ЛАРРИ УЛЬМАН

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА PHP

- Язык для быстрой разработки Web-ресурсов
- Создание динамических сайтов
- Готовые сценарии

САМОУЧИТЕЛЬ

Peachpit Press

ДМК Пресс
Издательство

УДК 004.438PHP
ББК 32.973.26-018.2
У51

У51 **Ульман Л.**
Основы программирования на PHP: Пер. с англ. – М.: ДМК Пресс. – 288 с.: ил. (Самоучитель).

ISBN 5-94074-124-X

Представленная книга посвящена PHP – серверному межплатформенному встроенному в HTML языку написания сценариев. Рассматриваются следующие вопросы: синтаксис языка, строки и управляющие структуры, массивы и регулярные выражения, функции; описываются приемы отладки ваших сценариев. Особое внимание уделяется получению введенной в форму информации, работе с файловой системой, базами данных, cookie и др.

Включенные в состав книги приложения содержат информацию об установке и настройке Web-сервера, инсталляции языка PHP. Здесь же обсуждаются вопросы безопасности скриптов, даются ссылки на Web-ресурсы, посвященные PHP.

Книга будет полезна как начинающим Web-мастерам, которые только собираются создавать динамические сайты, так и профессиональным дизайнерам, желающим внести элементы динамики в проектируемые ими ресурсы.

ISBN 0-201-72787-0 (англ.)
ISBN 5-94074-124-X (рус.)

Содержание

Глава 1 Первые шаги с PHP

Изучение любого языка программирования всегда должно начинаться с понимания синтаксиса, ведь нарушение правил синтаксиса является распространенной причиной возникновения ошибок в коде. В связи с этим главное внимание в данной главе уделено основам языка, также сюда включены рекомендации, которые помогут избежать ошибок в будущем.

К концу главы мы успешно напишем и выполним наши первые сценарии на языке PHP.

Основы синтаксиса

Разработаем нашу первую страницу на языке PHP точно так же, как начали бы с нуля документ HTML.

Между стандартными HTML- и PHP-документами есть два основных различия. Во-первых, файлы PHP-сценария сохраняются с расширением .php (например, index.php). А во-вторых, PHP-код заключается в *тэги* `<?PHP` и `?>` для отделения кода PHP от HTML.

Тэги PHP и код HTML в первом сценарии

1. Откройте текстовый редактор SimpleText, WordPad или любой другой.
2. Выберите команду **File ➤ New** для создания нового пустого документа.
3. Напечатайте такую строку:

```
<HTML><HEAD><TITLE>First PHP Script</TITLE></HEAD><BODY>
```

Для большей наглядности можно расположить каждый элемент или группу элементов на отдельной строке.

4. На новой строке наберите `<?PHP`.

5. Нажмите клавишу **Enter** для создания новой строки и наберите символы `?>`.
6. Напечатайте `</BODY></HTML>`.
7. Выберите команду **File ➤ Save As**. В появившемся диалоговом окне выберите формат **Text Only** (или **ASCII**).
8. Определите место для сохранения сценария.
9. Сохраните сценарий как first.php (листинг 1.1).

Листинг 1.1 ▾ Основная структура HTML-документа с тэгами PHP. Все PHP-сценарии должны быть выделены специальными тэгами. Тогда сервер сможет обрабатывать то, что нужно, как PHP-код. Внутри PHP-тэгов все интерпретируется как сценарий PHP, а прочая информация посылается в браузер как стандартный код HTML.

```
1. <HTML>
2. <HEAD>
3. <TITLE>First PHP Script</TITLE>
4. </HEAD>
5. <BODY>
6. <?PHP
7. ?>
8. </BODY>
9. </HTML>
```

Узнайте у вашего провайдера, какие расширения можно использовать для PHP-документов. Мы применяем расширение .php, хотя вместо этого вы можете использовать .phtml. На серверах с третьей версией PHP по умолчанию используется расширение .php3. Расширение файла дает серверу указание, как интерпретировать файл: file.php обрабатывается модулем PHP, file.asp будет обработан как ASP, а file.html является статическим HTML-документом.

Следует также узнать у провайдера, можно ли использовать короткие тэги (`<?` и `?>` вместо `<?PHP` и `?>`) или ASP-тэги (`<%` и `%>`). Такие программы, как Macromedia Dreamweaver, лучше работают с PHP-страницами, если используются тэги ASP.

Передача данных в браузер

Теперь, когда вы создали свой первый PHP-сценарий, самое время попробовать с ним что-нибудь сделать. Как упоминалось в предисловии, PHP «говорит» серверу, какие данные посылать в браузер. Для начала мы используем функцию `phpinfo()` для печати служебной информации. При вызове данная функция пошлет в Web-браузер таблицу с полным перечнем характеристик самого сервера и установленного на этом сервере модуля PHP.

Добавление функции phpinfo

1. Откройте в текстовом редакторе сценарий first.php.
2. Установите курсор между PHP-тэгами (`<?PHP` и `?>`) и создайте новую строку, нажав клавишу **Enter**.

3. На новой строке напечатайте `phpinfo();`.
4. Сделайте иным название страницы, заменив `First` на `Test` в третьей строке HTML (листинг 1.2).
5. Сохраните сценарий как test.php.

Листинг 1.2 ▾ Так как этот файл сохраняется отдельно, мы изменили титульную строку HTML при добавлении функции `phpinfo()`.

```
1. <HTML>
2. <HEAD>
3. <TITLE>Test PHP Script</TITLE>
4. </HEAD>
5. <BODY>
6. <?PHP
7. phpinfo();
8. ?>
9. </BODY>
10. </HTML>
```

Каждая инструкция PHP-кода должна заканчиваться знаком «точка с запятой» (`;`). Пропуск этого символа – самая распространенная ошибка. Вы можете размещать несколько инструкций на одной строке, отделяя их друг от друга этим знаком. Однако для ясности программ я бы не рекомендовал этого делать.

Инструкция в PHP – это исполняемая строка кода, такая как `print()` или `phpinfo()`. Точка с запятой в конце строк означает указание выполнить команду. И наоборот, строки комментариев, PHP-тэги, управляющие структуры (условные операторы, циклы и т.п) и некоторые другие конструктивные элементы, обсуждаемые далее, не требуют использования данного знака.

Каждый из компонентов нужен, чтобы указывать обстоятельства выполнения инструкций. Тэг PHP указывает только то, что начинается PHP-код; символы комментариев поясняют текст в программе и т.п. Таким образом, точка с запятой завершает конкретное действие и не требуется для конструктивных элементов, которые создают условия.

Хорошо это или нет, PHP достаточно либерален в отношении использования разных регистров во встроенных функциях, таких как `PHPINFO()`. Конечный результат функций `PHPinfo()` и `PHPINFO()` будет одним и тем же. Во второй главе приведены такие примеры, в которых регистр играет важную роль. Кстати, в языке HTML регистр букв не имеет никакого значения.

`Phpinfo()` – пример стандартной встроенной функции PHP. Более подробная информация о функциях и их создании содержится в главе 9.

Очень удобно всегда держать под рукой копию файла test.php. Его можно использовать для проверки возможностей PHP на новом сервере или для того, чтобы узнать, какая дополнительная функциональность (базы данных, работа с GIF-изображениями и т.д) поддерживается. Файл test.php можно использовать и для

экспериментов с различными расширениями файлов. Проведя несколько таких опытов, вы узнаете, какие файлы сервер будет обрабатывать правильно, а какие – нет.

Тестирование сценария

В противоположность коду HTML, который можно протестировать на своем компьютере с помощью Web-браузера, результаты PHP-сценария удастся посмотреть только после сохранения сценария на сервере, поддерживающем технологию PHP. Если вы работаете прямо на сервере, надо лишь позаботиться о сохранении сценариев в нужном каталоге. Если же вы создаете сценарий в текстовом редакторе на домашнем компьютере, для пересылки его на сервер вам потребуется клиент FTP (протокол передачи файлов). Провайдер Web-хостинга также должен обеспечить вам доступ к FTP-серверу. Потребуется установить на компьютере клиентское приложение FTP, такое как Fetch для Macintosh или WS_FTP для Windows.

Загрузка сценария на сервер с помощью FTP

1. Запустите программу FTP-клиента.
2. Установите соединение с сервером, введя его адрес, имя пользователя и пароль, присвоенные вам провайдером (рис. 1.1).
3. Найдите каталог для HTML-страниц (обычно это www/ или htdocs/).
4. Сохраните сценарий (test.php) на сервере. (Как правило, большинство FTP приложений сохраняют переданные на сервер страницы под тем именем, которое вы использовали на своем компьютере). Если вы пользуетесь программой, позволяющей указывать имя файла, назовите его test.php.

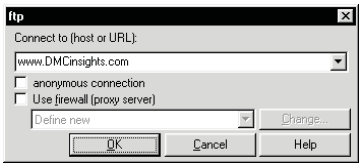

Рис. 1.1 ▼ Введите имя пользователя и пароль вашего провайдера. Если вам известен каталог, в котором следует сохранить файлы, его название также можно ввести в этом окне

Тестирование сценария в браузере

1. Откройте браузер.
2. Введите адрес сайта, на котором вы сохранили сценарий. (В моем случае это http://www.DMCinsights.com/php.)
3. Добавьте к адресу запись /test.php.
4. Нажмите клавишу **Enter**. Страница должна загрузиться в окне браузера (рис. 1.2).

Функция `phpinfo()` выводит на экран системную информацию модуля PHP, инсталированного на сервере. Полезно использовать эту функцию после

Рис. 1.2 ▼ Если вы увидите текст `phpinfo();`, то либо модуль PHP установлен некорректно, либо используемое расширение (в данном случае .php) не интерпретируется как PHP-файл

установки новой версии PHP, чтобы определить, какие расширения можно использовать и какие функции PHP поддерживаются.

Некоторые текстовые редакторы, такие как BBEdit, имеют встроенную функциональность FTP, позволяющую сохранять сценарии прямо на сервере.

Передача простого текста в браузер

Если бы PHP использовался только для просмотра конфигурации PHP на сервере, от него было бы мало толку. В основном этот язык применяется для отправки информации в браузер в виде обычного текста и HTML-тэгов. Для этого используется функция `print()`.

Печать простого сообщения

1. Откройте файл first.php в текстовом редакторе.
2. Установите курсор между PHP-тэгами и создайте новую строку, нажав клавишу **Enter**.
3. Наберите `print ("Hello, world!");` (листинг 1.3).
4. Сохраните сценарий.
5. Загрузите сценарий на сервер и проверьте результат в браузере (рис. 1.3).

Листинг 1.3 ▾ Вставив инструкцию `print` между PHP-тэгами, мы даем команду серверу послать приветствие «Hello, world!» в браузер. Это аналогично тому, что мы ввели данный текст в HTML-код.

```
1. <HTML>
2. <HEAD>
3. <TITLE>First PHP Script</TITLE>
4. </HEAD>
5. <BODY>
6. <?PHP
7. print ("Hello, world! ");
8. ?>
9. </BODY>
10. </HTML>
```

Печать фразы «Hello, world!» – первый шаг, которому учат во многих учебниках по программированию. Банально использовать для этого PHP, но я подчиняюсь традиции в демонстрационных целях.

Для отправки текста в браузер, включая `echo()` и `printf()`. Функция `echo` фактически является синонимом `print`, поэтому мы не будем рассматривать ее более подробно. О функции `printf()` говорится в главе 13.

Скобки в описании некоторых функций можно опускать, но кавычки необходимы всегда, например `print "Hello, world!";`. Хотя в книге для выделения аргументов функций мы используем скобки, многие программисты не делают этого. Я бы посоветовал вам определиться с этим вопросом и в дальнейшем придерживаться принятого решения.

Пропуск открывающих или закрывающих кавычек, или скобок, или точки с запятой после каждой инструкции – распространенная причина возникновения ошибок при использовании функции `print()`. Если при исполнении сценария возникают ошибки, прежде всего проверьте эти знаки.

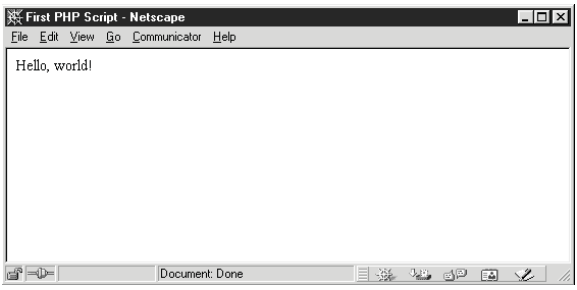

Рис. 1.3 ▾ Так будет выглядеть окно браузера, если сценарий выполнен правильно (не захватывающе, но работает)

Передача страницы HTML в браузер

Те, кто начинает изучать HTML, быстро понимают, что вид простого текста в Web-браузере оставляет желать лучшего. Действительно, язык HTML был создан для разметки простого текста. Так как HTML работает на основе добавления тэгов к тексту, мы можем использовать PHP для отправки HTML-тэгов в браузер вместе с другими нашими данными.

Передача страницы HTML в браузер с помощью PHP

1. Откройте сценарий first.php в текстовом редакторе.
2. Отредактируйте текст «Hello, world!» в строке 7, добавив тэги для выделения текста полужирным шрифтом и выровняв текст по центру

   ```
   print ("<B><CENTER>Hello, world!</CENTER></B>");
   ```

3. Загрузите сценарий (листинг 1.4) на сервер, перезагрузите страницу в браузере (рис. 1.4).

Листинг 1.4 ▼ С помощью функции `print` HTML-тэги можно вместе с текстом послать в браузер, где и произойдет форматирование.

```
1.  <HTML>
2.  <HEAD>
3.  <TITLE>First PHP Script</TITLE>
4.  </HEAD>
5.  <BODY>
6.  <?PHP
7.  print ("<B><CENTER>Hello, world!</CENTER></B>");
8.  ?>
9.  </BODY>
10. </HTML>
```

Рис. 1.4 ▼ Более привлекательная версия нашего сценария. Любой тэг HTML может быть отправлен в браузер из PHP – просто не забывайте про условные обозначения HTML (закрывающий тэг, например)

HTML-тэги, требующие кавычек (например, `<FONT COLOR="#000000">`), могут вызвать проблемы при печати из PHP, поскольку функция `print()` также использует кавычки. В таких случаях необходимо использовать обратный слеш (`\`). В нашем примере инструкция будет выглядеть следующим образом: `print "<FONT COLOR=\"#000000\">";`. После этого PHP напечатает кавычки, не интерпретируя их как начало или конец самой строки. В книге дано множество примеров подобного экранирования, представлены некоторые другие специальные символы.

Использование пробельных символов в PHP и HTML

Программисты, создающие HTML-код вручную, хорошо понимают, что использование промежутков (пробелов, пустых строк, табуляции и прочих пробельных символов) в коде помогает избежать ненужного загромождения при написании программы, но не влияет на то, что видит пользователь в окне браузера. Вставляя пустые строки между фрагментами кода, отделяя вложенные элементы табуляцией и пробелами, мы делаем сценарий более читаемым. Это облегчает программирование и последующую отладку сценария. Таким образом, разумное использование пробельных символов всячески одобряется и может применяться как в PHP, так и в полученном HTML-коде.

Вы помните, что в книге рассматриваются все три этапа, которые проходит каждое Web-приложение. Вначале это PHP-сценарий. Затем данные, посылаемые после выполнения PHP-инструкций в браузер (в основном HTML). Наконец, интерпретация и изображение этих данных в браузере клиента. Коротко остановимся на значении пробельных символов на каждом этапе.

При написании PHP-кода необходимо понимать, что промежутки обычно (но не всегда) игнорируются. Любая пустая строка или несколько таких строк подряд абсолютно не влияют на конечный результат. Табуляция и пробелы также обычно несущественны для PHP.

Код HTML без PHP (листинг 1.4, строки 1–5) может быть размещен обычным образом, как если бы вы размечали обычную HTML-страницу. Чтобы получить такую же разметку из PHP (листинг 1.4, строка 7), нужно явно использовать необходимые HTML-тэги.

Разбиение на строки PHP-кода и данных, посылаемых в браузер

1. Откройте файл first.php в текстовом редакторе.
2. Вставьте новые строки до и после команды печати с помощью клавиши **Enter**.

Новые строки служат только для придания сценарию более структурированной и ясной формы.

3. В конце команды печати (строка 8) перед кавычками добавьте символ `\n` (листинг 1.5).

4. Сохраните сценарий, загрузите его на сервер и просмотрите с помощью браузера (рис. 1.5).

Листинг 1.5 ▾ Добавление пустых строк не влияет на вид страницы в браузере, но делает код более читабельным. Для каждого символа `\n`, вставленного в инструкцию печати, в HTML-коде появится новая строка (не путать с HTML-тэгом `<BR>`, который вставляет новую строку в изображение страницы в браузере).

```
1. <HTML>
2. <HEAD>
3. <TITLE>First PHP Script</TITLE>
4. </HEAD>
5. <BODY>
6. <?PHP
7.
8. print ("<B><CENTER>Hello, world!</CENTER></B>\n");
9.
10. ?>
11. </BODY>
12. </HTML>
```

Символ `\n` посылает в браузер команду начать новую строку в HTML-коде. Можно считать, что это эквивалентно нажатию клавиши **Enter**.

Вышеописанные шаги могут сделать код PHP и HTML более читабельным, а использование промежутков не повлияет на вид страницы в браузере (рис. 1.5). Для этого необходимо использовать HTML-тэги (код для создания неразрывного пробела в браузере – ` `, эквивалент нажатия клавиши **Enter** в HTML – `<BR>`).

Дополнительные пробелы имеют значение только в инструкции печати. В результате они попадают в HTML-код, однако затем обычно игнорируются браузером.

Для просмотра кода, посланного в браузер, используйте команду **View➤Source** или **View➤Page➤Source**. Вы быстро увидите разницу между использованием

Рис. 1.5 ▾ После добавления новой пустой строки страница в браузере выглядит, как прежде, так как символ `\n` – это не HTML-тэг, а пустые строки в PHP-коде (строки 7 и 9) не пересылаются в браузер

и неиспользованием новой строки (рис. 1.6 и 1.7). Достаточно сложно оценить преимущества применения пробелов в сценарии из двенадцати строк. Но по мере увеличения и усложнения сценариев значение промежутков будет ясно видно.

Существует мнение, что код HTML нужно сжимать как можно плотнее, избегая любых лишних промежутков. Считается, что это увеличивает скорость загрузки страницы, так как ее пустые места не передаются. Хотя идея и заслуживает внимания, она не очень применима на практике и для обучения.

Рис. 1.6 ▼ Просмотр исходного текста Web-страницы – хороший способ определить, где могут возникать проблемы форматирования. Это код нашего сценария до того, как мы добавили символ `\n` и пустые строки

Рис. 1.7 ▼ Введя символ `\n` в инструкцию печати, мы отделили строку с текстом «Hello, world!» от других HTML-тэгов. При отправке более сложного HTML-кода в браузер использование новых строк помогает улучшить вид исходного текста

Добавление комментариев в сценарий

Каждый программист понимает, что делать заметки для себя – это спасательное средство, когда вы возвращаетесь к проекту для изменения, копирования или отладки фрагментов кода после большого перерыва. Использование заметок, или *комментариев*, помогает вспомнить, что вы думали в тот момент. Это не всегда просто сделать несколько месяцев спустя. При обработке сценария компьютер игнорирует комментарии, а PHP поддерживает три метода их создания.

Есть два способа закомментировать строку кода, поставив символы // или # в самое начало строки. Их можно также использовать для вставки комментария после строки PHP:

```
Print ("Hello."); // Просто приветствие.
```

Комментирование строки кода

1. Откройте сценарий first.php в текстовом редакторе.
2. Перед командой печати на строке 8 введите символ // или # (листинг 1.6).
3. Сохраните сценарий, загрузите его на сервер и просмотрите страницу с помощью браузера (рис. 1.8).

Листинг 1.6 ▼ Использование символа // или # перед строкой кода означает, что эта строка закомментирована и не будет обработана сервером.

```
1.  <HTML>
2.  <HEAD>
3.  <TITLE>First PHP Script</TITLE>
4.  </HEAD>
5.  <BODY>
6.  <?PHP
7.
8.  // print ("<B><CENTER>Hello, world!</CENTER></B>\n");
9.
10. ?>
11. </BODY>
12. </HTML>
```

Если все работает правильно, вы ничего не увидите в браузере. Не беспокойтесь, это не ошибка! Браузер не напечатал фразу «Hello, world!», так как не было команды на это действие (мы ведь закоментировали инструкцию с помощью символа // или #).

Используя обозначения /* до, а */ после сегмента кода, вы даете команду серверу проигнорировать все, что попало между этими скобками, от слова до нескольких строк.

Рис. 1.8 ▼ С закомментированной командой печати страница выглядит так, будто бы команды печати вообще не существует. Так как сервер не обрабатывает команду печати, браузер не получает текст «Hello, world!»

Комментирование нескольких строк кода

1. Откройте сценарий first.php в текстовом редакторе.
2. Удалите операторы # и // перед командой `print()`.
3. В любом месте перед командой `print()` на строке 8, но после открывающего PHP-тэга (`<?PHP`) на строке 6 наберите символ `/*`.
4. В любом месте после команды `print()` на строке 8 (после точки с запятой), но до закрывающего PHP-тэга (`?>`) на строке 10 введите символ `*/` (листинг 1.7).
5. Сохраните сценарий, загрузите его на сервер и просмотрите страницу с помощью браузера (рис. 1.9).

Листинг 1.7 ▼ Может показаться избыточным использование операторов `/* */` для комментирования одной строки кода, хотя в этом нет ничего страшного, а полученный результат тот же, что и при работе с операторами # или // (рис. 1.8 и 1.9).

```
1.  <HTML>
2.  <HEAD>
3.  <TITLE>First PHP Script</TITLE>
4.  </HEAD>
5.  <BODY>
6.  <?PHP
7.  /*
8.  print ("<B><CENTER>Hello, world!</CENTER></B\n");
9.  */
10. ?>
11. </BODY>
12. <HTML>
```

Посредством операторов `/*` и `*/` можно закомментировать как одну строку (в нашем примере), так и несколько, а операторами // и # – только одну строку.

Программисты комментируют код по-разному. Выберите понравившийся вам метод и придерживайтесь его в своей работе. Тот, кто программирует на JavaScript, по всей видимости, будет пользоваться операторами // и `/**/`. Программистам на Perl более известен символ #.

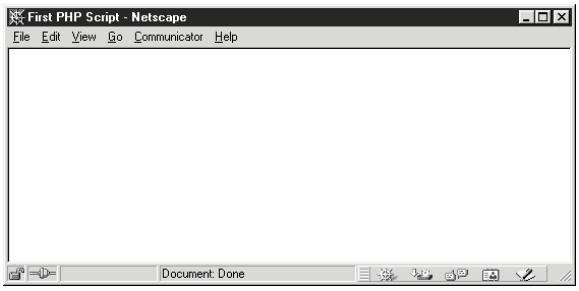

Рис. 1.9 ▼ Неважно, какие операторы используются для комментирования сценария, главное – использовать их правильно: // и # – для одной строки, а `/*` и `*/` – для любого количества строк. (Обещаю, это последний рисунок, на котором абсолютно ничего не изображено!)

Обратите внимание на то, что для комментариев кода в PHP нельзя использовать символы комментариев HTML (<!-- и -->). PHP можно использовать для печати этих элементов на странице, в этом случае комментарий появится в исходном тексте HTML на клиентском компьютере (но не в окне браузера). Комментарии PHP никогда не видны на мониторе пользователя.

Так как закомментированный в PHP текст не пересылается в браузер (рис. 1.10), это хорошее место для хранения замечаний, видимых только программисту.

В хороших текстовых редакторах, таких как BBEdit, закомментированный код выделяется цветом (рис. 1.11). Очень полезная функциональность при работе с большими сценариями.

Рис. 1.10 ▼ Сравните этот код с кодом до того, как мы закомментировали команду печати (рис. 1.7), и вы увидите, что PHP-код не был передан в браузер. Закомментированный с помощью операторов <!-- и --> HTML-код все же появляется в браузере, но не показывается им

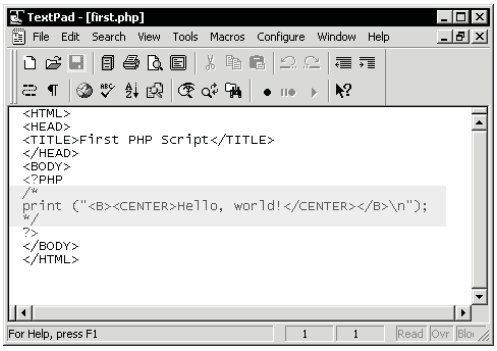

Рис. 1.11 ▼ Если текстовый редактор позволяет выделять логические блоки кода разным цветом, это значительно облегчает программирование. Вы видите, что закомментированный код помечен другим цветом, следовательно, этот конкретный код является неактивным

Глава 2 Переменные

В первой главе говорилось об использовании PHP для передачи простого текста и HTML-кода в Web-браузер, хотя это можно сделать и без PHP. Не стоит беспокоиться: книга научит вас, как использовать инструкцию `print()` и другие возможности языка для создания действительно интересных вещей на вашем Web-сайте.

Для перехода от простых статических страниц к созданию динамических Web-приложений и интерактивных Web-сайтов вам необходимо научиться манипулировать данными. Для этих целей используют переменные. Это неотъемлемый инструмент языка PHP, а также JavaScript, Java, Perl и любого другого языка программирования.

Переменные позволяют временно хранить данные и манипулировать ими. Они наделяют любой язык программирования его истинной силой. Понимание того, что есть переменная, знание типов поддерживаемых в языке переменных и умение их использовать необходимы для работы. В этой главе анализируется само понятие «переменные, используемые в PHP», а в главах 4–6 говорится о том, что именно можно делать с различными типами переменных.

Что такое переменная

Переменная – это своего рода контейнер для данных. Как только данные сохранены в переменной (иначе говоря, как только переменной присвоено значение), они могут быть изменены, напечатаны в Web-браузере, сохранены в базе данных, посланы по электронной почте и т.п. (Под словом «напечатаны» имеется в виду то, что данные отправлены в Web-браузер, но это выполняется именно инструкцией print, поэтому приемлемы оба термина.)

Переменные гибки: можно поместить данные в этот «контейнер», извлечь их оттуда (что не влияет на значение самой переменной), поместить туда новые