

Первый пример интернет-магазина, рассматриваемый в книre, называется "Знание — сила". Этот интернет-магазин предназначен для продажи контента. Ему присущи следующие основные характеристики:

- использование HTML, PHP и MySQL для создания простого приложения;
- зависимость от учетных записей пользователей;
- предоставление администраторам возможности добавлять контент в форматах HTML и PDF;
- обработка платежей с помощью PayPal.

Этот пример интернет-магазина относительно прост и может применяться фирмами, относящимися к категории малого и среднего бизнеса. В части III будет рассмотрен более сложный проект интернет-магазина "Кофе", для создания которого требуется знание профессиональных методик работы с HTML, PHP и MySQL. Для покупок в интернет-магазине "Кофе" не нужно создавать учетную запись, существует возможность предварительного заказа товаров и используется альтернативная платежная система. Сказанное вовсе не означает, что первый интернет-магазин никуда не годится. Это всего лишь свидетельствует о том, что первый пример интернет-магазина проще в реализации.

Система учетных записей пользователей включает несколько компонентов, предназначенных для регистрации пользователей, входа в систему, выхода из системы, восстановления забытых паролей либо изменения существующих паролей. В первом примере интернет-магазина также применяется защищенный способ обработки паролей пользователей. Платный контент, просматриваемый клиентами, может храниться в двух форматах: HTML и PDF. Для обработки контента в HTML-формате предусмотрен WYSIWYG-редактор, встроенный в HTML-форму. С помощью этого редактора администраторы могут легко создавать HTML-код, не имея ни малейшего представления о языке разметки HTML. Для обработки контента в PDF-формате используется *прокси-сценарий*, предназначенный для работы с защищенными файлами, которые недоступны при использовании протокола HTTP либо для пользователей, не прошедших процедуру верификации.

В этой главе содержатся сведения, необходимые на начальной стадии разработки сайта интернет-магазина. Рассматриваются такие темы, как проектирование базы данных, структура файлов на сервере, HTML-шаблон и пара вспомогательных файлов, используемых сценариями PHP. Код интернет-магазина можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-5-8459-1939-7.html

# ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

База данных, разработанная для этого примера, проста и функциональна. Она включает пять таблиц (рис. 3.1) и называется ecommerce1.



Рис. 3.1. Структура базы данных

В таблицах categories, pages и pdfs хранится информация, относящаяся к "товарам" интернет-магазина. В таблице categories перечислены категории, применяемые для упорядочения HTML-контента. Категория может включать одну или несколько страниц, но каждая страница может относиться только к одной категории.



#### Совет

Разработку нового сайта я всегда начинаю с создания схемы базы данных и выбора способов создания HTML-кода. Но можно начать с пользовательского интерфейса, а затем перейти на нижний уровень проектирования базы данных.

В таблице pages хранится актуальный HTML-контент сайта. На каждой HTML-странице находятся три важных столбца: title, description и content. Столбец title используется в качестве ссылки на остальные страницы, а также может применяться в качестве заголовка браузера. В столбце description содержится небольшой текстовый блок, применяемый для описания контента страницы. Этот блок доступен для просмотра пользователям и в поисковых системах. В столбце content хранится фактический HTML-контент. Подробнее работа с HTML-контентом рассматривается в главе 5.

В таблице pdfs перечислены подробные сведения о каждом PDF-документе, находящемся на сайте. Здесь находится заголовок PDF-документа, напоминающий заголовок HTML-страницы, краткое описание, доступное для просмотра любому пользователю, реальное название PDF-файла и его размер, выраженный в килобайтах. Загруженный на сайте PDF-файл сохраняется под именем, сгенерированным случайным образом, например:

#### ce0347c0045658845967215025e56d53d31b4c6a-51fa6740714da8.33955614

Это имя должно храниться в базе данных, поскольку применяется для поиска файла на сервере. Если же PDF-файл представляется пользователю, отображается его исходное название. В отличие от HTML-контента PDF-файлы не упорядочиваются с применением информационных категорий.



#### Примечание

Большинство таблиц включают столбец, предназначенный для хранения даты и времени добавления записи. В таблице users также имеется поле, в котором отображаются дата и время последнего изменения записи.

В таблице users хранится минимум информации о клиентах, которые могут создавать свои пользовательские имена. Также в таблице users хранятся адрес электронной почты, пароль, имя и фамилия клиента. Пароль хранится в виде *xeu-кoдa*, который является представлением определенного значения. (Это зашифрованная версия пароля, которую можно расшифровать.) Хеш-код всегда имеет фиксированную длину, поэтому столбец, в котором хранится код, *может* иметь тип CHAR фиксированной длины. В данной ситуации для этого столбца лучше выбирать тип VARCHAR, чтобы обеспечить возможность изменения алгоритма в дальнейшем. Вполне возможно, что для выполнения подобных изменений потребуется перерегистрация пользовательских паролей.

Допускаются два типа пользователей сайта: member (участник) и admin (администратор). В столбце ENUM (нумерованный список параметров) хранится тип пользователя, по умолчанию member (участник). Несмотря на то что администраторы имеют привилегии (например, не оплачивают доступ к сайту), для них и для обычных пользователей применяется одна и та же методика входа в систему. Поэтому администраторы также должны регистрироваться в базе данных. Таблица users включает столбец date\_expires, в котором хранится дата окончания действия учетной записи пользователя, применяемой для оплаты контента. Как только пользователь создаст первые подписки и выполнит первые платежи, срок действия учетной записи устанавливается равным одному году. Если пользователь обновляет членство на сайте, то срок действия учетной записи продлевается на год. Пользователи с истекшим сроком действия учетной записи также могут входить на сайт, но они не получат доступа к контенту. Им будут разосланы уведомления о необходимости продления регистрации на сайте.



#### Совет

Лучше хранить в базе данных избыточную информацию, чтобы потом не сожалеть о том, что какие-то важные данные были утеряны.

Для выполнения платежей используется платежная система PayPal. Несмотря на то что PayPal perистрирует детали каждой транзакции, сохраняйте сведения о выполненных транзакциях на вашем сайте. Информация о транзакциях PayPal находится в таблице orders. Она связана с идентификатором пользователя (ID), хранящимся в таблице users. Каждый заказ связан с единственным пользователем, а каждому пользователю может соответствовать одна или несколько записей в таблице orders. Также записываются три информационных фрагмента, относящихся к транзакции PayPal: transaction\_id (уникальный идентификатор), payment\_status (код подтверждения) и payment\_amount (сумма платежа). Значение payment\_amount хранится в виде целого числа, которое компьютеры легче обрабатывают, чем числа с плавающей точкой. При выполнении кода суммы платежей будут преобразовываться из формата с плавающей точкой в целочисленный формат (и обратно), если возникнет такая необходимость.

Благодаря хранению базовых сведений о платежах возможно создание простого интерфейса администрирования, предназначенного для просмотра общего количества заказов, сумм платежей и другой финансовой информации. При этом не требуется обращаться к PayPal за соответствующими данными. Идентификатор transaction\_id является наиболее важной частью информации, хранящейся в базе данных, поскольку позволяет получить доступ ко многим деталям транзакции. Поэтому доступ к этому идентификатору может получить лишь авторизованный владелец связанной учетной записи PayPal.

Я предполагаю, что вы владеете методиками создания базы данных и относящихся к ней таблиц с помощью таких инструментов, как phpMyAdmin, клиента командной строки mysql и прочих подобных инструментов. Если же вы не имеете опыта создания баз данных, прочтите одну из книг по MySQL, то найдите интересующие вас сведения в Интернете либо задайте вопросы на одном из форумов, посвященных программированию на MySQL. Приведенный ниже код SQL можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-0-3219-4936-3.html



#### Совет

В части IV приведены дополнительные таблицы, расширяющие этот пример кода.

```
CREATE TABLE 'categories' (
    'id' SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
    'category' VARCHAR(45) NOT NULL,
    PRIMARY KEY ('id'),
    UNIQUE INDEX 'category_UNIQUE' ('category' ASC)
) ENGINE = InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE 'orders' (
    'id' INT UNSIGNED NOT NULL AUTO_INCREMENT,
    'users_id' INT UNSIGNED NOT NULL,
    'transaction_id' VARCHAR(45) NOT NULL,
    'payment_status' VARCHAR(45) NOT NULL,
    'payment_amount' INT UNSIGNED NOT NULL,
    'date_created' TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY ('id'),
```

```
INDEX 'date created' ('date created' ASC),
   INDEX 'transaction id' ('transaction id' ASC),
   CONSTRAINT 'fk orders users1' FOREIGN KEY ('id')
      REFERENCES 'users' ('id')
      ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE = InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE 'pages' (
   'id' INT UNSIGNED NOT NULL AUTO INCREMENT,
   'categories id' SMALLINT UNSIGNED NOT NULL,
   'title' VARCHAR(100) NOT NULL,
   'description' TINYTEXT NOT NULL,
   'content' LONGTEXT NULL,
   'date created' TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP,
   PRIMARY KEY ('id'),
   INDEX 'date created' ('date created' ASC),
   INDEX 'fk_pages_categories idx' ('categories id' ASC),
   CONSTRAINT 'fk pages categories' FOREIGN KEY ('categories id')
      REFERENCES 'categories' ('id')
      ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE = InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE 'pdfs' (
   'id' INT UNSIGNED NOT NULL AUTO INCREMENT,
   'title' VARCHAR(100) NOT NULL,
   'description' TINYTEXT NOT NULL,
   'tmp name' CHAR(63) NOT NULL,
   'file name' VARCHAR(100) NOT NULL,
   'size' MEDIUMINT UNSIGNED NOT NULL,
   'date created' TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP,
   PRIMARY KEY ('id'),
   UNIQUE INDEX 'tmp name UNIQUE' ('tmp name' ASC),
   INDEX 'date created' ('date created' ASC)
) ENGINE = InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE 'users' (
   'id' INT UNSIGNED NOT NULL AUTO INCREMENT,
   'type' ENUM('member', 'admin') NOT NULL DEFAULT 'member',
   'username' VARCHAR(45) NOT NULL,
   'email' VARCHAR(80) NOT NULL,
   'pass' VARCHAR(255) NOT NULL,
   'first name' VARCHAR(45) NOT NULL,
   'last name' VARCHAR(45) NOT NULL,
   'date created' TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP,
   'date expires' DATE NOT NULL,
   'date modified' TIMESTAMP NOT NULL DEFAULT CURRENT TIMESTAMP
   ON UPDATE CURRENT TIMESTAMP,
   PRIMARY KEY ('id'),
   UNIQUE INDEX 'username UNIQUE' ('username' ASC),
   UNIQUE INDEX 'email UNIQUE' ('email' ASC),
   INDEX 'login' ('email' ASC, 'pass' ASC)
) ENGINE = InnoDB DEFAULT CHARSET=utf8;
```

В соответствии со стандартами производительности и нормализации столбцы таблиц должны определяться как NOT NULL. Также соответствующим образом установлены заданные по умолчанию значения. Индексы нужно создавать в столбцах первичных ключей, в столбцах с уникальными значениями, в столбцах, используемых в объединениях и в инструкциях WHERE и ORDER BY. Не экономьте на индексах, используйте их чаще.

Ограничения внешнего ключа имеют место в двух таблицах: pages и orders. В соответствии с этими ограничениями соответствующее значение для внешнего ключа должно существовать в качестве первичного ключа в связанной таблице. Например, заказ, для которого идентификатор users\_id равен 23890, может быть сохранен только в том случае, если в таблице users существует запись с идентификатором id, равным 23890. Не предусмотрены какие-либо действия, выполняемые при обновлении или удалении связанной записи первичного ключа (например, записи в таблице users с идентификатором id, равным 23890). Поэтому невозможно удалить запись первичного ключа либо изменить значение первичного ключа в таблицах базы данных сайта.



#### Совет

Если вы не знакомы с ограничениями внешних ключей, просмотрите соответствующие книги по MySQL, воспользуйтесь поиском в Интернете либо обратитесь к одному из многочисленных форумов в Интернете.

По умолчанию в MySQL используется механизм хранения данных InnoDB, который, как правило, выбирается в качестве типа таблиц. Для базы данных и сайта применяется кодировка UTF-8, обеспечивающая поддержку всех письменных языков.

# СТРУКТУРНАЯ ОРГАНИЗАЦИЯ СЕРВЕРА

Прежде чем приступить к созданию HTML-документов или PHP-сценариев, рассмотрим, каким образом должен быть организован сервер. Большинству сайтов присуща структура, предусматривающая наличие в корневом каталоге сайта папок, предназначенных для размещения следующих элементов:

- файлы, применяемые для целей администрирования;
- CSS-файлы;
- изображения;
- другие медиаобъекты;
- фрагменты кода PHP;
- сценарии JavaScript.



#### Совет

Чтобы радикально улучшить защиту сайта, переименуйте папки includes и administration, дав им нестандартные имена.

Если речь идет о больших сайтах, включающих несколько общедоступных областей, для каждой области следует выделить отдельную подпапку: магазин, форумы, блог, учетная запись и другие подобные области. Рассматриваемый в этой части книги сайт относительно

невелик и содержит около десятка общедоступных страниц, которые находятся в корневой папке сайта.

Для двух страниц администрирования сайта не требуется отдельная папка. Также не нужны отельные папки для хранения изображений либо других медиаобъектов, поскольку для создания сайта применяется базовый HTML-шаблон. В корневой папке сайта находятся три подпапки, которые называются css, includes и js.



# Совет

В главе 7 рассматриваются альтернативные способы защиты папок сайта.

В папке css находятся два файла, заимствованные из шаблона сайта: bootstrap.min. css и sticky-footer-navbar.css. Подробнее этот шаблон рассматривается далее.

В папке includes находятся сценарии PHP, которые будут включаться другими сценариями. Другими словами, папка includes предназначена для хранения файлов, которые не могут вызываться сами по себе. В следующих трех главах будут созданы шесть сценариев, которые находятся в папке includes:

- config.inc.php: сценарий, определяющий общее поведение сайта и разные константы;
- footer.html: представляет завершающую часть шаблона HTML;
- form\_functions.inc.php: определяет функцию, используемую каждой формой;
- header.html: первая часть шаблона HTML;
- login.inc.php: обрабатывает процесс входа в систему;
- login\_form.inc.php: включает форму входа в систему.

Чтобы облегчить поддержку сайта, функциональные средства помещаются в отдельные файлы.

В папке js находится единственный файл, требуемый шаблоном сайта: bootstrap. min.js. Подробнее шаблон рассматривается в одном из следующих разделов. В папке js также хранятся файлы, используемые WYSIWYG-редактором.

Наравне со сценариями PHP, применяемыми для выполнения стандартных для страниц операций, таких как регистрация, выход и другие подобные действия, для сайта потребуется дополнительный сценарий PHP под названием mysql.inc.php. Этот сценарий выполняет подключение к базе данных. Поскольку этот сценарий определяет важную информацию, его не следует помещать в корневом каталоге сайта.

Для сайта также требуется папка, в которой хранятся PDF-документы, доступные для платных подписчиков. Эти документы помещаются в папку с помощью сценария загрузки. Веб-серверу назначаются разрешения, позволяющие выполнять запись данных в эту папку. Поскольку подобный алгоритм приводит к появлению потенциальной угрозы безопасности, эту папку не рекомендуется помещать в корневой каталог сайта. Доступ к PDF-файлам открывается только для платных подписчиков. При этом используется прокси-сценарий, верифицирующий пользователей.



#### Примечание

В книге вы не найдете описания файла robots.txt, показанного на рисунках. Этот файл можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-5-8459-1939-7.html

Структура сервера показана на рис. 3.2. В качестве корневой папки сайта используется папка html. Для получения доступа к этой папке применяется ссылка www.example.com.

			_		_		_				
joj ntmi											크비스
Файл Правка Вид Избранное Сервис Справка											
🕞 Назад 🔹 🐑 • 🥂 🔎 Поиск 👔	Папки	3× 7	8.5. <b>-</b>								
Адрес: C:\Site\ex1\html											🕶 🔿 Переход
Папки	<							Case D			
Share	• I D	þ	6		6	0	2		0 % H ×		
E SIERRA	ajax	CSS	images	indudes	js	Button	template	robots	.DS Store	add page	add pdf
🗆 🔜 Site											
🗆 🔜 ex1		- 3		- 3		- 3					
🖃 🕜 html		3	•		•		•				
ajax	add to fay	cancel	category	change pas	favorites	forgot pas	history	index	index-bonus	ion	inn log
CSS	1		,.,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,				,,				
images -											
🗄 🔜 indudes		0	0		0						
±js			-	_	-						
pdfs	logout	page	pdfs	pdt	register	remove_fro	renew	reset	thanks	view_pdf	
	<b>-</b>										
<u>↓</u>											

Рис. 3.2. Структура веб-сервера

Если вы пользуетесь общим хостом, то не сможете поместить какие-либо файлы в корневую папку сайта. В этом случае нужно воспользоваться структурой, подобной показанной на рис. 3.3. Сценарий mysql.inc.php и папка pdfs были перемещены в папку includes. С помощью инструментов веб-сервера ограничьте доступ к папке includes. Чтобы запретить доступ к этой папке, воспользуйтесь файлами Apache .htaccess. В результате посетители сайта не смогут загружать контент, находящийся в папке includes (и в подпапке pdfs). Получить доступ к этому контенту можно лишь с помощью сценариев PHP, находящихся на сервере. Подробные сведения о блокировании доступа к папкам приведены в главе 7.



#### Примечание

На рис. 3.2 и 3.3 показаны файлы и папки, которые будут созданы в следующих четырех главах.



Рис. 3.3. Структура веб-сервера в случае выбора общего хоста

### Расширения файлов

Как только пользователь запрашивает страницу с расширением .html, сервер передает ее содержимое непосредственно браузеру, не выполняя какую-либо дополнительную обработку. Если же пользователь запрашивает страницу с расширением . php, то сервер сначала запускает на выполнение контент страницы. При этом используется интерпретатор РНР, обрабатывающий код РНР. Если же изменить стандартные настройки сервера, то обработка расширений файла будет выполняться иным образом. Поэтому для главных страниц сайта, к которым получает непосредственный доступ пользователи, используется расширение . php. С помощью этого расширения автоматически запускается код РНР. Если же страницы включаются с помощью других сценариев РНР и не запускаются непосредственно браузером, то они не будут обрабатываться сервером. В этом

случае можно выбрать расширение страницы на свое усмотрение.

Расширение .html присуще файлам, включающим код HTML, например файлам заголовков и футеров. Если же страницы в основном состоят из кода РНР и не предназначены для использования в качестве включаемых файлов, то применяется комбинированное расширение.inc.php. Расширение.inc указывает на то, что файл будет включен в процессе выполнения основного кода, а расширение . php предотвращает отображение кода при передаче страницы непосредственно веб-браузеру. Если же для файла выбрать расширение .inc, то сервер не отсылает содержимое интерпретатору РНР, в результате чего появляется риск пересылки потенциально важной информации браузеру. Не рискуйте понапрасну.

# ПОДКЛЮЧЕНИЕ К БАЗЕ ДАННЫХ

Сценарии РНР, находящиеся на сайте, требуют подключения к базе данных. Для выполнения этой задачи нужно создать отдельный файл.



#### Примечание

Настройте текстовый редактор или среду IDE таким образом, чтобы сохранять созданные страницы в кодировке UTF-8, используемой для сайта.

- С помощью текстового редактора или в среде IDE создайте новый сценарий PHP и присвойте ему имя mysql.inc.php.
   Значок этого файла показан на рис. 3.3.
- 2. Присвойте значения константам, применяемым для получения доступа к базе данных.

```
<?php
DEFINE ('DB_USER', 'username');
DEFINE ('DB_PASSWORD', 'password');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'ecommerce1');
```

Вместо значений username и password подставьте имя пользователя и пароль, применяемые для подключения к базе данных MySQL. Для рассматриваемого примера были созданы база данных ecommercel и учетная запись пользователя MySQL с привилегиями SELECT, INSERT и UPDATE по отношению к этой базе данных. Выбирайте имя пользователя и пароль, которые трудно было бы подобрать сразу!

#### 3. Подключитесь к базе данных с помощью следующей строки кода:

\$dbc = mysqli connect (DB HOST, DB USER, DB PASSWORD, DB NAME);

Для подключения к базе данных применяется функция mysqli\_connect(). Строка подключения присвоена переменной <code>\$dbc</code>, которая может использоваться многими функциями в других сценариях.



#### Примечание

Для обработки ошибок подключения применяется настраиваемый обработчик ошибок, заданный в файле конфигурации.

4. С помощью следующей строки выберите применяемый набор символов:

```
mysqli set charset($dbc, 'utf8');
```

С помощью этой функции задается набор символов, используемый для обмена данными между PHP и базой данных. Таблицы базы данных, HTML-страницы и подключение PHP–MySQL должны использовать один и тот же набор символов.

 Начните определять функцию, обеспечивающую безопасное использование данных в запросах:

function escape data (\$data, \$dbc) {

Эта функция применяет фрагмент данных в качестве первого аргумента, а затем возвращает версию этого фрагмента, которую можно безопасно применять в запросах к базе данных. Данная функция сводит к нулю вероятность атак внедрения SQL-кода (см. главу 2) и выполняет следующие три действия:

- удаляет дополнительные символы косой черты, если активизированы "волшебные кавычки" (Magic Quotes);
- устраняет дополнительные пробелы из массива данных;

• обрабатывает данные с помощью функции mysqli\_real\_escape\_string(). Последняя операция является наиболее важной, поскольку функция mysqli\_real\_ escape\_string() подготавливает данные для безопасного использования запросом. В результате выполнения этой функции учитываются конфигурация базы данных и применяемый набор символов. Эта функция нуждается в подключении к базе данных, информация о котором передается во втором аргументе.



#### Примечание

"Волшебные кавычки" (Magic Quotes) не рекомендуются к применению в PHP 5.3 и исключены из PHP 5.4.

6. Если задействованы "волшебные кавычки", то следующая строка кода удаляет лишние символы косой черты:

```
if (get magic quotes gpc()) $data = stripslashes($data);
```

С помощью "волшебных кавычек" обеспечивается общий уровень безопасности входящих данных, который будет все же ниже, чем в случае использования функции mysqli\_real\_escape\_string(). Поэтому режим "волшебных кавычек" обычно отключается для сервера. Если же "волшебные кавычки" активизированы, то в потоке входных данных символы косой черты будут выделять потенциально

опасные символы, которые могут привести к сбою запроса. Эта проблема может проявиться после выполнения функции mysqli\_real\_escape\_string(), которая создает дополнительные символы косой черты. В результате вместо одного символа появятся два. Чтобы устранить лишние символы косой черты, выполните строку кода из этого пункта, которая проверяет значение настройки **Magic Quotes** ("волшебные кавычки"). Если эта настройка активизирована, то вызывается функция stripslashes().

7. Следующая строка кода возвращает "обрезанную" защищенную версию данных:

return mysqli real escape string (\$dbc, trim (\$data));

Разница между функцией mysqli\_real\_escape\_string() и "волшебными кавычками" либо функцией addslashes() заключается в том, что функция mysqli\_ real\_escape\_string() идентифицирует "проблемные" символы на основе информации в базе данных, применяемого набора символа и других факторов.

- 8. Завершите создание функции escape\_data():
  - } // завершение функции escape\_data()
- 9. Сохраните файл.

В принципе, использование закрывающего тега PHP в данном случае необязательно. Если применяется закрывающий тег, после которого случайно вставляется пробел или пустая строка, то в результате включения этого файла с помощью других сценариев браузеру отсылаются заголовки. Если сценарий, включающий файл (родительский сценарий), позднее пытается отправить куки-файл, запускает сеанс или перенаправляет браузер, то на экране появляется сообщение об ошибке "Headers already sent" (заголовки уже отправлены). Подобную ситуацию можно предотвратить, отказавшись от применения закрывающего тега (по крайней мере, в том случае, если файл включается с помощью сценария). Также отказ от закрывающего тега PHP (по мере возможности) способствует повышению быстродействия кода.



#### Совет

Файл тузql.inc.php, включает дополнительный код, рассматриваемый в следующих главах. Этот файл можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-0-3219-4936-3.html

# ФАЙЛ КОНФИГУРАЦИИ

Рассматриваемый в этом разделе сценарий PHP создает файл конфигурации. Подобно сценарию mysql.inc.php, этот сценарий может применяться любым другим сценарием на сайте. В то время как на некоторых сайтах имеются страницы, которые не требуют доступа к базе данных, сценарии PHP, находящиеся на любом сайте, используют файл конфигурации. Назначение этого файла следующее:

- хранение легко изменяемых системных настроек;
- создание полезных констант, которые могут применяться разными сценариями;
- запуск сеанса;
- задание порядка обработки ошибок.

Начнем с определения файла конфигурации, в который позднее будет добавлен дополнительный код.

- 1. С помощью текстового редактора или в среде IDE создайте новый сценарий PHP, присвойте ему имя config.inc.php и сохраните в папке includes (см. рис. 3.3).
- 2. Задайте значения констант LIVE и CONTACT\_EMAIL.

```
<?php
if (!defined('LIVE')) DEFINE('LIVE', false);
DEFINE('CONTACT EMAIL', 'you@example.com');
```

Наиболее важной настройкой является LIVE, поскольку именно она определяет порядок обработки ошибок. В зависимости от используемого платежного шлюза, значение этой константы может также применяться для переключения между тестированием системы обработки платежей до фактического использования этой системы. Сказанное не относится к платежной системе PayPal, имеющей свои особенности. С помощью константы CONTACT\_EMAIL задается адрес электронной почты, используемый для получения сообщений электронной почты при эксплуатации сайта. Эта константа может также применяться для форм контактов либо для указания других адресов электронной почты.

С помощью инструкции IF, находящейся в первой строке кода, можно выбрать настройку LIVE для единственной страницы. Если, например, нужно отладить отдельную страницу, не затрагивая весь сайт, то воспользуйтесь следующим кодом.

```
DEFINE('LIVE', true);
require('./includes/config.inc.php');
```

#### 3. Присвойте значения другим константам.

```
define ('BASE_URI', '/путь/к/папке/');
define ('BASE_URL', 'www.example.com/');
define ('MYSQL', BASE URI . 'mysql.inc.php');
```

Это первые три константы, которые будут использованы сайтом. Измените значения этих констант в соответствии с выбранной средой. Первая константа задает родительскую папку корневого каталога сайта (при наличии подобной папки). При этом применяется ссылка на папку, находящуюся на сервере, на котором смонтирована файловая система. На Macintosh эта выглядит так: /Users/<*ваше-имя*>/Sites/ еcom1/. В Windows эта ссылка принимает вид C:\xampp\htdocs\ecom1\.

Если настроить BASE\_URI в соответствии со структурой, показанной на рис. 3.2, то ссылка BASE\_URI . pdfs применяется для хранения PDF-файлов, а BASE\_URI. mysql . inc.php указывает местоположение сценария, задающего подключение к базе данных MySQL. Если невозможно получить доступ к папкам, находящимся на более высоком уровне, чем корневая папка сайта, то присвойте BASE\_URI название самой папки сайта. Как показано на рис. 3.3, PDF-документы хранятся в папке, которая задается с помощью ссылки BASE\_URI. 'includes/pdfs'.

Вторая константа задает базовую URL-ссылку на сайт, не включающую название протокола, например www.example.com/. Другими словами, эта ссылка представляет собой основную часть адреса сайта, с помощью которой можно получить доступ к сайту в окне браузера. Часть http:// была устранена, поскольку некоторые вебстраницы используют префикс https://.

Третья константа указывает на только что созданный сценарий подключения MySQL. С помощью ссылки BASE URI облегчается создание полного пути к сайту. Например,

если ссылка включает значения, относящиеся к нескольким сценариям, находящимся в нескольких папках, то довольно сложно последовательно поддерживать корректные ссылки. Ну а абсолютные ссылки всегда будут корректны. Во-вторых, при глобальных изменениях сайта (выбор другого доменного имени или хостинга) достаточно изменить значения ссылок. Например, если сайт разрабатывается на вашем собственном компьютере, то значение ссылки BASE\_URL может быть localhost/ ecom1/, а после ввода сайта в эксплуатацию — www.example.org/.

#### Примечание

84

Ссылки BASE URI и BASE URL завершаются символом косой черты.

4. С помощью следующей строки кода откройте сеанс:

session start();

Для отслеживания подключенных к сайту пользователей используются сеансы. Поскольку каждая страница сайта требует файла конфигурации, следует запустить сеанс, чтобы каждая страница могла получать доступ к данным сеанса. При необходимости настроить сеанс (например, название сеанса или длительность) выполните эти операции в одном месте.

С другой стороны, сеансы запускаются даже в тех случаях, когда они не требуются, например, во время просмотра пользователем начальной страницы либо страницы, содержащей перечень контента (без выполнения дальнейших действий).

5. Начните создавать функцию обработки ошибок.

```
function my_error_handler($e_number, $e_message, $e_file,
%$e line, $e vars) {
```

В PHP можно создавать собственные функции, предназначенные для обработки ошибок. С помощью этих функций можно управлять отображением сообщений об ошибках, задавать порядок обработки ошибок и отображаемые сведения об ошибках. В случае возникновения ошибки PHP либо генерирования подобной ошибки с помощью функции trigger\_error() функция обработки ошибок вызывается сценарием, который будет создан позднее. Этот пользовательский обработчик ошибок не совместим с ошибками синтаксического анализа (парсинга) и другими серьезными ошибками PHP. В случае возникновения подобных ошибок выполнение этого сценария будет невозможным.

Количество аргументов функции обработки ошибок может варьироваться от двух до пяти. В рассматриваемом случае используются все пять аргументов. Первый аргумент представляет числовой идентификатор ошибки, которому могут присваиваться такие значения, как 2. Этот идентификатор представляет сообщение об ошибке  $E\_WARNING$ . Второй аргумент соответствует полученному сообщению об ошибке. Следующий аргумент представляет имя файла, в котором произошла ошибка, а четвертый аргумент — строку с ошибкой. Пятый аргумент представляет массив переменных, существовавших в тот момент, когда произошла ошибка. Вся эта информация значительно облегчает процесс отладки.



#### Примечание

Можно создать обработчик исключений, предназначенный для обработки исключений, генерируемых кодом или включенной библиотекой. 6. Начните создавать детализированное сообщение об ошибке.

```
$message = "Ошибка произошла в сценарии '$e_file', в строке
%$e line:\n$e message\n";
```

Детализированное сообщение об ошибке начинается с названия файла (в кавычках), номера строки с ошибкой и строки сообщения.

7. Добавьте данные обратной трассировки:

```
$message .= "" .print r(debug backtrace(), 1) . "\n";
```

В процессе выполнения *обратной трассировки* контролируются все действия, происходящие до момента появления ошибки. Отслеживаются выполняемые файлы, вызванные функции, аргументы, переданные функциям, и созданные переменные. Чтобы получить информацию обратной трассировки, вызовите функцию debug\_ backtrace(), которая возвращает массив результатов. Чтобы добавить массив в сообщение об ошибке, передайте ero (либо вызов функции, создающей массив) в качестве первого аргумента функции print\_r(), а затем передайте 1 либо true в качестве второго значения аргумента функции print\_r(). Если в качестве второго аргумента функции print\_r() указывается положительное число, то осуществляется возврат значения вместо его вывода на печать. Чтобы облегчить чтение кода, заключите его в теги предварительного форматирования HTML: cpre>.... Если не требуется выполнение детализированной обратной трассировки, то можно добавить список переменных и значений в сообщение:

```
$message .= "" . print r ($e vars, 1) . "\n";
```

8. Если сайт не функционирует, то в окне браузера будет выведено сообщение об ошибке.

```
if (!LIVE) {
    echo '<div class="alert alert-danger">' . nl2br($message)
    %. '</div>';
```

Если сайт не работает, то нужно тут же известить пользователя о возникшей проблеме. Сообщение об ошибке заключается в теги <DIV> и выводится на печать с помощью классов alert и alert-danger. (Эти классы соответствуют классам, заданным в файле CSS.)

Чтобы включить символы новой строки (\n) в HTML-теги, выполняющие разбиение текста, воспользуйтесь функцией nl2br(). На рис. 3.4 показана первая часть примера детализированного сообщения об ошибке, отображаемого в окне браузера.

An error oc Jndefined	curred in script '/Users/larryullman/Sites/ex1/html/index.php' on line 27: /ariable: var
Array	
C	
[0]	=> Array
	C
	[file] => /Users/larryullman/Sites/ex1/html/index.php
	[line] => 27
	[function] => my_error_handler

Рис. 3.4. Часть сообщения об ошибке, выводимого в окне браузера

 Если сайт функционирует, то отсылается сообщение электронной почты, включающее сообщение об ошибке.

```
} else {
    error_log($message, 1, CONTACT_EMAIL,
    %'From:admin@example.com');
```

Функция error\_log() может регистрировать ошибки разными способами. Первый аргумент этой функции представляет собой само сообщение об ошибке. Второй аргумент представляет тип получателя сообщений об ошибках. Значение 1 задает пересылку сообщения об ошибке по электронной почте. (Если используется заданное по умолчанию значение 0, то сообщение об ошибке регистрируется в системном журнале.) Третий аргумент задает получателя сообщения. Если сообщение об ошибке отправляется по электронной почте, то этот аргумент задает электронный адрес "Кому". Четвертый аргумент предназначен исключительно для отправки сообщений электронной почты и применяется для добавления произвольных дополнительных заголовков сообщений, таких как электронный адрес "От".

**10.** Если сайт находится в рабочем режиме и ошибка не является предупреждением, то отображается обобщенное сообщение об ошибке.

```
if ($e_number != E_NOTICE) {
    echo '<div class="alert alert-danger">Произошла системная
    %ошибка. Приносим извинения за доставленные неудобства.</div>';
}
```

Если сайт работает в обычном режиме, то вместо детализированного сообщения об ошибке отобразится маловразумительный отклик. Некоторые сообщения об ошибках свидетельствуют не о реальных проблемах, а представляют собой некие технические отчеты, которые не отражаются на работоспособности сайта. Подобные отчеты трактуются как *предупреждения*. Фактически сообщение об ошибке, приведенное на рис. 3.4, не появилось бы, если бы сайт работал в обычном режиме. На рис. 3.5 показано сообщение, отображаемое для пользователя при возникновении настоящей ошибки.

Произошла системная ошибка. Приносим извинения за доставленные неудобства.

Рис. 3.5. Сообщение о системной ошибке



86

#### Примечание

Если файл является полностью работоспособным и всесторонне протестирован, то существует большая вероятность того, что посетители вообще не увидят сообщений об ошибках, даже относящихся к категории предупреждений.

**11.** Завершите создание функции my error handler().

```
} // завершение условного выражения $live IF-ELSE return true;
```

} // завершение определения функции my\_error\_handler()

Если функция обработки ошибок возвращает значение, которое отличается от false, значит, ошибка была обработана. Если функция отображает ложное значение, то будет также вызван заданный по умолчанию обработчик ошибок PHP (эту операцию выполнять не нужно для реально функционирующего сайта).

12. Вызовите обработчик ошибок:

set error handler('my error handler');

Эта строка указывает PHP на то, что следует воспользоваться пользовательской функцией для выполнения обработки ошибок. Если вы не вызываете эту функцию, то PHP по-прежнему может использовать заданный по умолчанию браузер. Поэтому ошибка синтаксического разбора не обрабатывается пользовательским обработчиком ошибок и приводит к прекращению выполнения сценария PHP.

13. Сохраните файл.

# **ШАБЛОН НТМL**

Чтобы создать браузерную сторону сайта, начните с разработки одного или нескольких шаблонов HTML, которые формируют элегантный внешний вид, способствующий привлечению внимания покупателей виртуального контента. В качестве шаблона была выбрана структура "Sticky footer with fixed navbar" (examples.getbootstrap.com), относящаяся к фреймворку Twitter Bootstrap (getbootstrap.com). Эта структура была изменена в соответствии с потребностями сайта. Соответствующий файл, template.html, можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-0-3219-4936-3.html



#### Совет

Можете использовать собственный дизайн либо изменить сгенерированную тему сайта в соответствии со своими потребностями.

Если пользователь не зарегистрировался на сайте, то в левой части экрана выводится форма регистрации (рис. 3.6). Если пользователь зарегистрирован на сайте, то форма регистрации не отображается, но в верхней части страницы появляются ссылки, используемые для управления учетными записями (рис. 3.7). Если зарегистрированный пользователь является администратором, то на экране появляется дополнительная ссылка, открывающая доступ к параметрам администратора (рис. 3.8).

Контент	Добро пожаловать				
Безопасность JavaScript	Добро пожаловать на сайт Знание - сила, на котором публикуется самая				
Безопасность РНР	современная информация о безопасности в Интернете и программировании. Да, да,				
Безопасность базы данных	да. Конечно, конечно, конечно.				
Безопасность в Интернете	Lorem ipsum dolor sil amet, consectetur adipiscing elit. Praesent consectetur voludpat nunc, eget vulputate quam tristique si amet. Dones suscipti molis rent in egestas. Morti di rous quam. Sed vita erat eu torfor tempos consequat. Morti quam massa, vierra sed ulamcoper si amet, uttrices ulamcoper eros. Nauris uttricis thoncus leo, a velvicui as en condimentum vel. Morto varus nutrum laoreet. Maecenas vitae turpis turpis. Class aglent tacil sociosya ad litora torquent per consulta nostra, per inceptos himenaeos. Fusce leo turpis functios et concequitat eget adjectar qui turpis. Checi adjectar de la concentrative advectar purus in imperdiet. Vivanus at nulla diam, sagittis rutrum diam. Integer porta imperdiet eusimod.				
Общие атаки					
Справочники в формате PDF					
Due -	Lorem Ipsum				
Бход	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent consectetur volutpat nunc, eget vulputate quam tristique sit amet. Donec suscipit mollis erat in egestas. Morbi id risus quam. Sed vitae erat eu tortor tempus consequat. Morbi quam massa, viverra sed				
	ullamcorper sit amet, ultrices ullamcorper eros. Mauris ultricles rhoncus leo, ac vehicula sem condimentum vel. Morbi varius rutrum laoreet. Maecenas vitae turpis turpis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.				
	Fusce leo turpis, faucibus et consequat eget, adipiscing ut turpis. Donec lacinia sodales nulla nec pellentesque. Fusce fringilla dictum purus in imperdiet. Vivamus at nulla diam, sagittis rutrum diam. Integer porta imperdiet euismod.				
Забыли пароль?					
Bxon →					

Рис. 3.6. Зарегистрируйтесь на сайте



Рис. 3.7. С помощью этого меню можно управлять учетными записями



Рис. 3.8. Для администратора открывается доступ к дополнительным параметрам



#### Совет

Подробную документацию по фреймворку Twitter Bootstrap можно найти в Интернете.

Контент сайта упорядочен по категориям, названия которых отображаются в левой части окна. Посетители сайта, которые не оплатили доступ к контенту, смогут лишь просмотреть список доступных статей (названия и краткое описание). Если же пользователь оплатил доступ к контенту, то он получит полный доступ к тексту выбранной статьи.

Чтобы применить HTML-шаблон к каждой странице сайта, воспользуйтесь стандартной методикой, предусматривающей разбиение шаблона на файл заголовка и файл футера. Каждая страница включает заголовок, затем отображает специфичный для страницы контент, а потом — футер. После завершения создания базового шаблона (либо шаблонов) можно приступать к генерированию отдельных файлов.



#### Совет

Если при создании сайта применяется несколько шаблонов, то сформируйте несколько файлов заголовков и футеров, а затем включите требуемые файлы на каждую страницу.

### Создание заголовка

HTML-страница начинается с заголовка. В него включаются применяемые таблицы стилей CSS и код, относящийся к основной части страницы (до области, в которой начинается контент, специфичный для страницы).

 В окне текстового редактора или в среде IDE откройте заранее созданный файл шаблона, если вы до сих пор этого не сделали. Завершенный файл template.html, с помощью которого формируется дизайн сайта, можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-0-3219-4936-3.html

 Скопируйте код HTML из файла шаблона, от первой строки до строки, задающей специфичный для страницы контент.

В рассматриваемом случае заголовок начинается с определения типа документа:

<!DOCTYPE html>

Последняя строка файла заголовка создает элемент DIV со значением класса col-9 (сетка шириной 9 столбцов).

<div class="col-9"> <!-- KOHTEHT -->

- 3. Создайте новый файл, присвойте ему имя header.html и сохраните в папке includes.
- 4. Вставьте скопированный код.
- 5. Сохраните файл.

### Добавление динамических областей в заголовок

Чтобы добавить динамические свойства, внедрите в HTML-шаблон соответствующие фрагменты кода PHP. Благодаря динамическим средствам можно оперативно изменять внешний вид и функции, доступные на страницах сайта. В следующем перечне перечислены динамические области заголовка:

- название страницы, отображаемое в окне браузера;
- выделенная область, находящаяся в верхней навигационной ссылке (см. рис. 3.6);
- два дополнительных раскрывающихся меню (см. рис. 3.7 и 3.8);
- специфические элементы в списке контента;
- форма регистрации.

Рассмотрим изменения, которые нужно внести в шаблон, чтобы получить требуемый динамический эффект.

**1.** В файл header.html добавьте инструкцию if-else, с помощью которой будет генерироваться название страницы.

```
<title><?php if (isset($page_title)) {
    echo $page_title;
} else {
    echo 'Знание - сила, но за все нужно платить';
}
?></title>
```

Заголовок страницы, отображающийся в верхней части окна браузера, на панели закладок и на панели истории, должен изменяться. Это изменение реализовано на основании значения переменной  $page_title$ , доступной в файле заголовка. В соответствии с правилами хорошего программирования не следует полагаться на то, что этой переменной всегда присвоено значение. Поэтому сначала проверяется наличие значения этой переменной. Если результат проверки положителен, то в качестве заголовка страницы используется значение этой переменной. Если же переменной  $page_title$  значение не присвоено, то отображается заголовок, заданный по умолчанию.

2. Удалите элементы списка, формирующие верхние навигационные вкладки.

```
class="active"><a href="index.php">Домой</a><a href="about.php">О наc</a><a href="contact.php">Контакты</a><a href="register.php">Регистрация</a>
```

К навигационной вкладке текущей просматриваемой страницы применен специфический класс CSS (в данном примере — active), который изменяет ее внешний вид. Для создания подобного эффекта можно воспользоваться кодом JavaScript, но если вы хотите обеспечить совместимость со всеми браузерами, то лучше обратиться к коду PHP.

3. Вместо элементов списка в шаблоне HTML начните блок кода PHP:

<?php

4. Создайте массив, элементами которого являются страницы сайта.

```
$pages = array (
    'Домой' => 'index.php',
    'O нас' => '#',
    'Контакты' => '#',
    'Регистрация' => 'register.php'
);
```

Этот массив включает основные навигационные элементы, которые содержат текст, отображаемый на соответствующей вкладке. Значение каждого элемента соответствует определенной странице (т.е. странице, с которой связана вкладка).



#### Примечание

Некоторые страницы, связанные с заголовком и футером, в этой книге создаваться не будут. Поскольку создание подобных страниц не представляет особой сложности, выполните эту операцию самостоятельно.

5. Идентифицируйте текущую просматриваемую страницу:

```
$this page = basename($ SERVER['PHP SELF']);
```

Чтобы динамически применить класс к текущей странице, сценарий должен получить информацию о текущей странице и идентифицировать изменение значения PHP, присвоенного переменной \$\_SERVER['PHP\_SELF']. Если пользователь просматривает страницу, доступную по ссылке http://www.example.com/папка/ file.php, то переменная \$\_SERVER['PHP\_SELF'] получает значение сценария / лапка/file.php. Чтобы получить часть сценария file.php, реализующего подобное поведение, примените функцию basename(). 6. Выполните циклический обход каждой страницы.

```
foreach ($pages as $k => $v) {
    echo '<li';</pre>
```

Этот цикл предназначен для обхода каждого элемента массива. Для создания навигационных вкладок применяются переменные цикла \$k и \$v. Первая строка кода в цикле начинает создавать новый элемент списка HTML.

7. Если выбрана текущая страница, добавьте следующий класс:

```
if ($this page == $v) echo ' class="active"';
```

Для создания ссылки на текущую страницу используется следующий код HTML:

.



#### Примечание

Условные выражения рекомендуется заключать в фигурные скобки. Но если отказаться от использования фигурных скобок (например, для экономии места в книге), тогда нужно поместить условное выражение в одну строку (см. п. 7).

8. Завершите элемент списка, который был начат в п. 6.

```
echo '><a href="' . $v . '">' . $k . '</a>
```

Сначала закрывается открывающий тег Затем создается ссылка на соответствующую страницу, представленная переменной v. Отображаемый текст заключается в теги SPAN, затем завершаются теги link и list item. Чтобы добавить символ новой строки в HTML-код, создающий страницу, оператор echo завершается в следующей строке (рис. 3.9).

```
<a href="index.php">Домой</a>
<a href="#"><0 наc</a>
<a href="#">Контакты</a>
class="active"><a href="register.php">Регистрация</a>
```

#### Рис. 3.9. Код HTML-страницы

- 9. Завершите цикл foreach и блок кода PHP:
  - } // конец цикла FOREACH
- **10.** Затем начините условный блок if, отображающий меню для зарегистрированных пользователей:

```
if (isset($ SESSION['user id'])) {
```

Эта страница получает информацию о том, что пользователь зарегистрирован, если присвоено соответствующее значение переменной \$ SESSION['user id'].



#### Совет

Если вы собираетесь исследовать HTML-код, отформатируйте его соответствующим образом с помощью сценария PHP.

**11.** С помощью кода РНР добавьте пункты меню, предназначенные для работы с учетными записями.

```
echo '
<a href="#" class="dropdown-toggle" datatoggle="
dropdown">Учетная запись<b class="caret"></b></a>
<a href="logout.php">Выход</a>
<a href="renew.php">Обновить</a>
<a href="renew.php">Обновить</a>
<a href="change_password.php">Изменить пароль</a>
<a href="favorites.php">Избранное</a>
<a href="favorites.php">Уизбранное</a>
<a href="recommendations.php">Рекомендации</a>
```

Этот код PHP создает те же самые ссылки, что и оригинальный шаблон. Связанные страницы будут созданы в следующих главах. Большая часть HTML-кода заимствована из шаблона Twitter Bootstrap и применяется для создания раскрывающихся меню.

12. С помощью следующего кода создаются пункты меню, доступные администратору.

```
if (isset($_SESSION['user_admin'])) {
    echo '
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
    %Admin <b class="caret"></b></a>

        <a href="add_page.php">Добавить страницу</a>
        <a href="add_pdf.php">Добавить PDF-документ</a>
        <a href="#"><Другое действие</a>

        </l
```

Если зарегистрированный пользователь является администратором, то в его распоряжении окажутся дополнительные средства. Приведенный выше код отображает вторую панель ссылок на боковой панели. Сценарии add\_page.php и add\_pdf. php будут созданы в главе 5.

```
13. Завершите условное выражение $ SESSION['user_id'].
```

} ?>

92

В данном случае закрывающий тег PHP является необходимым, поскольку закрывает блок PHP, находящийся в большом блоке HTML.

**14.** Позднее в файле вместо статического кода HTML будут генерироваться динамические ссылки.

```
<?php
$q = 'SELECT * FROM categories ORDER BY category';
$r = mysqli_query($dbc, $q);
while (list($id, $category) = mysqli_fetch_array($r,
%MYSQLI_NUM)) {
    echo '<a href="category.php?id=' . $id . '"
    %class="list-group-item" title="' . $category . '">' .
    %htmlspecialchars($category) . '
```

```
</a>';
}
?>
```

На этом создание базового кода PHP и MySQL завершено. В результате выполнения запроса отображаются результаты, а для каждой возвращаемой записи создается один элемент списка. Ссылки вместе с идентификатором категории передаются в URL-ссылке, которая будет использована сценарием category.php (об этом будет говориться в главе 5).

Чтобы устранить вероятность осуществления XSS-атаки, к категории применяется функция htmlspecialchars(). (Поскольку столбец id представлен целым числом, по отношению к нему эту функцию можно не применять.)

Ссылка на PDF-страницу отделена от других ссылок, поскольку она не представляет категорию контента в базе данных.



#### Примечание

Если категории контента не будут часто изменяться, тогда жестко закодируйте ссылки на категории с помощью HTML-кода. Это позволит отказаться от создания дополнительных запросов к базе данных.

**15.** После списка ссылок на контент включите форму регистрации, которая отобразится в том случае, если пользователь не зарегистрирован на сайте.

```
<?php
if (!isset($_SESSION['user_id'])) {
    require('includes/login_form.inc.php');
}
?>
```

Форма регистрации будет создана в следующей главе.

 Сохраните файл. На этом создание динамического заголовка завершается.

# Создание футера

Файл футера включается после контента, специфического для страницы. Этот файл создает элементы бокового меню и футер страницы (например, символ авторского права и другую подобную информацию), а также завершает HTML-страницу.

- 1. С помощью текстового редактора или в среде IDE откройте выбранный файл шаблона.
- Начиная со строки, завершающей контент страницы, скопируйте код HTML из файла шаблона.

```
<!-Завершение контента -->
</div><!--/col-9-->
</div><!--/row-->
</div><!--/container-->
</div><!--/wrap-->
<div id="footer">
<div class="container">
<div class="container">
<div class="text-muted credit"><span class="pull-left">
<a href="site_map.php">Kapta caйтa</a> | <a href="</pre>
```

```
"policies.php">Политики</a></span> <span class=
"pull-right">© Знание - сила, 2013</span>
</div>
</div>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/
jquery.min.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Код футера закрывает большинство имеющихся тегов <DIV>, добавляет футер и включает два файла JavaScript. Первый файл JavaScript представляет собой библиотеку jQuery, которая поддерживается сетью доставки контента Google (Content Distribution Network — CDN). Второй файл представляет шаблон Twitter Bootstrap. Оба файла применяются для создания раскрывающегося меню (как и многие другие динамические компоненты фреймворка Twitter Bootstrap).

- **3.** Создайте новый файл, присвойте ему имя footer.html и сохраните в папке includes.
- 4. Вставьте скопированный код.
- 5. Сохраните файл.

94

### Создание начальной страницы

Чтобы объединить все компоненты, нужно создать начальную страницу, которая формируется на основе комбинирования четырех файлов (конфигурации, заголовка, подключения к серверу MySQL и футера).

- 1. С помощью текстового редактора или в среде IDE создайте новый сценарий PHP, присвойте ему имя index.php и сохраните в корневой папке сайта.
- 2. С помощью ключевого слова require подключите файл конфигурации:

```
require('./includes/config.inc.php');
```

С помощью файла конфигурации задаются системные настройки, обрабатываются ошибки и запускается сеанс, поэтому этот файл всегда должен быть первым фрагментом кода (незакомментированным), используемого для создания страниц.

 С помощью строки, включающей ключевое слово require, задайте подключение к базе данных:

require (MYSQL);

Для добавления сценария подключения к базе данных можно использовать ссылку на константу MYSQL, определенную в файле конфигурации. Даже если изменить название или местоположение сценария mysql.inc.php, то достаточно заменить всего лишь одну строку в файле конфигурации, и все страницы будут корректно включать этот сценарий.

Файл подключения к базе данных включается с помощью ключевого слова require, находящегося перед строкой файла заголовка, который запускает запрос.



#### Совет

При использовании ключевых слов require и include круглые скобки не требуются, хотя в книге эти скобки будут встречаться. **4.** С помощью ключевого слова include подключите файл заголовка:

include('./includes/header.html');

#### 5. Создайте контент, специфичный для страницы.

?><h3>Добро пожаловать</h3>

Добро пожаловать на сайт Знание – сила, на котором публикуется самая современная информация о безопасности в Интернете и программировании. Да, да, да. Конечно, конечно, конечно.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Praesent consectetur volutpat nunc, eget vulputate quam tristique sit amet. Donec suscipit mollis mollis erat in egestas. Morbi id risus quam. Sed vitae erat eu tortor tempus consequat. Morbi quam massa, viverra sed ullamcorper sit amet, ultrices ullamcorper eros. Mauris ultricies rhoncus leo, ac vehicula sem condimentum vel. Morbi varius rutrum laoreet. Maecenas vitae turpis turpis. Class arlent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaous. Fusce leo turpis, faucibus et concequat eget, adipiscing ut turpis. Donec lacinia sodates nulla nec pellentesque. Fusce fringilla dictum purus in imperdiet. Vivamus at nulla diam, saggitis rutrum diam. Integer porta imperdiet euismod.

#### 6. Добавьте футер.

```
<?php
include('./includes/footer.html');
?>
```

#### 7. Сохраните файл.



#### Совет

В файле index.php находится дополнительный код, который будет рассмотрен в главе 4. Этот файл можно загрузить по следующему адресу:

http://www.williamspublishing.com/Books/978-0-3219-4936-3.html

- 8. Откройте файл в окне браузера и обратите внимание на полученный результат. Поскольку в данном случае идет речь о коде PHP, для запуска сценария index.php на выполнение используется URL-ссылка вида http://иmo-либо), а не ссылка типа file://.
- **9.** Чтобы увидеть, как изменится страница после входа обычного пользователя, после строки, включающей файл конфигурации, добавьте следующую строку кода:

```
$ SESSION['user id'] = 1;
```

**10.** Чтобы увидеть, как изменится страница после входа администратора, после строки, включающей файл конфигурации, добавьте следующую строку кода:

```
$_SESSION['user_admin'] = true;
```

04 ch03.indd 95

# ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

Прежде чем приступить к рассмотрению основных сценариев в следующей главе, нужно определить две вспомогательные функции. Первая функция выполняет перенаправление браузера, избавляя пользователя от необходимости соответствовать определенным требованиям для доступа к выбранной странице. Вторая функция существенно облегчает обработку некоторых форм сайта.

Обратите внимание на следующие преимущества, обеспечиваемые вспомогательными функциями:

- разбиение на фрагменты сложной логики для улучшения восприятия кода;
- повторное использование одной и той же логики;
- возможность быстрого изменения кода.

Последние два преимущества являются ключевыми. Благодаря разделению логики отпадает необходимость в повторном использовании кода. И если в дальнейшем вы решите внести изменения в код, то сможете легко выполнить эту операцию.

## Перенаправление браузера

Первая вспомогательная функция будет применяться для ограничения доступа к некоторым страницам. Например, доступ к нескольким общедоступным страницам открывается для обычных пользователей, а к двум страницам администрирования — только администраторами. Если пользователь не соответствует критериям страницы, то браузер будет перенаправлен на другую страницу, а текущая страница закроется (рис. 3.10). Если этот процесс реализован в виде функции, то произвольная страница, требующая авторизации, будет вызывать лишь данную функцию (без какой-либо дополнительной логики).



Рис. 3.10. Процесс перенаправления браузера

После включения файла конфигурации (config.inc.php) в любой сценарий, выполняющийся на сайте, можно определить первую вспомогательную функцию.

```
function redirect_invalid_user($check = 'user_id', $destination =
%'index.php', $protocol = 'http://') {
    if (!isset($ SESSION[$check])) {
```

```
$url = $protocol . BASE_URL . $destination;
header("Расположение: $url");
exit();
}
```



}

#### Совет

На сайте, применяющем куки-файлы, та же самая функция может верифицировать пользователя с помощью константы \$ COOKIE (вместо \$ SESSION).

Эта функция принимает три аргумента, которые являются необязательными. Первый аргумент представляет собой элемент массива сеанса, применяемый для выполнения проверки (по умолчанию — user\_id). Другими словами, если значение \$\_SESSION['user\_id'] не установлено, то пользователь не зарегистрирован на сайте и не может просматривать страницу. В главе 5 эта же функция применяется для ограничения доступа к сценариям администраторам.

Второй аргумент функции представляет собой страницу, на которую будет перенаправляться пользователь. По умолчанию это будет начальная страница, хотя можно перенаправить пользователя на страницу регистрации или любую другую страницу, изменив передаваемое этой функции значение (в качестве второго аргумента).

Третий аргумент представляет собой применяемый протокол; по умолчанию используется протокол http://. Поскольку в рассматриваемом примере используется именно этот аргумент, пользователи могут перенаправляться на SSL-страницы или на страницы, не поддерживающие протокол SSL.

При выполнении этой функции проверяется значение переменной сеанса с помощью условного выражения. Если этой переменной значение не присвоено, то URL-ссылка, используемая для перенаправления, формируется путем конкатенации протокола и целевого адреса. Результат выполнения этой операции присваивается константе BASE\_URL (также определяется в файле конфигурации). Затем выполняется реальное перенаправление путем вызова функции header(). И наконец, функция exit(), которая технически является конструкцией языка, прекращает выполнение сценария, вызвавшего эту функцию. Подобное действие необходимо, поскольку PHP будет продолжать вызывать на выполнение сценарий после вызова функции header(), даже если перенаправление браузера было завершено.

В процессе создания сценария регистрации (см. главу 4) следует учитывать, каким образом работает функция перенаправления. Вообще говоря, авторизация основана на факте присваивания значения, а не на величине этого значения. Например, зарегистрированному пользователю присуще произвольное значение user\_id, а администратору — произвольное значение user\_admin.

Эта функция должна вызываться незамедлительно после включения файла конфигурации (рис. 3.10). Она не проверяет, что заголовки еще не отправлены, и в результате перенаправление браузера не происходит. Чтобы выполнить подобную проверку, воспользуйтесь функцией headers\_sent(), включив ее в условное выражение, находящееся в составе текущей функции. Если возвращается значение false, то происходит перенаправление пользователя. Если же возвращается значение true, то включаются заголовок и футер и выводится сообщение об ошибке.

```
if (!headers_sent()) {
    // Код перенаправления
} else {
```

```
include_once('./includes/header.html');
trigger_error('У вас нет разрешений на доступ к этой странице.
%Войдите на сайт и попробуйте еще раз.');
include_once('./includes/footer.html');
```



98

#### Совет

В данном блоке кода использовалась функция include\_once() вместо функции include(), поскольку заголовки уже были отправлены. Это возможно, поскольку файл заголовка был включен до строки, выполняющей вызов функции.

### Создание форм ввода данных

Практически все функции сценариев, которые будут рассмотрены в следующей главе, основаны на применении форм. С помощью форм пользователь может выполнять регистрацию, входить на сайт, изменять пароль либо восстанавливать забытый пароль. Все эти формы используют три типа вводимых данных: текст, пароль и адрес электронной почты (не учитывая кнопку отправки данных). Поле, используемое для ввода данных, создается с помощью следующего простого кода HTML:

```
<input type="type" name="name" id="name">
```

#### Например:

<input type="type" name="username" id="username">

Если предпринимается попытка передачи данных после некорректного ввода данных в поля формы, то эта форма должна вновь отобразиться на экране.



### Совет

Во фреймворке Twitter Bootstrap поля формы имеют дополнительные свойства (например, классы), как показано в следующем примере кода.

Для удобства ввода данных форма должна запоминать введенные значения (эти значения "прилипают" к экрану). Чтобы добиться подобного эффекта, к коду, формирующему поля формы, нужно добавить параметр value="любое значение". Для добавления этого параметра можно воспользоваться следующим кодом PHP.

```
<input type="text" name="username" id="username"
value="<?php echo $_POST['username']; ?>">
```

Если форма загружается впервые, то переменной *\$\_POST['username']* значение не присвоено, поэтому код изменится следующим образом.

```
<input type="text" name="username" id="username"
value="<?php if (isset($_POST['username']))
%echo $_POST['username']; ?>">
```

Совет

При желании можно изменять размеры полей ввода данных. Если этого не сделать, то выбирается заданный по умолчанию размер поля, задаваемый браузером.

Если пользователь в силу каких-либо причин использует кавычки для выделения значения, то это может привести к нарушению корректной работы кода HTML. На рис. 3.11 представлен результат ввода пользователем имени Алекс "Шеф" Петров.

```
Желательное имя пользователя
Алекс
```

#### Рис. 3.11. Результат некорректной обработки вводимых данных

Чтобы исключить некорректную обработку вводимых данных, можно воспользоваться функцией htmlspecialchars () (рис. 3.12 и 3.13).

```
<input type="text" name="username" id="username"
value="<?php if (isset($_POST['username']))
%echo htmlspecialchars($_POST['username']); ?>">
```

Желательное имя пользователя

Алекс "Шеф" Петров

#### Рис. 3.12. Результат корректной обработки вводимых данных

<label for="username" class="control-label">желательное имя пользователя</label> <input type="text" name="usernane" id="username" class="form-control" value="Алекс &quot;Шеф&quot; Петров">

#### Рис. 3.13. Код, выполняющий обработку вводимых данных

Если включены "волшебные кавычки", то значение обрабатывается с помощью функции stripslashes(). Соответствующий код будет добавлен позднее, ну а пока что рассмотрим еще одну возможную проблему. Если поля формы заполнены некорректно, то было бы неплохо отобразить красную рамку. Для этого в форму ввода нужно добавить соответствующий класс CSS. Следующий код выполняет эту задачу (с помощью фреймворка Twitter Bootstrap, версия 3).

```
<div class="form-group<?php if (/* ошибка ввода */)

%echo ' class="has-error"'; ?>">

<label for="username" class="control-label">Желательное

%имя пользователя</label>

<input type="text" name="username" id="username"

%class="form-control" value="<?php if (isset($_POST['username']))

%echo htmlspecialchars($_POST['username']); ?>">

<span class="help-block">Пожалуйста, введите желательное

%имя!</span>

</div>
```

Также в этом случае ниже поля ввода данных должно отображаться сообщение об ошибке (рис. 3.14).

```
Желательное имя пользователя
Алекс *Шеф* Петров
Пожалуйста, введите имя, используя только буквы и цифры!
```

Рис. 3.14. Сообщение, отображаемое в случае ошибки при вводе данных

04\_ch03.indd 99

```
<div class="form-group<?php if (/* ошибка ввода данных */)
%echo ' has-error'; ?>">
<label for="username" class="control-label">Желательное имя/label>
<input type="text" name="username" id="username"
class="form-control" value="<?php if (isset($_POST['username']))
echo htmlspecialchars($_POST['username']); ?>">
<?php if (/* ошибка ввода данных */) echo '<span class=
"help-block">Пожалуйста, введите имя, используя только буквы
и цифры!</span>'; ?>
</div>
```

Обратите внимание на то, что для обработки вводимых данных и ошибок требуется большой объем кода даже без использования "волшебных кавычек". Приведенный выше пример кода выглядит довольно неаккуратно, к тому же, если вы соберетесь использовать его несколько раз либо захотите полностью переделать, у вас это может занять целый день. Вместо того чтобы выполнять эти операции вручную (для каждого поля ввода данных), можно создать функцию, выполняющую эти действия в автоматическом режиме. Рассматриваемые в главе 5 формы также включают текстовые области, для обработки которых может использоваться данная функция.



#### Примечание

В имени файла используется слово *functions*, даже если определена только одна функция. Это связано с тем, что позднее будут добавлены другие функции.

- 1. С помощью текстового редактора или в среде IDE создайте новый файл PHP и присвойте ему имя form\_functions.inc.php. Этот файл будет находиться в папке includes.
- 2. Начните создавать функцию.

```
<?php
function create_form_input($name, $type, $label = '',
%$errors = array(), $options = array()) {</pre>
```

Эта функция принимает пять аргументов, из которых лишь два являются обязательными. Первый аргумент представляет собой имя, присваиваемое элементу. Второй аргумент задает тип элемента: text, password, email (в этой главе) и textarea (в следующей главе). Третий аргумент представляет значение надписи поля. Четвертый аргумент задает массив ошибок. Пятый аргумент позволяет передавать функции дополнительные параметры. Например, с помощью этого аргумента можно передать значение заполнителя.

3. Проверьте значение и выполните его обработку.

При выполнении этой функции сначала предполагается, что значение отсутствует. Если же существует значение для данных, вводимых с помощью переменной \$\_POST, то оно присваивается переменной \$value.

Затем удаляются избыточные символы косой черты, присвоенные значению при включенных "волшебных кавычках".

При выполнении этой функции предполагается, что форма использует действие POST. Можно создать другой аргумент, который принимает действия POST либо GET, а затем проверяет наличие соответствующей суперглобальной области действия значения, которая позволит сделать функцию более гибкой.



#### Совет

Условные выражения рекомендуется записывать в нескольких строках и заключать в фигурные скобки. В этой же книге некоторые условные выражения записываются в одной строке для экономии места.

**4.** Создайте теги <DIV>, в которые заключается текущий элемент.

```
echo '<div class="form-group';
if (array_key_exists($name, $errors)) echo ' has-error';
echo '">';
```

Этот код строго привязан к синтаксису, применяемому для создания элементов формы с помощью фреймворка Twitter Bootstrap 3. Каждый элемент вместе с соответствующей меткой, относящийся к данному фреймворку, заключен в теги <DIV> вместе с классом form-group. Все они находятся в первой строке кода.

Для обработки ошибок, связанных с созданным элементом, нужно добавить класс has-error. В случае вызова функции массив ошибок передается параметру \$errors. Этот массив будет включать ошибки формы, проиндексированные названием поля ввода данных. (Подобные ошибки могут появляться при выполнении сценариев, обрабатывающих формы.) Следовательно, если массив включает элемент с тем же именем, что и название поля ввода данных, то класс has-error должен быть добавлен в область действия элемента DIV.

Если подобный элемент массива отсутствует, то ввод завершается без выполнения дополнительной стилизации класса (последняя строка кода).

5. Создайте надпись (если они нужна).

```
if (!empty($label)) echo '<label for="' . $name . '"
%class="control-label">' . $label . '</label>';
```

Надпись необязательна, но если она поддерживается, то будет создана с помощью этого кода. При этом используется синтаксис, рекомендуемый фреймворком Twitter Bootstrap 3.

6. Создайте условное выражение, применяемое для проверки типа поля ввода данных.

```
if ( ($type === 'text') || ($type === 'password') ||
%($type === 'email')) {
```

Эта функция будет создавать поля ввода текста, пароля, адреса электронной почты и текстовые области. Первые три типа полей имеют практически одинаковый синтаксис (за исключением значения type, применяемого в коде HTML). Выполнение функции начинается с обработки этих трех типов полей.

7. Начните создавать поля ввода данных.

```
echo '<input type="' . $type . '" name="' . $name . '"
%id="' . $name . '" class="form-control";</pre>
```

Это начальная оболочка HTML-кода, применяемого для поля ввода данных. Она включает свойства type, name, id и class.



#### Примечание

При создании шаблона используется HTML5, в котором определен тип поля ввода данных *email*.

8. Добавьте значение поля ввода данных (при наличии):

if (\$value) echo ' value="' . htmlspecialchars(\$value) . '"';

Значение, присвоенное переменной \$value, будет добавлено в поток ввода данных с помощью функции htmlspecialchars().

#### 9. Проверьте наличие дополнительных параметров.

```
if (!empty($options) && is_array($options)) {
   foreach ($options as $k => $v) {
      echo " $k=\"$v\"";
   }
}
```

В результате выполнения этих действий появляется возможность добавить заполнители или другие атрибуты элемента.

10. Завершите элемент:

echo '>';

11. Отобразите сообщение об ошибке (в случае появления ошибки).

```
if (array_key_exists($name, $errors)) echo '<span
%class="help-block">' . $errors[$name] . '</span>';
```

Этот код сначала проверяет значение переменной Serrors, чтобы идентифицировать ошибку. При наличии ошибки соответствующее сообщение добавляется после поля ввода данных (см. рис. 3.14).

**12.** Проверьте, будет ли тип вводимых данных textarea:

```
} elseif ($type === 'textarea') {
```

Синтаксис, применяемый при создании областей ввода текста, отличается от синтаксиса, используемого для создания других типов вводимых данных. Поэтому текстовые области генерируются особым образом с помощью этой функции.

13. Сначала отобразите сообщение об ошибке.

```
if (array_key_exists($name, $errors)) echo '<span class=
%"help-block">' . $errors[$name] . '</span>';
```

В отличие от ввода текста и паролей, когда сообщение об ошибке отображается ниже вводимых данных, при вводе данных в текстовые области сообщения об ошибках отображаются выше области. Благодаря этому ошибку легче заметить.

14. Начните создавать область ввода текста.

```
echo '<textarea name="' . $name . '" id="' . $name . '"
class="form-control"';</pre>
```

В результате выполнения этого кода создается открывающий тег текстовой области, поддерживающий динамические значения name и id.

15. Проверьте наличие дополнительных параметров.

```
if (!empty($options) && is_array($options)) {
  foreach ($options as $k => $v) {
```

echo " \$k=\"\$v\""; }

С помощью подобных параметров можно, например, установить количество строк и столбцов.

16. Закройте открывающий тег:

echo '>';

}

17. Добавьте значение в текстовую область:

if (\$value) echo \$value;

Значение для текстовых областей записывается между открывающим и закрывающим тегами текстовой области. В п. 16 закрывается открывающий тег, а в п. 18 будет создан закрывающий тег. В результате выводится на печать значение.

18. Завершите создание области ввода текста:

echo '</textarea>';

19. Завершите формирование функции.

```
} // завершение основной конструкции IF-ELSE
echo '</div>';
```

```
} // завершение функции create_form_input()
```

Обязательно закройте тег <DIV>, который включает весь элемент (начал определяться в п. 4).

20. Сохраните файл.

И снова в силу описанных выше причин закрывающий тег РНР не применяется.