Web-разработка на PHP-технологиях

Курс лекций и семинаров для студентов,
желающих научиться основам Web-разработки на PHP

Осень-Зима 2014

Лекция №4

PHP — Синтаксис языка
Операторы

Автор: Дмитрий Левин, Senior PHP Developer, Sibers

Sibers®

Типы операторов

- Арифметические операторы PHP
- Логические операторы PHP
- Строковые операторы PHP
- Побитовые операторы PHP
- Операторы присвоения PHP
- Операторы сравнения PHP
- Операции инкремента и декремента PHP
- Операторы эквивалентности PHP
- Операции с символьными переменными PHP
- Приоритеты операторов PHP
- Операторы исполнения PHP
- Операторы работы с массивами
- Операторы управления ошибками PHP

Арифметические операторы PHP

Пример	Название	Результат
-$a	Отрицание	Смена знака $a.
$a + $b	Сложение	Сумма $a и $b.
$a - $b	Вычитание	Разность $a и $b.
$a * $b	Умножение	Произведение $a и $b.
$a / $b	Деление	Частное от деления $a на $b.
$a % $b	Деление по модулю	Целочисленный остаток от деления $a на $b.

Логические операторы PHP

Пример	Название	Результат
$a and $b	Логическое 'и'	**TRUE** если и $a, и $b **TRUE**.
$a or $b	Логическое 'или'	**TRUE** если или $a, или $b **TRUE**.
$a xor $b	Исключающее 'или'	**TRUE** если $a, или $b **TRUE**, но не оба.
! $a	Отрицание	**TRUE** если $a не **TRUE**.
$a && $b	Логическое 'и'	**TRUE** если и $a, и $b **TRUE**.
$a || $b	Логическое 'или'	**TRUE** если или $a, или $b **TRUE**.

Строковые операторы PHP

В PHP есть два оператора для работы со строками.

Первый — оператор конкатенации ('.'), который возвращает объединение левого и правого аргумента.

Второй — оператор присвоения с конкатенацией, который присоединяет правый аргумент к левому.

```php
<?php
$a = "Hello ";
$b = $a . "World!"; // $b содержит строку "Hello World!"

$a = "Hello ";
$a .= "World!";     // $a содержит строку "Hello World!"
?>
```

Побитовые операторы PHP

Побитовые операции предназначены для работы (установки/снятия/проверки) групп битов в целой переменной.

Биты целого числа — это не что иное, как отдельные разряды того же самого числа, записанного в двоичной системе счисления

0000 0000 0000 0000 0000 0000 0000 0000 — это ноль;
0000 0000 0000 0000 0000 0000 0000 0001 — это 1;
0000 0000 0000 0000 0000 0000 0000 0010 — это 2;
0000 0000 0000 0000 0000 0000 0000 0011 — это 3;
0000 0000 0000 0000 0000 0000 0000 0100 — это 4;
0000 0000 0000 0000 0000 0000 0000 0101 — это 5.

Побитовые операторы PHP

Пример	Название	Результат
$a & $b	Побитовое 'и'	Устанавливаются только те биты, которые установлены и в $a, и в $b.
$a | $b	Побитовое 'или'	Устанавливаются те биты, которые установлены либо в $a, либо в $b.
$a ^ $b	Исключающее или	Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b
~ $a	Отрицание	Устанавливаются те биты, которые в $a не установлены, и наоборот.
$a << $b	Сдвиг влево	Все биты переменной $a сдвигаются на $b позиций влево (каждая позиция подразумевает 'умножение на 2')
$a >> $b	Сдвиг вправо	Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает 'деление на 2')

Операторы присвоения PHP

Базовый оператор присвоения обозначается как =.
На первый взгляд может показаться, что это оператор «равно». На самом деле это не так. В действительности, оператор присвоения означает, что левый операнд получает значение правого выражения, (т. е. устанавливается результирующим значением)

```php
<?php

$a = ($b = 4) + 5; // результат: $a установлена значением 9, переменной $b присвоено 4.

?>
<?php

$a = 3;
$a += 5; // устанавливает $a значением 8, аналогично записи: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // устанавливает $b строкой "Hello There!",  как и $b = $b . "There!";

?>
```

Операторы сравнения PHP

Операторы сравнения, как это видно из их названия, позволяют сравнивать между собой два значения.

Это в своем роде уникальные операции, потому что независимо от типов своих аргументов они всегда возвращают одно из двух: false или true.

Операции сравнения позволяют сравнивать два значения между собой и, если условие выполнено, возвращают true, а если нет — false

Операторы сравнения PHP

Пример	Название	Результат
$a == $b	Равно	**TRUE** если $a равно $b.
$a === $b	Тождественно равно	**TRUE** если $a равно $b и имеет тот же тип. (Добавлено в PHP 4)
$a != $b	Не равно	**TRUE** если $a не равно $b.
$a <> $b	Не равно	**TRUE** если $a не равно $b.
$a !== $b	Тождественно не равно	**TRUE** если $a не равно $b или в случае, если они разных типов (Добавлено в PHP 4)
$a < $b	Меньше	**TRUE** если $a строго меньше $b.
$a > $b	Больше	**TRUE** если $a строго больше $b.
$a <= $b	Меньше или равно	**TRUE** если $a is меньше или равно $b.
$a >= $b	Больше или равно	**TRUE** если $a больше или равно $b.

Операторы инкремента и декремента в PHP

Пример	Название	Действие
++$a	Префиксный инкремент	Увеличивает $a на единицу и возвращает значение $a.
$a++	Постфиксный инкремент	Возвращает значение $a, а затем увеличивает $a на единицу.
--$a	Префиксный декремент	Уменьшает $a на единицу и возвращает значение $a.
$a--	Постфиксный декремент	Возвращает значение $a, а затем уменьшает $a на единицу.

```
$a=10;
$b=$a++;
echo "a=$a, b=$b"; // Выводит a=11, b=10
```

Операторы эквивалентности PHP

В PHP, начиная с PHP4 есть оператор тождественного сравнения — тройной знак равенства **===**, или оператор проверки на эквивалентность. PHP довольно терпимо относится к тому, что строки неявно преобразуются в числа, и наоборот.

```
$a=0; // ноль
$b=""; // пустая строка
if($a==$b) echo "a и b равны"; // Выводит "a и b равны"
```

```
$a=0; // ноль
$b=""; // пустая строка
if($a===$b) echo "a и b равны"; // Ничего не выводит
```

```
$a=array('a'=>'aaa');
$b=array('b'=>'bbb');
if($a==$b) echo "С использованием == a=b<br>";
if($a===$b) echo "С использованием === a=b<br>";
```

Операции с символьными переменными в PHP

PHP следует соглашениям Perl (в отличие от C) касательно выполнения арифметических операций с символьными переменными

```php
<?php
$i = 'W';
for($n=0; $n<6; $n++)
   echo ++$i . "\n";

/*
   Результат работы будет следующий:

X
Y
Z
AA
AB
AC

*/
```

Приоритеты выполнения операторов в PHP

Приоритет операторов определяет, насколько «тесно» связанны между собой два выражения. Например, выражение 1 + 5 * 3 вычисляется как 16, а не 18, поскольку операция умножения ("*") имеет более высокий приоритет, чем операция сложения ("+").

В случае, если операторы имеют одинаковый приоритет, они будут выполняться слева направо.

Круглые скобки могут использоваться для принудительного указания необходимого порядка выполнения операторов. Например, выражение (1 + 5) * 3 вычисляется как 18.

```php
<?php
    $a = 3 * 3 % 5; // (3 * 3) % 5 = 4
    $a = true ? 0 : true ? 1 : 2; // (true ? 0 : true) ? 1 : 2 = 2

    $a = 1;
    $b = 2;
    $a = $b += 3; // $a = ($b += 3) -> $a = 5, $b = 5
```

Приоритеты выполнения операторов в PHP

Приоритет	Оператор	Порядок выполнения
13	(постфикс)++ (постфикс)--	слева направо
12	++(префикс) --(префикс)	справа налево
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	||	слева направо
1	= += -= *= /= %= >>= <<== &= ^= |=	справа налево

Операторы исполнения внешних команд в PHP

PHP поддерживает один оператор исполнения: обратные кавычки (``).

Обратите внимание, что это не одиночные кавычки. PHP пытается выполнить строку, заключенную в обратные кавычки, как консольную команду, и возвращает полученный вывод.

Использование обратных кавычек аналогично использованию функции shell_exec()

```php
<?php
$output = `ls -al`;
echo "<pre>$output</pre>";
```

Операторы для работы с массивами PHP

Пример	Название	Результат
$a + $b	Объединение	Объединение массива $a и массива $b.
$a == $b	Равно	**TRUE** в случае, если $a и $b содержат одни и те же элементы.
$a === $b	Тождественно равно	**TRUE** в случае, если $a и $b содержат одни и те же элементы в том же самом порядке.
$a != $b	Не равно	**TRUE** если массив $a не равен массиву $b.
$a <> $b	Не равно	**TRUE** если массив $a не равен массиву $b.
$a !== $b	Тождественно не равно	**TRUE** если массив $a не равен тождественно массиву $b.

Операторы для работы с массивами PHP

```php
<?php
$a = array("a" => "apple", "b" => "banana");
$b = array("a" => "pear", "b" => "strawberry", "c" => "cherry");

$c = $a + $b; // Объеденение $a и $b
echo "Union of \$a and \$b: \n";
var_dump($c);

$c = $b + $a; // Объединение $b и $a
echo "Union of \$b and \$a: \n";
var_dump($c);
```

Операторы для работы с массивами PHP

```
Union of $a and $b:
array(3) {
   ["a"]=>
   string(5) "apple"
   ["b"]=>
   string(6) "banana"
   ["c"]=>
   string(6) "cherry"
}
Union of $b and $a:
array(3) {
   ["a"]=>
   string(4) "pear"
   ["b"]=>
   string(10) "strawberry"
   ["c"]=>
   string(6) "cherry"
}
```

```php
<?php
$a = array("apple", "banana");
$b = array(1 => "banana", "0" => "apple");

var_dump($a == $b); // bool(true)
var_dump($a === $b); // bool(false)
```

Операторы управления ошибками PHP

PHP поддерживает один оператор управления ошибками: знак @.

В случае, если он предшествует какому-либо выражению в PHP-коде, любые сообщения об ошибках, генерируемые этим выражением, будут проигнорированы.

```php
<?php
// Преднамеренная ошибка при работе с файлами
$my_file = @file ('non_existent_file') or
     die ("Failed opening file: error was '$php_errormsg'");

// работает для любых выражений, а не только для функций
$value = @$cache[$key];
// В случае если ключа $key нет, сообщение об ошибке не будет отображено
```

Операторы управления ошибками PHP

Внимание: Оператор @ работает только с выражениями.

Есть простое правило: если произвольная языковая конструкция возвращает значение, значит вы можете использовать предшествующий ей оператор @.

Например, вы можете использовать @ перед именем переменной, произвольной функцией или вызовом include(), константой и так далее. В то же время вы не можете использовать этот оператор перед определением функции или класса, условными конструкциями, такими как if или foreach.

Внимание: Оператор @ не подавляет вывод ошибок, возникающих на стадии синтаксического разбора скрипта.

На сегодняшний день оператор @ подавляет вывод сообщений даже о критических ошибках прерывающих работу скрипта. Помимо всего прочего, это означает, что если вы использовали @ для подавления ошибок, возникающих при работе какой-либо функции, в случае если она недоступна или написана неправильно, дальнейшая работа скрипта будет остановлена без каких-либо уведомлений.

Операторы классов PHP

Оператор instanceof используется для определения того, является ли текущий объект экземпляром указанного класса.

Оператор instanceof был добавлен в PHP5. До этого использовалась функция is_a(), которая на данный момент не рекомендуется к применению, более предпочтительно использовать оператор instanceof.

```
<?php
class A { }
class B { }

$thing = new A;
if ($thing instanceof A) {
echo 'A';
} if ($thing instanceof B) {
echo 'B';
}
```

Заключение:

В четвёртой лекции мы разобрали основные операторы в языке PHP и приоритеты выполнения операторов

В следующей лекции:

PHP — Синтаксис языка

Управляющие конструкции языка PHP. Функции обработки строк. Функции для работы с массивами.

Полезные ссылки:

http://www.php.su/learnphp/operators/
http://php.net/manual/ru/language.operators.php