On-line-поддержка книги

ЕЛЕНА БЕНКЕН

PHP, MySQL, XML

ПРОГРАММИРОВАНИЕ ДЛЯ ИНТЕРНЕТА

2-е издание

Основы PHP
PHP и MySQL
Базовые сведения о XML
PHP и XML
Новостная лента RSS
Генерация и обработка XML-документов
Практические примеры и задания для самостоятельной работы

Елена Бенкен

PHP, MySQL, XML
ПРОГРАММИРОВАНИЕ ДЛЯ ИНТЕРНЕТА
2-е издание

Санкт-Петербург
«БХВ-Петербург»
2008

УДК 681.3.068+800.92
ББК 32.973.26-018.1
Б46

Бенкен Е. С.

Б46 PHP, MySQL, XML: программирование для Интернета: 2-е изд. перераб. и доп. — СПб.: БХВ-Петербург, 2008. — 352 с.: ил.

ISBN 978-5-9775-0280-1

Рассмотрены актуальные вопросы применения PHP для работы с базами данных MySQL и XML-документами. Описана установка и настройка сервера Apache с модулем PHP 5 и сервера MySQL 5. Изложены основы языка PHP и его расширения. Подробно излагается работа с базами данных MySQL от построения запросов до использования утилит командной строки. Приведены базовые сведения о языке XML. Описан формат новостной ленты RSS и представлены практические примеры обработки XML-документов с помощью расширений PHP 5, таких как SimpleXML, DOM-функциями и функциями событийного программирования SAX. Во втором издании расширено описание объектной модели PHP, уделено внимание проблеме русификации Web-приложений.

Для Web-программистов

УДК 681.3.068+800.92
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

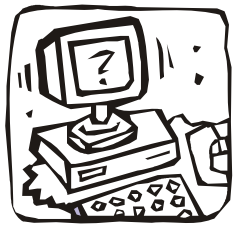
Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.05.08.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 28,38.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.002108.02.07 от 28.02.2007 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ОАО "Техническая книга"
190005, Санкт-Петербург, Измайловский пр., 29.

ISBN 978-5-9775-0280-1

Оглавление



Глава 3

Создание сценариев на PHP. Типы данных, переменные, операторы

Редакторы для работы с PHP

В качестве редактора для скриптов PHP можно использовать любой текстовый редактор, сохраняющий данные в текстовом формате (не так, как MS Word), подойдет и обычный Блокнот (Notepad). Обычно выбор разработчика падает на один из следующих редакторов.

- Редактор PHP Expert Editor (для русскоговорящих граждан — бесплатная регистрация, **http://www.phpexperteditor.com**). Этот редактор предназначен для работы под Windows. Файл PHP размечен по строкам, и на каждой строке можно поставить так называемую точку останова. При запуске скрипта отладчик дойдет только до этой точки.
- Прекрасные редакторы, имеющие коммерческий статус, — Macromedia HomeSite и Macromedia Dreamweaver.
- Zend Development Environment (ZDE) — идеальное решение для профессиональных PHP-разработчиков. ZDE — коммерческий продукт, эта среда разработана фирмой-производителем PHP. Он позволяет отлаживать код PHP 4 и PHP 5, не требуя установленного Web-сервера. У него нет проблем с кодировками, и вам не нужно скачивать русификаторы для того, чтобы меню были на русском. Кроме того, в поставку входит PHP Manual.

Базовый синтаксис

Документация расшифровывает PHP как "рекурсивный акроним словосочетания ”PHP: Hypertext Preprocessor“".

В ходе своей работы модуль PHP читает и исполняет сценарий — текстовый файл, содержащий набор команд. Результат выполнения сценария обычно

представляет собой HTML-документ, который PHP передает Apache, а тот уже клиентскому браузеру.

PHP-сценарии следует сохранять в файлах с расширением php (в соответствии с выполненными настройками Web-сервера) в каталоге htdocs сервера Apache. Запустить сценарий our_script.php, хранимый в корневом каталоге сервера, можно только из браузера, в адресной строке которого следует набрать **http://localhost/our_script.php**. Таким образом мы создаем запрос по протоколу HTTP. В этом случае Apache передаст PHP-сценарий на обработку модулю PHP.

> ***Примечание***
>
> Частая ошибка начинающих состоит в том, что они, желая запустить скрипт на выполнение, щелкают дважды по имени сценария в окне Проводника. Но эта программа не умеет отправлять HTTP-запросы!

Модуль PHP будет обрабатывать и выполнять только те команды скрипта, которые заключены в специальные теги — дескрипторы PHP. PHP поддерживает следующие теги:

- XML-стиль

```
<?php Здесь идет код на PHP ?>
```

- HTML-стиль

```
<script language="php"> Здесь идет код на PHP </script>
```

- краткий стиль

```
<? Здесь идет код на PHP ?>
```

- ASP-стиль

```
<% Здесь идет код на PHP %>
```

Использование двух последних стилей в настоящее время обычно запрещается администраторами Web-серверов. Найдите директивы, задающие этот запрет (`short_open_tag` и `asp_tags`), в файле конфигурации модуля PHP php.ini.

Действия, которые должен выполнить PHP-модуль, указываются PHP-операторами, помещаемыми между открывающим и закрывающим дескрипторами.

Сохраните следующий пример в файл, скажем, 1.php в каталоге htdocs вашего Web-сервера.

```
<?php
echo "Этот текст вы увидите на экране!";
?>
```

Запустите браузер и наберите в адресной строке: localhost. В *главе 2* об установке Apache мы удалили все индексные файлы, поэтому в окне браузера вы должны увидеть список файлов, хранящихся в каталоге htdocs, а внизу — подпись сервера:

```
Apache/2.0.45 (Win32) PHP/5.1.4 Server at localhost Port 80
```

Найдите в списке сценарий 1.php и запустите его. Если вы правильно установили и настроили Apache и PHP, то увидите в окне браузера сообщение "Этот текст вы увидите на экране!". Инструкция `echo` в PHP означает, что текст, следующий за ней, должен быть выведен в браузер.

Команды на языке PHP можно встраивать в HTML-документ. Модуль PHP передаст Web-серверу HTML-теги без изменений, начнет обработку только тех команд, которые встроены в HTML-документ с помощью специальных тегов.

Пример:

```
<html>
  <head>
    <title>Our first page</title>
  </head>
<body>
<?php
  echo "Этот текст вы увидите на экране!";
?>
</body>
</html>
```

Просмотрите исходный код этой страницы — вы не увидите кода PHP: скрипт был выполнен на сервере, а в браузер был передан только результирующий HTML-документ.

Типы данных

Основой любого языка программирования является хранение и обработка данных. При этом важно знать, данные каких типов могут использоваться в конструкциях языка. PHP может оперировать следующими типами данных:

- `integer` — для целых чисел;
- `double` — для действительных чисел (с плавающей точкой).

  В PHP не поддерживается европейская нотация записи действительных чисел с помощью запятой, отделяющей дробную часть числа. Число

с плавающей точкой может быть представлено в экспоненциальном представлении. `1.2e2` — это число 120.

Кроме десятичных чисел в скрипте можно использовать и шестнадцатеричные числа. Перед таким числом должны стоять символы `0x`. Допустимо также применять восьмеричные числа. Перед таким числом надо ставить `0`. Таким образом, число `012` — это восьмеричное число, в десятичным выражении равное 10;

- `boolean` — логические данные. Могут принимать значение `true` (истина) или `false` (ложь);
- `string` — для строк символов;
- `array` — для хранения нескольких элементов данных одного типа;
- `object` — для хранения экземпляров классов;
- `NULL` — специальное значение, указывающее, что данные не имеют значения, даже нуля или пустой строки, т. е. это говорит об отсутствии значений какого-либо типа;
- `resource` — специальный тип данных, содержащих ссылку на какой-то внешний по отношению к скрипту источник данных, например, текстовый файл на диске или графические данные в памяти сервера.

Комментарии

Комментарии используются для пояснения назначения сценария, почему он написан именно таким образом и т. п. Считается, что комментарий должен составлять около 30% от объема кода. Следует писать комментарии сразу на этапе создания и отладки сценария — позже до этого руки у вас никогда не дойдут. Перед телом функции всегда помещайте комментарий, объясняющий ее назначение. Добавляйте комментарии в сомнительных участках кода, когда нет уверенности, что он будет работать как надо. Если назначение кода неочевидно, внесите информацию о предназначении этого участка.

PHP-модуль не анализирует комментарии, которые для него равнозначны пробелам. Многострочные комментарии должны начинаться с символов `/*` и завершаться символами `*/`:

```
/*
  Данный сценарий позволяет
  распечатать все предложения фирмы "IT консалтинг".
*/
```

Часто используются и однострочные комментарии:

```
echo "Привет";  // Вывод сообщения на экран
```

Все символы, следующие за символом комментария (//) вплоть до конца строки или до завершающего дескриптора PHP, в зависимости от того, что встретится раньше, рассматриваются как комментарий.

Выражения и операторы

Для того чтобы работать с данными, необходимо идентифицировать область памяти, в которой они хранятся. Идентификаторы используются для именования данных или блоков кода (имена функций).

Идентификаторы могут начинаться с буквы или символа подчеркивания. Последующие символы в идентификаторе могут быть буквами, цифрами или знаками подчеркивания.

Почти все в PHP является выражением. Простое и точное определение выражения — "все что угодно, имеющее значение". Основными формами выражений являются константы и переменные. Если вы записываете `$a = 5`, то присваиваете значение `5` переменной `$a`.

Оператор состоит из одного или более выражений. Операторы бывают трех видов:

- во-первых, это унарные операторы, которые работают только с одним аргументом, например, `!` (оператор отрицания) или `++` (инкремент);
- вторую группу составляют бинарные операторы, работающие с двумя аргументами;
- третью группу составляет тернарный оператор `?:`.

Операторы PHP разделяются точками с запятой (`;`). Отсутствие точки с запятой в конце команды заставляет PHP выводить на экран сообщение об ошибке.

Константы

Для задания значений, которые не будут изменяться в ходе выполнения программы, можно применять *константы*, которые определяются с помощью конструкции `define`, например:

```
<?php
  define('OUR_CONST', 55);
  echo OUR_CONST;
?>
```

В PHP имена констант традиционно записываются в верхнем регистре. Если вы хотите, чтобы другие программисты могли легко воспринимать ваш код, придерживайтесь этого правила.

Переменные

Переменная — это поименованная ячейка памяти, в которой хранится значение. Идентификаторы переменных в PHP начинаются со знака доллара (`$`). Имена переменных чувствительны к регистру, т. е. `$var` и `$Var` — это разные переменные. При назначении переменных и функций старайтесь давать им осмысленные имена. Иногда ничего не приходит в голову, и появляется желание назвать переменную как попало — остерегайтесь этого. В свое время было потрачено немало часов из-за неудачно названных переменных, иногда отладить код удавалось лишь в том случае, если переменные были переименованы подобающим образом.

Оператор присваивания используется для задания переменной некоторого значения. Например:

```
$a = 25;
```

В PHP тип переменной определяется присвоенным ей значением. В следующем примере переменная `$b` имеет тип `double`:

```
$b = 1.23;
```

При этом в программе дальше может следовать такая строка:

```
$b = "Привет";
```

В этом случае переменная `$b` поменяет свой тип на тип `string`. PHP в любой момент времени изменяет тип переменной в соответствии с данными, хранящимися в ней.

Строковую переменную можно определить тремя способами.

Во-первых, с помощью одинарных кавычек:

```
$str ='Голубое небо';
```

Если в строковое значение необходимо добавить одинарную кавычку, то надо предварить ее обратным слэшем:

```
$str = 'Д\'Артаньян';
```

Второй способ — определить строковое значение в двойных кавычках.

```
$str = "синее море, белый пароход";
```

При этом можно вставить некоторые специальные символы, используя escape-последовательности:

- `\n` — конец строки;
- `\r` — возврат каретки;
- `\t` — горизонтальная табуляция;
- `\\` — обратный слэш.

Слово escape можно перевести глаголом "управлять", т. е. escape-последовательности управляют способом отображения данных. Например:

```
$str="синее море, \n белый пароход";
```

Написав затем в скрипте конструкцию `echo $str;`, вы не увидите в браузере перехода на новую строку, но откройте исходный код — все на месте, просто браузер удалил символ новой строки.

Третий способ задания строкового значения с помощью heredoc-синтаксиса:

```
$str= <<<MARKER
Это сам текст, который может занимать
несколько строк и должен заканчиваться таким же
маркером, каким начинался, причем маркер должен быть
единственным текстом в строке кроме точки с запятой.
MARKER;
```

Приведем еще один пример встраивания PHP-кода в HTML-документ. Пусть переменная `$jpg` получает строковое значение (им будет имя файла):

```
<?php
  $jpg="1.jpg";
  echo "<img border=1 src=\"$jpg\" width=200 height=120>";
?>
```

Сохраните любой рисунок в формате JPEG под именем 1.jpg в тот же каталог, в который сохранили только что приведенный пример, запустите пример и посмотрите исходный код.

Можно включать имя переменной в строку, при этом происходит подстановка значения переменной, если использовать двойные кавычки или heredoc-синтаксис. Следующий пример выведет в браузере строку `"5 котят"`:

```
<?php
  $num = 5;
  echo "$num котят";
?>
```

Поменяйте в приведенном примере двойные кавычки на одинарные, и вы увидите, что PHP не подставляет значение переменной в строку, ограниченную одинарными кавычками. Поэтому для ускорения выполнения сценария рекомендуется использовать при задании строк именно одинарные кавычки — PHP не разбирает текст, заключенный в них.

Для строк определена операция *конкатенации* — объединение строк:

```
<?php
  $start = "Первая часть строки ";
  $stop = " Вторая часть строки";
  echo $start.$stop;
?>
```

Сравните два оператора:

```
<?php
  echo "Другое начало строки ".$stop;
  echo "Другое начало строки $stop";
?>
```

Конструкции `echo` выведут в браузер одну и ту же строку, но первая конструкция работает на 30% быстрее.

Ссылки

Внутри механизма PHP (Zend engine) существует разделение между именем переменной и ее значением. Создадим три переменные

```
$a = 2; $b = 4; $c = 5;
```

Выведем их значения

```
print $a;   print $b;   print $c;
```

Внутреннее представление переменных в PHP будет таким, как представлено на рис. 3.1.

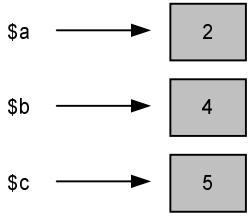


Рис. 3.1. Создание переменных

В PHP механизм ссылок позволяет создать несколько имен для одной и той же переменной. Важно понять это, чтобы правильно использовать ссылки в своем коде. Создадим теперь переменную так:

```
$b = &$a;
```

Знак `&` перед именем переменной указывает на использование ссылки. Переменная `$b` определена как ссылка на переменную `$a`, т. е. она не имеет своей области данных, а пользуется той, которая есть у переменной `$a`. Механизм здесь тот же, что и в жестких связях в UNIX (hard link). При изменении значения любой из переменных, `$b` или `$a`, будет меняться и значение второй переменной. Посмотрим, как в этом случае будут выглядеть внутренние связи между именами переменных и их значениями (рис. 3.2).

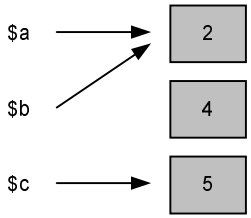


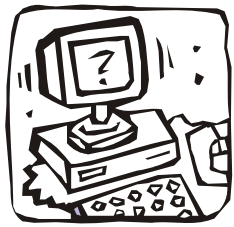
Рис. 3.2. Создание ссылки

Старое значение переменной `$b` теперь не имеет ни одного имени, связанного с ней, и поэтому больше недоступно из PHP-кода. Ядро PHP автоматически уничтожает такие переменные, поэтому значение 4, ранее бывшее значением переменной `$b`, перестает существовать с этого момента.

Поскольку теперь значение первой переменной в таблице значений имеет два имени (`$a` и `$b`), то обращение к любому из этих имен будет рассматриваться как обращение к этой переменной:

```
print $a;       // Результат: 2
print $b;       // Результат: 2
print $c;       // Результат: 5
```

При задании переменной путем приравнивания ее другой переменной (`$b = $a`) происходит копирование значения переменной `$a` в область памяти, которая обозначается как `$b`. При этом при изменении значения переменной `$a` значение переменной `$b` не меняется.



Глава 4

Операции и управляющие конструкции

Программа обычно состоит из последовательности инструкций *(statements)*. Инструкции могут представлять собой объявление переменных, вызов функции и включать в себя операции. *Операции* являются базовыми элементами языка программирования. Наиболее часто применяются арифметические, логические и строковые операции.

Арифметические операции

В PHP поддерживаются обычные *арифметические операции*, работающие с двумя операндами:

- сложение (`+`);
- вычитание (`-`);
- умножение (`*`);
- деление (`/`);
- остаток от деления (`%`).

Можно сохранять результат операции:

```
$a = 5 + 7;
```

Кроме обычных операций можно применять комбинированные операции. Например, если надо взять значение переменной `$a`, прибавить к нему число 5 и сохранить как новое значение переменной `$a`, то это можно сделать двумя способами:

```
$a = 10;
$a = $a + 5;
```

или

```
$a = 10;
$a+=5;
```

Разрешены комбинированные операции, представленные в табл. 4.1.

Таблица 4.1. *Объединенные операции присваивания PHP*

Символ операции	Пример использования	Эквивалентная операция
+=	`$a += $b`	`$a = $a + $b`
-+	`$a -= $b`	`$a = $a - $b`
*=	`$a *= $b`	`$a = $a * $b`
/=	`$a /= $b`	`$a = $a / $b`
%= (остаток от деления)	`$a %= $b`	`$a = $a % $b`
.= (конкатенация)	`$a .= $b`	`$a = $a . $b`

Примечание

Остаток `$a % $b` будет отрицательным для отрицательных значений `$a`.

Кроме того, PHP заимствовал из языка C операции префиксного и постфиксного инкремента (`++`) и декремента (`--`).

Префиксный инкремент увеличивает значение переменной, перед которой записывается, на 1, а затем возвращает ее новое значение. Например, `++$a`.

Постфиксный инкремент возвращает значение переменной, после которой записывается, а потом увеличивает ее значение на 1. Например, `$a++`.

Префиксный декремент уменьшает значение переменной, перед которой записывается, на 1, а затем возвращает ее новое значение. Например, `--$a`.

Постфиксный декремент возвращает значение переменной, после которой записывается, а потом уменьшает ее значение на 1. Например, `$a--`.

Поэкспериментируйте с этими операциями, чтобы хорошенько с ними разобраться. Например:

```
<?php
  $a=10;
  $b = $a++;
  echo $a;
?>
```

Вот один пример, дающий неочевидный результат:

```
<?php
  $a = 1;
```