ДЕНИС КОЛИСНИЧЕНКО

bhv®

PHP 5/6 и MySQL 6

Разработка Web-приложений

2-е издание

- СИНТАКСИС ЯЗЫКА PHP
- ОСОБЕННОСТИ НОВЕЙШЕЙ ВЕРСИИ PHP 6
- ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА PHP 6
- ИСПОЛЬЗОВАНИЕ СЕРВЕРОВ БАЗ ДАННЫХ MySQL 6 И SQLite
- ФУНКЦИИ ДЛЯ РАБОТЫ С MySQL 6
- САМЫЕ ПОЛЕЗНЫЕ PHP-ФУНКЦИИ
- ШАБЛОНИЗАТОР Smarty
- ТЕХНОЛОГИЯ Ajax
- РАБОТА С RSS
- PHP EXTENSION AND APPLICATION REPOSITORY (PEAR)
- СИСТЕМА КОНТРОЛЯ ВЕРСИЙ CVS
- ТЕСТИРОВАНИЕ С ПОМОЩЬЮ PHPUnit
- СТАТИСТИКА САЙТА
- СОЗДАНИЕ СИСТЕМЫ ОПРОСОВ
- СОЗДАНИЕ СИСТЕМЫ ПРОДАЖИ НЕДВИЖИМОСТИ
- SSL-СОЕДИНЕНИЯ

+CD

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

УДК 681.3.06
ББК 32.973.26-018.2
К60

Колисниченко Д. Н.

К60 PHP 5/6 и MySQL 6. Разработка Web-приложений. —
2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 560 с.: ил.
+ CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0581-9

На практических примерах описана разработка Web-приложений на языке PHP версий 5 и 6. Большая часть кода примеров совместима с обеими версиями PHP, но особое внимание уделено новым функциям PHP 6. Даны начала программирования на PHP: установка и настройка PHP и MySQL, выбор редактора PHP-кода, основы синтаксиса и самые полезные функции PHP. Рассмотрено создание собственного движка сайта и ряда дополнительных модулей — фотогалереи, RSS-граббера, модуля для работы с MP3, модуля продажи недвижимости, гостевой книги, а также применение мощного шаблонизатора Smarty и создание простейшего собственного шаблонизатора. В качестве хранилища данных использованы два сервера — самая современная версия MySQL 6 и "суперлегкий" сервер баз данных SQLite. Показано, как с помощью технологии Ajax добиться обновления данных на странице без ее перезагрузки. Во втором издании описаны новые инструменты для создания сложных проектов PEAR, CVS и PHPUnit. Прилагаемый компакт-диск содержит дополнительные главы, все листинги из книги, а также необходимое программное обеспечение.

Для Web-программистов

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	*Владимир Красовский*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 01.04.10.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 45,15.
Тираж 2000 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 978-5-9775-0581-9

Оглавление

Глава 3

Основы синтаксиса PHP

3.1. Переменные

3.1.1. Правила объявления переменных. Имена переменных

Если вы писали до этого на других языках программирования, то наверняка знакомы с понятием переменной, которое присутствует и в PHP. Говоря формально, переменная — это поименованная область памяти. Области памяти присваивается выбранное программистом имя, по которому потом программа обращается к данным. Все данные, с которыми вы будете работать, будут храниться в переменных.

В других языках программирования переменные нужно объявлять до их первого использования, например в языке Pascal:

```
var
      I : integer;
```

PHP не требует явного объявления переменных. Но если переменные не объявлены, как тогда PHP знает, что́ есть переменная, а что́ — обычный символ? Для выделения переменных из кода программы PHP требует, чтобы перед именем переменной указывался символ `$`. Например: `$i`, `$string`.

То, что PHP не требует объявления переменной, не освобождает вас от ее инициализации, то есть от задания переменной начального значения. Инициализация не обязательна, но весьма желательна. Предположим, что вы используете переменную `$i` как счетчик с шагом 2, то есть после определенного события, например после каждой итерации цикла, выполняется следующий оператор:

```
$i = $i + 2;
```

Если вы забыли присвоить переменной $i начальное значение до выполнения этого оператора, результат будет непредсказуемым. Поэтому возьмите за правило инициализировать каждую переменную, которую вы собираетесь использовать, например:

```
$i = 0;
```

Вернемся к именам переменных. Как уже было сказано, перед именем переменной нужно обязательно указывать символ доллара. Например:

```
i = $i + 2;          // этот код неправильный!
i = i + 2;           // этот код неправильный!
$i = $i + 2;         // этот код ПРАВИЛЬНЫЙ
```

Имя переменной может состоять из латинских символов, знака подчеркивания и цифр, причем не может начинаться с цифры.

Имена переменных чувствительны к регистру символов, то есть $a и $A — две разные переменные:

```
$a = 1;
$A = 2;


echo $a;
echo $A;
```

Наш код выведет, как и следовало ожидать, 12. Если бы переменные не были чувствительны к регистру символов, код вывел бы 11.

3.1.2. Типы данных переменных

Обратите внимание, что при объявлении переменной на языке Pascal были указаны имя переменной и ее тип. PHP не требует явного объявления типа переменной, но это не означает, что в PHP нет типов переменных. Просто PHP по-особому работает с переменными.

Предположим, что вы объявили две переменные. Одна переменная ($a) содержит число, а вторая переменная ($b) — массив. Присвоим второй переменной значение первой переменной:

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

Вы только задумайтесь, что происходит. Вы присваиваете число массиву! В любом другом языке программирования вы бы получили ошибку несовместимости типов переменных, но не в PHP. PHP возьмет да и скопирует в

переменную $b не только значение переменной $a, но и тип (структуру) переменной. Да, в результате массив будет потерян, $b станет обычной целой переменной и будет содержать значение переменной $a.

Такая "молчаливость" PHP довольно удобна, но с другой стороны, если вы невнимательны, она может стать причиной трудно отслеживаемых ошибок.

Вот основные типы переменных в PHP:

- int — целой переменной вы можете присвоить значения от –2 147 483 648 до 2 147 483 647;
- double — переменная, содержащая вещественное число большой точности;
- string — наверное, самый важный тип данных в PHP, поскольку очень много операций связано именно с обработкой строк;
- array — массивам и спискам посвящен целый раздел книги;
- object — PHP является объектно-ориентированным языком программирования, что, впрочем, требуется далеко не всегда — вы можете написать превосходный сценарий, работающий и без объектов;
- bool — логическая переменная (может принимать значения true или false);
- link — ссылка, особый тип, о котором мы поговорим отдельно.

Для проверки типа переменной используются функции, приведенные в табл. 3.1.

Таблица 3.1. *Функции проверки типа переменной*

Функция	Описание
is_int($x)	Возвращает true, если указанная переменная — целое число
is_double($x)	Возвращает true, если указанная переменная — вещественное число
is_string($x)	Проверяет, является ли переменная строкой
is_array($x)	Проверяет, является ли переменная массивом
is_object($x)	Возвращает true, если указанная переменная — объект
is_bool($x)	Проверяет, является ли переменная логической переменной

Таблица 3.1 *(окончание)*

Функция	Описание
`gettype($x)`	Возвращает строку, которая описывает тип переменной: ❒ `"boolean"` — логическое значение; ❒ `"integer"` — целое значение; ❒ `"double"` — вещественное значение; ❒ `"string"` — строка; ❒ `"array"` — массив; ❒ `"object"` — объект; ❒ `"resource"` — ресурс (например, результат обработки MySQL-запроса); ❒ `"NULL"` — переменная не имеет значения (не инициализирована), следовательно, невозможно установить ее тип; ❒ `"unknown type"` — неизвестный тип (тип определить не удалось)

3.1.3. Булевы переменные

Отдельного разговора заслуживают логические переменные. Мы привыкли, что в других языках программирования значения `true` и `false` ассоциируются соответственно со значениями 1 и 0. Но в PHP значение `false` ассоциируется с пустой строкой, а `true` — с любым ненулевым значением, например:

```
$a = 10001;
if ($a == true) echo 'Переменная $a — истинна';
```

> ***Примечание***
>
> Обратите внимание, что в операторе `echo` используются одинарные кавычки, а не двойные.

Поскольку со значением `true` ассоциируется любое ненулевое значение, то мы увидим следующую строку:

```
Переменная $a — истинна
```

Рассмотрим еще один пример:

```
$a = 1001; $b = true;
if ($a == $b) echo '$a = $b';     // сравниваем переменные
echo "\n".gettype($b);            // выводим тип переменной $b
```

Самое интересное, что наш сценарий сообщит, что переменные `$a` и `$b` равны (рис. 3.1). Это происходит потому, что переменной `$b` присвоен логический тип (это так и есть), и PHP считает, что и переменная `$a` тоже логического типа.

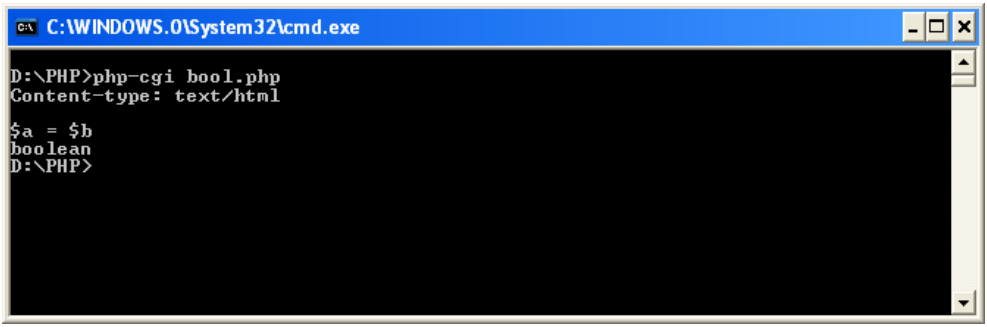

Рис. 3.1. Сравнение логической и целой переменных

3.1.4. Операции над переменными

Над переменной вы можете выполнить всего три операции:

- присвоить значение переменной — константу, результат выполнения функции или выражения:

```
$x = 1;
$x = pi;
$x = func($y);
$x = ($b * 2) / 100;
```

- уничтожить переменную — для этого используется функция `unset`:

```
unset($a);
```

- проверить существование переменной — используется функция `isset()`:

```
if ($isset($x)) echo 'Переменная $x установлена';
```

3.1.5. Ссылки

Если вы знакомы с программированием на C, то вы знакомы с указателями. В PHP нет указателей, зато есть ссылки. Ссылки можно расценивать как псевдонимы переменных. Предположим, что у вас есть переменная-массив `$arr1` объемом 1 Мбайт. Если вы присвоите ее значение другой переменной, `$arr2`, то у вас будет два одинаковых массива общим размером 2 Мбайт.

Ссылки позволяют обращаться к одним и тем же данным под разными именами:

```
$a = 100;
$link_on_a = &$a;

echo $link_on_a;

echo " ";     // пробел

$link_on_a = 5;
echo $a;
```

Сначала мы объявляем переменную `$a`. Затем мы объявляем ссылку `$link_on_a` на переменную `$a`. Синтаксис объявления ссылки следующий:

```
ссылка = &переменная;
```

После чего мы выводим переменную `$link_on_a`. Она равна 100, как и следовало ожидать. Затем мы присваиваем ссылке `$link_on_a` значение 5 и выводим значение переменной `$a`. Сценарий выведет значение 5 (рис. 3.2).

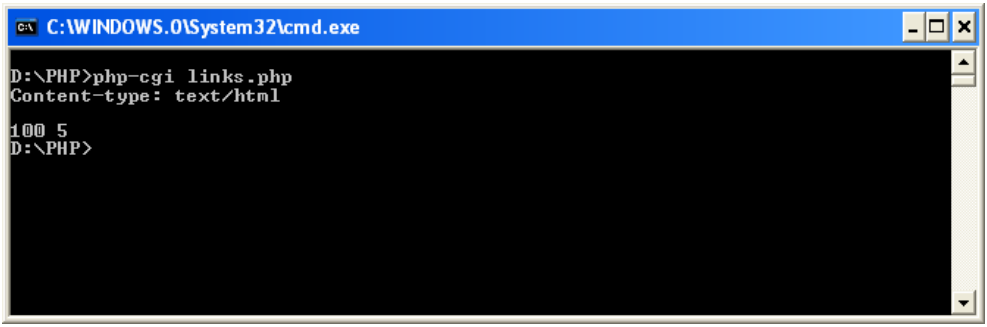

Рис. 3.2. Пример использования ссылок

Приведенные ссылки называются *жесткими*. Существуют еще "мягкие" (символические) ссылки. Символическая ссылка — это не псевдоним переменной, а переменная, содержащая имя другой переменной.

Работать с мягкими ссылками нужно следующим образом. Сначала нужно создать переменную, на которую будет указывать ссылка. Затем нужно другой переменной присвоить имя первой переменной. Чтобы создать мягкую ссылку, нужно обратиться ко второй переменной так:

```
$$переменная
```

Рассмотрим следующий фрагмент кода:

```
$a = 100;                    // исходная переменная
$soft_link = "a";            // эта переменная будет играть роль
                            // ссылки

echo $$soft_link;            // это уже ссылка, выведет 100

echo " ";                    // выводим пробел

echo $soft_link;             // выведет просто 'a' (без кавычек)

$$soft_link = 5;             // присваиваем ссылке значение 5

echo " ";                    // еще один пробел

echo $a;                     // выводим переменную $a, будет выведено 5
```

3.2. Константы

Константы содержат постоянные значения, изменить которые в процессе выполнения программы вы не можете. Для объявления констант используется функция `define()`, синтаксис ее таков:

```
define(имя, значение, чувствительность_к_регистру);
```

Вот примеры объявления констант:

```
define("const", "константа", true);
define("A", "1");
```

Первый параметр функции `define()` — имя константы, второй — значение. Третий параметр особый. Если он `true`, имя константы будет чувствительно к регистру. По умолчанию константы чувствительны к регистру, то есть `CONST` и `const` — это две разные константы.

При использовании константы указывать знак доллара не нужно:

```
// выводим константы
echo const;
echo A;
```

Вы можете использовать следующие стандартные константы:

- `_FILE_` — имя текущего сценария;
- `PHP_OS` — название операционной системы, под которой выполняется PHP;
- `PHP_VERSION` — версия PHP.

3.3. Выражения и операции

3.3.1. Что такое выражение

Выражение — это часть инструкции или сама инструкция, возвращающая значение. Практически все инструкции программы являются выражениями. Даже оператор присваивания:

```
$a = 1;
```

Результат этой операции — 1, он будет присвоен переменной `$a`. Само выражение в этом случае выглядит так:

```
1
```

Выражением может быть не только значение, но и инициализированная переменная:

```
$a = 1;
$b = $a;
```

В PHP оператор присваивания может выглядеть довольно необычно:

```
$a = $b = 1;
```

Интересно, что в выражении можно даже инициализировать переменную:

```
$a = 50 * ($c = 7) * $c;
```

3.3.2. Арифметические операции

Как и в любом другом языке программирования, вы можете использовать следующие арифметические операции:

- `+` — сложение;
- `-` — вычитание;
- `*` — умножение;
- `/` — деление;
- `A % B` — остаток от деления `A` на `B`.

Кроме того, вы можете использовать операции инкремента и декремента:

- `$a++` — увеличить переменную `$a` на 1;
- `$b--` — уменьшить переменную `$b` на 1.

3.3.3. Логические выражения

Логическим называется выражение, результатом которого является истина или ложь. Обычно в логических выражениях используются следующие операторы сравнения:

- `<` — меньше;
- `>` — больше;
- `!=` — не равно;
- `==` — равно;
- `<=` — меньше или равно;
- `>=` — больше или равно.

Также вы можете использовать следующие сугубо логические операции:

- `&&` — бинарная операция И (AND); истинна, если оба операнда истинны;
- `||` — бинарная операция ИЛИ (OR); истинна, если один из операндов `true`;
- `!` — унарная операция отрицания (NOT).

Примеры использования логических операций:

```
$a = true;
$b = false;
```

```
$c = $a || $b;       // $c = true;
$d = $a && $b;       // $d = false;

$e = !$a;            // $e = false;
```

3.3.4. Приоритеты операций

Операции в PHP выполняются в соответствии с приоритетом выполнения операций, приведенным в табл. 3.2. Для изменения порядка выполнения операций можно использовать скобки, например:

```
$a = (2 + 2) * 2;
```

Таблица 3.2. *Приоритет операций*

Приоритет	Порядок выполнения	Операторы
13	справа налево	=, +=, .=, *=, /=, %=, >>=, <<=, &=, ^=, |=
12	слева направо	||
11	слева направо	&&
10	слева направо	|
9	слева направо	^
8	слева направо	&
7	слева направо	==, !=
6	слева направо	<, <=, >, >=
5	слева направо	<<, >>
4	слева направо	+, -
3	слева направо	*, /, %
2	справа налево	++(префикс), --(префикс)
1	слева направо	(постфикс)++, (постфикс)--

Примечание

Приоритет операций в табл. 3.2 представлен от низшего к высшему.

Примечание

Операции >> и << называются битовыми. Мы их не рассматриваем, поскольку при Web-разработке они используются очень редко, а на роль системного языка программирования (как C) PHP явно не тянет!

3.3.5. Операторы эквивалентности == и ===

Для сравнения значений используется оператор равенства ==, например:

```
if ($a == $b) ...
```

Иногда этот оператор работает не совсем так, как мы ожидаем. Рассмотрим следующий код:

```
$x = 1;
$y = 1;

$a = 0;
$b = "";

if ($x == $y) echo 'Переменные $x и $y равны';
if ($a == $b) echo 'Переменные $a и $b равны';
```

Переменные $x и $y действительно равны, поэтому мы увидим сообщение:

```
Переменные $x и $y равны
```

Но переменные $a и $b вообще не могут быть равны, они даже разных типов, однако PHP сообщает, что они равны:

```
Переменные $a и $b равны
```

Почему так происходит? Просто PHP воспринимает $a и $b как логические. А 0 и пустая строка соответствуют false, поэтому он и сообщает, что переменные равны.

Чтобы избежать подобной неразберихи, нужно использовать оператор сравнения === (три знака равенства):

```
if ($a === $b) echo 'Переменные $a и $b равны';
```

В этом случае строка 'Переменные $a и $b равны' не будет выведена.

Помните, что для оператора === не предусмотрен обратный оператор !===, вместо него вы должны использовать следующую конструкцию:

```
if (!($a === $b)) echo '$a <> $b';
```