МАКСИМ КУЗНЕЦОВ
ИГОРЬ СИМДЯНОВ
СЕРГЕЙ ГОЛЫШЕВ

PHP 5

НА ПРИМЕРАХ

СЕКРЕТЫ PHP

ЗАЩИТА САЙТОВ
ОТ ВЗЛОМА

ОПТИМИЗАЦИЯ КОДА

РАБОТА С ГРАФИКОЙ,
FLASH, PDF-ДОКУМЕНТАМИ

РАБОТА
С БАЗАМИ ДАННЫХ

bhv

Максим Кузнецов
Игорь Симдянов
Сергей Голышев

PHP 5

НА ПРИМЕРАХ

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.068+800.92PHP 5
ББК 32.973.26-018.1
К89

Кузнецов М. В., Симдянов И. В., Голышев С. В.

К89 PHP 5 на примерах. — СПб.: БХВ-Петербург, 2005. — 576 с.: ил.

ISBN 5-94157-670-6

Предметом книги является освоение приемов программирования на языке PHP 5. Обучение производится с использованием многочисленных примеров, взятых из реальной практики Web-программирования: работа с СУБД MySQL, вопросы защиты Web-приложений, работа с графикой, Flash и PDF-документами, оптимизация кода и другие задачи. Книга ориентирована как на читателей, не имеющих большого опыта программирования на PHP, так и на Web-программистов, уже знакомых с этим языком и желающих рассмотреть различные способы использования PHP 5.

Для Web-разработчиков

УДК 681.3.068+800.92PHP 5
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 23.05.05.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 46,44.
Тираж 3000 экз. Заказ № 1056
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 5-94157-670-6

Оглавление

Введение

Для кого и о чем эта книга

В феврале 2005 г. в издательстве "БХВ-Петербург" в серии "Профессиональное программирование" вышла наша книга "PHP 5. Практика разработки Web-сайтов"[1], в которой есть глава с названием "Хитрости PHP". В нее были помещены небольшие, но поучительные и интересные приемы из разных областей PHP-программирования. Конечно, многое осталось за "бортом", поэтому было принято решение расширить эту главу до книги. Так появилась книга, которую вы держите в руках.

Основная идея книги — рассмотрение большего количества разнообразных практических примеров работы с использованием PHP 5. Все представленные примеры и задачи — это реальные вопросы посетителей форума по Web-программированию на Web-портале SoftTime и ответы авторов на эти вопросы. Посещаемость указанного форума на момент написания книги составляла около 3000 посетителей в день; каждый день на форуме появляется около 100 новых сообщений. При такой обратной связи с читателями авторы имели возможность собрать уникальную статистику по тем проблемам, которые возникают у читателей различных профессиональных категорий, работающих в разных областях: от создателей интернет-магазинов и порталов до разработчиков интернет-игр и WAP-сайтов. Материал этой книги подбирался с учетом реальных потребностей современного Web-разработчика.

Специальная часть книги *(приложения 1—5)* посвящена рассмотрению сложных примеров работы с HTML/DHTML/XHTML, CSS и JavaScript и взаимодействия их с PHP, а также вопросам оптимизации Web-графики. Этот материал включен авторами в настоящую книгу по просьбам как читателей книги

---

[1] Кузнецов М. В., Симдянов И. В., Голышев С. В. PHP 5. Практика разработки Web-сайтов. — СПб.: БХВ-Петербург, 2005. — 960 с.

"Самоучитель PHP 5"[1], так и посетителей форума PHP на сайте поддержки книг авторов.

Книга ориентирована на читателей, владеющих языком разметки гипертекста HTML, элементарными основами языка PHP и желающих повысить свой профессиональный уровень в разработке Web-приложений с использованием PHP 5.

На Web-сайте IT-студии SoftTime, сотрудниками которой являются авторы книги, работает форум (**http://www.softtime.ru/forum/**), посвященный программированию на PHP, на который авторы с удовольствием приглашают читателей этой книги обсудить с авторами и участниками форума свои вопросы по материалу данной книги, а также любые проблемы в области Web-программирования.

Благодарности

Авторы выражают признательность сотрудникам издательства "БХВ-Петербург", благодаря которым наша рукопись увидела свет.

Авторы благодарны читателям, приславшим отзывы на две наши предыдущие книги, вышедшие в издательстве "БХВ-Петербург", а также участникам форума SoftTime.

Если при чтении книги у вас возникнут вопросы по изложенному материалу, вы можете обращаться на наш форум, который располагается по адресу **http://www.softtime.ru/forum/**.

[1] Кузнецов М. В., Симдянов И. В. Самоучитель PHP 5. — СПб.: БХВ-Петербург, 2004.— 560 с.

ГЛАВА 1

Установка и настройка PHP 5 и Web-сервера Apache 2.0.0

В этой главе мы рассмотрим установку и настройку Web-сервера Apache 2.0.0 и PHP 5 для использования их на локальной машине под управлением операционной системы Windows 2000/XP/2003 Server. Применение локальных серверов может понадобиться по многим причинам, например, необходимо изучить PHP или MySQL, а тестирование своих Web-приложений на хостинге либо дорого, либо невозможно. В этом случае вам необходима связка "Apache + PHP" на локальной машине. В данной главе будут рассмотрены вопросы установки PHP, Web-сервера Apache и библиотек расширений для PHP — MySQL. В конце главы будут приведены типичные вопросы и ошибки, которые возникают при установке данных приложений.

Замечание

Порядок установки, а также настройка PHP 5 и Web-сервера Apache могут меняться с появлением новых версий. В этом случае авторы будут вносить изменения в текст статьи, посвященной настройке связки PHP, Apache и MySQL, которая расположена по адресу **http://www.softtime.ru/info/articlephp.php?id_article=24**. Кроме того, в случае возникновения проблем с установкой любого из компонентов вы можете обращаться на форум **http://www.softtime.ru/forum/**.

Примечание

Установка Apache 2.0.0 и PHP 5 под ОС UNIX описана в нашей предыдущей книге по языку PHP — "PHP 5. Практика разработки Web-сайтов"[1].

Дистрибутивы

Для начала необходимо найти дистрибутивы серверов Apache и MySQL, а также архив PHP.

---

[1] Кузнецов М. В., Симдянов И. В., Голышев С. В. PHP 5. Практика разработки Web-сайтов. — СПб.: БХВ-Петербург, 2005. — 960 с.

Скачать Apache можно с "зеркальных" серверов, приведенных на официальном сайте **http://www.apache.org/dyn/closer.cgi**. При поиске следует помнить, что Apache также может называться httpd, по имени своего демона в UNIX. На "зеркалах" обычно много различных файлов, например:

- httpd-2.0.51-win32-src.zip — архив с исходными кодами (src) для Windows (win32) Web-сервера Apache (httpd) версии 2.0.51;
- httpd-2.0.51.tar.gz — то же самое, но для ОС Linux, для которой программы принято распространять в исходных кодах;
- apache_2.0.50-win32-x86-no_ssl.exe — откомпилированный под архитектуру (x86) для Windows (win32) без поддержки SSL (no_ssl) сервер Apache (apache) версии 2.0.50. Именно он и нужен.

Замечание

Бинарные (двоичные) коды дистрибутивов Apache для Windows распространяются в нескольких вариантах, как с расширением exe, так и msi, и имеют название вида httpd_*версия*_win32_*_.msi.

Замечание

Скачать дистрибутивы Apache для Windows можно по адресу **http://apache.rin.ru/dist/httpd/binaries/win32/**.

Замечание

Вторая и третья цифры в версии могут отличаться от приведенных выше — нужно выбирать самую последнюю стабильную версию, т. к. в ней устранены ошибки, обнаруженные в предыдущих версиях. Также не следует использовать версии с индексами alpha, поскольку они являются тестовыми и могут работать нестабильно.

PHP 5 можно обнаружить на "зеркалах" **http://www.php.net/downloads.php**. На сайте дистрибутив PHP доступен в двух формах: исходных кодах (Complete Source Code) и предкомпилированном варианте (Windows Binaries). Нас будет интересовать предкомпилированная версия, которая так же распространяется в двух вариантах: в виде инсталлятора (php-5.0.0-installer.exe) и в виде zip-архива (php-5.0.0-Win32.zip). Автоматический инсталлятор удобен, но содержит лишь ограниченную версию PHP (для сравнения: инсталлятор занимает 2 Мбайт, а zip-архив 7 Мбайт). Кроме того, использование автоматического инсталлятора не избавляет от необходимости настройки конфигурационного файла httpd.conf сервера Apache. Поэтому рекомендуется все же загрузить zip-архив.

С этого же ресурса можно скачать документацию к PHP. Для пользователей Windows наиболее удобен chm-формат, позволяющий осуществлять поиск (**http://www.php.net/download-docs.php**).

Замечание

Дистрибутив MySQL можно загрузить со страницы **http://dev.mysql.com/downloads/**. Полное справочное руководство на русском языке легко найти на официальном сайте MySQL по адресу **http://dev.mysql.com/doc/mysql/ru/index.html**.

После того как мы запаслись всеми необходимыми дистрибутивами, можно приступать к установке. Приоритетность установки Apache и PHP не имеет значения. Начнем с Web-сервера Apache.

Установка Web-сервера Apache под Windows

Сервер Apache под Windows доступен как в виде исходных кодов, так и в виде откомпилированного пакета. Исходные коды могут понадобиться в том случае, если вы решили при установке перекомпилировать Apache. Перекомпилирование исходных кодов требуется, когда нужно исключить из исполняемой версии неиспользуемые функции или включить поддержку функций, не входящих в стандартную поставку. В этом случае необходимо наличие установленного на машине пакета Microsoft Visual Studio. Если стандартная компиляция сервера вас устраивает, можно приступить к установке.

Примечание

Несмотря на то, что Web-сервер Apache уже давно перенесен и успешно функционирует под Windows, только с версии Apache 2.0.0 произведена оптимизация исходных кодов с целью использования системных вызовов Windows. До этого использовался уровень POSIX (Portable Operating System Interface, переносимый интерфейс операционной системы), что приводило к недостаточной производительности как приложений, так и самого Apache при работе с Apache под Windows.

Замечание

Бинарные коды дистрибутивов Apache распространяются в нескольких вариантах, как с расширением exe, так и msi, и имеют название вида httpd_*версия*_win32_*_.msi.

Итак:

1. После двойного щелчка на файле httpd_*версия*_win32_*_.msi запустится Microsoft Installer.
2. Для начала установки нажмите кнопку **Next**. Появится окно с лицензионным соглашением. После принятия лицензионного соглашения следует перейти к следующему окну с краткой информацией о нововведениях во второй версии Apache.

3. Следующее окно, показанное на рис. 1.1, позволяет ввести в соответствующие поля информацию о сервере: доменное имя сервера (**Network Domain (e.g. somenet.com)**), имя сервера (**Server Name (e.g. www.somenet.com)**) и адрес электронной почты администратора (**Administrator's Email Address (e.g. webmaster@somenet.com)**). Если установка происходит на локальную машину, то в поля для доменного имени и имени сервера следует ввести `localhost` (рис. 1.1). В нижней части окна предлагается выбрать номер порта, по которому сервер будет принимать запросы (80 или 8080) — переключатель **for All Users, on Port 80, as a Service – Recommended.**

Замечание

Обычно используется порт 80, стандартный для протокола HTTP. Если для запуска сервера Apache по порту 80 не достаточно прав, следует выбрать порт 8080.

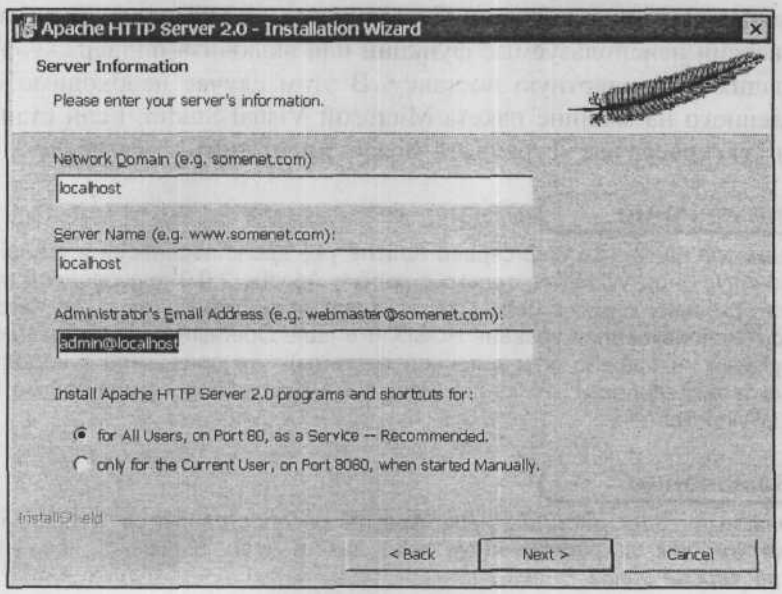


Рис. 1.1. Настройка параметров сервера

4. После этого будет предложен способ установки (рис. 1.2): типичный (**Typical**) или выборочный (**Custom**), позволяющий указать компоненты сервера вручную.
5. Следующее окно предоставляет возможность выбрать каталог установки сервера, по умолчанию это C:\Program Files\Apache Group.

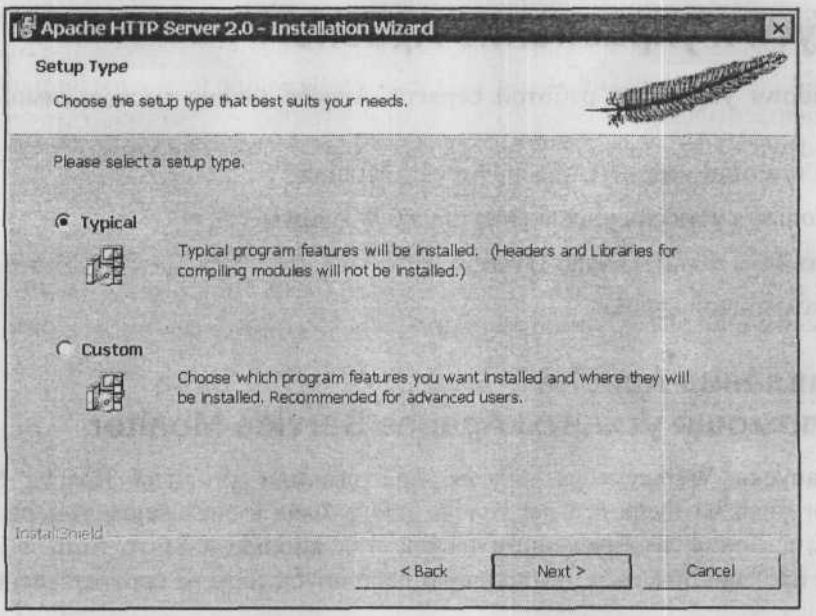


Рис. 1.2. Выбор способа установки

6. Далее мастер установки сообщит о готовности к процессу установки, и после нажатия кнопки **Install** будет произведено копирование файлов сервера.

Если установка прошла успешно, Windows автоматически запустит Apache. По умолчанию вместе с сервером запускается утилита мониторинга работы сервера Apache Service Monitor, значок которой помещается в системном триере[1].

Замечание

Работа утилиты Apache Service Monitor не влияет на работу сервера, и ее можно отключить. Если далее она потребуется ее можно найти в подкаталоге bin каталога установки Apache.

Проверить работоспособность сервера можно, набрав в браузере **http://localhost** (если установка проводилась для локальной машины). В случае успешной установки вы увидите приветственную страницу Apache.

[1] Триер — область панели задач (обычно справа), в которой расположены значки резидентных приложений.

Запуск и управление Apache

В Windows управлять работой сервера Apache можно несколькими способами:

- при помощи утилиты Apache Service Monitor;
- используя консоль управления служб Windows;
- используя пункты меню **Пуск**;
- из командной строки.

Управление Apache при помощи утилиты Apache Service Monitor

Для запуска Web-сервера Apache при помощи утилиты Apache Service Monitor дважды щелкните на значке программы в системном триере. В появившемся окне, изображенном на рис. 1.3, кнопками **Start**, **Stop** и **Restart** можно производить пуск, остановку и перезапуск сервера соответственно.

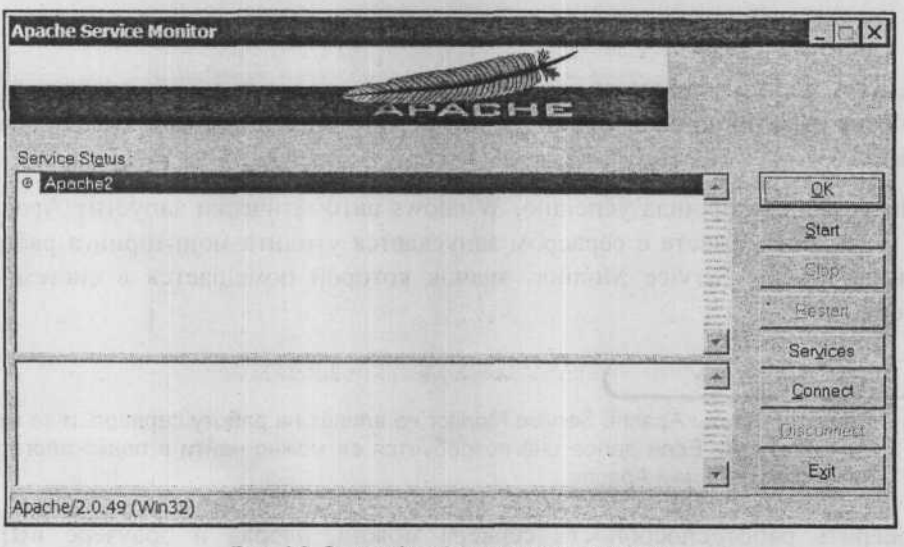


Рис. 1.3. Запуск Apache из триера программ

Управление Apache из консоли управления служб Windows

Если при установке сервера в качестве порта, по которому Apache принимает запросы, был выбран порт 80 (см. рис. 1.1), допускается запуск Apache в

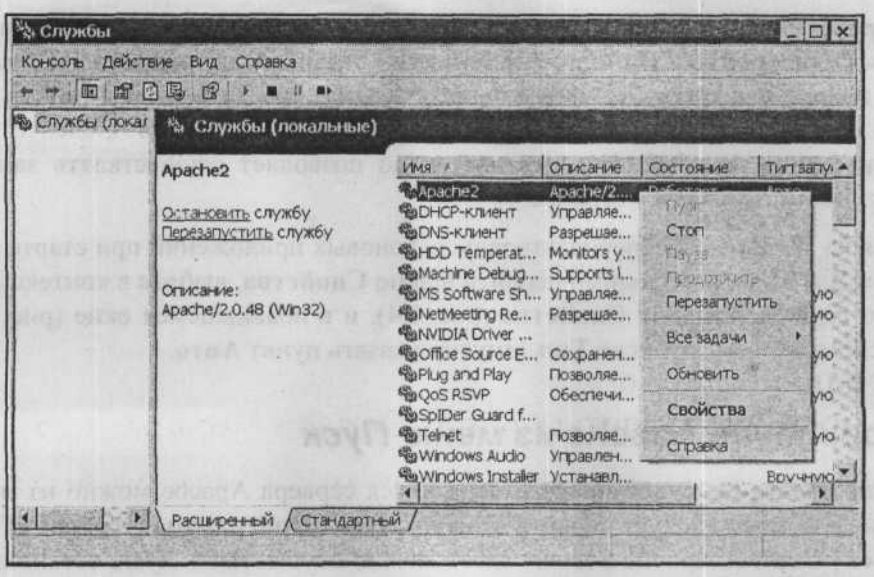


Рис. 1.4. Службы Windows

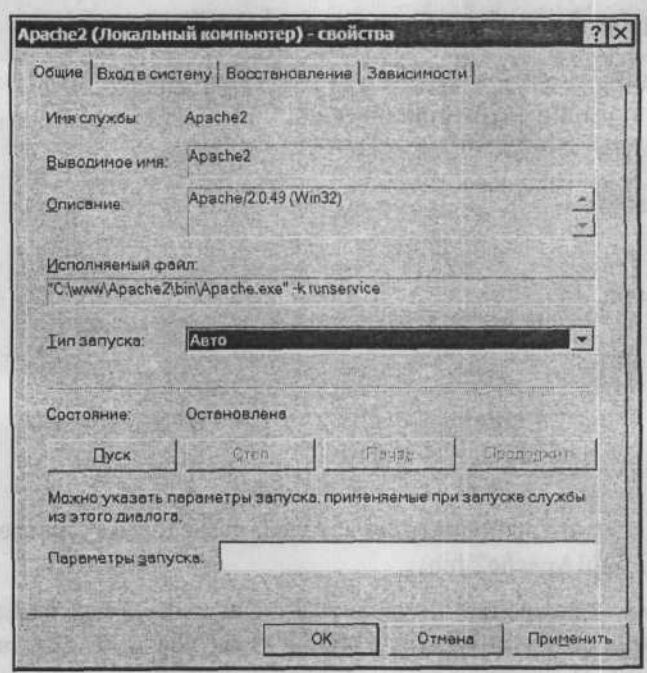


Рис. 1.5. Окно свойства сервиса Apache2

качестве сервиса. Для запуска консоли управления выполните команду **Пуск | Настройка | Панель управления | Администрирование | Службы** или нажмите кнопку **Services** в окне утилиты Apache Service Monitor (см. рис. 1.3). В появившемся окне консоли, приведенном на рис. 1.4, следует выбрать сервис Apache2. Контекстное меню позволяет осуществлять запуск, остановку и перезапуск сервиса.

Службы Windows обеспечивают запуск фоновых приложений при старте системы. Для этого необходимо перейти в окно **Свойства**, выбрав в контекстном меню сервиса команду **Свойства** (рис. 1.4), и в появившемся окне (рис. 1.5) в раскрывающемся списке **Тип запуска** указать пункт **Авто**.

Управление Apache из меню *Пуск*

Осуществлять пуск, остановку и перезапуск сервера Apache можно из меню **Пуск**. Для этого следует перейти в меню: **Пуск | Программы | Apache HTTP Server | Control Apache Server**. Здесь можно осуществить пуск, остановку и перезапуск сервера путем выбора пунктов **Start**, **Stop** и **Restart**.

Управление Apache из командной строки

Запускать, останавливать и перезапускать сервер Apache из командной строки можно при помощи следующих команд:

- старт

```
Apache -k start
```

- перезапуск

```
Apache -k restart
```

- стоп

```
Apache -k stop
```

или

```
Apache -k shutdown
```

Все команды следует выполнять из каталога bin сервера Apache (C:\Program Files\Apache Group\Apache2\bin\).

Команда `Apache -t` позволяет проверить конфигурационные файлы Apache на предмет наличия синтаксических ошибок. В случае их отсутствия выдается строка `Syntax OK`. Если же в конфигурационных файлах имеются ошибки, то в результате тестирования программа выдаст сообщение об ошибке.

Проблемы с установкой Apache и их устранение

Сервер Apache не установился, как служба Windows, автоматически

Если по каким-либо причинам сервер Apache не установился, как служба Windows, автоматически, то можно попытаться сделать это вручную. Но сначала удостоверьтесь, что Apache действительно отсутствует в списке служб. Для открытия списка служб следует выполнить команду **Пуск | Настройка | Панель управления | Администрирование | Службы**.

Для установки Apache, как службы, вручную, возможно, будет необходима корректировка директив конфигурационного файла httpd.conf. Поэтому если на первом этапе вам не хочется изучать его директивы, то более простым решением будет переустановка Apache через инсталлятор. После его запуска вам будут предложены на выбор два варианта: восстановление текущей установленной версии Apache, либо ее удаление (рис 1.6). Выберите нужную опцию и нажмите кнопку **Next**.

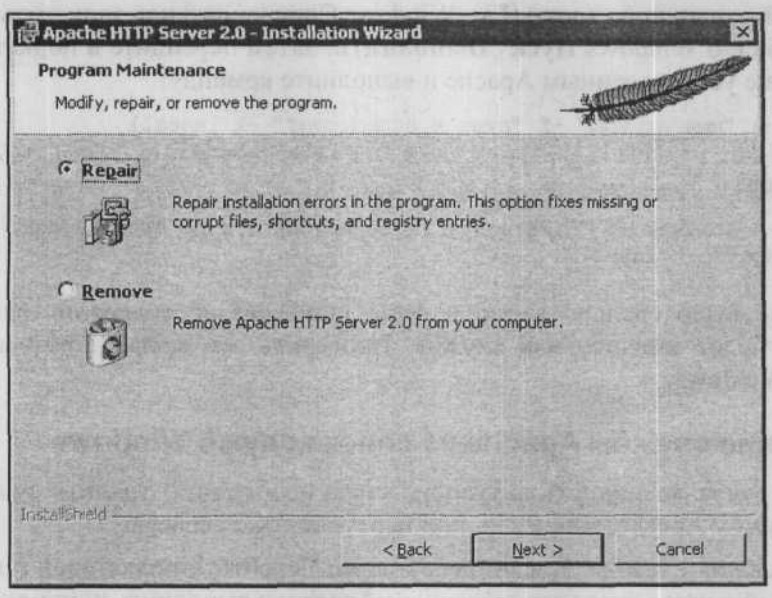


Рис. 1.6. Окно переустановки Apache

Установка Apache, как службы Windows, вручную

Перед установкой Apache, как службы Windows, вручную необходимо проверить директивы главного конфигурационного файла Apache httpd.conf. Для этого перейдите в каталог conf, расположенный в каталоге с установленным

сервером Apache, и откройте указанный файл для редактирования. Нас будут интересовать значения директив, где указаны пути к каталогам Apache, например:

- `ServerRoot "C:/Program Files/Apache Group/Apache2/"`
- `DocumentRoot "C:/Program Files/Apache Group/Apache2/htdocs"`
- `<Directory "C:/Program Files/Apache Group/Apache2/htdocs">`
- `Alias /icons/ "C:/Program Files/Apache Group/Apache2/icons/"`
- `<Directory "C:/Program Files/Apache Group/Apache2/icons">`
- `ScriptAlias /cgi-bin/ "C:/Program Files/Apache Group/Apache2/cgi-bin/"`
- `<Directory "C:/Program Files/Apache Group/Apache2/cgi-bin">`

Вам необходимо проверить, что все эти пути действительно существуют и указаны верно.

Далее, для установки новой службы нам понадобится выполнить команду из подкаталога bin, который находится в каталоге с установленным сервером Apache. Для этого следует воспользоваться любой shell-оболочкой с поддержкой командной строки (Far, Windows Commander) или запустить утилиту cmd из меню Windows **Пуск | Выполнить**. Затем перейдите в подкаталог bin каталога с установленным Apache и выполните команду:

```
Apache -n "имя_службы" -f "путь_к_httpd.conf" -k install
```

Например:

```
Apache -n Apache2 -f "C:/Program Files/Apache Group /Apache2/conf
/httpd.conf" -k install
```

Если все было сделано верно и файл httpd.conf не содержит ошибок, то Apache будет запущен как служба. Проверить это можно, открыв список служб Windows.

Удаление службы Apache из списка служб Windows

Иногда могут возникнуть ситуации, когда необходимо удалить существующую службу Apache, например, при переустановке сервера.

Для удаления службы Apache необходимо перейти в подкаталог bin, расположенный в каталоге с установленным Apache, и выполнить команду:

```
Apache -n "существующее_имя_службы" -k uninstall
```

Например:

```
Apache -n Apache2 -k uninstall
```

После перезагрузки Windows служба с именем `Apache2` будет удалена.

Конфигурирование Apache

Web-сервер — сложный программный продукт, работающий на разных платформах и в разных операционных системах по всему миру. Поэтому для корректной работы на установленной системе его необходимо настроить (сконфигурировать).

По умолчанию настройки Apache расположены в файле httpd.conf в каталоге conf. Далее будут описаны основные директивы файла httpd.conf и их общеупотребительные значения.

Пути к файлам

В конфигурационных файлах Apache и PHP вам часто придется указывать пути к различным каталогам и файлам. В операционных системах UNIX и Windows применяются различные разделители каталогов. В UNIX используется прямая косая черта (/), например /usr/bin/perl, в Windows — обратная (\), например c:\Apache\bin. Вообще, в некоторых директивах Apache и PHP работают оба вида разделителей каталогов: прямой (/) и обратный (\), но т. к. и Apache, и PHP изначально разрабатывались под UNIX, то, применяя их "родной" формат, вы сможете избежать ряда проблем. Поэтому пути в настроечных файлах (httpd.conf и php.ini) рекомендуется писать через слеш в формате UNIX — /.

Например:

```
ScriptAlias "/php_dir/" "c:/php/"
```

Директивы файла httpd.conf

Перечислим директивы конфигурационного файла httpd.conf.

Port

```
Port 80
```

Задает порт TCP, который используется Apache для установки соединения. По умолчанию указывается порт 80.

Примечание

Единственная причина задания нестандартного порта — это отсутствие прав на использование стандартного порта. При использовании нестандартного порта, например, 8080, номер порта следует указывать в адресе **http://localhost:8080/**.

ServerAdmin

```
ServerAdmin mymail@yandex.ru
```

Содержит адрес электронной почты администратора Web-сервера. Именно этот адрес будет отображаться при ошибках работы сервера.

ServerName

```
ServerName myserver
```

Содержит имя компьютера для сервера.

ServerRoot

```
ServerRoot "C:/Apache2"
```

Указывает на каталог, содержащий файлы Web-сервера Apache.

Примечание

Не путайте директиву `ServerRoot` с директивой `DocumentRoot`, которая указывает каталог для файлов Web-сайта.

DocumentRoot

```
DocumentRoot "C:/Apache2/htdocs"
```

Определяет каталог, в котором расположены файлы Web-сайта.

Контейнер <Directory />

Сфера действия директив внутри этого контейнера распространяется на все файлы и подкаталоги внутри каталога, заданного в директиве `DocumentRoot`.

```
<Directory />
  Options FollowSymLinks Includes Indexes
  AllowOverride All
</Directory>
```

- Директива `AllowOverride`, установленная в значение `All`, разрешает переопределять значения главного конфигурационного файла httpd.conf в файлах .htaccess.
- Директива `Options FollowSymLinks` разрешает Apache следовать символическим ссылкам.
- Директива `Options Includes` разрешает выполнение директив SSI (Server Side Includes, включения на стороне сервера) в коде страниц Web-сайта.
- Директива `Options Indexes` указывает, что нужно возвращать содержимое каталога, если отсутствует индексный файл.

DirectoryIndex

```
DirectoryIndex index.html index.phtml index.php
```

Содержит список индексных файлов, которые следует отображать при обращении к каталогу без указания имени файла (например, **http://localhost/test/**).

ScriptAlias

```
ScriptAlias /cgi-bin/ "C:/Apache2/cgi-bin/"
```

Директива `ScriptAlias` используется для создания псевдонима каталога /cgi-bin/, в котором располагаются программы и сценарии CGI. Далее необходимо настроить права и ограничения на каталог cgi-bin.

```
<Directory "C:/Apache2/cgi-bin">
  AllowOverride None
  Options ExecCGI
  Order allow,deny
  Allow from all
</Directory>
```

Данное действие необходимо для гарантии, что опции каталога не будут изменены, т. к. это может создать проблемы безопасности.

- Директива `AllowOverride None` говорит о том, что опции этого каталога не могут быть переопределены файлами .htaccess.
- Директива `Options ExecCGI` разрешает выполнение CGI-сценариев.
- Директивы `Order allow,deny` и `Allow from all` разрешают доступ к каталогу.

AddHandler

```
AddHandler cgi-script .bat .exe
```

Эта директива заставляет Apache рассматривать файлы с расширениями exe и bat, как CGI-скрипты.

DefaultType

```
DefaultType text/plain
```

Устанавливает заголовок файлов, тип которых не может быть определен по расширению. В данном случае все неизвестные файлы воспринимаются на обычные текстовые файлы. Для того чтобы все неизвестные расширения файлов обрабатывать, как HTML, измените директиву следующим образом:

```
DefaultType text/html
```

AddDefaultCharset

```
AddDefaultCharset windows-1251
```

Устанавливает кодировку по умолчанию, если кодировка не задана в заголовке HTML-документа. Также вам может потребоваться указывать значение кодировки KOI8-R.

Создание виртуальных хостов

На одном Web-сервере Apache можно установить несколько Web-сайтов. Эта функция сервера называется *виртуальным хостингом*. Далее рассмотрим создание виртуальных узлов на основе имен. Виртуальные узлы обычно расположены в конце файла httpd.conf.

Сначала требуется указать, какой IP-адрес используется для виртуальных хостов.

```
NameVirtualHost 127.0.0.1:80
```

Затем нужно прописать директивы для контейнера `<VirtualHost>`, которые будут определять конфигурацию виртуального хоста (листинг 1.1).

Листинг 1.1. Файл httpd.conf. Контейнер `<VirtualHost>`

```
<VirtualHost 127.0.0.1>
  ServerAdmin Webmaster@may_domain.ru
  DocumentRoot c:/www/mysite
  ServerName www.mysite.ru
  ServerAlias www.site.ru www.host2.ru
  ErrorLog logs/ mysite-error.log
  CustomLog logs/ mysite-access.log common
</VirtualHost>
```

Рассмотрим директивы виртуального узла:

- `DocumentRoot` указывает каталог, где расположены файлы (страницы) данного виртуального узла (Web-сайта);
- `ServerName` определяет имя виртуального узла, по которому к нему можно обратиться (в данном случае, по адресу **http://www.mysite.ru/**);
- `ServerAlias` содержит псевдонимы имен виртуального узла (в данном случае к виртуальному узлу можно также обратиться, используя имена **http://www.site.ru/** и **http://www.host2.ru/**);
- `ErrorLog` и `CustomLog` указывают имена log-файлов сервера для этого виртуального хоста.

Контейнеры `<VirtualHost>` обычно располагают один за другим в конце файла httpd.conf (листинг 1.2).

Листинг 1.2. Файл httpd.conf. Настройка виртуальных хостов

```
<VirtualHost 127.0.0.1>
  # Директивы виртуального хоста 1
</VirtualHost>
<VirtualHost 127.0.0.1>
```

```
  # Директивы виртуального хоста 2
</VirtualHost>
<VirtualHost 127.0.0.1>
  # Директивы виртуального хоста 3
</VirtualHost>
```

Для вступления в силу изменений, внесенных в файл httpd.conf, Apache следует перезагрузить.

Для того чтобы обращаться к виртуальным узлам по именам, их следует прописать в базе данных DNS-сервера. Если вы используете Apache для тестирования файлов на локальной машине, то имена ваших виртуальных узлов нужно прописать в файле hosts. Для Windows 2000 и XP он расположен в каталоге C:\Window\System32\Drivers\ets\. Файл hosts содержит записи, подобные представленным в листинге 1.3.

Листинг 1.3. Формат записей файла hosts

```
127.0.0.1       www.mysite.ru
127.0.0.1       www.site.ru
127.0.0.1       www.host2.ru
```

Если виртуальных хостов много, то работа по их конфигурированию в едином настроечном файле httpd.conf может усложниться. Решением этой проблемы является создание специального настроечного файла для каждого виртуального узла и присоединение этих настроечных файлов к файлу httpd.conf с помощью директивы `Include`. В настроечные файлы помещаются только контейнер `<VirtualHost>` и директивы внутри него.

Подключение к файлу httpd.conf происходит так, как указано в листинге 1.4.

Листинг 1.4. Файл httpd.conf с внешними файлами настроек

```
NameVirtualHost 127.0.0.1:80
Include conf/mysite.conf
Include conf/site.conf
Include conf/host2.conf
```

Установка и настройка PHP

Для установки PHP:

1. Следует создать каталог c:/php и разместить в нем файлы из zip-архива дистрибутива.

2. После этого нужно переименовать конфигурационный файл php.ini-dist в php.ini и скопировать его в каталог Windows.

3. Затем необходимо сообщить Web-серверу о наличии установленного PHP. Установка PHP возможна двумя вариантами: как модуль Apache и как внешнее CGI-приложение. Далее будут рассмотрены оба варианта установки.

Установка PHP в качестве модуля

Установка PHP в качестве модуля немного повышает быстродействие, т. к. модуль PHP загружается один раз при запуске Web-сервера.

Примечание

При установке PHP в качестве модуля настройки из php.ini читаются один раз при запуске Web-сервера. Поэтому при внесении изменений в php.ini необходимо перегрузить Apache для того, чтобы внесенные изменения вступили в силу.

Для установки PHP откройте главный настроечный файл Apache httpd.conf для редактирования и удалите символы комментариев со следующих строк, при необходимости изменив их (листинг 1.5).

Листинг 1.5. Файл httpd.conf. Подключение PHP как модуль Apache

```
AddType application/x-httpd-php phtml php
LoadModule php5_module c:/php/php5apache2.dll
```

Установка PHP, как CGI-приложения

При установке PHP, как CGI-приложения, интерпретатор PHP будет загружаться каждый раз при вызове PHP-сценария. В связи с этим возможно некоторое ухудшение быстродействия. Если PHP установлен, как CGI-приложение, то при внесении изменений в файл php.ini Apache перезагружать не следует, т. к. установки читаются каждый раз при выполнении PHP-сценария. Установка PHP, как CGI, немного ускоряет внесение изменений в конфигурацию PHP, поскольку она не требует перезагрузки Web-сервера.

Примечание

При установке PHP, как CGI-приложения, перестанут работать некоторые заголовки. Например, вы не сможете организовать авторизацию пользователей средствами PHP. Ее можно будет реализовать только средствами самого Apache с помощью файлов .htaccess.

Для установки PHP откройте главный настроечный файл httpd.conf для редактирования, найдите в нем директивы, подключающие PHP, как CGI, и сверьте их с директивами листинга 1.6.

Листинг 1.6. Файл httpd.conf. Подключение PHP, как CGI-приложения

```
AddType application/x-httpd-php phtml php
<Directory "c:/php">
  Options ExecCGI
</Directory>
ScriptAlias "/php_dir/" "c:/php/"
Action application/x-httpd-php "/php_dir/php-cgi.exe"
```

При установке PHP, как CGI, директивы, подключающие PHP в качестве модуля, должны быть закомментированы.

Директивы файла php.ini

Рассмотрим наиболее употребительные директивы файла php.ini.

Примечание

Объем книги не позволяет рассмотреть все директивы полностью, подробно их описание приведено в нашей книге "PHP 5. Практика разработки Web-сайтов."

Конфигурирование PHP

engine

```
engine = On|Off
```

Разрешает обработку PHP-скриптов.

short_open_tag

```
short_open_tag = On|Off
```

Код PHP может обрамляться упрощенным тегом с синтаксисом <?. Если эта директивы выключена, то код PHP необходимо выделять символами <?php.

asp_tags

```
asp_tags = On|Off
```

Разрешает для выделения кода PHP пользоваться тегами в стиле ASP — <% %>.

Precision

```
Precision = число
```

Определяет количество цифр после запятой для чисел с плавающей точкой. Например:

```
Precision = 12
```

output_buffering

```
output_buffering = On|Off
```

Включает и отключает буферизацию вывода. Если данная директива включена, то это означает, что заголовки HTTP можно выводить в любом месте сценария. Для управления буферизацией вывода также можно использовать функции `ob_start()` и `ob_end_flush()`.

safe_mode

```
safe_mode = On|Off
```

Включение или выключение безопасного режима для интерпретатора PHP, работающего в режиме CGI. В безопасном режиме сценариям PHP запрещен доступ к файлам, лежащим за пределами Web-узла, к которому они принадлежат.

disable_functions

```
disable_functions = имя_функции1 [, имя_функции2...]
```

Данная директива позволяет запретить вызовы функций, которые перечислены в ней через запятую. Например:

```
disable_functions = fopen, fwrite, popen
```

disable_classes

```
disable_classes = имя_класса1 [, имя_класса2...]
```

Директива позволяет запретить вызовы классов, которые перечислены в ней через запятую.

Ограничение по ресурсам

max_execution_time

```
max_execution_time = число
```

Устанавливает максимально возможное время выполнения скрипта в секундах. Если сценарий превысил это время, то интерпретатор PHP его отключает.

max_input_time

```
max_input_time = число
```

Устанавливает максимально возможное время в секундах, которое скрипт может потратить на обработку загружаемых данных.

memory_limit

```
memory_limit = объем_в_байтах [объем_в_мегабайтах]
```

Максимальный объем памяти, выделяемый сценарию. Например:

```
memory_limit = 8M
```

Обработка ошибок и ведение журнала

error_reporting

```
error_reporting = [битовое поле][предопределенная константа]
```

Директива `error_reporting` задается в виде битового поля, т. е. ее значение устанавливается с помощью перечисленных далее констант, объединенных оператором | (OR). Такими константами могут быть:

- `E_ALL` (все предупреждения и ошибки);
- `E_ERROR` (фатальные ошибки во время выполнения);
- `E_WARNING` (некритические ошибки, т. е. предупреждения, во время выполнения);
- `E_PARSE` (ошибки трансляции);
- `E_NOTICE` (замечания во время выполнения, которые говорят о возможных логических ошибках в скрипте; к этому типу ошибок относятся, к примеру, предупреждения об использовании неинициализированных переменных);
- `E_CORE_ERROR` (фатальные ошибки при старте PHP);
- `E_CORE_WARNING` (предупреждения при старте PHP);
- `E_COMPILE_ERROR` (фатальные ошибки при компиляции);
- `E_COMPILE_WARNING` (предупреждения при компиляции);
- `E_USER_ERROR` (ошибки, вызванные действиями пользователя);
- `E_USER_WARNING` (предупреждения, вызванные действиями пользователя);
- `E_USER_NOTICE` (замечания, вызванные действиями пользователя).

Таким образом, для того чтобы показывать все ошибки, за исключением замечаний о возможных логических ошибках в сценарии, нужно установить директиву `error_reporting` так:

```
error_reporting = E_ALL & ~E_NOTICE
```

Для того чтобы показывать только ошибки, директива должна быть настроена так, как показано ниже:

```
error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
```

display_errors

```
display_errors = On|Off
```

Разрешает или запрещает вывод ошибок и предупреждений в браузер. При отладке приложений оставьте эту директиву включенной, в реальной же работе лучше ее отключить (в PHP 5 по умолчанию она отключена), поскольку при включенной `display_errors` пользователь может получить доступ к секретной информации. Вместо этой директивы можно использовать возможность ведения журнала с помощью директив `log_error` и `error_log`.

display_startup_errors

```
display_startup_errors = On|Off
```

Включает или отключает отображение ошибок, возникающих при запуске PHP.

log_errors

```
log_errors = On|Off
```

Включает или отключает сохранение сообщений об ошибках PHP в файле журнала. Путь к журналу и режим его использования задается директивой `error_log`.

error_log

```
error_log = полный_путь_к_журналу
```

Позволяет вести журнал в указанном файле. Если в качестве пути к файлу задать ключевое слово `syslog`, то в системе Windows запись будет производиться в журнал событий, а в UNIX регистрация ошибок будет вестись в системе syslog.

track_errors

```
track_errors = On|Off
```

Включает или отключает сохранение последнего сообщения об ошибке в переменной `$php_errormsg`.

Обработка данных

variables_order

```
variables_order = "EGPCS"
```

Эта директива определяет порядок регистрации переменных Environment, GET, POST, Cookie, и Session (соответственно E, G, P, C и S). Регистрация осуществляется путем чтения этой строки слева направо. Если убрать какой-либо символ из данной директивы, то соответствующие переменные не смогут использоваться в сценариях PHP.

Примечание

Переменные Environment — это переменные окружения Web-сервера. GET — переменные, передаваемые методом GET (в адресной строке браузера), POST — переменные, передаваемые методом POST. Cookie — переменные, получаемые из файлов cookies. Session — сессионные переменные.

register_globals

```
register_globals = On|Off
```

Включает или отключает возможность регистрации EGPCS-переменных как глобальных переменных. Дело в том, что использование глобальных переменных может создать "дыры" в защите сценария, поэтому в PHP5 по умолчанию эта директива отключена.

Примечание

Подробнее о "дырах" в системе безопасности, возникающей при включенной директиве register_globals, написано в *гл. 9*.

register_long_arrays

```
register_long_arrays = On|Off
```

Директива, разрешающая или запрещающая использование для передачи переменных в виде длинных массивов вида $HTTP_***_VARS.

Замечание

Выставленное по умолчанию значение этой директивы является весьма спорным, поскольку реальных опасений с точки зрения защиты при включенной директиве register_long_arrays возникать не должно. Многие же разработчики предпочитают в силу привычки суперглобальным массивам использование массивов вида HTTP_***_VARS. Если же указанная директива выключена, естественно, такой код перестает быть рабочим.

post_max_size

```
post_max_size = объем_в_байтах [объем_в_мегабайтах]
```

Максимальный размер данных, передаваемых методом POST. Например:

```
post_max_size = 8M
```

magic_quotes_gpc

```
magic_quotes_gpc = On|Off
```

Включает или отключает автоматическое экранирование специальных символов (одинарная кавычка, двойная кавычка, завершающий ноль, косая черта), передаваемых методами GET, POST и Cookie.

magic_quotes_runtime

```
magic_quotes_runtime = On|Off
```

Включает или отключает заключение в кавычки данных, сгенерированных во время выполнения сценария, к примеру, для SQL-данных, данных из функции exec() и т. д.

auto_prepend_file

```
auto_prepend_file = имя_файла
```

Предназначена для добавления верхнего колонтитула на каждую страницу PHP-сценария.

auto_append_file

```
auto_append_file = имя_файла
```

Предназначена для добавления нижнего колонтитула на каждую страницу PHP-сценария.

default_mimetype

```
default_mimetype = "text/html"
```

Эта директива указывает, какую информацию PHP сообщает браузеру об используемой кодировке в заголовке Content-type. По умолчанию задается значение text/html без указания кодировки.

doc_root

Параметру doc_root необходимо передать значение директивы DocumentRoot Web-сервера. Например:

```
doc_root = "c:/www/"
```

Загрузка файлов

file_uploads

```
file_uploads = On|Off
```

Разрешает или запрещает загрузку файлов на сервер.

upload_tmp_dir

```
upload_tmp_dir = каталог
```

Эта директива устанавливает временный каталог для загруженных файлов.

upload_max_filesize

```
upload_max_filesize = объем_в_байтах [объем_в_мегабайтах]
```

Устанавливает максимальный размер файла, который может быть загружен с удаленного компьютера средствами PHP.

Подключение библиотек расширений

extension

```
extension = имя_модуля
```

Данная директива предназначена для загрузки модуля динамического расширения PHP. Директиву можно повторять произвольное количество раз для загрузки разных расширений. Например:

```
extension = php_mbstring.dll
extension = php_exif.dll
extension = php_gd2.dll
```

extension_dir

```
extension_dir = "c:/php/ext/"
```

Указывает каталог, в котором расположены библиотеки расширений для PHP.

Подключение MySQL

Если сервер MySQL уже установлен на вашей машине, то следующим шагом будет настройка PHP для работы с базами данных MySQL.

1. Откройте для редактирования файл php.ini из каталога Windows.
2. Для подключения библиотеки расширения MySQL вам нужно убрать символ комментария ; (точка с запятой) из строки:

   ```
   extension=php_mysql.dll
   ```

3. Также проверьте значение директивы `extension_dir`

   ```
   extension_dir ="c:/php-5.0/ext"
   ```

   Она должна указывать на каталог, где хранятся расширения PHP. Разделители каталогов рекомендуется писать в формате UNIX (/) — прямой слеш.

4. Если PHP у вас подключен как модуль, то вам также необходимо скопировать библиотеку libmysql.dll из каталога с установленным PHP в системный каталог C:\Windows\System32.

5. Чтобы внесенные изменения вступили в силу, перезагрузите Apache.

Чтобы проверить подключение к базе данных, создайте тестовый файл phpinfo.php с вызовом функции `phpinfo()` — листинг 1.7.

Листинг 1.7. Файл phpinfo.php. Проверка работы PHP

```
<?
  phpinfo();
?>
```

Скопируйте его в каталог документов Web-сервера и обратитесь к нему из адресной строки браузера: **http://localhost/phpinfo.php**.

Если настройка подключения расширения MySQL правильная, то в таблицах будет выведен раздел с именем MySQL, содержащий номер версии MySQL и значения некоторых настроек.

Типичные ошибки, возникающие при установке Apache, PHP и MySQL

Не запускается инсталлятор Apache

Иногда при запуске инсталлятора Apache, имеющего расширение msi, например, apache_2.0.49-win32-x86-no_ssl.msi выдается сообщение, гласящее: "Приложение, выполняющее эту операцию, указанному файлу не сопоставлено". Такая ошибка может возникать, если вы пытаетесь запускать инсталлятор из shell-оболочки Far и ее аналогов. Попробуйте запустить инсталлятор из Проводника Windows.

Если вы запускаете инсталлятор из Проводника, но он также не запускается, либо процесс установки не доходит до завершения, то, возможно, проблема в отсутствии или наличии старой версии Windows Installer. Обновите пакет Windows Installer у себя на компьютере.

Ошибка "Internal Server Error" при подключении PHP

Проверьте, нет ли у вас ошибок при подключении PHP в файле httpd.conf (листинг 1.8).

Листинг 1.8. Файл httpd.conf. Подключение PHP как CGI

```
AddType application/x-httpd-php phtml php
<Directory "c:/php">
  Options ExecCGI
</Directory>
ScriptAlias "/php_dir/" "c:/php/"
Action application/x-httpd-php "/php_dir/php-cgi.exe"
```

Особое внимание обратите на имя подключаемого модуля: php-cgi.exe. Для применения изменений, вносимых в файл httpd.conf, Apache следует перезагрузить.

При использовании устаревшего имени php.exe, применявшегося в более ранних версиях вместо php-cgi.exe, также возможно появление ошибки:

```
403 Forbidden You don't have permission to access /__php_dir__/php.exe
/test.php on this server
```

Не исполняются PHP-скрипты

При ненастроенном подключении PHP при обращении к файлам с расширением php, например, **http:/localohost/index.php**, открывается окно с запросом на загрузку такого файла. Это говорит о том, что не настроена обработка файлов с расширением php. Проверьте в файле httpd.conf существование следующей строки:

```
AddType application/x-httpd-php phtml php
```

Сообщение "Notice: Undefined variable:"

На новом, только что установленном, PHP можно часто видеть сообщения вида:

```
Notice: Undefined variable: msg in C:\Main\addrec.php on line 7
```

Это не ошибка, а это рекомендации по кодированию, сообщающие о неинициализированных переменных. Следует отключить вывод подобных рекомендаций в файле php.ini. Найдите директиву `error_reporting` и установите ее значение:

```
error_reporting = E_ALL & ~E_NOTICE
```

Не подключается MySQL

При запуске Apache и при обращении к скриптам выдается сообщение о невозможности загрузки библиотеки php_mysql.dll (рис. 1.7).

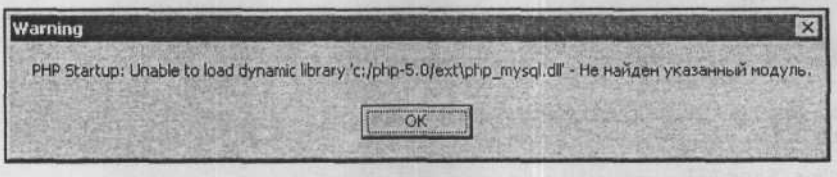


Рис. 1.7. Ошибка при загрузке библиотеки php_mysql.dll

Еще раз сверьтесь с инструкциями из раздела, где описывается подключение к PHP библиотеки для работы с MySQL. Удостоверьтесь, что вы используете "правильную" версию файла php_mysql.dll (именно для той версии PHP, которая установлена в системе). Версии файла php_mysql.dll различаются для разных версий PHP, хотя и имеют одно и то же название.

Проверьте, скопирована ли библиотека libmysql.dll из каталога с установленным PHP в системный каталог C:\Windows\System32.

Неизвестные ошибки

Если все же настроить связку Apache, PHP и MySQL не получается, то причины могут быть в следующем:

- использование русских букв в именах файлов и каталогах, прописываемых в настроечных файлах httpd.conf и php.ini;
- использование пробелов в именах файлов и каталогах в настроечных файлах. Если в именах файлов и каталогов есть пробелы, то их нужно обрамлять кавычками";
- указание разделителей каталогов формата Windows (обратный слеш): c:\apache\bin. Для надежной работы следует использовать разделители в формате UNIX (прямой слеш), например: c:/apache/bin;
- существование нескольких настроечных файлов php.ini на машине, либо отсутствие такого файла. Нужный файл php.ini должен находиться в каталоге Windows. Проведите поиск по дискам компьютера, найдите все лишние версии настроечных файлов и удалите их.

ГЛАВА 2

Приемы конфигурирования Web-сервера Apache 2

Web-сервер Apache — это мощный и многофункциональный программный продукт с разнообразными возможностями. В данной главе будут рассмотрены приемы конфигурирования Apache, наиболее часто встречающиеся при разработке Web-сайтов.

Как известно, все настройки сервера Apache находятся в файле httpd.conf, доступ к которому имеется не всегда. Например, если используется виртуальный сервер на хостинге, когда один сервер Apache обслуживает сотни сайтов, то, естественно, нельзя позволить владельцу одного сайта менять конфигурацию сервера, которая отразится на всех остальных сайтах. Тем не менее Web-сервер Apache допускает конфигурирование на уровне отдельных каталогов при помощи файлов .htaccess. Именно на работу с этими файлами, как единственными конфигурационными файлами, которые доступны большинству Web-разработчиков, и будет сделан основной упор в этой главе.

Особенности конфигурирования в Windows

Web-сервер Apache в настоящее время является, действительно, многоплатформной системой, работающей на различных клонах UNIX, в Windows и других операционных системах. Но т. к. корни Apache идут из UNIX-систем, при конфигурировании Apache под Windows следует учитывать некоторые особенности, не очевидные для пользователей этой системы.

Во-первых, в системах UNIX и Windows используются различные разделители каталогов и файлов. В UNIX для разделения каталогов применяется прямой слеш (/), например, /pub/server/bin, в то время как в Windows традиционно используется обратный слеш (\), например, c:\apache\bin\, хотя данная операционная система поддерживает и прямой слеш. При конфигурировании Apache следует придерживаться UNIX-нотации, т. е. в качестве разделителей каталогов указывать прямой слеш, например, c:/apache/bin/.

Еще одной особенностью файловой системы Windows является наличие пробелов в именах файлов и каталогов. Для использования таких путей в конфигурационных файлах их следует обрамлять двойными кавычками (").

Файл .htaccess

Файл .htaccess (с точкой в начале имени) — это конфигурационный файл, который дает возможность настраивать работу сервера на уровне отдельных каталогов: устанавливать права доступа к файлам в каталогах, менять названия индексных файлов, самостоятельно обрабатывать коды ответов протокола HTTP, модифицировать адреса запрошенных страниц.

Замечание

В UNIX точка в начале файла является признаком скрытого файла. Поэтому большинство конфигурационных файлов предваряются точкой. Именно отсюда берет начало обозначение текущего каталога точкой ("."), а родительского каталога двумя точками ("..").

Файл .htaccess может быть размещен в любом каталоге. Директивы этого файла действуют на все файлы в текущем каталоге и во всех его подкаталогах (если эти директивы не переопределены директивами файлов .htaccess во вложенных каталогах).

Изменения, вносимые в файлы .htaccess, вступают в силу немедленно и не требуют перезагрузки сервера в отличие от изменений, вносимых в главный конфигурационный файл httpd.conf.

Для того чтобы файлы .htaccess можно было использовать, необходимы соответствующие настройки главного конфигурационного файла httpd.conf, где должны быть прописаны директивы, которые разрешат файлу .htaccess переопределять конфигурацию Web-сервера в каталоге. Список этих директив задается директивой `AllowOverride`.

Директива `AllowOverride` может включать в себя одну из следующих директив или их комбинацию:

- `AuthConfig` — разрешает использование директив аутентификации и управления доступом (таких как `AuthDBMGroupFile`, `AuthDBMUserFile`, `AuthType`, `AuthName`, `AuthUserFile`, `AuthGroupFile`, `Require`);
- `FileInfo` — разрешает использование директив, управляющих типами документов (`AddEncoding`, `AddLanguage`, `AddType`, `DefaultType`, `ErrorDocument`, `LanguagePriority`);
- `Indexes` — разрешает использование директив, управляющими индексами каталогов (таких как `AddDescription`, `AddIcon`, `AddIconByEncoding`,

AddIconByType, DefaultIcon, DirectoryIndex, FancyIndexing, HeaderName, IndexIgnore, IndexOptions, ReadmeName);

- Limit — разрешает применение директив, управляющих доступом к хостам (Allow, Deny, Order);
- Options — разрешает использование директив, управляющих специфическими свойствами каталогов (Options, XBitHack).

Синтаксис директивы следующий:

```
AllowOverride опция_1, опция_2, опция_3 ...
```

Для того чтобы дать директивам файлов .htaccess максимальные права на изменения директив, значение директивы AllowOverride в файле httpd.conf должно быть равно All. Оно является значением по умолчанию.

```
AllowOverride All
```

Запретить переопределение любых директив в конфигурационных файлах .htaccess можно при помощи значения None:

```
AllowOverride None
```

При установке значения директивы AllowOverride в None сервер не читает файлы .htaccess и соответственно управление сервером с помощью этих файлов невозможно. Установка значения None может ускорить ответ сервера.

Замечание

Название конфигурационного файла можно изменить, и например, назвать его не .htaccess, а access.conf. За название этого файла отвечает директива AccessFileName в файле httpd.conf. Изменение названия конфигурационного файла .htaccess не рекомендуется, т. к. это может усложнить дальнейшую поддержку сервера.

Синтаксис файла .htaccess

Перед тем, как будут рассмотрены примеры, остановимся на синтаксисе директив в файлах .htaccess. Пути к файлам и каталогам должны указываться от корня сервера, например, /pub/home/server1/html/.

Если абсолютный путь от корня сервера не известен, то его можно узнать, спросив у администратора сервера, либо посмотрев самостоятельно, запустив на сайте функцию PHP phpinfo(). Данная функция выведет на экран конфигурацию PHP — значение переменной doc_root будет содержать путь от корня сервера до корневого каталога виртуального хоста. Иногда эта переменная не инициализирована, поэтому следует проверить значения переменных: open_basedir, DOCUMENT_ROOT и SCRIPT_FILENAME.

При указании абсолютных URL обязательно должны быть заданы протоколы, например:

```
Redirect / http://www.newsite.ru
```

В файлах .htaccess недопустимы пробелы в указаниях путей к файлам и в названиях самих файлов, т. к. это приводит к генерации кода ответа 500 — ошибка конфигурации сервера: "Internal Server Error". В листинге 2.1 можно видеть пример такого ошибочного файла.

Листинг 2.1. Ошибка в файле .htaccess

```
AuthType Basic
AuthName admin
AuthUserFile  c:/web sites/.htpasswd
<Limit GET POST>
  require user admin
</Limit>
```

Ошибки с указанием пути с пробелами характерны для Windows. Если в имени файла или каталога есть пробелы, то заключите соответствующий параметр в кавычки, как это сделано в листинге 2.2.

Листинг 2.2. Правильный файл .htaccess

```
AuthType Basic
AuthName admin
AuthUserFile  "c:/web sites/.htpasswd"
<Limit GET POST>
  require user admin
</Limit>
```

Далее будут рассмотрены часто встречающиеся задачи конфигурирования сайтов и их решения с помощью файлов .htaccess.

Индексные страницы

Обычно названия индексных страниц по умолчанию определены в директиве `DirectoryIndex` конфигурационного файла httpd.conf, например:

```
DirectoryIndex index.html index.shtml index.htm
```

Могут возникнуть ситуации, когда необходимо изменить состав индексных файлов, например, если нужна индексная страница index.php, а в основном конфигурационном файле httpd.conf она не прописана. Эту задачу можно ре-

шить при помощи файла .htaccess, в котором необходимо создать директиву `DirectoryIndex`, где будут перечислены имена индексных страниц (листинг 2.3).

Листинг 2.3. Определение индексных страниц

```
DirectoryIndex index.php index.html index.shtml index.htm
```

При запросе каталога без указания имени файла сначала будет осуществлен поиск страницы с именем index.php. Если страницы с таким именем нет в каталоге, то аналогичные операции будут произведены с файлом index.html и т. д. до конца списка, пока не будет найдена и открыта соответствующая страница.

Запрет на отображение содержимого каталога при отсутствии индексного файла

Часто требуется запретить отображение списка файлов в каталоге, если не указан или отсутствует индексный файл (листинг 2.4). Например, запретить отображение содержимого каталога с изображениями. Если такой запрет не поставить, то пользователь, обратившийся напрямую к такому каталогу, получит список всех изображений.

Листинг 2.4. Запрет на отображение содержимого каталога

```
Options -Indexes
```

Обработка кодов ответов Web-сервера Apache

Ни один сайт не застрахован от возникновения ошибок. Самой частой ошибкой является переход по ссылке на несуществующую страницу. В этом случае Apache генерирует код ответа 404 и отображает автоматически сгенерированную страницу с сообщением об ошибке. Наличие несуществующих страниц производит плохое впечатление на посетителей сайта. Это впечатление можно сгладить, если вместо стандартных страниц, приевшихся посетителю, подставлять собственные страницы с сообщением об ошибке, на которых будут принесены извинения и предоставлено меню для того, чтобы посетитель мог продолжить работу с сайтом. За назначение страниц-обработчиков кодов ответа протокола HTTP несет ответственность директива `ErrorDocument` (листинг 2.5).

Листинг 2.5. Обработка ошибок Apache

```
ErrorDocument 401 /401.php
ErrorDocument 403 /403.php
ErrorDocument 404 /index.php
ErrorDocument 500 /500.php
```

После директивы `ErrorDocument` следует указать код ответа и страницу, на которую необходимо перенаправить посетителя при возникновении данного кода ответа. Страницы-обработчики можно использовать не только для отображения или маскировки ошибок, но также для их подсчета и сохранения в log-файлах, анализ которых может предотвратить взлом системы администрирования сайта или указать на ошибку в проектировании системы навигации по сайту. В этом случае в код страниц нужно вставить соответствующие скрипты.

В табл. 2.1 приведена расшифровка возможных кодов ответов.

***Таблица 2.1.** Возможные коды ответов*

Ответ	Описание
"400 Bad Request"	В запросе клиента сервер нашел синтаксическую ошибку
"401 Unautorized"	Запрос требует аутентификации пользователя
"403 Forbidden"	Доступ к запрашиваемому ресурсу запрещен. Клиент не должен повторять запрос
"404 Not Found"	Запрашиваемый документ на сервере отсутствует
"405 Method Not Allowes"	Метод запроса, используемый клиентом, неприемлем
"406 Not Acceptable"	Запрашиваемый ресурс недоступен в том формате, который может принимать клиент
"407 Proxy Authentification Required"	Несанкционированный запрос доступа к proxy-серверу. Сервер отправляет заголовок `Proxy-Authentificate` со схемой аутентификации и областью запрашиваемого ресурса
"408 Request Time-Out"	Клиент не завершил свой запрос за время ожидания запроса, заданное серверу
"409 Conflict"	Возник конфликт запроса клиента с другим запросом
"410 Gone"	Запрашиваемый ресурс удален с сервера

Таблица 2.1 (окончание)

Ответ	Описание
"411 Length Required"	В своем запросе клиент должен предоставить заголовок Content-Length, в котором указан размер запроса
"413 Request Entity Too Large"	Сервер отказывается обрабатывать запрос: слишком большое тело сообщения. Сервер может прервать соединение, чтобы клиент не продолжал отправлять этот запрос
"414 Request-URI Too Long"	Сервер отказывает обрабатывать запрос: слишком большой размер заданного URI
"415 Unsupported Media Type"	Сервер отказывается обрабатывать запрос: отсутствует поддержка формата тела сообщения
"500 Internal Server Error"	Ошибка конфигурации сервера или внешней программы
"501 Not Implemented"	Сервер не поддерживает требуемые функции для выполнения запроса
"502 Bad Gateway"	Неверный ответ вышестоящего сервера или proxy-сервера
"503 Service Unavailabel"	Служба временно недоступна
"505 HHTP Version Not Supported"	Версия HTTP, используемая клиентом, не поддерживается

Как выполнять код PHP в файлах HTML?

Обычно PHP-код выполняется в файлах с расширением php. Иногда возникают ситуации, когда необходимо выполнять PHP-код в файлах с другим расширением. Наиболее часто такие задачи встречаются при обновлении сайта, когда в статические страницы нужно внести динамические вставки на PHP, а переименовывать все страницы сайта (изменять расширение) не хочется, т. к. это влечет серьезную работу по изменению всех ссылок в HTML-страницах. В этом случае можно дать указание Web-серверу выполнять PHP-код не только в файлах с расширением php, но и в файлах с расширением html (листинг 2.6).

Листинг 2.6. Выполнять код PHP в файлах HTML

```
RemoveHandler .html .htm
AddType application/x-httpd-php .php .htm .html .phtml
```

Первая строка в листинге 2.6 удаляет обработчик файлов с расширениями html и htm, а вторая строка сообщает серверу о необходимости использовать для файлов с расширениями htm и html обработчик PHP.

Замечание

Следует отметить, что введение дополнительных расширений увеличивает нагрузку на Web-сервер.

Задание кодировки файлов на сервере

Для того чтобы указать серверу, с применением какой кодировки созданы файлы в каталоге, следует использовать директиву `AddDefaultCharset` (листинг 2.7). Указанная кодировка отправляется браузеру в заголовке `Content-Type` и позволит браузеру клиента автоматически переключиться на требуемую кодировку.

Листинг 2.7. Задание кодировки файлов

```
AddDefaultCharset Windows-1251
```

По умолчанию в Apache значение этой директивы установлено в `ISO-8859-1`, что исключает поддержку кириллицы. Далее приведены имена нескольких часто применяемых кодировок:

- `ISO-8859-1` — западноевропейская;
- `ISO-8859-15` — западноевропейская с поддержкой символа Евро;
- `Windows-1251` — кириллица Windows;
- `KOI8-R` — кириллица UNIX;
- `UTF-8` — 8-битовая кодировка Unicode;
- `UTF-7` — 7-битовая кодировка Unicode.

Задание кодировки загружаемых файлов

При загрузке файлов на сервер можно указать, в какой кодировке сервер должен ожидать файл. Для этого служит директива `CharsetSourceEnc` (листинг 2.8).

Листинг 2.8. Задание кодировки загружаемых файлов

```
CharsetSourceEnc windows-1251
```

Иногда возникают ситуации, когда Apache некорректно перекодирует загружаемые на сервер двоичные файлы. В результате файлы оказываются *"битыми"*. Это проблема актуальна при использовании *"русского Apache"*. Для того чтобы исправить такое поведение Apache, следует отменить перекодировку (листинги 2.9 и 2.10).

Листинг 2.9. Отмена перекодировки

```
CharsetDisable On
```

Листинг 2.10. Отмена перекодировки конкретного файла

```
<FILES filename.php>
  CharsetDisable On
</FILES>
```

Отключение директивы *MultiViews*

Включенная на хостинге опция `MultiViews` может вызвать неожиданные проблемы, например, отображение несуществующих страниц сайта. Скажем, на сайте существует страница с адресом **http://www.server.ru/downloads.php**, и если посетители обратятся к несуществующему каталогу **http://www.server.ru/downloads/**, то включенная опция `MultiViews` вместо этого каталога подставит файл downloads.php. Однако подстановка будет выполнена не полностью — пути к изображениям, таблицам стилей и т. п. будут подставлены неверно. То есть страница будет отображена с искажениями. Это может испортить репутацию сайта, особенно если URL такого вида попадут в каталоги поисковых систем, т. к. посетители не осведомлены, что это проделки Web-сервера, а не халатность Web-разработчиков. Для подавления такого поведения Apache опцию `MultiViews` следует отключить (листинг 2.11).

Листинг 2.11. Отключение директивы `MultiViews`

```
Options -MultiViews
```

Запрет доступа к файлам

Для того чтобы посетители не могли получить доступ к служебным файлам из окна браузера, следует запретить доступ к таким файлам. В листингах 2.12—2.17 приведены примеры использования директив запрета `Deny` и разрешения доступа `Allow`.

Примечание

Использование директив `Deny` и `Allow` управляет только доступом к файлам из браузера, либо из другой программы-клиента. Подобные запреты не распространяются на скрипты сервера.

Листинг 2.12. Запрет доступа к файлам из браузера

```
Deny from all
```

При использовании такой директивы будет запрещен доступ из браузера ко всем файлам и каталогам текущего каталога.

Листинг 2.13. Запрет доступа к определенному файлу

```
<Files config.php>
  Deny from all
</Files>
```

Здесь запрещен доступ только к файлам с именем config.php.

Листинг 2.14. Запрет доступа к файлам расширения inc

```
<Files "*.inc">
  Deny from all
</Files>
```

Здесь запрещен доступ к файлам с расширением inc. Директива `<Files>`, при указании имени файлов, позволяет использовать подстановочные символы:

- `?` — любой одиночный символ;
- `*` — любая последовательность символов, исключая символ слеша (/).

Директива `<Files>`, по умолчанию, не работает с регулярными выражениями, но их можно включить, поставив символ тильды (~) в опциях директивы. Синтаксис директивы в этом случае следующий:

`[~] [пробел] [`*регулярное_выражение*`]`

Листинг 2.15. Запретить доступ к файлам с несколькими типами расширений

```
<Files ~ "\.(inc|conf|cfg)$">
  Deny from all
</Files>
```

Запрещен доступ к файлам с расширением inc, conf и cfg.

Для выбора файлов также может применяться директива <FilesMatch>. Она выполняет действия, аналогичные действиям директивы <Files>, но вместо имени файла в качестве параметра принимает регулярное выражение по поиску файлов. Для того чтобы решить задачу из листинга 2.15 с помощью директивы <FilesMatch>, следует модифицировать код следующим образом:

```
<FilesMatch "\.(inc|conf|cfg)$">
  Deny from all
</Files>
```

Листинг 2.16. Запретить доступ с определенного IP-адреса

```
Deny from 195.135.232.70
```

Листинг 2.17. Разрешить доступ только с определенного IP-адреса

```
Order deny,allow
Deny from all
Allow from 195.135.232.70
```

Директива Order позволяет задать порядок, в котором будут выполняться директивы. Сначала выполняется директива запрета доступа (директива Deny), а затем разрешается доступ только для IP-адреса 195.135.232.70 (директива Allow). Если в первой строке поменять порядок следования директив на Order allow,deny, то доступ для IP-адреса 195.135.232.70 не будет открыт, т. к. директива Deny, выполняемая последней, перекроет действие директивы Allow.

Примечание

Следует отметить, что разрешение доступа только с определенного IP-адреса иногда может не срабатывать. Например, в том случае, если на хостинге установлен обратный кэширующий proxy-сервер. Если директивы разрешения доступа не работают, то вам нужно проконсультироваться по этому вопросу со службой поддержки хостинга.

Перенаправление на другой адрес

Часто встречаются задачи, когда все запросы к определенному каталогу или странице нужно перенаправить (redirect) на другой адрес. Например, при реорганизации сайта, когда скрипты переносятся из одного каталога в другой или сайт меняет доменное имя. Для того чтобы посетители, пришедшие по ссылкам из поисковых систем, по ссылкам с других ресурсов или из закладок своих браузеров, не испытывали неудобств, следует организовать переадресацию всех несуществующих URL на новые адреса.

Это можно сделать с помощью директив `Redirect` и `RedirectMatch`. Они сообщают, что ресурс по запрошенному URL отсутствует, и указывают адрес, по которому следует перейти. Директивы `Redirect` посылают браузеру соответствующий заголовок, и уже браузер осуществляет перенаправление (листинги 2.18—2.20).

Листинг 2.18. Глобальное перенаправление на новый адрес

```
Redirect / http://www.mysite.ru/
```

Если в условиях поиска указать /, то перенаправление на новый адрес **http://www.mysite/** будет осуществляться при обращении к корню сайта: **http://www.server.ru/**.

Листинг 2.19. Перенаправление при обращении к определенному файлу

```
Redirect /about/index.php http://www.mysite.ru/company/
```

Листинг 2.20. Перенаправление при обращении к каталогу

```
Redirect /about http://www.mysite.ru/company/
```

Рассмотрим, как будет работать перенаправление, если запросить страницу **http://www.server.ru/about/page1.php**. При обращении в каталог about будет сделана попытка перенаправить запрос по адресу **http://www.mysite.ru /company/page1.php**. То есть имя запрашиваемой страницы присоединится к новому URL, хотя оно не было указано в условиях перенаправления. Если по новому адресу такой страницы нет, то будет выдана стандартная страница с 404-й ошибкой о недоступности страницы. Директива `Redirect` является регистрозависимой. То есть если в запросе написать имя каталога About с большой буквы (**http://www.server.ru/About/**), то перенаправление, указанное в листинге 2.20, не сработает, т. к. в условиях поиска каталог about написан с маленькой буквы.

Директива `RedirectMatch` расширяет функциональность директивы `Redirect`. С ее помощью можно применять регулярные выражения при указании URL, используемых при перенаправлении. Например, нужно сделать перенаправление при запросе любых страниц из каталога about и в том числе при обращении к каталогу без указания страницы (листинги 2.21 и 2.22).

Листинг 2.21. Перенаправление при обращении к любым страницам каталога

```
RedirectMatch /about/.* http://www.mysite.ru/company/
```

Листинг 2.22. Перенаправление при обращении к любым страницам сайта

```
RedirectMatch /.* http://www.mysite.ru/
```

У директив `Redirect` и `RedirectMatch` имеется дополнительный параметр, с помощью которого можно управлять состоянием перенаправления. Данный параметр может принимать следующие значения:

- `permanent` — ресурс перемещен навсегда (листинг 2.23); код состояния 301;
- `temp` — ресурс перемещен временно; код состояния 302;
- `seeother` — ресурс был заменен другим ресурсом; код состояния 303;
- `gone` — ресурс удален навсегда; код состояния 410. Параметр URL директивы `Redirect` должен быть опущен.

В зависимости от передаваемого кода состояния, браузеры могут предпринять различные действия, например, при коде состояния 303 интеллектуальные браузеры могут заменить адреса ссылок в закладках.

Листинг 2.23. Ресурс перемещен навсегда

```
RedirectMatch permanent /.* http://www.mysite.ru/
```

Преобразование адресов

В предыдущем разделе были рассмотрены директивы `Redirect`, с помощью которых можно сделать подмену одних запросов другими. Более широкие возможности по преобразованию URL обеспечивают директивы модуля `mod_rewrite`, который должен быть доступен на сервере.

Чтобы проверить наличие этого модуля, выполните функцию PHP `phpinfo()` и проверьте в таблице `apache` значение строки с именем `Loaded Modules`, в которой описаны загруженные модули Apache.

Примечание

Использование директив модуля `mod_rewrite` увеличивает нагрузку на сервер, поэтому без крайней необходимости не следует их применять.

Для того чтобы правила преобразования работали в опциях вашего сайта (виртуальный хост сервера), в файле httpd.conf должна быть включена директива:

```
Options FollowSymlinks
```

Чтобы включить механизм преобразования адресов, установите директиву `RewriteEngine` в значение `On`:

```
RewriteEngine On
```

Установка этой директивы в значение `Off` отключит механизм преобразования. Таким образом, можно быстро включать и отключать правила преобразования, описанные в файле .htaccess.

Директива `RewriteRule` определяет правила преобразования. Она имеет следующий синтаксис:

```
RewriteRule шаблон_поиска строка_для_замены [флаги]
```

В шаблоне поиска записывается регулярное выражение, применяемое к запрашиваемому URL. В строке для замены могут быть использованы: текст, ссылки на подстроки из регулярного выражения шаблона поиска, значения серверных переменных.

Перечислим некоторые серверные переменные, которые могут быть использованы для преобразования адресов:

- `SERVER_NAME` — имя Web-сервера, например: **www.domain.ru**;
- `SERVER_PORT` — номер порта Web-сервера;
- `DOCUMENT_ROOT` — каталог документов верхнего уровня для Web-сайта, например: /usr/host/mysite/html;
- `HTTP_FORWARDED` — переадресованная ссылка;
- `HTTP_HOST` — имя компьютера Web-сервера;
- `HTTP_REFERER` — адрес страницы, с которой был осуществлен переход на текущую страницу;
- `HTTP_USER_AGENT` — информация о Web-клиенте, который запросил текущую страницу;
- `REMOTE_ADDR` — IP-адрес посетителя;
- `REMOTE_HOST` — имя компьютера посетителя;
- `REQUEST_METHOD` — метод запроса, который был использован при обращении к текущей странице;
- `SCRIPT_FILENAME` — физический путь к запрошенной странице, например: /usr/host/mysite/html/page.php;
- `PATH_INFO` — путь к запрошенной странице от `DOCUMENT_ROOT`;
- `QUERY_STRING` — параметры запроса к странице, например: id=3&parent=4;
- `AUTH_TYPE` — тип используемой аутентификации;

- `REQUEST_URI` — запрошенный URL. Содержит строку запроса после имени сервера, например: /company/test.php?id=3&parent=4;
- `REQUEST_FILENAME` — то же самое, что и `REQUEST_URI`;
- `TIME_YEAR` — текущий год;
- `TIME_MON` — текущий месяц;
- `TIME_DAY` — текущий день;
- `TIME_HOUR` — текущий час;
- `TIME_MIN` — текущая минута;
- `TIME_SEC` — текущая секунда;
- `TIME_WDAY` — текущий день недели;
- `TIME` — текущее время.

Для того чтобы использовать серверные переменные в директивах модуля `mod_rewrite`, их следует писать в формате: `%{`*имя_переменной*`}`. Например:

```
RewriteCond %{HTTP_USER_AGENT} Opera
```

Рассмотрим пример (листинг 2.24).

Листинг 2.24. Скрытая подмена страницы oldpage.html на страницу newpage.html

```
RewriteEngine on
RewriteBase /
RewriteRule  ^oldpage\.html$  newpage.html
```

Здесь, при запросе страницы oldpage.html, URL преобразовывается в запрос к странице newpage.html. При этом подмена URL происходит незаметно для посетителя. В адресной строке продолжает отображаться имя страницы oldpage.html.

Если необходимо использовать внешнюю переадресацию, чтобы посетителю был отображен реальный адрес страницы, то в директиве `RewriteRule` нужно указать флаг `[R]` (листинг 2.25).

Листинг 2.25. Переадресация запроса страницы oldpage.html на страницу newpage.html

```
RewriteEngine On
RewriteBase /
RewriteRule ^oldpage\.html$  newpage.html [R]
```

Рассмотрим задачу подмены обращений к пользовательским каталогам формата **http://www.server.ru/~username/** на обращение по адресам **http://www.server.ru/users/username/** (листинг 2.26).

Листинг 2.26. Подмена URL каталогов пользователей

```
RewriteEngine on
RewriteBase /
RewriteRule   /~([^/]+)?(/*)/  /users/$1/$2 [R]
```

Директива `RewriteCond` позволяет задавать условия при преобразовании адресов. Синтаксис директивы `RewriteCond`:

```
RewriteCond контрольная_строка шаблон_поиска [флаги]
```

Если шаблон поиска найден в контрольной строке, т. е. условие выполнено, то выполняются директивы, указанные сразу за директивой `RewriteCond`, в противном случае управление переходит к следующему блоку директив.

Рассмотрим задачу изменения адресов в зависимости от типа браузера (листинг 2.27).

Листинг 2.27. Изменение адреса в зависимости от типа браузера

```
RewriteEngine On
RewriteBase /
RewriteCond %{HTTP_USER_AGENT}  Opera
RewriteCond %{REQUEST_FILENAME}  server\.ru/$|server\.ru/index\.php
RewriteRule ^(.*) index_opera.php?%{QUERY_STRING} [L]

RewriteCond %{HTTP_USER_AGENT}  Netscape
RewriteCond %{REQUEST_FILENAME}  server\.ru/$|server\.ru/index\.php
RewriteRule ^(.*) index_netscape.php?%{QUERY_STRING} [L]
```

Первое условие проверяет наличие в заголовке `User-Agent` подстроки `Opera`. Если подстрока `Opera` найдена, то проверяется следующее условие: направлен запрос к странице index.php, либо к корню сайта без указания имени файла. В обоих случаях следует подставить индексный файл, адаптированный под браузер Opera — index_opera.php. Флаг `[L]` (Last) говорит о том, что дальнейшее выполнение директив следует прекратить.

Если же первое условие:

```
RewriteCond %{HTTP_USER_AGENT}  Opera
```

не выполнено, то управление переходит на следующий блок, начинающийся с условия:

```
RewriteCond %{HTTP_USER_AGENT}  Netscape
```

В нем проверяется принадлежность браузера к семейству Netscape, и если условие истинно, то выполняется переадресация на страницу index_netscape.php, адаптированную под браузер Netscape. Если же условие проверки принадлежности браузера к Netscape также не выполнено, то преобразование адресов не происходит.

С помощью директив модуля `mod_rewrite` легко подвергать URL модификации в зависимости от времени суток, месяца, года и т. п. (листинг 2.28).

Листинг 2.28. Изменение адреса в зависимости от времени суток

```
RewriteEngine On
RewriteBase /
RewriteCond %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule ^page\.php$ page_day.php?%{QUERY_STRING} [L]
RewriteRule ^page\.php$ page_night.php?%{QUERY_STRING} [L]
```

С помощью серверных переменных `TIME_HOUR` и `TIME_MIN` проверяется текущее время. Если оно лежит в диапазоне от 7:00 до 19:00, т. е. выполняются оба условия `RewriteCond`, то запрошенный адрес преобразуется в запрос к странице page_day.php. Ключ `[L]` прекращает дальнейшую обработку директив. Если хотя бы одно условие не выполнено, то осуществляется запрос к странице page_night.php.

Рассмотрим задачу запрета посещений сайта нежелательными роботами[1]. Управление посещениями сайта роботами возложено на файл robots.txt, но далеко не все роботы следуют указаниям этого файла. В таком случае для блокирования посещений сайта некорректно работающими роботами можно применить директивы модуля `mod_rewrite` (листинг 2.29).

Примечание

Файл robots.txt — это текстовый файл, который используется для управления посещения сайта роботами поисковых систем. С его помощью можно запретить роботам посещать определенные разделы и страницы сайта. Директивы файла robots.txt имеют рекомендательный характер и могут игнорироваться некорректно работающими роботами, поэтому для достижения гарантированного запрета следует использовать директивы Apache.

[1] В терминах протокола HTTP робот поисковой системы относится к интеллектуальному агенту. Это автономная программа, осуществляющая обход Web-ресурса для сбора текстовой информации, расположенной на сайте. Полученная роботом информация используется для построения каталога поисковой системы.

Листинг 2.29. Блокирование посещений сайта нежелательными роботами

```
RewriteEngine On
RewriteBase /
RewriteCond %{HTTP_USER_AGENT} ^robot
RewriteCond %{REMOTE_ADDR} ^196\.56\.78\.18
RewriteRule ^(.*) for_bad_robots.php
```

Здесь в двух условиях `RewriteCond` проверяется имя робота, переданное в заголовке `User-Agent`, а затем IP-адрес, с которого робот пришел на сайт. Если оба условия выполняются, то осуществляется запрос к странице for_bad_robots.php.

Примечание

Более подробно о директивах модуля `mod_rewrite` вы можете почитать по адресу **http://httpd.apache.org/docs-2.0/mod/mod_rewrite.html**, а также в книге М. Дж. Кабира "Сервер Apache 2. Библия пользователя"[1].

Защита сайта с помощью файлов .htaccess и .htpasswd

Защита сайта средствами сервера Apache является одним из самых простых и в то же время достаточно надежных способов. В этом случае не нужно досконально продумывать стратегию безопасности, осуществлять ее проектирование и реализацию в PHP-коде. В данном разделе будут подробно описаны шаги и действия, которые необходимо выполнить для защиты сайта, и приведены примеры файлов .htaccess с директивами аутентификации.

Определение

Аутентификация — процесс, с помощью которого проверяется, что некто является именно тем, за кого он себя выдает. Как правило, проверка включает в себя ввод имени и пароля.

Далее будет рассмотрен самый простой и доступный способ защиты — базовая аутентификация.

При обращении посетителя в защищаемый каталог сервер Apache в ответ на запрос посылает заголовок с кодом 401 (`"authentication required header"`).

---

[1] Мохамед Дж. Кабир. Сервер Apache 2. Библия пользователя: Пер. с англ. — М.: Издательский дом "Вильямс", 2002. — 672 с.

Браузер посетителя принимает заголовок с кодом 401 и выводит окно с полями для ввода имени пользователя и пароля. После ввода имени и пароля эти данные отсылаются назад серверу, который проверяет имя пользователя на предмет нахождения в специальном списке, а пароль на соответствие имени, введенного пользователем. Если все верно, то посетитель получает доступ к ресурсу. Кэширование параметров аутентификации (имя, пароль, область действия) обычно осуществляет только в пределах одного сеанса.

Примечание

При базовой аутентификации имя пользователя и его пароль передаются в сеть в открытом виде в течение всего сеанса, когда посетитель работает с защищенным каталогом. Злоумышленник может перехватить эту информацию, используя сетевой анализатор пакетов. Данный вид аутентификации не должен использоваться там, где нужна реальная защита коммерчески ценной информации.

Примечание

Web-сервер Apache поддерживает еще один вид защиты — digest-аутентификацию. При digest-аутентификации пароль передается не в открытом виде, а в виде хеш-кода, вычисленного по алгоритму необратимого шифрования MD5. Поэтому пароль, перехваченный при сканировании трафика, практически невозможно подобрать, особенно если он имеет большую длину. Для использования digest-аутентификации на сервере должен быть установлен специальный модуль — `mod_auth_digest`.

Для того чтобы защитить сайт, нужно выполнить следующую последовательность действий: создать файл с паролями, переписать его на сервер, создать файл .htaccess с директивами аутентификации и тоже переписать его на сервер в защищаемый каталог.

Создание файла с паролями

Файл с паролями создается утилитой htpasswd.exe. Если на локальной машине установлен Web-сервер Apache, то данная утилита находится в подкаталоге bin корневого каталога сервера.

Для работы с утилитой htpasswd.exe необходим интерфейс работы с командной строкой. Интерфейсом работы с командной строкой обладают такие программы, как Far, Windows Commander и т. п. Здесь будет рассмотрена работа с командной строкой с помощью утилиты cmd, которая входит в поставку Windows 2000/XP. В меню **Пуск** выберите команду **Выполнить**, введите в строку ввода `cmd` (рис. 2.1) и нажмите кнопку **OK**. В результате будет открыто окно утилиты cmd (рис. 2.2).

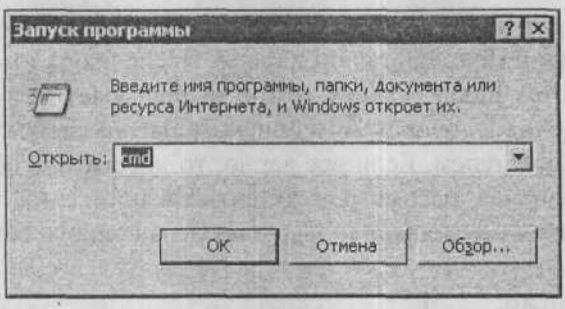


Рис. 2.1. Запуск утилиты cmd

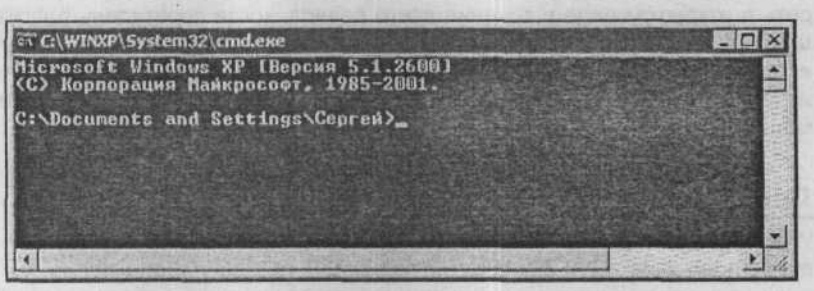


Рис. 2.2. Окно утилиты cmd

Далее необходимо перейти в каталог, где находится утилита htpasswd.exe. Если сервер Apache установлен в каталог c:/Apache2, тогда следует ввести команду:

```
cd../../apache2/bin
```

и нажать клавишу <Enter> (рис. 2.3).

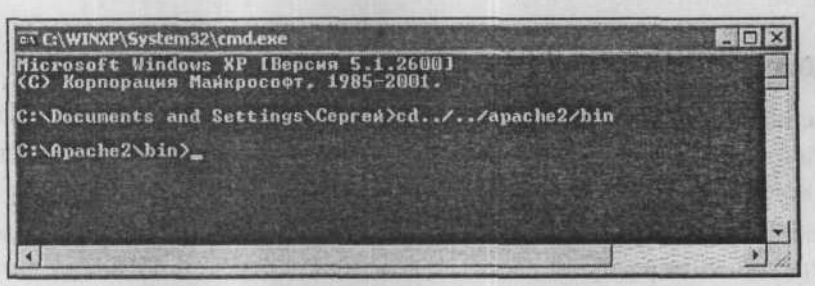


Рис. 2.3. Переход в каталог Apache2/bin

После того как осуществлен переход в каталог bin, следует ввести команду для создания файла с паролем:

```
htpasswd -cm .htpasswd admin
```

Здесь:

- `-cm` — ключи утилиты htpasswd.exe. Ключ `c` указывает на необходимость создания нового файла с паролями. Если файл с таким именем уже существует, он будет перезаписан. Ключ `m` определяет шифрование по алгоритму MD5;
- `.htpasswd` — имя файла с паролями (можно использовать любое имя);
- `admin` — имя посетителя, для которого создается пароль.

В ответ на введенную команду утилита htpasswd.exe предложит ввести пароль и его подтверждение (рис. 2.4). Если оба пароля совпадают, то в завершении отобразится сообщение "Adding password for user admin" и в каталоге c:/Apache2/bin появится файл .htpasswd, в котором будет находиться строка с именем пользователя и хеш-кодом его пароля.

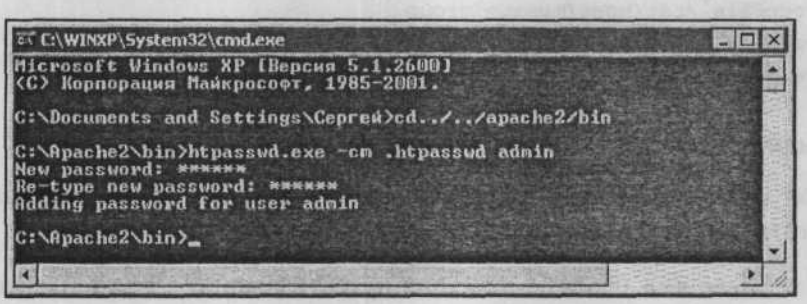


Рис. 2.4. Создание пароля для пользователя admin

Для того чтобы в тот же файл .htpasswd добавить сведения еще об одном пользователе, следует убрать ключ `-c` из команды запуска утилиты htpasswd.exe:

```
htpasswd -m .htpasswd admin
```

Примечание

Если файл с паролями не был создан, то, возможно, некоторые ключи утилиты не поддерживаются в текущей операционной системе. Например, иногда не поддерживается ключ `-m`. Чтобы посмотреть ключи и параметры работы утилиты, следует ввести команду `htpasswd.exe /?`.

Итак, файл с паролями создан. Теперь необходимо переместить его на сервер. Файлы с паролями желательно располагать выше корневого каталога сайта, где к нему нельзя получить доступ из окна браузера.

Если это невозможно, то файлы с паролями следует обязательно защитить, например, с помощью директив файлов .htaccess, описанных в *разд. "Запрет доступа к файлам" ранее в этой главе*:

```
<Files .htpasswd>
   deny from all
</Files>
```

Созданный файл .htaccess с директивами запрета доступа следует поместить в тот каталог, где находится файл с паролями.

Файл с паролем создан и защищен от несанкционированного доступа. Теперь необходимо создать файл .htaccess (листинг 2.30), который будет определять параметры аутентификации.

Листинг 2.30. Пример файла .htaccess

```
AuthType Basic
AuthName "Private zone. Only for administrator!"
AuthGroupFile /usr/host/mysite/group
AuthUserFile  /usr/host/mysite/.htpasswd
require group admins
```

Для защиты каталога используются директивы:

- `AuthType` — тип используемой аутентификации. Для базовой аутентификации эта директива должна иметь значение `Basic`;
- `AuthName` — имя области действия аутентификации. Текст, помогающий посетителю понять, куда он пытается получить доступ. Например, может быть написано: `"Private zone. Only for administrator!"`;
- `AuthGroupFile` — путь к файлу групп, если он существует;
- `AuthUserFile` — путь к файлу с паролями (.htpasswd);
- `Require` — одно или несколько требований, которые должны быть выполнены для получения доступа к закрытой области.

Рассмотрим директивы `AuthUserFile` и `AuthGroupFile` подробнее. В них прописываются абсолютные пути к соответствующим файлам от корня сервера.

Внимание!

Относительные пути работать не будут!

Если защита доступа тестируется на локальной машине в операционной системе Windows и в путях к файлам есть пробелы, то пути к файлам следует заключить в двойные кавычки, например:

```
AuthUserFile  "c:/web sites/.htpasswd"
```

Директива `Require` определяет, кому разрешен доступ к закрытой области. Например:

- `require valid-user` — разрешен доступ всем прошедшим проверку;
- `require user admin alex mango` — разрешен доступ только посетителям с именами `admin`, `alex` и `mango`. Естественно, они должны пройти аутентификацию;
- `require group admins` — разрешен доступ всем пользователям из группы `admins`.

Если к защищаемой области сайта должна иметь доступ большая группа людей, то удобно объединить пользователей в группы и разрешать доступ, определяя принадлежность посетителя к группе.

Формат файла групп очень прост. Это текстовый файл, каждая строка которой описывает отдельную группу. Первым в строке должно идти название группы с двоеточием. А затем через пробел перечисляются посетители, входящие в группу (листинг 2.31).

Листинг 2.31. Пример файла групп

```
admins: admin alex mango
users: guest user max23
```

В группу `admins` входят посетители с именами `admin`, `alex` и `mango`, а в группу `users` — посетители с именами `guest`, `user` и `max23`.

В листингах 2.32—2.35 приведены примеры файлов .htaccess, определяющих параметры аутентификации.

Листинг 2.32. Разрешение доступа всем пользователям, прошедшим авторизацию

```
AuthType Basic
AuthName "Private zone. Only for administrator!"
AuthUserFile  /usr/host/mysite/.htpasswd
require valid-user
```

Листинг 2.33. Разрешение доступа только пользователям `admin` и `root`

```
AuthType Basic
AuthName "Private zone. Only for administrator!"
AuthUserFile  /usr/host/mysite/.htpasswd
require user admin root
```

Листинг 2.34. Разрешение доступа только пользователям из группы admins

```
AuthType Basic
AuthName "Private zone. Only for administrator!"
AuthUserFile  /usr/host/mysite/.htpasswd
AuthGroupFile /usr/host/mysite/group
require group admins
```

Листинг 2.35. Запрет доступа только к файлу private.zip

```
<Files private.zip>
  AuthType Basic
  AuthName "Private zone. Only for administrator!"
  AuthUserFile  /usr/host/mysite/.htpasswd
  require valid-user
</Files>
```

ГЛАВА 3

Массивы

Массивы являются одной из основных и часто встречающихся структур для хранения данных. По определению, массив представляет собой индексированную совокупность переменных одного типа. Каждая переменная или элемент массива имеет свой *индекс*, т. е. все элементы массива последовательно пронумерованы от 0 до *N*, где *N* — *размер* массива.

Создание одномерных массивов

Существует два метода создания одномерных массивов:

- простое присвоение значений элементам массива;
- использование конструкции `array()`.

Первый способ: присвоение значений

Первый способ создания массивов представлен в листинге 3.1. Значение присваивается массиву без указания индекса, что приводит к добавлению элемента в конец массива.

Листинг 3.1. Создание одномерного массива присвоением значений его элементам

```
<?php
  $number[] = "1";
  $number[] = "2";
  $number[] = "3";
  echo($number[0]);
?>
```

В результате выполнения этого скрипта будет выведена цифра 1.

Внимание!

Индексация массивов в PHP начинается с 0, как в любом С-подобном языке программирования.

Посмотреть всю структуру массива сразу можно, если вместо функции `echo()` для вывода значений воспользоваться функцией `print_r()`, как это показано в листинге 3.2.

Листинг 3.2. Вывод структуры массива функцией `print_r()`

```
<?php
  $number[] = "1";
  $number[] = "2";
  $number[] = "3";
  print_r($number);
?>
```

В этом случае будет выведена структура всего массива:

```
Array
(
  [0] => 1
  [1] => 2
  [2] => 3
)
```

При присвоении новых значений элементам массива индекс можно указывать явно (листинг 3.3).

Листинг 3.3. Использование явного указания индекса элементов массива

```
<?php
  $number[2] = "1";
  $number[0] = "2";
  $number[5] = "3";
  print_r($number);
?>
```

В этом случае, запросив вывод значений элементов массива, мы получим следующий результат:

```
Array
(
  [0] => 2
```

```
  [2] => 1
  [5] => 3
)
```

Внимание!

Если при объявлении элементов массива смешиваются переменные с явной индексацией и без индексации, то тому элементу, индекс которого не задан, PHP присвоит первый доступный индекс, после самого большего использованного до сих пор индекса. Например, если мы создадим массив с элементами, индексы которых будут равны, скажем, 5, 15 и 90, а потом создадим элемент, индекс которого явно не укажем, то ему автоматически присвоится индекс 91.

Второй способ: использование конструкции *array()*

Второй способ определения массивов состоит в использовании конструкции `array()` так, как это показано в листинге 3.4.

Листинг 3.4. Использование конструкции `array()`

```
<?php
  $number = array("1","2","3");
  echo($number[1]);    // выводит число 2
?>
```

Для явного указания индексов можно применить оператор => (листинг 3.5).

Листинг 3.5. Указание индекса с помощью оператора =>

```
<?php
  $number = array("1", 5 => "2", "3","4");
  print_r($number);
?>
```

Результат:

```
Array
(
  [0] => 1
  [5] => 2
  [6] => 3
  [7] => 4
)
```

Индексами массива в PHP могут быть не только числа, но и строки, в этом случае массив называется *ассоциативным*, а индексы — *ключами* (листинг 3.6).

Листинг 3.6. Определение ассоциативного массива

```
<?php
  $number = array("one" => "1", "two" => "2");
  print_r($number);
?>
```

Результат:

```
Array
(
  [one] => 1
  [two] => 2
)
```

Создание многомерных массивов

Принцип создания многомерных массивов аналогичен созданию одномерных. Массивы можно создавать, обращаясь к элементам или используя вложенные конструкции `array()`. В листинге 3.7 показан пример формирования многомерного массива.

Листинг 3.7. Создание многомерного массива

```
<?php
  $ship = array(
       "Пассажирские корабли" => array("Лайнер","Яхта","Паром"),
       "Военные корабли" => array("Авианосец","Линкор","Эсминец"),
       "Грузовые корабли" => array("Сормовский","Волго-Дон","Окский")
       );
  print_r($ship);
?>
```

В результате такой инициализации будет создан массив следующей структуры:

```
Array
(
  [Пассажирские корабли] => Array
```

```
    (
        [0] => Лайнер
        [1] => Яхта
        [2] => Паром
    )
    [Военные корабли] => Array
        (
            [0] => Авианосец
            [1] => Линкор
            [2] => Эсминец
        )
    [Грузовые корабли] => Array
        (
            [0] => Сормовский
            [1] => Волго-Дон
            [2] => Окский
        )
)
```

В этом примере создан двумерный массив с количеством элементов 3 × 3, т. е. получилось три массива, каждый из которых содержит в себе по три элемента. Следует отметить, что полученный массив является *смешанным*, т. е. в нем присутствуют как индексы, так и ключи ассоциативного массива — обращение к элементу `$ship['Пассажирские корабли'][0]` возвратит значение `"Лайнер"`.

Обход массива в цикле

Существует три способа обхода массива в цикле:

- с помощью цикла `foreach`;
- с помощью цикла `for`;
- с помощью цикла `while`, используя конструкцию `each—list`.

Цикл *foreach*

Обход массива в цикле можно организовать при помощи цикла `foreach`, который специально создан для ассоциативных массивов и имеет следующий синтаксис:

```
foreach (array as [$key =>] $value)
{
    операторы;
}
```

Смысл этого цикла прост: при проходе каждого элемента массива в переменную $key помещается ключ этого элемента, а в переменную $value — его значение. Имена этих переменных могут быть любыми. Пример — в листинге 3.8.

Листинг 3.8. Обход массива в цикле foreach

```
<?php
  $number = array("first" => "1", "second" => "2", "third" => "3");
  foreach($number as $index => $val) echo "$index = $val <br>";
?>
```

Результат:

```
first = 1
second = 2
third = 3
```

Переменная $key для ключа массива необязательна и может быть опущена (листинг 3.9).

Листинг 3.9. Обход массива в цикле foreach без использования ключа

```
<?php
  $number = array("first" => "1", "second" => "2", "third" => "3");
  foreach($number as $val) echo $val;      // выведет 123
?>
```

Цикл *for*

Обходить массивы в цикле можно не только с помощью цикла foreach, но и используя цикл for (листинг 3.10).

Листинг 3.10. Обход массива в цикле for

```
<?php
  $number = array("1", "2", "3");
  for($i=0; $i < count($number); $i++)
  {
    echo $number[$i];
  }
?>
```

Результат: 123.

Функция `count()`, используемая в этом примере, предназначена для вывода количества элементов массива и имеет простой синтаксис:

```
int count(mixed array)
```

Для массива `$number` из листинга 3.10 эта функция возвратит число 3.

Цикл *while*

С помощью комбинации функций `list`—`each`, используя цикл `while`, также можно организовать перебор элементов массива. Но сначала остановимся на самих функциях `list` и `each`.

Функция `list()` предназначена для помещения элементов массива в отдельные переменные (листинг 3.11).

Листинг 3.11. Использование функции `list()`

```
<?php
  $info = array('sasha','programmer');
  // Присваиваем переменным, перечисленным в list,
  // значения элементов массива $inso
  list($name, $profession) = $info;
  echo "Имя: $name<br>";
  echo "Профессия: $profession";
?>
```

Результат работы функции:

```
Имя: sasha
Профессия: programmer
```

Функция `each()` возвращает пару "индекс — значение" текущего элемента массива и сдвигает курсор (указатель) массива на следующий элемент. Синтаксис функции таков:

```
array each(array arr)
```

В результате функция возвращает массив из четырех элементов:

```
[1] => "значение"
[value] => "значение"
[0] => индекс
[key] => индекс
```

Если курсор достиг конца массива, функция возвращает `false`.

Пример — в листинге 3.12.

Листинг 3.12. Применение функции each()

```
<?php
  $name = array("sasha ", "masha");
  $each_name = each($name);
  print_r($each_name);echo("<br>");
  $each_name = each($name);
  print_r($each_name);
?>
```

Результат:

```
Array ([1] => sasha [value] => sasha [0] => 0 [key] => 0)
Array ([1] => masha [value] => masha [0] => 1 [key] => 1)
```

При каждом применении этой функции происходит сдвиг курсора массива на следующий элемент.

Теперь все готово к тому, чтобы произвести перебор элементов массива с помощью цикла `while` (листинг 3.13).

Листинг 3.13. Обход массива в цикле while

```
<?php
  $number = array("1", "2", "3");
  reset($number);   // Устанавливаем указатель массива на первый элемент
  while(list($key, $val) = each($number))
  {
    echo ($val);    // Результат: 123
  }
?>
```

Функция `reset()` предназначена для установки указателя массива на первый элемент и возвращает значение первого элемента массива.

Замечание

Обход массива с помощью функций `list`—`each` сейчас считается устаревшим. Он применялся при программировании в третьей версии PHP, т. к. в этой версии не было цикла `foreach` (введен в PHP 4).

Обход многомерных массивов

Обход многомерных массивов проводится с помощью вложенных циклов `foreach`, при этом число вложенных циклов соответствует размерности мас-

сива. Для того чтобы нам в цикле вывести элементы массива, созданные в листинге 3.7, надо организовать следующий цикл (листинг 3.14).

Листинг 3.14. Обход многомерных массивов в цикле

```
<?php
  foreach($ship as $key => $type)
  {
    // вывод значений основных массивов
    echo("<b>$key</b>\n"."\n");
    foreach($type as $ship)
    {
      // вывод значений для каждого из массивов
      echo("\t<li>$ship</li>\n");
    }
  }
?>
```

Результат выполнения этого скрипта показан ниже:

Пассажирские корабли

- Лайнер
- Яхта
- Паром

Военные корабли

- Авианосец
- Линкор
- Эсминец

Грузовые корабли

- Сормовский
- Волго-Дон
- Окский

Способы сортировки элементов массивов

Сортировка массива — это расположение его элементов по возрастанию или убыванию значений элементов. Технически это реализуется перестановкой элементов массива в цикле в соответствии с заданными критериями сортировки. PHP достаточно удобен в плане сортировки массивов — в нем имеется немало встроенных функций для сортировки, и вместо того, чтобы думать над циклами, часто достаточно просто вызвать нужную функцию.

Сортировка по возрастанию

Этот вид сортировки производится с помощью функции `sort()`, как показано в листинге 3.15.

Листинг 3.15. Сортировка массива по возрастанию

```
<?php
  $number = array("2", "1", "4", "3","5"); // исходный массив
  echo "до сортировки: <br>";
  for($i=0; $i < count($number); $i++)
  {
    echo "$number[$i] ";
  }
  sort($number); // сортируем массив по возрастанию
  echo "<br>после сортировки: <br>";
  for($i = 0; $i < count($number); $i++)
  {
    echo "$number[$i] ";
  }
?>
```

Результат:

```
до сортировки:
2 1 4 3 5
после сортировки:
1 2 3 4 5
```

Как видно из листинга 3.15, достаточно просто вызвать эту функцию, передав ей в качестве параметра сортируемый массив.

Общий синтаксис функции таков:

```
bool sort(array array[, int sort_flags])
```

Необязательный параметр *sort_flags* указывает, как именно должны сортироваться элементы (или, как говорят, *задает флаги сортировки*). Допустимы следующие значения этого параметра:

- `SORT_REGULAR` — задает нормальное сравнение элементов, т. е. сравнивает элементы, как есть;
- `SORT_NUMERIC` — сравнивает элементы как числа;
- `SORT_STRING` — сравнивает элементы как строки.

Сортировка строк происходит по старшинству первой буквы в алфавите. Такую сортировку часто называют сортировкой в альфа-бета порядке. К примеру, если имеется массив

```
array("one", "two", "abs", "three", "uic", "for", "five"),
```

то применение к нему функции `sort()` приведет к следующему результату:

```
0:abs 1:five 2:for 3:one 4:three 5:two 6:uic
```

Сортировка по убыванию

Сортировка по убыванию производится с помощью функции `rsort()`. Способ применения функции и синтаксис полностью аналогичны функции `sort()` (листинг 3.16).

Листинг 3.16. Сортировка массива по убыванию

```
<?php
  $number = array("2", "1", "4", "3","5"); // исходный массив
  echo("до сортировки: <br>");
  for($i=0; $i < count($number); $i++)
  {
    echo "$number[$i] ";
  }
  rsort($number);   // сортируем массив по убыванию
  echo "<br>после сортировки: <br>";
  for($i=0; $i < count($number); $i++)
  {
    echo "$number[$i] ";
  }
?>
```

Результат:

```
до сортировки:
2 1 4 3 5
после сортировки:
5 4 3 2 1
```

Естественная сортировка

Под *естественной сортировкой* понимается сортировка элементов таким образом, как их отсортировал бы человек ("естественным образом"). Приведем пример. Пусть у нас есть несколько файлов с именами

```
file1.txt
file10.txt
file2.txt
file12.txt
file20.txt
```

При обычной сортировке файлы будут расположены в следующем ("машинном") порядке:

```
file1.txt
file10.txt
file12.txt
file2.txt
file20.txt
```

Естественная же сортировка приведет к следующему результату:

```
file1.txt
file2.txt
file10.txt
file12.txt
file20.txt
```

Выполняется естественная сортировка с помощью функции `natsort()`, принимающей в качестве параметра сортируемый массив (листинг 3.17).

Листинг 3.17. Естественная сортировка

```
<?php
  $array1 = $array2 =
            array("file12.txt", "file10.txt", "file2.txt", "file1.txt");
  sort($array1);
  echo "Обычная сортировка<br>";
  print_r($array1);
  natsort($array2);
  echo "<br>Естественная сортировка<br>";
  print_r($array2);
?>
```

Результат:

```
Обычная сортировка
Array ([0] => file1.txt [1] => file10.txt [2] => file12.txt [3] =>
file2.txt)
Естественная сортировка
Array ([3] => file1.txt [2] => file2.txt [1] => file10.txt [0] =>
file12.txt)
```

Основные операции с массивами

Некоторые операции (создание, обход, сортировку массивов) мы уже рассмотрели в предыдущих разделах, теперь изучим остальные.

Поиск элемента в массиве

Поиск элемента в массиве осуществляется с помощью функции `in_array()`:

```
bool in_array(mixed element, array arr [, bool strict])
```

Эта функция ищет в массиве *arr* значение *element* и возвращает `true`, если оно найдено, и `false` — в противном случае (листинг 3.18).

Листинг 3.18. Поиск элемента в массиве

```
<?php
  $number = array(0.57, '21.5', 40.52);
  if (in_array(21.5, $number)) echo "Значение 21.5 найдено";
  else echo "Ничего не найдено";
?>
```

В результате будет выведена фраза:

```
Значение 21.5 найдено
```

Заметим, что найденный элемент массива взят в одинарные кавычки, т. е. относится к строковому типу, в отличие от других элементов этого же массива. В приведенном варианте использования функции это различие не фиксируется. Для того чтобы функция различала типы элементов в массиве, необходимо третий необязательный параметр *strict* установить в значение `true` (листинг 3.19).

Листинг 3.19. Поиск элемента в массиве с различием по типу

```
<?php
  $number = array(0.57, '21.5', 40.52);
  if (in_array(21.5, $number, true)) echo "Значение 21.5 найдено";
  else echo "Ничего не найдено";
?>
```

В результате будет выведена фраза:

```
Ничего не найдено
```

В этом случае ничего найдено не будет, т. к. тип элемента, передаваемого функции `in_array()` (float), отличается от типа элемента в массиве (string).

При таком вызове функции она найдет элемент 21.5 только, если он будет взят в кавычки (т. е. тоже будет строкового типа):

```
if (in_array('21.5', $number, true))
```

Аналогично функции `in_array()` для поиска заданного ключа в массиве можно воспользоваться функцией `array_key_exists()`:

```
bool array_key_exists(mixed key, array arr)
```

Функция возвращает `true`, если ключ *`key`* найден в массиве *`arr`* (листинг 3.20).

Листинг 3.20. Поиск ключей массива

```
<?php
  $array = array("first_numb" => 1, "second_numb" => 2);
  if (array_key_exists("first_numb", $array) echo "OK";
?>
```

Выборка ключей массива

Задача выборки ключей массива решается с помощью функции `array_keys()`:

```
array array_keys(array arr [,value_of_argument])
```

Эта функция возвращает массив как числовых, так и строковых ключей, содержащихся в массиве *`arr`* (листинг 3.21).

Листинг 3.21. Выборка ключей массива

```
<?php
  $arr = array(0 => 1, "name" => "Саша");
  print_r(array_keys($arr));
?>
```

В результате выполнения данного скрипта мы получим массив ключей массива `$arr`:

```
Array
(
  [0] => 0
  [1] => name
)
```

Пусть требуется вывести не все ключи массива, а только соответствующие определенным элементам. К примеру, из массива имен необходимо извлечь все ключи, соответствующие имени Саша. Для этого следует задать данное значение в необязательном параметре *`value_of_argument`* (листинг 3.22).

Листинг 3.22. Выборка ключей массива по заданному элементу

```
<?php
  $arr = array (0 => "Саша", "1" => "Боря", "2" => "Глеб", 3 => "Саша",
               4 => "Иван", 5 => "Саша");
  print_r(array_keys ($arr, 'Саша'));
?>
```

В результате выполнения этого скрипта мы получим массив ключей, соответствующий элементу "Саша":

```
Array
(
  [0] => 0
  [1] => 3
  [2] => 5
)
```

Суперглобальные массивы

В PHP, для того чтобы передать данные из формы в скрипт-обработчик, можно просто написать в коде формы, к примеру,

```
<input type=text name=name_box value '<? echo $name?> '>
```

В обработчике значение из формы будет помещено в переменную `$name_box`. Это связано с определенным риском, т. к. злоумышленник может передать методом `GET` или `POST` параметры, имена которых совпадут с переменными, используемыми в скрипте, и если последние не инициализируются должным образом, это может повлиять на ход выполнения скрипта.

Примечание

Переход с глобальных переменных на суперглобальные массивы связан в первую очередь с проблемами безопасности создаваемых приложений. Подробнее об этом *см. в гл. 9*.

В связи с этим PHP предоставляет дополнительный набор предопределенных массивов, содержащих переменные Web-сервера, окружения и пользовательского ввода.

Замечание

Директива `register_globalas` конфигурационного файла php.ini, ответственная за прямую передачу параметров в переменные скрипта, отключена по умолчанию в PHP 5.

С введением суперглобальных массивов, при отключенной директиве register_globals, для получения доступа к данным, переданным методами GET и POST, следует обращаться к суперглобальным массивам $_GET и $_POST, соответственно (листинг 3.23).

Листинг 3.23. Использование суперглобальных массивов

```
<?php
  $name = $_GET['name'];
  $test = $_POST['test'];
?>
```

Примечание

Иногда эти переменные называют "автоглобальными", поскольку они определяются автоматически, и пользователь изменить их не может (в PHP нет механизма определяемых пользователем суперглобальных переменных).

Типы суперглобальных массивов

Перечислим суперглобальные массивы, доступные в PHP.

Примечание

Смысл работы с каждым конкретным суперглобальным массивом подробно разобран в соответствующих главах.

- $_GLOBALS

  Содержит ссылку на каждую переменную, доступную в данный момент в глобальной области видимости скрипта. Ключами этого массива являются имена глобальных переменных.

  Дело в том, что по умолчанию переменные, используемые в функциях, имеют локальную область видимости (т. е. если локальная и внешняя переменные имеют одинаковые названия, то работа с локальной переменной никак не скажется на внешней переменной). Для того чтобы локальную переменную сделать глобальной, ранее было необходимо поместить перед ней ключевое слово globals. В этом случае доступ к такой переменной был возможен из любой функции. Теперь для этого необходимо воспользоваться суперглобальным массивом $_GLOBALS. Пример — в листинге 3.24.

Листинг 3.24. Использование глобальных переменных

```
<?php
$var = 10;
```

```
function get_var()
{
  $_GLOBALS["var"] = 20;
  echo($_GLOBALS["var"]);
}
get_var(); // выводит 20, а не 10, т. к. внешняя переменная изменена
?>
```

- `$_SERVER`

  Переменные, установленные Web-сервером либо напрямую связанные с окружением выполнения текущего скрипта.

- `$_GET`

  Переменные, передаваемые скрипту методом `GET`.

- `$_POST`

  Переменные, передаваемые скрипту методом `POST`.

- `$_SESSION`

  Переменные, передаваемые скрипту через механизм сессий *(см. гл. 10)*.

- `$_COOKIE`

  Переменные, передаваемые скрипту через механизм cookies *(см. гл. 10)*.

- `$_FILES`

  Параметры файла, передаваемого скрипту методом `POST`.

- `$_ENV`

  Переменные окружения.

- `$_REQUEST`

  Переменные, передаваемые скрипту через методы `GET`, `POST` и `COOKIE`. Наличие и порядок включения переменных в этот массив определяются в соответствии с директивой конфигурации PHP `variables_order`.

Определение IP-адреса посетителя

Для определения IP-адреса посетителя, пришедшего на страницу, следует обратиться к элементу суперглобального массива `$_SERVER['REMOTE_ADDR']` (листинг 3.25).

Листинг 3.25. Определение IP-адреса посетителя

```
<?php
  echo($_SERVER["REMOTE_ADDR"]);
?>
```

Часто по одному IP-адресу в Интернет выходит не один человек, а целая группа. Это связано с тем, что в большинстве организаций стоят proxy-серверы, обеспечивающие выход всех сотрудников через одну машину (а следовательно, с одного IP-адреса), т. к. один компьютер легче защитить, и в этом случае проще контролировать трафик. Кроме того, в Интернете имеется большое число разнообразных proxy-серверов, решающих различные задачи — анонимный выход в Интернет, доступ в Интернет с мобильных средств связи и т. п. Все пользователи, которые прибегают к услугам proxy-сервера, будут иметь один IP-адрес. В некоторых случаях удается получить адреса машин в подсети, если в качестве пользователя выступает сотрудник организации. Для этого используется переменная окружения `HTTP_X_FORWARDED_FOR` (листинг 3.26).

Листинг 3.26. Определение машин в подсети

```
<?php
  echo(getenv(HTTP_X_FORWARDED_FOR));
?>
```

Блок кода, определяющий IP-адрес с учетом подсети, может выглядеть, к примеру, так, как представлено в листинге 3.27.

Листинг 3.27. Определение IP-адреса с учетом подсети

```
<?php
  // определяем IP-адрес
  $ip = $_SERVER["REMOTE_ADDR"];
  // определяем подсеть, если она доступна
  $subnet = getenv(HTTP_X_FORWARDED_FOR);
  // формируем строку с полным IP-адресом
  if (($subnet != NULL) && ($subnet != $_SERVER["REMOTE_ADDR"]))
    $_SERVER["REMOTE_ADDR"] = $_SERVER["REMOTE_ADDR"]."/".$subnet;
  echo "Страницу $_SERVER[PHP_SELF] посетил
        пользователь с IP-адресом ".$_SERVER["REMOTE_ADDR"];
?>
```

Результат работы скрипта:

```
Страницу /test/test.php посетил пользователь с IP-адресом 127.0.0.1
```

Запрет посещений с определенного IP-адреса

Для запрета посещения сайта с определенного IP-адреса необходимо создать массив IP-адресов и проверять элемент суперглобального массива

$_SERVER['REMOTE_ADDR'] на предмет вхождения его в этот массив (листинг 3.28).

Листинг 3.28. Запрет посещений с определенного IP-адреса

```
<?php
  // провинившиеся IP
  $ip[] = "10.15.242.247";
  $ip[] = "10.15.242.248";
  $ip[] = "10.15.242.249";
  if(empty($_SERVER['REMOTE_ADDR']))
    die("<b>Ваш IP-адрес не определен!</b>");
  elseif (in_array($_SERVER['REMOTE_ADDR'], $ip))
    die("<b>Прочь!</b>");
?>
```

Как узнать, на какой странице я нахожусь?

Задача определения имени текущего скрипта является довольно частой, например, при создании счетчиков посетителей, когда требуется выяснить наиболее посещаемые страницы сайта. Имя текущей страницы записывается в элемент суперглобального массива $_SERVER['PHP_SELF'] (листинг 3.29).

Листинг 3.29. Определение страницы, на которой отрабатывается скрипт

```
<?php
  echo $_SERVER['PHP_SELF'];
?>
```

Результат:

```
/test/test.php
```

Имя сервера помещается в элемент суперглобального массива $_SERVER['SERVER_NAME']. Поэтому полный адрес текущей страницы можно получить из комбинации этих двух элементов (листинг 3.30).

Листинг 3.30. Определение полного адреса страницы, на которой отрабатывается скрипт

```
<?php
  echo "http://".$_SERVER['SERVER_NAME'].$_SERVER['PHP_SELF'];
?>
```

Замечание

Узнать, имеются ли параметры в строке запроса, можно, обратившись к элементу суперглобального массива `$_SERVER['QUERY_STRING']`.

Как узнать, с какой страницы пришел посетитель?

Это тоже очень частая задача, возникающая при разработке систем сбора статистики посещения сайта. Всегда интересно знать, сколько посетителей приходят на сайт с поисковых систем (и с каких именно), с ресурсов, которые ссылаются на ваш сайт, и т. д. Выяснить это можно, обратившись к элементу суперглобального массива `$_SERVER['HTTP_REFERER']` (листинг 3.31).

Листинг 3.31. Определение страницы, с которой пришел посетитель

```
<?php
  echo "Вы пришли со страницы $_SERVER["HTTP_REFERER"]
?>
```

Ловим пауков, или учет пользовательских агентов

Определить, кем является посетитель, можно, анализируя элемент суперглобального массива `$_SERVER['HTTP_USER_AGENT']`, содержащий строку, возвращаемую браузером клиента. В состав этой строки входит информация о типе и версии браузера и операционной системы посетителя.

Вот типичное содержание этой строки:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

Наличие подстроки `"MSIE 6.0"` говорит о том, что посетитель просматривает страницу при помощи Internet Explorer версии 6.0. Строка `"Windows NT 5.1"` сообщает, что в качестве операционной системы используется Windows NT.

Замечание

Для Windows 2000 переменная `$HTTP_USER_AGENT` выглядит следующим образом: `"Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)"`, в то время как для Windows XP — `"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"`.

Если посетитель воспользуется браузером Opera, то содержание переменной `$HTTP_USER_AGENT` может выглядеть следующим образом:

```
Mozilla/4.0 (compatible; MSIE 5.0; Windows 98) Opera 6.04 [ru]
```

Подстрока `"MSIE 5.0"` здесь также присутствует, сообщая, что браузер Opera является совместимым с браузером Internet Explorer и использует те же дина-

мические библиотеки Windows. Поэтому при анализе строки, возвращаемой браузером, следует иметь в виду, что к Internet Explorer относится строка, содержащая подстроку "MSIE 6.0" и не содержащая подстроки "Opera". Кроме того, из данной строки можно заключить, что пользователь работает в операционной системе Windows 98.

При использовании браузера Netscape содержание переменной $HTTP_USER_AGENT может выглядеть следующим образом:

```
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4) Gecko/20030624
Netscape/7.1
```

Принадлежность к этому браузеру можно определить по наличию подстроки "Netscape". Кроме того, можно узнать, что посетитель выходит в Интернет, используя операционную систему Linux с ядром, оптимизированным под Pentium 4, находясь в графической оболочке X-Window.

Этот механизм удобно использовать для сбора статистической информации, которая позволяет дизайнерам оптимизировать страницы под наиболее распространенные браузеры.

Например, запретить загрузку страницы можно так же и по $_SERVER['HTTP_USER_AGENT'] (листинг 3.32), остановив выполнение скрипта PHP, если содержимое переменной $_SERVER['HTTP_USER_AGENT'] нас по каким-то причинам не устраивает.

Листинг 3.32. Запрет на загрузку страницы браузером

```
<?php
  // Если в строке $_SERVER['HTTP_USER_AGENT'], передаваемой браузером,
  // присутствует хоть что-то напоминающее слово Windows, остановить
  // выполнение скрипта, тем самым обезопасив свой ресурс от
  // 98-процентной армии посетителей, использующих Windows в качестве
  // операционной системы.
  if(strpos($_SERVER['HTTP_USER_AGENT'], "Win")!==false) exit();
?>
```

Разумеется, следует поступать очень осторожно и не действовать по принципу "все что не разрешено — запрещено", как это продемонстрировано в листинге 3.32. Не стоит сразу же запрещать посещение страницы роботам, которые кажутся подозрительными. Поисковые роботы всегда работают во благо ресурса, позволяя посетителям быстро находить ваш ресурс. Гораздо интереснее учитывать число посещений роботами страницы, например, для того чтобы выяснить, какие страницы были проиндексированы поисковыми системами.

Замечание

В англоязычной литературе роботов часто называют Web-spiders (Web-пауками).

Если страницы сайта посещаются роботами поисковых систем, по строке `$_SERVER['HTTP_USER_AGENT']` можно определить их принадлежность. Ниже перечислены строки, которые обычно возвращают наиболее известные поисковые роботы:

- `"StackRambler/2.0"` или `"StackRambler/2.0 (MSIE incompatible)')"` — Rambler;
- `"Yandex/1.01.001 (compatible; Win16; I)"` — поисковый робот Yandex;
- `"Googlebot/2.1 (+http://www.googlebot.com/bot.html)"` — Google;
- `"Aport"` — Aport.

Замечание

Если необходимо, чтобы поисковые роботы не индексировали часть страниц вашего ресурса (внутренние форумы, страницы администрирования, служебную информацию), не следует прибегать к вышеописанному способу, для этого нужно разместить в заголовках HTTP-документа строку `<META NAME="robots" CONTENT="noindex">`.

Интересно содержимое `$_SERVER['HTTP_USER_AGENT']` поискового робота Google, в нем приводится адрес, где можно ознакомиться с задачами поискового робота. Это достаточно распространенная практика.

Вот несколько примеров роботов:

```
msnbot/0.11 (+http://search.msn.com/msnbot.htm)
SurveyBot/2.3 (Whois Source)
LWP::Simple/5.50
```

Приведем несколько примеров строк, возвращаемых серверу менеджерами загрузки, которые являются кандидатами на запрет:

- `"Teleport Pro/1.29"` — один из популярнейших менеджеров закачки, позволяющий выкачать сайт с сохранением его структуры;
- `"Download Master"`;
- `"GetRight/4.3"`;
- `"FlashGet`;
- `"ia_archiver"`;
- `"DA 5.0"`.

Замечание

Описываемый в данной главе способ борьбы с нежелательной закачкой содержимого сайта при помощи менеджеров не является универсальным. Так как строку `$_SERVER['HTTP_USER_AGENT']` формирует клиентская сторона, а тем более менеджер, цель которого — во что бы то ни стало скачать страницу, она часто подделывается или просто остается пустой. Так строка `"Mozilla/4.04 [en] (Win95; I; Nav)"`, якобы относящаяся к посетителю с операционной системой Windows 95, является на самом деле некогда распространенным менеджером NetVampire. Тем не менее предложенный способ позволит оградить ваш сайт от неискушенных пользователей.

Не следует использовать именно эти строки для запрета. Сначала нужно выяснить, есть ли вообще проблема, связанная с использованием менеджеров, и какими именно менеджерами злоупотребляют посетители. Для этого необходимо собрать статистику, создав ловушку в начале каждого файла, фиксирующую переменную `$_SERVER['HTTP_USER_AGENT']` каждого из посетителей с сохранением ее в файле или базе данных.

Поддержка нескольких языков

Каждый браузер при запросе к серверу отправляет ему HTTP-заголовок `Accept-Language`, в котором сообщает о своем языковом предпочтении, а точнее, о языковых предпочтениях своего пользователя.

Например, следующий HTTP-заголовок

```
Accept-Language: ru, en; q=0.7
```

можно интерпретировать так: вообще я предпочитаю документы на русском языке (`ru`), но могу читать и английский, если у вас по-русски ничего нет, но знайте, что коэффициент предпочтения английского языка у меня составляет 30% (`en; q=0.7` — `q` является снижением коэффициента предпочтения).

Значение HTTP-заголовка помещается в элемент суперглобального массива `$_SERVER['HTTP_ACCEPT_LANGUAGE']`. Так определить заголовок для браузера можно просто осуществив вывод его значения в окно браузера (листинг 3.33).

Листинг 3.33. Определяем языковые предпочтения посетителя

```
<?php
  echo $_SERVER['HTTP_ACCEPT_LANGUAGE'];
?>
```

Например, браузер Opera, работающий под управлением русской версии операционной системы Windows XP, отправляет серверу следующий HTTP-заголовок:

```
ru;q=1.0,en;q=0.9
```

который можно интерпретировать так: в 90% из 100% пользователь может прочитать английский вариант документа, но все же русскому следует отдавать предпочтение.

Браузер Internet Explorer, работающий под управлением той же операционной системы, лаконичен:

```
ru
```

считая, что если на машине пользователя установлена русская версия Windows XP, то английский язык пользователю не знаком. Анализируя эти строки, всегда можно решить, какую страницу лучше подсунуть посетителю. Следует помнить, что обозначения языков двухбуквенные: `ru` — Российская Федерация, `ua` — Украина, `li` — Литва, `uk` — Англия, `us` — США и т. п.

Вывод случайного элемента массива

Задача выбора случайного элемента массива встречается при организации блоков "Совет дня", "Цитата дня" и т. д. Выбрать случайный элемент можно с помощью функции `array_rand()`:

```
mixed array_rand(array input[, int num_req])
```

В качестве первого параметра *input* передается массив, из которого выбирается случайный элемент. Необязательный параметр *num_req* задает количество сортируемых элементов массива.

Пример — в листинге 3.34.

Листинг 3.34. Вывод случайного элемента массива с помощью функции `array_rand()`

```
<?php
$input = array("Приказы должны обсуждаться только в сторону лучшего их
               исполнения", "Глупость, чтобы не бросаться в глаза, должна
               быть оглушающей", "Мы уже там, куда вас еще и не звали",
               "Он не ведущий научный сотрудник, а уводящий", "Люди
               благодарны тем, кто их бьет, поскольку хорошее отношение
               не ценится так же, как и хорошее здоровье");
$rand_sent = array_rand($input, 2);
echo $input[$rand_sent[0]];
?>
```

Примечание

В версиях PHP до 4.2.0 необходимо перед выборкой случайного элемента инициализировать генератор случайных чисел функцией `srand()` или `mt_srand()`. Начиная с версии PHP 4.2.0, это происходит автоматически.

Выбрать случайный элемент можно также с помощью функции `shuffle()`, принимающей в качестве своего единственного параметра массив *arr*:

```
void shuffle(array arr)
```

Пример — в листинге 3.35.

Листинг 3.35. Вывод случайного элемента массива с помощью функции `shuffle()`

```
<?php
  $str = "Нас еще не согнули";
  $input = array(" годы", " уроды",
                 " законы природы", " любовные походы");
   shuffle($input);
   echo $str.$input[1];
?>
```

Задания

3.1. Создайте многомерный массив, характеризующий кадровый состав университета, который будет включать в себя: названия кафедр, имена преподавателей, их должность, возраст, научную степень. Однородная информация не должна дублироваться в разных элементах массива.

3.2. Выведите в браузер список преподавателей каждой кафедры, отсортированный по алфавиту в обратном порядке.

ГЛАВА 4

Работа со строками

Строки являются основными переменными в PHP, они выступают основными посредниками при операциях с файлами и базами данных, с браузерами и серверами. Можно сказать, что работа со строками — это основная задача PHP. PHP, являясь наследником Perl, предоставляет разнообразные и необычайно гибкие возможности для работы с текстом. В данной главе будут рассмотрены приемы работы со строками при помощи строковых функций.

Кавычки

В языках программирования обычно поддерживают два варианта кавычек: одинарные (') и двойные ("), для того чтобы при необходимости применять одинарные кавычки для обрамления двойных (листинг 4.1).

Листинг 4.1. Одинарные кавычки

```
<?php
  echo 'Проект "Бездна" - самый дорогой проект в истории...';
?>
```

А двойные — для обрамления одинарных (листинг 4.2).

Листинг 4.2. Двойные кавычки

```
<?php
  echo "Переменная принимает значение '345'";
?>
```

В PHP функциональность кавычек была расширена (вернее, заимствована из Perl). Так, если поместить в двойные кавычки переменную, ее значение будет подставлено в текст (листинг 4.3).

Листинг 4.3. Подстановка переменной

```
<?php
  $str = 123;
  echo "Значение переменной - $str"; // Значение переменной - 123
?>
```

Иногда требуется подавить такое поведение переменных. Для этого применяется экранирование знака $ обратным слешем, как это показано в листинге 4.4.

Листинг 4.4. Экранирование знака $

```
<?php
  $str = 123;
  echo "Значение переменной - \$str"; // Значение переменной - $str
?>
```

Точно так же можно применять экранирование для размещения двойных кавычек в строке, обрамленных двойными же кавычками (листинг 4.5).

Листинг 4.5. Экранирование двойных кавычек

```
<?php
  $str = 123;
  echo "Проект \"Бездна\" - самый дорогой проект в истории...";
?>
```

Применение обратного слеша с рядом других символов интерпретируется особым образом. Список таких символов и их значения представлены в табл. 4.1.

Таблица 4.1. *Специальные символы и их значения*

Значение	Описание
\n	Перевод строки
\r	Возврат каретки
\t	Символ табуляции
\\	Обратный слеш
\"	Двойная кавычка
\'	Одинарная кавычка

Размещение переменных и специальных символов (за исключением \') в одинарных кавычках не приводит к их специальной интерпретации (листинг 4.6).

Листинг 4.6. Размещение переменной в строке, обрамленной одинарными кавычками

```
<?php
  $str = 123;
  echo 'Значение переменной - $str';   // Значение переменной - $str
?>
```

Иногда при размещении переменной внутри строки требуется точно указать границы переменной. Для этого имя переменной, значение которой следует подставить в строку, обрамляют фигурными скобками (листинг 4.7).

Листинг 4.7. Применение фигурных скобок

```
<?php
  $text = "Паро";
  echo "Едет $textвоз";       // Неправильно. Выведет только "Едет"
  echo "Плывет $textход";     // Неправильно. Выведет только "Плывет"
  echo "Едет {$text}воз";     // Выведет "Едет Паровоз"
  echo "Плывет {$text}ход";   // Выведет "Плывет Пароход"
?>
```

В первом случае PHP встретит знак доллара ($), найдет пробел и посчитает переменными $textвоз и $textход. Во втором случае мы явно указываем границы переменных.

Точно так же можно использовать в строковых переменных массивы (листинг 4.8).

Листинг 4.8. Работа с массивами

```
<?php
  $wet['root'] = "password";
  // Все записи ниже эквивалентны
  echo "Значение элемента массива ".$wet['root'];
  echo "Значение элемента массива $wet[root]";
  echo "Значение элемента массива {$wet['root']}";
?>
```

PHP унаследовал от Perl третий вид кавычек — так называемые *обратные кавычки* (`). Заключенные в них строки воспринимаются как команды опера-

ционной системы, которые выполняются, а все, что команда печатает в стандартное устройство вывода, возвращается скрипту (листинг 4.9).

Листинг 4.9. Работа с обратными кавычками

```
<?php
  // echo `ls -l`; // UNIX
  echo `dir`;      // Windows
?>
```

Результат работы скрипта:

```
 Том в устройстве E имеет метку MEDIUM
 Серийный номер тома: EC68-EF90
 Содержимое папки E:\main

25.03.2004  20:41    <DIR>          .
25.03.2004  20:41    <DIR>          ..
23.10.2004  00:19               411 config.php
24.07.2004  12:43                54 index.php
03.07.2004  12:55             1 815 main.php
18.09.2003  19:43             2 789 top.php
25.03.2004  20:42    <DIR>          images
11.04.2004  16:30    <DIR>          Projects
25.03.2004  20:43    <DIR>          admin
30.03.2004  22:17    <DIR>          scripts
04.04.2004  12:27    <DIR>          Books
25.04.2004  12:52    <DIR>          tools
12.06.2004  09:46    <DIR>          test
26.09.2004  12:53    <DIR>          Site
               4  файлов          5 069 байт
              10 папок  18 157 846 528 байт свободно
```

Замечание

Обратные кавычки не работают при безопасном режиме.

Форматирование

Строки можно выводить непосредственно в окно браузера, но заимствованные из C функции `printf()` и `sprintf()` позволяют осуществить предварительное форматирование строки и чисел.

Замечание

Функция `printf()` отличается от `sprintf()` тем, что первая функция выводит результат непосредственно в окно браузера, а вторая возвращает строку, которую можно сохранить в переменной.

В качестве первого аргумента функция `printf()` принимает строку форматирования, а в качестве последующих аргументов — переменные, определяемые строкой форматирования (число не ограничено).

Строка форматирования, помимо обычных символов, может содержать специальные последовательности символов, начинающиеся со знака `%`, которые называют *определителями преобразования*. В листинге 4.10 определитель преобразования `%d` подставляет в строку число, которое передается в качестве второго аргумента функции.

Листинг 4.10. Использование определителя преобразования

```
<?php
  printf("Первое число - %d", 26); // Первое число - 26
?>
```

Замечание

Помимо функций `printf()` и `sprintf()` существуют их аналоги — `vprintf()` и `vsprintf()`, которые принимают не переменное число аргументов, а строку форматирования и массив с переменными для определителей преобразования.

Буква `d`, следующая за знаком `%`, определяет тип аргумента (целое, строка и т. д.), поэтому называется *определителем типа*. Различают следующие определители типа:

- `b` — целое число, представляемое в двоичном виде;
- `c` — целое число, представляемое в виде символа с тем же ASCII-кодом;
- `d` — целое число, представляемое в десятичном виде;
- `f` — число с плавающей точкой, представляемое в виде десятичной дроби;
- `o` — целое число, представляемое в восьмеричном виде;
- `s` — строка;
- `x` — целое число, представляемое в шестнадцатеричном виде в нижнем регистре;
- `X` — целое число, представляемое в шестнадцатеричном виде в верхнем регистре.

Пример — в листинге 4.11.

Листинг 4.11. Работа с определителями типа

```
<?php
  $number = 1024;
  printf("Двоичное число: %b<br>", $number);
  printf("ASCII-эквивалент: %c<br>", $number);
  printf("Десятичное число: %d<br>", $number);
  printf("Число с плавающей точкой: %f<br>", $number);
  printf("Восьмеричное число: %o<br>", $number);
  printf("Строковое представление: %s<br>", $number);
  printf("Шестнадцатеричное число (нижний регистр): %x<br>", $number);
  printf("Шестнадцатеричное число (верхний регистр): %X<br>", $number);
?>
```

Результат работы скрипта:

```
Двоичное число: 1011011101011
ASCII-эквивалент: л
Десятичное число: 5867
Число с плавающей точкой: 5867.000000
Восьмеричное число: 13353
Строковое представление: 5867
Шестнадцатеричное число (нижний регистр): 16eb
Шестнадцатеричное число (верхний регистр): 16EB
```

Использование определителя типа `x` для шестнадцатеричных чисел удобно при формировании цвета в HTML-тегах (листинг 4.12).

Листинг 4.12. Преобразование цвета из десятичного в шестнадцатеричный формат

```
<?php
  $red = 256;
  $green = 256;
  $blue = 100;
  printf("#%X%X%X", $red, $green, $blue); // #FFFF64
?>
```

Между символом `%` и определителем типа может быть расположен *определитель заполнения*. Он состоит из символа заполнения и числа, которое определяет, сколько символов отводится под вывод. Все не занятые параметром символы будут заполнены символом заполнителя. Так в листинге 4.13 под вывод числа 45 отводится 5 символов. Так как само число занимает лишь 2 символа, три ведущих символа будут содержать символ заполнения.

Листинг 4.13. Использование определителя заполнения

```
<?php
  echo "<pre>";
  printf("% 4d\n",45);    // "   45"
  printf("%04d\n",45);    // "00045"
  echo "</pre>";
?>
```

Если за знаком % следует число (а не пробел или 0), то оно воспринимается как *определитель ширины поля*, т. е. число символов, отводимых под выводимое значение. При помощи данного поля можно осуществлять выравнивание строк по правому краю (листинг 4.14).

Листинг 4.14. Выравнивание строк по правому краю

```
<?php
  echo "<pre>";
  printf("%20s\n","Новости");
  printf("%20s\n","Форум");
  printf("%20s\n","Фотогалерея");
  printf("%20s\n","Каталог продукции");
  printf("%20s\n","Контакты");
  echo "</pre>";
?>
```

Результат работы скрипта:

```
            Новости
              Форум
         Фотогалерея
   Каталог продукции
           Контакты
```

Применение определителя типа f позволяет вывести число в десятичном формате. Результатом этого является вывод числа с шестью знаками после запятой. Очень часто требуется отобразить строго определенное число символов после запятой, например, для вывода денежных единиц, где обязательным требованием является два знака после запятой. В этом случае прибегают к *определителю точности*, который следует сразу за определителем ширины и представляет собой точку и число символов, отводимых под дробную часть числа (листинг 4.15).

Листинг 4.15. Использование определителя точности

```
<?php
  printf("%8.2f\n",1000.45684);  // 1000.46
  printf("%.2f\n",12.92869);     // 12.93
?>
```

В первом случае под все число отводится 8 символов, два из которых используется для мантиссы числа. Во втором случае ограничение накладывается только на количество цифр после запятой.

Если после числа следует вывести знак процента (`%`), его необходимо передавать в качестве аргумента для определителя типа `s`. Помещение его в строку форматирования без определителя типа приведет к синтаксической ошибке (листинг 4.16).

Листинг 4.16. Вывод числа в процентном формате

```
<?php
  // printf("%.2f%s\n",12.92869); // Ошибка
  printf("%.2f%s\n",12.92869,"%"); // 12.93%
?>
```

Помимо семейства функций `printf()` в PHP также имеется функция `number_format()`, позволяющая форматировать числа. Функция имеет следующий синтаксис:

```
string number_format(float number,
                     [[int decimals,]
                     string dec_point,
                     string thousands_sep])
```

Функция возвращает отформатированное число *number* с *decimals* знаками после запятой. При этом в качестве десятичного разделителя будет использован символ *dec_point*, а тысячи будет разделять символ *thousands_sep*.

Пример — в листинге 4.17.

Листинг 4.17. Использование функции `number_format()`

```
<?php
  $number = 1234.56;
  // английский формат (по умолчанию)
  $english_format_number = number_format($number);
  // 1,234
```

```
  // французский формат
  $nombre_format_francais = number_format($number, 2, ',', ' ');
  // 1 234,56

  $number = 1234.5678;
  // английский формат без разделителей групп
  $english_format_number = number_format($number, 2, '.', '');
  // 1234.57
?>
```

Сравнение строк

Осуществлять сравнение строк можно при помощи оператора ==, как и любые другие переменные PHP. При сравнении учитывается регистр строк. Поэтому, для того чтобы сравнение строк выполнялось без учета регистра, необходимо привести строки к верхнему регистру при помощи функции `strtoupper()` или к нижнему с использованием функции `strtolower()`. Пример — в листинге 4.18.

Листинг 4.18. Сравнение строк

```
<?php
  $str = "строка текста";
  // Строки равны
  if($str == "строка текста") echo "Строки равны<br>";
  else echo "Строки не равны<br>";
  // Строки не равны
  if(strtoupper($str) == $str) echo "Строки равны<br>";
  else echo "Строки не равны<br>";
  // Строки равны
  if(strtolower($str) == $str) echo "Строки равны<br>";
  else echo "Строки не равны<br>";
?>
```

В PHP также имеется специальная функция, осуществляющая сравнение строк — `strcmp()`. Она принимает два строковых аргумента и возвращает положительное число, если первая строка больше второй, ноль, если строки равны, и отрицательное, если первая строка больше второй. "Величина" строки вычисляется согласно кодам символов.

Примечание

Имеется аналог данной функции — функция `strcasecmp()`, которая осуществляет сравнение без учета регистра. Кроме этого, для обеих функций сравнения

существует еще два варианта — это функции `strncmp()` и `strncasecmp()`, которые помимо первых двух параметров принимают третий числовой аргумент, определяющий число символов в строках, которые подвергаются сравнению.

Примечание

В PHP также имеются функции `ucfirst()` и `ucwords()`. Первая из функций производит преобразование первого символа передаваемой ей в качестве аргумента строки в верхний регистр. Вторая функция осуществляет такое преобразование для каждого слова в строке.

К сожалению, приведенные выше функции не всегда корректно преобразуют символы кириллицы из верхнего регистра в нижний и наоборот. Эту проблему поможет решить код из листинга 4.19, который принудительно преобразует символы из одного регистра в другой с помощью функции `ststr()`.

Листинг 4.19. Преобразование символов кириллицы из верхнего регистра в нижний

```
<?php
  $from = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
  $from = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
  $str = "ВСЕ БОЛЬШИЕ БУКВЫ";
  $str = strtr($str, $from, $to);
  echo $str;
?>
```

Этот скрипт выведет строку, преобразованную в нижний регистр: `"все большие буквы"`.

Поиск в тексте

Задача поиска в строках является очень распространенной при создании Web-приложений. Существуют два способа решения этой проблемы: при помощи строковых функций и регулярных выражений. Регулярные выражения, которые будут рассмотрены в *гл. 5*, позволяют решать более сложные задачи, и в то же время они менее производительны, т. к. поиск с использованием регулярных выражений требует значительных расчетов.

Одной из основных строковых функций этого класса является функция `substr()`, имеющая следующий синтаксис:

```
string substr(string str, int start [, int length])
```

Функция `substr()` возвращает часть строки. Первый аргумент функции *`str`* — исходная строка, из которой вырезается текст; второй *`start`* — положение

в строке, которую нужно вернуть, первого символа (отсчет начинается с нуля); третий *length* — длина возвращаемой строки в символах. Если третий аргумент не указан, то возвращается вся оставшаяся часть строки.

Пример — в листинге 4.20.

Листинг 4.20. Применение функции substr()

```
<?php
  $str = "04.05.2005";
  echo "день - ".substr($str,0,2)."<br>";     // день - 04
  echo "месяц - ".substr($str,3,2)."<br>";    // месяц - 05
  echo "год - ".substr($str,6)."<br>";        // год - 2005
?>
```

Еще одной функцией поиска является strpos(), которая имеет следующий синтаксис:

```
string strpos(string str, string needle[, int offset])
```

Эта функция возвращает позицию в строке *str*, с которой начинается переданная ей подстрока *needle*.

Пример — в листинге 4.21.

Листинг 4.21. Использование функции strpos()

```
<?php
  $str = "PHP является интерпретируемым языком";
  echo substr($str, strpos($str, "является"));
  // Выводит "является интерпретируемым языком"
?>
```

Пример в листинге 4.21 ищет позицию, с которой начинается подстрока "является", и выводит в окно браузера строку, начиная с этой позиции до конца строки.

Еще одной востребованной задачей является разбивка строки типа "ключ значение" на подстроки "ключ" и "значение" (листинг 4.22).

Листинг 4.22. Разбиение строки на две части

```
<?php
  $str = "ключ значение";
  $key = substr($str, 0, strpos($str, " "));      // ключ
  $value = substr($str, strpos($str, " ") + 1);   // значение
?>
```

Разбивка строк вида "первый второй третий" требует уже других способов, которые будут рассмотрены в *разд. "Разбивка строк на подстроки" далее в этой главе*.

Замечание

Помимо функции `strpos()`, в PHP имеется функция `strrpos()`, которая аналогична функции `strpos()`, за исключением того, что ищется не первое вхождение подстроки, а последнее.

Функции поиска редко используются сами по себе. В основном они применяются в комбинации с остальными функциями, что будет продемонстрировано в следующей главе.

Замена в тексте

Функция замены `str_replace()` позволяет заменить подстроку в тексте на другую подстроку и имеет следующий синтаксис:

```
string str_replace(string from, string to, string str)
```

Функция заменяет в строке *str* все вхождения подстроки *from* на *to* и возвращает результат. Функция `str_replace()` может работать также с двоичными строками.

Одной из распространенных задач является замена собственных тегов форматирования на их HTML-эквиваленты (листинг 4.23).

Листинг 4.23. Пример использования функции `str_replace()`

```
<?php
    // Исходная строка
    $postbody = "[b]Это[/b] очень жирный [b]текст[/b].";
    $postbody = str_replace("[b]","<b>",$postbody);
    $postbody = str_replace("[/b]","</b>",$postbody);
    echo $postbody;
?>
```

Результатом работы скрипта в листинге 4.23 будет замена всех символов `[b]` на `<b>`, а `[/b]` на `</b>`.

В листинге 4.23 представлено самое простое решение — если пользователь системы забудет добавить завершающий символ `[/b]`, весь текст после этого отсутствующего символа будет выделен жирным стилем. Для того чтобы избежать этого, необходимо перед заменой выполнить поиск завершающего тега и осуществить замену только в случае удачного поиска, как это продемонстрировано в листинге 4.24.

Листинг 4.24. Поиск завершающих тегов

```
<?php
  // Исходная строка
  $postbody = "[b]Это[/b] очень жирный [b]текст[/b].";
  // Результирующая строка
  $result = "";
  // Вспомогательные переменные
  $lastocc = 0;
  $sndocc = 1;
  while($sndocc)
  {
    // Начало жирного фрагмента
    $fstocc = strpos($postbody, "[b]", $lastocc);
    // Завершение жирного фрагмента
    $sndocc = strpos($postbody, "[/b]", $fstocc);
    if(($fstocc>0 && $sndocc>0 && $lastocc>0) ||
       ($fstocc >= 0 && $sndocc>0 && $lastocc == 0))
    {
      // Помещаем фрагмент до тега [b] в строку $result
      $result .= substr($postbody, $lastocc, $fstocc - $lastocc);
      // Жирный фрагмент
      $result .= "<b>".substr($postbody,
                              $fstocc + 3,
                              $sndocc - $fstocc - 3)."</b>";
      $lastocc = $sndocc + 4;
    }
    else
    {
      // Подбираем остатки строки
      $result .= substr($postbody, $lastocc, strlen($postbody)-$lastocc);
      // Выходим из цикла
      break;
    }
  }
  echo $result;
?>
```

Как видно из листинга, нам даже не понадобилась функция `str_replace()`.

Еще одной часто встречающейся задачей является замена символа перевода строки `\n` на его HTML-эквивалент — `<br>` (браузеры не воспринимают символ перевода строки `\n`). Данную задачу также можно решить при помощи функции `str_replace()` — листинг 4.25.

**Листинг 4.25. Замена символа перевода строки на тег
**

```
<?php
  $str = "Текст, содержащий\nперевод строки";
  echo str_replace("\n", "<br>", $str);
?>
```

Для задачи, решаемой в листинге 4.25, имеется специальная функция — `nl2br()`, которой следует отдавать предпочтение, т. к. она короче в написании и быстрее в исполнении.

Специальным видом замены является удаление подстрок из строки. Например, скрипт из листинга 4.26 удаляет из текста все теги и .

**Листинг 4.26. Удаление тегов и **

```
<?php
  $str = "<b>Это</b> очень жирный <b>текст</b>.";
  $str = str_replace("<b>", "", $str);
  $str = str_replace("</b>", "", $str);
  echo $str;
?>
```

Из всех операций удаления подстрок наиболее часто встречающейся является операция удаления начальных и конечных пробелов. Для этого предназначены функции семейства `trim()`:

- `ltrim()` — удаляет из строки начальные пробелы;
- `rtrim()` — удаляет из строки конечные пробелы;
- `trim()` — удаляет из строки и начальные, и конечные пробелы.

Листинг 4.27. Семейство функций `trim()`

```
<?php
  $str = "  строка  ";
  $lstr = ltrim($str); // "строка  "
  $rstr = rtrim($str); // "  строка"
  $str = trim($str);   // "строка"
?>
```

Функция `substr_replace()` заменяет в исходной строке одни подстроки на другие и имеет следующий синтаксис:

```
string substr_replace (string str, string replacement,
                       int start[, int length])
```

Она возвращает строку, в которой часть исходной строки *str* от символа с позицией *start* и длиной *length* заменяется строкой *replacement*. Если аргумент длины *length* не указан, замена проводится до конца строки. Если значение аргумента *start* положительно, то отсчет выполняется от начала строки *str*, в противном случае — от конца строки. В случае неотрицательного значения *length* данный аргумент указывает длину заменяемого фрагмента. Если же *length* отрицательно, то обозначает количество символов от конца строки *str* до последнего символа заменяемого фрагмента.

Разбивка строк на подстроки

Ранее в *разд. "Поиск в тексте" данной главы* был приведен пример извлечения подстрок, разделенных пробелом (см. листинг 4.22). Разбиение строки вручную не всегда является приемлемым, особенно в том случае, когда за один раз следует извлечь большое число подстрок. PHP обладает большим набором функций для разбивки строки на подстроки по определенному символу, которые будут рассмотрены в этом разделе. Первая функция `explode()` предназначена для разбивки строки по определенному разделителю и имеет следующий синтаксис:

```
array explode(string separator, string str[, int limit])
```

Функция возвращает массив строк, каждая из которых соответствует фрагменту исходной строки *str*, находящемуся между разделителями, определяемыми аргументом *separator*. Необязательный параметр *limit* задает максимальное количество элементов в массиве. Оставшаяся (неразделенная) часть будет содержаться в последнем элементе.

Пример — в листинге 4.28.

Листинг 4.28. Функция `explode()`

```
<?php
  $str = "Имя, Фамилия, e-mail";
  $exp_str = explode(",", $str);
  print_r ($exp_str);
?>
```

Результат:

```
Array
(
  [0] => Имя
  [1] => Фамилия
  [2] => e-mail
)
```

Замечание

Функция `implode()` является обратной к `explode()` функцией и осуществляет объединение элементов массива в строку.

Если число элементов в массиве известно заранее, можно воспользоваться функцией `list()`, которая разбивает массив на переменные (листинг 4.29).

Листинг 4.29. Функция `list()`

```
<?php
  $str = "04.05.2005";
  list($day, $month, $year) = explode(".", $str);
  echo "$day<br>";   // 04
  echo "$month<br>"; // 05
  echo "$year<br>";  // 2005
?>
```

Другой функцией, позволяющей разбить строку на подстроки, является `strtok()`, которая имеет следующий синтаксис:

```
string strtok(string str, string separate)
```

Функция `strtok()` возвращает строку по частям, а именно возвращает часть строки *str* до разделителя *separate*. При последующих вызовах функции возвращается следующая часть до следующего разделителя, и так до конца строки. При первом вызове функция принимает два аргумента: исходную строку *str* и разделитель *separate*. Обратите внимание, что при каждом последующем вызове *str* указывать не следует, иначе будет возвращаться первая часть строки (листинг 4.30).

Листинг 4.30 Функция `strtok()`

```
<?php
  $str = "Предложение может содержать\nпробелы,
          запятые и символы\nпереноса";
  $tok = strtok($str,",\n");
  while($tok)
  {
    echo "word=$tok<br>";
    $tok = strtok(",\n");
  }
?>
```

При помощи данной функции можно извлекать параметры и их значения из строки запроса (листинг 4.31).

Листинг 4.31. Извлечение подстрок из строки запроса

```
<?php
  $str = "http://www.softtime.ru/forum/
          read.php?id_forum=1&id_theme=961&id_post=6806";
  $tok = strtok($str,"?&");
  while($tok = strtok("?&"))
  {
    echo "$tok<br>";
  }
?>
```

Результат:

```
id_forum=1
id_theme=961
id_post=6806
```

Достаточно удобной является функция `str_word_count()`, позволяющая как разбивать строку на отдельные слова, так и возвращать число слов в строке. Функция имеет следующий синтаксис:

```
mixed str_word_count(string str[, int format])
```

Функция принимает строку *str* и необязательный параметр *format*, определяющий, какую информацию следует возвратить о строке. В случае его отсутствия возвращается количество слов в строке. Ниже описаны допустимые значения аргумента *format* и соответствующие им возвращаемые значения:

- 1 — возвращается массив, содержащий все слова, входящие в строку *str*;
- 2 — возвращается массив, индексами которого являются позиции в строке, а значениями — соответствующие слова.

Пример — в листинге 4.32.

Листинг 4.32. Функция `str_word_count()`

```
<?php
  $str = "Hello world!    Hello PHP!";

  $a = str_word_count($str, 1);
  $b = str_word_count($str, 2);
  $c = str_word_count($str);

  print_r($a);
  print_r($b);
```

```
  echo $c;
?>
```

Результат работы функции:

```
Array
(
    [0] => Hello
    [1] => world
    [2] => Hello
    [3] => PHP
)
Array
(
    [0] => Hello
    [6] => world
    [16] => Hello
    [22] => PHP
)
4
```

Функция `str_split()` преобразует строку в массив и имеет следующий синтаксис:

```
array str_split(string str[, int split_length])
```

Строка *str* преобразуется в массив. Если указан необязательный аргумент *split_length*, возвращаемый массив будет содержать части исходной строки длиной *split_length* каждая, иначе каждый элемент будет содержать один символ.

Если *split_length* меньше 1, возвращается `false`. Если *split_length* больше длины строки *str*, вся строка будет возвращена в первом и единственном элементе массива.

Пример — в листинге 4.33.

Листинг 4.33. Функция `str_split()`

```
<?php
  $str = "Потрошим строку";
  $arr1 = str_split($str);
  $arr2 = str_split($str, 3);
  print_r($arr1);
  print_r($arr2);
?>
```

Результат работы скрипта:

```
Array
(
    [0] => П
    [1] => о
    [2] => т
    [3] => р
    [4] => о
    [5] => ш
    [6] => и
    [7] => м
    [8] =>
    [9] => с
    [10] => т
    [11] => р
    [12] => о
    [13] => к
    [14] => у
)
Array
(
    [0] => Пот
    [1] => рош
    [2] => им
    [3] => стр
    [4] => оку
)
```

Еще одной задачей по разбивке строки является перенос. Часто в HTML требуется ограничить количество символов на одной строке, т. к. слишком длинное слово или предложение может нарушить дизайн страницы. Для этого предназначена функция `wordwrap()`, которая осуществляет перенос на заданное количество символов с использованием символа разрыва строки. Функция имеет следующий синтаксис:

```
string wordwrap(string str
                [, int width [, string break [, boolean cut]]])
```

Функция разбивает блок текста *str* на несколько строк, которые завершаются символами *break* (по умолчанию это перенос строки — \n), так, чтобы в одной строке было не более *width* букв (по умолчанию 75). Поскольку разбиение происходит по границам слов, текст остается вполне читаемым (листинг 4.34).

Листинг 4.34. Разбиение текста функцией `wordwrap()`

```
<?php
  $str = "Здесь может быть любой текст";
  $mod_str = wordwrap($str,10,"<br>");
  echo($mod_str);
?>
```

Результат:

```
Здесь
может быть
любой
текст
```

Если аргумент *cut* установлен в 1, разрыв делается точно в заданной колонке (листинг 4.35). Поэтому если исходная строка содержит слово длиннее, чем указанное количество символов, то в этом случае слово будет разорвано.

Листинг 4.35. Разрыв строки в конкретной колонке

```
<?php
  $text = "Очень длинное слоооооооооооооооово.";
  $newtext = wordwrap($text, 8, "<br>", 1);
  echo "$newtext";
?>
```

Результат:

```
Очень
длинное
слоооооо
оооооооо
оово.
```

Работа с символами

Следует помнить, что строки являются по своей сути массивами символов, завершающихся нулевым символом, поэтому к любому символу строки можно обратиться по его индексу, который начинается с 0: `$str[0]`, `$str[1]`, `$str[2]` и т. д. Пример — в листинге 4.36.

Листинг 4.36. Обращение к символам строки по их индексам

```
<?php
  $text = "Строка - это массив символов";
  echo $text[0];   // "С"
  echo $text[1];   // "т"
?>
```

Некоторые базы данных формируют отчет или дамп базы данных, размещая периодически через определенное число строк символы перевода страницы (символ с кодом 12 из таблицы ASCII-кодов), что затрудняет автоматическую обработку таких данных. Проблему можно решить, если заменить этот символ переводом строки, используя ранее рассмотренную функцию `str_replace()`. Но символ перевода страницы не так просто набрать с клавиатуры.

Замечание

В Windows символы перевода строки и перевода страницы предваряются символом возврата каретки — `\r`.

Для решения таких задач в PHP введена функция `chr()`, которая возвращает символ по его ASCII-коду и имеет следующий синтаксис:

```
string chr(int ascii)
```

Пример — в листинге 4.37.

Листинг 4.37. Пример использования функции `chr()`

```
<?php
  $text = str_replace(chr(12), "\n", $text);
?>
```

В файлах могут быть различные управляющие символы, не воспроизводимые редактором. Для определения их значения может быть полезной функция `ord()`, которая возвращает ASCII-код символа:

```
int ord(string str)
```

Если в строке *str* содержится более одного символа, ASCII-код будет возвращен только для первого.

Пример — в листинге 4.38.

Листинг 4.38. Пример использования функции ord()

```
<?php
  $str = ord('$');
  echo($str);
?>
```

Результат: 36.

Еще одной интересной функцией, позволяющей получить статистическое распределение символов в строке, является функция count_chars(), которая имеет следующий синтаксис:

```
mixed count_chars(string str [, int mode])
```

Функция подсчитывает количество вхождений каждого из символов с ASCII-кодом из диапазона 0—255 в строке *str* и возвращает эту информацию в различных форматах. Необязательный аргумент *mode* по умолчанию равен 0. В зависимости от его значения возвращается:

- 0 — массив, индексами которого являются ASCII-коды, а значениями — число вхождений соответствующего символа;
- 1 — то же, что и для 0, но информация о символах с нулевым числом вхождений не включается в массив;
- 2 — то же, что и для 0, но в массив включается информация только о символах с нулевым числом вхождений;
- 3 — строка, состоящая из символов, которые входят в исходную строку хотя бы раз;
- 4 — строка, состоящая из символов, которые не входят в исходную строку.

Пример — в листинге 4.39.

Листинг 4.39. Функция count_chars()

```
<?php
  $data = "Две в и одна с";
  $result = count_chars($data, 0);
  for ($i=0; $i < count($result); $i++)
  {
    if ($result[$i] != 0)
       echo "'".chr($i)."' встречается в строке $result[$i] раз(а).<br>";
  }
?>
```

Результат:

```
' ' встречается в строке 4 раз(а).
'Д' встречается в строке 1 раз(а).
'а' встречается в строке 1 раз(а).
'в' встречается в строке 2 раз(а).
'д' встречается в строке 1 раз(а).
'е' встречается в строке 1 раз(а).
'и' встречается в строке 1 раз(а).
'н' встречается в строке 1 раз(а).
'о' встречается в строке 1 раз(а).
'с' встречается в строке 1 раз(а).
```

Преобразование кодировок

По историческим причинам в русскоязычном секторе Интернета используется большое число кодировок. Для преобразования строк из одной кодировки в другую предназначена функция `convert_cyr_string()`, которая имеет следующий синтаксис:

```
string convert_cyr_string(string str, string from, string to)
```

Функция `convert_cyr_string` преобразует строку *str* из кодировки *from* в кодировку *to*. Значения аргументов *from* и *to* — одиночные символы, определяющие кодировку:

- `k` — KOI8-R;
- `w` — Windows-1251;
- `i` — ISO8859-5;
- `a` — X-CP866;
- `m` — X-Mac-Cyrillic.

Пример использования функции `convert_cyr_string()` приведен в листинге 4.40, где перекодируется слово "определяющий" из кодировки Windows-1251 в KOI8-R и обратно.

Листинг 4.40. Перекодирование строки функцией `convert_cyr_string()`

```
<?php
  $str = "определяющий";
  echo "'$str' в KOI8-R является '".
       convert_cyr_string($str, "w", "k").
       "'<br>";
?>
```

И вот результат:

```
'определяющий' в KOI8-R является 'ПРТЕДЕМСАЭЙК'
```

Иногда возникает задача конвертирования символов из формата Unicode. Наиболее часто эта задача встречается при работе с форматом XML.

Примечание

Unicode — это новый стандарт кодирования символов, когда один символ может кодироваться несколькими байтами. Это позволяет в одной кодовой таблице закодировать все символы основных мировых языков и таким образом избежать проблем с разночтением. Стандарт Unicode поддерживается тремя формами, 32-битной (UTF-32), 16-битной (UTF-16) и 8-битной (UTF-8).

Для преобразования кодировок многобайтовых строк предназначена функция `mb_convert_encoding()`.

```
string mb_convert_encoding(string str,
                           string to-encoding [, mixed from-encoding])
```

Функция возвращает строку *str*, преобразованную из кодировки *from-encoding* в кодировку *to-encoding*. Рассмотрим блок кода, преобразующий текст в формате UTF-16 в формат Windows-1251 (листинг 4.41).

Листинг 4.41. Преобразование текста

```
<?php
  $data = mb_convert_encoding($data, "Windows-1251", "UTF-16");
?>
```

Работа с URL

Так как PHP в основном ориентирован на разработку Web-приложений, в его арсенале имеется большое число функций для работы с URL.

Функция `parse_url()` позволяет разбить адрес на отдельные компоненты и имеет следующий синтаксис:

```
array parse_url(string url)
```

Функция `parse_url()` обрабатывает URL (листинг 4.42), переданный строкой *url*, и возвращает его компоненты. Массив, возвращаемый функцией, включает множество различных существующих компонентов URL: "scheme", "host", "port", "user", "pass", "path", "query" и "fragment".

Листинг 4.42. Обработка URL

```
<?php
  $url = "http://www.softtime.ru/forum/read.php?id_forum=1&id_theme=80";
  $arr = parse_url($url);
  print_r($arr);
?>
```

Результат:

```
Array
(
    [scheme] => http
    [host] => www.softtime.ru
    [path] => /forum/read.php
    [query] => id_forum=1&id_theme=80
)
```

Отдельные компоненты уже гораздо проще разбить на подстроки при помощи ранее рассмотренных функций. Впрочем, для разбора строки с параметрами ("query") имеется специальная функция `parse_str()`, которая имеет следующий синтаксис:

```
void parse_str(string str [, array arr])
```

Функция `parse_str()` интерпретирует строку *str* так, как если бы эта строка содержала в себе переменные и их значения и передавалась бы в URL. Если задан второй необязательный параметр *arr*, то значения, найденные при помощи этой функции, сохраняются не в глобальных переменных, а в элементах указанного массива.

Пример — в листинге 4.43.

Листинг 4.43. Работа с функцией `parse_str()`

```
<?php
  $str = "id_forum=1&id_theme=80";
  parse_str($str);
  echo $id_forum;          // 1
  echo id_theme;           // 80

  parse_str($str, $arr);
  echo $arr['id_forum'];   // 1
  echo $arr['id_theme'];   // 80
?>
```

Согласно спецификации RFC 1738 (**http://www.ysn.ru/docs/cie/RFC/1738/index.htm**) в URL не допускается использование пробелов, а также символов национальных алфавитов, поэтому для передачи русских слов через строку запроса URL или значение параметра следует преобразовать в безопасный режим при помощи функции `urlencode()`, которая имеет следующий синтаксис:

```
string urlencode(string str)
```

Функция `urlencode()` возвращает строку (листинг 4.44), в которой все не алфавитно-цифровые символы, за исключением дефиса, знака подчеркивания и точки заменены знаком процента (`%`), за которым следуют две шестнадцатеричные цифры, обозначающие код символа.

Замечание

RFC (Request For Comments, просьба прокомментировать) — это серия документов IETF (Internet Engineering Task Force, проблемная группа проектирования Интернета — одна из групп, отвечающая за решение инженерных задач глобальной сети), начатая в 1969 г. и содержащая описания набора протоколов Интернета и связанную с ними информацию.

Листинг 4.44. Работа с функцией `urlencode()`

```
<?php
  echo "http://www.yandex.ru/yandsearch?stype=www&nl=0&text=".
       urlencode("Программирование");
?>
```

Результат:

```
http://www.yandex.ru/yandsearch?stype=www&nl=0&text=%CF%F0%EE%E3%F0%E0%EC
%EC%E8%F0%EE%E2%E0%ED%E8%E5
```

Обратная функция `urldecode()` расшифровывает закодированную `urlencode()` строку и имеет следующий синтаксис:

```
string urldecode(string str)
```

Работа с путями к файлам и каталогам

Основной функцией при работе с путями к файлам и каталогам является функция `pathinfo()`, которая имеет следующий синтаксис:

```
array pathinfo(string path)
```

Функция принимает путь к файлу *path* и возвращает ассоциативный массив, хранящий в своих элементах путь, по которому расположен файл, имя файла и его расширение.

Пример — в листинге 4.45.

Листинг 4.45. Работа с функцией pathinfo()

```
<?php
  $path_parts = pathinfo("C:\www\htdocs\index.html");
  echo $path_parts["dirname"]."<br>";
  echo $path_parts["basename"]."<br>";
  echo $path_parts["extension"]."<br>";
?>
```

Результат:

```
C:\www\htdocs
index.html
html
```

Для того чтобы определить абсолютный путь к файлу, необходимый для функции pathinfo(), можно воспользоваться функцией realpath(), которая имеет следующий синтаксис:

```
string realpath(string path)
```

Функция принимает относительный путь *path* и возвращает абсолютный. При работе с функцией следует помнить, что точкой (".") обозначается текущий каталог, а двумя точками ("..") — каталог на уровень выше (листинг 4.46).

Листинг 4.46. Работа с функцией realpath()

```
<?php
  echo realpath(".");              // Путь к текущему каталогу
  echo realpath("../index.php");   // Путь к файлу index.php уровнем выше
?>
```

Замечание

При разработке Web-приложений чаще требуется путь к текущей странице от DocumentRoot, который можно узнать, обратившись к элементу суперглобального массива $_SERVER['PHP_SELF'].

Для извлечения имени файла существует отдельная функция basename():

```
string basename(string path [, string suffix])
```

Функция принимает путь *path* и необязательное расширение файла *suffix*. Эта функция вернет имя файла, чей путь был передан в качестве параметра. Если имя файла имеет расширение *suffix*, оно будет отброшено (листинг 4.47).

Листинг 4.47. Работа с функцией basename()

```
<?php
  $path = "/home/httpd/html/index.html";
  echo basename ($path);          // Выведет "index.html"
  echo basename ($path, ".html"); // Выведет "index"
?>
```

Результат работы скрипта:

```
index.html
index
```

Функция dirname() извлекает из пути каталог и имеет следующий синтаксис:

```
string dirname(string path)
```

Функция принимает путь *path* и возвращает только каталог (листинг 4.48).

Листинг 4.48. Работа с функцией dirname()

```
<?php
  $path = "c:\www\html\index.html";
  echo dirname($path); // c:\www\html
?>
```

Работа с датой и временем

Для получения текущего времени и даты предназначена функция time(), которая возвращает количество секунд, прошедших с 0:00:00 1 января 1970 г. Такой формат является общепринятым для UNIX-подобных операционных систем, позволяет хранить дату в одной целочисленной переменной и легко выполнять операции сложения и вычитания, не заботясь о числе дней в месяце и високосных годах (листинг 4.49).

Листинг 4.49. Работа с функцией time()

```
<?php
  echo time(); // 1102283188
?>
```

Замечание

В PHP имеется функция `microtime()`, возвращающая текущее время в микросекундах.

Для формирования произвольной временной метки следует воспользоваться функцией `mktime()`, которая имеет следующий синтаксис:

```
int mktime([int hour
           [, int minute
           [, int second
           [, int month
           [, int day
           [, int year
           [, int is_dst]]]]]])
```

Все принимаемые ею аргументы являются необязательными. Если не передан ни один из аргументов, функция вернет текущее время. Значение семи аргументов функции таковы:

- *hour* — часы;
- *minute* — минуты;
- *second* — секунды;
- *month* — месяц;
- *day* — день;
- *year* — год;
- *is_dst* — аргумент *is_dst* может быть установлен в 1, если заданной дате соответствует летнее время, 0 — в противном случае, или –1 (значение по умолчанию), если неизвестно, действует ли летнее время на заданную дату. В последнем случае PHP пытается определить это самостоятельно.

Пример — в листинге 4.50.

Листинг 4.50. Работа с функцией `mktime()`

```
<?php
  echo mktime(0, 0, 0, 2, 24, 2005); // 1109192400
?>
```

Параметр *year* может быть двух- или четырехзначным числом. Значения от 0 до 69 соответствуют 2000—2069, а значения от 70 до 99 соответствуют 1970—1999 (в большинстве современных систем, где время представляется 32-битным целым со знаком, допустимыми являются значения *year* между 1901 и 2038).

Замечание

Ни одна из версий Windows не поддерживает отрицательные метки времени. Поэтому для Windows допустимыми являются значения *year* между 1970 и 2038.

Используя комбинацию функций `time()` и `mktime()`, можно выяснить, например, количество дней между какой-либо датой и сегодняшним днем (листинг 4.51).

Листинг 4.51. Подсчет количества дней за определенный временной интервал

```
<?php
  // Число дней между сегодняшним днем и 26.12.2002
  echo (time() - mktime(0, 0, 0, 12, 26, 2002))/86400;
?>
```

Более удобной в практическом применении является функция `getdate()`, которая имеет следующий синтаксис:

```
array getdate([int timestamp])
```

Через необязательный параметр *timestamp* функции можно передать время (в секундах с 1 января 1970 г.). В случае отсутствия данного параметра функция работает с текущим временем. В качестве результата работы `getdate()` возвращает ассоциативный массив, содержащий ключи, перечисленные в табл. 4.2.

Таблица 4.2. *Ключи ассоциативного массива, возвращаемого функцией* `getdate()`

Ключ	Описание	Диапазон
`"seconds"`	Секунды	От 0 до 59
`"minutes"`	Минуты	От 0 до 59
`"hours"`	Часы	От 0 до 23
`"mday"`	Порядковый номер дня месяца	От 1 до 31
`"wday"`	Порядковый номер дня	От 0 (воскресенье) до 6 (суббота)
`"mon"`	Порядковый номер месяца	От 1 до 12
`"year"`	Порядковый номер года, 4 цифры	Примеры: 1999, 2003
`"yday"`	Порядковый номер дня в году (нумерация начинается с 0)	От 0 до 366

Таблица 4.2 (окончание)

Ключ	Описание	Диапазон
"weekday"	Полное название дня недели	От Sunday (воскресенье) до Saturday (суббота)
"month"	Полное название месяца	От January (январь) до December (декабрь)
0	Количество секунд, прошедших с 1 января 1970 г.	Платформозависимое, в большинстве случаев от –2 147 483 648 до 2 147 483 647

Пример — в листинге 4.52.

Листинг 4.52. Работа с функцией `getdate()`

```
<?php
  $today = getdate();
  print_r($today);
?>
```

Результат:

```
Array
(
    [seconds] => 26
    [minutes] => 35
    [hours] => 1
    [mday] => 6
    [wday] => 1
    [mon] => 12
    [year] => 2004
    [yday] => 340
    [weekday] => Monday
    [month] => December
    [0] => 1102286126
)
```

Функция `date()` позволяет форматировать дату и имеет следующий синтаксис:

```
string date(string format [, int timestamp])
```

Необязательный параметр *timestamp* позволяет задать время (в секундах с 1 января 1970 г.). При его отсутствии функция возвращает текущее время.

Параметр `format` определяет строку форматирования, в которой символы интерпретируются согласно табл. 4.3.

Таблица 4.3. *Символы форматирования функции* `date()`

Символ	Описание	Пример возвращаемого значения
a	Ante meridiem или Post meridiem в нижнем регистре	am или pm
A	Ante meridiem или Post meridiem в верхнем регистре	AM или PM
B	Время в стандарте Swatch Internet	От 000 до 999
c	Дата в формате ISO-8601 (добавлено в PHP 5)	2004-02-12T15:19:21+00:00
d	День месяца, 2 цифры с ведущими нулями	От 01 до 31
D	Сокращенное наименование дня недели, 3 символа	От Mon до Sun
F	Полное название месяца, например January или March	От January до December
g	Часы в 12-часовом формате без ведущих нулей	От 1 до 12
G	Часы в 24-часовом формате без ведущих нулей	От 0 до 23
h	Часы в 12-часовом формате с ведущими нулями	От 01 до 12
H	Часы в 24-часовом формате с ведущими нулями	От 00 до 23
i	Минуты с ведущими нулями	От 00 до 59
I	Признак летнего времени	1, если дата соответствует летнему времени, иначе 0
j	День месяца без ведущих нулей	От 1 до 31
l	Полное наименование дня недели	От Sunday до Saturday
L	Признак високосного года	1, если год високосный, иначе 0
m	Порядковый номер месяца с ведущими нулями	От 01 до 12
M	Сокращенное наименование месяца, 3 символа	От Jan до Dec
n	Порядковый номер месяца без ведущих нулей	От 1 до 12
O	Разница со временем по Гринвичу в часах	Например: +0200
r	Дата в формате RFC 2822	Например: Thu, 21 Dec 2000 16:01:07 +0200

Таблица 4.3 (окончание)

Символ	Описание	Пример возвращаемого значения
s	Секунды с ведущими нулями	От 00 до 59
S	Английский суффикс порядкового числительного дня месяца, 2 символа	st, nd, rd или th. Применяется совместно с j
t	Количество дней в месяце	От 28 до 31
T	Временная зона на сервере	Примеры: EST, MDT
U	Количество секунд, прошедших с 1 января 1970 г.	См. также функцию time()
w	Порядковый номер дня недели	От 0 (воскресенье) до 6 (суббота)
W	Порядковый номер недели года по ISO-8601, первый день недели — понедельник	Например: 42 (42-я неделя года)
Y	Порядковый номер года, 4 цифры	Примеры: 1999, 2003
y	Номер года, 2 цифры	Примеры: 99, 03
z	Порядковый номер дня в году (нумерация с 0)	От 0 до 365
Z	Смещение временной зоны в секундах. Для временных зон западнее UTC это отрицательное число, восточнее UTC — положительное	От –43 200 до 43 200

Пример — в листинге 4.53.

Листинг 4.53. Работа с функцией date()

```
<?php
  echo date("d.m.Y G:i:s"); // 06.12.2004 10:51:53
?>
```

Представленный в листинге 4.54 скрипт выводит приветствие в зависимости от времени суток.

Листинг 4.54. Учет времени суток

```
<?php
  $hour = date("G"); // Получаем текущий час
  if ($hour>=5 &&  $hour<11) echo "Доброе утро!";
  if ($hour>=11 && $hour<17) echo "Добрый день!";
```

```
  if ($hour>=17 && $hour<23) echo "Добрый вечер!";
  if ($hour>=23 || $hour<5)  echo "Доброй ночи!";
?>
```

Еще одной функцией форматирования является функция `strftime()`, которая имеет следующий синтаксис:

```
string strftime(string format [, int timestamp])
```

Необязательный параметр *timestamp* позволяет задать время (в секундах с 1 января 1970 г.). При его отсутствии функция возвращает текущее время. Параметр *format* определяет строку форматирования. В форматирующей строке распознаются следующие символы:

- `%a` — сокращенное название дня недели в текущей локали;
- `%A` — полное название дня недели в текущей локали;
- `%b` — сокращенное название месяца недели в текущей локали;
- `%B` — полное название месяца недели в текущей локали;
- `%c` — предпочтительный формат даты и времени в текущей локали;
- `%C` — столетие (год, деленный на 100 и округленный до целого, от 00 до 99);
- `%d` — день месяца в виде десятичного числа (от 01 до 31);
- `%D` — аналогично `%m/%d/%y`;
- `%e` — день месяца в виде десятичного числа, если это одна цифра, то перед ней добавляется пробел (от 1 до 31);
- `%g` — подобно `%G`, но без столетия;
- `%G` — год, четырехзначное число, соответствующее номеру недели по ISO (см. `%V`). Аналогично `%Y`, за исключением того, что если номер недели по ISO соответствует предыдущему или следующему году, используется соответствующий год;
- `%h` — аналогично `%b`;
- `%H` — номер часа от 00 до 23;
- `%I` — номер часа от 01 до 12;
- `%j` — номер дня в году (от 001 до 366);
- `%m` — номер месяца (от 01 до 12);
- `%M` — минуты;
- `%n` — символ `\n`;
- `%p` — `am` или `pm`, или соответствующие строки в текущей локали;

- %r — время в формате a.m. или p.m.;
- %R — время в 24-часовом формате;
- %S — секунды;
- %t — символ табуляции (\t);
- %T — текущее время, аналогично %H:%M:%S;
- %u — номер дня недели от 1 до 7, где 1 соответствует понедельнику;
- %U — порядковый номер недели в текущем году. Первым днем первой недели в году считается первое воскресенье года;
- %V — порядковый номер недели в году по стандарту ISO-8601 от 01 до 53, где 1 соответствует первой неделе в году, в которой, как минимум, 4 дня принадлежат этому году. Первым днем недели считается понедельник. (Используйте %G or %g для определения соответствующего года.);
- %W — порядковый номер недели в текущем году. Первым днем первой недели в году считается первый понедельник года;
- %w — номер дня недели, 0 соответствует воскресенью;
- %x — предпочтительный формат даты без времени в текущей локали;
- %X — предпочтительный формат времени без даты в текущей локали;
- %y — год без столетия (от 00 до 99);
- %Y — год, включая столетие;
- %Z — временная зона в виде смещения, аббревиатуры или полного наименования;
- %% — символ %.

В заключение раздела рассмотрим скрипт, позволяющий вывести список дней месяца, выходные дни в котором подсвечены красным цветом и жирным стилем (листинг 4.55).

Листинг 4.55. Пример использования временных функций

```
<?php
  // Декабрь 2004 года
  $text = "01.12.2004";
  $hour = 0;
  $minute = 0;
  $second = 0;
  $month = substr($text,3,2);
  $days = substr($text,0,2);
```

```
  $year = substr($text,6,4);
  $timesec = mktime($hour, $minute, $second, $month, $days, $year);
  for($i = 0; $i < date("t",$timesec); ++$i)
  {
   $timesec = mktime($hour, $minute, $second, $month, $days + $i, $year);
    if(date("D", $timesec) == "Sun" ||
       date("D", $timesec) == "Sat")
    {
      echo "<font color=red>".(date("d.m.Y", $timesec))."</font><br>";
    }
    else
    {
      echo "<font color=black>".(date("d.m.Y", $timesec))."</font><br>";
    }
  }
?>
```

Результатом работы скрипта будет серия дат, от 1 до 31 декабря 2004 года:

```
01.12.2004
02.12.2004
03.12.2004
04.12.2004
05.12.2004
06.12.2004
07.12.2004
...
23.12.2004
24.12.2004
25.12.2004
26.12.2004
27.12.2004
28.12.2004
29.12.2004
30.12.2004
31.12.2004
```

Хранение данных

Строки очень часто используются для хранения разнообразных данных. Поэтому PHP имеет целый арсенал функций для упаковки данных в строки и извлечению их. Первыми будут рассмотрены функции для работы с бинарными данными, их две — `pack()` и `unpack()`. Первая осуществляет пакетиро-

вание данных в двоичную строку, а вторая — распаковывает данные из двоичной строки. Синтаксис функции `pack()` следующий:

```
string pack(string format [,mixed $args, ...])
```

Функция `pack()` упаковывает заданные в ее параметре аргументы в бинарную строку. Формат параметров и их количество задается аргументом *format*, при помощи следующих спецификаторов форматирования:

- `a` — строка, свободные места в поле заполняются символом с кодом 0;
- `A` — строка, свободные места заполняются пробелами;
- `h` — шестнадцатеричная строка, младшие разряды в начале;
- `H` — шестнадцатеричная строка, старшие разряды в начале;
- `c` — знаковый байт;
- `C` — беззнаковый байт;
- `s` — знаковое короткое целое;
- `S` — беззнаковое короткое целое;
- `n` — беззнаковое целое, 16 битов, старшие разряды в конце;
- `v` — беззнаковое целое, 16 битов, младшие разряды в конце;
- `i` — знаковое целое;
- `I` — беззнаковое целое;
- `l` — знаковое длинное целое, 32 бита;
- `L` — беззнаковое длинное целое;
- `N` — беззнаковое длинное целое, 32 бита, старшие разряды в конце;
- `V` — беззнаковое целое, 32 бита, младшие разряды в конце;
- `f` — число с плавающей точкой;
- `d` — число двойной точности;
- `x` — символ с нулевым кодом;
- `X` — возврат назад на 1 байт;
- `@` — заполнение нулевым кодом до заданной абсолютной позиции.

После каждого спецификатора может стоять число, которое говорит о том, сколько информации будет обработано данным спецификатором. Для форматов `a`, `A`, `h` и `H` это число задает количество символов, которые будут помещены в бинарную строку из тех, что находятся в параметре-строке (фактически, определяется размер поля вывода строки). Если используется спецификатор `@`, то в этом случае определяется абсолютная позиция, в которую будут помещены данные. Для всех остальных спецификаторов следующие за ними

числа задают количество аргументов, на которые распространяется действие данного формата. Вместо числа можно указать символ *, в этом случае спецификатор действует на все оставшиеся данные. Функция возвращает упакованные данные в шестнадцатеричном формате.

Пример — в листинге 4.56.

Листинг 4.56. Упаковка данных в двоичный формат с помощью функции pack()

```
<?php
  $bin = pack("nvn*",0x5722,0x1148, 65, 66); // запаковываем
  $var = bin2hex($bin); // перекодируем из шестнадцатеричного формата
  echo($var);
?>
```

Результат:

```
0x57, 0x22, 0x48, 0x11, 0x00, 0x41, 0x00, 0x42
```

Согласно заданному нами формату `nvn*`, первое число мы возвращаем как беззнаковое целое со старшими разрядами в конце, второе тоже как беззнаковое целое, только в конце младшие разряды (поэтому получилось `0x48, 0x11`, а не `0x11, 0x48`), и все остальное до конца мы возвращаем как беззнаковое целое со старшими разрядами в конце.

Функция `unpack()` имеет следующий синтаксис:

```
array unpack(string format, string data)
```

Функция `unpack()` распаковывает данные из двоичной строки согласно формату и возвращает массив, содержащий распакованные элементы.

Следующая пара функций — `serialize()` и `unserialize()` — позволяет осуществлять упаковку и распаковку, соответственно, массивов и объектов.

Замечание

Сериализация впервые возникла в объектно-ориентированных библиотеках (первой из которых была MFC), потом сериализация стала появляться в объектно-ориентированных языках (Java). Идея сериализации заключается в том, что объекты и массивы очень сложны по своей структуре и на сохранение путем перебора каждого элемента требуется значительный объем кода — самым простым решением является сохранение таких структур в виде единой закодированной последовательности — байт-коде. В PHP функции сериализации упаковывают данные не в виде байт-кода, а в виде строки.

Синтаксис функции `serialize()` можно представить следующим образом:

```
string serialize(mixed value)
```

В качестве аргумента функция принимает массив или объект, возвращая его в виде закодированной строки. Симметричная ей функция `unserialize()` принимает в качестве аргумента закодированную строку, возвращая массив или объект.

```
mixed unserialize(string str)
```

Пример — в листинге 4.57.

Листинг 4.57. Упаковка данных в двоичный формат с помощью сериализации

```
<?php
  $poll[0] = 23;
  $poll[1] = 45;
  $poll[2] = 34;
  $poll[3] = 2;
  $poll[4] = 12;
  // Упаковываем массив в строку
  $str = serialize($poll);
  echo $str."<br>";
  // Извлекаем массив из строки
  $arr = unserialize($str);
  print_r($arr);
?>
```

Результат:

```
a:5:{i:0;i:23;i:1;i:45;i:2;i:34;i:3;i:2;i:4;i:12;}
Array
(
    [0] => 23
    [1] => 45
    [2] => 34
    [3] => 2
    [4] => 12
)
```

Последние две функции часто применяются при работе с регулярными и плоскими файлами, что будет продемонстрировано в соответствующих главах.

Подсветка кода PHP

Во многих Web-приложениях требуется подсветить PHP-код: такая задача часто возникает при разработке форумов, посвященных вопросам PHP-программирования, при динамическом отображении PHP-скриптов на страницах

ресурса, в отладочных Web-приложениях и т. д. Для этого предназначена функция `highlight_string()` — листинг 4.58.

Замечание

Помимо стандартной функции `highlight_string()`, обеспечивающей подсветку синтаксиса кода переданной в качестве параметра строки, существует также функция `highlight_file()`, обеспечивающая подсветку синтаксиса файла, имя которого передается в качестве аргумента.

Листинг 4.58. Подсветка кода

```
// Строка, содержащая скрипт PHP
$code = '<?php
        if(!$flag)
        {
          // пишем какой-либо код
          echo("Hello");
          $var = 1;
        }
        else break;
       ?>';
// вызываем функцию highlight_string
highlight_string($code);
?>
```

Как видно из листинга 4.58, строку, содержащую PHP-код, необходимо передать функции `highlight_string()`. Если в качестве второго необязательного параметра передать значение `true`, функция вместо вывода результата в окно браузера возвратит строку с подсвеченным синтаксисом.

Замечание

Стандартные функции `highlight_string()` и `highlight_file()` подсвечивают синтаксис кода только в том случае, если он заключен в теги `<?php` и `?>`. Это не всегда удобно, поэтому в *гл. 5* будет рассмотрено построение собственной функции подсветки синтаксиса.

Задания

4.1. В листинге 4.9 продемонстрировано, как при помощи обратных кавычек можно извлечь содержимое текущего каталога. Извлеките из полученной строки имена всех файлов и каталогов и разместите их в массивах `$files` и `$dirs` соответственно. Выведите содержимое этих массивов, предварительно отсортировав их.

4.2. Создайте массив из 10 элементов, значения которых равны факториалу индекса массива (0=>0!, 1=>1!, 2=>2!, ..., 9=>9!). Преобразуйте каждый элемент массива в строку из 20 символов таким образом, чтобы цифры были выровнены по левому краю, и выведите содержимое массива в окно браузера.

4.3. Создайте функцию, которая, принимая строку с HTML-страницей, возвращала бы ее название, заключенное между тегами <title> и </title>.

4.4. Разбейте строку со временем в формате СУБД MySQL "2004-11-26 14:56:08" на подстроки, содержащие год, месяц, число, часы, минуты и секунды с выводом их в окно браузера.

4.5. Реализуйте функцию, принимающую дату в формате "2004-11-26" и возвращающую в формате "26.11.2004".

4.6. Реализуйте скрипт замены тегов [code] и [/code] на <code> и </code>, соответственно, таким образом, чтобы замена происходила лишь в том случае, если каждому открывающему тегу соответствует закрывающий.

4.7. Преобразуйте скрипт в листинге 4.31 таким образом, чтобы значения параметров id_forum, id_theme и id_post в строке запроса попадали в переменные $forum, $theme и $post, соответственно.

4.8. Создайте аналог функции wordwrap(), не прибегая к этой функции.

4.9. В PHP имеется функция strrev(), принимающая в качестве параметра строку и возвращающая строку, в которой порядок следования символов изменен на обратный. Попробуйте реализовать такую функцию самостоятельно.

4.10. Запрос к поисковой системе Yandex имеет следующий вид **http://www.yandex.ru/yandsearch?stype=www&nl=0&text=%EF%F0...** В параметр запроса **text** помещается закодированная при помощи функции urlencode() поисковая фраза. Создайте HTML-форму с текстовым полем и кнопкой, обработчик которой загружал бы страницу с результатами поиска по фразе, введенной в поле формы. Поэкспериментируйте с другими поисковыми системами.

4.11. Выведите содержимое корневого каталога жесткого диска, на котором расположен скрипт. Обязательным условием является работоспособность скрипта во всех каталогах диска не зависимо от степени их вложенности.

4.12. Создайте скрипт, вычисляющий количество минут, прошедших с полуночи 1 января 2000 г. до текущего момента.

4.13. Реализуйте самостоятельно функции serialize() и unserialize() для упаковки и распаковки одномерного массива.

ГЛАВА 5

Регулярные выражения

Регулярные выражения — это специализированный язык поиска и осуществления манипуляций с подстроками в тексте.

Замечание

Синтаксис регулярных выражений является достаточно сложным и его изучение требует серьезных усилий. Наилучшим руководством по регулярным выражениям на сегодняшний день является книга Дж. Фридла "Регулярные выражения"[1], позволяющая, по словам автора, "научиться мыслить регулярными выражениями".

Базовый синтаксис и создание регулярных выражений

В настоящий момент существует несколько диалектов регулярных выражений. В данной главе будет рассмотрен синтаксис Perl-совместимых регулярных выражений, как самого распространенного и развитого диалекта.

Регулярное выражение — это шаблон, применяемый к заданному тексту слева направо. Большая часть символов сохраняет свое значение в шаблоне и означает совпадение с соответствующим символом. Так регулярное выражение, содержащее обычный текст, например "грам", соответствует строке, содержащей указанную подстроку, например "программирование".

В Perl для регулярного выражения обязательно задание границ. Так приведенный выше пример можно записать следующим образом:

```
"/грам/"
```

[1] Дж. Фридл. Регулярные выражения: Пер. с англ. 2-е изд. — СПб.: Питер, 2003. — 464 с.

Символ / применяется для задания границ регулярного выражения, т. е. регулярное выражение действует до тех пор, пока не встретится второй символ прямого слеша (/). После регулярного выражения следуют *модификаторы шаблона* — инструкции, действующие на все регулярное выражение. Мы рассмотрим только один модификатор i. При его использовании поиск по регулярному выражению осуществляется без учета регистра. Так рассмотренный выше пример можно записать следующим образом:

```
"/грам/i"
```

Данное регулярное выражение будет соответствовать как строке "программирование", так и "ПРОГРАММИРОВАНИЕ".

Замечание

С остальными модификаторами шаблонов можно ознакомиться в документации PHP, которая доступна для свободной загрузки с адреса **http://www.php.net**. Данный раздел документации переведен на русский язык.

Рассмотренное регулярное выражение осуществляет поиск по всему тексту, но чаще следует привязать регулярное выражение к началу слова, т. е. чтобы регулярное выражение "/грам/i" соответствовало строке, начинающейся со слова "грампластинка", но не подходило бы слову "программирование". Для этого используется символ ^, соответствующий началу строки:

```
"/^грам/i"
```

Знак доллара $ означает конец строки:

```
"/^грампластинка$/i"
```

Это регулярное выражение соответствует любой строке "грампластинка", но не подходит строке "грампластинка — это вам не программирование", т. к. после искомого слова идет текст.

Регулярное выражение

```
"/^$/i"
```

соответствует пустой строке.

В поисковой строке может понадобиться найти подстроку, содержащую символ /, который у нас используется для обозначения границ. В этом случае необходимо прибегнуть к экранированию этого символа с помощью символа обратного слеша (\). Так подстроки "Программирование/PHP" регулярное выражение выглядит следующим образом:

```
"/программирование\/php/i"
```

Если символ / встречается часто в поисковой строке (например, в HTML-тексте), то можно изменить границы регулярного выражения, в качестве них

может выступать любой другой символ, например, следующее регулярное выражение эквивалентно только что рассмотренному:

```
"|программирование/php|i"
```

Использование символа вертикальной черты (|) в качестве границы слова не всегда приемлемо, т. к. он может присутствовать в регулярном выражении для задания альтернативных масок:

```
"/abc|абв/"
```

Этому регулярному выражению соответствует любая строка, содержащая подстроки "abc" или "абв". Вертикальную черту удобно применять при проверке расширений и имен файлов, зон доменных имен и т. д. К примеру, следующее регулярное выражение проверяет, содержатся ли в строке подстроки "ru", "com" или "net":

```
"/ru|com|net/"
```

Это выражение соответствует любой строке, которая содержит подстроку "abc".

Подстроки в регулярных выражениях можно группировать при помощи скобок:

```
"/домен - (ru|com|net)/i"
```

Это регулярное выражение будет соответствовать строке вида "домен - ru", вместо ru может быть как com, так и net. Если необходимо использовать скобки как часть искомой строки, их следует экранировать. Так для поиска подстроки "программирование (PHP)" следует использовать следующее регулярное выражение:

```
"/ программирование \(PHP\)/i"
```

Скобки при программировании в PHP и Perl имеют еще одно значение помимо группировки символов. Все найденные в скобках выражения сохраняются интерпретатором, и к ним можно обратиться при замене или поиске по номеру скобки \\1, \\2 и т. д., но это будет рассмотрено в *разд. "Функции для работы с регулярными выражениями" далее в этой главе*.

Для задания класса символов используются квадратные скобки ([]). Они ограничивают поиск теми символами, которые в них заключены:

```
"/[abc]/"
```

Этому регулярному выражению соответствует подстрока, содержащая *один* символ: либо a, либо b, либо c.

Так для создания регулярного выражения, соответствующего всем буквам русского алфавита, можно, конечно, перечислить все буквы в регулярном вы-

ражении. Это допустимо, но утомительно и неэлегантно. Более коротко такое регулярное выражение можно записать следующим образом:

```
"/[а-Я]/"
```

Данное выражение подходит всем буквам русского алфавита, поскольку любые два символа, разделяемые дефисом, задают соответствие диапазону символов, находящихся между ними. Регулярное выражение `"/[а-Я]/"` описывает символы как нижнего, так и верхнего регистров, поэтому более подробно это выражение можно записать так:

```
"/[а-яА-Я]/"
```

Точно таким же образом задаются регулярные выражения, соответствующие цифре:

```
"/[0-9]/"
```

или

```
"/[0123456789]/"
```

Оба выражения эквивалентны.

В регулярных выражениях действуют также экранирование, применение обратного слеша с определенными символами, что приводит к их специальной интерпретации:

- `\d` — любая десятичная цифра;
- `\D` — любой символ, кроме десятичной цифры;
- `\s` — любой пробельный символ;
- `\S` — любой непробельный символ;
- `\w` — любой символ, образующий "слово";
- `\W` — любой символ, не образующий "слово";
- `\t` — символ табуляции;
- `\f` — конец файла;
- `\n` — символ перевода строки;
- `\\` — символ обратного слеша (\);
- `\.` — символ точки (".").

Замечание

Символ точки (".") используется для обозначения любого символа в регулярном выражении, поэтому для поиска точки следует экранировать этот символ.

Рассмотренное выше регулярное выражение для числа можно записать следующим образом:

```
"/[\d]/"
```

Для исключения класса символов из поиска первым ставится символ ^, который в квадратных скобках действует уже не как указатель границы строки, а как отрицание:

```
"[^0-9]"
```

Это регулярное выражение отвечает любому символу, не содержащемуся в диапазоне 0—9, т. е. все что угодно, только не цифра.

Замечание

Классы символов \d, \s и т. п. могут применяться в любой части регулярного выражения, а не только в квадратных скобках, т. е. вполне допустимо выражение вида "/\d/".

Выражение в квадратных скобках соответствует только одному символу и часто применяется совместно с так называемыми *квантификаторами*. Это символы ?, + и *, которые следуют сразу за символом и изменяют число вхождений конкретного символа в строку:

- ? — символ либо входит в строку один раз, либо вообще в нее не входит;
- * — любое число вхождений символа в строку, в том числе и 0;
- + — одно или более число вхождений символа в строку.

Символ ? позволяет сократить выражения вида

```
"/программирование|программирование\/PHP/i"
```

до

```
"/программирование (PHP)?/i"
```

Метасимвол + обозначает один или несколько экземпляров элемента непосредственно предшествующего элемента, а * — любое количество экземпляров элемента, в том числе и нулевое. Так, если необходимо найти подстроку, содержащую одну или более цифр, следует воспользоваться выражением вида

```
"/[\d]+/"
```

Символ * используется для любого числа вхождений строки в подстроку, т. е. регулярное выражение

```
"/^[\d]*$/"
```

соответствует либо пустой строке, либо строке, содержащей только цифры, причем их количество не ограничено.

Помимо круглых и квадратных скобок в регулярных выражениях так же применяются фигурные скобки ({}). Они предназначены для указания числа или диапазона чисел повторения элемента:

- "xy{2}" — соответствует строке "xyy";
- "xy{2,}" — соответствует строке, в которой за x следует не менее двух y (может быть и больше);
- "xy{2,6}" — соответствует строке, в которой за x следует от двух до шести y.

Для указания количества вхождений не одиночного символа, а их последовательности, используются круглые скобки:

- "x(yz){2,6}" — соответствует строке, в которой за x следует от двух до шести последовательностей yz;
- "x(yz)*" — соответствует строке, в которой за x следует ноль и более последовательностей yz.

Функции для работы с регулярными выражениями

Рассмотрим основные функции для работы с регулярными Perl-выражениями в PHP. В данном разделе в основном представлен только синтаксис, а конкретные примеры использования функций приведены в остальных разделах данной главы и других главах книги.

Первая рассматриваемая функция, `preg_match()` осуществляет поиск в строке по регулярному выражению и имеет следующий синтаксис:

```
int preg_match(string pattern, string subject
               [, array matches [, int flags [, int offset]]])
```

Эта функция ищет в строке *subject* соответствие регулярному выражению *pattern*. Если задан необязательный параметр *matches*, то результаты поиска помещаются в массив. Элемент *$matches*[0] будет содержать часть строки, соответствующую вхождению всего шаблона, *$matches*[1] — часть строки, соответствующей первым круглым скобкам, *$matches*[2] — вторым и т. д. Необязательный флаг *flag* может принимать единственное значение PREG_OFFSET_CAPTURE, при указании которого изменяется формат возвращаемого массива *$matches* — каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение. Поиск осуществляется слева направо, с начала строки. Дополни-

тельный параметр *offset* может быть использован для указания альтернативной начальной позиции для поиска. Функция `preg_match()` возвращает количество найденных соответствий. Это может быть 0 (совпадения не найдены) и 1, поскольку `preg_match()` прекращает свою работу после первого найденного совпадения.

Если необходимо найти либо сосчитать все совпадения, следует воспользоваться функцией `preg_match_all()`, которая имеет следующий синтаксис:

```
int preg_match_all(string pattern, string subject, array matches
                   [, int flags [, int offset]])
```

Функция ищет в строке *subject* все совпадения с регулярным выражением *pattern* и помещает результат в массив *matches* в порядке, определяемом комбинацией флагов *flags*. Так же как и в случае функции `preg_match()`, можно задать смещение *offset*, начиная с которого будет осуществляться поиск в строке *subject*.

После нахождения первого соответствия последующие поиски будут осуществляться не с начала строки, а от конца последнего найденного вхождения.

Дополнительный параметр *flags* может комбинировать следующие значения (использование `PREG_PATTERN_ORDER` одновременно с `PREG_SET_ORDER` бессмысленно):

- `PREG_PATTERN_ORDER` — если этот флаг установлен, результат будет упорядочен следующим образом: элемент `$matches[0]` содержит массив полных вхождений регулярных выражений, элемент `$matches[1]` хранит массив вхождений первых круглых скобок, `$matches[2]` — вторых и т. д. То есть если строка содержит три соответствия регулярному выражению, то подстроку для последнего соответствия всему регулярному выражению можно найти в элементе `$matches[0][3]`, а для первых круглых скобок данного соответствия — в элементе `$matches[1][3]`;
- `PREG_SET_ORDER` — если этот флаг установлен, результат будет упорядочен следующим образом: элемент `$matches[0]` содержит первый набор вхождений, элемент `$matches[1]` — второй набор вхождений и т. д. В таком случае массив `$matches[0]` содержит первый набор вхождений, а именно: элемент `$matches[0][0]` хранит первое вхождение всего регулярного выражения, элемент `$matches[0][1]` — первое вхождение первых круглых скобок, `$matches[0][1]` — вторых и т. д. Аналогично массив `$matches[1]` содержит второй набор вхождений, и так для каждого найденного набора;
- `PREG_OFFSET_CAPTURE` — в случае, если этот флаг указан, для каждой найденной подстроки будет указана ее позиция в исходной строке. Необхо-

димо помнить, что данный флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение.

Функция `preg_match()` возвращает количество найденных вхождений шаблона (может быть нулем) либо `false`, если во время выполнения возникли какие-либо ошибки.

Обобщением функций `preg_match()` и `preg_match_all()` является функция `preg_grep()`, которая имеет следующий синтаксис:

```
array preg_grep(string pattern, array input [, int flags])
```

Функция возвращает массив, состоящий из элементов массива *input*, которые соответствуют заданному регулярному выражению *pattern*. Параметр *flag* может принимать единственное значение `PREG_GREP_INVERT`, задание которого приводит к тому, что функция возвращает те элементы массива *input*, которые не соответствуют регулярному выражению *pattern*.

Функция `preg_replace()` осуществляет поиск и замену по регулярному выражению и имеет следующий синтаксис:

```
mixed preg_replace(mixed pattern, mixed replacement, mixed subject
                   [, int limit])
```

Эта функция ищет в строке *subject* соответствие регулярному выражению *pattern* и заменяет его на *replacement*. Необязательный параметр *limit* задает число соответствий, которые необходимо заменить. Если этот параметр не указан, или равен –1, то заменяются все найденные соответствия.

Параметр `replacement` может содержать ссылки вида `\\n`. Каждая такая ссылка будет заменена на подстроку, соответствующую `n` раз повторяющимся круглым скобкам. `n` может принимать значения от 0 до 99, причем ссылка `\\0` соответствует вхождению всего шаблона. Выражения в круглых скобках нумеруются слева направо, начиная с единицы.

Если во время выполнения функции были обнаружены совпадения с шаблоном, будет возвращено измененное значение *subject*, в противном случае будет возвращен исходный текст *subject*.

Первые три параметра функции `preg_replace()` могут быть одномерными массивами. В случае если массив использует ключи, при обработке массива они будут взяты в том порядке, в котором расположены в массиве.

Если параметры *pattern* и *replacement* являются массивами, `preg_replace()` поочередно извлекает из обоих массивов по паре элементов и использует их для операции поиска и замены. Если массив *replacement* содержит больше элементов, чем *pattern*, вместо недостающих элементов для замены будут

взяты пустые строки. В случае, если *pattern* является массивом, а *replacement* — строкой, по каждому элементу массива *pattern* будет осуществлен поиск и замена на *pattern* (шаблоном будут поочередно все элементы массива, в то время как строка замены остается фиксированной). Вариант, когда *pattern* является строкой, а *replacement* — массивом, не имеет смысла.

Функция `preg_replace_callback()` осуществляет поиск по регулярному выражению и замену с использованием функции обратного вызова:

```
mixed preg_replace_callback(mixed pattern, callback callback,
                            mixed subject [, int limit])
```

Поведение этой функции во многом сходно с `preg_replace()`, за исключением того, что вместо параметра *replacement* необходимо указывать функцию *callback*, которой в качестве входящего параметра передается массив найденных вхождений. Функция обратного вызова *callback* возвращает строку, в которой будет произведена замена. Пример использования данной функции будет приведен в *разд. "Подстановка с использованием собственных тегов форматирования" далее в этой главе*.

Последняя функция из группы Perl-совместимых регулярных выражений является функция `preg_split()`, которая разбивает строку по регулярному выражению:

```
array preg_split(string pattern, string subject
                 [, int limit [, int flags]])
```

Функция возвращает массив, состоящий из подстрок заданной строки *subject*, которая разбита по границам, соответствующим шаблону *pattern*.

В случае если параметр *limit* указан, функция возвращает не более, чем *limit* подстрок, при его отсутствии или равенстве –1 функция действует без ограничений. Последний параметр *flags* может быть произвольной комбинацией следующих флагов (соединение происходит при помощи оператора ИЛИ (|)):

- `PREG_SPLIT_NO_EMPTY` — если этот флаг указан, функция `preg_split()` вернет только непустые подстроки;
- `PREG_SPLIT_DELIM_CAPTURE` — если этот флаг указан, выражение, заключенное в круглые скобки в разделяющем шаблоне, также извлекается из заданной строки и возвращается функцией;
- `PREG_SPLIT_OFFSET_CAPTURE` — если этот флаг задан, для каждой найденной подстроки будет указана ее позиция в исходной строке. Этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом — смещение.

Конвертация даты из формата *YYYY-MM-DD* в *DD.MM.YYYY*

Продолжая рассмотренную в предыдущей главе тему конвертации даты, рассмотрим скрипт (листинг 5.1), осуществляющий преобразование даты из формата `YYYY-MM-DD` в формат `DD.MM.YYYY`. Эта задача решается при помощи функции `preg_match()`:

```
int preg_match(string pattern, string str [, array regs])
```

Данная функция ищет в строке *str* соответствие регулярному выражению, заданному в шаблоне *pattern*. Если соответствия подвыражений с шаблоном будут найдены, то они сохраняются в массиве соответствий *regs*. При этом *$regs*[0] содержит копию строки *string*, *$regs*[1] — подстроку, начинающуюся с первой сохраняющей скобки, $regs[2] — подстроку, начинающуюся со второй сохраняющей скобки и т. д.

Функция возвращает число найденных соответствий.

Листинг 5.1. Преобразование даты

```
<?php
$date = "2003-03-21";
if (preg_match("|([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})|i", $date, $regs))
{
  echo "$regs[3].$regs[2].$regs[1]";
}
else
{
  echo "Неверный формат даты: $date";
}
?>
```

Результатом работы скрипта будет строка

```
21.03.2003
```

Проверка правильности ввода e-mail

В следующем примере на предмет корректности проверяется адрес электронной почты, вводимый пользователем.

Будем исходить из того, что адрес должен иметь вид ***something@server.com***. У адреса имеются две составляющие — имя пользователя и имя домена, которые разделены знаком `@`. В имени пользователя могут присутствовать бук-

вы нижнего и верхнего регистров, цифры, знаки подчеркивания, минуса и точки. Для проверки разделителя между именем пользователя и именем домена в выражение требуется добавить @. Таким образом, регулярное выражение, проверяющее имя пользователя и наличие разделителя, имеет следующий вид:

```
"[0-9a-z_]+@[0-9a-z_^\.]+"
```

Для проверки последнего доменного имени (**.ru**, **.com**) необходимо добавить следующее выражение:

```
"\.[a-z]{2,3}"
```

Символ "." в регулярных выражениях используется для обозначения любого символа, поэтому для поиска соответствия точки в регулярном выражении этот символ экранируется: "\.".

Объединяя эти строки, можно получить следующее регулярное выражение в формате Perl для проверки адресов электронной почты:

```
"|[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}|i"
```

Проверка правильности ввода URL

Проверка правильности ввода URL является достаточно сложной задачей. Построить универсальное регулярное выражение, которое соответствовало бы всем URL, очень не просто.

Замечание

Данная задача решается в первой части книги Дж. Фридла "Регулярные выражения", который на протяжении нескольких страниц демонстрирует различные подводные камни при создании универсального регулярного выражения, постепенно подводя читателя к мысли, что необходимо искать "компромисс между сложностью и точностью" регулярного выражения.

При построении регулярного выражения будем исходить из того, что URL имеет следующий формат: **http://*хост*/*путь***.

Простейшим вариантом проверки, не позволяющим посетителю ошибиться при наборе URL, но в то же время не предотвращающий неверный формат, является:

```
"#http://[^ ]*\.html?#i"
```

Данное выражение учитывает только URL расширения файла html либо htm, для обобщения на другие файлы следует расширить регулярное выражение до

```
"#http://[^ ]*\.(html?|php|pl|cgi)#i"
```

Уточняя регулярное выражение, разместим за префиксом `http://` регулярное выражение для хоста — `[-a-z0-9_.]+`. Структура пути может быть более разнообразной. Для него необходимо использовать выражение вида: `[-a-z0-9_:@&?=+,.!/~*'%$]*`.

Объединяя все в одно регулярное выражение, получаем следующее:

```
"#http://[-a-z0-9_.]+[-a-z0-9_:@&?=+,.!/~*'%$]*\.(html?|php|pl|cgi)#i"
```

Проверка правильности ввода имени

Вводимая посетителем информация (имя и пароль) как с использованием метода `GET`, так и метода `POST` должна быть тщательно проверена на содержание инъекционных SQL-запросов, функции `system()` и т. п. *(см. гл. 9)*. Если имя пользователя не должно содержать ничего кроме букв, то для этого следует воспользоваться следующим регулярным выражением в формате Perl:

```
"|^[\w]+$|i"
```

которое соответствует строке, содержащей одну или более букв. Если кроме букв в имени или пароле следует разрешить использование чисел и пробельных символов, то регулярное выражение следует расширить следующим образом:

```
"|^[\w\d\s]+$|i"
```

Впрочем, использование символов табуляции, конца файла и т. п. является не лучшим решением, поэтому целесообразно символ `\s` заменить просто пробелом:

```
"|^[\w\d ]+$|i"
```

Если в имени используются инициалы, оно может включать завершающие точки, например "Симдянов И. В.". Использовать точки напрямую нельзя и их следует экранировать, т. к. символ точки является специальным символом, соответствующим любому одному символу:

```
"|^[\w\d\. ]+$|i"
```

В тексте, вводимом посетителем, могут быть любые символы, но символы прямых одинарных кавычек (') и точки с запятой (;) способны приводить к возможности осуществления зловредного кода. Запретить использование данных символов можно при помощи следующего регулярного выражения:

```
"|^[^';]+$|i"
```

Замечание

Конечно, символы одинарной кавычки и точки с запятой являются допустимыми и достаточно часто встречающимися символами, поэтому полное их запреще-

ние не является целесообразным, и чаще прибегают к их замене на другие символы.

Создадим HTML-форму (листинг 5.2).

Листинг 5.2. HTML-форма

```
<form action=handler.php method=post>
  Имя : <input type=text name=name><br>
  Пароль : <input type=password name=password><br>
  e-mail : <input type=text name=email><br>
  URL : <input type=text name=url><br>
  Сообщение : <textarea name=message cols=76 rows=3></textarea><br>
  <input type=submit name=send value=Отправить>
</form>
```

Форма в листинге содержит четыре текстовых поля: `name` — для имени посетителя, `password` — для его пароля, `email` — для адреса электронной почты и `url` — для ссылки на ресурс, текстовую область `message` для ввода сообщения и кнопку `send` для отправки данных обработчику handler.php, код которого представлен в листинге 5.3.

Листинг 5.3. Обработчик формы с проверкой вводимых данных

```
<?php
  // Извлекаем данные из суперглобального массива $_POST
  $name = $_POST['name'];
  $password = $_POST['password'];
  $email = $_POST['email'];
  $message = $_POST['message'];
  $url = $_POST['url'];
  if(!preg_match("|^[\w\d\. ]+$|i", $name))
    exit("Неверный формат имени");
  if(!preg_match("|^[\w\d\. ]+$|i", $password))
    exit("Неверный формат пароля");
  if(!preg_match("|[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}|i", $email))
    exit("Неверный формат e-mail");
  if(!preg_match("|^[^';]+$|i", $message))
    exit("Неверный формат e-mail");
  if(!preg_match("#http://[-a-z0-9_.]+[-a-z0-
9_:@&?=+,.!/~*'%$]*\.(html?|php|pl|cgi)#i", $url))
    exit("Неверный формат URL");
  // Дальнейшая обработка данных - проверка правильности пароля
  // и занесение сообщения в базу данных или файл
?>
```

Проверка правильности ввода числа

В HTML-формах Web-приложений часто требуется ввод чисел. Кроме того, практически все Web-приложения, имеющие в своей основе базу данных, передают первичные ключи через строку запроса. Например, URL

```
http://www.softtime.ru/forum/read.php?id_forum=1&id_theme=495
```

содержит два числовых параметра: `id_forum` и `id_theme`, которые принимают значения `1` и `495` соответственно. Дотошные посетители страницы могут попытаться присвоить этим параметрам совершенно другие значения, а то и подставить вместо цифр текст с SQL-инъекцией *(см. гл. 9)*. Поэтому все числовые параметры следует проверять — они должны содержать только цифры от 0 до 9 (листинг 5.4).

Листинг 5.4. Проверка числовых параметров в URL

```
<?php
  if(!preg_match("|^[\d]*$|", $_GET['id_theme'])) exit();
  if(!preg_match("|^[\d]*$|", $_GET['id_forum'])) exit();
?>
```

Приведенный в листинге 5.4 код допускает использование в качестве параметров `id_theme` и `id_forum` либо целые числа, либо пустое значение. Если пустое значение параметров недопустимо, следует заменить символ `*` (любое число символов) на `+` (хотя бы один символ).

Часто могут встречаться задачи проверки правильности ввода чисел определенного формата, например, почтового индекса или паспортных данных. В этом случае следует использовать фигурные скобки (листинг 5.5).

Листинг 5.5. Проверка правильности ввода чисел определенного формата

```
<?php
  $zip = "600603";                    // Почтовый индекс
  $passport_seria = "2202";           // Серия паспорта
  $passport_number = "856432";        // Номер паспорта
  if(!preg_match("|^[\d]{6}$|", $zip))
    exit("Введите почтовый индекс в формате ######");
  if(!preg_match("|^[\d]{4}$|", $passport_seria))
    exit("Введите серию паспорта в формате ####");
  if(!preg_match("|^[\d]{6}$|", $passport_number))
    exit("Введите номер паспорта в формате ######");
?>
```

Обработка целочисленных значений является самой простой и распространенной задачей. Более сложной проблемой является обработка чисел с плавающей точкой. В этом случае число может предваряться знаком плюс или минус, которые, впрочем, могут и отсутствовать. Это решается при помощи квантификатора ?, допускающего и отсутствие символа, и присутствие его в одном экземпляре:

```
"[-+]?"
```

Мантиссе числа соответствует рассмотренное в предыдущем разделе выражение

```
"[\d]+"
```

После точки (которую необходимо экранировать) следует дробная часть, которая описывается так же выражением

```
"[\d]+"
```

Теперь, объединяя фрагменты, можем получить шаблон для чисел с плавающей точкой:

```
"|^[-+]?[\d]+\.[\d]+$|"
```

Завершающим штрихом будет учет разного формата разделителя мантиссы и дробной части, в качестве которой может выступать как точка (`23.56`), так и запятая (`23,56`). Для учета чисел в обоих форматах, регулярное выражение следует исправить следующим образом:

```
"|^[-+]?[\d]+[\.,][\d]+$|"
```

Корректность ввода даты

Корректность ввода даты может быть проверена при помощи скрипта, представленного в листинге 5.6.

Листинг 5.6. Проверка корректности введенной даты

```
<?php
  $date = "2005.02.18";     // ГГГГ.ММ.ДД
  if(!preg_match("|^[0-9]{4}.[0-9]{2}.[0-9]{2}$|i", $date))
  {
    echo "Введите дату в формате ГГГГ.ММ.ДД";
  }
  else
  {
    echo "Дата введена правильно";
  }
?>
```

Только русский текст!

Одной из распространенных задач в русскоязычном Интернете является ограничение на ввод посетителем только кириллических букв. Это ограничение можно осуществить при помощи регулярного выражения

```
"|^[а-я]+$|i"
```

Данное регулярное выражение не допускает использование никаких разделителей, для их разрешения следует описать их явно:

```
"|^[-а-я\s\.,;:\?!]+$|i"
```

Это регулярное выражение помимо русского текста разрешает использовать пробельные символы, тире, точку, точку с запятой, двоеточие, знак вопроса и восклицания.

Автоподсветка URL

В гостевых книгах, форумах и чатах — везде, где посетители оставляют длинные сообщения с ворохом ссылок на другие ресурсы, встает задача автоматического преобразования URL в гиперссылку. То есть при размещении текста

```
"... на ресурсе, посвященном PHP-программированию, http://www.softtime.ru
вы можете найти..."
```

его следует преобразовать в текст вида

```
"... на ресурсе, посвященном PHP-программированию, <a
href=http://www.softtime.ru>http://www.softtime.ru</a> вы можете
найти..."
```

Эту задачу выполняет скрипт из листинга 5.7.

Листинг 5.7. Автоподсветка URL

```
<?php
  $text = "... на ресурсе, посвященном PHP-программированию,
http://www.softtime.ru вы можете найти...";
  $patern = "/(http:\/\/|)(\S+)/i";
  $replacment = '<a href="http://\\2">\\2</a>';
  echo preg_replace($patern, $replacment, $text);
?>
```

Результат работы скрипта выглядит следующим образом

```
... на ресурсе, посвященном PHP-программированию, www.softtime.ru
вы можете найти...
```

Конвертирование тегов в стиль форума phpBB и обратно

В Интернете получили большое распространение теги в квадратных скобках, именуемые также *тегами в стиле phpBB* (известного и широко распространенного форума). Удобство использования таких тегов заключается в том, что все теги HTML можно запретить, преобразуя их при помощи функции `htmlspecialchars()` в безопасную форму, и в то же время разрешить посетителям использовать их эквиваленты. Например, `[i]` вместо `<i>` и `[code]` вместо `<code>`. Теги в квадратных скобках можно заменить на теги в угловых скобках уже после преобразования текста при помощи функции `htmlspecialchars()`. Чаще всего прибегают к тегам `[url]`, которые имеют следующий синтаксис

```
[url = ссылка] имя_ссылки [/url]
```

При выводе на страницу этот шаблон следует преобразовать в

```
<a href=ссылка>имя_ссылки</a>
```

Для этого предназначен скрипт, представленный в листинге 5.8.

Листинг 5.8. Преобразование тегов ссылок, представленных в стиле phpBB

```
<?php
$text = "... на следующем [url = http://www.softtime.ru] ресурсе [/url]
        вы можете найти...";
$patern = "#\[[\s]*url[\s]*=[\s]*([^\]]*)\]([^\[]*)\[[\s]*/url[\s]*\]#i";
$replacment = '<a href=\\1>\\2</a>';
echo preg_replace($patern, $replacment, $text);
?>
```

Результат работы функции:

```
... на следующем ресурсе вы можете найти...
```

Квадратные скобки в искомой фразе необходимо экранировать. Кроме того, во все позиции, где могут встречаться пробелы, необходимо поместить подстроку `[\s]*`, соответствующую любому количеству пробельных символов, в том числе и нулевому. В Web-приложении может потребоваться решение и обратной задачи — преобразование тега `<a>` в тег с квадратными скобками (листинг 5.9).

Листинг 5.9. Преобразование тегов ссылок в теги стиля phpBB

```
<?php
  $str = "<a href=http://www.softtime.ru>Ресурс по PHP</a>";
```

```
  $pat="#<[\s]*a[\s]*href[\s]*=[\s]*([^>]*)>([^<]*)<[\s]*/[\s]*a[\s]*>#";
  $replacement = "[url = \\1]\\2[/url]";
  echo preg_replace($pat, $replacement, $str);
?>
```

Результат работы функции:

```
[url = http://www.softtime.ru]Ресурс по PHP[/url]
```

Сходной задачей является конвертация тегов с изображениями, которые имеют следующий синтаксис:

```
[img = ссылка] имя_ссылки [/img]
```

При выводе на страницу этот шаблон следует преобразовать в

```
<img src=ссылка>имя_ссылки</a>
```

Для этого предназначен скрипт из листинга 5.10.

Листинг 5.10. Преобразование тегов изображений, представленных в стиле phpBB

```
<?php
  $text = "... следующее изображение
           [img = http://www.softtime.ru/images/softtime3.gif]";
  $patern = "#\[[\s]*img[\s]*=[\s]*([^\]]*)\]([^\[]*)#i";
  $replacment = '<img src=\\1>';
  echo preg_replace($patern, $replacment, $text);
?>
```

Обратную задачу по преобразованию тегов <img> в теги с квадратными скобками решает скрипт, приведенный в листинге 5.11.

Листинг 5.11. Преобразование тегов <img> в теги с квадратными скобками

```
<?php
  $str = "<img src=http://www.softtime.ru/images/softtime3.gif>";
  $pattern = "/<img[\s]*src[\s]*=[\s]*([^>]*)>/i";
  $replacement = "[img = \\1]";
  echo preg_replace($pattern, $replacement, $str);
?>
```

Результат работы скрипта:

```
[img = http://www.softtime.ru/images/softtime3.gif]
```

Работа с HTML-тегами: извлечение параметров и текста

Часто в Web-приложениях встречается задача извлечения текста из HTML-страницы. В предыдущем разделе были рассмотрены приемы извлечения параметров из тегов <a> и <img>. Рассмотрим извлечение названия страницы, заключенное между тегами <title> и </title>. Эту задачу выполняет скрипт из листинга 5.12.

Листинг 5.12. Извлечение названия HTML-страницы

```
<?php
  $text = "<html><head><title>Название страницы</title></head>
           <body><p>Текст в абзаце</p>текст без абзаца</body></html>";
  $patern = "#<[\s]*title[\s]*>([^<]*)<[\s]*/title[\s]*>#i";
  if(preg_match($patern, $text, $matches)) echo $matches[1];
  else echo "Ничего не найдено";
?>
```

Результат работы скрипта:

```
Название страницы
```

Точно так же можно извлекать информацию из других тегов, например, текст между тегами и можно извлечь при помощи следующего скрипта (листинг 5.13).

**Листинг 5.13. Извлечение текста, расположенного между тегами и **

```
<?php
  $text = "<html><head><title>Название
страницы</title></head><body><p>Текст в абзаце</p>текст без
абзаца</body></html>";
  $patern = "#<[\s]*p[\s]*>([^<]*)<[\s]*/p[\s]*>#i";
  if(preg_match($patern, $text, $matches)) echo $matches[1];
  else echo "Ничего не найдено";
?>
```

Результат работы скрипта:

```
Текст в абзаце
```

Замена прямых кавычек на парные

Еще одна распространенная задача — преобразование прямых кавычек (") на парные кавычки (" и "). Эту задачу решает скрипт из листинга 5.14.

Листинг 5.14. Замена прямых кавычек на парные

```
<?php
  $string = '"Разный" "текст". Мда... кавычек может быть много. ""';
  $pattern = '|"([^"]*)"|i';
  $replacement = "“\\1”";
  echo preg_replace($pattern, $replacement, $string);
?>
```

Результат работы скрипта:

```
"Разный" "текст". Мда... кавычек может быть много. ""
```

Подстановка с использованием собственных тегов форматирования

Часто требуется использование своих собственных тегов, которые могут приобретать довольно сложную форму. Например, пусть в тексте присутствуют теги, имеющие следующий синтаксис:

```
[d]2D5+8[/d]
```

Данный тег применяется в ряде ролевых интернет-игр. Необходимо извлечь из тега все три числа и получить по ним результат, который вычисляется по следующему алгоритму:

```
$sum = 0;
for($i=0; $i<2; $i++) $sum = $sum + rand(1,5);//случайное число от 1 до 5
$sum = $sum + 8;
```

Данную задачу удобно решать при помощи функции `preg_replace_callback()`, как это продемонстрировано в листинге 5.15.

Листинг 5.15. Использование функции `preg_replace_callback()`

```
<?php
  // Текст для разбора
  $message = "текст текст текст
             [d] 2D5+8 [/d] текст текст [D]1d4+1 [/D]";
```

```
// Функция обратного вызова
function summ_number($matches)
{
  // Как обычно: $matches[0] - полное вхождение шаблона
  // $matches[1] - вхождение первой подмаски,
  // заключенной в круглые скобки, и т. д.
  for($i=1; $i<$matches[2]; $i++) $sum = rand(1,$matches[3])+1;
  $sum = $sum + $matches[4];
  $result = $matches[1]." = ".$sum;
  return $result;
}
echo preg_replace_callback(
          "|\[d][\s]*(([0-9]+)D([0-9]+)\+([0-9]+))[\s]*\[/d]|i",
          "summ_number",
          $message);
?>
```

Результат работы функции может быть следующим:

```
текст текст текст 2D5+8 = 14 текст текст 1d4+1 = 1
```

В листинге 5.15 используется функция обратного вызова `summ_number()`, имя которой передается в качестве второго аргумента функции `preg_replace_callback()`. Функция принимает массив с результатами поиска и осуществляет вычисления, возвращая строку с результатом, который подставляется вместо найденной подстроки.

Подсветка синтаксиса PHP: собственная функция

PHP имеет две стандартные функции для подсветки кода *(см. гл. 4)*: `highlight_string()` и `highlight_file()`. Данные функции имеют два серьезных недостатка: поддерживается только подсветка PHP-кода и только кода, размещенного между тегами `<?php` и `?>` (а так же `<?` и `?>`). Поэтому в большинстве случаев приходится применять собственную функцию подсветки синтаксиса.

Рассмотрим задачу построения простейших аналогов функций `highlight_string()` и `highlight_file()`, реализующих собственную схему подсветки кода и, следовательно, допускающих реализацию подсветки синтаксиса языков, отличных от PHP. Строковый вариант функции назовем `shighlight()`. Она будет принимать в качестве параметра строку, содержащую код PHP. Файловый вариант пусть называется `fhighlight()` и будет принимать имя файла с PHP-скриптом. Обе функции будут возвращать строки с разметкой HTML,

обеспечивающей подсветку синтаксиса. Функции должны, по возможности, подсвечивать максимальное количество элементов.

Построение функции подсветки следует решать с привлечением механизма регулярных выражений, специально созданных для такого рода задач. В листинге 5.16 приведен код функции `shighlight()`.

Листинг 5.16. Функция `shighlight()`

```
<?php
function shighlight($document)
{
  // Преобразуем угловые скобки для отображения HTML-тегов
  $document = str_replace("<", "&lt;", $document);
  $document = str_replace(">", "&gt;", $document);
  // Преобразуем теги PHP <?php и ? >
  $tegs = array("'&lt;\?php'si","'&lt;\?'si","'\?&gt;'si");
  $replace = array("<font color=#95001E>&lt;?php</font>",
                   "<font color=#95001E>&lt;?</font>",
                   "<font color=#95001E>?&gt;</font>");
  $document = preg_replace($tegs, $replace, $document);
  // Преобразуем комментарии
  $document = preg_replace("'((?:#|//)[^\n]*|/\*.*?\*/)'si",
                           "<font color=#244ECC>\\1</font>",
                           $document);
  // Осуществляем переносы строк
  $document = preg_replace("'(\n)'si","<br>\\1", $document);
  // Преобразуем функции
  $document = preg_replace ("'([\w]+)([\s]*)[\(]'si",
                            "<font color=#0000CC><b>\\1</b></font>\\2(",
                            $document);
  // Преобразуем операторы
  $separator = array("'\,'si",
                     "'\-'si",
                     "'\+'si",
                     "'\('si",
                     "'\)'si",
                     "'\{'si",
                     "'\}'si");
  $replace = array("<font color=#1A691A>,</font>",
                   "<font color=#1A691A>-</font>",
                   "<font color=#1A691A>+</font>",
                   "<font color=#1A691A>(</font>",
                   "<font color=#1A691A>)</font>",
```

```
                    "<font color=#1A691A>{</font>",
                    "<font color=#1A691A>}</font>");
  $document = preg_replace($separator,$replace,$document);
  // Преобразуем переменные PHP
  $document = preg_replace("'([\$]{1,2}[A-Za-z_]+)'si",
                           "<b><font color=#000000>\\1</font></b>",
                           $document);
  // Преобразуем строки, заключенные в одинарные и двойные кавычки
  $str = array("'(\"[^\"]*\")'si",
               "'(\'[^\']*\')'si");
  $replace = array("<font color=#FFCC00>\\1</font>",
                   "<font color=#FFCC00>\\1</font>");
  $document = preg_replace($str, $replace, $document);
  // Преобразуем зарезервированные слова
  $str = array("'(echo)'si",
               "'(print)'si",
               "'(while)'si",
               "'(for)'si",
               "'(if)'si",
               "'(else)'si",
               "'(switch)'si",
               "'(function)'si",
               "'(array)'si");
  $replace = array_fill(0,
                        count($str),
                        "<b><font color=#0000CC>\\1</font></b>");
  $document = preg_replace($str, $replace, $document);

  // Возвращаем результат работы функции
  return "<code>$document</code>";
}
?>
```

Работа функции начинается с преобразования угловых скобок < и > в их HTML-представление: `<` и `<` соответственно.

Следующим этапом является подсветка тегов <?php, <? и ?> темно-красным цветом. Так как на предыдущем этапе все угловые скобки были преобразованы, то регулярные выражения для этих трех тегов приобретают вид: `'<\?php'si`, `'<\?'si`, `'\?>'si`. Эта операция осуществляется при помощи функции `preg_replace()`, принимающей в качестве первых двух параметров массив регулярных выражений в стиле языка Perl и массив со строками замены. В качестве третьего параметра выступает строка, в которой осуществляется замена.

После преобразования тегов PHP происходит подсветка комментариев PHP светло-синим цветом. Регулярное выражение, ответственное за такое преобразование

```
'((?:#|//)[^\n]*|/\*.*?\*/)'si
```

можно разбить на три части. Подвыражение `/\*.*?\*/` ответственно за подсветку многострочных комментариев в стиле языка C (`/* */`) — символы `*` экранируются обратным слешем (`\`), а между последовательностями `/*` и `*/` допускается произвольное число любых символов (`.*?`). Подвыражение

```
((?:#|//)[^\n]*
```

является ответственным за однострочные комментарии: `//` (стиль C++) и `#` (стиль командных оболочек), после которых следует произвольное количество символов (включая их отсутствие), не содержащих символ перевода строки — `[^\n]*`. Последовательность `?:` предписывает не сохранять результат для данных круглых скобок. Это позволяет не задумываться о числе сохраненных результатов и при замене с использованием функции `preg_replace()`, а во втором параметре использовать единственный сохраненный результат `\\1`, который относится к внешним круглым скобкам.

Для осуществления переноса строки все символы переноса строки (`\n`) предваряются HTML-тегом переноса строки — `<br>`. Такой прием позволяет сохранить форматирование исходного текста PHP-программы и отразить его в окне браузера.

Следующим этапом является подсветка функций. В PHP существует огромное число функций. Кроме того, программисты в программах могут вводить собственные функции. Поэтому создание списка возможных функций, как это часто делают, нецелесообразно. Все функции объединяет то, что после их объявления обязательно следует список параметров в круглых скобках, который, в общем случае, может быть пустым. Между именем функции и скобкой может быть произвольное число пробельных символов или даже переводы строк, поэтому регулярное выражение принимает вид:

```
'([\w]+)([\s]*)[\(]'si
```

Имя функции состоит из одного или более символа, включающего обобщенный символ слов (`[\w]+`), после которого следует произвольное количество обобщенных пробелов (`[\s]*`) и экранированный символ открывающей круглой скобки (`[\(]`).

Замечание

Обобщенный символ слова `[\w]` эквивалентен `[a-Az-Z0-9_]`, а символ обобщенного пробела `[\s]` — `[ \f\n\r\t\v]`.

Во втором параметре функции `preg_replace()` используется результат, сохраненный в двух круглых скобках: `\\1` соответствует имя функции, `\\2` — пробелы между именем и открывающей круглой скобкой.

После этого происходит подсветка операторов темно-зеленым цветом.

Заключительный этап состоит в подсветке темно-синим цветом зарезервированных ключевых слов языка PHP, таких как `while`, `if`, `else` и т. д. Так как выделить эти ключевые слова на фоне других элементов программы достаточно сложно, они помещаются в массив и обрамляются круглыми скобками, что обеспечивает сохранение их в качестве первого параметра (`\\1`). Массив замены `$replace`, выступающий в качестве второго аргумента функции `preg_replace()`, автоматически формируется при помощи функции `array_fill()`. Для обеспечения подсветки других языков программирования в массив `$str` необходимо добавить зарезервированные в этих языках слова.

После построения строковой функции подсветки синтаксиса языка PHP не составляет труда создать файловую версию этой функции — `fhighlight()`, код которой приведен в листинге 5.17.

Листинг 5.17. Функция `fhighlight()`

```
<?php
function fhighlight($filename)
{
  // Открываем файл, имя которого передано в параметре $filename
  $file = fopen($filename, "r");
  // Помещаем его содержимое в буфер $document
  $document = fread($file, filesize($filename));
  // Закрываем файл
  fclose($file);
  // Вызываем функцию shighlight()
  return shighlight($document);
}
?>
```

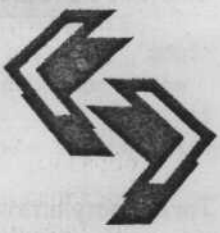
Работа функции сводится к открытию переданного в параметре `$filename` файла и передаче его содержимого ранее созданной функции `shighlight()`.

Задания

5.1. Сохраните на жестком диске любую HTML-страницу и удалите из нее все теги `<img>`. Для этого необходимо прочесть содержимое файла *(см. гл. 6)* и заменить все теги `<img>` пустой строкой, после чего результат следует вывести в окно браузера.

5.2. Возьмите любой объемный текст, содержащий несколько предложений, и поместите каждое предложение текста в элементы массива `$text` так, чтобы первое предложение оказалось в элементе с индексом 0 — `$text[0]`, второе в элементе с индексом 1 — `$text[1]` и т. д. Далее в цикле `foreach` преобразуйте массив `$text` в двумерный массив таким образом, чтобы в элементе `$text[0][0]` хранилось первое слово первого предложения, в элементе `$text[0][1]` хранилось второе слово первого предложения и т. д. Проконтролируйте результаты работы, отправив дамп массива в окно браузера при помощи функции `print_r()`.

5.3. Выберите объемный текст и посчитайте, сколько в нем содержится одно-, двух-, ..., десятибуквенных слов.



ГЛАВА 6

Работа с файлами и каталогами

Протокол HTTP не поддерживает сохранение состояния, поэтому файлы в Web-приложениях играют важную роль как средства хранения информации между сеансами. В данной главе будут рассмотрены приемы работы с файлами и каталогами, позволяющие использовать возможности файловой системы на 100%.

Включение файлов в документ

Начиная главу, посвященную файлам, следует остановиться на функциях `include()` и `require()`, позволяющих включить файл в PHP-скрипт. Обе функции принимают единственный аргумент — путь к включаемому файлу и результатом их действия является подстановка содержимого файла в место их вызова в исходном скрипте. Если в качестве включаемого скрипта выступает PHP-скрипт, то сначала происходит его подстановка в исходный скрипт, а затем интерпретация результирующего скрипта. Пример — в листинге 6.1.

Листинг 6.1. Использование функции `include()`

```
<?php
  echo "first<br>";
  include("second.php");
  echo "first<br>";
?>
```

Пусть файл second.php содержит код, представленный в листинге 6.2.

Листинг 6.2. Файл second.php

```
<?php
  echo "second<br>";
?>
<h3>Текст не обязательно должен выводиться оператором echo</h3><br>
```

Тогда результатом работы скрипта, приведенного в листинге 6.1, будут строки:

```
first
second
Текст не обязательно должен выводиться оператором echo
first
```

Можно явно указать возвращаемое значение, объявив во включаемом файле оператор `return`. Исправим содержимое файла second.php (листинг 6.3).

Листинг 6.3. Файл second.php

```
<?php
  $q = 12
  return $q*2;
  echo "second";
?>
```

Результат выполнения скрипта из листинга 6.1 будет таким:

```
first
24
first
```

Следует обратить внимание на то, что вызов оператора `return` действует точно так же, как в обычных функциях — программа немедленно выходит из файла, а все последующие за `return` операторы игнорируются.

Замечание

В качестве пути к файлу может быть указан сетевой путь. В этом случае следует помнить, что функция `include()` вернет файл в том виде, в котором она получает его от Web-сервера, т. е. не следует ждать PHP-код — в сетевом варианте функции `include()` будет получен лишь результат работы скрипта, но не его код.

Включаемый функцией `include()` файл не обязательно должен быть PHP-скриптом. К этой функции часто прибегают для включения объемных текстовых вставок, которые бы могли затруднить чтение кода.

Как было упомянуто в начале раздела, помимо функции `include()`, существует функция `require()`, выполняющая аналогичные действия. Различия в этих функциях заключаются в их реакции на отсутствие включаемого файла. Если в случае функции `include()` включаемый файл отсутствует, то реакцией на это является единственно вывод в окно браузера соответствующего предупреждения, который можно подавить, разместив перед `include()` символ `@`. Отсутствие файла по пути, который передается в качестве аргумента функции `require()`, приводит к остановке скрипта.

Замечание

Для обеих функций существуют аналоги с суффиксом `_once`: `include_once()` и `require_once()`, позволяющие включить файл в документ только один раз, не зависимо от того, сколько попыток включения предпринимается. Это удобно использовать при вложенных включениях, во избежание ошибок при повторном включении файлов, содержащих объявления функций.

Создание файлов и работа с ними

Любая операция с файлами, будь то создание нового файла, чтение или запись в уже существующий файл, предваряется операцией открытия файла. Исключение составляют ряд операций, которые обсуждаются далее. Открытие файлов в файловой системе сервера производится при помощи функции `fopen()`:

```
int fopen(string filename, string mode [, int use_include_path])
```

Первый аргумент *filename* — относительный или абсолютный путь к файлу. Второй аргумент *mode* позволяет задать режим работы с открываемым файлом и может принимать следующие значения:

- `r` — открыть файл только для чтения; после открытия указатель файла устанавливается в начало файла;
- `r+` — открыть файл для чтения и записи; после открытия указатель файла устанавливается в начало файла;
- `w` — создать новый пустой файл только для записи; если файл с таким именем уже есть, вся информация в нем уничтожается;
- `w+` — создать новый пустой файл для чтения и записи; если файл с таким именем уже есть, вся информация в нем уничтожается;
- `a` — открыть файл для дозаписи; данные будут записываться в конец файла;
- `a+` — открыть файл для дозаписи и чтения данных; данные будут записываться в конец файла;

- `b` — флаг, указывающий на работу — чтение и запись с двоичным файлом; задается только в Windows.

Третий необязательный аргумент *use_include_path* определяет, должны ли искаться файлы в каталоге *include_path*. (Параметр include_path устанавливается в файле php.ini.)

В случае удачного открытия файла функция `fopen()` возвращает дескриптор файла, в случае неудачи — `false`. *Дескриптор файла* представляет собой указатель на открытый файл, который используется операционной системой для поддержки операций с этим файлом и является уникальным числом.

Замечание

Не следует путать дескриптор файла и файловый указатель, который предназначен для указания положения точки ввода/вывода относительно начала открытого файла.

Возвращенный функцией дескриптор файла необходимо затем задавать во всех функциях, которые в дальнейшем будут работать с этим файлом.

Код в листинге 6.4 создает файл file.txt.

Листинг 6.4. Создание файла

```
<?php
  $fd = fopen("file.txt","w");
?>
```

После того как работа с файлом закончена, его необходимо закрыть. Закрытие файлов осуществляется с помощью функции `fclose()`:

```
int fclose(int fd)
```

Аргумент *fd* представляет собой дескриптор файла, который необходимо закрыть.

Замечание

Если скрипт не закрывает файл, то его закрытие осуществляется автоматически при завершении работы скрипта. Тем не менее, по возможности, следует закрывать файл сразу же после прекращения работы с ним, т. к. в то время, когда файл открыт, с ним не может работать другой скрипт, точно так же, как нельзя сменить режим работы с файлом в текущем скрипте.

Запись в файл осуществляется функциями `fputs()` и `fwrite()`, которые абсолютно идентичны друг другу как по синтаксису, так и по выполняемым действиям:

```
int fputs(int fd, string str [, int length]))
int fwrite(int fd, string str [, int length]))
```

Первый аргумент *fd* представляет собой дескриптор файла, который возвращается функцией `fopen()`. Второй аргумент *str* является строкой, которая должна быть записана в файл. Третий необязательный аргумент задает количество символов в строке, которые должны быть записаны. Если третий аргумент не указан, записывается вся строка.

В листинге 6.5 в файл file.txt записывается строка `"Hello, world!"`.

Листинг 6.5. Запись в файл

```
<?php
  // Открываем файл для чтения
  $fd = fopen ("file.txt","r");
  // Записываем файл
  fwrite($fd, "Hello world!");
  // Закрываем файл
  fclose ($fd);
?>
```

Чтение содержимого открытого файла можно осуществить при помощи функции `fread()`, которая имеет следующий синтаксис:

```
string fread(int fd, int length)
```

Эта функция возвращает строку длиной *length* байтов.

Для чтения файла целиком часто прибегают к функции `filesize()`, которая возвращает число байтов, содержащихся в файле, имя которого передано данной функции в качестве аргумента. Код в листинге 6.6 считывает содержимое файла и выводит его в окно браузера.

Листинг 6.6. Чтение из файла

```
<?php
  // Имя файла
  $filename = "file.txt";
  // Открываем файл для чтения
  $fd = fopen($filename, "r");
  // Читаем содержимое файла в переменную $bufer
  $bufer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Выводим содержимое файла в окно браузера
  echo $bufer;
}
?>
```

Для чтения из файла можно также пользоваться функцией `fgets()`:

```
string fgets(int file, int length)
```

Эта функция читает и возвращает строку длиной *length* – 1 байтов. Чтение прекращается, когда достигнута новая строка или конец файла. При достижении конца файла функция возвращает пустую строку.

Для чтения файла с удалением из него тегов HTML применяется функция `fgetss()`:

```
string fgetss(int file, int length [, string allowable_tags])
```

Необязательный третий параметр *allowable_tags* может содержать строку со списком тегов, которые не должны быть отброшены, при этом теги в строке записываются через запятую.

Если необходимо записать содержимое файла в массив, применяется функция `file()`:

```
array file(string filename [, int use_include_path])
```

Функция считывает файл с именем *filename* и возвращает массив, каждый элемент которого соответствует строке в прочитанном файле. В листинге 6.7 с помощью функции читается файл, информация из которого затем выводится в браузер.

Листинг 6.7. Использование функции `file()`

```
<?php
  $content = file("file.txt");
  foreach($content as line) echo "$line<br>";
?>
```

Эта функция удобна также тем, что с ее помощью можно легко подсчитать количество строк в файле (листинг 6.8).

Листинг 6.8. Подсчет количества строк в файле

```
<?php
  $content = file("file.txt");
  echo count($content);
?>
```

Для чтения файлов с расширением csv применяется функция `fgetcsv()`:

```
array fgetcsv(int file, int length, char delim)
```

Функция читает строку из файла и разбивает ее по символу *delim*. Строка *delim* должна состоять из одного символа, иначе принимается во внимание только первый символ этой строки. Функция возвращает получившийся массив или `false`, если достигнут конец файла. Пустые строки в файле не игнорируются, а возвращаются как массив из одного элемента — пустой строки. Параметр *length* задает максимальную длину строки точно так же, как это делается в функции `fgets()`.

Примечание

CSV-файл (Comma Separated Value) — это текстовый файл, в котором данные построены следующим образом: строки файла представляют собой строки таблицы, а столбцы таблицы разделены в файле заранее определенным символом-разделителем, например, `;`.

Кроме рассмотренных, в PHP имеется еще ряд функций, облегчающих ввод/вывод из файлов. При чтении данных из файла указатель текущей позиции перемещается к очередному непрочитанному символу. Существует несколько функций, с помощью которых можно управлять положением этого указателя.

Установка указателя текущей позиции в начало файла производится функцией `rewind()`:

```
int rewind(int fd)
```

Аргумент *fd* является дескриптором файла, который возвращает функция `fopen()`. Узнать текущее положение указателя можно при помощи функции `ftell()`:

```
int ftell(int fd)
```

Установить указатель в любое место файла можно, используя функцию `fseek()`:

```
int fseek(int fd, int offset [, int whence])
```

Функция `fseek()` устанавливает указатель файла на байт со смещением *offset* (от начала файла, от его конца или от текущей позиции, в зависимости от значения параметра *whence*). Аргумент *fd* представляет собой дескриптор файла. Аргумент *whence* задает, с какого места отсчитывается смещение *offset*, и может принимать одно из следующих значений:

- `SEEK_SET` — отсчитывает позицию от начала файла;
- `SEEK_CUR` — отсчитывает позицию относительно текущего положения указателя;
- `SEEK_END` — отсчитывает позицию относительно конца файла.

По умолчанию аргумент *whence* имеет значение `SEEK_SET`.

Узнать, находится ли указатель в конце файла, можно с помощью функции `feof()`:

```
int feof(int fd)
```

Если указатель находится в конце файла, функция возвращает `true`, в противном случае — `false`.

Функцию `feof()` удобно использовать при чтении файла, как это сделано в листинге 6.9.

Листинг 6.9. Чтение файла

```
<?php
  // Открываем файл для чтения
  $fd = fopen("file.txt","r");
  while(!feof($fd))
  {
    echo fgets($fd)."<br>";
  }
  // Закрываем файл
  fclose($fd);
?>
```

В завершение раздела следует упомянуть функцию `unlink()`, позволяющую уничтожить файл. Данная функция имеет следующий синтаксис:

```
bool unlink(string filename)
```

Функция принимает в качестве аргумента имя файла и возвращает `true` при успешном удалении и `false` — в противном случае.

Атрибуты файлов

Для получения дополнительной информации об атрибутах файла вы можете воспользоваться перечисленными ниже функциями.

- Функция `file_exists()` проверяет, существует ли файл, и возвращает `true`, если файл существует, и `false` — в противном случае:

  ```
  bool file_exists(string filename)
  ```

- Функция `fileatime()` возвращает время последнего обращения к файлу:

  ```
  int fileatime(string filename)
  ```

- Функция `filemtime()` возвращает время последней модификации содержимого файла:

  ```
  int filemtime(string filename)
  ```

- Функция `filesize()` возвращает размер файла в байтах:

```
int filesize(string filename)
```

- Функция `filetype()` возвращает тип файла:

```
string filetype(string filename)
```

Строка, возвращаемая этой функцией, содержит один из следующих типов файла:

- `char` — специальное символьное устройство;
- `dir` — каталог;
- `fifo` — именованный канал;
- `link` — символическая ссылка;
- `block` — специальное блочное устройство;
- `file` — обычный файл;
- `unknown` — тип не установлен.

Поскольку использование функций, возвращающих характеристики файла, весьма ресурсоемко, во избежание потери производительности при вызовах таких функций, PHP кэширует информацию о файле. Очистить этот кэш можно с помощью функции `clearstatcache()`:

```
<?php
  clearstatcache();
?>
```

Загрузка файлов на сервер

Еще одной распространенной задачей при работе с файлами является их загрузка на удаленный сервер. Для загрузки файлов на сервер понадобится HTML-форма index.html и скрипт upload.php для ее обработки. Простая форма для отправки файла на сервер может выглядеть так, как показано в листинге 6.10.

Листинг 6.10. HTML-форма для загрузки файлов на сервер — index.html

```
<html><head><title> Загрузка файлов на сервер </title></head><body>
  <h2><b> Форма для загрузки файлов </b></h2>
   <form action="upload.php" method="post" enctype="multipart/form-data">
     <input type="file" name="filename"><br>
     <input type="submit" value="Загрузить"><br>
```

```
    </form>
  </body>
</html>
```

Атрибут `entype` формы определяет вид кодировки, которую браузер применяет к параметрам формы. Для того чтобы отправка файлов на сервер действовала, атрибуту `entype` необходимо присвоить значение `"multipart/form-data"`. По умолчанию этот атрибут имеет значение `"application/x-www-form-urlencoded"`.

Если все сделано правильно, форма для отправки файлов на сервер должна выглядеть так, как это показано на рис. 6.1.

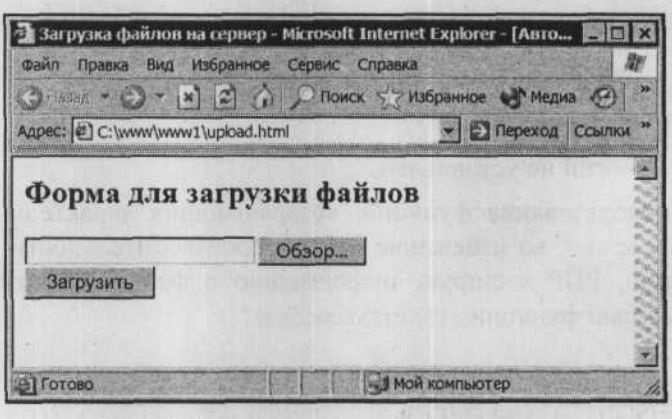


Рис. 6.1. Форма для загрузки файлов на сервер

После того как получен HTTP-запрос, содержимое загруженного файла записывается во временный файл, который создается в каталоге сервера, заданном по умолчанию для временных файлов, если другой каталог не задан в файле php.ini (директивой `upload_tmp_dir`).

Характеристики загруженного файла доступны через двумерный массив `$_FILES`. При этом переменная со значениями этого массива может иметь следующий вид:

- `$_FILES["filename"]["name"]` — содержит исходное имя файла на клиентской машине;
- `$_FILES["filename"]["size"]` — содержит размер загруженного файла в байтах;
- `$_FILES["filename"]["type"]` — содержит MIME-тип файла;
- `$_FILES["filename"]["tmp_name"]` — содержит имя временного файла, в который сохраняется загруженный файл.

В листинге 6.11 приведен скрипт upload.php, который загружает файл на сервер и копирует его из временного каталога в каталог temp.

Листинг 6.11. Загрузка файлов на сервер — upload.php

```
<html>
  <head>
    <title> Результат загрузки файла </title>
  </head>
  <body>
  <?php
    if(copy($_FILES["filename"]["tmp_name"],
       "temp/".$_FILES["filename"]["name"]))
    {
      echo("Файл успешно загружен");
    }
    else
    {
      echo("Ошибка загрузки файла");
    }
  ?>
  </body>
</html>
```

После выполнения этого скрипта выбранный для загрузки файл будет помещен в подкаталог temp каталога, в котором расположен скрипт, а браузер выдаст фразу "Файл успешно загружен".

Замечание

Проверить успешность загрузки файла на сервер можно при помощи специальной функции `is_uploaded_file()`, которая принимает в качестве единственного параметра имя файла (`$_FILES["filename"]["name"]`) и возвращает `true` в случае успешной загрузки и `false` в случае неудачи.

Скрипт из листинга 6.12 позволяет вывести характеристики загруженного файла. Для этого необходимо модифицировать скрипт в листинге 6.11.

Листинг 6.12. Вывод характеристик загруженного файла

```
<html>
  <head>
    <title> Результат загрузки файла </title>
  </head>
```

```
<body>
<?php
  if(copy($_FILES["filename"]["tmp_name"],
     "temp/".$_FILES["filename"]["name"]))
  {
    echo("Файл успешно загружен <br>");
    // Далее выводится информация о файле
    echo("Характеристики файла: <br>");
    echo("Имя файла: ");
    echo($_FILES["filename"]["name"]);
    echo("<br>Размер файла: ");
    echo($_FILES["filename"]["size"]);
    echo("<br>Каталог для загрузки: ");
    echo($_FILES["filename"]["tmp_name"]);
    echo("<br>Тип файла: ");
    echo($_FILES["filename"]["type"]);
  }
  else
  {
    echo("Ошибка загрузки файла");
  }
?>
</body>
</html>
```

Результат выглядит так, как представлено на рис. 6.2.

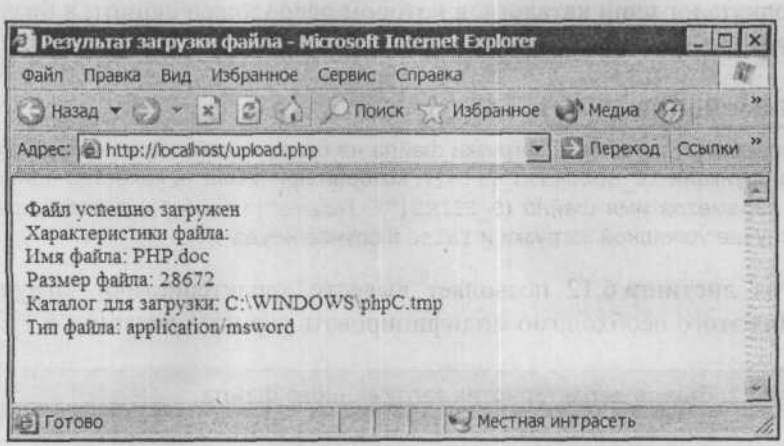


Рис. 6.2. Информация о загруженном файле

В некоторых случаях требуется ограничить размер файла, который может быть загружен на сервер. К примеру, чтобы разрешить загрузку на сервер

файлов размером не более 3 Мбайт, нужно изменить скрипт следующим образом (листинг 6.13).

Листинг 6.13. Ограничение объема загружаемых на сервер файлов

```
<?php
  if($_FILES["filename"]["size"] > 1024*3*1024)
  {
    exit("Размер файла превышает три мегабайта");
  }
  if(copy($_FILES["filename"]["tmp_name"],
    "temp/".$_FILES["filename"]["name"]))
  {
    echo("Файл успешно загружен <br>");
  }
  else
  {
    echo("Ошибка загрузки файла");
  }
?>
```

Максимальный размер загружаемого файла можно также задать при помощи директивы `upload_max_filesize`, значение которой по умолчанию равно 2 Мбайт:

```
if($_FILES["filename"]["size"] > upload_max_filesize)
```

Замечание

Значение директивы `upload_max_filesize` можно изменить в конфигурационном файле php.ini. Следует помнить, что изменения вступают в силу после перезагрузки Web-сервера.

Загрузка файлов с сервера

Обычно никаких специальных действий для загрузки файла с сервера предпринимать не нужно, достаточно указать ссылку на файл и переход по ней приведет к инициализации процесса загрузки файла.

Проблему составляют текстовые файлы, для которых не открывается окно, предлагающее сохранить файл, а осуществляется вывод в окно браузера, что не всегда удобно. Для того чтобы подавить вывод текстовых файлов в окно браузера, используются заголовки, приведенные в листинге 6.14.

Листинг 6.14. Подавление вывода текстовых файлов в окно браузера

```
<?php
  header("Content-Disposition: attachment; filename=text.txt");
  header("Content-type: application/octet-stream");
?>
```

Загрузка файла частями, или как разрезать и "склеить" файл?

Время действия PHP-скрипта на сервере составляет 30 секунд, объемные файлы при медленном соединении часто не успевают загрузиться за это короткое время. Кроме того, контролировать загрузку большого числа мелких файлов проще, чем одного большого. Поэтому часто прибегают к процедуре разбивки файла на отдельные части с последующим их объединением на сервере.

Пусть имеется файл site.rar, который необходимо разбить на части по 10 000 байт. Скрипт, выполняющий эту задачу, может выглядеть следующим образом (листинг 6.15).

Листинг 6.15. Разбивка файла на части

```
<?php
  // Имя файла
  $filename = "site.rar";
  // Разбиваем файл на части по 10 000 байт
  $piece = 10000;
  // Открываем исходный файл для чтения
  $fp = fopen($filename, "r");
  // Читаем содержимое файла в буфер
  $bufer = fread($fp, filesize($filename));
  // Закрываем файл
  fclose($fp);
  // Подсчитываем число частей, на которые необходимо разбить файл
  $count = (int)filesize($filename)/$piece;
  if((float)(filesize($filename)/$piece) - $count != 0) $count++;
  // В цикле разбиваем содержимое файла в переменной $bufer на части
  for($i=0; $i<$count; ++$i)
  {
    $part = substr($bufer,$i*$piece,$piece);
    // Сохраняем текущую часть в отдельном файле
    $fp = fopen("Site.tm".$i,"w");
```

```
    fwrite($fp,$part);
    fclose($fp);
  }
?>
```

Обратную задачу по сбору файла из отдельных частей выполняет скрипт, приведенный в листинге 6.16.

Листинг 6.16. Объединение частей файла в единое целое

```
<?php
  $buffer = "";
  for($i=0; $i<100000; ++$i)
  {
    // Генерируем имя файла
    $filename = "site.tm".$i;
    // Если такой файл существует, добавляем его содержимое к $buffer
    if(file_exists($filename))
    {
      $fp = fopen($filename,"r");
      $buffer .= fread($fp,filesize($filename));
      fclose($fp);
    }
    else
    {
      // Если файл с таким именем не существует, выходим из цикла
      break;
    }
    // Склеенные в переменной $bufer части помещаем в конечный файл
    $fp = fopen("site_final.rar","w");
    fwrite($fp, $buffer);
    fclose($fp);
  }
?>
```

Как посмотреть список файлов в каталоге?

PHP обладает внушительным списком функций для работы с каталогами. Для установки текущего каталога применяется функция `chdir()`:

```
int chdir(string path)
```

Работать с этой функцией можно следующим образом:

```
chdir("c:/tmp/data"); // переход по абсолютному пути
chdir("./js");        // переход в подкаталог текущего каталога
chdir("..");          // переход в родительский каталог
```

Чтобы узнать имя текущего каталога, можно воспользоваться функцией `getcwd()`:

```
string getcwd(string path)
```

По аналогии с файлами можно открыть каталог. Для этого предназначена функция `opendir()`, открывающая каталог, заданный параметром *path*:

```
int opendir(string path)
```

После того, как каталог открыт, прочитать его можно функцией `readdir()`:

```
string readdir(int dir)
```

Эта функция возвращает имена элементов, содержащихся в каталоге. Кроме файлов и папок в каталогах находятся также элементы "." и "..". Первый элемент указывает на текущий каталог, а второй — на родительский. Текущий каталог, кстати, можно открыть, указав его имя как ".":

```
$dir = opendir(".");
```

После того, как работа с каталогом закончена, его нужно закрыть. Закрытие каталога выполняется при помощи функции `closedir()`:

```
void closedir(dir)
```

В листинге 6.17 приведен скрипт, позволяющий вывести содержимое текущего каталога.

Листинг 6.17. Вывод списка файлов каталога

```
<?php
  // Открываем каталог
  $dir = opendir(".");
  // В цикле выводим его содержимое
  while (($file = readdir($dir)) !== false) echo "$file<br>";
  // Закрываем каталог
  closedir($dir);
?>
```

При просмотре каталога необходимо явно проверять успешность операции `$file = readdir($dir)`, т. к. если файл или подкаталог называется "0" или лю-

бым другим именем, начинающимся с нуля, то возвращаемое функцией `readdir()` имя может быть истолковано как `false`.

Вместо использования комбинации функций `opendir()` и `readdir()` имена файлов и каталогов можно также получить при помощи функции `scandir()`:

```
array scandir(string dir [, int sorting_order])
```

Эта функция возвращает массив, содержащий имена файлов и каталогов, расположенных в каталоге *dir*. Необязательный параметр *sorting_order* указывает, в каком порядке должна проводиться сортировка элементов массива. По умолчанию сортировка выполняется в алфавитном порядке по возрастанию. Если указан необязательный параметр *sorting_order*, равный 1, сортировка производится в алфавитном порядке по убыванию.

В листинге 6.18 показан простой пример, возвращающий массив с именами файлов и каталогов в заданном каталоге.

Листинг 6.18. Функция `scandir()`

```
<?php
  $dir = 'dir';
  // Сортировка по возрастанию
  $files = scandir($dir);
  // Сортировка по убыванию
  $sort_files = scandir($dir, 1);
  // Выводим содержимое массивов
  print_r($files);
  print_r(sort_files);
?>
```

Результат выглядит следующим образом:

```
Array
(
    [0] => array.php
    [1] => file.txt
    [2] => somedir
)
Array
(
    [0] => somedir
    [1] => file.txt
    [2] => array.php
)
```

Как определить: перед нами каталог или файл, или подсчет файлов в каталоге

Функции `readdir()` и `scandir()` не делают различий между содержимым каталога, возвращая все элементы каталога, будь то файлы или подкаталоги. В первую очередь может стоять задача в исключении каталогов "." и ".." из списка содержимого каталога. Этого можно добиться при помощи скрипта из листинга 6.19.

Листинг 6.19. Исключение каталогов "." и ".." из списка содержимого каталога

```
<?php
  // Открываем каталог
  $dir = opendir(".");
  // В цикле выводим его содержимое
  while (($file = readdir($dir)) !== false)
  {
    if($file != "." && $file != "..") echo "$file<br>";
  }
  // Закрываем каталог
  closedir($dir);
?>
```

При создании Web-приложений часто встает задача определения количества файлов или каталогов в подкаталоге, что требует различать файлы и подкаталоги. Для этого предназначены функции `is_file()` и `is_dir()`, которые имеют следующий синтаксис:

```
bool is_file(string filename)
bool is_dir(string filename)
```

Обе функции принимают в качестве аргумента имя файла или каталога. Функция `is_file()` возвращает `true` в том случае, если по переданному ей пути в качестве аргумента находится файл, и `false` — в противном случае. Аналогичным образом ведет себя и функция `is_dir()`, только по отношению к каталогу.

Замечание

В PHP имеется функция `file_exists()`, которая проверяет наличие по пути, переданного ей в качестве аргумента файла или каталога. Если такой файл или каталог существует, функция возвращает `true`, если нет — `false`.

С учетом рассмотренных функций задачу подсчета файлов и подкаталогов в каталоге можно решить так, как показано в листинге 6.20.

Листинг 6.20. Подсчет количества файлов и подкаталогов в каталоге

```
<?php
  $file_count = 0;      // Счетчик файлов
  $dir_count = 0;       // Счетчик подкаталогов
  $arr = scandir("."); // Возвращаем массив с содержимым каталога
  foreach($arr as $file)
  {
    // Если это текущая или вышележащая папка, пропускаем обработку
    // и переходим к следующему циклу
    if($file == "." || $file == "..") continue;
    // Если это файл, увеличиваем счетчик файлов
    if(is_dir($file)) ++$dir_count;
    // Если это каталог, увеличиваем счетчик каталогов
    if(is_file($file)) ++$file_count;
  }
  // Выводим результаты
  echo "В текущем каталоге содержится $dir_count подкаталогов,<br>";
  echo "а также $file_count файлов";
?>
```

Работа с правами доступа

Так сложилось, что большинство Web-разработчиков при создании Web-приложений используют в качестве операционной системы Windows, в то время как большинство серверов работает под управлением UNIX-подобной операционной системы, которая имеет отличную от Windows систему прав доступа.

В UNIX права доступа выставляются для трех групп:

- владельца файла;
- группы, в которую входит владелец файла;
- всех остальных.

Для каждой из групп права доступа задаются восьмеричным числом. При этом праву чтения соответствует цифра 4, праву записи — 2, а исполнению — 1. Общие права для групп задаются суммой этих чисел, так значение 6 (4 + 2) обеспечивает возможность чтения и записи, а значение 7 (4 + 2 + 1) предоставляет полный доступ к файлу или каталогу.

Для файлов наиболее приемлемые права доступа таковы: чтение и запись для владельца и чтение для всех остальных — 644. Для того чтобы иметь возможность "заходить" в каталог, для него необходимо выставить права доступа и на исполнение, поэтому для каталогов следует выставлять 755.

Для смены прав доступа в PHP предназначена функция `chmod()`, которая имеет следующий синтаксис:

```
bool chmod(string filename, int mode)
```

Функция принимает в качестве первого параметра имя файла *filename*, в качестве второго *mode* — восьмеричное число, задающее права доступа к файлу.

Замечание

Восьмеричное число в PHP предваряется 0, например, 0755. Число 755 не является восьмеричным и будет интерпретироваться как десятичное.

Пример — в листинге 6.21.

Листинг 6.21. Изменение прав доступа к файлу

```
<?php
  // Изменяем права доступа к файлу. Для указания принадлежности числа
  // к восьмеричной системе перед ним записывается 0
  chmod("index.php", 0644);
?>
```

Если требуется изменить права доступа сразу ко всем файлам каталога, следует воспользоваться скриптом из листинга 6.22.

Листинг 6.22. Изменение прав доступа ко всем файлам каталога

```
<?php
  // Открываем каталог
  $dir = opendir("dir_name");
  // В цикле считываем его содержимое
  while(($file = readdir($dir)))
  {
    // Если текущий объект является файлом, изменяем права доступа
    if(is_file($file)) chmod($file, 0644);
  }
  // Закрываем каталог
  closedir($dir);
?>
```

Создание каталога

Создание каталога производится с помощью функции `mkdir()`:

```
bool mkdir(string dirname, int mode)
```

Эта функция создает каталог с именем *dirname* и правами доступа, определяемыми восьмеричным числом *mode*. В случае неудачи возвращает `false`. В листинге 6.23 приведен пример создания каталога `test`.

Листинг 6.23. Создание каталога

```
<?php
  if(mkdir("test", 0700)) echo "Каталог успешно создан";
  else echo "Ошибка создания каталога";
?>
```

Удаление каталогов

Удалить каталог можно с помощью функции `rmdir()`, которая имеет следующий синтаксис:

```
bool rmdir(string dirname)
```

Функция принимает имя каталога *dirname* и возвращает `true` в случае удачного удаления каталога и `false` при неудаче.

Пример — в листинге 6.24.

Листинг 6.24. Удаление каталога

```
<?php
  if(rmdir("c:/temp/test")) echo("Каталог успешно удален");
  else echo("Ошибка удаления каталога");
?>
```

Функция `rmdir()` удаляет только пустые каталоги. Для уничтожения непустого каталога необходимо предварительно удалить все файлы, содержащиеся в нем, как это показано в листинге 6.25.

Листинг 6.25. Удаление каталогов

```
<?php
  // Рекурсивная функция удаления каталога
  // с произвольной степенью вложенности
  function full_del_dir($directory)
  {
    $dir = opendir($directory);
    while(($file = readdir($dir)))
```

```
    {
      // Если функция readdir() вернула файл, удаляем его
      if(is_file("$directory/$file")) unlink("$directory/$file");
      // Если функция readdir() вернула каталог и он
      // не равен текущему или родительскому, осуществляем
      // рекурсивный вызов full_del_dir() для этого каталога
      else if (is_dir("$directory/$file") &&
               $file != "." &&
               $file != "..")
      {
        full_del_dir("$directory/$file");
      }
    }
    closedir($dir);
    rmdir($directory);
    echo("Каталог успешно удален");
  }
  full_del_dir("temp");
?>
```

При рекурсивном вызове функции не следует передавать в качестве аргументов записи "." и "..".

Редактирование файлов на сервере

Используя функции для работы с файлами, можно организовать редактирование файлов на сервере при помощи Web-интерфейса, код которого представлен в листинге 6.26.

Листинг 6.26. Редактирование файлов на сервере (edit.php)

```
<?php
  // Файл edit.php
  // Если передано исправленное содержимое файла,
  // открываем файл и перезаписываем его
  if(isset($_POST['content']))
  {
    // Открываем файл
    $fd = @fopen($_POST['filename'], "w");
    // Если файл не может быть открыт, сообщаем
    // об этом предупреждением в окне браузера
    if(!$fd) exit("Такой файл отсутствует");
```

```
    // Перезаписываем содержимое файла
    fwrite($fd, stripslashes($_POST['content']));
    // Закрываем файл
    fclose($fd);
    // Помещаем в суперглобальный массив $_GET имя файла
    $_GET['filename'] = $_POST['filename'];
  }
?>
<form action = "edit.php" name=first method="get">
   Имя файла <input type="text" name="filename"
                   value=<?php echo $_GET['filename']; ?>><br>
  <input type="submit" value="Отправить">
</form>
<?php
  // Если в строке запроса передано имя файла,
  // открываем его для редактирования
  if(isset($_GET['filename']))
  {
    // Открываем файл
    $fd = @fopen($_GET['filename'], "r");
    // Если файл не может быть открыт, сообщаем
    // об этом предупреждением в окне браузера
    if(!$fd) exit("Такой файл отсутствует");
    // Помещаем содержимое файла в переменную $bufer
    $bufer = fread($fd, filesize($_GET['filename']));
    // Закрываем файл
    fclose($fd);
    ?>
      <form action = "edit.php" name=second method="post">
        <textarea cols=76 rows=10 name="content">
                  <?php echo $bufer; ?></textarea><br>
        <input type="hidden" name=filename
               value='<?php echo $_GET['filename']; ?>'>
        <input type="submit" name=edit value="Редактировать">
      </form>
    <?php
  }
?>
```

При первой загрузке скрипта открывается форма `first`, имеющая единственное текстовое поле `filename` для имени файла и кнопку, отправляющую данные методом `GET` файлу edit.php, в котором расположен этот скрипт (рис. 6.3). Если элемент суперглобального массива `$_GET['filename']` не пуст, то проис-

ходит загрузка содержимого файла во временную переменную `$bufer`, содержимое которой помещается в текстовую область формы `second` (рис. 6.4). Помимо текстовой области `content`, форма содержит скрытое поле `filename`, через которое передается имя файла. После редактирования и нажатия кнопки второй формы `second` данные повторно отправляются скрипту edit.php, но уже методом `POST`. В начале скрипта размещен обработчик, который в том случае, если элемент `$_POST['content']` принимает не пустое значение, переписывает содержимое редактируемого файла. Для корректного поведения

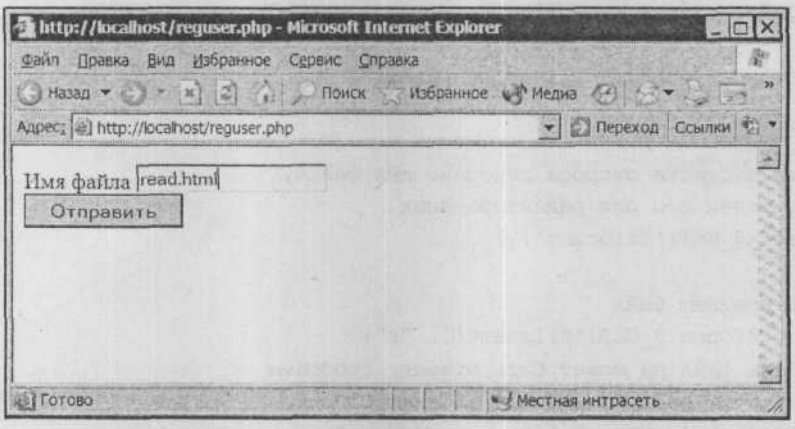


Рис. 6.3. Первая форма Web-интерфейса для редактирования файла

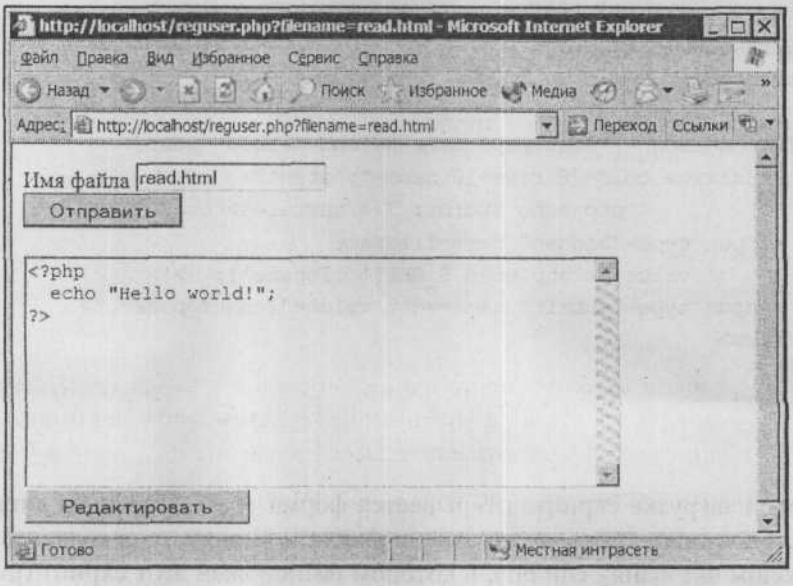


Рис. 6.4. Вторая форма редактирования Web-интерфейса

скрипта в конце обработчика имя файла из суперглобального массива $_POST переписывается в суперглобальный массив $_GET. Функция fopen() предваряется знаком @, который подавляет вывод предупреждений в окно браузера, в том случае если функция не может найти текстовый файл.

Автоматическое редактирование текстовых файлов

Часто при создании Web-приложений, основанных на хранении информации в файлах, требуется изменение их содержимого. Решение такого рода задач осуществляется по единой схеме, представленной в листинге 6.27.

Листинг 6.27. Редактирование файла

```
<?php
  // Имя файла
  $filename = "text.txt";
  // Открываем файл
  $fd = fopen($filename, "r");
  // Читаем его содержимое в буфер
  $buffer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Редактируем содержимое переменной
  // $buffer
  //....
  // Записываем в файл новое содержимое
  $fd = fopen($filename, "w");
  // Записываем содержимое строки $buffer
  fwrite($fd, $buffer);
  // Закрываем файл
  fclose($fd);
?>
```

Файл, над которым необходимо совершить преобразования, открывается, и его содержимое подвергается редактированию. Затем файл повторно открывается и перезаписывается уже измененным содержимым.

Пусть имеется текстовый файл proba.txt следующего содержания:

```
107 Разное
108 Экскурсии и туризм
109 Пищевые продукты
```

```
3202 9-этажный жилой дом (Автор: Молчанова Т.А.)
16734 The Tower of London (Автор: Репка Николай)
3206 Автобусная остановка
```

Задача состоит в удалении цифр, в начале каждой строки файла, и выводе результата в окно браузера. Для решения данной задачи удобно использовать регулярные выражения (листинг 6.28).

Листинг 6.28. Редактирование файла с использованием регулярных выражений

```
<?php
  // Имя файла
  $filename = "proba.txt";
  // Помещаем содержимое файла в массив $line
  $lines = file($filename);
  // В цикле извлекаем из строки текстовую часть строки
  foreach($lines as $line)
  {
    echo preg_replace("/\d{1,}\s([^\(]*)\s\(?.*/","\\1",$line);
    echo "<br>";
  }
?>
```

Результатом работы скрипта будет следующий столбец:

```
Разное
Экскурсии и туризм
Пищевые продукты
9-этажный жилой дом
The Tower of London
Автобусная остановка
```

Рассмотрим следующую очень распространенную задачу, связанную в первую очередь с созданием различных файловых счетчиков посещений и учета обращений к ресурсам. Пусть имеется файл count.txt следующего формата:

```
first = 34
second = 56
third = 14
fourth = 48
```

В задачу создаваемого нами скрипта будет входить увеличение числа при передаче скрипту в качестве параметра `page` левого значения из текстового

файла count.txt. Эту проблему решает скрипт, представленный в листинге 6.29, в котором используется функция обратного вызова.

Листинг 6.29. Создание счетчика

```
<?php
  // Извлекаем из строки запроса параметр page
  $page = $_GET['page'];
  // Имя файла
  $filename = "count.txt";
  // Открываем файл для чтения
  $fd = fopen($filename, "r");
  // Читаем его содержимое в буфер
  $bufer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Редактируем содержимое переменной
  $bufer = preg_replace_callback("|($page =)([\d]+)|i",
                                 "increment_number",$bufer);
  // Открываем файл для записи
  $fd = fopen($filename, "w");
  // Записываем содержимое строки $buffer
  fwrite($fd, $bufer);
  // Закрываем файл
  fclose($fd);
  // функция обратного вызова
  function increment_number($matches)
  {
    return $matches[1].(++$matches[2]);
  }
?>
```

Удаление строк из середины файла

Удаление данных из середины файла является достаточно распространенной задачей. Будем использовать рассмотренный в предыдущем разделе файл count.txt, но теперь станем не увеличивать значение счетчика на единицу при обращении к странице, а уменьшать все числовые значения, причем, если число принимает значение 0, то его будем удалять из файла. Решение этой задачи представлено в листинге 6.30.

Листинг 6.30. Удаление строк из середины файла

```
<?php
  // Имя файла
  $filename = "count.txt";
  // Получаем содержимое файла в виде массива строк
  $lines = file($filename);
  // В эту строку будем складывать новые строки файла
  $data = "";
  // Обработку данных производим в цикле
  foreach($lines as $index => $line)
  {
    $patern = "|^([^ ]+) = ([^\n]+)|i";
    preg_match ($patern, $line, $out);
    // Если число равно 0, то пропускаем цикл,
    // исключая тем самым строку из конечной строки $data
    $out[2]--;
    if($out[2] == 0) continue;
    $data .= "$out[1] = $out[2]\n";
    echo "$out[1] = $out[2]<br>";
  }
  // Записываем новые данные в файл
  $fd = fopen($filename, "w");
  fwrite($fd, $data);
  fclose($fd);
?>
```

Случайный вывод из файла

При создании блока "Афоризм дня" или ленты "Анекдоты" часто встает задача выборки из текстового файла случайных строк. Для решения данной задачи необходимо использование функции `rand()`, которая имеет следующий синтаксис:

```
int rand( [int min, int max])
```

Функция генерирует случайное целое число между двумя целочисленными параметрами *min* и *max*. Если необязательные параметры *min* и *max* не указаны, число выбирается из диапазона [0; `RAND_MAX(32768)`].

Замечание

Традиционно генератор случайных чисел инициировался временной отметкой вручную, в противном случае генератор всегда формировал одну и ту же последовательность случайных цифр. В PHP эту задачу выполняла функция

srand(). Начиная с версии PHP 4.2, было принято решение делать это автоматически, и теперь в инициализации генератора нет необходимости, хотя это не возбраняется и оставлено для обратной совместимости со старым кодом.

Пусть имеется текстовый файл, в каждой строке которого содержатся данные для вывода в окно браузера. Будем для этого использовать файл count.txt, рассмотренный выше. Тогда задачу случайного вывода можно решить при помощи скрипта, представленного в листинге 6.31.

Листинг 6.31. Случайный вывод данных из текстового файла

```
<?php
  // Имя файла
  $filename = "count.txt";
  // Помещаем содержимое файла count.txt в массив $lines
  $lines = file($filename);
  // Генерируем случайный индекс массива $lines
  $index = rand(0, count($lines) - 1);
  // Выводим строку с номером $index
  echo $lines[$index];
?>
```

Результат работы скрипта может быть следующим:

```
third = 14
```

Работа с индексным файлом: запись, извлечение, редактирование и удаление

Еще одной распространенной задачей является работа с *индексными файлами*. Пусть имеется файл index.txt следующего формата:

```
1 Программирование
2 Программирование на PHP
3 Программирование на JavaScript
4 Программирование на ASP.NET
```

Каждому номеру, который обычно называется *индексом*, соответствует строка, называемая *значением*. Одной из насущных задач является определение наличия в файле заданного индекса и вывод соответствующего ему значения, если такой индекс существует. Данную проблему решает скрипт из листинга 6.32.

Листинг 6.32. Поиск в файле заданного индекса

```
<?php
  // Ищем строку с индексом 2
  $index = 2;
  // Имя файла
  $filename = "index.txt";
  // Открываем файл для чтения
  $fd = fopen($filename, "r");
  // Читаем содержимое файла
  $bufer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Находим строку с индексом $index
  preg_match("|$index ([^\n]*)|", $bufer, $matches);
  // Выводим результат
  if(isset($matches[1])) echo $matches[1];
  else echo "Позиция не найдена";
?>
```

Результат работы скрипта выглядит следующим образом:

```
Программирование на PHP
```

Для вставки уникального индекса в файл необходимо определить максимальный индекс и увеличить его на единицу. Эта задача решается в листинге 6.33.

Листинг 6.33. Вставка в файл уникального индекса

```
<?php
  // Имя файла
  $filename = "index.txt";
  // Помещаем содержимое файла в массив $lines
  $lines = file($filename);
  // В цикле формируем массив индексов
  foreach($lines as $line)
  {
    // Обнуляем массив
    unset($matches);
    // Находим строку с индексом $index
    preg_match("|^[\d]+|", $line, $matches);
    if(isset($matches)) $index[] = $matches[0];
  }
```

```
  // Находим максимальное значение массива
  // $index и увеличиваем его на единицу
  echo max($index) + 1;
?>
```

Результат работы скрипта выглядит следующим образом:

```
5
```

Замечание

При таком подходе при увеличении объема файла время на поиск максимального значения будет расти, поэтому целесообразно хранить максимальный индекс элемента в отдельном файле, переписывая его при каждом добавлении уникального индекса в файл.

Теперь, когда новый уникальный индекс получен, можно добавить новую запись в файл (листинг 6.34).

Листинг 6.34. Добавление записи в файл

```
<?php
  // Имя файла
  $filename = "index.txt";
  // Открываем файл для дозаписи
  $fd = fopen($filename, "a");
  // Добавляем новую запись
  fwrite($fd, "5 Новая строка\n");
  // Закрываем файл
  fclose($fd);
?>
```

В конце строки следует поместить символ перевода строки \n, иначе новая запись окажется не на новой строке, а в конце последней записи. После работы функции будет добавлена новая строка, в результате чего файл index.txt примет вид:

```
1 Программирование
2 Программирование на PHP
3 Программирование на JavaScript
4 Программирование на ASP.NET
5 Новая строка
```

Еще одной интересной задачей является редактирование отдельных записей файла. Эту задачу решает небольшое Web-приложение, расположенное в файле edit.php, код которого представлен в листинге 6.35.

Листинг 6.35. Файл edit.php

```
<?php
  // edit.php
  // Имя файла
  $filename = "index.txt";
  // Если передано исправленное содержимое файла,
  // открываем файл и перезаписываем его
  if(isset($_POST['content']))
  {
    // Открываем файл для чтения
    $fd = @fopen($filename, "r");
    // Читаем содержимое буфера в переменную $bufer
    $bufer = fread($fd, filesize($filename));
    // Закрываем файл
    fclose($fd);
    // Изменяем строку с индексом $_POST['index']
    $patern = "|".$_POST['index']." [^\n]*\n|";
    $replacement = $_POST['content'];
    $bufer = preg_replace($patern, $replacement, $bufer);
    // Открываем файл для записи
    $fd = @fopen($filename, "w");
    // Перезаписываем содержимое файла
    fwrite($fd, $bufer);
    // Закрываем файл
    fclose($fd);
    // Помещаем в суперглобальный массив $_GET имя файла
    $_GET['index'] = $_POST['index'];
  }
?>
<form action = "edit.php" name=first method="get">
   Номер записи <input type="text" name="index"
                      value=<?php echo $_GET['index']; ?>><br>
  <input type="submit" value="Отправить">
</form>
<?php
  // Если в строке запроса передан индекс,
  // открываем его для редактирования
  if(isset($_GET['index']))
  {
    // Помещаем содержимое файла в массив $lines
    $lines = file($filename);
    if($_GET['index']<1 || $_GET['index']>count($lines))
```

```
    {
      exit("К сожалению, такая запись не существует");
    }
    ?>
      <form action = "edit.php" name=second method="post">
        <textarea cols=50 rows=10 name="content">
        <?php echo $lines[$_GET['index']-1]; ?></textarea><br>
        <input type="hidden" name=index
               value='<?php echo $_GET['index']; ?>'>
        <input type="submit" name=edit value="Редактировать">
      </form>
    <?php
  }
?>
```

В листинге 6.35 определяются две формы, первая из которых, `first`, содержит текстовое поле `index` для номера записи из файла index.txt и кнопку для отправки данных обработчику, в качестве которого выступает этот же самый скрипт. Внешний вид формы представлен на рис. 6.5.

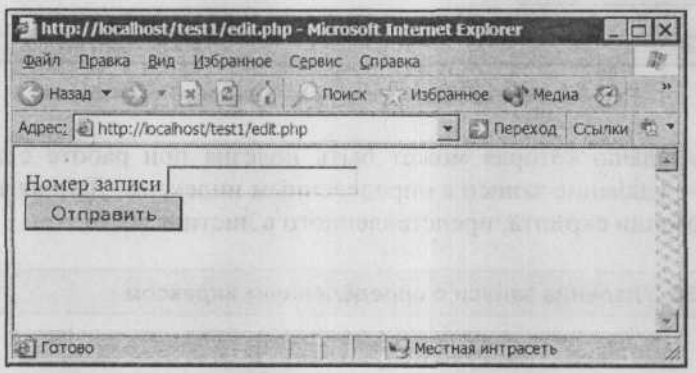


Рис. 6.5. Редактирование индексного файла. Первая форма

После ввода в форму номера записи страница перегружается, и обработчик, следующий сразу за формой, помещает содержимое файла в массив `$lines`. Если массив содержит элемент с индексом, переданным из первой формы, то происходит вывод второй формы `second`, содержащей текстовую область для редактирования записи (рис. 6.6). Кроме этого, форма содержит скрытое поле `index`, через которое передается индекс редактируемой записи.

После редактирования содержимого записи данные повторно отправляются скрипту в файле edit.php, но уже методом `POST`. В дело вступает обработчик, расположенный в начале файла. Он помещает содержимое редактируемого

файла во временную переменную `$bufer`, содержимое которой редактируется при помощи функции `preg_replace()`. После того как содержимое файла исправлено в переменной `$bufer`, происходит перезапись файла ее содержимым.

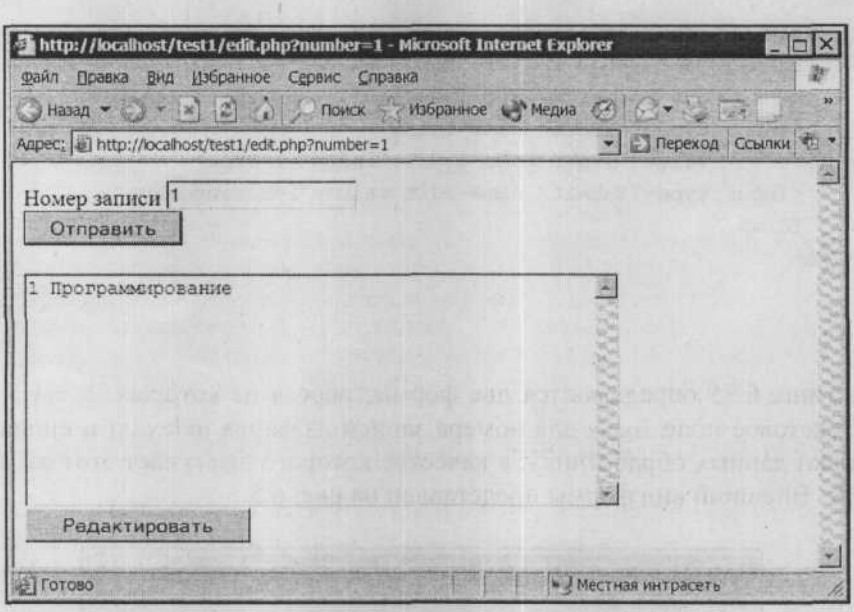


Рис. 6.6. Редактирование индексного файла. Вторая форма

Последняя задача, которая может быть полезна при работе с индексными файлами, — удаление записи с определенным индексом. Данная задача решается при помощи скрипта, представленного в листинге 6.36.

Листинг 6.36. Удаление записи с определенным индексом

```
<?php
  // Индекс строки
  $index = 2;
  // Имя файла
  $filename = "index.txt";
  // Открываем файл для чтения
  $fd = fopen($filename, "r");
  // Читаем содержимое файла
  $bufer = fread($fd, filesize($filename));
  // Закрываем файл
  fclose($fd);
  // Удаляем строку с индексом $index
  $bufer = preg_replace("|$index [^\n]*\n|", "", $bufer);
```

```
  // Открываем файл для записи
  $fd = fopen($filename, "w");
  // Перезаписываем содержимое файла
  fwrite($fd, $bufer);
  // Закрываем файл
  fclose($fd);
?>
```

В результате работы скрипта содержимое файла index.txt изменится следующим образом:

```
1 Программирование
3 Программирование на JavaScript
4 Программирование на ASP.NET
5 Новая строка
```

Блокировка файла

Рассмотренные выше скрипты осуществляют запись в файл в предположении, что скрипт владеет файлом единолично, но при записи в файл одновременно несколькими пользователями могут возникать конфликтные ситуации, приводящие к разрушению информации в файле.

Замечание

На Web-сленге операцию блокировки файла часто называют "залачиванием". Сленговое словечко "залачивать" происходит от английского глагола *to lock* — закрывать.

В PHP для предотвращения такой ситуации имеется специальная функция `flock()`, которая блокирует файл, не позволяя другим пользователям читать этот файл или писать в него до тех пор, пока процесс, наложивший блокировку, не закончит работу с файлом. Синтаксис функции `flock()` следующий:

```
bool flock(resource fd, int operation [, int &wouldblock])
```

В качестве первого аргумента функция принимает дескриптор файла, возвращаемый функцией `fopen()`. В качестве второго аргумента функции передается константа, сообщающая тип операции, совершаемой над файлом. Допустимы следующие значения:

- `LOCK_SH` — чтение файла несколькими потоками;
- `LOCK_EX` — блокировка для использования файла одним потоком (для записи);
- `LOCK_UN` — снятие блокировки с файла, когда оно больше не требуется, чтобы остальные участники могли получить к нему доступ.

Третий необязательный параметр *wouldblock* устанавливается в `true`, если файл, к которому происходит обращение, уже блокирован другим скриптом.

Функцию `flock()` следует вызывать сразу после открытия файла для установки блокировки и непосредственно перед закрытием файла для снятия блокировки.

Пример — в листинге 6.37.

Листинг 6.37. Блокировка файла

```
<?php
  // Имя файла
  $filename = "index.txt";
  // Открываем файл для записи
  $fd = fopen($filename, "w");
  // Блокируем файл
  flock($fd, LOCK_EX);
  // Перезаписываем содержимое файла
  fwrite($fd, "Текст для записи в файл index.txt");
  // Снимаем блокировку с файла
  flock($fd, LOCK_UN);
  // Закрываем файл
  fclose($fd);
?>
```

В реальных Web-приложениях необходимы более сложные схемы блокировки файлов. Так при добавлении записи в файловый вариант гостевой книги, чата или счетчика посещений, если файл заблокирован предыдущей операцией, информация будет потеряна, поэтому следует организовывать цикл ожидания, как это продемонстрировано в листинге 6.38.

Листинг 6.38. Реализация сложной схемы блокировки

```
<?php
  function work_with_file($filename, $str)
  {
    // Открываем файл
    if($fd = @fopen($filename, "a"))
    {
      // В течение 10 секунд пытаемся получить
      // единоличный доступ к файлу
      for($i = 0; $i<10; ++$i)
```

```
    {
      // Как только доступ получен, выходим из цикла
      if(flock($fd, LOCK_EX)) break();
      // Задерживаем время выполнения программы на 1 секунду,
      // если доступ получить не удалось
      else sleep(1);
    }
    // Записываем строку в файл
    fwrite($fd, $str);
    // Очищаем буфер записи
    flush($fd);
    // Снимаем блокировку с файла
    flock($fd, LOCK_UN);
    // Закрываем файл
    fclose($fd);
    // Покидаем функцию
    return true;
  }
  else
  {
    error("Нет доступа к файлу или он не существует!", $file);
    return false;
  }
}
?>
```

В листинге представлена функция `work_with_file()`, которая в качестве первого параметра `$filename` принимает имя файла, а в качестве второго параметра `$str` — информацию, которую необходимо дописать в файл. При этом блокировка файла осуществляется в цикле: при неудачной попытке заблокировать файл происходит задержка выполнения программы с помощью функции `sleep()`, которая принимает единственный параметр — время задержки в секундах.

Сохранение и извлечение из файла массивов и объектов

В *гл. 4*, посвященной строкам, были рассмотрены функции `serialize()` и `unserialize()`, позволяющие осуществлять упаковку и распаковку, соответственно, массивов и объектов. Эти функции очень удобно использовать при работе с файлами, т. к. упакованный в строку массив нужно где-то хранить (листинг 6.39).

Листинг 6.39. Использование функции `serialize()` при работе с файлами

```
<?php
  $poll[0] = 23;
  $poll[1] = 45;
  $poll[2] = 34;
  $poll[3] = 2;
  $poll[4] = 12;
  // Упаковываем массив в строку
  $str = serialize($poll);
  // Помещаем строку в файл
  $fd = fopen("text.txt","w");
  // Сохраняем в файле упакованный массив
  fwrite($fd, $str);
  // Закрываем файл
  fclose($fd);
?>
```

После того как упакованный массив сохранен в файле, в другом скрипте его можно извлечь при помощи кода, представленного в листинге 6.40.

Листинг 6.40. Использование функции `unserialize()` при работе с файлами

```
<?php
  // Открываем файл
  $fd = fopen("text.txt", "r");
  // Читаем содержимое файла во временную переменную $bufer
  $bufer = fread($fd, filesize("text.txt"));
  // Закрываем файл
  fclose($fd);
  // Извлекаем массив из переменной $bufer
  $arr = unserialize($bufer);
  // Выводим дамп массива
  print_r($arr);
?>
```

Результатом работы скрипта из листинга 6.40 будет следующий дамп:

```
Array
(
    [0] => 23
    [1] => 45
    [2] => 34
```

```
    [3] => 2
    [4] => 12
)
```

Работа с csv-файлами, или как загрузить данные из MS Excel

Для чтения файлов с расширением csv применяется функция `fgetcsv()`:

```
array fgetcsv(int file, int length, char delim)
```

Функция читает строку из файла и разбивает ее по символу *delim*. Строковый параметр *delim* должен содержать только один символ, иначе учитывается лишь первый символ строки. Функция возвращает получившийся массив или `false`, если достигнут конец файла. Пустые строки в файле не игнорируются, а возвращаются как массив из одного элемента — пустой строки. Параметр *length* задает максимальную длину строки точно так же, как это делается в функции `fgets()`.

Формат CSV является одним из форматов, в котором может сохранять файлы MS Excel. В листинге 6.41 приведен пример чтения созданного MS Excel файла file.csv, содержащего пароли пользователей.

Листинг 6.41. Чтение csv-файла

```
<?php
  // Имя файла
  $filename = "file.csv";
  // Открываем файл
  $fd = fopen ("file.csv", "r");
  // В цикле читаем строки файла
  while ($data = fgetcsv ($fd, 1000, ";"))
  {
    foreach($data as $element) echo "$element   ";
    echo "<br>";
  }
  // Закрываем файл
  fclose ($fd);
?>
```

Задания

6.1. Создайте скрипт, который осуществляет подсчет числа байтов в файлах текущего каталога.

6.2. Осуществите рекурсивный спуск по каталогу C:/Windows, подсчитав количество файлов в этом каталоге и во всех вложенных каталогах.

6.3. Создайте текстовый файл, содержащий несколько текстовых строк. Перепишите файл таким образом, чтобы порядок следования строк файле поменялся: первая строка — на последнем месте, вторая — на предпоследнем, ..., последняя — на первом.

6.4. Откройте любой текстовый файл и подсчитайте число пробелов, входящих в его состав.

6.5. Откройте любой текстовый файл и удалите из него все пробельные символы.

ГЛАВА 7

Плоские файлы

Операции записи и чтения осуществляются быстрее при работе с файловой системой, по сравнению с иерархической базой данных и, конечно, быстрее, чем при работе с реляционной базой данных. Поэтому там, где нужна скорость (например, в поисковых системах), прибегают к *плоским файлам* — API (набор функций) к драйверу, обеспечивающему простейшие операции помещения записи в файл и извлечения их. То есть создается подобие примитивной базы данных, обеспечивающей работу с файлами как с набором строк. Разумеется, в случае плоских файлов не может идти речи о поиске или каких-либо встроенных функциях — для этого предназначены реляционные базы данных, например MySQL, которая рассмотрена в следующей главе.

Что такое плоские файлы и для чего они нужны?

В предыдущей главе были продемонстрированы приемы работы с индексными файлами, состоящими из записей, каждая из которых представляет собой пару "ключ — значение". Плоские файлы — это точно такие же файлы, но часто в двоичном или упакованном виде, позволяющем быстрее осуществлять операции извлечения, удаления и редактирования записей при помощи стандартного набора функций, избавляя программиста от забот о создании собственных библиотек для работы с текстовыми файлами. В состав PHP входит большое число расширений, позволяющих работать с различными видами плоских файлов и разновидностей баз данных Berkeley. Стандартным в настоящий момент считается доступ через расширение dba-функций (Database Abstraction Layer) — единый интерфейс к такого рода файлам.

Замечание

Для подключения расширения dba, которое, как и все остальные расширения, по умолчанию отключено, следует снять комментарий с директивы `extension=php_dba.dll` в конфигурационном файле php.ini.

Создание файла

Для открытия и создания плоского файла предназначена функция `dba_open()`, которая имеет следующий синтаксис:

```
resource dba_open(string name, string mode, string handler)
```

В качестве первого аргумента функция принимает параметр *name* с указанием имени файла. Второй параметр *mode* содержит символы, задающие режим открытия файла, и может принимать следующие значения:

- `r` — чтение;
- `w` — запись;
- `c` — создание.

Последний параметр *handler* определяет тип базы данных или плоского файла и может принимать следующие значения: `dbm`, `ndbm`, `gdbm`, `db2`, `db3`, `db4`, `cdb`, `cdb_make`, `flatfile`, `inifile` и `qdbm`. В основном это различные разновидности баз данных Berkeley DB. Нас будет интересовать тип `flatfile` — плоский файл.

Замечание

Не все форматы доступны по умолчанию, ряд из них конфликтует друг с другом. Для того чтобы выяснить, какие базы данных доступны в системе, следует воспользоваться функцией `dba_handlers()`, которая не принимает никаких аргументов и возвращает массив из имен доступных баз данных. Самый простой способ получить список баз данных — выполнить команду `print_r(dba_handlers())`.

Замечание

Функция `dba_list()`, которая не принимает никаких аргументов, возвращает массив из полных путей к файлам, открытых в настоящий момент.

В случае успешного выполнения функция возвращает дескриптор открытого файла, а в случае не удачи — `false`. В листинге 7.1 приведен пример создания плоского файла.

Листинг 7.1. Создание плоского файла

```
<?php
  // Создаем файл test.db
  $id = dba_open("test.db", "c", "flatfile");
  // Закрываем соединение
  dba_close($id);
?>
```

Результатом работы будет пустой файл test.db. Обратите внимание, что, как и обычные файлы, в конце скрипта следует закрыть соединение, для чего предназначена специальная функция `dba_close()`.

Заполнение файла

После того как файл создан, можно приступать к заполнению его информацией. Для вставки в файл записей типа "ключ — значение" предназначена функция `dba_insert()`, которая имеет следующий синтаксис:

```
bool dba_insert(string key, string value, resource handle)
```

В качестве первого аргумента *key* функция принимает строку с ключом записи, в качестве второго *value* — значение записи. Последний параметр определяет значение, которое передается дескриптору открытого функцией `mda_open()` файла. При успешном выполнении функция возвращает `true`, в случае неудачи — `false`.

Пример — в листинге 7.2.

Листинг 7.2. Заполнение файла

```
<?php
  // Открываем файл test.db
  $id = dba_open("test.db", "w", "flatfile");
  // Проверяем корректность открытия файла
  if (!$id) exit("Невозможно открыть файл");
  // Функция dba_insert принимает три параметра:
  // первый - ключ, второй - значение, третий - дескриптор
  dba_insert ("1", "Hello world!", $id);
  dba_insert ("2", "Hello PHP!", $id);
  // Закрываем файл
  dba_close($id);
?>
```

Код в листинге 7.2 заносит в ранее созданный файл test.db две записи.

Замечание

Формат `flatfile` не позволяет помещать записи с одинаковыми значениями ключей. Если в файл необходимо поместить несколько записей с одинаковыми ключами, следует воспользоваться форматом, поддерживающим такое поведение, например, cdb.

Замечание

Плоские файлы не допускают вставку нескольких значений для одной записи. Для того чтобы сохранить в одной записи несколько значений, следует размес-

тить их в массиве и упаковать его в строку при помощи функции `serialize()`, а при извлечении записи распаковать обратно в массив при помощи функции `unserialize()`. Подробнее работа с данными функциями описана в *гл. 6*.

Чтение информации из файла

После того как файл создан, из него можно извлекать занесенную ранее информацию. Для этого предназначен набор функций, первой из которых является функция `dba_exists()`, которая проверяет существование ключа в плоском файле и имеет следующий синтаксис:

```
bool dba_exists(string key, resource handle)
```

Функция принимает в качестве первого аргумента ключ *key*, а в качестве второго — дескриптор файла, возвращаемый функцией `mda_open()`. Если ключ *key* найден в файле, возвращается `true`, в противном случае — `false`. В листинге 7.3 приведен пример работы функции `dba_exists()`.

Листинг 7.3. Функция `dba_exists()`

```
<?php
  // Открываем файл test.db
  $id = dba_open("test.db", "r", "flatfile");
  // Проверяем корректность открытия файла
  if (!$id) exit("Невозможно открыть файл");
  // Функция dba_insert принимает три параметра:
  // первый - ключ, второй - значение, третий - дескриптор
  if(dba_exists("1",$id)) echo "Ключ существует<br>";
  else echo "Такой ключ отсутствует<br>";
  if(dba_exists("4",$id)) echo "Ключ существует<br>";
  else echo "Такой ключ отсутствует<br>";
  // Закрываем файл
  dba_close($id);
?>
```

Первый вызов функции приводит к выводу строки `"Ключ существует"`, т. к. ранее в файл test.db нами была добавлена запись с ключом `"1"`. Проверка на существование записи с ключом `"4"` приводит к выводу записи `"Ключ отсутствует"`.

После того как существование ключа проверено, можно осуществлять извлечение записи. Для этого предназначена функция `dba_fetch()`, которая по ключу извлекает из файла соответствующее ему название и имеет следующий синтаксис:

```
string dba_fetch(string key, resource handle)
```

В качестве первого аргумента *key* передается ключ файла, в качестве второго — дескриптор, возвращаемый функцией `mda_open()`. Функция возвращает строковое значение, соответствующее ключу.

Замечание

Если используется база данных, поддерживающая создание записей с одинаковыми ключами, например cdb, после параметра *key* допускается использование дополнительного параметра *skip*, задающего число записей с ключом *key*, которые следует пропустить.

Пример — в листинге 7.4.

Листинг 7.4. Функция `dba_fetch()`

```
<?php
  // Открываем файл test.db
  $id = dba_open("test.db", "r", "flatfile");
  // Проверяем корректность открытия файла
  if (!$id) exit("Невозможно открыть файл");
  $key = "2";
  // Если ключ существует
  if(dba_exists($key, $id))
  {
    // Извлекаем соответствующее ему значение
    echo dba_fetch($key, $id);
  }
  // Закрываем файл
  dba_close($id);
?>
```

В результате работы скрипта из листинга 7.4 в окно браузера будет выведена строка

```
Hello PHP!
```

Если ключи плоского файла заранее не известны, имеется возможность последовательного извлечения записей из файла, начиная с первой записи. Функция `dba_firstkey()` возвращает ключ первой записи и имеет следующий синтаксис:

```
string dba_firstkey(resource handle)
```

Функция принимает дескриптор открытого файла *handle* и возвращает ключ первой записи файла. В случае неудачи функция возвращает `false`.

Функция dba_nextkey() возвращает ключ следующей записи. Последовательным вызовом данной функции можно перебрать весь файл. После того как функция вернет ключ последней записи, она возвращает flase.

Пример — в листинге 7.5.

Листинг 7.5. Последовательное чтение плоского файла

```
<?php
  // Открываем файл test.db
  $id = dba_open("test.db", "r", "flatfile");
  // Проверяем корректность открытия файла
  if (!$id) exit("Невозможно открыть файл");
  // Извлекаем ключ первой записи
  $key = dba_firstkey($id);
  // Выводим значение первой записи
  echo dba_fetch($key, $id)."<br>";
  // В цикле извлекаем следующие записи
  while($key = dba_nextkey($id))
  {
   echo dba_fetch($key, $id)."<br>";
  }
  // Закрываем файл
  dba_close($id);
?>
```

В листинге 7.5 продемонстрировано последовательное чтение плоского файла — в цикле выводятся значения всех записей файла.

Замена записи

По ключу можно заменить значение уже существующей записи на новое. Для этого предназначена функция dba_replace(), которая имеет следующий синтаксис:

```
bool dba_replace(string key, string value, resource handle)
```

Функция принимает в качестве первого параметра *key* ключ записи, в качестве второго *value* — значение, в качестве третьего *handle* — дескриптор, возвращенный функцией dba_open(). Функция возвращает true в случае успешного выполнения и false в случае неудачи.

Замечание

Если в качестве ключа *key* функции dba_replace() будет передан ключ несуществующей записи, такая запись будет создана.

Пример — в листинге 7.6.

Листинг 7.6. Функция dba_replace()

```
<?php
  // Открываем файл test.db
  $id = dba_open("test.db", "w", "flatfile");
  // Проверяем корректность открытия файла
  if (!$id) exit("Невозможно открыть файл");
  // Извлекаем ключ первой записи
  $key = "1";
  // Заменяем запись с ключом $key
  dba_replace($key, "Hello", $id);
  // Выводим значение первой записи
  echo dba_fetch($key, $id)."<br>";
  // Закрываем файл
  dba_close($id);
?>
```

В результате работы скрипта в окно браузера будет выведена строка "Hello".

Удаление записи

Последней операцией, которую поддерживают плоские файлы, является операция удаления записей. Для удаления записей предназначена функция dba_delete(), которая имеет следующий синтаксис:

```
bool dba_delete(string key, resource handle)
```

Функция принимает в качестве первого параметра значение ключа *key* удаляемой записи, а в качестве второго параметра *handle* — дескриптор открытого файла. В случае удачного выполнения функции возвращается true, а в случае не удачи — false.

В листинге 7.7 производится удаление всех записей из файла test.db.

Листинг 7.7. Пример удаления записей из плоского файла

```
<?php
  // Открываем файл test.db
  $id = dba_open("test.db", "w", "flatfile");
  // Проверяем корректность открытия файла
  if (!$id) exit("Невозможно открыть файл");
  // Извлекаем ключ первой записи
  $key = dba_firstkey($id);
```

```
  // Удаляем первую запись
  dba_delete($key, $id);
  // Удаляем записи
  while($key = dba_nextkey($id))
  {
    dba_delete($key, $id);
  }
  // Закрываем файл
  dba_close($id);
?>
```

ГЛАВА 8

Работа с MySQL

Плоские файлы реализуют лишь базовые функции при работе с данными и представляют собой примитивные текстовые базы данных, лишенные собственных средств программирования, возможностей поиска и фильтрации данных. Все это с лихвой компенсирует база данных MySQL — одна из лучших баз данных в мире по скорости доступа к данным, ставшая стандартом дефакто в Web.

Замечание

MySQL является базой данных с открытым кодом и доступна для свободной загрузки с сайта **http://www.mysql.com** как в виде кодов, так и в откомпилированном виде для различных операционных систем, включая Windows и Linux.

Замечание

Полное справочное руководство на русском языке можно найти на официальном сайте MySQL по адресу **http://dev.mysql.com/doc/mysql/ru/index.html**.

Работа с базой данных имеет как свои преимущества, так и недостатки. К достоинствам можно отнести значительное сокращение кода (иногда в 2—3 раза) по сравнению с файловыми вариантами Web-приложений, что сокращает время разработки и упрощает процесс отладки. К недостаткам причисляется зависимость приложения не только от работоспособности Web-сервера, но и от работоспособности сервера баз данных. Разумеется, вероятность того, что один из двух серверов может выйти из строя, выше по сравнению, если бы работа Web-приложения зависела только от одного сервера. Кроме того, непосредственная работа с файловой системой осуществляется быстрее по сравнению со случаем, когда в качестве посредника для работы с ней выбирается СУБД. Так поисковая система Google, известная высокой скоростью обработки запросов, основана на модели плоских файлов и не использует базы данных. С другой стороны, PHP, как интерпретируемый язык, не может достичь производительности баз данных, реализованных на C.

Типы таблиц — почему их так много?

СУБД MySQL в настоящее время поддерживает несколько видов таблиц: ISAM, MyISAM, MERGE, HEAP, BDB и InnoDB.

Изначально СУБД MySQL планировалась как одна из быстрых баз данных и поддерживала единственный тип таблицы — ISAM, который обеспечивает индексно-последовательный метод доступа. Для ускорения выполнения запросов дескриптор ISAM не поддерживает транзакции и ряд операторов SQL.

Тип таблицы ISAM является устаревшим. Он применялся в ранних версиях MySQL, использование его в настоящее время нежелательно, т. к. в будущем его поддержка может быть удалена. Этот вид таблиц был заменен на новый — MyISAM, являющийся стандартным типом по умолчанию. Основные преимущества по сравнению с таблицами типа ISAM заключаются в следующем:

- поддерживается больший размер таблиц;
- содержимое таблиц хранится в платформонезависимом формате;
- более эффективная работа с индексами и атрибутом `AUTO_INCREMENT`;
- более эффективная проверка целостности таблицы;
- поддержка полнотекстового поиска с использованием индекса `FULLTEXT`.

Таблицы MERGE предназначены для объединения нескольких таблиц MyISAM в одну, позволяя при помощи одного запроса обращаться ко всем таблицам, входящим в ее состав. Главным преимуществом такого объединения является то, что итоговая таблица может превышать предельный размер, установленный для таблиц MyISAM.

Таблицы HEAP — это вре́менные таблицы, предназначенные для хранения в оперативной памяти. Для повышения эффективности в них применяются только строки фиксированной длины.

Таблицы типа BDB поддерживаются дескриптором Berkeley DB, разработанным компанией Sleepycat. Дескриптор BDB обеспечивает:

- обработку таблиц с использованием транзакций;
- автоматическое восстановление после сбоев;
- блокирование на уровне страниц, обеспечивающее хорошую производительность при обработке параллельных запросов.

Таблицы типа InnoDB являются самыми новыми, недавно добавленными в СУБД MySQL. Этот тип таблиц поддерживается дескриптором `InnoDB`, который был разработан компанией Innobase Oy. Он обеспечивает следующие возможности:

- обработку таблиц с использованием транзакций;
- автоматическое восстановление после сбоев;
- поддержку ключей, в том числе каскадное удаление;
- блокирование на уровне строк, обеспечивающее хорошую производительность при обработке параллельных запросов;
- распределение таблиц по нескольким файлам или даже разделам диска, что позволяет выходить за пределы, установленные файловой системой.

Основы SQL

Язык структурированных запросов SQL (Structure Query Language) позволяет выполнять различные операции с базами данных: создавать таблицы, помещать, обновлять и удалять из них данные, производить запросы из таблиц и т. д. Далее мы последовательно рассмотрим все эти операции.

Замечание

Несмотря на то, что последний стандарт SQL принят в 1992 г., на сегодняшний день нет ни одной базы данных, где бы он полностью выполнялся. Более того, в различных базах данных часть операций осуществляется по-разному. Мы будем придерживаться диалекта SQL, характерного для СУБД MySQL, поэтому не все запросы могут выполняться для других баз данных.

Замечание

В состав дистрибутива MySQL входит консольный клиент mysql, найти который можно в каталоге /mysql/bin. Это не единственное средство общения с базой данных. На сайте **http://www.mysql.com** доступны для свободной загрузки некоторые графические клиенты, такие как MySQL Control Center, MySQL Query Browser и др.

Далее в главе будут рассмотрены основные операторы SQL.

CREATE DATABASE

Эта команда создает новую базу данных:

```
CREATE DATABASE [IF NOT EXISTS] db_name;
```

Здесь *db_name* является именем создаваемой базы данных. Для того чтобы создать новую базу данных `forum`, наберите в строке-приглашении[1] клиента mysql эту команду и укажите название базы данных (листинг 8.1).

[1] Здесь и далее подчеркнутая строка означает приглашение клиента. Разумеется, это приглашение вводить в командной строке не надо. — *Ред.*

Листинг 8.1. Создание базы данных

```
mysql> CREATE DATABASE forum;
```

Замечание

Каждый запрос MySQL в клиенте mysql завершается точкой с запятой (;). При выполнении запросов из PHP-скрипта отсутствие завершающей точки с запятой не приводит к ошибке.

Замечание

Клиент mysql расположен в каталоге bin базы данных MySQL. По умолчанию при установке MySQL создаются два пользователя: анонимный, в качестве имени которого выступает пустая строка, и суперпользователь, имя которого — `root`. Для анонимного пользователя по умолчанию разрешен не полный набор SQL-операторов, поэтому из-под него может не получиться создать базу данных. Тогда клиент mysql следует загрузить с правами суперпользователя: `mysql -u root`.

Не обязательная ключевая фраза `IF NOT EXISTS` сообщает, что базу данных следует создавать, только если база данных с таким именем отсутствует, что позволяет предотвратить завершение запроса ошибкой. Особенно это актуально при использовании SQL в PHP-скриптах.

Для того чтобы убедиться, что база данных `forum` успешно создана, можно выполнить команду `SHOW DATABASES` (листинг 8.2), которая покажет, какие базы данных существуют в вашей системе.

Листинг 8.2. Просмотр созданных баз данных

```
mysql> SHOW DATABASES;
```

Как видим, среди различных баз данных присутствует и только что созданная база данных `forum` (рис. 8.1).

```
+----------+
| Database |
+----------+
| auth     |
| count    |
| forum    |
| guest    |
| local    |
| mysql    |
| test     |
+----------+
7 rows in set (0.00 sec)
```

Рис. 8.1. Созданные базы данных

Замечание

Команда SHOW является внутренней командой MySQL, отсутствующей в стандарте SQL и неподдерживаемой другими базами данных. Далее по мере рассмотрения других операторов SQL будут приведены иные варианты использования команды SHOW.

Замечание

Изначально в MySQL присутствуют только две базы данных: test и mysql. В последней хранится системный каталог привилегий пользователей СУБД MySQL.

USE

Для того чтобы начать работу с таблицами, необходимо сообщить MySQL, с какой базой данных вы намерены работать. Это осуществляется при помощи команды USE:

```
USE db_name;
```

Здесь *db_name* — название выбираемой базы данных. Выберем созданную базу forum (листинг 8.3).

Листинг 8.3. Выбор базы данных

```
mysql> CREATE DATABASE forum;
Database changed;
```

Замечание

При работе с клиентом mysql базу данных можно выбрать непосредственно при его загрузке, передав имя базы данных в качестве параметра, к примеру: mysql forum.

CREATE TABLE

Команда CREATE TABLE создает новую таблицу в выбранной базе данных. В простейшем случае команда имеет следующий синтаксис:

```
CREATE TABLE table_name [(create_definition, ...)][table_options];
```

Здесь *table_name* — имя создаваемой таблицы; *create_definition* — объявление столбца, его типа и атрибутов. В конце оператора может следовать необязательное указание типа таблицы *table_options*, например, TYPE = MyISAM.

Создадим таблицу базы данных forum, которая называется authors и содержит различные данные о зарегистрированных посетителях форума: имя (name),

пароль (passw), e-mail (email), Web-адрес сайта посетителя (url), номер ICQ (icq), сведения о посетителе (about), строку, содержащую путь к файлу фотографии посетителя (photo), время добавления запроса (time), последнее время посещения форума (last_time), счетчик сообщений, оставленных посетителем на форуме (themes), статус посетителя — является ли он модератором, администратором или обычным посетителем (statususer). Кроме перечисленных полей в таблице имеется поле id_author, являющееся первичным ключом таблицы.

SQL-запрос[1], создающий эту таблицу, приведен в листинге 8.4.

Листинг 8.4. Создание таблицы authors базы данных forum

```
mysql> CREATE TABLE authors (
    -> id_author INT NOT NULL AUTO_INCREMENT,
    -> name TINYTEXT,
    -> passw TINYTEXT,
    -> email TINYTEXT,
    -> url TINYTEXT,
    -> icq TINYTEXT,
    -> about TINYTEXT,
    -> photo TINYTEXT,
    -> time DATETIME DEFAULT NULL,
    -> last_time DATETIME DEFAULT NULL,
    -> themes INT DEFAULT NULL,
    -> statususer INT DEFAULT NULL,
    -> PRIMARY KEY (id_author)
    ->) TYPE=MyISAM;
```

Выполнив SQL-команду SHOW TABLES, можно убедиться, что таблица authors успешно создана (рис. 8.2).

```
+---------------+
| Tables_in_forum |
+---------------+
| authors       |
+---------------+
1 row in set (0.05 sec)
```

Рис. 8.2. Таблица authors успешно создана

[1] Символ -> означает продолжение строки запроса. Набираться с клавиатуры он не должен, клиент mysql сам автоматически выставляет его при переходе на другую строку.

DESCRIBE

Команда DESCRIBE показывает структуру созданных таблиц и имеет следующий синтаксис:

```
DESCRIBE table_name;
```

Здесь *table_name* — имя таблицы, структура которой запрашивается.

Замечание

Команда DESCRIBE не входит в стандарт SQL и является внутренней командой СУБД MySQL.

Просмотреть структуру таблицы authors можно, выполнив SQL-запрос из листинга 8.5.

Листинг 8.5. Команда DESCRIBE

```
mysql> DESCRIBE authors;
```

После выполнения этой команды интерпретатор MySQL выведет таблицу, изображенную на рис. 8.3.

Field	Type	Null	Key	Default	Extra
id_author	int(11)		PRI	NULL	auto_increment
name	tinytext	YES		NULL	
passw	tinytext	YES		NULL	
email	tinytext	YES		NULL	
url	tinytext	YES		NULL	
icq	tinytext	YES		NULL	
about	tinytext	YES		NULL	
photo	tinytext	YES		NULL	
time	datetime	YES		NULL	
last_time	datetime	YES		NULL	
themes	int(11)	YES		NULL	
statususer	int(11)	YES		NULL	

Рис. 8.3. Структура таблицы authors

Замечание

Более полное описание структуры таблицы authors, включающее права доступа и комментарии, можно получить, воспользовавшись оператором SHOW FULL COLUMNS FROM authors;.

ALTER TABLE

Команда ALTER TABLE предназначена для изменения структуры таблицы. Эта команда позволяет добавлять и удалять столбцы, создавать и уничтожать ин-

дексы, переименовывать столбцы и саму таблицу. Команда имеет следующий синтаксис:

```
ALTER TABLE table_name alter_spec;
```

Основные значения параметра *alter_spec* приведены в табл. 8.1.

***Таблица 8.1.** Основные преобразования, выполняемые оператором* `ALTER TABLE`

Синтаксис	Описание команды
`ADD create_definition [FIRST|AFTER column_name]`	Добавление нового столбца *create_definition*. Этот параметр представляет собой название нового столбца и его тип. Конструкция `FIRST` добавляет новый столбец перед столбцом *column_name*. Конструкция `AFTER` добавляет новый столбец после столбца *column_name*. Если место добавления не указано, по умолчанию столбец добавляется в конец таблицы
`ADD INDEX [index_name] (index_col_name, ...)`	Добавление индекса *index_name* для столбца *index_col_name*. Если имя индекса *index_name* не указывается, ему присваивается имя, совпадающее с именем столбца *index_col_name*
`ADD PRIMARY KEY (index_col_name, ...)`	Делает столбец *index_col_name* или группу столбцов первичным ключом таблицы
`CHANGE old_col_name new_col_name type`	Изменение столбца с именем *old_col_name* на столбец с именем *new_col_name* и типом *type*
`DROP col_name`	Удаление столбца с именем *col_name*
`DROP PRIMARY KEY`	Удаление первичного ключа таблицы
`DROP INDEX index_name`	Удаление индекса *index_name*

Добавление в таблицу `authors` нового столбца `test` с размещением его после столбца `name` можно выполнить SQL-запросом из листинга 8.6.

Листинг 8.6. Добавление столбца в таблицу

```
mysql> ALTER TABLE authors ADD test INT(10) AFTER name;
```

Выполнив команду `DESCRIBE forums`, можно увидеть, что столбец `test` успешно добавлен после столбца `name` (рис. 8.4).

Переименование столбца `test` в текстовый столбец `new_test` можно осуществить следующим образом (листинг 8.7).

Field	Type	Null	Key	Default	Extra
id_author	int(11)		PRI	NULL	auto_increment
name	tinytext	YES		NULL	
test	int(10)	YES		NULL	
passw	tinytext	YES		NULL	
email	tinytext	YES		NULL	
url	tinytext	YES		NULL	
icq	tinytext	YES		NULL	
about	tinytext	YES		NULL	
photo	tinytext	YES		NULL	
time	datetime	YES		NULL	
last_time	datetime	YES		NULL	
themes	int(11)	YES		NULL	
statususer	int(11)	YES		NULL	

Рис. 8.4. Новый столбец успешно добавлен в таблицу

Листинг 8.7. Переименование столбца

```
mysql> ALTER TABLE authors CHANGE test new_test text;
```

Как видно из рис. 8.5, столбец успешно переименован.

Field	Type	Null	Key	Default	Extra
id_author	int(11)		PRI	NULL	auto_increment
name	tinytext	YES		NULL	
new_test	text	YES		NULL	
passw	tinytext	YES		NULL	
email	tinytext	YES		NULL	
url	tinytext	YES		NULL	
icq	tinytext	YES		NULL	
about	tinytext	YES		NULL	
photo	tinytext	YES		NULL	
time	datetime	YES		NULL	
last_time	datetime	YES		NULL	
themes	int(11)	YES		NULL	
statususer	int(11)	YES		NULL	

Рис. 8.5. Столбец переименован

Удаление столбца `new_test` можно осуществить следующим запросом из листинга 8.8.

Листинг 8.8. Удаление столбца из таблицы

```
mysql> ALTER TABLE authors DROP new_test;
```

DROP TABLE

Команда `DROP TABLE` предназначена для удаления одной или нескольких таблиц:

```
DROP TABLE table_name [,table_name, ...];
```

К примеру, для удаления таблицы forums нужно выполнить SQL-запрос из листинга 8.9.

Листинг 8.9. Удаление таблиц

```
mysql> DROP TABLE authors;
```

DROP DATABASE

Команда DROP DATABASE удаляет базу данных со всеми таблицами, входящими в ее состав:

```
DROP DATABASE database_name
```

Так удалить базу данных forum можно следующим SQL-запросом (листинг 8.10).

Листинг 8.10. Удаление базы данных

```
mysql> DROP DATABASE forum;
```

INSERT INTO...VALUES

Команда INSERT INTO...VALUES вставляет новые записи в существующую таблицу. Синтаксис команды:

```
INSERT INTO table_name VALUES (values, ...);
```

После оператора VALUES в скобках через запятую перечисляются все значения полей таблицы в соответствии с их типами.

Для того чтобы вставить в базу данных authors несколько записей, в которых расположена информация о зарегистрированных посетителях форума, можно воспользоваться несколькими операторами INSERT (листинг 8.11).

Листинг 8.11. Оператор INSERT

```
mysql> INSERT INTO authors VALUES (0, 'Maks', '123', 'maks@mail.ru',
    ->                             'www.softtime.ru ', '', 'программист',
    ->                             '', '', '', 0, 0);
mysql> INSERT INTO authors VALUES (0, 'Igor', '123', 'igor@mail.ru',
    ->                             'www.softtime.ru', '', 'Программист',
    ->                             '', '', '', 407, 0);
mysql> INSERT INTO authors VALUES (0, 'Sergey', '212', 'sergey@mail.ru',
    ->                             'www.softtime.ru', '', 'Дизайнер',
    ->                             '', '', '', 408, 0);
```

Начиная с версии MySQL 3.22.5, можно добавить сразу несколько записей с помощью многострочного оператора INSERT (листинг 8.12).

Листинг 8.12. Многострочный оператор

```
mysql> INSERT INTO authors VALUES (0, 'Maks', '123', 'maks@mail.ru',
    ->                             'www.softtime.ru ', '', 'программист',
    ->                             '', '', '', 0, 0),
    ->                             (0, 'Igor', '123', 'igor@mail.ru',
    ->                             'www.softtime.ru', '', 'Программист',
    ->                             '', '', '', 407, 0),
    ->                             (0, 'Sergey', '212', 'sergey@mail.ru',
    ->                             'www.softtime.ru', '', 'Дизайнер', '',
    ->                             '', '', 408, 0);
```

При использовании такого оператора записи приводятся в круглых скобках через запятую после ключевого слова VALUES.

Порядок добавления столбцов можно задавать самостоятельно, воспользовавшись формой оператора INSERT:

```
INSERT INTO tbl_name (col_name1,col_name2,...) VALUES
(value1,value2,...);
```

Например, запись в таблицу authors можно осуществить при помощи SQL-запроса, представленного в листинге 8.13. При этом значения полей, не указанных в списке, следующим за названием таблицы authors, устанавливаются в значения по умолчанию. Так, первичный ключ получает значение NULL, которое интерпретируется для полей с атрибутом AUTO_INCREMENT точно так же, как и значение 0 — происходит генерация уникального числа.

Листинг 8.13. Определение порядка добавления столбцов

```
mysql> INSERT INTO authors (name, mail, url, about)
VALUES ('Maks','maks@mail.ru','www.softtime.ru','программист');
```

Версия MySQL 3.22.10 позволяет задавать значения полей в операторе INSERT в форме *col_name* = *value* (листинг 8.14).

Листинг 8.14. Альтернативная форма оператора INSERT

```
mysql> INSERT INTO authors SET name='Maks', mail='maks@mail.ru',
mysql>                         url='www.softtime.ru ',
mysql>                         about='программист',);
```

Здесь также для полей, не получивших значения, будут выставлены значения по умолчанию.

DELETE

Команда DELETE удаляет из таблицы *table_name* записи, удовлетворяющие заданным в *definition* условиям, и возвращает число удаленных записей.

```
DELETE FROM table_name [WHERE definition];
```

Вот как можно удалить все записи из таблицы authors (листинг 8.15).

Листинг 8.15. Удаление записей

```
mysql> DELETE FROM authors;
```

Важной частью запросов DELETE, UPDATE и SELECT является оператор WHERE, который позволяет задать условия для выбора записей, на которые будут действовать эти команды. Запрос из листинга 8.16 удаляет из таблицы посетителя, первичный ключ для которого равен 1.

Листинг 8.16. Удаление записей по указанному критерию

```
mysql> DELETE FROM authors WHERE id_author = 1;
```

Условия отбора могут быть значительно сложнее. Так в листинге 8.17 удаляются все авторы с паролем '123', первичный ключ которых превышает значение 10.

Листинг 8.17. Удаление записей со сложным критерием

```
mysql> DELETE FROM authors WHERE passw = '123' AND id_author > 10;
```

Оператор AND реализует логическое И. В запросах можно так же применять логическое ИЛИ — OR.

Замечание

СУБД MySQL в качестве альтернативы операторами AND и OR поддерживает синтаксис, принятый в С-подобных языках программирования, т. е. вместо AND можно применять &&, а вместо OR — ||.

SELECT

Команда SELECT предназначена для извлечения строк данных из одной или нескольких таблиц и имеет в общем случае следующий синтаксис:

```
SELECT column, ...
  [FROM table WHERE definition]
  [ORDER BY col_name [ASC|DESC], ...]
  [LIMIT [offset], rows];
```

Здесь *column* — имя выбираемого столбца. Можно указать несколько столбцов через запятую. Если необходимо выбрать все столбцы, можно просто ввести символ звездочки (*). Ключевое слово FROM указывает таблицу *table*, из которой извлекаются записи. Ключевое слово WHERE определяет, так же как и в операторе DELETE, условия отбора строк. Ключевое слово ORDER BY сортирует строки запросов по столбцу *col_name* в прямом (ASC) или обратном порядке (DESC). Ключевое слово LIMIT сообщает MySQL о выводе только *rows* запросов, начиная с позиции *offset*.

Для выборки из базы данных при помощи оператора SELECT создадим таблицу forums и добавим в нее несколько записей (листинги 2.18, 2.19).

Листинг 8.18. Создание таблицы forums

```
mysql> CREATE TABLE forums (
    -> id_forum INT(6) NOT NULL AUTO_INCREMENT,
    -> name TINYTEXT,
    -> rule TEXT,
    -> logo TEXT,
    -> pos INT DEFAULT NULL,
    -> hide TINYINT(1) DEFAULT NULL,
    -> PRIMARY KEY (id_forum)
    ->) TYPE=MyISAM;
```

В таблице forums присутствуют следующие поля: первичный ключ (id_forum), название раздела (name), правила форума (rule), краткое описание форума (logo), порядковый номер (pos), флаг, принимающий значение 1, если форум скрытый, и 0, если общедоступный (hide).

Листинг 8.19. Вставка 5 записей в таблицу forums

```
mysql> INSERT INTO forums VALUES (0, 'Форум1', '', '', 5, 0);
    -> INSERT INTO forums VALUES (0, 'Форум2', '', '', 4, 0);
    -> INSERT INTO forums VALUES (0, 'Форум3', '', '', 3, 0);
    -> INSERT INTO forums VALUES (0, 'Форум4', '', '', 2, 0);
    -> INSERT INTO forums VALUES (0, 'Форум5', '', '', 1, 0);
```

Для того чтобы посмотреть всю таблицу forums, выполняется запрос из листинга 8.20.

Листинг 8.20. Оператор SELECT

```
mysql> SELECT * FROM forums;
```

В данном запросе происходит выборка всех столбцов из таблицы forums без ограничений. Результат запроса показан на рис. 8.6.

id_forum	name	rule	logo	pos	hide
1	Форум1			5	
2	Форум2			4	
3	Форум3			3	
4	Форум4			2	
5	Форум5			1	

Рис. 8.6. Результат выполнения запроса из листинга 8.20

Можно выбрать не все столбцы таблицы, а лишь часть, для этого необходимо явно задать список выбираемых столбцов (листинг 8.21).

Листинг 8.21. Частичная выборка

```
mysql> SELECT id_forum, name FROM forums;
```

В этом случае MySQL выведет лишь два столбца с первичным ключом id_forum и названием форума name (рис. 8.7).

id_forum	name
1	Форум1
2	Форум2
3	Форум3
4	Форум4
5	Форум5

Рис. 8.7. Результат выполнения запроса из листинга 8.21

Оператор LIMIT используется для ограничения количества строк, возвращаемых командой SELECT (листинг 8.22).

Листинг 8.22. Использование ключевого слова LIMIT в операторе SELECT

```
mysql> SELECT * FROM forums LIMIT 3;
```

В результате этого запроса будут выведены только первые 3 записи из 5 (рис. 8.8).

```
+----------+--------+------+------+-----+------+
| id_forum | name   | rule | logo | pos | hide |
+----------+--------+------+------+-----+------+
|        1 | Форум1 |      |      |   5 |      |
|        2 | Форум2 |      |      |   4 |      |
|        3 | Форум3 |      |      |   3 |      |
+----------+--------+------+------+-----+------+
```

Рис. 8.8. Результат выполнения запроса из листинга 8.22

Оператор `LIMIT` может также принимать два целочисленных аргумента. В этом случае последний аргумент задает максимальное количество возвращаемых строк, а первый сообщает MySQL, начиная с какой строки производить отсчет (листинг 8.23).

Листинг 8.23. Альтернативная форма задания `LIMIT`

```
mysql> SELECT * FROM forums LIMIT 1,3;
```

В этом случае будут возвращены строки 2, 3 и 4 (рис. 8.9).

```
+----------+--------+------+------+-----+------+
| id_forum | name   | rule | logo | pos | hide |
+----------+--------+------+------+-----+------+
|        2 | Форум2 |      |      |   4 |      |
|        3 | Форум3 |      |      |   3 |      |
|        4 | Форум4 |      |      |   2 |      |
+----------+--------+------+------+-----+------+
```

Рис. 8.9. Результат выполнения запроса из листинга 8.23

Оператор `WHERE` применяется в команде `SELECT` точно так же, как и в команде `DELETE`. Выберем из таблицы только те записи, у которых значение `id_forum` больше 2 (листинг 8.24).

Листинг 8.24. Условный вариант оператора `SELECT`

```
mysql> SELECT * FROM forums WHERE id_forum > 2;
```

Результат показан на рис. 8.10.

```
+----------+--------+------+------+-----+------+
| id_forum | name   | rule | logo | pos | hide |
+----------+--------+------+------+-----+------+
|        3 | Форум3 |      |      |   3 |      |
|        4 | Форум4 |      |      |   2 |      |
|        5 | Форум5 |      |      |   1 |      |
+----------+--------+------+------+-----+------+
```

Рис. 8.10. Результат выполнения запроса из листинга 8.24

Порядок сортировки выводимых записей можно задавать при помощи оператора `ORDER BY` (листинг 8.25).

Листинг 8.25. Сортировка результатов запроса

```
mysql> SELECT * FROM forums WHERE id_forum > 2 ORDER BY pos;
```

В этом запросе выводятся все записи со значением поля `id_forum`, не меньше 2, которые при этом сортируются по значению поля `pos`. Результат такого запроса показан на рис. 8.11.

```
+----------+--------+------+------+-----+------+
| id_forum | name   | rule | logo | pos | hide |
+----------+--------+------+------+-----+------+
|        5 | Форум5 |      |      |   1 |      |
|        4 | Форум4 |      |      |   2 |      |
|        3 | Форум3 |      |      |   3 |      |
+----------+--------+------+------+-----+------+
```

Рис. 8.11. Результат выполнения запроса из листинга 8.25

Для дальнейшего рассмотрения оператора `SELECT` нам понадобятся еще несколько записей в таблице `forums`. Разместим при помощи SQL-запроса, приведенного в листинге 8.26, группу новых форумов со значением поля `hide=1`. Форумы с таким значением поля `hide` являются скрытыми и не доступны пользователям.

Листинг 8.26. Добавление группы скрытых форумов в таблицу `forums`

```
mysql> INSERT INTO forums VALUES (0, 'Форум6', '', '', 5, 1),
    ->                           (0, 'Форум7', '', '', 4, 1),
    ->                           (0, 'Форум8', '', '', 3, 1),
    ->                           (0, 'Форум9', '', '', 2, 1),
    ->                           (0, 'Форум10', '', '', 1, 1);
```

Часто стоит задача группировки значений, которая осуществляется при помощи оператора `GROUP BY`, пример использования которого приведен в листинге 8.27.

Листинг 8.27. Пример использования оператора `GROUP BY`

```
mysql> SELECT * FROM forums GROUP BY hide;
```

```
+----------+--------+------+------+-----+------+
| id_forum | name   | rule | logo | pos | hide |
+----------+--------+------+------+-----+------+
|        1 | Форум1 |      |      |   5 |      |
|        6 | Форум6 |      |      |   5 | show |
+----------+--------+------+------+-----+------+
```

Рис. 8.12. Результат выполнения SQL-запроса из листинга 8.27

Результат выполнения SQL-запроса, приведенного в листинге 8.27, показан на рис. 8.12.

UPDATE

Оператор UPDATE обновляет столбцы таблицы *table* в соответствии с их новыми значениями в строках существующей таблицы.

```
UPDATE table
  SET col_name1=expr1 [, col_name2=expr2 ...]
  [WHERE definition]
  [LIMIT rows];
```

В выражении SET указывается, какие именно столбцы следует модифицировать и какие величины должны быть в них установлены. В выражении WHERE, если оно присутствует, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Ключевое слово LIMIT позволяет ограничить число обновляемых строк.

В листинге 8.28 для раздела форума с первичным ключом 2 задается новое название (PHP) и устанавливается атрибут hide равным 1, делая форум невидимым.

Листинг 8.28. Применение оператора UPDATE

```
UPDATE forums SET name='PHP', hide=1 WHERE id_forum=2;
```

SHOW

Оператор SHOW может принимать множество форм и предназначен для мониторинга таблиц, баз данных и сервера MySQL. При помощи оператора SHOW CHARACTER SET можно выяснить кодировки, поддерживаемые сервером MySQL (рис. 8.13).

Оператор SHOW COLUMN TYPES позволяет вывести перечень информации о типах столбцов, которые использовались при создании таблиц MySQL.

Замечание

Операторы SHOW CHARACTER SET и SHOW COLUMN TYPES поддерживаются, начиная с версии MySQL 4.1.

Так, вывести список всех столбцов выбранной таблицы можно при помощи запроса из листинга 8.29.

Charset	Description	Default collation	Maxlen
big5	Big5 Traditional Chinese	big5_chinese_ci	2
dec8	DEC West European	dec8_swedish_ci	1
cp850	DOS West European	cp850_general_ci	1
hp8	HP West European	hp8_english_ci	1
koi8r	KOI8-R Relcom Russian	koi8r_general_ci	1
latin1	ISO 8859-1 West European	latin1_swedish_ci	1
latin2	ISO 8859-2 Central European	latin2_general_ci	1
swe7	7bit Swedish	swe7_swedish_ci	1
ascii	US ASCII	ascii_general_ci	1
ujis	EUC-JP Japanese	ujis_japanese_ci	3
sjis	Shift-JIS Japanese	sjis_japanese_ci	2
hebrew	ISO 8859-8 Hebrew	hebrew_general_ci	1
tis620	TIS620 Thai	tis620_thai_ci	1
euckr	EUC-KR Korean	euckr_korean_ci	2
koi8u	KOI8-U Ukrainian	koi8u_general_ci	1
gb2312	GB2312 Simplified Chinese	gb2312_chinese_ci	2
greek	ISO 8859-7 Greek	greek_general_ci	1
cp1250	Windows Central European	cp1250_general_ci	1
gbk	GBK Simplified Chinese	gbk_chinese_ci	2
latin5	ISO 8859-9 Turkish	latin5_turkish_ci	1
armscii8	ARMSCII-8 Armenian	armscii8_general_ci	1
utf8	UTF-8 Unicode	utf8_general_ci	3
ucs2	UCS-2 Unicode	ucs2_general_ci	2
cp866	DOS Russian	cp866_general_ci	1
keybcs2	DOS Kamenicky Czech-Slovak	keybcs2_general_ci	1
macce	Mac Central European	macce_general_ci	1
macroman	Mac West European	macroman_general_ci	1
cp852	DOS Central European	cp852_general_ci	1
latin7	ISO 8859-13 Baltic	latin7_general_ci	1
cp1251	Windows Cyrillic	cp1251_general_ci	1
cp1256	Windows Arabic	cp1256_general_ci	1
cp1257	Windows Baltic	cp1257_general_ci	1
binary	Binary pseudo charset	binary	1
geostd8	GEOSTD8 Georgian	geostd8_general_ci	1

Рис. 8.13. Результат выполнения запроса `SHOW CHARACTER SET`

Листинг 8.29. Вывод структуры таблицы при помощи оператора `SHOW`

```
mysql> SHOW FIELDS FROM authors;
```

Результат приведен на рис. 8.14.

Field	Type	Null	Key	Default	Extra
id_author	int(11)		PRI	NULL	auto_increment
name	tinytext	YES		NULL	
new_test	text	YES		NULL	
passw	tinytext	YES		NULL	
email	tinytext	YES		NULL	
url	tinytext	YES		NULL	
icq	tinytext	YES		NULL	
about	tinytext	YES		NULL	
photo	tinytext	YES		NULL	
time	datetime	YES		NULL	
last_time	datetime	YES		NULL	
themes	int(11)	YES		NULL	
statususer	int(11)	YES		NULL	

Рис. 8.14. Результат выполнения запроса из листинга 8.29

Замечание

Получить результат, представленный на рис. 8.14, можно при помощи альтернативных операторов `DESCRIBE authors` и `SHOW COLUMNS FROM authors`.

Замечание

При помощи оператора `SHOW FULL COLUMNS FROM authors` можно получить расширенное описание структуры таблицы.

Отображение информации обо всех индексах конкретной таблицы легко получить с помощью листинга 8.30.

Листинг 8.30. Отображение информации об индексах

```
mysql> SHOW INDEX FROM authors;
```

Оператор `SHOW PROCESSLIST` позволяет получить информацию о потоках, выполняющихся на сервере (рис. 8.15).

```
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+
| Table   | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+
| authors |          0 | PRIMARY  |            1 | id_author   | A         |           3 |     NULL | NULL   |      |
+---------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+
```

Рис. 8.15. Список выполняемых запросов

Оператор `SHOW STATUS` позволяет получить значения переменных состояния сервера. Информацию о таблицах текущей базы данных можно получить при помощи оператора `SHOW TABLE STATUS`, выводящего многочисленные сведения: имя таблицы, тип, формат хранения строк, среднее число байтов, занимаемых таблицами, реальный размер файла данных таблицы, файла индекса, следующее значение для столбца с атрибутом `AUTO_INCREMENT` и т. п.

Как уже упоминалось, при помощи операторов `SHOW DATABASES` и `SHOW TABLES` можно просматривать список баз данных, размещенных на сервере, и список таблиц текущей базы данных.

Замечание

Программа mysqlshow, входящая в дистрибутив MySQL, позволяет осуществить просмотр информации, предоставляемой по запросу оператором `SHOW`. Так, просмотр списка баз данных можно осуществить, запустив в командной строке mysqlshow без параметров. Передав в качестве параметра имя базы данных (к примеру, `mysqlshow forum`), можно получить список таблиц этой базы данных.

Соответствие шаблону (*LIKE* и *NOT LIKE*)

СУБД MySQL позволяет осуществлять запросы, которые отвечают определенным шаблонам, т. е. не задавая точно всего искомого значения. Для этого используются специальные операторы — `LIKE` и `NOT LIKE`, которым передается строка с символами-заместителями. Символ `_` соответствует любому одному символу, аналогично символу "." в регулярных выражениях, а символ `%` соответствует любой последовательности символов или их отсутствию и является аналогом символа `*` в регулярных выражениях.

Так SQL-запрос

```
SELECT name, url FROM authors WHERE name LIKE 'S%';
```

к таблице `authors`, сформированной в листинге 8.4, приводит к выводу зарегистрированных посетителей форума, чье имя начинается с буквы `S` (рис. 8.16).

```
+--------+-----------------+
| name   | url             |
+--------+-----------------+
| Sergey | www.softtime.ru |
+--------+-----------------+
```

Рис. 8.16. Выборка по шаблону

Функция *COUNT*

Функция `COUNT` позволяет получить количество строк, выбранных запросом. Например, SQL-запрос, приведенный в листинге 8.31, позволяет выяснить общее количество посетителей форума.

Листинг 8.31. Общее количество строк в таблице `authors`

```
mysql> SELECT COUNT(*) FROM authors;
```

В качестве аргумента функции может выступать имя столбца таблицы. Запрос, приведенный в листинге 8.32, аналогичен по результату запросу из листинга 8.31.

Листинг 8.32. Альтернативный способ подсчета посетителей

```
mysql> SELECT COUNT(name) FROM authors;
```

Часто данные в таблицах дублируются. Получить выборку уникальных значений позволяет ключевое слово `DISTINCT`, которое помещается перед именем столбца. В листинге 8.33 приводится выборка URL пользователей. Несмотря

на то, что у некоторых пользователей URL могут совпадать, каждый из URL, удовлетворяющий запросу, выводится только по одному разу.

Листинг 8.33. Использование ключевого слова DISTINCT

```
mysql> SELECT DISTINCT url FROM authors;
```

Ключевое слово DISTINCT может использоваться совместно с функцией COUNT. Все зарегистрированные участники форума должны иметь уникальные имена, не совпадающие с именами других посетителей. Проверить выполнение этого требования можно при помощи SQL-запроса, приведенного в листинге 8.34.

Листинг 8.34. Проверка уникальности имен посетителей форума

```
mysql> SELECT COUNT(DISTINCT name) FROM authors;
```

Если полученный результат не совпадает с результатом работы листинга 8.32, предъявленное выше требование не соблюдается.

Наиболее продуктивно функция COUNT используется совместно с оператором GROUP BY. Так подсчет количества скрытых и доступных форумов можно осуществить одним SQL-запросом, приведенным в листинге 8.35.

Листинг 8.35. Совместное использование COUNT и GROUP BY

```
mysql> SELECT hide, COUNT(*) FROM forums GROUP BY hide;
```

Результат работы запроса представлен на рис. 8.17, где показано, что в базе данных имеются 5 скрытых и 5 открытых форумов.

```
+------+----------+
| hide | COUNT(*) |
+------+----------+
|    0 |        5 |
|    1 |        5 |
+------+----------+
```

Рис. 8.17. Результат выполнения SQL-запроса из листинга 8.35

Соединение с базой данных — подводные камни

Соединение с базой данных MySQL осуществляется при помощи функции mysql_connect(), которая имеет следующий синтаксис:

```
resource mysql_connect([string server [,
                       string username [,
                       string password [,
                       bool new_link [,
                       int client_flags]]]]])
```

Эта функция устанавливает соединение с сервером MySQL, сетевой адрес которого задается параметром *server*. Вторым и третьим аргументами этой функции являются имя пользователя базы данных *username* и его пароль *password* соответственно.

По умолчанию повторный вызов функции `mysql_connect()` с теми же аргументами не приводит к установлению нового соединения, вместо этого функция возвращает дескриптор уже существующего соединения. Если четвертому параметру *new_link* присвоить значение `true`, будет открыто новое соединение с сервером.

Параметр *client_flags* должен быть комбинацией из следующих констант:

- `MYSQL_CLIENT_COMPRESS` — предписывает использование протокола сжатия при обмене информации между сервером и клиентом;
- `MYSQL_CLIENT_IGNORE_SPACE` — в SQL-запросах после имен функций разрешается использование пробелов;
- `MYSQL_CLIENT_INTERACTIVE` — ждать `interactive_timeout` секунд до закрытия соединения, если между сервером и клиентом не происходит обмена данными.

Замечание

Все аргументы функции являются необязательными. В случае их отсутствия, по умолчанию, для этой функции устанавливаются следующие параметры: `server = 'localhost:3306'`, `username` принимает значение владельца процесса сервера, а `password` — пустую строку.

В случае успеха функция возвращает дескриптор соединения с сервером, при неудаче возвращает значение `false`.

Пример — в листинге 8.36.

Листинг 8.36. Функция `mysql_connect()`

```
<?php
  $dblocation = "localhost"; // Имя сервера
  $dbuser = "root";          // Имя пользователя
  $dbpasswd = "";            // Пароль
  $dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
```

```
  if (!$dbcnx)     // Если дескриптор равен 0, соединение не установлено
  {
    exit("<P>В настоящий момент сервер базы данных недоступен, поэтому
          корректное отображение страницы невозможно.</P>");
  }
  else
  {
    echo("<P>Соединение установлено.</P>");
  }
?>
```

Замечание

Для подавления вывода сообщений об ошибках, генерируемых PHP в окно браузера, в листинге 8.36 перед функцией `mysql_connect` размещен символ `@`.

Переменные `$dblocation`, `$dbuser` и `$dbpasswd` хранят имя сервера, имя пользователя и пароль и, как правило, прописываются в отдельном файле (к примеру, config.php), который потом вставляется в каждый PHP-файл, содержащий код для работы с MySQL.

Если код из листинга 8.36 не срабатывает, это может означать, что сервер базы данных не запущен. Если вы работаете в среде операционной системы Windows, запустите диспетчер задач и убедитесь в наличии процесса сервера MySQL — mysqld-nt.exe. Если такой процесс отсутствует, следует запустить утилиту WinMySQLAdmin, которая входит в состав дистрибутива (/mysql/bin/winmysqladmin.exe). После запуска WinMySQLAdmin в системном триере появляется значок в виде светофора, контекстное меню которого позволяет запускать и останавливать сервер MySQL, а сигнал "светофора" отображает состояние сервера: зеленый цвет, если сервер запущен, и красный, если приостановлен (рис. 8.18).

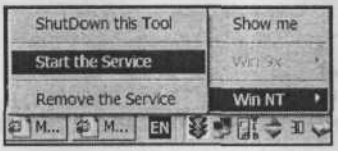


Рис. 8.18. Контекстное меню утилиты WinMySQLAdmin

При работе в среде Windows NT/2000/XP можно установить MySQL в качестве сервиса, что обеспечит запуск сервера `mysqld` при старте системы. Для этого в контекстном меню WinMySQLAdmin необходимо выбрать пункт **Install the Service**.

Замечание

Процесс установки связки "PHP+Apache+MySQL" описывается в статье по ссылке **http://www.softtime.ru/info/articlephp.php?id_article=24**.

Замечание

При переустановке базы данных MySQL сервер всегда следует останавливать, т. к. в противном случае можно сбить сервис запуска, и занятный процессом файл mysql-nt.exe может не установиться.

Если сервер запущен, но в окне браузера при обращении к любым функциям PHP для работы с MySQL наблюдается чистое окно без малейшего признака сообщений об успешности или невозможности соединения сервера, это означает, что связка "MySQL+PHP" настроена некорректно. Дело в том, что, начиная с версии PHP 5, все расширения в PHP, в том числе и расширение для работы с базой данных MySQL, по умолчанию отключены. Для того чтобы подключить расширение для работы с базой данных, следует в конфигурационном файле PHP php.ini убрать комментарий (;) с соответствующего расширения:

```
extension=php_mysql.dll
```

Замечание

Изменения в конфигурационном файле php.ini вступают в силу после перезапуска Web-сервера.

Если PHP не может найти библиотеку расширения, то при любом обращении к функциям MySQL будет выдано предупреждение, представленное на рис. 8.19.

Самым простым решением этой проблемы будет копирование файла php_mysql.dll из каталога php/ext в системный каталог C:/Windows/system32 или в корневой каталог PHP. Другим способом решения проблемы будет указание в директиве `extension_dir` пути к каталогу php/ext:

```
extension_dir = "C:/PHP/ext"
```

Замечание

PHP очень плохо реагирует на обратные слеши (\), принятые для указания пути в Windows. Для разделения каталогов следует использовать прямые слеши (/).

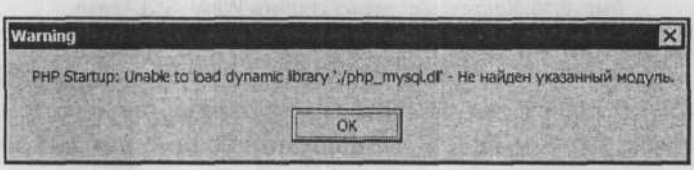


Рис. 8.19. Невозможно загрузить динамическую библиотеку

Если предупреждение о невозможности загрузить библиотеку php_mysql.dll по-прежнему выводится, следует скопировать библиотеки libmysql.dll в системный каталог C:/Windows/system32.

Замечание

Дополнительные динамические библиотеки требуются для многих расширений PHP. Ознакомиться со списком этих расширений и требуемых ими библиотек можно в файле snapshot.txt, расположенном в корневом каталоге PHP.

После того как соединение установлено, необходимо выбрать базу данных. Это осуществляется при помощи функции `mysql_select_db()`, которая имеет следующий синтаксис:

```
bool mysql_select_db(string database_name [, resource link_identifier])
```

Использование этой функции эквивалентно вызову команды `USE` в SQL-запросе, т. е. функция `mysql_select_db()` выбирает базу данных для дальнейшей работы, и все последующие SQL-запросы применяются к выбранной базе данных. Функция принимает в качестве аргументов название выбираемой базы данных *database_name* и дескриптор соединения *link_identifier*. Функция возвращает `true` при успешном выполнении операции и `false` в противном случае.

Имеет смысл помещать функции для соединения и выбора базы данных в тот же файл (config.php), где объявлены переменные с именами сервера, пользователя и паролем (листинг 8.37).

Листинг 8.37. Файл config.php

```
<?php
  $dblocation = "localhost";
  $dbname = "test";
  $dbuser = "root";
  $dbpasswd = "";
  $dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
  if (!$dbcnx)
  {
    exit("<P>В настоящий момент сервер базы данных недоступен, поэтому
          корректное отображение страницы невозможно.</P>");
  }
  if (!@mysql_select_db($dbname, $dbcnx))
  {
    exit("<P>В настоящий момент база данных недоступна, поэтому
          корректное отображение страницы невозможно.</P>");
  }
?>
```

При использовании в скрипте функций MySQL достаточно включить в начале скрипта файл config.php и соединение с базой данных будет установлено:

```
include "config.php";
```

Закрытие соединения

Закрытие соединения осуществляется при помощи функции `mysql_close()`. Ее синтаксис таков:

```
bool mysql_close([resource link_identifier])
```

В качестве необязательного параметра функция принимает дескриптор открытого соединения *link_identifier*. Если этот параметр не указан, закрывается последнее открытое соединение. Функция возвращает `true` в случае успеха и `false` при возникновении ошибки.

В листинге 8.38 приведен пример использования функции.

Листинг 8.38. Функция `mysql_close()`

```
<?php
  // Устанавливаем соединение с базой данных
  $dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
  if (!$dbcnx)
  {
    // Выводим предупреждение
    exit ("<P>В настоящий момент сервер базы данных недоступен, поэтому
           корректное отображение страницы невозможно.</P>");
  }
  if(mysql_close($dbcnx))    // Разрываем соединение
  {
    echo("Соединение с базой данных прекращено");
  }
  else
  {
    echo("Не удалось завершить соединение");
  }
?>
```

Выполнение запросов

Выполнение SQL-запросов осуществляется при помощи функции `mysql_query()`, которая имеет следующий синтаксис

```
resource mysql_query(string query[, resource link_identifier])
```

Первый аргумент функции представляет собой строку с запросом *query*, второй (*link_identifier*) — дескриптор соединения, возвращаемый функцией mysql_connect().

Замечание

Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции mysql_connect() без параметров.

Замечание

При передаче запроса функции mysql_query() точку с запятой в конце запроса, обязательную при работе с клиентом mysql, можно не ставить.

Функция возвращает дескриптор запроса в случае успеха и false в случае неудачного выполнения запроса. В листинге 8.39 представлен скрипт, создающий новую таблицу author.

Листинг 8.39. Функция mysql_query()

```
<?php
  include "config.php";
  if(mysql_query("CREATE TABLE author VALUES(INT id_author, TEXT name)"))
  {
     echo "Таблица создана успешно";
  }
  else
  {
     exit(mysql_erorr());
  }
?>
```

В листинге 8.39 при неудачном выполнении функции exit(), останавливающей работу скрипта, в качестве параметра передается значение ошибки, возвращаемой функцией mysql_erorr().

Как осуществить выборку?

Дескриптор, возвращаемый функцией mysql_query(), используется далее для получения значений, возвращаемых СУБД. Обычно это осуществляется при помощи одной из пяти функций: mysql_result(), mysql_fetch_row(), mysql_fetch_assoc(), mysql_fetch_array() и mysql_fetch_object().

Пусть имеется таблица пользователей authors, содержащая три поля: первичный ключ (id_author), имя пользователя (name) и его пароль (passw). SQL-запрос, создающий эту таблицу, приведен в листинге 8.40.

Листинг 8.40. Создание таблицы authors

```
mysql> CREATE TABLE authors (
  id_author INT(6) NOT NULL AUTO_INCREMENT,
  name TEXT,
  passw TEXT,
  PRIMARY KEY (id_author)
) TYPE=MyISAM;
```

Первая функция `mysql_result()` возвращает результат запроса, выполненного функцией `mysql_query()`. С ее помощью можно получить доступ к отдельному полю записи. Синтаксис функции таков:

```
mixed mysql_result(resource result, int row [, mixed field])
```

В качестве первого аргумента *result* функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Второй аргумент *row* представляет собой номер столбца, который необходимо вернуть. Третий необязательный параметр *field* — это имя поля таблицы.

Вывести имя автора, который первым найдется в базе данных, можно при помощи скрипта из листинга 8.41.

Листинг 8.41. Поиск первой записи с именем автора

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Выполняем SQL-запрос
  $ath = mysql_query("SELECT name FROM authors");
  // Обрабатываем результаты запроса
  if($ath) echo mysql_result($ath,0,'name');
  else exit(mysql_error());
?>
```

Функция `mysql_fetch_row()` обрабатывает результаты запроса, выполненного функцией `mysql_query()`, и возвращает неассоциативный массив. Синтаксис функции следующий:

```
array mysql_fetch_row(resource result)
```

В качестве единственного аргумента *result* функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Возвращает массив, содержащий данные обработанного ряда, или `false`, если рядов больше нет. При работе с данным массивом удобно использовать функцию `list()`, преобразующую элементы массива в переменные (листинг 8.42).

Листинг 8.42. Применение функции mysql_fetch_row()

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Выполняем SQL-запрос
  $ath = mysql_query("SELECT * FROM authors");
  // Проверяем успешность выполнения SQL-запроса
  if(!$ath) exit(mysql_erorr());
  // Определяем таблицу и заголовок
  // Так как запрос может возвращать несколько строк, применяем цикл
  echo "<table>";
  while(list($id_author, $name, $password) = mysql_fetch_row($ath))
  {
    echo "<tr>
            <td>$id_author</td>
            <td>$name</td>
            <td>$password</td>
          </tr>";
  }
  echo "</table>";
?>
```

Функция `mysql_fetch_assoc()` обрабатывает результаты запроса и возвращает ассоциативный массив. Синтаксис функции:

```
array mysql_fetch_assoc(resource result)
```

В качестве единственного аргумента *result* функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Возвращает массив, содержащий данные обработанного ряда, или `false`, если рядов больше нет. Пример использования `mysql_fetch_assoc()` приведен в листинге 8.43.

Листинг 8.43. Использование функции mysql_fetch_assoc()

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Выполняем SQL-запрос
  $ath = mysql_query("select * from forums");
  // Проверяем успешность выполнения SQL-запроса
  if(!$ath) exit(mysql_error());
  // Определяем таблицу и заголовок
  echo "<table>";
```

```
  while($author = mysql_fetch_assoc($ath))
  {
    echo "<tr>
            <td>".$author['id_author']."</td>
            <td>".$author['name']."</td>
            <td>".$author['password']."</td>
          </tr>";
  }
  echo "</table>";
?>
```

Функция `mysql_fetch_array()` возвращает результаты запроса, выполненного функцией `mysql_query()`, в виде массива. Причем тип массива (численный или ассоциативный) может быть задан. Синтаксис данной функции:

```
array mysql_fetch_array(resource result [, int result_type])
```

В качестве первого аргумента *result* функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Второй необязательный параметр может принимать три значения:

- `MYSQL_ASSOC` — возврат результата работы в виде ассоциативного массива;
- `MYSQL_NUM` — возврат результата работы в виде численного массива;
- `MYSQL_BOTH` — возврат результата работы в виде массива, содержащего как численные, так и ассоциативные индексы.

Замечание

По умолчанию второй аргумент принимает значение `MYSQL_BOTH`.

Замечание

Режимы работы функции `mysql_fetch_array()`, принимающей в качестве второго аргумента константы `MYSQL_ASSOC` и `MYSQL_NUM`, аналогичны функциям `mysql_fetch_assoc()` и `mysql_fetch_row()` соответственно.

При возврате ассоциативного массива в качестве индексов выступают имена столбцов таблицы, из которой производится выборка. Вот как с помощью этой функции можно вывести все строки таблицы `authors` (листинг 8.44).

Листинг 8.44. Извлечение всех строк из таблицы

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
```

```
  // Выполняем SQL-запрос
  $ath = mysql_query("SELECT * FROM authors");
  // Проверяем успешность выполнения SQL-запроса
  if(!$ath) exit(mysql_error());
  // Определяем таблицу и заголовок
  echo "<table>";
  // Так как запрос возвращает несколько строк, применяем цикл
  while($author = mysql_fetch_array($ath))
  {
    echo "<tr>
            <td>".$author['id_author']."</td>
            <td>".$author['name']."</td>
            <td>".$author['password']."</td>
          </tr>";
  }
  echo "</table>";
?>
```

Функция `mysql_fetch_object()` обрабатывает результат запроса функции `mysql_query()` и возвращает поля таблицы в виде объекта, имена членов которого совпадают с именами полей таблицы. Синтаксис функции:

```
object mysql_fetch_object(resource result)
```

В качестве единственного аргумента *result* функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`. Возвращает объект со свойствами, соответствующими колонкам в обработанном ряду, или `false`, если рядов больше нет.

Замечание

Имена полей, возвращаемых этой функцией, не зависят от регистра.

В листинге 8.45 приведен пример, в котором с помощью данной функции из таблицы `authors` выводятся имя, URL и e-mail авторов.

Листинг 8.45. Вывод некоторых полей таблицы `authors`

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Выполняем SQL-запрос
  $ath = mysql_query("SELECT * FROM authors");
  // Проверяем успешность выполнения SQL-запроса
  if(!$ath) exit(mysql_error());
```

```
  while($row = mysql_fetch_object($ath))
  {
    echo "<p>name: ".$row->name."</p>";
    echo "<p>url: ".$row->url."</p>";
    echo "<p>email: ".$row->email."</p>";
  }
?>
```

Сколько строк в выборке?

Одной из распространенных задач при выборке из базы данных является определение числа записей, которые возвращает функция `mysql_query()`. Для этого предназначена функция `mysql_num_rows()`, которая имеет следующий синтаксис:

```
int mysql_num_rows(resource result)
```

В качестве единственного аргумента *result* функция принимает дескриптор запроса, возвращаемый функцией `mysql_query()`.

Замечание

Эта функция работает только с запросами `SELECT`. Чтобы получить количество рядов, обработанных командами `INSERT`, `UPDATE`, `DELETE`, следует использовать функцию `mysql_affected_rows()`.

Замечание

Функция `mysql_num_fields()` позволяет определить число столбцов в результате.

Функция `mysql_query()` в случае успеха всегда возвращает дескриптор соединения и `flase` только в случае ошибочного синтаксиса SQL-запроса, поэтому проверка соответствия пароля в листинге 8.46 ошибочна, т. к. такая проверка будет срабатывать в любом случае, даже если число возвращенных записей окажется равным нулю.

Листинг 8.46. Пример ошибочной проверки пароля

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Формируем SQL-запрос
  $query = "SELECT * FROM authors
            WHERE name = 'имя' AND pass = 'пароль'";
```

```
  // Выполняем SQL-запрос
  $ath = mysql_query($query);
  // Проверяем успешность выполнения SQL-запроса
  if(!$ath)
  {
     exit("Ошибка авторизации!")
  }
  else
  {
     echo("Авторизация пройдена!")
  }
?>
```

В этом случае следует использовать функцию `mysql_num_rows()`, как это продемонстрировано в листинге 8.47.

Листинг 8.47. Пример корректной проверки пароля

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Формируем SQL-запрос
  $query = "SELECT * FROM authors
            WHERE name = 'имя' AND pass = 'пароль'";
  // Выполняем SQL-запрос
  $ath = mysql_query($query);
  // Проверяем успешность выполнения SQL-запроса
  if(mysql_num_rows($ath) == 0)
  {
     exit("Ошибка авторизации!")
  }
  else
  {
     echo("Авторизация пройдена!")
  }
?>
```

Число строк можно так же определить при помощи отдельного SQL-запроса, воспользовавшись встроенной функцией MySQL — `COUNT()`:

```
SELECT COUNT(*) FROM authors;
```

Подробно о транзакциях

Транзакция — это последовательность операторов SQL, выполняющихся как единая операция, которая не прерывается другими клиентами, т. е. пока происходит работа с записями таблицы (их обновление или удаление), никто другой не может получить доступ к этим записям, поскольку СУБД MySQL автоматически блокирует доступ к ним.

Замечание

Таблицы ISAM, MyISAM и HEAP не поддерживают транзакции. В настоящий момент их поддержка осуществляется только в таблицах BDB и InnoDB.

Транзакции позволяют объединять операторы в группу и гарантировать, что все операции внутри группы будут выполнены успешно. Если часть транзакции выполняется со сбоем, результаты работы всех операторов транзакции до места сбоя отменяются, приводя базу данных к виду, в котором она была до выполнения транзакции.

Системы, поддерживающие возможности транзакций, часто еще характеризуют как обеспечивающие свойства ACID (от англ. *Atomic*, *Consistent*, *Isolated*, *Durable* — атомарный, целостный, изолированный, длительный).

- *Атомарность*: операторы транзакции формируют единый логический блок, каждый элемент которого невозможно выполнить без обработки всех остальных элементов блока.
- *Целостность*: база данных является целостной до и после выполнения транзакции.
- *Изоляция*: отдельные транзакции не влияют на работу друг друга.
- *Длительность*: при выполнении транзакции все ее результаты сохраняются в базе данных.

Для выяснения механизма транзакций рассмотрим ситуацию, которая возможна при работе интернет-магазина, торгующего книгами. Предположим, что на складе имеется 100 экземпляров книги. При этом в один момент времени оформление покупки книги начинают два покупателя.

Первый покупатель оформляет покупку книги, при этом происходят два события: регистрируется факт покупки и обновляется количество книг, существующих на складе. Регистрация покупки разрешена, если количество книг, возвращаемое оператором `SELECT`, больше нуля. Далее из этого значения вычитается количество заказываемых покупателем книг и происходит обновление базы данных склада. Первый раз программное обеспечение по запросу `SELECT` получает количество книг на складе, равное 100.

Одновременно второй покупатель начинает оформление покупки. Запрос на выборку количества книг для него так же возвращает значение 100.

После регистрации покупки книги первым покупателем число книг, оставшихся на складе, обновляется, и 100 заменяется значением 99.

После регистрации покупки книги вторым посетителем интернет-магазина так же происходит обновление базы данных склада при помощи оператора UPDATE, но поскольку поток, в котором выполняется программа обработки покупки, ничего не знает о параллельно выполняющемся другом запросе, то в результате двух покупок остаток на складе составит 99 книг, вместо 98.

Для предотвращения такой ситуации выполнение запроса для подсчета остатка на складе при помощи оператора SELECT и обновление данного значения в базе данных при помощи оператора UPDATE должно выполняться как одна транзакция.

Другая ситуация, связанная с выполнением группы операций, заключается в том, что при возникновении ошибки в середине группы база данных останется в модифицированном состоянии лишь наполовину.

Если при оплате покупки происходит перевод денежной суммы со счета клиента на счет интернет-магазина, то счет клиента должен уменьшиться на сумму sum, а счет интернет-магазина увеличится на ту же сумму:

```
UPDATE account SET balance = balance - sum WHERE name = 'client';
UPDATE account SET balance = balance + sum WHERE name = 'bookshop';
```

Если в момент завершения первого запроса и перед самым выполнением второго произойдет аварийный сбой, транзакция окажется незавершенной.

Замечание

Ряд ситуаций можно обработать без привлечения механизма транзакций. Например, блокируя таблицу, с последующей разблокировкой при помощи операторов LOCK TABLES и UNLOCK TABLES. Кроме того, в ситуации с интернет-магазином можно воспользоваться так называемым *относительным изменением*, используя запрос UPDATE store SET count = count - 1;.

По умолчанию СУБД MySQL работает в режиме auto-commit (автовыполнения), т. е. каждый отдельный оператор рассматривается как отдельная транзакция и выполняется немедленно. Для того чтобы выполнять транзакции явным образом, необходимо отключение режима автовыполнения, которое осуществляется заданием оператора BEGIN. Завершение транзакции отмечается выполнением оператора COMMIT. Если во время операции происходит сбой, ее результаты можно отменить с помощью оператора ROLLBACK.

Замечание

После выполнения одного из двух операторов COMMIT или ROLLBACK режим восстанавливается таким, каким он был до выполнения оператора BEGIN.

Замечание

Существует еще один способ выполнения транзакций — это установка и сброс режима автовыполнения при помощи операторов `SET AUTOCOMMIT = 1` и `SET AUTOCOMMIT = 0`.

Для демонстрации работы транзакций создадим таблицу `forums` из листинга 8.18, назначив ей тип InnoDB (листинг 8.48).

Листинг 8.48. Создание таблицы `forums`

```
mysql> CREATE TABLE forums (
    -> id_forum INT(6) NOT NULL AUTO_INCREMENT,
    -> name TINYTEXT,
    -> rule TEXT,
    -> logo TEXT,
    -> pos INT DEFAULT NULL,
    -> hide TINYINT(1) DEFAULT NULL,
    -> PRIMARY KEY (id_forum)
    ->) TYPE=InnoDB;
```

Обозначим начало транзакции оператором `BEGIN`, добавим в таблицу две записи и подтвердим выполнение транзакции оператором `COMMIT`. После чего произведем выборку значений из таблицы (листинг 8.49).

Листинг 8.49. Выполнение транзакции

```
mysql> BEGIN;
    -> INSERT INTO forums SET name = 'Форум I', hide = 0, pos = 1;
    -> INSERT INTO forums SET name = 'Форум II', hide = 0, pos = 2;
    -> COMMIT;
    -> SELECT id_forum, name FROM forums;
```

Результат выполнения группы SQL-запросов из листинга 8.49 представлен на рис. 8.20.

```
+----------+----------+
| id_forum | name     |
+----------+----------+
|        1 | Форум I  |
|        2 | Форум II |
+----------+----------+
```

Рис. 8.20. Результат работы SQL-запросов из листинга 8.49

Если запустить второй экземпляр клиента mysql, выборка до выполнения оператора `COMMIT` возвратит пустое множество.

Если во время отработки транзакции происходит сбой, ее работу можно отменить посредством оператора ROLLBACK (листинг 8.50).

Листинг 8.50. Отмена транзакции

```
mysql> BEGIN;
    -> INSERT INTO forums SET name = 'Форум III', hide = 0, pos = 1;
    -> INSERT INTO forums SET id_forum = 1, name = 'Форум IV', hide = 0;
ERROR 1062 (23000): Duplicate entry '1' for key 1
    -> ROLLBACK;
    -> SELECT id_forum, name FROM forums;
```

Как видно из рис. 8.21, успешно размещенная запись "Форум III" была удалена из базы данных forums после выполнения оператора ROLLBACK, производящего откат таблицы к исходному состоянию.

```
+----------+----------+
| id_forum | name     |
+----------+----------+
|        1 | Форум I  |
|        2 | Форум II |
+----------+----------+
```

Рис. 8.21. Результат работы SQL-запросов из листинга 8.50

Проверка результатов запроса на значение *NULL*

Определить, возвращает ли оператор SELECT значение NULL, можно при помощи функции isset().

Замечание

PHP интерпретирует значение NULL как установленное значение.

Пример проверки на значения NULL для выборки из таблицы test, содержащей три столбца (first, second и third), приведен в листинге 8.51.

Листинг 8.51. Проверка на равенство значению NULL

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT first, second, third FROM test";
  $tst = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if($tst)
```

```
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Выводим заголовок таблицы
  echo "<table>";
  while(list($first, $second, $third) = mysql_fetch_row($tst))
  {
      echo "<tr>";
      if(!isset($first)) echo "<td>-</td>";
      else echo "<td>$first</td>";
      if(!isset($second)) echo "<td>-</td>";
      else echo "<td>$second</td>";
      if(!isset($third)) echo "<td>-</td>";
      else echo "<td>$third</td>";
      echo "</tr>";
  }
  // Выводим тег завершения таблицы
  echo "</table>";
?>
```

Замечание

Использование функции `empty()` в данном случае будет не точным, т. к. она возвращает `true` и для значения `NULL`, и для пустой строки (`""`).

Аналогично функции `isset()`, на равенство пустому значению можно проверять при помощи оператора идентичности `===` и константы `NULL`, как это продемонстрировано в листинге 8.52.

Листинг 8.52. Использование оператора `===` и константы `NULL`

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT first, second, third FROM test";
  $tst = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if($tst)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Выводим заголовок таблицы
  echo "<table>";
```

```
  while(list($first, $second, $third) = mysql_fetch_row($tst))
  {
      echo "<tr>";
      if($first === NULL) echo "<td>-</td>";
      else echo "<td>$first</td>";
      if($second === NULL) echo "<td>-</td>";
      else echo "<td>$second</td>";
      if($third === NULL) echo "<td>-</td>";
      else echo "<td>$third</td>";
      echo "</tr>";
  }
  // Выводим тег завершения таблицы
  echo "</table>";
?>
```

Близко к рассмотренной лежит задача определения пустых выборок из таблиц. Так код в листинге 8.52 выведет таблицу с единственной строкой, содержащей три прочерка, если в таблице нет ни одной записи. В таком случае, логично было бы вообще не выводить строки из таблицы. Решить эту проблему можно при помощи уже упоминавшейся функции `mysql_num_rows()`, возвращающей число строк в результатах запроса (листинг 8.53).

Листинг 8.53. Определение количества строк в результате выполнения запроса

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT first, second, third FROM test";
  $tst = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$tst)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($tst)>0)
  {
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($first, $second, $third) = mysql_fetch_row($tst))
    {
      echo "<tr>";
```

```
      if($first === NULL) echo "<td>-</td>";
      else echo "<td>$first</td>";
      if($second === NULL) echo "<td>-</td>";
      else echo "<td>$second</td>";
      if($third === NULL) echo "<td>-</td>";
      else echo "<td>$third</td>";
      echo "</tr>";
    }
    // Выводим тег завершения таблицы
    echo "</table>";
  }
?>
```

Избежание повторных запросов

Пусть имеется таблица `members`, содержащая имена (`name`), электронные адреса (`email`) и домашние странички (`url`) членов виртуального клуба. При этом поле `name` является обязательным, а поля `email` и `url` — опциональными и, в случае отсутствия у посетителя `email` или `url`, принимающими значения `NULL`. Оператор `CREATE`, создающий данную таблицу, представлен в листинге 8.54.

Листинг 8.54. Таблица `members`

```
CREATE TABLE members (
  name VARCHAR(20) NOT NULL DEFAULT '',
  email TINYTEXT,
  url TINYTEXT
) TYPE=MyISAM;
```

Одна из задач, стоящая перед администратором клуба, состоит в выводе двух списков: членов клуба, имеющих e-mail, и членов клуба, имеющих свои виртуальные странички. Один из вариантов скрипта, выполняющего эту задачу, приведен в листинге 8.55.

Листинг 8.55. Вывод списков разных членов клуба *(вариант 1)*

```
<?php
  // I таблица
  // Формируем и выполняем SQL-запрос
  $query = "SELECT name, email FROM members";
  $mem = mysql_query($query);
```

```
  // Проверяем успешность выполнения запроса
  if(!$mem)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($mem)>0)
  {
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($name, $email) = mysql_fetch_row($mem))
      if($email !== NULL) echo "<tr><td>$name</td><td>$email</td></tr>";
    // Выводим завершение таблицы
    echo "</table>";
  }

  // II таблица
  // Формируем и выполняем SQL-запрос
  $query = "SELECT name, url FROM members";
  $mem = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$mem)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($mem)>0)
  {
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($name, $url) = mysql_fetch_row($url))
      if($email !== NULL) echo "<tr><td>$name</td><td>$url</td></tr>";
    // Выводим завершение таблицы
    echo "</table>";
  }
?>
```

Из листинга 8.55 видно, что два раза подряд выполняется практически одинаковый код. Первое, что приходит в голову, — поместить после оператора `SELECT` первого SQL-запроса поле `url` и использовать дескриптор запроса повторно, без выполнения второго SQL-запроса. Однако повторно использовать

дескриптор `$mem` без дополнительного кода не удастся, т. к. внутренний указатель находится на последнем элементе. Решить проблему может функция `mysql_data_seek()`, способная перемещать внутренний указатель и имеющая следующий синтаксис:

```
bool mysql_data_seek(resource result_identifier, int row_number)
```

Функция принимает два параметра, первый из которых *result_identifier* — дескриптор запроса, возвращаемый функцией `mysql_query()`, второй — смещение внутреннего указателя относительно первой позиции. Функция возвращает `true` в случае успеха и `false` в случае неудачи.

С использованием этой функции код из листинга 8.55 можно переписать следующим образом (листинг 8.56).

Листинг 8.56. Вывод списков разных членов клуба *(вариант 2)*

```
<?php
  // Формируем и выполняем SQL-запрос
  $query = "SELECT name, email, url FROM members";
  $mem = mysql_query($query);
  // Проверяем успешность выполнения запроса
  if(!$mem)
  {
    echo "<p><b>Error: ".mysql_error()."</b><p>";
    exit();
  }
  // Проверяем, содержит ли результат хоть одну строку
  if(mysql_num_rows($mem)>0)
  {
    // I таблица
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($name, $email, $url) = mysql_fetch_row($mem))
      if($email !== NULL) echo "<tr><td>$name</td><td>$email</td></tr>";
    // Выводим завершение таблицы
    echo "</table>";
    // Устанавливаем внутренний указатель в начало
    mysql_data_seek($mem,0);
    // II таблица
    // Выводим заголовок таблицы
    echo "<table>";
    while(list($name, $email, $url) = mysql_fetch_row($mem))
      if($email !== NULL) echo "<tr><td>$name</td><td>$url</td></tr>";
```

```
    // Выводим завершение таблицы
    echo "</table>";
}
?>
```

Как видно, для вывода обеих таблиц оказалось достаточно одного SQL-запроса.

Полнотекстовый поиск

Полнотекстовый поиск в СУБД MySQL на сегодняшний день поддерживается только для таблиц типа MyISAM и только для текстовых столбцов переменной длины (TEXT и VARCHAR). Эта возможность активируется созданием индекса типа FULLTEXT для столбцов, по которым необходимо осуществлять поиск.

Замечание

Полнотекстовый поиск в СУБД MySQL не чувствителен к регистру. Кроме того, при поиске игнорируются так называемые "общеупотребительные" слова. К ним относятся слишком короткие слова (по умолчанию состоящие меньше, чем из четырех символов), а также слова, встречающиеся, по крайней мере, в половине записей. Так если в таблице имеются только две записи, то поиск не даст результатов, поскольку каждое слово будет присутствовать в половине записей.

Для использования возможности полнотекстового поиска в таблице, содержащей текстовые поля, необходимо создать индексы FULLTEXT. Изменить уже существующие таблицы можно при помощи SQL-оператора ALTER. В листинге 8.57 для текстового поля name таблицы tbl создается индекс FULLTEXT.

Листинг 8.57. SQL-оператор ALTER для добавления индекса FULLTEXT для поля name

```
ALTER TABLE tbl ADD FULLTEXT name (name);
```

В листинге 8.57 сразу после ключевого слова FULLTEXT следует имя создаваемого индекса, за которым в скобках указано имя текстового столбца.

Индекс FULLTEXT можно создать сразу по нескольким столбцам. Для этого в скобках после имени индекса нужно указать имена двух столбцов через запятую:

```
ALTER TABLE tbl ADD FULLTEXT name (name, description);
```

Замечание

Если индекс создается только по одному столбцу и его имя совпадает с именем столбца, то имя индекса, размещаемое после ключевого слова FULLTEXT, можно опустить.

На количество индексов в таблице ограничения также не накладываются, допустимо создание нескольких индексов:

```
ALTER TABLE tbl ADD FULLTEXT (name),
                ADD FULLTEXT (description),
                ADD FULLTEXT together (name, description);
```

Поиск выполняется при помощи встроенной функции MATCH(), в качестве аргумента которой выступает один или более индексов типа FULLTEXT, а также функции AGAINST(), которая принимает в качестве аргумента искомую строку (листинг 8.58).

Листинг 8.58. Оператор SELECT, осуществляющий полнотекстовый поиск

```
SELECT * FROM tbl WHERE MATCH(name) AGAINST('программирование');
```

В листинге 8.58 из таблицы tbl извлекаются все записи, соответствующие статьям, в названии которых встречается слово "программирование". Синтаксис допускает использование нескольких конструкций MATCH() — AGAINST() после оператора WHERE, разделенных логическими операторами OR или AND (листинг 8.59).

Листинг 8.59. Использование нескольких конструкций MATCH() — AGAINST()

```
SELECT * FROM tbl WHERE MATCH(name) AGAINST('программирование')
                     OR MATCH(name) AGAINST('кодирование');
```

В листинге 8.59 из таблицы tbl извлекаются все записи, соответствующие статьям, в теле которых присутствует либо слово "программирование", либо слово "кодирование".

Замечание

Результаты поиска сортируются в порядке уменьшения релевантности. Величина релевантности представляет собой неотрицательное число с плавающей точкой. Релевантность вычисляется на основе количества слов в данной строке столбца, количества уникальных слов в этой строке, общего количества слов в тексте и числа документов (строк), содержащих отдельное слово.

Замечание

Начиная с версии 4.0.1, СУБД MySQL поддерживает поиск в так называемом логическом режиме, позволяющем задать обязательное отсутствие слова в результатах запроса, изменять величину релевантности для отдельных слов и многое другое. За более подробной информацией следует обратиться к официальному справочному руководству по СУБД MySQL.

Временные таблицы

База данных MySQL допускает создание врéменных таблиц, предназначенных для хранения IP-адресов посетителей и состоящих из четырех полей:

- `id_ip` — первичный ключ таблицы;
- `ip` — IP-адрес посетителя;
- `id_user` — вторичный ключ, устанавливающий дополнительную связь с таблицей посетителей `users`. В таблице `users` этот ключ выступает в качестве первичного;
- `puttime` — время посещения.

Необходимо решить следующую задачу — выбрать последний IP-адрес для каждого посетителя и время, когда он с него зашел. Данную задачу можно решить в два этапа. Сначала выбрать вторичные ключи пользователей `id_user` и соответствующее им время последнего посещения (листинг 8.60), а затем в цикле выбрать соответствующие им IP-адреса (листинг 8.61).

Листинг 8.60. Выбор вторичных ключей пользователей и времени последнего посещения

```
SELECT DISTINCT(id_user) AS id_user, MAX(puttime) AS max_time
  FROM ips
  GROUP BY id_user
```

Листинг 8.61. Выбор IP-адресов

```
SELECT * FROM ips WHERE id_user=id_user AND puttime=max_time
```

PHP-код, реализующий эту схему, может выглядеть так, как представлено в листинге 8.62.

Листинг 8.62. Выбор IP-адреса каждого посетителя и времени посещения

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
```

```
  // Формируем SQL-запрос
  $query = "SELECT DISTINCT(id_user) AS id_user,
                   MAX(puttime) AS max_time
            FROM ips
            GROUP BY id_user";
  // Выполняем SQL-запрос
  $first = mysql_query($query);
  if($first)
  {
    while($res = mysql_fetch_array($first))
    {
      // В цикле формируем второй SQL-запрос
      $query = "SELECT * FROM ips
                WHERE id_user = ".$res['id_user']."
                      AND puttime=".$res['max_time'];
      $final = mysql_query($query);
      $result = mysql_fetch_array($final);
      echo $result['ip']." - ".$result['puttime']."<br>";
    }
  }
?>
```

Недостатком такого подхода является выполнение второго запроса в цикле, что может создать серьезную нагрузку на сервер MySQL. Выходом из этой ситуации будет создание временной таблицы с результатами первого запроса, как это показано в листинге 8.63.

Листинг 8.63. Создание временной таблицы с результатами запроса

```
<?php
  $query = "CREATE TEMPORARY TABLE ttt
            SELECT id_user, puttime, ip FROM ips
            ORDER BY id_user,time DESC";
  if(mysql_query($query))
  {
    $sel = "SELECT id_user, puttime, ip
            FROM ttt GROUP BY id_user";
    $res = mysql_query($sel);
    if($res)
    {
      while($result = mysql_fetch_array($res))
      {
        echo $result["id_user"]." ".$result["puttime"].
```

```
            " ".$result["ip"]."<br>";
        }
    }
}
?>
```

Как видно из листинга, выполнение второго SQL-запроса вынесено из тела цикла. После выполнения скрипта таблица `ttt`, созданная в памяти, будет уничтожена автоматически.

Удаление и выборка нескольких записей

В Web-программировании, особенно при создании различных панелей администрирования, часто встает задача множественного выбора или удаления сразу нескольких записей из базы данных. Пусть имеется таблица `tbl`, содержащая два поля: первичный ключ таблицы (`id`), текстовое поле (`name`).

Для удаления записей обычно создается HTML-форма с набором флажков, как это показано в листинге 8.64.

Листинг 8.64. Файл index.php

```
<form action=handler.php method=post>
  <input type='checkbox' name='type[]' value='1'>Первый<br>
  <input type='checkbox' name='type[]' value='2'>Второй <br>
  <input type='checkbox' name='type[]' value='3'>Третий<br>
  <input type='checkbox' name='type[]' value='4'>Четвертый<br>
  <input type='checkbox' name='type[]' value='5'>Пятый<br>
  <input type='checkbox' name='type[]' value='6'>Шестой<br>
  <input type='checkbox' name='type[]' value='7'>Седьмой<br>
  <input type=submit>
</form>
```

Обработчик формы handler.php может выглядеть так, как представлено в листинге 8.65.

Листинг 8.65. Файл handler.php

```
<?php
  // Удалить сразу несколько записей можно
  // при помощи запроса "DELETE FROM tbl WHERE id IN (1,3,5,7)"
  // Получаем список отмеченных флажков
  $type = $_POST['type'];
```

```
  if(!empty($type))
  {
    // Начинаем формировать переменную, содержащую этот список
    // в формате "(3,5,6,7)"
    $query = "(";
    foreach($type as $val) $query .= "$val,";
    // Удаляем последнюю запятую, заменяя ее закрывающей скобкой)
    $query = substr($query, 0, strlen($query) - 1).")";
    // Завершаем формирование SQL-запроса на удаление
    $query = "DELETE FROM tbl WHERE id IN ".$query;
    // Выполняем запрос
    if(!mysql_query($query))
    {
      echo mysql_error()."<br>";
      echo $query."<br>";
    }
  }
?>
```

Если перед Web-приложением стоит задача не уничтожения записей, а их выборка, то вместо SQL-оператора `DELETE` следует использовать SQL-оператор `SELECT`, соответствующим образом изменив скрипт из листинга 8.65.

Перенос данных из MySQL в dbf-формат

Часто перед Web-программистом стоит задача переноса данных из какого-либо старого формата в базу данных MySQL или обратная задача: созданную при помощи Web-интерфейса базу данных преобразовать в плоский файл или базу данных старого формата.

Замечание

Перенос данных из MySQL в другую реляционную базу данных, поддерживающую SQL, производится за счет создания дампа базы данных при помощи утилиты mysqldump, которая формирует текстовый файл с SQL-инструкциями, воссоздающих базу данных. Эта утилита входит в состав дистрибутива MySQL и находится в каталоге bin. Например, для создания дампа базы данных `tbl` следует вызвать утилиту со следующими параметрами `mysqldump tbl>tbl.sql`. В результате будет создан дамп базы данных в файле tbl.sql. Загрузить его в базу данных можно при помощи клиента mysql, вызвав его со следующими параметрами: `mysql tbl<tbl.sql`.

Перенос данных будет рассмотрен на примере dbf-формата базы данных dBase, для работы с файлами которого в PHP имеется расширение.

Замечание

Для подключения расширения следует снять комментарий с директивы `extension=php_dbase.dll`. Объем книги не позволяет полностью ознакомиться с функциями данного расширения, но они во многом напоминают функции для работы с плоскими файлами, рассмотренные в *гл. 7*. С синтаксисом dbf-функций можно ознакомиться в документации к PHP.

Пусть имеется таблица `tbl`, SQL-оператор `CREATE` которой представлен в листинге 8.66.

Листинг 8.66. Таблица `tbl`

```
CREATE TABLE tbl (
  data DATETIME NOT NULL default '0000-00-00 00:00:00',
  numer INT(11) NOT NULL,
  podrazd NAME(50) NOT NULL default ''
) TYPE=MyISAM;
```

Тогда dbf-файл из данных таблицы можно создать при помощи скрипта, представленного в листинге 8.67.

Листинг 8.67. Создание dbf-файла из таблицы `tbl`

```
<?php
  include "config.php";
  // Определим структуру таблицы tbl в dbf-формате
  $db_name = "tbl.dbf";
  $def =
    array(
        array("date",   "C", 50),
        array("number", "N", 11, 0),
        array("name",   "C", 50)
    );
  // Создаем таблицу
  if (!dbase_create($db_name, $def))
  {
    echo "Ошибка при создании dbf-таблицы";
    exit();
  }
  // Извлекаем записи из MySQL
  $query = "SELECT * FROM tbl";
  $tbl = mysql_query($query);
  if(!$tbl)
```

```
  {
    echo "Ошибка в синтаксисе SQL-запроса";
    exit();
  }
  // Открываем созданный dbf-файл
  $dbh = dbase_open($db_name,2)
    or die("Ошибка - невозможно открыть '$db_name'");
  // В цикле извлекаем записи из MySQL и заносим их в dbf
  while($table = mysql_fetch_array($tbl))
  {
    echo $table['data'];
    // Формируем структуру записи
    $record =
      array(
        $table['data'],
        $table['number'],
        $table['name'],
      );
    // Добавляем запись в dbf-файл
    if(!dbase_add_record($dbh,$record))
    {
      echo "Ошибка при добавлении записи в dbf-файл";
      exit();
    }
  }
  // Закрываем dbf-файл
  dbase_close($dbh);
?>
```

В отличие от плоских файлов, dbf-формат допускает несколько полей в записи, которые определяются двумерным массивом. Первое измерение определяет число полей в записи — каждому полю соответствует свой собственный массив. Первый элемент каждого такого подмассива содержит текстовую переменную с названием поля, второй элемент — строку с единственной буквой, определяющей тип переменной. Допускаются следующие значения:

- `L` — логический (boolean) тип;
- `D` — дата в формате `ГГГГММДД`;
- `N` — число;
- `C` — строка.

Третий элемент определяет длину строки или количество символов, отводимых под число, для числа (`N`) указывается также количество цифр после запятой.

Создание dbf-файла осуществляется при помощи функции dbase_create(), которая имеет следующий синтаксис:

```
int dbase_create(string filename, array fields)
```

Функция принимает в качестве первого параметра имя создаваемого dbf-файла, а в качестве второго — описанный выше двумерный массив. В случае успеха функция возвращает дескриптор dbf-файла, а в случае неудачи — false.

Дальнейшая работа с dbf-файлом протекает по сходному для всех файлов сценарию. Открытие файла осуществляется при помощи функции dbase_open(), имеющей следующий синтаксис:

```
int dbase_open(string filename, int flags)
```

В качестве первого параметра функция принимает имя dbf-файла, а в качестве второго — число: 0, если файл открывается только для чтения, 1, если только для записи, и 2, если файл отрывается и для чтения, и для записи. Функция возвращает дескриптор файла в случае успеха и false в случае неудачи.

Функция dbase_close() осуществляет закрытие dbf-файла, принимая в качестве единственного аргумента дескриптор, возвращенный функцией dbase_open().

Для помещения записи в dbf-файл используется функция dbase_add_record(), которая имеет следующий синтаксис:

```
bool dbase_add_record(int dbase_identifier, array record)
```

В качестве первого аргумента функция принимает дескриптор, возвращенный функцией dbase_open(). В качестве второго аргумента функция ожидает массив, число элементов которого совпадает с количеством полей, определенных в dbf-файле. Функция возвращает true при успешном выполнении и false в противном случае.

Для удаления записей из файла предназначена функция dbase_delete_record(), которая имеет следующий синтаксис:

```
bool dbase_delete_record(int dbase_identifier, int record)
```

В качестве первого аргумента функция принимает дескриптор, возвращенный функцией dbase_open(). В качестве второго аргумента функции передается номер удаляемой записи. Аналогично dbase_add_record(), функция возвращает true при успешном выполнении и false в противном случае.

Преобразование времени

MySQL поддерживает несколько календарных типов, которые представлены в табл. 8.2. MySQL самостоятельно заботится о преобразовании типов, поэтому все внутренние функции одинаково применимы ко всем типам данных.

Таблица 8.2. *Календарные типы*

Тип	Объем памяти	Диапазон
DATE	3 байта	От '1000-01-01' до '9999-12-31'
TIME	3 байта	От '–828:59:59' до '828:59:59'
DATATIME	8 байт	От '1000-01-01 00:00:00' до '9999-12-31 00:00:00'
TIMESTAMP[(M)]	4 байта	От 19700101000000 до неопределенной даты в 2037 году
YEAR[(M)]	1 байт	От 1901 до 2155 для YEAR(4) От 1970 до 2069 для YEAR(2)

При выборке календарных типов по умолчанию время выводится в формате MySQL, например, текущее время можно получить вызовом SQL-запроса из листинга 8.68.

Листинг 8.68. Текущее время

```
mysql> SELECT NOW();
2005-05-05 15:16:57
```

Преобразовать время можно при помощи внутренней функции MySQL DATE_FORMAT(*date*,*format*), которая форматирует величину *date* в соответствии со строкой *format*. В строке *format* могут использоваться определители, представленные в табл. 8.3.

Таблица 8.3. *Определители формата*

Определитель	Описание
%M	Название месяца (январь, ..., декабрь)
%W	Название дня недели (воскресенье, ..., суббота)

Таблица 8.3 (продолжение)

Определитель	Описание
%D	День месяца с английским суффиксом (0st, 1st, 2nd, 3rd и т. д.)
%Y	Год: число, 4 разряда
%y	Год: число, 2 разряда
%X	Год для недели, где воскресенье считается первым днем недели: число, 4 разряда, используется с '%V'
%x	Год для недели, где воскресенье считается первым днем недели: число, 4 разряда, используется с '%v'
%a	Сокращенное наименование дня недели (Вс, ..., Сб)
%d	День месяца: число (00—31)
%e	День месяца: число (0—31)
%m	Месяц: число (00 — 12)
%c	Месяц, число (0..12)
%b	Сокращенное наименование месяца (Янв, ..., Дек)
%j	День года (001—366)
%H	Час (00—23)
%k	Час (0—23)
%h	Час (01—12)
%I	Час (01—12)
%l	Час (1—12)
%i	Минуты: число (00—59)
%r	Время, 12-часовой формат (hh:mm:ss [AP]M)
%T	Время, 24-часовой формат (hh:mm:ss)
%S	Секунды (00—59)
%s	Секунды (0—59)
%p	AM или PM
%w	День недели (0 — воскресенье, ..., 6 — суббота)
%U	Неделя (00—52), где воскресенье считается первым днем недели
%u	Неделя (00—52), где понедельник считается первым днем недели
%V	Неделя (01—53), где воскресенье считается первым днем недели. Используется с '%X'

Таблица 8.3 *(окончание)*

Определитель	Описание
%v	Неделя (01—53), где понедельник считается первым днем недели. Используется с '%x'
%%	Литерал %

При помощи встроенной функции DATE_FORMAT() можно преобразовать время в любой формат (листинг 8.69).

Листинг 8.69. Преобразование времени

```
mysql> SELECT DATE_FORMAT(NOW(),'%d.%m.%Y');
05.05.2005
mysql> SELECT DATE_FORMAT(putdate,'%d.%m.%Y %k:%i') FROM tbl
07.10.2004 13:26
```

Перенос данных из SQL-файла в базу данных

Как было указано в *разд. "Перенос данных из MySQL в dbf-формат" ранее в этой главе*, утилита mysqldump.exe, входящая в состав дистрибутива MySQL, позволяет создавать дамп базы данных.

Например, в листинге 8.70 создается дамп базы данных base, который помещается в файл base.sql.

Листинг 8.70. Создание дампа базы данных

```
mysqldump base>base.sql
```

Часто встает задача перенести базу данных с локальной машины на сервер. Для этого можно использовать скрипт из листинга 8.71.

Листинг 8.71. Перенос базы данных с локальной машины на сервер

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Имя файла с SQL-инструкциями
  $filename = "base.sql";
```

```
  // Открываем его и читаем в буфер
  $fp = fopen($filename, "r");
  $bufer = fread($fp,filesize($filename));
  fclose($fp);
  // Разбиваем содержимое файла по точке с запятой
  $quer = preg_split("#(\);\r\n|M;\r\n)#i", $bufer);
  // Выполняем SQL-запросы
  foreach($quer as $query)
  {
    if(!mysql_query($query.")")) exit(mysql_error());
  }
?>
```

Задания

8.1. Создайте таблицу, состоящую из двух полей — первичного ключа и текстового поля. Создайте Web-приложение, позволяющее по номеру первичного ключа редактировать текст записи в таблице.

8.2. Дополните предыдущее Web-приложение возможностью полнотекстового поиска по полям таблицы.

8.3. Создайте Web-приложение, формирующее для произвольно существующей таблицы файл, в который помещен SQL-запрос CREATE, создающий данную таблицу.

8.4. Выведите список всех баз данных MySQL. Под каждой из баз данных выведите подсписок таблиц, которые входят в состав базы данных.

8.5. Создайте Web-приложение, представляющее собой HTML-форму, принимающую четыре значения: имя сервера базы данных, имя пользователя, его пароль и имя базы данных. Результатом работы Web-приложения должен быть список таблиц введенной в форму базы данных. Причем рядом с каждой таблицей должно быть выведено ее количество записей.

8.6. Напишите конвертор данных из плоских файлов, которые были рассмотрены в предыдущей главе, в таблицу MySQL и обратно.

8.7. Создайте скрипт, преобразующий dbf-формат в таблицу MySQL.

ГЛАВА 9

Безопасное программирование Web-сайтов

В этой главе рассмотрены вопросы безопасности создаваемых Web-приложений. Безопасность в Интернете включает в себя много различных аспектов, таких как система защиты Web-сервера, безопасность баз данных, проверка данных, вводимых пользователями, приемы и методы шифрования и др.

Проверка данных, вводимых пользователем

Все, что приходит в скрипт из сети, необходимо подвергать самой тщательной проверке, т. к. в принимаемых данных, помимо случайных ошибок посетителей, может быть добавлен зловредный код, подобранный таким образом, что ошибки в проектировании Web-приложения приведут к его выполнению. Для того чтобы избежать этого, вводимый текст необходимо обработать функциями удаления HTML-тегов (для исключения возможности написания скриптов на JavaScript и VBScript) и обратных слешей (для исключения возможности написания скриптов на Perl). Таким образом, минимальный набор действий, необходимый для проверки корректности данных, вводимых пользователем, включает следующие этапы:

- проверка заполнения посетителем обязательных для ввода полей;
- проверка допустимости вводимых данных (как правило, осуществляется при помощи регулярных выражений);
- обработка введенного текста функцией `htmlspecialchars()` для удаления HTML-тегов;
- обработка введенного текста функцией `stripslashes()` для удаления обратных слешей.

Пусть имеется HTML-форма с двумя текстовыми полями для имени пользователя и его e-mail, размещаемая в файле index.php (листинг 9.1).

Листинг 9.1. Файл index.php

```
<form action=handler.php method=post>
  Имя: <input type=text name=name><br>
  e-mail: <input type=text name=email><br>
  <input type=submit value=Отправить>
</form>
```

Проверка ввода данных посетителем может осуществляться с помощью функции `isset()` — листинг 9.2.

Листинг 9.2. Файл handler.php

```
<?php
  // Проверяем, передано ли значение поля name
  if (!isset($_POST['name']))
  {
    // Если переменная $name не установлена, просим повторить ввод имени
    echo "Не введено обязательное к заполнению поле name<br>";
    echo "<a href=# onClick='history.back()'>Вернуться к правке</a>";
    exit();
  }
  else
  {
    // Производим обработку данных HTML-формы
  }
?>
```

В листинге 9.2 приведен обработчик HTML-формы из файла index.php, посылающей методом `POST` значение поля `name`. Если элемент суперглобального массива `$_POST['name']` не установлен, это означает, что текстовое поле `name` не было заполнено. В этом случае в окно браузера выводится предупреждение и ссылка возврата, реализованная при помощи функции JavaScript — `history.back()`.

Код, приведенный в листинге 9.2, не решает проблему пробелов, которые можно ввести в качестве значения текстового поля `name`. Для решения этой проблемы следует использовать функцию `trim()`, которая уничтожит все пробелы в начале и конце переменной. В этом случае вместо функции `isset()` следует использовать функцию `empty()` — листинг 9.3.

Листинг 9.3. Другой вариант файла handler.php

```
<?php
  // Удаляем пробельные символы в начале и конце строки
  if(isset($_POST['name'])) $name = trim($_POST['name']);
  else $name = "";
  // Если значение переменной пусто, выводим сообщение об ошибке
  if (empty($name))
  {
    // Если переменная $name не установлена, просим повторить ввод имени
    echo "Не введено обязательное к заполнению поле name<br>";
    echo "<a href=# onClick='history.back()'>Вернуться к правке</a>";
    exit();
  }
  else
  {
    // Производим обработку данных HTML-формы
  }
?>
```

После того как осуществлена первичная проверка введенных данных, следует воспользоваться регулярными выражениями для более детального анализа данных *(см. гл. 5)*.

Замечание

Не забывайте о том, что, пользуясь вашим приложением, посетитель не должен чувствовать себя словно в клетке. По этой причине не стоит задавать жесткий формат, например, для поля **Имя**. Следует давать возможность пользователю ввести то имя, которое он хочет, а оно может состоять из любых видимых символов.

Замечание

Если в текстовое поле не требуется вводить много текста, ограничивайте ввод пользователя при помощи параметра `maxlength` тега `<input>`. Например, для ввода номера дома не следует отводить более четырех символов `<input name="house" maxlength="4">`.

Функция *htmlspecialchars()*

Вводимый посетителем текст необходимо обрабатывать функциями удаления HTML-тегов (для исключения возможности написания скриптов на JavaScript и VBScript) и обратных слешей (для исключения возможности написания скриптов на Perl). К примеру, если переменная `$name` содержит текст с име-

нем пользователя, то обработка этого текста может выглядеть так, как представлено в листинге 9.4.

Листинг 9.4. Обработка имени пользователя

```
<?php
  $name = htmlspecialchars($_POST['name']);
?>
```

Применение функции `htmlspecialchars()` гарантирует, что любой введенный пользователем код (php, javascript и т. д.) будет отображен, но выполняться не станет. Функция имеет следующий синтаксис:

```
string htmlspecialchars(string str[, int quote_style[, string charset]])
```

Через первый обязательный параметр *str* функции передается обрабатываемый текст, который возвращается после преобразования как результат работы.

Второй необязательный параметр *quote_style* задает режим обработки одинарных и двойных кавычек. По умолчанию этот параметр соответствует константе `ENT_COMPAT`, в данном режиме двойные кавычки заменяются символом `&quot;`, при этом одиночные остаются без изменений. Кроме этого, параметр может принимать два других значения: `ENT_QUOTES` и `ENT_NOQUOTES`. В первом случае, помимо двойных кавычек преобразованию подвергаются также одинарные кавычки, которые заменяются символом `&#039;`. Значение параметра `ENT_NOQUOTES` задает режим, в котором не один из видов кавычек не подвергается преобразованию.

Последний параметр *charset* определяет кодировку, например `"cp1251"` или `"KOI8-R"`.

Межсайтовый скриптинг

Межсайтовый скриптинг (Cross Site Scripting, XSS) позволяет злоумышленнику включать свой HTML-код страницы. Наиболее уязвимы для такого вида атак являются гостевые книги и форумы, где посетителям предоставляется наибольшая свобода в динамическом формировании страниц. Возможности кода, который злоумышленник может вставить в код сайта, практически не ограничены, например, вставка тега `<img>` со ссылкой на скрипт, который расположен на подконтрольном сайте, позволяет собирать различную информацию (например, cookie). В основе атаки лежит использование специальных символов, позволяющих внедрить свой код в страницу. В качестве таких символов чаще используют ' (одинарную кавычку), " (двойную кавыч-

ку), ` (обратную кавычку), > (знак "больше"). Защита от этого вида атак сводится к фильтрованию данных, отосланных пользователем.

Замечание

Подробнее с XSS-атаками можно познакомиться по ссылкам:
http://www.bugtraq.ru/library/www/xssanatomy.html и
http://www.bugtraq.ru/library/www/advancedxss.html.

Так для тега `<input>`, представленного в листинге 9.5, можно провести такого рода атаку.

Листинг 9.5. Небезопасный тег `<input>`

```
<input name="username" value="<? echo $_GET['username'] ?>">
```

В этом случае злоумышленнику достаточно сформировать URL следующего вида:

http://www.server.com/index.php?username="><script>alert(document.cookie)</script>

и код HTML-страницы примет вид, представленный в листинге 9.6.

Листинг 9.6. Модифицированный код тега `<input>`

```
<input name="username" value=""><script>alert(document.cookie)</script>">
```

Данный код не причиняет вреда, а лишь демонстрирует уязвимость XSS, отображая ваши cookies. Следует помнить, что код может быть другим, более опасным и, будучи оставленным, например, в сообщении гостевой книги станет запускаться на машине каждого из посетителей гостевой книги. Для защиты от такого вида атак следует проверять содержимое параметров при помощи регулярных выражений, как это продемонстрировано в *гл. 5*.

Защита имени от подделки

Одним из распространенных способов дискредитации посетителя форума или чата является подделка его имени. Подделка производится путем замены в имени пользователя английских букв сходными по начертанию русскими буквами или наоборот.

Пусть имеется таблица `authors`, содержащая поле `name`, в котором хранятся имена пользователей. Задача будет состоять в проверке нового имени на предмет его уникальности и схожести с другими именами. Решение представлено в листинге 9.7.

Листинг 9.7. Защита от подделки имени

```
<?php
  // Выясняем, не создается ли новое имя для дискредитации
  // Возможны три ситуации, которые необходимо предотвратить:
  // 1. Вводится имя, полностью совпадающее с уже существующим
  // 2. Вводится уже существующее русское имя, в котором
  //    одна или несколько букв заменены на английские
  // 3. Вводится уже существующее английское имя, в котором
  //    одна или несколько букв заменены на русские
  // Массив русских букв
  $rus = array("А", "а", "В", "Е", "е", "К", "М", "Н", "О", "о", "Р",
              "р", "С", "с", "Т", "Х", "х");
  // Массив английских букв
  $eng = array("A", "a", "B", "E", "e", "K", "M", "H", "O", "o", "P",
              "p", "C", "c", "T", "X", "x");
  // Заменяем русские буквы английскими
  $eng_author = str_replace($rus, $eng, $author);
  // Заменяем английские буквы русскими
  $rus_author = str_replace($eng, $rus, $author);
  // Формируем SQL-запрос
  $query = "SELECT * FROM authors
            WHERE name LIKE '$author' OR
                  name LIKE '$eng_author' OR
                  name LIKE '$rus_author'";
  $ath = mysql_query($query);
  if(!$ath) exit("Ошибка при регистрации нового посетителя");
  // Если выборка содержит хотя бы одну запись,
  // прекращаем регистрацию нового посетителя
  if(mysql_num_rows($ath)>0) exit("К сожалению, данное имя уже
                                  зарегистрировано.
                                  Попробуйте другое.");
  // Код регистрации нового посетителя...
?>
```

Для решения проблемы в скрипте формируются два массива: `$rus` и `$eng`, содержащие русские и английские буквы, сходные по написанию. Так как имена обычно состоят полностью из русских или полностью из английских букв, проверяются три варианта, подтверждение каждого из которых приводит к прекращению регистрации:

- имя нового посетителя полностью совпадает с именем из таблицы `authors`;
- имя нового посетителя совпадает с одним из таблицы `authors` после замены всех английских букв в имени русскими;

- имя нового посетителя совпадает с одним из таблицы authors после замены всех русских букв в имени английскими.

Если ни одно из условий не выполняется, новое имя посетителя уникально, и оно добавляется в таблицу authors.

Как просто и быстро стереть весь сайт, или загружаем файлы с исполняемым кодом

Часто Web-приложения, такие как форум или фотогалерея, позволяют посетителям публиковать собственные изображения или файлы. Это могут быть фотографии, музыкальные, текстовые или бинарные файлы. Передача пользовательских файлов на сервер уже несет в себе потенциальную опасность, и при невнимательном кодировании может привести к серьезным последствиям. Для демонстрации угрозы, исходящей из пользовательских файлов, разработаем небольшое Web-приложение, которое закачивает файл пользователя на сайт и предоставляет ссылку для скачивания или его просмотра (если это изображение или текстовый файл).

Приложение будет состоять из двух файлов: index.php, являющегося HTML-формой для отправки файлов на сервер, и handler.php — обработчика формы. Содержимое файла index.php представлено в листинге 9.8.

Листинг 9.8. Файл index.php

```
<form enctype='multipart/form-data' action=handler.php method=post>
  Файл : <input type=file name=filename><br>
  <input type=submit value=Отправить>
</form>
```

HTML-форма имеет единственное поле ввода типа file, а также кнопку, для отправки данных обработчику handler.php, содержимое которого приведено в листинге 9.9.

Листинг 9.9. Файл handler.php

```
<?php
  // Проверяем содержимое параметра filename, передаваемого обработчику,
  // из формы в файле index.php
  if(isset($_POST['filename'])) $filename = trim($_POST['filename']);
  else $filename = "";
  // Если обработчику передано пустое поле filename
  // или содержащее пробелы, то
  // игнорируем обращение, в противном случае выводим ссылку на файл
```

```
  if (!empty($filename))
  {
    // Если копирование произведено удачно, выводим ссылку на файл
    if (copy($filename, $_FILES['filename']['name']))
      echo "<a href=".$_FILES['filename']['name'].">Посмотреть</a>";
    else
      echo "Ошибка при передаче файла на сервер.";
  }
  else
  {
    // Если переменная $filename пуста, просим повторить загрузку файла
    echo "Имя файла не введено, повторите, пожалуйста, операцию.<br>";
    echo "<a href=# onClick='history.back()'>Вернуться к отправке</a>";
    exit();
  }
?>
```

После отправки при помощи HTML-формы, приведенной в листинге 9.6, изображения или текстового файла посетителю предоставляется возможность посмотреть содержимое отправленного файла при переходе по ссылке **Посмотреть**.

На первый взгляд, код в листинге 9.9 производит впечатление достаточно проработанного и предусматривающего все возможные варианты развития ситуации. Тем не менее данное Web-приложение позволяет полностью уничтожить сайт, на котором оно расположено. Отправка при помощи данной формы файла, например, приведенного в листинге 9.10, приводит к рекурсивному удалению файлов с сервера.

Листинг 9.10. Удаление файлов с сервера

```
// Вызываем функцию, удаляющую файлы с сервера, с параметром "..", чтобы
// подняться на один уровень выше: это позволит уничтожить больше файлов,
// т. к. наверняка Web-приложение находится в отдельном каталоге.
delfiles("..");
// Функция, удаляющая каталоги и файлы сервера
function delfiles($catalog)
{
  $dir = opendir($catalog);
  while(($file = readdir($dir)))
  {
    // Если текущий объект является файлом - уничтожаем его
    if(is_file($catalog."/".$file))  unlink($$catalog."/".$file);
```

```
    // Если текущий объект является каталогом – рекурсивно вызываем
    // функцию delfiles(), заботливо избегая каталогов "." и "..",
    // соответствующих текущему и вышележащему каталогам.
    else if (is_dir($catalog."/".$file) &&
            ($file != ".") &&
            ($file != "..")) delfiles($catalog."/".$file);
  }
  // Закрываем каталог
  closedir($dir);
  // Удаляем каталог
  rmdir($catalog);
}
```

В результате работы приведенного в листинге 9.8 кода, будут уничтожены все файлы приложения и файлы, расположенные уровнем выше. После небольшой модификации рекурсию можно направить вверх по дереву каталогов и, если скрипт обладает достаточными правами, можно уничтожить файлы всего сервера.

Решить данную проблему может проверка расширения файла. При этом не нужно запрещать расширения, например php, phtml или php3. Следует разрешать расширения для загружаемых файлов, например jpg или gif, и публиковать только то, что непосредственно разрешено (листинг 9.11). В противном случае, можно не учесть другие расширения. Например, расширения Perl-скриптов pl, файлы с расширениями html и htm позволяют не только публиковать безобидный HTML-код, но и получать файлы с сервера при помощи HTTP-заголовков.

Если же не предоставляется возможность проводить разрешающую политику, необходимо сверится с конфигурационным файлом сервера Apache и запретить все зарегистрированные файлы.

Листинг 9.11. Файл handler.php

```
<?php
  // Проверяем содержимое параметра filename, передаваемого обработчику,
  // из формы в файле index.php
  if(isset($_POST['filename'])) $filename = trim($_POST['filename']);
  else $filename = "";
  // Если обработчику передано пустое поле filename
  // или содержащее пробелы, игнорируем обращение,
  // в противном случае выводим ссылку на файл
  if (!empty($filename))
```

```
  {
    // Формируем массив разрешенных к публикации файлов
    $extentions = array(".gif",".txt",".jpg");
    // Если расширение файла совпадает с одним из расширений файлов
    // массива $extentions, закачиваем файл на сервер
    if(in_array(strrchr($_FILES['attach']['name'], "."),$extentions))
    {
      // Если копирование произведено удачно, выводим ссылку на файл
      if (copy($filename, $_FILES['filename']['name']))
        echo "<a href=".$_FILES['filename']['name'].">Посмотреть</a>";
      else
        echo "Ошибка при передаче файла на сервер.";
    }
    else
    {
      // Если расширение файла имеет недопустимое значение,
      // выводим предупреждение и ссылку возврата
      echo "Файл имеет недопустимое расширение.";
      echo "<a href=# onClick='history.back()'>Вернуться к отправке</a>";
      exit();
    }
  }
  else
  {
    // Если переменная $filename пуста, просим повторить загрузку файла
    echo "Имя файла не введено, повторите, пожалуйста, операцию.<br>";
    echo "<a href=# onClick='history.back()'>Вернуться к отправке</a>";
    exit();
  }
?>
```

Продемонстрированная "дыра" в безопасности показывает, насколько легко можно допустить ошибку при проектировании Web-приложения. Подобного рода ошибки совершить очень легко. Приемы, описанные в данной главе, не оберегут ото всех возможных ошибок. Поэтому необходимо проводить регулярное резервное копирование файлов сервера и дампов баз данных. Лучше, если это будет выполнять не человек, а автоматическая система.

Необратимое шифрование MD5 — зачем?

При необратимом шифровании информация зашифровывается таким образом, что не подлежит обратной расшифровке. На первый взгляд это может показаться странным, в действительности же такой метод шифрования ис-

пользуется очень часто. Функции, с помощью которых реализуется однонаправленное шифрование, называются *функциями хеширования*. При использовании таких функций создается уникальный "отпечаток" строки. Наиболее часто в качестве алгоритма хеширования применяется алгоритм MD5, реализовать который можно с помощью одноименной функции:

```
string md5(string str[, bool raw_output])
```

В качестве обязательного аргумента эта функция принимает строку `str`, которую необходимо зашифровать, и возвращает ее уникальный 128-битовый отпечаток *(хеш-код)*. Если необязательный аргумент `raw_output` имеет значение `true`, то возвращается бинарная строка из 16 символов. Вероятность того, что две строки дадут одинаковый хеш-код, стремится к нулю.

Замечание

Аналогичная функция `md5_file()` часто используется для создания уникального хеш-кода объемных файлов, которые передаются по сети. Загрузив файлы, всегда можно проверить целостность хеш-кода, вычислив его по алгоритму MD5 и сравнив полученный результат с хеш-кодом, предоставляемым распространителем. Это позволяет отследить повреждения файла, вызванные передачей через сеть, а так же предотвращает фальсификацию файла. Такой способ часто применяют при распространении объемных дистрибутивов.

При помощи этой функции можно зашифровывать различные данные, к примеру, пароли пользователей. Это предоставляет возможность организовать следующий алгоритм авторизации пользователей. При первой регистрации пользователя сохраняется хеш-код его пароля (к примеру, в базе данных). При дальнейших посещениях странички хеш-код вводимого пользователем пароля сравнивается с сохраненным ранее хеш-кодом. Если эти отпечатки совпадают, авторизация считается успешной.

Замечание

Такая схема авторизации не позволяет получить непосредственный доступ к паролям, даже если происходит хищение базы данных. В этом случае злоумышленник вынужден тратить значительное машинное время на перебор паролей по словарю, поэтому пароли вида W5t7,9yuP практически не поддаются расшифровке, в то же время необратимое шифрование не сможет защитить от перебора при пароле вида 12345.

Обратимое шифрование с библиотекой mcrypt

При симметричном шифровании строки шифруются с помощью ключа, который известен как отправителю, так и получателю. Алгоритмы симметричного шифрования в PHP реализованы в библиотеке mcrypt, и ознакомиться с пол-

ным набором функций и алгоритмов этой библиотеки можно на сайте **http://www.php.net**. Функции mcrypt работают только, если подключена сама библиотека.

Замечание

Как и любое расширение, библиотека mcrypt отключена по умолчанию в PHP 5. Для того чтобы ее подключить, необходимо убрать комментарий напротив строки `extension=php_mcrypt.dll` в конфигурационном файле php.ini. Кроме этого, в системный каталог C:/Windows/sysyem32 нужно скопировать дополнительную библиотеку libmcrypt.dll, которая не входит в состав дистрибутива PHP, и ее следует загрузить из сети, например, по ссылке **http://www.softtime.ru/files/libmcrypt.dll**.

В листинге 9.12 приведен пример, в котором строка зашифровывается и расшифровывается при помощи алгоритма 3DES (Triple-DES).

Листинг 9.12. Шифрование и дешифрование строки

```
<?php
  // Зашифровываем пароль
  $user_password = "gfkjxrb99";
  $key = "Это секретный ключ";
  // Шифруем пароль с использованием ключа $key
  $user_crypt = mcrypt_ecb(MCRYPT_3DES, $key, $maks_password,
                           MCRYPT_ENCRYPT);
  echo "Зашифрованный пароль - ".$user_crypt;
  // Расшифровываем пароль
  $user_crypt = mcrypt_ecb(MCRYPT_3DES, $key, $user_crypt,
                           MCRYPT_DECRYPT);
  echo "Расшифрованный пароль - ".$user_crypt;
?>
```

В листинге 9.12 пароль, размещенный в переменной `$user_password`, сначала шифруется функцией `mcrypt_ecb()` с применением ключа `$key`, а затем проводится его обратная дешифрация.

Если *register_globals = On*

Директива `register_globals` конфигурационного файла php.ini несет ответственность за непосредственную передачу данных, переданных методами `GET`, `POST`, `SESSION`, `COOKIE` в переменные скрипта. Другими словами, если значение этой директивы установлено в `On`, то, например, содержимое текстового поля `name` передается в обработчике непосредственно в переменную `$name`. При

такой установке нет надобности в проверке источника поступления переменной (от GET, POST или COOKIE), поэтому значения переменных легко могут оказаться поддельными.

Обычно приводят следующий классический пример. Из HTML-формы index.php в обработчик handler.php передаются два параметра: user и password, для имени и пароля, по которым посетителю предоставляется доступ к какой-либо странице, например, управлению его виртуальным счетом (листинг 9.13).

Листинг 9.13. Файл hadler.php

```
<?php
  // Файл handler.php, принимающий от формы index.php два параметра:
  // $user и $password для имени пользователя и пароля соответственно
  if($user == "user1" && $password == "password1")
  {
    $access = 1;
  }
  if($access)
  {
    // Доступ к виртуальному счету
  }
?>
```

В случае установки директивы register_globals в файле php.ini в значение On, логика работы скрипта может быть нарушена. Так, если в строке запроса обратиться по адресу **handler.php?access=1**, авторизация произойдет, минуя HTML-форму и части кода, ответственного за авторизацию. Разумеется, в данном случае защитить Web-приложение очень просто, но в больших Web-приложениях могут быть неочевидные ситуации. На фоне того, что злоумышленник может автоматически обращаться и анализировать ответы, или коды выложены для всеобщего доступа, вероятность взлома возрастает значительно, т. к. отследить все такие ситуации сложно.

Для работы с данными, возвращаемыми методами POST, GET, сессиями, cookie, в PHP были введены так называемые *суперглобальные массивы* — предопределенные массивы, содержащие внешние переменные, которыми и необходимо пользоваться.

Замечание

Возможна безопасная работа и при включенной директиве register_globals, однако в этом случае необходима явная инициализация всех переменных.

Таким образом, следует помнить, что необходимо либо отключить директиву `register_globals`, либо обеспечить явную инициализацию всех переменных. Это делает невозможным 95% всех возможных атак.

Насколько опасны cookies?

Cookies — это механизм хранения данных на компьютере клиента. Иногда в cookie приходится хранить конфиденциальные данные. В этом случае разработчик Web-приложения должен позаботиться о том, чтобы информация, хранящаяся в cookie, не попала третьим лицам. Существует несколько методов защиты информации, хранящейся в cookie:

- установка области видимости cookie;
- шифрование;
- ограничение доступа для доменов;
- отправка cookie по защищенному запросу.

Наилучшим решением является комплексное применение всех этих способов.

Замечание

Более подробно познакомиться с технологией cookie можно в спецификации RFC2965 (**www.ietf.org/rfc/rfc2965.txt**).

Замечание

Ни в коем случае нельзя полностью полагаться на значения из cookie для предоставления доступа к определенным ресурсам сайта, поскольку они находятся на компьютере клиента и легко подвергаются подделке. Так пользователь может изменить свой идентификационный номер на соседний и получить доступ к данным другого посетителя.

Для обеспечения дополнительных мер безопасности информацию, хранящуюся в cookie, можно подвергать шифрованию. Часто в интернет-магазинах и других Web-приложениях, требующих авторизации, предоставляется следующий сервис: авторизовавшись один раз, посетитель в течение некоторого времени имеет возможность не вводить имя и пароль, т. к. постоянный ввод данных может его раздражать. Такой сервис реализуется либо при помощи механизма сессий, либо cookie. Сессии формируют достаточно длинный URL, затрудняющий продвижение ресурса в поисковых системах. Так как поисковые роботы предпочитают не работать с длинными URL, поэтому часто прибегают к механизму cookie. При сохранении данных в cookie их необходимо шифровать. Рассмотрим Web-приложение, которое будет состоять из двух файлов: HTML-формы index.php (листинг 9.14) и ее обработчика handler.php (листинг 9.15).

Листинг 9.14. Файл index.php

```
<?php
  // Вектор начального состояния остается неизменным
  $key = "Это секретный ключ";
  // Расшифровываем пароль, извлеченный из cookie
  $password = mcrypt_ecb(MCRYPT_3DES, $key, $_COOKIE['password'],
                         MCRYPT_DECRYPT);
?>
<form action=handler.php method=post>
  Имя посетителя: <input type=text name=name
                         value=<?php echo $_COOKIE['name']?>><br>
  Пароль: <input type=password name=password
                 value=<?php echo $password;?>><br>
  <input type=submit value=Отправить>
</form>
```

HTML-форма, приведенная в листинге 9.14, имеет два текстовых поля: для ввода имени `name` и пароля `password`, после заполнения которых введенная в них информация отправляется в обработчик формы handler.php (листинг 9.15).

Листинг 9.15. Файл handler.php

```
<?php
  // Создаем вектор начального состояния для шифрования
  $key = "Это секретный ключ";
  $password = mcrypt_ecb(MCRYPT_3DES, $key, $maks_password,
                         MCRYPT_ENCRYPT);
  // Устанавливаем cookie с именем посетителя и паролем (зашифрованным)
  // на двое суток
  setcookie("user", $_POST['name'], time() + 3600*24*2);
  setcookie("password", $password, time() + 3600*24*2);
  // Выводим приветствие
  echo "Здравствуйте, ".$_COOKIE['user'];
  echo "<br><a href=index.php>Вернуться к форме</a>"
?>
```

В обработчике handler.php полученный из формы пароль шифруется по симметричному алгоритму 3DES. И имя пользователя, и зашифрованный пароль сохраняются в cookie со сроком действия двое суток. Теперь, если вернуться к форме, поля формы будут уже заполнены.

Безопасная настройка PHP

В конфигурационном файле php.ini содержатся различные параметры, при помощи которых можно существенно повысить защищенность создаваемых скриптов.

- `display_errors`

  Этот параметр отвечает за вывод ошибок в браузер и должен быть установлен на рабочем сервере в значение `false`, поскольку при возникновении ошибок PHP предоставляет информацию о путях и именах переменных, что может дать атакующему нужную информацию. В директиве конфигурации `error_log` необходимо задать путь к файлу, в который будет производиться запись о произошедших ошибках.

- `error_reporting`

  Данная директива позволяет установить виды ошибок, о которых должно сообщаться. Подробно этот параметр описан в *гл. 1*. При отладке лучше всего установить значение этого параметра в значение `E_ALL`, а в реальной работе лучше оставить только сообщения об ошибках, а вывод предупреждений и уведомлений отключить.

- `variables_order`

  При помощи этого параметра устанавливается порядок, в котором PHP регистрирует переменные окружения (Environment) и переменные `POST`, `GET`, `COOKIE` и `SERVER`. По умолчанию значение этого параметра равно `EPGCS`, что означает регистрацию переменных в указанном порядке. Те переменные, которые регистрируются позднее, переопределяют ранее установленные переменные, т. е. переменные окружения заменяются `GET`-переменными, которые в свою очередь заменяются на `POST`-переменные и т. д. При помощи этого параметра можно отключить регистрацию переменных из конкретного источника, к примеру, от переменных окружения, для чего параметр `variables_order` нужно установить в значение `GPCS`.

- `open_basedir`

  Этот параметр задает каталоги, доступные для функции `fopen()`. К примеру, если необходимо открывать файлы только из каталога /home/php/, то это значение нужно установить для параметра `open_basedir`. Для того чтобы задать несколько каталогов, их следует разделить точкой с запятой (в Windows) и двоеточием (в UNIX). Значение, установленное для этого параметра по умолчанию, позволяет открывать любой файл, что является нежелательным при работе PHP на реальном сервере.

- `disable_functions`

  При помощи этой директивы задается список функций, выполнение которых запрещено. По умолчанию разрешено выполнение любых функций.

В число функций, запрещение которых желательно, входят следующие:

- `exec()` — выполняет указанную команду;
- `passthru()` — выполняет указанную команду и возвращает все результаты в браузер;
- `system()` — то же, что и `passthru()`, только не обрабатывает двоичные данные;
- `shell_exec()` — эквивалент оператора обратного штриха, который PHP выполняет в качестве команды оболочки;
- `popen()` — выполняет указанную команду;
- `phpinfo()` — позволяет узнать настройки и расположение сервера Apache и интерпретатора PHP.

Это вовсе не значит, что использование перечисленных здесь функций следует запретить. Продуманный с точки зрения безопасности код не позволит злоумышленнику воспользоваться гибкостью, предоставляемой данными функциями.

- `allow_url_fopen`

  При помощи этого параметра можно запретить открытие удаленных файлов, расположенных на других серверах. Если отсутствует жесткая необходимость в таком обращении, то значение данного параметра следует установить в `Off`. В ином случае, атакующий может выполнить произвольный код, вызвав какой-либо из ваших скриптов со строкой запроса, к примеру, **?page=http://www.myserver.com/haker.txt**. Это очень частый вид атак, особенно для тех случаев, когда имена страниц сайта передаются через URL путем создания ссылок типа **script.php?page=index.php**. Поэтому *крайне желательно* также не передавать имена страниц через URL, а присвоить номер каждой странице и создать массив с номерами файлов страниц, где индексами массива являются эти номера.

Безопасная установка MySQL

После установки сервера базы данных MySQL система имеет две базы данных:

- `mysql` — база данных, содержащая таблицы разрешений;
- `test` — база данных, обычно используемая для тестирования.

По умолчанию привилегии в таблице разрешений базы данных `mysql` устанавливаются таким образом, что подключиться к серверу может любой пользователь. При этом в качестве пароля используется пустая строка.

Изначально, в системе заводятся две учетные записи: `root` — для администратора (имеющего привилегии суперпользователя) и учетная запись анонимного пользователя. Обе они имеют пустой пароль.

Замечание

Вместо имени анонимного пользователя в СУБД MySQL используется пустая строка.

Под операционной системой Windows привилегии между этими двумя учетными записями распределяются согласно табл. 9.1.

Таблица 9.1. *Распределение привилегий учетных записей*

Хост	Пользователь	Пароль	Привилегии суперпользователя
`localhost`	`root`		Все
`%`	`root`		Все
`localhost`			Все
`%`			Отсутствуют

Как видно из табл. 9.1, изначально и суперпользователю (`root`), и анонимному пользователю назначены пустые пароли и доступ к системе как с локальной машины, так и с любого хоста сети.

Замечание

Символ `%` в СУБД MySQL соответствует любому хосту сети. Это означает, что пользователь, для которого указан данный символ, может работать с любой машины в сети с использованием пароля, указанного в столбце "Пароль".

UNIX

В среде операционной системы UNIX не используется предустановленная таблица разрешений, она создается после инсталляции MySQL при помощи сценария `mysql_install_db`.

При работе с СУБД MySQL на локальном компьютере выставленных по умолчанию разрешений вполне достаточно, однако при функционировании системы на реальном сервере необходимо защитить учетные записи паролем и ограничить доступ к базе данных, разрешив делать это только с проверенных сетевых узлов.

Замечание

Если СУБД MySQL установлена на той же самой машине, что и сервер Apache, имеет смысл оставить только доступ с узла `localhost`.

Существует несколько способов выставить пароль для учетной записи. Первый из них заключается в использовании утилиты mysqladmin, входящей в состав дистрибутива MySQL (листинг 9.16).

Листинг 9.16. Назначение пароля пользователю root *(вариант 1)*

```
# mysqladmin -h localhost -u root password "new_password"
# mysqladmin -h www.softtime.ru -u root password "new_password"
```

Приведенные выше две команды устанавливают для учетной записи root новый пароль, заданный строкой "new_password".

Второй способ связан с возможностями диалекта языка запросов SQL, реализованного в СУБД MySQL. В нем для установки паролей применяется оператор SET PASSWORD (листинг 9.17).

Листинг 9.17. Назначение пароля пользователю root *(вариант 2)*

```
mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('new_password');
mysql> SET PASSWORD FOR 'root'@'%' = PASSWORD('new_password');
```

Так как пароли и разрешения — это просто записи в таблицы user базы данных mysql, то существует еще один способ смены пароля пользователя, основанный на применении оператора UPDATE (листинг 9.18).

Листинг 9.18. Назначение пароля пользователю root *(вариант 3)*

```
mysql> UPDATE user SET Password = PASSWORD('new_password')
mysql> WHERE User='root';
mysql> FLUSH PRIVILEGES;
```

Для того чтобы не перегружать сервер явным образом, в листинге 9.16 используется оператор FLUSH PRIVILEGES.

Для управления пользователями базы данных предназначены два оператора SQL:

- GRANT — оператор, создающий пользователя MySQL и позволяющий настроить его привилегии;
- REVOKE — оператор, позволяющий удалить привилегии пользователя.

Создание нового пользователя в системе осуществляется с помощью оператора GRANT, который имеет следующий синтаксис:

```
GRANT privileges [(columns)], [privileges [(columns)] ...]
   ON table
   TO user [IDENTIFIED BY [PASSWORD] 'password'];
```

Если учетная запись не существует — она создается. Если же учетная запись, определяемая параметром *user*, уже имеется в системе — происходит ее модификация.

Оператор позволяет задать привилегии, определяемые элементом *privileges* для столбцов (*columns*), которые указывать необязательно, таблицы *table*. Элемент *privileges* является оператором (или списком операторов), разрешенным к использованию. В качестве таких операторов могут выступать INSERT, SELECT, UPDATE, ALTER и др. Если в качестве параметра *privileges* указано ключевое слово ALL, привилегии коснутся всех допустимых операторов.

Замечание

За списком всех допустимых операторов следует обращаться к документации MySQL.

Замечание

Вместо имени таблицы можно использовать групповой символ *. Тогда изменения коснутся всех таблиц в базе данных. При этом можно уточнить имя базы данных, используя формат dbname.*. Если параметр *table* равен *.*, привилегии будут применяться ко всем таблицам базы данных.

Параметр *user* определяет учетную запись, которой присваиваются привилегии. Значение этого параметра состоит из имени пользователя и имени узла в формате *'имя_пользователя'@'узел'*. Необязательный параметр *password* задает пароль для учетной записи.

Для удаления привилегий пользователей предназначен оператор REVOKE, имеющий следующий синтаксис:

```
REVOKE privileges [(columns)], [privileges [(columns)] ...]
   ON table
   TO user;
```

Оператор REVOKE применяется для отмены привилегий, но не для удаления пользователей. В таблице user базы данных mysql все равно остается запись для пользователя, даже если все привилегии для него отменены. Для полного удаления пользователя необходимо явно удалить его из таблицы user при помощи SQL-оператора DELETE.

Инъекционные SQL-запросы

Пусть имеется база данных пользователей users, содержащая три поля: первичный ключ (id_user), имя пользователя (name) и его пароль (pass). SQL-оператор CREATE, создающий данную таблицу, приведен в листинге 9.19.

Листинг 9.19. Таблица users

```
CREATE TABLE users (
  id_user int(11) NOT NULL auto_increment,
  name tinytext,
  pass tinytext,
  PRIMARY KEY (id_user)
) TYPE=MyISAM;
INSERT INTO users VALUES (1, 'user1', 'pass1');
INSERT INTO users VALUES (2, 'user2', 'pass2');
INSERT INTO users VALUES (3, 'user3', 'pass3');
```

Авторизация пользователей производится через HTML-форму, приведенную в листинге 9.20.

Листинг 9.20. HTML-форма

```
<form action=handler.php method=post>
  Имя посетителя: <input type=text name=name ><br>
  Пароль: <input type=password name=pass ><br>
  <input type=submit value=Отправить>
</form>
```

Обработчик handler.php проверяет соответствие введенного имени паролю (листинг 9.21).

Листинг 9.21. Обработчик HTML-формы handler.php

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Осуществляем проверку соответствия имени паролю
  $query = "SELECT * FROM users
            WHERE name = '$_POST[name]' AND
                  pass = '$_POST[pass]'";
```

```
  $usr = mysql_query($query);
  if(!$usr) exit("Ошибка в SQL-запросе");
  if(mysql_num_rows($usr)>0)
  {
    // Вход в защищенную область сайта
  }
?>
```

Теперь если в форму из листинга 9.20 в качестве имени пользователя ввести любую строку, например, `chacker`, а вместо пароля пользователя ввести строку:

```
' OR 1=1
```

SQL-запрос будет всегда возвращать все записи таблицы, т. к. условие 1 = 1 будет выполнимо для всех записей. Результирующий SQL-запрос будет выглядеть следующим образом:

```
SELECT * FROM users
WHERE name = 'chacker' AND pass = '' OR 1=1
```

Методы борьбы против такого вида атаки были рассмотрены в *гл. 5*, посвященной регулярным выражениям. Против этого вида запроса можно использовать замену в передаваемых данных всех прямых кавычек (') на обратные (`) — листинг 9.22.

Листинг 9.22. Замена прямых кавычек на обратные

```
<?php
  $_POST['name'] = str_replace("'", "`", $_POST['name']);
  $_POST['pass'] = str_replace("'", "`", $_POST['pass']);
?>
```

В этом случае SQL-запрос примет вид:

```
SELECT * FROM users
WHERE name = 'chacker' AND pass = '` OR 1=1'
```

Если на странице, куда выводятся данные, обратные кавычки не приемлемы, их всегда можно перед выводом заменить на прямые.

Использование параметров, переданных методом `GET`, также не является безопасным. Пусть страницу с информацией о пользователе формирует скрипт user.php, который принимает в качестве параметра первичный ключ `id_user` из листинга 9.19. Код скрипта user.php приведен в листинге 9.23.

Листинг 9.23. Файл user.php

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Осуществляем запрос к базе данных,
  // извлекая запись из таблицы user
  if(isset($_GET['id_user']))
  {
    // Формируем SQL-запрос
    $query = "SELECT * FROM users WHERE id_user = ".$_GET['id_user'];
    // Выполняем SQL-запрос
    $usr = mysql_query($query);
    if($usr) exit("Ошибка при обращении к таблице пользователей");
    $user = mysql_fetch_array($usr);
    // Выводим имя пользователя
    echo $user['name'];
  }
?>
```

Результат работы скрипта представлен на рис. 9.1.

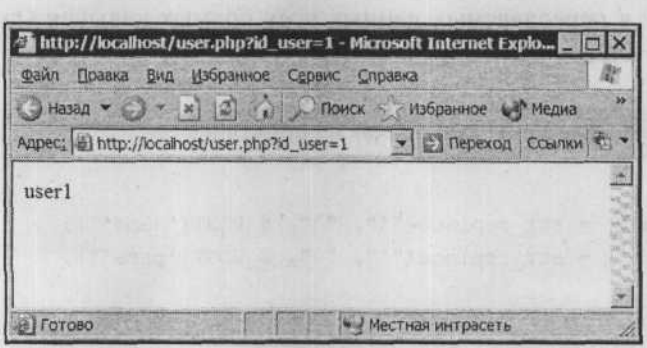


Рис. 9.1. Штатная работа скрипта user.php

Теперь, помещая в адресную строку следующий адрес:

```
http://localhost/user.php?id_user=-1
%20UNION%20SELECT%201,pass,name%20FROM%20users%20WHERE%20id_user=1
```

можно получить пароль пользователя с первичным ключом `id_user = 1` (рис. 9.2).

Пробелы в строке запроса заменяются символом `%20`, который Web-сервер автоматически заменяет на пробелы. Числовые данные, передаваемые в качестве параметров, следует подвергать проверке, как это описано в *гл. 5*.

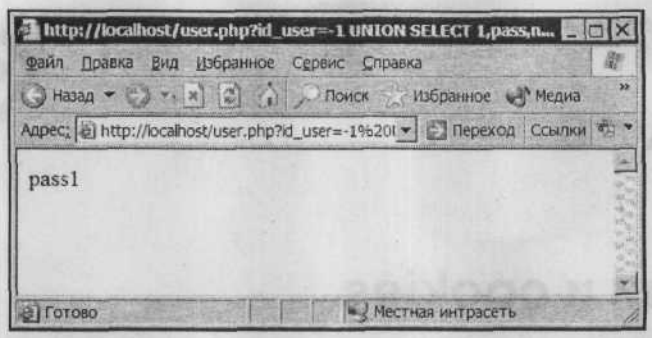


Рис. 9.2. Инъекционный SQL-запрос позволяет узнать пароль пользователя

ГЛАВА 10

Сессии и cookies

HTTP-протокол, лежащий в основе Web, не сохраняет информации о состоянии сеанса. Это означает, что любое обращение клиента сервер воспринимает как обращение нового клиента, и даже если для загрузки картинок с текущей страницы клиент формирует запрос, сервером он воспринимается как запрос нового клиента, никак не связанного с тем, который только что загрузил страницу. Данная схема работала достаточно хорошо для статических страниц, но стала совершенно неприемлемой для динамических. В связи с этим в Web были введены механизмы сессий и cookies, которые в настоящий момент поддерживают все участники Web: клиенты, proxy-серверы и конечные серверы.

Как передать переменную из одного скрипта в другой?

Невозможность сохранения состояния в рамках HTTP-протокола означает, что переменные и массивы, определенные в одном скрипте, не запоминаются при переходе к другому скрипту или при перезагрузке текущей страницы. Существует несколько способов передачи данных из одного скрипта в другой. Первый способ связан с передачей параметров через строку запроса методом `GET`. Так переход по ссылке **http://www.softtime.ru/forum /read.php?id_forum=1&id_theme=495** приведет к тому, что в скрипте, расположенном в файле read.php, передаваемые через параметры `id_forum` и `id_theme` значения будут доступны при обращении к элементам суперглобального массива `$_GET['id_forum']` и `$_GET['id_theme']` (листинг 10.1).

Листинг 10.1. Вывод на экран параметров из строки запроса

```
<?php
  echo $_GET['id_forum'];
```

```
  echo $_GET['id_theme'];
?>
```

Через строку запроса можно передать не только значения отдельных переменных, но и массивы. Например, передача в строке массива параметров **http://www.softtime.ru/forum/index.php?arr[]=1&arr[]=2&arr[]=3** приведет к тому, что в элемент суперглобального массива `$_GET['arr']` будет помещен массив из трех элементов (листинг 10.2).

Листинг 10.2. Передача скрипту массива

```
<?php
  // Выводим содержимое массива $_GET['arr']
  print_r($_GET['arr']);
?>
```

Результат:

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

Точно так же можно передать данные методом `POST`, например, через HTML-форму. В этом случае данные, введенные в элементы управления HTML-формы, будут размещены в элементах суперглобального массива `$_POST`, ключи которых будут совпадать с названиями элементов управления HTML-формы.

Второй способ передачи данных между скриптами заключается в сохранении промежуточных данных в файлах или базе данных. Этому способу были посвящены *гл. 6—8*.

Третий способ заключается в сохранении значений переменных в cookies. Cookies — это текстовые файлы на компьютере клиента, где сохраняют пары вида "имя/значение".

Замечание

Cookie хранятся в системном каталоге Cookies в файлах формата *имя_пользователя@хост*.txt.

Замечание

Файл для cookie создается только в том случае, если выставляется время жизни cookie, в противном случае cookie действует только до конца сеанса, т. е. до того момента, когда пользователь закроет окно браузера.

Для создания cookie предназначена функция `setcookie()`, которая имеет следующий синтаксис:

```
bool setcookie(string name[, string value[, int expire[, string path
                [, string domain[, int secure]]]]])
```

Функция принимает следующие аргументы:

- *name* — имя cookie;
- *value* — значение, хранящееся в cookie с именем `name`;
- *expire* — время в секундах с 1 января 1970 г. По истечении этого времени cookie удаляется с машины клиента;
- *path* — путь, по которому доступен cookie;
- *domain* — домен, из которого доступен cookie;
- *secure* — директива, определяющая, доступен ли cookie по защищенному протоколу HTPPS. По умолчанию эта директива имеет значение 0, что означает возможность доступа к cookie по стандартному незащищенному протоколу HTTP.

Функция возвращает `true` при успешной установки cookie на компьютере клиента и `false` в противном случае. После того как cookie установлен, его значение можно получить на всех страницах Web-приложения, обращаясь к суперглобальному массиву `$_COOKIE`, используя в качестве ключа имя cookie.

Так как cookie передается через заголовки HTTP-запроса, то вызов функции `setcookie()` необходимо размещать до любого вывода в окно браузера функциями `echo()`, `print()` и т. п., а также до любого включения в файл HTML-тегов. Это связано с тем, что данные страницы передаются в теле HTTP-запроса и, встретив вывод в окно браузера, PHP вынужден отправить все заголовки. Когда же дело доходит до функции `setcookie()`, отправить HTTP-заголовок, требующий от браузера пользователя установить cookie, уже невозможно, т. к. заголовки уже ушли клиенту и выполняется передача данных. За выводом в окно браузера следует следить очень внимательно, пробел или перевод строки перед тегом `<?php` тоже считается выводом в окно браузера и приводит к преждевременной отправке заголовков.

Работа с cookie без установки времени жизни продемонстрирована в листинге 10.3.

Листинг 10.3. Подсчет числа обращений к странице

```
<?php
  $_COOKIE['counter']++;
```

```
  setcookie("counter", $_COOKIE['counter']);
  echo("Вы посетили эту страницу $_COOKIE[counter] раз");
?>
```

В листинге 10.3 определяется cookie с именем `counter`, значение которого увеличивается при каждой перезагрузке страницы.

Установка времени жизни cookie осуществляется при помощи функций `time()` и `mktime()`. Функция `time()` возвращает число секунд, прошедших с 00:00:00 1 января 1970 г., и имеет следующий синтаксис:

```
int time(void)
```

Функция `mktime()` позволяет задать произвольное время и имеет следующий синтаксис:

```
int mktime([int hour [, int minute [, int second [, int month
          [, int day [, int year [, int is_dst]]]]]]])
```

Параметры *hour*, *minute*, *second*, *month*, *day*, *year* задают соответственно часы, минуты, секунды, дни и год. Аргумент *is_dst* этой функции определяет, попадает ли дата в период летнего времени, и может принимать следующие значения:

- –1 — по умолчанию, означает, что свойство не задано;
- 0 — временной интервал не приходится на период летнего времени;
- 1 — временной интервал приходится на период летнего времени.

Примеры установки cookie со временем жизни приведены в листинге 10.4.

Листинг 10.4. Установка cookie с определенным временем жизни

```
<?php
  // cookie действителен в течение 10 минут после создания
  setcookie("name", "value", time() + 600);
  // действие этого cookie прекращается в полночь 25 января 2010 года
  setcookie("name", "value", mktime(0,0,0,1,25,2010));
  // действие этого cookie прекращается в 18:00 25 января 2010 года
  setcookie("name", "value", mktime(18,0,0,1,25,2010));
?>
```

Замечание

В большинстве современных систем, где время представляется 32-битным целым числом, допустимыми являются значения года между 1901 и 2038. Типичной ошибкой при работе с cookie является указание года, выходящего за этот интервал, например, 2100. В этом случае cookie будут устанавливаться только как сессионные.

Ограничить доступ к cookie для всех страниц, кроме расположенных в конкретном каталоге, к примеру /web, можно, задав четвертый параметр с указанием в нем каталога на сервере, файлы которого могут получить доступ к cookie (листинг 10.5).

Листинг 10.5. Ограничение доступа с cookies

```
<?php
  setcookie("name", "value", time() + 600, "/web");
?>
```

Однако и в этом случае каталоги /web, /web1 и т. д. будут удовлетворять этому ограничению. Если это является нежелательным, можно ограничить область видимости cookie до конкретной страницы (листинг 10.6).

Листинг 10.6. Сужение области видимости cookie

```
<?php
  setcookie("name", "value", time() + 600, "/web/index.php");
?>
```

Но и такой способ в полной мере не решает проблему, поскольку в этом случае доступ к информации, содержащейся в cookie, может получить, к примеру, скрипт /web/index.php-script/anti_cookie.php. Во избежание доступа с посторонних хостов можно еще более сузить область видимости, задав хост, страницы которого имеют доступ к cookie, например, как это продемонстрировано в листинге 10.7.

Листинг 10.7. Устанавливаем хост, с которого видны cookie

```
<?php
  setcookie("name", "value", time() + 600, "/", "www.softtime.ru");
?>
```

Уничтожить cookies можно, установив им нулевое время жизни (листинг 10.8).

Листинг 10.8. Удаление cookie

```
<?php
  setcookie("name", "value");
?>
```

Замечание

Следует помнить, что для уничтожения cookie с ограничениями по каталогу необходимо выставлять точно такие же ограничения при удалении cookie, в противном случае cookie не будет уничтожен.

Четвертый способ передачи данных между отдельными Web-страницами заключается в инициализации сессии. *Сессия* во многом походит на cookie и представляет собой текстовый файл, хранящий пары "ключ/значение", но уже не на компьютере клиента, а на сервере. Во многих случаях сессии являются более предпочтительным вариантом, чем cookie, т. к. последние часто отключают посетители страниц.

Поскольку на сервере скапливается большое число файлов, принадлежащих сессиям разных клиентов, то для их идентификации каждому новому клиенту назначается уникальный номер — идентификатор сессии, который передается либо через строку запроса, либо через cookies, если они доступны. К недостаткам сессий относится невозможность контроля времени их жизни из PHP-скриптов, т. к. этот параметр задается в конфигурационном файле php.ini директивой `session.cookie_lifetime`.

Замечание

Оба механизма, сессии и cookies, взаимодополняют друг друга. Cookies находятся на компьютере посетителя, и время их жизни определяет разработчик, а сессии хранятся на сервере и время их жизни (обычно небольшое) определяет администратор сервера.

Если cookies обычно применяют для долгосрочных задач (от нескольких часов) и хранения информации, которая относится всецело к конкретному посетителю (личные настройки, учетные записи, пароли и т. п.), то сессии предназначены для краткосрочных задач (до нескольких часов) и хранения и обработки информации обо всех посетителях в целом (число посетителей online и т. п.). Поэтому использовать тот или иной механизм следует в зависимости от задачи.

Замечание

Директива `session.save_path` в конфигурационном файле php.ini позволяет задать путь к каталогу, в котором сохраняются файлы сессий. Это может быть удобным для отладки Web-приложений на локальном сервере. Если данная директива не задана, сессии хранятся в оперативной памяти сервера.

Инициируется сессия при помощи функции `session_start()`, которая имеет следующий синтаксис:

```
bool session_start(void)
```

Функция возвращает `true` в случае успешной инициализации сессии и `false` в противном случае. Для работы с сессией функция `session_start()` должна

вызываться на каждой странице, где происходит обращение к переменным сессии.

Замечание

Точно так же, как и функция `setcookie()`, функция `session_start()` должна вызываться до любого вывода в окно браузера.

Замечание

Если необходимо применять функции `setcookie()` и `session_start()` после вывода в окно браузера, необходимо воспользоваться функциями управления вывода, позволяющими задержать отправку страницы клиенту. Данные функции рассмотрены в *гл. 17*.

После инициализации сессии появляется возможность сохранять информацию в суперглобальном массиве `$_SESSION`. В листинге 10.9 на странице index.php в массив `$_SESSION` сохраняется переменная и массив.

Листинг 10.9. Помещаем данные в суперглобальный массив $_SESSION

```
<?php
  // Инициируем сессию
  session_start();
  // Помещаем значение в сессию
  $_SESSION['name'] = "value";
  // Помещаем массив в сессию
  $arr = array("first", "second", "third");
  $_SESSION['arr'] = $arr;
  // Выводим ссылку на другую страницу
  echo "<a href=other.php>другая страница</a>"
?>
```

На страницах, где происходит вызов функции `session_start()`, значения переменных можно извлечь из суперглобального массива `$_SESSION`. Так в листинге 10.10 на странице other.php появляется возможность извлечь помещенные ранее значения переменных.

Листинг 10.10. Извлечение данных из суперглобального массива $_SESSION

```
<?php
  // Инициируем сессию
  session_start();
  // Выводим содержимое суперглобального массива $_SESSION
  print_r($_SESSION);
?>
```

Результат работы скрипта:

```
Array
(
    [name] => value
    [arr] => Array
        (
            [0] => first
            [1] => second
            [2] => third
        )
)
```

Завершить работу сессии можно, вызвав функцию `session_destroy()`, которая имеет следующий синтаксис:

```
bool session_destroy(void)
```

Функция возвращает `true` при успешном уничтожении сессии и `false` в противном случае.

Если уничтожения текущей сессии не требуется, а необходимо лишь обнуление всех значений, хранящихся в сессии, следует вызвать функцию `session_unset()`, которая уничтожит все элементы в суперглобальном массиве `$_SESSION` текущей сессии. Данная функция, точно так же как и функция `session_destroy()`, не принимает ни одного параметра и возвращает `true` в случае успеха и `false` в случае неудачи.

Если требуется уничтожить какой-то один элемент в суперглобальном массиве `$_SESSION`, то следует использовать функцию `unset()`, например:

```
unset($_SESSION['name']);
```

Еще одной полезной функцией является функция `session_id()`, помогающая узнать текущий идентификатор сессии или задать собственный идентификатор и имеющая следующий синтаксис:

```
string session_id([string id])
```

Функция возвращает текущий идентификатор сессии. Необязательный параметр *id* позволяет задать собственный идентификатор сессии, который должен состоять из строчных или прописных букв английского алфавита и цифр. При задании собственного идентификатора функция `session_id()` должна вызываться до функции `session_start()` — листинг 10.11.

Листинг 10.11. Узнаем текущий идентификатор сессии

```
<?php
  // Инициируем сессию
  session_start();
```

```
  // Узнаем текущий идентификатор сессии
  echo session_id();
?>
```

Результатом работы скрипта из листинга 10.11 может быть строка:

```
a27e8c4724f07cae913c50e55cea1b38
```

Таким образом, передача данных между страницами возможна четырьмя способами: методами GET и POST, путем сохранения передаваемых данных в файлах или базе данных, за счет сохранения данных на компьютере клиента по механизму cookie и за счет сохранения данных на сервере с помощью механизма сессий. Все способы, применяемые в настоящее время для сохранения состояния между сеансами в Web, сводятся к использованию одного из этих методов или их комбинации.

Как проверить, включены ли cookies?

Нередко посетители отключают cookies в настройках своих браузеров. Для корректной работы в Web-приложение, использующее cookie, необходимо помещать код, проверяющий, включены ли cookies у посетителя. Такая проверка позволит использовать другой механизм аутентификации, например сессии, или просто разрешит вывести сообщение о необходимости включить cookies. Пример такой проверки приведен в листинге 10.12.

Листинг 10.12. Проверка, включены ли cookies

```
<?php
if(!isset($_GET['probe']))
{
  // Устанавливаем cookie с именем "test"
  if(setcookie("test","set"))
  {
    // Посылаем заголовок переадресации на страницу,
    // с которой будет предпринята попытка установить cookie
    header("Location: $PHP_SELF?probe=set");
  }
}
else
{
  if(!isset($_COOKIE["test"]))
  {
    echo("Для корректной работы приложения необходимо включить cookie");
  }
```

```
  else
  {
    // cookie включены, переходим на нужную страницу,
    // послав заголовок, содержащий адрес этой страницы
    header("Location: $PHP_SELF");
  }
}
?>
```

Для проверки корректности работы с cookie при помощи функции `setcookie()` устанавливается пробное значение cookie. Функция `setcookie()` в данном случае принимает два параметра, первый из которых — имя, а второй — значение cookie.

Защита от заполнения формы с другого сайта

Воссоздав HTML-форму, расположенную на вашем Web-сайте, например, подсмотрев ее параметры в исходном HTML-коде, злоумышленник получает возможность автоматически подбирать пароли пользователей или предоставлять ресурсы вашего сайта со своей страницы. В конце концов, он имеет возможность поместить в вашу гостевую книгу или форум сотни две сообщений рекламного характера, удаление которых может занять время и силы.

Замечание

Размещение однотипных сообщений на форуме или в гостевой книге на жаргоне Web-разработчиков называется флудом (от англ. *flood* — поток, избыток).

Для защиты от такого рода атак в HTML-форму следует добавить скрытое поле, в которое необходимо поместить идентификатор сессии. После того как данные будут отправлены скрипту-обработчику, в нем следует проверить соответствие переданного идентификатора и текущего идентификатора сессии (листинг 10.13).

Листинг 10.13. Защищенная HTML-форма

```
<?php
  // Инициируем сессию
  session_start();
?>
<form action=handler.php method=post>
  Имя: <input type=text name=name><br>
```

```
  Пароль: <iput type=password name=pass><br>
  <input type=submit name=send value=Отправить>
  <input type=hidden name=session_id
         value=<?php echo echo session_id();?>>
</form>
```

Как видно из листинга 10.13, HTML-форма содержит дополнительное поле с именем `session_id`. После того как пользователь нажимает кнопку **Отправить**, данные отправляются обработчику, код которого приведен в листинге 10.14.

Листинг 10.14. Обработчик формы из листинга 10.12

```
<?php
  // Инициируем сессию
  session_start();
  // Сравниваем переданный идентификатор из формы
  // с текущим идентификатором сессии
  if($_POST['session_id'] != session_id())
  {
    exit("Попытка передачи данных с другого хоста. Скрипт остановлен.");
  }
  // Дальнейшая обработка данных...
?>
```

Авторизация с помощью сессий

Проблема авторизации посетителей встает достаточно часто: при предоставлении доступа администрации сайта к системе администрирования, для предоставления доступа пользователям к их персональным данным, например для редактирования, и т. п. В создаваемой системе авторизации имена и пароли будут храниться в таблице MySQL — `users`. В листинге 10.15 приведен SQL-оператор `CREATE`, с помощью которого можно воссоздать данную таблицу.

Листинг 10.15. Таблица `users`

```
CREATE TABLE users (
  id_user INT NOT NULL AUTO_INCREMENT,
  name TINYTEXT,
  pass TINYTEXT,
  PRIMARY KEY (id_user)
) TYPE=MyISAM;
```

Таблица users имеет три поля:

- id_user — первичный ключ таблицы, снабженный атрибутом AUTO_INCREMENT;
- name — имя пользователя;
- pass — пароль пользователя.

Для того чтобы иметь возможность протестировать систему авторизации, необходимо добавить нескольких пользователей в таблицу users, как это продемонстрировано в листинге 10.16.

Листинг 10.16. Несколько записей в таблице users

```
INSERT INTO users VALUES(NULL, 'user1', MD5('pass1'));
INSERT INTO users VALUES(NULL, 'user2', MD5('pass2'));
INSERT INTO users VALUES(NULL, 'user3', MD5('pass3'));
```

Пароли каждого из пользователей подвергаются необратимому шифрованию при помощи встроенной функции MySQL — MD5(). В том случае, если злоумышленник завладеет базой данных, он не сможет воспользоваться зашифрованными паролями, или, по крайней мере, на их расшифровку (осуществляемую перебором) будет затрачено длительное время, за которое пароли могут смениться.

Замечание

В реальных Web-приложениях необходимо создавать систему администрирования, позволяющую управлять пользовательскими учетными записями. К сожалению, объем и формат книги не позволяют рассмотреть приемы создания такой системы.

Для авторизации пользователя необходимо создать HTML-форму index.php, которая представлена в листинге 10.17.

Листинг 10.17. HTML-форма для авторизации (файл index.php)

```
<?php
  // Это файл index.php
  session_start();
?>
<form action=handler.php method=post>
  Имя посетителя: <input type=text name=name
                         value=<?php echo $_SESSION['user']; ?>><br>
  Пароль: <input type=password name=password
                 value=<?php echo $_SESSION['password']; ?>><br>
```

```
  <input type=submit value=Отправить>
  <input type=hidden name=session_id
         value=<?php echo echo session_id();?>>
</form>
<?php
  // Если посетитель авторизовался, приветствуем его
  if(isset($_SESSION['user']))
  {
    echo "Здравствуйте, ".$_SESSION['user']."!";
  }
?>
```

Обработчик handler.php данной формы представлен в листинге 10.18.

Листинг 10.18. Файл handler.php

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Сравниваем переданный идентификатор из формы
  // с текущим идентификатором сессии
  if($_POST['session_id'] != session_id())
  {
    exit("Попытка передачи данных с другого хоста. Скрипт остановлен.");
  }
  // Формируем и выполняем SQL-запрос: имеется ли пользователь
  // с именем $_POST['name']
  $query = "SELECT pass=MD5('$_POST[password]') FROM users
            WHERE name='$_POST[name]'";
  $nme = mysql_query($query);
  if(!$nme) exit ("Ошибка выполнения запроса");
  // Если запрос вернул результат, производим дальнейшую обработку
  if(mysql_num_rows($nme) > 0)
  {
    // Идентификация прошла успешно, разрешаем
    // "вход" посетителя. Для того чтобы в течение текущей
    // сессии посетитель не вводил свое имя, пароль повторно,
    // передаем их через сессию
    if(session_start())
    {
      $_SESSION['user'] = $_POST['name'];
      $_SESSION['password'] = $_POST['pass'];
```

```
        // Осуществляем автоматический переход
        // на страницу index.php, чтобы убедиться, что
        // посетитель "вошел"
        echo "<HTML><HEAD>
              <META HTTP-EQUIV='Refresh' CONTENT='0; URL=index.php'>
              </HEAD><body>";
    }
    else exit("Ошибка идентификации: неправильный пароль");
    // Если в результате запроса не получено ни одной
    // строки, посетитель с таким именем не зарегистрирован
    else exit("Ошибка идентификации: посетитель не зарегистрирован");
?>
```

Если идентификация посетителя проходит успешно, выполняется автоматический переход посетителя на страницу index.php.

Замечание

При идентификации по механизму cookies вместо суперглобального массива `$_SESSION` следует использовать суперглобальный массив `$_COOKIE`, а после успешной авторизации регистрировать переменные `user` и `password` при помощи функции `setcookie()`.

Определение посетителей online, или как отслеживать "уход" посетителей

Во многих Web-приложениях, таких как чаты, форумы, интернет-магазины, требуется отслеживать посетителей, находящихся в данный момент на сайте, а также момент ухода их с сайта. Будем считать, что авторизация посетителей на сайте происходит так, как это описано в предыдущем разделе.

Протокол HTTP не позволяет устанавливать сеансы и, как следствие, не предоставляет возможности отслеживать длительность работы посетителя с Web-ресурсом. Единственное, что можно зафиксировать, — это время обращения клиента к ресурсам сервера. После этого посетитель может часами читать загруженную страницу — Web-сервер не сможет узнать момент прекращения работы посетителя с загруженными ресурсами. Поэтому будем считать, что посетитель покинул Web-ресурс, если с момента последней загрузки их страницы прошло более 20 минут.

Для фиксирования времени обращения посетителей к страницам ресурса необходимо создать таблицу MySQL `session`, в которой будут храниться имена пользователей и время последнего обращения их к странице (листинг 10.19).

Листинг 10.19. Таблица session

```
CREATE TABLE session (
  id_session tinytext NOT NULL,
  putdate datetime NOT NULL default '0000-00-00 00:00:00',
  user tinytext NOT NULL
) TYPE=MyISAM;
```

Таблица содержит три поля:

- `id_session` — идентификатор сессии;
- `putdate` — время последнего обращения посетителя к страницам сайта;
- `user` — имя пользователя.

Тогда для регистрации посетителей, которые находятся в данный момент на сайте, следует создать скрипт, представленный в листинге 10.20.

Листинг 10.20. Регистрация посетителей

```
<?php
  // Начинаем сессию
  session_start();
  // Получаем уникальный id сессии
  $id_session = session_id();
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Проверяем, присутствует ли такой id в базе данных
  $query = "SELECT * FROM session
            WHERE id_session = '$id_session'";
  $ses = mysql_query($query);
  if(!$ses) exit("<p>Ошибка в запросе к таблице сессий</p>");
  // Если сессия с таким номером уже существует, значит,
  // пользователь online. Обновляем время его последнего посещения
  if(mysql_num_rows($ses)>0)
  {
    $query = "UPDATE session SET putdate = NOW(),
                                 user = '$_SESSION[user]'
              WHERE id_session = '$id_session'";
    mysql_query($query);
  }
  // Иначе, если такого номера нет - посетитель только что
  // вошел, - помещаем в таблицу нового посетителя
```

```
  else
  {
    $query = "INSERT INTO session
              VALUES('$id_session', NOW(), '$_SESSION[user]')";
    if(!mysql_query($query))
    {
      echo $query."<br>";
      echo "<p>Ошибка при добавлении пользователя</p>";
      exit();
    }
  }
  // Будем считать, что пользователи, которые отсутствовали
  // в течение 20 минут, покинули ресурс.
  // Удаляем их id_session из базы данных
  $query = "DELETE FROM session
            WHERE putdate < NOW() - INTERVAL '20' MINUTE";
  mysql_query($query);
?>
```

В начале скрипта при помощи функции `session_id()` определяется текущий идентификатор сессии. Если такой идентификатор отсутствует в таблице `session`, происходит добавление новой записи, что эквивалентно приходу на ресурс нового посетителя. Если же идентификатор в таблице `session` существует, следовательно, данный посетитель уже зашел на ресурс, и требуется обновить время его посещения и имя на тот случай, если он авторизовался под другим именем или осуществил авторизацию только сейчас. В конце скрипта происходит удаление всех записей таблицы, время посещения для которых было обновлено более чем 20 минут назад — таким образом фиксируется уход посетителя с ресурса.

Замечание

В чатах принято сообщать, что посетитель покинул Web-ресурс. Для этого пред удалением имеет смысл выбрать все записи, время обновления которых было произведено более чем 20 минут назад, и сообщить, что посетители с такими-то именами покинули чат.

Скрипт в листинге 10.20 следует подключить при помощи инструкции `include` ко всем часто посещаемым страницам сайта, для того чтобы надежно отслеживать присутствие посетителя на сайте.

После того как организовано динамическое обновление таблицы `session`, не составляет труда вывести список текущих посетителей, как это продемонстрировано в листинге 10.21.

Листинг 10.21. Выводим список текущих посетителей

```
<?php
  // Устанавливаем соединение с базой данных
  include "config.php";
  // Выводим имена всех посетителей, записи о которых имеются
  // в таблице session
  $query = "SELECT * FROM session";
  $ath = mysql_query($query);
  if(!$ath) exit("<p>Ошибка в запросе к таблице сессий</p>");
  // Если хоть кто-нибудь есть, выводим таблицу
  if(mysql_num_rows($ath)>0)
  {
    echo "<table>";
    while($author = mysql_fetch_array($ath))
    {
      // Если посетитель не зарегистрирован,
      // выводим вместо его имени - "аноним"
      if(empty($author['user'])) echo "<tr><td>аноним</td></tr>";
      else echo "<tr><td>".$author['user']."</td></tr>";
    }
    echo "</table>";
  }
?>
```

В цикле формируем таблицу с именами посетителей. Если имя не известно, например, посетитель не прошел авторизацию — выводим надпись "аноним".

Задания

10.1. Создайте скрипт, сохраняющий в cookie массив, в том числе многомерный, не прибегая к его упаковке в строку.

10.2. Модифицируйте систему определения посетителей online таким образом, чтобы в таблицу session помещалась информация об IP-адресе посетителя, а сама система имела возможность запретить просмотр ресурса сайта с определенного IP-адреса.

ГЛАВА 11

Объектно-ориентированное программирование и исключения

Объектно-ориентированная технология создания программного кода создана в ответ на все возрастающий объем кода современных приложений. Используя структурный подход, становится все труднее отследить те многочисленные связи между блоками кода, которые возникают в большом приложении. Поэтому технологии создания программного кода выходят на другой абстрактный уровень — оперирование объектами, которые сами отслеживают связи внутри своего кода и определяют интерфейс взаимодействия с другими Web-приложениями.

Объектно-ориентированный подход за последние два десятилетия получил широкое распространение и введен во многие языки программирования, в том числе и в PHP 5.

Замечание

Более полно принципы объектно-ориентированного программирования описываются в книге Кузнецова М. В., Симдянова И. В. "Самоучитель PHP 5"[1].

Механизм исключений введен в PHP для обработки особых ситуаций, называемых *исключительными ситуациями* или *исключениями*. Большие объемы кода современных приложений приводят к тому, что предусмотреть все возникающие ошибки или ситуации, приводящие к ним, невозможно, поскольку это требует дополнительного расширения кода, что в свою очередь способно привести к новым ошибкам. Поэтому практически во всех современных языках программирования вводится особый механизм обработки таких ошибочных ситуаций. Теперь для того чтобы проверять определенные ошибки (например, деление на ноль) в различных местах Web-приложения, больше не требуется анализировать корректность исполнения блока кода непосредст-

[1] Кузнецов М. В., Симдянов И. В. Самоучитель PHP 5. — СПб.: БХВ-Петербург, 2004. — 560 с.

венно в этом участке программы — такая проверка производится в специальной части программы, так называемом *обработчике ошибок*. Данный подход улучшает структуру и читабельность программы, т. к. части программы, отвечающие за выполнение прикладных задач, отделяются от частей, ответственных за обработку ошибок.

Создание классов

Класс — это конструкция, задающая образец, по которому будет построен объект. Разработав класс один раз, можно затем создавать любое количество объектов этого класса, которые будут содержать в себе все переменные и методы, относящиеся к их классу. Изменение кода класса приводит к автоматическому изменению свойств всех объектов, во всех Web-приложениях, которые используют данный класс.

Объявление класса осуществляется при помощи ключевого слова `class`, за которым следуют название класса и компоненты класса в фигурных скобках.

Примечание

Объявление класса нельзя разрывать, т. е. класс должен полностью находиться в одном файле и в одном блоке ограничителей `<?php ... ?>`.

Компонентами класса являются переменные и функции, которые, как уже было сказано выше, называют *членами* и *методами класса*. В листинге 11.1 приведен пример простого класса.

Листинг 11.1. Очень простой класс

```
<?php
  // Начало класса
  class PrimitiveClass
  {
    // Объявляем член класса
    var $member = "Это член класса PrimitiveClass <br>";
    // Объявляем метод класса
    function method()
    {
      echo "Вызван метод класса PrimitiveClass <br>";
    }
  }
  // Конец класса

  // Объявляем объект класса PrimitiveClass
  $object = new PrimitiveClass();
```

```
  // Вызываем метод method()
  $object->method();
  // Обращаемся к члену $member
  echo $object->member;
?>
```

Объект класса объявляется при помощи ключевого слова `new`, за которым следует название класса. Как только объект класса объявлен, появляется возможность обращаться к его компонентам при помощи уточненного имени, которое включает имя объекта, следующую за ним стрелку (->) и имя метода или члена класса.

Результатом исполнения скрипта из листинга 11.1 будут две строки, первая из которых сообщит, что вызван метод `method()` класса `PrimitiveClass`, а вторая, что произошло обращение к члену класса `$member`.

Примечание

При обращении к переменным класса знак `$` перед ними не указывается.

Методы и члены класса могут вызываться не только из внешней программы, но и из самого класса. Для того чтобы обратиться к переменной или методу класса внутри класса, их вызов необходимо предварить конструкцией `$this->`. Переменная `$this`, которая неявно присутствует в каждом классе, является ссылкой на текущий объект класса и сообщает интерпретатору PHP, что происходит обращение к переменной данного класса, а не создание новой переменной. В листинге 11.2 продемонстрирован вызов одного метода класса другим.

Листинг 11.2. Вызов одного метода класса другим методом

```
<?php
  class PrimitiveClass
  {
    // Объявляем член класса
    var $member = "Это член класса PrimitiveClass <br>";
    // Объявляем метод класса
    function method()
    {
      echo "Вызван метод класса PrimitiveClass <br>";
    }
    function use_method()
    {
```

```
      echo "При помощи метода use_method() можно вызвать
            метод method() <br>";
      $this->method();
    }
  }
  // Объявляем объект класса PrimitiveClass
  $object = new PrimitiveClass();
  // Вызываем метод use_method()
  $object->use_method();
?>
```

Результат выполнения этого кода — две строки, одна из которых будет выведена методом `use_method()`, а вторая — методом `method()`.

Членам класса можно присваивать значения как обычным переменным программы. В листинге 11.3 члену `member` класса `PrmitiveClass` присваивается числовое значение, а затем оно выводится при помощи метода `PrintMember()` этого класса.

Листинг 11.3. Присваивание значения члену класса

```
<?php
  class PrmitiveClass
  {
    var $member;
    // Объявляем метод
    function PrintMember()
    {
      echo "Член member = {$this->member}";
    }
  }
  // Объявляем объект класса PrmitiveClass
  $obj = new PrmitiveClass;
  // Присваиваем члену member значение
  $obj->member = 3;
  // Обратимся к методу PrintMember() объекта $obj
  $obj->PrintMember();
?>
```

Среди методов класса различают два особых метода: конструктор и деструктор. *Конструктор* — это специальный метод класса, предназначенный для инициализации членов класса. Этот метод выполняется раньше всех остальных методов класса во время объявления объекта. В конструкторе обычно

осуществляют инициализацию членов класса и резервирование ресурсов, необходимых для работы объекта — выполняется открытие файлов, соединение с базой данных и т. д. Для того чтобы использовать в классе конструктор, необходимо объявить в нем метод с именем `__construct()`.

Примечание

Ключевое слово `__construct` предваряется двумя знаками подчеркивания.

Деструктор — это специальный метод класса, предназначенный для освобождения ресурсов, занятых объектом во время его существования. Этот метод всегда вызывается после всех остальных методов, во время уничтожения объекта. В нем обычно закрываются открытые файлы и соединения с базой данных.

Примечание

Конструктор и деструктор — это ключевые методы класса. Их наличие является одним из признаков объектно-ориентированной технологии. Это не значит, что в каждом классе обязательно должны присутствовать конструктор и деструктор — это необязательные элементы класса и их следует применять только при необходимости.

В листинге 11.4 на примере работы с файлом продемонстрирована работа с конструктором и деструктором.

Листинг 11.4. Использование конструктора и деструктора

```
<?php
  class WorkWithFile
  {
    // Буфер
    var $buff;
    // Имя файла
    var $filename;

    // Конструктор класса
    function __construct($filename)
    {
      // Имя файла помещаем в член filename
      $this->filename = $filename;
      // Проверяем существование файла
      if(!file_exists($this->filename)) exit("Файл не существует");
      // открытие файла
      $fd = fopen($filename, "r");
      if(!$fd) exit("Ошибка открытия файла");
```

```
        // чтение из файла
        $this->buff = fread($fd,filesize($this->filename));
        // закрытие файла
        fclose($fd);
      }

      // Метод выводит содержимое файла
      function getContent()
      {
        return $this->buff;
      }

      // Метод выводит размер файла
      function getSize()
      {
        return filesize($this->filename);
      }

      // Метод выводит число строк в файле
      function getCount()
      {
        // Если файл загружен
        if(!empty($this->filename))
        {
          $arr = file($this->filename);
          return count($arr);
        }
        else return 0;
      }
    }
?>
```

Методы класса, в том числе конструктор и деструктор, представляют собой обычные функции, которые могут принимать аргументы. Это придает классам дополнительную гибкость. К примеру, можно передавать названия файлов в качестве параметра конструктора. В этом случае можно работать с несколькими файлами, объявив несколько объектов (листинг 11.5).

Листинг 11.5. Передача аргументов конструктору и деструктору

```
<?php
  echo "<pre>";
  $first = new WorkWithFile("count.txt");
  $second = new WorkWithFile("file_new.txt");
```

```
  echo "Размер (в байтах): {$first->getSize()}<br>";
  echo "Число строк: {$first->getCount()}<br>";
  echo "{$first->getContent()}<br>";
  echo "{$second->getContent()}<br>";
  echo "</pre>";
?>
```

Конструктор так же применяется для помещения объектов других классов в качестве члена класса. Если класс содержит в качестве члена объект другого класса, то обращение к членам такого объекта также осуществляется при помощи стрелки: `$obj->object->member`. В листинге 11.6 приведен пример класса, содержащего объект другого класса.

Листинг 11.6. Помещение объекта одного класса в другой класс

```
<?php
  // Внутренний класс Inner
  class Inner
  {
    var $member = "Член member класса Inner";
    // Объявляем метод
    function method()
    {
      echo "Метод method() класса Inner";
    }
  }

  // Внешний класс Outer
  class Outer
  {
    // Конструктор
    function __construct()
    {
      $this->objt = new Inner;
    }
  }

  // Объявляем объект класса Outer
  $obj = new Outer;
  // Обратимся к члену member объекта $object класса Inner
  echo "{$obj->objt->member}<br>";
  // Обратимся к методу method() объекта $object класса Inner
  echo "{$obj->objt->method()}<br>";
?>
```

Число цепочек из стрелок -> не ограниченно, т. е. можно создавать любое количество вложенных объектов.

Примечание

Для того чтобы объявить член класса, так же как и в случае обычных переменных, необязательно обращаться к нему явно. При первом обращении к переменной при помощи конструкции "$this->*переменная*" создается новый член класса.

Создание объектов

Объекты объявляются при помощи ключевого слова new и являются экземплярами класса. Объекты являются переменными, тип и свойства которых определяются исходным классом. При работе с объектами можно создавать массив объектов, как это показано в листинге 11.7.

Листинг 11.7. Работа с массивом объектов

```
<?php
  class work
  {
    var name;
  }
  // Объявляем массив из десяти объектов класса work
  $works = array();
  for ($i = 0; $i < 10; $i++)
  {
    $works[$i] = new work();
    $works[$i]->name = "Это объект номер $i";
  }
  for ($i = 0; $i < 10; $i++)
  {
    echo $works[$i]->name;
    echo("<br>");
  }
?>
```

Объекты, как и обычные переменные, можно передавать в качестве аргумента функции. В листинге 11.8 приведен такой пример.

Листинг 11.8. Передача объекта через функцию

```
<?php
  class Brand
```

```
  {
    var $name;
  }
  function change_name($drinkobj, $name)
  {
    $drinkobj->name = $name;
  }
  $drink = new Brand;
  $drink->name = 'Сок';
  var_dump($drink->name);
  change_name($drink, 'Кофе');
  echo "Напиток: {$drink->name} \n";
?>
```

В результате выполнения скрипта из листинга 11.8 будет выведено слово "Кофе".

Объекты могут выступать не только как параметры функций, но и в качестве возвращаемых значений. Объект, возвращаемый функцией, можно присвоить другому объекту или вызвать его методы или члены непосредственно, как это показано в листинге 11.9.

Листинг 11.9. Непосредственный вызов методов класса

```
<?php
  // Объект "круг"
  class Circle
  {
    // метод, описывающий класс Circle
    function describe()
    {
      echo "Круг<br>";
    }
  }

  // Объект "квадрат"
  class Square
  {
    // метод, описывающий класс Square
    function draw()
    {
      echo "Квадрат<br>";
    }
  }
```

```
  // Функция, возвращающая объект
  function ShapeFactoryMethod($shape)
  {
    switch ($shape)
    {
      case "Circle":
        return new Circle();
      case "Square":
        return new Square();
    }
  }

  // Можно непосредственно вызвать методы объекта
  ShapeFactoryMethod("Circle")->draw();
  ShapeFactoryMethod("Square")->draw();
?>
```

Присваивание одного объекта другому тоже происходит по ссылке, т. е. после того, как один объект присвоен другому, ссылки указывают на один и тот же объект. Особенности операции присваивания продемонстрированы в листинге 11.10.

Листинг 11.10. Присваивание объектов

```
<?
  class TestClass
  {
    // Объявляем член класса
    var $member;
    // Объявляем метод класса
    public function method()
    {
      echo "Вызван метод класса.";
    }
  }

  // Объявляем два объекта класса TestClass
  $first = new TestClass;
  $second = new TestClass;
  // Члену объекта $first присвоим значение
  $first->member = 3;
  // Присвоим один объект другому
  $second = $first;
  echo $second->member."<br>";    // 3
```

```
  // Изменим значение члена member объекта $second
  $second->member = 5;
  // Член member объекта $second тоже изменил значение
  echo $first->member."<br>";   // 5
?>
```

После того как объекту `$second` был присвоен объект `$first`, значение члена `$member` стало равно 3, но изменение этого члена в объекте `$second`, так же изменяет его и в объекте `$first`. Данную особенность объектов нужно всегда учитывать, т. к. в противном случае это может стать причиной сложно-отлаживаемых ошибок. В объектно-ориентированной модели PHP существует также метод точного копирования объектов, в результате которого получаются два независимых объекта одинакового содержания, но изменения, которым подвергается один объект, не отражаются на другом объекте. Механизм такого присваивания называется *клонированием объектов*.

Клонирование объектов

Часто от объектов требуется присваивание не по ссылке, а точное копирование всех переменных из одного объекта в другой, как это происходит в случае обычных переменных. Для этого в PHP 5 было введено специальное ключевое слово `clone`, возвращающее копию объекта.

Примечание

Ранние версии PHP использовали для клонирования объекта предопределенный метод `__clone()`, который и в настоящий момент является зарезервированным, но не используется. Разработчики PHP пришли к выводу, что новый способ клонирования объектов при помощи ключевого слова `clone` более читабельный.

В листинге 11.11 показано применение данного ключевого слова.

Листинг 11.11. Клонирование объектов

```
<?php
  class TestClass
  {
    // Объявляем член класса
    public $member;
  }

  // Объявляем два объекта класса TestClass
  $first = new TestClass;
  $second = new TestClass;
```

```
  // Члену объекта $first присвоим значение
  $first->member = 3;
  // Присвоим клон объекта другому объекту
  $second = clone $first;
  echo $second->member."<br>";    // 3
  // Изменим значение члена member объекта $second
  $second->member = 5;
  // Член member объекта $first не изменил свое значение
  echo $first->member."<br>";    // 3
?>
```

В отличие от листинга 11.10, в листинге 11.11 после присваивания клона объекта `$first` объекту `$second` оба объекта ведут себя независимо, как два различных объекта.

Подсчет объектов или статических членов и методов

Для того чтобы воспользоваться членами или методами, не всегда обязательно объявлять объект класса. Объявление внутренних компонентов класса статическими при помощи ключевого слова `static` делает их доступными в любой момент без объявления объекта класса. Обращение к таким компонентам осуществляется при помощи следующей конструкции:

класс`::`*компонент*`(`*член или метод*`)`

В листинге 11.12 приведен пример простейшего класса, содержащего один статический метод и одну статическую переменную.

Листинг 11.12. Статические члены и методы класса

```
<?php
  class StaticClass
  {
    // Объявляем статический член класса
    static $member = "Это статическаий член класса<br>";
    // Объявляем статический метод класса
    static function method()
    {
      echo "Вызван статический метод класса<br>";
    }
  }
```

```
  // Вызываем статический метод класса,
  // объявление объекта не требуется.
  StaticClass::method();
  // Выводим статическую переменную в окно браузера
  echo StaticClass::$member;
?>
```

Как видно из листинга 11.12, для обращения к статическим методам и членам не требуется явное объявление объекта.

Статические члены класса удобно применять для счетчиков объектов. Статические переменные принадлежат всем объектам одновременно, т. е. изменения, производимые над статическим членом в одном объекте, распространяются на все остальные объекты. В листинге 11.13 приведен пример простейшего счетчика объектов.

Листинг 11.13. Счетчик объектов

```
<?php
  class Counter
  {
    // Объявляем статический член класса
    static $counter = 0;
    // Объявляем конструктор класса
    function __construct()
    {
      // Увеличиваем счетчик объектов
      Counter::$counter++;
    }
    function __destruct()
    {
      // Уменьшаем счетчик объектов
      Counter::$counter--;
    }
  }

  // Объявляем 5 объектов класса Counter
  $obj1 = new Counter;
  $obj2 = new Counter;
  $obj3 = new Counter;
  $obj4 = new Counter;
  $obj5 = new Counter;
  // Выводим число объектов
  echo Counter::$counter;    // 5
?>
```

Статическая переменная $counter увеличивается на единицу всякий раз, когда происходит обращение к конструктору класса, и уменьшается, если выполняется обращение к деструктору. Таким образом, можно отслеживать количество объектов в сложных системах, где непосредственный их подсчет затруднен. В листинге 11.14 приведен пример класса, осуществляющий открытие файла в конструкторе и закрытие его в деструкторе. Число одновременно открытых файлов отслеживается при помощи счетчика файлов.

Листинг 11.14. Открытие класса в конструкторе и закрытие в деструкторе

```
<?php
  class Counter
  {
    // Счетчик одновременно открытых файлов
    static $counter = 0;
    // Имя файла
    var $name;
    // Дескриптор файла
    var $file;

    // Объявляем конструктор класса
    function __construct()
    {
      // Увеличиваем счетчик открытых файлов
      Counter::$counter++;
      $this->name = Counter::$counter;
      if(Counter::$counter<=3)
      {
        $this->file = @fopen($this->name,"r");
        if(!$this->file)
        {
          echo("Ошибка открытия файла");
          $this->is_open = false;
        }
        // Флаг is_open равен true, если удалось открыть файл,
        // и false - в случае неудачи
        $this->is_open = true;
      }
      else
      {
        $this->is_open = false;
        echo "Невозможно открыть файл - исчерпан лимит <br>";
```

```
        Counter::$counter--;    // Файл не был открыт
      }
    }

    function __destruct()
    {
      // Если файл был открыт, закрываем его
      if($this->is_open)
      {
        @fclose($this->file);
      }
      // Уменьшаем счетчик открытых файлов
      Counter::$counter--;
    }
  }

  // Объявляем 5 объектов класса Counter
  $obj1 = new Counter;
  $obj2 = new Counter;
  $obj3 = new Counter;
  $obj4 = new Counter;
  $obj5 = new Counter;
  // Выводим число открытых файлов
  echo Counter::$counter;    // выводит 3
?>
```

В конструкторе поставлено ограничение на число одновременно открытых файлов, которое равно трем. Как только лимит файлов исчерпан, браузер сообщает о невозможности открытия дополнительных файлов.

Обработка исключительных ситуаций

Для реализации механизма исключений в PHP 5 введены следующие три ключевых слова: `try` (контролировать), `throw` (генерировать, бросать) и `catch` (ловить).

Служебное слово `try` позволяет выделить в любом месте скрипта так называемый *контролируемый блок*, за которым следует один или несколько блоков обработки исключений, вводимых с помощью ключевого слова `catch`:

```
try
{
  // операторы
}
```

```
catch (Exception $exp)
{
  // блок кода обработки исключительной ситуации
}
```

Обработчики всегда располагаются после контролируемого оператором `try` блока кода. Среди операторов контролируемого блока могут быть любые операторы и объявления PHP. Исключительная ситуация генерируется при помощи ключевого слова `throw`:

```
throw выражение_генерации_исключения;
```

После оператора `throw` следует объявление объекта исключения. При генерации исключения выполнение основной программы прекращается и происходит выполнение кода обработчика исключительной ситуации. Как и все объекты, объект класса исключений (`Exception`) объявляется при помощи ключевого слова `new`, при этом конструктор принимает два параметра: строку с сообщением об ошибке и код ошибки. В листинге 11.15 продемонстрирован пример обработки ошибки, генерируемой в контролируемом блоке.

Листинг 11.15. Простой пример обработки ошибки

```
<?php
  // Начало контролируемого блока
  try
  {
    // Генерируем исключение
    throw new Exception('Произошла исключительная ситуация', 9);
  }
  // Начало блока обработки исключительной ситуации
  catch (Exception $e)
  {
    echo "Исключение ". $e->getCode(). ": ". $e->getMessage()."<br>";
    echo "в файле ". $e->getFile()."<br>";
    echo "строка ".$e->getLine(). "<br>";
  }
?>
```

Если скрипт из листинга 11.15 разместить в файле test.php, то результатом работы будет сообщение:

```
Исключение 9: Произошла исключительная ситуация
в файле C:\www\html\test.php
строка 6
```

Класс исключений (Exception) содержит четыре метода:

- getCode() — возвращает код ошибки (первый параметр конструктора);
- getMessage() — возвращает строку сообщения об ошибке (второй параметр конструктора);
- getFile() — возвращает имя файла, в котором было сгенерировано исключение;
- getLine() — возвращает номер строки, в которой произошла генерация исключения.

Генерация исключений может происходить не только непосредственно в контролируемом блоке, но и во всех функциях, вызываемых в данном блоке. В листинге 11.16 приведен пример функции OutputPositiveNumber, которая принимает в качестве аргумента положительное число, а при передаче отрицательного числа генерируется исключение.

Листинг 11.16. Пример обработки исключительной ситуации

```
<?php
  function OutputPositiveNumber($number)
  {
    if($number>=0)
    {
      echo "Функции OutputPositiveNumber передано
            положительное число: $number<br>";
    }
    else
    {
      throw new Exception("Функции OutputPositiveNumber передано
                           отрицательное число: $number", 1);
    }
  }
  // Начало контролируемого блока
  try
  {
    OutputPositiveNumber(45);
    OutputPositiveNumber(-100);  // Эта строка вызовет исключение
    OutputPositiveNumber(100);
  }
  // Начало блока обработки исключительной ситуации
  catch (Exception $e)
  {
    echo "Исключение ". $e->getCode(). ": ". $e->getMessage()."<br>";
```

```
    echo "в файле ". $e->getFile()."<br>";
    echo "строка ".$e->getLine(). "<br>";
  }
?>
```

Первый вызов функции `OutputPositiveNumber` инициирует вывод числа в браузер, второй — генерацию исключения. Третьего вызова функции не произойдет — генерация исключения означает сбой, не совместимый с дальнейшим выполнением программы.

Удобство исключений заключается в том, что для однотипных исключений, которые могут быть сгенерированы в разных функциях, можно использовать единый обработчик, не дублируя код обработки в каждой из функций.

Механизм исключений можно применять также для обработки ошибочных ситуаций в классах. В листинге 11.4 в конструкторе происходит открытие файла, в листинге 11.17 продемонстрировано использование механизма исключений применительно к этому классу.

Листинг 11.17. Применение механизма исключений при работе с классами

```
<?php
  class WorkWithFile
  {
    // Буфер
    var $buff;
    // Имя файла
    var $filename;

    // Конструктор класса
    function __construct($filename)
    {
      // Имя файла помещаем в член filename
      $this->filename = $filename;
      // Проверяем существование файла
      if(!file_exists($this->filename))
        throw new Exception("файл $filename не существует", 21);
      // открытие файла
      $fd = fopen($filename, "r");
      if(!$fd) exit("Ошибка открытия файла");
      // чтение из файла
      $this->buff = fread($fd,filesize($this->filename));
      // закрытие файла
      fclose($fd);
    }
```

```
    // Метод выводит содержимое файла
    function getContent()
    {
      return $this->buff;
    }

    // Метод выводит размер файла
    function getSize()
    {
      return filesize($this->filename);
    }

    // Метод выводит число строк в файле
    function getCount()
    {
      // Если файл загружен
      if(!empty($this->filename))
      {
        $arr = file($this->filename);
        return count($arr);
      }
      else return 0;
    }
  }
?>
```

Теперь объявление объектов класса `WorkWithFile` можно поместить в контролируемый блок, и в случае передачи конструктору класса несуществующего пути работа скрипта будет остановлена с выдачей сообщения о произошедшей исключительной ситуации (листинг 11.18).

Листинг 11.18. Обработка исключения

```
<?php
  try
  {
    echo "<pre>";
    $first = new WorkWithFile("count.txt");
    $second = new WorkWithFile("file_new.txt");
    echo "Размер (в байтах): {$first->getSize()}<br>";
    echo "Число строк: {$first->getCount()}<br>";
    echo "{$first->getContent()}<br>";
```

```
    echo "{$second->getContent()}<br>";
    echo "</pre>";
  }
  catch (Exception $e)
  {
    echo "Исключение ". $e->getCode(). ": ". $e->getMessage()."<br>";
    echo "в файле ". $e->getFile()."<br>";
    echo "строка ".$e->getLine(). "<br>";
  }
?>
```

ГЛАВА 12

Работа с FTP

Протокол передачи файлов FTP (File Transfer Protocol) является одним из старейших прикладных протоколов, появившийся задолго до Web в 1971 г. До начала 90-х годов прошлого века на долю FTP приходилась половина трафика в сети Интернет. Этот протокол и по сегодняшний день используется для распространения программного обеспечения и доступа к удаленным хостам. В данной главе будет рассмотрена разработка Web-приложения, обеспечивающего доступ по протоколу HTTP к удаленному серверу FTP как с другого удаленного сервера, так и с локальной машины, на которой работает сервер Apache с поддержкой PHP.

Замечание

Протокол FTP описан в документе RFC959, доступном по адресу **http://www.faqs.org/rfcs/rfc959.html**.

Установка соединения с FTP-сервером

Соединение с сервером устанавливается при помощи функции `ftp_connect()`, имеющей следующий синтаксис:

```
resource ftp_connect(string host [, int port [, int timeout]])
```

Функция принимает в качестве первого параметра *host* адрес FTP-сервера. Через второй необязательный параметр *port* можно передать значение порта, по которому необходимо установить соединение. По умолчанию соединение устанавливается по стандартному 21 порту. При помощи третьего параметра *timeout* можно указать время в секундах, в течение которого функция будет ожидать ответа от сервера. По умолчанию, если данный параметр не задан, функция ожидает ответ 90 секунд.

В случае если установить соединение не удалось, функция возвращает `false`. При успешном соединении с FTP-сервером функция возвращает дескриптор соединения, который далее используется для работы с FTP-сервером.

После установки соединения с FTP-сервером осуществляется регистрация пользователя при помощи функции `ftp_login()`, имеющей следующий синтаксис:

```
bool ftp_login(resource link, string username, string password)
```

Функция принимает три параметра: дескриптор потока *link*, возвращаемый функцией `ftp_connect()`, имя пользователя *username* и его пароль *password*. При успешной регистрации на FTP-сервере функция возвращает `true`, в противном случае — `false`.

Закрыть FTP-соединение можно при помощи функции `ftp_close()`, которая имеет следующий синтаксис:

```
void ftp_close(resource link)
```

В качестве единственного аргумента *link* функция принимает дескриптор открытого FTP-соединения.

Замечание

Если доступ осуществляется к анонимному FTP-серверу, то в качестве имени пользователя можно ввести "anonymous", а в качестве пароля — адрес электронной почты.

Пример — в листинге 12.1.

Листинг 12.1. Соединение с FTP-сервером

```
<?php
  // Адрес FTP-сервера
  $ftp_server = "ftp.server.ru";
  // Пользователь
  $ftp_user = "user";
  // Пароль
  $ftp_password = "password";
  // Задаем время исполнения скрипта, равное 120 с
  set_time_limit(120);
  // Пытаемся установить соединение с FTP-сервером
  $link = ftp_connect($ftp_server);
  if(!$link) puterror("К сожалению, не удается установить
                соединение с FTP-сервером $ftp_server");
  // Осуществляем регистрацию на сервере
  $login = ftp_login($link, $ftp_user, $ftp_password);
```

```
  //$login = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);
  if(!$login) puterror("К сожалению, не удается зарегистрироваться на
          сервере. Проверьте регистрационные данные.");
?>
```

Для установки соединения в переменные `$ftp_server`, `$ftp_user` и `$ftp_password` необходимо поместить адрес FTP-сервера, имя пользователя и его пароль соответственно.

Замечание

Для дальнейшей работы с протоколом FTP код из листинга 12.1 удобно поместить в отдельный файл — config.php, который далее следует подключать ко всем скриптам инструкцией `include`, чтобы не переписывать код установки соединения в каждом скрипте.

FTP-сервер может работать под управлением различных операционных систем, каждая из которых имеет свои особенности. Поэтому для дальнейшей работы с ним требуется выяснить тип операционной системы, что осуществляется при помощи функции `ftp_systype()`, которая имеет следующий синтаксис:

```
string ftp_systype(resource link)
```

Функция `ftp_systype()` принимает единственный параметр *link* — дескриптор соединения с сервером, возвращаемый функцией `ftp_connect()`, и возвращает строку с типом операционной системы, например `"UNIX"` для UNIX-подобной операционной системы или `"Windows_NT"` для Windows (листинг 12.2).

Листинга 12.2. Функция `ftp_systype()`

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Выводим тип операционной системы FTP-сервера
  echo ftp_systype($link);
?>
```

Замечание

В большинстве случаев для работы с FTP-сервером достаточно функций, предоставляемых PHP, но при помощи функции `ftp_raw()` возможна отправка серверу команд FTP.

Навигация по FTP-серверу

После того как получен доступ к ресурсам FTP-сервера, можно начинать работу. Одна из первых задач, которые встают при создании Web-приложений, работающих с FTP-сервером, — перемещение по каталогу сервера и получение содержимого каталогов. Получить список файлов и подкаталогов текущего каталога можно при помощи функции `ftp_rawlist()`, имеющей следующий синтаксис:

```
array ftp_rawlist(resource link, string directory)
```

В качестве первого параметра функция принимает дескриптор FTP-соединения, а качестве второго аргумента — путь к каталогу. Функция возвращает массив строк, каждая из которых содержит информацию о подкаталоге или файле (листинг 12.3).

Листинг 12.3. Функция `ftp_rawlist()`

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Получаем все файлы корневого каталога
  $file_list = ftp_rawlist($link, "/");
  // Выводим массив $file_list
  foreach($file_list as $line) echo $line."<br>";
?>
```

Результат работы кода из листинга 12.3 может выглядеть следующим образом:

```
drwx------ 9 root root 512    Dec 27 19:07 .
drwx------ 9 root root 512    Dec 27 19:07 ..
-rw-r--r-- 1 root root 27154  Jan 6  13:30 readme.txt
-rw-r--r-- 1 root root 1144   Dec 31 11:14 licence.txt
-rw-r--r-- 1 root root 73373  Jan 6  12:23 version.txt
drwxrwx--- 2 root root 512    Sep 18 2003  db
drwxr-xr-x 2 root hosting 512 Jan 6  01:48 logs
```

Замечание

Информация, возвращаемая функцией `ftp_rawlist()`, предоставляется в формате UNIX-утилиты ls с ключом `-l`. Данная утилита возвращает содержимое текущего каталога в расширенном формате.

Первая подстрока вида "drwx------" содержит 10 символов и определяет тип файла и права доступа к нему. Первый символ характеризует тип файла:

- - — обычный файл;
- d — каталог;
- l — символическая ссылка;
- s — сокет;
- p — именованный канал.

На практике, при работе с FTP встречаются только первые два вида файлов: каталоги и обычные файлы.

Оставшиеся девять символов определяют режим доступа к файлу или каталогу. При этом подстрока разбивается на три группы, по три символа каждая:

- права владельца файла;
- права группы владельца;
- права остальных пользователей.

В рамках каждой из этих трех групп существуют три вида разрешений:

- r — право чтения данного файла;
- w — право записи/изменения данного файла;
- x — право выполнения данного файла, если он является сценарием или программой (для каталога право его просмотра).

Таким образом, строка "-rw-r--r--" может быть интерпретирована как принадлежащая обычному файлу, для владельца которого установлено право чтения и записи, а для пользователей, входящих в группу владельца файлов, и всех остальных пользователей установлено только право чтения. Строка "drwxr-xr-x" принадлежит каталогу, для владельца которого установлены все права: чтения, записи и просмотра, а для всех остальных пользователей — только чтения и просмотра.

После этого далее в строке следует число блоков в файле, группа-владелец файла или каталога, пользователь-владелец файла или каталога, размер файла в байтах, дата создания и имя файла или каталога.

Информация, возвращаемая функцией ftp_rawlist(), избыточна. Поэтому вместо нее часто используют функцию ftp_nlist(), которая возвращает массив файлов из указанного каталога без дополнительных параметров. Функция имеет следующий синтаксис:

```
array ftp_nlist(resource link, string directory)
```

В качестве параметров функция принимает дескриптор FTP-соединения *link* и имя каталога *directory*.

Для определения имени текущего каталога предназначена функция `ftp_pwd()`, которая принимает в качестве единственного параметра дескриптор FTP-соединения *link*:

```
string ftp_pwd(resource link)
```

Изменение текущего рабочего каталога на указанный осуществляется при помощи функции `ftp_chdir()` (листинг 12.4).

Листинг 12.4. Изменение текущего каталога

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Изменяем текущий каталог
  ftp_chdir($link, "logs");
?>
```

Как видно из листинга 12.4, функция `ftp_chdir()` имеет два параметра: дескриптор соединения (`$link`) и имя нового каталога (`$new_dir`).

Вернуться на уровень выше можно при помощи функции `ftp_cdup()`, которая принимает в качестве единственного параметра дескриптор FTP-соединения `link`:

```
bool ftp_cdup(resource link)
```

Функция возвращает `true` при успешной смене каталога и `false` в противном случае.

Работа с каталогами

Для создания каталога на FTP-сервере предназначена функция `ftp_mkdir()`, которая имеет следующий синтаксис:

```
string ftp_mkdir(resource link, string directory)
```

Функция принимает в качестве первого параметра дескриптор FTP-соединения *link*, а в качестве второго — имя создаваемого каталога *directory*. При успешном выполнении функция возвращает имя только что созданного каталога, при неудачном — `false`.

Пример — в листинге 12.5.

Листинг 12.5. Создание каталога

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Создаем каталог new
  ftp_mkdir($link, "new");
?>
```

Для переименования каталогов предназначена функция `ftp_rename()`. Синтаксис функции таков:

```
bool ftp_rename(resource link, string from, string to)
```

В качестве первого аргумента функция принимает дескриптор соединения с FTP-сервером, возвращаемый функцией `ftp_connect()`. Через второй аргумент *from* передается имя переименовываемого каталога. В качестве третьего параметра *to* выступает новое имя каталога. В листинге 12.6 созданный ранее (см. листинг 12.5) каталог `new` переименовывается в каталог `old`.

Листинг 12.6. Переименование каталога

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Переименовываем каталог
  ftp_rename ($link, "new", "old");
?>
```

Для удаления каталога предназначена функция `ftp_rmdir()`, имеющая следующий синтаксис:

```
bool ftp_rmdir(resource link, string directory)
```

Функция удаляет каталог *directory* и возвращает `true` в случае успеха и `false` в противном случае. Функция удаляет только пустые каталоги.

Работа с файлами

Загрузка файлов на FTP-сервер

Добавление файлов в текущий каталог осуществляется путем загрузки их на FTP-сервер со стороны клиента. Инициализация загрузки файла на FTP-сервер осуществляется функцией `ftp_nb_put()`, которая имеет следующий синтаксис:

```
int ftp_nb_put(resource link, string remote_file,
               string local_file, int mode [, int startpos])
```

В качестве первого параметра *link* функция принимает дескриптор соединения с FTP-сервером. Второй аргумент *remote_file* определяет имя и путь к файлу на удаленном FTP-сервере. Третий параметр *local_file* определяет путь к локальному файлу. Четвертый параметр *mode* указывает режим передачи информации: FTP_ASCII для текста и FTP_BINARY для бинарных файлов. Необязательный пятый параметр *startpos* позволяет задать позицию в байтах, начиная с которой следует загружать файл.

Функция возвращает одну из трех констант:

- FTP_FAILED — если не удалось передать файл на FTP-сервер;
- FTP_FINISHED — если передача файла на сервер успешно завершена;
- FTP_MOREDATA — если в настоящий момент передача данных продолжается.

Код загрузки файла на FTP-сервер приведен в листинге 12.7. Если функция возвращает константу FTP_MOREDATA, скрипт ожидает завершения загрузки файла в цикле while. Статус процесса контролируется при помощи функции ftp_nb_continue(), принимающей в качестве единственного параметра дескриптор соединения link, который возвращается функцией ftp_connect(). Функция ftp_nb_continue() возвращает те же самые константы, что и функция ftp_nb_put().

После завершения процесса загрузки файла на FTP-сервер функция ftp_nb_continue() возвращает константу FTP_FINISHED и скрипт выходит из цикла while.

Листинг 12.7. Загрузка файла на FTP-сервер

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Инициируем загрузку файла на FTP-сервер
  $ret = ftp_nb_put($link, "/path/file.zip", "C:\\file.zip", FTP_BINARY);
  // Цикл загрузки
  while ($ret == FTP_MOREDATA)
  {
    // Выводим точки, чтобы пользователь
    // знал, что процесс загрузки идет
    echo ".";
    // Продолжаем загрузку
    $ret = ftp_nb_continue($link);
  }
```

```
  if ($ret != FTP_FINISHED)
    exit ("<br>Во время загрузки файла произошла ошибка...");
?>
```

Переименование, удаление файлов на FTP-сервере

За переименование файлов на FTP-сервере несет ответственность функция `ftp_rename()`, которая имеет следующий синтаксис:

```
bool ftp_rename(resource link, string from, string to)
```

В качестве первого параметра *link* функция, как и большинство FTP-функций, принимает дескриптор FTP-соединения, вторым параметром указывается путь к файлу, который подвергается переименованию, последний параметр функции указывает новое имя файла.

Удаление файла осуществляется при помощи функции `ftp_delete()`, которая принимает в качестве первого аргумента дескриптор соединения с FTP-сервером, а в качестве второго — путь к удаляемому файлу (листинг 12.8). В случае успешного удаления файла функция возвращает `true`, иначе — `false`.

Листинг 12.8. Удаление файла с FTP-сервера

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Удаление файла
  ftp_delete ($link, "/path/file.zip");
?>
```

Загрузка файлов с FTP-сервера

Загрузка файлов с FTP-сервера осуществляется функцией `ftp_nb_get()`, которая имеет следующий синтаксис:

```
int ftp_nb_get(resource link, string local_file,
               string remote_file, int mode [, int resumepos])
```

В качестве первого параметра *link* функция принимает дескриптор соединения с FTP-сервером. Второй аргумент *local_file* определяет путь к локальному файлу. Третий параметр *remote_file* указывает имя и путь к файлу на удаленном FTP-сервере. Четвертый параметр *mode* определяет режим переда-

чи информации: FTP_ASCII для текста и FTP_BINARY для бинарных файлов. Необязательный пятый параметр startpos позволяет задать позицию в байтах, начиная с которой следует загружать файл.

Функция возвращает одну из трех констант:

- FTP_FAILED — если не удалось загрузить файл с FTP-сервера;
- FTP_FINISHED — если загрузка файла с сервера успешно завершена;
- FTP_MOREDATA — если в настоящий момент передача данных продолжается.

Точно так же, как и в листинге 12.7, контроль за состоянием загрузки файла осуществляется в цикле while при помощи рассмотренной выше функции ftp_nb_continue() — листинг 12.9.

Листинг 12.9. Загрузка файлов с FTP-сервера

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Инициируем загрузку файла с FTP-сервера
  $ret = ftp_nb_get($link, "C:\\file.zip", "/path/file.zip", FTP_BINARY);
  // Цикл загрузки
  while ($ret == FTP_MOREDATA)
  {
    // Выводим точки, чтобы пользователь
    // видел, что процесс идет
    echo ".";
    // Продолжаем загрузку
    $ret = ftp_nb_continue($link);
  }
  // Если происходит ошибка при загрузке файла,
  // уведомляем об этом пользователя
  if ($ret != FTP_FINISHED)
  {
    echo "<br>Во время загрузки файла произошла ошибка...";
    exit();
  }
?>
```

Работа с правами доступа

Помимо рассмотренной в *разд. "Навигация по FTP-серверу" ранее в этой главе* текстовой формы для задания прав доступа в операционной системе UNIX, права доступа можно задавать восьмеричным числом. При этом праву чтения

соответствует цифра 4, праву записи — 2, а исполнению — 1. Общие права для групп задаются суммой этих чисел. Так значение 6 (4 + 2) обеспечивает возможность чтения и записи, а значение 7 (4 + 2 + 1) предоставляет полный доступ к файлу или каталогу. Тогда для каталога восьмеричное число 0755 означает, что владелец каталога имеет полный доступ (`rwx`), а все остальные — право читать файлы в нем и просматривать содержимое каталога (`r-x`).

Замечание

В PHP восьмеричные числа предваряются нулем.

Права доступа для файлов и каталогов устанавливаются при помощи функции `ftp_chmod()`, которая имеет следующий синтаксис:

```
string ftp_chmod(resource link, int mode, string directory)
```

В качестве первого аргумента *link* эта функция, так же как и большинство других функций для работы с FTP, принимает дескриптор соединения, возвращаемый функцией `ftp_connect()`. Второй параметр *mode* принимает восьмеричное число, задающее права доступа к каталогу. Последний параметр *directory* определяет имя каталога, права доступа к которому изменяются. Функция возвращает только что установленные права в случае успеха и `false` в противном случае.

Пример — в листинге 12.10.

Листинг 12.10. Установка прав доступа к каталогам и файлам

```
<?php
  // Устанавливаем соединение с FTP-сервером
  require_once("config.php");
  // Устанавливаем права доступа к каталогу
  ftp_chmod($link, 0755, "logs");
  // Устанавливаем права доступа к файлу
  ftp_chmod($link, 0644, "/path/file.zip");
?>
```

ГЛАВА 13

Примеры работы с сетевыми протоколами

В этой главе мы рассмотрим различные примеры, в основе которых лежит подключение к удаленному серверу с помощью функций для работы с сокетами. *Сокет* — библиотека функций, реализуемая операционной системой для установки сетевых соединений.

Подключение к удаленному серверу

Подключение к удаленному серверу осуществляется с помощью функции `fsockopen()`:

```
resource fsockopen(string target, int port [, int errno
                   [, string errstr [, float timeout]]])
```

Функция принимает пять параметров, первый из которых содержит адрес соединения, а второй — порт. Если при этом функции переданы два необязательных параметра `errno` и `errstr`, в них размещается код и текстовое описание ошибки, соответственно. Пятый необязательный параметр — значение тайм-аута, определяющего максимальное время (в секундах) ожидания соединения. При успешной установке соединения функция возвращает дескриптор соединения, при неудаче — `false`.

Примечание

PHP предоставляет немало функций для работы с сокетами, однако рассмотрение их всех, во-первых, не входит в задачи этой книги, а во-вторых, для большинства приложений, с которыми приходится сталкиваться Web-разработчику, вполне достаточно обращений к сокетам с помощью функции `fsockopen()`.

Пример работы с функцией `fsockopen()` приведен в листинге 13.1, где происходит загрузка главной страницы портала **http://www.mail.ru**.

Листинг 13.1. Пример использования функции `fsockopen()`

```
<?php
  function get_content($hostname, $path)
  {
    $line = "";
    // Устанавливаем соединение, имя которого
    // передано в параметре $hostname
    $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
    // Проверяем успешность установки соединения
    if (!$fp) echo "$errstr ($errno)<br />\n";
    else
    {
      // Формируем HTTP-запрос для передачи его серверу
      $headers = "GET $path HTTP/1.1\r\n";
      $headers .= "Host: $hostname\r\n";
      $headers .= "Connection: Close\r\n\r\n";
      // Отправляем HTTP-запрос серверу
      fwrite($fp, $headers);
      // Получаем ответ
      while (!feof($fp))
      {
        $line .= fgets($fp, 1024);
      }
      fclose($fp);
    }
    return $line;
  }
  $hostname = "www.mail.ru";
  $path = "/";
  // Устанавливаем большее время работы скрипта:
  // пока вся страница не будет загружена, она не будет отображена
  set_time_limit(180);
  // Вызываем функцию
  echo get_content($hostname, $path);
?>
```

При работе с сокетами мы вынуждены брать на себя всю черновую работу по отправке серверу HTTP-запросов, получению и обработке ответов на них. В скрипте обращение к функции `fsockopen()` оформлено в виде функции `get_content()`, которая получает два параметра: имя сервера `$hostname`, с которым устанавливается соединение, и путь `$path` к странице относительно имени сервера.

Замечание

Следует помнить, что имя сервера не содержит префикса **http://**, который необходим для сообщения браузерам типа протокола, по которому следует работать с сервером.

После установки соединения с сервером в переменной `$headers` формируется HTTP-запрос серверу по методу `GET`. Если подставить значения всех переменных, то серверу отправляются следующие заголовки (листинг 13.2).

Листинг 13.2. Заголовки, отправляемые серверу www.mail.ru

```
GET / HTTP/1.1\r\n
Host: www.mail.ru\r\n
Connection: Close\r\n\r\n
```

После этого из сокета производится чтение ответа функцией `fgets()` блоками по 1024 байта до тех пор, пока не встретится символ конца файла, определяемый функцией `feof()`. Затем результат возвращается и выводится в окно браузера.

В первой строке листинга 13.2 у сервера запрашивается индексный файл корневого каталога сервера (/) методом `GET` по протоколу HTTP/1.1. Вторая строка сообщает адрес сервера, а третья требует от сервера закрыть соединение после передачи данных.

Замечание

Вместо символа / можно передать адрес страницы на сервере, например, `/all/index.php?r=1`. В результате чего будет загружена другая страница сайта — именно так и действуют браузеры. При переходе пользователей по ссылке они извлекают часть адреса после доменного имени и передают HTTP-запрос, подставив эту часть после ключевого слова `GET`.

Три строки из листинга 13.2 эквивалентны запросу в окне браузера **http://www.mail.ru/**. Результат работы скрипта в листинге 13.1 приведен на рис. 13.1.

В начале страницы идут заголовки, которые браузер скрывает от пользователя, работа с ними будет рассмотрена в следующем разделе. Потом начинается тело страницы, которое интерпретируется браузером — обильная реклама, которая есть на **mail.ru**, подгружена уже браузером, т. к. скрипт из листинга 13.1 загружает только HTML-код, на загрузку каждого изображения нужно открывать новое сетевое соединение, как того требует протокол HTTP. За нас это делает сам браузер, который разбирает текст HTML-страницы, извлекает адреса всех изображений, открывает для каждого изображения новое соеди-

нение и загружает по нему изображение примерно так, как это сделано в листинге 13.1.

Рис. 13.1. Просмотр портала **mail.ru** с локального хоста

Замечание

Можно передавать данные серверу не только по методу `GET`, но и по методу `POST`, что будет продемонстрировано в *разд. "Отправление данных методом POST" далее в этой главе.*

Извлечение заголовков HTTP-ответа

Часто при работе с данными удаленного хоста не требуется тело ответа, представляющее собой объемную HTML-страницу или файл. Достаточно загрузить лишь заголовки. Модифицируем функцию из листинга 13.1 для получения лишь заголовков HTTP-ответа (листинг 13.3).

Листинг 13.3. Загружаем только заголовки HTTP-ответа

```
<?php
  // А это сама функция выяснения размера
  function get_content($hostname, $path)
  {
    $line = "";
    // Устанавливаем соединение, имя которого
    // передано в параметре $hostname
    $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
```

```
    // Проверяем успешность установки соединения
    if (!$fp) echo "$errstr ($errno)<br />\n";
    else
    {
      // Формируем HTTP-запрос для передачи его серверу
      $headers = "GET $path HTTP/1.1\r\n";
      $headers .= "Host: $hostname\r\n";
      $headers .= "Connection: Close\r\n\r\n";
      // Отправляем HTTP-запрос серверу
      fwrite($fp, $headers);
      $end = $false;
      // Получаем ответ
      while (!$end)
      {
        $line = fgets($fp, 1024);
        if (trim($line) == "") $end = true;
        else $out[] = $line;
      }
      fclose($fp);
    }
    return $out;
  }
  $hostname = "www.mail.ru";
  $path = "/";
  // Устанавливаем большее время работы скрипта:
  // пока вся страница не загрузится, она не будет отображена
  set_time_limit(180);
  // Вызываем функцию
  $out = get_content($hostname, $path);
  // Выводим содержимое массива
  print_r($out);
?>
```

Результат работы функции может выглядеть следующим образом:

```
Array
(
    [0] => HTTP/1.1 200 OK
    [1] => Date: Sat, 08 Jan 2005 23:16:16 GMT
    [2] => Server: Apache/1.3.31 (Unix) mod_deflate/1.0.21 rus/PL30.20
    [3] => Set-Cookie: Mpopl=594605484; expires=Sat, 08 Jan 2005 23:31:16
GMT; path=/; domain=.mail.ru
    [4] => Set-Cookie: mrcu=5BDE41E069C01AD9775FCF4DD052; expires=Mon, 08
Jan 2007 23:16:16 GMT; path=/; domain=.mail.ru
```

```
    [5] => Connection: close
    [6] => Transfer-Encoding: chunked
    [7] => Content-Type: text/html; charset=windows-1251
    [8] => Vary: user-agent
)
```

Из ответов сервера можно почерпнуть массу интереснейших сведений. Первая строка является стандартным ответом сервера, сообщающим, что запрос успешно обработан (код ответа 200). Если запрашиваемый ресурс не будет существовать, то будет возвращен код ответа 404 (`HTTP/1.1 404 Not Found`).

Второй заголовок сообщает время формирования документа на сервере, необходимое для механизма кэширования, который рассматривается ниже.

Третий заголовок сообщает тип и версию Web-сервера.

Четвертый и пятый HTTP-заголовки просят браузер установить cookie, первый из которых с именем `Mpopl` имеет значение `594605484` и действует 20 минут, для домена `.mail.ru`. Второй cookie является сессионным и устанавливает без времени жизни переменную `mrcu` со значением `5BDE41E069C01AD9775FCF4DD052`, которое является, скорее всего, хеш-кодом MD5 для домена `.mail.ru`. Их установка при помощи функции `setcookie()` может выглядеть так, как представлено в листинге 13.4.

Листинг 13.4. Установка cookies

```
<?php
  setcookie("Mpopl", 594605484, time() + 60*20, "/", ".mail.ru");
  setcookie("mrcu", "5BDE41E069C01AD9775FCF4DD052", time(), "/",
            ".mail.ru");
?>
```

Шестой заголовок передает клиенту просьбу закрыть соединение после получения ответа.

Седьмой заголовок `Transfer-Encoding` (имеющий значение `chuncked`) указывает получателю, что ответ разбит на фрагменты.

Восьмой заголовок `Content-Type` указывает тип загружаемого документа (`text/html`) и его кодировку (`charset=windows-1251`).

Девятый заголовок `Vary` используется в качестве ключа при сохранении и извлечении ответа из кэша и предназначен для поддержания механизма кэширования.

Определение размера файла на удаленном хосте

Одной из распространенных задач является определение размера файла на удаленном хосте. Для этого воспользуемся функцией `get_content()` из листинга 13.3, с помощью которой узнаем число байтов в одном из самых популярных zip-архивах RuNet на момент написания книги (листинг 13.5).

Листинг 13.5. Определение размера файла на удаленном хосте

```
<?php
  $hostname = "book-world.narod.ru";
  $path = "/detectiv/marinina24.zip";
  // Вызываем функцию
  $out = get_content($hostname, $path);
  // Объединяем содержимое массива в одну строку
  $lines = implode(" ",$out);
  // Определяем число байтов в закачиваемом файле
  // по регулярному выражению
  preg_match("|Content-Length:[\s]+([\d]+)|i", $lines, $matches);
  // Выводим результат
  echo "Число байтов в самом популярном zip-архиве - ".$matches[1];
?>
```

Результат работы функции

```
Число байтов в самом популярном zip-архиве — 841602
```

Размер может помочь сориентироваться при загрузке файла с удаленного хоста, хватит ли отведенных PHP-скрипту 30 секунд для загрузки файла или нет.

Отправление данных методом *POST*

Создадим простейшую форму, состоящую из одного текстового поля `name` и кнопки для отправки данных (листинг 13.6).

Листинг 13.6. HTML-форма

```
<form action=handler.php method=post>
  Имя: <input type=text name=name><br>
  Пароль: <input type=text name=pass><br>
  <input type=submit name=send value=Отправить>
</form>
```

После заполнения текстового поля и нажатия кнопки **Отправить** методом POST данные отправляются обработчику handler.php, код которого представлен в листинге 13.7.

Листинг 13.7. Обработчик handler.php

```
<?php
  echo "Имя - ".$_POST['name'];
  echo "<br>Пароль - ".$_POST['pass'];
?>
```

Обработчик выводит в окно браузера текст, введенный в текстовое поле name в HTML-форме. В листинге 13.8 представлен код, который отправляет данные обработчику handler.php в обход HTML-формы.

Листинг 13.8. Отправка данных методом POST через сокеты

```
<?php
  $hostname = "localhost";
  $path = "/test2/handler.php";
  $line = "";
  // Устанавливаем соединение, имя которого
  // передано в параметре $hostname
  $fp = fsockopen($hostname, 80, $errno, $errstr, 30);
  // Проверяем успешность установки соединения
  if (!$fp) echo "$errstr ($errno)<br />\n";
  else
  {
    // Данные HTTP-запроса
    $data =
      "name=".urlencode("Игорь")."&pass=".urlencode("пароль")."\r\n\r\n";
    // Заголовок HTTP-запроса
    $headers = "POST $path HTTP/1.1\r\n";
    $headers .= "Host: $hostname\r\n";
    $headers .= "Content-type: application/x-www-form-urlencoded\r\n";
    $headers .= "Content-Length: ".strlen($data)."\r\n\r\n";
    // Отправляем HTTP-запрос серверу
    fwrite($fp, $headers.$data);
    // Получаем ответ
    while (!feof($fp))
    {
      $line .= fgets($fp, 1024);
    }
```

```
    fclose($fp);
  }
  echo $line;
?>
```

Результатом работы скрипта будет строка

```
Игорь
пароль
```

так, как если бы мы ввели это имя в HTML-форме в листинге 13.6.

Замечание

При отправке данных методом `POST` обязательно следует отправлять заголовок `Content-Length`, в качестве значения которого передается число байтов в данных `POST`-запроса, идущих после заголовков, иначе сервер не будет знать, когда следует ожидать конца поступления данных.

Замечание

При отправке данных методом `POST` переменные отделяются друг от друга символом `&`.

Создание Whois-сервиса

Рассмотрим разработку Web-приложения, позволяющего посетителю узнать, на кого и где зарегистрирован какой-либо IP-адрес, при помощи сервиса Whois. Для этого мы с помощью функции `fsockopen()` установим соединение с сервером **whois.ripe.net**, предоставляющим подобный сервис. Код приложения приведен в листинге 13.9.

Замечание

Сервис Whois (от англ. *Who is?*) является официальным сервисом центра RIPE, предоставляющим информацию о зарегистрированных на данный момент доменных именах, о DNS-серверах, поддерживающих IP-адреса, и о владельцах IP-адресов. Когда мы на различных сайтах проверяем, существует ли на данный момент какое-либо доменное имя, то пользуемся именно сервисом Whois. RIPE (Reseaux IP Europeens) — это Европейская континентальная сеть TCP/IP. Первоначально все IP-адреса, которые мы используем в RuNet, получают в RIPE Network Coordination Centre (Координационный центр RIPE), который является одним из трех региональных интернет-регистраторов, поддерживающих глобальную инфраструктуру Интернета. Сеть RIPE управляется сетью EUnet. Последняя представляет собой сеть UNIX для Европы и является старейшим и крупнейшим поставщиком услуг Интернета в Европе, частью которого является российское АО "РЕЛКОМ", являющееся крупнейшим поставщиком услуг Интернета в России.

Листинг 13.9. Whois-сервис

```
<html>
  <body>
    <center>
    <form action=ip.php method=post>
      <br>
      </i></b></font>
      <br>
      <INPUT type=text name=ip size=35>
      <input type=submit value='Введите IP-адрес'>
    </form>
    </center>
    <?php
      if(!empty($_POST["ip"]))
      {
        // Соединение с сокетом TCP, ожидающим на сервере
        // whois.ripe.net по 43 порту.
        // В результате возвращается дескриптор соединения $sock.
        $sock = fsockopen("whois.ripe.net", 43, $errno, $errstr);
        if (!$sock) exit("$errno($errstr)");
        else
        {
          // Записываем строку из переменной $_POST["ip"]
          // в дескриптор сокета.
          fputs ($sock, $_POST["ip"]."\r\n");
          // Осуществляем чтение из дескриптора сокета.
          while (!feof($sock))
          {
            echo (str_replace
                  (":",":      ",
                   fgets ($sock, 128))."<br>");
          }
        }
        // Закрываем соединение
        fclose($sock);
      }
    ?>
  </body>
</html>
```

Как видно из листинга 13.9, скрипт содержит как саму форму для заполнения, так и ее обработчик. После передачи обработчику IP-адреса происходит от-

крытие сокета при помощи функции `fsockopen()`. В случае если в базе Whois-сервиса, к которому мы обращаемся, присутствует информация о запрашиваемом IP-адресе, мы получим стандартный ответ, увидеть который можно, запустив этот скрипт.

Однако если возникнет желание узнать, кому принадлежит IP-адрес 65.54.188.74 (принадлежащий Microsoft Corporation) с помощью этого скрипта, то он потерпит неудачу, поскольку данный диапазон IP-адресов не обслуживается сервисом **whois.ripe.net**. Закономерен вопрос: каким же образом эту информацию предоставляют регистраторы доменных имен? Дело в том, что существует много различных Whois-сервисов, отвечающих за определенный диапазон IP. К примеру, за обсуждаемый нами IP-адрес ответственен сервис **whois.arin.net**, обратившись к которому можно узнавать информацию об IP-адресах, лежащих в диапазоне 65.52.0.0 — 65.55.255.255. А вот получить с помощью **arin.net** информацию о каком-либо нижегородском IP-адресе уже не получится — не его диапазон. Однако вышесказанное не означает, что для того чтобы получать информацию об IP-адресах из любого диапазона, необходимо перебирать наугад все известные нам Whois-серверы. Все не так печально. Если Whois-сервер, к которому происходит обращение, не может предоставить информацию по данному IP-адресу, то он в своем отрицательном ответе указывает, какой именно Whois-сервер ответственен за диапазон IP-адресов, к которому принадлежит запрашиваемый IP-адрес (листинг 13.10).

Листинг 13.10. Отрицательный ответ Whois-сервиса

```
ReferralServer: whois: //whois.ripe.net: 43
NetRange:       62.0.0.0 - 62.255.255.255
CIDR:           62.0.0.0/8
NetName:        RIPE-C3
NetHandle:      NET-62-0-0-0-1
Parent:
NetType:       Allocated to RIPE NCC
NameServer:       NS-PRI.RIPE.NET
NameServer:       SEC1.APNIC.NET
NameServer:       SEC3.APNIC.NET
NameServer:       NS2.NIC.FR
NameServer:       SUNIC.SUNET.SE
NameServer:       AUTH03.NS.UU.NET
NameServer:       TINNIE.ARIN.NET
Comment:       These addresses have been further assigned to users in
Comment:       the RIPE NCC region. Contact information can be found in
Comment:       the RIPE database at http:       //www.ripe.net/whois
```

Обращать внимание при анализе отрицательного ответа Whois-сервиса надо на строку

```
ReferralServer: whois: //whois.ripe.net: 43
```

В этой строке указан справочный сервер и порт, к которому необходимо обращаться для того, чтобы получить информацию о нужном IP-адресе.

О региональных Whois-сервисах

> Региональные доменные имена третьего уровня, как правило, обслуживают региональные Whois-сервисы. К примеру, доменные имена вида **spb.ru**, **msk.ru** на момент написания книги находятся на обслуживании ООО "Релком. ДС", и получить информацию о них можно, обратившись к Whois-сервису **whois.relcom.ru**.

PHP и DNS

Наиболее часто возникает задача получения имени хоста по его IP-адресу, другие задачи встречаются значительно реже. Рассмотрим функции, которые предоставляет PHP для работы с DNS.

Функция `gethostbyname()`, имеющая синтаксис:

```
string gethostbyname(string hostname)
```

принимает в качестве аргумента строку, представляющую доменное имя компьютера, и возвращает соответствующий IP-адрес (листинг 13.11).

Листинг 13.11. Получение IP-адреса по доменному имени

```
<?php
  $hostname = "localhost";
  $ip_address = gethostbyname($hostname);
  echo ("IP-адрес $hostname: $ip_address");
?>
```

Примечание

> Конечно, если вместо `localhost` подставить какое-либо другое имя и запустить этот скрипт на локальной машине, то мы ничего не добьемся. Работа с DNS должна происходить именно на том сервере, который имеет доступ к реальным DNS-серверам.

Многие компьютеры имеют несколько IP-адресов, особенно типична такая ситуация для различных серверов. Получить полный список IP-адресов, соответствующих данному имени компьютера, можно с помощью функции `gethostbynamel()`, действующей аналогично функции `gethostbyname()`. Другая ситуация, в которой полезно применение этой функции, возникает, когда од-

но имя DNS соответствует нескольким компьютерам. Это бывает при работе с DNS-серверами, поддерживающими *механизм кругового распределения нагрузки*, при котором одно имя DNS-сервера отображается на несколько компьютеров в локальной сети этого сервера.

Функция `gethostbynamel()` имеет следующий синтаксис:

```
string gethostbynamel(string hostname)
```

Возвращаемый список IP-адресов функция `gethostbynamel()` помещает в массив (листинг 13.12).

Листинг 13.12. Получение списка IP-адресов

```
<?php
  $hostname = "localhost";
  $ip_addresses = gethostbyname($hostname);
  echo("IP-адрес '$hostName': <br>\n");
  foreach($ip_adresses as $index => $val)
  {
    echo("$val");
  }
?>
```

Функция `gethostbyaddr()`, имеющая синтаксис:

```
string gethostbyaddr(string ip_address)
```

принимает в качестве аргумента IP-адрес *ip_address* и возвращает соответствующее ему имя хоста (листинг 13.13).

Листинг 13.13. Определение имени хоста

```
<?php
  $ip_address = "127.0.0.1";
  $hostname = gethostbyaddr ($ip_address);
  echo ("Имя хоста с IP-адресом $ip_address: $hostname");
?>
```

DNS-сервер сохраняет немало полезной информации о хосте, для чего используются так называемые *записи ресурсов*, которые имеют следующие типы:

- `A` — запись содержит IP-адрес хоста;
- `CNAME` — запись содержит псевдоним хоста;

- NS — запись содержит имя DNS-сервера, являющегося авторитетным для поддомена. Авторитетным является такой DNS-сервер, который замыкает цепочку DNS-запросов;
- MX — запись содержит имя хоста почтового ретранслятора в домене, которому принадлежит данный хост. В этой записи также хранится значение коэффициента предпочтения для почтового ретранслятора;
- PTR — запись хранит отображения IP-адресов в имена.

Функция `checkdnsrr()` находит на DNS-сервере записи ресурсов вида *type* для хоста *hostname*:

```
string checkdnsrr(string hostname [, string type])
```

Примечание

Эта функция не поддерживается на Windows-платформах.

Функция `getprotobyname()` принимает в качестве аргумента имя протокола *name* и возвращает номер протокола TCP/IP, соответствующее заданному имени:

```
int getprotobyname(string name)
```

Функция `getprotobynumber()` решает задачу, обратную функции `getprotobyname()`: по номеру протокола *number* она возвращает его имя:

```
string getprotobynumber(int number)
```

Функция `getservbyname()` принимает в качестве аргумента имя службы *service* и один из транспортных протоколов *protocol* (TCP или UDP) и возвращает номер порта, через который работает эта служба:

```
string getservbyname(string service, string protocol)
```

Пример — в листинге 13.14.

Листинг 13.14. Определение номера порта

```
<?php
  $service = "pop";
  $protocol = "tcp";
  $port = getservbyname ($service, $protocol);
  echo("Сервису $service отвечает порт $port");
?>
```

Функция `getservbyport()` выполняет задачу, обратную функции `getservbyname()`: возвращает имя службы, соответствующее переданному ей номеру порта *port* и транспортному протоколу *protocol*.

```
string getservbyport(int port, string protocol)
```

Пример — в листинге 13.15.

Листинг 13.15. Определение сервиса

```
<?php
  $port = 25;
  $protocol = "tcp";
  $service = getservbyport ($port, $protocol);
  echo("Через порт $port работает служба $service ");
?>
```

Подробно о кэшировании

Механизм кэширования применяется с целью оптимизации пересылки данных между клиентом и сервером. Запрошенный по HTTP-протоколу пользователем документ, может быть сохранен в кэше промежуточного сервера и при повторном его запросе будет выдаваться посетителю без обращения к источнику. PHP-скрипт, использующий заголовки, взаимодействует с браузером посредством отправки соответствующих статус-кодов, о которых говорилось в *разд. "Извлечение заголовков HTTP-ответа" ранее в этой главе*.

Кэши принято подразделять на два вида: локальные и глобальные. Локальный кэш — это кэш, создаваемый браузером клиента, а глобальным называется кэш, расположенный на proxy-сервере провайдера (или организации, в которой имеется свой внутренний proxy-сервер).

Замечание

Proxy-сервером, в отличие от обычного сервера, предоставляющего доступ к какому-либо ресурсу, называют сервер-посредник, расположенный между клиентом и обычным сервером. В отличие от шлюз-сервера, осуществляющего обычное транслирование запросов от клиента и ответов сервера, в задачи proxy-сервера входит предоставление дополнительных услуг, таких как преобразование медиатипа, анонимная фильтрация, сжатие протокола, кэширование и др.

В большинстве случаев кэширование позволяет ускорить работу с Web и значительно снизить трафик в сети, но иногда кэширование может мешать работе Web-приложений. Если посетители часто загружают страницы Web-сайта с редко изменяющейся информацией, к примеру, расписание пригородного транспорта на весенне-летний период, в этом случае кэширование полезно, поскольку экономит трафик и уменьшает время запроса. Когда пользователь загружает динамические страницы, например, активного форума, кэширование, без сомнения, вредно, поскольку содержимое таких страниц меняется слишком часто.

Рассмотрим некоторые наиболее часто встречающиеся ситуации. Как правило, в основном возникает задача запрета на кэширование Web-страниц. Однако при этом необходимо учитывать ряд моментов:

- никогда не сохраняются в кэше данные, передаваемые методом `POST`;
- страницы, обращение к которым происходит по URL, содержащим параметр в запросе (`index.php?id=4`), по умолчанию в кэше не сохраняются, т. е. кэширование таких страниц происходит только при специальном указании, которое рассмотрено далее.

Таким образом, остается не так много случаев, когда необходим принудительный запрет на кэширование. В случае, когда это действительно необходимо, решить эту задачу можно путем помещения в код одного из заголовков, приведенных в листинге 13.16.

Листинг 13.16. Запрет на кэширование

```
<?php
  // любая дата в прошлом
  header("Expires: Mon, 23 May 1995 02:00:00 GMT");
  header("Last-Modified: ".gmdate("D, d M Y H:i:s")." GMT");
  header("Cache-Control: no-cache, must-revalidate");
  header("Pragma: no-cache");
?>
```

Поле `Expires` задает дату, по достижении которой документ считается устаревшим. Поэтому задание для этого заголовка уже прошедшей даты предотвращает кэширование данной страницы.

Поле `Last-Modified` определяет дату последнего изменения Web-страницы. Если с момента последнего обращения к ней proxy-сервера значение этого заголовка изменилось, происходит повторная загрузка страницы. Поэтому приравнивание данной директиве текущего времени при каждом обращении запрещает кэширование.

Замечание

Указание этого поля актуально только для статических Web-страниц, т. к. при формировании динамических страниц, например сгенерированных при помощи PHP, происходит автоматическое внесение изменения заголовка `Last-Modified` для динамически сформированного документа.

Поле `Cache-Control` предназначено для управления кэшированием, и указание его значения, равным `no-cache`, также приводит к запрету кэширования. В устаревшем стандарте HTTP/1.0 для запрета кэширования нужно присвоить значение `no-cache` полю `Pragma`.

Приведенный в листинге 13.16 код предусматривает все возможные варианты запрета кэширования. Как правило, такая избыточность не нужна, и в большинстве случаев достаточно отправки только первого из заголовков:

```
header("Expires: Mon, 23 May 1995 02:00:00 GMT");
```

Теперь рассмотрим случаи, когда кэширование может быть полезно. Допустим, часть страниц Web-сайта меняется очень редко и их уместно сохранять в кэше. Это можно сделать при помощи заголовков, приведенных в листинге 13.17.

Листинг 13.17. Кэширование документов

```
<?php
  header("Cache-control: public");
  header("Cache-control: private");
?>
```

Атрибут `public` применяется в тех случаях, когда необходимо разрешить кэширование на proxy-серверах. Атрибут `private` указывает, что информацию нельзя хранить на публичных серверах, и ее необходимо сохранять только в локальном кэше браузера пользователя.

Можно поступать более конструктивно. Допустим, на сайте раз в час происходит обновление новостей, соответственно до наступления нового часа эти данные можно хранить в кэше (листинг 3.18).

Листинг 13.18. Кэширование документов раз в час

```
<?php
  header("Cache-control: public");
  header("Cache-control: max-age=3600");
?>
```

Аналогично можно поступать с данными, обновляющимися раз в неделю и т. д. Необходимо выделить временны́е интервалы, в течение которых содержимое страницы остается постоянным.

Замечание

Ряд предкомпилированных серверов, например русский Apache, по умолчанию снабжают все страницы запретами на кэширование.

Более совершенный способ управления кэшем основан на анализе предоставляемых данных. Этот прием удобно использовать при выдаче изображений,

больших документов, т. е. всех данных, которые изменяются очень редко или не изменяются вообще.

Пусть при обращении по адресу **http://www.softtime.ru/dic.php?id=25** посетителю отправляется изображение, которое извлекается из базы данных. Тогда скрипт, сохраняющий изображение в кэше, может выглядеть так, как представлено в листинге 13.19.

Листинг 13.19. Сохранение изображения в кэше

```
<?php
  // Адрес сервера MySQL
  $dblocation = "localhost";
  // Пользователь
  $dbuser = "user";
  // Пароль
  $dbpasswd = "password";
  // Осуществляем соединение с базой данных
  @mysql_connect($dblocation,$dbuser,$dbpasswd);
  // Осуществляем запрос на извлечение картинки из таблицы documents
  $doc = mysql_query("dbase",
                     "SELECT docs, type FROM documents WHERE id=$id");
  // Отправляем HTTP-заголовки
  header("Cache-Control: public, must-revalidate");
  header("Vary: Content-ID");
  header("Content-ID: ".md5(mysql_result($doc, 0, "docs")));
  header("Content-type: ".mysql_result($doc, 0, "type"));
  // Выводим картинку в окно браузера
  echo mysql_result($doc, 0, "docs");
  // Закрываем соединение с базой данных
  mysql_close();
?>
```

В листинге 13.19 вычисляется хеш-код извлекаемого из базы данных документа по алгоритму MD5. Полученное число передается в качестве значения заголовка `Content-ID` — пока содержимое документа не меняется, контрольная сумма будет постоянной. Как только значение хеш-кода изменится, это послужит сигналом для proxy-сервера о том, что документ в кэше потерял актуальность и его необходимо обновить.

Кроме рассмотренных способов управления кэшированием с использованием отправки заголовков, PHP 5 предоставляет еще несколько функций, с помощью которых можно управлять кэшированием. Мы рассмотрим две основные из них.

Функции *session_cache_limiter()* и *session_cache_expire()*

Функция `session_cache_limiter()` возвращает и/или устанавливает ограничитель текущего кэша и имеет следующий синтаксис:

```
string session_cache_limiter([string cache_limiter])
```

Если определен необязательный параметр *cache_limiter*, то текущий ограничитель меняется на новое значение.

Ограничитель кэша контролирует заголовки HTTP, посланные клиенту. Выше говорилось, что установка ограничителя кэша в `no-cache` позволяет отключить клиентское кэширование. Значение `public`, наоборот, установит постоянное кэширование. Так же, как и при использовании соответствующего заголовка, параметр может принимать значение `private`.

Замечание

В режиме `private` заголовок `Expire`, посланный клиенту, может вызвать неоднозначность в браузерах, построенных на основе Mozilla. Этой проблемы можно избежать, используя вместо `private` режим `private_no_expire`. В таком режиме заголовок `Expire` никогда не будет посылаться клиенту.

Ограничитель кэша восстанавливается в значение по умолчанию, хранимое в `session.cache_limiter`, в начале запроса. Из этого следует, что нужно вызывать `session_cache_limiter` для каждого запроса (и до вызова функции `session_start()`).

Рассмотрим пример, использующий эту функцию, в котором ограничитель кэша устанавливается в режим `public` (листинг 13.20).

Листинг 13.20. Использование функции `session_cache_limiter()`

```
<?php
  // Устанавливаем ограничитель кэша в 'public'
  session_cache_limiter('public');
  $cache_limiter = session_cache_limiter();
?>
```

Функция `session_cache_expire()` возвращает текущее время жизни кэша и имеет следующий синтаксис:

```
int session_cache_expire([int new_cache_expire])
```

Необязательный параметр *new_cache_expire* определяет время жизни кэша в минутах (по умолчанию этот параметр равен 180). Если вызов функции `session_cache_expire()` осуществляется с параметром *new_cache_expire*, то

текущее время жизни (180 минут) заменяется на значение, переданное в этом параметре (листинг 13.21).

Листинг 13.21. Использование функции `session_cache_expire()`

```
<?php
  // Устанавливаем ограничитель кэша в 'private'
  session_cache_limiter('private');
  $cache_limiter = session_cache_limiter();
  // Устанавливаем время жизни кэша, равное 30 минутам
  session_cache_expire(30);
  $cache_expire = session_cache_expire();
  // Запускаем сессию
  session_start();
?>
```

Задания

13.1. Усовершенствуйте разработанный в этой главе Whois-сервис таким образом, чтобы он самостоятельно находил Whois-серверы, ответственные за данный диапазон IP-адресов.

13.2. Получите методом `GET` любую HTML-страницу из Интернета или с локального компьютера и извлеките ее название (текст между тегами `<title>` и `</title>`).

13.3. Получите методом `GET` любое изображение из Интернета или с локального компьютера и выведите его в окно браузера с указанием размера загруженного изображения.

ГЛАВА 14

Электронная почта

В этой главе мы рассмотрим основные примеры отправки почтовых сообщений средствами PHP. Тематика данной книги не предусматривает подробного введения в принципы работы электронной почты, об этом вы можете прочитать в наших предыдущих книгах: "Самоучитель PHP 5"[1] и "PHP 5. Практика разработки Web-сайтов"[2]. Эти вопросы изложены там достаточно подробно.

Отправка сообщений с помощью стандартной функции *mail()*

Отправка почты средствами PHP осуществляется при помощи функции `mail()`:

```
bool mail(string to, string subject, string body
          [,string extra_headers] [,string extra_parameters])
```

Эта функция принимает следующие аргументы:

- *to* — адрес электронной почты получателя;
- *subject* — тема сообщения;
- *body* — текст сообщения;
- *extra_headers* — дополнительные заголовки, которые можно задать в сообщении;
- *extra_parameters* — дополнительные параметры, которые можно задать в сообщении.

---

[1] Кузнецов М. В., Симдянов И. В. Самоучитель PHP 5. — СПб.: БХВ-Петербург, 2004. — 560 с.

[2] Кузнецов М. В., Симдянов И. В., Голышев С. В. PHP 5. Практика разработки Web-сайтов. — СПб.: БХВ-Петербург, 2005. — 960 с.

Замечание

Поля *subject* и *body* для темы сообщения и его тела могут быть пустыми строками.

Рассмотрим несколько примеров работы с функцией `mail()` и разработаем скрипт для отправки сообщений.

Начнем с простого примера (листинг 14.1).

Листинг 14.1. Простой пример использования функции `mail()`

```
<?php
  mail("kuznetsov@softtime.ru", "Привет!", "1строка\n2строка\n3строка")
?>
```

В этом примере на адрес `kuznetsov@softtime.ru` будет отправлено сообщение с темой `"Привет"` и текстом:

```
1строка
2строка
3строка
```

Замечание

Для совместимости с ОС Windows каждую строку в почтовых сообщениях необходимо заканчивать символом `\r\n`.

В листинге 14.2 показан пример отправки сообщения `$msg` с использованием необязательного параметра *extra_headers*, содержащего дополнительный заголовок `From`.

Листинг 14.2. Пример использования функции `mail()` с дополнительным заголовком

```
<?php
  mail("kuznetsov@softtime.ru", "Привет!", $msg,
       "From: admin <admin@softtime.ru>")
?>
```

Замечание

E-mail можно указывать в угловых скобках, при этом текст, предшествующий им, будет использован в качестве имени отправителя.

Если требуется указать несколько дополнительных полей, они разделяются символом `\r\n` (листинг 14.3).

Листинг 14.3. Пример использования функции mail() с дополнительным заголовком

```
<?php
  $msg = "Содержимое письма";
  mail("kuznetsov@softtime.ru", "Привет!", $msg,
       "From: admin@{$_SERVER['SERVER_NAME']}\r\n".
       "Reply-To: admin@{$_SERVER['SERVER_NAME']}\r\n".
       "X-Mailer: PHP/".phpversion());
?>
```

В этом примере в поля `From` и `Reply-To` подставляется адрес вида `admin@`*имя_сервера*. При этом имя сервера возвращается из суперглобального массива `$_SERVER`. В пользовательском поле `X-Mailer` в качестве программы отправителя указывается PHP той версии, которая установлена на сервере. Версия PHP возвращается с помощью функции `phpversion()`.

Пример использования необязательного параметра *extra_parameters* приведен в листинге 14.4.

Листинг 14.4. Пример использования функции mail() с дополнительным заголовком

```
<?php
  $msg = "Содержимое письма";
  mail("kuznetsov@softtime.ru", "Привет!", $msg,
       "From: admin@{$_SERVER['SERVER_NAME']}",
       "-fwebmaster@{$_SERVER['SERVER_NAME']}");
?>
```

В следующем примере (листинг 14.5) разработан типичный скрипт отправки сообщений с использованием функции `mail()`.

Листинг 14.5. Сценарий отправки сообщений

```
<?php
// Принимаем данные, которые пользователь ввел в форму отправки сообщений
$action = $_POST["action"];
if (!empty($action)) // Проверяем, переданы ли данные в принципе
{
  if(!empty($mail_to)) // Проверяем, введен ли адрес
  {
    // Проверяем правильность заполнения адреса
    // с помощью регулярного выражения
```

```
    if (!preg_match("/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i", $mail_to))
    {
      $action = "";
      echo("Введите адрес в виде somebody@server.com"); exit();
    }
    $mail_to = substr($_POST["mail_to"],0,32);
    $mail_to = trim($mail_to);
    $mail_to = htmlspecialchars(stripslashes($mail_to));
    $mail_subject = substr($_POST["mail_subject"],0,64);
    $mail_subject = trim($mail_subject);
    $mail_subject = htmlspecialchars(stripslashes($mail_subject));
    $mail_msg = substr($_POST["mail_msg"],0,1024);
    $mail_msg = trim($mail_msg);
    $mail_msg = htmlspecialchars(stripslashes($mail_msg));
    // Формируем сообщение
    if(mail($mail_to, $mail_subject, $mail_msg)) {
      echo("Сообщение успешно отправлено");
    } else {
      echo("При отправлении сообщения произошла ошибка");
    }
  } else {
    echo("Введите адрес получателя");
  }
}
?>
```

В переменных `$mail_to`, `$mail_subject` и `$mail_msg` содержатся соответственно адрес получателя, тема и тело сообщения, переданные из какой-либо формы отправки сообщения. После этого для каждой из переменных следует стандартный блок обработки (листинг 14.6).

Листинг 14.6. Обработка переменных `$mail_to`, `$mail_subject` и `$mail_msg`

```
// Выделяем данные, относящиеся к конкретной переменной из массива $_POST
$mail_to = substr($_POST["mail_to"],0,32);
// Удаляем пробельные символы
$mail_to = trim($mail_to);
// Удаляем теги и обратные слеши в целях повышения безопасности
$mail_to = htmlspecialchars(stripslashes($mail_to));
```

Замечание

Функция `mail()` будет работать только в том случае, если в системе, где выполняется скрипт с ее использованием, установлен почтовый транспортный

агент (MTA). Если у вас на компьютере нет какого-либо запущенного MTA, либо удаленного SMTP-сервера, предоставляющего доступ к вашей машине, то функция `mail()` работать не будет. В состав практически любого дистрибутива UNIX входит транспортный агент sendmail, выполняющий его функции, в то время как в операционной системе Windows для этого требуются установки стороннего транспортного агента (MTA).

Собственная функция *mail()* или отправка сообщений через удаленный SMTP-сервер

Отправить сообщение с локальной машины можно, не прибегая к функции `mail()`. Для этого необходимо разработать приложение SMTP-клиента, которое будет соединяться с удаленным SMTP-сервером. В PHP отсутствуют функции для работы с SMTP-серверами, поэтому далее рассмотрим разработку собственной функции для работы с удаленным SMTP-сервером.

Примечание

Отправка сообщений подобным образом будет работать только в тех случаях, если почтовый сервер является открытым ретранслятором, или ваш IP-адрес является для него разрешенным. Однако в последнее время в целях борьбы со спамерскими рассылками ретрансляторы стараются закрывать, а системные администраторы организаций не разрешают отправку почты с адресов, не находящихся в собственной подсети.

Таким образом, наша задача состоит в создании функций для отправки сообщений электронной почты, которая будет соединяться с удаленным SMTP-сервером и организовывать SMTP-сеанс, во время которого будет происходить отправка сообщения. Код функции приведен в листинге 14.7.

Листинг 14.7. Отправка сообщения через удаленный SMTP-сервер

```
<?php
// Функция отправки сообщения: открывает сокет, ведет диалог с сервером,
// записывает данные, закрывает сокет
function send($server, $to, $from, $subject="",$msg, $headers="")
{
  // Формируем поля заголовка
  $headers="To: $to\nFrom: $from\nSubject: $subject\nX-Mailer: My
           Mailer\n$headers";
  // Соединяемся с сервером по порту 25,
  // при этом переменная $fp содержит дескриптор соединения
  $fp = fsockopen($server, 25, &$errno, &$errstr, 30);
  if (!$fp) die("Server $server. Connection failed: $errno, $errstr");
```

```
  // Если соединение прошло успешно, производим запись данных в сокет,
  // т. е. открываем наш SMTP-сеанс с удаленным сервером $server
  fputs($fp,"HELO $server\n");     // Здороваемся с сервером
  // Посылаем поле from
  fputs($fp,"MAIL FROM: $from\n");
  // Посылаем поле To
  fputs($fp,"RCPT TO: $to\n");
  // Посылаем поле Data
  fputs($fp,"DATA\n");
  // Посылаем сообщение, которое содержится в переменной $msg
  fputs($fp,"$msg\r\n".".""\r\n");
  // Посылаем заголовки
  fputs($fp,$this->headers);
  if (strlen($headers))
    fputs($fp,"$headers\n");
  // Завершаем SMTP-сеанс
  fputs($fp,"\n.\nQUIT\n");
  // Завершаем соединение
  fclose($fp);
  }
}
// Отправка сообщения
send('mx2.yandex.ru', // Почтовый ретранслятор, к примеру, сервера yandex
     'mail@yandex.ru', // Кому
     'mail@softtime.ru',  // От кого
     'Hello!',            // Тема
     'Привет!');          // Сообщение
?>
```

В начале функции происходит соединение с почтовым ретранслятором удаленного SMTP-сервера по 25-му порту с помощью функции `fsockopen()`, работа с которой была подробно рассмотрена в *гл. 13*. Затем серверу отправляются SMTP-команды, которые приведены ранее в этой главе. Текст сообщения должен заканчиваться строчкой на отдельной странице, поэтому точка обязательно должна присутствовать в строке отправки текста сообщения:

```
fputs($fp, "$msg\r\n.\r\n");
```

Как узнать адреса почтовых ретрансляторов?

Узнать, какое имя имеют почтовые ретрансляторы конкретного SMTP-сервера, можно при помощи скрипта из листинга 14.8, выводящего список почтовых ретрансляторов заданного сервера и их приоритет.

Листинг 14.8. Получение списка почтовых ретрансляторов

```
<?php
  // Некоторый адрес электронной почты
  $email="mail@yandex.ru";
  // Разбиваем адрес на массив из двух элементов: имени почтового ящика и
  // имени почтового сервера
  $email_arr = explode("@", $email);
  // Заносим в переменную $host имя почтового сервера
  $host = $email_arr[1];
  // Далее определяем список почтовых ретрансляторов сервера $host
  // и выводим их с указанием значений предпочтения
  getmxrr($host, $mxhostsarr, $weight);
  echo "На $email письма могут отправляться через следующие хосты:<br>";
  for ($i=0; $i < count($mxhostsarr); $i++)
  {
    echo ("$mxhostsarr[$i] = $weight[$i]<br>");
  }
?>
```

В этом скрипте используется функция `getmxrr()`, возвращающая список почтовых ретрансляторов. Ее синтаксис таков:

```
string getmxrr(string hostname, array mxhost, [, array weight])
```

Эта функция принимает в качестве аргумента имя хоста *hostname* в данном домене и заполняет массив *mxhost* списком почтовых ретрансляторов этого домена. Если указан третий необязательный аргумент *weight*, то функция заполняет его значениями предпочтения, которые возвращает ей почтовый ретранслятор. К примеру, для адреса `mail@yandex.ru` скрипт из листинга 14.8 вернет следующий список почтовых ретрансляторов, через которые происходит отправка писем SMTP-сервером Яндекса:

```
mx2.yandex.ru = 10
mx1.yandex.ru = 0
```

Как присоединить вложения к сообщениям и что такое спецификация MIME?

Для отправки сообщений электронной почты с вложениями необходимо использование MIME — спецификации, расширяющей возможности стандартной электронной почты.

Замечание

Спецификация MIME (Multipurpose Internet Mail Extensions, многоцелевые расширения почты Интернета) описана в RFC 2045-2049.

Спецификация MIME позволяет расширить возможности обычной электронной почты для решения следующих задач:

- пересылки восьмибитовых текстов и восьмибитовых символов в заголовке сообщения;
- пересылки двоичных данных любого типа (графики, аудио, видео и т. д.);
- поддержки сложных типов сообщений (к примеру, сообщений, содержащих данные различных типов).

Примечание

Подробно спецификация MIME изложена в *гл. 13* нашей книги "PHP 5. Практика разработки Web-сайтов". Здесь мы, в целях экономии места, не можем позволить себе детальное рассмотрение этого вопроса.

Пример — в листинге 14.9.

Листинг 14.9. Отправка сообщений с вложениями

```
<?php
  // Текст сообщения
  $msg .= "Привет!";
  // Путь к файлу. В данном случае предполагается, что файл
  // расположен в том же каталоге, что и скрипт
  $path = "t.doc";
  // Читаем файл
  $fp = fopen($path,"r");
  if (!$fp)
  {
    print "Файл $path не может быть прочитан";
    //return;
  }
  $file = fread($fp, filesize($path));
  fclose($fp);

  $boundary = "--".md5(uniqid(time()));  // Генерируем разделитель
  $headers .= "MIME-Version: 1.0\n";
  $headers .="Content-Type: multipart/mixed; boundary=\"$boundary\"\n";
  $multipart .= "--$boundary\n";
  $kod='koi8-r';                          // Указываем кодировку
  $multipart .= "Content-Type: text/html; charset=$kod\n";
```

```
  $multipart .= "Content-Transfer-Encoding: Quot-Printed\n\n";
  $multipart .= "$msg\n\n";

  $message_part = "";
  $message_part .= "Content-Type: application/octet-stream";
  $message_part .= "; file_name = \"$path\"\n";
  $message_part .= "Content-Transfer-Encoding: base64\n";
  $message_part .= "Content-Disposition: attachment; filename =
  \"".$path."\"\n\n";
  $message_part .= chunk_split(base64_encode($file))."\n";
  $multipart .= "--$boundary\n".$message_part."--$boundary--\n";
  // Отправляем сообщение
  if(mail("ttt@softtime.ru", "Привет", $multipart, $headers))
  {
    echo "Письмо успешно отправлено";
  }
?>
```

ГЛАВА 15

Работа с библиотеками расширений

В этой книге мы рассмотрим три библиотеки расширений для работы: с графикой (библиотека GD), PDF-документами (библиотека CLibPDF) и Flash (библиотека Ming). Взаимодействию PHP и Flash посвящено *приложение 5*, а в данной главе мы рассмотрим работу с графикой и PDF-документами.

Формат PDF был создан фирмой Adobe и предназначен для создания документов с независящим от операционной системы представлением, т. е. документ PDF, созданный в одной среде, будет выглядеть точно так же в другой.

Замечание

PDF-формат (Portable Document Format, портативный документный формат) стал стандартом де-факто в Интернете для хранения, обмена и распространения документов, чье представление должно оставаться неизменным на всех платформах. Формат получил широкое распространение, т. к. программа просмотра PDF-документов — Acrobat Reader — распространяется свободно, и ее можно загрузить с сайта Adobe Systems Incorparated. Следует отметить, что практически вся научная и техническая информация в мире распространяется в данном формате. В настоящий момент PDF серьезно рассматривается в качестве замены HTML во многих областях, к примеру, в электронных научных библиотеках.

Замечание

Для успешной работы с PDF-форматом на клиентской машине должна быть установлена программа Acrobat Reader, последнюю версию которой можно загрузить по ссылке **http://www.adobe.com/products/acrobat/readstep2.html**.

Для работы с PDF-документами PHP 5 предоставляет две библиотеки: ClibPDF и PDFLib. Обе библиотеки очень похожи друг на друга как с точки зрения функциональности, так и с точки зрения работы с ними средствами PHP. В этой главе мы рассмотрим библиотеку ClibPDF, которая входит в стандартную поставку дистрибутива PHP 5.

Замечание

Пожалуй, одним из немногих отличий библиотеки ClibPDF от PDFLib является то, что первая не позволяет работать одновременно с несколькими PDF-документами.

Обе библиотеки способны создавать PDF-документ в памяти без использования временных файлов.

Замечание

Как и любое расширение в PHP 5, библиотека ClibPDF по умолчанию отключена. Для ее подключения следует снять комментарий со строки `extension=php_cpdf.dll` в конфигурационном файле php.ini.

Основные операции с PDF-документами

Открытие

Для открытия нового PDF-документа используется функция `cpdf_open()`:

```
int cpdf_open(int compression [, string filename])
```

Первый параметр *compression* определяет сжатие документа. Значение параметра, равное 0, говорит о том, что документ открывается в обычном (не сжатом) виде. Необязательный параметр *filename* указывает имя файла, в который записывается документ. При отсутствии этого параметра документ создается в памяти и в последующем может быть сохранен в файл с помощью функции `cpdf_save_to_file()`, либо передан в стандартный вывод функцией `cpdf_output_buffer()`.

Функция возвращает дескриптор PDF-документа, который используется в качестве первого параметра во всех остальных функциях библиотеки.

Сохранение

Для того чтобы сохранить PDF-документ, нужно воспользоваться функцией `cpdf_save_to_file()`:

```
void cpdf_save_to_file(int cpdf, string filename)
```

Эта функция сохраняет созданный в памяти PDF-документ *cpdf* в файл с именем *filename*.

Примечание

Все функции библиотеки ClibPDF, за исключением функции `cpdf_open()`, принимают в качестве первого параметра дескриптор открытого PDF-документа *cpdf*, возвращаемый функцией `cpdf_open()`, поэтому в дальнейшем при описании остальных функций на этом параметре мы останавливаться не будем.

Второй способ сохранения PDF-документа состоит в передаче второго необязательного параметра *filename* функции cpdf_open(). В этом случае в использовании функции cpdf_save_to_file() уже нет необходимости.

Создание новой страницы

Для создания новой страницы в уже открытом PDF-документе предназначена функция cpdf_page_init():

```
bool cpdf_page_init(int cpdf, int page_number, int orientation,
                    float height, float width [, float unit])
```

Функция cpdf_page_init() создает новую страницу высотой *height*, шириной *width*, номером *page_number* и ориентацией *orientation*. Параметр *orientation* может принимать значения 0 (для простого изображения) и 1 (для пейзажа). В том случае, когда необходим вывод текста, значение этого параметра устанавливается равным 0.

Замечание

Точка отсчета координат в PDF-документе начинается не с левого верхнего угла, как в большинстве библиотек, а с *левого нижнего*. Изменить начало координат можно при помощи функции cpdf_translate(), описываемой далее.

Необязательный параметр *unit* устанавливает значение единицы измерения для системы координат. Это значение должно являться числом PostScript-точек на единицу расстояния (в данном случае — на дюйм). По умолчанию данное значение равно 72, что означает, что в одном дюйме содержатся 72 точки.

Добавление закладки

Закладка (bookmark) добавляется с помощью функции cpdf_add_outline():

```
int cpdf_add_outline(int cpdf, int lastoutline, int sublevel,
                     int open, int pagenr, string text)
```

Функция cpdf_add_outline() добавляет закладку с текстом *text*, указывающую на текущую страницу.

Пример использования функции в листинге 15.1.

Листинг 15.1. Добавление закладки в PDF-документ

```
<?php
  $cpdf = cpdf_open(0);
```

```
  cpdf_page_init($cpdf, 1, 0, 595, 842);
  cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
  /*
      {блок вывода нужного текста или рисунка}
  */
  cpdf_close($cpdf);
?>
```

Закладка позволяет осуществлять быстрый переход к указываемому ей месту в PDF-документе и служит для создания содержания документа.

Работа с текстом

Для того чтобы приступить к работе с текстом, нужно начать текстовый раздел с помощью функции `cpdf_begin_text()`:

```
bool cpdf_begin_text(int cpdf)
```

Единственным параметром функции является дескриптор *cpdf* открытого PDF-документа.

По окончании работы текстовый раздел обязательно должен явно завершаться вызовом функции `cpdf_end_text()`:

```
bool cpdf_end_text(int cpdf)
```

Между этими двумя разделами обычно располагаются функции вывода текста, установки шрифтов выводимого текста и выбора способа воспроизведения выводимого текста. Пример — в листинге 15.2.

Листинг 15.2. Добавление начала и окончания текстового раздела в документ

```
<?php
  cpdf_begin_text($pdf);              // начало текстового раздела
    cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
    cpdf_set_text_rendering($cpdf, 1);
    cpdf_text($pdf, 100, 100, "Выводимый текст");
  cpdf_end_text($pdf);                // конец текстового раздела
?>
```

Вывод текста осуществляется функцией `cpdf_text()`, которая для этого и предназначена. Текст передается в параметре *text* в точке документа с координатами *x* и *y*:

```
bool cpdf_text(int cpdf, string text, float x, float y
               [, int mode [, float orientation [, int alignmode]]])
```

Необязательный параметр *mode* определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения. Необязательные параметры *orientation* и *alignmode* определяют поворот текста в градусах и способ выравнивания, соответственно.

Нужный способ воспроизведения текста в PDF-документе выбирается с помощью функции `cpdf_set_text_rendering()`:

```
bool cpdf_set_text_rendering(int cpdf, int rendermode)
```

Возможными значениями параметра *rendermode* являются следующие:

- 0 — заполненный текст;
- 1 — перечеркнутый текст;
- 2 — заполненный и перечеркнутый текст;
- 3 — невидимый текст;
- 4 — заполненный текст, добавленный в отсеченную область;
- 5 — перечеркнутый текст, добавленный в отсеченную область;
- 6 — заполненный и перечеркнутый текст, добавленный в отсеченную область;
- 7 — текст, добавленный в отсеченную область.

Примечание

Отсеченной областью (clipping part) называется область, ограничивающая скрытый графический объект, воспроизводимый только при печати.

Рассмотрим пример, в котором мы по традиции выведем в PDF-документ текст "Hello, World!" (листинг 15.3).

Листинг 15.3. Вывод текста в PDF-документ

```
<?php
  // Открываем новый PDF-документ
  $cpdf = cpdf_open(0);
  // Создаем новую страницу
  cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
  // Добавляем закладку
  cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");

  /* Начало блока вывода текста */
  cpdf_begin_text($cpdf);
```

```
    // Устанавливаем тип и размер шрифта, а также кодировку текста
    cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
    // Устанавливаем способ воспроизведения текста
    cpdf_set_text_rendering($cpdf, 1);
    // Выводим текст
    cpdf_text($cpdf, "Hello world!", 50, 500);
  cpdf_end_text($cpdf);
  /* Окончание блока вывода текста */

  // Рисуем линию вдоль текущего пути
  cpdf_stroke($cpdf);
  // Завершаем PDF-документ
  cpdf_finalize($cpdf);
  // Отсылаем браузеру соответствующий заголовок
  Header("Content-type: application/pdf");
  // Выводим документ в браузер
  cpdf_output_buffer($cpdf);
  // Закрываем PDF-документ
  cpdf_close($cpdf);
?>
```

Результат выполнения этого скрипта показан на рис. 15.1.

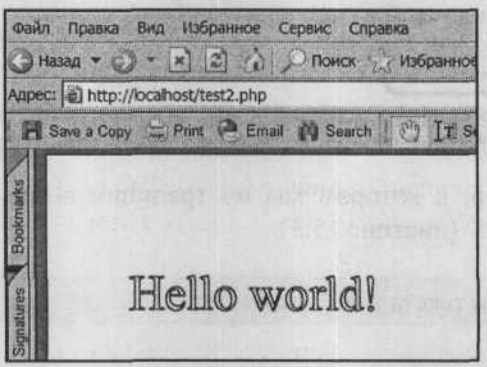


Рис. 15.1. Вывод текста в PDF-документ

Все функции, которые были необходимы для вывода текста с помощью скрипта, показанного в листинге 15.3, мы рассмотрели ранее. После того как PDF-документ был завершен с помощью функции `cpdf_finalize()` и перед тем как вывести его в браузер функцией `cpdf_output_buffer()`, ему был послан заголовок

```
Header("Content-type: application/pdf")
```

Это необходимо для того, чтобы браузер мог определить тип пересылаемого ему документа, как документ PDF.

Внимание!

Подобный заголовок нужно посылать браузеру в любом скрипте, где происходит работа с PDF-документом, который потом необходимо вывести в браузер. Иначе документ будет рассматриваться браузером, как текстовый, и в окно будет выведено его двоичное представление.

Работа со шрифтами

Вид *font_name* и размер *font_size* шрифта, которым выводится текст, устанавливается функцией `cpdf_set_font()`:

```
bool cpdf_set_font(int cpdf, string font_name, float font_size,
                   string encoding)
```

Примечание

В текущей версии библиотеки поддерживаются только стандартные PostScript-шрифты.

Параметр *encoding* определяет кодировку и может принимать следующие значения: `MacRomanEncoding`, `MacExpertEncoding`, `WinAnsiEncoding` и `NULL`. Значение параметра, равное `NULL`, означает встроенную кодировку шрифта.

Рисование линий в PDF-документе

Для того чтобы нарисовать линию, необходимо сначала установить текущую точку с помощью функции `cpdf_moveto()`, которая имеет следующий синтаксис:

```
bool cpdf_moveto(int cpdf, float x, float y [, int mode])
```

Функция `cpdf_moveto()` устанавливает текущую точку с координатами *x* и *y*. Необязательный параметр *mode* определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

После того как текущая точка установлена, нарисовать линию можно, используя функцию `cpdf_lineto()`, которая имеет следующий синтаксис:

```
bool cpdf_lineto(int cpdf, float x, float y [, int mode])
```

Эта функция рисует линию от текущей точки до точки с координатами *x* и *y*. Необязательный параметр *mode* определяет размер единиц измерения. Если данный параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

Загрузка изображения в PDF-документ

Загрузить JPG-изображение в PDF-документ можно с помощью функции `cpdf_import_jpeg()`, которая имеет следующий синтаксис:

```
int cpdf_import_jpeg(int cpdf, string filename,
                     float x, float y,
                     float angle, float width, float height,
                     float x-scale, float y-scale, int gsave
                     [, int mode])
```

Функция `cpdf_import_jpeg()` открывает JPEG-изображение, содержащееся в файле *filename*. Изображение помещается на текущей странице в точке с координатами *x* и *y*. Параметр *angle* определяет, на сколько градусов должно быть повернуто изображение. Параметры *width* и *height* задают ширину и высоту изображения, соответственно. Параметры *x-scale* и *y-scale* устанавливают значение коэффициента масштабирования по направлениям *x* и *y*, соответственно. Для того чтобы функция работала корректно, параметр *gsave* должен иметь ненулевое значение.

Необязательный параметр *mode* определяет размер единиц измерения. Если этот параметр равен 0 или отсутствует, то по умолчанию используются единицы измерения, установленные для страницы. В ином случае координаты измеряются в PostScript-пунктах без учета текущих единиц измерения.

Вывод PDF-документа в браузер

Вывод PDF-документа в браузер производится функцией `cpdf_output_buffer()`, принимающей в качестве своего единственного параметра дескриптор PDF-документа:

```
cpdf_output_buffer(int cpdf)
```

Завершение работы с PDF-документом

После того как основная работа (вывод текста и т. д.) с документом закончена, его необходимо завершить с помощью функции `cpdf_finalize()`:

```
bool cpdf_ finalize(int cpdf)
```

Далее документ можно выводить в браузер, сохранять и т. д. А уже после завершения всех операций с документом его нужно закрыть функцией `cpdf_close()`:

```
cpdf_close(int cpdf)
```

Пример: рисование квадрата в PDF-документе

В этом разделе мы рассмотрим создание PDF-документа и нарисуем в нем прямоугольник с заданными параметрами (листинг 15.4).

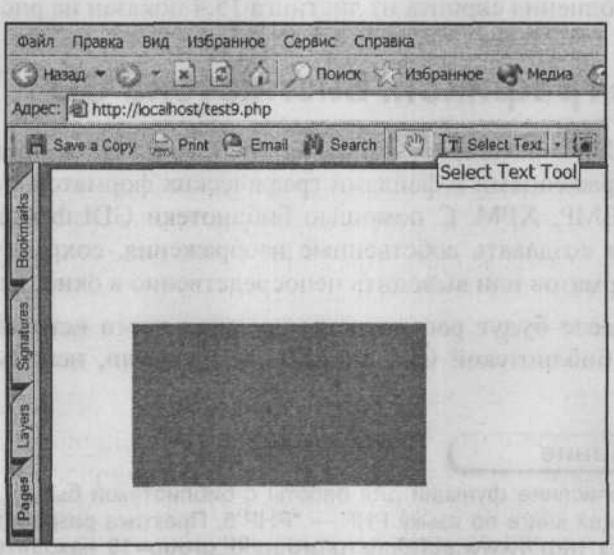


Рис. 15.2. Красный прямоугольник в PDF-документе

Листинг 15.4. Рисуем квадрат в PDF-документе

```
<?php
  // Открываем PDF-документ
  $cpdf = cpdf_open(0);
  // Создаем новую страницу
  cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
  // Устанавливаем RGB-значения цвета
  cpdf_setrgbcolor($cpdf, 1, 0, 0);
  // Рисуем прямоугольник
  cpdf_rect($cpdf, 50, 400, 180, 100);
```

```
  // Закрашиваем внутреннюю часть прямоугольника цветом заполнения
  cpdf_fill($cpdf);
  // Завершаем страницу
  cpdf_finalize($cpdf);
  // Отправляем заголовок браузеру
  Header("Content-type: application/pdf");
  // Выводим содержимое документа в браузер
  cpdf_output_buffer($cpdf);
  // Закрываем документ
  cpdf_close($cpdf);
?>
```

Результат выполнения скрипта из листинга 15.4 показан на рис. 15.2.

Работа с графикой. Библиотека GD

Библиотека GDLib расширяет возможности языка PHP и предназначена для работы с изображениями и файлами графических форматов, таких как JPEG, GIF, PNG, WBMP, XPM. С помощью библиотеки GDLib можно программным способом создавать собственные изображения, сохранять их в файлы различных форматов или выводить непосредственно в окно браузера.

В данном разделе будут рассмотрены решения часто встречающихся задач при работе с библиотекой GD, описаны ее функции, используемые в примерах.

Замечание

Полное описание функций для работы с библиотекой было сделано в нашей предыдущей книге по языку PHP — "PHP 5. Практика разработки Web-сайтов". По адресу **http://www.softtime.ru/group/id_group=15** находится полный справочник по функциям GDLib на русском языке, а по адресу **http://www.boutell.com/gd/** — домашняя страница библиотеки GDLib.

Замечание

Начиная с версии 1.6, поддержка формата GIF в библиотеке GDLib осуществляется не полностью — доступно только чтение файлов в указанном формате. Это связано с тем, что алгоритм сжатия LZW, применяемый при создании файлов в формате GIF, был запатентован. На момент написания книги действие патента истекло и к выходу книги, вероятнее всего, новая версия библиотеки с возобновленной полной поддержкой формата GIF будет доступна для свободной загрузки.

Замечание

Как и любое расширение в PHP 5, библиотека GD по умолчанию отключена, для ее подключения следует снять комментарий со строки `extension=`

php_cpdf.dll в конфигурационном файле php.ini. Для подключения exif-функций, читающих информацию из заголовков файлов форматов JPEG и TIFF, необходимо таким же образом подключить библиотеки php_mbstring.dll и php_exif.dll. Библиотека php_mbstring.dll должна быть в списке подключаемых библиотек первой.

Функция *getimagesize()*

Функция `getimagesize()` возвращает размер изображения в пикселах и различную информацию об изображении.

```
array getimagesize(string filename [, array imageinfo])
```

Здесь:

- *filename* — имя файла с изображением;
- *imageinfo* — при указании этого параметра в него заносится расширенная информация об изображении.

Возвращаемый функцией массив состоит из 4 элементов. Элемент с индексом 0 содержит ширину изображения в пикселах. Элемент 1 — высоту. Элемент 2 — целое число, определяющее тип файла. Расшифровка числовых значений элемента с индексом 2 приведена в табл. 15.1. Элемент 3 содержит строку формата `width="xxx" height="yyy"`, которая может быть вставлена в HTML-тег `<img>`.

Если к файлу нет доступа или изображение не удается прочитать, то возвращается `NULL` и генерируется предупреждение.

Замечание

Функция не входит в библиотеку GD и не может вызываться без наличия установленной библиотеки.

***Таблица 15.1.** Соответствие констант идентификаторов форматов файлов*

Возвращаемое значение	Константа
1	IMAGETYPE_GIF
2	IMAGETYPE_JPEG
3	IMAGETYPE_PNG
4	IMAGETYPE_SWF
5	IMAGETYPE_PSD
6	IMAGETYPE_BMP
7	IMAGETYPE_TIFF_II

Таблица 15.1 (окончание)

Возвращаемое значение	Константа
8	IMAGETYPE_TIFF_MM
9	IMAGETYPE_JPC
10	IMAGETYPE_JP2
11	IMAGETYPE_JPX
12	IMAGETYPE_JB2
13	IMAGETYPE_SWC
14	IMAGETYPE_IFF
15	IMAGETYPE_WBMP
16	IMAGETYPE_XBM

Функция *imagecreatetruecolor()*

Функция `imagecreatetruecolor()` создает пустое полноцветное изображение в памяти с размерами *x_size* и *y_size* и возвращает дескриптор созданного изображения. Созданное изображение имеет черный фон.

```
resource imagecreatetruecolor(int x_size, int y_size)
```

Функция *imagecreatefromjpeg()*

Функция `imagecreatefromjpeg()` создает изображение в памяти из файла *filename* формата JPEG. В качестве имени файла может использоваться URL. Возвращает дескриптор созданного изображения. При ошибке создания возвращается пустая строка и выводится соответствующее сообщение.

```
resource imagecreatefromjpeg(string filename)
```

Функция *imagecreatefromgif()*

Создает изображение в памяти из файла *filename* формата GIF.

```
resource imagecreatefromjpeg(string filename)
```

Функция *imagecreatefrompng()*

Создает изображение в памяти из файла *filename* формата PNG.

```
resource imagecreatefromjpeg(string filename)
```

Функция *imagecopyresampled()*

Функция `imagecopyresampled()` копирует прямоугольные области с одного изображения на другое.

```
bool imagecopyresampled(resource dst_im, resource src_im,
                        int dstX, int dstY,
                        int srcX, int srcY,
                        int dstW, int dstH,
                        int srcW, int srcH)
```

Если *dst_im* равно *src_im*, то области копируются в пределах одного изображения, но при перекрытии областей копирования и вставки результаты непредсказуемы. Если размеры исходной области и области назначения различны, то происходит соответствующее сжатие или растяжение копируемой области. Таким образом, функцию можно использовать для масштабирования изображений.

Параметры, передаваемые функции, таковы:

- *dst_im* — изображение назначения;
- *src_im* — изображение источник;
- *dstX*, *dstY* — точка на изображении назначения, определяющая левый верхний угол прямоугольника, в который будет вставляться копируемая область;
- *dstW*, *dstH* — ширина и высота прямоугольника, в который будет вписана копируемая область;
- *srcX*, *srcY* — точка на изображении-источнике, которая определяет левый верхний угол прямоугольника, содержащего копируемую точку;
- *srcW*, *srcH* — ширина и высота копируемой области на изображении-источнике.

Замечание

Функция неустойчиво работает с неполноцветными изображениями, цветовая палитра которых ограничена 256 цветами.

Замечание

Функция была введена в PHP 4.0.6 и требует для работы библиотеку GD версии 2.0.1 и выше.

Функция *imagejpeg()*

Функция `imagejpeg()` записывает изображение *image* на диск под именем *filename* в формате JPEG.

```
int imagejpeg(resource image [, string filename [, int quality]])
```

Качество сжатия указывается параметром *quality*, который может принимать значения от 0 до 100. 0 соответствует максимальному сжатию, но минимальному качеству изображения. По умолчанию параметр `quality` равен 75.

Если параметр *filename* не указан, то изображение выводится в выходной поток браузера. При выводе изображения непосредственно в браузер необходимо передать обозревателю MIME-тип выводимых данных. Это следует сделать с помощью функции `Header()`:

```
Header("Content-type: image/jpeg");
```

Замечание

Для создания файлов в формате Progressive JPEG следует включить чересстрочное создание изображений функцией `imageinterlace()`.

Функция *imagegif()*

Функция `imagegif()` записывает изображение *image* на диск под именем *filename* в формате GIF.

```
int imagegif(resource image [, string filename])
```

Если параметр *filename* не указан, то изображение будет выведено в выходной поток браузера. При выводе изображения непосредственно в браузер необходимо передать обозревателю MIME-тип выводимых данных. Это следует сделать с помощью функции `Header()`.

```
Header("Content-type: image/gif");
```

Функция *imagepng()*

Функция `imagepng()` записывает изображение *image* на диск под именем *filename* в формате PNG. Если параметр *filename* не указан, то изображение выводится в выходной поток браузера.

```
int imagepng(resource image [, string filename])
```

При выводе изображения непосредственно в браузер необходимо передать обозревателю MIME-тип выводимых данных. Это следует сделать с помощью функции `Header()`.

```
Header("Content-type: image/png");
```

Другие функции, используемые в данной главе, будут рассмотрены непосредственно перед их использованием в примерах кода.

Автоматическое масштабирование изображения

Задача автоматического масштабирования изображений очень часто встречается при разработке Web-приложений, например, при создании фотогалереи, когда необходимо создавать копии изображений маленького размера, предназначенные для предварительного просмотра, или на форуме, для ограничения размера добавляемой посетителем фотографии. Рассмотрим пример скрипта для автоматического масштабирования изображения по заданным размерам (листинг 15.5).

Листинг 15.5. Функция масштабирования изображений. Сохранение в файлы

```
<?php
  // Функция автоматического масштабирования изображения
  // $filename – имя исходного изображения
  // $smallimage – имя файла уменьшенной копии
  // $w и $h – максимальные размеры изображения по ширине и высоте
  function resizeimg($filename, $smallimage, $w, $h)
  {
    // Определение коэффициента сжатия уменьшенной копии изображения
    $ratio = $w/$h;
    // Получение размеров исходного изображения
    $size_img = getimagesize($filename);
    // Если размеры меньше, то масштабирование не нужно
    if (($size_img[0]<$w) && ($size_img[1]<$h)) return true;
    // Определение коэффициента сжатия исходного изображения
    $src_ratio=$size_img[0]/$size_img[1];
    // Вычисление размеров уменьшенной копии, чтобы при
    // масштабировании были сохранены пропорции исходного изображения
    if ($ratio<$src_ratio) $h = $w/$src_ratio;
    else  $w = $h*$src_ratio;
    // Создание пустого изображения по заданным размерам
    $dest_img = imagecreatetruecolor($w, $h);
    // Вызов функции создания изображения
    // в зависимости от расширения исходного файла
    if ($size_img[2]==2)  $src_img = imagecreatefromjpeg($filename);
    else if ($size_img[2]==1) $src_img = imagecreatefromgif($filename);
    else if ($size_img[2]==3) $src_img = imagecreatefrompng($filename);
    // Масштабирование изображения
    if (!imagecopyresampled($dest_img, $src_img,
                            0, 0, 0, 0, $w, $h,
                            $size_img[0], $size_img[1])) return false;
```

```
    // В зависимости от расширения имени файла с уменьшенной копией,
    // переданного в параметрах функции,
    // вызываем функцию сохранения уменьшенной копии в файл
    $path_parts=pathinfo($smallimage);
    if ($path_parts["extension"] == "jpg")
                imagejpeg($dest_img, $smallimage);
    else if ($path_parts["extension"] == "gif")
                imagegif($dest_img, $smallimage);
    else if ($path_parts["extension"] == "png")
                imagepng($dest_img, $smallimage);
    // Очищение памяти от созданных изображений
    imagedestroy($dest_img);
    imagedestroy($src_img);
    return true;
  }
?>
```

В качестве параметров функции `resizeimg()` передаются имя исходного файла с изображением (`$filename`), имя файла, в который следует сохранить уменьшенную копию изображения (`$smallimage`), и максимальные размеры уменьшенной копии изображения по ширине (`$w`) и высоте (`$h`). Если пропорции исходного изображения и его уменьшенной копии не совпадают, то при масштабировании будут использованы пропорции исходного изображения. Но при этом размеры уменьшенной копии не должны превышать переданные в функцию максимальные размеры по ширине и высоте.

Пример кода, вызывающий функцию `resizeimg()`, представлен в листинге 15.6. Код функции `resize()`, приведенный в листинге 15.5, находится в файле util.php.

Листинг 15.6. Создание уменьшенной копии изображения

```
<?php
  // Подключение файла с функцией resizeimg()
  include "util.php";
  // Вызываем функцию автоматического масштабирования изображения
  // img.jpg, уменьшенную копию которого следует записать
  // в файл под именем img_small.jpg
  // Максимальные размеры изображения ограничены
  // 130 пикселами по ширине и 100 по высоте
  resizeimg("img.gif", "img_small.jpg", 130, 100);
?>
```

Выше были рассмотрены функции, сохраняющие изображение, созданное с помощью библиотеки GD, в файл на диске. Но часто возникают задачи, когда требуется вывести уменьшенную копию изображения в браузер, не сохраняя изображение в файле. Например, для добавления на каждое выводимое изображение значка авторских прав или при отображении статистики на счетчике посещений сайта. Эти примеры будут рассмотрены далее. Модифицируем код функции `resizeimg()`, для вывода изображений непосредственно в браузер (листинг 15.7).

Листинг 15.7. Функция масштабирования изображений. Вывод в браузер

```
<?php
  function resizeimg($filename, $w, $h)
  {
    $ratio = $w/$h;
    $size_img = getimagesize($filename);
    if (($size_img[0]<$w) && ($size_img[1]<$h)) return true;
    $src_ratio = $size_img[0]/$size_img[1];
    if ($ratio<$src_ratio) $h = $w/$src_ratio;
    else  $w = $h*$src_ratio;
    $dest_img = imagecreatetruecolor($w, $h);
    if ($size_img[2]==2)  $src_img = imagecreatefromjpeg($filename);
    else if ($size_img[2]==1) $src_img = imagecreatefromgif($filename);
    else if ($size_img[2]==3) $src_img = imagecreatefrompng($filename);
    if (!imagecopyresampled($dest_img, $src_img,
                            0, 0, 0, 0, $w, $h,
                            $size_img[0], $size_img[1])) return false;
    $path_parts = pathinfo($filename);
    // Вывод изображений в браузер
    if ($path_parts["extension"] == "jpg")
    {
      header ("Content-type: image/jpeg");
      imagejpeg($dest_img);
    }
    else if ($path_parts["extension"] == "gif")
    {
      header ("Content-type: image/gif");
      imagegif($dest_img);
    }
    else if ($path_parts["extension"] == "png")
    {
      header ("Content-type: image/png");
      imagepng($dest_img);
    }
```

```
    imagedestroy($dest_img);
    imagedestroy($src_img);
    return true;
  }
?>
```

Изменения коснулись параметров функции — отсутствует параметр `$smallimage`, т. к. теперь изображение не нужно сохранять в файл. И блок сохранения изображений в файлы заменен на блок вывода изображений в браузер с выводом соответствующих заголовков. Формат выводимого в браузер изображения определяется форматом исходного файла (`$path_parts=pathinfo($filename)`), в отличие от предыдущей версии функции `resizeimg()`, где формат определялся расширением файла, в который следовало сохранить уменьшенную копию.

Для создания уменьшенной копии изображения и вывода его в браузер можно воспользоваться кодом из листинга 15.8.

Листинг 15.8. Создание уменьшенной копии изображения и вывод в браузер

```
<?php
  include "util.php";
  resizeimg("img.gif, 130, 100);
?>
```

Следует обратить внимание на то, что изображение в браузере будет отображено только в том случае, если до вызова функции `resizeimg()` не было вывода в браузер никаких символов, в том числе пробелов и пустых строк, т. к. функция отправляет HTTP-заголовок. Пример ошибочного файла приведен в листинге 15.9.

Листинг 15.9. Ошибка при выводе изображений в браузер

```
<html>   // ошибка — вывод текста в браузер
<?php
  echo "Вывод в браузер";  // ошибка — вывод текста в браузер
  include "util.php";
  resizeimg("img.gif, 130, 100);
?>
```

Если до вызова функции `resizeimg()` в браузер был осуществлен вывод текста, то вместо изображения на странице появится двоичный код файла с изображением. Для того чтобы двоичный код файла был воспринят как изобра-

жение, необходимо послать в браузер соответствующие заголовки, например, для изображений формата JPEG должен быть послан заголовок:

```
Header("Content-type: image/jpeg");
```

HTTP-заголовки могут быть посланы и восприняты браузером только в том случае, если еще не было вывода текста в выходной поток браузера.

Как видно из приведенного примера, данный скрипт еще не применим для сайто-строительства, т. к. невозможен вывод изображения, внедренного в дизайн страницы. Задача вывода изображения внутри HTML-кода решается в следующем разделе.

Вывод сгенерированного изображения в HTML-коде

Создадим файл imggd.php, внутри которого будет располагаться функция `resizeimg()` — листинг 15.10.

Листинг 15.10. Файл imggd.php

```
<?php
  // Название файла с исходным изображением получаем как параметр
  $img = $_GET['img'];
  resizeimg($img,130,100);
  function resizeimg($filename, $w, $h)
  {
    // Код функции полностью соответствует коду функции из листинга 15.7
  }
?>
```

Этот файл будет использоваться для подстановки вместо пути к изображению в параметре `src` HTML-тега `<img>` (листинг 15.11).

Листинг 15.11. HTML-код для вставки сгенерированного изображения на страницу

```
<img src="resize.php?img=my_img.jpg" border="0" alt="">
```

Таким образом, скрипт, находящийся в файле resize.php, формирует изображение, которое выводится в HTML-теге `<img>`. Имя изображения, которое нужно масштабировать и вывести на странице, передается в параметре `img`:

```
resize.php?img=my_img.jpg
```

Добавление текста на изображение

Задача добавления текста на существующее изображение встречается достаточно часто в Web-разработках. Например, при создании счетчика посещений сайта, когда количество хостов и хитов накладываются на изображение-баннер. Или при создании фотогалереи, когда необходимо на каждое изображение поместить значок авторских прав и имя автора.

Рассмотрим функции, которые могут быть использованы при добавлении текста на изображение.

Перед добавлением текста на существующее изображение следует определить цвет, которым будет создано изображение. Функции `imagecolorallocate()` и `imagecolorallocatealpha()` используются для определения цвета.

Функция *imagecolorallocate()*

Функция `imagecolorallocate()` возвращает идентификатор цвета для изображения `image`, представленный RGB-компонентами (Red, Green, Blue). Полученный идентификатор цвета далее может использоваться для функций рисования. Функция `imagecolorallocate()` должна вызываться для каждого цвета, используемого при рисовании изображения *image*.

```
int imagecolorallocate(resource image,
                       int red, int green, int blue)
```

Параметры *red*, *green*, *blue* являются RGB-компонентами и их значения лежат в диапазоне от 0 до 255 или от 0x00 до 0xFF в шестнадцатеричном формате.

Функция *imagecolorallocatealpha()*

Поведение функции `imagecolorallocatealpha()` и ее параметры аналогичны функции `imagecolorallocate()` за исключением параметра *alpha*, который определяет прозрачность.

```
int imagecolorallocatealpha(resource image,
                            int red, int green, int blue,
                            int alpha)
```

Значения параметра *alpha* лежат в диапазоне от 0 до 127: 0 определяет полную непрозрачность, 127 соответствует полностью прозрачному цвету.

Библиотека GD обеспечивает поддержку нескольких типов шрифтов: TrueType, FreeType, PostScript Type1. В примерах этой главы будут использованы функции для работы со шрифтами TrueType.

Примечание

Если работа скрипта осуществляется в Windows, то шрифты TrueType можно найти в системной папке Fonts каталога Windows.

Примечание

Для работы функций, поддерживающих шрифты TrueType, требуется наличие библиотеки FreeType. Проверить ее наличие можно, выполнив функцию `phpinfo()`. Сведения о ней будут находиться в разделе GD.

Функция *imagestring()*

Функция `imagestring()` осуществляет горизонтальное рисование строки *s* на изображении *image*.

```
int imagestring(resource image, int font, int x, int y,
                string s, int color)
```

Здесь:

- *font* — идентификатор шрифта. Если *font* равен 1, 2, 3, 4 или 5, то используется встроенный шрифт (большее число соответствует шрифту большего размера);
- *x*, *y* — координаты левого верхнего угла рисуемого символа;
- *color* — идентификатор цвета символа.

Функция *imagettfbbox()*

Функция `imagettfbbox()` вычисляет размеры прямоугольника, в который вписана строка *text*, написанная TrueType-шрифтом.

```
array imagettfbbox(int size, int angle, string fontfile, string text)
```

Здесь:

- `size` — размер текста в пикселах;
- *angle* — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол *0* градусов соответствует направлению на 3 часа;
- *fontfile* — имя файла, содержащего TrueType-шрифт.

Функция возвращает массив координат левого нижнего и правого верхнего углов прямоугольника. Описание индексов массива приведено в табл. 15.2.

Функция `imagettfbbox()` может использоваться для проверки, не выходит ли записываемый текст за границы изображения.

Таблица 15.2. Описание массива координат вершин прямоугольника

Индекс массива	Описание
0	x-координата левого нижнего угла
1	y-координата левого нижнего угла
2	x-координата правого верхнего угла
3	y-координата правого верхнего угла

Функция *imagettftext()*

Функция `imagettftext()` записывает строку *text* на изображении *image* TrueType-шрифтом.

```
array imagettftext(resource image, int size, int angle,
                   int x, int y, int color,
                   string fontfile, string text)
```

Здесь:

- *size* — размер шрифта в пикселах;
- *angle* — угол поворота текста. Поворот осуществляется против часовой стрелки. Угол 0 градусов соответствует направлению на 3 часа;
- *x*, *y* — координаты точки, с которой начинается запись текста. Приблизительно соответствует левому верхнему углу первого символа;
- *color* — идентификатор цвета текста;
- *fontfile* — имя файла со шрифтом.

Функция возвращает массив координат 4 вершин прямоугольника, в который будет вписан текст. Вершины перечисляются в следующем порядке: нижняя левая, нижняя правая, верхняя правая, верхняя левая (рис. 15.3).

Пример — в листинге 15.12.

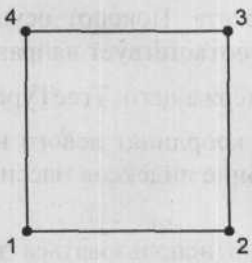


Рис. 15.3. Порядок вершин прямоугольника, возвращаемого функцией `imagettftext()`

Листинг 15.12. Файл writetext.php. Наложение текста поверх изображения

```
<?php
  $img = $_GET['img'];
  if ($img == "") exit();
  writeTextOnImage($img, "IT-студия SoftTime");

  function writeTextOnImage($filename, $text)
  {
    if ($filename == "") exit();
    $size_img = getimagesize($filename);
    if ($size_img[2]==2)  $src_img = imagecreatefromjpeg($filename);
    else if ($size_img[2]==1) $src_img = imagecreatefromgif($filename);
    else if ($size_img[2]==3) $src_img = imagecreatefrompng($filename);
    if (!$src_img) exit();
    // Назначаем цвет
    $color = imagecolorallocatealpha($src_img, 0, 255, 0, 50);
    $height_font = 40;
    $angle = 0;
    $font_file = "arbat.ttf";
    // Запись текста поверх изображения
    $box = imagettftext($src_img, $height_font, $angle, 10, 150,
                        $color, $font_file, $text);
    // Вывод изображения в браузер
    if ($size_img[2]==2)
    {
      header ("Content-type: image/jpeg");
      imagejpeg($src_img);
    }
    else if ($size_img[2]==1)
    {
      header ("Content-type: image/gif");
      imagegif($src_img);
    }
    else if ($size_img[2]==3)
    {
      header ("Content-type: image/png");
      imagepng($src_img);
    }
    imagedestroy($src_img);
    return true;
  }
?>
```

Функция наложения текста на изображение сохранена в файле writetext.php. В этот файл в качестве параметра передается имя изображения. Затем проверяется, что данный параметр не пуст, и вызывается функция наложения текста "IT-студия SoftTime" на изображение, имя которого передано в файл в параметре `$img`. Файл с TrueType-шрифтом, определяемый переменной `$font_file` (`$font_file = "arbat.ttf"`), должен находиться в той же папке, что и сам скрипт.

Чтобы наложить текст на изображение, создайте код из листинга 15.13.

Листинг 15.13. Вставка сгенерированного изображения с текстом на HTML-страницу

```
<?php
  $size = getimagesize("img.jpg");
?>
<img src="writetext.php?img=img.jpg" border="0"
                              <?= $size[3] ?> alt="" >
```

Рис. 15.4. Наложение текста на изображение

Конструкция `<?= $size[3] ?>` внутри тега `<img>` вставляет размеры изображения в формате: `width="`*число_пикселов*`" height="`*число_пикселов*`"`. Этот скрипт

выведет в браузер изображение с наложенным на него текстом "IT-студия SoftTime" (рис. 15.4).

Построение круговой диаграммы

Динамическое рисование диаграмм средствами библиотеки GD может потребоваться при выводе статистических данных, например, статистики голосования, статистики посещений сайта и т. п.

Рассмотрим функции библиотеки GD, которые используются в скрипте рисования диаграммы и не были рассмотрены ранее.

Функция *imagecreatetruecolor()*

Функция `imagecreatetruecolor()` создает пустое полноцветное изображение с размерами *x_size* и *y_size*. Возвращает дескриптор созданного изображения. Созданное изображение имеет черный фон.

```
resource imagecreatetruecolor(int x_size, int y_size)
```

Функция *imagefilledellipse()*

Функция `imagefilledellipse()` рисует закрашенный эллипс с центром в точке с координатами *cx*, *cy* шириной *w* и высотой `h`. Цвет контура и заливки определяется идентификатором цвета *color*.

```
bool imagefilledellipse(resource image, int cx, int cy,
                        int w, int h, int color)
```

Функция *imagefilledarc()*

Функция `imagefilledarc()` рисует сектор эллипса и осуществляет его заливку.

```
bool imagefilledarc(resource image, int cx, int cy, int w, int h,
                    int s, int e, int color, int style)
```

Здесь:

- *cx*, *cy* — координаты центра эллипса;
- *w*, *h* — ширина и высота эллипса;
- *color* — идентификатор цвета;
- *s*, *e* — начальный и конечный углы сектора. Углы задаются в градусах. Отсчет производится против часовой стрелки. Позиция, равная 0 градусов, соответствует направлению "3 часа";

- *style* — стиль рисования; может принимать значения, которые перечисляются через побитовое ИЛИ:
  - IMG_ARC_PIE — соединяет начальную и конечную точки на окружности кривой линией, повторяющей окружность;
  - IMG_ARC_CHORD — соединяет начальную и конечную точки на окружности прямой линией;
  - IMG_ARC_NOFILL — рисуемый сектор не закрашивается;
  - IMG_ARC_EDGED — начальная и конечная точки соединяются с центром окружности.

  Значения IMG_ARC_PIE и IMG_ARC_CHORD — взаимоисключающие и не могут применяться вместе.

Замечание

Функции imagecreatetruecolor(), imagefilledellipse() и imagefilledarc() были введены в PHP 4.0.6 и требуют для работы библиотеку GD версии 2.0.1 и выше.

Пример — в листинге 15.14.

Листинг 15.14. Рисование диаграмм

```
<?
  $sectors = array(50,25,74,16);
  // Создание пустого изображения, размером 200x200 пикселов
  $img =  imagecreatetruecolor (200, 200);
  // Если изображение не создано, выполнение скрипта останавливается
  if (!$img) exit();
  // Определение белого цвета на изображении
  $white = imagecolorallocate($img, 255, 255, 255);
  // Заливка изображения белым цветом
  imagefill($img, 1, 1, $white);
  // Определение цвета фона диаграммы
  $background = imagecolorallocate($img, 240, 240, 240);
  // Переменные $cy и $cy определяют центр диаграммы
  $cx = $cy = 100;
  // Переменные $w и $h определяют ширину и высоту диаграммы
  $w = $h = 100;
  // Рисование окружности
  imagefilledellipse($img, $cx, $cy, $w, $h, $background);
  // Инициализация начального угла сектора
  $start = 0;
```

```
  foreach ($sectors as $value)
  {
    // Формирование цвета для каждого сектора
    // Цвет формируется случайным образом
    $color = imagecolorallocate($img,
                    rand(0, 255), rand(0, 255), rand(0, 255));
    // Определение конечного угла сектора
    $angle_sector = $start + $value;
    // Рисование сектора
    imagefilledarc($img, $cx, $cy, $w, $h, $start, $angle_sector,
                    $color, "IMG_ARC_PIE || IMG_ARC_EDGED");
    // Увеличение начального угла сектора
    $start += $value;
  }
  // Вывод изображения в браузер в формате PNG
  header("Content-type: image/png");
  imagepng($img);
?>
```

В самом начале скрипта определяется массив `$sectors`, который содержит угловые размеры секторов. Скрипт будет более универсальным, если представить данные в массиве `$sectors` размерами секторов в процентах. Для этого следует провести нормирование данных перед их использованием:

```
$value = $value*360/100;
```

Данный скрипт выводит в браузер круговую диаграмму, изображенную на рис. 15.5.

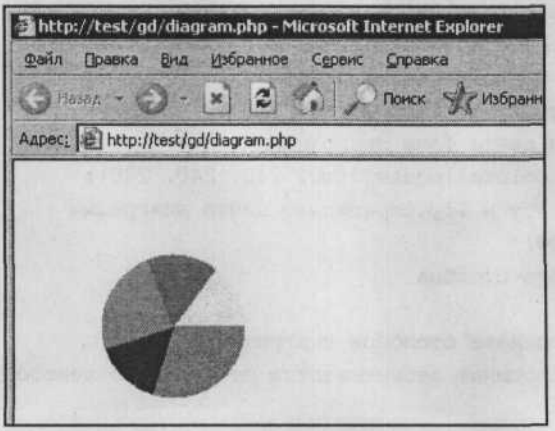


Рис. 15.5. Круговая диаграмма

Построение гистограммы

Скрипт построения гистограммы (столбцовой диаграммы) в целом аналогичен скрипту построения круговой диаграммы, но статистика отображается не в секторах окружности, а с помощью столбцов. Рассмотрим функцию `imagefilledrectangle()`, которая служит для построения такого типа диаграмм.

Функция *imagefilledrectangle()*

Функция `imagefilledrectangle()` рисует закрашенный прямоугольник на изображении *image*. Прямоугольник определяется координатами левого верхнего и правого нижнего углов — *x1*, *y1* и *x2*, *y2* соответственно. Цвет прямоугольника задается идентификатором цвета *color*.

```
int imagefilledrectangle(resource image, int x1, int y1,
                         int x2, int y2, int color)
```

Пример — в листинге 15.15.

Листинг 15.15. Построение гистограммы

```
<?
  $sectors = array(80,15,54,16);
  // Создание пустого изображения, размером 200x200 пикселов
  $img =  imagecreatetruecolor (200, 200);
  // Если изображение не создано, выполнение скрипта останавливается
  if (!$img) exit();
  // Определение белого цвета на изображении
  $white = imagecolorallocate($img, 255, 255, 255);
  // Заливка изображения белым цветом
  imagefill($img, 1, 1, $white);
  // Определение цвета фона диаграммы
  $color = imagecolorallocate($img, 240, 240, 240);
  // Переменные $cy и $cy определяют центр диаграммы
  $cx = $cy = 100;
  // Ширина одного столбца
  $w = 30;
  // Нижняя координата столбцов диаграммы
  // Значения координат отсчитываются от верхнего левого угла
  $y1 = 200;
  // Максимальный размер изображения по высоте
  $max_y = 200;
```

```
  // Координата x, с которой начнется построение диаграммы
  $x1 = 0;
  foreach ($sectors as $value)
  {
    // Формирование цвета для каждого столбца
    $color = imagecolorallocate($img,
                    rand(0, 255), rand(0, 255), rand(0, 255));
    // Нормирование высоты столбца. Перевод процентов в пикселы
    $y2 = $y1 - $value*$max_y/100;
    // Определение второй координаты прямоугольника
    $x2 = $x1 + $w;
    // Рисование прямоугольника
    imagefilledrectangle($img, $x1, $y1, $x2, $y2, $color);
    // Определение начальной x-кооринаты для следующего столбца
    $x1 = $x2 + 5;
  }
  // Выводим изображение в браузер в формате PNG
  header ("Content-type: image/png");
  imagepng($img);
?>
```

Массив `$sectors`, определяемый в начале скрипта, содержит высоту столбцов в процентах. Данный скрипт выводит в браузер гистограмму, изображенную на рис. 15.6.

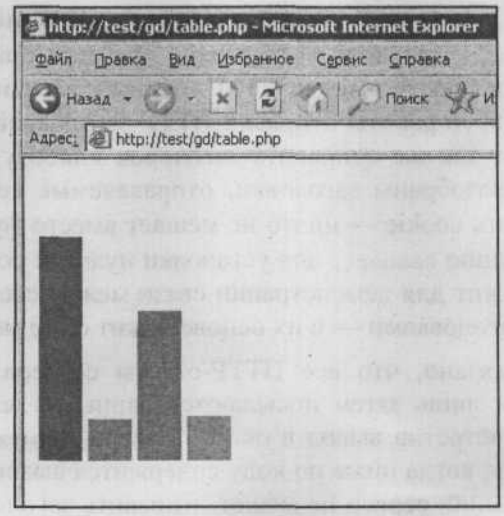


Рис. 15.6. Гистограмма

ГЛАВА 16

Часто встречающиеся ошибки

В данной главе будут рассмотрены типичные ошибки, которые могут возникать при программировании на PHP. Отладка Web-приложений представляет собой сложную задачу, т. к. Web-приложение само по себе распределено по многим машинам. Кроме того, положение осложняется тем, что большинство приложений разрабатываются на Windows-машинах, а затем переносятся на UNIX-машины.

Преждевременная отправка HTTP-заголовков

Наиболее частой ошибкой является вызов функций `session_start()`, `setcookie()` и `header()` после вывода в окно браузера при помощи функций `echo()`, `print()` или просто вывода HTML-данных. Механизмы cookie и сессий требуют для своей работы отправки HTTP-заголовков клиенту. Функция `header()` позволяет так же отправить заголовок клиенту — в *гл. 13*, посвященной сокетам, разобраны заголовки, отправляемые сервером браузеру с просьбой установить cookie — ничто не мешает вместо функции `setcookie()` использовать функцию `header()` для установки нужных cookies. Это длительное пояснение служит для демонстрации связи между сессиями, cookie и остальными HTTP-заголовками — в их основе лежит один механизм.

В *гл. 13* было показано, что все HTTP-ответы сервера вначале содержат HTTP-заголовки, и лишь затем посылаются данные, т. е. HTML-текст. Интерпретатор PHP, встретив вывод в окно браузера, вынужден отправить все заголовки. Поэтому, когда ниже по коду содержится вызов одной из перечисленных ранее функций, сервер не может отправить заголовки повторно, т. к. клиенту передаются данные, а HTTP-заголовки уже ушли. В результате сервер выдает предупреждение, одно из которых можно видеть в листинге 16.1.

Листинг 16.1. Предупреждение сообщает о некорректном завершении функции `session_start()`

```
Warning: session_start() [function.session-start]: Cannot send session
cookie - headers already sent by (output started at E:\main\get.php:2)
in E:\main\get.php on line 3
```

Предупреждение в листинге 16.1 сообщает о некорректном завершении работы функции `session_start()`, вызов которой осуществляется в 3-й строке скрипта E:\main\get.php, т. к. во второй строке скрипта HTTP-заголовки были отправлены клиенту. В листинге 16.2 приводится код, вызывающий вывод данного предупреждения. Виновницей бойкота функции `session_start()` является лишняя строка перед тегом `<?php`. Точно такую же реакцию вызовет пробел или вывод текста функцией `echo()`, как это показано в листинге 16.3.

Листинг 16.2. Ошибочный вызов функции `session_start()`

```
<?php
  session_start();
?>
```

Листинг 16.3. Преждевременный вывод текста

```
<?php
  echo "Hello"; // Данный текст вызовет сбой в работе session_start()
  session_start();
?>
```

Если разместить функции `session_start()`, `setcookie()` и `header()` до вывода информации в окно браузера не удается, можно задержать вывод при помощи функций управления вывода, рассматриваемых в *гл. 17*.

Кроме этого, перед выводом текста можно проверить, какие HTTP-заголовки подготовлены к отправке. Это осуществляется при помощи функции `headers_list()`, имеющей следующий синтаксис:

```
array headers_list(void)
```

Функция возвращает массив, в элементах которого помещаются отправляемые HTTP-заголовки. В листинге 16.4 продемонстрирована работа функции `headers_list()`.

Листинг 16.4. Функция `headers_list()`

```
<?php
  // Устанавливаем cookie при помощи setcookie()
  setcookie('name', 'value', time() + 600);
```

```
  // Сообщаем браузеру, что содержимое передаваемых
  // ему данных будет отправлено в виде обычного текста
  header('Content-type: text/plain');
  // Выясняем, какие заголовки будут отправлены
  print_r(headers_list());
?>
```

В функции устанавливается cookie `name` и отправляется HTTP-заголовок функцией `header()`. Результат работы скрипта выглядит следующим образом:

```
Array
(
    [0] => X-Powered-By: PHP/5.0.1
    [1] => Set-Cookie: name=value; expires=Tue, 11-Jan-2005 10:57:14 GMT
    [2] => Content-type: text/plain
)
```

Как видно, помимо отправленных функцией двух заголовков, PHP по собственной инициативе отправляет дополнительный заголовок `"X-Powered-By: PHP/5.0.1"`, который, скорее всего, будет проигнорирован большинством браузеров.

Еще одной функцией, позволяющей выяснить текущий статус заголовков, является функция `headers_sent()`, которая проверяет, отправлены ли HTTP-заголовки клиенту и, если отправлены, то откуда. Функция имеет следующий синтаксис:

```
bool headers_sent([string &file [, int &line]])
```

Функция возвращает `false`, если HTTP-заголовки еще не были отправлены клиенту, и `true` в противном случае. Если были указаны необязательные параметры *`file`* и *`line`*, функция `headers_sent()` возвращает имя файла и номер строки, где был начат вывод данных в переменные *`file`* и *`line`* соответственно. Пример обработки ситуации преждевременной отправки заголовков приведен в листинге 16.5.

Листинг 16.5. Функция `headers_sent()`

```
<?php
  if (headers_sent($filename, $linenum))
  {
    exit("HTTP-заголовки уже отправлены в
          $filename в строке $linenum\n");
  }
?>
```

Орфографические ошибки

Рассмотрим одно из сообщений на форуме **http://www.softtime.ru/forum/**, где авторы обсуждают с читателями вопросы, посвященные PHP:

"Вот сейчас изучаю массивы и прочитал, что массивы можно создавать следующим образом:

```
$b1[] = "WindowsXP";
$b1[] = "Windows 98";
$b1[] = "Windows 95";
echo $b1[0];
```

При этом выводится `WindowsXP`.

А можно писать так:

```
$b2 = array("chicken", "steak", "turkey");
echo $b2[1];
```

При этом должно появиться: `steak`.

Но вместо этого скрипт выводит:

```
Fatal error: Call to undefined function: array()
```

Подскажите, пожалуйста, в чем причина?"

На первый взгляд ничего криминального в коде нет и вы никогда не найдете ошибку, поскольку она заключается в том, что первая буква ключевого слова `array()` набрана в русской раскладке.

Замечание

Старайтесь использовать редактор, подсвечивающий русские и английские слова разными стилями — это позволит сохранить массу времени и нервов.

К разряду этих же ошибок относится появление в коде невидимых символов, мешающих работе интерпретатора PHP. Поэтому необходимо иметь под рукой несколько редакторов. Если совершенно правильный с вашей точки зрения код отказывается работать, имеет смысл открыть его в различных редакторах и посмотреть, не проявятся ли невидимые символы.

Замечание

Старайтесь выбирать редактор, позволяющий менять кодировки, т. к. в некоторых кодировках невидимые символы не отображаются.

В качестве следующего примера ошибок данного класса можно привести форму в листинге 16.6.

Листинг 16.6. Ошибка в HTML-форме

```
<form action=handler.php metod=post>
  Имя: <input type=text name=name><br>
  Пароль: <input type=password name=pass><br>
  <input type=submit value=Отправить>
</form>
```

В качестве обработчика HTML-формы выступает скрипт handler.php, выводящий содержимое суперглобального массива `$_POST`, который должен содержать два элемента `name` и `pass`, по числу текстовых полей в форме (листинг 16.7).

Листинг 16.7. Обработчик handler.php

```
<?php
  print_r($_POST);
?>
```

Ввод информации в форму из листинга 16.6 и отправка ее обработчику приведет к следующему результату:

```
Array ()
```

Суперглобальный массив `$_POST` оказался пустым. Ошибка, приводящая к такому результату, заключается в неверном написании параметра `method` (в орме — `metod`), задающего метод отправки данных, в результате чего параметр игнорируется, и данные отправляются методом по умолчанию (`GET`).

Замечание

Следует очень аккуратно составлять HTML-код, т. к. об ошибках в нем никто не сообщает (ни браузер, ни PHP-интерпретатор).

HTML-форма, приведенная в листинге 16.8, не загрузит файл с машины клиента на сервер.

Листинг 16.8. HTML-форма не загружает файл на сервер

```
<form action="download.php" method=post>
  <input type=file name="putfile">
  <input tyme=submit>
</form>
```

Ошибка проектирования формы заключается в отсутствие параметра `enctype="multipart/form-data"`.

Замечание

Атрибут `enctype` формы определяет вид кодировки, которую браузер применяет к параметрам формы. Для того чтобы отправка файлов на сервер действовала, атрибуту `entype` необходимо присвоить значение `multipart/form-data`. По умолчанию этот атрибут имеет значение `application/x-www-form-urlencoded`.

Права доступа

При переносе Web-приложений, разработанных в среде Windows, на UNIX-сервер часто можно наблюдать ошибки, связанные с правами доступа (листинг 16.9). Особенно это характерно для скриптов, использующих файлы для хранения информации.

Листинг 16.9. Ошибки, связанные с правами доступа

```
Warning fopen(records/rec.1100029015): failed to stream: Permission
denied in /home/broker/public_html/gb/addrec.php on line 101
```

Предупреждение в листинге 16.9 сообщает о том, что функция `fopen()`, расположенная в 101 строке файла addrec.php, не может открыть файл rec.1100029015 в каталоге records. Это связано с тем, что у скрипта addrec.php не достаточно прав для записи в каталог records и он не может ни создать в нем файлы, ни редактировать уже существующие. Для исправления данной ошибки следует выставить корректные права доступа для каталога и файла, к которому обращается скрипт.

Замечание

Дочерние процессы сервера Apache часто запускаются от имени пользователя `nobody`, не имеющего никаких прав в системе. Если злоумышленник произведет атаку на переполнение буфера, то получит в распоряжение лишь учетную запись пользователя `nobody`. Обратной стороной такой меры защиты является необходимость установки для рабочих каталогов прав доступа `0777`, а для файлов — `0666`, т. к. права пользователя `nobody` по отношению к файлам другого пользователя определяются последней цифрой.

Замечания

По умолчанию в PHP 5 выставлен максимально высокий уровень обработки ошибок. Это приводит к тому, что в окно браузера выводятся не только сообщения об ошибках и предупреждения, но и замечания, например, об не инициализированных переменных, которые следует воспринимать как советы по кодированию.

Замечание

С уровнями обработки ошибок можно ознакомиться в *гл. 1*.

Вывод замечаний приводит к нарушению дизайна страницы, и их устранение сильно ухудшает читабельность кода. Поэтому на большинстве серверов выставляют более низкий уровень обработки ошибок, например:

```
error_reporting = E_ALL & ~E_NOTICE
```

Это приводит к подавлению вывода малоинформативных замечаний.

Привилегии в MySQL

Основной ошибкой при работе с СУБД MySQL является совершение операций от имени пользователя, не имеющего прав доступа на выполнение операции.

Замечание

По умолчанию при первой установке MySQL добавляются два пользователя — суперпользователь (`root`) и анонимный пользователь, в качестве имени которого выступает пустая строка. Суперпользователь обладает всеми правами в системе и может выполнять все операции, в то время как права анонимного пользователя и любого другого пользователя, добавленного при помощи SQL-оператора `GRANT`, ограничены.

При запуске программы-клиента mysql с параметрами по умолчанию вход осуществляется из-под учетной записи анонимного пользователя (листинг 16.10). Попытка создания базы данных из-под анонимного пользователя приводит к выдаче 1044-й ошибки (пользователь `'@localhost'` не имеет право выполнять SQL-оператор `CREATE DATABASE`), выходу из клиента mysql. Последующий вход из-под учетной записи пользователя `root` позволяет создать базу данных.

Листинг 16.10. Неудачная попытка осуществить создание базы данных

```
mysql
mysql> CREATE DATABASE new_database;
ERROR 1044: Access denied for user: '@localhost' to database
'new_database'
mysql>exit
Bye
mysql -u root
mysql> CREATE DATABASE new_database;
Query OK, 1 row affected (0.01 sec)
```

ГЛАВА 17

Разное

В этой главе будут рассмотрены различные приемы программирования на PHP, не вошедшие в предыдущие главы.

Когда необходима рекурсия?

Рекурсия — это вызов функции из самой себя. Пример рекурсивной функции приведен в листинге 17.1.

Листинг 17.1. Рекурсивная функция `callself()`

```
<?php
  function callself($counter)
  {
    // Если параметр $counter больше 0, продолжаем рекурсивный спуск
    if($counter>0)
    {
      // Уменьшаем значение параметра $counter и выводим его значение
      // в окно браузера
      echo ($counter--)."<br>";
      // Осуществляем рекурсивный вызов функции callself()
      callself($counter);
    }
    // Если значение параметра меньше или равно 0, прекращаем
    // рекурсивный спуск
    else return;
  }
  // Вызываем функцию callself()
  callself(4);
?>
```

Результат работы функции будет последовательность цифр:

```
4
3
2
1
```

Функция `callself()` вызывает саму себя до тех пор, пока ее параметр `$counter` положителен и не равен нулю. Рекурсивных функций по возможности стараются избегать, т. к. они относятся к трудным по восприятию конструкциям языка и отладка их является достаточно сложным делом, особенно когда приходится иметь дело не с простейшей рекурсивной функцией, представленной в листинге 17.1, а сложной функцией, осуществляющей рекурсивный вызов в нескольких местах функции.

Замечание

Опасность использования неотраженных рекурсивных функций заключается в возможности перехода их в режим бесконечной рекурсии, когда условие, прекращающее спуск вниз по рекурсии, из-за ошибки не наступает, в результате чего, как и в случае бесконечных циклов, происходит зависание программы.

Практически в любом случае можно избежать рекурсивных функций. Исключение составляют задачи, так или иначе связанные с обходом деревьев. К таким задачам относится, например, удаление каталогов *(см. гл. 6)*, когда число файлов и подкаталогов заранее не известно, и необходимо вызывать функцию удаления до тех пор, пока не будут удалены файлы на самом глубоком уровне вложенности. К таким задачам относится и подсчет вложенных сумм, когда количество вложений заранее не известно, а также любая другая задача, являющаяся аналогом бинарного дерева.

Создание переменных, или как динамически формировать PHP-код

Функция `eval()` позволяет динамически формировать PHP-код, переданный ей в качестве параметра (листинг 17.2).

Листинг 17.2. Функция `eval()`

```
<?php
  $code = '$str = "Hello world!<br>";
                  echo $str;';
  eval($code);
  echo substr($str, 0, 5);
?>
```

Результатом работы функции будут следующие строки:

```
Hello world!
Hello
```

Таким образом, переменная `$str`, которая определена в строке `$code`, доступна не только в пределах данной строки, но и после выполнения функции `eval()`. Поэтому данную функцию удобно использовать для формирования динамических переменных, например восьмеричных чисел, определяющих права доступа (листинг 17.3).

Листинг 17.3. Динамическое формирование значения переменных

```
<?php
  $own = 7;
  $grp = 5;
  $oth = 5;
  eval("\$mode = 0".$own.$grp.$oth.";");
?>
```

Код в листинге 17.3 эквивалентен определению переменной `$mode` в скрипте следующим образом: `$mode = 0755`.

При помощи функции `eval()` можно динамически формировать не только значения переменных, но и сами переменные (листинг 17.4).

Листинг 17.4. Динамическое формирование имени переменной

```
<?php
  $id_menu = 3;
  // Первый символ будет интерпретироваться как знак $, а
  // вместо $id_menu будет подставлено значение 3
  eval("\$active$id_menu = 1;");
?>
```

Код в листинге 17.4 эквивалентен определению переменной: `$active3 = 1`. Функция `eval()` позволяет сформировать переменную с любым именем.

Замечание

Не следует злоупотреблять динамическим формированием имени переменной, т. к. это приводит к снижению читабельности кода и значительно усложняет его отладку и сопровождение.

PHP предоставляет еще один способ динамического формирования переменных — через двойную последовательность символа `$`. Код в листинге 17.5 приводит к результату, аналогичному листингу 17.4.

Листинг 17.5. Альтернативный способ формирования имени переменной

```
<?php
  $id_menu = 3;
  $str = "active$id_menu"; // "active3"
  $$str = 1;               // $active3 = 1;
?>
```

Значение переменной, следующей после первого знака $, воспринимается как имя для новой переменной. В результате, если строка `$str` имеет значение `"active3"`, то имя новой переменной будет `$active3`.

Символ @ — подавление вывода ошибок

Символ `@` в PHP предназначен для подавления вывода предупреждений и сообщений об ошибках в окно браузера. Так код в листинге 17.6 в случае невозможности установить соединение с сервером базы данных, без использования символа `@` перед вызовом функции `mysql_connect()`, помимо сообщений, выводимых функцией `echo()`, выводит также предупреждение: `"Warning: mysql_connect()..."`.

Листинг 17.6. Пример использования символа @

```
<?php
  $dbcnx = @mysql_connect($dblocation, $dbuser, $dbpasswd);
  if (!$dbcnx)
  {
    exit("<p>Извините, к сожалению, не доступен сервер MySQL.
          Попробуйте посетить наш ресурс позже</p>");
  }
?>
```

Вывод предупреждений и сообщений может нарушить дизайн Web-страницы и произвести плохое впечатление на посетителя, который ожидает вместо маловразумительной фразы на английском языке получить детальные объяснения и извинения на русском. Использование символа `@` позволяет подавить вывод не только нежелательных сообщений, но и вообще любого вывода функции, включая полезный вывод.

Замечание

Следует крайне аккуратно использовать символ `@`, особенно на стадии отладки Web-приложения, т. к. чрезмерное подавление вывода может замаскировать серьезные ошибки в приложении.

Есть еще один интересный прием использования символа @, который представлен в листинге 17.7.

Листинг 17.7. Использование символа @ перед include

```
<?php
  @include "index.php";
?>
```

Такой прием позволяет подавить весь вывод в окно браузера из файла index.php.

Скрипт предзагрузки страницы

С развитием Интернета и увеличением скорости доступа посетители страницы становятся все более и более требовательны к скорости формирования контента сайта. Если посетитель вынужден ждать более 10 секунд, он просто уходит с ресурса. Однако и в настоящий момент встречаются задачи, требующие длительных вычислений, например, запросы из базы данных или обсчет ресурсоемкой задачи. В этом случае узким местом становится не пропускная способность канала, а вычисления на сервере. Для того чтобы удержать посетителя на ресурсе, часто достаточно просто вежливо предупредить его о том, что в настоящий момент производятся вычисления, требующие значительного машинного времени, и результатов этих вычислений придется подождать. Посетителя отпугивает не перспектива ожидания в течение 15—20 секунд, а неизвестность.

Для эмуляции ресурсоемкого скрипта, производящего длительные вычисления, будем использовать скрипт, представленный в листинге 17.8, который производит задержку на 5 секунд при помощи функции `sleep()`.

Листинг 17.8. "Ресурсоемкий" скрипт get.php

```
<?php
  // Эмулируем длительные вычисления
  sleep(5);       // Время в секундах
  echo "Результат работы длительного процесса";
?>
```

Вместо того чтобы сразу направлять посетителя на страницу get.php, от которой у него сложится впечатление "зависания" сайта, следует направить его на страницу index.php, код которой представлен в листинге 17.9.

Листинг 17.9. Страница предзагруки

```
<?php
  // Это файл index.php
  echo "<p>Вычисление результата может занять некоторое время.
          Пожалуйста, подождите 5 секунд</p>";
  // Осуществляем перенаправление на другую страницу get.php
  echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=get.php'>
          </HEAD></HTML>";
?>
```

В результате, если загрузить страницу index.php, пока браузер ожидает результатов вычисления сервера, посетителю будет отображаться надпись: "Вычисление результата может занять некоторое время. Пожалуйста, подождите 5 секунд", по истечении этих 5 секунд будет осуществлен редирект (перенаправление) на страницу get.php и выведена надпись "Результат работы длительного процесса".

Совет

Если на страницу предзагрузки вывести GIF-анимацию переворачивающихся песочных часов, постепенно заполняющегося секционного прямоугольника, это скрасит время ожидания посетителя, и он будет терпеливо ждать, понимая, что сервер выполняет серьезные вычисления.

Время генерации страницы

Потратив свое время на ожидание длительного процесса в ситуации, описанной в предыдущем разделе, посетитель вправе узнать время генерации страницы. Значение времени генерации страницы, выведенное после длительных вычислений, окончательно убедит посетителя в серьезности решаемых сервером задач и настроит его на ожидание при работе с таким сервисом.

Для решения данной задачи используется функция `microtime()`, возвращающая строку вида `"msec sec"`, где `msec` — составляющая в микросекундах, а `sec` — время в секундах, прошедшее с 1 января 1970 г.

Применение этой функции приведет примерно к такому результату: 0.48023600 1076713360.

Такое представление не очень удобно для вычисления разницы во времени, поэтому его необходимо преобразовать в числовой вид. Иначе говоря, строку `"msec sec"` следует преобразовать к виду `"sec.msec"`. В листинге 17.10 приведен модифицированный скрипт из листинга 17.8, вычисляющий время своей работы.

Листинг 17.10. Вычисление времени генерации страницы

```
<?php
  // Помещаем время старта в переменную $begin_time
  $part_time = explode(' ',microtime());
  $begin_time = $part_time[1].substr($part_time[0],1);
  // Здесь формируется страница после редиректа
  echo "Результат работы длительного процесса<br>";
  // Эмулируем длительные вычисления
  sleep(5);       // Время в секундах
  // Помещаем время окончания вычислений в переменную $end_time
  $part_time = explode(' ',microtime());
  $end_time = $part_time[1].substr($part_time[0],1);
  // Выводим время работы скрипта
  echo $end_time - $begin_time;
?>
```

Результат работы скрипта может выглядеть следующим образом:

```
Результат работы длительного процесса
4.98896002769
```

Задержка вывода

Часто требуется осуществить задержку вывода страницы в окно браузера. Это может быть связано как с необходимостью отправки HTTP-заголовков после использования функций `echo()` и `print()`, так и для постобработки, например, для реализации шаблонов или подсветки искомых слов при формировании результатов поиска. Для решения этой задачи предназначены функции управления выводом. Данные функции позволяют поместить весь вывод для окна браузера в буфер и отправлять страницу с сервера единым блоком. Буферизация вывода инициализируется функцией `ob_start()`, которая сообщает интерпретатору PHP, что вывод в окно браузера следует задерживать и размещать во внутреннем буфере. Получить содержимое буфера можно при помощи функции `ob_get_contents()`, которая имеет следующий синтаксис:

```
string ob_get_contents (void)
```

Функция возвращает содержимое буфера в строковой переменной.

Очистить буфер можно при помощи функции `ob_end_clean()`, которая возвращает `true` в случае успешного выполнения и `false` в противном случае. В листинге 17.11 приведен пример работы с функциями управления вывода.

Листинг 17.11. Функции управления вывода

```
<?php
  ob_start();
?>
Код гостевой книги. В теле сообщений содержатся ссылки в виде текста.
<?php
  // Инициируем сессию
  session_start();
  // Занесение содержимого буфера в переменную
  $strtmp = ob_get_contents();
  // Очистка буфера вывода и отключение буферизации вывода
  ob_end_clean();
  // Замена текстовой ссылки ссылкой HTML. Заменяются строки формата
  // http://www.ресурс и www.ресурс
  $strtmp = preg_replace("#(http://www|www)(\S+)#si",
                         "<a href=http://www\\2>www\\2</a>",
                         $strtmp);
  // Вывод модифицированного содержимого страницы в браузер
  echo $strtmp;
?>
```

В листинге 17.11 весь вывод направляется в буфер, в результате чего появляется возможность вызова функции `session_start()`, т. к. клиенту еще ничего не отправлено. После этого содержимое буфера переносится в переменную `$strtmp`, а сам буфер уничтожается функцией `ob_end_clean()`. С текстом страницы можно совершать различные манипуляции, например, осуществлять в нем замену по регулярному выражению. Отправить текст клиенту можно, воспользовавшись функцией `echo()`.

Два обработчика для одной формы

Часто требуется в зависимости от выбора пользователя вызывать вместо одного обработчика другой. При решении этой задачи существует несколько подходов. В HTML-форме можно создать несколько кнопок, а в обработчике отслеживать, какая из кнопок нажата (листинг 17.12).

Листинг 17.12. Несколько кнопок в HTML-форме

```
<form action=handler.php method=post>
  <input type=text name=name><br>
  <input type=submit name=first value='Первая кнопка'><br>
  <input type=submit name=second value='Вторая кнопка'><br>
</form>
```

В обработчике handler.php можно узнать, какая кнопка нажата (first или second), в зависимости от того установлена переменная $_POST['first'] или $_POST['second'] (листинг 17.13).

Листинг 17.13. Обработчик handler.php

```
<?php
  if(isset($_POST['first']))
  {
    // Нажата кнопка first
  }
  if(isset($_POST['second']))
  {
    // Нажата кнопка second
  }
?>
```

Второй подход к решаемой проблеме связан с динамической подстановкой имени обработчика в параметр action тега <form>. Скрипт, представленный в листинге 17.14, при первом нажатии кнопки перегружает форму, а при вторичном нажатии отправляет данные обработчику handler.php.

Листинг 17.14. Динамическое изменение обработчика

```
<?php
  // Страница называется index.php
  // Проверяем значение скрытого поля first
  if($_POST['first'] == "first")
  {
    // Если происходит перезагрузка страницы,
    // т. е. кнопку нажимаем второй раз, то подставляем
    // в качестве обработчика handler.php
    $action = "handler.php";
  }
  else
  {
    // Если кнопка не нажималась ни разу,
    // подставляем в качестве обработчика текущую страницу
    $action = "index.php";
  }
?>
<form action=<?php echo $action; ?> method=post>
  <input type=text name=name value=<?php echo $_POST['name']; ?>><br>
```

```
  <input type=hidden name=first value='Первый обработчик'><br>
  <input type=submit name=send  value='Второй обработчик'>
</form>
```

Третий подход связан с так называемыми каскадными обработчиками, когда имеется один обработчик и одна кнопка, но в HTML-форме присутствует скрытое поле или выпадающий список, в зависимости от состояния которого первичный обработчик передает управление вторичному.

Пусть имеется HTML-форма, содержащая выпадающий список, позволяющий выбрать страницу, на которую осуществляется переход (листинг 17.15).

Листинг 17.15. HTML-форма с каскадным обработчиком

```
<form action=handler.php method=post>
  <select type=text name='switchfield'>
    <option value=1>Редактирование заголовка
    <option value=2>Редактирование абзаца
    <option value=3>Редактирование ссылки
  </select><br>
  <input type=submit name=send value=Отправить>
  <input type=hidden name=id_article value=<? echo $id_article; ?>>
  <input type=hidden name=id_page value=<? echo $id_page; ?>>
  <input type=hidden name=pos value=<? echo $pos; ?>>
</form>
```

После выбора из выпадающего списка действия и нажатия кнопки данные формы отправляются обработчику, расположенному в файле handler.php, код которого представлен в листинге 17.16.

Листинг 17.16. Обработчик handler.php

```
<?php
  $action = "index.php";
  switch($_POST['switchfield'])
  {
    case 1:   // Заголовок
      $action = "addtitlform.php";
      break;
    case 2:   // Абзац
      $action = "addparform.php";
      break;
    case 3:   // Ссылка
      $action = "addanchform.php";
```

```
        break;
    }
    echo "<HTML><HEAD>
          <META HTTP-EQUIV='Refresh' CONTENT='0; URL=$action?
                              id_article=".$_POST['id_article']."&
                              pos=".$_POST['pos']."&
                              id_page=".$_POST['id_page']."'>
          </HEAD></HTML>";
?>
```

Обработчик принимает данные и в зависимости от содержания параметра `switchfield` отправляет данные методом `GET` одному из вторичных обработчиков:

- addtitlform.php;
- addparform.php;
- addanchform.php.

Счетчик загрузки файлов

Если подсчет посещаемости страниц осуществляется достаточно просто, то подсчет числа обращений на загрузку файлов с сервера кажется на первый взгляд нетривиальной задачей. В то же время именно скачивание файлов зачастую является определяющим фактором при формировании общего трафика сервера.

Данная задача имеет несколько решений: использование JavaScript, подсчет до отправки HTTP-заголовка браузеру, анализ журналов сервера. Ни один из предложенных способов не является универсальным: JavaScript поддерживается не всеми браузерами. При использовании HTTP-заголовка посетитель может установить точный путь к файлу, анализируя исходный текст HTML-страницы, и в следующий раз воспользоваться им, минуя скрипт подсчета загрузки. Доступ к журналам сервера не всегда возможен и их анализ является достаточно ресурсоемкой задачей, т. к. в них фиксируется множество дополнительной информации, которую необходимо отсеять.

В каждом конкретном случае следует искать компромиссное решение, наиболее удовлетворяющее поставленным целям и имеющимся ресурсам. В данном разделе будет рассмотрен второй вариант системы подсчетов файлов, как наиболее подходящий почти во всех случаях. Это связано с тем, что HTTP-заголовки поддерживаются подавляющим числом браузеров, а среднестатистический посетитель вряд ли будет осуществлять поиск точного пути к файлу, т. к. это не дает никаких преимуществ и выгод.

Допустим, на странице index.php имеется ссылка на файл test.zip, предназначенный для загрузки с сервера. Необходимо регистрировать каждую загрузку этого файла. В листинге 17.17 представлена схема, по которой будет разрабатываться счетчик загрузки файлов.

Листинг 17.17. Файл index.php

```
<HTML><HEAD>
<?php
  // Если пользователь перешел по ссылке закачки файла, учитываем
  // хит и отправляем файл клиенту
  if($_GET['down'] == "downloads")
  {
      // Тут размещаем код подсчета хита, например,
      // путем записи в базу данных или в файл
      ...
      // В заголовке отправляем ссылку на загружаемый файл:
      echo "<META HTTP-EQUIV='Refresh' CONTENT='0; URL=test.zip'>";
  }
?>
</HEAD>
  <BODY>
    <a href=index.php?down=downloads>Скачать</a>
  </BODY>
</HTML>
```

В ссылке, предоставляемой посетителю для загрузки файла, скрипту index.php передается параметр `down`, содержание которого будет определять, какой файл необходимо загрузить на машину клиента. Извлекая из ассоциативного массива `$_GET` значение параметра `down`, скрипт в index.php размещает в заголовках файла строку

```
<META HTTP-EQUIV='Refresh' CONTENT='0; URL=test.zip'>
```

сообщающую браузеру о необходимости принять файл test.zip. Перед выводом этой строки можно осуществить подсчет загрузки, сохранив информацию в файл или базу данных.

Вывод текущего курса валют

Источником информации об официальном курсе валюты служит сайт Центробанка Российской Федерации. Обратившись по адресу сайта Центробанка **http://www.cbr.ru/currency_base/D_print.asp?date_req=$date**, где **$data** — дата в формате `ДД/ММ/ГГГГ`, можно узнать курс валют, установленных в за-

прошенный день. К примеру, узнать каков был курс валюты на 14 января 2005 года можно по адресу **http://www.cbr.ru/currency_base/D_print.asp?date_req=14/01/2005**. В результате будет открыта страница, содержащая таблицу с курсами валют, установленными в этот день. Остается только загрузить страницу и разобрать HTML-код при помощи регулярных выражений (листинг 17.18).

Листинг 17.18. Определяем текущий курс ЕВРО и доллара США

```
<?php
  // Формируем сегодняшнюю дату
  $date = date("d/m/Y");
  // Формируем ссылку
  $link = "http://www.cbr.ru/currency_base/D_print.asp?date_req=$date";
  // Загружаем HTML-страницу
  $fd = fopen($link, "r");
  $text="";
  if (!$fd) echo "Запрашиваемая страница не найдена";
  else
  {
    // Чтение содержимого файла в переменную $text
    while (!feof ($fd)) $text .= fgets($fd, 4096);
  }
  // Закрыть открытый файловый дескриптор
  fclose ($fd);
  // Извлекаем курс доллара
  preg_match("|Доллар США[^>]*>[^>]*>([\d,\.]*)|i", $text, $out);
  echo "Курс доллара - ".$out[1];
  // Извлекаем курс ЕВРО
  preg_match("|ЕВРО[^>]*>[^>]*>([\d,\.]*)|i", $text, $out);
  echo "<br>Курс ЕВРО - ".$out[1];
?>
```

Результат работы скрипта может выглядеть следующим образом:

```
Курс доллара - 27,8677
Курс ЕВРО - 36,9414
```

Постраничная навигация

Пусть имеется простейшая таблица `tbl`, состоящая из двух полей: первичного ключа таблицы (`id`) и поля для содержимого записи (`name`) — листинг 17.19. Таблица хранит несколько сотен записей (например, URL), и задача состоит

в выводе на странице 25 записей и организации ссылок на другие страницы, содержащие остальные записи.

Листинг 17.19. Таблица tbl

```
CREATE TABLE tbl (
  id_news int(11) NOT NULL auto_increment,
  name tinytext NOT NULL,
  PRIMARY KEY (id_news)
) TYPE=MyISAM;
```

Код, организующий такой вывод, представлен в листинге 17.20.

Листинг 17.20. Постраничная навигация

```
<?php
  // Число записей на одной странице
  $pnumber = 25;
  // Устанавливаем соединение с базой данных
  include 'config.php';
  // Проверяем значение параметра start
  if(empty($_GET['start'])) $start = 0;
  else $start = $_GET['start'];
  // Выясним число новостей в таблице tbl
  $cnt = mysql_query("SELECT COUNT(*) FROM tbl");
  if(!$cnt) exit("Ошибка в синтаксисе - ".mysql_error()."<br>");
  // Поместим число записей в переменную $count
  $count = mysql_result($cnt, 0);
  // Выбираем записи из таблицы tbl
  $res = mysql_query("SELECT * FROM tbl LIMIT $start, $pnumber");
  for($i=0; $i<mysql_num_rows($res); $i++)
  {
    $order = mysql_fetch_array($res);
    echo $order['name']."<br>";
  }
  // Выводим ссылки на предыдущие записи
  if ($start != 0)
  {
    echo "<A href=http://".$_SERVER["SERVER_NAME"].$_SERVER['PHP_SELF'].
         "?start=".($start - $pnumber).">Предыдущие</A> ";
  }
  // Выводим ссылки на следующие записи
  if ($count > $start + $pnumber)
```

```
  {
    echo "<A href=http://".$_SERVER["SERVER_NAME"].$_SERVER['PHP_SELF'].
          "?start=".($start + $pnumber).">Следующие</A>";
  }
?>
```

Переменная `$start` хранит позицию, начиная с которой выводятся записи. Значение этой позиции передается по методу `GET` через строку запроса `index.php?start=5`. Поэтому в начале скрипта производится проверка: если значение не передано, переменной `$start` присваивается значение 0.

После этого базе данных отправляется запрос на выборку записей, начиная с позиции `$start`, включающий в себя не более `$pnumber` (25) записей:

```
SELECT * FROM tbl LIMIT $start, $pnumber
```

Для вывода ссылок на следующие позиции необходимо знать число всех записей в базе данных:

```
SELECT COUNT(*) FROM tbl
```

Затем в цикле `for` происходит вывод записей на страницу, а также ссылок **Следующие** и **Предыдущие**.

ГЛАВА 18

Полезные советы

Кто в будущее двинулся, держись,
Взад и вперед,
Взад и вперед до пота.
Порой подумаешь:
Вся наша жизнь
Сплошная ледокольная работа.

К. Симонов

В нашей книге "Самоучитель PHP 5"[1] существует глава "20 советов или как стать хорошим программистом". Получив от читателей много добрых отзывов на эту главу, мы решили в переработанном и дополненном варианте привести ее и в этой книге.

Примечание

Эта глава ориентирована скорее на начинающих программистов, поэтому опытные разработчики могут ее пропустить.

Взаимодействие с заказчиком

Нередко, даже те, кто изучал PHP только для того, чтобы сделать собственный сайт, в дальнейшем переходят к выполнению сторонних заказов. Поэтому мы решили включить в эту главу небольшой раздел, касающийся работы с заказчиком. Эта часть работы программиста крайне важна, потому что направлена непосредственно на оплату его труда. Независимо от того, работаете ли вы частным образом, либо в организации, от результатов ваших переговоров с заказчиком зависит ваша зарплата.

Примечание

Эта часть главы написана от первого лица, поскольку отражает только субъективные наблюдения одного из авторов, сделанные во время переговоров с клиентами, участия в различных собеседованиях и т. д.

---

[1] Кузнецов М. В., Симдянов И. В. Самоучитель PHP 5. — СПб.: БХВ-Петербург, 2004. — 560 с.

По большому счету, дело даже не в зарплате, а в самореализации. С этой точки зрения талантливый программист ничем не отличается от талантливого певца (гитариста, пианиста и т. д.). Певцу, для того чтобы "пробиться", так или иначе нужно взаимодействовать с шоу-бизнесом. Иначе о нем с большой вероятностью никто не узнает, даже если у него великолепные голосовые данные. К сожалению, во всех наших школах (высших, музыкальных и др.) практически не преподают прикладную психологию, менеджмент и т. д., а все внимание уделяют только технике (игры, программирования, выполнения различных работ). В результате на эстраде вместо талантливых исполнителей, которых приятно слушать, мы по большей части имеем разновозрастных дебилов, не знающих азов нотной грамоты, слушать которых можно только будучи в сильно нетрезвом виде. А нормальные люди чем только не занимаются, кроме того, чем должны. В программировании — то же самое. Я знаю не одного прекрасного программиста, которые программируют только вечерами, потому что днем они работают стропальщиками, каменщиками, слесарями. И часто годам к 30 спиваются.

Зададимся вопросом: почему же так происходит? Почему талантливые люди, которых в России как продавцов на центральном рынке, зачастую бедствуют, занимаются не тем, чем должны, а всякая, по нашим меркам, малообразованная шушера наводнила Олимп?

Примечание

Вы можете возразить, что в программировании не так. Что тут, чтобы чего-то добиться, все-таки нужно обладать какими-то техническими умениями. Бросьте! Каждый третий наш проект — это доделка или переделка того, что уже было начато другими "коллегами", причем сполна получившим деньги за сделанную работу.

Потому что встречают и в прямом, и в переносном смысле по одежке. Почти всегда. По походке, по разговору, по умению отвечать на вопросы. И много еще "почему", и отдельные "почему" мы попытаемся рассмотреть в этой небольшой главе.

Встречают по одежке

В прямом и в переносном смысле. Сначала о самом прямом. Подойдете ли вы в электричке к бомжу, если даже точно будете знать, что он выдающийся математик? Вряд ли. Хотя, между прочим, среди бомжей есть очень образованные люди. Я точно знаю, что на маршруте Н. Новгород — Вязники есть бомж, бывший в своей прошлой жизни аж кандидатом физико-математических наук.

Примечание

Этот пример, кстати, лишний раз говорит еще о двух серьезных вещах. Во-первых, о том, что даже очень высшее образование не гарантирует ничего. Это всего лишь одна из ступеней, которую стоит одолеть. Фундамент, после закладки которого постройка дома только начинается. Не более. Во-вторых, о том, что стать бомжем в прямом смысле слова намного легче, чем многие из нас думают. Нужно только позволить себе чуток расслабиться...

Теперь представьте, что к заказчику пришел этот бомж-математик. Ни ученая степень, ни что иное ему не поможет. Конечно, я утрирую, но, в частностях, некоторые из нас, бывает, иногда смахивают на этого бомжа. Я, к примеру, спеша на встречу в машине и куря по дороге, имею устойчивую тенденцию, пребывая в задумчивости, обсыпать себя пеплом с ног до головы. Один мой знакомый нередко при переговорах, задумавшись, переходит на свист. Другой никак не отвыкнет от привычки хрустеть костяшками пальцев в момент принятия судьбоносного решения. Это, так сказать, примеры внешних проявлений и далеко не самых худших. А сколько внутренних? Особенно хорошо они замечаются на собеседованиях.

Номер первый. "Ой, простите. Я, это, значит, к вам, ну на это, ну для собеседования". Садится на краешек стула, всем своим видом демонстрируя полную готовность немедленно упасть в обморок при любом осложнении ситуации.

Номер второй. Вытаскивает из заднего кармана брюк вчетверо сложенную и немало помятую бумажку, глядя на которую рождаются самые нехорошие предположения относительно ее происхождения. Ан, нет, — оказывается, это рекомендации.

Номер третий. При решении задания иногда исподлобья смотрит на членов комиссии взглядом, в котором такая смесь тоски и агрессивности, что впору рисовать картину "Допрос пионера-героя гестаповцами-садистами".

Номер четвертый. Забыл свою ручку, а потом в задумчивости обмуслякал ручку, любезно предложенную ему председателем комиссии.

Номер пятый. Говорит достаточно уверенно, складно, но от волнения плюется. Понятно, что более всего достается председателю, поскольку по большей части испытуемый обращает свое внимание на него.

Номер шестой. Все тестовые задания сделал правильно, но практически ничего не говорит, ограничиваясь кивками головы. На просьбу пояснить свое решение, сжал скулы и с ненавистью осмотрел всех членов комиссии. Уже дважды за сегодня пострадавший от чужих слюней председатель на всякий случай отодвигается подальше.

Номер седьмой. На вопрос, чем не устроило предыдущее место работы, честно ответил, что там начальник — придурок. Понимая, что в скором времени также будут говорить про них, члены комиссии осуждающе качают головами.

И так далее... Беседа с потенциальным клиентом от собеседования практически ничем не отличается. С кем бы вы не говорили, с работодателем или клиентом, смотрите ему в глаза и будьте спокойны и уверенны. (Понимаю, что легко сказать, да не легко сделать. Тренируйтесь. Изображайте спокойствие, и уверенность — только правдиво, — со временем действительно станете уверенными.) Говорите четко, внятно, без ненужных подробностей. Если клиент игриво сообщает вам, что после вчерашней вечеринки у него побаливает голова и он не очень соображает, не стоит отвечать ему тем же и сочувственно говорить, что вы его прекрасно понимаете, потому что вчера тоже "надрызгались вхламину". Вас могут различными способами вызывать на откровенность, — не надо говорить ничего лишнего.

Когда мы говорим "Встречают по одежке", то имеем в виду не только прямой смысл этой фразы, но и следующее: чтобы достичь осязаемых успехов, мало одних технических знаний. Нужно еще уметь "себя подать".

Заказчик — дурак?

Заказчик — дурак? Конечно, дурак. Такой же, как и мы с вами в той сфере деятельности, которой занимается он. Однако самая популярная тема для разговоров у дизайнеров и программистов — "Какой заказчик кретин, не знает, что такое домен". "Ой, что ты", — вторит ему другой, — "вчера работал с таким идиотом, который не знает даже, что такое IP-адрес!". А с какой стати он должен это знать? Увы, такая нехорошая черта характера, как профессиональный снобизм, сильно развита у многих программистов. Но наша профессия ничем не лучше и ничем не хуже других — просто одна из многих профессий. И глупо требовать от человека другой профессии (заказчика), чтобы он в ней разбирался. Представьте себе, что вы пришли на прием к платному врачу, он вам дает кардиограмму, в расшифровке которой написано, что у вас нарушение проводимости по пучку Гиса. На ваш вопрос, что это означает, врач смотрит на вас взглядом, в котором смешано презрение вашими умственными способностями с восхищением собственными. После этого он выходит в коридор и начинает жаловаться своему коллеге: "Слушай, ну у меня, что не пациент, то дебил! Представляешь, какой попался — не знает, что такое пучок Гиса! А?! Ну, как с такими можно работать?".

Рожденный ползать — уйди со взлетной полосы!

По отношению к заказчику у исполнителей крайне нередка фраза: "Я ему кучу писем послал, столько его ждал, теперь он пусть меня подождет". Такому исполнителю можно только повторить фразу, которой назван этот подраздел. Потому что никто нас ждать не будет — Web-студий сейчас на рынке, как песка в пустыне. И заказчик, когда платит деньги за сайт, рассчитывает на то, что исполнитель понимает, что на данном моменте сайт для него в лучшем

случае на 10-м месте среди предметов второй необходимости. Нет ничего плохого в том, чтобы ему напомнить о себе. Иначе, через несколько дней, заказчик вспомнит о сайте, сам вам позвонит и услышит: "А вы мне ничего не прислали-и-и, хотя я вас просил две недели назад". И не надо после этого печалиться, что заказчик разнервничался и возмущался тому, что не идет работа. А потом вообще взял и разорвал договор. Безынициативные исполнители — это кандидаты для естественного отбора.

Невербальное общение

Грубо говоря, невербальные общение — это общение без слов: жестами, мимикой и т. д. По этой теме психологами написаны целые трактаты, и обсудить даже основные моменты данной темы в этом небольшом разделе не представляется возможным. Важно следующее: наше внутреннее состояние проявляется нашими невербальными реакциями и может быть "считано" подсознанием собеседника. По этой причине, когда вы идете на встречу, если у вас нет положительного внутреннего настроя, не стоит удивляться, что встреча будет провалена. Как бы вы не "гримировались", что-нибудь да выдаст: жесты, оттенки голоса, мимика лица, поза. Обратное тоже верно. Если вы настроены на успех, хоть и немного волнуетесь (не без этого), действительно доброжелательно относитесь к вашим собеседникам, то все ваши невербальные реакции будут работать на вас.

Примером одной из самых распространенных невербальных реакций является скрещивание собеседником рук на груди при его настороженности. (Этот жест символизирует защиту. Его аналогом является сцепление кистей рук.) Когда во время разговора собеседник скрещивает руки на груди, значит, вы сказали что-то, с чем он не согласен. Даже если внешне ваш собеседник соглашается с вами, продолжать разговор бессмысленно до того, как напряжение спадет. Потому что до тех пор, пока руки остаются скрещенными, негативное отношение сохраняется. Есть простой, но эффективный метод борьбы с этой позой. Если есть возможность, подайте собеседнику что-нибудь — к примеру, ручку с блокнотом, чтобы записать вам его адрес электронной почты, который как раз время спросить. В этом случае он будет вынужден изменить позу, раскрыть ладони и наклониться вперед, т. е. вы сможете сделать собеседника более открытым. Другой способ заключается в том, чтобы попросить собеседника наклониться вперед, чтобы что-то рассмотреть, допустим, какую-то схему, которую вы начертили. Так он тоже будет вынужден отказаться от принятой позы.

Примечание

Вариант беседы, когда вы что-то рисуете или чертите и просите собеседника наклоняться и разглядывать нарисованное, является очень неплохим, т. к. во-

первых, не дает собеседнику принимать защитные позы, а во-вторых, незаметно происходит уменьшение расстояния между собеседниками.

Примечание

У многих людей, которые постоянно бывают на людях, вырабатывается замаскированный вариант защитной позы. То есть жест, целью которого является скрещивание рук, не доводится до конца, а ограничивается касанием часов, браслета, или какого-то предмета, находящегося поблизости от руки. В таком случае барьер тоже образуется, что дает человеку чувство уверенности.

Если вы часто общаетесь с одним и тем же собеседником, вы можете достаточно точно "срисовать" его индивидуальные положительные и отрицательные невербальные реакции на те или иные события. Для этого нужно искусственно вызвать в нем положительное и отрицательное настроение и запомнить его поведение в этих случаях. Наибольшее внимание следует обращать на жесты, голос, движения глаз. На первых порах развивать такую наблюдательность достаточно сложно, однако потом это воздастся сторицей. Постепенно приучившись обращать внимание на подобные "мелочи", вы в один миг вдруг поймаете себя на мысли, что, к примеру, ваш собеседник вам врет или чего-то недоговаривает. Хотя вербально (словесно, подконтрольной мимикой и т. д.) как будто бы все нормально. К примеру, некоторое время назад, обсуждая с коллегой план будущих работ, я после разговора почувствовал, что что-то меня гнетет, хотя внешне все было нормально: собеседник ругался, кривился и не соглашался со мной (в той ситуации именно так и должно было быть). На следующий день, вернувшись к разговору, я попытался, насколько смог, проследить его невербальные реакции. И понял, что меня гнетет. Они все расходились со словами: на словах он ругался, а жестами своими, интонациями голоса, движениями глазных яблок вел себя точно так же, как когда предвкушал поездку в отпуск. Своевременное выяснение причин того, чему же он так возрадовался, помогло предотвратить нежелательное для нас развитие ситуации.

Примечание

Все мы слышали о "женской интуиции". Действительно, по статистике интуиция у женщины развита выше, чем у мужчин. Психологи связывают это с тем, что женщины более четко на подсознательном уровне понимают невербальные реакции. По мнению ряда исследователей, это является одним из проявлений материнского инстинкта (женщины растят детей, которые в первые годы, а особенно месяцы жизни общаются сугубо невербально). По этой причине, мужчины, когда ваш заказчик женского пола, лукавство, особенно если ваш собеседник вас старше, на неосознанном уровне распознается очень быстро.

В любом из нас спит гений. И с каждым днем все крепче

Часто возникает вопрос: "О советах по применению тех или иных приемов общения наверняка слышал мой собеседник. Он же тоже их будет применять, и что из этого получится?" Здесь два варианта ответа. Если ваш собеседник действительно может квалифицированно общаться, то вы в любом случае в выигрыше. Потому что быстрее достигнете желаемой цели. Однако это, увы, случается не так часто. Дело здесь в том, что наличие тех или иных книг на полке не означает владение теми приемами, которые в них описаны. Наверняка у многих из ваших знакомых есть классические книги Д. Карнеги. А теперь вспомните, кто из них в совершенстве овладел ораторским искусством или "перестал беспокоиться и начал жить"? В лучшем случае процентов пять. Потому что устойчивые знания и навыки достигаются только каждодневными тренировками. К умению общаться это относится так же, как к умению программировать, драться и т. д.

Немного о виктимологии

Виктимология — это наука о жертвах, т. е. о людях, которым все время падает кирпич на голову, даже если они идут в чистом поле. Помните, у Чехова в пьесе "Вишневый сад" есть такой персонаж — Епиходов, которого все звали "тридцать три несчастья"? Вот именно таких людей и почему они стали такими, и изучает виктимология. Тема эта очень обширная даже для книги, а не только для этого маленького раздела. По своей сути вся практическая психология ставит своей целью помощь человеку перестать быть жертвой. Здесь я лишь повторю мысль, многократно высказанную в этой главе. Работайте на успех, а не на неудачу! Помните, что вы — человек успеха, а не неудачник. А от последних бегите как можно дальше, какими бы прекрасными людьми они не были. Потому что помощь жертвам — это удел высококвалифицированнейших психологов-профессионалов, а не наш с вами. А быть втянутым в орбиту жертвы очень просто.

Поэтому, когда идете на встречу с заказчиком, создавайте в себе максимально позитивный настрой. Для этого, кстати, непосредственно перед встречей не мешает вспомнить что-нибудь приятное. Если же вы идете с ощущением собственного ничтожества перед вашими конкурентами, не удивляйтесь, что вам ничего не закажут. Помните, что "если назвался червяком, не обижайся на то, что тебя раздавят".

Личная встреча — лишний шаг к успеху

Не доверяйте важные переговоры электронной почте и телефону. По возможности организовывайте личную встречу. При личной встрече у вас всегда

есть возможность сразу видеть, как реагирует собеседник на ваши слова, и быстро подстроиться под него. Есть мудрая пословица: "Хотите провалить переговоры — договаривайтесь по телефону". То же самое относится и к электронной почте. Даже после того, как договор подписан, первое время и ключевые моменты всегда следует обсуждать лично, даже если заказчик сам предлагает "переслать ему дизайн по e-mail". По электронной почте стоит общаться только для обсуждения незначительных технических деталей (т. е. уже ближе к концу работы над проектом), либо с уже знакомыми партнерами.

Не возражайте "в лоб"

Проведите над своими знакомыми такой простой эксперимент: начертите на бумаге линию, вверху напишите "Самый умный", внизу "Самый глупый". И попросите их оценить себя по этой шкале. Почти все поставят свои точки выше середины. Тест этот называется *тестом Дембо*. Его результаты означают, что почти никто на самом деле, а не на словах, не считает себя дураком. Когда вы идете на встречу, то знайте, что ваш собеседник считает себя как минимум не глупее вас. Если вам двадцать, а ему сорок, то почти всегда — умнее. Теперь, после этого краткого предисловия, представьте, что сразу в лоб возражаете на какое-то предложение вашего собеседника. Ему это не понравится, т. к. это не нравится никому. Уважайте своего клиента. Возьмите за правило, даже если вы не согласны, некоторое время подумать. После того, как ваш собеседник увидит, что к его предложению отнеслись со вниманием, ваше возражение на его слова уже не будет воспринято болезненно.

ЯЗВа

Когда-то в Интернете нашел такое определение заказчика — ЯЗВа (сокращение от "Я Знаю Все"). Увы, не помню фамилию того остроумного человека, который это придумал, но определение чудесное. Если вы столкнулись с таким типом заказчика, то в первую очередь помните, что ваша цель — все-таки получить деньги, а не учить кого-то бесплатно уму разуму. От этого и пляшите. Если же хотите вдобавок, чтобы в портфолио попал сайт, который было не стыдно показать людям — без привлечения психологии не обойтись. "Птичку вверху, чтоб крыльями махала? Пожалуйста!" А на следующей встрече можно сказать, что птичка не очень хорошо смотрится. Отложенное несогласие будет восприниматься вашим всезнающим собеседником не так болезненно, как если бы вы возразили сразу. Кстати, если вам встретился заказчик типа "ЯЗВа", это значит, что у него для вас припасено немало достоинств. К примеру, его несложно испугать. И пугайте на здоровье! Если человек принадлежит типу "Я знаю все", значит, его давно никто не пугал, и вы

просто восполняете допущенный природой пробел. Несколько примеров. Заказчик хочет Flash, вы о нем не договаривались, а заказчик типичный холерик по темпераменту и сразу рвет договор при малейшем с ним несогласии. С выражением лица "только вам по секрету" говорите ему, что Flash отрицательно влияет на индексацию поисковыми системами, и сайты с Flash индексируются в последнюю очередь. И так далее, с подробным объяснением того, чем это грозит заказчику, и парой-тройкой ужасных примеров "из жизни".

АнтиЯЗВа

Иной пример. Вы пришли к заказчику, и слышите что-то типа: "Господа, я в этом деле ничего не понимаю, вы уж на меня, дурака, внимания не обращайте". Вот в этом случае — максимум внимания и серьезности. На 50% перед вами достаточно компетентный человек, и если вы получаете от него отказ, это означает, что количество лапши на его ушах превысило предельно допустимую концентрацию. Еще на 30% — компетентный человек находится или где-то рядом, или потом прослушает запись разговора с вами. Ну и 20% — перед вами честный с самим собой и с вами человек, работать с которым просто удовольствие.

Закон об объеме оперативной памяти

Этот закон гласит, что оперативная память человека способна усвоить от среды от пяти до девяти (7 ± 2) информационных знаков, независимо от их природы. (Число семь вообще магическое: семь нот, семь цветов радуги, семь дней недели, "семеро одного не ждут", "Семь раз отмерь...".) Наша оперативная память работает около 5—20 секунд. По истечении этого времени информация либо поступает в кратковременное хранилище (где может находиться до 5—7 дней), а затем в долговременное хранилище (от 5—7 дней до бесконечности), либо забывается. Установлено, что более половины взрослых людей не в состоянии запоминать на слух предложение, насчитывающее 14 слов. Одна треть взрослых людей забывает начало фразы уже тогда, когда произносится одиннадцатое по счету слово. Если цепочка произносимых слов длится более 6 секунд, слушатели теряют нить рассуждения.

Примечание

Закон об объеме оперативной памяти называется *законом Миллера*, и был открыт в 50-х годах прошлого века группой психологов под руководством Д. Миллера во время их работы по заказу ВМФ США.

По этой причине, если вы хотите, чтобы заказчик вас понял и усвоил, что вы говорите, в разговоре произносите не более 2 слов в секунду, при этом стараясь строить фразы, состоящие из 7—9 слов.

Зная о законе оперативной памяти, понятно также, что делать, если нужно, наоборот, чтобы заказчик и окружающие мало что поняли (тоже нередкая задача).

Закон края (закон Эббингауза)

Этот закон многим известен: лучше всего запоминается то, что находится сначала и в конце (речи, текста и т. д.). Применительно к нашей ситуации, то, что вы хотите, чтобы лучше запомнилось собеседником, говорите в начале и в конце беседы. По этой же причине, будьте наиболее доброжелательными в начале и в конце беседы. Это и запомнится. И когда ваш собеседник будет потом в памяти прокручивать разговор с вами, у него останется приятное впечатление. Соответственно, старайтесь не начинать и не заканчивать беседу с выяснения "щекотливых" вопросов.

Закон контрастов

Запоминание становится более эффективным при чередовании контрастных объектов умственной деятельности, учебных предметов, идей.

Примечание

По этой причине в расписании занятий стараются чередовать дисциплины естественно-математического и гуманитарного циклов. Такое чередование балансирует работу левого и правого полушарий и способствует успешному усвоению учебной информации.

Что дает нам этот закон применительно к беседе с заказчиком? Очень много. На этом законе построена масса технологий убеждения, запоминания развития способностей. К примеру, при прямой попытке изменить точку зрения собеседника, стоящего на противоположной позиции, неминуемо приходим к конфликту. Здесь срабатывает эффект контраста — чем больше мы его убеждаем, тем он становится упрямее, а наши шансы убедить его — все меньше. Изменить позицию собеседника можно лишь терпеливым и последовательным убеждением, помня при этом, что каждый шаг в данном направлении приводит лишь к частичному (!) изменению позиции вашего партнера. Желательно делать перерывы в беседе, во время которых партнер постепенно переходит на вашу позицию, а вы частично принимаете его позицию.

Принцип ледокола или еще одно следствие из закона контрастов

Убеждая собеседника, можно поступать примерно так. Через какое-то время уводить собеседника от обсуждаемого вопроса. Потом снова к нему возвра-

щаться. Одно "но": переводить тему надо естественно, а без тренировок это сделать непросто. Если подходящего повода для смены темы нет, то желательно делать перерывы в беседе, во время которых партнер постепенно переходит на вашу позицию, а вы частично принимаете его позицию. Перерывы тоже должны быть естественными: вы можете попросить воды (в горле пересохло), попросить срочно позвонить по телефону и т. д.

Примечание

Этот прием я называю *"принципом ледокола"*. Вспомните, как работает ледокол: для того чтобы расколоть льдину, он наваливается на нее и затем отходит назад. Набирает нужную скорость и снова наваливается. И так до тех пор, пока не расколет. Это единственный способ расколоть льдину. Если просто в нее упереться и все время "работать полный вперед", это ни к чему не приведет (хорошему). А теперь вспомним, как часто мы поступаем наоборот при убеждении собеседника. При этом удивляясь тому, как он непроходимо туп, когда с нами не соглашается.

Не уходите от скользких вопросов

Точнее говоря, от них надо уходить всеми силами, но если уж неприятная тема затронута, то не уходите от нее явно. Убегая от неприятного вопроса, вы пробуждаете в собеседнике инстинкт преследователя, и следующие вопросы будут только хуже предыдущего. В конце концов, раззадорившийся собеседник, скорее всего, вам откажет. По такому сценарию срывается много собеседований. Ситуация для примера: нервный заказчик на предыдущей встрече сказал вам что-то дорисовать, вы согласились с его мнением, а дорисовать забыли. Рассмотрим несколько возможных вариантов поведения.

Заказчик: "А где тот рисунок, который я вас просил нарисовать?"

Вы: "Ой, простите, забыл".

Этой фразой вы вводите себя в роль жертвы, которую надо догнать и уничтожить. Конечно, лучше не забывать, но если такое случилось, сделайте по рецепту Д. Карнеги "из лимона лимонад".

Заказчик: "А где тот рисунок, который я вас просил нарисовать?"

Вы: "Не волнуйтесь, я о нем помню, эта идея очень хорошая и обязательно будет реализована, просто для ее полноценной реализации необходимо провести некоторые подготовительные работы".

При таком варианте вы не только отводите огонь от себя, но и делаете скрытый комплимент заказчику (фраза "эта идея очень хорошая ").

Еще вариант.

Заказчик (с раздражением, переходящим в гнев): "Я хотел бы знать, где тот рисунок, который я вас просил нарисовать?"

Вы, тоже с гневом смотря то на заказчика, то на монитор: "Я тоже!" (При таком варианте развития событий заказчик теряется, т. к. он ожидает от вас всего, чего угодно, только не гнева в его адрес.) "Я тоже хотел бы это знать!", — повторяете вы, еще больше распаляясь. И продолжаете: "Как мне надоели эти хостинг-провайдеры, не могущие настроить свои proxy-серверы, которые кэшируют все на свете!". После этого можете понизить тон и объяснить заказчику, что картинку вы поместили, но она просто не отображается, т. к. негодный proxy-сервер еще не обновил свой кэш.

Минус эмоции

Вы должны научиться бесстрастно относиться к работе с Заказчиком. Никаких эмоций! Вам важен только результат! Когда вы уже работаете над проектом, результат — завершение проекта и получение оплаты. Вы не должны испытывать к собеседнику ни жалости, ни ненависти, ни человеческой привязанности. Беседа с заказчиком — это ваша работа, а вы — машина для ее выполнения. Чувства можно позволить себе потом. При выполнении работы можно позволить себе только два чувства: радости (за хорошо сделанную работу) и злости (на себя в противном варианте).

Примечание

Чувства в бизнесе и вообще в любом деле — вещь крайне опасная. Их можно себе позволить дома, на отдыхе с друзьями, но не во время проведения переговоров. При переговорах вам важно максимально улавливать информацию и адекватно на нее реагировать. Чувства этому только мешают. Только в спокойной воде вы можете четко увидеть свое отражение, если на воде даже небольшое волнение — изображение "поплывет". Так и с эмоциями.

Немного о НЛП

НЛП расшифровывается как нейролингвистическое программирование и представляет собой одно из направлений психологии. Часть "нейро" этой аббревиатуры относится к нервной системе, а часть "лингвистическое" — к нашей способности использования речи и к тому, каким образом употребляемые нами речевые обороты характеризуют наш внутренний мир. Ну а программирование здесь появилось затем, чтобы показать, что наши мысли, чувства и действия являются результатом выполнения наших "внутренних программ", поменяв которые можно изменить свою жизнь.

Нас в этой главе из НЛП заинтересует лишь небольшая часть, которая серьезно может облегчить общение с клиентом, а именно *системы представления*. Согласно концепции НЛП мы обрабатываем информацию, пользуясь четырьмя основными системами представления:

- визуальной (зрение);
- аудиальной (слух);
- кинестетической (ощущения);
- аудиально-дискретной (внутренний диалог с самим собой).

В той или иной мере мы используем все системы представления, но у каждого из нас есть "своя любимая". Знать систему представления собеседника неплохо, т. к. подстройка к его системе помогает значительно повысить конструктивность диалога. Приведу пример. Допустим, вы склонны к аудиальной системе представления, а ваш собеседник — к визуальной. Это значит, что вы лучше всего воспринимаете информацию со слуха, а он — путем анализа зрительных образов. То есть то, что вы говорите, ему гораздо легче было бы воспринимать, если бы он видел перед собой какие-то схемы. Поэтому не поленитесь, понаблюдайте немного за вашим партнером, и постарайтесь подстроиться под его систему представления. Основные признаки этих систем мы сейчас и рассмотрим.

Примечание

В НЛП существует понятие "сенсорных предикатов", которое нам понадобится в дальнейшем. Иначе, это "слова-ощущения" или "мыслеформы". То есть, к примеру выражение "звучит неплохо" — аудиальный сенсорный предикат, характерный для человека с аудиальной системой представления.

- Визуальная система.

  Люди этого типа запоминают информацию с помощью мысленных образов, и их внимание сложно отвлечь звуками. В разговоре пользуются визуальными предикатами, к примеру: "у меня сложилось такое видение ситуации", "как мне видится, это надо сделать так-то", "итак, что мы видим". Такому человеку в разговоре лучше начертить схему и сказать: "Давайте посмотрим на эту схему". Если человек с визуальной системой долго слушает только одну речь, то его внимание начинает "плыть", что тоже нужно учитывать. Люди визуальной системы практически всегда собранны и опрятны (чего, кстати, ждут и от вас). Манера речи — быстрая, высота голоса, как правило, немного повыше нормальной, взгляд направлен чуть вверх.

- Аудиальная система.

  Люди этой системы склонны пользоваться аудиальными предикатами (в скобках для сравнения приведены визуальные предикаты): "звучит неплохо" (вместо "смотрится неплохо" для визуальщика), "что я слышу?!" (вместо "что я вижу?!"), "вы слышали, что вчера по телевизору сказали?" (вместо "вы видели..."). Люди с аудиальной системой говорят ритмично,

с богатыми голосовыми интонациями. При обдумывании проблемы склонны отводить глаза в сторону, как бы прислушиваясь к чему-то. Аудиальщики легко усваивают информацию на слух и предпочитают слышать отклик собеседника на свои высказывания.

- Кинестетическая система.

В речи кинестетики склонны пользоваться кинестетическими предикатами, обозначающими различные ощущения: "меня в озноб ударило от этой мысли", "все идет гладко", "я получил заряд бодрости", "это нагоняет тоску", "проблема обрушилась на меня" и т. д. Когда кинестетик пребывает в задумчивости, взгляд его направлен, обычно, вправо вниз. Темп речи в спокойном состоянии медленный, нередко с большими паузами между словами. Кинестетики хорошо реагируют на дружеские прикосновения, похлопывания по плечу, и сами могут себе такое позволять. Процесс запоминания нового происходит путем мысленного повторения всего процесса (к примеру, если надо запомнить основные моменты беседы, кинестетик мысленно "проиграет" ее всю от начала до конца). В общем, это человек ощущений. Кстати, такие люди неплохо относятся к решению различных вопросов во время застолий, совместных походов в баню и т. д.

- Аудиально-дискретная система.

Люди с такой системой представления много времени уделяют мысленному общению с самими собой. Для них важно только то, что имеет логику. Обороты их речи достаточно сложны. В речи любят пользоваться обилием подробностей. Как правило, очень эгоцентричны. Не дай вам Бог такого человека похлопать по плечу, или предложить "порешать вопрос в баньке" — такого обращения они не терпят, считая это недопустимым панибратством. Подобные методы они могут позволить лишь только очень узкому кругу близких людей. Во время внутреннего диалога смотрят, как правило, влево вниз.

Примечание

Не знаю, везение это или нет, но мне, как правило, нередко попадаются люди именно с этой системой представления. С одной стороны, — везение. Потому что это, в основном, очень глубокие, неординарно мыслящие, натуры, встреча с которыми — счастье. Но, как говорится, чем более блестяща одна сторона медали, тем более темна другая: если человек, обладающий такой системой представления, излишне самозациклен, то общение с ним никакого удовольствия не доставляет. Потому что никто, кроме него, ему не интересен. Да, он может быть крайне эрудирован, но мало кому хочется выслушивать незапланированные двухчасовые лекции на тему "Ах, неужто, вы не читали дневников Набокова? Напрасно, напрасно". И говорит он это, конечно, не для того, чтобы вас просветить, а для того, чтобы продемонстрировать собственную эрудицицию. Остановлюсь еще на двух отрицательных моментах, характерных для излишне

самозацикленных аудиодискретчиков. Это совершенно потрясающие хронофаги (пожиратели времени). С ними можно говорить три часа и ничего полезного не вынести. Вы узнаете их мнение по поводу палестино-израильского конфликта, будете все знать об их болезнях и сопутствующих им душевных переживаниях, но — ничего по делу. Поэтому берегите свое время и не давайте уводить себя в сторону. Кроме всего прочего, после такого разговора вы еще будете выжаты как лимон и раздосадованы на то, что столько времени пропало даром. Другой момент состоит в том, что такие люди как магнитом притягивают к себе различные неприятности, рискуя и вас затянуть в свою жертвенную орбиту *(см. разд. "Немного о виктимологии" ранее в этой главе)*. Но, возвращаясь назад, повторюсь, что если вам встретится не самозацикленный аудиодискретчик, а действительно глубокий человек, это — подарок судьбы.

О разработке программных продуктов

А теперь приведем советы, касающиеся методологии разработки программного продукта.

Ни дня без кода!

Этот совет помещен одним из первых, потому что только постоянная практика и совершенствование своего мастерства делает людей профессионалами. Особенно это относится к тем, кто только начал обучаться программированию. Возьмите себе за правило кодировать и изучать что-то новое если не каждый день, то не менее 5 раз в неделю. Только так можно привить себе те основные качества, которые отличают профессионалов от любителей — автоматизм, умение четко рассчитывать и оценивать свои действия и профессиональную интуицию.

Не занимайтесь необдуманным копированием!

Одна из фатальных ошибок, которую допускают начинающие программисты, — бездумное копирование чужого кода в свои программы. Нужно писать свои программы, а не заниматься многосуточными поисками по Интернету фрагментов чужого кода, которые потом вставляются в вашу программу. Дело здесь даже не в том, что такая программа если и будет работать, то крайне неэффективно и ее невозможно расширять. И не в том, что нельзя использовать чужой код. Самое главное в том, что при таком подходе вы никогда не станете программистом! А использование чужого кода в смысле изучения его работы очень даже приветствуется. Смотреть, как работает чужой код, и строить на его основе собственные решения — это очень неплохой вариант обучения программированию.

Не пишите в стол!

Даже если вы очень любите программирование и способны им заниматься только потому, что вам это очень нравится, писать программы лишь для себя не стоит, по крайней мере по трем причинам. Во-первых, человеку все-таки нужно, чтобы результатами его труда пользовались другие люди. И программы пишутся для того, чтобы ими пользовались. Иначе на определенном этапе станет просто неинтересно, и возникнет состояние, которое можно назвать "кризисом жанра" или "комплексом неудачника". Конечно, очень приятно, когда за созданный программный продукт получаешь неплохие деньги. Но не менее приятно, даже когда твоей программой люди просто пользуются, и иногда ее даже хвалят. Давайте не будем отказывать себе в этих удовольствиях. Во-вторых, когда вы пишете только для себя, на определенном этапе вы перестанете идти в ногу со временем, и, когда вдруг все же решитесь сказать миру о себе, можете с удивлением обнаружить, что ваш поезд уже давно ушел, рельсы заросли травой, а вы один стоите на заброшенном перроне... И то, что вы делаете уже давно, никому не нужно. Согласитесь, обидно и горько оказаться в такой ситуации. И в-третьих, когда вы варитесь в собственном соку, то можете не увидеть ошибок, на которые указали бы вам пользователи или ваши коллеги.

Поэтому, если даже пока вы не можете брать за свои программы деньги, все равно ориентируйтесь на пользователя! Разместите в Интернете свой сайт, на котором выкладывайте ваши программы, и просите пользователей писать отзывы. В общем, работайте для людей!

Создавайте рабочие программы!

На первый взгляд, этот совет может показаться надуманным, однако это, к сожалению, не совсем так. Некоторые программисты настолько стремятся к красоте кода и к своему видению внутренней гармонии программы, что забывают о том, что программа в первую очередь должна работать. Помните, что программист — это не тот, кто пишет красивый код, а тот, кто пишет работающие программы.

Не бойтесь отладки

Немало программистов, которые любят кодировать, но всеми фибрами души ненавидят отладку. Эта ошибка из разряда очень плохих, потому что отладка является более важной стадией в программировании, чем собственно кодирование. Если вы не научитесь отлаживать свои программы, можете забыть о программистской карьере.

Тестируйте приложения

Тестируйте работу программы в самых различных ситуациях, и желательно, в условиях, максимально приближенных к реальным. Лучше узнать о неполадках в программе на этапе тестирования, чем от разгневанного заказчика, который из-за брешей в системе защиты вашей программы понес ощутимый материальный урон.

Ориентируйтесь на пользователя!

Одной из самых распространенных ошибок при планировании и реализации проектов является исключение пользователя из процесса создания программного продукта. Поэтому нередки ситуации, когда при сдаче проекта оказывается, что создано совсем не то, что хотел бы видеть заказчик. Не допускайте подобной ошибки и старайтесь как можно чаще показывать заказчику промежуточные результаты работы над проектом.

Читайте книги!

Вложение средств в свои знания — одно из самых выгодных капиталовложений. Сюда же можно отнести пожелание посещать различные тематические конференции, форумы и т. д.

Не забывайте про проектирование!

Проектирование — одна из самых важных частей разработки программного обеспечения. В классической книге Ф. Брукса[1] говорится о том, что проектирование должно занимать не менее одной трети от общего времени всей работы над проектом, а кодирование — одну шестую этого времени. Все остальное время отводится на отладку и тестирование, т. е. проектированию надо уделять большую часть времени работы над проектом, а кодирование должно превращаться просто в перенос того, что спроектировано, на конкретный язык программирования. Зачастую же происходит так, что проектированию вообще не уделяется времени, и разработка проекта начинается непосредственно с этапа кодирования.

Более того, имеет место общепризнанное мнение, суть которого можно выразить словами: "ошибки в кодировании неприятны, ошибки в проектировании — фатальны". Действительно, ведь большинство программных продуктов создается под конкретного заказчика, который платит деньги за ваш про-

[1] Брукс Ф. Мифический человеко-месяц или как создаются программные системы. — СПб.: Символ-Плюс, 2000.

граммный продукт. И если некоторые ошибки в кодировании приводят, к примеру, к медленной работе программы, то это неприятно, но поправимо. Вы можете исправить ошибки во время гарантийного обслуживания и т. д. А неправильное проектирование приводит к самому неприятному — полному срыву всех работ по проекту. Много потенциально хороших специалистов "выпало из обоймы" по одной причине — неумению сдавать работу в срок.

Примечание

Существует хорошее определение слова "профессионал", которое гласит, что профессионал — это специалист, который получает деньги за свою работу. Вы можете блестяще кодировать, но если вы не умеете рассчитывать свои силы и не в состоянии подстраиваться под этот быстро меняющийся мир, вас просто вытеснят с рынка, и о том, что вы прекрасный специалист, просто никто не узнает.

Четко оценивайте время работы над проектом!

Одна из самых распространенных и опасных ошибок состоит в неумении многих программистов оценивать время, необходимое для разработки проекта. Результатом неправильной оценки временных затрат является срыв сроков работы над проектом, недовольство заказчика, а иногда и штрафные санкции. Поэтому постарайтесь четко спланировать все этапы разработки, определить время реализации каждого из них и все просуммировать. Затем полученный результат умножьте на коэффициент, учитывающий непредвиденные вами факторы. Это и будет более-менее реальным временем разработки проекта. Для кого-то этот коэффициент равен двум, для кого-то трем, но почти ни для кого не равен единице. Определите для себя свой собственный поправочный коэффициент, и вам с большой долей вероятности не придется, глядя в пол, оправдываться перед заказчиком за сорванные сроки.

Не занимайтесь взломом

Этот совет адресован тем, кому не дают спокойно спать лавры героев романтически-криминальных романов. Не занимайтесь хакерством, — в реальности все куда прозаичнее и жестче. А вот изучать, как хакеры действуют, — дело нужное, т. к. лучший способ защиты от нападающего состоит в том, чтобы понять его образ мыслей. Но целенаправленно что-то ломать вряд ли стоит. Не нужно наивно думать, что найти взломщика — это большая проблема. Очень многие остаются безнаказанными по одной простой причине — лень тратить время и силы на поиски и наказание. Но если уж кого-то достают особенно сильно — поверьте, безнаказанным это не остается. И, как правило, наказание не афишируется. Поэтому и создается ложное впечатление, что хакеров у нас никто не ловит, и можно творить все, что душе угодно. Если уж

задумали что-нибудь взломать, то давайте себе отчет, что игры эти опасны. Я неоднократно видел, сколь жалкими становятся эти кибершпионы, когда их прижимают соответствующие инстанции или те, кого они достали... А еще более жалкими — их мамы и папы. И это еще один из аргументов.

Точно так же надо понимать, что мысли типа "вот еще немного и брошу" редко бывают состоятельными. Лучше не начинать. Мой знакомый врач сказал примерно так про наркоманов: "Я знал одного наркомана, который закончил хорошо. Он вылечился... и умер только через 5 лет". Это я к тому, что ваши детские шалости не могут пройти бесследно, даже если вы "завяжете" с этим ремеслом. Точно так же, как бывшему наркоману надо прыгнуть через три головы, чтобы хоть как-то восстановить свои подсаженные органы, так и вас в будущем могут настигнуть "залпы давно смолкнувших орудий". Примеров много. Один мой знакомый баловался этим делом на втором курсе своего обучения в вузе. Попался! Чуть не отчислили, родители по крохам собирали на штраф... Но все это — ерунда. Уже окончив институт и работая вполне мирным системным администратором, он столкнулся с ребятишками, попросившими его вспомнить "дела давно минувших дней" и поработать на них. Серьезно так попросили, убедительно. Если вкратце, то безопасность той организации, которую он когда-то взламывал, обеспечивал полукриминальный ЧОП (Частное Охранное Предприятие). Уже не было ни этой организации, ни ЧОПа, а люди, которые взяли его на заметочку, остались. И вспомнили, спустя немалый промежуток времени. Причем, надо сказать, взломщик он был так себе, да и звали его не за этим. Он просто должен был взять в случае чего на себя вину (естественно, о том не подозревая) вместо того человека, который действительно стал бы заниматься взломом. Этому парню повезло, поскольку у него нашлись неслабые знакомые, которые смогли за него заступиться. Что бывает достаточно редко.

Предложить что-нибудь взломать вам могут не обязательно "братки", а вполне приличные люди, занимающие высокие должности в серьезных организациях. Таких примеров тоже не мало. Не под каким соусом на это не соглашайтесь! Даже если вас просит об этом ваш непосредственный руководитель. Помните, что в 90% случаев все будет обставлено так, словно вы это сделали по собственной инициативе. Но даже если и удастся доказать, что вас к этому кто-то принудил, — вам, по большому счету, это мало поможет.

Технические советы

Эта группа советов тоже немаловажна, поскольку несоблюдение некоторых из них приводит к таким же печальным последствиям, как и ошибки в проектировании.

Не стоит недооценивать PHP

При поверхностном знакомстве с PHP возникает ощущение его простоты, т. к. он, являясь скриптовым языком, позволяет решать многие задачи меньшей кровью, чем компилируемые языки программирования. Это обманчивое впечатление, у PHP имеется много уровней, на которых можно работать, начиная от самого простейшего кода и заканчивая виртуозно закрученным в хакерском стиле.

При более глубоком знакомстве новички в PHP начинают чувствовать, что почва уходит из-под ног, а рядом имеется что-то большое и глубокое и оно почему-то проходит мимо...

Дело в том, что PHP — достаточно новый и динамично развивающийся язык, каждая последующая версия буквально перечеркивает предыдущую. В таких условиях имеющаяся документация быстро устаревает, а специалисты предпочитают не писать книг, поскольку информация, размещенная в книге, начинает устаревать до ее выхода в свет. Авторы не раз поминали крепким словцом разработчиков PHP при написании предыдущих книг, когда внесенные изменения в язык перечеркивали написанное в книгах.

Почему же PHP так популярен и каким образом столько людей разом обучилось PHP? Дело в том, что большинство PHP-программистов не изучало язык с нуля, а постигало его либо уже зная C/C++, либо Perl. Perl является C-подобным языком программирования (как и PHP) и предшественником PHP, можно сказать, что PHP — это Web-ориентированный Perl. Поскольку PHP просто логически следует из Perl и широкое распространение получил именно из-за того, что программистам было легко переходить с Perl на PHP (ну и с C на PHP тоже). Следует отметить, что Perl-сообщество является своеобразным миром, со своей культурой, этикой и огромным багажом документации, поэтому многие вещи, которые заимствованы из Perl, в PHP не находят нужным даже пояснять.

Замечание

Язык Perl появился в 1986 г. по воле системного программиста Ларри Уолла. Созданный первоначально как средство обработки текстовых файлов, призванное облегчить жизнь системному администратору UNIX, он превратился в настоящий язык программирования. Традиционные области, в которых Perl применяется особенно часто и успешно, — создание приложений CGI, системное администрирование UNIX, обработка текста. Сам Ларри Уолл кроме всего прочего является лингвистом, поэтому его язык задумывался как наиболее близкий к естественному.

Кроме того, свой вклад вносит предметная область — программирование для сети Интернет, для серьезного понимания которого желательно знание очень

многих вещей (HTML, SQL, UNIX, прикладных протоколов — хотя бы HTTP, стилей программирования на C, Perl, Java). Основной вклад вносит среда UNIX, которой буквально пропитан PHP, да и вообще программирование для Интернета. Если вы хотите познать то большое, которое ускользает от вас, — поставьте в качестве второй операционной системы Linux — многие непонятные до этого моменты прояснятся.

Не пренебрегайте словами "хороший стиль программирования"!

К сожалению, еще немало программистов при фразе "хороший стиль программирования" часто презрительно усмехаются. Не повторяйте их ошибок и не относитесь беспечно к этим словам, поскольку они выстраданы потерянными нервными клетками тех программистов, которые, небрежно относясь к своему коду, в конце концов, сами в нем запутывались и срывали проект. Не имеет значения, какой стиль будет использован при написании Web-приложений — главное придерживаться его от начала до конца. Пусть стимулом вам будет тот факт, что следование правилам хорошего тона позволяет увеличить скорость разработки и сопровождения приложений на 30%.

Расстановка фигурных скобок и отступы

Существует несколько стилей расстановки фигурных скобок. Все они диктуются существующими стилями в других C-подобных языках программирования.

- *Рациональный стиль.*

  Это один из наиболее распространенных стилей, т. к. им пользовались Керниген (Kernighan) и Ричи (Ritchie), авторы языка C.

  Пример — в листинге 18.1.

Листинг 18.1. Рациональный стиль программирования

```
<?php
  if($flag){ echo "Hello, World!"; }
?>
```

Преимущество этого подхода заключается в экономии вертикального пространства, жизненно важного при отладке большого блока кода. Оборотной стороной такого подхода является то, что может оказаться трудным найти символ {, спрятанный в конце строки. Этого стиля придерживаются и Java-программисты, как-то предписывает корпорация Sun.

- *Стиль Алмена.*

  Эрик Алмен (Eric Allman) написал утилиты BSD в этом стиле, поэтому данный стиль часто называют "стилем BSD" (листинг 18.2).

Листинг 18.2. Стиль Алмена

```
<?php
  if($flag)
  {
    echo "Hello, World!";
  }
?>
```

Аргументом в поддержку такого стиля является тот факт, что область видимости блочного оператора ясна и визуально ассоциируется с управляющим оператором.

- *Стиль Whitesmith.*

  Данный стиль предписывает использование такой расстановки фигурных скобок, которая представлена в листинге 18.3.

Листинг 18.3. Стиль Whitesmith

```
<?php
  if($flag)
    {
    echo "Hello, World!";
    }
?>
```

Этот стиль имеет преимущество в том, что скобки более тесно ассоциируются с кодом, который они включают и разграничивают, однако при визуальном просмотре текста отыскать скобки оказывается чуть более сложно.

- *Стиль GNU.*

  Программисты GNU фонда Free Software Foundation используют следующий стиль расстановки фигурных скобок (листинг 18.4).

Листинг 18.4. Стиль GNU

```
<?php
  if($flag)
```

```
    {
      echo "Hello world!";
    }
?>
```

Внутри любых управляющих конструкций операторы следует располагать с отступом на одинаковое число пробелов, например, для операторов `if-then-else` код должен выглядеть так, как представлен в листинге 18.5.

Листинг 18.5. Стиль GNU для операторов `if-then-else`

```
<?php
  $flag = true;
  if($flag)
  {
    echo "Переменная равна true";
    exit();
  }
  else
  {
    echo "Переменная равна false";
    exit();
  }
?>
```

Количество пробелов может быть любым, обычно используют 2, 4 или 8 пробелов. Старайтесь придерживаться этого правила, некоторые программисты приходят в бешенство, когда это число не кратно 2. Наиболее оптимальным является использование 2-х пробелов, т. к. при их большем числе вложенные блоки становятся "растянутыми" и их сложно воспринимать.

Пробелы вокруг символов

Бинарные операторы следует обрамлять пробелами (листинг 18.6).

Листинг 18.6. Обрамление бинарных операторов пробелами

```
<?php
  // Неправильно
  $a=$b+$c*$d;
  // Правильно
  $a = $b + $c * $d;
?>
```

Символ пробела ассоциируется с новым словом, поэтому формула читается не как непонятный набор символов, а как нечто осмысленное.

Комментарии

Расставляйте комментарии по принципу "чем больше, тем лучше" — пройдет некоторое время, и вы забудете, что делал тот или иной программный блок. Вообще принято комментировать код на английском языке или не комментировать вообще, т. к. для русского языка существует масса кодировок, да и вообще так исторически сложилось. Не обращайте на это внимания — код вы комментируете в первую очередь для себя, а не для других! А раз уж вы делаете это для себя, делайте это в удобной для вас кодировке.

PHP собрал в себе практически все комментарии современных языков программирования, наряду с однострочными комментариями в стиле shell-скриптов (#) и C++ (//) — листинги 18.7 и 18.8.

Листинг 18.7. Однострочный комментарий в стиле shell-скриптов

```
<?php
  # Программный модуль index.php
  echo "Hello world!";
?>
```

Листинг 18.8. Однострочный комментарий в стиле C++

```
<?php
  // Программный модуль index.php
  echo "Hello world!";
?>
```

Можно использовать многострочный комментарий в стиле C (листинг 18.9).

Листинг 18.9. Многострочный комментарий в стиле C

```
<?php
  /* Это многострочный комментарий в стиле C.
     Он охватывает несколько строк - не допускается
     вложенных комментариев
  */
  echo "Hello world!";
?>
```

К хорошему стилю программирования относится использование однострочных комментариев для короткого комментария, а многострочного — для

комментария, охватывающего несколько строк. Не возбраняется использовать однострочные комментарии для большого текста, особенно в начале файла или важного блока кода (листинг 18.10).

Листинг 18.10. Использование однострочного комментария для большого текста

```
<?php
  ////////////////////////////////////////////////////////
  // Гостевая книга
  ////////////////////////////////////////////////////////
?>
```

Как и при работе с отступами и фигурными скобками, основным требованием является необходимость придерживаться одного стиля во всех программных блоках.

При расстановке однострочных комментариев возможны два варианта: непосредственно перед выполняемым оператором (листинг 18.11) и после точки с запятой (листинг 18.12).

Листинг 18.11. Однострочный комментарий *перед* оператором

```
<?php
  // Вывод текстовой строки в окно браузера
  echo "Hello world!";
?>
```

Листинг 18.12. Однострочный комментарий *после* точки с запятой

```
<?php
    echo "Hello world!"; // Вывод текстовой строки в окно браузера
?>
```

Лучше придерживаться первого правила, т. к. во втором случае строка получается длинной и плохо воспринимается читающим. Единственным оправданием использования такого комментария является комментирование закрывающейся скобки длинного программного блока, содержащего много вложенных блоков (листинг 18.13).

Листинг 18.13. Комментирование закрывающейся скобки длинного программного блока

```
<?php
  if($tot)
```

```
{
    while($position = next($tot))
    {
        /* Очень длинный код,
           содержащий много
           вложенных блоков
           ...
        */
        if($flag)
        {
          echo "Ошибка";
          exit();
        }
    } // Конец while($position = next($tot))
}
?>
```

Имена переменных и функций

Существует несколько стилей названия переменных:

- `$var_bell` — стиль C: нижний регистр, знак подчеркивания;
- `$VarBell` — стиль Pascal: каждая подстрока в названии начинается с прописной буквы;
- `$varBell` — стиль Java: первая подстрока начинается со строчной (маленькой) буквы, все последующие — с прописной.

Не имеет значения, какой стиль будет вами выбран — главное придерживаться в коде одного стиля.

Совет

В программировании константы традиционно записываются в верхнем регистре `YANDEX_BOT`. Если вы хотите, чтобы другие программисты могли легко воспринимать ваш код, придерживайтесь этого правила.

Не шутите с именами!

Непродуманная система имен переменных и функций проекта является одной из серьезных ошибок программистов, причем не только начинающих. Называйте переменные и функции осмысленно — этим вы сэкономите много времени и себе, и вашему коллеге, который работает с вами в команде. От того, как вы назовете переменные и функции, зависит читабельность вашего кода. Имя переменной (функции) должно отражать характер ее содержания (действия). Не используйте в названиях одному вам понятные сокращения — не

пройдет и месяца, как вы сами забудете их смысл, не говоря уже о других людях, которые его не знали изначально. Крайне сложно отлаживать код, в котором переменные имеют имена a1, dcr1_x15, nm. Старайтесь, чтобы имена были не слишком короткими, но и не слишком длинными. Названия функций типа MyNewFunctionForReadInformationFromForm тоже не способствуют улучшению читабельности кода. Хорошим правилом считается, что имя переменной должно содержать не больше двух слов и 10 символов. Старайтесь эти слова отделять друг от друга символом подчеркивания или выделять заглавными буквами. Согласитесь, когда переменная имеет имя $user_address, это удобнее чем просто $useraddress. По возможности, избегайте употребления числительных в именах. Сложно работать с кодом, в котором переменные имеют имена $str1, $str2, $str3 и т. д.

Не забывайте, что имена переменных в PHP *регистрозависимы* и переменные $user_address и $User_address — разные переменные. Не используйте в программе переменные, которые различаются только регистром букв — это часто приводит к ошибкам и усложняет восприятие кода.

Сохраняйте файлы проекта под осмысленными именами. Работать с такими именами намного проще, чем с ничего не говорящими названиями типа sbw1fg5.php. Кроме того, у вашего заказчика может возникнуть глубокое недоумение, когда, нажав на какую-либо ссылку, он увидит страницу с загадочным и интригующим названием **http://www.myserver.com /r762vvGh178a9a.php**.

Помните, что необдуманная система имен в том случае, если вы программируете один, добавит вам несколько десятков часов бессмысленной работы, а если вы работаете в команде, небрежное обращение с именами — просто элементарное неуважение к вашим коллегам и их труду.

Рассмотрим несколько примеров. Часто временные строки для хранения SQL называют $query (запрос) — это очень удачное название, ассоциирующееся именно с SQL-запросом. Обычно на этом все и заканчивается. При появлении второго запроса, вторая переменная получает имя $query1 — это порочная практика. Обычно запросы в рамках одного скрипта отличаются своим действием: один SQL-запрос может выполнять обновление (UPDATE), другой выборку (SELECT), поэтому переменные лучше называть с указанием действия оператора $query_update и $query_select соответственно.

Другим примером может служить код из листинга 18.14.

Листинг 18.14. Ошибочное именование переменных

```
<?php
  $query = "SELECT * FROM catalog";
  $query1 = mysql_query($query);
```

```
  while($result = mysql_fetch_array($query1))
  {
     /* Код обработки */
  }
?>
```

Здесь дескриптор запроса, возвращаемый функцией `mysql_query()`, назван `$query1`, это здорово запутывает как разработчика, так и читающего текст программы. Дескриптор не несет физического смысла — это лишь ключ к результату, поэтому его лучше называть сокращенным именем таблицы (листинг 18.15).

Листинг 18.15. Правильное именование переменных

```
<?php
  $query = "SELECT * FROM catalog";
  $cat = mysql_query($query);
  while($catalog = mysql_fetch_array($cat))
  {
     /* Код обработки */
  }
?>
```

Структурируйте ваш код

Старайтесь как можно четче структурировать свой код:

- в каждой строке кода пишите только один оператор;
- не пренебрегайте комментариями — через некоторое время вы забудете, что делает та или иная функция;
- структурируйте код при помощи отступов;
- оформляйте код в виде отдельных функций, при этом каждая функция должна выполнять одну конкретную задачу;
- не перегружайте головной файл проекта дополнительным кодом — все необходимые действия делайте в отдельных модулях программы, а в головном файле размещайте вызовы требуемых функций или методов класса.

Соблюдение этих правил сделает код более читабельным и сэкономит много времени при дальнейшем сопровождении.

Не делайте средствами PHP то, что можно сделать с помощью СУБД

При работе с базами данных не перекладывайте те задачи, которые можно решить средствами СУБД, на код PHP. Если для поиска информации в базе данных будут использоваться средства PHP, а не SQL — не стоит удивляться, что скрипт работает медленно. Язык SQL был специально разработан для того, чтобы его средствами отсеивать ненужные вам данные и получать только те, которые действительно необходимы для дальнейшей обработки. Если при выборке получаете больше данных, чем вам нужно, — это верный признак того, что вы не до конца продумали SQL-запросы.

Рассмотрим пример кода, в котором выводится список адресов электронной почты пользователей с определенным `id` (листинг 18.16).

Листинг 18.16. Вывод списка адресов пользователей с определенным `id` (средствами PHP)

```
<?php
  // Установление соединения с базой данных
  // $dbcnx - дескриптор соединения
  $query = "SELECT name, address, id FROM mail_table";
  $result = mysql_query ($query, $dbcnx);
  if (!$result) exit ("Ошибка выполнения запроса");
  if (mysql_num_rows ($result) <= 0) echo ("Получено ноль значений");
  while ($row = mysql_fetch_array ($result))
  {
    if ($row[id] == 5)
    {
      print "Имя: $row[name]\n<br>\n";
      print "e-mail: $row[address]\n<br>\n";
      break;
    }
  }
?>
```

Этот код имеет большой недостаток — поиск по базе данных реализуется средствами PHP, что при большом объеме данных приведет к неминуемому снижению скорости. Выход очень прост — для поиска использовать средства SQL, добавив в SQL-запрос оператор `WHERE`, а PHP применять только для вывода найденной информации (листинг 18.17).

Листинг 18.17. Вывод списка адресов пол зателей с определенным id (средствами SQL)

```
<?php
  // Установление соединения с базой д ых
  // $dbcnx - дескриптор соединения
  $query = "SELECT name, address, id F M mail_table WHERE id=5";
  $result = mysql_query ($query, $dbcn ;
  if (!$result) exit("Ошибка выполнени запроса");
  if (mysql_num_rows ($result) != 1)
    echo ("Ошибка - неверное количество рядов");
  $row = mysql_fetch_array ($result);
  print "Имя: $row[name]\n<br>\n";
  print "Телефон: $row[address]\n<br>\n";
?>
```

Делайте свои скрипты устойчивыми к ошибкам

Слабое тестирование на устойчивость скриптов к ошибкам — беда многих начинающих программистов. При недоработках такого рода может возникнуть множество неприятных ситуаций. Для того чтобы их избежать, обязательно проверяйте результаты вызова всех используемых вами функций на предмет возможных ошибок, чтобы в случае возникновения последних корректно завершить работу программы (листинг 18.18).

Листинг 18.18. Проверка результатов вызовов используемых функций

```
<?php
  $dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
  if (!$dbcnx)
  {
    echo "<p>К сожалению, не доступен сервер mySQL</p>";
    exit();
  }
  if (!@mysql_select_db($dbname,$dbcnx))
  {
    echo "<p>К сожалению, не доступна база данных</p>";
    exit();
  }
?>
```

По крайней мере, на этапе отладки выставите наивысший уровень директивы отображения сообщений об ошибках `error_reporting` (`error_reporting = E_ALL`) в файле настроек php.ini. При ином значении этой директивы многие ошибки ускользнут от вашего внимания.

Взаимодействуйте с д ими программистами

Когда дело доходит до взлома, зек проявляет всю заложенную в него природой сообразительность. В т время, когда дело доходит до защиты, программист демонстрирует верх рофессионализма, т. к. ему чисто психологически не хочется ломать св обственное творение. Не доверяйте себе — приглашайте сторонних люде пообещайте им вознаграждение за найденную ошибку — это вам окупитс

Не злоупотребляйте регулярными выражениями

Регулярные выражения представляют собой мощный инструмент для поиска данных и проверки корректности информации, вводимой пользователем. Однако надо помнить, что регулярные выражения работают намного медленнее других функций PHP. Поэтому не используйте регулярные выражения без надобности и по возможности ищите им альтернативу.

Не "изобретайте велосипед"

Многое из того, что вам может понадобиться, содержится в многочисленных функциях PHP. Однако нередко приходится видеть, как программисты попросту "изобретают велосипед" — пишут километры собственных функций для действий, которые уже давно запрограммированы и внесены в стандартные функции PHP. Потратив некоторое количество времени на более подробное изучение языка и используя стандартные функции, вы избавите ваших пользователей от необходимости томиться в ожидании, пока ваш скрипт выполнит свою работу.

Не используйте без надобности функции форматного вывода

Используйте функции форматного вывода только по их прямому назначению — для форматного вывода. Не вызывайте их для простого вывода данных, т. к. это в разы замедлит работу вашего скрипта.

Не используйте устаревшие конструкции языка

Масса программистов до сих пор пользуется многими конструкциями языка PHP 3. Существуют, по крайней мере, две причины, почему этого не стоит делать. Во-первых, ваш код просто не будет никто читать и использовать. Особенно это неприятно, если вы работаете в команде. И во-вторых, в будущих версиях языка устаревшие конструкции могут быть просто отменены.

ПРИЛОЖЕНИЕ 1

HTML, XTML, DHTML.

Что такое XHTML?

XHTML — это новая редакция языка HTML, записываемая в соответствии с синтаксическими правилами языка XML. XML (eXtensible Markup Language) — это расширяемый язык разметки. В отличие от языка HTML, в нем можно создавать собственные теги и формировать структуру документа. XHTML имеет более строгие правила верстки, в отличие от языка HTML, который допускает значительные вольности при кодировании. Браузеры, поддерживающие HTML, склонны прощать Web-разработчикам незначительные ошибки при верстке HTML-страниц. XML-парсер, который просматривает XHTML-страницы, не пропустит ни одной найденной ошибки и выдаст сообщение о ней на экран. Таким образом, при создании XTML-страниц на Web-разработчика накладываются дополнительные требования к синтаксической корректности HTML-кода.

Далее приведены основные правила, которые следует соблюдать при верстке XHTML.

Обязательные теги

В начало каждого документа должно быть помещено объявление о его типе DOCTYPE. Это может быть одна из следующих директив:

- *переходной тип документа*; обладает более мягкими правилами и обеспечивает обратную совместимость

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
```

- *строгий тип документа*; данный тип документа должен полностью соответствовать синтаксису XML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN">
```

- *шаблон DTD для фреймов*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN">
```

Также следует обязательно прописывать теги <html>, <head>, <title>, <body>.

Новый обязательный атрибут в теге <html>

Тег <html> должен быть записан с новым атрибутом xmlns:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Нижний регистр тегов

Все теги, используемые на странице, должны быть записаны в *нижнем регистре* согласно правилам XML.

Правила вложенности тегов обязательны

Несмотря на то, что правила вложенности тегов также являются обязательными и для HTML, браузеры часто игнорируют некритичные ошибки, правильно отображая содержимое страницы. Парсер XML требует строго соблюдения правил вложенности. Пример неправильной вложенности, которая приведет к ошибкам:

```
<b><p>текст</b></p>
```

Данный код должен быть записан следующим образом:

```
<p><b>текст</b></p>
```

Закрывающие теги обязательны

В HTML некоторые теги могли не иметь закрывающего тега, например тег <p>. По правилам XHTML все теги *обязательно* должны иметь закрывающий тег. Теги, которые, в принципе, не имеют закрывающего тега, например тег <br>, должны записываться со слешем в конце, например, <br />. Пробел между именем тега и слешем нужен для обеспечения совместимости со старыми браузерами, которые иначе могут не распознать тег <br/>.

Значения атрибутов в кавычках

Синтаксис XML требует, чтобы все значения атрибутов тегов были записаны в *кавычках*, например:

```
<p align="center">Текст</p>
```

В HTML некоторые атрибуты могут быть записаны в сокращенном формате — одним словом, например атрибут `selected`:

```
<option value="1" selected>Элемент списка</option>
```

В XHTML такие атрибуты необходимо приводить в полном формате

```
<option value="1" selected="selected">Элемент списка</option>
```

Атрибут *name* заменяется атрибутом *id*

Атрибута с именем `name` нет в спецификации XHTML. Вместо него используется атрибут `id`.

Символы разметки < и &

Символы `<` и `&` рассматриваются как символы разметки. Для отображения этих символов в тексте страницы следует воспользоваться их кодами:

- `<` — `<`;
- `&` — `&`.

Более подробную информацию о языке XHTML можно получить по адресам:

- спецификация XTML: **http://www.w3.org/HR/xhtml1/**;
- фундаментальная спецификация XHTML: **http://www.w3.org/HR/xhtm-basic/**;
- модуль XHTML 1.1: **http://www.w3.org/HR/xhtml11/**.

Что такое DHTML?

DHTML расшифровывается как Dynamic HTML (динамический HTML). Это технология, позволяющая создавать динамические HTML-страницы. DHTML нельзя назвать языком, у него нет спецификации и точных правил. В настоящее время, DHTML — это термин, объединяющий три технологии: язык разметки HTML, таблицы стилей CSS и скриптовый язык JavaScript. Связующим ядром для них является объектная модель документа (Document Object Model, DOM). Объектная модель документа предоставляет интерфейс для доступа к элементам HTML, с помощью которого возможно управление элементами и их свойствами. Именно DOM делает документы HTML динамическими. Любое событие, такое как щелчок мыши, перемещение указателя мыши в окне браузера, нажатие клавиши и т. д., является событием, которое может быть обработано инструкциями сценария.

DHTML — активно развивающаяся технология, однако основным препятствием ее распространения является отсутствие стандартов, вследствие чего

возникает различная реализация DHTML в браузерах разных производителей. Наиболее сильное различие — это разные объектные модели, которые используются в линейках браузеров Internet Explorer и Netscape (в том числе все браузеры, основанные на ядре Mozilla). Вследствие этого Web-разработчикам необходимо создавать, как минимум, два варианта кода, обеспечивающего одинаковую функциональность для разных браузеров.

Оптимизация HTML-кода

Рассматривать вопросы оптимизации HTML-кода, как это не парадоксально, следует еще до того, как он будет написан. Непродуманность вопросов верстки может привести к тому, что в итоге качество сайта не будет удовлетворять, а внесение серьезных изменений без переделки всего HTML-кода страницы станет уже невозможным.

Оптимизация ради оптимизации не имеет смысла, это необходимо делать ради конкретных задач: увеличения скорости загрузки сайта, корректного воспроизведения сайта разными браузерами, поддержки различных разрешений экрана и т. п. Все эти мероприятия оптимизации направлены на то, чтобы сделать сайт более доступным для посетителей. Через верстку кода Web-мастер воплощает визуальное представление, которое может быть оформлено либо в виде макета дизайна, либо еще только формируется в его голове. Но в любом случае, перед началом верстки уже сформулирована какая-либо исходная задача, которую нужно реализовать средствами языка HTML. Все вопросы верстки HTML очень индивидуальны, точно так же как индивидуальны решаемые задачи, и индивидуальны предпочтения Web-разработчиков. Поэтому невозможно дать точный и исчерпывающий список рецептов оптимизации. Далее будут рассмотрены лишь общие принципы, соблюдение которых может повысить качество кода, и некоторые распространенные вопросы по верстке и оптимизации HTML-кода.

Чем проще — тем лучше

Следует придерживаться правила: "Чем проще — тем лучше". Web-мастер должен видеть структуру созданного им HTML-кода. Чем проще код, тем легче внести в него изменения, тем более предсказуемым будет реакция браузеров. Если Web-мастер не может разобраться в хитросплетениях тегов — это первый признак того, что код нуждается в оптимизации. Практически всегда для сложного участка кода можно найти более простое и элегантное решение.

Больше пишите руками

Автоматизированные средства разработки могут помочь и существенно облегчить задачу верстки HTML-кода, но их использование также несет в себе

и минусы. Код, сверстанный автоматизированными программами, может содержать ненужные или неоптимальные блоки кода, приводящие к более запутанной структуре и большему объему страницы. При ручном создании кода этого можно избежать. Совет писать код руками не означает, что нужно отказаться от применения автоматизированных программ верстки. При достижении определенной степени профессионализма написание кода вручную будет быстрее, чем использование специальных программ, а сами программы станут для вас многофункциональным справочником.

Табличная верстка против верстки на слоях

Таблицы — очень мощная конструкция HTML, но не стоит ею злоупотреблять. Огромное достоинство таблиц в том, что до сих пор они являются единственным элементом, с помощью которого можно структурировать данные на странице, не опасаясь проблем с отображением. Даже технология CSS (Cascading Style Sheet, каскадная таблица стилей) не смогла заменить табличной верстки. Недостаток табличной верстки проявляется при использовании большого числа вложенных таблиц. Это верное средство запутать код и сделать его нечитаемым. В последнее время широкое распространение получила верстка на слоях (теги `<div>`) с применением технологии CSS. Однако пока это еще не может служить адекватной альтернативой таблицам. Технология CSS позволяет упростить код, но может внести проблемы с отображением сайта разными браузерами и при разных разрешениях, т. к. поддержка браузерами CSS осуществляется еще не в полной мере.

Самый главный недостаток верстки на слоях заключается в ответе на вопрос: а что будет, если пользователь отключит стили? Структура страницы окажется разрушенной. Верхняя часть может сползти вниз, нижняя подняться наверх — сайт станет нерабочим. Немаловажным является тот факт, что стилевые таблицы CSS могут не успеть погрузиться на сайт при использовании посетителем медленного модемного соединения. В этом случае страница будет отражена без стилей в самом неприглядном виде.

Компромиссом, учитывающим сильные и слабые стороны обоих способов верстки, является их сочетание, которое можно сформулировать в следующих советах:

- используйте табличную верстку для построения структуры страницы, поскольку это гарантирует, что структура страницы при любых условиях будет отражена так, как вы задумали;
- используйте верстку на слоях для решения оформительских задач — это упростит код и сделает его прозрачнее.

Тестируйте с отключенными изображениями

Посетители при просмотре сайтов часто отключают загрузку изображений и работают в текстовом режиме. Причиной такого поведения может быть и низкая скорость, и высокая стоимость доступа к Интернету. Web-разработчику, ориентирующемуся на широкую аудиторию, следует учитывать этот вид навигации и тестировать работу своих сайтов при отключенных изображениях. Далее перечислены самые распространенные ошибки, выявляемые при подобном тестировании.

- Отсутствие поясняющих подписей к изображениям в свойстве `alt`. Увидев интересную подпись, посетитель может специально включить просмотр графики, чтобы посмотреть на изображение.
- Отсутствие поясняющих надписей у графических кнопок. Данная ошибка относится к первому пункту, но в случае, когда изображением является кнопка, эта ошибка переходит в разряд фатальных, т. к. это может сделать сайт неработоспособным.
- Отсутствие размеров изображений `width` и `height` в свойствах тега `<img src="image.jpg" width="100px" height="200px">`. Если размер изображения не прописан в свойствах тега, это приведет к тому, что все изображения будут выводиться как заглушки одинакового размера. Это может полностью испортить дизайн страницы. Особенно если при верстке используются изображения-пустышки — прозрачные изображения размером в один пиксел. Представьте, что все они станут размером 28 × 30 пикселов, а именно такая ситуация наблюдается в самом популярном браузере — Internet Explorer.
- Основной цвет рисунка не соответствует цвету фона. Приведем пример. В качестве фона используется темное изображение, поверх которого размещается белый текст. Цвет фона при этом останется по умолчанию белым. Если на такой сайт зайдет посетитель с отключенной графикой, то он ничего не увидит, поскольку белый текст будет написан на белом фоне.

Оптимизация под разные настройки экрана

Разброс экранных разрешений, используемых посетителями в Интернете, достаточно велик. Более половины всех посетителей используют разрешение 1024 × 768 пикселов. Около четверти работают с разрешением 800 × 600 пикселов. Меньшие экранные разрешения встречаются значительно реже, поэтому нижний предел разрешений, для которых следует проводить оптимизацию, можно ограничить разрешением 800 × 600. Оптимизацию следует проводить не только для малых разрешений, но и для больших (более 1024 × 768), т. к. отображение страницы может существенно измениться. На-

пример, шрифт будет слишком мелким, и его невозможно будет прочесть, текстовый параграф растянется в одну строку на всю ширину окна браузера, и его неудобно будет читать.

Также могут возникнуть проблемы с отображением графики на сайте из-за недостаточного количества цветов, указанных в настройках рабочего стола посетителей. Следует проверить отображение сайта при использовании 24- и 16-битного цвета в настройках рабочего стола. Отображение полноцветных изображений в таких режимах работы может существенно измениться. В таком случае следует провести оптимизацию изображений для уменьшения влияния цветовых настроек. Также необходимо проверить качество отображения на жидкокристаллических (LCD) мониторах. В последнее время они получают большое распространение, а качество цветопередачи у них отличается от мониторов с электронно-лучевыми трубками.

Оптимизация под разные браузеры

Одна из целей оптимизации сайта — это корректное отображение при его просмотре посредством разных браузеров. Лучшим способом решения этой задачи является тестирование работы сайта с использованием всех распространенных версий браузеров. В настоящий момент число браузеров разных производителей и их версий настолько велико, что проверить работу во всех версиях даже популярных браузеров не представляется возможным, не говоря уже о качественной оптимизации. Проблемы отображения сайта существуют не только при просмотре сайта в браузерах разных производителей, например Internet Explorer и Opera, но и в разных версиях браузеров одного и того же производителя, например Internet Explorer 5.0 и Internet Explorer 6.0. Если необходимо обеспечить поддержку старых версий, то нужно отказаться от использования возможностей, которые поддерживаются в более новых версиях обозревателей. Есть несколько вариантов решения проблем совместимости браузеров.

1. Не проводить оптимизацию совсем и разрабатывать сайт только под одну, самую популярную версию браузера. Этот подход имеет право на существование, если хорошо известна аудитория сайта. Например, это корпоративный сервер во внутренней сети предприятия.
2. Предупредить посетителей с помощью сообщения о том, что сайт оптимизирован только под определенную версию браузера, и при просмотре его другими браузерами могут возникнуть проблемы с отображением. Например: "Сайт оптимизирован для работы в Internet Explorer 5.5 и выше".
3. Создать несколько версий HTML-кода и выдавать посетителям версию, оптимизированную под тот браузер, который они используют для про-

смотра сайта. Автоматическое определение типа и версии браузера рассмотрено в *приложении 4*.

4. Выбрать группу самых популярных браузеров (4—5 штук) и провести оптимизацию кода только под эти браузеры.

Наиболее приемлемым является последний 4-й способ. Таким образом, можно обеспечить поддержку до 99% посетителей. Конечно, при этом придется лишиться части возможностей, индивидуальных для каждого из браузеров и, вероятно, придется многократно переделывать HTML-код, чтобы добиться примерно одинакового отображения сайта. Но это небольшая плата за полноценную поддержку подавляющего числа посетителей.

С технической точки зрения лучшим способом оптимизации является 3-й способ — создание нескольких версий HTML-кода одной и тоже страницы. Однако этот прием нечасто используется Web-разработчиками. Причина — многократное увеличение трудозатрат на поддержку и внесение изменений в HTML-код. Ведь каждое изменение нужно выполнить в разных файлах и каждую модифицированную страницу протестировать.

Предупреждения посетителей о том, что сайт оптимизирован только под определенный браузер (способ 2), хотя и вполне корректны, но могут создать негативный имидж. Логика здесь может быть такой: раз сайт не смогли оптимизировать под разные браузеры, значит, у разработчиков низкая квалификация, а хозяева сайта (если это сайт фирмы) не смогли найти профессиональных разработчиков для своего ресурса. Если сайт оптимизируется только под один самый популярный браузер, то, возможно, будет лучше совсем не говорить об этом посетителям. Тем более что большинство из них не увидит проблем, т. к. будет просматривать сайт с использованием нужного браузера.

Замечание

Вопросы определения типа браузера с помощью JavaScript рассматриваются в *приложении 4*. Посредством PHP информацию о типе браузера можно получить из элемента суперглобального массива `$_SERVER["HTTP_USER_AGENT"]`.

Самые популярные браузеры

Бесспорное лидерство среди браузеров занимает продукт компании Microsoft — Internet Explorer (MSIE). По разным оценкам им пользуются от 80 до 90% всех посетителей в RuNet. На втором месте по популярности идет браузер Opera — около 4%. На третьем — браузер Mozilla. Ниже приведена статистика использования браузеров, полученная по данным популярного интернет-счетчика Hotlog (**http://gs.hotlog.ru/**):

- MSIE 6.0 — 62,2%;
- MSIE 5.0 — 21%;
- MSIE 5.5 — 7,8%;
- Opera 7 — 3,8%;
- Mozilla — 2,52%;
- Opera 6 — 0,63%;
- MSIE 4 — 0,62%;
- другие — 0,22%;
- Netscape 3 — 0,19%;
- Netscape 7 — 0,17%;
- Netscape 4.7 — 0,16%.

Процент используемых посетителями браузеров зависит от характера аудитории интернет-ресурса.

Проверяйте грамматику и орфографию

Этот совет достаточно банален, однако существует множество сайтов, которые невозможно воспринимать из-за обилия грамматических и орфографических ошибок. Не пренебрегайте проверкой текстов. Если в качестве одной из целей оптимизации рассматривать воздействие на аудиторию, то проверка текстов на сайте должна стоять на первом месте в списке задач оптимизации. Многие люди органически не переносят тексты с ошибками. Такие тексты не только сложно читать — они вредны для раскрутки сайта в поисковых системах. Ведь большинство людей при запросе в поисковиках пишут слова правильно. Для минимальной проверки можно использовать проверку правописания в редакторе MS Word.

Приемы работы с HTML

Рассмотрим часто возникающие вопросы верстки HTML-кода и приведем варианты их решения.

Проблемы относительных ссылок и тег <base>

Проблемы относительных ссылок часто возникают при использовании конструкции `include` в PHP для подключения элементов дизайна, в которых используются относительные ссылки. Таким элементом может быть меню, подключаемое ко всем страницам. Например, файл с меню расположен в корневом каталоге сайта и подключается на страницу index.php с помощью кода:

```
include "menu.php";
```

И тот же самый файл menu.php подключается в файлах из каталога company:

```
include "../menu.php";
```

Для того чтобы из каталога company подключить файл menu.php, необходимо подняться на один уровень вверх. Для этого путь к файлу записан в виде `../menu.php`.

Однако возникнет проблема с относительными ссылками. Прекрасно функционирующее меню в индексном файле не станет работать на страницах из каталога company, т. к. относительные пути для индексного файла и файлов из каталога company будут разными. Одним из решений этой проблемы может стать использование тега <base>. Формат тега:

```
<base href="полный_URL">
```

Тег <base> помещается внутри контейнера <head> и устанавливает полный базовый адрес для текущего документа. При наличии этого тега все относительные ссылки станут автоматически отсчитываться не от местоположения текущего документа, а от значения базового адреса из тега <base>. Например, если на странице прописать корневой адрес сайта <base href="http://www.server.ru/">, то все относительные ссылки на этой странице будут отсчитываться от корневого каталога сайта независимо от того, где эта страница находится. Использование тега <base> позволяет перемещать страницы из каталога в каталог, не опасаясь, что будут нарушены относительные пути. Относительные пути к изображениям в теге <img> также будут отсчитываться от значения базового адреса в теге <base>.

Резиновый табличный дизайн

Резиновым дизайном называется страница, растягиваемая на всю область окна браузера независимо от разрешения экрана. Для создания резинового дизайна следует использовать одну главную таблицу, в которую будет помещен весь оставшийся HTML-код. Страница обычно делится на 3 логические части: верхняя часть, основной информационный блок и нижняя часть страницы. Подобную структуру можно сверстать из таблицы с тремя строками и одним столбцом. Для растяжения таблицы по ширине и высоте на все окно браузера необходимо у тега <table> определить параметры width=100% и height=100% (листинг П1.1).

Листинг П1.1. Резиновый дизайн

```
<html>
  <head>
  </head>
  <body>
    <table border="1" width="100%" height="100%">
      <tr align="center">
        <td height="50px"><h1>Верхняя шапка</h1></td>
      </tr>
      <tr align="center" valign="top">
```

```
      <td>Основной информационный блок</td>
    </tr>
    <tr align="center">
      <td height="50px"><h4>Низ страницы</h4></td>
    </tr>
  </table>
 </body>
</html>
```

Результат выполнения этого кода изображен на рис. П1.1.

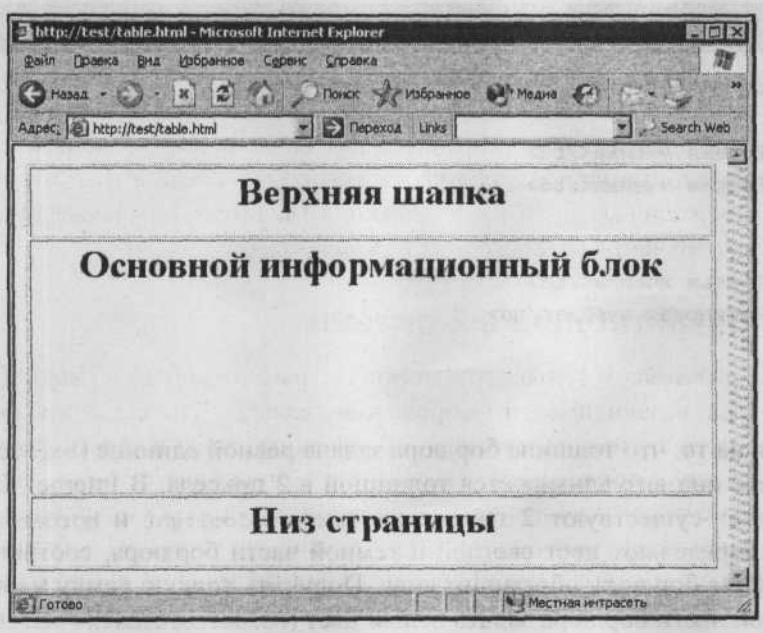


Рис. П1.1. Резиновый дизайн

Замечание

Размещение HTML-кода в одной таблице имеет свои недостатки. Так, например, самый популярный браузер Internet Explorer не станет отображать содержимое страницы до тех пор, пока не будет загружена таблица целиком.

Тонкие рамки таблицы

У таблиц, создаваемых тегами `<table>`, сложно задать тонкие красивые границы. Существует решение для Internet Explorer, но оно, к сожалению, не работает в других браузерах. Пример таблиц с обычными и тонкими рамками приведен на рис. П1.2. и П1.3.

Первая ячейка	Вторая ячейка
Третья ячейка	Четвертая ячейка

Рис. П1.2. Обычные рамки

Первая ячейка	Вторая ячейка
Третья ячейка	Четвертая ячейка

Рис. П1.3. Тонкие рамки

Тонкие рамки для Internet Explorer

Рассмотрим задание тонких границ для браузера Internet Explorer (листинг П1.2).

Листинг П1.2. Задание тонких границ для браузера Internet Explorer

```
<table border="1" cellpadding="0" cellspacing="0">
  <tr>
    <td>Первая ячейка</td>
    <td>Вторая ячейка</td>
  </tr>
  <tr>
    <td>Третья ячейка</td>
    <td>Четвертая ячейка</td>
  </tr>
</table>
```

Несмотря на то, что толщина бордюра задана равной единице (`border="1"`), на самом деле она визуализируется толщиной в 2 пиксела. В Internet Explorer у тега `<table>` существуют 2 параметра: `bordercolorlight` и `bordercolordark`, которые определяют цвет светлой и темной части бордюра, соответственно, для предания бордюру объемного вида. Получить тонкую рамку можно, если для темной части бордюра задать белый цвет (`bordercolordark="white"`), а для светлой, наоборот, — черный (`bordercolorlight="black"`):

```
<table border="1" cellpadding="0" cellspacing="0"
                  bordercolorlight="black" bordercolordark="white">
```

Указание тега `<table>` с такими параметрами отобразит таблицу с тонкими рамками, подобную изображенной на рис. П1.3. Но при отображении в других браузерах рамки останутся объемными (см. рис. П1.2).

Тонкие рамки: универсальное решение

Рассмотрим универсальное решение, которое позволит создавать таблицы с тонкими рамками и работает во всех основных браузерах. Для этого применим таблицы стилей CSS (листинг П1.3).

Листинг П1.3. Универсальное решение для создания тонких рамок

```
<html>
  <head>
    <style>
      /* стиль контейнера (фон) */
      .fon{background-color: #373737;}
      /* фоновый цвет строк таблицы */
      .mytable tr{background-color: #F5F5F5;}
      /* внутренний отступ для ячеек таблицы */
      .mytable td {padding: 5px}
    </style>
  </head>
  <body>
    <div class="fon">
      <table class=mytable border=0 cellspacing="1" >
        <tr>
          <td>Первая ячейка</td>
          <td>Вторая ячейка</td>
        </tr>
        <tr>
          <td>Третья ячейка</td>
          <td>Четвертая ячейка</td>
        </tr>
      </table>
    </div>
  </body>
</html>
```

В результате визуализации данного кода на экран будет выведена таблица, изображенная на рис. П1.4.

Таблица помещается внутрь тега `<div class="fon">`, который играет роль контейнера таблицы. С помощью строки `class="fon"` данному тегу `<div>` присваивается класс стилей с именем `fon`. Таблица стилей `fon` определяет темный фоновый цвет для тега `<div>`. Если больше никакие стили не определять, то таблица также будет темной, поскольку сквозь ячейки будет виден темный фон тега `<div>`. Для того чтобы это исправить, необходимо определить фоновый цвет строк таблицы. Поэтому назначим таблице класс стилей `<table class=mytable>`. Теперь с помощью этого класса стилей можно назначить светлый фоновый цвет строкам таблицы. Это можно сделать при помощи следующей строки:

```
.mytable tr{background-color: #F5F5F5;}
```

Таким образом, все строки таблицы имеют светлый фон, а контейнер (теперь не видный сквозь таблицу) — темный фон. Для того чтобы цвет контейнера отображался в виде тонких рамок, нужно установить параметр cellspacing="1" для тега <table>. Установка этого параметра задаст расстояние между ячейками таблицы в один пиксел, через которое будет виден темный фон тега . Теперь таблица обрамлена тонкими темными рамками. Для завершения создания рамок определим внутренний отступ у ячеек таблицы с помощью стиля .mytable td {padding: 5px}. Внешний вид созданной таким образом таблицы изображен на рис. П1.4.

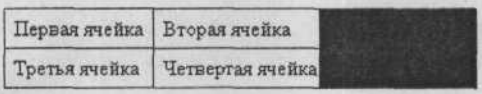


Рис. П1.4. Созданная в листинге П1.3 таблица

Избавиться от черного прямоугольника справа от таблицы можно либо указав ширину тега меньше, чем ширина таблицы, например, , либо растянув таблицу на всю ширину страницы <table class=mytable width=100%>.

Если жесткое указание размеров нежелательно, то в качестве контейнера с фоном следует применить таблицу вместо тега . Модифицированный код приведен в листинге П1.4. Внесенные изменения выделены жирным шрифтом.

Листинг П1.4. Модифицированное универсальное решение для создания тонких рамок

```
<table class="fon" cellspacing="0" cellpadding="0"><tr><td>
  <table class=mytable border=0 cellspacing="1" >
    <tr>
      <td>Первая ячейка</td>
      <td>Вторая ячейка</td>
    </tr>
    <tr>
      <td>Третья ячейка</td>
      <td>Четвертая ячейка</td>
    </tr>
  </table>
</td></tr></table>
```

Тонкие рамки: использование стилей CSS

Для создания тонких рамок, подобных тем, что изображены на рис. П1.5, необходимо с помощью стилей CSS задать бордюры у ячеек таблицы. Контейнер, определяющий фон, в этом случае уже не нужен (листинг П1.5).

Листинг П1.5. Создание тонких рамок с помощью стилей CSS

```
<html>
  <head>
    <style>
      .table td {padding: 5px;
                 border-style: solid; /* указание рисовать бордюр */
                 border-width: 1px;   /* толщина бордюра */
                 border-color: black  /* цвет бордюра */
                }
    </style>
  </head>
  <body>
    <table class=table border="0" cellspacing="1" >
      <tr>
        <td>Первая ячейка</td>
        <td>Вторая ячейка</td>
      </tr>
      <tr>
        <td>Третья ячейка</td>
        <td>Четвертая ячейка</td>
      </tr>
    </table>
  </body>
</html>
```

Первая ячейка	Вторая ячейка
Третья ячейка	Четвертая ячейка

Рис. П1.5. Тонкие рамки: стиль второй

Применение "распорок"

При верстке страниц очень часто используется сложное форматирование таблиц с объединением столбцов (`colspan`) и строк (`rowspan`). В этом случае браузеры часто неправильно рассчитывают ширину ячеек таблиц, если в них использованы подобные группировки ячеек.

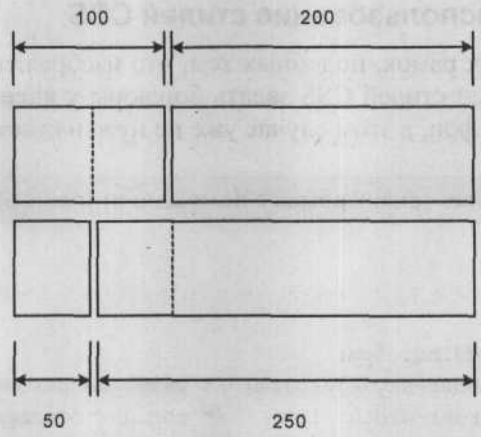


Рис. П1.6. Схема таблицы

Рассмотрим пример верстки таблицы, схема которой изображена на рис. П1.6.

Такую таблицу можно сверстать с помощью таблиц из двух строк и трех колонок. В первой строке первая и вторая ячейки будут объединены в одну установкой параметра `colspan=2`, во второй строке также будут объединены вторая и третья ячейки. В листинге П1.6 приведен HTML-код, реализующий данную схему.

Листинг П1.6. Верстка таблицы по схеме из рис. П1.6

```
<table class="table" border="0" cellspacing="1" width="300px">
  <tr>
    <td colspan="2" width="100px">1</td>
    <td width="200px">2</td>
  </tr>
  <tr>
    <td width="50px">3</td>
    <td colspan="2" width="250px">4</td>
  </tr>
</table>
```

Несмотря на то, что HTML-код написан корректно, таблица будет отображена неправильно (рис. П1.7), т. к. ширина ячеек не соответствует задуманной (см. рис. П1.6).

Таким образом, установка параметров ширины не может сформировать задуманную таблицу. Для решения подобных проблем можно применить распорки. *Распорками* называются вставки прозрачных изображений размером 1 × 1

пиксел и назначение им требуемой ширины с помощью параметров тега <img width="x" height="x">. Для формирования распорки следует создать прозрачное изображение размером 1 × 1 пиксел и сохранить в формате GIF. Затем вставить его в третью и четвертую ячейки, установив ширину изображения 50 и 250 пикселов соответственно (листинг П1.7).

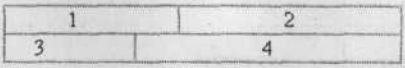


Рис. П1.7. Неправильное отображение таблицы

Листинг П1.7. Использование "распорок" для указания размеров ячеек

```
<table border="1" cellspacing="0" width="300px" cellpadding="0">
  <tr align="center">
    <td width="100px" colspan="2">1</td>
    <td>2</td>
  </tr>
  <tr align="center">
    <td width="50px"><img src="../site/images/pic.gif" border="0"
                          width="50px" height="1px"><br>3</td>
    <td colspan="2"><img src="../site/images/pic.gif" border="0"
                          width="250px" height="6"><br>4</td>
  </tr>
</table>
```

Полученная таблица соответствует задуманной по ширине ячеек и изображена на рис. П1.8.

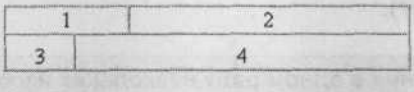


Рис. П1.8. Таблица, созданная с помощью "распорок"

Определяйте цвет фона

По умолчанию фон страницы зависит от системных настроек — цветовой схемы рабочего стола. Поэтому, разрабатывая сайты, необходимо быть готовым к тому, что у посетителей окажется иная цветовая схема рабочего стола, чем та, что была установлена при разработке сайта. Если цвет фона для тега <body> не определяется явно, то у посетителя с иной цветовой схемой фон страницы может быть другим и дизайн страницы может сильно пострадать.

Для определения цвета фона удобно использовать таблицы стилей (листинг П1.8).

Листинг П1.8. Определение цвета фона

```
<html>
  <head>
    <style>
      body {background-color: #FFFFFF}
    </style>
  </head>
  ...
</html>
```

Здесь приведен пример таблицы стилей, помещенной внутрь шапки <head>.

Совет

Не следует использовать белый цвет текста для печати, т. к. фоновые изображения и цвет фона не печатаются.

Точное позиционирование изображений

Одна из частых проблем при решении задач точного позиционирования изображений — наличие пробелов и отступов, мешающих достижению поставленной цели. Эта проблема возникает из-за того, что теги <img> чутко реагируют на наличие пробелов и символов перевода строк вокруг них. Для того чтобы убрать нежелательные отступы вокруг изображений, следует удалить все пробелы и переводы строк вокруг тегов <img>.

Рассмотрим задачу размещения в одном ряду нескольких изображений без пробелов (листинг П1.9).

Листинг П1.9. Размещение в одном ряду нескольких изображений

```
<img src="images/bank_1.jpg" border="0" width="100" height="75" alt="">
<img src="images/bank_4.jpg" border="0" width="100" height="75" alt="">
<img src="images/bank_2.jpg" border="0" width="100" height="75" alt="">
```

Каждый тег <img> начинается с новой строки. Отображение этого кода в браузере показано на рис. П1.9.

Браузер автоматически добавил пробелы между изображениями, хотя это не было задано в HTML-коде. Причина такого поведения — переводы строк, после каждого тега <img>. Для того чтобы все три изображения были отобра-

жены слитно и без пробелов, HTML-код в листинге должен быть записан в одну строку без пробелов между тегами <img>. Модифицированный вариант кода выведет на экран изображение, показанное на рис. П1.10.

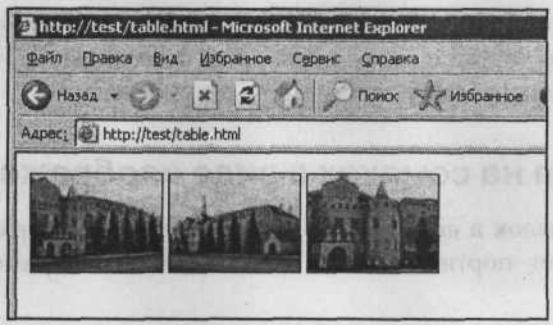


Рис. П1.9. Пробелы между изображениями

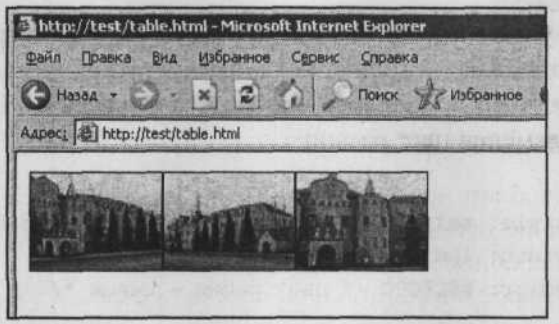


Рис. П1.10. Изображения без пробелов между ними

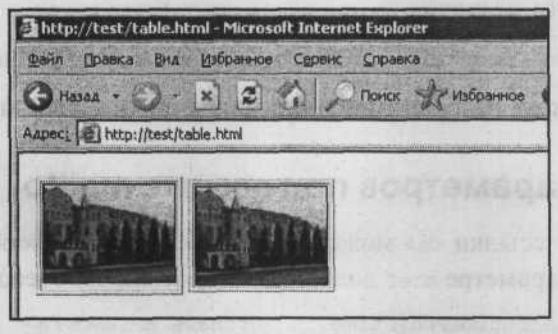


Рис. П1.11. Отступ между изображениями в таблице

Сходная проблема точного позиционирования возникает при вставке изображения в ячейку таблицы. Если между завершающей скобкой тега <img> и за-

вершающим тегом ячейки </td> будет расположен пробел или перевод строки, то в браузере изображение будет отображено с отступом от нижнего края ячейки. Пример такого дефекта показан на рис. П1.11. В правой таблице виден результат выполнения исправленного HTML-кода (без пробелов).

Данный дефект особенно заметен, когда одно большое изображение разрезается на несколько частей и затем собирается с помощью HTML-кода. В этом случае пробелы между изображениями недопустимы.

Синяя рамка на ссылках в виде изображений

При создании ссылок в виде изображений вокруг них появляется синяя рамка, которая может портить внешний вид страницы. Пример такой ссылки приведен ниже.

```
<a href="mailto:softtime@softtime.ru">
            <img src="image.gif" border="1" width="50" height="50"></a>
```

Для того чтобы убрать рамку, следует установить параметр border=0 в теге <img>. Однако возможна ситуация, когда рамка необходима, но нужно изменить ее цвет. Для этого можно воспользоваться стилями CSS. Создадим класс myimg, где будет изменен цвет рамки:

```
.myimg {
        border-style: solid;  /* тип рамки - непрерывная линия */
        border-width: 1px;    /* толщина рамки */
        border-color: #8C8C8C /* цвет рамки - серый */
       }
```

Далее следует применить этот стиль к тегу <img> с помощью строки class="myimg":

```
<img class="myimg" src="image.gif" border="1" width="50" height="50">
```

Теперь рамка вокруг изображения имеет спокойный серый цвет.

Передача параметров при ссылке mailto

Стандартный тег ссылки <a> может использоваться для формирования письма. Для этого в параметре href должно быть указано ключевое слово mailto.

```
<a href="mailto:softtime@softtime.ru">Послать письмо</a>
```

При щелчке по такой ссылке откроется окно почтового клиента, где в качестве адресата будет стоять адрес **softtime@softtime.ru**. Но кроме e-mail через параметры ссылки можно также указать тему и текст письма. Для того чтобы

это сделать, нужно перевести передаваемые параметры в URL-безопасный вид, т. к. через параметры ссылки беспрепятственно можно передавать только алфавитно-цифровые символы. Все символы, не попадающие под это определение, например символы пробела, перевода строки, кавычек, должны быть закодированы.

```
<a href="mailto:softtime@softtime.ru?
                subject=Письмо%20с%20сайта&
                body=первая%20строка%0D%0A
                вторая%20строка">
                Послать письмо</a>
```

При щелчке по этой ссылке откроется окно почтового клиента, где будет указан адресат, тема письма и две строки текста в теле письма (рис. П1.12).

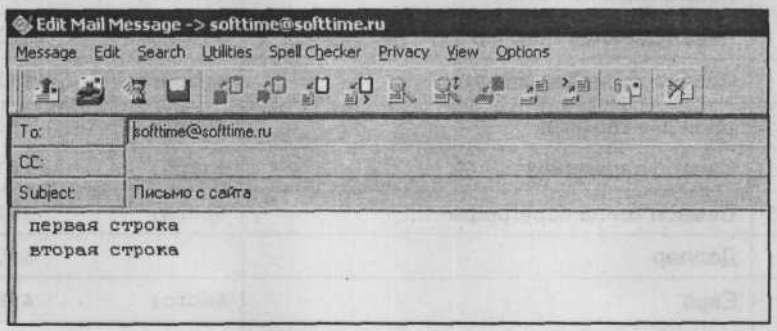


Рис. П1.12. Результат передачи параметров письма в ссылке

Ниже приведены наиболее часто используемые неалфавитно-цифровые символы и их кодировка:

- пробел — `%20`;
- перевод строки — `%0D0A`;
- двойная кавычка — `%22`;
- одинарная кавычка — `%27`.

Коды спецсимволов

С помощью специальных кодов в текст HTML-страницы легко вставлять множество спецсимволов и знаков, которые не могут быть набраны обычным текстом, например знаки авторского права © и зарегистрированной торговой марки ®. В табл. П1.1 приведен список наиболее распространенных спецсимволов, их цифровые коды и именованные обозначения.

Таблица П1.1. *Спецсимволы*

Символ	Описание	Именованное обозначение	Цифровой код
®	Зарегистрированная торговая марка	®	®
©	Знак авторского права (copyright)	©	©
™	Торговая марка	™	™
@	Символ e-mail		@
	Неразрывный пробел		
`	Обратная кавычка	´	´
»	Правые кавычки-лапки	»	»
«	Левые кавычки-лапки	«	«
"	Левая двойная кавычка	“	“
"	Правая двойная кавычка	”	”
"	Двойные кавычки	"	"
§	Начало параграфа	§	§
¶	Символ конца параграфа	¶	¶
$	Доллар		$
€	Евро	€	€
¢	Цент	¢	¢
¥	Йена	¥	¥
£	Фунт	£	£
<	Символ "меньше"	<	<
>	Символ "больше"	>	>
&	Амперсанд	&	&
#	Решетка (диез)		#
°	Градус	°	°
±	Плюс/минус	±	±
²	Математический символ "в квадрате"	²	²
³	Математический символ "в кубе"	³	³
¼	1/4	¼	¼
½	1/2	½	½
¾	3/4	¾	¾

Таблица П1.1 (окончание)

Символ	Описание	Именованное обозначение	Цифровой код
×	Умножение	×	×
÷	Деление	÷	÷
≈	Приближенно равно	≈	≈
≠	Не равно	≠	≠
≤	Меньше, либо равно	≤	≤
≥	Больше, либо равно	≥	≥
·	Жирная точка	·	·
µ	Символ микро	µ	µ
∞	Символ бесконечности	∞	∞

Метатеги

Метатеги — это специальные теги в HTML-страницах, содержащие служебную информацию. Метатеги и их содержимое не отображаются на страницах. Они предназначены для чтения роботами поисковых систем, браузерами посетителей просматривающих страницы, кэширующими proxy-серверами и другими программами, работающими с HTML-страницами. Существует большое число различных метатегов. За время развития среды Интернет многие метатеги потеряли актуальность и практически не используются. Далее будут рассмотрены основные группы метатегов, которые могут пригодиться при разработке сайтов.

Метатеги записываются в блоке <head> HTML-страницы:

```
<html>
  <head>
    <META NAME="author" CONTENT="IT-студия SoftTime">
  </head>
  <body>
    <!-- текст страницы -->
  </body>
</html>
```

Метатеги могут быть записаны в одном из двух форматов:

```
<META NAME="имя" CONTENT="значение метатега">
```

Это описательные метатеги. В них приводятся свойства страниц, такие как авторские права, краткое содержание страницы, указания для роботов поисковых систем. Вторая форма метатега имеет следующий вид:

```
<META HTTP-EQUIV="имя" CONTENT="значение метатега">
```

Информация из метатегов, содержащих строку HTTP-EQUIV, используется для передачи соответствующих HTTP-заголовков. С помощью таких метатегов можно управлять кэшированием документов, указывать кодировку документа, посылать запросы на переадресацию.

Взаимодействие с поисковыми системами

Рассмотрим метатеги, предназначенные для взаимодействия с поисковыми системами. Следует отметить, что это самая противоречивая группа метатегов. Роботы поисковых систем все меньше обращают внимание на информацию в этих тегах и все больше ориентируются на текстовое содержание самой страницы, которое видит обычный посетитель через браузер. Это произошло вследствие того, что подобные метатеги часто содержали информацию, не соответствующую действительности, для того чтобы обмануть поисковые системы и любыми способами привлечь большое количество посетителей на сайт.

Description

Метатег с именем Description содержит краткую аннотацию документа:

```
<META NAME="Description" CONTENT="Описание вашей страницы.">
```

Каждая поисковая система определяет свои рекомендации относительно длины описания. Обычно рекомендуется использовать не более 255 символов.

Keywords

Метатег Keywords определяет список ключевых слов, перечисляемых через запятую или пробел, которые характеризуют содержание страницы. Например, для сайта, посвященного программированию, набор ключевых слов может быть следующим:

```
<META NAME="Keywords" CONTENT="php форум программирования
                               скрипты web php-скрипты download">
```

Revisit

Метатеги с именем Revisit дают рекомендацию роботам поисковых систем, через какое количество дней следует заново проиндексировать страницу:

```
<META NAME="Revisit" CONTENT="N days">
<META NAME="Revisit-after" CONTENT="N days">
```

Здесь N — число дней. Следует отметить, что этот метатег практически потерял свою актуальность и не учитывается роботами известных поисковых систем.

Robots

Метатеги с именем Robots позволяют управлять индексацией сайта. В частности, с их помощью можно запретить индексирование определенных страниц. Если на сайте существует много страниц, которые не содержат полезной и интересной информации, то можно запретить их индексацию поисковыми роботами. В этом случае, роботы станут уделять больше внимания действительно нужным информационным страницам. Метатеги имеют следующий формат:

```
<META NAME="Robots" CONTENT="параметры">
```

В качестве параметров могут выступать значения:

- index — индексировать страницу;
- follow — следовать по ссылкам с данной страницы;
- all — идентично двум совместно используемым опциям index и follow;
- noindex — запретить индексацию страницы;
- nofollow — запретить переход по ссылкам с данной страницы;
- none — идентично применению параметров noindex и nofollow.

Примеры:

```
<META NAME="robots" CONTENT="noindex">
<META NAME="robots" CONTENT="index, nofollow">
<META NAME="robots" CONTENT="nofollow">
<META NAME="Robots" CONTENT="index, follow">
```

Замечание

Приведенные метатеги носят рекомендательный характер. Они принимаются во внимание корректно работающими роботами, соблюдающими данное соглашение об индексации. Для гарантированного запрета индексации страниц, например, страниц системы администрирования, следует применять другие средства, скажем, закрывать доступ с помощью соответствующей настройки Web-сервера *(см. гл. 2)*.

Описание документа

Content-Type

Данный метатег указывает MIME-тип (медиатип) документа и кодовую страницу. Тег записывается в формате:

```
<META HTTP-EQUIV="Content-Type"
      content="MIME-тип; charset=кодовая_страница">
```

Пример:

```
<META HTTP-EQUIV="Content-Type"
      content="text/html; charset=windows-1251">
```

Здесь указано, что страница имеет формат HTML в кодировке Windows-1251.

Замечание

Этот метатег следует применять с осторожностью. В настоящее время все современные браузеры могут сами определять тип и кодировку документа. Если указанная в метатеге кодировка документа не совпадет с фактической, то документ будет отображен в нечитаемом виде.

Content-Language

Заголовок `Content-Language` позволяет указать языковую аудиторию, для которой существует страница. Если страница предназначена для многоязыковой аудитории, то в теле заголовка можно перечислить несколько языков через запятую. Пример:

```
<META HTTP-EQUIV="Content-Language" CONTENT="ru">
<META HTTP-EQUIV="Content-Language" CONTENT="en,ru">
```

Могут использоваться только фиксированные коды языков, например:

- `ru` — русский;
- `ua` — украинский;
- `en` — английский;
- `fr` – французский;
- `de` — немецкий;
- `it` — итальянский;
- `us` — американский вариант английского.

Document-state

Метатег `Document-state` позволяет указывать формат документа: статический или динамический.

```
<META NAME="Document-state" CONTENT="Dynamic">
<META NAME="Document-state" CONTENT="Static">
```

Данный метатег предназначен для работы с роботами поисковых систем. По умолчанию подразумевается значение `Dynamic`. Оно означает, что документ является динамическим и его нужно периодически переиндексировать. Значение `Static` предназначено для статических, редко меняющихся документов, которые не нуждаются в переиндексации.

Resource-type

Метатег `Resource-type` указывает тип документа. Для нормальной индексации страницы должно быть установлено значение `document`.

```
<META NAME="Resource-type" CONTENT="Document">
```

Также этот метатег может принимать значения: `Build`, `Classification`, `Creation`, `Formatter`, `Host`, `Operator`, `Rating`, `Site-languages`, `Subject`, `Template`, `Version`.

Generator

Имя программы, в которой была создана страница, например:

```
<META NAME="Generator" CONTENT="Название программы">
```

Author

Метатег, содержащий указание на авторство документа. Значение тега записывается в произвольном формате.

```
<META NAME="author" content="Автор документа">
```

Copyrigh

Указание авторских прав на документ. Значение тега записывается в произвольном формате.

```
<META NAME="Copyright" content="Авторские права">
```

Кэширование

При кэшировании страница сохраняется во временном хранилище и при запросе этой страницы выдается не ее реальная версия из сети, а сохраненная локальная копия. Кэширование может осуществляться в браузере посетителя и на proxy-серверах, через которые проходит трафик. Так как трафик — это деньги, то администраторы proxy-серверов стремятся его уменьшить всеми возможными способами, в том числе кэшируя все посещенные страницы. Для управления кэшированием служат специальные метатеги.

Замечание

Иногда неправильные настройки кэширования на proxy-серверах приводят к невозможности получения свежей информации. В таком случае никакие метатеги не работают и помогает только принудительное, ручное обновление каждой страницы с помощью кнопки **Обновить**.

Pragma

Метатег посылает заголовок с именем `Pragma`, который служит для указания браузерам и proxy-серверам, работающим по протоколу HTTP 1.0, не кэшировать страницу. Заголовок `Pragma` может содержать только одно значение `no-cache`.

```
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
```

Cache-Control

Заголовок с именем `Cache-Control` появился в протоколе HTTP 1.1 и служит для управления кэшированием.

```
<META HTTP-EQUIV="Cache-Control" CONTENT="значение">
```

Директивы этого заголовка должны выполняться по всей цепочке "запрос — ответ" и переписывать директивы кэширования, установленные по умолчанию. Этот заголовок может содержать следующие значения:

- `public` — разрешение кэширования во всех видах кэшей, даже если в обычных условиях ответ не кэшируется;
- `private` — весь ответ, либо его часть может кэшироваться только одним авторизованным пользователем. Для всех других кэширование запрещается;
- `no-cache` — запрет кэширования, где бы то ни было;
- `no-store` — разрешается только временное кэширование. Сохранение данных в долговременную память запрещается. Однако эта директива не дает никаких гарантий по соблюдению конфиденциальности информации. Пользователь может вручную сохранить страницу через меню браузера;
- `no-transform` — запрещает трансформацию передаваемых данных, которая иногда происходит на промежуточных proxy-серверах, например, для уменьшения трафика;
- `must-revalidate` — данная директива необходима для поддержания надежной работы определенных функций протокола. При наличии данной директивы всегда должно следовать обращение к исходному серверу для сверки данных. Если обращение к исходному серверу не удалось, то должен быть выдан соответствующий ответ. Выдача информации без проверки ее актуальности запрещена;

- proxy-revalidate — данная директива имеет то же значение, что и директива must-revalidate, но влияет только на proxy-серверы и не распространяет свое действие на кэш браузеров;
- max-age — директива управления временем жизни данных в кэше. Данная директива имеет более высокий приоритет, чем заголовок Expires, также управляющий временем актуальности кэшируемых данных.

Примеры:

```
<META HTTP-EQUIV="Cache-Control" CONTENT="public">
<META HTTP-EQUIV="Cache-Control" CONTENT="must-revalidate">
<META HTTP-EQUIV="Cache-Control" CONTENT="max-age=3600">
```

Expires

Заголовок Expires содержит дату, начиная с которой данные в кэше следует считать устаревшими. Устаревшие данные должны быть запрошены из их адреса в сети, а не из кэша. В качестве значения должна быть указана абсолютная дата. Пример:

```
<META HTTP-EQUIV="Expires" CONTENT="Wed, 2 Mar 2005 00:00:05 GMT">
```

При ошибочном формате даты, в том числе когда значение равно 0, данная дата должна относится к прошлому. Заголовок Expires не будет иметь значение при наличии заголовка Cache-Control с директивой max-age.

Переадресация

Refresh

Заголовок refresh определяет время в секундах, по истечении которого браузер производит переадресацию страницы по указанному адресу. Пример:

```
<META HTTP-EQUIV="Refresh" CONTENT="5; URL=newpage.html">
```

Здесь происходит переадресация текущей страницы на страницу newpage.html через 5 секунд после получения этого заголовка. Если не указать параметр URL, то страница будет переадресовывать сама на себя.

Location

Заголовок Location определяет адрес документа в Интернете и может использоваться для переадресации. Пример:

```
<META HTTP-EQUIV="Location" CONTENT="url=http.www.newsite.ru">
```

ПРИЛОЖЕНИЕ 2

CSS

Аббревиатура CSS расшифровывается как *Cascading Style Sheets*, что в переводе означает "каскадные таблицы стилей". Идея таблиц стилей состоит в разделении логического представления HTML-кода и его оформления (дизайна). Несмотря на то, что CSS не решает полностью эту задачу, тем не менее использование CSS несет ряд преимуществ, которыми нельзя пренебрегать:

- структуризация и формализация возможностей оформления и дизайна за счет применения правил CSS. CSS предоставляет ряд стандартных правил, благодаря чему процесс верстки дизайна упрощается;
- вынос оформления в отдельные файлы (файлы стилей), в результате чего HTML-код страницы становится более легким для понимания, а работа с оформлением более простой;
- возможность использования единой схемы дизайна на всех страницах сайта и быстрого внесения изменений в дизайн. Так как стили можно хранить в файле, едином для всего сайта, то при внесении изменений в этот файл они автоматически применяются ко всем страницам сайта.

Стилем, в данном случае, называется совокупность свойств элемента, характеризующая его внешний вид (цвет, размер, рамка, фоновое изображение и т. п.).

Каскадными таблицы называются потому, что к одному документу можно применять несколько таблиц стилей (каскады стилей). В этом случае одному элементу могут предназначаться сразу несколько стилей, которые способны конфликтовать друг с другом. Какой именно стиль выберет браузер для отображения страниц, определяется правилами каскадирования.

- Самый низший приоритет имеют стили, заданные по умолчанию. Эти стили определяются браузером. Назначаемые стили всегда имеют более высокий приоритет.

- Если элемент поддерживает свойства стилей родительского элемента, то для его оформления используются стили родительского элемента.

Замечание

Родительским элементом для тега является тот элемент, внутри которого он находится. Например, `<p>CSS – <em>Каскадные Таблицы Стилей</em></p>`. Для тега `<em>` родительским элементом является тег `<p>`.

- Если стили, применяемые к элементу, конфликтуют между собой, то применяется стиль, содержащий более точное описание элемента, к которому этот стиль будет применен. Например, в ситуации, когда конфликтуют стили, назначенные всем тегам параграфа `<p>`, и стили конкретного параграфа, то предпочтение будет отдано второму, т. к. его применение более специфично, и он применяется к конкретному параграфу.

Спецификации стилей

CSS является развивающимся стандартом. В связи с этим он постоянно дополняется новыми спецификациями, которые расширяют возможности CSS с учетом возможностей браузеров новых поколений.

В настоящий момент существует три спецификации CSS: CSS Level 1, CSS Level 2 и CSS Level 3.

Спецификация CSS Level 1 определяет следующие правила:

- управление отображением текста: гарнитура шрифта, размер, начертание, межбуквенный интервал;
- цвет и фон;
- управление полями и границами объектов;
- форматирование списков.

Спецификация CSS Level 2 определяет следующие возможности:

- подготовка документа для печати;
- поддержка речевых браузеров;
- загружаемые шрифты;
- определение точного позиционирования элементов (расширение CSSP — CSS-Position);
- форматирование таблиц;
- поддержка CSS для XML;
- настройка интерфейса (курсор мыши);
- ограниченные модели поведения (наведение указателя мыши на ссылку).

Спецификация CSS Level 3 определяет следующие возможности:

- поддержка вертикального размещения текста;
- возможность размещения текста в колонках;
- улучшенная поддержка моделей поведения и стилей;
- интеграция с другими технологиями для работы с цветом, графикой и шрифтами.

С выходом новых версий браузеры все более полно осуществляют поддержку данных спецификаций. При этом, как правило, браузеры дополнительно реализуют собственную модель стилей, специфичную для конкретного браузера.

Применение стилей к документу HTML

Существует несколько способов применения стилей CSS к HTML-странице:

- внедренные стили (свойство `style` у элемента);
- тег `<style>`;
- внешний css-файл с описаниями стилей;
- импорт css-файла.

Рассмотрим все эти способы.

Внедренные стили: свойство *style*

У каждого элемента можно прописать свой стиль с помощью свойства `style`:

```
<div style="border-style: solid"></div>
```

Этот способ определения стилей имеет наивысший приоритет, и при конфликте стилей предпочтение будет отдано стилю, определенному в свойствах конкретного элемента. Одновременно данный способ является наименее эффективным. При таком подходе каждому элементу стили нужно назначать индивидуально. При этом теряется большинство преимуществ CSS. Данный способ следует применять только в том случае, если используемый стиль является уникальным и не может быть применен больше ни к какому элементу.

Тег *<style>*

Тег `<style>` позволяет описывать стили непосредственно внутри HTML-документа. Синтаксис:

```
<style type="text/css">
  <!--
```

```
  Правила CSS
  -->
</style>
```

Символы комментариев (<!-- -->) необходимы для того, чтобы старые версии браузеров, не поддерживающие тег <style>, не выводили содержимое таблицы стилей непосредственно на страницу. Данный способ можно считать эффективным для применения только к одной HTML-странице. Чтобы применить одни и те же стили к нескольким страницам, следует копировать таблицы стилей в каждую HTML-страницу и следить, чтобы скопированные стили были идентичны. То есть, при внесении изменений в стиль, модификации должны подвергнуться все страницы одновременно, т. к. описания стилей на каждой странице независимы.

Внешний css-файл

Внешний css-файл удобен, когда необходимо создание единого стилевого оформления для нескольких страниц. В этом случае таблицы стилей выносятся во внешние css-файлы и подключаются на каждую страницу внутри блока head при помощи конструкции:

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Таким образом, для внесения изменений в оформление на всех страницах сайта необходимо отредактировать только один файл стилей styles.css. Изменения в оформлении страниц будут произведены автоматически. Вынос стилевого оформления во внешний файл уменьшает размер страниц, что положительно сказывается на скорости загрузки страниц и их индексации поисковыми системами.

Внешний файл стилей представляет собой текстовый файл с расширением css, в котором расположены таблицы стилей. Следует отметить, что внешний файл стилей не обязательно должен находиться на том же сервере, что и сами страницы. Для подключения внешнего css-файла с другого сервера следует написать его полный URL в параметре href.

Замечание

У внешнего файла стилей есть недостаток, проявляющийся при активной разработке сайта. Очень часто файл стилей кэшируется proxy-серверами, и вместо обновленных таблиц стилей выдается их устаревшая версия из кэша. В особо трудных случаях решить эту проблему можно переименованием внешнего файла со стилями и соответствующего изменения имени файла в теге <link rel="stylesheet" type="text/css" href="styles.css">.

Импорт css-файла

Инструкции по импортированию css-файлов выполняются в современных браузерах, поддерживающих спецификацию CSS2.

```
<style type="text/css">
  <!--
  @import url(styles.css)
  -->
</style>
```

Импортируемый файл должен иметь расширение css. Импортирование css-файлов можно использовать в Internet Explorer, начиная с 4 версии, и Netscape Navigator 6.

Все перечисленные способы применений стилей CSS можно комбинировать и применять одновременно на одной странице.

Спецификация требует точного указания языка для таблиц стилей `type="text/css"`. Документы, в которых отсутствует определение языка, считаются некорректными, хотя большинство браузеров прощают Web-мастерам такую ошибку и обрабатывают документы без указания языка стилей правильно.

Синтаксис стилей CSS

Таблицы стилей содержат собой несложные правила форматирования со следующим синтаксисом:

Селектор {*свойство*: *значение*; *свойство*: *значение*...}

Селектор представляет собой элемент, к которому будет применяться описываемое правило. Пример — в листинге П2.1.

Листинг П2.1. Пример синтаксиса стилей

```
/* стилевое правило для всех тегов <h1> */
h1 {font-size: 18px; color: #00FF00; font-family: "Arial Cyr"}
/* правило для элемента с присвоенным ему классом стилей myclass */
.myclass {border-style: solid; border-width: 1px}
/* правило для элемента с идентификатором titl */
#titl {font-size: 14px; text-style: italic; color: #FFFFFE}
```

В качестве селектора может быть имя тега, имя класса (`class`), идентификатор элемента (`id`), а также их комбинации, которые будут рассмотрены далее.

В листинге П2.1 определены стили для тега <h1>, класса стилей myclass и элемента с идентификатором titl.

Правила, применяемые к селектору, перечисляются в фигурных скобках через точку с запятой. Если значения свойств содержат пробелы, то их нужно обрамлять кавычками. В листинге П2.1 так определяется шрифт Arial Cyr для тега <h1>.

При применении стилей внутри тегов с помощью свойства style синтаксис меняется: отсутствуют селекторы и фигурные скобки, а сами правила помещаются в кавычки.

```
<p style="font-size: 14px; color: #FF0000">Красный текст</p>
```

Комментарии в стилевых таблицах заключаются в следующие символы:

```
/* комментарий */
```

Пример использования комментариев приведен в листинге П2.2.

Базовые селекторы

Базовым селектором является универсальный селектор, либо селекторы типа. *Универсальный селектор* отображается звездочкой * и применяется ко всем без исключения элементам. Пример:

```
* {font-size: 12px; font-family: Arial, sans-serif }
```

Селекторами типа являются имена элементов, например, h1 для тега <h1>, p для тега <p>, em для тега и т. д. В этом случае описываемые стили применяются ко всем элементам, имеющим такие имена. Пример:

```
p { color: #FF0000}
```

Описываемый стиль, в котором определяется цвет текста, применяется ко всем тегам <p>, т. е. любой текст, записанный внутри тегов <p>, примет красный цвет.

Приемы стилей с базовыми селекторами приведены в листинге П2.2.

Листинг П2.2. Стили, применяемые к тегам

```
<style type="text/css">
  /* стиль для тега <body> */
  body {font-size: 12px; color: #000000; background-color: #FFFFFF}
  h1 {font-size: 24px}                    /* стиль для заголовка */
  p {font-size: 14px; color: #00FF00}     /* стиль для параграфа */
</style>
```

Здесь стили применяются ко всем тегам заданного типа (<body>, <h1> и <p>).

Селектор *class*

Для применения стилей к группам тегов следует использовать атрибут HTML-тега `class`. Ниже приведены примеры использования атрибута `class` в HTML-коде.

```
<p class=txt>Текст</p>
<div class=txt>Текст</p>
<table><tr><td class=txt>Текст</td></tr></table>
```

В данном случае к элементам применяется один стиль с именем `txt`. При определении стилей для классов селекторы записываются через точку перед именем класса (листинг П2.3).

Листинг П2.3. Стили, применяемые к группам тегов (классы)

```
<style type="text/css">
  .txt{font-size: 14px; color: #00FF00}
</style>
```

Селектор *id*

Для назначения стилей конкретному элементу следует воспользоваться атрибутом `id`. Атрибут `id` задает уникальное имя элемента, с помощью которого к этому элементу можно обращаться из сценариев JavaScript и таблиц стилей. Ниже показан пример определения идентификатора `id` в HTML-коде:

```
<p id="info">Текст</p>
```

Значение атрибута `id` должно быть уникально в пределах страницы. Если в качестве селектора выступает идентификатор элемента, то его следует записывать через символ решетки `#` перед именем идентификатора (листинг П2.4).

Листинг П2.4. Стили, применяемые к конкретным элементам

```
<style type="text/css">
  #info {font-size: 14px; color: #00FF00}
</style>
```

Приоритеты применения стилей CSS

Нередки ситуации, когда стили, применяемые на странице, конфликтуют между собой, и к одному элементу может быть применено несколько разных

стилей. Какой именно стиль будет выбран, определяется браузером на основе правил каскадирования. Используется стиль, содержащий более точное описание элемента, к которому он применяется. Например, можно определить стили для всех параграфов, а также стиль для одного конкретного параграфа.

Например, если всем тегам <p> задать стиль с помощью тега <style>:

```
<style>
  p {fobt-size: 12px}
</style>
```

а у одного параграфа (тег <p>) определить стили через атрибут style:

```
<p style="font-size: 14px">
```

то для этого параграфа стили, определенные через атрибут, будут иметь больший приоритет, чем стили, назначаемые ко всем тегам <p>. Рассмотрим это подробнее.

Наименьший приоритет имеют стили родительских документов (листинг П2.5).

Листинг П2.5. Определение родительского стиля

```
<html>
<head>
  <style type="text/css">
    body{font-size: 12px; color: #000000; font-family: Arial, sans-serif}
    p{font-size: 16px}
  </style>
</head>
<body>
  <div>Текст внутри тега div</div>
  <p>Текст внутри тега параграфа</p>
</body>
</html>
```

В данном примере определен стиль для тега <body>, который является родительским по отношению к тегам и <p>, и, следовательно, имеет меньший приоритет. Как можно видеть на рис. П2.1, при отображении тега были применены стили родительского тега <body>, в то время как для тега <p> стили родительского элемента были переопределены стилями, заданными непосредственно для тега p{font-size: 16px}.

Стили, применяемые к классу, имеют более высокий приоритет, чем стили тегов (листинг П2.6).

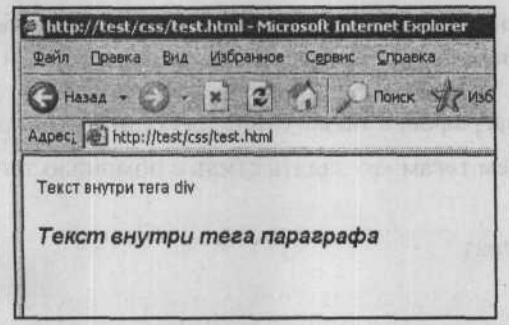


Рис. П2.1. Применение стилей

Листинг П2.6. Стили, применяемые к классу

```
<html>
<head>
<style type="text/css">
  p{font-size: 12px; color: #000000; font-family: Arial, sans-serif;
  .txt{font-size: 16px; font-style: italic}
</style>
</head>
<body>
  <p>Обычный текст внутри тега параграфа</p>
  <p class=txt>Текст внутри тега параграфа,
              с назначенным классом стилей</p>
</body>
</html>
```

Результат выполнения HTML-кода из листинга П2.6 приведен на рис. П2.2. Как видно из рисунка, стиль класса `txt` имеет более высокий приоритет и переопределяет стили, назначаемые всем тегам `<p>`.

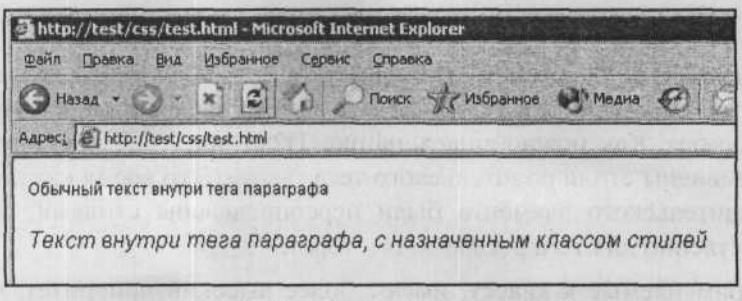


Рис. П2.2. Использование класса стилей

Стили, определенные через идентификатор id, имеют более высокий приоритет, чем стили родительского элемента, стили тегов и стили классов. Самый высокий приоритет имеют внедренные стили, определяемые в атрибуте style.

```
<div style="font-size: 13px; border-style: solid"></div>
```

Комбинация селекторов стилей

Гибкость применения стилей значительно расширяется при использовании контекстных селекторов, которые позволяют определять стили элементов в зависимости от их вложенности друг в друга. Например:

```
p em {font-size: 16px; font-weight: bold; color: #00FF00}
```

Данный стиль определен для тегов , которые находятся внутри тегов <p>.

Область действия селекторов классов, определяемых с помощью атрибута class, может быть более детализирована и применяться к разным тегам.

```
a.link{color: #003399}
p.link{background-image: url(link.jpg); padding-left: 20px}
```

Здесь стиль с одним и тем же именем link определяет разные свойства для тега <a> и тега <p> . Вложенность контекстных селекторов может быть неограниченной:

```
div.inform p em {background-color: #0000FF}
```

Данный стиль применяется к тегу , который находится внутри тега <p>, а тот, в свою очередь, — внутри тега с назначенным ему классом стилей inform.

Если один и тот же стиль применяется к нескольким элементам, то их можно сгруппировать и применить к ним одно правило. Пример:

```
h1 { font-family: Arial, sans-serif; color: #00FF00 }
h2 { font-family: Arial, sans-serif; color: #00FF00 }
h3 { font-family: Arial, sans-serif; color: #00FF00 }
```

Здесь определены три идентичных друг другу стиля для тегов <h1>, <h2>, <h3>. Их можно записать в сокращенном формате, перечисляя имена тегов (селекторы) через запятую:

```
h1, h2, h3 { font-family: Arial, sans-serif; color: #00FF00 }
```

Эти записи идентичны.

Разные стили для разных устройств вывода

Язык HTML одинаково интерпретируется разными устройствами отображения, например, дисплей монитора, принтер, проектор и т. д. В то же время таблицы стилей могут применяться к документу в зависимости от медиасреды, в которой происходит отображение страницы. Например, при выводе страниц на печать целесообразно убрать ненужные элементы дизайна страницы и вывести на принтер только тестовую информацию или совсем поменять оформление страницы. Определение таблиц стилей для разных медиасред задается в свойстве `media`. По умолчанию свойство `media` имеет значение `screen`, которое определяет таблицу стилей для экрана монитора.

```
<link rel="stylesheet" type="text/css" href="display.css">
<link rel="stylesheet" type="text/css" href="print.css" media="print">
```

В приведенном примере при отображении на экране монитора будет применена таблица стилей из файла display.css. А при выводе на принтер к документу будет применена таблица стилей из файла print.css, т. к. значение свойства `media` для этой таблицы стилей установлено равным `print`. Перечислим возможные значения свойства `media`:

- `screen` — подразумевается экран монитора;
- `tty` — носитель, использующий сетку символов фиксированного размера, такой как телетайп, текстовые терминалы и т. п.;
- `tv` — устройство типа телевизора, которое имеет небольшое разрешение, низкую цветопередачу, отсутствие прокрутки экрана;
- `projection` — проектор;
- `handheld` — портативные устройства, которые характеризуются маленькими экранами, растровой графикой, частотными ограничениями, монохромным дисплеем;
- `embossed` — принтер, печатающий азбукой Брайля (для слабовидящих);
- `print` — для использования при печати на страничном непрозрачном материале;
- `braille` — устройства для слабовидящих;
- `aural` — речевой синтезатор;
- `all` — применяется ко всем устройствам.

Шрифты

Разнообразие используемых для HTML-страниц шрифтов достаточно ограничено. Фактически, на всем многообразии HTML-страниц царствуют два типа шрифтов — с засечками и без засечек. Наиболее распространенные шрифты этих типов — Times New Roman и Arial соответственно. Хотя сами технологии позволяют разнообразить шрифтовое пространство страницы, но фактически эти возможности не используются. Дело в том, что шрифты, используемые на страницах, должны быть доступны на компьютерах посетителей, иначе они не смогут быть отображены. На фоне большого количества компьютерных систем и существующих шрифтов нельзя дать никакой гарантии, что шрифт, используемый Web-мастером, будет иметься в наличии на компьютере посетителя. И в основном, разработчики не хотят рисковать и ограничиваются использованием стандартных шрифтов. В настоящее время появились технологии, решающие проблемы шрифтов, но они еще не до конца отработаны и поддерживаются не всеми типами браузеров. Кроме того, их распространению препятствует невозможность поддержки данных технологий в старых версиях браузеров.

Спецификация CSS позволяет управлять видом шрифтов (начертание), размером, стилем (прямой, наклонный и курсивный) и толщиной шрифта.

Определение типа шрифта

Для определения типа шрифта используется свойство `font-family`. В нем задается список шрифтов или семейств шрифтов, например:

```
font-family: "Arial Cyr", Arial, Helvetica, Sans-Serif
```

Когда страница загружается в браузер, сначала проводится поиск шрифта с именем Arial Cyr, если он не найден, то ищется следующий шрифт с именем Arial и так до конца списка. Если имя шрифта содержит пробелы, то его следует помещать в кавычки.

В CSS можно определять не только конкретный шрифт, а целое семейство шрифтов. Таким образом, если ни один желаемый шрифт не найдется на компьютере посетителя, то для отображения текста будет использован любой шрифт из указанного семейства. Существуют пять семейств шрифтов:

- `serif` (например, Times New Roman);
- `sans-serif` (например, Arial);
- `cursive` (например, Zapf-Chancery);
- `fantasy` (например, Western);
- `monospace` (например, Courier).

Также существуют комбинации шрифтов, рекомендованные к использованию на Web-страницах. В них перечислены шрифты, установленные на компьютерах большинства пользователей. Например:

```
Arial, Helvetica, sans-serif
Times New Roman, Times, serif
Courier New, Courier, monospace
Georgia, Times, New Roman, Times, serif
Verdana, Arial, Helvetica, sans-serif
```

Загружаемые шрифты

К достоинствам использования загружаемых шрифтов относится то, что разработчику не следует беспокоиться об их наличии на компьютерах посетителей Web-страницы. Единственным, но важным недостатком этой технологии является неполная поддержка ее современными браузерами.

Загружаемые шрифты определяются с помощью свойства `@font-face`, записываемого в формате:

```
@font-face { font-family: MyFont;
             scr:url(http://www.myserver.com/MyFont.eot); }
```

Здесь определяется шрифт с именем `MyFont` и указывается `url`, по которому находится файл со шрифтом, необходимым для загрузки. Загружаемый шрифт должен быть записан в особом формате и иметь расширение eot.

Для того чтобы применить загруженный шрифт к тексту, следует указать его в свойстве `font-family` для элемента.

```
p {font-family: MyFont}
```

Замечание

Для создания загружаемых шрифтов следует воспользоваться Web-программой, размещенной на сайте Microsoft по адресу **http://www.microsoft.com /typography/**. Там же вы сможете более подробно ознакомиться с технологией загружаемых шрифтов.

Стили шрифтов

Стиль шрифта определяется свойством `font-style`, которое может принимать три значения:

- `normal` — нормальный;
- `italic` — курсивный;
- `oblique` — наклонный.

Например:

```
p { font-style: italic }
```

Замечание

В типографском деле наклонный — это скошенный прямой шрифт, а курсивный — это специально спроектированный скошенный шрифт. В настоящее время, при просмотре в браузерах, отображения курсивного и наклонного стиля шрифтов не отличается.

Варианты шрифтов

Свойство `font-variant` позволяет вывести текст прописными буквами уменьшенного размера.

```
.txt { font-variant: small-caps }
```

Применим этот стиль к тексту:

```
<p>Обычный текст</p>
<p class=txt>Текст со свойством font-variant: small-caps</p>
```

Результат изображен на рис. П2.3.

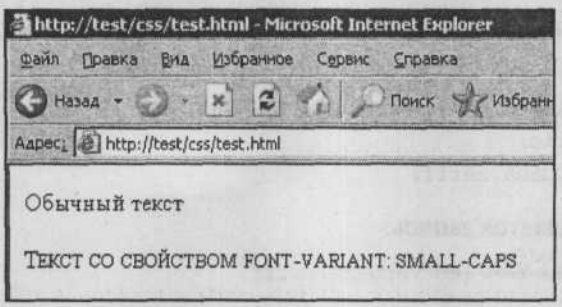


Рис. П2.3. Использование свойства `font-variant`

Полнота шрифта

Степень полноты или жирности шрифта устанавливается в свойстве `font-weight`. Например:

```
P { font-weight: bold }
```

Существует 9 градаций полноты шрифта со значениями от 100 до 900 с шагом 100. Значение 100 соответствует самому бледному шрифту, 900 — самому жирному. Обычный текст имеет значение 400 для данного свойства. Так-

же для определения полноты применяются ключевые слова: `normal` (обычный), `bold` (полужирный), `bolder` (более жирный), `lighter` (бледнее).

Замечание

К сожалению, в настоящее время большинство браузеров поддерживает только две градации полноты: обычный (`normal`) и полужирный (`bold`) — в связи с чем, указание цифровых значений теряет смысл. Некоторые семейства шрифтов также имеют только две выше обозначенные градации.

Сокращенный формат записи стилей шрифтов

Сокращенный формат записи стилей служит для их более компактной записи, но накладывает ограничения на порядок перечисления свойств. Если при определении стилей в полном формате свойства могут быть перечислены в произвольном порядке, то при сокращенной записи нужно соблюдать следующий формат:

```
font: font-style, font-variant, font-weight,
      font-size, font-height, font-family
```

Первые 3 свойства могут быть опущены. Свойства `font-size` и `font-height` должны быть записаны через наклонную черту.

Например, для стилей

```
font-size: 2em;
font-weight: bold;
font-style: italic;
font-family: verdana,serif;
```

Сокращенной является запись:

```
font: italic bold 2em/1em verdana,serif
```

Единицы измерения и размеры шрифтов

Спецификация CSS определяет несколько единиц измерения, которыми Web-мастер может оперировать при верстке HTML-страниц. Их можно условно разделить на две группы: абсолютные и относительные единицы измерения.

Абсолютные единицы измерения:

- `in` — дюймы (1 дюйм = 2,54 см);
- `cm` — сантиметры;
- `mm` — миллиметры;

- pt — пункты (1 пункт = 1/72 дюйма);
- pc — пики (1 пика = 12 пунктам).

Относительные единицы измерения:

- em — размер шрифта;
- ex — размер символа x;
- % — проценты;
- px — пикселы;
- относительные ключевые слова (larger и smaller);
- относительные ключевые слова для задания абсолютного размера шрифтов.

Замечание

Пикселы хотя и принадлежат группе относительных единиц измерения, на самом деле имеют двойственную природу и сочетают в себе качества также и абсолютных единиц измерения. Подробно пикселы будут рассмотрены в *разд. "Использование пикселов" далее в этом приложении.*

Особое значение единицы измерения имеют применительно к размеру шрифтов. Именно о единицах измерения шрифтов далее пойдет речь. Применение единиц измерения к другим элементам, как правило, не вызывает проблем. В качестве задания абсолютных величин измерения обычно используются пикселы, а в качестве относительных — проценты.

Недостатки абсолютных единиц измерения

По признанию большинства Web-мастеров, абсолютные единицы измерения обладают более выраженными недостатками. Задание абсолютных единиц измерения не позволяет посетителю изменить размер шрифтов через меню браузера и адаптировать страницу под свои настройки. Наиболее часто встречающаяся проблема — это маленький размер шрифта, делающий текст на экране нечитаемым. В большинстве случаев невозможно заранее предусмотреть, на каком мониторе, какого размера и разрешения будет просматриваться сайт. Текст, нормально воспринимаемый на экране с небольшим или средним разрешением, может стать абсолютно нечитаемым при высоком разрешении. Эти проблемы становятся критичными при использовании абсолютных единиц измерения.

Проблемы при использовании пунктов

Наиболее часто применяется абсолютная единица измерения — пункты (pt, пт). 1 пункт равен 1/72 дюйма, и именно с данным соотношением связана

часть проблем пунктов, как единиц измерения. Дело в том, что разрешение экрана монитора вычисляется в пикселах, и для того чтобы отобразить текст, заданный в пунктах, на экране его нужно перевести в пикселы. Стандартное разрешение компьютеров PC равно 96 dpi (dots per inch, точек на дюйм), т. е. в одном дюйме находятся 96 пикселов. Таким образом, 1 пункт, как 1/72 дюйма на экране PC равен примерно 1,3 пиксела. Но не все системы имеют такое же стандартное разрешение. Разрешение компьютеров Macintosh равно 72 dpi, т. е. в одном дюйме содержится 72 пиксела и 1 пункт равен 1 пикселу. В результате один и тот же текст, размер которого задан в дюймах, на экране Macintosh выглядит в 1,3 раза меньшим, чем тот же самый текст на компьютерах PC. При использовании шрифта, высотой 9 пунктов и меньше, текст на экране Macintosh становится нечитаемым.

Используйте пункты для печати

Задание размеров в пунктах или других абсолютных единицах измерения удобно применять при использовании альтернативных таблиц стилей, применяемых для печати страниц. Размеры листа бумаги однозначно вычисляются в дюймах, и результат визуализации будет предсказуем.

Относительные единицы измерения *em, ex, %*

Относительные единицы измерения разрешают пользователям изменять размеры шрифтов через меню браузера и, таким образом, адаптировать страницу под настройки своей компьютерной системы и собственные предпочтения.

Размер шрифта в `1em` соответствует 100-процентному размеру шрифта родительского элемента. А размер шрифта в `1ex` соответствует размеру символа `x` шрифта родительского элемента. Размеры, заданные в процентах, применительно к шрифтам фактически соответствуют единице измерения `em`. Далее при рассмотрении в этой главе будем использовать единицу измерения `em`.

Если задать размер шрифта, например `0.9em`, то он будет визуализирован как 90% от размеров шрифта родительского элемента. Например, создадим таблицу стилей:

```
body{font-size: .9em}
p{font-size: .9em}
```

И применим ее к странице:

```
<body>
  Текст внутри тега body
  <p>Текст внутри тега параграфа</p>
</body>
```

Допустим, настройки браузера устанавливают размер шрифта равным `16pt`. Таким образом, первая строка текста будет визуализирована в размере 90% от размера шрифта в настройках браузера. Это равно `14.4pt`. Вторая строка, помещенная в тег `<p>`, будет выведена на экран еще меньшим шрифтом — в размере 90% от шрифта, заданного в теге `<body>`, т. к. именно он является родительским элементом. Таким образом, размер шрифта второй строки будет равен примерно `13pt` (рис. П2.4).

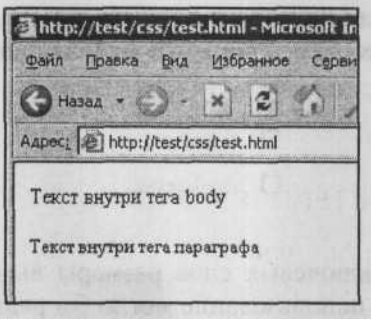


Рис. П2.4. Относительные размеры шрифтов

Использование пикселов

Пикселы, хотя и принадлежат относительным единицам, на самом деле сочетают в себе качества и абсолютных, и относительных единиц измерения. Относительность определяется тем, что видимый размер, указанный в пикселах, зависит от разрешения монитора. На мониторе с небольшим разрешением видимый размер будет больше, чем тот же самый размер в пикселах на мониторе с высоким разрешением. Это связано с тем, что сами размеры пикселов неизменны, но на большом мониторе для отображения содержимого экрана используется большее количество пикселов. Таким образом, при увеличении монитора размеры, заданные в пикселах, уменьшаются.

Абсолютность пикселов обусловлена независимостью данной единицы измерения от настроек браузера, т. е. шрифт, заданный в размере `12px`, всегда станет визуализироваться в этом размере и его будет невозможно изменить настройками браузера. Таким образом, пикселы можно назвать относительными единицами измерения, независящими от настроек браузера.

Относительные ключевые слова

Задать относительный размер шрифта можно указанием ключевых слов `larger` и `smaller`. Ключевое слово `larger` указывает, что размер шрифта будет на один размер больше шрифта родительского элемента, а ключевое слово

smaller, наоборот, уменьшает размер шрифта. Недостатки относительных ключевых слов в том, что сложно точно определить величину шрифта при их использовании, т. к. это зависит от браузера.

Относительные ключевые слова для задания абсолютного размера шрифтов

Применение этих ключевых слов должно было заменить систему измерения, реализованную в теге <font>. Для указания размеров доступно семь ключевых слов, определяющих размер шрифтов от меньшего к большему:

- xx-small;
- x-small;
- small;
- medium;
- large;
- x-large;
- xx-large.

При применении этих ключевых слов размеры вычисляются относительно настроек браузера, и их использование могло бы решить проблемы со шрифтами. Однако до настоящего момента производители браузеров не обеспечили корректную поддержку данных ключевых слов. В связи с чем их применение в реальной практике не рекомендуется.

Проблемы использования относительных единиц измерения

Хотя относительные единицы измерения обладают многими преимуществами по сравнению с абсолютными, они тоже не лишены недостатков. В основном, проблемы вызывает неаккуратная расстановка стилей на странице. Применение относительных единиц измерения требует от Web-мастера большой внимательности при верстке HTML-кода, т. к. в этом случае размеры шрифтов зависят не только от назначенных элементам стилей, но и от размеров шрифтов родительских элементов. Приведем просто пример HTML-кода, иллюстрирующий данный вид проблем (листинг П2.7).

Листинг П2.7. Проблемы использования относительных единиц измерения

```
<html>
  <head>
    <style type="text/css">
      body{font-size: 16pt}
      div{font-size: .8em}
    </style>
  </head>
```

```
  <body>
    <div>Текст внутри тега div
      <div>Текст внутри вложенного тега div
        <div>Текст внутри вложенного тега div повторно</div>
      </div>
    </div>
  </body>
</html>
```

На рис. П2.5 изображен результат вывода этой страницы в браузер.

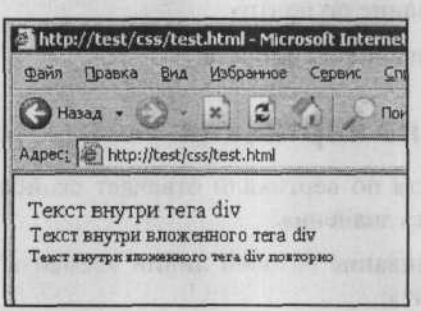


Рис. П2.5. Проблемы использования относительных единиц измерения

Как видно из рисунка, текст в третьей строке стал практически нечитаемым, хотя маловероятно, что это входило в задачи Web-мастера.

Этого недостатка относительных единиц измерения можно избежать, если стараться применять стили не к тегам, а к классам и использовать вложенные контекстные селекторы, например, как это показано ниже. Изменяем описание стилей:

```
body{font-size: 16pt}
.txt{font-size: .8em}
```

И расстановку стилей на странице:

```
<div class=txt>Текст внутри тега div
  <div>Текст внутри вложенного тега div
    <div>Текст внутри вложенного тега div повторно</div>
  </div>
</div>
```

Форматирование текста

Спецификация CSS определяет свойства, позволяющие изменять текст: выравнивание, межстрочный интервал, межсимвольный интервал и т. п.

Выравнивание по горизонтали

Выравнивание текста по горизонтали определяется с помощью свойства `text-align`, например:

```
div { text-align: left }
```

Свойство `text-align` может принимать значения:

- `left` — выравнивание по левому краю;
- `right` — выравнивание по правому краю;
- `center` — выравнивание по центру;
- `justify` — выравнивание по ширине.

Выравнивание по вертикали

За выравнивание текста по вертикали отвечает свойство `vertical-align`, которое может принимать значения:

- `baseline` — выравнивание базовой линии элемента по базовой линии родительского элемента;
- `middle` — выравнивание средней точки элемента по базовой линии родительского элемента;
- `text-top` — выравнивание верха элемента по верху шрифта родительского элемента;
- `text-bottom` — выравнивание нижней линии элемента по нижней линии родительского элемента;
- `top` — выравнивание верха элемента по верху самого высокого элемента строки;
- `bottom` — выравнивание низа элемента по низу элемента строки, расположенного ниже всех;
- `sub` — отображение элемента в виде нижнего индекса;
- `super` — отображение элемента в виде верхнего индекса;
- абсолютное значение — увеличение или уменьшение (отрицательное значение) текущего межстрочного интервала на данную величину;
- проценты — задание межстрочного интервала в процентах от текущего.

Определение

Базовой является линия, на которой лежит текст без учета некоторых символов, таких как "р", "д", "щ", "ц" и т. п.

Замечание

Свойство `vertical-align` не наследуется от родительских элементов.

Оформление текста

Свойство `text-decoration` управляет оформлением текста и принимает значения.

- `underline` — подчеркнутый текст;
- `overline` — надчеркнутый текст;
- `line-through` — перечеркнутый текст;
- `blink` — мерцающий текст. В настоящий момент это значение не поддерживается браузерами;
- `none` — убирает свойства `text-decoration`, унаследованные от родительских элементов.

Межсимвольное расстояние

Свойство `letter-spacing` управляет межсимвольным расстоянием. Значением данного свойства является число с указанием единиц измерения, либо ключевое слово `normal`. При использовании относительных единиц измерения их значения отсчитываются относительно значений текущего шрифта. Ключевое слово `normal` устанавливает межсимвольное расстояние в размеры, определенные для текущего шрифта.

Примеры:

```
p { letter-spacing: 0.5em }
p { letter-spacing: 0.5cm }
```

На рис. П2.6 можно видеть результат применения этого стиля со значением `letter-spacing`, равным `0.5em`.

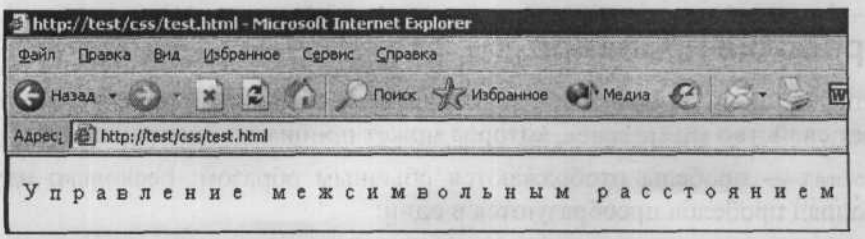


Рис. П2.6. Управление межсимвольным расстоянием

Расстояние между словами

Свойство `word-spacing` управляет расстоянием между словами. Значением этого свойства является число с указанием единиц измерения, либо ключевое слово `normal`. При использовании относительных единиц измерения их значения отсчитываются относительно значений текущего шрифта. Ключевое слово `normal` устанавливает межсловное расстояние в размеры, определенные для текущего шрифта.

Пример:

```
p { word-spacing: 1.5em }
```

Результат применения этого стиля со значением `word-spacing`, равным `1.5em`, изображен на рис. П2.7.

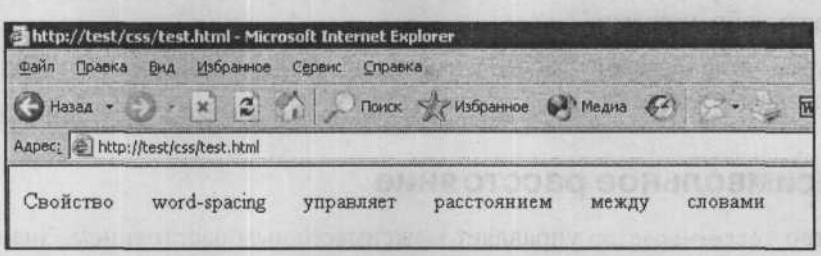


Рис. П2.7. Управление расстоянием между словами

Межстрочное расстояние

Межстрочное расстояние определяет, насколько одна строка находится выше другой. Межстрочное расстояние устанавливается с помощью свойства `line-height`, значением которого является число с указанием единиц измерения. При использовании относительных единиц измерения их значения отсчитываются относительно значений текущего шрифта.

Пример:

```
p { line-height: 14px}
```

Обработка пробелов

Спецификация CSS позволяет задать правила обработки пробелов. За это отвечает свойство `white-space`, которое может принимать значения:

- `normal` — пробелы отображаются обычным образом: несколько идущих подряд пробелов преобразуются в один;
- `pre` — несколько идущих подряд пробелов не преобразуются в один, а отображаются, как есть;

- `nowrap` — запрет на перенос строки. Возможен только принудительный разрыв строки с помощью тега `<br>`.

Замечание

Для Internet Explorer 6 значение `pre` поддерживается только при установке режима совместимости со спецификацией CSS.

Прописные и строчные буквы

Свойство `text-transform` позволяет отображать текст прописными или строчными буквами и может принимать значения:

- `uppercase` — прописные буквы;
- `lowercase` — строчные буквы;
- `capitalize` — каждое слово начинается с большой буквы;
- `none` — обычный текст. Необходимо для снятия стилей родительских элементов.

Пример:

```
.upper {text-transform: uppercase}
.lower {text-transform: lowercase}
.capitalize {text-transform: capitalize}
```

Результат применения этих стилей показан на рис. П2.8.

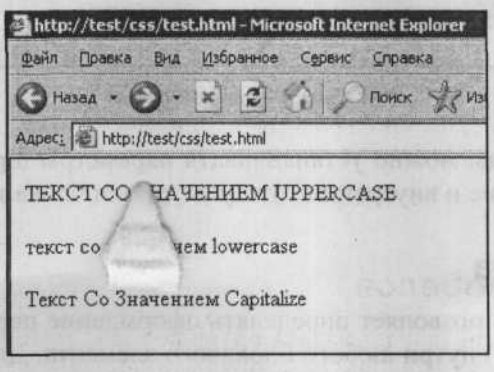


Рис. П2.8. Прописные и строчные буквы

Замечание

Свойство `text-transform` не наследуется от родительских элементов.

Отступ первой строки

Для задания отступа первой строки следует использовать свойство `text-indent`, значением которого является число с указанием единиц измерения. При использовании относительных единиц измерения их значения отсчитываются относительно значений текущего шрифта.

Пример:

```
p { text-indent: 20px }
```

Буквица

Буквицей называют выделенную первую букву в абзаце. Создать буквицу и определить ее параметры можно с помощью псевдоэлемента `first-letter`, например:

```
p:first-letter { font-size: 300%; vertical-align: baseline }
```

Результат применения этого стиля изображен на рис. П2.9.

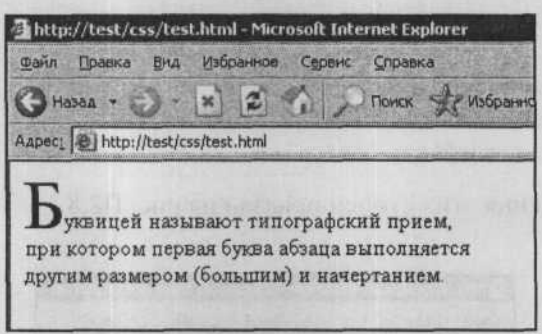


Рис. П2.9. Создание буквицы

В свойствах буквицы можно устанавливать параметры шрифтов, цвет, параметры фона, внешние и внутренние отступы, рамки блока и т. п.

Первая строка

Спецификация CSS позволяет определять оформление первой строки текста, который находится внутри любого блокового элемента. Для этого предназначен псевдоэлемент `first-line`, к которому можно применять стили для определения свойств шрифта, цвета, фона, стили форматирования текста.

Пример:

```
div:first-line { text-transform: uppercase; font-weight: bold }
```

На рис. П2.10 показан результат применения этого стиля HTML-коду:

```
<table width="200px" border="1">
  <tr>
    <td>Оформление первой строки. Спецификация CSS позволяет определять
        оформление первой строки текста, который находится внутри любого
        блокового элемента.
    </td>
  </tr>
</table>
```

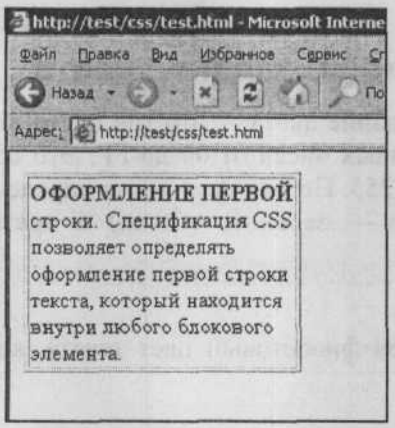


Рис. П2.10. Оформление первой строки текста

Тень текста

С помощью свойства `text-shadow` можно задавать тень у текста. Свойство `text-shadow` содержит список эффектов, разделенных запятыми.

`text-shadow: [цвет] смещение_вправо смещение_вниз [радиус_пятна], ...`

Параметры *смещение_вправо* и *смещение_вниз* определяют смещение тени вправо и вниз от текста и могут иметь отрицательные значения. Необязательный элемент *цвет* задает цвет тени. Если цвет тени не задан, то используется цвет текста. Параметр *радиус_пятна* тени также является необязательным элементом. Алгоритм вычисления пятна зависит от браузера.

Замечание

В настоящее время свойство `text-shadow` не поддерживается большинством браузеров.

Определение цвета

CSS предлагает несколько способов определения цвета:

- шестнадцатеричное;
- трехзначное шестнадцатеричное;
- RGB-значения;
- RGB-значения в процентах;
- ключевые слова;
- системные цвета.

Шестнадцатеричное задание цвета

Шестнадцатеричное задание цвета — это три цветовых составляющих в диапазоне шестнадцатеричных чисел от 00 до FF. Это соответствует диапазону десятичных чисел 0 до 255. Первая пара чисел определяет компонент красного цвета (Red), вторая — зеленого (Green) и третья — голубого (Blue): `#RRGGBB`. Например:

```
p { color: #663399}
```

В этом стиле определен фиолетовый цвет текста, который будет помещен в теги `<p>`.

Трехзначное шестнадцатеричное задание цвета

Шестнадцатеричный формат можно записывать в упрощенном виде `#RGB`, что соответствует формату `#RRGGBB`. Таким образом, запись вида `#F00` соответствует записи `#FF0000`.

RGB-значения

CSS позволяет указывать цвет в виде тройки RGB-значений, записанных десятичными числами.

```
em {color: rgb (0,255,100) }
```

Значения цветовых компонентов лежат в диапазоне о 0 до 255.

RGB-значения в процентах

Кроме десятичных чисел, при указании цвета тройками RGB-значений можно применять проценты. Значения лежат в диапазоне от 0 до 100%.

```
em {color: rgb (0%,100%,55%) }
```

Ключевые слова

При назначении цвета можно применять именованные цвета, определенные в спецификации HTML:

- `Black` — черный (`#000000`);
- `White` — белый (`#FFFFFF`);
- `Gray` — серый (`#808080`);
- `Silver` — светло-серый (`#C0C0C0`);
- `Green` — темно-зеленый (`#008000`);
- `Lime` — ярко-зеленый (`#00FF00`);
- `Olive` — оливковый (`#808000`);
- `Yellow` — желтый (`#FFFF00`);
- `Aqua` — светлый сине-зеленый (`#00FFFF`);
- `Teal` — темный сине-зеленый (`#000808`);
- `Blue` — синий (`#0000FF`);
- `Navy` — темно-синий (`#000080`);
- `Fuchsia` — ярко-розовый (`#FF00FF`);
- `Purple` — фиолетовый (`#800080`);
- `Red` — красный (`#FF0000`);
- `Maroon` — красно-коричневый (`#800000`).

Совет

Полный список именованных цветов и их шестнадцатеричных значений можно посмотреть по адресу **http://www.htmlref.com/reference/appe/colorchart.htm**.

Системные цвета

В спецификации CSS2 появилась возможность использовать системные цвета, связанные с графическим окружением пользователя, например, учитывать цвет фона рабочего стола, цвет активной границы окна и т. п. Это позволит создавать страницы, встроенные в графическое окружение пользователя, которое он сам себе выбрал.

Ниже приведен список ключевых слов для определения цвета:

- `ActiveBorder` — цвет активной границы окна;
- `ActiveCaption` — цвет активного заголовка окна;
- `AppWorkspace` — фоновый цвет многооконного интерфейса;

- `Background` — фоновый цвет рабочего стола;
- `ButtonFace` — цвет лицевой стороны трехмерных элементов интерфейса;
- `ButtonHightlight` — темный участок трехмерных элементов;
- `ButtonShadow` — цвет тени трехмерных элементов;
- `ButtonText` — цвет текста на нажимаемых кнопках;
- `CaptionText` — цвет текста заголовка;
- `GrayText` — цвет затененного (неактивного) текста;
- `Highlight` — цвет позиции, выбранной в элементе управления;
- `HighlightText` — цвет текста, выбранного в элементе управления;
- `InactiveBorder` — цвет неактивной границы окна;
- `InactiveCaption` — цвет неактивного заголовка окна;
- `InactiveCaptionText` — цвет текста на неактивном заголовке;
- `InfoBackground` — фоновый цвет для всплывающих подсказок;
- `InfoText` — цвет текста всплывающих подсказок;
- `Menu` — фоновый цвет меню;
- `MenuText` — цвет текста меню;
- `Scrollbar` — серая зона полосы прокрутки;
- `ThreeDDarkShadow` — цвет темного участка трехмерных элементов изображений;
- `ThreeDFace` — цвет лицевой стороны трехмерных элементов изображений;
- `ThreeDHightlight` — цвет подсветки трехмерных элементов изображений;
- `ThreeDLightShadow` — цвет светлого участка трехмерных элементов изображений;
- `ThreeDShadow` — цвет темного участка трехмерных элементов изображений;
- `Window` — фоновый цвет окна;
- `WindowFrame` — цвет обрамления окна;
- `WindowText` — цвет текста в окне.

Фон

В CSS любые отображаемые теги могут иметь фон. Таким образом, фон может быть назначен и заголовку, и текстовым параграфам, и тегам выделения,

таким как <b>, и т. д. Фон может быть определен либо в виде одноцветного поля, либо в виде фонового изображения с различными параметрами.

Цвет фона

Для задания цвета фона следует использовать свойство background-color:

```
p { background-color: #FF56A6 }
```

Цвет фона можно задавать любыми способами, определяющими цвет, которые были рассмотрены ранее.

Фоновое изображение

За определение фонового изображения отвечает свойство background-image, записываемое в следующем формате:

```
background-image: url("fon.gif")
```

Фоновое изображение задается указанием пути к нему. Если фоновое изображение содержит прозрачные области, то они будут закрашены фоновым цветом, определяемым свойством background-color. Если фоновый цвет не установлен, то сквозь прозрачные области будут просвечивать нижележащие элементы.

Замечание

Путь к изображению определяется от местоположения файла стилей, а не от страницы, к которой таблица стилей была применена.

Повторяемость фрагмента фона

Очень часто фоновое изображение используется в виде небольшого фрагмента, размноженного по полю элемента. Характером размножения (повторяемостью) фонового фрагмента управляет свойство background-repeat, которое может принимать следующие значения:

- repeat — изображение повторяется по направлениям осей *x* и *y*;
- repeat-x — изображение повторяется по направлению оси *x*;
- repeat-y — изображение повторяется по направлению оси *y*;
- no-repeat — изображение не повторяется ни по одной из осей.

Если свойство background-repeat не установлено, то значением по умолчанию является repeat, которое полностью заполняет элемент фоновым изображением.

Пример:

```
td { background-image: url("fon.gif"); background-repeat: repeat-x}
```

Прокрутка фона

С помощью свойства `background-attachment` определяется, прокручивается фон листания страницы или остается неподвижным. Свойство `background-attachment` может принимать значения:

- `scroll` — фон прокручивается вместе со страницей;
- `fixed` — фон остается неподвижным при прокрутке страницы.

Если свойство `background-attachment` не определено, то значением по умолчанию является `scroll`.

Пример:

```
td { background-image: url("fon.gif"); background-attachment: fixed }
```

Позиционирование фона

Свойство `background-position` определяет положение фона относительно границ элемента. Значения этого свойства могут быть определены в абсолютных единицах измерения, в процентах и с помощью комбинаций ключевых слов. При использовании абсолютных величин левый верхний угол фонового изображения смещается на указанное значение от верхнего левого угла элемента, например:

```
td { background-image: url("fon.gif"); background-position: 20px 30px }
```

При указании позиционирования в процентах позиция фона вычисляется следующим образом:

- `background-position: 0% 0%` — помещает левый верхний угол изображения в левый верхний угол элемента;
- `background-position: 100% 100%` — помещает правый нижний угол изображения в правый нижний угол элемента.
- `background-position: 22% 79%` — помещает точку, лежащую на расстоянии 22% длины изображения левее и 79% высоты изображения ниже от верхнего левого угла изображения в точку элемента, лежащую на соответствующих расстояниях, но отсчитанных от размеров элемента.

Для задания положения используются также ключевые слова: `top`, `bottom`, `left`, `right`, `center`. Ниже приведены комбинации ключевых слов и их эквивалентные значения в процентах.

- top left, left top, 0% 0%
- top, top center, center top, 50%, 0%
- top right, right top, 100% 0%
- left, left center, center left, 0% 50%
- center, center center, 50% 50%
- right, right center, center right, 100% 50%
- bottom left, left bottom, 0% 100%
- bottom, bottom center, center bottom, 50% 100%
- bottom right, right bottom, 100% 100%

Сокращенная запись свойств фона

Для сокращенной записи свойств фона предназначено свойство background, в котором могут быть перечислены все описанные выше свойства:

```
background: [background-color] [background-image]
            [background-repeat] [background-attachment]
            [background-position]
```

Свойства перечисляются через пробел. Если какое-либо свойство не указано, то оно принимает значение по умолчанию. Например:

```
td {background: url("fon.gif") repeat-x 0% 50%}
```

Замечание

Не все браузеры понимают сокращенную запись свойств фона и игнорируют перечисленные в этом свойстве стили.

Внешний отступ, внутренний отступ, рамка

Структура блоковых элементов содержит различные параметры, влияющие на их визуальное представление. Кроме содержимого, блоковый элемент включает в себя внутренний и внешний отступы и рамку. Структура блоковых элементов представлена на рис. П2.11.

CCS позволяет управлять визуализацией параметров блокового элемента:

- внутренний отступ (padding): указание размера, цвет наследуется от фона содержимого;
- рамка (border): указание цвета и толщины рамки;
- внешний отступ (margin): указание ширины и цвета.

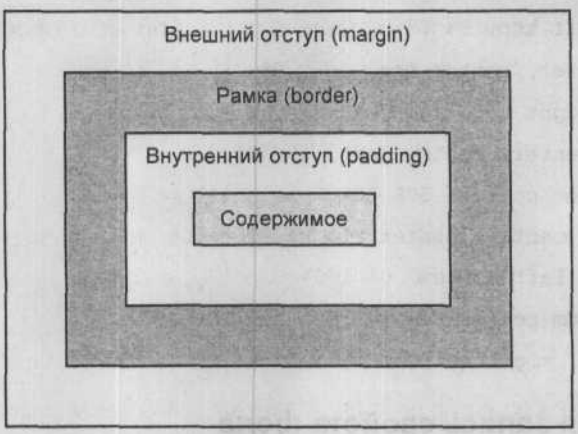


Рис. П2.11. Формат блокового элемента

Все вышеперечисленные элементы могут быть определены отдельно для каждой стороны: верх, низ, левая и правая стороны.

Определение

Объединение блока содержимого и блока внутреннего отступа называется *вмещающим блоком*. Это определение будет использоваться в дальнейшем.

Внутренний отступ

Параметры внутреннего отступа определяются свойством `padding`. Значения свойства `padding` могут быть заданы в виде абсолютных величин или в виде процентов, которые отсчитываются относительно размеров вмещающего блока. Таким образом, указание размера внутреннего отступа в 25% определяет размер внутреннего блока, равный 1/4 от размера блока содержимого. Размер отступа задается для каждой стороны. Таким образом, сумма внешних отступов слева и справа равна ширине блока содержимого (25% + 25% = 50%).

Пример:

```
div { padding: 25px}
```

Такая запись устанавливает одинаковые размеры внутреннего отступа для всех сторон.

CSS определяет свойства, позволяющие устанавливать внутренние отступы для каждой из сторон: `padding-right`, `padding-left`, `padding-top`, `padding-bottom`. Примеры:

```
p { padding-right: 10px, padding-left: 10px,
    padding-top: 5px, padding-bottom: 5px }
```

Возможна запись значений отступов для каждой из сторон в сокращенном формате.

```
padding: <верх> <право> <низ> <лево>
```

Например:

```
padding: 10px 0px 10px 0px
```

Если заданы два или три значения, то недостающие размеры принимаются равными размерам противоположных сторон.

Внешний отступ

Параметры внешнего отступа определяются свойством `margin`. Внешний отступ всегда прозрачный, в отличие от внутреннего отступа. Значения внешнего отступа могут быть заданы в виде абсолютных величин или в виде процентов, которые отсчитываются относительно размеров вмещающего блока.

Пример:

```
div { margin: 20px}
```

Такая запись устанавливает одинаковый размер внешнего отступа для всех сторон.

CSS определяет свойства, позволяющие устанавливать внешние отступы для каждой из сторон: `margin-right`, `margin-left`, `margin-top`, `margin-bottom`. Пример:

```
p { margin-right: 10px, margin-left: 10px,
    margin-top: 5px, margin-bottom: 5px }
```

Возможна запись значений отступов для каждой из сторон в сокращенном формате.

```
margin: <верх> <право> <низ> <лево>
```

Например:

```
padding: 10px 0px 10px 0px
```

Если заданы два или три значения, то недостающие размеры принимаются равными размерам противоположных сторон.

Рамка

Рамка определяет границу вмещающего блока и является отображаемым элементом. С помощью CSS у рамки может быть задан стиль отображения, толщина и цвет.

Стиль отображения

Включение отображения рамки и задание ее стиля определяется с помощью свойства `border-style`, которое может принимать одно из значений:

- `dotted` — рамка отображается точками;
- `dashed` — штрихпунктирная рамка;
- `solid` — сплошная линия;
- `double` — двойная рамка;
- `groove` — рамка отображается в трехмерном виде, как вдавленная;
- `ridge` — рамка отображается в трехмерном виде, как выдавленная;
- `inset` — весь элемент отображается, как вдавленный в лист;
- `outset` — весь элемент отображается, как выдавленный;
- `none` — рамка не отображается. Свойство `border-width` устанавливается в 0.

Пример:

```
p { border-style: solid }
```

Стиль отображения можно указывать отдельно для каждой из сторон:

- `border-top-style` — стиль верхней рамки;
- `border-right-style` — стиль правой рамки;
- `border-bottom-style` — стиль нижней рамки;
- `border-left-style` — стиль левой рамки.

Свойство `border-style` служит также для сокращенного формата описания стиля рамки и может содержать от 1 до 4 значений. Значения определяются в следующем формате:

```
border-style: <верх> <право> <низ> <лево>
```

Если какая-либо сторона не определена, то ее стиль принимается равным стилю противоположной рамки.

Размер рамки

Размер рамки определяется свойством `border-width`. Значением этого свойства может быть одно из ключевых слов, либо указание размера рамки в абсолютных величинах.

Для задания толщины рамки могут использоваться ключевые слова:

- `thin` — тонкая рамка;
- `medium` — средняя рамка;
- `thick` — толстая рамка.

Примеры:

```
td { border-style: solid;  border-width: 1px }
td { border-style: solid; border-width: thin }
```

Для указания размера рамки отдельно для каждой стороны применяются свойства:

- `border-top-width` — толщина верхней рамки;
- `border-right-width` — толщина правой рамки;
- `border-bottom-width` — толщина нижней рамки;
- `border-left-width` — толщина левой рамки.

Свойство `border-width` служит также для записи сокращенного формата и может содержать от 1 до 4 значений. Значения определяются в следующем формате:

```
border-width: <верх> <право> <низ> <лево>
```

Если какая-либо сторона не определена, то ее ширина принимается равной ширине противоположной рамки.

Цвет рамки

Для определения цвета рамки служит свойство `border-color`, например:

```
div { border-color: #FFFF00 }
```

Цвет может быть определен как для всех сторон одновременно, так и для каждой стороны в отдельности. Для этого предназначены свойства:

- `border-top-color` — цвет верхней рамки;
- `border-right-color` — цвет правой рамки;
- `border-bottom-color` — цвет нижней рамки;
- `border-left-color` — цвет левой рамки.

Свойство `border-color` служит также для записи сокращенного формата и может содержать от 1 до 4 значений. Значения определяются в следующем формате:

```
border-color: <верх> <право> <низ> <лево>
```

Если цвет какой-либо из сторон не определен, то он принимается равным цвету противоположной рамки.

Сокращенный формат описания рамок

Для записи значений рамок для каждой из сторон предназначены свойства: `border-top`, `border-right`, `border-bottom`, `border-left`, которые записываются в формате

`border-<сторона>: <размер рамки> <стиль рамки> <цвет рамки>`

Например:

```
border-top: thick solid red
```

Если какое-либо свойство не задано в сокращенном формате, то оно принимается равным значению по умолчанию. Для записи всех значений рамок всех сторон одновременно используется свойство `border`, записываемое в формате:

`border: <размер рамки> <стиль рамки> <цвет рамки>`

Позиционирование

При генерации HTML-документа браузер последовательно отображает элементы в том порядке, в котором они определены в HTML-коде. CSS позволяет изменить такой порядок отображения и определить позиционирование каждого элемента на экране. Для этого предназначено свойство `position`, которое может принимать следующие значения:

- `static` — статичное позиционирование; отображается последовательно в выходном потоке браузера;
- `relative` — относительное позиционирование; позиция элемента отсчитывается относительно его контейнера, выведенного в выходном потоке браузера;
- `absolute` — абсолютное позиционирование; позиция элемента отсчитывается относительно его абсолютно позиционированного предка, либо если такового нет, то относительно элемента `body`.

Абсолютно-позиционированные элементы выпадают из выходного потока браузера, и их положение не зависит от других элементов. Если у элемента есть абсолютно-позиционированные предки, то их положение отсчитывается относительно положения ближайшего абсолютно-позиционированного предка.

Пример:

```
.logo { position: absolute; width: 100px; height: 100px; top: 50px }
```

К абсолютно-позиционированному элементу могут применяться свойства `width` и `height`, устанавливающие ширину и высоту элемента. Если эти свой-

ства не заданы, то по ширине элемент распространяется до правого края окна браузера, а его высота равна значению, необходимому для отображения его содержимого. Если при установке ширины и высоты элемента его содержимое не умещается в установленные размеры, то эти части содержимого не отображаются на экране.

Замечание

При установке абсолютного позиционирования элемент может выйти за пределы отображения окна браузера и исчезнуть с экрана.

Управление перекрытием

Если абсолютно позиционированные элементы отображаются в одном и том же месте, то они будут перекрывать друг друга.

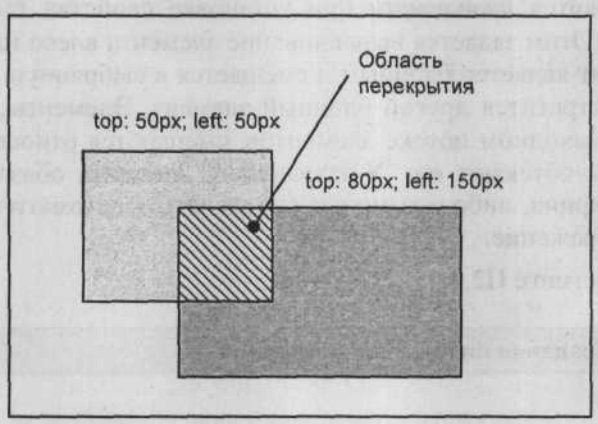


Рис. П2.12. Перекрытие абсолютно-позиционированных элементов

Управление перекрытием осуществляется с помощью свойства `z-index`, значением которого является число.

```
.num { z-index: 3 }
```

Чем больше число, тем выше находится элемент. Таким образом, элемент с бóльшим значением `z-index` будет перекрывать элемент с мéньшим значением `z-index`.

Пример:

```
.logo   {position: absolute; width: 100px; height: 100px; top: 50px;
         z-index: 2 }
.regard {position: absolute; width: 200px; height: 200px; top: 50px;
         z-index: 1 }
```

Установка смещений

Смещения для абсолютно- и относительно-позиционированных элементов устанавливаются с помощью свойств `left`, `top`, `right`, `bottom`. Значением этих свойств могут быть абсолютные величины, проценты и ключевое слово `auto`. При указании смещения в процентах они высчитываются относительно ширины и высоты обрамляющего блока. Ключевое слово `auto` определяет обычное положение элемента в выходном потоке элементов.

Примеры:

```
div.txt { top: 25px; left: 200px }
div.txt { right: 25px; top: 20px }
```

Плавающие элементы

Элемент становится *плавающим* при установке свойства `float` в значение `left` или `right`. Этим задается выравнивание элемента влево или вправо. Плавающий элемент является блочным и смещается в выбранную сторону до тех пор, пока не встретится другой блочный элемент. Элементы, следующие за плавающим в выходном потоке элементов, смещаются относительно его позиции и как бы обтекают его. У плавающего элемента обязательно должна быть задана ширина, либо она может определяться автоматически, если это, например, изображение.

Пример — в листинге П2.8.

Листинг П2.8. Создание плавающих элементов

```
<html>
<head>
<style type="text/css">
  div{border-style: solid; border-width: 1px; width: 100px; height: 10px;
      text-align: center}
</style>
</head>
<body>
  <div style="float: left">float=left</div>
  <div style="float: right">float=right</div>
  Текст...
</body>
</html>
```

На рис. П2.13 изображен результат применения стилей и создания плавающих элементов. Текст обтекает как левый, так и правый плавающий элемент.

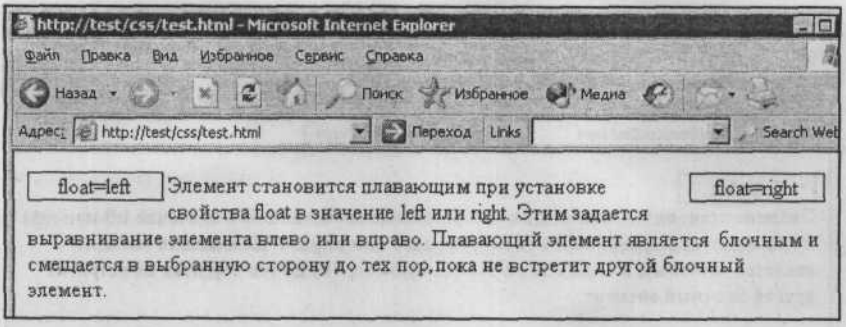


Рис. П2.13. Создание плавающих элементов

Обтекание текстом

Свойство `clear` управляет обтеканием предыдущих плавающих элементов и может принимать значения:

- `left` — запрещает обтекание предыдущего плавающего элемента слева;
- `right` — запрещает обтекание предыдущего плавающего элемента справа;
- `both` — запрещает обтекание предыдущего плавающего элемента с обеих сторон;
- `none` — разрешает обтекание предыдущего плавающего элемента с обеих сторон.

Указание запрета обтекания отображает элемент ниже плавающего элемента.

Пример — в листинге П2.9.

Листинг П2.9. Запрет обтекания текста слева

```
<html>
<head>
<style type="text/css">
 div{border-style: solid; border-width: 1px; width: 100px; height: 10px;
     text-align: center}
</style>
</head>
<body>
  <div style="float: left">float=left</div>
  <span style="clear:left">Текст...</span>
</body>
</html>
```

Результат изображен на рис. П2.14.

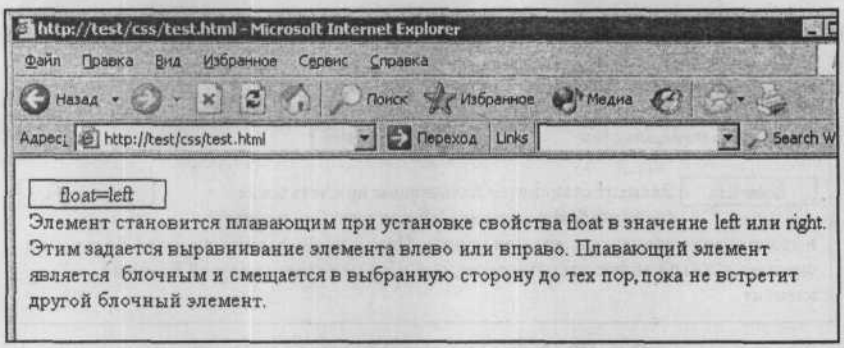


Рис. П2.14. Запрет обтекания текста слева

Псевдоклассы ссылок

Браузеры по-разному отображают ссылки, по которым был осуществлен переход и по которым переход не осуществлялся. Обычно они отличаются друг от друга цветами. Спецификация CSS1 определяет превдоклассы, которые применяются к ссылкам в зависимости от их состояния:

- `a:link {color: red}` — ссылка, по которой был осуществлен переход;
- `a:visited {color: blue}` — ссылка, по которой не был осуществлен переход;
- `a:active {color: green}` — активная ссылка;

Определение

Активная ссылка — это ссылка, выбранная в данный момент, например, с помощью клавиши <Tab>, или только что нажатая ссылка.

- `a:hover {color: navy}` — ссылка, над которой находится курсор мыши.

По умолчанию данные псевдоклассы применяются к тегам `<a>`, и при записи стилей тег `<a>` можно опустить, т. е. нижеприведенные описания стилей будут равнозначны:

```
a:hover {color: red}
:hover {color: red}
```

Списки

Таблицы стилей CSS позволяют разнообразить оформление стандартных списков HTML: назначать различные типы маркеров, использовать вместо маркеров изображения и определять положения маркера относительно списка.

Типы маркеров

Типы маркеров устанавливаются с помощью свойства `list-style-type`, которое может принимать одно из перечисленных ниже значений:

- `disc` — черный кружок;
- `circle` — светлый кружок;
- `square` — черный квадрат;
- `decimal` — десятичные числа (1, 2, 3, ...);
- `decimal-leading-zero` — десятичные числа с ведущим нулем (01, 02, 03, ...);
- `lower-roman` — строчные римские числа (i, ii, iii, iv, v, ...);
- `upper-roman` — прописные римские числа (I, II, III, IV, V, ...);
- `lower-greek` — строчные греческие буквы (α, β, γ, ..., ω);
- `lower-alpha` — строчные латинские буквы (a, b, c, ..., z);
- `lower-latin` — строчные латинские буквы (a, b, c, ..., z);
- `upper-alpha` — прописные латинские буквы (A, B, C, ..., Z);
- `upper-latin` — прописные латинские буквы (A, B, C, ..., Z);
- `hebrew` — нумерация на иврите;
- `armenian` — армянская нумерация;
- `georgian` — грузинская нумерация;
- `cjk-ideographic` — дальневосточная идеографическая нумерация;
- `hiragana` — японская нумерация азбукой хирагана;
- `katakana` — японская нумерация азбукой катакана;
- `hiragana-iroha` — японская нумерация хирагана-ироха;
- `katakana-iroha` — японская нумерация катакана-ироха.

Пример:

```
ol { list-style-type: upper-roman }
```

Изображения в качестве маркеров

Для использования в качестве маркеров собственных изображений необходимо воспользоваться свойством `list-style-image`, в качестве значения которого нужно указать URL к изображению, например:

```
ul { list-style-image: url(../images/circle.gif) }
```

Чтобы отменить унаследованные в качестве маркеров изображения, свойству `list-style-image` следует назначить `none`:

```
ul { list-style-image: none }
```

Замечание

При установке изображения в качестве маркера свойство `list-style-type` перестает действовать.

Положение маркера

Для указания положения маркера относительно текста служит свойство `list-style-position`, которое может принимать одно из значений:

- `inside` — маркер отображается как бы внутри текстового блока;
- `outside` — маркер отображается вне текстового блока.

Пример:

```
ul { list-style-position: outside }
```

Разница между значениями `inside` и `outside` показана на рис. П2.15.

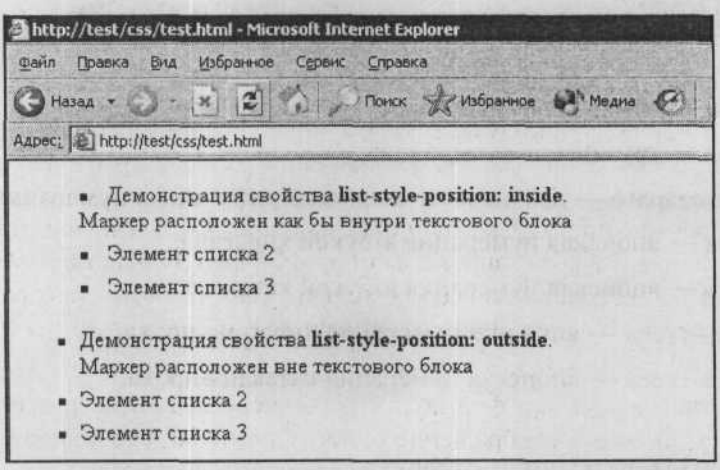


Рис. П2.15. Демонстрация значений свойств `list-style-position`

ПРИЛОЖЕНИЕ 3

Секреты быстрой загрузки сайтов или оптимизация графики под Web

Большинство браузеров поддерживает только три формата изображений: GIF, JPEG и PNG. Каждый из них имеет свои сильные и слабые стороны и вопрос выбора оптимального формата для использования в HTML-страницах должен определяться в каждом конкретном случае индивидуально. В данном приложении рассмотрены особенности распространенных графических форматов.

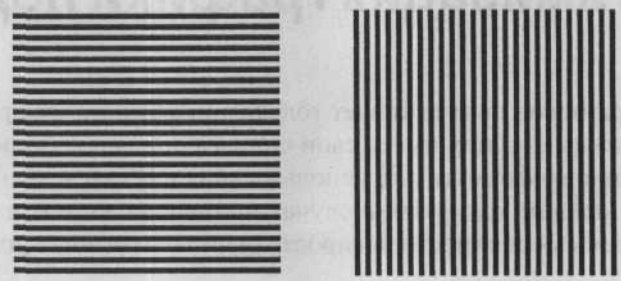
Формат GIF

Аббревиатура GIF расшифровывается как Graphics Interchange Format — формат для обмена графикой. Это растровый формат, цветовой диапазон которого ограничен 256 цветами, т. к. для хранения информации о цвете используются только 8 битов. Для уменьшения размера графических файлов возможно сократить количество используемых цветов до 2. Формат GIF поддерживает прозрачность. Сквозь пикселы, которым назначен прозрачный цвет, будут видны нижележащие объекты, либо фон. Однако в изображениях формата GIF может использоваться только один уровень прозрачности — прозрачность 100%, в отличие от формата PNG, который поддерживает 256 уровней прозрачности.

Еще одно преимущество формата GIF — чересстрочная развертка. При включении этой опции изображение будет постепенно увеличивать четкость по мере его загрузки. Сначала будут отображена каждая 8 строчка, затем каждая 4, каждая 2 и, наконец, будет выведено полное изображение. Таким образом, изображение появляется на экране почти сразу после начала загрузки страницы и, не дожидаясь полной загрузки, можно понять, что представлено на картинке.

Еще одно сильное качество формата GIF — поддержка анимации. Анимация поддерживается в версии формата GIF89a. В этом случае изображение представляется в виде сменяющих друг друга кадров.

Сжатие в формате GIF осуществляется по строкам, т. е. если строка имеет однородный цвет, то при сохранении в GIF к ней будет применено сжатие. Если однородный цвет используется в столбцах, то сжатия не происходит. На рис. П3.1 представлены 2 изображения размером 100 × 100 пикселов. В одном из них однородным цветом закрашены строки *(слева)*, а в другом — столбцы *(справа)*. Под изображениями указан их размер в байтах. В этом случае, уменьшение размера достигает почти 20%. Конечно, в реальной ситуации выигрыш будет не очень велик, но иногда знание этого свойства может сослужить хорошую службу.



247 байтов 301 байт

Рис. П3.1. Размер сжатых изображений формата GIF

Более подробную информацию о формате GIF можно получить по адресу **http://www.w3.org/Graphic/GIF/spec-gif89a.txt/**.

Применение GIF

Формат GIF следует использовать в том случае, если цветовой диапазон исходных изображений не превышает 256 цветов, либо количество цветов может быть уменьшено без существенного ухудшения качества. Это, как правило, изображения с большими площадями однородных одноцветных областей, изображения, полученные конвертированием из векторных форматов, изображения с текстом.

Для полноцветных изображений, в том числе для фотографий, формат GIF мало применим. В этом случае следует использовать другие форматы сжатия.

Формат JPEG

JPEG расшифровывается как Joint Photographic Experts Group — объединенная группа экспертов в области фотографии. Изображения в формате JPEG

поддерживают 24-битные цвета, и вследствие этого им хорошо пользоваться для сохранения полноцветных изображений. Формат предполагает сжатие с потерями. JPEG-сжатие основано на разложении изображений на составляющие, близкие к тем, которые используются в человеческом зрении при отбрасывании информации, не сказывающейся на зрительное восприятие образа. За счет этого достигается высокое сжатие изображений при незначительном ухудшении качества. Степень сжатия и качество изображений находятся в обратной зависимости: чем сильнее сжато изображение, тем ниже его качество. Обычно эти параметры определяются в процентах в диапазоне от 0 до 100.

Подробную информацию о JPEG можно получить по адресам:

- **http://www.w3.org/Graphics/JPEG/**;
- **http://www.jpeg.org**.

Применение JPEG

JPEG хорошо подходит для изображений с богатой цветовой гаммой, плавным переходом цветов, для фотографий и изображений с градиентными областями.

Не следует использовать JPEG для сжатия изображений, цветовая гамма которых ограничена несколькими цветами, изображений с мелким текстом, изображений, которые должны сохранить четкие границы или содержат мелкие детали.

Формат JPEG получил свое развитие в формате JPEG 2000, который вводит новые возможности и улучшает качество сжатия изображений, но пока этот новый формат не получил поддержки в браузерах и недоступен для использования в Интернете.

Формат PNG

Формат PNG расшифровывается как Portable Network Graphics — переносимая сетевая графика. Это относительно новый формат, призванный заменить собой формат GIF. Формат PNG существует в двух вариантах PNG-8 и PNG-24. PNG-8 практически полностью аналогичен формату GIF, за исключением улучшенного сжатия и отсутствия возможности анимации.

PNG-24 обладает рядом дополнительных преимуществ, таких как:

- наличие альфа-прозрачности — метода определения прозрачных областей, который в отличие от формата GIF обеспечивает 256 уровней прозрачности;

- существование гамма-коррекции — автоматической коррекции яркости изображения при воспроизведении на разных системах;
- применение улучшенного сжатия.

Более полную информацию о формате PNG можно получить по адресам:

- **http://www.libpng.org/pub/png/**;
- **http://www.w3.org/Graphics/PNG/**.

Использование PNG

Распространение формата PNG сдерживается старыми версиями браузеров, не поддерживающих данный формат, а также недостаточной и неполной поддержкой возможностей PNG в новых версиях. Так, например, обстоит дело с альфа-прозрачностью, поддержка которой отсутствует в браузерах.

При использовании формата PNG-24 для сжатия полноцветных изображений он проигрывает формату JPEG в размере созданного файла, т. к. использует сжатие без потерь.

PNG-24 рекомендуется выбирать для полноцветных изображений с четкими краями и мелкими деталями, изображений с мелким текстом, а также для изображений с прозрачными областями.

Оптимизация графики

Для качественной оптимизации графики следует использовать специальные графические программы, такие как Adobe Photoshop, ImageReady, Ulead Smart Saver, Macromedia Fireworks, PhotoImpact. Далее будут рассмотрены правила оптимизации на примере программы Adobe Photoshop. Диалоговое окно, предназначенное для оптимизации графики для Web, открывается по команде **Save for Web** из меню **File**. Из-за того, что в разных форматах применяются разные алгоритмы сжатия и хранений информации, для них существуют разные правила оптимизации.

Формат PNG-8 практически полностью соответствует формату GIF и имеет общие с ним принципы оптимизации.

Так как PNG-24 использует сжатие без потерь, то для него нет необходимости устанавливать качество сжатия — оно является максимально возможным. По той же причине размер файлов в формате PNG-24 будет больше, чем размер тех же самых изображений в формате JPEG.

Далее будут рассмотрены правила оптимизации, характерные для изображений форматов GIF и JPEG, а также общие рекомендации по оптимизации изображений и HTML-кода.

Оптимизация GIF

Рассмотрим приемы оптимизации, характерные для формата GIF.

Уменьшение количества цветов

Цветовой диапазон GIF-изображений ограничен 256 цветами. Однако иногда можно ограничиться еще меньшим количеством цветов. Минимальное количество цветов, до которого можно сократить цветовое пространство, равно 2.

Результат уменьшения количества цветов показан на рис. П3.2 на примере градиента, хотя сам градиент не предназначен для сохранения в формате GIF, но он очень наглядно показывает изменения качества в зависимости от количества цветов.

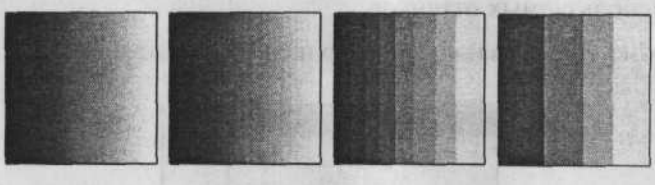


Рис. П3.2. Уменьшение количества цветов в цветовой палитре

Из рисунка наглядно видно, как ухудшается качество изображения при уменьшении количества цветов. Количество используемых цветов может быть равно 256, 128, 64, 2, 16, 8, 4 или 2. Подбор этого значения должен проводиться экспериментально для каждого изображения.

Оптимизация палитры

Все цвета, которые используются в формате GIF, хранятся в цветовой палитре. Упрощенно, это перечень цветов, используемых в изображении. Для формата GIF она не может содержать больше 256 цветов. Для сохранения изображений можно использовать предопределенные палитры или вычислять палитры динамически на основе самого изображения. Предопределенными палитрами являются: системная палитра Windows, градации серого, Web-палитра и т. п. Если, например, цветному изображению назначить Web-палитру, то для каждого пиксела будет подобран наиболее близкий цвет из назначенной палитры, а если назначить палитру градаций серого, то изображение станет черно-белым. Использование предопределенных палитр, как правило, приводит к не очень хорошему результату.

Определение

Web-палитра или, иначе, *палитра безопасных цветов*, состоит из цветов, которые будут без искажения воспроизведены во всех браузерах и на всех системах.

С целью улучшения качества изображений рекомендуется использовать палитры, вычисляемые по свойствам изображений. Программа Adobe Photoshop предоставляет выбор из трех палитр:

- **Perceptual** — палитра на основе текущих цветов изображения и учитывает при этом восприятия цветов человеком;
- **Selective** — палитра на основе текущих цветов изображения. В отличие от палитры **Perceptual**, она более направлена на сохранение цветов однородных участков и безопасных цветов;
- **Adaptive** — палитра на основе участка спектра, где представлено большинство используемых оттенков.

Диалоговое окно, показывающее выбор палитр, приведено на рис. П3.3.

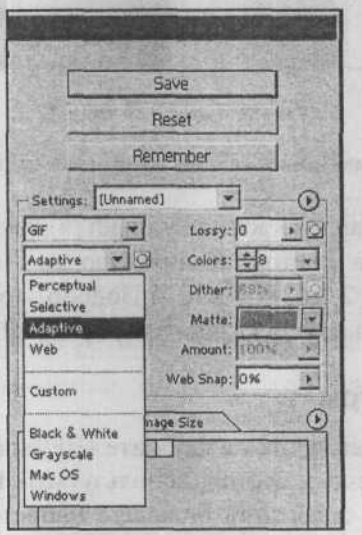


Рис. П3.3. Диалоговое окно выбора палитры в Adobe Photoshop

Диалоговое окно оптимизации и сохранения изображений для Web открывается по команде **Save for Web** из меню **File**.

Кроме выбора палитры, при сохранении GIF можно использовать **Dithering** — имитацию цветов. В этом случае, недостающие цвета получаются смешиванием существующих. Например, в цветовой палитре отсутствует оранжевый цвет, который необходимо получить, но присутствуют желтый и

красный. При включении смешивания цветов оранжевый цвет будет получен путем размещения рядом пикселов красного и желтого цветов. При включении смешивания размер файла увеличивается и в некоторых случаях может стать больше, чем был до оптимизации. Если количество используемых цветов мало, то при смешивании отдельные пикселы будут явно видны на изображении.

Художественная обработка изображений

Если исходное изображение плохо сжимается, и нет особых причин его не изменять, то можно применить художественную обработку или иначе — стилизацию. Стилизация необходима для незаметного изменения цветового диапазона изображения, т. е. его нужно обработать таким образом, чтобы создалось впечатление, что именно такое изображение и задумывалось автором изначально. Скажем, для полноцветного изображения можно применить фильтр, стилизующий изображение под работу карандашом. Пример такой стилизации приведен на рис. П3.4.

Рис. П3.4. Стилизация изображений

В результате стилизации, изображенной на рис. П3.4, цветовой диапазон сократился до 2 цветов, а размер изображения в формате GIF уменьшился с 77 до 9 Кбайт.

Таким образом, задача стилизации сводится к художественной обработке изображения, результатом которого является уменьшение количества используемых в нем цветов.

Оптимизация прозрачных изображений

При использовании прозрачности часто возникает проблема ореола вокруг изображения, у которого нет четких краев. Пример такого изображения приведен на рис. П3.5.

Рис. П3.5. Ореол вокруг прозрачных областей

Как видно из рис. П3.5, прозрачные области отображаются с белым ореолом. Если бы фон страницы был белым, то это было бы незаметно, т. е. при сохранении изображений с прозрачными областями цвет прозрачных пикселов следует подбирать под цвет фона. В Adobe Photoshop за назначения фона отвечает опция **Matte**, при сохранении изображений для Web в формате GIF (см. рис. П3.3).

Если фоновый рисунок неоднороден, то для удаления ореола следует ластиком стереть сглаженные области по краям (обычно 1—2 пиксела) и добиться их четкости.

Оптимизация JPEG

Задача оптимизации JPEG-изображений сводится в основном к выбору коэффициента качества, который обратно пропорционален коэффициенту сжатия. Для этого следует сохранить изображение с несколькими вариантами степени сжатия и визуально оценить их качество. На основе полученных данных нужно выбрать оптимальную степень сжатия и сохранить изображение окончательно. Подбор степени сжатия удобно делать в специализированных программах. В Adobe Photoshop для этого используется упомянутая выше команда **Save for Web**.

JPEG-изображения можно сохранять с использованием прогрессивной развертки. В графических программах эта опция обычно называется **Progressive**. При ее использовании изображение быстро появляется на экране в плохом качестве, и его детализация увеличивается по мере загрузки.

Оптимизация изображений при работе с HTML

Теперь рассмотрим приемы оптимизации, независящие от формата используемых изображений. Они будут основаны на особенностях отображения графики средствами HTML.

Искажение размеров

Этот метод оптимизации применим к изображениям с однородными областями и простыми формами.

Допустим, необходимо нарисовать прямоугольник однородного желтого цвета размером 100 × 200 пикселов. Для этого не обязательно создавать изобра-

жение указанного размера и помещать его на страницу. Достаточно создать изображение желтого цвета размером 1 × 1 пиксел и при размещении на странице назначить ему размеры 100 × 200 пикселов в теге `<img>`.

Рассмотрим использование этого приема на примере градиента. В шапку страницы необходимо поместить фон в виде градиента, изображенного на рис. П3.6 (левый прямоугольник). Для этого нет необходимости сохранять весь градиент, следует лишь сохранить полоску шириной в 1 пиксел и высотой, равной высоте градиента. Сохраняемая область показана выделением в правом прямоугольнике на рис. П3.6. Далее эту полоску следует поместить на страницу, задав ей нужную ширину с помощью свойства `width`. Если градиент должен размещаться по всей ширине страницы, то для полоски должна быть указана ширина 100% (`width=100%`).

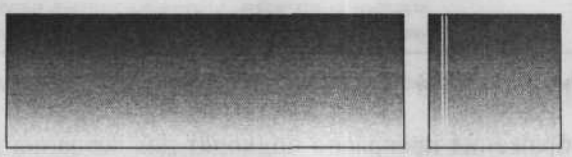


Рис. П3.6. Оптимизация градиентной заливки

Подобные изображения удобно делать фоновыми и описывать их в стилях CSS. В листинге П3.1 показан стиль для применения описываемого выше градиента к тегу `<div>`.

Листинг П3.1. Стиль градиента для тега `<div>`

```
div {
    background-color: url(gradient.jpg); /* фоновое изображение */
    background-repeat: repeat-x;  /* повторение только по оси x */
    height: 100px;                /* высота тега <div> */
    width: 100%                   /* максимально возможная ширина */
}
```

Использование фоновых изображений

Использование фоновых изображений вместо обычных изображений в теге `<img>` может положительно сказаться на восприятии HTML-страницы. Фоновые изображения обладают следующими достоинствами.

- Если посетитель будет просматривать страницу с отключенной графикой, то ему не будут показаны портящие дизайн специальные заглушки на месте отсутствующих изображений (рис. П3.7).

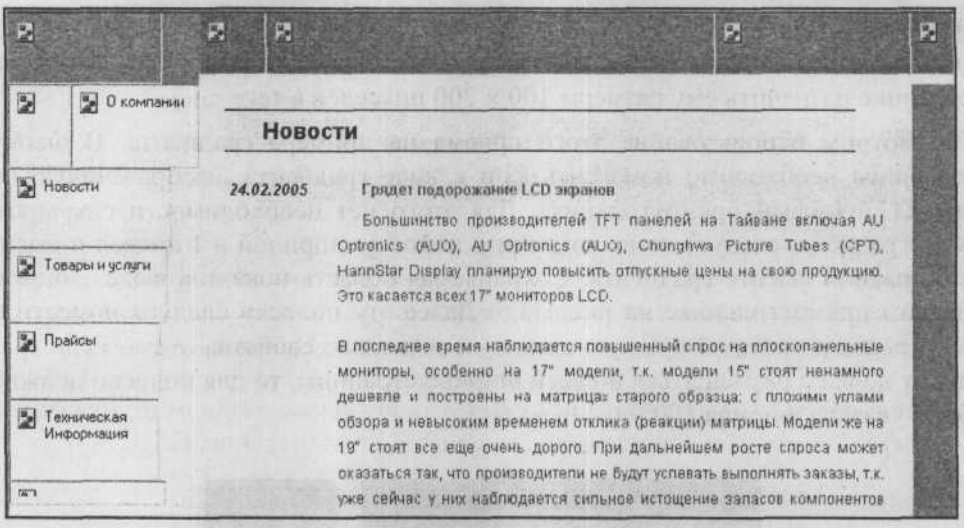


Рис. П3.7. Просмотр сайта с отключенными изображениями

- Фоновые изображения прекрасно подходят для создания дизайна, независящего от разрешения экрана. Допустим, имеется сверстанный табличный дизайн, независящий от экранного разрешения. При изменении разрешения экрана и соответственно уменьшении размера окна браузера столбцы таблицы автоматически изменяют свои размеры. Если в ячейку таблицы будет вставлено изображение в теге `<img>`, то это заставит жестко определить размер ячейки. Уменьшение размера ячейки будет невозможно, т. к. уменьшить ее не даст вставленный внутрь ячейки тег `<img>`. Если вместо вставки тега `<img>` ячейке таблицы назначить фоновое изображение, то ширина ячейки сможет беспрепятственно изменяться в зависимости от размера окна браузера. На рис. П3.8 показан пример с табличным дизайном для разрешения 1024 × 768 пикселов. При уменьшении разрешения до 800 × 600 пикселов страница не умещается на экране и появляется горизонтальная полоса прокрутки (правая страница на рис. П3.8).

  В листинге П3.2 приведен HTML-код страницы, изображенной на рис. П3.8.

Листинг П3.2. HTML-код страницы, изображенной на рис. П3.8

```
<table width=100% border="1" summary="">
  <tr>
    <td><img src="min.jpg" border="0" width="500" height="375"></td>
    <td><img src="logo.gif" border="0" width="400" height="200"></td>
  </tr>
</table>
```

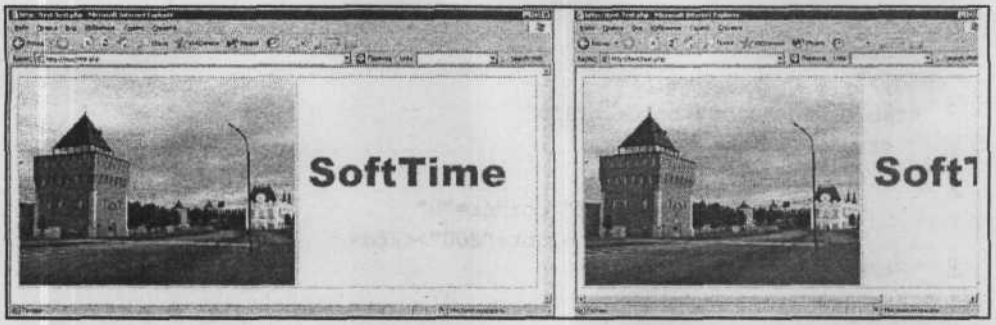


Рис. П3.8. Страница не уменьшается на экране при уменьшении разрешения

Для создания дизайна, независящего от разрешения, вставим изображение в первой ячейке в качестве фона. Результат показан на рис. П3.9.

Рис. П3.9. Фоновые изображения не мешают масштабированию страницы

HTML-код, соответствующий изображению на рис. П3.9, приведен в листинге П3.3.

Листинг П3.3. HTML-код страницы, изображенной на рис. П3.9

```
<html>
  <head>
    <style>
     /* стиль для ячейки с фоновым изображением */
    .myimg {background-image: url(min.jpg);
            background-repeat: no-repeat;
            width: 500px; height: 375px}
```

```
    </style>
  </head>
<body>
  <table width=100% border="1">
    <tr>
      <td class=myimg></td>
      <td><img src="logo.gif" border="0"
               width="400" height="200"></td>
    </tr>
  </table>
</body>
</html>
```

Фоновое изображение к ячейке было применено с помощью стилей CSS `<td class=myimg>`. Описание класса стиля `myimg` помещено в контейнер `<style>`.

Как видно из рис. П3.9, страница уместилась на экране при уменьшении размеров окна браузера (полосы прокрутки отсутствуют), т. к. фоновое изображение в левой ячейке не мешает масштабированию таблицы. Однако само фоновое изображение не уместилось в ней полностью, и была показана только его часть. Таким образом, применение этого способа оптимизации пригодно только для изображений, которые могут быть частично "урезаны". Либо нужно учитывать возможность подобной оптимизации еще при их создании.

Разделение изображения на фрагменты

Если изображение достаточно велико, то рационально разделить его на несколько частей и оптимизировать каждую часть отдельно. Этот прием действенен для изображений, созданных с помощью различных графических программ и мало применим к фотографическим изображениям. Одним из классических способов фрагментирования является разделение изображения на прямоугольные фрагменты и их компоновка на странице с помощью таблиц.

Следует отметить, что при использовании табличного способа компоновки желательно сохранять фрагменты только в одном выбранном графическом формате. Это могут быть форматы GIF, JPEG или PNG. Если разные фрагменты будут сохранены в разных форматах, то у посетителей с 24- или 16-битным цветом в настройках рабочего стола они могут быть отображены в разной цветовой гамме и вследствие этого им будет хорошо видна фрагментация исходного изображения. Защищены от такого дефекта только посетители с 32-битным цветом в настройках рабочего стола. Если необходимо сохранять разные фрагменты в различных графических форматах, то нужно обязательно протестировать полученную страницу при работе с глубиной цвета 24 и 16 битов.

Рассмотрим ситуацию, когда деление изображения на прямоугольные части и табличная фрагментация не дает значительной оптимизации.

Пусть необходимо разместить на странице большое изображение, являющееся элементом дизайна. Пример такого изображения представлен на рис. П3.10.

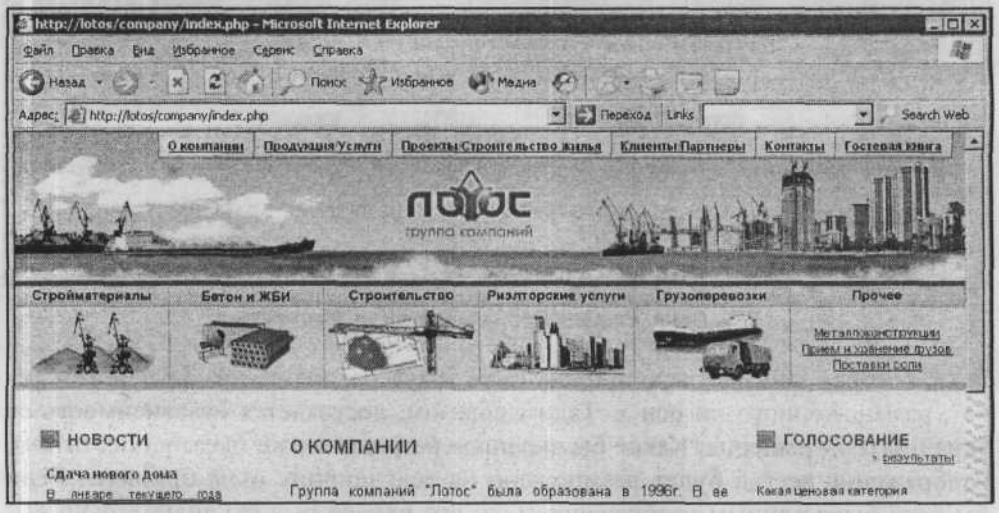


Рис. П3.10. Большое изображение на шапке сайта

Изображение на шапке сайта, изображенного на рис. П3.10, состоит из семи отдельных изображений, которые были скомпонованы в одно изображение с помощью таблиц стилей CSS. Отдельные части этого изображения показаны на рис. П3.11.

На рис. П3.11 показаны шесть фрагментов изображения, седьмым фрагментом является фон, который размножен по оси *x*, как это было показано в приеме оптимизации в *разд. "Искажение размеров" ранее в этой главе*.

Отдельные фрагменты были помещены в теги `<div>`, которым было назначено абсолютное позиционирование (`position: absolute`). Благодаря этому фрагменты изображения были размещены одно над другим с помощью свойства `z-index` тега `<div>`. Чем больше значение свойства `z-index`, тем "выше" расположено изображение. Так, например, фрагмент "судно" (`z-index=3`) располагается над фрагментом "карьер" (`z-index=2`), которое в свою очередь размещено над фрагментом "облако" (`z-index=1`) и даже частично закрывает его. Поэтому при изменении размеров окна браузера фрагменты будут корректно смещаться относительно друг друга. Благодаря этому дизайн становится независим от экранного разрешения.

Рис. П3.11. Разделение изображения на фрагменты

Река, которая занимает всю ширину окна браузера, состоит из фрагмента "река", размноженного по оси *x*. Таким образом, достигается независимость от экранного разрешения. Какое бы экранное разрешение не было у посетителя, изображение всегда будет размножено на всю ширину окна браузера. Если бы река была единым изображением, то его размер был бы значительно выше, чем размер фрагмента.

Фрагменты "логотип" и "порт и город" также помещены в теги `<div>` с абсолютным позиционированием. Реализация данного изображения в HTML-код показана в листинге П3.4.

Листинг П3.4. Реализация деления большого изображения на фрагменты

```
<div id="ship"></div>
<div id="town"></div>
<div id="karjer"></div>
<div id="cloud"></div>
<div id="logo"></div>
<table id="toptable" border="0" cellpadding="0" cellspacing="0"
       background="../images/sky.jpg">
  <tr valign="bottom">
    <td>
      <div id="river"></div>
    </td>
  </tr>
</table>
```

HTML-код в листинге П3.4 практически не содержит форматирования, за исключением таблицы, в которую помещен стиль слой с фрагментом "река" (`<div id="river"></div>`) и которой назначен фон для отображения неба (`background="../images/sky.jpg"`). Все оформление вынесено в таблицы стилей CSS, которые приведены в листинге П3.5.

Листинг П3.5. Стилевые таблицы для HTML-кода из листинга П3.4

```
/* фрагмент "судно" */
#ship{position: absolute; top: 81px; left: 9%; z-index: 3;
      width: 187; height: 29;
      background-image: url(../images/ship2.gif);
      background-repeat: no-repeat; }
/* фрагмент "порт и город" */
#town{position: absolute; top: 16px; right: 0%; z-index: 3;
      width: 320; height: 91;
      background-image: url(../images/town.gif);
      background-repeat: no-repeat; }
/* фрагмент "карьер" */
#karjer{position: absolute; top: 56px; left: -3%; z-index: 2;
        width: 168; height: 49;
        background-image: url(../images/karjer.gif);
        background-repeat: no-repeat; }
/* фрагмент "облако" */
#cloud{position: absolute; top: 20px; left: 0.5%; z-index: 1;
       width: 265; height: 75;
       background-image: url(../images/cloud.gif);
       background-repeat: no-repeat; }
/* фрагмент "логотип" */
#logo{position: absolute; top: 30px; left: 41%; z-index: 1;
      width: 111; height: 49;
      background-image: url(../images/logo.gif);
      background-repeat: no-repeat; }
/* фрагмент "река" */
#river{width: 100%; height: 27;
       background-image: url(../images/river.jpg);
       background-repeat: repeat-x;}
```

С помощью описанного приема "разделения изображения на фрагменты" был сверстан дизайн, независящий от экранного разрешения, а размер изображений уменьшен в несколько раз по сравнению с тем, если бы изображение не было разделено на фрагменты.

ПРИЛОЖЕНИЕ 4

Работа с JavaScript

Web-программирование относится к клиент-серверной технологии и подразумевает создание сервисов, работающих как на стороне сервера, так и на стороне клиента. Многие Web-технологии, такие как Java или ASP.NET объединяют в себе как серверную, так и клиентскую часть. PHP работает только на стороне сервера, поэтому для создания полноценных Web-приложений его необходимо использовать совместно с клиентским языком программирования. Традиционно для этих целей применяется JavaScript. PHP завершает свою работу, как только сгенерирован полный HTML-код страницы. После этого HTML-код через сеть передается в браузер, где начинает свою работу JavaScript. Таким образом, PHP не может вмешаться в работу JavaScript, и в качестве обработчиков JavaScript не могут выступать PHP-функции. Хотя код JavaScript не может взаимодействовать с PHP напрямую, он может обращаться к серверу и передавать ему необходимые данные.

В листинге П4.1 приведен пример обращения JavaScript к PHP-скрипту.

Листинг П4.1. Обращение к PHP-скрипту

```
<script language="JavaScript">
<!-
  window.location.replace("http://www.server.ru/index.php?var=3")
//-->
</script>
```

При выполнении этого кода происходит переадресация страницы по адресу **http://www.server.ru/index.php**, т. е. идет обращение к PHP-скрипту, которому передается параметр `var=3`.

PHP также не может выполнять функции JavaScript, но он способен сформировать HTML-код, содержащий функцию JavaScript, которая будет выполнена, как только страница загрузится в браузер.

В листинге П4.2 приведен пример формирования JavaScript с помощью PHP.

Листинг П4.2. Формирование JavaScript-кода на PHP *(вариант 1)*

```
<?php
echo "<script language=\"JavaScript\">
      <!-
        alert (\"Привет! Меня зовут JavaScript - я здесь живу.\");
      //-->
      </script>";
?>
```

Следует обратить внимание на то, что все двойные кавычки, написанные внутри конструкции `echo`, записаны через обратный слеш (`\"`). Экранирование необходимо, т. к. двойные кавычки уже используются для обрамления строки передаваемой конструкции `echo`. Вместо двойных кавычек внутри `echo` можно применять одинарные кавычки. В листинге П4.3 приведен тот же самый код, но с применением одинарных кавычек.

Листинг П4.3. Формирование JavaScript-кода на PHP *(вариант 2)*

```
<?
echo "<script language='JavaScript'>
      <!-
        alert ('Привет! Меня зовут JavaScript - я здесь живу.');
      //-->
      </script>";
?>
```

Сложности создания переносимого кода

Одной из задач Web-разработчика является создания *переносимого кода*, который будет одинаково работать с большинством используемых посетителями браузеров. Ситуация усугубляется тем, что браузеры отличаются друг от друга поддержкой языка JavaScript и объектной модели документа. Наиболее сильное отличие имеют браузеры линейки Internet Explorer и браузеры Netscape, в том числе браузеры Mozilla. Вследствие этого, для создания кода, который будет выполняться у большинства посетителей, Web-разработчикам необходимо знать, каким браузером пользуется посетитель и какие возможности он поддерживает.

Ранее функциональность браузера была тесно связана с версией поддерживаемого им языка JavaScript, и очень часто определение возможностей брау-

зера основывалось на выяснении версии языка. В настоящее время из-за различной поддержки объектной модели документа (Document Object Model, DOM) этот метод уже нельзя назвать удовлетворительным. Более правильным является определение версии модели объектов DOM, чтобы на основе этого выбрать нужную ветвь программного кода. Далее будут рассмотрены способы определения типа браузера и выяснения возможностей браузера на основе анализа поддерживаемых им объектов и свойств DOM.

Определение типа браузера

Для определения производителя браузера можно использовать свойство `navigator.appName`.

```
if (navigator.appName=="Netscape") var  isNets = true;
if (navigator.appName=="Microsoft Internet Explorer") var isIE = true;
```

Однако этот способ не всегда приводит к приемлемому результату, т. к. существуют браузеры, которые позволяют изменить это свойство. Например, браузер Opera позволяет пользователю самому определить содержимое этой строки: браузер может идентифицироваться как Netscape, Microsoft Internet Explorer или Opera. Более надежным является определение типа браузера с помощью анализа содержимого свойства `UserAgent`. Ниже приведены примеры значений этого свойства для разных браузеров:

- Internet Explorer 6.0

  ```
  Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
  ```

- Mozilla 0.9.3

  ```
  Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7) Gecko/20040803
  Firefox/0.9.3
  ```

- Opera

  В браузере Opera значение `UserAgent` также зависит от выбранных пользователем настроек:

  - `Opera/7.23 (Windows NT 5.1; U) [ru]` — для Opera;
  - `Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.23 [ru]` — для Internet Explorer 6.0;
  - `Mozilla/5.0 (Windows NT 5.1; U) Opera 7.23 [ru]` — для Mozilla 5.0.

  Как видно из вышеприведенных строк, Opera не полностью подделывает значение `UserAgent`, вставляя свое имя в конец строки. Таким образом, для определения браузера Opera нужно осуществлять поиск подстроки `Opera` в свойстве `UserAgent`.

В листинге П4.4 приведен пример кода, определяющего тип браузера.

Листинг П4.4. Определение типа браузера

```
<script language="JavaScript">
<!-
  if (navigator.userAgent.indexOf("Opera")!= -1) var isOpera = 1;
  else if (navigator.userAgent.indexOf("MSIE")!= -1) var isIE = 1;
  else if (navigator.userAgent.indexOf("Netscape")!= -1) var isNets = 1;
  else if (navigator.userAgent.indexOf("Firefox")!= -1) var isMoz = 1;
  if (isOpera) document.write("Opera");
  if (isIE)    document.write("Internet Explorer");
  if (isNets)  document.write("Netscape");
  if (isMoz)   document.write("Mozilla FireFox");
//-->
</script>
```

Не рекомендуется пользоваться вышеприведенным кодом для выяснения функциональности браузера, исходя из его типа. Более точные результаты дает метод определения объектов, приведенный в следующем разделе.

Проверка объектов

Для создания переносимого кода необходимо проверять наличие используемых объектов перед их применением и организовывать ветвления кода в зависимости от результатов проверки. Этот прием более надежен, чем определение типа браузера и построение предположений о поддерживаемых им объектах.

Для проверки следует обратиться к проверяемому объекту в блоке `if`. Если объект не доступен, то будет возвращено значение `false`. Такими проверками нужно предварять использование всех спорных объектов.

В листинге П4.5 приведен пример кода, проверяющий наличие массива `images`, поддержка которого отсутствовала в старых браузерах.

Листинг П4.5. Проверка доступности массива `images`

```
if (document.images)
{
  // Выполняемый код, если массив images доступен
}
else
{
  // Выполняемый код, если массив images не доступен
}
```

Проверка свойств и методов объектов

Помимо проверки наличия объектов, необходимо также выяснять доступность их методов и свойств. Рекомендуется использовать следующий метод проверки:

```
if (объект && объект.свойство != "undefined")
{
  // Выполняемый код, если свойство доступно
}
```

Проверку свойства в блоке `if` необходимо предварять проверкой существования объекта. Этим обеспечивается защита от обращения к свойству несуществующего объекта. Проверка наличия свойства должна проводиться сравнением со значением `"undefined"`, т. к. если свойство существует, но равно 0, либо пустой строке, то проверка вида `if(`*объект.свойство*`)` выдаст ошибочный результат.

Аналогично необходимо проверять спорные методы объектов, поддержка которых может отсутствовать в браузерах, используемых пользователями:

```
if (объект && объект.метод)
{
  // Выполняемый код, если метод доступен
}
```

Если метод существует, то ссылка на *объект.метод* возвратит значение `true`.

Пример:

```
if (document.getElementById)
{
  // Выполняемый код, если метод getElementById() доступен
}
```

Здесь проверяется наличие метода `getElementById()` у объекта `document`. Наличие самого объекта `document` можно не проверять, т. к. он существует во всех реализациях JavaScript.

Работа с окнами

Создание и открытие окна

Создать новое окно можно вызовом метода `open()` объекта `window`. Метод `open()` содержит три параметра: URL, имя окна, опции открываемого окна:

```
window.open("http://www.softtime.ru", "main",
            "top=100,left=100,width=500")
```

В качестве третьего параметра метода `open()` передаются опции, которые определяют внешний вид и расположение нового окна. Опции и их описание приведены в табл. П4.1. Последние два столбца содержат номера версий браузеров Internet Explorer (IE) и Netscape (NN), в которых впервые появилась поддержка описываемых свойств.

Таблица П4.1. *Опции окна, открываемого методом* `open`

Имя опции	Описание	IE	NN
`toolbar`	Отображение панели инструментов	3	2
`resizable`	Размер окна может быть изменен пользователем	3	2
`scrollbars`	Отображение полосы прокрутки, если документ не умещается в размерах окна	3	2
`status`	Отображение строки-подсказки	3	2
`directories`	Отображение кнопок папок	3	2
`top`	Смещение верхней границы окна браузера от верхней границы экрана	4	6
`left`	Отображение полосы прокрутки, если документ не умещается в размерах окна	4	6
`height`	Высота окна в пикселах	3	2
`width`	Ширина окна в пикселах	3	2
`location`	Отображение адресной строки	3	2
`menubar`	Отображение строки меню	3	2
`channelMode`	Отображение панели каналов	4	—
`fullscreen`	Запрет отображения заголовка и меню	4	—
`alwaysLowered`	Окно всегда располагается под другими открытыми окнами браузера	—	4
`alwaysRaised`	Окно всегда располагается над другими открытыми окнами браузера	—	4
`dependent`	Закрытие дочернего окна при закрытии главного окна	—	4
`hotkeys`	Отключение горячих клавиш доступа к меню	—	4
`innerHeight`	Высота содержимого	—	4
`innerWidth`	Ширина содержимого	—	4
`outerWidth`	Наружная ширина окна	—	4
`outerHeight`	Наружная высота окна	—	4

Таблица П4.1 (окончание)

Имя опции	Описание	IE	NN
screenX	Смещение левой границы окна браузера от левой границы экрана	—	4
screenY	Смещение верхней границы окна браузера от верхней границы экрана	—	4
titlebar	Отображение строки меню	—	4
z-lock	Фиксация окна под окнами браузера	—	4

Если при открытии нового окна указать только часть параметров, то пропущенные параметры будут установлены в значение `false`.

Закрытие окна

Для закрытия окна следует выполнить метод `close()`:

```
window.close()
```

Для закрытия текущего окна вместо объекта `window` можно обращаться к свойству `self`, которое является синонимом текущего окна:

```
self.close()
```

При попытке применить это свойство к основному окну будет выдано окно предупреждения (рис. П4.1), спрашивающее разрешение закрыть окно.

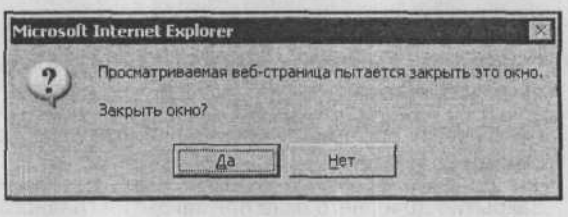


Рис. П4.1. Предупреждение о закрытии основного окна

Беспрепятственно методом `close()` можно закрывать только дочерние окна, открытые с помощью JavaScript методом `open()`, либо открытые по ссылке с указанием в параметре `target` имени окна или ключевого слова `_blank`. Пример:

```
<a hreh="index.php" target="_blank">Новое окна</a>
```

Приемы работы с формой

Приведем два совета, которые помогут сделать работу с формой более удобной для посетителя. Первый совет — это установка начального фокуса на элемент ввода формы. В этом случае посетителю не потребуется вручную устанавливать текстовый курсор в поле ввода — можно начать с ввода данных сразу же после открытия страницы с формой. Для установки фокуса необходимо выполнить соответствующие инструкции по событию загрузки страницы `onload` в теге `<body>`:

```
<body onload="document.имяФормы.имяПоляВвода.focus()">
```

Пример:

```
<body onload="document.myFrm.name.focus()">
```

Второй совет — очистка поля ввода при щелчке по нему мышью. Такое поведение может быть полезно, если в поле ввода располагается подсказка, например, "Введите данные в это поле", которая стирается после того, как пользователь щелкнул по полю мышью и намеревается ввести свои данные. Очистка данных в поле осуществляется по событию `onFocus`:

```
<input type="text" name="phrase"
       value="Введите данные в это поле" onFocus="this.value=''">
```

Ввод только цифр или букв

Для увеличения удобства работы пользователя и предотвращения ввода нежелательных символов в форму Web-разработчики часто прибегают к ограничению символов, которые пользователь может ввести в поле ввода. Часто требуется ограничить ввод только цифрами, буквами или каким-либо заранее определенным набором символов. Для перехвата ввода символа в текстовом поле используется событие `onkeypress`. Обработчик события принимает единственный параметр, в котором передается код вводимого символа. Анализ данного кода позволяет разрешить или запретить ввод символа. Если необходимо разрешить ввод символа, обработчик должен возвратить значение `true`, в противном случае — `false`.

В листинге П4.6 приведена функция `enterNumber()`, которая контролирует работу пользователя, разрешая ввод только цифр, и функция `enterLetter()`, позволяющая вводить только символы букв.

Листинг П4.6. Ограничение на ввод данных в форму

```
<html>
  <head>
```

```
    <script language="JavaScript">
    <!-
      // Функция, разрешающая ввод только цифр
      function enterNumber(event)
      {
        if (event.charCode) var charCode = event.chatCode;
          else if (event.keyCode) var charCode = event.keyCode;
          else if (event.which) var charCode = event.which;
          else var charCode = 0;
        if (charCode > 31 && (charCode < 48 || charCode > 57))
        {
          alert("Разрешен ввод только цифр.");
          return false;
        }
      }

      // Функция, разрешающая ввод только латиницы и кириллицы
      function enterLetter(event)
      {
        if (event.charCode) var charCode = event.chatCode;
          else if (event.keyCode) var charCode = event.keyCode;
          else if (event.which) var charCode = event.which;
          else var charCode = 0;
        if (charCode > 31 &&
            (charCode < 65 || charCode > 90) &&
            (charCode < 97 || charCode > 122) &&
            (charCode < 1040 || charCode > 1103))
        {
          alert("Разрешен ввод только символов латиницы и кириллицы.");
          return false;
        }
      }
    //-->
    </script>
  </head>
  <body>
    <form action="index.php" method="post">
      <input type="text" name="account" size="40"
             onkeypress="return enterNumber(event)"><br>
      <input type="text" name="name" size="40"
             onkeypress="return enterLetter(event)">
    </form>
  </body>
</html>
```

Блокирование элементов формы

Элементы формы сделать неактивными (заблокировать) можно с помощью свойства `disabled`. Например, в листинге П4.7 по выбору переключателя **Физ. лицо** блокируется поле ввода `firma`, предназначенное для ввода названия фирмы, и снимается блокировка с поля ввода `fio`, которое служит для ввода имени.

Листинг П4.7. Блокирование элементов формы

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function freeze(form, value)
      {
        if (value==1)
        {
          form.fio.disabled = false;
          form.firma.disabled = true;
        }
        else
        {
          form.fio.disabled = true;
          form.firma.disabled = false;
        }
      }
    //-->
    </script>
  </head>
  <body>
    <form action="index.php" method="post">
      <input type="radio" name="type" value="1"
             onclick="freeze(this.form, 1)"> Физ. лицо<br>
      <input type="radio" name="type" value="2"
             onclick="freeze(this.form, 2)"> Юр. лицо<br>
      Имя: <input type="text" name="fio" size="40"><br>
      Название фирмы: <input type="text" name="firma" size="40">
    </form>
  </body>
</html>
```

Можно не только блокировать поля для ввода, но и совсем скрывать их на странице. Однако в некоторых случаях пользовательский интерфейс выигрывает, если пользователь видит все элементы ввода формы.

Проверка введенных данных перед отправкой

Процедуры проверки введенной информации нередко используются в Web-приложениях. Наиболее часто встречаются задачи проверки заполнения полей, правильности ввода e-mail и даты.

Рассмотрим пример из листинга П4.8. Событие отправки формы по нажатию кнопки **Обработать** перехватывается сценарием JavaScript, который производит проверку введенных данных на корректность. Эта функциональность реализуется помещением в тег `<form>` обработчика события `onsubmit` (`onsubmit="return checkForm(this)"`). Если функция `checkForm()` возвращает `true`, то форма отсылается скрипту, указанному в параметре `action`. Если функция `checkForm()` находит ошибки во введенных данных, то возвращается `false` и отправки формы не происходит. В качестве параметра функции `checkForm()` передается ссылка на саму форму с помощью ключевого слова `this` — `checkForm(this)`.

Листинг П4.8. Проверка введенных данных

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function checkForm(form)    // Функция проверки введенных данных
      {
        var err="";
        // Проверка введенных данных в поле E-mail
        var errEmail = checkEmail(form.email.value);
        if (errEmail) err += errEmail+"\n";
        // Проверка введенных данных в поле Дата
        var errData = checkData(form.data.value);
        if (errData) err += errData+"\n";
        // Проверка введенных данных в текстовое поле
        var errTxr = checkTxt(form.txt.value);
        if (errTxr) err += errTxr+"\n";
        if (err=="") return true; // Возврат true, если ошибок не найдено
        else
        {
          alert(err);      // Вывод сообщения об ошибках
          return false;    // Возврат false
        }
      }
```

```
    function checkTxt(value) // Проверка ввода данных в текстовое поле
    {
      // Регулярное выражение, проверяющее,
      // введены ли данные в текстовое поле (пробелы не учитываются).
      if (!value.match(/[^\s]/g)) return "Не введен текст "
    }
  //-->
  </script>
 </head>
 <body>
  <!-- Обрабатываемая форма -->
  <form action="sendmail.php" method="post"
        onsubmit="return checkForm(this)">
   E-mail: <input type="text" name="email" size="40"><br>
   Дата: <input type="text" name="data" size="40"><br>
   Текст <input type="text" name="txt" size="40"><br>
   <input type="submit" value="Обработать">
  </form>
 </body>
</html>
```

Функция проверки правильности e-mail

Функция проверки электронного адреса выясняет корректность синтаксиса введенного e-mail с помощью регулярного выражения. Адрес электронной почты должен соответствовать формату ***user@domail.ru***. В имени пользователя и домена могут быть использованы цифры, буквы и дефис. В случае несоответствия синтаксису возвращается сообщение об ошибке. Если синтаксис корректный, то возвращается `false` (листинг П4.9).

Листинг П4.9. Функция проверки правильности e-mail

```
function checkEmail(value)
{
  // Проверка, не пустое ли значение
  if (!value.match(/[^\s]/g)) return "Не введен e-mail"
  var re = /^[\w-]+(\.[\w-]+)*@([\w-]+)\.+[a-zA-Z]{2,3}$/;
  if (!value.match(re)
     return "Неправильный формат адреса электронной почты"
  return false;
}
```

Функция проверки правильности даты

Функция анализа правильности даты основана на последовательном выполнении двух проверок (листинг П4.10). Первая проверка осуществляется по регулярному выражению на соответствие формату `dd.mm.yyyy`. Вводимые день и месяц могут содержать 1 или 2 цифры. Год должен содержать 4 цифры. Если проверка на соответствие формату пройдена, то далее анализируется правильность введенных данных для того, чтобы исключить ввод некорректных данных, например 45.23.2003. Проверка базируется на создании проверочного объекта `Date` (на основе введенных посетителем данных), последующего разложения этого проверочного объекта на отдельные компоненты (день, месяц, год) и сравнении их с допустимыми данными. Проверка основана на том, что при создании объекта `Date` JavaScript-сценарий автоматически откорректирует значения, выходящие за пределы допустимого диапазона. После подобного преобразования введенные пользователем данные и компоненты даты, полученные разложением объекта `Date`, будут не равны.

Функция возвращает сообщение об ошибке при некорректном синтаксисе и `false`, если ошибок не найдено.

Листинг П4.10. Функция проверки правильности даты

```
function checkData(value)
{
  // Проверка, не пустое ли значение
  if (!value.match(/[^\s]/g)) return "Не введена дата"
  // Регулярное выражение для проверки правильности ввода
  // формата даты dd.mm.yyyy
  var re = /^[\d]{1,2}(\.)[\d]{1,2}(\.)[\d]{4}$/i
  if (!value.match(re))
    return "Неправильный формат даты. Введите дату в формате dd.mm.yyyy"
  pos1 = value.indexOf(".");
  pos2 = value.lastIndexOf(".");
  dd = value.substring(0,pos1);          // день
  mm = value.substring(pos1+1,pos2);     // месяц
  yyyy = value.substring(pos2+1,10);     // год
  tmpData = new Date(yyyy,mm-1,dd);      // тестовая дата
  var err = "";
  if (tmpData.getDate()!=dd) return "Неправильно записан день."
  if (tmpData.getMonth()!=mm-1) return "Неправильно записан месяц."
  if (tmpData.getFullYear()!=yyyy) return "Неправильно записан год."+yyyy
  return false;
}
```

Отправка формы без нажатия кнопки *Отправить*

Отправка формы без нажатия кнопки **Отправить** часто применяется при выборе элемента из выпадающего списка, выборе одного из переключателей, либо по какому-то иному событию, которое может означать, что пользователь закончил выбор и готов к отправке формы. В листинге П4.11 приведен пример отправки формы по событию выбора элемента выпадающего списка. Кнопка отправки формы отсутствует. Отправка формы производится при изменении выбранного элемента `onchange="sendForm(this.form)"` и вызова функции-обработчика, которая выполняет отправку формы вызовом метода `submit()` у формы.

Листинг П4.11. Отправка формы без нажатия кнопки Отправить

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function sendForm(form)
      {
        form.submit();
      }
    //-->
    </script>
  </head>
  <body>
    <form action="action.php" method="post">
      <select name="name" onchange="sendForm(this.form)">
        <option value="1"> Форум по PHP</option>
        <option value="2"> Форум по Apache</option>
      </select>
    </form>
  </body>
</html>
```

Средствами JavaScript можно не только отправлять форму по адресу, указанному в параметре `action`, но и подменять этот адрес в зависимости, например, от выбранного элемента списка. Пример подобной функциональности приведен в листинге П4.12. По событию выбора элемента списка вызывается функция

```
sendForm(this.form, this.options[this.selectedIndex].value)"
```

Функции sendForm() в качестве первого параметра передается имя формы, в качестве второго — значение выбранного элемента списка. В зависимости от этого значения меняется значение параметра action. После смены этого параметра форма отсылается выполнением метода submit().

Листинг П4.12. Подмена параметра action у HTML-формы

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function sendForm(form, value)
      {
        if (value==1) form.action="http://www.softtime.ru/php/index.php"
        if (value==2)
            form.action="http://www.softtime.ru/apache/index.php"
        form.submit();
      }
    //-->
    </script>
  </head>
  <body>
    <form action="http://www.softtime.ru/php/index.php" method="post">
      <select name="name" onchange="sendForm(this.form,
                                this.options[this.selectedIndex].value)">
        <option value="1"> Форум по PHP</option>
        <option value="2"> Форум по Apache</option>
      </select>
    </form>
  </body>
</html>
```

Извлечение переменных из адресной строки

Для извлечения значений переменных из адресной строки применяется свойство location.search, которое содержит параметры запроса к странице вида ?id=3&parent_id=2&option=link.

Полученная строка после отбрасывания первого символа раскладывается в массив по символу-разделителю &. Полученный массив состоит из элементов:

```
param[0] = "id=3"
param[1] = "parent_id=2"
param[2] = "option=link"
```

Для разделения имени переменной и ее значения производится разбиение элементов массива `param` на временный массив `tmp`, первый элемент которого содержит имя переменной, а второй — ее значение. Разбиение на временный массив происходит по символу =. Далее инициализируется массив `values`. Идентификатор элемента массива `values` приравнивается имени переменной из строки запроса, а значение — значению переменной из строки запроса.

Полученный массив `values` будет иметь вид:

```
values["id"] = 3
values ["parent_id "] = 2
values ["option "] = link
```

Пример — в листинге П4.13.

Листинг П4.13. Получение переменных из адресной строки

```
<script language="JavaScript">
<!--
  // Получение строки запроса без символа ?
  var query=location.search.substring(1);
  // Разложение строки в массив по символу-разделителю &
  var param=query.split('&');
  var values = new Array();
  // Каждая пара (имя=значение)
  // раскладывается на имя переменной и ее значение
  for(i=0; i<=param.length-1; i++)
  {
    var tmp = param[i].split('=');
    values[tmp[0]] = tmp[1];
  }
//-->
</script>
```

Для получения значения переменной из адресной строки необходимо обратиться к элементу массива `values` по имени переменной.

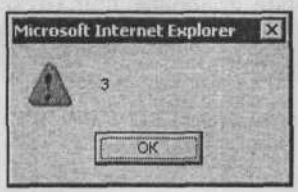


Рис. П4.2. Проверка получения значения из адресной строки

Например, строка

```
alert(values["id"])
```

выведет в окне сообщения значение 3 (рис. П4.2).

Подсветка ячейки при наведении указателя мыши

Таблицы смотрятся очень привлекательно, если они реагируют на перемещение указателя мыши над ними. В листинге П4.14 приведен код, который при наведении указателя мыши на ячейку подсвечивает фон черным, а текст белым цветом. Если внутри ячейки расположена ссылка, то она также меняет цвет на белый. Скрипт основан на динамической смене стилей элементов. Ячейка становится подсвеченной по событию `onmouseover="over(this)"` и принимает обычный вид по событию `onmouseout="out(this)"`. Назначаемые стили предварительно описаны в блоке `<head>`.

Листинг П4.14. Подсветка ячейки при наведении указателя мыши

```
<html>
  <head>
    <style type="text/css">
      /* Стиль select предназначен для выделенной ячейки */
      /* фон ячейки и цвет текста*/
      td.select{background-color: #000000; color: #FFFFFF}
      td.select a{color: #FFFFFF}  /* цвет ссылки внутри ячейки */
      /* Стиль normal предназначен для обычной ячейки */
      td.normal{background-color: #FFFFFF}
      td.normal a{color: #000000}
    </style>
    <script language="JavaScript">
    <!-
      function over(id)     // Подсветка ячейки
      {
        id.className="select";
      }
      function out(id)      // Удаление подсветки ячейки
      {
        id.className="normal";
      }
    //-->
    </script>
  </head>
  <body>
```

```
    <table border="1" summary="">
      <tr>
        <td onmouseover="over(this)" onmouseout="out(this)">
           <a href="#">Ячейка 1</a></td>
        <td onmouseover="over(this)" onmouseout="out(this)">
           <a href="#">Ячейка 2</a></td>
      </tr>
    </table>
  </body>
</html>
```

Смена изображения при наведении указателя мыши

Аналогично динамической смене стилей можно осуществлять смену изображения при наведении на него курсора мыши (листинг П4.15). Для того чтобы в первый раз изображение поменялось без задержки, необходимо применять предварительную загрузку изображений. Для этого создаются два массива объектов `image`, свойства `src` которых указывают на загружаемые изображения. Один массив содержит изображение, появляющееся при наведении указателя мыши, а другой — исходные изображения. Смена изображений происходит по событиям `onmouseover="over(this.id)"` и `onmouseout="out(this.id)"`, которые вызывают соответствующие функции смены изображений. В качестве параметра в этих функциях передается идентификатор изображения.

Листинг П4.15. Меняющиеся изображения при наведении указателя мыши

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      if (document.images)
      {
        var imgOver = new Object();
        imgOver["image_1"] = new Image();
        imgOver["image_1"].src = "images/bank_1.jpg";
        var imgOut = new Object();
        imgOut["image_1"] = new Image();
        imgOut["image_1"].src = "images/bank_2.jpg";
      }
```

```
      function over(id)
      {
        if(document.getElementById)
           document.getElementById(id).src = imgOver[id].src;
      }
      function out(id)
      {
        if(document.getElementById)
           document.getElementById(id).src = imgOut[id].src;
      }
    //-->
    </script>
  </head>
  <body>
    <img id="image_1" src="images/bank_2.jpg" border="0"
         onmouseover="over(this.id)"
         onmouseout="out(this.id)">
  </body>
</html>
```

Предварительная загрузка может не работать, если на странице запрещено кэширование, например, используются заголовки, запрещающие кэширование страницы:

```
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
```

Кэширование страницы также может быть отключено в браузере пользователя.

Передача данных между фреймами

Фреймы не пользуются у Web-разработчиков большой популярностью. Основная причина этому — неудовлетворительная работа с фреймами поисковых систем. Однако фреймы имеют свои преимущества, которыми не стоит пренебрегать. Если задача раскрутки сайта в Интернете не является первоочередной, а основное внимание должно быть направлено на построение удобного пользовательского интерфейса, то стоит присмотреться к фреймам — они предоставляют много возможностей.

Рассмотрим задачу организации передачи информации между фреймами без обращения к серверу.

При навигации по сайту с фреймами меняется только содержимое самих фреймов, а страница, содержащая структуру фреймов, остается постоянной. Этим можно воспользоваться для хранения глобальных переменных, с кото-

рыми могут работать все фреймы. Предположим, что в одном из фреймов располагается форма с элементом ввода, значение которого нужно сохранить (листинг П4.16).

Листинг П4.16. Сохранение переменной из фрейма

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function save(frm)
      {
        top.name = frm.name.value;
        return true;
      }
    //-->
    </script>
  </head>
  <body>
    <form action="index.php" method="post" onsubmit="return save(this)">
      <input type="text" name="name" size="40" maxlength="256">
      <input type="submit" value="Послать">
    </form>
  </body>
</html>
```

По событию отправки `onsubmit` вызывается функция `save()`, которая сохраняет введенное в поле `name` значение в глобальную переменную `name` и затем возвращает значение `true`, что приводит к отправке формы скрипту, указанному в параметре `action`. Теперь в глобальной переменной `name` содержится введенное значение, которое можно прочитать из других фреймов. Пример чтения глобальной переменной `name` приведен в листинге П4.17.

Листинг П4.17. Чтение глобальной переменной из фрейма

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function read()
      {
        if (typeof top.name != "undefined")
```

```
        {
          document.myfrm.name.value=top.name;
        }
      }
    //-->
    </script>
  </head>
  <body onload="read()">
    <form name=myfrm action="action.php" method="post">
      <input type="text" name="name" size="40">
      <input type="submit" value="Послать">
    </form>
  </body>
</html>
```

Этот код может быть использован на любой странице, подгружаемый в качестве фрейма. По окончании загрузки страницы вызывается функция `read()`, которая читает значение глобальной переменной `name` и использует его для инициализации поля `name` формы `myfrm`, расположенной на странице.

Передача данных в два фрейма одновременно

Передача данных более чем в один фрейм очень часто встречается при разработке пользовательского интерфейса. Например, один фрейм содержит меню и в него нужно передать номер активного пункта меню, чтобы выделить, а второй фрейм содержит страницу, соответствующую выбранному пункту меню. Рассмотрим структуру из трех фреймов. При щелчке по ссылке в первом фрейме данные должны быть переданы в два соседних фрейма. Эта задача может быть решена с помощью JavaScript. В листинге П4.18 приведена структура фреймов в файле main.html.

Листинг П4.18. Файл main.html: структура фреймов

```
<frameset rows="80,*,80" frameborder="NO" border="0" framespacing="0">
  <frame src="topframe.php" name="topframe" scrolling="NO" noresize >
  <frame src="middleframe.php" name="middleframe">
  <frame src="bottomframe.php" name="bottomframe" noresize>
</frameset>
<noframes></noframes>
```

Необходимо по щелчку на ссылке в верхнем фрейме `topframe` передать значение переменной `var` во фреймы с именами `middleframe` и `bottomframe`. Со-

держание файла topframe.php, который отображается во фрейме topframe, приведено в листинге П4.19.

Листинг П4.19. Файл topframe.php

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function send(var)
      {
         top.frames.middleframe.location.href="middleframe.php?id="+var;
         top.frames.bottomframe.location.href="bottomframe.php?id="+var;
      }
    //-->
    </script>
  </head>
  <body>
    <h2>Верхний фрейм</h2>
    <a href="#" onclick="send(2); return false">Ссылка</a>
  </body>
</html>
```

По щелчку мыши на ссылке вызывается функция send(), в параметре которой находится значение для передачи фрейму. Строка return false, помещенная после вызова функции send(), запрещает срабатывание ссылки по адресу в параметре href.

Функция send() заменяет параметр location у среднего (middleframe) и нижнего (bottomframe) фреймов, тем самым перезагружает фреймы и передает им параметры id.

Проконтролировать процесс передачи параметра можно, разместив в файлах middleframe.php и bottomframe.php PHP-код, который будет выводить содержимое переменной var на экран:

```
<?php
  echo $_GET["id"];
?>
```

Горизонтальное выпадающее меню

Меню — один из основных элементов пользовательского интерфейса Web-сайта. Удобство навигации по сайту через меню в большой степени определяет удобство пользования всем сайтом. За время существования Web было

разработано множество самых разнообразных типов меню: горизонтальные, вертикальные, выпадающие, раскрывающиеся и т. д. Рассмотрим программную реализацию горизонтального выпадающего меню (рис. П4.3).

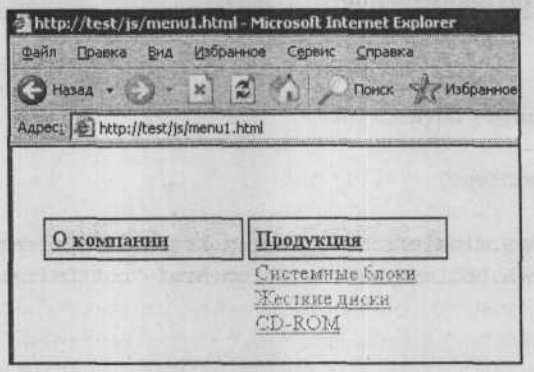


Рис. П4.3. Выпадающее меню

Основное достоинство выпадающего меню в том, что оно занимает мало экранного места, т. к. большую часть времени раскрывающиеся подменю скрыты и не мешают просмотру сайта. При наведении мыши на один из главных пунктов меню верхнего уровня раскрывается выпадающее подменю. Когда указатель мыши покидает область меню — подменю сворачивается.

Все меню помещено в тег `<div>`, которому назначается абсолютное позиционирование с помощью таблиц стилей `<div id=menu>`. Благодаря абсолютному позиционированию, меню можно разместить в любом месте страницы. Для этого необходимо отредактировать параметры `top` и `left` в стиле `#menu`. Класс стилей `.rootmenu` предназначен для оформления пунктов меню верхнего уровня, а класс стилей `.submenu` описывает оформление блоков выпадающего меню. Установка в классе `.submenu` правила `display: none` скрывает блоки выпадающих меню при начальной загрузке страницы.

В листинге П4.20 приведен HTML-код страницы с выпадающим меню. Функции JavaScript, используемые при организации выпадающего меню, приведены в листингах П4.21 и П4.22.

Листинг П4.20. HTML-код и таблицы стилей выпадающего меню

```
<html>
  <head>
  <style type="text/css">
    /* Контейнер меню */
    #menu{position: absolute; top: 50px; left: 20px}
```

```
    // Оформление пунктов меню верхнего уровня
    .rootmenu {border-style: solid; border-width: 1px; padding: 5px;
               background-color: #EEEEEE}
    // Стиль контейнеров выпадающего меню
    .submenu {display: none; padding: 0px 0px 0px 5px}
  </style>
  </head>
  <body>
    <div id=menu>
      <table border=0><tr valign=top>
        <td width="150px">
          <div class=rootmenu>
            <a href="#" onmouseover="showmenu('m1')"
               onmouseout="hidemenu()"><b>О компании</b></a></div>
            <div id="m1" class="submenu" onmouseover="showmenu('m1')"
                                         onmouseout="hidemenu()">
            <a href="http://">О компании</a><br>
            <a href="http://">Руководство</a><br>
            <a href="http://">Контакты</a><br>
            <a href="http://">Схема проезда</a> </td>
          </div>
        </td>
        <td width="150px">
          <div class=rootmenu>
            <a href="#" onmouseover="showmenu('m2')"
               onmouseout="hidemenu()"><b>Продукция</b></a></div>
            <div id="m2" class="submenu" onmouseover="showmenu('m2')"
                                         onmouseout="hidemenu()">
            <a href="http://">Системные блоки</a><br>
            <a href="http://">Жесткие диски</a><br>
            <a href="http://">CD-ROM</a> </td>
          </div>
        </td></tr></table>
    </div>
  </body>
</html>
```

Отображение блоков выпадающего меню происходит по событию наведения мыши на пункты меню, при котором вызывается функция `showmenu()`:

```
onmouseover="showmenu(id)"
```

В качестве параметра функции `showmenu()` передается идентификатор выпадающего меню, которое необходимо отобразить.

После того как указатель мыши покидает область выпадающего меню, по событию `onmouseout` вызывается функция `hidemenu()`, которая скрывает блок открывшегося меню. Функции `showmenu()` и `hidemenu()`, реализующие работу выпадающего меню, приведены в листинге П4.21. Скрытие выпадающего меню в функции `hidemenu()` организовано по таймеру:

```
setTimeout("выражение", миллисекунды)
```

который выполняет код, указанный в выражении через время в миллисекундах. Скрытие по таймеру необходимо для предотвращения преждевременного закрытия выпавшего блока подменю, когда указатель мыши перемещается от пункта меню верхнего уровня к ссылкам в подменю. Например, от ссылки **Продукция** к ссылке **CD-ROM** (см. рис. П4.3).

Листинг П4.21. Функции JavaScript для организации выпадающего меню

```
// Идентификатор предыдущего открытого меню
var oldsubmenu;
// Открытое меню закрывается по таймеру
var timeOnMenu;
// Функция скрытия блока выпадающего меню
function showmenu(obj)
{
  if (timeOnMenu) {
  // Если установлено скрытие меню по таймеру,
  // то удалить таймер и отменить скрытие
    clearTimeout(timeOnMenu);
  }
  // Скрыть существующий открытый блок выпадающего меню
  if (oldsubmenu) hide(oldsubmenu);
  // Показать блок выпадающего меню с идентификатором obj
  show(obj);
  // Идентификатор текущего открытого меню
  oldsubmenu=obj;
}
// Функция скрытия блока выпадающего меню
function hidemenu()
{
  // Если есть открытий блок выпадающего меню,
  // то скрыть его через 0.5 секунды по таймеру
  if (oldsubmenu) timeOnMenu = setTimeout("hide(oldsubmenu)",500)
}
```

Для скрытия и отображения блоков меню используются функции `show()` и `hide()`, приведенные в листинге П4.22. Они реализуют отображение и скры-

тие элементов по их идентификатору и обеспечивают корректную работу кода в большинстве браузеров. Коды функций `show()`, `hide()` и `getObject()` являются универсальными и могут использоваться в качестве библиотечных функций для других задач.

Листинг П4.22. Функции для скрытия и отображения элементов по их идентификатору

```
function getObject(obj)
{
  var theObj
  if (document.layers) {
    if (typeof obj == "string") return document.layers[obj]
    else return obj
  }
  if (document.all) {
    if (typeof obj == "string")
    {
      if(document.all(obj)!=null) return document.all(obj).style;
      else return null;
    }
    else return obj.style
  }
  if (document.getElementById)
  {
    if (typeof obj == "string")
       return document.getElementById(obj).style
    else return obj.style
  }
  return null
}

// Функция для отображения объектов
function show(obj) {
  var theObj = getObject(obj);
  if (typeof theObj.visibility != "undefined")
      theObj.visibility = "visible"
  if (typeof theObj.display != "undefined") theObj.display = "block"
}

// Функция для скрытия объектов
function hide(obj)
{
  var theObj = getObject(obj)
```

```
  if (typeof theObj.visibility != "undefined")
      theObj.visibility = "hidden"
  if (typeof theObj.display != "undefined") theObj.display = "none"
}
```

Скрытие и отображение объектов по событию

Часто возникают ситуации, когда необходимо скрыть или показать какой-либо элемент на странице, не прибегая к ее перезагрузке. Это может быть отображение интерактивной подсказки, предоставление разных вариантов формы в зависимости от выбранных предыдущих значений, создание раскрывающегося меню и т. д. Для реализации этого применяется свойство элементов `display`. При его установке в значение `none` элемент не отображается и его место занимают другие элементы. При установке свойства `display` в `block` элемент снова появляется на странице, "раздвигая" собой элементы, занявшие его место.

Рассмотрим пример, скрывающий и отображающий элементы `<div>` по событию выбора переключателей. Внутри элементов `<div>` могут быть размещены любые элементы, например формы. Скрытие и отображение элементов `<div>` происходит по событию выбора одного из переключателей. Событие `onclick="change(1)"` вызывает отображение блока с идентификатором `id`, равным `block1`, и скрытие блока с идентификатором `id`, равным `block2`. Для скрытия и отображения вызываются функции `show()` и `hide()` соответственно, приведенные в листинге П4.22.

Листинг П4.23. Скрытие и отображение объектов по событию

```
<html>
  <head>
    <script language="JavaScript">
    <!--
      function change(id)  // Смена отображаемых и визуализируемых блоков
      {
        if (id==1)
        {
          show("block1");
          hide("block2");
        }
        else
        {
          show("block2");
```

```
        hide("block1");
      }
    }
  //-->
  </script>
</head>
<body>
  <form action="#" method="post">
    <input type="radio" name="rb" value="1" onclick="change(1)">Блок 1
    <input type="radio" name="rb" value="2" onclick="change(2)">Блок 2
  </form>
  <div id=block1>
    Содержимое первого блока
  </div>
  <div id=block2 style="display: none">
    Содержимое второго блока
  </div>
</body>
</html>
```

ПРИЛОЖЕНИЕ 5

Flash, PHP и JavaScript

Flash — это мультимедийный графический формат, обеспечивающий работу с векторной и растровой графикой, а также со звуком. Flash имеет внутренний язык сценариев, с помощью которого можно создавать интерактивные мультимедийные компоненты (Flash-фильмы), использующиеся в мультимедиасистемах, презентациях, анимации и т. д. Благодаря небольшому размеру получаемых Flash-фильмов и интерактивности, технология Flash успешно применяется в оформлении Web-сайтов. В основном, это создание анимационных роликов, целью которых является повышение привлекательности сайта. Не менее важное использование технологии Flash — реализация пользовательского интерфейса. В этом случае Flash можно рассматривать в качестве замены JavaScript и технологии DHTML, которые также призваны обеспечивать интерактивную работу с пользователями.

Далее будут рассмотрены способы применения технологии Flash при разработке Web-сайтов: генерация Flash-фильмов на сервере с помощью языка PHP, внедрение Flash-фильмов в HTML-код, метод определения наличия Flash-проигрывателя в браузере посетителя и передача параметров из HTML-страницы во Flash с помощью JavaScript.

Создание Flash с помощью PHP

Для создания Flash-фильмов с помощью PHP предназначена библиотека ming. С ее помощью можно динамически генерировать Flash-фильмы и отображать их на страницах сайта. Задача динамического создания Flash актуальна в том случае, если используется большое число уникальных Flash-фильмов, которые затруднительно создать вручную, либо параметры создаваемых Flash-фильмов невозможно определить до начала выполнения скриптов, а также при создании страниц по технологии Flash.

Библиотека ming, позволяющая работать с Flash, поддерживает почти все основные объекты и свойства Flash 4, такие как формы, градиенты, изображения в форматах JPEG и PNG, трансформацию форм из одной в другую, текст, кнопки, механизм событий, movie-фильмы, загрузку потокового звука в формате MP3, трансформацию цветов. Единственная неподдерживаемая опция — это звуковые события.

Библиотека ming — это разработка с открытым кодом по соглашению LGPL. Новости об этой библиотеке, примеры кода, новые версии доступны на сайте, посвященном библиотеке ming, по адресу **http://ming.sourceforge.net**. Библиотека ming была введена в PHP с версии 4.0.5.

Замечание

Подробное описание функций для работы с библиотекой ming можно найти в нашей предыдущей книге по языку PHP — "PHP 5. Практика разработки Web-сайтов".

Замечание

Как и любое расширение в PHP 5, библиотека ming по умолчанию отключена. Для ее подключения следует снять комментарий со строки `extension = php_ming.dll` в конфигурационном файле php.ini.

Ниже будет рассмотрено несколько примеров, демонстрирующих создание Flash-фильмов с помощью PHP.

Создание Flash-фильма с градиентом

В листинге П5.1 первоначально создается объект Flash-фильма `SWFMovie` и устанавливаются его размеры. Затем создаются объекты формы `SWFShape` и градиента `SWFGradient`. С помощью метода `addEntry()` добавляются 3 цвета градиента. Методом `addFill()` градиент добавляется к форме, и тем самым создается объект заливки `SWFFill`, над которым затем производятся операции трансформации. Окончательная установка заливки осуществляется методом `setRightFill()`. После всех необходимых действий по настройке объектов градиента и заливки рисуются линии формы методами `drawLine()`. Последним этапом объект формы добавляется к объекту Flash-фильма (`add($s)`) и выводится в браузер методом `output()`.

Листинг П5.1. Создание Flash-фильма с градиентом

```
<?php
  // Создание объекта Flash-фильма
  $m = new SWFMovie();
  $m->setDimension(400, 400);
```

```
// Создание объекта формы
$s = new SWFShape();
// Создание объекта градиента
$g = new SWFGradient();
$g->addEntry(0.0, 255, 0, 50);
$g->addEntry(0.5, 0, 120, 0);
$g->addEntry(1.0, 50, 255, 255);
// Присоединение градиента к форме и создание объекта линейной заливки
$f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
// Трансформации объекта заливки SWFFill
$f->scaleTo(0.05, 0.05);
$f->moveTo(70, 50);
$f->rotateTo(25);
$s->setRightFill($f);
// Рисование формы
// Установка стиля линий
$s->setLine(1, 0, 0, 0);
// Перемещение позиции рисования
$s->movePenTo(20, 10);
// Рисование линий
$s->drawLine(0, 100);
$s->drawLine(100, 0);
$s->drawLine(0, -100);
$s->drawLine(-100, 0);
// Добавление формы к объекту SWFMovie
$m->add($s);
// Вывод созданного Flash-фильма в браузер
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

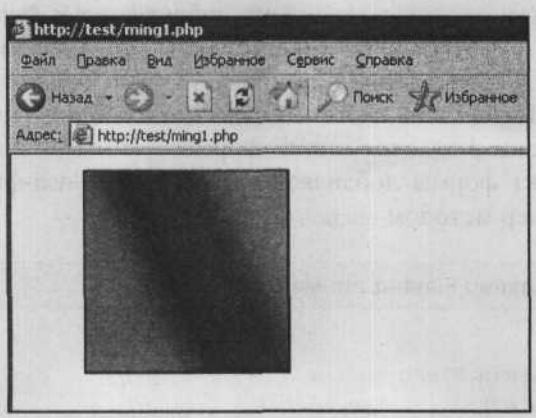


Рис. П5.1. Flash-фильм с градиентом

Результат выполнения кода, приведенного в листинге П5.1, показан на рис. П5.1.

Создание Flash-фильма с трансформацией объектов

В данном примере рассмотрена работа с объектом `SWFMorph`, с помощью которого одна форма трансформируется в другую. Процесс трансформации анимируется в объекте `SWFMovie` (листинг П5.2).

Листинг П5.2. Трансформация одной формы в другую

```
<?php
  // Создание Flash-фильма
  $m = new SWFMovie();
  // Установка размеров
  $m->setDimension(320, 240);
  // Создание объекта трансформации
  $p = new SWFMorph();
  // Получение указателя на первую форму трансформации
  $s = $p->getShape1();
  // Рисование формы
  $s->setLine(0,0,0,0);
  $f = $s->addFill(255, 0, 0);
  $s->setLeftFill($f);
  $s->movePenTo(0,30);
  $s->drawLine(200,0);
  $s->drawLine(0,30);
  $s->drawLine(-200,0);
  $s->drawLine(0,-30);
  // Получение указателя на вторую форму трансформации
  $s = $p->getShape2();
  // Рисование формы
  $s->setLine(1,0,0,0);
  $f = $s->addFill(0, 0, 255);
  $s->setLeftFill($f);
  $s->movePenTo(100,50);
  $s->drawLine(50,-50);
  $s->drawLine(50,50);
  $s->drawLine(-50,50);
  $s->drawLine(-50,-50);
  // Добавление объекта трансформации во Flash-фильм
  $i = $m->add($p);
```

```
// Установка коэффициента выполнения трансформации
// одной формы в другую для последовательности кадров
for($r=0.0; $r<=1.0; $r+=0.1)
{
  // Установка коэффициента выполнения трансформации
  // для текущего кадра
  $i->setRatio($r);
  // Переход к следующему кадру
  $m->nextFrame();
}
// Вывод Flash-фильма в браузер
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

В листинге П5.2 первоначально создаются объекты Flash-фильма `SWFMovie` и трансформации `SWFMorph`. Затем с помощью метода `getShape1()` возвращается указатель на объект первой формы, над которой будет произведена операция трансформации. На объект формы добавляется заливка и линии несколькими методами `drawLine()`. Аналогичные действия произведены со второй формой, которая возвращается методом `getShape2()`. Затем методом `add()` объект трансформации `SWFMorph` добавляется во Flash-фильм.

Для создания анимированной трансформации необходимо не только создать объект `SWFMorph`, но и установить коэффициент выполнения трансформации для каждого кадра анимации. Это достигается вызовом метода `setRatio()`, который выполняется в цикле `for`. Переход к следующему кадру осуществляется методом `nextFrame()`. По окончании настройки анимации объект `SWFMovie` выводится в браузер.

Результатом выполнения кода, приведенного в листинге П5.2, является анимация трансформации прямоугольника в ромб, три кадра из которой показаны на рис. П5.2.

На рис. П5.2 зафиксированы три точки выполнения трансформации: 0, 50 и 100%.

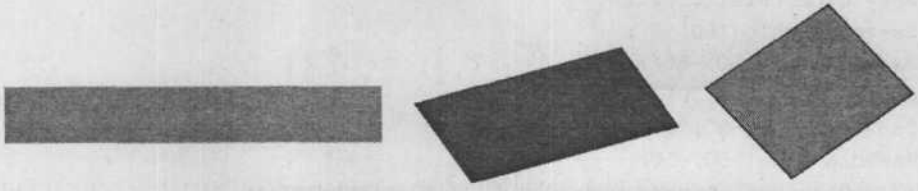


Рис. П5.2. Процесс трансформации прямоугольника в ромб

Определение наличия Flash у посетителя

Самый надежный способ определения, работает какая-либо технология (Flash, cookies, ActiveX) на компьютере посетителя или нет, — попытаться выполнить код, использующий данную технологию. И уже по результатам выполнения этого проверочного кода делать вывод о наличии либо отсутствии соответствующих компонентов. Именно таким способом будет осуществлена проверка наличия установленного Flash-дополнения на компьютере посетителя.

Для осуществления проверки понадобятся:

- проверочный Flash-фильм;
- входная страница index.html, на которой будет размещен проверочный Flash-фильм.

Проверочный Flash-фильм внедрен в HTML-код входной страницы и состоит из одного кадра, выполняющего единственное действие — переход на специальную страницу, созданную с использованием технологии Flash, например, так:

```
GetURL("flash.html");
```

Таким образом, если Flash-дополнение установлено в браузере пользователя, то при загрузке проверочного Flash-фильма будет осуществлена переадресация на страницу с использованием Flash-компонентов. Если Flash-дополнение не установлено, то Flash-фильм не будет выполнен и переадресации не произойдет.

Если переадресация на Flash-страницу не выполнена, то нужно перенаправить посетителя на страницу без использования Flash. Это можно сделать, поместив на страницу мегатег переадресации в блок <head>:

```
<META HTTP-EQUIV="Refresh" CONTENT="2; URL=normal.html">
```

Данная конструкция осуществляет переадресацию посетителя на страницу normal.html через 2 секунды после загрузки страницы. Задержка в 2 секунды перед выполнением переадресации необходима для того, чтобы дать время отработать проверочному Flash-фильму.

Вставка Flash в HTML-страницу

Для вставки Flash-фильмов в HTML-страницу используются одновременно два тега — <object> и <embed>. Тег <embed> вставляется внутрь тега <object>. Такая конструкция является следствием того, что часть браузеров не поддерживает технологию ActiveX и тег <object>.

Браузеры, поддерживающие ActiveX (Internet Explorer и браузеры на его основе), используют для отображения Flash-фильма параметры тега <object> и игнорируют вставленный внутрь него тег <embed>.

Браузеры без поддержки ActiveX (Netscape, Mozilla, Opera) игнорируют тег <object>, т. к. он ими не поддерживается, и используют для отображения Flash-фильма известный им тег <embed>. Пример вставки Flash-фильма в HTML-страницу приведен в листинге П5.3.

Листинг П5.3. Вставка Flash-фильма в HTML-страницу

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
   codebase="http://download.macromedia.com/pub/shockwave/
                                    cabs/flash/swflash.cab#version=6,0,0,0"
   width="550" HEIGHT="400" id="myflash">
   <param name=name value="myflash">
   <param name=movie value="myflash.swf">
   <param name=quality value=high>
   <embed src="myflash.swf" quality=high width="550" height="400"
     name="myflash"
     swliveconnect=true
     type="application/x-shockwave-flash"
     pluginspage="http://www.macromedia.com/go/getflashplayer">
   </embed>
</object>
```

В листинге П5.3 в HTML-код вставляется Flash-фильм из файла myflash.swf шириной 550 и высотой 400 пикселов. Для описания Flash-фильма в тегах <object> и <embed> могут использоваться следующие параметры:

- `src` — путь к загружаемому фильму (только для <embed>);
- `movie` — путь к загружаемому фильму (только для <object>);
- `classid` — идентификатор ActiveX (только для <object>);
- `width` — ширина Flash-фильма в пикселах;
- `height` — высота Flash-фильма в пикселах;
- `codebase` — URL, по которому браузер будет пытаться загрузить Flash-дополнение, если оно не установлено у посетителя (для тега <object>);
- `pluginspage` — URL, по которому браузер будет пытаться загрузить Flash-дополнение, если оно не установлено у посетителя (для тега <embed>);
- `play` — определяет, начинать ли проигрывание фильма сразу после загрузки или нет. Может принимать значение `true` или `false`;

- `loop` — определяет, будет ли фильм проигрываться в бесконечном цикле. Может принимать значение `true` или `false`;
- `quality` — качество отображения фильма. Может принимать значения: `best`, `high`, `medium`, `autohigh`, `autolow`, `low`. Значение `best` соответствует самому высокому качеству, а `low` — самому низкому. Чем выше качество проигрывания фильма, тем больше системных ресурсов он потребляет;
- `bgcolor` — цвет фона фильма;
- `scale` — определяет параметры масштабирования фильма, если размеры в свойствах `width` и `height` указаны в процентах. Может принимать одно из значений:
  - `showall` — растягивает фильм для заполнения заданной для него области с сохранением пропорции фильма (используется по умолчанию). Если пропорции фильма и заданной для него области не совпадают, то при отображении могут появиться рамки по сторонам фильма;
  - `noborder` — полностью заполняет заданную для него область с сохранением пропорций фильма. Неуместившиеся части фильма, выходящие за пределы заданной области, отсекаются;
  - `noscale` — фильм не масштабируется;
  - `exactfit` — масштабирует фильм по размерам заданной для него области. Пропорции не сохраняются;
- `base` — основной URL, который используется при относительной переадресации, если фильм состоит из нескольких фильмов, хранящихся в разных каталогах;
- `menu` — тип контекстного меню, появляющегося при нажатии правой кнопки мыши в области фильма. Может принимать значение `true` или `false`. Значение `true` отображает полное меню. При установке в значение `false` контекстное меню содержит только пункт **About Flash**.
- `wmode` — определяет параметры прозрачности. Может принимать значения:
  - `window` — отображается в собственном прямоугольном окне; используется по умолчанию;
  - `opaque` — фильм помещается на задний план страницы;
  - `transparent` — устанавливает прозрачный фон фильма, сквозь который будет видно нижележащие элементы страницы.

  Используется только для тега `<object>`;
- `swliveconnect` — применяется для разрешения передачи данных между JavaScript и Flash-фильмом (для тега `<embed>`). Значение `true` разрешает передачу данных, `false` (по умолчанию) — запрещает.

Передача параметров из JavaScript во Flash

Передача параметров из JavaScript во Flash может использоваться для организации динамических интерактивных страниц без обращения к серверным скриптам. Передача данных осуществляется посредством функций JavaScript. Для организации передачи данных должно быть выполнено несколько условий:

- тег `<object>` должен иметь параметр `id`, например `id="myflash"`;
- тег `<embed>` должен иметь параметр `name`, например `name="myflash"`;
- для тега `<embed>` должен быть установлен параметр `SWLIVECONNECT=true`, разрешающий передачу данных между Flash и JavaScript.

Передача данных осуществляется выполнением метода `SetVariable()` у Flash-фильма. Метод имеет следующий формат:

```
window.document.флешФильм.SetVariable(variableName, variableValue)
```

Пример:

```
window.document.myflash.SetVariable("param",12345);
```

В листинге П5.4 приведен пример, устанавливающий переменной `param`, находящейся на корневом уровне Flash-фильма, значение 12345. Передача данных осуществляется вызовом функции `send()` при нажатии кнопки **Передать данные**.

Листинг П5.4. Передача параметров из JavaScript во Flash

```
<head>
<SCRIPT language=javascript>
<!--
function send(param)
{
   window.document.myflash.SetVariable("_root.param",param);
}
// -->
</script>
</head>
<body>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
   codebase="http://download.macromedia.com/pub/shockwave/
                  cabs/flash/swflash.cab#version=6,0,0,0"
   width="550" HEIGHT="400" id="myflash">
   <param name=name value="myflash">
```

```
    <param name=movie value="myflash.swf">
    <param name=quality value=high>
    <embed src="myflash.swf" quality=high width="550" height="400"
      name="myflash"
      swliveconnect=true
      type="application/x-shockwave-flash"
      pluginspage="http://www.macromedia.com/go/getflashplayer">
    </embed>
</object>
<input type="submit" onclick="send(12345)" value="Передать данные">
```

Предметный указатель

А

Б

В

Д

З

И

К

М

О

П

Р

С

Т

У

Ф

Продолжение рубрики см. на с. 574

Х

Ц

Ч

Э

Я