Web-разработка на PHP-технологиях

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

Осень-Зима 2014

Лекция №5

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

Автор: Дмитрий Левин, Senior PHP Developer, Sibers

Sibers®

Условные операторы PHP

Конструкция IF

Согласно выражениям PHP, конструкция if содержит логическое выражение. Если логическое выражение истинно (true), то оператор, следующий за конструкцией if будет исполнен, а если логическое выражение ложно (false), то следующий за if оператор исполнен не будет.

```php
$data = 12;

if($data > 10){
    $data--;
}
```

Условные операторы PHP

Конструкция ELSE

Часто возникает потребность исполнения операторов не только в теле конструкции if, если выполнено какое-либо условие конструкции if, но и в случае, если условие конструкции if не выполнено.

В данной ситуации нельзя обойтись без конструкции else

```
$data = 12;

if($data > 10){
    $data--;
} else {
    $data++;
}
```

Условные операторы PHP

Конструкция ELSEIF

```php
$a = 12;
$b = 10;

if ($a > $b) {
    echo "a больше, чем b";
} elseif ($a == $b) {
    echo "a равен b";
} else {
    echo "a меньше, чем b";
}
```

Конструкции выбора в PHP

Конструкция switch-case чем-то напоминает конструкцию if-else, который, по сути, является её аналогом. Конструкцию выбора можно использовать, если предполагаемых вариантов много, например, более 5, и для каждого варианта нужно выполнить специфические действия. В таком случае, использование конструкции if-else становится действительно неудобным.

```php
$data = 10;

if($data == 10){
    echo 'case1';
} elseif($data == 11){
    echo 'case2';
} elseif($data == 12){
    echo 'case3';
} elseif($data == 13){
    echo 'case4';
} elseif($data == 14){
    echo 'case5';
} elseif($data == 15){
    echo 'case6';
}
```

```php
$data = 10;

switch($data){
    case 10 : echo 'case1';  break;
    case 11 : echo 'case2';  break;
    case 12 : echo 'case3';  break;
    case 13 : echo 'case4';  break;
    case 14 : echo 'case5';  break;
    case 15 : echo 'case6';  break;
    default: echo 'case default';  break;
}
```

Циклы в PHP

Циклы позволяют повторять определенное (и даже неопределенное — когда работа цикла зависит от условия) количество раз различные операторы. Данные операторы называются **телом цикла**. Проход цикла называется **итерацией**.

PHP поддерживает три вида циклов:

- Цикл с предусловием (**while**);
- Цикл с постусловием (**do-while**);
- Цикл со счетчиком (**for**);
- Специальный цикл перебора массивов (**foreach**).

При использовании циклов есть возможность использования операторов **break** и **continue**. Первый из них прерывает работу всего цикла, а второй — только текущей итерации.

Циклы в PHP: **WHILE**

Цикл с предусловием while работает по следующим принципам:

- Вычисляется значение логического выражения.
- Если значение истинно, выполняется тело цикла, в противном случае — переходим на следующий за циклом оператор.

```php
$x = 0;

while($x < 10){
    $x++;
    echo $x;
}

// Выводит 12345678910
```

```php
$x = 0;

while($x < 10):
    $x++;
    echo $x;
endwhile;

// Выводит 12345678910
```

Sibers®

Циклы в PHP: **DO WHILE**

Цикл с постусловием **do while**

В отличие от цикла **while**, этот цикл проверяет значение выражения не до, а после каждого прохода (итерации). Таким образом, тело цикла выполняется хотя бы один раз.

```php
$x = 1;
do {
    echo $x;
} while ($x++<10);

// Выводит 12345678910
```

Циклы в PHP: **FOR**

Цикл со счетчиком **for**

Цикл со счетчиком используется для выполнения тела цикла определенное число раз. С помощью цикла **for** можно (и нужно) создавать конструкции, которые будут выполнять действия совсем не такие тривиальные, как простая переборка значения счетчика.

```
for (инициализирующие_команды; условие_цикла; команды_после_итерации) { тело_цикла; }
```

```php
for($x = 0; $x<10; $x++){
    echo $x;
}
// Выводит 012345678910
```

```php
for($x = 0; $x<10; $x++):
    echo $x;
endfor;
// Выводит 012345678910
```

Циклы в PHP: **FOREACH**

Цикл перебора массивов **foreach**

Данный цикл предназначен специально для перебора массивов.

```php
$array = array(
    'os_1' => 'Windows',
    'os_2' => 'Linux',
    'os_3' => 'Mac'
);

foreach ($array as $key => $value){
    echo "$key - $value \r\n";
}
/**
    os_1 - Windows
    os_2 - Linux
    os_3 - Mac
*/
```

```php
$array = array(
    'os_1' => 'Windows',
    'os_2' => 'Linux',
    'os_3' => 'Mac'
);

foreach ($array as $value){
    echo "$value \r\n";
}
/**
    Windows
    Linux
    Mac
*/
```

Циклы в PHP: **Break**

Конструкция break

Очень часто для того, чтобы упростить логику какого-нибудь сложного цикла, удобно иметь возможность его прервать в ходе очередной итерации (к примеру, при выполнении какого-нибудь особенного условия).

Для этого и существует конструкция break, которая осуществляет немедленный выход из цикла. Она может задаваться с одним необязательным параметром — числом, которое указывает, из какого вложенного цикла должен быть произведен выход. По умолчанию используется 1, т. е. выход из текущего цикла, но иногда применяются и другие значения.

```php
$x = 0;
while ($x++<10) {
    if ($x == 3) break;
    echo "<b>Итерация $x</b><br>";
}

// Итерация 1
// Итерация 2
```

Циклы в PHP: **Continue**

Конструкция **continue** так же, как и **break**, работает только в «паре» с циклическими конструкциями. Она немедленно завершает текущую итерацию цикла и переходит к новой (конечно, если выполняется условие цикла для цикла с предусловием).

Точно так же, как и для **break**, для **continue** можно указать уровень вложенности цикла, который будет продолжен по возврату управления.

```
$x=0;
while ($x++<5) {
    if ($x==3) continue;
    echo "<b>Итерация $x</b><br>";
}

// Итерация 1
// Итерация 2
// Итерация 4
// Итерация 5
```

Функции обработки строк

Функции обработки строк

addslashes — экранирует спецсимволы в строке

Возвращает строку str, в которой перед каждым спецсимволом добавлен обратный слэш (\), например для последующего использования этой строки в запросе к базе данных.

Экранируются одиночная кавычка ('), двойная кавычка ("), обратный слэш (\) и NUL (байт NULL).

```php
$str = "Is your name O'reilly?";

// выводит: Is your name O\'reilly?
echo addslashes($str);
```

Функции обработки строк

stripslashes — удаляет экранирование символов, произведенное функцией addslashes()

Удаляет экранирующие бэкслэши. (\' преобразуется в ', и т.д.).
Двойные бэкслэши (\\) преобразуется в одиночные(\).

```php
$str = "Is your name O\'reilly?";

// выводит: Is your name O'reilly?
echo stripslashes($str);
```

Функции обработки строк

explode — разбивает строку на подстроки

Возвращает массив строк, полученных разбиением строки string с использованием separator в качестве разделителя. Если передан аргумент limit передан, массив будет содержать максимум limit элементов, при этом последний элемент будет содержать остаток строки string.

```php
$pizza  = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
echo $pieces[0]; // piece1
echo $pieces[1]; // piece2
```

Функции обработки строк

implode — разбивает строку на подстроки

Возвращает строку, полученную объединением строковых представлений элементов массива, со вставкой строки разделителя между соседними элементами.

```php
$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);

echo $comma_separated; // lastname,email,phone
```

Функции обработки строк

htmlentities — преобразует символы в соответствующие HTML сущности.

htmlentities() преобразует все символы в соответствующие HTML сущности (для тех символов, для которых HTML сущности существуют).

```php
$str = "A 'quote' is <b>bold</b>";

// выводит: A 'quote' is &lt;b&gt;bold&lt;/b&gt;
echo htmlentities($str);
```

Функции обработки строк

md5 — возвращает MD5 хэш строки

Вычисляет MD5 хэш строки **str** используя алгоритм MD5 RSA Data Security, Inc. и возвращает этот хэш. Хэш представляет собой 32-значное шестнадцатеричное число. Если необязательный аргумент **raw_output** имеет значение **TRUE**, то возвращается бинарная строка из 16 символов.

```php
$str = 'apple';

if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f') {
    echo "Would you like a green or red apple?";
    exit;
}
```

Функции обработки строк

str_replace — заменяет строку поиска на строку замены

```php
// присваивает <body text='black'>
$bodytag = str_replace("%body%", "black", "<body text='%body%'>");

// присваивает: Hll Wrld f PHP
$vowels = array("a", "e", "i", "o", "u", "A", "E", "I", "O", "U");
$onlyconsonants = str_replace($vowels, "", "Hello World of PHP");

// присваивает: You should eat pizza, beer, and ice cream every day
$phrase  = "You should eat fruits, vegetables, and fiber every day.";
$healthy = array("fruits", "vegetables", "fiber");
$yummy   = array("pizza", "beer", "ice cream");
```

Функции обработки строк

strip_tags — удаляет HTML и PHP тэги из строки

```php
$text = '<p>Параграф.</p><span>Еще немного текста</span>';
echo strip_tags($text);
// Параграф.Еще немного текста
echo strip_tags($text, '<p>');
// <p>Параграф.</p>Еще немного текста
```

Функции обработки строк

strlen — возвращает длину строки

```php
$str = 'abcdef';
echo strlen($str); // 6

$str = ' ab cd ';
echo strlen($str); // 7
```

Функции обработки строк

strpos — возвращает позицию первого вхождения подстроки

Возвращает позицию первого найденного совпадения в строке. Если подстрока искомого не найдена, strpos() возвращает FALSE.

```php
$mystring = 'abc';
$findme   = 'a';
$pos = strpos($mystring, $findme);

if ($pos === false) {
    echo "Строка '$findme' не найдена в строке '$mystring1'";
} else {
    echo "Строка '$findme' найдена в строке '$mystring1'";
    echo " в позиции $pos";
}

// Ищем , начиная со второго символа
$newstring = 'abcdef abcdef';
$pos = strpos($newstring, 'a', 1); // $pos = 7, not 0
```

Функции обработки строк

strtolower — преобразует строку в нижний регистр

Возвращает строку string, в которой все буквенные символы переведены в нижний регистр.

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
echo $str; // выводит: mary had a little lamb and she loved it so
```

strtoupper — преобразует строку в верхний регистр

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper($str);
echo $str; // выводит: MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

Функции обработки строк

trim — удаляет пробелы из начала и конца строки

Эта функция возвращает строку str с удаленными из начала и конца строки пробелами.

```php
$str = "  some text  ";

echo trim($str); // "some text"
```

- " " (ASCII 32 (0x20)), символ пробела.
- "\t" (ASCII 9 (0x09)), символ табуляции.
- "\n" (ASCII 10 (0x0A)), символ перевода строки.
- "\r" (ASCII 13 (0x0D)), символ возврата каретки.
- "\0" (ASCII 0 (0x00)), NUL-байт.
- "\x0B" (ASCII 11 (0x0B)), вертикальная табуляция.

Функции обработки строк

substr — возвращает подстроку

substr (string string, int start [, int length])
substr() возвращает подстроку строки string длиной length, начинающегося с start символа по счету.

Если start неотрицателен, возвращаемая подстрока начинается в позиции start от начала строки, считая от нуля. Например, в строке 'abcdef', в позиции 0находится символ 'a', в позиции 2 - символ 'c', и т.д.

```
$rest = substr("abcdef", 1);    // возвращает "bcdef"
$rest = substr("abcdef", 1, 3); // возвращает "bcd"
$rest = substr("abcdef", 0, 4); // возвращает "abcd"
$rest = substr("abcdef", 0, 8); // возвращает "abcdef"
```

```
$string = 'abcdef';
echo $string{0};                // выводит a
echo $string{3};                // выводит d
```

Функции обработки строк

substr —возвращает подстроку

substr (string **string**, int **start** [, int **length**])

```php
$rest = substr("abcdef", -1);    // возвращает "f"
$rest = substr("abcdef", -2);    // возвращает "ef"
$rest = substr("abcdef", -3, 1); // возвращает "d"

$rest = substr("abcdef", 0, -1);  // возвращает "abcde"
$rest = substr("abcdef", 2, -1);  // возвращает "cde"
$rest = substr("abcdef", 4, -4);  // возвращает ""
$rest = substr("abcdef", -3, -1); // возвращает "de"
```

Функции обработки строк

ucwords — преобразует в верхний регистр первый символ каждого слова в строке

```
$foo = 'hello world!';
$foo = ucwords($foo);                    // Hello World!

$bar = 'HELLO WORLD!';
$bar = ucwords($bar);                    // HELLO WORLD!
$bar = ucwords(strtolower($bar)); // Hello World!
```

Функции обработки строк

ucfirst — преобразует первый символ строки в верхний регистр

```php
$foo = 'hello world!';
$foo = ucfirst($foo);                  // Hello world!

$bar = 'HELLO WORLD!';
$bar = ucfirst($bar);                  // HELLO WORLD!
$bar = ucfirst(strtolower($bar)); // Hello world!
```

Функции для работы с массивами

Функции для работы с массивами

array_chunk — разбить массив на части

Функция array_chunk() разбивает массив на несколько массивов размером n значений.

Последний массив из полученных может содержать меньшее количество значений, чем указано. В качестве результата функция возвращает многомерный массив с индексами, начинающимися с нуля и элементами, которыми являются массивы, полученные в результате разбивки.

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

Функции для работы с массивами

array_chunk — разбить массив на части

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
```

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

Sibers®

Функции для работы с массивами

array_combine — создать новый массив, используя один массив в качестве ключей, а другой в качестве соответствующих значений.

```php
$a = array('green', 'red', 'yellow');
$b = array('avocado', 'apple', 'banana');
$c = array_combine($a, $b);

print_r($c);
```

```
Array
(
    [green] => avocado
    [red] => apple
    [yellow] => banana
)
```

Функции для работы с массивами

array_keys — выбрать все ключи массива

```php
$array = array (0 => 100, "color" => "red");
print_r(array_keys ($array));

$array = array ("blue", "red", "green", "blue", "blue");
print_r(array_keys ($array, "blue"));
```

```
Array
(
    [0] => 0
    [1] => color
)
```

```
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
```

Функции для работы с массивами

array_key_exists — проверить, присутствует ли в массиве указанный ключ или индекс

Функция array_key_exists() возвращает TRUE, если в массиве присутствует указанное значение ключ. Параметр ключ может быть любым значением, которое подходит для описания индекса массива.

```php
$search_array = array("first" => 1, "second" => 4);

if (array_key_exists("first", $search_array)) {
    echo "The 'first' element is in the array";
}
```

Функции для работы с массивами

in_array — проверить, присутствует ли в массиве значение

```php
$os = array("Mac", "NT", "Irix", "Linux");

if (in_array("Irix", $os)) {
    echo "Got Irix";
}
if (in_array("mac", $os)) {
    echo "Got mac";
}
```

```php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if (in_array(array('p', 'h'), $a)) {
    echo "'ph' найдено";
}
```

Функции для работы с массивами

array_push — добавить один или несколько элеметов в конец массива

array_push() использует array как стэк, и добавляет переданные значения в конец массива array. Длина array увеличивается на количество переданных значений.

Имеет тот же эффект, что и выражение:

```
$array[] = $var;
```

```
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");

print_r($stack);
```

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

Функции для работы с массивами

array_reverse — возвращает массив с элементами в обратном порядке

Функция array_reverse() берёт массив array и возвращает новый массив, порядок элементов в котором обратный исходному, сохраняя ключи, если параметр preserve_keys равен TRUE.

```php
$input  = array("php", 4.0, array("green", "red"));

$result = array_reverse($input);
$result_keyed = array_reverse($input, true);
```

```
Array
(
    [0] => Array
        (
            [0] => green
            [1] => red
        )

    [1] => 4
    [2] => php
)
```

```
Array
(
    [2] => Array
        (
            [0] => green
            [1] => red
        )

    [1] => 4
    [0] => php
)
```

Функции для работы с массивами

array_search — осуществляет поиск данного значения в массиве и возвращает соответствующий ключ в случае удачи

```php
$array = array(
            0 => 'blue',  1 => 'red', 2 => 'grey',
            3 => 'green', 4 => 'red'
        );

$key = array_search('red', $array);   // $key = 1;
$key = array_search('grey', $array);  // $key = 2;
$key = array_search('green', $array); // $key = 3;
```

Функции для работы с массивами

array_values — выбрать все значения массива

array_values() возвращает индексный массив, содержащий все значения массива input.

```php
$array = array("size" => "XL", "color" => "gold");
print_r(array_values($array));
```

```
Array
(
    [0] => XL
    [1] => gold
)
```

Функции для работы с массивами

arsort — отсортировать массив в обратном порядке, сохраняя ключи

Эта функция сортирует массив в обратном порядке таким образом, что сохраняются отношения между ключами и значениями. Она полезна, в основном, при сортировке ассоциативных массивов, когда важно сохранить отношение ключ => значение.

```php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");
arsort($fruits);
reset($fruits);
while (list($key, $val) = each($fruits)) {
    echo "$key = $val\n";
}

//a = orange, d = lemon, b = banana, c = apple
```

Функции для работы с массивами

ksort — отсортировать массив по ключам

Сортирует массив по ключам, сохраняя отношения между ключами и значениями. Функция полезна, в основном, для работы с ассоциативными массивами.

```php
$fruits = array("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort($fruits);
reset($fruits);
while (list($key, $val) = each($fruits)) {
    echo "$key = $val";
}

//a = orange, b = banana, c = apple, d = lemon
```

Функции для работы с массивами

uksort — отсортировать массив по ключам, используя пользовательскую функцию для сравнения ключей

uksort() отсортирует массив, используя для сравнения его ключей функцию, определённую пользователем. Если массив должен быть отсортирован по какому-либо необычному признаку, вы должны использовать эту функцию.

```php
function cmp($a, $b)
{
    if ($a == $b) {
        return 0;
    }
    return ($a > $b) ? -1 : 1;
}

$a = array(4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");
uksort($a, "cmp");

//20: twenty, 10: ten, 4: four, 3: three
```

Заключение:

- в пятой лекции мы рассмотрели условные операторы и конструкции выбора в PHP
- познакомились с шестью циклами
- рассмотрели 16 функций обработки строк и 13 функций для работы с массивами

В следующей лекции:

PHP и Cookies.
Сессии в PHP.