ВИКТОР ПЕТИН

bhv

Сайт на AJAX под ключ

Готовое решение для интернет-магазина

САЙТЫ БЕЗ ПЕРЕЗАГРУЗКИ СТРАНИЦЫ

ГОТОВОЕ РЕШЕНИЕ: ИНТЕРНЕТ-МАГАЗИН

PHP

xajax и jQuery

AJAX и Smarty

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+CD

УДК 681.3.06
ББК 32.973.26-018.2
П29

Петин В. А.

П29 Сайт на AJAX под ключ. Готовое решение для интернет-магазина. — СПб.: БХВ-Петербург, 2011. — 432 с.: ил. + CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0629-8

Описана разработка высокоинтерактивных Web-сайтов, основанных на передовой технологии AJAX, работающих без перезагрузки страниц и обладающих функциональностью настольных приложений. Обучение построено на сквозном примере создания с нуля готового решения: интернет-магазина цифровых товаров, а также системы его администрирования. При этом использован язык PHP, фреймворки xajax и jQuery, шаблонизатор Smarty и другие популярные технологии динамического формирования контента. Разработанный сайт создан полностью по технологии AJAX и готов к размещению в сети.

Прилагаемый компакт-диск содержит исходные коды описанного в книге интернет-магазина, а также бесплатные программы для создания и отладки сайтов на локальной машине.

Для Web-разработчиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 26.10.10.
Формат 70×100 $^{1}/_{16}$. Печать офсетная. Усл. печ. л. 34,83.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 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 978-5-9775-0629-8

Оглавление

Глава 1

Инструменты создания Web-страниц

Основа программирования документов для Web — язык разметки HTML — позволяет создавать только статические страницы, обновляемые с сервера. В отличие от обычного HTML, динамический HTML (DHTML) обеспечивает взаимодействие Web-документов с пользователем и дает возможность изменять документ на компьютере клиента без обращения на сервер. Инструментом для манипулирования страницами на компьютере клиента служат языки сценариев JavaScript и VBScript, из которых в настоящее время наиболее популярен JavaScript. Однако для создания по-настоящему динамических Web-приложений (взаимодействие с посетителями, получение от них информации, настройка страниц под конкретного пользователя и т. д.) необходимо взаимодействие страниц с сервером. Было создано несколько серверных языков для написания сценариев на стороне сервера и формирования динамических страниц. PHP — один из самых успешных таких языков — быстро нашел свое применение и приобрел большую популярность. При разработке Web-приложений нам понадобится сервер баз данных. В этой главе рассмотрим один из наиболее подходящих для нас — MySQL. Существенно облегчит вашу работу отладочный пакет Денвер, описанный в последнем разделе.

1.1. HTML и CSS

Всемирная "паутина" основана на языке гипертекстовой разметки HTML. HTML не вполне обычный язык: он не относится к языкам высокого уровня. Это язык разметки, и код, написанный на нем, исполняется на компьютере клиента в приложении Web-браузера. Web-страница, загружаемая в браузер, представляет собой HTML-файл. Для просмотра HTML-кода документа щелкните правой кнопкой мыши в окне документа и в появившемся контекстном меню выберите в зависимости от браузера команду **Просмотр HTML-кода** или **Исходный код страницы**.

1.1.1. Теговая модель

Разметка HTML-документов выполняется с помощью тегов, которые записывают с соблюдением определенных правил. Теговая модель предполагает разбиение документа на отдельные фрагменты, которые заключаются в теги или начинаются тегом. Все теги начинаются с открывающейся угловой скобки <, за которой следует

текст, определяющий содержание тега. Оканчивается тег закрывающейся угловой скобкой >. Содержанием тега может быть просто его имя, либо имя и набор атрибутов тега.

Большинство тегов парные, для каждого начального тега <ИМЯ> есть конечный тег </ИМЯ>, например:

```
<TABLE>. . .</TABLE>
<FORM>. . .</FORM>
```

Многоточие означает, что между начальным и конечным тегами может находиться текст или другие теги.

Кроме парных, существуют одиночные теги, в которых имеется только открывающий тег. В соответствии с инструкциями одиночных тегов браузер выполняет определенные действия, например:

- <P> — формирование нового абзаца;
- <HR> — вставка горизонтальной линии;
- <IMG> — вставка изображения.

1.1.2. Элементы HTML

Документ HTML включает в себя элементы, которые представляют абзацы, заголовки, списки, таблицы, гиперссылки, рисунки и пр. Весь документ можно рассматривать как совокупность определенных элементов. Элемент — это пара тегов и символьные данные (код или текст), заключенные между ними. Иначе говоря, элемент состоит из начального тега, содержимого и конечного тега. В некоторых элементах конечный тег может быть опущен (в случае одиночных тегов). Элементы HTML не чувствительны к регистру символов, т. е. браузер одинаково воспринимает теги <Table>, <TABLE>, <table>. Список элементов HTML утвержден спецификацией HTML 4.01. Если браузер находит незнакомый элемент, он его просто игнорирует.

1.1.3. Классификация элементов HTML

Все элементы, предусмотренные в HTML, можно условно разбить на несколько категорий:

- структурные — обязательны для документа, соответствующего стандарту HTML (например, HTML, HEAD, BODY, TITLE);
- блоковые — предназначены для форматирования целых текстовых блоков (например, DIV, H1, H2, PRE);
- текстовые — создают разметку текста (EM, STRONG) и разметку шрифта (I, B, U);
- специальные — элементы пустой строки (BR, HR), ссылка A, внедренные элементы (IMG, EMBED, OBJECT), элементы формы (INPUT, SELECT, TEXTAREA), элементы таблицы (TABLE, TR, TD) и др.

1.1.4. Атрибуты тегов

Часто теги, помимо имени, содержат дополнительные элементы, которые называют атрибутами:

```
<H1 id="zagolovok" color="red" ></H1>
```

Атрибут записывается после имени тега перед закрывающейся скобкой > и состоит из пары `имя атрибута=значение`. В теге может быть несколько атрибутов. Атрибуты отделяют друг от друга пробелами, очередность записи атрибутов в теге не имеет значения. Атрибуты записываются в начальных тегах и отсутствуют в конечных. Одно из главных назначений атрибутов — управлять видом элемента на странице положением, цветом.

1.1.5. Листы стилей CSS

Каскадные листы (таблицы) стилей (CSS) — это язык, используемый в документах HTML для определения способа представления содержимого. Представление задается с помощью стилей, помещаемых непосредственно в элементы HTML, заголовок HTML-документа или отдельные таблицы стилей. В листах стилей значение свойства присоединяется к стилю при помощи двоеточия:

```
background-color : green
font-size: 12 pt
```

1.1.5.1. Определение встроенного стиля

Простейший способ задания стиля элемента HTML с помощью атрибута `style`:

```
<div style="font-size:12pt;color:green;background-color:yellow">
```

Введение стиля мало чем отличается от форматирования средствами HTML. В то же время встроенный стиль — простейший способ задания стилевых свойств, который можно оперативно применить к отдельным элементам в процессе создания документа. При этом нужно понимать, что встроенные стили нарушают основную концепцию CSS, заключающуюся в том, что форматирование документа должно быть отделено от содержания.

1.1.5.2. Формирование листа стилей

Стилевые свойства вводят с помощью определения стиля, которое принято обозначать фигурными скобками:

```
{ font-family:Arial; background-color:yellow }
{visibility:hidden}
```

Назначение стиля тому или иному элементу состоит в установлении связи:

```
span {color:red;font-style:italic}
div.p {text-size:12 px; margin-left: 10 px}
#div1 {border-color:green}
```

Элемент, к которому относится определенный стиль, называется селектором. Селекторы, записанные прописными буквами, обозначают классы, а селекторы, начинающиеся со знака #, отвечают уникальным идентификаторам элементов.

1.1.5.3. Внутренние листы стилей

Встроенные стили имеют большой недостаток — они не позволяют отделить средства форматирования документа от его содержания. Кроме того, объявления встроенного стиля приходится повторять для каждого форматируемого элемента на протяжении всего документа. От этих недостатков свободен другой способ введения стилей — размещение листа стилей в заголовочной части документов. Согласно этому способу, называемому заголовочным стилем, можно единым образом управлять содержимым всего документа. Для изменения отображения одинаково оформленных элементов достаточно один раз изменить соответствующие свойства в листе стилей. Встроенные и заголовочные стили относятся к внутренним листам стилей. Для введения заголовочного стиля в заголовочную часть HTML-документа вставляется специальный контейнер `<style></style>`:

```
<style type="text/css">
<!--  описание листа стилей  -- >
</style>
```

Пример

```
<head>
<style="text/css">
h4.style1 {color:red}
#style2 {color:green;background-color:yellow}
</style>
</head>
```

Сопоставление правил CSS с конкретными элементами документа выполняется с помощью атрибутов `class` и `id`:

```
<body>
<h4 class="style1">Заголовок 1</h4>
<div id="style2">Текст 1</div>
</body>
```

1.1.5.4. Внешние листы стилей

Внешние листы стилей записываются в отдельных файлах и применяются для оформления набора HTML-документов. Использование внешних листов стилей позволяет единым образом оформлять множество Web-страниц и даже сайтов. Удобство внешних стилей заключается также и в том, что можно изменять стили, не затрагивая содержания документов. Описание стилей хранится в отдельном файле, который имеет расширение css. Содержательная часть CSS-файла состоит только из листа стилей. Основным инструментом подключения к HTML-документу

внешних листов стилей является одиночный тег <LINK>, который располагается в заголовочной части <HEAD>:

```
<HEAD>
<LINK type="text/css" href="http://www.my_site.ru/css/
site.css" rel="stylesheet" ">
</HEAD>
```

где:

- type="text/css — указывает браузеру, что применяется текст формата CSS;
- href — задает URL файла внешнего листа стилей;
- rel="stylesheet" — указывает на то, что элемент LINK устанавливает связь с внешним листом стилей.

1.2. Язык сценариев JavaScript

JavaScript — это один из основных языков разработки Web-страниц, который поддерживают все популярные браузеры. Для просмотра Web-страниц, содержащих инструкции JavaScript, пользователю не нужно устанавливать дополнительное программное обеспечение. Язык JavaScript разработан компанией Netscape Communications и является языком сценариев. Этот язык призван был расширить возможности HTML по переработке информации из форм и добавлению динамики на Web-страницы. JavaScript вначале был задуман как клиентский язык, предназначенный для работы на компьютере клиента-пользователя. Идея создания JavaScript заключалась именно в возможности размещения на Web-страницах исполняемого содержимого, благодаря чему можно было бы выйти за рамки статического HTML, обеспечить взаимодействие с пользователем, управление браузером и т. д. Однако по мере своего развития JavaScript вышел за рамки отдельно взятого браузера и стал выполнять также функции серверной части.

1.2.1. Встраивание сценария JavaScript в документ

На стороне клиента сценарий содержится в HTML-файле. Код сценария заключен между тегами <SCRIPT> и </SCRIPT>, которые могут быть размещены в любом месте HTML-документа вслед за тегами <HEAD> и <BODY>. В документе может быть несколько сценариев, ограниченных тегами <SCRIPT> и </SCRIPT>, причем эти фрагменты не должны перекрываться. Сценарии будут исполняться в порядке их расположения в документе. Функции исполняются при обращении к ним обработчиков событий или при вызове из других функций. В теге <SCRIPT> обязательно нужно указывать язык сценария:

```
<SCRIPT language="JavaScript">
<!--  Операторы языка JavaScript -- >
</SCRIPT>
```

Возможны следующие случаи размещения сценария в HTML-документе:

- в теле программы (между тегами <BODY>), в этом случае сценарий исполняется при загрузке страницы в браузер;
- в заголовке документа (между тегами <HEAD>), в этом случае сценарий не исполняется сразу при загрузке, а может использоваться как функция другими сценариями;
- между тегами HTML (между угловыми скобками < …>), при этом сценарий является обработчиком событий, для его записи не требуются теги <SCRIPT>;
- в отдельном файле. Язык JavaScript допускает создание собственных файлов с расширением js, в которых размещаются сценарии:

```
<SCRIPT language="JavaScript" src="js/jquery-1.4.2.js">
```

1.2.2. Обработка событий в JavaScript

Очень важное место в программировании Web-страниц занимают события. События генерируются в результате действий пользователя (щелчков мыши, нажатия клавиш и пр.). Разрабатывая Web-страницы, можно составить сценарий таким образом, что страница будет реагировать на некоторые из событий. Это делается с помощью специальных программ, которые называются обработчиками событий. Программы обработчиков событий представляют собой фрагменты кода и обычно оформляются в виде функций. Обработчик событий, написанный на JavaScript, вводится в сценарий просто — буквально одной строкой, например:

```
<input name=input1 type=text onclick="alert(this.value);">
<input name=input2 type=text onchange="function1(this.value)">
```

В табл. 1.1 приведены события и элементы HTML, в которых эти события могут происходить.

Таблица 1.1. *События и объекты*

Обработчик события	Поддерживающие HTML-элементы	Описание
onAbort	IMG	Прерывание загрузки изображения
onBlur	A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA	Потеря текущим элементом фокуса, т. е. переход к другому элементу. Возникает при щелчке мышью вне элемента либо при нажатии клавиши табуляции
onChange	INPUT, SELECT, TEXTAREA	Изменение значений элементов формы. Возникает после потерей элементом фокуса, т. е. после события blur
onClick	Практически все HTML-элементы	Одинарный щелчок (нажата и отпущена кнопка мыши)
onDblClick	То же	Двойной щелчок

Таблица 1.1 *(окончание)*

Обработчик события	Поддерживающие HTML-элементы	Описание
onError	IMG, WINDOW	Возникновение ошибки выполнения сценария
onFocus	A, AREA, BUTTON, INPUT, LABEL, SELECT, TEXTAREA	Получение элементом фокуса (щелчок мышью на элементе или очередное нажатие клавиши табуляции)
onKeyDown	Практически все HTML-элементы	Нажата клавиша на клавиатуре
onKeyPress	То же	Нажата и отпущена клавиша на клавиатуре
onKeyUp	То же	Отпущена клавиша на клавиатуре
onLoad	BODY, FRAMESET	Закончена загрузка документа
onMouseDown	Практически все HTML-элементы	Нажата кнопка мыши в пределах текущего элемента
onMouseMove	То же	Перемещение курсора мыши в пределах текущего элемента
onMouseOut	То же	Курсор мыши выведен за пределы текущего элемента
onMouseOver	То же	Курсор мыши наведен на текущий элемент
onMouseUp	То же	Отпущена кнопка мыши в пределах текущего элемента
onMove	WINDOW	Перемещение окна
onReset	FORM	Сброс данных формы (щелчок по кнопке <input type="reset">)
onResize	WINDOW	Изменение размеров окна
onSelect	INPUT, TEXTAREA	Выделение текста в текущем элементе
onSubmit	FORM	Отправка данных формы (щелчок на кнопке <input type="submit">)
onUnload	BODY, FRAMESET	Попытка закрытия окна браузера и выгрузки документа

1.3. Динамический HTML

Консорциум W3C (**http://www.w3.org/dom**) определяет DHTML как спецификацию открытой объектной модели, которая обеспечивает полный доступ к документу и позволяет свободно манипулировать всем документом и его содержимым. Все элементы документа являются программируемыми объектами, управляемыми событиями мыши и клавиатуры. Благодаря DHTML такие операции, как добавле-

ние содержимого, изменение какой-либо части Web-страницы, не требуют обращения к серверу и перезагрузки страницы. DHTML расширяет возможности традиционного HTML, ориентированного в основном на оформление страниц, позволяет создавать страницы, которые могут в интерактивном режиме взаимодействовать с пользователем.

Одной из особенностей языка JavaScript является то, что на стороне клиента язык интегрирован с функциями браузера. Достигается это благодаря тому, что объектная модель браузера строится по принципу совместимости с объектной моделью JavaScript.

Все объекты браузера организованы в иерархическую структуру (рис. 1.1). Поскольку основные функции браузера реализуются в окне приложения, в котором отображается сам HTML-документ, центральным объектом иерархии является окно браузера. Оно представляется объектом `window`. Все другие объекты HTML рассматриваются как свойства этого объекта. На основе `window` можно определить объекты, свойства и методы, необходимые для полноценной работы с документами. Объекту `window` подчинены объекты следующего уровня, которые можно разделить на две группы:

- объекты браузера, предоставляющие доступ к свойствам, методам и событиям, происходящим в окне браузера:
  - `events`;
  - `history`;
  - `location`;
  - `navigator`;
  - `screen`;
- объекты документа и фреймов, позволяющие управлять элементами документов и фреймов, загруженных в браузер:
  - `document`;
  - `frames`.

Вторая группа объектов вместе с содержащимися в ней свойствами, методами и событиями образует объектную модель документа DOM (Document Object Model). Это все, что пользователь видит в окне документа: текст, ссылки, рисунки и т. д.

Для эффективного управления содержимым блоков HTML-страниц и их оформлением необходимо хорошо представлять себе иерархию объектов объектной модели. Вверху иерархии расположен самый старший класс `window`. Атрибуты и свойства этого класса относятся, как правило, ко всему окну в целом. Доступ к подчиненным классам и далее к семействам, элементам и атрибутам элементов осуществляется через dot-нотацию, т. е. через точку, как во многих объектно-ориентированных языках, например:

```
window.document.forms
// все элементы семейства форм документа
```

Внутри объекту присваивается порядковый номер и может быть присвоено имя (идентификатор). Нумерация начинается с нуля (листинг 1.1).

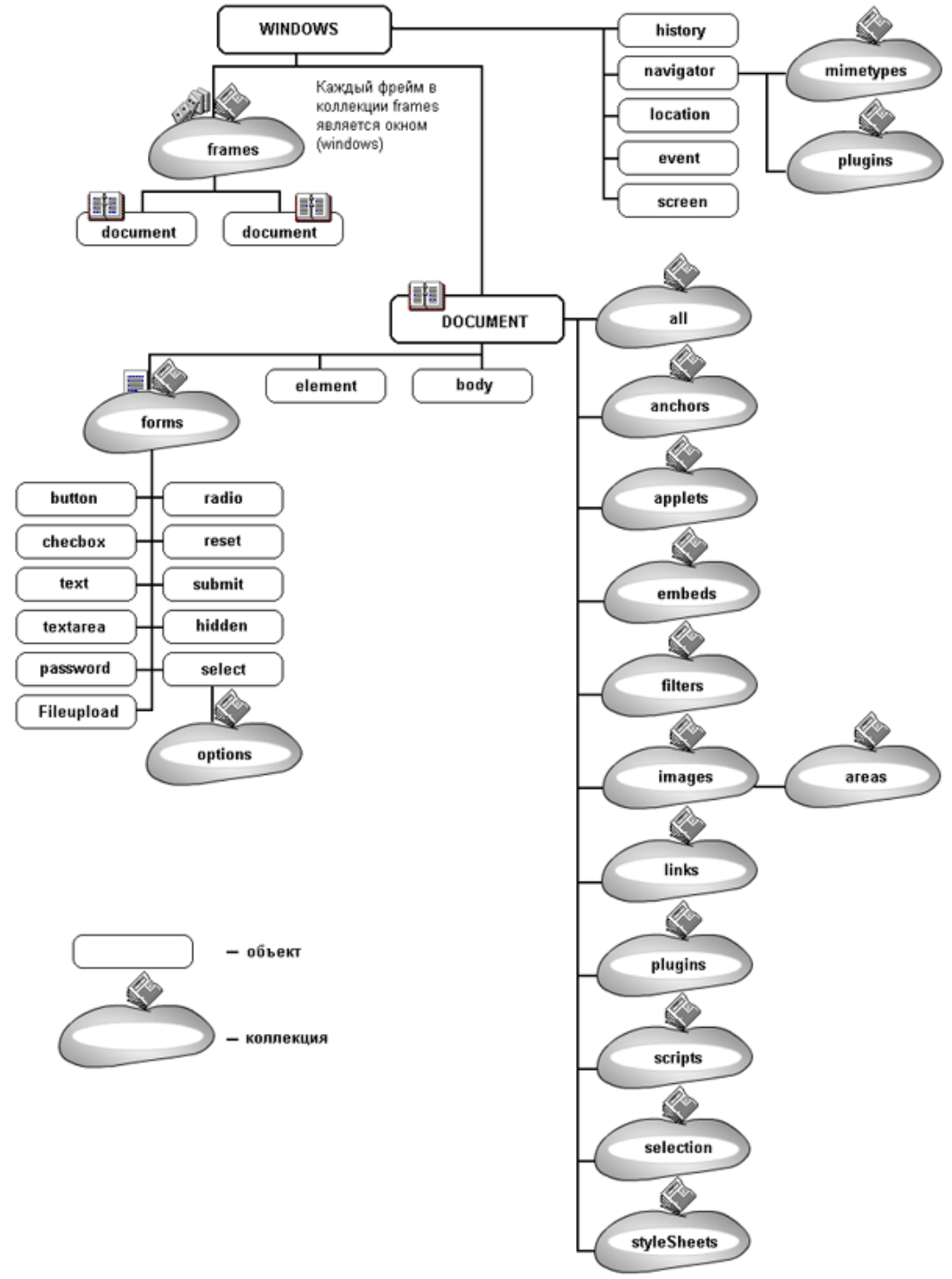

Рис. 1.1. Объектная модель

Листинг 1.1

```
window.document.forms[0]
// первая форма семейства форм
window.document.forms.form1
// форма семейства форм form1
window.document.forms.form1.elements.elements[1]
// второй элемент формы form1
window.document.forms.form1.elements.input1
// элемент input формы form1
```

Для изменения атрибутов HTML-элементов, необходимо указать путь к этому элементу и установить значение необходимого атрибута:

```
window.document.forms.form1.elements.input1.value="Значение 1";
window.document.forms.form1.elements.input1.style.color="red";
window.document.forms.form1.elements.input1.size=10;
```

Если элементу присвоен уникальный идентификатор (ID), доступ к элементу можно получить через ID:

```
window.document.getElementById("div1").style.display="none";
window.document.getElementById("div1").style.color="green";
```

В записи через dot-нотацию допускается опускать `window`, при этом запись будет относиться к текущему окну.

1.4. PHP — серверный язык программирования

PHP (Hypertext Preprocessor — препроцессор гипертекста) — это широко используемый язык сценариев общего назначения с открытым исходным кодом. PHP специально разработан для написания Web-приложений, исполняющихся на Web-сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и Perl. Он очень похож на C и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить Web-мастер, не знающий пока других языков программирования.

Огромным преимуществом PHP, в отличие, например, от JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. PHP не зависит от быстродействия компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML-файл или результат выполнения скрипта.

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в HTML-код страницы.

PHP способен генерировать и преобразовывать не только HTML-документы, но и изображения разных форматов (JPEG, GIF, PNG), файлы PDF и FLASH. PHP может формировать данные в любом текстовом формате, включая XHTML и XML.

PHP — кроссплатформенная технология. Дистрибутив PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix, Microsoft Windows, Mac OS и др. PHP поддерживается на большинстве Web-серверов, таких, как Apache, Microsoft Internet Information Server (IIS), Microsoft Personal Web Server и др.

Для большинства серверов PHP поставляется в двух вариантах: в качестве модуля и в качестве препроцессора CGI.

PHP поддерживает работу с ODBC и многими базами данных: MySQL, MSQL, Oracle, PostgreSQL, SQLite и др.

Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL — оптимальный вариант для создания интернет-сайтов различной сложности. Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков Web-программирования.

1.5. СУБД MySQL

В настоящее время ни одно серьезное Web-приложение не может обойтись без взаимодействия с базой данных, обеспечивающей разнообразные возможности при работе с данными: сортировку, поиск, преобразование, редактирование и многое другое. При этом все низкоуровневые операции с файловой системой скрыты для программиста за несложными SQL-запросами. Есть множество различных видов баз данных, но мы будем рассматривать MySQL. Почему именно MySQL? Потому что она является небольшим, очень быстрым, компактным и простым в использовании сервером баз данных, идеальным для приложений малого и среднего размера.

1.5.1. Типы данных

Типы данных, применяемые в таблицах MySQL, можно разделить на следующие группы:

- целые числа;
- дробные числа;
- строки;
- бинарные данные;
- календарные (дата и время).

1.5.1.1. Целые числа

Общий вид указания целого числового типа данных:

```
префиксINT [UNSIGNED]
```

Необязательный флаг `UNSIGNED` задает, что будет создано поле для хранения беззнаковых чисел (бóльших или равных нулю).

Типы целых числовых данных приведены в табл. 1.2.

Таблица 1.2. *Целые числовые типы*

Тип	Диапазон значений
TINYINT	–128 ... 127
SMALLINT	–32 768 ... 32 767
MEDIUMINT	–8 388 608 ... 8 388 607
INT	–2 147 483 648 ... 2 147 483 647
BIGINT	–9 223 372 036 854 775 808 ... 9 223 372 036 854 775 807

1.5.1.2. Дробные числа

MySQL поддерживает несколько типов дробных чисел (табл. 1.3).

В общем виде они записываются так:

```
ИмяТипа[(length, decimals)] [UNSIGNED]
```

где:

- `length` — количество знакомест (ширина поля), в которых будет размещено дробное число при его передаче;
- `decimals` — число знаков после десятичной точки, которые будут учитываться.

Таблица 1.3. *Дробные числовые типы*

Тип	Определение
UNSIGNED	Задает беззнаковые числа
FLOAT	Число с плавающей точкой небольшой точности
DOUBLE	Число с плавающей точкой двойной точности
REAL	Синоним для DOUBLE
DECIMAL	Дробное число, хранящееся в виде строки
NUMERIC	Синоним для DECIMAL

1.5.1.3. Строки

Строки представляют собой массивы символов. Обычно при поиске по текстовым полям по запросу `SELECT` не учитывается регистр символов, т. е. строки "Вася" и "ВАСЯ" считаются одинаковыми. Кроме того, если база данных настроена на автоматическую перекодировку текста при его помещении и извлечении, эти поля будут храниться в указанной вами кодировке. Для начала ознакомимся с типом строки, которая может хранить не более `length` символов, где `length` принадлежит диапазону от 1 до 255:

```
VARCHAR (length) [BINARY]
```

При занесении некоторого значения в поле такого типа из него автоматически вырезаются концевые пробелы. Если указан флаг BINARY, то при запросе SELECT строка будет сравниваться с учетом регистра.

Типы строковых данных приведены в табл. 1.4.

Таблица 1.4. *Строковые типы*

Тип	Максимальное число символов
VARCHAR	255
TINYTEXT	255
TEXT	65 535
MEDIUMTEXT	16 777 215
LONGTEXT	4 294 967 295

1.5.1.4. Бинарные данные

Бинарные данные почти аналогичны данным в формате TEXT, но только при поиске в них учитывается регистр символов.

Типы бинарных строковых данных приведены в табл. 1.5.

Таблица 1.5. *Бинарные типы*

Тип	Максимальное число символов
TINYBLOB	255
BLOB	65 535
MEDIUMBLOB	16 777 215
LONGBLOB	4 294 967 295

Бинарные данные не перекодируются автоматически, если при работе с установленным соединением включена возможность перекодирования текста "на лету".

1.5.1.5. Дата и время

MySQL поддерживает несколько типов полей, специально приспособленных для хранения даты и времени в различных форматах (табл. 1.6).

Таблица 1.6. *Типы дата и время*

Тип	Определение
DATE	Дата в формате ГГГГ-ММ-ДД
TIME	Время в формате ЧЧ:ММ:СС
DATETIME	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС
TIMESTAMP	Дата и время в формате timestamp. Однако при получении значения поля оно отображается не в формате timestamp, а в виде ГГГГММДДЧЧММСС, что сильно умаляет преимущества его использования в PHP

1.5.2. Таблицы MySQL

СУБД MySQL поддерживает в настоящее время несколько видов таблиц. Их можно разделить на два различных типа:

- Транзакционные:
  - InnoDB;
  - BDB.
- Без поддержки транзакций:
  - HEAP;
  - ISAM;
  - MERGE;
  - MyISAM.

Преимущества транзакционных таблиц (Transaction Safe Tables, TST):

- Высокая надежность. Даже если произойдет сбой в работе MySQL или возникнут проблемы с оборудованием, свои данные вы сможете восстановить либо методом автоматического восстановления, либо при помощи резервной копии и журнала транзакций.
- Можно сочетать несколько операторов и принимать их одной командой COMMIT.
- Внесенные изменения можно отменить командой ROLLBACK (если не установлен режим автоматической фиксации).
- Если произойдет сбой во время обновления, все изменения будут восстановлены (в нетранзакционных таблицах внесенные изменения нельзя отменить).

Преимущества таблиц без безопасных транзакций (Non Transaction Safe Tables, NTST):

- Работать с ними намного быстрее, т. к. не выполняются дополнительные транзакции.
- Для них требуется меньше дискового пространства.
- Для обновлений занято меньше памяти.

По умолчанию в MySQL принят тип таблиц MyISAM.

1.5.3. Структурированный язык запросов SQL

Структурированный язык запросов SQL позволяет выполнять различные операции с базами данных: создавать таблицы, вставлять, обновлять и удалять данные, проводить выборку данных и т. д. Рассмотрим основные операторы SQL.

- CREATE DATABASE — эта команда создает новую базу данных:

```
CREATE DATABASE [IF NOT EXIST] db_name
```

Здесь db_name — имя новой создаваемой базы.

- DROP DATABASE db_name — удаляет базу данных со всеми таблицами, входящими в ее состав; db_name — имя удаляемой базы.

- `USE db_name` — указывает MySQL, с какой базой данных вы намерены работать; `db_name` — имя выбираемой базы.
- `CREATE TABLE` — создает новую таблицу в выбранной базе данных. Синтаксис команды:

  ```
  CREATE TABLE table_name [(create_definitions,…)][table_options]
  ```

  Здесь
  - `table_name` — имя создаваемой таблицы;
  - `create_definitions` — объявление столбца, его типа и атрибутов;
  - `table_options` — тип таблицы.

  Пример:

  ```
  CREATE TABLE users (
  id INT NOT NULL AUTO_INCREMENT, login VARCHAR (32) NOT NULL ,
  password VARCHAR (12) NOT NULL , PRIMARY_KEY(id)) TYPE=MyISAM
  ```

- `DROP TABLE` — удаляет одну или несколько таблиц:

  ```
  DROP TABLE table_name1 [,table_name2,….]
  ```

  Здесь `table_name1`, `table_name2` — имена удаляемых таблиц.
- `INSERT INTO… VALUES` — вставляет новые записи в существующую таблицу базы данных:

  ```
  INSERT INTO table_name VALUES (values1,….)
  ```

  Пример:

  ```
  INSERT INTO users VALUES ("1","user1","password1")
  ```

  Порядок добавления столбцов можно устанавливать самостоятельно:

  ```
  INSERT INTO users (password,id,login) VALUES ("password2","2","user2").
  ```

- `INSERT INTO… SET` — вставляет новые записи в существующую таблицу базы данных:

  ```
  INSERT INTO table_name SET col_name1=value1[,col_name2=value2…]
  ```

  Пример:

  ```
  INSERT INTO users SET login="user3"
  ```

  Поля, не указанные в запросе, получат значения по умолчанию, поле `AUTO_INCREMENT` получит значение на единицу больше, чем последнее.
- `DELETE FROM` — удаляет записи из таблицы базы данных:

  ```
  DELETE FROM table_name [WHERE definition]
  ```

  Удаление из таблицы `table_name` данных, удовлетворяющих указанным в `definition` условиям, и возвращает число удаленных записей.

  Пример:

  ```
  DELETE FROM users  WHERE  id>3
  ```

- `SELECT` — извлекает строки данных из одной или нескольких таблиц. Синтаксис команды:

  ```
  SELECT column1,… [FROM table WHERE definition]
  [ORDER BY col_name [ASC|DESC],…][LIMIT [offset], rows]
  ```

Здесь column — имя выбираемого столбца (для всех *). WHERE — определяет условия отбора строк. ORDER BY — сортирует строки по столбцу col_name в прямом (ASC) или обратном(DESC) порядке. LIMIT — сообщает MySQL о выводе только rows запросов, начиная с позиции offset.

- UPDATE — обновляет столбцы таблицы table в соответствии с их новыми значениями в строках существующей таблицы. Синтаксис:

```
UPDATE table SET col_name1=expr1[,col_name2=expr2…]
[WHERE definition]
[LIMIT rows]
```

В выражении SET указывается, какие именно столбцы следует изменить и какие значения. В WHERE определяется, какие строки подлежат изменению. LIMIT позволяет ограничить число изменяемых строк.

1.5.4. Функции PHP для работы с MySQL

Рассмотрим основные функции API для работы с MySQL из PHP-скриптов.

1.5.4.1. *mysql_connect*

Открывает соединение с сервером MySQL и возвращает его указатель или false при неудаче. Синтаксис функции:

```
resource mysql_connect([ string $server [, string $username [,string $password]]])
```

Это урезанный вариант синтаксиса функции mysql_connect(). Здесь рассмотрены три основные строковые (string) переменные, которых обычно хватает для работы.

- $server — сокет (хост), к которому производится подключение. Значение переменной не имеет никакого отношения к домену вашего сайта. Название и порт $server зависят от настроек самого сервера. Обычно эта переменная имеет значение localhost, что можно изменить в настройках PHP;
- $username — имя пользователя владельца процесса сервера;
- $password — пароль владельца процесса сервера.

1.5.4.2. *mysql_close*

Закрывает соединение с сервером MySQL. Синтаксис функции:

```
bool mysql_close ([resource link_identifier])
```

Возвращает true в случае успешного завершения, false при возникновении ошибки. mysql_close() закрывает соединение с базой данных MySQL, на которое указывает переданный указатель. Если параметр link_identifier не указан, закрывается последнее открытое (текущее) соединение. Непостоянные соединения автоматически закрываются в конце скрипта и mysql_close() не требуется.

1.5.4.3. *mysql_select_db*

Выбирает базу данных MySQL. Синтаксис функции:

```
bool mysql_select_db (string database_name [, resource link_identifier])
```

Возвращает `true` в случае успешного завершения, `false` при возникновении ошибки. `mysql_select_db()` выбирает для работы указанную базу данных на сервере, на который ссылается переданный указатель. Если параметр указателя опущен, используется последнее открытое соединение. Если нет ни одного открытого соединения, функция попытается соединиться с сервером аналогично функции `mysql_connect()`, вызванной без параметров. Каждый последующий вызов функции `mysql_query()` будет работать с выбранной базой данных.

1.5.4.4. *mysql_query*

Посылает запрос MySQL. Синтаксис функции:

```
resource mysql_query ( string query [, resource link_identifier])
```

`mysql_query()` посылает запрос активной базе данных сервера, на который ссылается переданный указатель. Если параметр `link_identifier` опущен, используется последнее открытое соединение. Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции `mysql_connect()` без параметров. Результат запроса буферизируется. Только для запросов `SELECT`, `SHOW`, `EXPLAIN` и `DESCRIBE`, `mysql_query()` возвращает указатель на результат запроса, или `false`, если запрос не был выполнен. В остальных случаях, `mysql_query()` возвращает `true` при успешном запросе и `false` в случае ошибки. Значение, не равное `false`, свидетельствует лишь о том, что запрос был выполнен успешно, но не говорит о количестве затронутых или возвращенных рядов. Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда.

1.5.4.5. *mysql_fetch_row*

Обрабатывает ряд результата запроса и возвращает неассоциативный массив. Синтаксис функции:

```
array mysql_fetch_row (resource result)
```

Возвращает массив, содержащий данные обработанного ряда, или `false`, если рядов больше нет. `mysql_fetch_row()` обрабатывает один ряд результата, на который ссылается переданный указатель. Ряд возвращается в массиве. Каждая колонка располагается в следующей ячейке массива. Массив начинается с нулевого индекса. Последующие вызовы функции `mysql_fetch_row()` вернут следующие ряды или `false`, если рядов не осталось.

1.5.4.6. *mysql_fetch_assoc*

Обрабатывает ряд результата запроса и возвращает ассоциативный массив. Синтаксис функции:

```
array mysql_fetch_assoc (resource result)
```

Возвращает ассоциативный массив с названиями индексов, соответствующими названиям колонок или `false`, если рядов больше нет. Функция `mysql_fetch_assoc()` аналогична вызову функции `mysql_fetch_array()` со вторым параметром, равным `MYSQL_ASSOC`. Функция возвращает только ассоциативный массив. Если вам нужны как ассоциативные, так и численные индексы в массиве, обратитесь к функции