## **Общие правила построения PHP-программы**

PHP-программы состоят из простого текста, поэтому набирать их можно в любом текстовом редакторе. Популярные HTML-редакторы имеют встроенную поддержку для редактирования PHP-программ.

Расширение файлов PHP-программ по умолчанию в PHP4 - **.php**. На основании этого расширения сервер распознает файл как PHP-программу и запускает интерпретатор.

PHP-программа должна быть отделена от обычного HTML-текста. Существует четыре стиля обрамления PHP-кода:

|  |  |  |
| --- | --- | --- |
| **Стиль** | **Открывающий тег** | **Закрывающий тег** |
| Сокращенный | <? | ?> |
| XML (стандартный) | <?php | ?> |
| ASP | <% | %> |
| SCRIPT (программный) | <SCRIPT LANGUAGE="php"> | </SCRIPT> |

Из перечисленных тегов только стандартные и программные работают в любой конфигурации PHP.

Так выглядит простейший смешанный документ, состоящий из HTML-текста и PHP-кода:

#### Пример 1

<html>

<head>

<title>Документ, состоящий из HTML-текста и PHP-кода</title>

</head>

<body>

<?php

echo "А вот и PHP!<br>";

?>

Всё очень просто!

</body>

</html>

Здесь содержится один PHP-оператор **echo**. Этот оператор передает строку-аргумент "А вот и PHP!<br>" в HTML-страницу, которую генерирует сервер. При этом тег **<br>** сделает свое дело, т.е. осуществит переход на новую строку.

**РЕЗУЛЬТАТ ПРИМЕРА 1:**

А вот и PHP!

Всё очень просто!

Для разделения операторов (по аналогии с Си) используется точка с запятой.

Комментарии в PHP-программе могут быть трех стилей:

/\* Многострочный комментарий

в стиле классического Си \*/

// Однострочный комментарий в стиле С++

# Однострочный комментарий в стиле Perl

В одном документе можно чередовать HTML-текст и блоки PHP-операторов сколько угодно раз. При этом все переменные, функции и классы, определенные в первом блоке, будут доступны и в последующих блоках.

# Основы языка PHP

## **Константы**

**Константы** определяются в PHP-программе с помощью функции **define( )**. Например:

define("PI", 3.1415927);

После определения константа не может быть изменена. В имени константы обычно используются только заглавные буквы.

PHP имеет ряд **предопределенных констант**. Например:

* **\_FILE\_** содержит имя файла, включая полный путь, который в данный момент читает PHP
* **\_LINE\_** содержит номер строки этого файла
* **\_\_DIR\_\_** представляет только путь к файлу
* **\_\_CLASS\_\_** представляет имя текущего класса
* **\_\_FUNCTION\_\_** представляет имя текущей функции
* **\_\_METHOD\_\_** представляет имя текущего метода
* **\_\_NAMESPACE\_\_** представляет имя текущего пространства имен

## **Переменные**

Имя любой переменной в PHP начинается со знака **$**. Имена переменных *чувствительны к регистру*символов.

*Тип переменной* не требуется задавать специально. Конкретный тип переменной устанавливается и меняется в ходе выполнения программы.

PHP поддерживает восемь типов данных:

* логический (принимает значения **true** или **false**);
* целое число;
* вещественное число с плавающей точкой;
* строка;
* объект;
* массив;
* ресурс (специальный тип);
* null (специальный тип).

Тип переменной можно *проверить* с помощью функции [**gettype()**](https://htmlweb.ru/php/function/gettype.php).

#### Пример 1

<html>

<head>

<title>Автоматическое изменение и проверка типа переменной</title>

</head>

<body>

<?php

$t = "0";

echo "t = $t -- ".gettype($t)."<br>";

$t += 2;

echo "t = $t -- ".gettype($t)."<br>";

$t = $t + 3.5;

echo "t = $t -- ".gettype($t)."<br>";

$t = 5 + "5 поросят";

echo "t = $t -- ".gettype($t)."<br>";

$t = 5.0 + "5 поросят";

echo "t = $t -- ".gettype($t);

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 1:**

t = 0 -- string

t = 2 -- integer

t = 5.5 -- double

t = 10 -- integer

t = 10 -- double

Тип переменной можно *изменить* с помощью функции [**settype()**](https://htmlweb.ru/php/function/settype.php).

#### Пример 2

<html>

<head>

<title>Изменение типа переменной</title>

</head>

<body>

<?php

$t = 3.14;

echo "t = $t -- ".gettype($t)."<br>";

settype($t,"string");

echo "t = $t -- ".gettype($t)."<br>";

settype($t,"integer");

echo "t = $t -- ".gettype($t)."<br>";

settype($t,"double");

echo "t = $t -- ".gettype($t)."<br>";

settype($t,"boolean");

echo "t = $t -- ".gettype($t);

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 2:**

t = 3.14 -- double

t = 3.14 -- string

t = 3 -- integer

t = 3 -- double

t = 1 -- boolean

*Приведение* типа переменной осуществляется с помощью операторов:

* **(bool)** - к логическому типу;
* **(int)** - к целому числу;
* **(double)** - к вещественному числу;
* **(string)** - к строке;
* **(array)** - к массиву;
* **(object)** - к объекту.

Отличие от изменения типа с помощью функции **settype()** состоит в том, что оператор приведения создает временную копию нового типа, оставляя саму переменную без изменений.

#### Пример 3

<html>

<head>

<title>Приведение типа переменной</title>

</head>

<body>

<?php

$t = 3.14;

$tmp = (string) $t;

echo "tmp = $tmp -- ".gettype($tmp)."<br>";

$tmp = (int) $t;

echo "tmp = $tmp -- ".gettype($tmp)."<br>";

$tmp = (double) $t;

echo "tmp = $tmp -- ".gettype($tmp)."<br>";

$tmp = (bool) $t;

echo "tmp = $tmp -- ".gettype($tmp)."<br>";

echo "t = $t -- ".gettype($t);

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 3:**

tmp = 3.14 -- string

tmp = 3 -- integer

tmp = 3.14 -- double

tmp = 1 -- boolean

t = 3.14 -- double

Иногда для упрощения логики программы удобнее использовать переменные имена переменных. PHP предоставляет такую возможность в виде **динамических переменных**. Динамической называют переменную, имя которой хранится в ней самой.

#### Пример 4

<html>

<head>

<title>Использование динамической переменной</title>

</head>

<body>

<?php

$t = "Всем"; // переменной t присваиваем значение "Всем"

$$t = "привет!"; // Переменной "Всем" присваиваем значение "привет!"

echo "$$t ".$$t."<br>";

echo "$t ${$t}<br>";

echo "$t $Всем";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 4:**

$Всем привет!

Всем привет!

Всем привет!

В PHP возможно обращение к одной и той же переменной с использованием различных имен. Для реализации этого используются **ссылки**. Ссылки позволяют двум или большему количеству переменных ссылаться на одну и ту же область памяти.

#### Пример 5

<html>

<head>

<title>Создание ссылки на переменную</title>

</head>

<body>

<?php

$t1 = 96;

$t2 = **&**$t1;

echo "t1 = $t1; t2 = $t2<br>";

$t1 = 315;

echo "t1 = $t1; t2 = $t2";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 5:**

t1 = 96; t2 = 96

t1 = 315; t2 = 315

## **Операторы**

Операторы PHP напоминают общеизвестные операторы языка Си.

### **Унарные операторы**

|  |  |
| --- | --- |
| - | Изменение знака на противоположный |
| ! | Дополнение. Используется для реверсирования значения логических переменных |
| ++ | Увеличение значения переменной. Может применяться и как префикс, и как суффикс |
| -- | Уменьшение значения переменной. Может применяться и как префикс, и как суффикс |

### **Арифметические операторы**

|  |  |
| --- | --- |
| - | Вычитание |
| + | Сложение |
| \* | Умножение |
| / | Деление |
| % | Остаток от деления |

### **Оператор конкатенации**

Оператор конкатенации "**.**" присоединяет правую строку к левой.

#### Пример 6

<html>

<head>

<title>Конкатенация строк</title>

</head>

<body>

<?php

$a = "Всем";

$b = $a." привет!";

echo $b;

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 6:**

Всем привет!

Оператор конкатенации обрабатывает операнды любых типов как строки. Результат его выполнения всегда является строкой.

### **Операторы присваивания**

|  |  |
| --- | --- |
| = | Присваивание |
| += | Сложение ($n += 777; аналогично $n = $n + 777;) |
| -= | Вычитание ($n -= 777; аналогично $n = $n - 777;) |
| \*= | Умножение |
| /= | Деление |
| %= | Остаток от деления |
| .= | Конкатенация ($n .= "777"; аналогично $n = $n."777";) |

### **Битовые операторы**

Битовые операторы позволяют изменять отдельные биты целых чисел.

|  |  |
| --- | --- |
| & | И |
| | | ИЛИ |
| ^ | Исключающее ИЛИ |
| ~ | Инверсия |
| >> | Сдвиг вправо |
| << | Сдвиг влево |

### **Операторы сравнения**

|  |  |
| --- | --- |
| > | Больше ( Больше ли первое значение, чем второе?) |
| >= | Больше или равно ( Верно ли, что первое значение не меньше второго?) |
| < | Меньше ( Меньше ли первое значение, чем второе?) |
| <= | Меньше или равно (Верно ли, что первое значение не больше второго?) |
| == | Равно (Равнозначны ли значения двух переменных?) |
| === | Идентично (Одинаковы ли как значения, так и типы двух переменных?) |
| != , <> | Не равно (Не равны ли значения двух переменных?) |
| !== | Не идентично (Не одинаковы ли значения или типы данных двух переменных?) |

### **Логические операторы**

Логические операторы отличаются от битовых тем, что работают не с числами, а с логическими значениями: TRUE и FALSE.

|  |  |
| --- | --- |
| and | И |
| or | ИЛИ |
| xor | Исключающее ИЛИ |
| ! | Инверсия |
| >> | Сдвиг вправо |
| << | Сдвиг влево |
| && | И |
| || | ИЛИ |

Логические операторы "И" и "ИЛИ" имеют два формата. Это не синонимы. Дело в том, что оператор **or** имеет приоритет ниже, чем **||**, а **and** - ниже, чем **&&**. Таким образом, при построении сложных условных выражений можно обойтись без скобок. Однако, в таком способе указания порядка проще и запутаться.

### **Проверка содержимого переменной**

Иногда необходимо проверить, существует ли переменная или какое она имеет значение. Ниже приведены функции, позволяющие выполнить такие действия.

isset($имя\_переменной) #Истина, если переменная объявлена даже без присваивания значения.  
empty($имя\_переменной) #Истина, если значение переменной равно нулю или пустой строке, либо переменная не объявлена.

РНР также позволяет проверить тип переменной. Например, для того чтобы проверить, является ли переменная целочисленной, следует воспользоваться функцией

is\_int($number)

Результатом выполнения этой функции является TRUE, если переменная $number имеет тип integer. Рассмотрим подобные функции.

is\_array ($var2) проверяет, является ли переменная $var2 массивом.  
is\_f loat ($number) проверяет, является ли переменная $number числом с плавающей точкой.  
is\_null ($varl) проверяет, равно ли значение переменной $varl нулю  
is\_numeric($string) проверяет, является ли переменная $string числовой строкой.  
is\_string ($string) проверяет, является ли переменная $string строкой.

Для проверки обратных условий следует воспользоваться символом восклицания (!). Например, при обработке следующего выражения будет получено значение TRUE, если переменная не объявлена:

! isset($имя\_переменной)

## **Управляющие структуры**

### **Условные операторы**

В языке PHP два условных оператора: **if** и **?**.

Существует три типа оператора **if**. Первый тип - базовый условный оператор. Например:

if ($a > $b)

{

echo "А больше Б";

}

Альтернативный синтаксис оператора **if**:

if ($a > $b):

echo "А больше Б";

endif;

Второй тип - условный оператор **if-else**. Например:

if ($a > $b)

{

echo "А больше Б";

}

else

{

echo "А не больше Б";

}

Альтернативный синтаксис оператора **if-else**:

if ($a > $b):

echo "А больше Б";

else:

echo "А не больше Б";

endif;

Третий тип - условный оператор **if-elseif**. Например:

if ($a > $b)

{

echo "А больше Б";

}

elseif ($a == $b)

{

echo "А равно Б";

}

else

{

echo "А меньше Б";

}

Альтернативный синтаксис оператора **if-elseif**:

if ($a > $b):

echo "А больше Б";

elseif ($a == $b):

echo "А равно Б";

else:

echo "А меньше Б";

endif;

Условный оператор **?** возвращает одно из двух значений, разделенных двоеточием. Использование оператора **?** может сделать более компактным текст программы. Например:

$text = ($a == $b) ? "А равно Б" : "А не равно Б";

echo $text;

### **Оператор выбора**

Оператор выбора **switch** оценивает одно выражение и в зависимости от его значения выполняет один из нескольких блоков программы. Выражение в операторе **switch** чаще всего бывает простой переменной. Например:

switch ( $a )

{

case 1:

echo "А равно 1";

break;

case 2:

echo "А равно 2";

break;

case 3:

echo "А равно 3";

break;

default:

echo "А не равно ни 1, ни 2, ни 3";

}

Альтернативный синтаксис оператора **switch**:

switch ( $a ):

case 1:

echo "А равно 1";

break;

case 2:

echo "А равно 2";

break;

case 3:

echo "А равно 3";

break;

default:

echo "А не равно ни 1, ни 2, ни 3";

endswitch;

### **Цикл с параметром**

Цикл с параметром **for** относится к наиболее старому и заслуженному виду цикла.

#### Пример 7

<html>

<head>

<title>Цикл for</title>

</head>

<body>

<?php

for ($a = 11; $a <= 19; $a++)

{

echo "квадрат $a равен ".($a\*$a)."<br>";

}

?>

</body>

</html>

Альтернативный синтаксис оператора **for**:

for ($a = 11; $a <= 19; $a++):

echo "квадрат $a равен ".($a\*$a)."<br>";

endfor;

### **Циклы с условием**

В языке PHP сущеcтвует два типа цикла с условием:

* **while** - цикл с предусловием;
* **do .. while** - цикл с постусловием.

Оператор **while** оценивает значение условия и, если оно истинно, выполняет действия в фигурных скобках (тело цикла). Как только значение условия станет ложным, выполнение цикла прекращается.

#### Пример 8

<html>

<head>

<title>Цикл while</title>

</head>

<body>

<?php

$a = 11;

while ( $a <= 19 )

{

echo "квадрат $a равен ".($a\*$a)."<br>";

$a++;

}

?>

</body>

</html>

Замечание для продвинутых: сокращенная запись не сработает как ожидается:

Альтернативный синтаксис оператора **while**:

while ( $a <= 19 ):

echo "квадрат $a равен ".($a\*$a)."<br>";

$a++;

endwhile;

Цикл **do .. while** отличается от цикла **while** лишь тем, что истинность условия проверяется не *до*, а *после*выполнения тела цикла.

#### Пример 9

<html>

<head>

<title>Цикл do .. while</title>

</head>

<body>

<?php

$a = 11;

do {

echo "квадрат $a равен ".($a\*$a)."<br>";

$a++;

}

while ( $a <= 19 );

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРОВ 7-9:**

квадрат 11 равен 121

квадрат 12 равен 144

квадрат 13 равен 169

квадрат 14 равен 196

квадрат 15 равен 225

квадрат 16 равен 256

квадрат 17 равен 289

квадрат 18 равен 324

квадрат 19 равен 361

В организации цикла могут участвовать еще два оператора: **break** (выход из цикла) и **continue**(переход на следующий шаг).

# Использование массивов

Основное назначение массивов в PHP - организация групп связанных значений. Каждый элемент массива имеет индекс (ключ) и значение. Индекс элемента массива указывается в квадратных скобках после имени массива. Для того, чтобы обратиться к пятому элементу массива **$array1**, надо написать:

$array1[5]

Помните, что по умолчанию массив начинается не с первого элемента, а с нулевого.

Индекс может быть как числом, так и текстовой строкой. Массив со строковыми индексами называют *ассоциативным*, а сами индексы - именами элементов. Например, цены на товары хранятся в ассоциативном массиве **$prices**, индексами которого являются наименования товаров. Чтобы получить значение цены на конкретный товар, надо написать:

$prices["яйца 10 шт. 1 сорт"]

Значение элемента массива может иметь любой тип. Возможна организация многомерных массивов, так как элемент массива может в свою очередь являться массивом. Примеры обращений к элементам многомерных массивов:

$array3[0][4][1]

$цены["овощи"]["помидор"]

$список[1]["фамилия"]

## **Создание массива**

Массив можно создать с помощью функции [**array()**](https://htmlweb.ru/php/function/array.php), параметры которой и составляют массив. Параметры могут задаваться парами "ключ=>значение". Если при создании массива ключ не указывается, то индекс определяется положением элемента в массиве (начиная с 0). Например:

$рост = array (174, 181, 166); //Массив с индексацией, начинающейся с нуля

$цена = array ("помидоры" => 15, "огурцы" => 12); //Ассоциативный массив

$данные = array (

"Иванов" => array ("рост" => 174, "вес" => 68),

"Петров" => array ("рост" => 181, "вес" => 90),

"Сидоров" => array ("рост" => 166, "вес" => 73)); //Двухмерный массив

Массивы можно создать и другим способом - непосредственно. Например:

$фрукты[] = "яблоко";

$фрукты[] = "груша";

$фрукты[] = "слива";

$цена["помидор"] = 15;

$цена["огурец"] = 12;

Индексы элементов неассоциативного массива можно не указывать. PHP автоматически вычислит их. Если же указать индексы таким образом:

$фрукты[0] = "груша";

$фрукты[5] = "яблоко";

то в массиве будет два элемента, причем последний с индексом 5. Элементы 1 - 4 не инициализируются.

Можно создать массив с помощью функции [**array()**](https://htmlweb.ru/php/function/array.php), а затем добавить к нему новый элемент:

$фрукты = array("яблоко","груша","слива");

...

$фрукты[] = "персик";

## **Подсчет количества элементов**

Количество элементов в массиве можно определить с помощью функций [**count()**](https://htmlweb.ru/php/function/count.php) или [**sizeof()**](https://htmlweb.ru/php/function/sizeof.php).

#### Пример 1

<html>

<head>

<title>Размер массива</title>

</head>

<body>

<?php

$фрукты = array("яблоко", "груша", "слива", "персик");

echo "Размер массива \$фрукты равен ".count($фрукты)."<br>";

echo "Последний элемент массива \$фрукты - ".$фрукты[count($фрукты)-1]."<br>";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 1:**

Размер массива $фрукты равен 4

Последний элемент массива $фрукты - персик

Для доступа к последнему элементу надо вычесть 1 из размера массива, так как индексация массива начинается с нуля. Для вывода зарезервированного символа "$" перед знаком доллара стоит символ обратной косой черты "\".

Частоту вхождения элементов в массив можно определить с помощью функции [**array\_count\_values()**](https://htmlweb.ru/php/function/array_count_values.php). Эта функция возвращает массив, в котором ключами являются элементы исследуемого массива, а значениями - частоты их вхождения в исследуемый массив.

#### Пример 2

<html>

<head>

<title>Размер массива</title>

</head>

<body>

<?php

$фрукты = array("яблоко", "груша", "слива", "персик", "груша");

print\_r (array\_count\_values($фрукты));

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 2:**

Array ( [яблоко] => 1 [груша] => 2 [слива] => 1 [персик] => 1 )

Функция [**print\_r()**](https://htmlweb.ru/php/function/print_r.php) отображает ключи и значения массива, указанного в аргументе.

## **Просмотр массива в цикле**

Для итерационного просмотра содержимого массива служит функция [**foreach**](https://htmlweb.ru/php/function/foreach.php). С ее помощью можно просмотреть и простой (проиндексированный числами) массив, и ассоциативный, и многомерный.

#### Пример 3

<html>

<head>

<title>Просмотр массива</title>

</head>

<body>

<?php

$фрукты = array("яблоко", "груша", "слива", "персик", "груша");

foreach ($фрукты as $фрукт)

{

echo "$фрукт<br>";

}

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 3:**

яблоко

груша

слива

персик

груша

#### Пример 4

<html>

<head>

<title>Просмотр ассоциативного массива</title>

</head>

<body>

<?php

$цена = array ("помидоры" => 15, "огурцы" => 12);

foreach ($цена as $овощи => $руб)

{

echo "$овощи стоят $руб руб.<br>";

}

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 4:**

помидоры стоят 15 руб.

огурцы стоят 12 руб.

#### Пример 5

<html>

<head>

<title>Просмотр двухмерного массива</title>

</head>

<body>

<?php

$данные = array (

"Иванов" => array ("рост" => 174, "вес" => 68),

"Петров" => array ("рост" => 181, "вес" => 90),

"Сидоров" => array ("рост" => 166, "вес" => 73));

foreach ($данные as $фамилия => $данные1)

{

echo "<br>$фамилия:<br>";

foreach ($данные1 as $параметр => $pp)

{

echo "$параметр = $pp<br>";

}

}

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 5:**

Иванов:

рост = 174

вес = 68

Петров:

рост = 181

вес = 90

Сидоров:

рост = 166

вес = 73

[**print\_r**](https://htmlweb.ru/php/function/print_r.php) - Вывод всех эллементов массива. Удобная функция для отладки.

<?php

$a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x','y','z'));

**print\_r** ($a);

?>

**РЕЗУЛЬТАТ ПРИМЕРА:**

Array

(

[a] => apple

[b] => banana

[c] => Array

(

[0] => x

[1] => y

[2] => z

)

)

## **Управление массивами**

Для управление массивами в PHP существует целый ряд специализированных функций. Наиболее употребимыми являются:

* **+** - объединение массивов. $ar=$ar1+$ar2

* **==** - сравнение массивов. Оператор вернёт true если все пары (ключ:значение) из первого массива совпадут с содержанием второго массива.

* **===** - идентичность массивов. Оператор вернёт true если:
  + содержат одно и тоже количество элементов;
  + ключи и значения элементов одинаковы в обоих массивах;
  + порядок элементов;
  + абсолютное совпадение по типам данных у значений элементов.

Если ключи одного массива типа string, а второго integer и все значения совпадают, то оператор вернёт true. Если же вместо integer воспользоваться float, то результат будет false.

* [**array\_merge()**](https://htmlweb.ru/php/function/array_merge.php) - объединение (слияние) массивов. Последующее значение элемента перекрывает предыдущее, если ключи строковые и одинаковые.

#### Пример 6

<html>

<head>

<title>Слияние массивов</title>

</head>

<body>

<?php

$мнение1 = array (1, "Катя" => "умная", 2, "Женя" => "красивая", 3);

$мнение2 = array (4, "Женя" => "глупая", 5, "Катя" => "милая");

echo "1 + 2:<br>";

print\_r (array\_merge ($мнение1, $мнение2));

echo "<br>2 + 1:<br>";

print\_r (array\_merge ($мнение2, $мнение1));

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 6:**

1 + 2:

Array ( [0] => 1 [Катя] => милая [1] => 2 [Женя] => глупая [2] => 3 [3] => 4 [4] => 5 )

2 + 1:

Array ( [0] => 4 [Женя] => красивая [1] => 5 [Катя] => умная [2] => 1 [3] => 2 [4] => 3 )

* [**array\_shift()**](https://htmlweb.ru/php/function/array_shift.php) - удаление первого элемента.

#### Пример 7

<html>

<head>

<title>Удаление первого элемента массива</title>

</head>

<body>

<?php

$фрукты = array("яблоко", "груша", "слива", "персик");

echo "В вазе лежали:";

foreach ($фрукты as $tmp)

{

echo " $tmp";

}

echo "<br>Первым съели ".array\_shift($фрукты)."<br>";

echo "В вазе остались:";

foreach ($фрукты as $tmp)

{

echo " $tmp";

}

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 7:**

В вазе лежали: яблоко груша слива персик

Первым съели яблоко

В вазе остались: груша слива персик

* [**array\_pop()**](https://htmlweb.ru/php/function/array_pop.php) - удаление последнего элемента.

#### Пример 8

<html>

<head>

<title>Удаление последнего элемента массива</title>

</head>

<body>

<?php

$фрукты = array("яблоко", "груша", "слива", "персик");

echo "В вазе лежали:";

foreach ($фрукты as $tmp)

{

echo " $tmp";

}

echo "<br>Съели ".array\_pop($фрукты)."<br>";

echo "В вазе остались:";

foreach ($фрукты as $tmp)

{

echo " $tmp";

}

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 8:**

В вазе лежали: яблоко груша слива персик

Съели персик

В вазе остались: яблоко груша слива

* [**array\_push()**](https://htmlweb.ru/php/function/array_push.php) - добавление элементов в конец массива.

#### Пример 9

<html>

<head>

<title>Добавление элементов в конец массива</title>

</head>

<body>

<?php

$a = array(1, 2, 3);

echo "Массив \$a:";

foreach ($a as $tmp) echo " $tmp";

$k = array\_push($a, 4, 5);

echo "<br>Массив \$a после добавления:";

foreach ($a as $tmp) echo " $tmp";

echo "<br>Его длина = $k";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 9:**

Массив $a: 1 2 3

Массив $a после добавления: 1 2 3 4 5

Его длина = 5

Имейте в виду, что если в качестве второго аргумента функции [**array\_push()**](https://htmlweb.ru/php/function/array_push.php) передать массив, то этот массив будет добавлен как элемент, т.е. будет создан двумерный массив.

* [**array\_unshift()**](https://htmlweb.ru/php/function/array_unshift.php) - добавление элементов в начало массива.

#### Пример 10

<html>

<head>

<title>Добавление элементов в начало массива</title>

</head>

<body>

<?php

$a = array(1, 2, 3);

echo "Массив \$a:";

foreach ($a as $tmp) echo " $tmp";

$k = array\_unshift($a, 4, 5);

echo "<br>Массив \$a после добавления:";

foreach ($a as $tmp) echo " $tmp";

echo "<br>Его длина = $k";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 10:**

Массив $a: 1 2 3

Массив $a после добавления: 4 5 1 2 3

Его длина = 5

* [**array\_slice()**](https://htmlweb.ru/php/function/array_slice.php) - выделение фрагмента массива.

Имеет три параметра: сам массив, смещение и длину выделяемого фрагмента. При положительном смещении отсчет выполняется от начала массива, при отрицательном - от конца. При положительной длине результирующий фрагмент будет содержать заданное число элементов. При отрицательной длине последним элементом фрагмента станет тот, который находится на указанном расстоянии от конца массива. Если параметр опущен, то фрагмент будет содержать все элементы от начального смещения и до конца массива.

#### Пример 11

<html>

<head>

<title>Выделение фрагмента массива</title>

</head>

<body>

<?php

$a = array("a", "b", "c", "d", "e", "f");

echo "\$a:"; foreach ($a as $t) echo " $t";

$b = array\_slice($a, 2);

echo "<br>array\_slice(\$a, 2):"; foreach ($b as $t) echo " $t";

$b = array\_slice($a, 2, -1);

echo "<br>array\_slice(\$a, 2, -1):"; foreach ($b as $t) echo " $t";

$b = array\_slice($a, -2, 1);

echo "<br>array\_slice(\$a, -2, 1):"; foreach ($b as $t) echo " $t";

$b = array\_slice($a, 0, 3);

echo "<br>array\_slice(\$a, 0, 3):"; foreach ($b as $t) echo " $t";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 11:**

$a: a b c d e f

array\_slice($a, 2): c d e f

array\_slice($a, 2, -1): c d e

array\_slice($a, -2, 1): e

array\_slice($a, 0, 3): a b c

## **Сортировки**

Сортировать можно как простые, так и ассоциативные массивы. Для сортировки массивов в PHP существуют определенные функции:

* [**sort()**](https://htmlweb.ru/php/function/sort.php) - сортирует массив в алфавитном порядке, если хотя бы один из его элементов является строкой, и в числовом порядке, если все его элементы - числа.
* [**rsort()**](https://htmlweb.ru/php/function/rsort.php) - работает как **sort( )**, но в обратном порядке.
* [**asort()**](https://htmlweb.ru/php/function/asort.php) - сортирует ассоциативный массив; работает как **sort( )**, но сохраняет имена элементов.
* [**arsort()**](https://htmlweb.ru/php/function/arsort.php) - работает как **asort( )**, но в обратном порядке.
* [**ksort()**](https://htmlweb.ru/php/function/ksort.php) - сортирует ассоциативный массив по именам элементов.
* [**krsort()**](https://htmlweb.ru/php/function/krsort.php) - работает как **ksort( )**, но в обратном порядке.

#### Пример 12

<html>

<head>

<title>Сортировка ассоциативного массива</title>

</head>

<body>

<?php

$a = array("первый" => 6, "второй" => 2, "третий" => 1);

echo "\$a:<br>"

asort ($a);

echo "asort (\$a):<br>"

foreach ($a as $k => $t) echo " $k = $t<br>";

ksort ($a);

echo "ksort (\$a):<br>"

foreach ($a as $k => $t) echo "$k = $t<br>";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 12:**

$a:

первый = 6

второй = 2

третий = 1

asort ($a):

третий = 1

второй = 2

первый = 6

ksort ($a):

**в**торой = 2

**п**ервый = 6

**т**ретий = 1

## **Поиск элемента**

Для проверки наличия элемента в массиве существуют функции:

* [**in\_array()**](https://htmlweb.ru/php/function/in_array.php) - если элемент найден, возвращает **true**, иначе - **false**.
* [**array\_search()**](https://htmlweb.ru/php/function/array_search.php) - если элемент найден, возвращает его ключ, иначе - **false**.

#### Пример 13

<html>

<head>

<title>Поиск элемента в массиве</title>

</head>

<body>

<?php

$a = array("первый" => 6, "второй" => 2, "третий" => 1);

if (in\_array (2, $a)) echo "2 нашли!<br>";

echo "ключ найденного элемента - ".array\_search(2, $a);

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 13:**

2 нашли!

ключ найденного элемента - второй

## **Использование итератора для обхода многомерного массива**

#### Пример 13

// Многомерный массив

$arr = [

["sitepoint", "phpmaster"],

["buildmobile", "rubysource"],

["designfestival", "cloudspring"],

"not an array"

];

$iter = new RecursiveArrayIterator($arr);

// Цикл по объекту

// Нужно создать экземпляр RecursiveIteratorIterator

foreach(new RecursiveIteratorIterator($iter) as $key => $value) {

echo $key . ": " . $value . "<br>";

}

**РЕЗУЛЬТАТ ПРИМЕРА 14:**

0: sitepoint

1: phpmaster

0: buildmobile

1: rubysource

0: designfestival

1: cloudspring

3: not an array

# Суперглобальный массив $\_SERVER

В массив **$\_SERVER** PHP-интерпретатор помещает переменные, полученные от сервера. Без данных переменных сложно организовать полноценную поддержку Web-приложений. Ниже приводится описание наиболее важных элементов суперглобального массива **$\_SERVER**.

#### Замечание

 Просмотреть полный список элементов массива $\_SERVERможно либо при помощи функции print\_r(), которая распечатывает дамп массива либо при помощи функции phpinfo(), которая выводит информацию о PHP-интерпретаторе.

## **$\_SERVER['DOCUMENT\_ROOT']**

Элемент $\_SERVER['DOCUMENT\_ROOT'] содержит путь к корневой директории сервера, если скрипт выполняется в виртуальном хосте, в данном элементе указывается путь к корневой директории виртуального хоста. Т.е. в конфигурационном файле httpd.conf виртуальный хост имеет директиву DocumentRoot, которой присвоено значение "D:/main", элемент $\_SERVER['DOCUMENT\_ROOT'] будет содержать значение "D:main".

/var/www/htmlweb.ru/web

## **$\_SERVER['HTTP\_ACCEPT']**

В элементе $\_SERVER['HTTP\_ACCEPT'] описываются предпочтения клиента относительно типа документа. Содержимое этого элемента извлекается из HTTP-заголовка Accept, который присылает клиент серверу. Содержимое данного заголовка может выглядеть следующим образом

image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/msword, \*/\*

Заголовок Accept позволяет уточнить медиа-тип, который предпочитает получить клиент в ответ на свой запрос. Этот заголовок позволяет сообщить серверу, что ответ ограничен небольшим множеством предпочитаемых типов.

Символ \* используется для группирования типов в медиа-ряду. К примеру, символом \*/\* задается использование всех типов, а обозначение type/\* определяет использование всех подтипов выбранного типа type.

#### Замечание

Медиа-типы отделяются друг от друга запятыми.

Каждый медиа-ряд характеризуется также дополнительным набором параметров. Одним из них является так называемый относительный коэффициент предпочтения q, который принимает значения от 0 до 1, соответственно, от менее предпочитаемых типов к более предпочитаемым. Использование нескольких параметров q, позволяет клиенту сообщить серверу относительную степень предпочтения для того или иного медиа-типа.

#### Замечание

По умолчанию параметр q принимает значение 1. Кроме того, от медиа-типа он отделяется точкой с запятой.

Пример заголовка типа Accept:

Accept: audio/\*; q=0.2, audio/basic или реальный нынешний:

\*/\*

В данном заголовке первым идёт тип audio/\* включающий в себя все музыкальные документы и характеризующийся коэффициентом предпочтения 0.2. Через запятую указан тип audio/basic, для которого коэффициент предпочтения не указан и принимает значение по умолчанию равное единице. Цитируя [RFС2616](http://www.softtime.ru/) данный заголовок можно интерпретировать следующим образом: "Я предпочитаю тип audio/basic, но мне можно также слать документы любого другого audio-типа, если они будут доступны, после снижения коэффициента предпочтения более чем на 80 %".

Пример может быть более сложным. Accept: text/plain; q=0.5, text/html,   
        text/x-dvi; q=0.8, text/x-c

#### Замечание

Следует учитывать, что элемент $\_SERVER['HTTP\_ACCEPT'] содержит точно такую же информацию, но без начального заголовка Accept.

Этот заголовок интерпретируется следующим образом: Типы документов text/html и text/x-c являются предпочтительными, но если они недоступны, тогда клиент отсылающий данный запрос, предпочтёт text/x-dvi, а, если и его нет, то он может принять тип text/plain.

## **$\_SERVER['HTTP\_ACCEPT\_LANGUAGE']**

В элементе $\_SERVER['HTTP\_ACCEPT\_LANGUAGE'] описываются предпочтения клиента относительно языка. Данная информация извлекается из HTTP-заголовка Accept-Language, который присылает клиент серверу. Можно привести следующий пример:

Accept-Language: ru, en; q=0.7

Который можно интерпретировать следующим образом: клиент предпочитает русский язык, но в случае его отсутствия согласен принимать документы на английском. Элемент $\_SERVER['HTTP\_ACCEPT\_LANGUAGE'] будет содержать точно такую же информацию, но без заголовка Accept-Language:

ru, en; q=0.7

Содержимое элемента $\_SERVER['HTTP\_ACCEPT\_LANGUAGE'] можно использовать для определения национальной принадлежность посетителей. Однако результаты будут приблизительными, так как многие пользователи используют английские варианты браузеров, которые будут извещать сервер о том, что посетитель предпочитает лишь один язык - английский.

ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4

## **$\_SERVER['HTTP\_HOST']**

В элементе $\_SERVER['HTTP\_HOST'] содержится имя сервера, которое, как правило, совпадает с доменным именем сайта, расположенного на сервере. Как правило, имя, указанное в данном параметре совпадает с именем $\_SERVER['SERVER\_NAME']. В параметре приводится лишь доменное имя без названия протокола (http://), т.е.

htmlweb.ru

## **$\_SERVER['HTTP\_REFERER']**

В элементе $\_SERVER['HTTP\_REFERER'] приводится адрес страницы, с которой посетитель пришёл на данную страницу. Переход должен осуществляться по ссылке. Создадим две страницы index.php и page.php.

#### Страница index.php

echo "<a href=page.php>Ссылка на страницу PHP</a><br />";

echo "Содержимое $\_SERVER['HTTP\_REFERER'] - ".

$\_SERVER['HTTP\_REFERER']

Страница page.php будет аналогичного содержания, но ссылка будет указывать на страницу index.php.

#### Страница page.php

echo "<a href=index.php>Ссылка на страницу PHP</a><br />";

echo "Содержимое $\_SERVER['HTTP\_REFERER'] - ".

$\_SERVER['HTTP\_REFERER']

При переходе с одной страницы на другую, под ссылкой будет выводится адрес страницы, с которой был осуществлён переход.

А на эту страницу вы перешли с:

https://htmlweb.ru/php/php\_array.php

## **$\_SERVER['HTTP\_USER\_AGENT']**

Элемент $\_SERVER['HTTP\_USER\_AGENT'] содержит информацию о типе и версии браузера и операционной системы посетителя.

Вот типичное содержание этой строки: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)Наличие подстроки "MSIE 6.0" говорит о том, что посетитель просматривает страницу при помощи Internet Explorer версии 6.0. Строка "Windows NT 5.1" сообщает, что в качестве операционной системы используется Windows XP.

#### Замечание

Для Windows 2000 элемент $\_SERVER['HTTP\_USER\_AGENT'] выглядит следующим образом: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)') , в то время как для Windows XP - Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

Если посетитель воспользуется браузером Opera, то содержание $\_SERVER['HTTP\_USER\_AGENT'] может выглядеть следующим образом: Mozilla/4.0 (compatible; MSIE 5.0; Windows 98) Opera 6.04 [ru]Подстрока "MSIE 6.0" здесь так же присутствует, сообщая, что браузер Opera является совместимым с браузером Internet Explorer и использует те же динамические библиотеки Windows. Поэтому, при анализе строки, возвращаемой браузером, следует иметь в виду, что к Internet Explorer относится строка, содержащая подстроку "MSIE 6.0" и не содержащая подстроки "Opera". Кроме того, из данной строки можно заключить, что пользователь использует операционную систему Windows 98.

#### Замечание

Пользовательский агент браузера Firefox может выглядеть следующим образом Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8) Gecko/20051111 Firefox/1.5

При использовании браузера Netscape, содержание элемент $\_SERVER['HTTP\_USER\_AGENT'] может выглядеть следующим образом: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.4) Gecko/20030624 Netscape/7.1 Принадлежность к этому браузеру можно определить по наличию подстроки "Netscape". Кроме того, можно узнать, что посетитель выходит в Интернет, используя операционную версию Linux, с ядром, оптимизированным под Pentium IV, находясь в графической оболочке X-Window. Этот механизм удобно использовать для сбора статистической информации, которая позволяет дизайнерам оптимизировать страницы под наиболее распространенные браузеры.

Ваш HTTP\_USER\_AGENT:

Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36

[Определение и расшифровка User\_agent](https://htmlweb.ru/analiz/browser.php)

## **$\_SERVER['REMOTE\_ADDR']**

В элемент $\_SERVER['REMOTE\_ADDR'] помещается IP-адрес клиента. При тестировании на локальной машине - этот адрес будет равен 127.0.0.1. Однако при тестировании в сети переменная вернёт IP-адрес клиента или последнего прокси-сервера через который клиент попал на сервер. Если клиент использует прокси-сервер узнать его IP-адрес можно при помощи переменной окружения HTTP\_X\_FORWARDED\_FOR, значение которой можно получить при помощи функции getenv().

213.87.225.245

#### Замечание

Прокси-сервера являются специальными промежуточными серверами, предоставляющими специальный вид услуг: сжатие трафика, кодирование данных, адаптация под мобильные устройства и т.п. Среди множества прокси-серверов различают так называемые анонимные прокси-сервера, которые позволяют скрывать истинный IP-адрес клиента, такие сервера не возвращают переменной окружения HTTP\_X\_FORWARDED\_FOR.

#### Извлечение переменной окружения **HTTP\_X\_FORWARDED\_FOR**

echo @getenv(HTTP\_X\_FORWARDED\_FOR);

## **$\_SERVER['SCRIPT\_FILENAME']**

В элемент $\_SERVER['SCRIPT\_FILENAME'] помещается абсолютный путь к файлу от корня диска. Так, если сервер работает под управлением операционной системы Windows, то такой путь может выглядеть следующим образом "d:main estindex.php", т.е. путь указывается от диска, в UNIX-подобной операционной системы путь указывается от корневой директории /, например "/var/share/www/test/index.php".

/var/www/htmlweb.ru/web/index.php

## **$\_SERVER['SERVER\_NAME']**

В элемент $\_SERVER['SERVER\_NAME'] помещается имя сервера, как правило, совпадающее с доменным именем сайта, расположенного на нём. Например,

htmlweb.ru

Содержимое элемента $\_SERVER['SERVER\_NAME'] часто совпадает с содержимым элемента $\_SERVER['HTTP\_HOST']. Помимо имени сервера суперглобальный массив $\_SERVER позволяет выяснить ещё ряд параметров сервера, например IP-адрес сервера, прослушиваемый порт, какой Web-сервер установлен и версию HTTP протокола. Эта информация помещается в элементы $\_SERVER['SERVER\_ADDR'], $\_SERVER['SERVER\_PORT'], $\_SERVER['SERVER\_SOFTWARE'] и $\_SERVER['SERVER\_PROTOCOL'], соответственно. Ниже приводится пример с использованием данных элементов.

#### Использование элементов массива $\_SERVER

echo "Имя сервера - ".$\_SERVER['SERVER\_NAME']."<br />";

echo "IP-адрес сервера - ".$\_SERVER['SERVER\_ADDR']."<br />";

echo "Порт сервера - ".$\_SERVER['SERVER\_PORT']."<br />";

echo "Web-сервер - ".$\_SERVER['SERVER\_SOFTWARE']."<br />";

echo "Версия HTTP-протокола - ".$\_SERVER['SERVER\_PROTOCOL']."<br />";

Имя сервера - htmlweb.ru  
IP-адрес сервера - 89.108.87.148  
Порт сервера - 443  
Web-сервер - Apache/2.4.6  
Версия HTTP-протокола - HTTP/1.1

## **$\_SERVER['REQUEST\_METHOD']**

В элемент $\_SERVER['REQUEST\_METHOD'] помещается метод запроса, который применяется для вызова скрипта: GET или POST.

echo $\_SERVER['REQUEST\_METHOD'];

GET

## **$\_SERVER['QUERY\_STRING']**

В элемент $\_SERVER['QUERY\_STRING'] заносятся параметры, переданные скрипту, если строка запроса представляет собой адрес

echo $\_SERVER['QUERY\_STRING'];

ajax=1

Например при обращении к:  
**http://www.mysite.ru/test/index.php?id=1&test=wet&id\_theme=512** в элемент $\_SERVER['QUERY\_STRING'] попадёт весь текст после знака "?":

echo $\_SERVER['QUERY\_STRING'];

id=1&test=wet&id\_theme=512

## **$\_SERVER['PHP\_SELF']**

В элемент $\_SERVER['PHP\_SELF'] помещается имя скрипта, начиная от корневой директории виртуального хоста, т.е. если строка запроса представляет собой адрес **http://www.mysite.ru/test/index.php?id=1&test=wet&id\_theme=512** то элемент $\_SERVER['PHP\_SELF'] будет содержать фрагмент **"/test/index.php"**. Как правило, этот же фрагмент помещается в элемент $\_SERVER['SCRIPT\_NAME'].

## **$\_SERVER['REQUEST\_URI']**

В элемент $\_SERVER['REQUEST\_URI'] содержит имя скрипта, начиная от корневой директории виртуального хоста и параметры, т.е. если строка запроса представляет собой адрес:**http://www.mysite.ru/test/index.php?id=1&test=wet&id\_theme=512** то элемент $\_SERVER['REQUEST\_URI'] будет содержать фрагмент **"/test/index.php?id=1&test=wet&id\_theme=512"**. Для того, чтобы восстановить в скрипте полный адрес, который помещён в строке запроса, достаточно использовать комбинацию элементов массива $\_SERVER, представленную ниже

#### Полный адрес к скрипту

echo "http://".$\_SERVER['SERVER\_NAME'].$\_SERVER['REQUEST\_URI'];

# Функции в PHP

**Подсказка:**  
Для получения полного списка встроенных PHP функций в правой колонке нажмите кнопочку Поиск при пустой строке поиска "PHP поиск".  
Если Вы хотите узнать подробное **описание** с примером конкретной **PHP функции** укажите её в строке PHP поиск. →

В PHP существует две основные формы функций: встроенные и пользовательские.

Для определения пользовательской функции используется ключевое слово **function**.

#### Пример 1

<html>

<head>

<title>Определение и вызов функции</title>

</head>

<body>

<?php

function заголовок ($str, $n)

{

echo "<H$n>$str</H$n>";

}

заголовок ("Часть", 1);

заголовок ("Глава", 2);

заголовок ("Раздел", 3);

?>

</body>

</html>

Начиная с версии PHP 4.0 можно использовать значения аргументов, подставляемые по умолчанию.

#### Пример 2

<html>

<head>

<title>Функция со значением аргумента по умолчанию</title>

</head>

<body>

<?php

function заголовок ($str, $n = 2)

{

echo "<H$n>$str</H$n>";

}

заголовок ("Часть", 1);

заголовок ("Глава");

заголовок ("Раздел", 3);

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРОВ 1-2:**

Часть

## **Глава**

### **Раздел**

**Примечание:** Указанный ниже код будет отрабатывать не так, как Вы возможно ожидаете. Переменная $full во всех случаях будет true.

fun($ar, false);

function fun($row, $full = true){ ... }

## **Передача аргумента по ссылке**

Аргументы можно передавать **по ссылке**. Тогда все изменения, сделанные с аргументом в теле функции, отразятся на самой переменной.

#### Пример 3

<html>

<head>

<title>Передача аргумента по ссылке</title>

</head>

<body>

<?php

function add ( &$a, $b )

{

$a += $b;

$b += $b;

}

$af = 10;

$bf = 6;

echo "af = $af; bf = $bf";

add ( $af, $bf );

echo "<br>После вызова функции:<br>";

echo "af = $af; bf = $bf";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 3:**

af = 10; bf = 6

После вызова функции:

af = 16; bf = 6

## **Возврат значений**

Для возврата значений из функции используется оператор **return**. Можно вернуть несколько значений, если применить список [**list()**](https://htmlweb.ru/php/function/list.php).

#### Пример 4

<html>

<head>

<title>Функция, возвращающая список значений</title>

</head>

<body>

<?php

function ФИО()

{

return array ("Иванов", "Иван", "Иванович");

}

**list** ($фамилия, $имя, $отчество) = ФИО();

echo "<table>";

echo "<tr><td>фамилия: <td>$фамилия";

echo "<tr><td>имя: <td>$имя";

echo "<tr><td>отчество: <td>$отчество</table>";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 4:**

|  |  |
| --- | --- |
| фамилия: | Иванов |
| имя: | Иван |
| отчество: | Иванович |

## **Динамические функции**

В PHP можно использовать **динамические функции**. Это означает, что если некоторой переменной присвоено имя функции, то с этой переменной можно обращаться точно так же, как с самой функцией.

#### Пример 5

<html>

<head>

<title>Переменные функции</title>

</head>

<body>

<?php

function funct1 ()

{

echo "Мы внутри функции funct1( );<br>";

}

function funct2 ()

{

echo "Мы внутри функции funct2( );<br>";

}

$myfunct = "funct1";

$myfunct();

$myfunct = "funct2";

$myfunct();

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 5:**

Мы внутри функции funct1( );

Мы внутри функции funct2( );

## **Глобальные переменные**

Чтобы переменная, созданная вне функции, была доступна внутри функции, надо объявить ее глобальной с помощью инструкции **global**.

#### Пример 6

<html>

<head>

<title>Доступ к глобальной переменной</title>

</head>

<body>

<?php

$gg = 45;

function funct ()

{

global $gg;

echo "Глобальная переменная имеет значение - $gg";

}

funct ();

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 6:**

Глобальная переменная имеет значение - 45

## **Статические переменные**

Чтобы переменная сохраняла свое значение между вызовами функции, нужно объявить ее статической с помощью инструкции **static**.

#### Пример 7

<html>

<head>

<title>Сохранение значения переменной между вызовами функции</title>

</head>

<body>

<?php

function funct (&str)

{

static $ss = 0;

$ss++;

echo "<h2>$ss. $str</h2>";

}

funct ("Понедельник");

echo "Король гулял в парке";

funct ("Вторник");

echo "Король охотился";

?>

</body>

</html>

Была создана и проинициализирована переменная **$ss**. По отношению к функции **funct( )** эта переменная останется локальной. При втором и последующих вызовах функции **funct( )** инициализация игнорируется, а запоминается последнее присвоенное значение.

**РЕЗУЛЬТАТ ПРИМЕРА 7:**

## **1. Понедельник**

Король гулял в парке

## **2. Вторник**

Король охотился

## **Управляющие последовательности**

Если строка заключена в двойные кавычки ("), PHP распознает большее количество управляющих последовательностей для специальных символов:

| **Управляющие последовательности** | |
| --- | --- |
| Последовательность | Значение |
| \n | новая строка (LF или 0x0A (10) в ASCII) |
| \r | возврат каретки (CR или 0x0D (13) в ASCII) |
| \t | горизонтальная табуляция (HT или 0x09 (9) в ASCII) |
| \v | вертикальная табуляция (VT или 0x0B (11) в ASCII) (с версии PHP 5.2.5) |
| \e | escape-знак (ESC или 0x1B (27) в ASCII) (с версии PHP 5.4.4) |
| \f | подача страницы (FF или 0x0C (12) в ASCII) (с версии PHP 5.2.5) |
| \\ | обратная косая черта |
| \$ | знак доллара |
| \" | двойная кавычка |
| \[0-7]{1,3} | последовательность символов, соответствующая регулярному выражению символа в восьмеричной системе счисления |
| \x[0-9A-Fa-f]{1,2} | последовательность символов, соответствующая регулярному выражению символа в шестнадцатеричной системе счисления |

# Манипулирование строками

Для работы со строками в PHP существует целый ряд специализированных функций.

## **Форматирование строк**

В PHP есть две функции, позволяющие отформатировать данные перед выводом:

* [**printf()**](https://htmlweb.ru/php/function/printf.php) - выводит отформатированную строку в окно браузера;
* [**sprintf()**](https://htmlweb.ru/php/function/sprintf.php) - выводит отформатированную строку в переменную. Далее можно использовать для записи в файл с помощью функции [**fputs()**](https://htmlweb.ru/php/function/fputs.php).

Этим функциям передаются строка управления форматом и сами форматируемые данные. Строка управления форматом включает в себя обычные символы, которые прямо копируются в результат, и определители преобразования. Каждый определитель преобразования начинается с символа процента (**%**) и включает следующие элементы:

* **указатель заполнения** используется, если необходимо увеличить размер строки. По умолчанию выполняется заполнение пробелами. Можно указать и 0. Перед любым другим символом заполнения надо поставить одинарную кавычку (**'**). Необязательный параметр;
* **указатель выравнивания**, заданный знаком минус (**-**), определяет выравнивание влево. По умолчанию выполняется выравнивание вправо. Необязательный параметр;
* **указатель ширины** задает количество символов, которое должно содержаться в выводимых данных. Необязательный параметр;
* **указатель точности** задает количество десятичных знаков, отображаемых для чисел с плавающей запятой. Необязательный параметр;
* **указатель типа** задает тип, в соответствии с которым следует обрабатывать выводимые данные. Возможные значения:

|  |  |
| --- | --- |
| d | десятичное число |
| b | двоичное число |
| c | ASCII-эквивалент |
| f | число с плавающей точкой двойной точности |
| o | восьмеричное число |
| s | строка |
| x | шестнадцатеричное число в нижнем регистре |
| X | шестнадцатеричное число в верхнем регистре |

#### Пример 1

<html>

<head>

<title>Форматирование вывода в окно браузера</title>

</head>

<body>

<?php

$товары = array ("помидоры" => 15, "огурцы" => 12, "картофель" => 10);

echo "<pre>";

printf ("%-25s%25s\n%'=50s\n", "Наименование", "Цена", "");

foreach ($товары as $наим => $цена)

printf ("%'.-25s%'.22.2f\n", $наим, $цена);

echo "</pre>";

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 1:**

Наименование Цена

==================================================

помидоры.....................................15.00

огурцы.......................................12.00

картофель....................................10.00

## **Анализ строк**

PHP содержит множество функций, дающих информацию о строках. Наиболее употребимыми являются:

* [**strlen()**](https://htmlweb.ru/php/function/strlen.php) - определяет длину строки.
* [**strstr()**](https://htmlweb.ru/php/function/strstr.php) - определяет наличие подстроки в строке. Аргументы: исходная строка и искомая подстрока. Если такой подстроки в строке нет, функция возвращает **false**. Если подстрока в строке есть, функция возвращает часть исходной строки, которая начинается с искомой подстроки.
* echo strstr ("Кто там?", "там"); // выводит "там?"
* [**stristr()**](https://htmlweb.ru/php/function/stristr.php) - работает так же, как и **strstr()**, но не различает регистры.
* [**strpos()**](https://htmlweb.ru/php/function/strpos.php) - определяет позицию подстроки в строке. Аргументы: исходная строка и искомая подстрока. Если такой подстроки в строке нет, функция возвращает **false**. Если подстрока в строке есть, функция возвращает номер символа, с которого начинается подстрока.
* echo strpos ("Кто там?", "там"); // выводит "4"
* [**substr()**](https://htmlweb.ru/php/function/substr.php) - возвращает часть строки, ограниченную параметрами. Аргументы: исходная строка, начальная позиция, длина подстроки (необязательно). Если начальная позиция отрицательна, то отсчет идет от конца строки. Если длина не указана, возвращается остаток строки. Если длина отрицательна, то подстрока завершается на указанной позиции от конца строки.
* echo substr ("раз-два-три", 4, 3); // выводит "два"
* echo substr ("раз-два-три", 4); // выводит "два-три"
* echo substr ("раз-два-три", -3, 3); // выводит "три"
* echo substr ("раз-два-три", 4, -3); // выводит "два-"

## **Управление строками**

PHP содержит множество функций управления строками. Наиболее употребимыми являются:

* [**ltrim()**](https://htmlweb.ru/php/function/ltrim.php), [**chop()**](https://htmlweb.ru/php/function/chop.php), [**trim()**](https://htmlweb.ru/php/function/trim.php) - удаляют все разделители соответственно в начале строки, в конце строки, в начале и конце строки. К разделителям относятся следующие символы: **"\n"** (новая строка), **"\r"**(возврат каретки), **"\t"** (табуляция), простой пробел.
* [**str\_replace()**](https://htmlweb.ru/php/function/str_replace.php) - заменяет все экземпляры указанной подстроки в строке на новую подстроку. Аргументы: заменяемая подстрока, замещающая подстрока, исходная строка. Например:
* echo str\_replace ("там", "тут", "Кто там?"); // выводит "Кто тут?"
* [**substr\_replace()**](https://htmlweb.ru/php/function/substr_replace.php) - заменяет часть строки, ограниченную параметрами. Аргументы: исходная строка, замещающая подстрока, начальная позиция, длина заменяемой подстроки (необязательно). Если начальная позиция отрицательна, то отсчет идет от конца строки. Если длина не указана, используется вся длина строки. Если длина отрицательна, то замена завершается на указанной позиции от конца строки. Например:
* echo substr\_replace ("раз-два-три", "пять", 4, 3); // выводит "раз-пять-три"
* echo substr\_replace ("раз-два-три", "пять", 4); // выводит "раз-пять"
* echo substr\_replace ("раз-два-три", "пять", -3, 3); // выводит "раз-два-пять"
* echo substr\_replace ("раз-два-три", "пять", 4, -3); // выводит "раз-пятьтри"
* [**strtoupper()**](https://htmlweb.ru/php/function/strtoupper.php) - переводит символы строки в верхний регистр.
* [**strtolower()**](https://htmlweb.ru/php/function/strtolower.php) - переводит символы строки в нижний регистр.
* [**ucwords()**](https://htmlweb.ru/php/function/ucwords.php) - переводит в верхний регистр первый символ каждого слова.
* [**explode()**](https://htmlweb.ru/php/function/explode.php) - разбивает строку на массив. Аргументы: строка-разделитель и исходная строка. Например:
* $строка = "Голова моя машет ушами";
* $массив = explode(" ",$str);
* // count ($массив) == 4
* // $массив[0] == "Голова"
* // $массив[1] == "моя"
* // $массив[2] == "машет"
* // $массив[3] == "ушами"
* [**implode()**](https://htmlweb.ru/php/function/implode.php) -объединяет элементы массива со строкой.
* $array = array('lastname', 'email', 'phone');
* $comma\_separated = implode(",", $array);

print $comma\_separated; // lastname,email,phone

# Функции работы со строками в PHP

* [addcslashes](https://htmlweb.ru/php/function/addcslashes.php) - Экранирует спецсимволы в стиле языка C
* [addslashes](https://htmlweb.ru/php/function/addslashes.php) - Экранирует спецсимволы в строке
* [bin2hex](https://htmlweb.ru/php/function/bin2hex.php) - Преобразует бинарные данные в шестнадцатиричное представление
* [chr](https://htmlweb.ru/php/function/chr.php) - Возвращает символ по его коду
* [chunk\_split](https://htmlweb.ru/php/function/chunk_split.php) - Разбивает строку на фрагменты
* [convert\_cyr\_string](https://htmlweb.ru/php/function/convert_cyr_string.php) - Преобразует строку из одной кириллической кодировки в другую
* [count\_chars](https://htmlweb.ru/php/function/count_chars.php) - Возвращает информацию о символах, входящих в строку
* [crc32](https://htmlweb.ru/php/function/crc32.php) - Вычисляет CRC32 для строки
* [crypt](https://htmlweb.ru/php/function/crypt.php) - Необратимое шифрование (хэширование)
* [echo](https://htmlweb.ru/php/function/echo.php) - Выводит одну или более строк
* [explode](https://htmlweb.ru/php/function/explode.php) - Разбивает строку на подстроки
* [fprintf](https://htmlweb.ru/php/function/fprintf.php) - Записывает отформатированную строку в поток
* [get\_html\_translation\_table](https://htmlweb.ru/php/function/get_html_translation_table.php) - Возвращает таблицу преобразований
* [hebrev](https://htmlweb.ru/php/function/hebrev.php) - Преобразует текст на иврите из логической кодировки в визуальную
* [hebrevc](https://htmlweb.ru/php/function/hebrevc.php) - Преобразует текст на иврите из логической кодировки в визуальную с преобразованием перевод
* [htmlentities](https://htmlweb.ru/php/function/htmlentities.php) - Преобразует символы в соответствующие HTML сущности
* [htmlspecialchars](https://htmlweb.ru/php/function/htmlspecialchars.php) - Преобразует специальные символы в HTML сущности
* [html\_entity\_decode](https://htmlweb.ru/php/function/html_entity_decode.php) - Преобразует HTML сущности в соответствующие символы
* [implode](https://htmlweb.ru/php/function/implode.php) - Объединяет элементы массива в строку(массив в строку)
* [localeconv](https://htmlweb.ru/php/function/localeconv.php) - Возвращает информацию о числовых форматах
* [ltrim](https://htmlweb.ru/php/function/ltrim.php) - Удаляет пробелы из начала строки
* [md5](https://htmlweb.ru/php/function/md5.php) - Возвращает MD5 хэш строки
* [md5\_file](https://htmlweb.ru/php/function/md5_file.php) - Возвращает MD5 хэш файла
* [metaphone](https://htmlweb.ru/php/function/metaphone.php) - Возвращает ключ metaphone для строки
* [nl2br](https://htmlweb.ru/php/function/nl2br.php) - Вставляет HTML код разрыва строки перед каждым переводом строки
* [number\_format](https://htmlweb.ru/php/function/number_format.php) - Форматирует число с разделением групп
* [ord](https://htmlweb.ru/php/function/ord.php) - Возвращает ASCII код символа
* [parse\_str](https://htmlweb.ru/php/function/parse_str.php) - Разбирает строку в переменные
* [print](https://htmlweb.ru/php/function/print.php) - Выводит строку
* [printf](https://htmlweb.ru/php/function/printf.php) - Выводит отформатированную строку
* [quoted\_printable\_decode](https://htmlweb.ru/php/function/quoted_printable_decode.php) - Раскодирует строку, закодированную методом quoted printable
* [quotemeta](https://htmlweb.ru/php/function/quotemeta.php) - Экранирует специальные символы
* [rtrim](https://htmlweb.ru/php/function/rtrim.php) - Удаляет пробелы из конца строки
* [sha1](https://htmlweb.ru/php/function/sha1.php) - Возвращает SHA1 хэш строки
* [sha1\_file](https://htmlweb.ru/php/function/sha1_file.php) - Возвращает SHA1 хэш файла
* [similar\_text](https://htmlweb.ru/php/function/similar_text.php) - Вычисляет степень похожести двух строк
* [soundex](https://htmlweb.ru/php/function/soundex.php) - Возвращает ключ soundex для строки
* [sprintf](https://htmlweb.ru/php/function/sprintf.php) - Возвращает отформатированную строку
* [sscanf](https://htmlweb.ru/php/function/sscanf.php) - Разбирает строку в соответствии с заданным форматом
* [strcasecmp](https://htmlweb.ru/php/function/strcasecmp.php) - Сравнение строк без учета регистра, безопасное для данных в двоичной форме
* [strcmp](https://htmlweb.ru/php/function/strcmp.php) - Сравнение строк, безопасное для данных в двоичной форме
* [strcoll](https://htmlweb.ru/php/function/strcoll.php) - Сравнение строк с учетом текущей локали
* [strcspn](https://htmlweb.ru/php/function/strcspn.php) - Возвращает длину участка в начале строки, не соответствующего маске
* [stripcslashes](https://htmlweb.ru/php/function/stripcslashes.php) - Удаляет экранирование символов, произведенное функцией addcslashes()
* [stripos](https://htmlweb.ru/php/function/stripos.php) - Возвращает позицию первого вхождения подстроки без учета регистра
* [stripslashes](https://htmlweb.ru/php/function/stripslashes.php) - Удаляет экранирование символов, произведенное функцией addslashes()
* [strip\_tags](https://htmlweb.ru/php/function/strip_tags.php) - Удаляет HTML и PHP тэги из строки
* [stristr](https://htmlweb.ru/php/function/stristr.php) - Аналог функции strstr, но независит от регистра
* [strlen](https://htmlweb.ru/php/function/strlen.php) - Возвращает длину строки
* [strnatcasecmp](https://htmlweb.ru/php/function/strnatcasecmp.php) - Сравнение строк без учета регистра с использованием алгоритма
* [strnatcmp](https://htmlweb.ru/php/function/strnatcmp.php) - Сравнение строк с использованием алгоритма "естественного упорядочения"
* [strncasecmp](https://htmlweb.ru/php/function/strncasecmp.php) - Сравнение первых n символов строк без учета регистра, безопасное для данных в двоичной форме
* [strncmp](https://htmlweb.ru/php/function/strncmp.php) - Сравнение первых n символов строк без учета регистра, безопасное для данных в двоичной форме
* [strpos](https://htmlweb.ru/php/function/strpos.php) - Находит первое вхождение подстроки в строку
* [strrchr](https://htmlweb.ru/php/function/strrchr.php) - Находит последнее вхождение символа в строку
* [strrev](https://htmlweb.ru/php/function/strrev.php) - Переворачивает строку
* [strripos](https://htmlweb.ru/php/function/strripos.php) - Возвращает позицию последнего вхождения подстроки без учета регистра
* [strrpos](https://htmlweb.ru/php/function/strrpos.php) - Находит последнее вхождение символа в строку
* [strspn](https://htmlweb.ru/php/function/strspn.php) - Возвращает длину участка в начале строки, соответствующего маске
* [strstr](https://htmlweb.ru/php/function/strstr.php) - Находит первое вхождение подстроки
* [strtok](https://htmlweb.ru/php/function/strtok.php) - Разбивает строку
* [strtolower](https://htmlweb.ru/php/function/strtolower.php) - Преобразует строку в нижний регистр
* [strtoupper](https://htmlweb.ru/php/function/strtoupper.php) - Преобразует строку в верхний регистр
* [strtr](https://htmlweb.ru/php/function/strtr.php) - Преобразует заданные символы
* [str\_ireplace](https://htmlweb.ru/php/function/str_ireplace.php) - Регистро-независимый вариант функции str\_replace().
* [str\_pad](https://htmlweb.ru/php/function/str_pad.php) - Дополняет строку другой строкой до заданной длины
* [str\_repeat](https://htmlweb.ru/php/function/str_repeat.php) - Возвращает повторяющуюся строку
* [str\_replace](https://htmlweb.ru/php/function/str_replace.php) - Заменяет строку поиска на строку замены
* [str\_rot13](https://htmlweb.ru/php/function/str_rot13.php) - Выполняет над строкой преобразование ROT13
* [str\_shuffle](https://htmlweb.ru/php/function/str_shuffle.php) - Переставляет символы в строке
* [str\_split](https://htmlweb.ru/php/function/str_split.php) - Преобразует строку в массив
* [str\_word\_count](https://htmlweb.ru/php/function/str_word_count.php) - Возвращает информацию о словах, входящих в строку
* [substr](https://htmlweb.ru/php/function/substr.php) - Функция возвращает часть строки
* [substr\_count](https://htmlweb.ru/php/function/substr_count.php) - Подсчитывает число вхождений подстроки в строку
* [substr\_replace](https://htmlweb.ru/php/function/substr_replace.php) - Заменяет часть строки
* [trim](https://htmlweb.ru/php/function/trim.php) - Удаляет пробелы из начала и конца строки
* [ucfirst](https://htmlweb.ru/php/function/ucfirst.php) - Преобразует первый символ строки в верхний регистр
* [ucwords](https://htmlweb.ru/php/function/ucwords.php) - Преобразует в верхний регистр первый символ каждого слова в строке
* [vprintf](https://htmlweb.ru/php/function/vprintf.php) - Выводит отформатированную строку
* [vsprintf](https://htmlweb.ru/php/function/vsprintf.php) - Возвращает отформатированную строку
* [wordwrap](https://htmlweb.ru/php/function/wordwrap.php) - Выполняет перенос строки на данное количество символов с использованием символа разрыва строки

# Особенности операторов сравнения применительно к строкам.

$one = 1; // Число один.

$zero = 0; // Присваиваем число нуль.

if ($one == "") echo 1; // Очевидно, не равно - не выводит 1.

if ($zero == "") echo 2; //\* Внимание! Вопреки ожиданиям печатает 2!

if ("" == $zero) echo 3; //\* И это тоже не поможет - печатает!..

if ("$zero" == "") echo 4; // Так правильно.

if (strval($zero) == "") echo 5; // Так тоже правильно - не выводит 5.

if ($zero === "") echo 6; // Лучший способ, но не действует в PHP 3.

#### chop()

Функция chop( ) возвращает строку после удаления из нее завершающих пропусков и символов новой строки. Синтаксис функции chop( ):

string chop(string строка)

В следующем примере функция chop( ) удаляет лишние символы новой строки:

$header = "Table of Contents\n\n";

$header = chop($header);

// $header = "Table of Contents"

#### str\_pad()

Функция str\_pad( ) выравнивает строку до определенной длины заданными символами и возвращает отформатированную строку. Синтаксис функции str\_pad( ):

string str\_pad (string строка, int длина\_дополнения [, string дополнение [, int тип\_дополнения]])

Если необязательный параметр дополнение не указан, строка дополняется пробелами. В противном случае строка дополняется заданными символами. По умолчанию строка дополняется справа; тем не менее, вы можете передать в параметре тип\_дополнения константу STR\_PAD\_RIGHT, STR\_PAD\_LEFT или STR\_PAD\_BOTH, что приведет к дополнению строки в заданном направлении. Пример демонстрирует дополнение строки функцией str\_pad( ) с параметрами по умолчанию:

$food = "salad";

print str\_pad ($food, 5): // Выводит строку "salad

В следующем примере используются необязательные параметры функции str\_pad( ):

$header = "Table of Contents";

print str\_pad ($header, 5, "=+=+=", STR\_PAD\_BOTH);

// В браузере выводится строка =+=+= Таbе of Contents=+=+="

#### trim()

Функция trim( ) удаляет псе пропуски с обоих краев строки и возвращает полученную строку. Синтаксис функции trim( ):

string trim (string страна]

К числу удаляемых пропусков относятся и специальные символы \n, \r, \t, \v и \0.

#### ltrim()

Функция lrim( ) удаляет все пропуски и специальные символы с левого края строки и возвращает полученную строку. Синтаксис функции ltrim( ):

string ltrim (string строка)

Функция удаляет те же специальные символы, что и функция trim( ).

#### strlen()

Определение длины строки

Длину строки в символах можно определить при помощи функции strlen( ). Синтаксис .функции strlen( ):

int strlen (string строка)

Следующий пример демонстрирует определение длины строки функцией strlen( ):

$string = "hello";

$length = strlen($string);

// $length = 5

## **Сравнение двух строк**

Сравнение двух строк принадлежит к числу важнейших строковых операций любого языка. Хотя эту задачу можно решить несколькими разными способами, в РНР существуют четыре функции сравнения строк:

* [strcmp()](https://htmlweb.ru/php/function/strcmp.php)
* [strcasecmp()](https://htmlweb.ru/php/function/strcasecmp.php)
* [strspn()](https://htmlweb.ru/php/function/strspn.php)
* [strcspn()](https://htmlweb.ru/php/function/strcspn.php)

#### strcmp()

Функция strcmp( ) сравнивает две строки с учетом регистра символов. Синтаксис функции strcmp( ): int strcmp (string строка1, string строка2)

После завершения сравнения strcmp( ) возвращает одно из трех возможных значений:

* 0, если строка1 и строка2 совпадают;
* < 0, если строка1 меньше, чем строка2;
* > 0, если строка2 меньше, чем строка1.

В следующем фрагменте сравниваются две одинаковые строки:

$sthng1 = "butter";

$string2 = "butter";

if ((strcmp($string1. $string2)) == 0) :

print "Strings are equivalent!"; endif;

// Команда if возвращает TRUE

#### strcasecmp()

Функция strcasecmp( ) работает точно так же, как strcmp( ), за одним исключением - регистр символов при сравнении не учитывается. Синтаксис функции strcasecmp( ):

int strcasecmp (string cтpoкa1, string строка2)

В следующем фрагменте сравниваются две одинаковые строки:

$string1 = "butter";

$string2 = "Butter";

if ((strcmp($string1, $string2)) == 0) :

print "Strings are equivalent!";

endif;

// Команда if возвращает TRUE

#### strspn()

Функция strspn( ) возвращает длину первого сегмента строки1, содержащего символы, присутствующие в строке2. Синтаксис функции strspn( ):

int strspn (string строка1, string строка2)

Следующий фрагмент показывает, как функция strspn( ) используется для проверки пароля:

$password = "12345";

if (strspn($password, "1234567890") != strlen($password)) :

print "Password cannot consist solely of numbers!";

endif:

#### strcspn()

Функция strcspn( ) возвращает длину первого сегмента строки1, содержащего символы, отсутствующие в строке2. Синтаксис функции strcspn( ):

int strcspn (string строка1, string строка2)

В следующем фрагменте функция strcspn( ) используется для проверки пароля:

$password = "12345";

if (strcspn($password, "1234567890") == 0) :

print "Password cannot consist solely of numbers!";

endif;

## **Обработка строковых данных без применения регулярных выражений**

При обработке больших объемов информации функции регулярных выражений сильно замедляют выполнение программы. Эти функции следует применять лишь при обработке относительно сложных строк, в которых регулярные выражения действительно необходимы. Если же анализ текста выполняется по относительно простым правилам, можно воспользоваться стандартными функциями РНР, которые заметно ускоряют обработку. Все эти функции описаны ниже.

#### strtok()

Функция strtok( ) разбивает строку на лексемы по разделителям, заданным вторым параметром. Синтаксис функции strtok( ):

string strtok (string строка, string разделители)

У функции strtok( ) есть одна странность: чтобы полностью разделить строку, функцию необходимо последовательно вызвать несколько раз. При очередном вызове функция выделяет из строки следующую лексему. При этом параметр строка задается всего один раз - функция отслеживает текущую позицию в строке до тех пор, пока строка не будет полностью разобрана на лексемы или не будет задан новый параметр строка. Следующий пример демонстрирует разбиение строки по нескольким разделителям:

$info = "WJ Gi1more:wjgilmore@hotmail.com | Columbus, Ohio";

// Ограничители - двоеточие (:), вертикальная черта (|) и запятая (.)

$tokens = ":|,";

$tokenized = strtok($info, $tokens);

// Вывести элементы массива $tokenized

while ($tokenized) :

echo "Element = $tokenized<br>";

// Обратите внимание: при последующих вызовах strtok

// первый аргумент не передается

$tokenized = strtok($tokens);

endwhile;

Результат: Element = WJGilmore Element = wjgilmore@hotmail.com Element = Columbus Element = Ohio

#### parse\_str()

Функция parse\_str( ) выделяет в строке пары <переменная-значение> и присваивает значения переменных в текущей области видимости. Синтаксис функции parse\_str( ):

void parse\_str (string строка)

Функция parse\_str( ) особенно удобна при обработке URL, содержащих данные форм HTML или другую расширенную информацию. В следующем примере анализируется информация, переданная через URL. Строка представляет собой стандартный способ передачи данных между страницами либо откомпилированных в гиперссылке, либо введенных в форму HTML:

$url = "fname=wj&lname=gilmore&zip=43210";

parse\_str($url);

// После выполнения parse\_str( ) доступны следующие переменные:

// $fname = "wj":

// $lname = "gilmore";

// $zip = "43210"

Поскольку эта функция создавалась для работы с URL, она игнорирует символ амперсанд (&).

#### explode()

Функция explode() делит строку на элементы и возвращает эти элементы в виде массива. Синтаксис функции explode():

array explode (string разделитель, string строка [, int порог])

Разбиение происходит по каждому экземпляру разделителя, причем количество полученных фрагментов может ограничиваться необязательным параметром порог.

Разделение строки функцией explode( ) продемонстрировано в следующем примере:

$info = "wilson | baseball | indians";

$user = explode("|", $info);

// $user[0] = "wilson";

// $user[1] = "baseball";

// $user[2] = "Indians";

Функция explode( ) практически идентична функции регулярных выражений POSIX split( ), описанной выше. Главное различие заключается в том, что передача регулярных выражений в параметрах допускается только при вызове split( ).

#### implode()

Если функция explode( ) разделяет строку на элементы массива, то ее двойник - функция implode( ) - объединяет массив в строку. Синтаксис функции implode( ):

string implode (string разделитель, array фрагменты)

Формирование строки из массива продемонстрировано в следующем примере:

$ohio\_cities = array ("Columbus", "Youngstown", "Cleveland", "Cincinnati");

$city\_string = implode("l", $ohio\_cities);

// $city\_string = "Columbus | Youngstown | Cleveland | Cincinnati";

У implode( ) имеется псевдоним - функция join( ).

#### strpos()

Функция strpos( ) находит в строке первый экземпляр заданной подстроки. Синтаксис функции strpos():

int strpos (string строка, string подстрока [, int смещение])

Необязательный параметр offset задает позицию, с которой должен начинаться поиск. Если подстрока не найдена, strpos() возвращает FALSE (0).

В следующем примере определяется позиция первого вхождения даты в файл журнала:

$log = "

206.169.23.11:/www/:2000-08-10

206.169.23.11:/www/logs/:2000-02-04

206.169.23.11:/www/img/:1999-01-31";

// В какой позиции в журнале впервые встречается 1999 год?

$pos = strpos($log, "1999");

// $pos = 95. поскольку первый экземпляр "1999"

// находится в позиции 95 строки, содержащейся в переменной $log

#### strrpos()

Функция strrpos( ) находит в строке последний экземпляр заданного символа. Синтаксис функции strrpos( ):

int strpos (string строка, char символ)

По возможностям эта функция уступает своему двойнику - функции strpos( ), поскольку она позволяет искать только отдельный символ, а не всю строку. Если во втором параметре strrpos( ) передается строка, при поиске будет использован только ее первый символ.

#### str\_replace()

Функция str\_replace( ) ищет в строке все вхождения заданной подстроки и заменяет их новой подстрокой. Синтаксис функции str\_replace( ):

string str\_replace (string подстрока, string замена, string строка)

Функция substr\_replace( ), описанная ниже в этом разделе, позволяет провести заме ну лишь в определенной части строки. Ниже показано, как функция str\_replace( ) используется для проведения глобальной замены в строке.

Если подстрока ни разу не встречается в строке, исходная строка не изменяется:

$favorite\_food = "My favorite foods are ice cream and chicken wings";

$favorite\_food = str\_replace("chicken\_wings", "pizza", $favohte\_food);

// $favorite\_food = "My favorite foods are ice cream and pizza"

#### strstr()

Функция strstr( ) возвращает часть строки, начинающуюся с первого вхождения заданной подстроки. Синтаксис функции strstr( ):

string strstr (string строка, string подстрока)

В следующем примере функция strstr( ) используется для выделения имени домена из URL:

$url = "http://www.apress.com"; $domain - strstr($url, ".");

// $domain = ".apress.com"

#### substr()

Функция substr( ) возвращает часть строки, начинающуюся с заданной начальной позиции и имеющую заданную длину. Синтаксис функции substr( ):

string substr (string строка, int начало [, int длина])

Если необязательный параметр длина не указан, считается, что подстрока начинается с заданной начальной позиции и продолжается до конца строки. При использовании этой функции необходимо учитывать четыре обстоятельства:

* если параметр начало положителен, возвращаемая подстрока начинается с позиции строки с заданным номером;
* если параметр начало отрицателен, возвращаемая подстрока начинается с позиции (длина строки - начало);
* если параметр длина положителен, в возвращаемую подстроку включаются все символы от позиции начало до позиции начало+длина. Если последняя величина превышает длину строки, возвращаются символы до конца строки;
* если параметр длина отрицателен, возвращаемая подстрока заканчивается на заданном расстоянии от конца строки.

Помните о том, что параметр начало определяет смещение от первого символа строки; таким образом, возвращаемая строка в действительности начинается с символа с номером (начало + 1).

Следующий пример демонстрирует выделение части строки функцией substr( ):

$car = "1944 Ford"; Smodel = substr($car, 6);

// Smodel = "Ford"

Пример с положительным параметром длина:

$car = "1944 Ford";

$model = substr($car, 0, 4);

// $model = "1944"

Пример с отрицательным параметром длина:

$car = "1944 Ford";

$model = substr($car, 2, -5);

// $model = "44"

#### substr\_count()

Функция substr\_count( ) возвращает количество вхождений подстроки в заданную строку. Синтаксис функции substr\_count( ): int substr\_count (string строка, string подстрока) В следующем примере функция substr\_count( ) подсчитывает количество вхождений подстроки ain:

$tng\_twist = "The rain falls mainly on the plains of Spain";

$count = substr\_count($tng\_twist, "ain");

// $count = 4

#### substr\_replace()

Функция substr\_replace( ) заменяет часть строки, которая начинается с заданной позиции. Если задан необязательный параметр длина, заменяется фрагмент заданной длины; в противном случае производится замена по всей длине заменяющей строки. Синтаксис функции substr\_replace( ):

string substr\_replace (string строка, string замена, int начало [, int длина])

Параметры начало и длина задаются по определенным правилам:

* если параметр начало положителен, замена начинается с заданной позиции;
* если параметр начало отрицателен, замена начинается с позиции (длина строки -начало);
* если параметр длина положителен, заменяется фрагмент заданной длины;
* если параметр длина отрицателен, замена завершается в позиции (длина строки -длина).

Простая замена текста функцией substr\_replace( ) продемонстрирована в следующем примере:

$favs = " 's favorite links";

$name = "Alessia";

// Параметры "0, 0" означают, что заменяемый фрагмент начинается

// и завершается в первой позиции строки.

$favs - substr\_replace($favs, $name, 0, 0);

print $favs:

Результат:

Alessia's favorite links

## **Преобразование строк и файлов к формату HTML и наоборот**

Преобразовать строку или целый файл к формату, подходящему для просмотра в web-браузере (или наоборот), проще, чем может показаться на первый взгляд. В РНР для этого существуют специальные функции.

### **Преобразование текста в HTML**

Быстрое преобразование простого текста к формату web-браузера - весьма распространенная задача. В ее решении вам помогут функции, описанные в этом разделе.

#### nl2br()

Функция nl2br() заменяет все символы новой строки (\n) эквивалентными конструкциями HTML.

Синтаксис функции nl2br():

string nl2br (string строка)

Символы новой строки могут быть как видимыми (то есть явно включенными в строку), так и невидимыми (например, введенными в редакторе). В следующем примере текстовая строка преобразуется в формат HTML посредством замены символов \n разрывами строк:

// Текстовая строка, отображаемая в редакторе.

$text\_recipe = "

Party Sauce recipe:

1 can stewed tomatoes

3 tablespoons fresh lemon juice

Stir together, server cold.";

// Преобразовать символы новой строки в

$htinl\_recipe = nl2br($text\_recipe)

При последующем выводе $html\_recipe браузеру будет передан следующий текст в формате HTML:

Party Sauce recipe:<br>

1 can stewed tomatoes<br>

3 tablespoons fresh lemon juice<br>

Stir together, server cold.<br>

#### htmlentities()

Функция htmlentities( ) преобразует символы в эквивалентные конструкции HTML. Синтаксис функции htmlentities:

string htmlentities (string строка)

В следующем примере производится необходимая замена символов строки для вывода в браузере:

$user\_input = "The cookbook, entitled Cafe Francaise' costs < $42.25.";

$converted\_input = htmlentities($user\_input);

// $converted\_input = "The cookbook, entitled 'Cafè

// Fracçiaise' costs < 42.25.";

Функция htmlentities( ) в настоящее время работает только для символов кодировки ISO-8559-1 (ISO-Latin-1). Кроме того, она не преобразует пробелы в  , как следовало бы ожидать.

#### htmlspecialchars()

Функция htmlspecialchars( ) заменяет некоторые символы, имеющие особый смысл в контексте HTML, эквивалентными конструкциями HTML. Синтаксис функции htmlspecialchars( ):

string htmlspecialchars (string строка)

Функция html special chars( ) в настоящее время преобразует следующие символы: & преобразуется в &; " " преобразуется в "; < преобразуется в <; > преобразуется в >.

В частности, эта функция позволяет предотвратить ввод пользователями разметки HTML в интерактивных web-приложениях (например, в электронных форумах). Ошибки, допущенные в разметке HTML, могут привести к тому, что вся страница будет формироваться неправильно. Впрочем, у этой задачи существует и более эффективное решение - полностью удалить теги из строки функцией strip\_tags( ).

Следующий пример демонстрирует удаление потенциально опасных символов функцией htmlspeclalchars( ):

$user\_input = "I just can't get <enough> of PHP & those fabulous cooking recipes!";

$conv\_input = htmlspecialchars($user\_input);

// $conv\_input = "I just can't <<enough>> of PHP & those fabulous cooking

recipes!"

Если функция htmlspecialchars( ) используется в сочетании с nl2br( ), то последнюю следует вызывать после htmlspecialchars( ). В противном случае конструкции <br>, сгенерированные при вызове nl2br( ), преобразуются в видимые символы.

#### get\_html\_translation\_table()

Функция get\_html\_translation\_table( ) обеспечивает удобные средства преобразования текста в эквиваленты HTML Синтаксис функции get\_htrril\_translation\_table( ):

string get\_html\_translation\_table (int таблица)

Функция get\_html\_translation\_table( ) возвращает одну из двух таблиц преобразования (определяется параметром таблица), используемых в работе стандартных функций htmlspecialchars( ) и htmlentities( ). Возвращаемое значение может использоваться в сочетании с другой стандартной функцией, strtr(), для преобразования текста в код HTML.

Параметр таблица принимает одно из двух значений:

* HTML\_ENTITIES;
* HTML\_SPECIALCHARS.

В следующем примере функция get\_html\_translation\_table( ) используется при преобразовании текста в код HTML:

$string = "La pasta e il piatto piu amato in Italia";

$translate = get\_html\_translation\_table(HTML\_ENTITIES);

print strtr($string, $translate);

// Специальные символы преобразуются в конструкции HTML

// и правильно отображаются в браузере.

Кстати, функция array\_flip( ) позволяет провести преобразование текста в HTML в обратном направлении и восстановить исходный текст. Предположим, что вместо вывода результата strtr( ) в предыдущем примере мы присвоили его переменной $translated string.

В следующем примере исходный текст восстанавливается функцией array\_flip( ):

$translate = array\_flip($translate);

$translated\_string - "La pasta é il piatto piú amato in Italia";

$original\_string = strtr($translated\_string, $translate);

// $original\_string = "La pasta e il piatto piu amato in Italia";

#### strtr()

Функция strtr( ) транслирует строку, то есть заменяет в ней все символы, входящие в строку источник, соответствующими символами строки приемник. Синтаксис функции strtr( ):

string strtr (string строка, string источник, string приемник)

Если строки источник и приемник имеют разную длину, длинная строка усекается до размеров короткой строки.

Существует альтернативный синтаксис вызова strtr( ) с двумя параметрами; в этом случае второй параметр содержит ассоциативный массив, ключи которого соответствуют заменяемым подстрокам, а значения - заменяющим подстрокам. В следующем примере теги HTML заменяются XML-подобными конструкциями:

<?

$source = array("<title>" => "<h1>", "</title>" => "</h1>");

$string = "<h1>Today In PHP-Powered News</h1>";

print strtr($string, $source);

// Выводится строка "<title>Today in PHP-Powered News</title>"

?>

### **Преобразование HTML в простой текст**

Иногда возникает необходимость преобразовать файл в формате HTML в простой текст. Функции, описанные ниже, помогут вам в решении этой задачи.

#### strip\_tags()

Функция strip\_tags( ) удаляет из строки все теги HTML и РНР, оставляя в ней только текст. Синтаксис функции strip\_tags( ):

string strip\_tags (string строка [, string разрешенные\_тerи])

Необязательный параметр разрешенные\_теги позволяет указать теги, которые должны пропускаться в процессе удаления.

Ниже приведен пример удаления из строки всех тегов HTML функцией strip\_tags( ):

$user\_input = "I just **love** РНР and *gourment* recipes!";

$stripped\_input = strip\_tags($user\_input);

// $stripped\_input = "I just love PHP and gourmet recipes!";

В следующем примере удаляются не все, а лишь некоторые теги:

$input = "I &lt;b>love&lt;/b> to &lt;a href = \"http://www.eating.com\">eat!&lt;/a>!";

$strip\_input = strip\_tags ($user\_input, "&lt;a>");

// $strip\_input = "I love to &lt;a href = \"http://www.eating.com\">eat!&lt;/a>!";

Удаление тегов из текста также производится функцией fgetss().

#### get\_meta\_tags()

Хотя функция get\_meta\_tags( ) и не имеет прямого отношения к преобразованию текста, зто весьма полезная функция, о которой следует упомянуть. Синтаксис функции get\_meta\_tags( ):

array get\_meta\_tags (string имя\_файла/URL [, int включение\_пути])

Функция get\_meta\_tags( ) предназначена для поиска в файле HTML тегов МЕТА.

Теги МЕТА содержат информацию о странице, используемую главным образом поисковыми системами. Эти теги находятся внутри пары тегов .... Применение тегов МЕТА продемонстрировано в следующем фрагменте (назовем его example.html, поскольку он будет использоваться в листинге 8.2): [/html] [/html] Функция get\_meta\_tags( ) ищет в заголовке документа теги, начинающиеся словом МЕТА, и сохраняет имена тегов и их содержимое в ассоциативном массиве. В листинге 8.2 продемонстрировано применение этой функции к файлу example.html. Листинг 8.2. Извлечение тегов МЕТА из файла HTML функцией get\_meta\_tags( )

$meta\_tags = get\_meta\_tags("example.html"):

// Переменная $meta\_tags содержит массив со следующей информацией:

// $meta\_tags["keywords"] = "PHP, code, recipes, web"

// $meta\_tags["description"] = "Информация о PHP"

// $meta\_tags["author"] = "KDG";

Интересная подробность: данные тегов МЕТА можно извлекать не только из файлов, находящихся на сервере, но и из других URL.

### **Преобразование строки к верхнему и нижнему регистру**

В РНР существует четыре функции, предназначенных для изменения регистра строки:

* [strtolower()](https://htmlweb.ru/php/function/strtolower.php);
* [strtoupper()](https://htmlweb.ru/php/function/strtoupper.php);
* [ucfirst()](https://htmlweb.ru/php/function/ucfirst.php);
* [ucwords()](https://htmlweb.ru/php/function/ucwords.php).

#### strtolower( )

Функция strtolower( ) преобразует все алфавитные символы строки к нижнему регистру. Синтаксис функции strtolower():

string strtolower(string строка)

Неалфавитные символы функцией не изменяются. Преобразование строки к нижнему регистру функцией strtolower() продемонстрировано в следующем примере:

$sentence = "COOKING and PROGRAMMING PHP are my TWO favorite!";

$sentence = strtolower($sentence);

// После вызова функции $sentence содержит строку

// "cooking and programming php are my two favorite!"

#### strtoupper()

Строки можно преобразовывать не только к нижнему, но и к верхнему регистру. Преобразование выполняется функцией strtoupper(), имеющей следующий синтаксис:

string strtoupper (string строка)

Неалфавитные символы функцией не изменяются. Преобразование строки к верхнему регистру функцией strtoupper() продемонстрировано в следующем примере:

$sentence = "cooking and programming PHP are my two favorite!";

$sentence = strtoupper($sentence);

// После вызова функции $sentence содержит строку

// "COOKING AND PROGRAMMING PHP ARE MY TWO FAVORITE!"

#### ucfirst()

Функция ucfirst( ) преобразует к верхнему регистру первый символ строки - при условии, что он является алфавитным символом. Синтаксис функции ucfirst():

string ucfirst (string строка)

Неалфавитные символы функцией не изменяются. Преобразование первого символа строки функцией ucfirst() продемонстрировано в следующем примере:

&sentence = "cooking and programming PHP are my two favorite!";

$sentence = ucfirst($sentence);

// После вызова функции $sentence содержит строку

// "Cooking and programming PHP are mу two favorite!"

#### ucwords()

Функция ucwords( ) преобразует к верхнему регистру первую букву каждого слова в строке. Синтаксис функции ucwords():

string ucwords (string строка")

Неалфавитные символы функцией не изменяются. "Слово" определяется как последовательность символов, отделенная от других элементов строки пробелами. В следующем примере продемонстрировано преобразование первых символов слов функцией ucwords( ):

$sentence = "cooking and programming PHP are my two favorite!";

$sentence = ucwords($sentence);

// После вызова функции $sentence содержит строку

// "Cooking And Programming PHP Are My Two Favorite!"

#### strrchr()

strrchr("строка", "о") - Находит последнее вхождение подстроки

Если подстрока не найдена, возвращает FALSE.

В отличие от strchr(), если искомая строка состоит более чем из одного символа, используется только первый символ.

Если второй параметр не является строкой, он приводится к целому и трактуется как код символа.

// получить последнюю директорию из $PATH

$dir = substr(strrchr($PATH, ":"), 1);

// получить все после последнего перевода строки

$text = "Line 1\nLine 2\nLine 3";

$last = substr(strrchr($text, 10), 1 );

#### highlight\_string()

highlight\_string- выделение синтаксиса строки.

mixed highlight\_string (string str [, bool return])

Функция highlight\_string() выводит версию с расцвеченным синтаксисом строки str, используя цвета, определённые во встроенном выделении синтаксиса PHP.

Если второй параметр return имеет значение TRUE, то highlight\_string() возвратит версию раскрашенного кода как строку, вместо её печати. Если второй параметр не имеет значение TRUE, highlight\_string() возвратит TRUE при успехе, FALSE при неудаче.

Примечание: 1.параметр return стал доступен, начиная с PHP 4.2.0. До этого он работал, как по умолчанию, т.е. FALSE.  
2.Функция Highlight\_String() различает PHP код по тегам <?php ?>. **show\_source()** - синоним**highlight\_file()**. Для изменения цвета выделения по умолчанию используйте следующие PHP директивы:

highlight.bg #FFFFFF

highlight.comment #FF8000

highlight.default #0000BB

highlight.html #000000

highlight.keyword #007700

highlight.string #DD0000

в .htaccess:

php\_flag highlight.bg #FFFFFF

в PHP:

if(@ini\_get('highlight.bg')=='') ...

#### addslashes()

- Экранирует спецсимволы в строке

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

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

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

// выводит: Is your name O\'reilly?

echo addslashes($str);

#### stripslashes

- Удаляет экранирование символов, произведенное функцией addslashes() Удаляет экранирующие бэкслэши. (\' преобразуется в ', и т.д.). Двойные бэкслэши (\\) преобразуется в одиночные(\).

#### wordwrap()

Выполняет перенос строки на данное количество символов с использованием символа разрыва строки

// Использование wordwrap().

function cite($ourText, $maxlen=60, $prefix="> ") {

$st = wordwrap($ourText, $maxlen-strlen($prefix), "\n");

$st = $prefix.str\_replace("\n", "\n$prefix", $st);

return $st;

}

echo cite("The first Matrix I designed was quite naturally

perfect, it was a work of art - flawless, sublime. A triumph

equalled only by its monumental failure. The inevitability

of its doom is apparent to me now as a consequence of the

imperfection inherent in every human being. Thus, I

redesigned it based on your history to more accurately reflect

the varying grotesqueries of your nature. However, I was again

frustrated by failure.", 20);

> The first Matrix I

> designed was quite

> naturally

> perfect, it was a

> work of art -

> flawless, sublime.

> A triumph

> equalled only by

> its monumental

> failure. The

> inevitability

> of its doom is

> apparent to me now

> as a consequence

> of the

> imperfection

> inherent in every

> human being. Thus,

> I

> redesigned it

> based on your

> history to more

> accurately reflect

> the varying

> grotesqueries of

> your nature.

> However, I was

> again

> frustrated by

> failure.

# Объектное программирование

### **Объектно-ориентированное программирование (ООП) на PHP**

Объект - это набор специальных переменных - *свойств* и специальных функций - *методов*. То, что в процедурном программировании называлось переменной — в ООП называется свойство. То, что в процедурном программировании называлось функцией — в ООП называется методом класса. Созданные на основе класса объекты называются экземплярами класса или просто объекты.

Обращение из метода к свойствам только через служебное слово $this: $this->name; (обратите внимание на отсутствие знака доллара перед name) Обращение внутри метода к другому методу тоже через $this: $this->foo(); Для доступа к свойствам и методам объекта служит оператор "->": $this->name; (обратите внимание на отсутствие знака доллара перед name)  
Обращение внутри метода к другому методу тоже через $this: $this->foo();. Объект создается с помощью оператора **new** на основании шаблона, называемого *классом*. Класс определяется ключевым словом **class**.

#### Пример 1

<html>

<head>

<title>Класс со свойством и методом</title>

</head>

<body>

<?php

class классN1

{

public $имя = "Маша"; // - это свойство класса доступное снаружи класса

private $Private\_name; // - это свойство доступно только методам класса

protected $Protected\_name; // это свойство доступно методам собственного класса, а также методам наследуемых классов

function Привет() // - это метод класса

{

echo "<H1>".$this->имя."! Привет!</H1>";

}

function Пока( $a )

{

$this->имя = $a;

echo "<H1>".$this->имя."! Пока!</H1>";

}

}

$obj = new классN1();

$obj->Привет();

$obj->имя = "Миша";

$obj->Привет();

$obj->Пока("Яша");

$obj->Привет();

?>

</body>

</html>

**Модификаторы доступа в ООП**:

* **public** — позволяет иметь доступ к свойствам и методам из любого места (глобальная область)
* **protected** — доступ к родительскому и наследуемому классу (область класса наследника)
* **private** — доступ только из класса, в котором объявлен сам элемент (область самого класса)

Метод по умолчанию — public. У свойств значения модификатора по умолчанию нет.

### **Константы класса в ООП**

const NAME = 2;

Таким образом можно создавать константы и вне класса. Это именно константы класса, они не принадлежат ни одному объекту, они общие на все объекты, поэтому использование внутри метода:

function printname(){

echo self::NAME;

}

self — это сам класс! Обращение вне класса (можно вызывать из глобальной области видимости без инициализации экземпляра класса):

echo OurClass::NAME;

### **this и self**

Внутри класса использована специальная переменная **this**. Это указатель, с помощью которого объект может ссылаться на самого себя.

Для обращения к статическим методам используется **self::**

Методу **Пока** передан аргумент точно так же, как и обычной функции. При вызове этого метода объект меняет свое свойство **имя**.

**РЕЗУЛЬТАТ ПРИМЕРА 1:**

# Маша! Привет!

# Миша! Привет!

# Яша! Пока!

# Яша! Привет!

Конструктор — это метод, который автоматически вызывается при создании нового объекта: public function \_\_construct(){}. При инициализации6 объекта через служебную конструкцию new, PHP ищет \_\_construct и если он есть, то вызывается.

Также можно создать метод, имя которого совпадает с именем класса, - такой метод также будет считаться конструктором. Конструктор может принимать аргументы, что значительно упрощает работу с классами.

#### Пример 2

<html>

<head>

<title>Класс с конструктором</title>

</head>

<body>

<?

class классN2

{

private $имя; // - это свойство класса НЕ доступное снаружи класса

function \_\_construct( $a="Кто-то там" )

{

$this->имя = $a;

}

function Привет()

{

echo "<H1>".$this->имя."! Привет!</H1>";

}

}

$obj0 = new классN2();

$obj1 = new классN2("Миша");

$obj2 = new классN2("Маша");

$obj0->Привет();

$obj1->Привет();

$obj2->Привет();

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 2:**

# Кто-то там! Привет!

# Миша! Привет!

# Маша! Привет!

Сложив все, изложенное выше, можно создать более осмысленный класс. Например, класс, который будет располагать данные в виде таблицы с поименнованными столбцами.

#### Пример 3

<html>

<head>

<title>Класс Table</title>

</head>

<body>

<?php

class Table

{

private $headers = [];

private $data = [];

function Table ( $headers )

{

$this->headers = $headers;

}

function addRow ( $row )

{

$tmp = [];

foreach ( $this->headers as $header )

{

if ( ! isset( $row[$header] )) $row[$header] = "";

$tmp[] = $row[$header];

}

array\_push ( $this->data, $tmp );

}

function output ()

{

echo "<PRE><B>";

foreach ( $this->headers as $header ) echo "$header ";

echo "</B><BR>";

foreach ( $this->data as $y )

{

foreach ( $y as $x ) echo "$x ";

echo "<BR>";

}

echo "</PRE>";

}

}

$test = new Table (array("a","b","c"));

$test->addRow(array("a"=>1,"b"=>3,"c"=>2));

$test->addRow(array("b"=>1,"a"=>3));

$test->addRow(array("c"=>1,"b"=>3,"a"=>4));

$test->output();

?>

</body>

</html>

Свойства класса **Table** - массив имен столбцов таблицы и двумерный массив строк данных. Конструктор класса **Table** получает массив имен столбцов таблицы. Метод **addRow** добавляет в таблицу новую строку данных. Метод **output** выводит таблицу на экран.

**РЕЗУЛЬТАТ ПРИМЕРА 3:**

**a b c**

1 3 2

3 1

4 3 1

### **Скрытые свойства и методы**

Свойства и методы класса могут быть как открытыми (public), так и скрытыми (private). Скрытые свойства и методы недоступны извне класса, т.е. из сценария, в котором используется данный класс, или из другого класса.

## **Наследование**

На основе существующих классов можно создавать новые, используя *механизм наследования*. Механизм наследования - это использование определенного ранее класса в качестве родительского. При этом набор свойств и методов родительского класса можно расширять. Имейте в виду, что производный класс имеет только одного родителя.

Чтобы создать новый класс, наследующий поведение существующего класса, надо использовать ключевое слово **extends** в его объявлении. Например:

class классN2 extends классN1

{

.......

}

Здесь **классN1** - родительский класс, **классN2** - производный.

Если производный класс не содержит собственного конструктора, то при создании его объекта используется конструктор родительского класса. Если в производном класса существует собственный конструктор, то конструктор родительского класса не вызывается. При необходимости вызвать конструктор родительского класса это надо сделать явно. Например:

классN1::классN1();

Производный класс будет иметь все свойства и методы родительского класса. Но их можно и переопределить в производном классе.

#### Пример 4

<html>

<head>

<title>Переопределение метода родительского класса</title>

</head>

<body>

<?php

class классN3

{

public $имя = "Маша";

function Привет()

{

echo "<H1>".$this->имя."! Привет!</H1>";

}

}

class классN4 extends классN3

{

function Привет()

{

echo "<H1>".$this->имя."! Какая встреча!</H1>";

}

}

$obj = new классN4();

$obj->Привет();

?>

</body>

</html>

Метод **Привет** переопределен для производного класса. Свойство **имя** наследуется от родительского.

**РЕЗУЛЬТАТ ПРИМЕРА 4:**

# Маша! Какая встреча!

Начиная с 4-й версии PHP, в объекте производного класса можно вызвать метод родительского класса, который был переопределен.

#### Пример 5

<html>

<head>

<title>Вызов метода родительского класса</title>

</head>

<body>

<?php

class классN5

{

public $имя = "Маша";

function Привет()

{

echo "<H1>".$this->имя."! Привет!</H1>";

}

function Пока()

{

echo "<H1>".$this->имя.", пока!</H1>";

}

}

/\*\*

\* Class классN6

\*/

class классN6 extends классN5

{

/\*\*

\*

\*/

function Привет()

{

echo "<H1>".$this->имя."! Какая встреча!</H1>";

классN5::Привет();

}

}

$obj = new классN6();

$obj->Привет();

$obj->Пока();

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 5:**

# Маша! Какая встреча!

# Маша! Привет!

# Маша, пока!

Итак, производный класс может наследовать, переопределять и дополнять свойства и методы другого класса.

В следующем примере создан класс **HTMLTable**, основанный на классе **Table** из примера 3. Новый класс формирует данные, сохраненные методом **addRow** родительского класса, и выводит их в HTML-таблицу. Свойства **$cellpadding** и **$bgcolor** дают возможность изменять соответствующие аргументы, при этом переменной **$cellpadding** присваивается значение по умолчанию, равное 2.

#### Пример 6

<html>

<head>

<title>Классы Table и HTMLTable</title>

</head>

<body>

<?php

class Tables

{

public $headers = [];

public $data = [];

function Tables( $headers )

{

$this->headers = $headers;

}

function addRow ( $row )

{

$tmp = [];

foreach ( $this->headers as $header )

{

if ( ! isset( $row[$header] )) $row[$header] = "";

$tmp[] = $row[$header];

}

array\_push ( $this->data, $tmp );

}

function output ()

{

echo "<PRE><B>";

foreach ( $this->headers as $header ) echo "$header ";

echo "</B><BR>";

foreach ( $this->data as $y )

{

foreach ( $y as $x ) echo "$x ";

echo "<BR>";

}

echo "</PRE>";

}

}

class HTMLTable extends Tables

{

public $cellpadding = "2";

public $bgcolor;

function HTMLTable ( $headers, $bg="FFFFFF" )

{

Tables::Tables( $headers );

$this->bgcolor = $bg;

}

function setCellpadding ( $padding )

{

$this->cellpadding = $padding;

}

function output ()

{

echo "<table cellpadding='".$this->cellpadding."'><tr>";

foreach ( $this->headers as $header )

echo "<th bgcolor='".$this->bgcolor."'>".$header;

foreach ( $this->data as $y )

{

echo "<tr>";

foreach ( $y as $x )

echo "<td bgcolor='".$this->bgcolor."'>$x";

}

echo "</table>";

}

}

$test = new HTMLTable ( array("a","b","c"), "#00FFFF" );

$test->setCellpadding ( 7 );

$test->addRow(array("a"=>1,"b"=>3,"c"=>2));

$test->addRow(array("b"=>1,"a"=>3));

$test->addRow(array("c"=>1,"b"=>3,"a"=>4));

$test->output();

?>

</body>

</html>

Обратите внимание на то, что значение свойства **сellpadding** меняется с помощью отдельного метода **setCellpadding**. Конечно, значения свойств можно менять непосредственно, вне объекта:

$test->сellpadding = 7 ;

Но это считается дурным тоном, т.к. в сложных объектах при изменении одного из свойств могут изменяться и другие свойства.

**РЕЗУЛЬТАТ ПРИМЕРА 6:**

|  |  |  |
| --- | --- | --- |
| a | b | c |
| 1 | 3 | 2 |
| 3 | 1 |  |
| 4 | 3 | 1 |

Использовать или нет технику объектного программирования? С одной стороны, проект, интенсивно использующий объектную технику, может занимать слишком много ресурсов во время выполнения. С другой стороны, правильно организованный объектный подход значительно сократит время разработки и сделает программу более гибкой.

### **Удаление объектов**

Удалить ранее созданный объект можно следующим образом:

unset($objName);

Ниже приведен пример, в котором объект класса Саг создается, а затем удаляется.

$myCar = new Car;

unset($myCar);

После вызова функции unset() объект больше не существует. В РНР имеется специальный метод \_\_destruct(), который автоматически вызывается при удалении объекта. Ниже приведен класс, содержащий этот метод.

class Bridge

{

function \_\_destruct()

{

echo "Мост разрушен";

}

}

$bigBridge = new Bridge;

unset($bigBridge);

При создании объекта класса Bridge, а затем его удалении отобразится следующее сообщение:

Мост разрушен

Оно отображается вследствие вызова метода \_\_destruct() при вызове функции unset(). При удалении объекта может потребоваться акрыть некоторые файлы или записать информацию в базу данных.

### **Копирование (клонирование) объекта**

Клонирование объекта:

$a = clone $b;

Конструктор не вызывается при клонировании, вызывается магический метод **\_\_clone()**{}. Он НЕ принимает аргументов и к нему нельзя обратиться как к методу.

### **Преобразование объекта в строку**

Для конвертации объекта в строку, и обратно, используются следующие функции:  
**serialize()** - принимает объект и возвращает строковое представление его класса и свойств;  
**unserialize()** - принимает строку, созданную при помощи serialize(), и возвращает объект.

serialize() и unserialize() работают со всеми типами данных, но они не работают с ресурсами.

Специальные методы для обслуживания функций serialize() и unserialize():  
**\_\_sleep()** - вызывается строго перед тем, как объект сериализуется с помощью функции serialize(). Функция \_\_sleep() должна будет вернуть список полей класса, которые функция serialize() включит в возвращаемую строку. Вы можете использовать это для того, чтобы исключить ненужные поля из строкового представления объекта. Например:

public function \_\_sleep() { // почистить

return array\_keys( get\_object\_vars( $this ) );

}

**\_\_wakeup()** - вызывается сразу после того, как объект десериализуется с помощью unserialize().

### **Абстрактный класс**

**Абстрактный класс** - это класс, который не может быть реализован, то есть, вы не сможете создать объект класса, если он абстрактный. Вместо этого вы создаете дочерние классы от него и спокойно создаете объекты от этих дочерних классов. Абстрактные классы представляют собой шаблоны для создания классов.

abstract class Person {

private $firstName = "";

private $lastName = "";

public function setName( $firstName, $lastName ) {

$this->firstName = $firstName;

$this->lastName = $lastName;

}

public function getName() {

return "$this->firstName $this->lastName";

}

abstract public function showWelcomeMessage();

/\* абстрактный метод showWelcomeMessage().

Так как он абстрактный, в нем нет ни строчки кода, это просто его объявление.

Любой дочерний класс обязан добавить и описать метод showWelcomeMessage() \*/

}

### **Интерфейс**

**Интерфейс** - это шаблон, который задает поведение одного или более классов. Вот основные отличия между интерфейсами и абстрактными классами:

* Ни один метод не может быть описан в интерфейсе. Они все абстрактны. В абстрактном классе могут быть и не абстрактные методы.
* Интерфейс не может содержать полей - только методы.
* Класс имплементирует интерфейс, и класс наследует или расширяет другой класс.
* Класс может имплементировать несколько интерфейсов одновременно. Этот же класс может наследовать другой класс. Но у дочернего класса может быть только один супер-класс (абстрактный или нет).

interface MyInterface {

public function aMethod();

public function anotherMethod();

}

class MyClass implements MyInterface {

public function aMethod() {

// (имплементация метода)

}

public function anotherMethod() {

// (имплементация метода)

}

}

### **Методы-перехватчики (магические методы)**

* **\_\_get**($property) - вызывается при обращении к неопределенному свойству
* **\_\_set**($property,$value) - вызывается, когда неопределенному свойству присваивается значение
* **\_\_unset**($property) - вызывается, когда функция unset() вызывается для неопределенного свойства
* **\_\_isset**($property) - вызывается, когда функция isset() вызывается для неопределенного свойства
* **\_\_call**($method,$arg array) - вызывается при обращении к неопределенному методу
* **\_\_callStatic**($method,$arg array) - вызывается при обращении к неопределенному статическому методу
* **\_\_toString**() - Вызывается, если есть попытка вывести объект, как строку.
* **\_\_debugInfo**() - В PHP 5.6 был добавлен новый магический метод, который позволяет менять свойства и значения объекта, когда он печатается с помощью функции var\_dump(класс).
* **\_\_invoke**() - для вызова объекта как функции. [Пример](https://htmlweb.ru/php/php6.php#invoke)

#### Пример использования необъявленных свойств класса

##### Где и зачем могут быть использованны методы-перехватчики?

Например есть у вас таблица в базе данных, называется user и есть в ней некие поля, например id, name, email, phone, password, avatar И Вы создали класс на для работы с юзерами, так его и назвали - User

Какие свойства будут у данного класса? Если вы сделаете такие же как в БД - id, name, email и так далее, то получается что при каждом изменении базы данных - вам нужно менять код в классе User, как то не очень удобно. Добавили вы например поле site - значит нужно его добавлять и в класс User, ну и так далее.  
Используя же методы \_\_get() и \_\_set() Вы можете это всё автоматизировать. У вас в классе User вообще не будет ни одного свойства из БД, у нас есть допустим только одно $data - мы туда взяли, да и загрузили всё что есть в базе данных на данного пользователя. А потом, когда программист что то запрашивает, например $user->email мы просто в \_\_get() методе можете посмотреть - если мы такую информацию загрузили из БД, и она лежит в $data['email'] - то вот мы её вам и возвращаем. А в \_\_set() наоборот. Есть такое поле в БД? Значит присвоим ему новое значение.

/\*\*

\* Class User

\* @property-read integer id текущего пользователя

\* @property-write String site возвращает ссылку на сайт пользователя

\*/

class User

{

private $data;

private $f\_write=false;

public function \_\_set($name, $value) {

$this->data[$name] = $value;

$this->f\_write=true; // признак, что нужно сохранить данные

}

public function \_\_get($name) {

if(empty($data)){

// читаем запись из БД в data

}

return $this->data[$name];

}

function \_\_destruct()

{

if(!empty($data)&&$this->f\_write){

// сохраняем изменения в БД

}

}

}

$user=new User();

$user->site='http://kdg.htmlweb.ru/'; //присваеваем переменной

echo $user->site; //выводим значение переменной

// записываем в БД. Можно это явно не делать, т.к. при окончании работы скрипта это поизойдет автоматически

unset($user);

#### Пример использование необъявленного свойства класса как элемент массива

Обратите внимание на то, что из \_\_get возвращается ссылка:

class Foo {

private $data = [];

public function \_\_set($name, $value) {

$this->data[$name] = $value;

}

public function & \_\_get($name) {

return $this->data[$name];

}

}

$foo = new Foo();

$foo->bar[2] = 'lol';

var\_dump($foo->bar);

#### Использоватние перехватчиков обращения к необъявленным методам класса

class OurClass

{

public function \_\_call($name,array $params)

{

echo 'Вы хотели вызвать $Object->'.$name.', но его не существует,

и сейчас выполняется '.\_\_METHOD\_\_.'()';

return;

}

public static function \_\_callStatic($name,array $params)

{

echo 'Вы хотели вызвать '.\_\_CLASS\_\_.'::'.$name.', но его не существует,

и сейчас выполняется '.\_\_METHOD\_\_.'()';

return;

}

}

$Object=new OurClass;

$Object->DynamicMethod();

OurClass::StaticMethod();

Пример обхода закрытых метов класса:

class \_byCallStatic{

// Пример обхода "закрытых" методов класса,

// при использовании метода "\_\_callStatic()" для вызова статического метода.

public static function \_\_callStatic($\_name, $\_param) {

return call\_user\_func\_array('static::'. $\_name, $\_param);

}

private static function \_newCall(){ echo 'Method: '. \_\_METHOD\_\_; }

}

echo \_byCallStatic::\_newCall(114, 'Integer', 157); # Результат: Method: \_byCallStatic::\_newCall

Как вызвать через статический метод любой динамический:

/\*\*

\* Class o

\* @method static void \_\_f(int $a1 = 1)

\*/

class o

{

public static function \_\_callStatic($method, $args)

{

$class = get\_called\_class();

$obj = new $class($args[0]);

$method = substr($method, 2);

$pass = array\_slice($args,1);

$reflection = new ReflectionMethod($obj, $method);

return $reflection->invokeArgs($obj, $pass);

}

public function f($a1 = 1) {

var\_dump('oo', func\_get\_args());

}

}

class a extends o

{

public function f($a1 = 1, $a2 = 2) { var\_dump('aa', $a1 ); }

}

class b extends o

{

public function f($b1 = 1) { var\_dump('bb', $b1); }

}

a::\_\_f(1,2,3);

b::\_\_f(4,5,6);

Полезное описание работы с ReflectionClass, когда вы можете проанализировать свойства и методы класса, проверить параметры по шаблонам и т.д.: http://habrahabr.ru/post/139649/

### **Как использовать объект как функцию?**

class Dog

{

private $name;

public function \_\_construct($dogName = 'Тузик') {

$this->name = $dogName;

}

public static function \_\_invoke() {

$args = func\_get\_args();

echo 'Собака получила: ' . implode(' и ', $args);

}

}

$dog = new Dog('Мухтар');

$dog('кость', 'поводок');

Собака получила: кость и поводок

### **Как обращаться к объекту как к массиву?**

Для этого необходимо создать такой объект который реализует интерфейс ArrayAccess из SPL. Следующий пример реализует объект доступ к данным которого можно получать как в стиле обращения к массиву, так и через получение свойств:

class MyArray implements ArrayAccess

{

protected $arr = array();

public function offsetSet($key, $value) {

$this->arr[$key] = $value;

}

public function offsetUnset($key) {

unset($this->arr[$key]);

}

public function offsetGet($key) {

return $this->arr[$key];

}

public function offsetExists($key) {

return isset($this->arr[$key]);

}

public function \_\_get($key)

{

return $this->offsetGet($key);

}

public function \_\_set($key, $val)

{

$this->offsetSet($key, $val);

}

}

$a = new MyArray();

$a['whoam'] = 'Я значение массива, или объекта? <br/ >';

echo $a['whoam'];

echo $a->whoam;

Я значение массива, или объекта?

Я значение массива, или объекта?

### **Автозагрузка классов**

Файлы автозагружаемых классов обычно располагаются в общем месте, например в /include/class/. Имя файла формируется в формате ИМЯ\_КЛАССА.php. Данный код необходимо подключить во все PHP-скрипты:

spl\_autoload\_register(function ($class\_name) {

//echo "Autoload ".$class\_name;

$file = $\_SERVER['DOCUMENT\_ROOT'] . "/include/class/" . strtolower($class\_name) . '.php';

if (file\_exists($file) == false) {

if($GLOBALS['DEBUG']) echo "Нет файла ".$file;

return false;

}

include\_once($file);

return true;

});

Для автоподгрузки классов можно также использовать определение функции \_\_autoload();

### **Обработка исключений в ООП**

Для обработки некритических ошибок используются исключения(Exception).

try {

$a = 1;

$b = 0;

if($b == 0)

throw new Exception ("деление на ноль!");

$c = $a/$b;

} catch (Exception $e) {

echo $e->getMessage();

echo $e->getLine();

}

Exception — встроенный класс. Если попали в throw, то код ниже не выполняется и осуществляется переход к блоку catch.

Блок try-catch используется как в процедурном, так и в ООП программировании. Он используется для отлова ошибок — большой блок try с множеством throw и все отлавливаются в одном месте — блоке catch.

Exception можно наследовать, желательно при этом перезагрузить конструктор:

class MyException extends Exception {

function \_\_construct($msg){

parent::\_\_construct($msg);

}

}

Блоков catch может быть несколько — для каждого класса наследника Exception.

# Работа с формами

Для передачи данных от пользователя Web-страницы на сервер используются HTML-формы. Для работы с формами в PHP предусмотрен ряд специальных средств.

## **Предварительно определенные переменные**

В PHP существует ряд предварительно определенных переменных, которые не меняются при выполнении всех приложений в конкретной среде. Их также называют переменными окружения или переменными среды. Они отражают установки среды Web-сервера Apache, а также информацию о запросе данного браузера. Есть возможность получить значения URL, строки запроса и других элементов HTTP-запроса.

Все предварительно определенные переменные содержатся в ассоциативном массиве $GLOBALS. Кроме переменных окружения этот массив содержит также глобальные переменные, определенные в программе.

#### Пример 1

<html>

<head>

<title>Просмотр массива $GLOBALS</title>

</head>

<body>

<?php

$a = "Hello!";

foreach ( $GLOBALS as $key=>$value )

echo "\$GLOBALS[\"$key\"] == $value<br>";

?>

</body>

</html>

В результате на экране появится список всех глобальных переменных, включая переменные окружения. Наиболее часто используемые из них:

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Описание** | **Cодержание** |
| **$\_SERVER['HTTP\_USER\_AGENT']** | Название и версия клиента | Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36 |
| **$\_SERVER['REMOTE\_ADDR']** | IP-адрес | 213.87.225.245 |
| **getenv('HTTP\_X\_FORWARDED\_FOR')** | Внутренний IP-адрес клиента |  |
| **$\_SERVER['REQUEST\_METHOD']** | Метод запроса (GET или POST) | GET |
| **$\_SERVER['QUERY\_STRING']** | При запросе GETзакодированные данные, передаваемые вместе с URL |  |
| **$\_SERVER['REQUEST\_URL']** | Полный адрес клиента, включая строку запроса |  |
| **$\_SERVER['HTTP\_REFERER']** | Адрес страницы, с которой был сделан запрос | https://htmlweb.ru/php/php6.php |
| **$\_SERVER['PHP\_SELF']** | Путь к выполняемой программе | /index.php |
| **$\_SERVER['SERVER\_NAME']** | Домен | htmlweb.ru |
| **$\_SERVER['REQUEST\_URI']** | Путь | /php/php\_form.php |

## **Обработка ввода пользователя**

PHP-программу обработки ввода можно отделить от HTML-текста, содержащего формы ввода, а можно расположить на одной странице.

#### Пример 2

<?php

if ( ! isset( $cardnumber ) )

$m = "Введите номер карточки";

elseif ( $cardnumber == "" )

$m = "Ваш номер пуст!";

else

$m = "Ваш номер: $cardnumber";

?>

<html>

<head>

<title>Пример обработки ввода</title>

</head>

<body>

<h1><?php echo $m?></h1>

<form name="form1" action="<?=$\_SERVER['PHP\_SELF']?>" method="post">

<p>Номер карточки:

<input name="cardnumber" type="text" size="20">

</form>

</body>

</html>

Здесь отсутствует кнопка передачи данных, т.к. форма, состоящая из одного поля, передается автоматически при нажатии клавиши <Enter>.

При обработки элемента с многозначным выбором для доступа ко всем выбранным значениям нужно к имени элемента добавить пару квадратных скобок. Для выбора нескольких эллементов следует удерживать клавишу Ctrl.

#### Пример 3.1

Файл ex.htm

<html>

<head>

<title>Список</title>

</head>

<body>

<form name="form1" action="ex1.php" method="post">

<select name="Item[]" size=5 multiple>

<option>Чай

<option>Кофе

<option>Молоко

<option>Ветчина

<option>Сыр

</select>

<input type="submit" value="ВВОД">

</form>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 3.1:**

Начало формы





Конец формы

#### Пример 3.2

Файл ex.php

<html>

<head>

<title>Обработка списка из файла ex1.htm</title>

</head>

<body>

<?php

echo "Ваш заказ: <p><ul>";

foreach ( $Item as $value ) echo "<li>$value";

echo "</ul>";

?>

</body>

</html>

#### Пример 4. Прием значений от checkbox-флажков

<?php

if (@$\_REQUEST['doGo']) {

foreach (@$\_REQUEST['known'] as $k=>$v) {

if($v) echo "Вы знаете язык программирования $k!<br>";

else echo "Вы не знаете языка программирования $k. <br>";

}

}

?>

<form action="<?=$\_SERVER['SCRIPT\_NAME']?>" method="post">

Какие языки программирования вы знаете?<br>

<input type=hidden name="known[PHP]" value="0">

<input type=checkbox name="known[PHP]" value="1"> PHP<br>

<input type=hidden name="known[Perl]" value="0">

<input type=checkbox name="known[Perl]" value="1"> Perl<br>

<input type=submit name="doGo" value="Go!">

</form>

**РЕЗУЛЬТАТ ПРИМЕРА 4:**

Начало формы

Какие языки программирования вы знаете?

 PHP

 Perl



Конец формы

#### Пример 5

<?php ## Модель скрипта, принимающего текст от пользователя.

if (@$\_REQUEST['text'])

echo htmlspecialchars($\_REQUEST['text'])."<hr>";

?>

<form action="<?=$\_SERVER['SCRIPT\_NAME']?>" method="post">

<textarea name="text" cols="60" rows="10">

< ?=@htmlspecialchars($\_REQUEST['text'])?>

</textarea><br>

<input type="submit">

</form>

Можно обрабатывать формы, не заботясь о фактических именах полей.

Для этого можно использовать (в зависимости от метода передачи) ассоциативный массив $HTTP\_GET\_VARS или $HTTP\_POST\_VARS. Эти массивы содержат пары имя/значение для каждого элемента переданной формы. Если Вам все равно, Вы можете использовать ассоциативный массив $\_REQUEST.

#### Пример 6

<html>

<head>

<title>Обработка произвольного ввода

        независимо от метода передачи</title>

</head>

<body>

<?php

$params = ( $\_SERVER['REQUEST\_METHOD'] == "GET" )

? $HTTP\_GET\_VARS : $HTTP\_POST\_VARS;

foreach ( $params as $key=>$value )

echo "$key == $value<br>";

?>

</body>

</html>

#### Пример 7. Обработка нажатия на кнопку с использованием оператора '@'.

<?php

if (@$\_REQUEST['submit']) echo "Кнопка нажата!"

?>

<form action="<?=$\_SERVER['SCRIPT\_NAME']?>">

<input type="submit" name="submit" value="Go!">

</form>

С помощью функции header(), послав браузеру заголовок "Location", можно перенаправить пользователя на новую страницу.

Например:

<? header("Location: ex2.php"); ?>

## **Передача файла на сервер. Залить файл. UpLoad**

PHP позволяет передавать на сервер файлы. HTML-форма, предназначенная для передачи файла, должна содержать аргумент enctype="multipart/form-data".

Кроме того в форме перед полем для копирования файла должно находиться скрытое поле с именем max\_file\_size. В это скрытое поле должен быть записан максимальный размер передаваемого файла (обычно не больше 2 Мбайт).

Само поле для передачи файла - обычный элемент INPUT с аргументом type="file".

Например:

<form enctype="multipart/form-data"

    action="<?echo $\_SERVER['PHP\_SELF']?>" method="post">

<input type="hidden" name="max\_file\_size" value="51200">

<input type="file" name="myfile"><br>

<input type="submit" value="ВВОД">

</form>

После того, как файл передан на сервер, он получает уникальное имя и сохраняется в каталоге для временных файлов. Полный путь к файлу записывается в глобальную переменную, имя которой совпадает с именем поля для передачи этого файла. Кроме этого PHP сохраняет еще некоторую дополнительную информацию о переданном файле в других глобальных переменных:

|  |  |
| --- | --- |
| **Переменная** | **Описание** |
| **$\_FILES['userfile']['name']** | оригинальное имя файла, такое, каким его видел пользователь, выбирая файл |
| **$\_FILES['userfile']['type']** | mime/type файла, к примеру, может быть image/gif; это поле полезно сохранить, если Вы хотите предоставлять интерфейс для скачивания загруженных файлов |
| **$\_FILES['userfile']['size']** | размер загруженного файла |
| **$\_FILES['userfile']['tmp\_name']** | полный путь к временному файлу на диске |
| **$\_FILES['userfile']['error']** | код ошибки, который равен 0, если операция прошла успешно |

#### Пример 8

<html>

<head>

<title>Обработка переданного файла</title>

</head>

<?php

if ( isset( $\_FILES['userfile']['tmp\_name'] ) )

{

echo "путь: ".$\_FILES['userfile']['tmp\_name']."<br>";

echo "имя: ".$\_FILES['userfile']['name']."<br>";

echo "размер: ".$\_FILES['userfile']['size']."<br>";

echo "тип: ".$\_FILES['userfile']['type']."<br>";

}

?>

<body>

<form enctype="multipart/form-data"

    action="<?echo $\_SERVER['PHP\_SELF']?>" method="post">

<input type="hidden" name="max\_file\_size" value="51200">

<input type="file" name="userfile"><br>

<input type="submit" value="ВВОД">

</form>

</body>

</html>

[Примеры загрузки файлов на сервер](https://htmlweb.ru/php/php_upload.php)

Если возникнут проблеммы с перекодировкой сервером загруженного файла, символ с кодом **0х00**заменен на пробел (символ с кодом **0х20**), допишите в файл **httpd.conf** из каталога Апача (/usr/local/apache) следующие строки.

<Location>

CharsetRecodeMultipartForms Off

</Location>

# Работа с файлами

Работа с файлами - важный инструмент PHP.

## **Включение внешних файлов**

В каждый PHP-документ можно включить файл с помощью инструкции [**include()**](https://htmlweb.ru/php/function/include.php). Её аргумент: путь к файлу. Этой инструкцией удобно пользоваться при наличии одинаковых кусков кода во многих PHP-программах. Содержимое включаемого файла обрабатывается как простой HTML-текст. Для того, чтобы содержимое этого файла обрабатывалось как PHP-программа, нужно обрамлять его открывающим и закрывающим тегами PHP.

#### Пример 1

<html>

<head>

<title>Использование include()</title>

</head>

<body>

<?php

include("top.php");

echo "<H2>...Основная часть...</H2>";

?>

</body>

</html>

Включаемый файл top.php с PHP-программой:

echo "&lt;H1 align=center&gt;...Общее приветствие...&lt;/H1&gt;"

**РЕЗУЛЬТАТ ПРИМЕРА 1:**

# ...Общее приветствие...

## **...Основная часть...**

В данном случае результат будет тем же, если включаемый файл top.php содержит только строку HTML-текста:

<H1 align=center>...Общее приветствие...</H1>

Включаемые файлы могут возвращать значения подобно функциям. Использование оператора **return**прерывает выполнение этого файла так же, как и функции.

#### Пример 2

<html>

<head>

<title>Использование инструкции include(), возвращающей значение</title>

</head>

<body>

<?php

$res = include("top.php");

echo "<H2>Включаемый файл вернул $res</H2>";

?>

</body>

</html>

Включаемый файл top.php с PHP-программой:

<?php

$a = 7 \* 8;

return $a;

?>

**РЕЗУЛЬТАТ ПРИМЕРА 2:**

## **Включаемый файл вернул 56**

Инструкцию **include()** можно использовать внутри цикла. В цикле **include()** выполняется при каждой итерации. Это можно использовать для включения нескольких файлов. Например:

for ($i=1; $i<=5; $i++)

include("incfile{$i}.htm");

Определение имени включаемого файла и его загрузка производятся повторно при каждом вызове **include()**. Это означает, что если содержание включаемого файла с момента предыдущего вызова изменилось, то загрузится новое содержание.

Оператор **include()** также можно включать в тело условного оператора.

Несмотря на сходство по внешнему виду с функцией, **include()** функцией не является, а представляет собой специальную конструкцию языка.

Для указания что файл нужно подключать только один раз используется оператор [include\_once()](https://htmlweb.ru/php/function/include_once.php)

## **Анализ файлов**

PHP содержит множество функций, дающих информацию о файлах. Наиболее употребимыми являются:

* [**file\_exists()**](https://htmlweb.ru/php/function/file_exists.php) - определяет существование файла. Например:
* if ( !file\_exists( "aaa.php" ) )
* echo "Внимание! Файл aaa.php не найден!";
* [**is\_file()**](https://htmlweb.ru/php/function/is_file.php) - определяет, является ли исследуемый объект файлом. Например:
* if ( is\_file( "bbb.txt" ) )
* echo "Можете не сомневаться, bbb.txt - это файл";
* [**is\_dir()**](https://htmlweb.ru/php/function/is_dir.php) - определяет, является ли исследуемый объект каталогом. Например:
* if ( is\_dir( "/tmp" ) )
* echo "Действительно, /tmp - это каталог";
* [**is\_readable()**](https://htmlweb.ru/php/function/is_readable.php) - определяет, доступен ли файл для чтения. Например:
* if ( is\_readable( "db.dbf" ) )
* echo "db.dbf можно читать";
* [**is\_writable()**](https://htmlweb.ru/php/function/is_writable.php) - определяет, доступен ли файл для записи. Например:
* if ( is\_writable( "db.dbf" ) )

echo "В db.dbf писать можно";

* [**filesize()**](https://htmlweb.ru/php/function/filesize.php) - определяет размер файла в байтах.
* [**filemtime()**](https://htmlweb.ru/php/function/filemtime.php) - определяет дату и время последнего изменения файла.
* [**fileatime()**](https://htmlweb.ru/php/function/fileatime.php) - определяет дату и время последнего обращения к файлу.

Время возвращается в формате Unix, т.е. представляет собой количество секунд, прошедших после 1 января 1970 г. В примере 2 это число преобразуется в понятный для человека формат с помощью функции **date( )**. Подробнее об этой функции можно узнать в разделе ["Дополнительные возможности"](https://htmlweb.ru/php/php10.php).

#### Пример 3

<html>

<head>

<title>Информация о файле</title>

</head>

<body>

<?php

$file = "top.php";

infoFile ( $file );

function infoFile ( $f )

{

if ( !file\_exists( $f ) )

{

echo "$f не найден!";

return;

}

echo "$f - ".( is\_file( $f ) ? "" : "не " )."файл<br>";

echo "$f - ".( is\_dir( $f ) ? "" : "не " )."каталог<br>";

echo "$f ".( is\_readable( $f ) ? "" : "не " )."доступен для чтения<br>";

echo "$f ".( is\_writable( $f ) ? "" : "не " )."доступен для записи<br>";

echo "размер $f в байтах - ".( filesize( $f ) )."<br>";

echo "последнее изменение $f - ".( date( "d M Y H:i", filemtime( $f ) ) )."<br>";

echo "последнее обращение к $f - ".( date( "d M Y H:i", fileatime( $f ) ) )."<br>";

}

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 3:**

top.php - файл

top.php - не каталог

top.php доступен для чтения

top.php доступен для записи

размер top.php в байтах - 732

последнее изменение top.php - 04 Oct 2005 20:21

последнее обращение к top.php - 20 Oct 2005 14:01

Внимание! С удаленными файлами эти функции не работают. Их можно применять только к локальной файловой системе.

## **Управление файлами**

PHP содержит множество функций управления файлами. Наиболее употребимыми являются:

* [**touch()**](https://htmlweb.ru/php/function/touch.php) - создает пустой файл с заданным именем. Если такой файл уже существует, то функция изменит дату модификации. Например:
* touch( "ex1.txt" );
* [**copy()**](https://htmlweb.ru/php/function/copy.php) - копирует файл. Для копирования файлов в php применяется функция copy ($source, $result). Ей нужно передать лишь два параметра - источник $source и имя файла-копии - $result. Стоит отметить, что следует указывать полные адреса к файлам. Пример применения функции copy:
* <?php
* if (copy('Z:/home/site/www/file.txt', 'Z:/home/site/www/file2.txt'))
* { echo "Копирование успешно выполнено"; }
* else
* { echo "Ошибка при копировании"; }

?>

* [**unlink()**](https://htmlweb.ru/php/function/unlink.php) - удаляет заданный файл. Например:
* <?php
* if (unlink('filename.txt'))
* { echo "Файл удален"; }
* else
* { echo "Ошибка при удалении файла"; }

?>

* **fopen( )** - открывает локальный или удаленный файл и возвращает указатель на него. Указатель используется во всех операциях с содержимым файла. Аргументы: имя файла и режим открытия.

|  |  |
| --- | --- |
| r | чтение. Указатель файла устанавливается на его начало |
| r+ | чтение и запись. Указатель файла устанавливается на его начало |
| w | запись. Указатель файла устанавливается на его начало. Все старое содержимое файла теряется. Если файл с указанным именем не существует, функция пытается его создать |
| w+ | чтение и запись. Указатель файла устанавливается на его начало. Все старое содержимое файла теряется. Если файл с указанным именем не существует, функция пытается его создать |
| a | запись. Указатель файла устанавливается на его конец. Если файл с указанным именем не существует, функция пытается его создать |
| a+ | чтение и запись. Указатель файла устанавливается на его конец. Если файл с указанным именем не существует, функция пытается его создать |

* Например:
* $fp = fopen( "http://www.php.net/", "r" ); // для чтения
* $fp = fopen( "ex1.txt", "w" ); // для записи
* $fp = fopen( "ex2.txt", "a" ); // для добавления в конец
* Если открыть файл не удалось, то можно прервать выполнение программы. Например:
* $fp = fopen( "ex1.txt", "w" ) or die ( "Не удалось открыть файл" );
* [**fclose()**](https://htmlweb.ru/php/function/fclose.php) - закрывает файл. Аргумент: указатель файла, полученный ранее от функции **fopen()**. Например:
* fclose( $fp );
* [**feof()**](https://htmlweb.ru/php/function/feof.php) - проверка конца файла. Аргумент: указатель файла.
* [**fgetc()**](https://htmlweb.ru/php/function/fgetc.php) - чтение очередного символа из файла. Аргумент: указатель файла.
* [**fgets()**](https://htmlweb.ru/php/function/fgets.php) - чтение очередной строки файла. Аргументы: указатель файла и длина считываемой строки. Операция прекращается либо после считывания указанного количества символов, либо после обнаружения конца строки или файла.

#### Пример 4

<html>

<head>

<title>Чтение строк из файла</title>

</head>

<body>

<?php

$fp = fopen( "ex1.txt", "r" ) or die ( "Не удалось открыть файл" );

while ( ! feof ( $fp ) )

echo ( fgets( $fp, 1024 ) )."<br>";

?>

</body>

</html>

* [**fread()**](https://htmlweb.ru/php/function/fread.php) - общая функция чтения из файла. Аргументы: указатель файла и количество считываемых символов.
* [**fseek()**](https://htmlweb.ru/php/function/fseek.php) - отступ от начала файла. Аргументы: указатель файла и смещение.

#### Пример 5

<html>

<head>

<title>Вывод на экран второй половины файла</title>

</head>

<body>

<?php

$f = "ex1.txt";

$fp = fopen( $f, "r" ) or die ( "Не удалось открыть $f" );

$fsize = filesize( $f );

$half = (int)( $fsize / 2 );

fseek ( $fp, $half );

echo ( fread( $fp, ($fsize - $half) ) );

?>

</body>

</html>

* [**fputs()**](https://htmlweb.ru/php/function/fputs.php) - запись строки в файл. Аргументы: указатель файла и строка.
* [**fwrite()**](https://htmlweb.ru/php/function/fwrite.php) - полный аналог функции **fputs( )**.

#### Пример 6

<html>

<head>

<title>Запись и добавление в файл</title>

</head>

<body>

<?php

$fp = fopen( "ex2.txt", "w" ) or die ( "Не удалось открыть файл" );

fputs( $fp, "Запись в файл\n" );

fclose( $fp );

$fp = fopen( "ex2.txt", "a" ) or die ( "Не удалось открыть файл" );

fputs( $fp, "Добавление в конец файла" );

fclose( $fp );

?>

</body>

</html>

* [**flock()**](https://htmlweb.ru/php/function/flock.php) - блокирует файл, т.е. не позволяет другим пользователям читать этот файл или писать в него, пока тот, кто наложил блокировку не закончит работу с данным файлом. Аргументы: указатель файла и номер режима блокировки.

|  |  |
| --- | --- |
| 1 | Можно читать, нельзя писать |
| 2 | Нельзя ни читать, ни писать |
| 3 | Снятие блокировки |

#### Пример 7

* <html>
* <head>
* <title>Блокировка файла</title>
* </head>
* <body>
* <?php
* $fp = fopen( "ex1.txt", "a" ) or die ( "Не удалось открыть файл" );
* flock( $fp, 2 ); // Полная блокировка
* // Запись в файл
* flock( $fp, 3 ); // Снятие блокировки
* fclose( $fp );
* ?>
* </body>
* </html>
* Блокировка с помощью **flock()** не является абсолютной. С ней будут считаться только те программы, которые тоже используют эту функцию.

В PHP версии >= 4.3.0 появились новые удобные функции работы с файлами:

* [**file\_get\_contents()**](https://htmlweb.ru/php/function/file_get_contents.php) - прочитать весь файл или URL
* [**file\_put\_contents()**](https://htmlweb.ru/php/function/file_put_contents.php) - записать файл

## **Работа с каталогами**

В PHP есть несколько функций для работы с каталогами:

* [**mkdir()**](https://htmlweb.ru/php/function/mkdir.php) - создание каталога. Аргументы: путевое имя каталога и режим доступа. Режим доступа - восьмеричное число из трех цифр с ведущим нулем. Первая цифра - право доступа для владельца, вторая - для группы, третья - для всех остальных. В системе Windows режим доступа игнорируется. В системе UNIX определены следующие права доступа:

|  |  |
| --- | --- |
| 0 | Нет доступа |
| 1 | Доступ для запуска |
| 2 | Доступ для записи |
| 3 | Доступ для записи и запуска |
| 4 | Доступ для чтения |
| 5 | Доступ для чтения и запуска |
| 6 | Доступ для чтения и записи |
| 7 | Полный доступ |

* Например, право полного доступа для владельца, чтения и запуска для группы и запуска для всех остальных:
* mkdir( "testdir", 0751 );
* [**rmdir()**](https://htmlweb.ru/php/function/rmdir.php) - удаление каталога. Аргумент: путевое имя каталога. Удалить можно лишь пустой каталог при наличии права на это. При успешном выполнении функция возвращает **true**.
* [**opendir()**](https://htmlweb.ru/php/function/opendir.php) - открытие каталога. Аргумент: путевое имя каталога. В случае возникновения ошибки функция возвращает **false**. Ошибка при открытии может быть вызвана тем, что каталог не существует или программа не имеет права его читать.
* [**readdir()**](https://htmlweb.ru/php/function/readdir.php) - чтение каталога. Аргумент: путевое имя каталога. Возвращает строку, содержащую имя найденного файла или подкаталога. По достижении конца каталога функция возвращает **false**.

#### Пример 8

<html>

<head>

<title>Чтение каталога</title>

</head>

<body>

<?php

$d = "testdir";

$dh = opendir( $d ) or die ( "Не удалось открыть каталог $d" );

while ( $f = readdir( $dh ) )

echo "$f<br>";

?>

</body>

</html>

Проверочное выражение цикла сработает не так, как хотелось бы, если в каталоге есть файл с именем "0". В таком случае имя "0" будет преобразовано в 0, что вызовет завершение цикла. Избежать подобного можно, если принять дополнительные меры безопасности, изменив проверочное выражение:

gettype( $f = readdir( $dh ) ) != "boolean"

Помимо использования функция fgetc() fgets() fgetss(), Вы можете использовать  
$file = file($filename). Она читает $filename в массив($file).  
  
Пример:

$file=file($filename);

for ($i=0; $i<=count($file); $i++) {

echo $file[$i], "<br />";

}

## **Передача файла с сервера. Скачать файл. DownLoad.**

Ручной Download, или как скриптом открыть пользователю окошко сохранения файла. Чтобы в браузере открылся диалог "Сохранить файл" с названием файла file.dat, скрипт должен послать такие заголовки:

header("Content-Disposition: attachment; filename=file.dat");

header("Content-Type: application/x-force-download; name=\"file.dat\"");

Готовый пример отправки файла с диска:

<?php

$filename = "DownloadedFile.mp3"; // имя файл предложенное для сохранения в окне браузера

$myFile = $\_SERVER['DOCUMENT\_ROOT']."/my/file.mp3"; // файл на серевере

$mm\_type="application/octet-stream";

header("Cache-Control: public, must-revalidate"); // кешировать

header("Pragma: hack");

header("Content-Type: " . $mm\_type);

header("Content-Length: " .(string)(filesize($myFile)) );

header('Content-Disposition: attachment; filename="'.$filename.'"');

header("Content-Transfer-Encoding: binary");

readfile($myFile); // прочитать файл и отправить в поток

?>

Не мешало бы также ссылаться на скрипт как http://.../download.php?file.dat, иначе браузеры, основанные на Mozilla, могут попытаться записать файл как file.dat.php. или указать атрибут download:

<a href="имя.php" download="file.dat">Скачать</a>

Примечание: Не используйте заголовок header("Cache-Control: no-cache") в таких скриптах, большинство версий Internet Explorer не смогут скачать файл.

Следующий пример скрипта демонстрирует **как организовать поддержку докачки**:

Предполагается, что $cd - путь к файлу, $fn - имя файла

<?php

$filename = $cd."/".$fn;

if (!file\_exists($filename))

{

header ("HTTP/1.0 404 Not Found");

exit;

}

$fsize = filesize($filename);

$ftime = date("D, d M Y H:i:s T", filemtime($filename));

$fd = @fopen($filename, "rb");

if (!$fd){

header ("HTTP/1.0 403 Forbidden");

exit;

}

// Если запрашивающий агент поддерживает докачку

if ($HTTP\_SERVER\_VARS["HTTP\_RANGE"]) {

$range = $HTTP\_SERVER\_VARS["HTTP\_RANGE"];

$range = str\_replace("bytes=", "", $range);

$range = str\_replace("-", "", $range);

if ($range) {fseek($fd, $range);}

}

$content = fread($fd, filesize($filename));

fclose($fd);

if ($range) {

header("HTTP/1.1 206 Partial Content");

}

else {

header("HTTP/1.1 200 OK");

}

header("Content-Disposition: attachment; filename=$fn");

header("Last-Modified: $ftime");

header("Accept-Ranges: bytes");

header("Content-Length: ".($fsize-$range));

header("Content-Range: bytes $range-".($fsize -1)."/".$fsize);

header("Content-type: application/octet-stream");

print $content;

exit;

?>

[Как передать файл на сервер?](https://htmlweb.ru/php/php_form.php#Upload)

## **Функции для работы с каталогами**

* [**chdir()**](https://htmlweb.ru/php/function/chdir.php)**- Сменить каталог**

boolean chdir (string str)

Изменяет текущий каталог PHP на указанный в качестве параметра str. Возвращает TRUE в случае успешного завершения, FALSE в случае возникновения ошибки

* [**chroot()**](https://htmlweb.ru/php/function/chroot.php)**- Сменить корневой каталог**

boolean chroot (string str)

Изменяет корневой каталог текущего процесса на переданный в качестве параметра str. Возвращает TRUE в случае успешного завершения, FALSE в случае возникновения ошибки..

Замечание: Для Windows-платформ эта функция не реализована.

* [**closedir()**](https://htmlweb.ru/php/function/closedir.php)**- Освободить дескриптор каталога**

void closedir (resourse catalog)

Закрывает поток, связанный с каталогом и переданный в качестве параметра catalog. Перед использованием данной функции, поток должен быть открыт с помощью функции opendir().

#### dir - Класс directory

class dir

{

dir (string str)

string path

resource handle

string read (void)

void rewind (void)

void close (void)

}

Псевдо-объектно-ориентированный механизм для чтения каталога, переданного в параметре каталог. С момента открытия каталога становятся доступными два свойства класса. Свойство "handle" может быть использовано с другими функциями для работы с каталогами, например, с функциями [readdir()](https://htmlweb.ru/php/function/readdir.php), [rewinddir()](https://htmlweb.ru/php/function/rewinddir.php)и [closedir()](https://htmlweb.ru/php/function/closedir.php). Свойство "path" содержит путь к открытому каталогу. Доступны три метода: read, rewind and close.

Пожалуйста, обратите внимание на способ, которым осуществляется проверка значения, возвращаемого методами класса dir() в примере, приведенном ниже. В этом примере проводится проверка значения на идентичность (выражения идентичны, когда они равны и являются одного типа - за более подробной информацией обратитесь к главе Операторы сравнения) значению FALSE, поскольку в ином случае, любой элемент каталога, чье имя может быть выражено как FALSE, остановит цикл.

Пример кода:

<?php

$d = dir("/etc");

echo "Дескриптор: ".$d->handle."<br>\n";

echo "Путь: ".$d->path."<br>\n";

while (false !== ($entry = $d->read())) {

echo $entry."<br>\n";

}

$d->close();

?>

Замечание: Порядок, в котором метод "read" возвращает элементы каталога, зависит от операционной системы.

Замечание: Также, PHP автоматически определяет внутренний класс Directory, что означает, что вы не сможете определять собственные классы с таким же именем.

* [**getcwd()**](https://htmlweb.ru/php/function/getcwd.php)**- Получить имя текущего рабочего каталога**

string getcwd (void)

Возвращает имя текущего рабочего каталога.

* [**opendir()**](https://htmlweb.ru/php/function/opendir.php)**- Открыть каталог**

resourse opendir (string path)

Возвращает дескриптор каталога для последующего использования с функциями closedir(), readdir() и rewinddir().

Если путь не существует или каталог, расположенный по указанному пути, не может быть открыт вследствие правовых ограничений или ошибок файловой системы, функция opendir() возвращает значение FALSE и генерирует сообщение PHP об ошибке уровня E\_WARNING. Вы можете запретить вывод сообщения об ошибке, предварив имя функции opendir() символом '@'.

Пример кода

<?php

$dir = "/tmp/";

// Открыть заведомо существующий каталог

// и начать считывать его содержимое

if (is\_dir($dir))

{

if ($dh = opendir($dir))

{

while (($file = readdir($dh)) !== false)

{

print "Файл: $file : тип: " . filetype($dir . $file) . "\n";

}

closedir($dh);

}

}

?>

Начиная с версии PHP 4.3.0, параметр путь может также являться любым URL'ом, обращение к которому приводит к получению списка его файлов и каталогов. Однако, данный способ работает только при использовании url-упаковщика file://. В версии PHP 5.0.0 была добавлена поддержка url-упаковщика ftp://.

* [**readdir**](https://htmlweb.ru/php/function/readdir.php)**- Получить элемент каталога по его дескриптору**

string readdir (resource catalog)

Возвращает имя следующего по порядку элемента каталога. Имена элементов возвращаются в порядке, зависящем от файловой системы.

Обратите внимание на способ проверки значения, возвращаемого функцией readdir() в приведенном ниже примере. В этом примере осуществляется проверка значения на идентичность (выражения идентичны, когда они равны и являются значениями одного типа - за более подробной информацией обратитесь к главе Операторы сравнения) значению FALSE, поскольку в ином случае, любой элемент каталога, чье имя может быть выражено как FALSE, остановит цикл (например, элемент с именем "0").

Пример кода

<?php

// Вывести список всех файлов в каталоге

// Оператор !== не существовал до версии 4.0.0-RC2

if ($handle = opendir('/path/to/files'))

{

echo "Дескриптор каталога: $handle\n";

echo "Файлы:\n";

/\* Именно этот способ чтения элементов

каталога является правильным. \*/

while (false !== ($file = readdir($handle)))

{

echo "$file\n";

}

// Этот способ НЕВЕРЕН.

while ($file = readdir($handle))

{

echo "$file\n";

}

closedir($handle);

}

?>

Обратите внимание, что функция readdir() также возвращает элементы с именами, и, если вы не хотите получать эти значения, просто отбрасывайте их:

Пример кода

<?php

if ($handle = opendir('.'))

{

while (false !== ($file = readdir($handle)))

{

if ($file != "." && $file != "..")

{

echo "$file\n";

}

}

closedir($handle);

}

?>

* [**rewinddir()**](https://htmlweb.ru/php/function/rewinddir.php)**- Сбросить дескриптор каталога**

void rewinddir (resource catalog)

Сбрасывает поток каталога, переданный в параметре catalog таким образом, чтобы тот указывал на начало каталога

* [**scandir()**](https://htmlweb.ru/php/function/scandir.php)**- Получить список файлов и каталогов, расположенных по указанному пути**

array scandir (string catalog [, int order])

Возвращает массив, содержащий имена файлов и каталогов, расположенных по пути, переданном в параметре catalog. Если каталог не является таковым, функция возвращает логическое значение FALSE и генерирует сообщение об ошибке уровня E\_WARNING.

По умолчанию, сортировка производится в алфавитном порядке по возрастанию. Если указан необязательный параметр order (равен 1), сортировка производится в алфавитном порядке по убыванию. Пример кода

<?php

$dir = '/tmp';

$files1 = scandir($dir);

$files2 = scandir($dir, 1);

print\_r($files1);

print\_r($files2);

?>

Результатом будет нечто вроде:

Array

(

[0] => .

[1] => ..

[2] => bar.php

[3] => foo.txt

[4] => somedir

)

Array

(

[0] => somedir

[1] => foo.txt

[2] => bar.php

[3] => ..

[4] => .

)

Пример кода

<?php

// Альтернативный вариант функции scandir() для PHP 4

$dir = "/tmp";

$dh = opendir($dir);

while (false !== ($filename = readdir($dh)))

{

$files[] = $filename;

}

sort($files);

print\_r($files);

rsort($files);

print\_r($files);

?>

**РЕЗУЛЬТАТ ПРИМЕРА:**

Array ( [0] => . [1] => .. [2] => bar.php [3] => foo.txt [4] => somedir ) Array ( [0] => somedir [1] => foo.txt [2] => bar.php [3] => .. [4] => . )

# Загрузка файлов на сервер

# Краткий эксурс в upload

Что такое Upload files, или почему не работает   
copy ("c:\images\sample.jpg", "http://mysite.ru/uploads/sample.jpg ")

Даже если у Вас в распоряжении всего один компьютер, на котором совмещен и сервер и рабочая станция, не стоит забывать о том, что php использует технологию клиент/сервер. Файл, который мы хотим загрузить, как правило, находится на машине клиента, т.е. пользователя, обыкновенного посетителя сайта. Место назначения - сервер. Для того чтобы совершить процесс передачи файла, нам понадобиться следующая форма:

<form enctype="multipart/form-data" action="/upload.php" method="post">

<input type="hidden" name="MAX\_FILE\_SIZE" value="30000">

Send this file: <input name="userfile" type="file">

<input type="submit" value="Send File">

</form>

При этом в поле action должен быть указан URL Вашего php-скрипта, который в дальнейшем будет заниматься обработкой загружаемых файлов. Скрытое поле MAX\_FILE\_SIZE должно предшествовать полю выбора файла, и содержать максимально допустимый размер файла в байтах. Его назначение - проверка размера файла еще до момента отправки файла на сервер. Это должно избавить пользователя от длительной и безрезультатной загрузки файла на сервер и образования лишнего трафика, но не стоит особо полагаться на это ограничение, так как его легко обойти.

Что происходит, когда пользователь выбрал файл на своем диске, и нажал на кнопку "Send file"? Браузер отсылает файл на сервер, где php-интерпретатор помещает его в свою временную директорию, присваивая ему случайное имя и выполняет скрипт, указанный в поле action.

Как должен выглядеть upload.php?

$uploaddir = '/var/www/uploads/';

if (move\_uploaded\_file($\_FILES['userfile']['tmp\_name'], $uploaddir.

$\_FILES['userfile']['name'])) {

print "File is valid, and was successfully uploaded.";

} else {

print "There some errors!";

}

При написании скрипта, возникает естественный вопрос: как получить информацию о загруженном файле и достучаться до самого файла. Если Вы используете PHP версии 4.1.0 и старше, лучше всего будет обратиться к глобальному массиву $\_FILES. Для каждого загруженного файла он содержит хеш-массив, со следующими данными:

* **$\_FILES['userfile']['name']** - оригинальное имя файла, такое, каким его видел пользователь, выбирая файл;
* **$\_FILES['userfile']['type']** - mime/type файла, к примеру, может быть image/gif; это поле полезно сохранить, если Вы хотите предоставлять интерфейс для скачивания загруженных файлов;
* **$\_FILES['userfile']['size']** - размер загруженного файла;
* **$\_FILES['userfile']['tmp\_name']** - полный путь к временному файлу на диске;
* **$\_FILES['userfile']['error']** - Начиная с версии 4.2.0, содержит код ошибки, который равен 0, если операция прошла успешно.

Для PHP версии ниже 4.1.0 этот массив называется $HTTP\_POST\_FILES. Не стоит забывать, что в отличие от $\_FILES этот массив не является суперглобальным и при обращении к нему, к примеру, из функции, необходимо явно указывать global $HTTP\_POST\_FILES;

Если в настройках Вашего сервера register\_globals=on, будут созданы дополнительные переменные вида $userfile\_name, $userfile\_type, $userfile\_size… Учитывая, что, начиная с версии 4.2.0, в настройках по умолчанию register\_globals=off использования этих переменных не рекомендовано, даже если они определены. Лучший способ получения информации о загружаемых файлах - использовать массив $\_FILES.

Для работы с загруженными файлами лучше всего использовать встроенные функции [is\_uploaded\_file()](https://htmlweb.ru/php/function/is_uploaded_file.php) и [move\_uploaded\_file()](https://htmlweb.ru/php/function/move_uploaded_file.php), которые проверяют, был ли загружен файл, и помещают его в указанную папку соответственно. Более детальную информацию Вы можете найти на страницах руководства. Не стоит изобретать велосипед и работать самому с временными файлами, копировать их, удалять. Это уже сделано до Вас и для Вас.

# Настройка сервера

Я все сделал правильно, но у меня что-то не работает. Может, у меня неправильно сконфигурирован сервер?

Если Вы "все сделали правильно", но Ваш код неработает, или работает неправильно, не спешите отчаиваться. Возможно проблема не в Ваших руках, а в неверных настройках сервера. Вот список директив, которые имеют отношения к загрузке файлов:

В файле php.ini:

* Если Вы хотите узнать, где расположен Ваш php.ini, выполните

<?php phpinfo(); ?>

* **file\_uploads** - возможность запретить или разрешить загрузку файлов в целом. По умолчанию On.
* **upload\_max\_filesize** - максимальный размер файла, который может быть загружен. Если Вам необходимо работать с большими файлами, измените эту настройку. По умолчанию 2М. Не забудьте изменить post\_max\_size.
* **post\_max\_size** - общее ограничение сверху на размер данных, передаваемых в POST запросе. Если Вам необходимо работать с большими файлами, или передавать несколько файлов одновременно, измените эту настройку. Значение по умолчанию 8М.
* **upload\_tmp\_dir** - временная директория на сервере, в которую будут помещаться все загружаемые файлы. Проверьте, какие на нее выставлены права(если на данном этапе у Вас возникли сложности, смотрите пояснения в конце статьи). Такая директория должна существовать и у пользователя, под которым выполняется Apache, также должны быть права на запись в эту директорию. Если Вы работаете с включенным ограничением open\_basedir - то временный каталог должен находиться внутри. Вам не нужно заботиться о ее чистке или об уникальности имен, PHP решает эту проблему за Вас.

В файле **httpd.conf**:

* Прежде всего, убедитесь, что Вы используете веб-сервер Apache 1.3 (последняя версия на момент написания статьи - 1.3.27). Если Вы используете Apache 2.0, Вам следует прочитать следующий отрывок из документации:

Do not use Apache 2.0 and PHP in a production environment neither on Unix nor on Windows.

* Если Вы получили сообщение "POST Method Not Allowed", это означает, что надо искать что-то похожее на следующие директивы, и использовать ключевое слово Allow:
* <Limit POST>
* Order allow,deny
* Allow from all
* </Limit>
* Проблемы с загрузкой бинарных файлов - классический вопрос "почему бьются файлы при upload". Вот способ решения, предложенный Димой Бородином (http://php.spb.ru): В директории, где лежит скрипт, делаем файл [.htaccess](https://htmlweb.ru/php/htaccess.php), в котором пишем:

CharsetDisable On

В файл **httpd.conf** дописать строки:

<Location />

CharsetRecodeMultipartForms Off

</Location>

Небольшие пояснения, к этому рецепту: вышеописанная проблема, когда загруженные на сервер архивы не распаковываются и картинки не отображаются, может возникать из-за того, что используется веб-сервер Russian Apache. Директива CharsetDisable отключает модуль charset-processing module, т.е. никакой перекодировки при скачивании файлов, находящихся в данной папке, происходить не будет. Директива CharsetRecodeMultipartForms выключает перекодировку данных, переданных методом POST с заголовком Content-Type: multipart/form-data. Т.е. двоичные данные, переданные с такой настройкой, будут оставлены в первоначальном виде, а все остальное наполнение сайта будет перекодировано согласно текущим настройкам сервера.

Но при этом могут возникнуть осложнения: будьте готовы к тому, что в некоторых случаях текстовые части запросов вам придется перекодировать самостоятельно. Вот что по этому поводу говорится в документации:

Используйте директиву CharsetRecodeMultipartForms, которая появилась в PL23, но при этом вам все-равно придется перекодировать вручную текстовые части запросов. Для этого можно использовать Russian Apache API, доступное в других модулях или Russian Apache Perl API, доступное из mod\_perl.

Один из примеров определения кодировки вы можете найти тут: http://tony2001.phpclub.net/detect\_charset/detect.phps

Самая свежая документация по Russian Apache находится на его официальном сайте: http://apache.lexa.ru/.

Не забывайте, что после любой смены конфигурации, Вам необходимо перезапустить Ваш веб-сервер.

Также возможна настройка параметров Apach с помощью .htaccess:

php\_value upload\_max\_filesize 50M

php\_value post\_max\_size 50M

# Дополнительные возможности

### **Загрузка нескольких файлов одновременно**

Пример формы загрузки нескольких файлов:

<form action="upload.php" method="post" enctype="multipart/form-data">

Send these files:<br>

<input name="userfile[]" type="file"><br>

<input name="userfile[]" type="file"><br>

<input type="submit" value="Send files">

</form>

И не забудьте увеличить **post\_max\_size**, если предполагается много файлов

### **Автоматическая загрузка файлов на сервер**

Не стоит забывать, что файлы на диске пользователя - конфиденциальная информация, к которой ни JavaScript, ни уж тем более PHP не имеют ни малейшего отношения. До тех пор, пока пользователь сам не выбрал файл при помощи <input type="file"> ни о какой работе с ним не может идти и речи. И не забывайте, что у данного поля ввода атрибут value защищен от записи.

### **Хранение файлов в базе данных mySQL**

Если Вы собрались хранить загружаемы файлы в базе данных, Вам необходимо помнить следующие моменты:

* Необходимо использовать поле типа BLOB
* Перед тем, как класть в базу, не забыть применить к строке [mysql\_escape\_string()](https://htmlweb.ru/php/function/mysql_escape_string.php)
* При отображении файла необходимо указывать заголовок content/type

Помните, что скрипт отображающий ваш HTML никак не связан со скриптом, который должен выводить изображение. Это должны быть два различные приложения.

Хранение картинок в базе не является хорошем стилем. Гораздо удобней хранить в базе лишь пути к файлам изображений.

### **Получение свойств изображения.**

Если перед вами возникла задача проверить тип или размеры картинки перед загрузкой файла на сервер, вам потребуется функция [getimagesize()](https://htmlweb.ru/php/function/getimagesize.php). В качестве аргумента она принимает имя файла на диске и возвращает массив, первые два элемента которого - ширина и высота соответственно, третий - тип изображения. В случае невозможности прочитать из указанного файла корректное изображение, функция возвращает ложь.

### **Загрузка файлов, имеющих русско-язычное название**

При загрузке на сервер файлов, необходимо проверять их оригинальные имена на предмет наличия "нестандартных" символов (к примеру русских букв). В случае их присутствия необходимо произвести замену. Оригинальное имя файла можно найти в переменной $\_FILES['userfile']['name']. Как перекодировать русскоязычную строку в транслит можно можно найти в [примерах PHP](https://htmlweb.ru/php/example/).

### **Отображения статуса загрузки (Progress bar)**

Необходимо учитывать, что до полной загрузки файла, PHP не может оперировать ни размером файла, ни процентом его загрузки. Только когда файл уже находится на сервере PHP, то он получает возможность обращаться к информации. Если вам все-таки крайне необходимо реализовать такую возможность, воспользуйтесь Java-аплетом.

# Права на файлы

Проблемы с правами на сервере (upload\_tmp\_dir)

В Unix-подобных операционных системах каждой папке, файлу, ссылке выставлены соответствие права доступа. Они могут выглядеть как rwx-rw-r- или же как число 754.

Доступность файла или каталога зависят от идентификатора пользователя и идентификатора группы, в которую он входит. Режим в целом описывается в терминах трех последовательностей, по три буквы в каждой:

Владелец Группа Прочие

(u) (g) (o)

rwx rwx rwx

Здесь владелец, члены группы и все прочие пользователи обладают правами чтения файла, записи в него и его выполнения. Права - любая осмысленная комбинация следующих букв:

r Право на чтение. (4)  
w Право на запись. (2)  
x Право на выполнение (поиск в каталоге). (1)

Для того, что бы загрузка файлов на сервер работала корректно, необходимо реализовать один из двух вариантов

 Установить владельцем каталога пользователя, с чьими привелегиями выполняется apache. Это можно узнать из файла httpd.conf или просмотрев список процессов на сервере. Права на каталог должны быть 700 (rwx------).

 Независимо от того, кто является владельцем каталога, установить права 777 (rwxrwxrwx).

# Пример реализации загрузки картинок на сервер.

$max\_image\_width = 380;

$max\_image\_height = 600;

$max\_image\_size = 64 \* 1024;

$valid\_types = array("gif","jpg", "png", "jpeg");

if (isset($\_FILES["userfile"])) {

if (is\_uploaded\_file($\_FILES['userfile']['tmp\_name'])) {

$filename = $\_FILES['userfile']['tmp\_name'];

$ext = substr($\_FILES['userfile']['name'],

1 + strrpos($\_FILES['userfile']['name'], "."));

if (filesize($filename) > $max\_image\_size) {

echo 'Error: File size > 64K.';

} elseif (!in\_array($ext, $valid\_types)) {

echo 'Error: Invalid file type.';

} else {

$size = GetImageSize($filename);

if (($size) && ($size[0] < $max\_image\_width)

&& ($size[1] < $max\_image\_height)) {

if (@move\_uploaded\_file($filename, "/www/htdocs/upload/")) {

echo 'File successful uploaded.';

} else {

echo 'Error: moving fie failed.';

}

} else {

echo 'Error: invalid image properties.';

}

}

} else {

echo "Error: empty file.";

}

} else {

echo '

<form enctype="multipart/form-data" method="post">

<input type="hidden" name="MAX\_FILE\_SIZE" value="64000">

Send this file: <input name="userfile" type="file">

<input type="submit" value="Send File">

</form>';

}

# Связь с базами данных MySQL

СУБД **MySQL** - одна из множества баз данных, поддерживаемых в PHP. Система MySQL распространяется бесплатно и обладает достаточной мощностью для решения реальных задач.

## **Краткое введение в MySQL**

**SQL** - это аббревиатура от слов *Structured Query Language*, что означает структурированный язык запросов. Этот язык является стандартным средством для доступа к различным базам данных.

Система MySQL представляет собой сервер, к которому могут подключаться пользователи удаленных компьютеров.

Для работы с базами данных удобно пользоваться средством, входящее в комплект Web-разработчика: [Denwer](http://denwer.ru/) [phpMyAdmin](http://localhost/). Здесь можно создать новую базу данных, создать новую таблицу в выбранной базе данных, заполнить таблицу данными, а также добавлять, удалять и редактировать данные.

В MySQL определены три базовых типа данных: числовой, дата и время и строчный. Каждая из этих категорий подразделяется на множество типов. Основные из них:

|  |  |
| --- | --- |
| Тип | Описание |
| INT | Целое число |
| TINYINT | Маленькое целое число (-127 до 128 или от 0 до 255) |
| FLOAT | Вещественное число с плавающей точкой |
| DATE | Дата. Отображается в виде ГГГГ-ММ-ДД |
| TIME | Время. Отображается в виде ЧЧ:ММ:СС |
| DATETIME | Дата и время. Отображается в виде ГГГГ-ММ-ДДЧЧ:ММ:СС |
| YEAR[(2|4)] | Год. Можно определить двух- или четырехциферный формат |
| CHAR(M) | Строка фиксированной длины М (M<=255) |
| VARCHAR(M) | Строка произвольной длины до М (M<=255) |
| TEXT | Длинные текстовые фрагменты (<=65535) |
| BLOB | Большие двоичные объекты (изображения, звуки) |

Каждый столбец после своего типа данных содержит и другие спецификаторы:

|  |  |
| --- | --- |
| Тип | Описание |
| NOT NULL | Все строки таблицы должны иметь значение в этом атрибуте. Если не указано, поле может быть пустым (**NULL**) |
| AUTO\_INCREMENT | Специальная возможность MySQL, которую можно задействовать в числовых столбцах. Если при вставке строк в таблицу оставлять такое поле пустым, MySQL автоматически генерирует уникальное значение идентификатора. Это значение будет на единицу больше максимального значения, уже существующего в столбце. В каждой таблице может быть не больше одного такого поля. Столбцы с **AUTO\_INCREMENT** должны быть проиндексированными |
| PRIMARY KEY | Столбец является первичным ключом для таблицы. Данные в этом столбце должны быть уникальными. MySQL автоматически индексирует этот столбец |
| UNSIGNED | После целочисленного типа означает, что его значение может быть либо положительным, либо нулевым |
| COMMENT | Название столбца таблицы |

**Создание новой базы данных MySQL** осуществляется при помощи SQL-команды **CREATE DATABASE**.

CREATE DATABASE IF NOT EXISTS `base`

DEFAULT CHARACTER SET cp1251 COLLATE cp1251\_bin

**Создание новой таблицы** осуществляется при помощи SQL-команды **CREATE TABLE**. Например, таблица **books** для книжного магазина будет содержать пять полей: ISBN, автор, название, цена и количество экземпляров:

CREATE TABLE books (ISBN CHAR(13) NOT NULL,

PRIMARY KEY (ISBN),

author VARCHAR(30),

title VARCHAR(60),

price FLOAT(4,2),

quantity TINYINT UNSIGNED);

Чтобы избежать сообщения об ошибке, если таблица уже есть необходимо изменить первую строчку, добавив фразу "IF NOT EXISTS":

CREATE TABLE IF NOT EXISTS books ...

Для создания **автообновляемого поля** с текущей датой типа TIMESTAMP или DATETIME используйте следующую конструкцию:

CREATE TABLE t1 (

ts TIMESTAMP DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

dt DATETIME DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

);

**Добавление** данных в эту таблицу осуществляется при помощи SQL-команды **INSERT**. Например:

INSERT INTO books ( ISBN, author, title, price, quantity )

VALUES ('5-8459-0184-7', 'Зандстра Мэт',

'Освой самостоятельно PHP4 за 24 часа', '129', '5');

Для извлечения данных из таблицы служит оператор **SELECT**. Он извлекает данные из базы, выбирая строки, которые отвечают заданному критерию поиска. Оператор **SELECT** сопровождает немалое количество опций и вариантов использования.

Символ **\*** означает, что необходимы все поля. Например:

SELECT \* FROM books;

Для получения доступа только к некоторому полю следует указать его имя в инструкции **SELECT**. Например:

SELECT author, title, price FROM books;

Чтобы получить доступ к подмножеству строк в таблице, следует указать критерий выбора, который устанавливает конструкция **WHERE**. Например, чтобы выбрать имеющиеся в наличии недорогие книги о PHP, надо составить запрос:

SELECT \* FROM books WHERE

price < 200 AND title LIKE '%PHP%' AND quantity != 0;

% Соответствует любому количеству символов, даже нулевых  
\_ Соответствует ровно одному символу

Для того, чтобы строки, извлеченные по запросу, перечислялись в определенном порядке, используется конструкция **ORDER BY**. Например:

SELECT \* FROM books ORDER BY price;

По умолчанию **порядок** **сортировки** идет по возрастанию. Изменить порядок сортировки на обратный можно с помощью ключевого слова **DESC**:

SELECT \* FROM books ORDER BY price DESC;

**Сортировать** можно и по нескольким столбцам. Вместо названий столбцов можно использовать их порядковые номера:

SELECT \* FROM books ORDER BY 4, 2, 3;

Для изменения ранее записанных в таблицу значений нужно воспользоваться командой **UPDATE**. Например, цену всех книг повысили на 10%:

UPDATE books SET price = price \* 1.1;

Конструкция **WHERE** ограничит работу **UPDATE** определенным строками. Например:

UPDATE books SET price = price \* 1.05 WHERE price <= 250;

Для удаления строк из базы данных используется оператор **DELETE**. Ненужные строки указываются при помощи конструкции **WHERE**. Например, какие-то книги проданы:

DELETE FROM books WHERE quantity = 0;

Если нужно удалить все записи

TRUNCATE TABLE table\_name

Для полного удаления таблицы используется:

DROP TABLE table\_name

## **Связь PHP с базой данных MySQL**

Поработав с phpMyAdmin над созданием базы данных, можно приступить к подключению этой базы данных к внешнему Web-интерфейсу.

Чтобы получить доступ к базе данных из Web, используя PHP, надо сделать следующие основные шаги:

* Подключение к серверу MySQL.
* Выбор базы данных.
* Выполнение запроса к базе данных:
  + **добавление**;
  + **удаление**;
  + **изменение**;
  + **поиск**;
  + **сортировка**.
* Получение результата запроса.
* Отсоединение от базы данных.

Для подключения к серверу базы данных в PHP есть функция **mysql\_connect( )**. Ее аргументы: имя компьютера, имя пользователя и пароль. Эти аргументы можно опустить. По умолчанию имя компьютера = **localhost**, тогда имя пользователя и пароль не требуется. Если PHP используется в сочетании с сервером Apache, то можно воспользоваться функцией [**mysql\_pconnect()**](https://htmlweb.ru/php/function/mysql_pconnect.php). В этом случае соединение с сервером не исчезает после завершения работы программы или вызова функции [**mysql\_close()**](https://htmlweb.ru/php/function/mysql_close.php). Функции [**mysql\_connect()**](https://htmlweb.ru/php/function/mysql_connect.php) и [**mysql\_pconnect()**](https://htmlweb.ru/php/function/mysql_pconnect.php) возвращают идентификатор подключения, если все прошло успешно. Например:

$link = mysql\_pconnect ();

if ( !$link ) die ("Невозможно подключение к MySQL");

После того, как соединение с сервером MySQL установлено, нужно выбрать базу данных. Для этого используется функция [**mysql\_select\_db()**](https://htmlweb.ru/php/function/mysql_select_db.php). Ее аргумент: имя базы данных. Функция возвращает **true**, если указанная база данных существует и доступ к ней возможен. Например:

$db = "sample";

mysql\_select\_db ( $db ) or die ("Невозможно открыть $db");

Для добавления, удаления, изменения и выбора данных нужно сконструировать и выполнить запрос SQL. Для этого в языке PHP существует функция [**mysql\_query()**](https://htmlweb.ru/php/function/mysql_query.php). Ее аргумент: строка с запросом. Функция возвращает идентификатор запроса.

#### Пример 1

<html>

<head>

<title>Добавление записи в таблицу</title>

</head>

<body>

<?php

$db = "sample";

$link = mysql\_pconnect ();

if ( !$link )

die ("Невозможно подключение к MySQL");

mysql\_select\_db ( $db ) or die ("Невозможно открыть $db");

$query = "INSERT INTO books

VALUES ('966-7393-80-1', 'Аллен Вайк',

'PHP. Справочник', '213', '4')";

mysql\_query ( $query );

mysql\_close ( $link );

?>

</body>

</html>

При каждом выполнении примера 1 в таблицу будет добавляться новая запись, содержащая одни и те же данные. Разумеется имеет смысл добавлять в базу данные, введенные пользователем.

В примере 2.1 приведена HTML-форма для добавления новых книг в базу данных.

#### Пример 2.1

<html>

<head>

<title>HTML-форма добавления новых книг</title>

</head>

<body>

<form aсtion="insert\_book.php" method="post">

<table>

<tr><td>ISBN</td><td><input name="isbn" maxlength=13 size=13></td></tr>

<tr><td>Автор</td><td><input name="author" maxlength=30 size=30></td></tr>

<tr><td>Название</td><td><input name="title" maxlength=60 size=30></td></tr>

<tr><td>Цена</td><td><input name="price" maxlength=7 size=7></td></tr>

<tr><td>Количество</td><td><input name="quantity" maxlength=3 size=3></td></tr>

<tr><td colspan=2><input type="submit" value="Ввод"></td></tr>

</table>

</form>

</body>

</html>

Результаты заполнения этой формы передаются в insert\_book.php.

#### Пример 2.2

<html>

<head>

<title>Программа добавления новых книг (файл insert\_book.php)</title>

</head>

<body>

<?php

if (!isset($\_POST['isbn']) || !isset($\_POST['author']) ||

!isset($\_POST['title']) || !isset($\_POST['price']) ||

!isset($\_POST['quantity'])){

die ("Не все данные введены.<br>

     Пожалуйста, вернитесь назад и закончите ввод");

}

$isbn = trim ( $\_POST['isbn'] );

$author = trim ( $\_POST['author'] );

$title = trim ( $\_POST['title'] ) ;

$isbn = addslashes ( $isbn );

$author = addslashes ( $author );

$title = addslashes ( $title ) ;

$db = "sample";

$link = mysql\_connect();

if ( !$link ) die ("Невозможно подключение к MySQL");

mysql\_select\_db ( $db ) or die ("Невозможно открыть $db");

$query = "INSERT INTO books VALUES ('"

.$isbn."', '".$author."', '".$title."', '"

.floatval($\_POST['price'])."', '".intval($\_POST['quantity'])."')";

$result = mysql\_query ( $query );

if ($result) echo "Книга добавлена в базу данных.";

mysql\_close ( $link );

?>

</body>

</html>

В примере 2.2 введенные строковые данные обработаны функцией [**addslashes()**](https://htmlweb.ru/php/function/addslashes.php). Эта функция добавляет обратные слеши перед одинарными кавычками ('), двойными кавычками ("), обратным слешем (\) и null-байтом. Дело в том, что по требованиям систаксиса запросов баз данных такие символы дожны заключаться в кавычки.

Для определения количества записей в результате запроса используется функция [**mysql\_num\_rows()**](https://htmlweb.ru/php/function/mysql_num_rows.php).

Все записи результата запроса можно просмотреть в цикле. Перед этим с помощью функции [**mysql\_fetch\_[]**](https://htmlweb.ru/php/function/mysql_fetch_array.php) для каждой записи получают ассоциативный массив.

В примере 3.1 приведена HTML-форма для поиска определенных книг в базе данных.

#### Пример 3.1

<html>

<head>

<title>HTML-форма поиска книг</title>

</head>

<body>

<form aсtion="search\_book.php" method="post">

Ищем по:<br>

<select name="searchtype" size=3>

<option value="author" selected>Автору

<option value="title">Названию

<option value="isbn">ISBN

</select> <br>

Что ищем:<br> <input name="searchterm"> <br>

<input type=submit value="Поиск">

</form>

</body>

</html>

Результаты заполнения этой формы передаются в search\_book.php.

#### Пример 3.2

<html>

<head>

<title>Программа поиска книг (файл search\_book.php)</title>

</head>

<body>

<?php

$searchterm = trim ( $\_POST['searchterm'] );

if (!$searchterm)

die ("Не все данные введены.<br>

    Пожалуйста, вернитесь назад и закончите ввод");

$searchterm = addslashes ($searchterm);

$link = mysql\_pconnect ();

if ( !$link ) die ("Невозможно подключение к MySQL");

$db = "sample";

mysql\_select\_db ( $db ) or die ("Невозможно открыть $db");

$query = "SELECT \* FROM books WHERE "

.$\_POST['searchtype']." like '%".$searchterm."%'";

$result = mysql\_query ( $query );

$n = mysql\_num\_rows ( $result );

for ( $i=0; $i<$n; $i++ )

{

$row = mysql\_fetch\_array($result);

echo "<p><b>".($i+1). $row['title']. "</b><br>";

echo "Автор: ".$row['author']."<br>";

echo "ISBN: ".$row['ISBN']."<br>";

echo "Цена: ".$row['price']."<br>";

echo "Количество: ".$row['quantity']."</p>";

}

if ( $n == 0 ) echo "Ничего не можем предложить. Извините";

mysql\_close ( $link );

?>

</body>

</html>

#### Альтернативный вариант

<html>

<head>

<title>Программа поиска книг (файл search\_book.php)</title>

</head>

<body>

<?

$searchterm=trim ( $\_POST['searchterm'] );

if (!$searchterm)

die ("Не все данные введены.<br>Пожалуйста, вернитесь назад и закончите ввод");

$searchterm = addslashes ($searchterm);

mysql\_connect() or die ("Невозможно подключение к MySQL");

mysql\_select\_db ( "sample" ) or die ("Невозможно открыть БД");

$result = mysql\_query ( "SELECT \* FROM books WHERE ".$\_POST['searchtype']." like '%".$searchterm."%'" );

$i=1;

while($row = mysql\_fetch\_array($result))

{

echo "<p><b>".($i++) . $row['title']."</b><br>";

echo "Автор: ".$row['author']."<br>";

echo "ISBN: ".$row['ISBN']."<br>";

echo "Цена: ".$row['price']."<br>";

echo "Количество: ".$row['quantity']."</p>";

}

if ( $i == 1 ) echo "Ничего не можем предложить. Извините";

mysql\_close( );

?>

</body>

</html>

Итак, как работает архитектура Web-баз данных:

1. Web-браузер пользователя выдает HTTP-запрос определенной Web-страницы. Например, пользователь, используя HTML-форму, ищет все книги о PHP. Страница обработки формы называется search\_book.php.
2. Web-сервер принимает запрос на search\_book.php, извлекает этот файл и передает на обработку механизму PHP.
3. PHP выполняет соединение с MySQL-сервером и отправляет запрос.
4. Сервер принимает запрос к базе данных, обрабатывает его и отправляет результат (список книг) обратно механизму PHP.
5. Механизм PHP завершает выполнение сценария, форматирует результат запроса в HTML. После этого результат в виде HTML возвращается Web-серверу.
6. Web-сервер пересылает HTML в браузер, и пользователь имеет возможность просмотреть запрошенный список книг.

### **Использование механизма транзакций**

Использование механизма транзакция на примере как передать деньги от одного человека другому

if(

mysql\_query ("BEGIN") &&

mysql\_query ("UPDATE money SET amt = amt - 6 WHERE name = 'Eve'") &&

mysql\_query ("UPDATE money SET amt = amt + 6 WHERE name = 'Ida'") &&

mysql\_query ("COMMIT")

){

echo "Успешно";

}else{

mysql\_query ("ROLLBACK");

echo "Не успешно";

}

### **SELECT … FOR UPDATE**

Если Вы запускаете несколько процессов, которые делают select запрос к одной и той же таблице, то они могут выбрать одну и ту же запись одновременно.

Чтобы избежать вышеупомянутой ситуации необходимо выполнить не просто SELECT запрос, а его расширенную версию, о которой многие и не подозревают: SELECT … FOR UPDATE.

Таким образом, при выполнении данного запроса, все затронутые записи в базе данных будут заблокированы до завершения сеанса работы с БД или до момента обновления данных записей. Другой скрипт не сможет выбрать заблокированные записи до тех пор, пока не наступит одно из упомянутых условий.

Однако не всё так просто. Вам нужно выполнить ещё несколько условий. Во-первых, ваша таблица должна быть создана на основе архитектуры InnoDB. В противном случае блокировка просто не будет срабатывать. Во-вторых, перед выполнением выборки необходимо отключить авто-коммит запроса. Т.е. другими словами автоматическое выполнение запроса. После того как вы укажите UPDATE запрос, необходимо будет ещё раз обратиться к базе и закоммитить изменения с помощью команды COMMIT:

mysql\_query("SET autocommit = 0");

$result = mysql\_query("SELECT \* FROM table WHERE locked = 0 LIMIT 1 FOR UPDATE");

$row = mysql\_fetch\_assoc($result);

mysql\_query("UPDATE table SET locked = 1 WHERE id = 1;");

mysql\_query("COMMIT;");

Представьте что у нас есть 2 записи в таблице. Мы запускаем два параллельных скрипта с данным содержанием. В результате, первый процесс, который сможет достучаться до базы сделает выборку записи и сразу же заменит значение в поле locked на 1. В результате второй скрипт проигнорирует первую запись. В данном случае, даже если скрипты достучались бы до БД одновременно, второй скрипт всё равно не смог бы выбрать первую запись, т.к. при выборке первым процессом она будет заблокирована.

# Работа с датой и временем в MySQL+PHP

Ниже представлен пример, в котором используются функции даты. Приведенный запрос выбирает все записи с величиной date\_col в течение последних 30 дней:

SELECT something FROM tbl\_name WHERE TO\_DAYS(NOW()) - TO\_DAYS(date\_col) <= 30;

**DAYOFWEEK**(date) - Возвращает индекс дня недели для аргумента date (1 = воскресенье, 2 = понедельник, ... 7 = суббота). Эти индексные величины соответствуют стандарту ODBC.

SELECT DAYOFWEEK('1998-02-03'); // -> 3

**WEEKDAY**(date) - Возвращает индекс дня недели для аргумента date (0 =понедельник, 1 = вторник, ... 6 = воскресенье):

SELECT WEEKDAY('1998-02-03 22:23:00'); // -> 1

SELECT WEEKDAY('1997-11-05'); // -> 2

**DAYOFMONTH**(date) - Возвращает порядковый номер дня месяца для аргумента date в диапазоне от 1 до 31:

SELECT DAYOFMONTH('1998-02-03'); // -> 3

**DAYOFYEAR**(date) - Возвращает порядковый номер дня года для аргумента date в диапазоне от 1 до 366:

SELECT DAYOFYEAR('1998-02-03'); // -> 34

**MONTH**(date) - Возвращает порядковый номер месяца в году для аргумента date в диапазоне от 1 до 12:

SELECT MONTH('1998-02-03'); // -> 2

**DAYNAME**(date) - Возвращает название дня недели для аргумента date:

SELECT DAYNAME("1998-02-05"); // -> 'Thursday'

**MONTHNAME**(date) - Возвращает название месяца для аргумента date:

SELECT MONTHNAME("1998-02-05"); // -> 'February'

**QUARTER**(date) - Возвращает номер квартала года для аргумента date в диапазоне от 1 до 4:

SELECT QUARTER('98-04-01'); // -> 2

**WEEK**(date), **WEEK**(date,first) - При наличии одного аргумента возвращает порядковый номер недели в году для date в диапазоне от 0 до 53 (да, возможно начало 53-й недели) для регионов, где воскресенье считается первым днем недели. Форма WEEK() с двумя аргументами позволяет уточнить, с какого дня начинается неделя - с воскресенья или с понедельника. Результат будет в пределах 0-53 или 1-52.

Вот как работает второй аргумент:  
0 - Неделя начинается с воскресенья; возвращаемое значение - в промежутке 0-53  
1 - Неделя начинается с понедельника; возвращаемое значение - в промежутке 0-53  
2 - Неделя начинается с воскресенья; возвращаемое значение - в промежутке 1-53  
3 - Неделя начинается с понедельника; возвращаемое значение - в промежутке 1-53 (ISO 8601)

SELECT WEEK('1998-02-20'); // -> 7

SELECT WEEK('1998-02-20',0); // -> 7

SELECT WEEK('1998-02-20',1); // -> 8

SELECT WEEK('1998-12-31',1); // -> 53

Примечание: в версии 4.0 функция WEEK(#,0) была изменена с целью соответствия календарю США.  
Заметьте, если неделя является последней неделей прошлого года, MySQL вернет 0 если вы не указали 2 или 3 как опциональный аргумент:

SELECT YEAR('2000-01-01'), WEEK('2000-01-01',0); // -> 2000, 0

SELECT WEEK('2000-01-01',2); // -> 52

Можно считать, что MySQL должен вернуть 52, так как данная дата и является 52-ой неделей года 1999. Мы решили возвращать 0, так как мы хотим, чтобы функция давала "номер недели в указанном году". Это делает функцию WEEK() более надежной при использовании совместно с другими функциями, которые вычисляют части дат.

Если вам все же важно уточнить корректную неделю в году, тогда вы можете использовать 2 или 3 как опциональный аргумент или использовать YEARWEEK()

SELECT YEARWEEK('2000-01-01'); // -> 199952

SELECT MID(YEARWEEK('2000-01-01'),5,2); // -> 52

**YEAR**(date) - Возвращает год для аргумента date в диапазоне от 1000 до 9999:

SELECT YEAR('98-02-03'); // -> 1998

**YEARWEEK**(date), **YEARWEEK**(date,first) - Возвращает год и неделю для аргумента date. Второй аргумент в данной функции работает подобно второму аргументу в функции WEEK(). Следует учитывать, что год может отличаться от указанного в аргументе date для первой и последней недель года:

SELECT YEARWEEK('1987-01-01'); // -> 198653

Обратите внимание, что номер недели отличается от того, который возвращает функция WEEK() (0), будучи вызванной с опциональным аргументом 0 или 1. Это потому, что WEEK() возвращает номер недели именно в указанном году. 

**HOUR**(time)- Возвращает час для аргумента time в диапазоне от 0 до 23:

SELECT HOUR('10:05:03'); //-> 10

**MINUTE**(time) - Возвращает количество минут для аргумента time в диапазоне от 0 до 59:

SELECT MINUTE('98-02-03 10:05:03'); // -> 5

**SECOND**(time) - Возвращает количество секунд для аргумента time в диапазоне от 0 до 59:

SELECT SECOND('10:05:03'); // -> 3

**PERIOD\_ADD**(P,N) - Добавляет N месяцев к периоду P (в формате YYMM или YYYYMM). Возвращает величину в формате YYYYMM. Следует учитывать, что аргумент периода P не является значением даты:

SELECT PERIOD\_ADD(9801,2); // -> 199803

**PERIOD\_DIFF**(P1,P2) - Возвращает количество месяцев между периодами P1 и P2. P1 и P2 должны быть в формате YYMM или YYYYMM. Следует учитывать, что аргументы периода P1 и P2 не являются значениями даты:

SELECT PERIOD\_DIFF(9802,199703); // -> 11

**DATE\_ADD**(date,INTERVAL expr type), **DATE\_SUB**(date,INTERVAL expr type), **ADDDATE**(date,INTERVAL expr type), **SUBDATE**(date,INTERVAL expr type) - Данные функции производят арифметические действия над датами. Обе являются нововведением версии MySQL 3.22. Функции ADDDATE() и SUBDATE() - синонимы для DATE\_ADD() и DATE\_SUB(). В версии MySQL 3.23 вместо функций DATE\_ADD() и DATE\_SUB() можно использовать операторы + и -, если выражение с правой стороны представляет собой столбец типа DATE или DATETIME (см. пример ниже). Аргумент date является величиной типа DATETIME или DATE, задающей начальную дату.

Выражение expr задает величину интервала, который следует добавить к начальной дате или вычесть из начальной даты. Выражение expr представляет собой строку, которая может начинаться с - для отрицательных значений интервалов. Ключевое слово type показывает, каким образом необходимо интерпретировать данное выражение. Вспомогательная функция EXTRACT(type FROM date) возвращает интервал указанного типа (type) из значения даты. В следующей таблице показана взаимосвязь аргументов type и expr:

SECOND - SECONDS  
MINUTE - MINUTES  
HOUR - HOURS  
DAY - DAYS  
MONTH - MONTHS  
YEAR - YEARS  
MINUTE\_SECOND - "MINUTES:SECONDS"  
HOUR\_MINUTE - "HOURS:MINUTES"  
DAY\_HOUR - "DAYS HOURS"  
YEAR\_MONTH - "YEARS-MONTHS"  
HOUR\_SECOND - "HOURS:MINUTES:SECONDS"  
DAY\_MINUTE - "DAYS HOURS:MINUTES"  
DAY\_SECOND - "DAYS HOURS:MINUTES:SECONDS"

В MySQL формат выражения expr допускает любые разделительные знаки. Разделители, представленные в данной таблице, приведены в качестве примеров. Если аргумент date является величиной типа DATE и предполагаемые вычисления включают в себя только части YEAR, MONTH, и DAY (т.е. не содержат временной части TIME), то результат представляется величиной типа DATE. В других случаях результат представляет собой величину DATETIME:

SELECT "1997-12-31 23:59:59" + INTERVAL 1 SECOND; // -> 1998-01-01 00:00:00

SELECT INTERVAL 1 DAY + "1997-12-31"; // -> 1998-01-01

SELECT "1998-01-01" - INTERVAL 1 SECOND; // -> 1997-12-31 23:59:59

SELECT DATE\_ADD("1997-12-31 23:59:59", INTERVAL 1 SECOND); // -> 1998-01-01 00:00:00

SELECT DATE\_ADD("1997-12-31 23:59:59", INTERVAL 1 DAY); // -> 1998-01-01 23:59:59

SELECT DATE\_ADD("1997-12-31 23:59:59", INTERVAL "1:1" MINUTE\_SECOND); // -> 1998-01-01 00:01:00

SELECT DATE\_SUB("1998-01-01 00:00:00", INTERVAL "1 1:1:1" DAY\_SECOND); // -> 1997-12-30 22:58:59

SELECT DATE\_ADD("1998-01-01 00:00:00", INTERVAL "-1 10" DAY\_HOUR); // -> 1997-12-30 14:00:00

SELECT DATE\_SUB("1998-01-02", INTERVAL 31 DAY); // -> 1997-12-02

Если указанный интервал слишком короткий (т.е. не включает все части интервала, ожидаемые при заданном ключевом слове type), то MySQL предполагает, что опущены крайние слева части интервала. Например, если указан аргумент type в виде DAY\_SECOND, то ожидаемое выражение expr должно иметь следующие части: дни, часы, минуты и секунды. Если в этом случае указать значение интервала в виде "1:10", то MySQL предполагает, что опущены дни и часы, а данная величина включает только минуты и секунды. Другими словами, сочетание "1:10" DAY\_SECOND интерпретируется как эквивалент "1:10" MINUTE\_SECOND. Аналогичным образом в MySQL интерпретируются и значения TIME - скорее как представляющие прошедшее время, чем как время дня. Следует учитывать, что при операциях сложения или вычитания с участием величины DATE и выражения, содержащего временную часть, данная величина DATE будет автоматически конвертироваться в величину типа DATETIME:

SELECT DATE\_ADD("1999-01-01", INTERVAL 1 DAY); // -> 1999-01-02

SELECT DATE\_ADD("1999-01-01", INTERVAL 1 HOUR); // -> 1999-01-01 01:00:00

При использовании некорректных значений дат результат будет равен NULL. Если при суммировании MONTH, YEAR\_MONTH или YEAR номер дня в результирующей дате превышает максимальное количество дней в новом месяце, то номер дня результирующей даты принимается равным последнему дню нового месяца:

SELECT DATE\_ADD('1998-01-30', INTERVAL 1 MONTH); // -> 1998-02-28

Из предыдущего примера видно, что слово INTERVAL и ключевое слово type не являются регистро-зависимыми.

**EXTRACT**(type FROM date) - Типы интервалов для функции EXTRACT() используются те же, что и для функций DATE\_ADD() или DATE\_SUB(), но EXTRACT() производит скорее извлечение части из значения даты, чем выполнение арифметических действий.

SELECT EXTRACT(YEAR FROM "1999-07-02"); // -> 1999

SELECT EXTRACT(YEAR\_MONTH FROM "1999-07-02 01:02:03"); // -> 199907

SELECT EXTRACT(DAY\_MINUTE FROM "1999-07-02 01:02:03"); // -> 20102

**TO\_DAYS**(date) - функция возвращает номер дня для даты, указанной в аргументе date, (количество дней, прошедших с года 0):

SELECT TO\_DAYS(950501); // -> 728779

SELECT TO\_DAYS('1997-10-07'); // -> 729669

Функция TO\_DAYS() не предназначена для использования с величинами, предшествующими введению григорианского календаря (1582), поскольку не учитывает дни, утерянные при изменении календаря.

**FROM\_DAYS**(N) - Возвращает величину DATE для заданного номера дня N:

SELECT FROM\_DAYS(729669); // -> '1997-10-07'

Функция FROM\_DAYS() не предназначена для использования с величинами, предшествующими введению григорианского календаря (1582), поскольку она не учитывает дни, утерянные при изменении календаря.

**DATE\_FORMAT**(date,format) - Форматирует величину date в соответствии со строкой format. В строке format могут использоваться следующие определители: %M Название месяца (январь...декабрь)  
%W Название дня недели (воскресенье...суббота)  
%D День месяца с английским суффиксом (0st, 1st, 2nd, 3rd и т.д.)  
%Y Год, число, 4 разряда  
%y Год, число, 2 разряда  
%X Год для недели, где воскресенье считается первым днем недели, число, 4 разряда, используется с '%V'  
%x Год для недели, где воскресенье считается первым днем недели, число, 4 разряда, используется с '%v'  
%a Сокращенное наименование дня недели (Вс...Сб)  
%d День месяца, число (00..31)  
%e День месяца, число (0..31)  
%m Месяц, число (00..12)  
%c Месяц, число (0..12)  
%b Сокращенное наименование месяца (Янв...Дек)  
%j День года (001..366)  
%H Час (00..23)  
%k Час (0..23)  
%h Час (01..12)  
%I Час (01..12)  
%l Час (1..12)  
%i Минуты, число (00..59)  
%r Время, 12-часовой формат (hh:mm:ss [AP]M)  
%T Время, 24-часовой формат (hh:mm:ss)  
%S Секунды (00..59)  
%s Секунды (00..59)  
%p AM или PM  
%w День недели (0=воскресенье..6=суббота)  
%U Неделя (00..53), где воскресенье считается первым днем недели  
%u Неделя (00..53), где понедельник считается первым днем недели  
%V Неделя (01..53), где воскресенье считается первым днем недели. Используется с '%X'  
%v Неделя (01..53), где понедельник считается первым днем недели. Используется с '%x'  
%% Литерал '%'.

Все другие символы просто копируются в результирующее выражение без интерпретации:

SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%W %M %Y'); // -> 'Saturday October 1997'

SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%H:%i:%s'); // -> '22:23:00'

SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%D %y %a %d %m %b %j'); // -> '4th 97 Sat 04 10 Oct 277'

SELECT DATE\_FORMAT('1997-10-04 22:23:00', '%H %k %I %r %T %S %w'); // -> '22 22 10 10:23:00 PM 22:23:00 00 6'

SELECT DATE\_FORMAT('1999-01-01', '%X %V'); // -> '1998 52'

SELECT DATE\_FORMAT(sale.time, '%Y-%m-%d') as dat // 2014-03-11

В MySQL 3.23 символ '%' должен предшествовать символам определителя формата. В более ранних версиях MySQL символ '%' необязателен.

Причина того, что промежутки для месяца и дня начинаются с нуля заключается в том, что MySQL позволяет использовать неполные даты, такие как '2004-00-00', начиная с MySQL 3.23.

**TIME\_FORMAT**(time,format) - Данная функция используется аналогично описанной выше функции DATE\_FORMAT(), но строка format может содержать только те определители формата, которые относятся к часам, минутам и секундам. При указании других определителей будет выдана величина NULL или 0.

**CURDATE()**, **CURRENT\_DATE** - Возвращает сегодняшнюю дату как величину в формате YYYY-MM-DD или YYYYMMDD, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

SELECT CURDATE(); // -> '1997-12-15'

SELECT CURDATE() + 0; //-> 19971215

**CURTIME**(), **CURRENT\_TIME** - Возвращает текущее время как величину в формате HH:MM:SS или HHMMS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

SELECT CURTIME(); // -> '23:50:26'

SELECT CURTIME() + 0; // -> 235026

**NOW**(), **SYSDATE**(), **CURRENT\_TIMESTAMP** - Возвращает текущую дату и время как величину в формате YYYY-MM-DD HH:MM:SS или YYYYMMDDHHMMSS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

SELECT NOW(); // -> '1997-12-15 23:50:26'

SELECT NOW() + 0; // -> 19971215235026

Заметьте, что NOW() вычисляется только единожды для каждого запроса, а именно - в начале его выполнения. Это позволяет быть уверенным в том, что множественные ссылки на NOW() в рамках одного запроса дадут одно и то же значение.

**UNIX\_TIMESTAMP**(), **UNIX\_TIMESTAMP**(date) - При вызове данной функции без аргумента она возвращает временную метку UNIX\_TIMESTAMP (секунды с 1970-01-01 00:00:00 GMT) как беззнаковое целое число. Если функция UNIX\_TIMESTAMP() вызывается с аргументом date, она возвращает величину аргумента как количество секунд с 1970-01-01 00:00:00 GMT. Аргумент date может представлять собой строку типа DATE, строку DATETIME, величину типа TIMESTAMP или число в формате YYMMDD или YYYYMMDD местного времени:

SELECT UNIX\_TIMESTAMP(); // -> 882226357

SELECT UNIX\_TIMESTAMP('1997-10-04 22:23:00'); // -> 875996580

При использовании функции UNIX\_TIMESTAMP в столбце TIMESTAMP эта функция будет возвращать величину внутренней временной метки непосредственно, без подразумеваемого преобразования строки во временную метку (``string-to-unix-timestamp'' ). Если заданная дата выходит за пределы допустимого диапазона, то функция UNIX\_TIMESTAMP() возвратит 0, но следует учитывать, что выполняется только базовая проверка (год 1970-2037, месяц 01-12, день 01-31). Если необходимо выполнить вычитание столбцов UNIX\_TIMESTAMP(), результат можно преобразовать к целым числам со знаком. See Раздел 6.3.5, «Функции приведения типов».

**FROM\_UNIXTIME**(unix\_timestamp) - Возвращает представление аргумента unix\_timestamp как величину в формате YYYY-MM-DD HH:MM:SS или YYYYMMDDHHMMSS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

SELECT FROM\_UNIXTIME(875996580); // -> '1997-10-04 22:23:00'

SELECT FROM\_UNIXTIME(875996580) + 0; // -> 19971004222300

**FROM\_UNIXTIME**(unix\_timestamp,format) - Возвращает строковое представление аргумента unix\_timestamp, отформатированное в соответствии со строкой format. Строка format может содержать те же определители, которые перечислены в описании для функции DATE\_FORMAT():

SELECT FROM\_UNIXTIME(UNIX\_TIMESTAMP(), '%Y %D %M %h:%i:%s %x'); // -> '1997 23rd December 03:43:30 1997'

**SEC\_TO\_TIME**(seconds) - Возвращает аргумент seconds, преобразованный в часы, минуты и секунды, как величину в формате HH:MM:SS или HHMMSS, в зависимости от того, в каком контексте используется функция - в строковом или числовом:

SELECT SEC\_TO\_TIME(2378); // -> '00:39:38'

SELECT SEC\_TO\_TIME(2378) + 0; // -> 3938

**TIME\_TO\_SEC**(time) - Возвращает аргумент time, преобразованный в секунды:

SELECT TIME\_TO\_SEC('22:23:00'); // -> 80580

SELECT TIME\_TO\_SEC('00:39:38'); // -> 2378

# Вопросы и ответы по MySQL и MySql+PHP

### **Как узнать следующее значение AUTO\_INCREMENT из таблицы в MySQL?**

$res=mysql\_query("SHOW TABLE STATUS LIKE 'tbl'");

if($row=mysql\_fetch\_assoc($res))

    return intval($row['Auto\_increment']);

else

    return 1;

Обратите внимание, в интернете часто встречается ошибочный способ определения номера(id) записи, которая будет добавлена:

SELECT max(id)+1 as q FROM tbl

**Это неверно!** Т.к. если последняя запись была удалена, то id значение добавленной записи не будет соответствовать MAX+1!

Кроме того, при большом количестве одновременных запросоов в базу от разных клиентов id может стать больше, чем только что полученный функцией SHOW TABLE STATUS LIKE 'tbl' т.к. мог проскочить INSERT. Например id: 1,2,3,4 Удалил 3. Следующее добавляемое 5, а не 3.

### **Как установить начальное значение для поля с auto increment в MySQL?**

Для сброса автоинкреметного поля в начальное значение уже существующей таблицы с данными выполните команду:

ALTER TABLE tab SET AUTO\_INCREMENT=1;

После этого, при добавлении, следующая запись будет имень номер самого большого id + 1

Если нужно при создании таблицы:

CREATE TABLE tab (id INT UNSIGNED AUTO\_INCREMENT,....PRIMARY KEY(id)) AUTO\_INCREMENT=123;

### **Как использовать свободные id записей?**

После удаления записи в таблице базы данных, автоинкрементное id освобождается и уже никогда не будет использовано. Это сделано специально для обеспечения уникальности операций. Например, при решении вопроса синхронизации в распределённых базах данных.

Если Вы используете id для нумерации, когторую показываете пользоветлю, то "дырки" в нумерации - не очень красиво.

Есть два варианта:

**первый:**

После каждого удаления сбрасывать AUTO\_INCREMENT в 1 с помощью команды:

alter table `tbl` auto\_increment=1

Пример:

mysql\_query("DELETE FROM `tbl` WHERE id='".$id."' LIMIT 1");

if(mysql\_affected\_rows()>0)

    mysql\_query("alter table `tbl` auto\_increment=1;");

**второй:**

После добавления записи для получения её id вместо стандартной функции mysql\_insert\_id() использовать следующую:

/\* получить id только что добавленной записи и

по возможности переместить её в свободный id \*/

function GetInsertId($tbl){

global $insert\_id;

$id0=mysql\_insert\_id(); if($id0<1) return $id0;

$id\_from=(isset($insert\_id[$tbl])?$insert\_id[$tbl]:1);

for($id=$id\_from; $id<$id0; $id++){

$result = mysql\_query('SELECT id from '.$tbl.' WHERE id='.$id.' LIMIT 1');

if(mysql\_num\_rows($result)==0){

    mysql\_query('UPDATE IGNORE '.$tbl.' SET id='.$id.' WHERE id='.$id0.' LIMIT 1');

    if(mysql\_affected\_rows()<1)continue; // уже заняли

$insert\_id[$tbl]=$id+1;

    return $id; // id - свободен

}

}

$insert\_id[$tbl]=$id;

return $id0;

}

Она ищет "дырку" в нумерации (свободные id номера) и пытается занять первый свободный, если успешно, то последняя добавленная запись будет иметь id "дырки" и возвращен номер. На случай частого использования данные кешируются в переменную памяти.

### **Вывод из связанных баз**

Вывести все поля из KATEG и поле name из GR. Осторожно с одинаковыми именами полей. Они берутся из второй базы.

$query = mysql\_query("SELECT \*,gr.name as name FROM kateg,gr WHERE kateg.gr=gr.id ORDER by name");

### **Проверка наличия записи по условию**

Если есть хотя бы одна запись удовлетворяющая условию:

if(mysql\_num\_rows(mysql\_query('SELECT \* from kateg WHERE gr='.$del.' LIMIT 1'))>0)

die('Не удаляю, т.к.....!');

### **Удаление в связанных базах**

Иногда хочется написать запрос типа такого:

mysql\_query("DELETE FROM price,supplier WHERE price.id=supplier.tovar and price.gr=".$del);

Но он работать не будет, вот эквивалентный работающий код:

$result=mysql\_query('SELECT \* from price WHERE price.gr='.$del);

while ($row = mysql\_fetch\_assoc($result)){ // удаляю во второй базе множество записей

mysql\_query('DELETE FROM supplier WHERE supplier.tovar='.$row['id']);

}

mysql\_query('DELETE FROM price WHERE price.gr='.$del); // удаляю в первой базе

### **Количество записей**

Подсчитать количество записей, удовлетворяющих условию

$q=mysql\_query('SELECT count(\*) as counter from price WHERE gr='.$row['id']);

$data=mysql\_fetch\_assoc($q);

$counter=($data?intval($data['counter']):0);

### **Получение ключа только что добавленной записи**

Для получение ключа только что добавленной записи используется функция

[mysql\_insert\_id()](https://htmlweb.ru/php/function/mysql_insert_id.php)

mysql\_query("INSERT INTO mytable (product) values ('kossu')");

echo "id последней добавленной записи ".mysql\_insert\_id();

### **Перенумеровать поле id UNIQUE AUTO\_INCREMENT в mysql**

В 99 процентах случаях это делать не нужно. Уникальный ключ для того и создан, чтобы при удалении записи оставалась "дырка" в нумерации. Но, иногда приходится. Например, я создавал сайт и набирал справочник для клиента на локальной машине, естественно, было много правок. Хотелось при "выкладывании вживую" чтобы все выглядело красиво. Вот для такого случая и сделан следующий скриптик:

$q=sql('SELECT count(\*) as counter from gr');

$data=mysql\_fetch\_assoc($q); $counter=($data?intval($data['counter']):0);

for($id=1; $id<=$counter; $id++){ // перенумеровываю все id

$result = sql('SELECT \* from gr WHERE id='.$id.' LIMIT 1');

if(mysql\_num\_rows($result)==0) {// id - свободен

$result = sql('SELECT MAX(id) as id from gr');

if (($row = mysql\_fetch\_assoc($result)) && ($id<$row['id'])){

sql('UPDATE gr SET id=LAST\_INSERT\_ID('.$id.') WHERE id='.$row['id'].' LIMIT 1');

// то же сделать во всех связанных базах!

sql('UPDATE price SET gr='.$id.' WHERE gr='.$row['id']);

sql('UPDATE kateg SET gr='.$id.' WHERE gr='.$row['id']);

}

else break;

}

}

### **Преобразование IP адреса для записи в базу MySql**

function int2ip($i)

{ // INET\_NTOA(3520061480)

$d[0]=(int)($i/256/256/256);

$d[1]=(int)(($i-$d[0]\*256\*256\*256)/256/256);

$d[2]=(int)(($i-$d[0]\*256\*256\*256-$d[1]\*256\*256)/256);

$d[3]=$i-$d[0]\*256\*256\*256-$d[1]\*256\*256-$d[2]\*256;

return "$d[0].$d[1].$d[2].$d[3]";

}

function ip2int($ip)

{ // INET\_ATON("209.207.224.40")

$a=explode(".",$ip);

return $a[0]\*256\*256\*256+$a[1]\*256\*256+$a[2]\*256+$a[3];

}

### **Как получить список id записей в строку по условию из MySql**

Чтобы выбрать список записей в виде строки id кодов воспользуется функцией MySql **GROUP\_CONCAT**

SELECT GROUP\_CONCAT(DISTINCT id SEPARATOR ',') AS plan\_id

FROM table

WHERE `Group` = 1

### **Найти в связанных таблицах потерянные записи (записи не имеющие значение в справочнике)**

SELECT \* FROM tovar

LEFT JOIN category\_link

ON tovar.id=category\_link.tovar

WHERE category\_link.tovar IS NULL

Если нужно найти и удалить потерянные записи:

SELECT swdist\_head.dist\_id

FROM swdist\_head LEFT JOIN swdist\_item

ON swdist\_head.dist\_id = swdist\_item.dist\_id

WHERE swdist\_item.dist\_id IS NULL;

Соответствующее предложение DELETE будет таким:

DELETE swdist\_head

FROM swdist\_head LEFT JOIN swdist\_item

ON swdist\_head.dist\_id = swdist\_item.dist\_id

WHERE swdist\_item.dist\_id IS NULL;

Запрос, определяющий потомков без родителей, выглядит так:

SELECT swdist\_item.dist\_id

FROM swdist\_item LEFT JOIN swdist\_head

ON swdist\_item.dist\_id = swdist\_head.dist\_id

WHERE swdist\_head.dist\_id IS NULL;

Соответствующее предложение DELETE удаляет их:

DELETE swdist\_item

FROM swdist\_item LEFT JOIN swdist\_head

ON swdist\_item.dist\_id = swdist\_head.dist\_id

WHERE swdist\_head.dist\_id IS NULL;

### **Сколько сообщений было отправлено в течение каждого часа дня?**

SELECT HOUR(t) AS hour, COUNT(HOUR(t)) AS count FROM mail GROUP BY hour;

### **Преобразование TIMESTAMP в DATETIME и обратно**

SELECT DATE\_FORMAT(t\_create,'%Y-%m-%d %T') FROM tbl; // CCYYMMDDhhmmss -> CCYY-MM-DD hh:mm:ss

SELECT DATE\_FORMAT(dt,'%Y%m%d%H%i%s'), dt+0 FROM tbl; // CCYY-MM-DD hh:mm:ss -> CCYYMMDDhhmmss

### **MyISAM или InnoDB ?**

MyISAM быстрее InnoDB - сомнительно. Они предназначены для разных видов приложений:

* **MyISAM** - таблички, предназначеные для приложений с очень большим перевесом в сторону чтения (например, они будут хорошо себя вести при 1000 чтений на 1 запись). При этом следует понимать, что они не поддерживают одновременную запись. Т.е. на время вставки записи в таблицу все остальные запросы висят и ждут!
* **InnoDB** - поддерживает одновременную запись при очень малой деградации производительности чтений по сравнению с MyISAM. Собственно, в условиях присутствия запросов на запись, параллельные чтения из той же таблички InnoDB возможны (в отличие от MyISAM), а, соответственно, производительность таких таблиц будет в среднем выше.

### **Mysql как игнорировать ошибки при загрузке дампа?**

Используйте ключ "-f" - он позволяет продолжать даже если в процессе произошла ошибка. Например:

mysql.exe --default-character-set=cp1251 -f -u root -p --database=base1 < !.!

### **Долгий MySQL запрос и на что его заменить?**

Данный запрос на таблице из 100тыс.записей выполняется более 10 секунд

SELECT user\_id FROM `user\_spy` WHERE key in (

SELECT key FROM `user\_spy` WHERE `user\_id` = 32602)

AND user\_id != 32602

Если его разбить на два, то каждый выполнится меньше чем за секунду:

SELECT group\_concat(distinct key) FROM `user\_spy` WHERE `user\_id` = 32602

+

SELECT user\_id FROM user\_spy WHERE key in ('xxxx','yyy') AND user\_id != 32602

## **Cookie в PHP**

[**setcookie()**](https://htmlweb.ru/php/function/setcookie.php) определяет куку для отправки вместе с остальной header-информацией. Куки обязаны быть отправлены *до* любых других шапок/headers (это ограничение кук, а не РНР). Это требует, чтобы вы помещали вызовы этой функции перед первым выводом на страницу, т.е. до тэга **<html>**.

int **setcookie** (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

Все аргументы, кроме ***name***, являются необязательными. Если имеется только аргумент name, кука с этим именем будет удалена с удалённого клиента. Вы можете также заместить любой аргумент пустой строкой (*""*), чтобы пропустить этот аргумент. Аргументы ***expire*** и ***secure*** это целые числа/integer и они не могут быть пропущены с помощью пустой строки. В них используйте нуль (*0*). Аргумент ***expire*** это обычное Unix time integer, возвращаемое функциями [**time()**](https://htmlweb.ru/php/function/time.php) или [**mktime()**](https://htmlweb.ru/php/function/mktime.php). Аргумент ***secure*** указывает, что данная кука должна передаваться только через секретное HTTPS-соединение.

После того как куки установлены, доступ к ним может быть получен при загрузке следующей страницы через массив **$\_COOKIE** (который вызывается **$HTTP\_COOKIE\_VARS** в версиях PHP до 4.1.0).

Обычные ловушки:

* Куки будут невидимы до тех пор, пока не будет загружена следующая страница.
* Куки обязаны быть удалены с теми же параметрами, с которыми были установлены.

В PHP 3 множественные вызовы **setcookie()** в том же скрипте могут быть выполнены в реверсном порядке. Если вы пытаетесь удалить одну куку до вставки другой, вы должны сделать вставку до удаления. В PHP 4 множественные вызовы **setcookie()** выполняются в порядке вызова.

Далее идут примеры отправки кук:

##### Пример 1. Отправка кук функцией setcookie()

setcookie ("TestCookie", $value);

setcookie ("TestCookie", $value,time()+3600); // период действия - 1 час

setcookie ("TestCookie", $value,time()+3600,

"/~rasmus/", ".utoronto.ca", 1);

При удалении куки вы должны убедиться, что дата окончания действия прошла, чтобы переключить механизм в вашем браузере. Далее идут примеры удаления куки, созданной в предыдущем примере:

##### Пример 2. Удаление куки с помощью setcookie()

// установить дату окончания действия на один час назад

setcookie ("TestCookie", "", time() - 3600);

setcookie ("TestCookie", "", time() - 3600,

"/~rasmus/", ".utoronto.ca", 1);

##### Пример 3. Удаление всех кук

foreach($\_COOKIE as $ind=>$val)

@setcookie($ind,'',time()-999, "/", ".".$\_SERVER['SERVER\_NAME'] );

##### Пример 4.

Обратите внимание, что часть value куки будет автоматически urlencoded при отправке куки, и, когда она получена, она автоматически декодируется и присваивается переменной с тем же именем, что и имя куки. Для просмотра содержимого нашей тестовой куки в скрипте просто используйте один из следующих примеров:

echo $TestCookie;

echo $\_COOKIE["TestCookie"];

##### Пример 5.

Вы можете также установить куки массива, используя нотацию в имени куки. Это даёт эффект установки стольких кук, сколько элементов в этом массиве, но, когда кука получается скриптом, значения помещаются в массив с именем куки:

setcookie ("cookie[three]", "cookiethree");

setcookie ("cookie[two]", "cookietwo");

setcookie ("cookie[one]", "cookieone");

foreach($\_COOKIE as $name=>$val) {

echo "$name = $value<br>\n";

}

##### Пример 6.

Если у Вашего сервера два доменных имени: "www.domain.com" и "other.domain.com", и Ваш аккаунт позволяет Вам обслуживать страницы из каталога ~/myhome, Вы должны вызывать функцию setcookie() следующим образом:

setcookie('name', $value, time()+3600, '~/myhome', '**.domain.com**');

##### Пример 7. Счетчик в куки с помощью setcookie()

<?

// Демонстрация работы с $\_COOKIES.

// Вначале счетчик равен нулю.

$count = 0;

// Если в Cookies что-то есть, берем счетчик оттуда.

if (isset($\_COOKIE['count'])) $count = $\_COOKIE['count'];

$count++;

// Записываем в Cookies новое значение счетчика.

setcookie("count", $count, 0x7FFFFFFF, "/");

// Выводим счетчик.

echo $count;

?>

# .htaccess PHP

**.htaccess** (с точкой в начале имени) - это файл-конфигуратор Apache-серверов, который дает возможность конфигурировать работу сервера в отдельных директориях (папках), не предоставляя доступа к главному конфигурационному файлу (apache/conf/httpd.conf). Например, устанавливать права доступа к файлам в директории, менять названия индексных файлов, самостоятельно обрабатывать ошибки Apache, перенаправляя посетителей на специальные страницы ошибок. **.htaccess** представляет собой обычный текстовый документ, расширение которого htaccess. Данный файл обычно находится в корне сайта, однако Вы можете создавать и дополнительные .htaccess-файлы для различных директорий Вашего сайта.

**Mod\_rewrite** - модуль, используемый веб-серверами для преобразования URL'ов.

Если вы хотите вести логи всех операций, выполненных с помощью mod\_rewrite, можно активировать это с помощью следующей записи в httpd.conf:

RewriteLog /usr/local/apache/logs/mod\_rewrite.log

RewriteLogLevel 1

## **Директивы модуля Mod\_rewrite**

* [RewriteBase](https://htmlweb.ru/php/htaccess-mod_rewrite-rewritebase.php)
* [RewriteCond](https://htmlweb.ru/php/htaccess-mod_rewrite-rewritecond.php)
* [RewriteEngine](https://htmlweb.ru/php/htaccess-mod_rewrite-rewriteengine.php)
* [RewriteLock](https://htmlweb.ru/php/htaccess-mod_rewrite-rewritelock.php)
* [RewriteLog](https://htmlweb.ru/php/htaccess-mod_rewrite-rewritelog.php)
* [RewriteLogLevel](https://htmlweb.ru/php/htaccess-mod_rewrite-rewriteloglevel.php)
* [RewriteMap](https://htmlweb.ru/php/htaccess-mod_rewrite-rewritemap.php)
* [RewriteOptions](https://htmlweb.ru/php/htaccess-mod_rewrite-rewriteoptions.php)
* [RewriteRule](https://htmlweb.ru/php/htaccess-mod_rewrite-rewriterule.php)

#### Наиболее часто используемые параметры

%{REQUEST\_URI} Строка запроса (без доменного имени, и GET параметров), пример "/server/htaccess/"  
%{HTTP\_HOST} Доменное имя, например "max22.ru"  
%{QUERY\_STRING} Строка GET параметров

## **Варианты реализации Редиректа с помощью файла .htaccess**

1. Простой редирект:
2. Redirect 301 / http://www.domainname.ru/

или

redirect /secret http://www.site.ru/nosecret

Ставится в файле **.htaccess** или **httpd.conf** для Apache. Первый "/" означает, что всё с верхнего уровня сайта, включая все подкаталоги, будет переадресовано (не забывайте поставить последний "/"). Если Вы хотите переадресовать только страницу, сохранив PR старой страницы, можно сделать так:

Redirect 301 /old/old.htm http://www.you.ru/new.htm где:  
/old/old.htm - путь и имя старой страницы   
http://www.you.com/new.htm - новый путь и новое имя перемещенной страницы

1. Редирект на любую страницу по ip пользователя или при запросе конкретной страницы (а также по маске имени).  
   Если у пользователя ip 192.152.37.125, то он будет перенаправлен на страницу user.php:
2. SetEnvIf REMOTE\_ADDR 192.152.37.125 REDIR="redir"
3. RewriteCond %{REDIR} redir
4. RewriteRule ^/$ /user.php
5. Редирект при запросе определённых файлов. Если запрашиваются файлы, расширение которых не указано в файле .htaccess (gif и jpg), то следует перенаправление:
6. RewriteEngine On
7. RewriteRule !.(gif|jpg)$ index.php
8. Использование mod\_rewrite:
9. Options +FollowSymLinks
10. RewriteEngine on
11. RewriteCond %{HTTP\_HOST} ^yourdomain\.ru
12. RewriteRule ^(.\*)$ http://www.yourdomain.ru/$1 [R=permanent,L]
13. Редирект с регулярным выражением:
14. RedirectMatch 301 (.\*) http://www.yourdomain.ru$1 Прописывается в файле .htaccess.

(.\*) **RedirectMatch** фактически соответствует регулярным образцам выражения после доменного имени. Таким образом, нельзя выполнить соответствие образца на ^/yourdomain.ru. Однако, можно преобразовать страницы с использованием .html расширения к файлам того же самого названия, но с .php расширением:

RedirectMatch 301 (.\*)\.html$ http://www.yourdomain.ru$1.php

Если необходимо сделать различное перенаправление для отдельных страниц, можно использовать следующее:

RedirectMatch Permanent ^/html/resources.html$ http://www.newdomain.com/resources.php

RedirectMatch Permanent ^/html/other\_page.html$ http://www.newdomain.com/other\_page.php

RedirectMatch Permanent ^/(.\*)$ http://www.newdomain.com/

"**RedirectMatch Permanent**" - это эквивалент "RedirectMatch 301", строка с "\*(Wildcard)" должна быть последней в этом списке.

1. Создание удобо читаемых URL  
   Чтобы преобразовать, например, www.site.ru/product.php?id=123 в www.site.ru/product/123 следующим образом:
2. RewriteEngine on
3. RewriteRule ^product/([^/\.]+)/?$ product.php?id=$1 [L]

В следующем примере преобразуем www.site.ru/script.php?product=123 в www.site.ru/cat/product/123/:

RewriteRule cat/(.\*)/(.\*)/$ /script.php?$1=$2

1. Редирект на PHP:
2. header("HTTP/1.1 301 Moved Permanently");
3. header("Location: http://www.newdomain.ru/newdir/newpage.htm");

exit();

Естественно, надо создать страницу, при обращении к которой и будет происходить Редирект, и разместить её на сервере. И лучше укажите HTTP/1.1 (а не HTTP/1.0 или HTTP/0.9, которые не поддерживают виртуальный хостинг)

1. Редирект всех файлов в папке на один файл.  
   Например вы больше не нуждаетесь в разделе сайта Super discount и хотите перенаправить все запросы к папке /superdiscount на один файл /hot-offers.php. Для этого добавляем в .htaccess следующий код.
2. RewriteRule ^superdiscount(.\*)$ /hot-offers.php [L,R=301]
3. Редирект всей папки кроме одного файла  
   В следующем примере все файлы из папки /superdiscount будут редиректится на на файл /hot-offers.php, КРОМЕ файла /superdiscount/my-ebook.html котоый должен редиректится на /hot-to-make-million.html
4. RewriteRule ^superdiscount/my-ebook.html /hot-to-make-million.html [L,R=301]
5. RewriteRule ^superdiscount(.\*)$ /hot-offers.php [L,R=301]
6. Редирект динамического URL на новый файл.  
   Данный вариант пригодится если вы хотите редиректить динамический URL с параметрами на новый статический файл.
7. RewriteRule ^article.jsp?id=(.\*)$ /latestnews.htm [L,R=301]

То есть теперь, запрос к файлу вида http://www.kass.ws/article.jsp?id=8632 и/или http://www.kass.ws/article.jsp?id=1245 будет отправлен на файл http://www.kass.ws/latestnews.htm.

1. Массовый редирект новых файлов.  
   Тепепь перейдем к самому сложному моменту, когда вам надо редиректить массу URL-ов, например после смены вашей CMS. Тут сразу возникает ряд проблем. Во-первых, внесение всех изменившихся адресов в .htaccess файл займет очень много времени, да и само по себе занятие малоприятное. Во-вторых, слишком много записей в .htaccess файле будут тормозить Apache сервера. И в третьих, при внесении такого количества информации высока вероятность, что вы где то ошибетесь. По этому, самый лучший выход, это нанять програмиста который вам напишет динамический редирект.  
   Нижеприведенный пример написан на PHP, но так же может быть выполнен на любом языке. Предположим вы перешли на новую систему ссылок на вашем сайте и все файлы оканчивающиеся на старый id должны быть средирекчены. Сначала создаем в базе таблицу, которая содержит старый id и новый URL для редиректа. old\_id INT new\_url VARCHAR (255) Далее пишем код который свяжет ваши старые id с новыми URL-ами  
   После этого, добавляем следующую строчку в .htaccess:
2. RewriteRule ^/product-(.\*)\_([0-9]+).php /redirectold.php?productid=$2

затем создаем PHP файл redirectold.php, который будет поддерживать 301 редирект:

<?php

function getRedirectUrl($productid) {

// Connect to the database

$dServer = "localhost";

$dDb = "mydbname";

$dUser = "mydb\_user";

$dPass = "password";

$s = @mysql\_connect($dServer, $dUser, $dPass)

or die("Couldn't connect to database server");

@mysql\_select\_db($dDb, $s)

or die("Couldn't connect to database");

$query = "SELECT new\_url FROM redirects WHERE old\_id = ". $productid;

mysql\_query($query);

$result = mysql\_query($query);

$hasRecords = mysql\_num\_rows($result) == 0 ? false : true;

if (!$hasRecords) {

$ret = 'http://www.yoursite.com/';

} else {

while($row = mysql\_fetch\_array($result))

{

$ret = 'http://www.yoursite.com/'. $row["new\_url"];

}

}

mysql\_close($s);

return $ret;

}

$productid = $\_GET["productid"];

$url = getRedirectUrl($productid);

header("HTTP/1.1 301 Moved Permanently");

header("Location: $url");

exit();

?>

Теперь все запросы к вашим старым URL-ам будут вызывать redirectold.php, который найдет новый URL и вернет 301 ответ с вашей новой ссылкой.

## **Редиректы в зависимости от времени**

Когда нужно применять уловки типа содержания зависящего от времени масса вебмастеров все ещё используют CGI скрипты которые производят редиректы на специальные страницы. Как это может быть сделано через mod\_rewrite?

Есть много переменных названных TIME\_xxx для условий редиректа. В связке со специальными лексикографическими образцами для сравнения <STRING, >STRING и =STRING мы можем производить редиректы зависящие от времени:

RewriteEngine on

RewriteCond %{TIME\_HOUR}%{TIME\_MIN} >0700

RewriteCond %{TIME\_HOUR}%{TIME\_MIN} <1900

RewriteRule ^foo\.html$ foo.day.html

RewriteRule ^foo\.html$ foo.night.html

Это выдает содержимое foo.day.html при запросе URL foo.html с 07:00 до 19:00 а в оставшееся время содержимое foo.night.html.

1. Убираем у всех запросов в начале **"WWW."**
2. RewriteEngine on # оглашаем, что хотим использовать mod\_rewrite
3. RewriteCond %{HTTP\_HOST} ^www\.(.\*) [NC]
4. RewriteRule ^/?(.\*) http://%1/$1 [L,R=permanent]

Проверяем доменное имя, если оно начинается с www, то сработает правило: "все, на http://%1/$1". Здесь %1 это наш домен без www (взят из условия), а $1 это адрес (взят из самого правила).

1. Убираем у всех запросов в конце index.php. Переадресуем на страницу без index.php

Поисковые системы плохо относятся к дубрям страниц. Чтобы этого избежать нужно удалить(склеить) страницы вида http://ваш\_домен/ и http://ваш\_домен/index.php

RewriteCond %{THE\_REQUEST} ^.\*/index.php

RewriteRule ^(.\*)index.php$ http://%{HTTP\_HOST}/$1 [R=301,L]

1. Меняем расширение **.html** на **.php**  
   Иногда бывает так, что у Вас статичный веб-сайт, а Вам необходимо, чтобы на нем срабатывал какой-нибудь php-скрипт. Для этого Вам необходимо сказать серверу, чтобы он обрабатывал эту страницу как php-файл.
2. AddHandler application/x-httpd-php .html

Этот прием можно использовать и для других расширений файлов:

AddHandler application/x-httpd-php .xml

AddHandler application/x-httpd-php .asp

### **Запрещение доступа в конкретную директорию**

1. для всех ко всем файлам в директории:
2. deny from all
3. к конкретному файлу:
4. <Files secret.php>
5. deny from all
6. </Files>
7. по ip пользователя:
8. order deny,allow
9. deny from all
10. allow from 192.152.37.125

Доступ в данную директорию будет разрешён только пользователю с ip 192.152.37.125.

А если хотите наоборот, запретить отдельным ip пользователям доступ к вашему сайту, то пропишем следующие строчки:

<Limit GET POST PUT>

order allow,deny

allow from all

deny from 192.152.37.125

deny from 123.456.177

</LIMIT>

1. Директива Options -Indexes - запрет на отображение содержимого каталога при отсутствии индексного файла Иногда нужно сделать так, чтобы в случае отсутствия в каталоге файла, который показывается по умолчанию, не выдавался список файлов в каталоге. Тогда можно добавить в .htaccess такую строчку :
2. Options -Indexes

В этом случае вместо списка файлов в каталоге посетитель получит [HTTP ошибку 403 - access forbidden.](https://htmlweb.ru/other/http_code.php)

1. Запретить доступ к файлам с несколькими типа расширений

<Files ~ "\.(inc|conf|cfg)$">

deny from all

</Files>

Запрещен доступ к файлам с расширением \*.inc, \*.conf и \*.cfg. Хотя директива, по умолчанию, не работает с регулярными выражениями, но их можно включить поставив символ тильды(~) в опциях директивы. Синтаксис следующий:

[тильда] [пробел] [далее\_все\_без\_пробелов] Чтобы блокировать этот доступ, запишем следующее:

RewriteRule ^.htaccess$ - [F]

Это правило переводится так:  
Если кто-то пробует обращаться к файлу .htaccess, система должна произвести код ошибки 'HTTP response of 403' или '403 Forbidden - You don't have permission to access /.htaccess on this server'.

Конструкция ^.htaccess$ в этом регулярном выражении означает:  
**^** - якорь начала строки  
**$** - якорь конца строки  
**.** - в регулярных выражениях точка '.' обозначает мета-символ и должна быть защищена обратным слэшем (backslash), если Вы все-таки хотите использовать именно фактическую точку.

Имя файла должно быть расположено точно между начальным и конечным якорем. Это будет гарантировать то, что только это определенное имя файла и никакое другое, сгенерирует код ошибки.  
**[F]** - специальный 'запрещающий' флажок (forbidden).  
**[NC]** - не учитывать регистр букв.  
**[OR]** - означает 'или следующее условие'.

### **Определение кодировки**

#### Определение кодировки, в которой сервер "отдает" файлы

AddDefaultCharset windows-1251

варианты: **KOI8-R**, **UTF-8**, **Windows-1251**

#### Определение кодировки на загружаемые файлы

CharsetSourceEnc windows-1251

### **Установка пароля на директорию с помощью .htaccess**

Для установки пароля на директорию можно воспользоваться системой базовой авторизации, предусмотренной в веб-сервере Apache. Создаем в каталоге, к которому хотим ограничить доступ по паролю, файл .htaccess с такими директивами:

AuthType Basic

AuthName "Some Name"

AuthUserFile /www/some\_login/www/htdocs/some\_dir/.htpasswd

require valid-user

Путь /www/some\_login/www/htdocs/some\_dir/.htpasswd обозначает полный путь к файлу паролей на диске нашего сервера. Если, например, вы поместите файл .htpasswd (в нем будут пароли) в домашний каталог, куда вы попадаете, зайдя на сервер по FTP, то путь к этому файлу будет иметь вид /www/some\_login/www/htdocs/some\_dir/.htpasswd, где some\_login - Ваш логин. В директиве AuthUserFile указываем абсолютный путь к файлу с логинами/паролями, который мы создадим чуть позже. Если вы создаете файл .htaccess на своем компьютере, а не сразу на сервере используя текстовый редактор, обратите особое внимание на то, что .htaccess должен передаваться по FTP строго в текстовом (ASCII) режиме.

Создаем файл паролей. Файл с паролями должен содержать строки вида login:password. Пароль должен быть зашифрован с использованием алгоритма MD5. Один из способов создать такой файл - воспользоваться программой, входящей в поставку Apache - htpasswd (на нашем сервере она находится в каталоге /usr/local/apache/bin, полный путь - /usr/local/apache/bin/htpasswd).

Рассмотрим, как создать файл паролей в unix shell прямо на сервере. Зайдем в shell и будем выполнять следующие команды:

htpasswd -mbc .htpasswd user1 7B1safkir

- создаем новый файл .htpasswd, в который добавляем запись для пользователя user1 с паролем, указанным в командной строке.

htpasswd .htpasswd user2

- добавляем в уже существующий файл .htpasswd пользователя user2, а пароль вводим вручную в ответ на соответствующий запрос программы.

После окончания заведения всех логинов файл нужно загрузить на сервер.

[О других способах установки паролей на страницу](https://htmlweb.ru/html/password.php)

### **Задаем собственные страницы ошибок**

Задать собственную страницу ошибок можно следующим образом:

ErrorDocument 404 http://www.site.ru/404.php

IE игнорирует страницы размером меньше 512 байт.

### **Индексация директорий и поддиректорий**

Чтобы избежать индексации поисковыми системами директорий и поддиректорий, необходимо прописать такую строку, к примеру:

DirectoryIndex index.php

Эта директива задает файл, который будет вызван при обращении к директории без указания имени файла.

Можно указать несколько индексных страниц. При запросе каталога они будут искаться в том порядке, в котором перечислены в директиве DirectoryIndex. Если не будет найден файл index.html, то будет произведен поиск файла index.php и т.д.

DirectoryIndex index.html index.php index.shtml

Лично я предпочитаю переадресовывать с пустых директорий либо на главную страницу сайта, либо на какую-либо другую подходящую страницу. Например, директорию www.site.ru/pic/ можно переадресовать на www.site.ru.

### **Защита изображений от скачивания**

Очень часто бывает, что веб-мастера нагло копируют контент с Вашего сайта вместе с рисунками, причем рисунки подгружаются с Вашего же сервера. Это создает лишний трафик, что, зачастую, приводит к ряду проблем. Как же защититься от таких веб-мастеров и не помешать поисковым роботам индексировать изображения? Все просто:

RewriteEngine on

RewriteCond %{HTTP\_REFERER} .

RewriteCond %{HTTP\_REFERER} !^http://([^.]+\.)?site\. [NC]

RewriteCond %{HTTP\_REFERER} !google\. [NC]

RewriteCond %{HTTP\_REFERER} !search\?q=cache [NC]

RewriteCond %{HTTP\_REFERER} !msn\. [NC]

RewriteCond %{HTTP\_REFERER} !yahoo\. [NC]

RewriteCond %{REQUEST\_URI} !^/hotlinker\.gif$

RewriteRule \.(gif|jpg|png)$ /hotlinker.gif [NC,L]

hotlinker.gif - изображение, которое будет отображаться, вместо истинных изображений. Рекомендую в этом изображении отобразить Ваш логотип и ссылку на Ваш сайт.

Еще один варинат запрета доступа к картинкам с неразрешенных сайтов:

SetEnvIfNoCase Referer "^$" local\_ref=1

SetEnvIfNoCase Referer "^http://(www\.)?htmlweb\.ru" local\_ref=1

SetEnvIfNoCase Referer "^http://(www\.)?images\.yandex\.ru" local\_ref=1

SetEnvIfNoCase Referer "^http://(www\.)?hghltd\.yandex\.com" local\_ref=1

<FilesMatch ".(jpg|gif|png)">

Order Allow,Deny

Allow from env=local\_ref

</FilesMatch>

Поисковые машини и разного рода сканеры создают коллосальный трафик на вашем сайте. Нижеприведенный блок кода позволит запретить доступ ботам на сайт.

RewriteCond %{HTTP\_USER\_AGENT} (Googlebot|Slurp|spider|Twiceler|heritrix|

Combine|appie|boitho|e-SocietyRobot|Exabot|Nutch|OmniExplorer|

MJ12bot|ZyBorg/1|Ask\ Jeeves|AskJeeves|ActiveTouristBot|

JemmaTheTourist| agadine3|BecomeBot|Clustered-Search-Bot|

MSIECrawler|freefind|galaxy|genieknows|INGRID|grub-client|

MojeekBot|NaverBot|NetNose-Crawler|OnetSzukaj|PrassoSunner|

Asterias\ Crawler|T-H-U-N-D-E-R-S-T-O-N-E|GeorgeTheTouristBot|

VoilaBot|Vagabondo|fantomBro wser|stealthBrowser|cloakBrowser|

fantomCrew\ Browser|Girafabot|Indy\ Library|Intelliseek|Zealbot|

Windows\ 95|^Mozilla/4\.05\ \[en\]$|^Mozilla/4\.0$) [NC]

RewriteRule ^(.\*)$ - [F]

#

RewriteCond %{HTTP\_USER\_AGENT} ^Mozilla.\* [NC,OR]

RewriteCond %{HTTP\_USER\_AGENT} ^Opera.\* [NC,OR]

RewriteCond %{HTTP\_USER\_AGENT} ^Firefox.\* [NC,OR]

RewriteCond %{HTTP\_USER\_AGENT} ^Netscape.\* [NC]

RewriteRule ^(.\*)$ - [L]

RewriteRule ^(.\*)$ - [F]

### **Отслеживание обращений к файлу robots.txt**

Чтобы иметь больше информации о посещении поисковиков, полезно иметь подробную информацио об обращении к файлу robots.txt Для того, чтобы оганизовать это, в '.htaccess' должны быть следующие записи:

RewriteEngine on

Options +FollowSymlinks

RewriteBase /

RewriteRule ^robots.txt$ /robot.php?%{REQUEST\_URI}

Теперь при запросе файла 'robots.txt' наш RewriteRule переадресует посетителя (робота) к обрабатывающему запросы скрипту robot.php. Кроме того, переменная передается скрипту, которая будет обработана в соответствии с вашими нуждами. 'REQUEST\_URI' определяет имя запрашиваемого файла. В данном примере это - 'robots.txt'. Скрипт прочтет содержание 'robots.txt' и отправит его web-браузеру или роботу поискового сервера. Таким образом, мы можем считать хиты посетителей и вести лог-файлы.

### **PHPSESSID**

Для отключения добавления PHPSESSID к URL вставьте в начало index.php:

ini\_set("session.use\_trans\_sid", 0);

Либо в .htaccess пропишите:

php\_flag session.use\_trans\_sid Off

Если вам все это показалось сложным, воспользуйтесь готовым [сервисом преобразования динамических URL в статические с помощью htaccess](http://www.seo.su/)

## **Директивы кеширования**

Кэширование для всех типов файлов по времени доступа

ExpiresActive on

ExpiresDefault "access plus 600 seconds"

Кэширование для всех типов файлов по времени изменения

ExpiresActive on

ExpiresDefault "modification plus 600 seconds"

Кэширование для определённых типов файлов

ExpiresByType text/css "modification plus 600 seconds"

ExpiresByType image/jpeg "modification plus 600 seconds"

ExpiresByType image/gif "modification plus 600 seconds"

ExpiresByType image/x-ico "modification plus 600 seconds"

ExpiresByType image/png "modification plus 600 seconds"

### **Запрет кеширования с помощью сервера Apache**

Откройте файл конфигурации сервера Apache **httpd.conf** и раскомментируйте следующие строчки:

LoadModule expires\_module modules/mod\_expires.so

LoadModule headers\_module modules/mod\_headers.so

...

AddModule mod\_expires.c

AddModule mod\_headers.c

Впишите в .htaccess следующее:

# Запрещение кеширования в этой папке

# Необходимо включение модулей

# mod\_headers.c и mod\_expires.c

#

# Заголовок Cache-Control

<IfModule mod\_headers.c>

Header append Cache-Control "no-store, no-cache, must-revalidate"

</IfModule>

# Заголовок Expires

<IfModule mod\_expires.c>

ExpiresActive On

ExpiresDefault "now"

</IfModule>

Необходимые заголовки будут передаваться автоматически, и специально их писать в PHP уже не нужно - кэш уже выключен!

[Описание http-заголовка кеширования Cache-control](https://htmlweb.ru/other/http_headers.php#Cache-control)

### **Кеширование с помощью файла .htaccess**

# Разрешение кеширования в этой папке

# Необходимо включение модулей

# mod\_headers.c и mod\_expires.c

#

<IfModule mod\_expires.c>

ExpiresActive on

#ExpiresDefault "access plus 1 hours"

#ExpiresDefault "access plus 10 years"

ExpiresDefault "access plus 1 month"

ExpiresByType text/cache-manifest "access plus 0 seconds"

ExpiresByType text/html "access plus 0 seconds"

ExpiresByType text/xml "access plus 0 seconds"

ExpiresByType application/xml "access plus 0 seconds"

ExpiresByType application/json "access plus 0 seconds"

ExpiresByType application/rss+xml "access plus 1 month"

ExpiresByType image/x-icon "access plus 1 week"

ExpiresByType image/gif "access plus 1 year"

ExpiresByType image/png "access plus 1 year"

ExpiresByType image/jpg "access plus 1 year"

ExpiresByType image/jpeg "access plus 1 year"

ExpiresByType video/ogg "access plus 1 year"

ExpiresByType audio/ogg "access plus 1 year"

ExpiresByType audio/mp3 "access plus 1 year"

ExpiresByType video/mp4 "access plus 1 year"

ExpiresByType video/webm "access plus 1 year"

ExpiresByType text/x-component "access plus 1 month"

ExpiresByType font/truetype "access plus 1 year"

ExpiresByType font/opentype "access plus 1 year"

ExpiresByType application/x-font-woff "access plus 1 year"

ExpiresByType image/svg+xml "access plus 1 month"

ExpiresByType application/vnd.ms-fontobject "access plus 1 year"

ExpiresByType text/css "access plus 2 months"

ExpiresByType application/javascript "access plus 2 months"

ExpiresByType text/javascript "access plus 2 months"

<IfModule mod\_headers.c>

Header append Cache-Control "public"

</IfModule>

</IfModule>

### **Кеширование javascript файлов с помощью файла .htaccess**

<FilesMatch .\*\.js$>

ExpiresDefault "access plus 3 days"

</FilesMatch>

Будьте осторожны при кешировании, т.к. при изменении файла, пользователь может получить новый вариант только через 3 дня!

### **Как заставить html-страницы обрабатывать php-код?**

Пропишите в своем файле .htaccess следующие строки:

RemoveHandler .php .htm .html

AddHandler application/x-httpd-php .php .htm .html

## **Как разместить несколько сайтов на одном виртуальном хостинге?**

Чтобы разместить два или более сайтов на одном виртуальном хостинге, вопреки отведенному вам тарифным планом количеству доменов необходимо в файле ".htaccess" прописать следующие строки:

RewriteEngine On

RewriteRule ^newdirectory/ - [L]

RewriteCond %{HTTP\_HOST} (www.)?newdomain.ru [NC]

RewriteRule (.\*) newdirectory/$1 [L]

Где:  
newdirectory/ - папка, в которой будет лежать второй сайт  
newdomain.ru - домен, для которого мы делаем перенаправление

Обратите внимание, что при этом у Вас будет единый почтовый аккаунт. Т.е. если, у Вас существует ящик admin@domain.ru, то после подключения домена newdomain.ru у ящика admin@domain.ru появляется второе имя - admin@newdomain.ru. А при создании любого нового ящика (например info), ему автоматически присваиваются два имени - info@domain.ru и info@newdomain.ru.

## **Поиск страниц больше чем в одном каталоге**

Иногда необходимо позволить веб-серверу искать страницы больше чем в одном каталоге.

RewriteEngine on

# во-первых попытаемся найти это в указанном месте/...

# ...и если нашли то заканчиваем поиск:

RewriteCond /your/docroot/dir1/%{REQUEST\_FILENAME} -f

RewriteRule ^(.+) /your/docroot/dir1/$1 [L]

# во-вторых - попытаемся найти это в pub/...

# ...и если нашли то заканчиваем поиск:

RewriteCond /your/docroot/dir2/%{REQUEST\_FILENAME} -f

RewriteRule ^(.+) /your/docroot/dir2/$1 [L]

# иначе продолжаем для других директив

RewriteRule ^(.+) - [PT]

## **Виртуальные хосты пользователей**

Если Вы хотите предоставлять адреса www.subdomain.domain.ru для страниц пользователей, Вы можете использовать следующий набор правил для преобразования http://www.subdomain.domain.ru/path во внутренний путь /home/subdomain/path:

RewriteEngine on

RewriteCond %{HTTP\_HOST} ^www\.[^.]+\.ru$

RewriteRule ^(.+) %{HTTP\_HOST}$1 [C]

RewriteRule ^www\.([^.]+)\.ru(.\*) /home/$1$2

## **Повреждаются файлы при загрузке на сервер**

Если при передаче файлов через формы (при указанном enctype="multipart/form-data") бинарные данные повреждаются, пропишите в /cgi-bin/.htaccess директиву:

CharsetRecodeMultipartForms Off.

## **Ошибка загрузки SWF файлов. Ошибки при обращении к страницам, содержащим ключевые слова, типа $\_REQUEST**

Такое может происходить из-за установленного модуля <mod\_security> в Apache. По умолчанию он блокирует в запросах строки с SQL аргументами и другими потенциально опасными командами.

Возможные сообщения об ошибке:

**Forbidden**

You don't have permission to access /adm/index.php on this server.

Additionally, a 404 Not Found error was encountered while

trying to use an ErrorDocument to handle the request.

или

Запрос небезопасен и был отвергнут.

Добавьте в .htaccess

<IfModule mod\_security.c>

SecFilterEngine Off

SecFilterScanPOST Off

</IfModule>

Для сообщения:

"POST /wp-admin/async-upload.php HTTP/1.1" 406 354 "-" "Shockwave Flash"

можно снять защиту только на загрузку файлов на сервер:

<IfModule mod\_security.c>

<Files async-upload.php>

SecFilterEngine Off

SecFilterScanPOST Off

</Files>

</IfModule>

Оптимально снимать защиту только с той папки, в которой это необходимо, не убирая защиту со всего сайта.

## **Переменные сервера**

Это переменные вида **%{NAME\_OF\_VARIABLE}**

где NAME\_OF\_VARIABLE может быть строкой взятой из следующего списка:

|  |  |  |
| --- | --- | --- |
| HTTP заголовки: | соединение & запрос: |  |
| HTTP\_USER\_AGENT HTTP\_REFERER HTTP\_COOKIE HTTP\_FORWARDED HTTP\_HOST HTTP\_PROXY\_CONNECTION HTTP\_ACCEPT | REMOTE\_ADDR REMOTE\_HOST REMOTE\_USER REMOTE\_IDENT REQUEST\_METHOD SCRIPT\_FILENAME PATH\_INFO QUERY\_STRING AUTH\_TYPE |  |
| внутренние сервера: | системные: | специальные: |
| DOCUMENT\_ROOT SERVER\_ADMIN SERVER\_NAME SERVER\_ADDR SERVER\_PORT SERVER\_PROTOCOL SERVER\_SOFTWARE | TIME\_YEAR TIME\_MON TIME\_DAY TIME\_HOUR TIME\_MIN TIME\_SEC TIME\_WDAY TIME | API\_VERSION THE\_REQUEST REQUEST\_URI REQUEST\_FILENAME IS\_SUBREQ |

Эти переменные полностью соответствуют названным похожим образом MIME-заголовкам HTTP, и переменным сервера Apache или полям struct tm систем Unix. Те, что являются для mod\_rewrite специальными включают:

**IS\_SUBREQ** - Будет содержать текст «true» если запрос выполняется в текущий момент как подзапрос, «false» в другом случае. Подзапросы могут быть сгенерированны модулями которым нужно иметь дело с дополнительными файлами или URI для того чтобы выполнить собственные задачи.

**API\_VERSION** - Это версия API модуля Apache (внутренний интерфейс между сервером и модулем) в текущей сборке сервера, что определено в include/ap\_mmn.h. API версия модуля соответствует используемой версии Apache (для версии Apache 1.3.14, к примеру это 19990320:10), однако это в основном интересно авторам модулей.

**THE\_REQUEST** - Полная строка HTTP запроса отправленная браузером серверу (т.е., «GET /index.html HTTP/1.1»). Она не включает какие-либо дополнительные заголовки отправляемые браузером.

**REQUEST\_URI** - Ресурс, запрошенный в строке HTTP запроса.

**REQUEST\_FILENAME** - Полный путь в файловой системе сервера к файлу или скрипту соответствующим этому запросу.

Примечания:

1. Переменные SCRIPT\_FILENAME и REQUEST\_FILENAME содержат одинаковые значения, т.е., значение поля filename внутренней структуры request\_rec сервера Apache. Первое имя это просто широко известное имя переменной CGI в то время как второе это постоянная копия REQUEST\_URI (содержащая значение поля uri структуры request\_rec).
2. Есть специальный формат: %{ENV:переменная} где переменная может быть любой переменной окружения. Это ищется во внутренних структурах Apache и (если там нет) с помощью вызова getenv()из процесса Apache сервера.
3. Есть специальный формат: %{HTTP:заголовок} где заголовок может быть любым именем HTTP MIME-заголовка. Это ищется в HTTP запросе. Пример: %{HTTP:Proxy-Connection} значение HTTP заголовка «Proxy-Connection:».
4. Есть специальный формат %{LA-U:переменная} опережающих запросов которые производятся внутренним (основанном на URL) подзапросом для определения конечного значения переменной. Используйте это когда вы хотите использовать переменную для преобразований, которая реально определяется позднее, в какой-либо фазе API, и таким образом недоступна на данном этапе. Для примера когда вы хотите преобразовать соответственно переменной REMOTE\_USER из контекста сервера (файл httpd.conf) вы должны использовать %{LA-U:REMOTE\_USER} потому что эта переменная устанавливается в фазах авторизации которые идут после фазы трансляции URL в которой и работает mod\_rewrite. С другой стороны, по причине реализации работы mod\_rewrite в контексте каталога (файл .htaccess) через Fixup фазу API и из-за того, фазы авторизации идут до этой фазы, вы просто можете там использовать %{REMOTE\_USER}.
5. Есть специальный формат: %{LA-F:переменная} который создает внутренний (основанный на имени файла) подзапрос для определения конечного значения переменной. В основном это то же самое что и формат LA-U приведенный выше.

### **Главная страница без дублирования**

Обычно код главной страницы физически расположен в файле index.html или index.php, но сайт должен открываться по любому из запросов: yoursite.ru, yoursite.ru/index.html, www.yoursite.ru и www.yoursite.ru/index.html. Но для поисковых систем это четыре разных URL! Если не настроить .htaccess верно, поисковик добавит в свой индекс четыре одинаковых страницы. Это признак некачественного сайта. Избежать этой проблемы можно с помощью такого кода в .htaccess:

Options +FollowSymLinks

RewriteEngine on

RewriteCond %{HTTP\_HOST} ^yoursite.ru

RewriteRule (.\*) http://www.yoursite.ru/$1 [R=301,L]

RewriteCond %{THE\_REQUEST} ^[A-Z]{3,9}\ /index\.html\ HTTP/

RewriteRule ^index\.html$ http://www.yoursite.ru/ [R=301,L]

Все страницы-дубли будут склеены редиректом с кодом [301](https://htmlweb.ru/other/http_code.php) с главной страницей – http://www.yoursite.ru/.

### **Дубли страниц без слэша в конце URL**

Чтобы предотвратить ситуацию с индексированием страниц www.yoursite.ru/about и www.yoursite.ru/about/ как разных, ставим следущий код:

Со страниц без слэша будет установлен редирект на «слэшевые».

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_URI} !(.\*)/$

RewriteRule ^(.\*)$ /$1/ [R=301,L]

### **Сохранение(загрузка) файлов вместо открытия**

Многие видели, как при попытке скачать архив с расширением .rar браузер открывает его в виде простого текста из мешанины символов. Это значит, что на сервере сайта не настроено принудительное сохранение типов файлов, которые не должны открываться в браузере.

AddType application/octet-stream .rar .doc .mov .avi .pdf .xls .mp4

### **Сжатие отправляемых страниц**

SetOutputFilter DEFLATE

Header unset ETag

FileETag None

### **Принудительная постановка замыкающего слеша**

Следующий код всегда будет добавлять слеш в адрес URL вашего сайта, что хорошо помогает в области SEO сайта.

RewriteCond %{REQUEST\_URI} /+[^\.]+$

RewriteRule ^(.+[^/])$ %{REQUEST\_URI}/ [R=301,L]

### **Междоменное использование шрифтов для FireFox**

При использовании встроенных шрифтов [Firefox](http://moiprogrammy.net/) не позволяет брать их с внешних сайтов. Следующий код для файла .htaccess позволит обойти данное ограничение.

<FilesMatch "\.(ttf|otf|eot|woff)$">

<IfModule mod\_headers.c>

#Замените yourdomain.com на адрес вашего блога

Header set Access-Control-Allow-Origin "http://yourdomain.com"

</IfModule>

</FilesMatch>

### **Выполнять PHP в файлах JavaScript**

При разработке кода JavaScript иногда требуется использовать PHP в файлах .js, например, для получения данных из базы данных.

AddType application/x-httpd-php .js

AddHandler x-httpd-php5 .js

<FilesMatch "\.(js|php)$">

SetHandler application/x-httpd-php

</FilesMatch>

### **Разложить файлы robots.txt, sitemap.xml и т.п. в папки доменов**

# Если есть запрашиваемый файл из корня сайта, в папке домена, то перенаправляем его туда

RewriteCond %{DOCUMENT\_ROOT}/domain/%{HTTP\_HOST}/root%{REQUEST\_URI} -f

RewriteRule ^(.\*)$ /domain/%{HTTP\_HOST}/root/$1 [L]

### **Блокируем User Agents через .htaccess**

Если вы столкнулись с проблемой, что какие-то типы User Agent нагружают ваш сервер ненужными запросами, то от них можно избавиться добавив в .htaccess следующие строчки:

SetEnvIfNoCase User-Agent "^Black Hole" bad\_bot

SetEnvIfNoCase User-Agent "^Titan" bad\_bot

SetEnvIfNoCase User-Agent "^WebStripper" bad\_bot

SetEnvIfNoCase User-Agent "^NetMechanic" bad\_bot

SetEnvIfNoCase User-Agent "^CherryPicker" bad\_bot

SetEnvIfNoCase User-Agent "^EmailCollector" bad\_bot

SetEnvIfNoCase User-Agent "^EmailSiphon" bad\_bot

SetEnvIfNoCase User-Agent "^WebBandit" bad\_bot

SetEnvIfNoCase User-Agent "^EmailWolf" bad\_bot

SetEnvIfNoCase User-Agent "^ExtractorPro" bad\_bot

SetEnvIfNoCase User-Agent "^CopyRightCheck" bad\_bot

SetEnvIfNoCase User-Agent "^Crescent" bad\_bot

SetEnvIfNoCase User-Agent "^Wget" bad\_bot

SetEnvIfNoCase User-Agent "^SiteSnagger" bad\_bot

SetEnvIfNoCase User-Agent "^ProWebWalker" bad\_bot

SetEnvIfNoCase User-Agent "^CheeseBot" bad\_bot

SetEnvIfNoCase User-Agent "^Teleport" bad\_bot

SetEnvIfNoCase User-Agent "^TeleportPro" bad\_bot

SetEnvIfNoCase User-Agent "^MIIxpc" bad\_bot

SetEnvIfNoCase User-Agent "^Telesoft" bad\_bot

SetEnvIfNoCase User-Agent "^Website Quester" bad\_bot

SetEnvIfNoCase User-Agent "^WebZip" bad\_bot

SetEnvIfNoCase User-Agent "^moget/2.1" bad\_bot

SetEnvIfNoCase User-Agent "^WebZip/4.0" bad\_bot

SetEnvIfNoCase User-Agent "^WebSauger" bad\_bot

SetEnvIfNoCase User-Agent "^WebCopier" bad\_bot

SetEnvIfNoCase User-Agent "^NetAnts" bad\_bot

SetEnvIfNoCase User-Agent "^Mister PiX" bad\_bot

SetEnvIfNoCase User-Agent "^WebAuto" bad\_bot

SetEnvIfNoCase User-Agent "^TheNomad" bad\_bot

SetEnvIfNoCase User-Agent "^WWW-Collector-E" bad\_bot

SetEnvIfNoCase User-Agent "^RMA" bad\_bot

SetEnvIfNoCase User-Agent "^libWeb/clsHTTP" bad\_bot

SetEnvIfNoCase User-Agent "^asterias" bad\_bot

SetEnvIfNoCase User-Agent "^httplib" bad\_bot

SetEnvIfNoCase User-Agent "^turingos" bad\_bot

SetEnvIfNoCase User-Agent "^spanner" bad\_bot

SetEnvIfNoCase User-Agent "^InfoNaviRobot" bad\_bot

SetEnvIfNoCase User-Agent "^Harvest/1.5" bad\_bot

SetEnvIfNoCase User-Agent "^Bullseye/1.0" bad\_bot

SetEnvIfNoCase User-Agent "^Mozilla/4.0 (compatible; BullsEye; Windows 95)" bad\_bot

SetEnvIfNoCase User-Agent "^Crescent Internet ToolPak HTTP OLE Control v.1.0" bad\_bot

SetEnvIfNoCase User-Agent "^CherryPickerSE/1.0" bad\_bot

SetEnvIfNoCase User-Agent "^CherryPicker /1.0" bad\_bot

SetEnvIfNoCase User-Agent "^WebBandit/3.50" bad\_bot

SetEnvIfNoCase User-Agent "^NICErsPRO" bad\_bot

SetEnvIfNoCase User-Agent "^Microsoft URL Control - 5.01.4511" bad\_bot

SetEnvIfNoCase User-Agent "^DittoSpyder" bad\_bot

SetEnvIfNoCase User-Agent "^Foobot" bad\_bot

SetEnvIfNoCase User-Agent "^WebmasterWorldForumBot" bad\_bot

SetEnvIfNoCase User-Agent "^SpankBot" bad\_bot

SetEnvIfNoCase User-Agent "^BotALot" bad\_bot

SetEnvIfNoCase User-Agent "^lwp-trivial/1.34" bad\_bot

SetEnvIfNoCase User-Agent "^lwp-trivial" bad\_bot

SetEnvIfNoCase User-Agent "^Wget/1.6" bad\_bot

SetEnvIfNoCase User-Agent "^BunnySlippers" bad\_bot

SetEnvIfNoCase User-Agent "^Microsoft URL Control - 6.00.8169" bad\_bot

SetEnvIfNoCase User-Agent "^URLy Warning" bad\_bot

SetEnvIfNoCase User-Agent "^Wget/1.5.3" bad\_bot

SetEnvIfNoCase User-Agent "^LinkWalker" bad\_bot

SetEnvIfNoCase User-Agent "^cosmos" bad\_bot

SetEnvIfNoCase User-Agent "^moget" bad\_bot

SetEnvIfNoCase User-Agent "^hloader" bad\_bot

SetEnvIfNoCase User-Agent "^humanlinks" bad\_bot

SetEnvIfNoCase User-Agent "^LinkextractorPro" bad\_bot

SetEnvIfNoCase User-Agent "^Offline Explorer" bad\_bot

SetEnvIfNoCase User-Agent "^Mata Hari" bad\_bot

SetEnvIfNoCase User-Agent "^LexiBot" bad\_bot

SetEnvIfNoCase User-Agent "^Web Image Collector" bad\_bot

SetEnvIfNoCase User-Agent "^The Intraformant" bad\_bot

SetEnvIfNoCase User-Agent "^True\_Robot/1.0" bad\_bot

SetEnvIfNoCase User-Agent "^True\_Robot" bad\_bot

SetEnvIfNoCase User-Agent "^BlowFish/1.0" bad\_bot

SetEnvIfNoCase User-Agent "^JennyBot" bad\_bot

SetEnvIfNoCase User-Agent "^MIIxpc/4.2" bad\_bot

SetEnvIfNoCase User-Agent "^BuiltBotTough" bad\_bot

SetEnvIfNoCase User-Agent "^ProPowerBot/2.14" bad\_bot

SetEnvIfNoCase User-Agent "^BackDoorBot/1.0" bad\_bot

SetEnvIfNoCase User-Agent "^toCrawl/UrlDispatcher" bad\_bot

SetEnvIfNoCase User-Agent "^WebEnhancer" bad\_bot

SetEnvIfNoCase User-Agent "^TightTwatBot" bad\_bot

SetEnvIfNoCase User-Agent "^suzuran" bad\_bot

SetEnvIfNoCase User-Agent "^VCI WebViewer VCI WebViewer Win32" bad\_bot

SetEnvIfNoCase User-Agent "^VCI" bad\_bot

SetEnvIfNoCase User-Agent "^Szukacz/1.4" bad\_bot

SetEnvIfNoCase User-Agent "^QueryN Metasearch" bad\_bot

SetEnvIfNoCase User-Agent "^Openfind data gathere" bad\_bot

SetEnvIfNoCase User-Agent "^Openfind" bad\_bot

SetEnvIfNoCase User-Agent "^Xenu's Link Sleuth 1.1c" bad\_bot

SetEnvIfNoCase User-Agent "^Xenu's" bad\_bot

SetEnvIfNoCase User-Agent "^Zeus" bad\_bot

SetEnvIfNoCase User-Agent "^RepoMonkey Bait & Tackle/v1.01" bad\_bot

SetEnvIfNoCase User-Agent "^RepoMonkey" bad\_bot

SetEnvIfNoCase User-Agent "^Zeus 32297 Webster Pro V2.9 Win32" bad\_bot

SetEnvIfNoCase User-Agent "^Webster Pro" bad\_bot

SetEnvIfNoCase User-Agent "^EroCrawler" bad\_bot

SetEnvIfNoCase User-Agent "^LinkScan/8.1a Unix" bad\_bot

SetEnvIfNoCase User-Agent "^Keyword Density/0.9" bad\_bot

SetEnvIfNoCase User-Agent "^Kenjin Spider" bad\_bot

SetEnvIfNoCase User-Agent "^Cegbfeieh" bad\_bot

<limit get="" post="" head="">

Order Allow,Deny

Allow from all

Deny from env=bad\_bot

</limit>

### **Переадресация по языку**

RewriteEngine on

RewriteCond %{HTTP:Accept-Language} (ru) [NC]

RewriteRule .\* КУДА [L]

### **Защита картинок от скачивания по ссылкам на других сайтах**

Включая переходы без REFERER, т.е. когда адрес указывают в строке браузера.

RewriteEngine on

RewriteCond %{HTTP\_REFERER} !^$

RewriteCond %{HTTP\_REFERER} !^http(s)?://(www\.)?yourdomain.com [NC]

RewriteRule \.(jpg|jpeg|png|gif)$ КУДА\_ПОСЛАТЬ [NC,R,L]

Только по сссылкам на чужих сайтах:

RewriteEngine on

RewriteCond %{HTTP\_REFERER} !^http(s)?://(www\.)?yourdomain.com [NC]

RewriteRule \.(jpg|jpeg|png|gif)$ КУДА\_ПОСЛАТЬ [NC,R,L]

### **Блокировать пользователей с определенным рефером**

Если вы не хотите, чтобы пользователи переходя с определенных сайтов попадали на ваш, вы можете это запретить.

RewriteEngine on

RewriteCond %{HTTP\_REFERER} bannedurl1.com [NC,OR]

RewriteCond %{HTTP\_REFERER} bannedurl2.com [NC,OR]

RewriteRule .\* - [F]

bannedurl1.com и bannedurl2.com — примеры запрещенных сайтов.

### **Ограничение размера файла загрузки в PHP, максимального размера запроса и максимального времени выполнения скрипта**

Объем загружаемого файла:

php\_value upload\_max\_filesize 15M

Максимальный размер запроса для загрузки в PHP:

php\_value post\_max\_size 10M

Время выполнения скрипта:

php\_value max\_execution\_time 240

Время для скрипта на разбор введенных данных:

php\_value max\_input\_time 180

### **Указываем для IE режим вывода**

Установка заголовка X-UA-Compatible:

Header set X-UA-Compatible "IE=Edge"

### **Установка Vary: Accept-Encoding**

Для поисковой оптимизации, и уменьшения времени загрузки страницы, Google рекомендует установить заголовок Vary: Accept-Encoding

Заголовок ответа HTTP/1.1 Vary позволяет серверу указать, что закэшированный ресурс может использоваться без проверки только, если указанные в Vary заголовки совпадают с заголовками запроса. Значения: Accept-Encoding, Host, User-Agent, Accept-Language.

<IfModule mod\_headers.c>

<FilesMatch "\.(js|css|xml|gz|html)$">

Header append Vary: Accept-Encoding

</FilesMatch>

</IfModule>

### **Общий файл картинки для всех доменов(глобальные алиасы)**

Alias /javascripts /usr/share/javascript/

<Directory "/usr/share/javascript/">

Options FollowSymLinks MultiViews

</Directory>

Положите общий файл, например, icon.png в директорию /usr/share/javascript/ В файл /etc/javascript-common/javascript-common.conf и добавьте строку:

Alias /apple-touch-icon-precomposed.png /usr/share/javascript/icon.png

### **При загрузке .doc, .docx файлы открываются в браузере с непонятной кодировкой**

Если у вас открываются .doc, .docx файлы в браузере с непонятной кодировкой и вы хотите чтобы при левом клике файл сохранялся, а не открывался, то вам необходимо прописать в .htaccess:

AddType application/force-download doc

AddType application/force-download docx

AddType application/force-download xls

AddType application/force-download xlsx

### **Перенаправление на безопасное соединение https**

Если вы используете https и хотите, чтобы все пользователи перенаправлялись на него, то вам поможет следующий код:

RewriteEngine On

RewriteCond %{HTTPS} !on

RewriteRule (.\*) https://%{HTTP\_HOST}%{REQUEST\_URI}

### **Доступ если установлена cookie с помощью htaccess**

Если Ваш логин на сайте admin и он устанавливается в куку с именем login Вы можеле легко закрыть доступ посторонним в любую директорию. Для этого создайте в этой директории(папке) файл .htaccess со следующим содержимым:

RewriteEngine On

RewriteCond %{HTTP\_COOKIE} !login=admin [NC]

RewriteRule .\* http://%{HTTP\_HOST}/ [L,R=301]

По этому же принципу можно закрывать сайт на профилактические работы для всех кроме админа.

### **Доступ к заголовкам «If-Modified-Since» и «If-None-Match» для реализации http 304 ответа**

Если PHP не установлен как модуль Apache, то для доступа к заголовкам «If-Modified-Since» и «If-None-Match» необходимо в корневом каталоге веб-сайта поместить файл .htaccess следующего содержания:

RewriteEngine On

RewriteRule .\* — [E=HTTP\_IF\_MODIFIED\_SINCE:%{HTTP:If-Modified-Since}]

RewriteRule .\* — [E=HTTP\_IF\_NONE\_MATCH:%{HTTP:If-None-Match}]

После этого, необходимые заголовки будут доступны как $\_SERVER['HTTP\_IF\_MODIFIED\_SINCE'] и $\_SERVER['HTTP\_IF\_NONE\_MATCH'].

Следует заметить, что заголовки «If-Modified-Since» и «If-None-Match» не отправляются браузером, если в предыдущих запросах к данной странице он не получал в ответе веб-сервера заголовок «Last-Modified». Кроме того, при использовании в веб-приложении сессий с установками по умолчанию, указанные заголовки также не будут присылаться браузером. Для того чтобы избежать такого поведения браузера, необходимо перед запуском сессии выполнять функцию session\_cache\_limiter, передавая в качестве аргумента параметр 'private\_no\_expire':

session\_cache\_limiter('private\_no\_expire');

session\_start();

### **Пред и пост обработка с помощью .htaccess**

# Запуск перед запрашиваемым файлом

php\_value auto\_prepend\_file "/dir/header.php"

# Запуск после запрашиваемого файла

php\_value auto\_append\_file "/dir/footer.php"

## **Приемы безопасного программирования на PHP**

Данная статья не претендует на роль всеобъемлющего руководства на тему "как сделать так, чтоб меня никто не поломал". Так не бывает. Единственная цель этой статьи - показать некоторые используемые мной приемы для защиты веб-приложений типа WWW-чатов, гостевых книг, веб-форумов и других приложений подобного рода. Итак, давайте рассмотрим некоторые приемы программирования на примере некоей гостевой книги, написанной на PHP.

Первой заповедью веб-программиста, желающего написать более-менее защищенное веб-приложение, должно стать "Никогда не верь данным, присылаемым тебе пользователем". Пользователи - это по определению такие злобные хакеры, которые только и ищут момента, как бы напихать в формы ввода всякую дрянь типа PHP, JavaScript, SSI, вызовов своих жутко хакерских скриптов и тому подобных ужасных вещей. Поэтому первое, что необходимо сделать - это жесточайшим образом отфильтровать все данные, присланные пользователем.

Допустим, у нас в гостевой книге существует 3 формы ввода: имя пользователя, его e-mail и само по себе тело сообщения. Прежде всего, ограничим количество данных, передаваемых из форм ввода чем-нибудь вроде:

<input type=text name=username maxlength=20>

На роль настоящей защиты, конечно, это претендовать не может - единственное назначение этого элемента - ограничить пользователя от случайного ввода имени длиннее 20-ти символов. А для того, чтобы у пользователя не возникло искушения скачать документ с формами ввода и подправить параметр maxlength, установим где-нибудь в самом начале скрипта, обрабатывающего данные, проверку переменной окружения web-сервера HTTP-REFERER:

<?php

$referer=getenv("HTTP\_REFERER");

if (!ereg("^http://www.myserver.com",$referer)) {

echo "hacker? he-he...\n";

exit;

}

?>

Теперь, если данные переданы не из форм документа, находящегося на сервере www.myserver.com, хакеру будет выдано деморализующее сообщение. На самом деле, и это тоже не может служить 100%-ой гарантией того, что данные действительно переданы из нашего документа. В конце концов, переменная HTTP\_REFERER формируется браузером, и никто не может помешать хакеру подправить код браузера, или просто зайти телнетом на 80-ый порт и сформировать свой запрос. Так что подобная защита годится только от Ну Совсем Необразованных хакеров. Впрочем, по моим наблюдениям, около 80% процентов злоумышленников на этом этапе останавливаются и дальше не лезут - то ли IQ не позволяет, то ли просто лень. Можно вынести этот фрагмент кода в отдельный файл, и вызываю его отовсюду, откуда это возможно.

Следующим этапом станет пресловутая жесткая фильтрация переданных данных. Прежде всего, не будем доверять переменной **maxlength** в формах ввода и ручками порежем строку:

$username=substr($username,0,20);

Не дадим пользователю использовать пустое поле имени - просто так, чтобы не давать писать анонимные сообщения:

if (empty($username)) {

echo "invalid username";

exit;

}

Запретим пользователю использовать в своем имени любые символы, кроме букв русского и латинского алфавита, знака "\_" (подчерк), пробела и цифр:

if (preg\_match("/[^(\w)|(\x7F-\xFF)|(\s)]/",$username)) {

echo "invalid username";

exit;

}

Я предпочитаю везде, где нужно что-нибудь более сложное, чем проверить наличие паттерна в строке или поменять один паттерн на другой, использовать Перл-совместимые регулярные выражения (Perl-compatible Regular Expressions). То же самое можно делать и используя стандартные PHP-шные ereg() и eregi(). Я не буду приводить здесь эти примеры - это достаточно подробно описано в мануале.

Для поля ввода адреса e-mail добавим в список разрешенных символов знаки "@" и ".", иначе пользователь не сможет корректно ввести адрес. Зато уберем русские буквы и пробел:

if (preg\_match("/[^(\w)|(\@)|(\.)]/",$usermail)) {

echo "invalid mail";

exit;

}

Поле ввода текста мы не будем подвергать таким жестким репрессиям - перебирать все знаки препинания, которые можно использовать, попросту лень, поэтому ограничимся использованием функций[nl2br()](https://htmlweb.ru/php/function/nl2br.php) и [htmlspecialchars()](https://htmlweb.ru/php/function/htmlspecialchars.php) - это не даст врагу понатыкать в текст сообщения html-тегов. Некоторые разработчики, наверное, скажут: "а мы все-таки очень хотим, чтобы пользователи \_могли\_ вставлять теги". Если сильно неймется - можно сделать некие тегозаменители, типа "текст, окруженный звездочками, будет высвечен bold'ом.". Но никогда не следует разрешать пользователям использование тегов, подразумевающих подключение внешних ресурсов - от тривиального <img> до супернавороченного <bgsound>.

Как-то раз меня попросили потестировать html-чат. Первым же замеченным мной багом было именно разрешение вставки картинок. Учитывая еще пару особенностей строения чата, через несколько минут у меня был файл, в котором аккуратно были перечислены IP-адреса, имена и пароли всех присутствовавших в этот момент на чате пользователей. Как? Да очень просто - чату был послан тег <img src=http://myserver.com/myscript.pl>, в результате чего браузеры всех пользователей, присутствовавших в тот момент на чате, вызвали скрипт myscript.pl с хоста myserver.com. (там не было людей, сидевших под lynx'ом :-) ). А скрипт, перед тем как выдать location на картинку, свалил мне в лог-файл половину переменных окружения - в частности QUERY\_STRING, REMOTE\_ADDR и других. Для каждого пользователя. С вышеупомянутым результатом.

Посему мое мнение - да, разрешить вставку html-тегов в чатах, форумах и гостевых книгах - это красиво, но игра не стоит свеч - вряд ли пользователи пойдут к Вам на книгу или в чат, зная, что их IP может стать известным первому встречному хакеру. Да и не только IP - возможности javascript'a я перечислять не буду :-)

Для примитивной гостевой книги перечисленных средств хватит, чтобы сделать ее более-менее сложной для взлома. Однако для удобства, книги обычно содержат некоторые возможности для модерирования - как минимум, возможность удаления сообщений. Разрешенную, естественно, узкому (или не очень) кругу лиц. Посмотрим, что можно сделать здесь.

Допустим, вся система модерирования книги также состоит из двух частей - страницы со списком сообщений, где можно отмечать подлежащие удалению сообщения, и непосредственно скрипта, удаляющего сообщения. Назовем их соответственно admin1.php и admin2.php.

Простейший и надежнейший способ аутентикации пользователя - размещение скриптов в директории, защищенной файлом [.htaccess](https://htmlweb.ru/php/htaccess.php). Для преодоления такой защиты нужно уже не приложение ломать, а web-сервер. Что несколько сложнее и уж, во всяком случае, не укладывается в рамки темы этой статьи. Однако не всегда этот способ пригоден к употреблению - иногда бывает надо проводить авторизацию средствами самого приложения.

Первый, самый простой способ - авторизация средствами HTTP - через код 401. При виде такого кода возврата, любой нормальный браузер высветит окошко авторизации и попросит ввести логин и пароль. А в дальнейшем браузер при получении кода 401 будет пытаться подсунуть web-серверу текущие для данного realm'а логин и пароль, и только в случае неудачи потребует повторной авторизации. Пример кода для вывода требования на такую авторизацию есть во всех хрестоматиях и мануалах:

if (!isset($PHP\_AUTH\_USER)){

Header("WWW-Authenticate: Basic realm=\"My Realm\"");

Header("HTTP/1.0 401 Unauthorized");

exit;

}

Разместим этот кусочек кода в начале скрипта admin1.php. После его выполнения, у нас будут две установленные переменные $PHP\_AUTH\_USER и PHP\_AUTH\_PW, в которых соответственно будут лежать имя и пароль, введенные пользователем. Их можно, к примеру, проверить по SQL-базе:

**\*\*\* Внимание!!!\*\*\***

*В приведенном ниже фрагменте кода сознательно допущена серьезная ошибка в безопасности. Попытайтесь найти ее самостоятельно.*

$sql\_statement="

select password from peoples where name='$PHP\_AUTH\_USER'";

$result = mysql($dbname, $sql\_statement);

$rpassword = mysql\_result($result,0,'password');

$sql\_statement = "select password('$PHP\_AUTH\_PW')";

$result = mysql($dbname, $sql\_statement);

$password = mysql\_result($result,0);

if ($password != $rpassword) {

Header("HTTP/1.0 401 Auth Required");

Header("WWW-authenticate: basic realm=\"My Realm\"");

exit;

}

Упомянутая ошибка, между прочим, очень распространена среди начинающих и невнимательных программистов. Когда-то я сам поймался на эту удочку - по счастью, особого вреда это не принесло, не считая оставленных хакером в новостной ленте нескольких нецензурных фраз.

Итак, раскрываю секрет: допустим, хакер вводит заведомо несуществующее имя пользователя и пустой пароль. При этом в результате выборки из базы переменная $rpassword принимает пустое значение. А алгоритм шифрования паролей при помощи функции СУБД MySQL Password(), так же, впрочем, как и стандартный алгоритм Unix, при попытке шифрования пустого пароля возвращает пустое значение. В итоге - $password == $rpassword, условие выполняется и взломщик получает доступ к защищенной части приложения. Лечится это либо запрещением пустых паролей, либо, на мой взгляд, более правильный путь - вставкой следующего фрагмента кода:

if (mysql\_numrows($result) != 1) {

Header("HTTP/1.0 401 Auth Required");

Header("WWW-authenticate: basic realm=\"My Realm\"");

exit;

}

То есть - проверкой наличия одного и только одного пользователя в базе. Ни больше, ни меньше.

Точно такую же проверку на авторизацию стоит встроить и в скрипт admin2.php. По идее, если пользователь хороший человек - то он приходит к admin2.php через admin1.php, а значит, уже является авторизованным и никаких повторных вопросов ему не будет - браузер втихомолку передаст пароль. Если же нет - ну, тогда и поругаться не грех. Скажем, вывести ту же фразу "hacker? he-he...".

К сожалению, не всегда удается воспользоваться алгоритмом авторизации через код 401 и приходится выполнять ее только средствами приложения. В общем случае модель такой авторизации будет следующей:

* Пользователь один раз авторизуется при помощи веб-формы и скрипта, который проверяет правильность имени и пароля.
* Остальные скрипты защищенной части приложения каким-нибудь образом проверяют факт авторизованности пользователя.

Такая модель называется сессионной - после прохождения авторизации открывается так называемая "сессия", в течение которой пользователь имеет доступ к защищенной части системы. Сессия закрылась - доступ закрывается. На этом принципе, в частности, строится большинство www-чатов: пользователь может получить доступ к чату только после того, как пройдет процедуру входа. Основная сложность данной схемы заключается в том, что все скрипты защищенной части приложения каким-то образом должны знать о том, что пользователь, посылающий данные, успешно авторизовался.

Рассмотрим несколько вариантов, как это можно сделать:

1. После авторизации все скрипты защищенной части вызываются с неким флажком вида adminmode=1. (Не надо смеяться - я сам такое видел).

Ясно, что любой, кому известен флажок adminmode, может сам сформировать URL и зайти в режиме администрирования. Кроме того - нет возможности отличить одного пользователя от другого.

1. Скрипт авторизации может каким-нибудь образом передать имя пользователя другим скриптам. Распространено во многих www-чатах - для того, чтобы отличить, где чье сообщение идет, рядом с формой типа text для ввода сообщения, пристраивается форма типа hidden, где указывается имя пользователя. Тоже ненадежно, потому что хакер может скачать документ с формой к себе на диск и поменять значение формы hidden. Некоторую пользу здесь может принести вышеупомянутая проверка [HTTP\_REFERER](https://htmlweb.ru/php/function/$_server.php#HTTP_REFERER) - но, как я уже говорил, никаких гарантий она не дает.
2. Определение пользователя по IP-адресу. В этом случае, после прохождения авторизации, где-нибудь в локальной базе данных (sql, dbm, да хоть в txt-файле) сохраняется текущий IP пользователя, а все скрипты защищенной части смотрят в переменную [REMOTE\_ADDR](https://htmlweb.ru/php/function/$_server.php#REMOTE_ADDR) и проверяют, есть ли такой адрес в базе. Если есть - значит, авторизация была, если нет - "Хакер!? Ха-Ха..." :-)

Это более надежный способ - не пройти авторизацию и получить доступ удастся лишь в том случае, если с того же IP сидит другой пользователь, успешно авторизовавшийся. Однако, учитывая распространенность прокси-серверов и IP-Masquerad'инга - это вполне реально.

1. Единственным, известным мне простым и достаточно надежным способом верификации личности пользователя является авторизация при помощи random uid. Рассмотрим ее более подробно.

После авторизации пользователя скрипт, проведший авторизацию, генерирует достаточно длинное случайное число:

mt\_srand((double)microtime()\*1000000);

$uid=mt\_rand(1,1000000);

Это число он:

а) заносит в локальный список авторизовавшихся пользователей;  
б) Выдает пользователю.

Пользователь при каждом запросе, помимо другой информации (сообщение в чате, или список сообщений в гостевой книге), отправляет серверу свой uid. При этом в документе с формами ввода будет присутствовать, наряду с другими формами, тег вида:

<input type=hidden name=uid value=1234567890>

Форма uid невидима для пользователя, но она передается скрипту защищенной части приложения. Тот сличает переданный ему uid с uid'ом, хранящимся в локальной базе и либо выполняет свою функцию, либо... "Хакер?! Ха-ха...".

Единственное, что необходимо сделать при такой организации - периодически чистить локальный список uid'ов и/или сделать для пользователя кнопку "выход", при нажатии на которую локальный uid пользователя сотрется из базы на сервере - сессия закрыта.

Некоторые программисты используют в качестве uid не "одноразовое" динамически генерирующееся число, а пароль пользователя. Это допустимо, но это является "дурным тоном", поскольку пароль пользователя обычно не меняется от сессии к сессии, а значит - хакер сможет сам открывать сессии. Та же самая модель может быть использована везде, где требуется идентификация пользователя - в чатах, веб-конференциях, электронных магазинах.

В заключение стоит упомянуть и о такой полезной вещи, как ведение логов. Если в каждую из описанных процедур встроить возможность занесения события в лог-файл с указанием IP-адреса потенциального злоумышленника - то в случае реальной атаки вычислить хакера будет гораздо проще, поскольку хакеры обычно пробуют последовательно усложняющиеся атаки. Для определения IP-адреса желательно использовать не только стандартную переменную REMOTE\_ADDR, но и менее известную HTTP\_X\_FORWARDED\_FOR, которая позволяет определить IP пользователя, находящегося за прокси-сервером. Естественно - если прокси это позволяет.

При ведении лог-файлов, необходимо помнить, что доступ к ним должен быть только у Вас. Лучше всего, если они будут расположены за пределами дерева каталогов, доступного через WWW. Если нет такой возможности - создайте отдельный каталог для лог-файлов и закройте туда доступ при помощи [.htaccess](https://htmlweb.ru/php/htaccess.php) (Deny from all).

## **register\_globals=On**

Почему опасно оставлять опцию register\_globals включенной?

Что представляет собой **register\_globals**?   
Это опция в php.ini, которая указывает на необходимость регистрации переменных, полученных методом POST или GET в глобальный массив $GLOBALS.

Для ясности приведу пример при **register\_globals=On**.  
Есть файл "index.php" с содержимым:

<? echo $asd.' - локальная переменная<br>'; echo $GLOBALS['asd'].' - ссылка в глобальном массиве $GLOBALS<br>'; echo $\_GET['asd'].' - $\_GET["asd"]'; ?>

В адресной строке напишем: **index.php?asd=123**

Получим:  
123 - локальная переменная   
123 - ссылка в глобальном массиве $GLOBALS  
123 - $\_GET['asd']

Как мы видим, создались 2 переменные: одна локальная (+ ссылка в $GLOBALS), другая в массиве $\_GET. Многие не используют массив $\_GET вообще, они продолжают обрабатывать переменную $asd после получения ее извне.

Но давайте вдумаемся, зачем нам "загрязнять" массив $GLOBALS? Для этого у нас есть специальные массивы, хранящие данные, переданные методами GET (массив $\_GET) и POST (массив $\_POST).

Тот же самый пример, но при **register\_globals=Off**:  
- глобальная переменная  
- ссылка в глобальном массиве $GLOBALS  
123 - $\_GET['asd']

Т.о. не была создана локальная переменная и для манипулирования с "$asd" мы должны использовать массив $\_GET.

Возможно, уже сейчас вы изменили свое мнение о register\_globals. Вероятно, вам придется, что-то переписать в своих программах, но оно того стоит.

А теперь я расскажу вам, как взломщик может воспользоваться этой опцией в своих целях, т.е. при register\_globals=On.

Начну от простого к сложному.   
Часто мы видим предупреждения:   
Notice: Undefined variable: asd(название переменной) in \*\*\*\*  
Что это значит? Это значит, что переменная $asd не была определена явно.   
Например, некоторые люди балуются подобным:

<?

for($i= 0;$i<10;$i++)

{

@$asd.=$i;

}

echo $asd

?>

Т.е. не определив переменную, сразу начинают ее использовать. Приведенный код по идее не страшен, но задумайтесь, а вдруг эта самая переменная $asd, в последствии записывается в файл? Например, напишем следующее в строке адреса: "index.php?asd=LUSER+" и получим: "LUSER 0123456789". Ну разве приятно будет увидеть такое? Не думаю.

Предположим мы пишем систему аутентификации пользователя:

<?

if($\_POST['login']=='login'&&$\_POST['pass']=='pass')

{

$valid\_user=TRUE; // Юзер корректный

}

if($valid\_user)

{

echo 'Здравствуйте, пользователь';

}

else echo 'В доступе отказано';

?>

Привел я заведомо дырявую систему, стоит нам только написать в адресной строке "index.php?valid\_user=1" и мы получим надпись "Здравствуйте, пользователь"

Этого бы не случилось, если бы мы написали так:

<?

if($\_POST['login']=='login'&&$\_POST['pass']=='pass')

{

$valid\_user=TRUE; // Юзер корректный

}

else $valid\_user=FALSE;

if($valid\_user)

{

echo 'Здравствуйте, пользователь';

}

else echo 'В доступе отказано'

?>

Т.е. сами определили переменную $valid\_user, как FALSE в случае неудачи.

Продолжим далее:   
Теперь использование функции [isset()](https://htmlweb.ru/php/function/isset.php) становиться небезопасно, т.к. любой может подменить переменную на угодную ему.

Приведу пример с sql-инъекцией:

<?

if(@$some\_conditions) // некоторые условия

{

$where='id=3';

}

echo $query='SELECT id, title, description FROM table '.

'WHERE '.(IsSet($where)?$where:'id=4')

?>

В адресной строке напишем :

"index.php?where=id=0 + UNION + ALL + SELECT + login, + password, + null + FROM + admin + where + login='admin'",

получим sql-инъекцию:

SELECT id, title, description FROM table WHERE id=0

UNION ALL SELECT login, password, null

FROM admin where login='admin'

И взломщик получает ваши явки и пароли.

Как вы видите, все примеры имеют дыры в защите, которые можно эксплуатировать через включенный register\_globals.

Справиться с подобным можно, если всегда определять переменную вне зависимости от условий. Или же использовать инкапсуляцию переменных в функциях, т.е. когда вы определяете функцию, то переменные внутри нее будут закрыты извне, например:

<?

function asd()

{

// Какие то действия

if(IsSet( $where )) {

echo $where ;

}

else echo '$where не существует' ;

}

asd();

?>

Теперь, если мы напишем в адресной строке: **index.php?where=123**, то получим: *"$where не существует"*.  
Но это при условии, что вы не устанавливаете переменную $where как глобальную, т.е. "global $where"

Если Вы установите опцию register\_globals=Off и попробуете заново все приведенные выше примеры, то убедитесь, что они не сработают.

Это можно сделать как в php.ini, но большинство хостинг провайдеров вам это не позволят, потому придется воспользоваться файлом [.htaccess](https://htmlweb.ru/php/htaccess.php)

Создаем файл с названием: .htaccess и запишем в него:

php\_flag register\_globals off

### **Защита от вредныз url-запросов**

Следующий PHP-код проверяет запрос на потенциальные инъекции

if (strlen($\_SERVER['REQUEST\_URI']) > 255 ||

strpos($\_SERVER['REQUEST\_URI'], "eval(") ||

strpos($\_SERVER['REQUEST\_URI'], "CONCAT") ||

strpos($\_SERVER['REQUEST\_URI'], "UNION+SELECT") ||

strpos($\_SERVER['REQUEST\_URI'], "base64")) {

header("HTTP/1.1 414 Request-URI Too Long");

    header("Status: 414 Request-URI Too Long");

    header("Connection: Close");

    exit;

}

Защититься также можно с помощью [.htaccess](https://htmlweb.ru/php/htaccess.php):

Options +FollowSymLinks

RewriteEngine On

RewriteCond %{QUERY\_STRING} (\<|%3C).\*script.\*(\>|%3E) [NC,OR]

RewriteCond %{QUERY\_STRING} GLOBALS(=|\[|\%[0-9A-Z]{0,2}) [OR]

RewriteCond %{QUERY\_STRING} \_REQUEST(=|\[|\%[0-9A-Z]{0,2})

RewriteRule ^(.\*)$ index.php [F,L]

# Этот код блокирует использование некоторых XSS-инъекций и попытки модифицировать переменные GLOBALS и\_REQUEST. Сессии в PHP

С самого начала PHP все приняли на ура, но как только на этом языке стали создавать достаточно крупные проекты, разработчики столкнулись с новой проблемой - в PHP отсутствовало понятие глобальных переменных! То есть, выполнялся некий скрипт, посылал сгенерированную страницу клиенту, и все ресурсы, используемые этим скриптом уничтожались. Попробую проиллюстрировать: предположим есть две страницы одного сайта, index.php и dothings.php. Исходники к этим страницам выглядят так:

**index.php**

<?php

$a = "Меня задали на index.php";

?>

<html><body>

<?php

echo $a;

?>

</body></html>

**dothings.php**

<html><body>

<?php

echo $a;

?>

</body></html>

Если выполнить эти два скрипта, то на первой странице мы увидим надпись "Меня задали на index.php", а вторая страница будет пустой.

Разработчики web-сайтов, недолго думая, стали использовать cookie для хранения глобальных переменных на стороне клиента. Процесс выглядел примерно так: пользователь приходит на главную страницу сайта, делает какие-то действия, и вся информация, связанная с этим пользователем, которая может потребоваться на других страницах сайта, будет храниться у него в браузере в виде cookie. Этот метод имеет довольно серьезные минусы, из-за которых от PHP в своё время отвернулось немало разработчиков. Например, нам нужно авторизовать пользователя, чтобы разрешить ему доступ к закрытым (или принадлежащим только ему) разделам сайта. Придется отправлять пользователю cookie, который будет служит его последующим идентификатором на сайте. Такой подход становится очень громоздким и не удобным, как только сайт начинает собирать всё больше и больше сведений о поведении пользователя, ведь всю информацию, посылаемую пользователю, желательно кодировать, чтобы её нельзя было подделать. Ещё совсем недавно подделкой cookie можно было "уложить" не один чат, а порой и пробраться в чужую почту. К тому же есть ещё на свете странные люди, у которых браузер cookie не поддерживает.

При использовании сессий вся информация хранится не на стороне клиента, а на стороне сервера, и потому лучше защищена от манипуляций злоумышленников. Да и работать с сессиями куда проще и удобнее, так как все данные автоматически проходят через алгоритмы криптографии модуля PHP. В броузере клиента, лишь хранится уникальный идентификатор номера сессии, либо в форме cookie, либо в виде переменной в адресной строке броузера, какой из двух способов использовать для передачи идентификатора сессии между страницами интерпретатор PHP выбирает сам. Это на 100% безопасно, так как идентификатор сессии уникален, и подделать его практически невозможно (об этом чуть далее, в разделе [о безопасности сессий](https://htmlweb.ru/php/php_session.php#protect)).

Я не буду вдаваться в технологические вопросы устройства механизма работы сессий, а только опишу, как правильно работать с сессиями в PHP.

#### Как работать с сессиями?

Если вы будете тестировать примеры из статьи (или ваши скрипты) на каком-либо коммерческом хостинге, проблем с работой с сессиями быть не должно. Если же вы сами настраивали ваш сервер (будь то реальный сервер, или эмулятор), могут появляться ошибки примерно такого содержания:

"Warning: open(/var/state/php/sess\_6f71d1dbb52fa88481e752af7f384db0, O\_RDWR) failed: No such file or directory (2)".

Это значит всего лишь, что у вас неправильно настроен PHP. Решить эту проблему можно, прописав правильный путь (на существующую директорию) для сохранения сессий в файле php.ini и перезапустить сервер.

Любой скрипт, который будет использовать переменные (данные) из сессий, должен содержать следующую строчку:

session\_start();

Эта команда говорит серверу, что данная страница нуждается во всех переменных, которые связаны с данным пользователем (браузером). Сервер берёт эти переменные из файла и делает их доступными. Очень важно открыть сессию до того, как какие-либо данные будут посылаться пользователю; на практике это значит, что функцию [session\_start()](https://htmlweb.ru/php/function/session_start.php) желательно вызывать в самом начале страницы, например так:

session\_start();

?>

<html>

<head>

</head>

...

Для задания директории в которой будут сохраняться файлы сессий используется функция [session\_save\_path()](https://htmlweb.ru/php/function/session_save_path.php):

session\_save\_path($\_SERVER['DOCUMENT\_ROOT'].'/session');

session\_start();

После начала сессии можно задавать глобальные переменные. Ари присвоении какого-либо значения любому полю массива $\_SESSION, переменная с таким же именем автоматически регистрируется, как переменная сессии. Этот массив доступен на всех страницах, использующих сессию. Для примера разберем програму:

**index.php**

<?php

// открываем сессию

session\_start();

// задаём значение переменной

$\_SESSION['a'] = "Меня задали на index.php";

?>

<html>

<body>

Всё ОК. Сессию загрузили!

Пройдём, посмотрим что <a href="dothings.php">там:</a>

</body>

</html>

**dothings.php**

<?php

// открываем сессию

session\_start();

?>

<html>

<body>

<?php

echo $\_SESSION['a'];

?>

</body>

</html>

При последовательном запуске этих файлов, первый скрипт "index.php" выдаст следующий результат:

*Всё ОК. Сессию загрузили! Пройдём, посмотрим что там:*

А второй "dothings.php" вот это:

*Меня задали на index.php*

Переменная $a теперь доступна на всех страницах данного сайта, которые запустили сессии.

Другие полезные функции и приемы для работы с сессиями:

* **unset($\_SESSION['a'])** - сессия "забывает" значение заданной сессионой переменной;
* [**session\_destroy**](https://htmlweb.ru/php/function/session_destroy.php)**()** - сессия уничтожается (например, если пользователь покинул систему, нажав кнопку "выход");
* [**session\_set\_cookie\_params**](https://htmlweb.ru/php/function/session_set_cookie_params.php)**(int lifetime [, string path [, string domain]])** - с помощью этой функции можно установить, как долго будет "жить" сессия, задав unix\_timestamp определяющий время "смерти" сессии. По умолчанию, сессия "живёт" до тех пор, пока клиент не закроет окно браузера.
* [**session\_write\_close**](https://htmlweb.ru/php/function/session_write_close.php)**()** - запись переменных сесии и закрытие ее. Это необходимо для открытия сайта в новом окне, если страница выполняет длительную обработу и заблокировала для вашего браузера файл сессий.

### **Примеры**

Теперь обратимся к практическому применению механизма сессий. Здесь мы рассмотрим пару довольно простых и в то же время полезных примеров.

#### Авторизация Пользователя

Вопросы по авторизации пользователей с помощью PHP-сессий постоянно задаются в конференциях по web-программированию. Механизм авторизации пользователей в системе с помощью сессий довольно хорош с точки зрения безопасности (см.раздел [Безопасность](https://htmlweb.ru/php/php_session.php#protect)).

Наш пример будет состоять из трёх файлов: index.php, authorize.php и secretplace.php. Файл index.php содержит форму, где пользователь введёт свой логин и пароль. Эта форма передаст данные файлу authorize.php, который в случае успешной авторизации допустит пользователя к файлу secretplace.php, а в противном случае выдаст сообщение об ошибке.

Примеры: **index.php**

<html>

<head>

<title>Вводи пароль</title>

</head>

<body>

<form action="authorize.php" method="post">

Логин: <input type="text" name="user\_name"><br>

Пароль: <input type="password" name="user\_pass"><br>

<input type="submit" name="Submit">

</form>

</body>

</html>

**authorize.php**

<?php

// открываем сессию

session\_start();

// данные были отправлены формой?

if($\_POST['Submit']){

// проверяем данные на правильность... в данном случае я

// вписал имя пользователя и пароль прямо в код, целесообразней

// было бы проверить логин/пароль в базе данных и при сов-

// падении дать доступ пользователю...

if(($\_POST['user\_name']=="cleo")&&($\_POST['user\_pass']=="password")){

// запоминаем имя пользователя

$\_SESSION['logged\_user'] = $\_POST['user\_name'];

// и переправляем его на <секретную> страницу...

header("Location: secretplace.php");

exit;

}

}

// если что-то было не так, то пользователь получит

// сообщение об ошибке.

?>

<html><body>

Вы ввели неверный пароль!

</body></html>

**secretplace.php**

<?php

// открываем сессию

session\_start();

/\*

просто зайти на эту страницу нельзя... если

имя пользователя не зарегистрировано, то

перенаправляем его на страницу index.php

для ввода логина и пароля... тут на самом деле

можно много чего сделать, например запомнить

IP пользователя, и после третьей попытки получить

доступ к файлам, его закрыть.

\*/

if(!isset($\_SESSION['logged\_user'])){

header("Location: index.php");

exit;

}

?>

<html>

<body>

Привет, <?php echo $\_SESSION['logged\_user']; ?>, ты на секретной странице!!! :)

</body>

</html>

#### Безопасность

Итак, мы умеем передавать идентификатор от одной страницы (PHP-скрипта) к другой (до следующего вызова с нашего сайта), а значит мы можем различать всех посетителей сайта. Так как идентификатор сессии - это очень большое число (128 бит), шансов, что его удастся подобрать перебором, практически нет. Поэтому злоумышленнику остаются следующие возможности:

* на компьютере пользователя стоит "троян", который ворует номера сессий;
* злоумышленник отлавливает трафик между компьютером пользователя и сервером. Конечно, есть защищенный (зашифрованный) протокол SSL, но им пользуются не все;
* к компьютеру нашего пользователя подошел сосед и стащил номер сессии.

Такие ситуации, основанные на том, что кто-то что-то у кого-то стащит, в общем, не входят в компетенцию программиста. Об этом должны заботиться администраторы и сами пользователи.

Впрочем, PHP очень часто можно "обмануть". Давайте рассмотрим возможные точки взлома в программе авторизации пользователя:

* Файл authorize.php - попытка подбора пароля с помощью стороннего скрипта;
* Файл secretplace.php - попытка обмануть программу путём вписывания значений переменной $logged\_user в адресной строке браузера, например так:   
  "http://www.yoursite.ru/secretplace.php?**logged\_user=hacker**"

Итак, в нашей программе явно видны две "дыры", одна маленькая и не особо заметная, а вот вторая - просто огромная, через которую большинство хакеров и лезет туда, куда не надо.

**Как "залатать" дыру номер 1?**

Не будем писать тонны кода по блокировке IP-адреса и т.п., а просто проверим, откуда приходит запрос, а точнее с какой страницы пришёл запрос, если это будет любая страница с нашего сайта, то всё нормально, а во всех остальных случаях пускать не будем. Подкорректируем файл authorize.php:

**authorize.php V2**

<?php

// открываем сессию

session\_start();

// полный путь к корневой директории где расположены скрипты

$SERVER\_ROOT = "http://localhost/test1/";

// если пользователь пришёл с любой страницы нашего сайта

// то он вроде наш...

// Переменная $HTTP\_REFERER всегда доступна по умолчанию

// и содержит полный адрес ссылающейся страницы...

// функция eregi() проверяет, начинается ли адрес ссылающейся страницы

// со значения в переменной $SERVER\_ROOT

if(preg\_match("/^$SERVER\_ROOT/",$\_SERVER['HTTP\_REFERER'])){

// данные были отправлены формой?

if($\_POST['Submit']){

// далее все как раньше

if(($\_POST['user\_name']=="cleo")&&($\_POST['user\_pass']=="password")){

// запоминаем имя пользователя

$\_SESSION['logged\_user'] = $\_POST['user\_name'];

// и переправляем его на <секретную> страницу...

header("Location: secretplace.php");

exit;

}

}

}

?>

<html><body>

Вы ввели неверный пароль!

</body></html>

**Как избавиться от "дыры" номер 2?**

Предположим, у вас есть сайт, где каждый смертный может зарегистрироваться чтобы добавлять сообщения в форум. Естественно, в форуме у некоторых пользователей (админов, модераторов), возможностей больше чем у других, они, например, могут удалять сообщения других пользователей. Уровень доступа пользователя вы храните в сессии, в переменной $user\_status, где $user\_status = 10 соответствует полному доступу к системе. Пришедшему на сайт злоумышленнику достаточно зарегистрироваться штатным образом, а потом дописать в адресной строке браузера **?user\_status=10**. Вот и завёлся у вас на форуме новый админ!

В принципе, любую переменную скрипта можно задать через адресную строку, просто дописав после полного адреса к скрипту вопросительный знак и название переменной с её значением. Давайте поправим наш код, чтобы этого избежать:

**secretplace.php V2**

<?php

// убираем всё лишнее из адресной строки

// функция unset() <освобождает> переменную

unset($\_SESSION['logged\_user']);

// открываем сессию

session\_start();

/\*

просто зайти на эту страницу нельзя... если

имя пользователя не зарегистрировано, то

перенаправляем его на страницу index.php

для ввода логина и пароля... тут на самом деле

можно много чего сделать, например запомнить

IP пользователя, и после третьей попытки получить

доступ к файлам, его перекрыть.

\*/

if(!isset($\_SESSION['logged\_user'])){

header("Location: index.php");

exit;

}

?>

<html>

<body>

Привет, <?php echo $\_SESSION['logged\_user']; ?>, ты на секретной странице!

</body>

</html>

**Итоги**

Механизм сессий - довольно удачная особенность языка PHP. Сессии просты, очень гибки в использовании. Кстати, есть одна, мало где документированная возможность сессий PHP (доступна начиная с версии 4.0.3) - в сессиях можно хранить не только переменные, но и объекты.

#### Примеры

<?php

// Автоматическая вставка SID в форму.

ini\_set("session.use\_trans\_sid", true);

session\_start();

?>

<form method="post">

</form>

?>

// Автоматическая вставка SID в ссылки.

ini\_set("session.use\_trans\_sid", true);

session\_start();

?>

<body>

<a href="/path/to/something.html">Click here!</a><br>

<a href="/path/to/something.php?param=value">Click here!</a><br>

<a href="http://htmlweb.ru/">Click here!</a><br>

</body>

// Пример работы с сессиями.

session\_start();

// Если на сайт только-только зашли, обнуляем счетчик.

if (!isset($\_SESSION['count'])) $\_SESSION['count'] = 0;

// Увеличиваем счетчик в сессии.

$\_SESSION['count'] = $\_SESSION['count'] + 1;

?>

<h2>Счетчик</h2>

В текущей сессии работы с браузером Вы открыли эту страницу

<?=$\_SESSION['count']?> раз(а).<br>

Закройте браузер, чтобы обнулить счетчик.<br>

<a href="<?=$\_SERVER['SCRIPT\_NAME']?>" target="\_blank">

    Открыть дочернее окно браузера</a>.

// Простой пример использования сессий без Cookies.

session\_name("test");

session\_start();

$\_SESSION['count'] = @$\_SESSION['count'] + 1;

?>

<h2>Счетчик</h2>

В текущей сессии работы с браузером Вы открыли эту страницу

<?=$\_SESSION['count']?> раз(а). <br>

Закройте браузер, чтобы обнулить этот счетчик.<br>

<a href="<?=$\_SERVER['SCRIPT\_NAME']?>?<?=SID?>">Нажмите сюда для обновления страницы!</a>

# Кэширование в PHP

## **Введение**

В старые добрые времена, когда создание web-сайтов представляло из себя такое простое занятие, как набор нескольких HTML-страниц, отправка web-страниц в браузер была простой отправкой файла web-сервером. Посетители сайта могли видеть эти небольшие, исключительно текстовые странички, почти мгновенно (если не считать пользователей медленных модемов). Как только страница была загружена, браузер кэширует её где-нибудь на локальном компьютере, чтобы в случае повторного запроса страницы, можно было взять его локальную версию из кэша, послав лишь короткий запрос, чтобы убедиться, что страница на сервере не была изменена. Запросы обрабатывались быстро и как можно эффективней, и все были счастливы (кроме использующих модемы 9600 бод).

Появление динамических web-страниц изменило положение вещей в худшую сторону, эффективно «сломав» эту модель обслуживания web-страниц благодаря наличию двух проблем:

1. Когда сервером получен запрос динамической web-странички, производится некоторая промежуточная обработка, например синтаксический анализ (парсинг) скрипта движком PHP, которая должна быть завершена. Благодаря этому получаем задержку перед тем, как web-сервер начнёт отправку вывода в браузер. Для простого PHP-скрипта это не существенно, но для более сложного приложения движок PHP может выполнить много действий прежде чем страница будет готова для отправки. Эти дополнительные действия приводят к заметной задержке между запросами пользователей и реальным отображением страниц в их браузерах.
2. Типичный web-сервер, например Apache, использует время модификации файла чтобы правильно сообщить web-браузеру состояние кэша запрашиваемой странички. Для динамических web-страниц, фактически PHP-скрипт может изменяться только изредка, в то время как отображаемый им контент, возможно располагающийся в базе данных, изменяется часто. Web-сервер не имеет возможности знать о наличии изменений в базе данных, тем не менее он не отправляет дату последней модификации. Если клиент (браузер) не получает никакого признака того, как долго данные являются корректными, он предполагает, что в следующий раз необходимо запросить страничку по новой. Web-сервер всегда будет отвечать обновлённой версией странички, независимо от того, изменились ли данные. Чтобы избежать этого недостатка большинство web-разработчиков используют мета-тэги или HTTP-заголовки, чтобы сообщить браузеру никогда не использовать кэшированную версию странички. Однако это отрицает естественную способность web-браузера кэшировать web-страницы и обладает некоторыми существенными недостатками. Например, содержание динамической странички может изменяться раз в сутки, поэтому выгода, получаемая от наличия даже 24-часового кэширования странички браузером, очевидна.

Обычно для маленьких PHP-приложений вполне можно игнорировать существование этих проблем, однако с увеличением сложности и повышением трафика Вашего сайта Вы можете столкнуться с проблемами. Тем не менее, обе эти проблемы могут быть решены, первая путём кэширования на стороне сервера, вторая путём управления кэшированием на стороне клиента из вашего приложения. Подход, который вы будете использовать для решения проблем, будет зависеть от вашей области применения, но в этой главе мы увидим, как вы можете решить обе проблемы используя PHP и некоторые классы библиотекиPEAR.

## **Как я предотвращаю кэширование страницы браузерами?**

Прежде чем мы рассмотрим методы клиентского и серверного кэширования, в первую очередь мы должны понять, как вообще предотвратить кэширование страниц web-браузером (и прокси-серверами). Основной способ достижения этого использует мета-тэги HTML:

<meta http-equiv="Expires" content="Mon, 26 Jul 1997 05:00:00 GMT" />

<meta http-equiv="Pragma" content="no-cache" />

Вставив прошедшую дату в мета-тэг Expires, вы сообщаете браузеру, что кэшированная копия странички всегда является устаревшей. Это значит, что браузер никогда не должен кэшировать страницу. Мета-тэг **Pragma: no-cache** довольно хорошо поддерживаемое соглашение, которому следует большинство web-браузеров. Обнаружив этот тэг, они обычно не кэшируют страницу (хотя никаких гарантий нет, это всего лишь соглашение).

Это хорошо звучит, но есть две проблемы, связанные с использованием мета-тэгов:

1. Если тэг не существовал когда страница была запрошена браузером впервые, но появляется позже (например, вы модифицировали включаемый файл **pageheader.php** который является шапкой каждой web-страницы), браузер останется в блаженном неведении и воспользуется свей кэшированной копей оригинала.
2. Прокси-серверы, кэширующие web-страницы, как например общий ISP, вообще не будет исследовать непосредственно содержимое HTML-документа. Вместо этого они полагаются только на web-сервер, с которого пришли документы, и [протокол HTTP](https://htmlweb.ru/other/http_protocol.php). Иными словами, web-браузер может считать, что не должен кэшировать страницу, но прокси-сервер между браузером и вашим web-сервером вероятно не знает этого – и продолжит отправлять клиенту ту же самую, уже устаревшую, страницу.

Лучший подход состоит в том, чтобы использовать непосредственно протокол HTTP с помощью функции PHP [**header()**](https://htmlweb.ru/php/function/header.php), эквивалентно приведённым выше двум мета-тэгам:

header('Expires: Mon, 26 Jul 1997 05:00:00 GMT');

header('Pragma: no-cache');

Мы можем пойти на один шаг вперёд, воспользовавшись заголовком **Cache-Control**  совместимым с браузерами, поддерживающими HTTP 1.1:

header('Expires: Mon, 26 Jul 1997 05:00:00 GMT');

header('Cache-Control: no-store, no-cache, must-revalidate');

header('Cache-Control: post-check=0, pre-check=0', FALSE);

header('Pragma: no-cache');

Это гарантирует, что никакой web-браузер или промежуточный прокси-сервер не будет кэшировать страницу, таким образом посетители всегда получат самую последнюю версию контента. Фактически, первый заголовок должен быть самодостаточным, это лучший способ гарантировать, что страница не кэшируется. Заголовки **Cache-Control** и **Pragma** добавлены с целью «подстраховаться». Хотя они не работают во всех браузерах или прокси, они отловят некоторые случаи, в которых **Expires** не работает должным образом (например, если дата на компьютере клиента установлена неправильно).

Конечно, полный отказ от кэширования обеспечивает нас проблемами, которые мы обсуждали в начале этой главы. Сейчас мы рассмотрим решение этих проблем.

### **Internet Explorer и кэширование загрузки файлов**

Если при обслуживании загрузки файла PHP-скриптом используются такие заголовки, как например**Content-Disposition: attachment, filename=myFile.pdf** или **Content-Disposition: inline, filename=myFile.pdf** у вас будут проблемы с Internet Explorer’ом, если вы сообщите браузеру не кэшировать страницу.

Internet Explorer оперирует загрузкой довольно необычным образом, выполняя два запроса к web-сайту. Первый запрос загружает файл и сохраняет его в кэше, пока не будет создан второй запрос (без сохранения отклика). Этот запрос вызывает процесс передачи файла конечному пользователю в соответствии с типом файла (например, запускает Acrobat Reader, если файл является PDF-документом). Это значит, что если вы отправили заголовки, запрещающие браузеру кэшировать страницу, Internet Explorerудалит файл между первым и вторым запросом, в  результате чего конечный пользователь ничего не получит. Если файл, который вы отдаёте PHP-скриптом, не изменяется, одним из простейших решений будет убрать «запрещающие кэширование» заголовки из скрипта.

Если загружаемый файл регулярно изменяется (т.е. вы хотите, чтобы браузер загружал новейшую версию), вы должны использовать заголовок **Last-Modified**, который будет рассмотрен в этой главе позднее, и гарантировать, что время модификации между двумя последовательными запросами не изменяется. Вы должны сделать это таким образом, чтобы не повлиять на пользователей браузеров, правильно оперирующих загрузкой. Одним из решений в этом случае будет сохранение файла на вашем web-сервере и предоставление простой ссылку к нему, предоставив web-серверу сообщать за вас заголовки кэширования. Конечно, это решение не может быть приемлемым, если предполагается авторизованный доступ к файлу, это решение допускает непосредственную загрузку сохранённого файла.

## **Как я могу захватить данные на стороне сервера для кэширования?**

Пришло время взглянуть на то, как мы можем уменьшить задержку при помощи кэширования вывода на стороне сервера. Общий подход начинает предоставлять страницу как обычно, выполняя запросы к базе данных и так далее на PHP. Тем не менее, перед отправкой результата в браузер, мы захватываем его и сохраняем готовую страницу, например, в файле. При следующем запросе, PHP-скрипт сначала проверяет наличие кэшированной версии страницы. Если она существует, скрипт отправляет в браузер версию из кэша, исключая таким образом задержку на повторное создание страницы.

### **Несколько слов о кэшировании при помощи шаблонов**

[Шаблонные движки](https://htmlweb.ru/cms.php) часто говорят о кэшировании шаблонов. Обычно эти движки предлагают встроенный механизм для сохранения откомпилированной версии шаблона (т.е. генерируют из шаблона PHP-исходник), что предохраняет нас от необходимости парсить шаблон каждый раз, когда запрашивается страница. Это не нужно путать с кэшированием вывода, которое имеет отношение к кэшированию предоставляемого HTML(или другого  вывода), который посылает PHP в браузер. Вы можете успешно использовать оба типа кэширования одновременно на одном и том же сайте.

Сейчас мы рассмотрим встроенный механизм кэширования на PHP, использующий буферизацию вывода, который может использоваться вами независимо от способа создания контента (с шаблонами или без шаблонов). Рассмотрим ситуацию в которой ваш скрипт отображает результат использую, к примеру, [**echo**](https://htmlweb.ru/php/function/echo.php) или [**print**](https://htmlweb.ru/php/function/print.php), чтобы выдать данные непосредственно в браузер. В таком случае вы можете использовать функции управления выводом PHP для хранения данных в буферной памяти, над которой ваш PHP-скрипт имеет и доступ, и контроль.

##### Пример 1. Файл: 1.php

// Начинаем буферизацию вывода ob\_start(); // Выводим некоторый текст (который сохраняется в буфере); echo '1. Выводим это в буфер  
'; // Получаем содержимое буфера $buffer = ob\_get\_contents(); // Останавливаем буферизацию и очищаем буфер вывода ob\_end\_clean(); // Выводим некоторый текст обычным образом echo '2. Нормальный вывод  
'; // Вывод содержимого буфера echo $buffer; [/php]

Сам буфер хранит вывод как строку. Так, в вышеприведённом скрипте мы начинаем буферизацию с [ob\_start()](https://htmlweb.ru/php/function/ob_start.php) и используем [**echo**](https://htmlweb.ru/php/function/echo.php), чтобы вывести что-либо. Затем мы используем [ob\_get\_contents()](https://htmlweb.ru/php/function/ob_get_contents.php), чтобы выбрать данные, помещённые в буфер оператором **echo**, и сохранить их в строке. Функция [ob\_end\_clean()](https://htmlweb.ru/php/function/ob_end_clean.php)останавливает буферизацию вывода и  уничтожает его содержимое, как альтернативу можно использовать [ob\_end\_flush()](https://htmlweb.ru/php/function/ob_end_flush.php), чтобы вывести содержимое буфера.

Вышеописанный скрипт выведет:

2. Нормальный вывод  
1. Выводим это в буфер

Другими словами, мы захватили вывод первого **echo**, затем послали его браузеру после второго **echo**. Как видно из этого простого примера, буферизация вывода является очень мощным инструментом для формирования вашего сайта, она обеспечивает решение для кэширования, как мы скоро увидим, и является отличным способом скрыть ошибки от посетителей вашего сайта (смотрите [Обработка ошибок](https://htmlweb.ru/php/php_error.php)). Она также обеспечивает альтернативную возможность для переадресации браузера в ситуациях типа аутентификации пользователя.

### **Заголовки HTTP и буферизация вывода**

Буферизация вывода может помочь решить наиболее общую проблему, связанную с функцией[*header()*](https://htmlweb.ru/php/function/header.php), не говоря уже о[*session\_start()*](https://htmlweb.ru/php/function/session_start.php)и[*setcookie()*](https://htmlweb.ru/php/function/setcookie.php). Обычно, если вы вызываете любую из этих функций после того, как начался вывод страницы, вы получите противное сообщение об ошибке. При включенной буферизации вывода единственным типом вывода, избегающим буферизации, являютсяHTTP-заголовки. Используя[***ob\_start()***](https://htmlweb.ru/php/function/ob_start.php)в самом начале выполнения вашего приложения, вы можете посылать заголовки в любой понравившейся точке программы, не сталкиваясь с обычными ошибками. Затем, как только вы будете уверены, что больше выводитьHTTP-заголовки не потребуется, вы можете сразу же вывести содержимое страницы из буфера.

(прим. переводчика: следует заметить что подобное использование данной функции крайне неоправдано. В большинстве случаев необходимости в использовании буферизации вывода для избавления ошибок указанного типа просто не существует и всё с лёгкостью может быть исправлено правильным проектированием приложения)

### **Использование буферизации вывода для кэширования на стороне сервера**

Вы уже видели базовый пример буферизации вывода, теперь следующий шаг, в котором буфер сохраняется в файл:

##### Пример 2. Файл: 2.php

// Если существует кэшированная версия:

if (file\_exists('./cache/2.cache')) {

// Читаем и выводим файл

readfile('./cache/2.cache');

exit();

}

// Начинаем буферизацию вывода

ob\_start();

// Выводим остальной HTML

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Кэшированная страница</title>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />

</head>

<body>

Эта страница кэшируется средствами PHP

<a href="http://www.php.net/outcontrol">Функции управления выводом</a>

</body>

</html>

<?

// Получаем содержимое буфера

$buffer = ob\_get\_contents();

// Останов буферирования и вывод буфера

ob\_end\_flush();

// Сохранение кэш-файла с контентом

$fp = fopen('./cache/2.cache', 'w');

fwrite($fp, $buffer);

fclose($fp);

?>

Сначала вышеописанный скрипт проверяет наличие существования версии странички в кэше, и, если она имеется, скрипт читает и выводит её. В противном случае, он использует буферизацию вывода для создания версии страницы в кэше. Она сохраняется как файл, после использования [ob\_end\_flush()](https://htmlweb.ru/php/function/ob_end_flush.php) для отображения страницы пользователю.

Файл 2.cache содержит точную копию HTML, которую предоставляет скрипт:

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Кэшированная страница</title>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />

</head>

<body>

Эта страница кэшируется средствами PHP

<a href="http://www.php.net/outcontrol">Функции управления выводом</a>

</body>

</html>

### **Блочная буферизация**

Упрощённый подход кэширует выводимый буфер как одну страницу. Однако этот подход лишает вас реальных возможностей, предоставляемых функциями управления выводом PHP, улучшающих производительность вашего сайта методом соответственно различающихся сроков жизни вашего контента.

Вне всякого сомнения, некоторые части отправляемой вами посетителю страницы изменяются очень редко, например, такие как шапку, меню и нижний колонтитул. Однако другие части, типа таблиц, содержащих обсуждения в форуме, могут изменяться довольно часто. Буферизация вывода может использоваться к кэшированию разделов страницы в отдельных файлах, затем создавать из них страницу – решение, устраняющее необходимость повторных запросов к базе данных, циклов while и т.д. Вы можете назначать каждому блоку страницы дату истечения срока, после которой пересоздаётся кэш-файл, или кроме того, вы можете включить в ваше приложение механизм, который будет удалять кэш-файл каждый раз, когда сохранённый в нём контент изменён.

Вот пример, демонстрирующий этот принцип:

##### Пример 3. Файл: 3.php

<?

/\* Запись кэш-файла

\* @param string contents - содержание буфера

\* @param string filename - имя файла, используемое при создании кэш-файла

\* @return void

\*/

function writeCache($content, $filename) {

$fp = fopen('./cache/' . $filename, 'w');

fwrite($fp, $content);

fclose($fp);

}

/\* Проверка кэш-файлов

\* @param string filename - имя проверяемого кэш-файла

\* @param int expiry - максимальный "возраст" файла в секундах

\* @return mixed содержимое кэша или false

\*/

function readCache($filename, $expiry) {

if (file\_exists('./cache/' . $filename)) {

if ((time() - $expiry) > filemtime('./cache/' . $filename))

return FALSE;

$cache = file('./cache/' . $filename);

return implode('', $cache);

}

return FALSE;

}

?>

Первые две определённые нами функции, **writeCache** и **readCache**, используются соответственно для создания кэш-файлов и проверки их существования. Функция **writeCache** получает данные для кэширования в первом аргументе, и имя файла, используемое при создании кэш-файла. Функция**readCache** получает имя кэш-файла в первом параметре, вместе со временем в секундах, после которого кэш-файл должен считаться устаревшим. Если она сочтёт кэш-файл допустимым, скрипт вернёт его содержимое, в противном случае он вернёт **FALSE**, чтобы показать, что-либо кэш-файла не существует, либо он устарел.

В этом примере я использовал процедурный подход. Однако я не советую делать это на практике, поскольку это закончится очень грязным кодом (смотри последующие решения с лучшей альтернативой) и, вероятно, вызовет проблемы с блокировкой файла (например, что случится, когда кто-то обращается к кэшу в момент его обновления?).

Давайте продолжим этот пример. После того, как запущена буферизация вывода, начинается обработка. Сначала скрипт вызывает **readCache**, чтобы узнать, существует ли файл **3\_header.cache**, он содержит шапку страницы – заголовок HTML и начало тела. Мы используем функцию [date()](https://htmlweb.ru/php/function/date.php) чтобы вывести время, когда страница фактически была сгенерирована, таким образом вы увидите различные кэш-файлы в работе, когда страница будет отображена.

##### Пример 4. Файл: 3.php

<?

// Начинаем буферизацию вывода

ob\_start();

// Обработка шапки

if (!$header = readCache('3\_header.cache', 604800)) {

// Вывод шапки

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Страница, кэшированная поблочно</title>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />

</head>

<body>

Время создания шапки: &lt;?=date('H:i:s'); ?> <br />

<?

$header = ob\_get\_contents();

ob\_clean();

writeCache($header,'3\_header.cache');

}

?>

Что же случается когда кэш-файл не найден? Выводится некоторый контент и присваивается переменной при помощи [ob\_get\_contents()](https://htmlweb.ru/php/function/ob_get_contents.php), после чего буфер очищается функцией [ob\_clean()](https://htmlweb.ru/php/function/ob_clean.php). Это позволяет нам перехватывать вывод по частям и сопоставлять их с индивидуальными кэш-файлами при помощи**writeCache**. Заголовок страницы теперь хранится как файл, который может быть использован без нашего вмешательства в пересборку страницы. Давайте вернёмся на секунду к началу условного оператора. Когда мы вызывали **readCache**, мы передали ей время жизни кэша в 604800 секунд (одна неделя), **readCache**использует время модификации кэш-файла, чтобы определить, является ли кэш-файл всё ещё  допустимым.

Для содержимого (тела) страницы мы по прежнему будем использовать тот же процесс. Однако на сей раз при вызове **readCache** мы будем использовать время жизни кэша в пять секунд, кэш-файл будет модифицироваться каждый раз, когда он «старше» 5 секунд:

##### Пример 5. Файл: 3.php

<?

// Обработка тела страницы

if (!$body = readCache('3\_body.cache', 5)) {

echo 'Время создания тела: ' . date('H:i:s') . '<br />';

$body = ob\_get\_contents();

ob\_clean();

writeCache($body, '3\_body.cache');

}

?>

Нижний колонтитул эффективно изменять так же, как заголовок. После этого буферизация вывода останавливается и отображается содержимое трёх переменных, содержащих данные страницы:

##### Пример 6. Файл: 3.php (окончание)

<?

// Обработка нижнего колонтитула страницы

if (!$footer = readCache('3\_footer.cache', 604800)) {

?>

Время создания нижнего колонтитула: &lt;?=date('H:i:s'); ?> <br />

</body>

</html>

<?

$footer = ob\_get\_contents();

ob\_clean();

writeCache($footer, '3\_footer.cache');

}

// останавливаем буферизацию

ob\_end\_clean();

// Выводим содержимое страницы

echo $header . $body . $footer;

?>

Конечный результат выглядит примерно так:

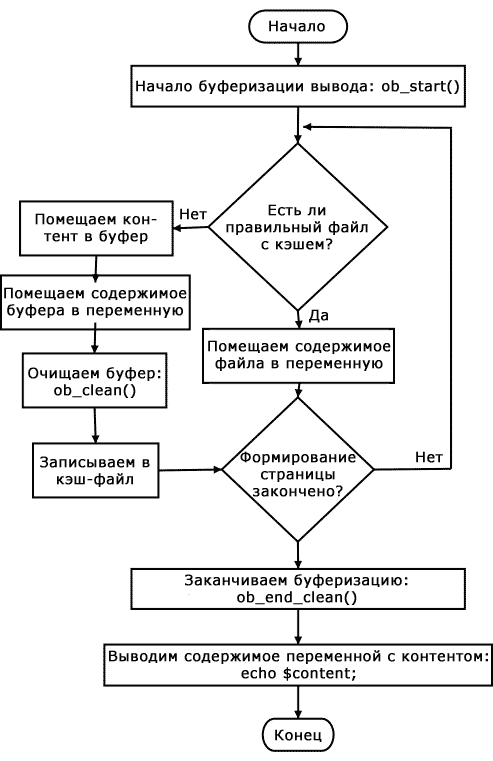
Время создания шапки: 17:10:42

Время создания тела: 18:07:40

Время создания нижнего колонтитула: 17:10:42

Заголовок и нижний колонтитул обновляются еженедельно, в время как тело модифицируется, когда оно старее 5 секунд.

Блок-схема на рисунке суммирует методологию блочной буферизации.

  
Блок-схема блочной буферизации вывода

### **Вложенные буферы**

Вы можете вкладывать один буфер в другой фактически до бесконечности, просто вызвав [**ob\_start()**](https://htmlweb.ru/php/function/ob_start.php)неоднократно. Это может быть полезным, если у вас имеется множество операций, использующих  буфер вывода, например, одни перехватывают сообщения PHP об ошибках, другие имеют дело с кэшированием. Вы должны удостовериться, что [**ob\_end\_flush()**](https://htmlweb.ru/php/function/ob_end_flush.php) или [**ob\_end\_clean()**](https://htmlweb.ru/php/function/ob_end_clean.php) вызываются каждый раз, когда используется [**ob\_start()**](https://htmlweb.ru/php/function/ob_start.php).

## **Как мне управлять кэшированием на стороне клиента средствами PHP?**

Пришло время посмотреть на механизм, который позволит нам контролировать кеш на стороне клиента средствами PHP. Этот подход будет работать только если вы используете PHP в связке с сервером Apache, поскольку мы будем использовать функцию [getallheaders()](https://htmlweb.ru/php/function/getallheaders.php), чтобы получить заголовки, передаваемые браузером. Эта функция работает только в Apache.

Новые имена функций

Если вы используетеPHP4.3.0 сApache, HTTP-заголовки доступны функцией[*apache\_request\_headers()*](https://htmlweb.ru/php/function/apache_request_headers.php)и[*apache\_response\_headers()*](https://htmlweb.ru/php/function/apache_response_headers.php). Функция[*getallheaders()*](https://htmlweb.ru/php/function/getallheaders.php)стала псевдонимом для новой функции[*apache\_request\_headers()*](https://htmlweb.ru/php/function/apache_request_headers.php).

Механизмом для работы с кэшем web-браузера вновь является HTTP. Множество заголовков вовлечёны в инструктирование web-браузеров и прокси-серверов независимо кэшировать страницу, ситуация осложняется тем фактом, что некоторые из них доступны только с HTTP 1.1.

Проверка HTTP-заголовков в вашем браузере

Простым но очень удобным инструментом для проверки заголовков запросов и откликов являетсяLiveHttpHeaders– аддон к браузеруMozilla. Необходимо точно знать, какие заголовки посылает ваш скрипт, особенно когда вы имеете дело с заголовками кэшированияHTTP.

Для простоты мы рассмотрим только заголовки кэширования HTTP 1.0, а именно **Expires**, **Last-Modified** и**If-Modified-Since**, а также статус-код **HTTP 304 (Not Modified)**.

Другие заголовки, доступные с HTTP 1.1, например **Cache-Control** и **ETag**, предназначены для обеспечения расширенного механизма, который может использоваться совместно с состоянием web-сессии, иными словами, версия данной страницы, отображаемой неавторизованному посетителю, может значительно отличаться от отображаемой авторизованному пользователю. Заголовки HTTP 1.1 изначально добавлялись для того, чтобы позволить кэшировать такие страницы.

### **Истечение срока жизни страницы**

Самым простым в использовании заголовком является заголовок **Expire**, который устанавливает дату (возможно, будущую), когда страница устареет. До этого момента web-браузеру разрешается использовать кэшированную версию страницы.

##### Пример 7. 6.php

<?

/\*\*

\* Посылает заголовок Expires HTTP 1.0.

\* @param int $expires - количество секунд до времени истечения срока жизни

\*/

function setExpires($expires){

header('Expires: ' .

    gmdate('D, d M Y H:i:s', time() + $expires) . 'GMT');

}

// Устанавливаем заголовок времени истечения срока жизни Expires

setExpires(10);

// Отображаем

echo 'Эта страница самоуничтожится через 10 секунд&lt;br /&gt;';

echo 'Сейчас ' . gmdate('H:i:s') . ' GMT&lt;br /&gt;';

echo '&lt;a href="' . $\_SERVER['PHP\_SELF'] .

    '"&gt;Посмотреть вновь &lt;/a&gt;&lt;br /&gt;';

?>

Функция **setExpires** отправляет заголовок HTTP **Expires** с будущим временем, заданном в секундах. Вышеприведённый пример показывает текущее время по Гринвичу и выводит ссылку, которая вам позволяет перейти на страницу вновь. Используя кнопку Refresh вашего браузера, вы можете сообщить браузеру о желании обновить кэш. Используя ссылку, вы увидите, что время изменяется только раз в 10 секунд.

Даты и время в HTTP

Даты в HTTP всегда вычисляются относительного меридиана времени Гринвича (GMT). Функция PHP[*gmdate()*](https://htmlweb.ru/php/function/gmdate.php)точно такая же функция, как[*date()*](https://htmlweb.ru/php/function/date.php), за исключением того, что она автоматически компенсирует время по Гринвичу, основанное на системных часах и настройках региона вашего сервера.

Когда браузер сталкивается с заголовком **Expires**, он кэширует страницу. Все последующие запросы страницы, сделанные до указанного времени истечения срока жизни, используют версию страницы из кэша, никаких запросов к web-серверу при этом не происходит.

Заголовок **Expires** преимущественно прост в реализации, но в большинстве случаев, если вы не высокоорганизованный человек, вы не можете знать точно, когда данная страница вашего сайта обновлена. Поскольку браузер войдёт в контакт с сервером только после того, как страница устареет, нет ни одного способа сообщить браузеру, что страница, находящаяся в его кэше, устарела. Вы также теряете некоторую часть трафика к вашему web-сайту, поскольку браузер не обращается к серверу при запросе страницы из кэша.

### **Время изменения страницы**

Более практично использовать заголовки **Last-Modified** и **If-Modified-Since**, доступные в HTTP 1.0. Технически он известно как выполнение условного GET-запроса, вы возвращаете любой контент, основываясь на условии пришедшего заголовка запроса **If-Modified-Since**.

При использовании этого метода вы должны отправлять заголовок **Last-Modified** каждый раз, когда обращаются к вашему PHP-скрипту. При следующем запросе страницы браузером, он отправит заголовок **If-Modified-Since**, содержащий время, по которому ваш скрипт может определить, обновлялась ли страница со времени последнего запроса. Если это не так, ваш скрипт посылает код статуса **HTTP 304**, чтобы указать, что страница не изменялась, не выводя при этом содержимого страницы.

Устанавливаем время модификации кэш-файла этой строкой:

$lastModified = filemtime($cache\_file);

Затем, используя время модификации кэш-файла, мы посылаем заголовок **Last-Modified**. Нам нужно посылать её для каждой предоставляемой страницы, чтобы вынудить браузер посылать нам заголовок **If-Modified-Since** с каждым запросом.

// Выдаём заголовок HTTP Last-Modified

header('Last-Modified: ' .

    gmdate('D, d M Y H:i:s', $lastModified) . ' GMT');

Использование функции [getallheaders()](https://htmlweb.ru/php/function/getallheaders.php) обеспечивает нам получение от PHP всех входящих заголовков в виде массива. Затем мы должны проверить, что заголовок **If-Modified-Since** действительно существует, если он есть, мы должны обработать специальный случай старых версий Mozilla (ниже 6й версии), который добавлял в конец дополнительное поле к заголовку **If-Modified-Since**. Используя функцию PHP [strtotime()](https://htmlweb.ru/php/function/strtotime.php), мы получаем таймштамп даты, переданной нам браузером. Если такого заголовка нет, мы присваиваем таймштампу ноль, вынуждая таким образом PHP отдать посетителю последнюю версию страницы.

<?

// Получаем заголовки запроса клиента - только для Apache

$request = getallheaders();

if (isset($request['If-Modified-Since'])) {

// Разделяем If-Modified-Since (Netscape &lt; v6 отдаёт их неправильно)

$modifiedSince = explode(';', $request['If-Modified-Since']);

// Преобразуем запрос клиента If-Modified-Since в таймштамп

$modifiedSince = strtotime($modifiedSince[0]);

} else {

// Устанавливаем время модификации в ноль

$modifiedSince = 0;

}

?>

Наконец, мы проверяем, был ли модифицирован кэш с тех пор как посетитель получал эту страницу в последний раз. Если это не так, мы просто посылаем в заголовке отклик **Not Modified** и прекращаем выполнение скрипта, не нагружая канал передачи данных и экономя процессорное время, инструктируя браузер отобразить кэшированную версию страницы.

<?

// Сравниваем время последней модификации контента с кэшем клиента

if ($lastModified <= $modifiedSince) {

    // Разгружаем канал передачи данных!

    header('HTTP/1.1 304 Not Modified');

    exit();

}

?>

И в заключение готовый кусок кода, позволяющий кешировать все статические ваши страницы.

**index.php** - через него загружаются все файлы с сервера. Настройка сделана с помощью [.htaccess](https://htmlweb.ru/php/htaccess.php)

<?

$url = $\_SERVER['REQUEST\_URI'];

if($url==""||$url=="/")

$url="index.htm" ;

else $url=substr($url, 1);

if ( !file\_exists( $url ) )

    { header("HTTP/1.0 404 Not Found"); exit();}

// Получаем время последней модификации кэш-файла

$lastModified = max(filemtime($url),filemtime('index.php'));

$slastModified = gmdate('D, d M Y H:i:s', $lastModified) . ' GMT';

// Выдаём заголовок HTTP Last-Modified

header('Last-Modified: ' . $slastModified );

// Получаем заголовки запроса клиента - только для Apache

$headers = getallheaders();

if (isset($headers['If-Modified-Since'])) {

// Разделяем If-Modified-Since (Netscape < v6 отдаёт их неправильно)

$modifiedSince = explode(';', $headers['If-Modified-Since']);

// Преобразуем запрос клиента If-Modified-Since в таймштамп

$modifiedSince = strtotime($modifiedSince[0]);

// Сравниваем время последней модификации контента с кэшем клиента

if ($lastModified <= $modifiedSince)

    // Разгружаем канал передачи данных!

    { header('HTTP/1.1 304 Not Modified'); exit(); }

}

echo "<DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 Transitional//EN\">\n<HTML>\n";

echo "<HEAD>\n";

echo "<META HTTP-EQUIV=\"Content-Type\" CONTENT=\"text/html; charset=windows-1251\">\n";

echo "<meta http-equiv=\"Last-Modified\" content=\"".$slastModified."\"&gt;\n";

?>

Если вы объедините подход времени последнего изменения со значением времени, являющимся уже доступным в вашем приложении (например, время самой последней новостной статьи, или время устаревания из системы серверного кэширования, которое мы видели в последнем решении), вы сможете воспользоваться преимуществами кэша web-браузера и разгрузите канал передачи данных, по возможности сэкономив информационный трафик с вашего сайта и улучшив его производительность.

Будьте осторожны при тестировании любого кэширования, выполненного в таком стиле, если вы сделаете это неправильно, вы можете заставить ваших посетителей всегда иметь устаревшие копии вашего сайта.

## **Кэширование ваших страниц в 5 шагов**

Оригинал: [Posted in PHP / JavaScript by ibzi on the February 17th, 2007](http://www.ibzi.net/)   
Перевод: Кузьма Феськов ([kuzma@russofile.ru](mailto:kuzma%40russofile.ru),[http://kuzma.russofile.ru](http://php.russofile.ru/)) 

Кэширование ваших страниц может оказаться красивым и полезным механизмом, особенно, если они генерируются средствами PHP и делают множество SQL запросов. Как только вы примените кэширование, ваш сервер тут же снизит нагрузку и перестанет съедать много памяти на генерацию страниц - он просто будет загружать их из КЭШа. Я покажу вам, как PHP может кэшировать страницы и, в дальнейшем, вы сможете тратить на это минут 5.

Расмотрим технологию кэширования пошагам:

1. В домашней директории создаем файлы **.htaccess**, **start\_cache.php**, **end\_cache.php**, а также папку с названием **cache\_files**.
2. Папке **cache\_files** необходимо проставить атрибуты **777**.
3. Внутри **.htaccess** файла пропишите следующие строки:
4. php\_value auto\_prepend\_file
5. /home/username/public\_html/start\_cache.php
6. php\_value auto\_append\_file
7. /home/username/public\_html/end\_cache.php

Строку /home/username/public\_html/ необходимо заменить на путь к вашей домашней директории.

1. В скрипт **start\_cache.php** помещаем следующий код:
2. <?php
3. // раздел настроек, которые вы можете менять
4. $settings\_cachedir = '/home/username/public\_html/cache\_files/';
5. $settings\_cachetime = 3600; //время жизни кэша (1 час)
6. // код
7. $thispage = 'http://' .
8. $\_SERVER['HTTP\_HOST'] . $\_SERVER['REQUEST\_URI'];
9. $cachelink = $settings\_cachedir.md5($thispage).".html";
10. if (file\_exists($cachelink)) {
11. $cachelink\_time = filemtime($cachelink);
12. if ((time() - $settings\_cachetime) < $cachelink\_time) {
13. readfile($cachelink);die();
14. }
15. }
16. ob\_start();
17. ?>

Не забывайте исправлять путь /home/username/public\_html/ на путь к вашей домашней директории.

1. А следующий код поместите в скрипт **end\_cache.php**:
2. <?php
3. $fp = fopen($cachelink, 'w');
4. fwrite($fp, ob\_get\_contents());
5. fclose($fp);
6. ob\_end\_flush();
7. ?>

Все ваши страницы будут кэшироваться на 3600 секунд = 1 час. Этот параметр вы легко можете поменять в скрипте **start\_cache.php**. Кэш страниц будет сохранен в папке **cache\_files**.

Совершенно очевидно, что в данном случае атрибуты **777** являются определенным нарушением безопасности. В связи с чем, рекомендую вынести папку **cahce\_files** за пределы **public\_html**, например, поместить ее на один уровень выше. Это закроет доступ к находящимся в ней файлам пользователей вашего сайта, но никак не повлияет на работоспособность системы.

Также, у данного метода есть еще один серьезный недостаток: автор статьи складывает весь кэш в одну папку, что, при достаточном количестве страниц на вашем сайте, вызовет проблему, например, в системах Unix наблюдается достаточное замедление работоспособности при наличие в папке более чем 1000 файлов. В связи с чем, в алгоритм необходимо внести ряд изменений и раскладывать файлы по отдельным подпапкам внутри папки **cache\_files**. Например, используя для этого первые 3-4 символа md5 КЭШа.

Для динамических ресурсов вполне возможно выбрать время кэширования в несколько (5-10) секунд или 1-2 минуты, что уже значительно снизит нагрузку на сервер, но не нанесет вреда интерактивности сайта.

Для страниц, для которых особо важна интерактивность, можно ввести исключения в [**.htaccess**](https://htmlweb.ru/php/htaccess.php), что позволит именно им постоянно изменяться, а для остальных страниц можно применять кэширование.

[все функции PHP для буферизации вывода](https://htmlweb.ru/php/function/index.php#буферизация)

## **Регенерация содержания на лету**

Динамически созданные, но статически обслуживаемые страницы, т.е. страницы которые должны передаваться как чисто статические (считываемые из файловой системы и затем передаваемые по запросу), однако они должны быть динамически сгенерированны веб-сервером если они отсутствуют в файловой системе. Таким образом вы можете иметь страницы сгенерированные PHP которые являются статически обслуживаемыми если только кто-либо (либо планировщик) не удалит статическое содержание. В таком случае содержание обновляется.

Это делается следующим набором директив:

RewriteCond %{REQUEST\_FILENAME} !-s

RewriteRule ^page\.html$ page.php [T=application/x-httpd-php,L]

Здесь, запрос к page.html приводит к внутреннему запуску соответствующего page.php, если page.html все-ещё отсутствует или имеет нулевой размер. Фокус здесь в том что page.php это обычный PHP скрипт который в дополнение к собственному выводу, записывает свой вывод в файл page.html. Запустив это один раз, сервер передает данные page.html. Когда вебмастер хочет обновить содержание, он просто удаляет page.html (обычно с помощью cronjob).

## **Проблема с кэшированием страниц у Internet Explorer.**

У IE при работе с заголовком "Vary" встречается одна неприятная ошибочка, связанная с кэшированием страниц. Проблема решается добавлением в .htaccess следующих строк:

BrowserMatch "MSIE" brokenvary=1

BrowserMatch "Mozilla/4.[0-9]{2}" brokenvary=1

BrowserMatch "Opera" !brokenvary

SetEnvIf brokenvary 1 force-no-vary

## **Представление даты и времени**

В распределенных системах, таких, как Интернет, время играет особую роль. Из-за незначительного расхождения системных часов игрок на рынке Forex может потерять десятки тысяч долларов в течение нескольких минут; система деловой разведки ошибется в составлении прогноза; серверы NNTP в процессе синхронизации потеряют важную информацию, нужную пользователю и т.д.

PHP содержит множество функций для работы с датой и временем. Наиболее употребимыми являются:

* **time( )** - возвращает текущее абсолютное время. Это число равно количеству секунд, которое прошло с полуночи 1 января 1970 года (с начала эпохи UNIX).
* **getdate( )** - считывает информацию о дате и времени. Возвращает ассоциативный массив, содержащий информацию по заданному или по текущему (по умолчанию) времени. Массив содержит следующие элементы:

|  |  |
| --- | --- |
| seconds | Секунды (0-59) |
| minutes | Минуты (0-59) |
| hours | Часы (0-23) |
| mday | День месяца (1-31) |
| wday | День недели (0-6), начиная с воскресенья |
| mon | Месяц (1-12) |
| year | Год |
| yday | День года (0-365) |
| weekday | Название дня недели (например, Friday) |
| month | Название месяца (например, January) |
| 0 | Абсолютное время |

#### Пример 1

* <html>
* <head>
* <title>Использование getdate()</title>
* </head>
* <body>
* <?php
* $d = getdate(); // использовано текущее время
* foreach ( $d as $key => $val )
* echo "$key = $val<br>";
* echo "<hr>Сегодня: $d[mday].$d[mon].$d[year]";
* ?>
* </body>
* </html>
* **РЕЗУЛЬТАТ ПРИМЕРА 1:**
* seconds = 21
* minutes = 4
* hours = 12
* mday = 10
* wday = 4
* mon = 11
* year = 2005
* yday = 313
* weekday = Thursday
* month = November
* 0 = 1131613461
* Сегодня: 10.11.2005
* **date( )** - форматирование даты и времени. Аргументы: строка формата и абсолютное время. Второй аргумент необязателен. Возвращает строку с заданной или текущей датой в указанном формате. Строка формата может содержать следующие коды:

|  |  |
| --- | --- |
| a | Включено обозначение "am" или "pm" |
| A | Включено обозначение "AM" или "PM" |
| d | День месяца (01-31) |
| D | Сокращенное название дня недели (три буквы) |
| F | Полное название месяца |
| g | Часы (12-часовой формат без ведущих нулей) |
| G | Часы (24-часовой формат без ведущих нулей) |
| h | Часы (12-часовой формат) |
| H | Часы (24-часовой формат) |
| i | Минуты (00-59) |
| j | День месяца без ведущих нулей (1-31) |
| l | Полное название дня недели |
| L | Признак високосного года (0 или 1) |
| m | Месяц (01-12) |
| M | Сокращенное название месяца (три буквы) |
| n | Месяц (1-12) |
| s | Секунды (00-59) |
| t | Количество дней в данном месяце (от 28 до 31) |
| U | Абсолютное время |
| w | Номер дня недели (0 - воскресенье, 6 - суббота) |
| y | Год (два разряда) |
| Y | Год (четыре разряда) |
| z | День года (0-365) |
| Z | Смещение часового пояса в секундах (от -43200 до 43200) |

* Любая другая информация, включенная в строку формата, будет вставлена в возвращаемую строку. Если в строку формата нужно добавить символы, которые сами по себе являются кодами формата, то перед ними надо поставить обратную косую черту "\". Символы, которые становятся кодами формата при добавлении к ним обратной косой, нужно предварять двумя косыми. Например, если необходимо добавить в строку "n", то надо ввести "\\n", поскольку "\n" является символом новой строки.

#### Пример 2

* <html>
* <head>
* <title>Использование date()</title>
* </head>
* <body>
* <?php
* echo date( "Сегодня d.m.y H:i" );
* ?>
* </body>
* </html>
* **РЕЗУЛЬТАТ ПРИМЕРА 2:**
* Сегодня: 10.11.05 13:03
* **mktime( )** - возвращает абсолютное время, которое затем можно использовать с функциями **date()**или **getdate()**. Принимает до шести целочисленных аргументов в следующем порядке:  
    
  часы  
  минуты  
  секунды  
  месяц  
  день месяца  
  год

#### Пример 3

<html>

<head>

<title>Использование mktime()</title>

</head>

<body>

<?php

echo date( "j of F Y, \a\\t g.i a, l", mktime( 13, 30, 0, 1, 22, 1971 ) );

?>

</body>

</html>

**РЕЗУЛЬТАТ ПРИМЕРА 3:**

22 of January 1971, at 1.30 pm, Friday

* **checkdate( )** - проверка правильности даты. Аргументы: месяц, день, год. Возвращает **true**, если дата правильная, т.е.   
  месяц - целое число от 1 до 12;  
  день - целое число, не превышающее общего количества дней в данном месяце. При этом високосные годы обрабатываются корректно;  
  год - целое число от 1 до 32767.

Например:

if ( !checkdate( $month, 1, $year ) )

{

$d = getdate();

$month = $d[mon];

$year = $d[year];

}

Внимание! Дата может находиться в допустимом диапазоне, но остальные функции работы с датами не примут это значение. Так, нельзя использовать **mktime()** для годов до 1902, а также следует использовать ее осторожно для годов до 1970.

* **strftime( )** - формирование локальной даты и времени. Аргументы: строка формата и абсолютное время. Второй аргумент необязателен. Возвращает строку с заданной или текущей датой в указанном формате. При этом названия месяцев и дней недели извлекается из локали, выбранной с помощью функции [setlocate( )](https://htmlweb.ru/php/php_date.php#local) Строка формата может содержать следующие коды:

|  |  |
| --- | --- |
| %a | Сокращенное название дня недели |
| %A | Полное название дня недели |
| %b | Сокращенное название месяца |
| %B | Полное название месяца |
| %c | Предпочтительный формат даты и времени |
| %C | Номер века |
| %d | День месяца (1-31) |
| %D | То же, что и **%m/%d/%y** |
| %e | Месяц (1-12) |
| %h | То же, что и **%b** |
| %H | Часы (24-часовой формат) |
| %I | Часы (12-часовой формат) |
| %j | День года (0-365) |
| %m | Месяц (1-12) |
| %M | Минуты |
| %n | Символ новой строки |
| %p | Включено обозначение "am" или "pm" |
| %r | Время с использованием a.m./p.m.-нотации |
| %R | Время в 24-часовом формате |
| %S | Секунды (00-59) |
| %t | Символ табуляции |
| %T | То же, что и **%H:%M:%S** |
| %u | Номер дня недели (1 - понедельник, 7 - воскресенье) |
| %U | Номер недели. Отсчет начинается с первого воскресенья года |
| %V | Номер недели по ISO 8601:1988. Первая неделя должна иметь не менее четырех дней, а понедельник считается первым днем |
| %W | Номер недели. Отсчет начинается с первого понедельника года |
| %w | Номер дня недели (0 - воскресенье, 6 - суббота) |
| %x | Предпочтительный формат даты без времени |
| %X | Предпочтительный формат времени без даты |
| %y | Год (два разряда) |
| %Y | Год (четыре разряда) |
| %Z | Часовой пояс (имя или сокращение) |
| %% | Символ "%" |

* Любая другая информация, включенная в строку формата, будет вставлена в возвращаемую строку.

#### Пример 4

* <html>
* <head>
* <title>Использование strftime()</title>
* </head>
* <body>
* <?php
* echo strftime("%A %d %B %Y %H:%M<br>");
* error\_reporting(E\_ALL & ~E\_WARNING);
* setlocale('LC\_ALL','');
* echo strftime("Сегодня %A %d %B %Y %X<br>");
* echo strftime("%Z<br>");
* ?>
* </body>
* </html>
* **РЕЗУЛЬТАТ ПРИМЕРА 4:**
* Tuesday 15 November 2005 13:11
* Сегодня вторник 15 Ноябрь 2005 13:11:14
* Московское время (зима)

#### Еще примеры:

date("l dS of F Y h:i:s A") :

Sunday 12th 2017f March 2017 07:38:40 PM

date("Сегодня d.m.Y") :

Сегодня 12.03.2017

date("Этот файл датирован d.m.Y", filectime(\_\_FILE\_\_)) :

Этот файл датирован 03.02.2016

// Активизируем текущую локаль (иначе дата будет на английском).

setlocale(LC\_ALL, '');

// Выводим 2 предложения.

echo strftime("%B %Y года, %d число.

Был %A, часы показывали %H:%M.");

March 2017 года, 12 число. Был Sunday, часы показывали 19:38.

Вчерашний день:

$time = strtotime("-1 day");

$fecha = date("Y-m-d", $time);

2017-03-11

Найти следующий рабочий день:

echo "Следующий рабочий день:".

date("d.m.Y",(intval(date("w")==0 ?

strtotime("next day") :

(intval(date("w")==6 ?

strtotime("+2 day") : strtotime("now") )))));

## **Математические функции**

Библиотека математических функций PHP реализует методы для тригонометрических вычислений, числовых преобразований и числовых операций. Тригонометрические функции воспринимают параметры в радианах, но существуют функции преобразования градусов в радианы и наоборот.

* **abs( )** – абсолютное значение числа.
* echo abs(-0.7); // Выводит 0.7
* **acos( )** – арккосинус, выраженный в радианах.
* echo acos(-0.7); // Выводит 2.3461938234056
* **asin( )** – арксинус, выраженный в радианах.
* echo asin(-0.7); // Выводит -0.77539749661075
* **atan( )** – арктангенс, выраженный в радианах.
* echo atan(-0.7); // Выводит -0.61072596438921
* **atan2( )** – арктангенс для координат **x** и **y**, выраженный в радианах. Отличие от выражения **atan(y/x)** состоит в том, что знаки обоих параметров используются для определения квадранта результата.
* echo atan(-0.7/2); // Выводит -0.33667481938673
* echo atan2(2, -0.7); // Выводит 1.9074711461816
* **base\_convert( )** – переводит число из одной системы счисления в другую. Аргументы: переводимое число, система счисления, из которой переводят, система счисления, в которую переводят.
* echo base\_convert(13, 10, 16); // Выводит d
* **bindec( )** – десятичный эквивалент двоичной строки. Наибольшее конвертируемое число содержит 31 разряд, что соответствует 2147483647.
* echo bindec(11101011); // Выводит 235
* **ceil( )** – округление числа в большую сторону.
* echo ceil(2.35); // Выводит 3
* **cos( )** – косинус аргумента, выраженного в радианах.
* echo cos(2.35); // Выводит -0.70271307677355
* **decbin( )** – двоичный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647, или 31 разряд.
* echo decbin(235); // Выводит 11101011
* **dechex( )** – шестнадцаричный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647 или 7fffffff в шестнадцаричном выражении.
* echo dechex(235); // Выводит eb
* **decoct( )** – восьмеричный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647 или 17777777777 в восьмеричном выражении.
* echo decoct(235); // Выводит 353
* **deg2rad( )** – преобразует градусы в радианы.
* echo deg2rad(90); // Выводит 1.5707963267949
* **exp( )** – экспонента числа.
* echo exp(1); // Выводит 2.718281828459
* **floor( )** – округление числа в меньшую сторону.
* echo floor(2.99); // Выводит 2
* **fmod(x,y)** – остаток от деления x на y.
* echo fmod(4.5,2); // Выводит 0.5
* **getrandmax( )** – максимальное число, которое может быть получено в результате вызова функции **rand( )**.
* echo getrandmax( ); // Выводит 32767
* **hexdec( )** – десятичный эквивалент шестнадцаричного числа, представленного строкой. Наибольшее конвертируемое число составляет 7fffffff или 2147483647 в десятичном выражении.
* echo hexdec('eb'); // Выводит 235
* **hypot(x,y)** – длина гипотенузы прямоугольного треугольника со сторонами x и y.
* echo hypot(4,5); // Выводит 6.4031242374328
* **is\_finite()** – проверяет, является ли число допустимым конечным числом на данной платформе.
* echo is\_finite(2000); // Выводит 1
* echo is\_finite(log(0)); // Ничего не выводит
* **is\_infinite()** – проверяет, является ли число бесконечным.
* echo is\_infinite(2000); // Ничего не выводит
* echo is\_infinite(log(0)); // Выводит 1
* **is\_nan()** – проверяет, является ли значение "не-числом".
* echo is\_nan(200); // Выводит 1
* echo is\_nan(acos(1.01)); // Ничего не выводит
* **lcg\_value( )** – псевдослучайное число в диапазоне от 0 до 1.
* **log( )** – натуральный логарифм.
* echo log(exp(1)); // Выводит 1
* echo log(2.718); // Выводит 0.99989631572895
* **log10( )** – десятичный логарифм.
* echo log(1000); // Выводит 3
* **log1p( )** – log(1+x), рассчитанное таким образом, что результат точен, даже когда значение x близко к нулю. Из-за недостатка точности log() в этом случае может вернуть просто log(1).
* **max( )** – наибольшее значение из списка параметров. Возможно сравнение неограниченного кличества значений. В качестве параметра может быть задан массив.
* echo max(12.23, 42.554, 58.234, 34.31); // Выводит 58.234
* $a = array(12.23, 42.554, 58.234, 34.31);
* echo max($a); // Выводит 58.234
* **min( )** – наибольшее значение из списка параметров. Возможно сравнение неограниченного количества значений. В качестве параметра может быть задан массив.
* echo min(12.23, 42.554, 58.234, 34.31); // Выводит 12.23
* $a = array(12.23, 42.554, 58.234, 34.31);
* echo min($a); // Выводит 12.23
* **mt\_getrandmax( )** – максимальное число, которое может быть получено в результате вызова функции **mt\_rand( )**.
* echo mt\_getrandmax( ); // Выводит 2147483647
* **mt\_rand( )** – случайное число. Для его получения используется генерация случайных чисел по методу Твистера. Необязательные аргументы указывают диапазон допустимых значений. Перед использованием этой функции необходимо установить начальное число с помощью функции **mt\_srand( )**.
* **mt\_srand( )** – устанавливает начальное число для генератора случайных чисел в соответствии с заданным параметром. Это позволяет получать различные результаты при вызове функции **mt\_rand( )**.
* mt\_srand(9);
* for ($i = 1; $i <= 10; $i++)
* echo mt\_rand(1, 9)." "; // Выводит 7 3 8 7 1 4 4 9 3 5
* **number\_format( )** – форматирует число. Аргументы: форматируемое число, число знаков после запятой, символ, используемый вместо десятичной точки (необязательно), символ разграничения тысяч (необязательно).
* $a = 3456787 \* log(2.718);
* echo number\_format($a, 4, ",", "'"); // Выводит 3'456'428,5856
* **octdec( )** – десятичный эквивалент восьмеричного числа, представленного строкой. Наибольшее конвертируемое число составляет 17777777777 или 2147483647 в десятичном выражении.
* echo octdec(353); // Выводит 235
* **pi( )** – приближенное значение числа **π**.
* echo pi(); // Выводит 3.1415926535898
* **pow( )** – возведение в степень. Аргументы: основание и показатель степени.
* echo pow(2, 3); // Выводит 8
* **rad2deg( )** – преобразует радианы в градусы.
* echo rad2deg(pi()/4); // Выводит 45
* **rand( )** – псевдослучайное число. Необязательные аргументы указывают диапазон допустимых значений. Если их не задавать, то число выбирается из диапазона от 0 до RAND\_MAX. Перед использованием этой функции необходимо установить начальное число с помощью функции **srand( )**.
* **round( )** – округление числа до ближайшего целого.
* echo round(6.45656); // Выводит 6
* **sin( )** – синус аргумента, выраженного в радианах.
* echo sin(pi()/2); // Выводит 1
* **sqrt( )** – квадратный корень числа.
* echo sqrt(121); // Выводит 11
* **srand( )** – устанавливает заданное начальное число для генератора псевдослучайных чисел. Это позволяет получать различные результаты при вызове функции **rand( )**.
* srand(9);
* for ($i = 1; $i <= 10; $i++)
* echo rand(1, 9)." "; // Выводит 9 1 6 2 3 4 6 3 4 6
* **tan( )** – тангенс аргумента, выраженного в радианах.
* echo tan(pi()/4); // Выводит 1

В языке PHP существуют также функции математических вычислений произвольной точности. Особенностью этих функций является параметр разрядности. Разрядность – количество знаков после десятичной точки в операндах и результате. По умолчанию принимается 0.

* **bcadd( )** – сумма первого и второго аргументов. Третий аргумент - параметр разрядности.
* echo bcadd(4.009, 4.009, 2); // Выводит 8
* **bccomp( )** – числовое сравнение. Если первый аргумент больше второго, то возвращается +1. Если первый аргумент меньше второго, то возвращается -1. Если аргументы равны, то возвращается 0.
* echo bccomp(4.001, 4.009, 2); // Выводит 0
* echo bccomp(4.01, 4.09, 2); // Выводит -1
* echo bccomp(4.09, 4.01, 2); // Выводит 1
* **bcdiv( )** – частное от деления первого аргумента на второй.
* echo bcdiv(4.001, 2.009, 2); // Выводит 2
* echo bcdiv(2.003, 3.009, 2); // Выводит 0.66
* echo bcdiv(4.00, 0.005, 2); // Выводит предупреждение о попытке деления на нуль
* **bcmod( )** – остаток от деления первого аргумента на второй.
* echo bcmod(1525, 3); // Выводит 1
* echo bcmod(1525.999, 3); // Выводит 1
* echo bcmod(1525.999, 3.123); // Выводит 1
* **bcmul( )** – произведение первого и второго аргументов. Третий аргумент – параметр разрядности.
* echo bcmul(2.008, 4.009, 2); // Выводит 8.00
* **bcpow( )** – возведение в степень. Аргументы: основание, показатель степени, параметр разрядности. Показатель степени не должен содержать знаков после десятичной точки.
* echo bcpow(2.005, 3, 2); // Выводит 8.00
* echo bcpow(2.25, 3, 2); // Выводит 11.39
* **bcscale( )** – значение параметра разрядности, который будет использоваться по умолчанию.
* echo bcpow(2.25, 3); // Выводит 8
* echo bcscale(2);
* echo bcpow(2.25, 3); // Выводит 11.39
* **bcsqrt( )** – квадратный корень числа.
* echo bcsqrt(121.00, 2); // Выводит 11.00
* echo bcsqrt(121.75, 2); // Выводит 11.03
* **bcsub( )** – разность. Из первого аргумента вычитается второй.
* echo bcsub(4.001, 2.009, 2); // Выводит 2.00
* echo bcsub(2.03, 3.09, 2); // Выводит -1.06

## **Разнородные функции**

Здесь собраны функции PHP, которые нельзя отнести к определенной группе: средства отладки; конструкции, связанные с выполнением сценария; функции, дающие информацию о браузере; функции, дающие информацию о текущем состоянии соединения с браузером.

* [**connection\_aborted**](https://htmlweb.ru/php/function/connection_aborted.php)**()** - возвращает значение **true** при отключении клиента. Обычно это происходит в результате щелчка на кнопке "Остановить" ("Stop") в окне пользовательского браузера.
* [**connection\_timeout**](https://htmlweb.ru/php/function/connection_timeout.php)**()** - возвращает значение **true** в случае тайм-аута сценария.
* [**define**](https://htmlweb.ru/php/function/define.php)**()** - объявление именованной константы. Аргументы: имя констаниы, значение, признак чувствительности к регистру. Третий аргумент необязателен. По умолчанию регистр учитывается. Для независимости от регистра в качестве третьего аргумента задается 1.
* define("e", 2.718281828);
* [**defined**](https://htmlweb.ru/php/function/defined.php)**()** - возвращает значение **true**, если заданный аргумент является существующей именованной константой.
* [**die**](https://htmlweb.ru/php/function/die.php)**()** - выводит сообщение и прекращает выполнение сценария.

if ( !file\_exists( "aaa.php" ) ) die "Файл aaa.php не найден!";

* [**eval**](https://htmlweb.ru/php/function/eval.php)**()** - выполняет строку как PHP-код. Строка должна соответствовать обычным требованиям к PHP-коду. Все переменные, созданные в этом коде, после выполнения функции продолжают существовать в основном коде.
* [**exit**](https://htmlweb.ru/php/function/exit.php)**()** - прекращает выполнение сценария.
* [**func\_get\_args**](https://htmlweb.ru/php/function/func_get_args.php)**()** - возвращает массив аргументов текущей функции пользователя. Базовое значние счетчика массива - нуль. При вызове вне рамок функции пользователя генерируется предупреждение.
* [**func\_num\_args**](https://htmlweb.ru/php/function/func_num_args.php)**()** - возвращает количество аргументов текущей функции пользователя. Базовое значние счетчика массива - нуль. При вызове вне рамок функции пользователя генерируется предупреждение.
* [**function\_exists**](https://htmlweb.ru/php/function/function_exists.php)**()** - возвращает значение **true**, если заданный аргумент - определенная функция.
* [**sleep**](https://htmlweb.ru/php/function/sleep.php)**()** - приостанавливает обработку сценария на количество секунд, заданное параметром.
* [**usleep**](https://htmlweb.ru/php/function/usleep.php)**()** - приостанавливает обработку сценария на количество микросекунд, заданное параметром.
* [**extension\_loaded**](https://htmlweb.ru/php/function/extension_loaded.php)**()** - возвращает значение **true**, если загружено расширение с именем, заданным параметром.
* [**getlastmod**](https://htmlweb.ru/php/function/getlastmod.php)**()** - возвращает метку времени UNIX последнего изменения сценария, который выполняется в данный момент. Формат метки времени пригоден для передачи в PHP функцию [**date()**](https://htmlweb.ru/php/function/date.php).
* [**phpinfo**](https://htmlweb.ru/php/function/phpinfo.php)**()** - выводит сведения о текущем состоянии PHP.
* **phpversion( )** - возвращает версию выполняемой в данный момент программы синтаксического анализа PHP. Например, **4.3.0RC2**.
* [**php\_logo\_guid**](https://htmlweb.ru/php/function/php_logo_guid.php)**()** - возвращает глобальный идентификатор регистрационных данных (logo).
* [**php\_sapi\_name**](https://htmlweb.ru/php/function/php_sapi_name.php)**()** - возвращает тип интерфейса, обеспечивающего взаимодействие Web-сервера и PHP. Например, **cgi** или **apache**.
* [**set\_time\_limit**](https://htmlweb.ru/php/function/set_time_limit.php)**()** - устанавливает количество секунд допустимой продолжительности выполнения сценария. По умолчанию ограничение составляет 30 секунд. Если парамер функции имеет нулевое значение, то ограничение по времени отсутствует. Если PHP выполняется в безопасном режиме, функция[**set\_time\_limit**](https://htmlweb.ru/php/function/set_time_limit.php)**()** не дает эффекта.

# Регулярные выражения в PHP.

Регулярные выражения позволяют найти в строке последовательности, соответствующие шаблону. Например шаблон "Вася(.\*)Пупкин" позволит найти последовательность когда между словами Вася и Пупкин будет любое количество любых символов. Если надо найти шесть цифр, то пишем "[0-9]{6}" (если, например, от шести до восьми цифр, тогда "[0-9]{6,8}"). Здесь разделены такие вещи как указатель набора символов и указатель необходимого количества:  
  
<набор символов><квантификатор><жадность>  
  
Вместо набора символов может быть использовано обозначение любого символа - точка, может быть указан конкретный набор символов (поддерживаются последовательности - упоминавшиеся "0-9"). Может быть указано "кроме данного набора символов".

Указатель количества символов в официальной документации по php называется "квантификатор". Термин удобный и не несет в себе кривотолков. Итак, квантификатор может иметь как конкретное значение - либо одно фиксированное ("{6}"), либо как числовой промежуток ("{6,8}"), так и абстрактное "любое число, в т.ч. 0" ("\*"), "любое натуральное число" - от 1 до бесконечности ("+": "document[0-9]+.txt"), "либо 0, либо 1" ("?"). По умолчанию квантификатор для данного набора символов равен единице ("document[0-9].txt").  
  
Для более гибкого поиска сочетаний эти связки "набор символов - квантификатор" можно объединять в метаструктуры.  
  
Как всякий гибкий инструмент, регулярные выражения гибки, но не абсолютно: зона их применения ограничена. Например, если вам надо заменить в тексте одну фиксированную строку на другую, фиксированную опять же, пользуйтесь str\_replace. Разработчики php слезно умоляют не пользоваться ради этого сложными функциями ereg\_replace или preg\_replace, ведь при их вызове происходит процесс интерпретации строки, а это серьезно потребляет ресурсы системы. К сожалению, это любимые грабли начинающих php-программистов.  
  
Пользуйтесь функциями регулярных выражений только если вы не знаете точно, какая "там" строка. Из примеров: поисковый код , в котором из строки поиска вырезаются служебные символы и короткие слова а так же вырезаются лишние пробелы (вернее, все пробелы сжимаются: " +" заменяется на один пробел). При помощи этих функций я проверяю email пользователя, оставляющего свой отзыв. Много полезного можно сделать, но важно иметь в виду: регулярные выражения не всесильны. Например, сложную замену в большом тексте ими лучше не делать. Ведь, к примеру, комбинация "(.\*)" в программном плане означает перебор всех символов текста. А если шаблон не привязан к началу или концу строки, то и сам шаблон "двигается" программой через весь текст, и получается двойной перебор, вернее перебор в квадрате. Нетрудно догадаться, что еще одна комбинация "(.\*)" означает перебор в кубе, и так далее. Возведите в третью степень, скажем, 5 килобайт текста. Получается 125 000 000 000 (прописью: сто двадцать пять миллиардов операций). Конечно же, если подходить строго, там стольких операций не будет, а будет раза в четыре-восемь меньше, но важен сам порядок цифр.

## **Набор символов**

|  |  |  |
| --- | --- | --- |
| . | точка | любой символ |
| [<символы>] | квадратные скобки | класс символов ("любое из"). Например [abcdef] |
| [^<символы>] |  | негативный класс символов ("любое кроме") |
| - | тире | обозначение последовательности в классе символов ("[0-9]" — цифры) |
| \d | [0-9] | Только цифры |
| \D | [^0-9] | Кроме цифр |
| \w | [a-z0-9] | Буквы и цифры |
| \W | [^a-z0-9] | Кроме букв и цифр |
| \s | [ ] | Пробельные символы: пробел, табуляция, перевод строки |
| \S | [^ ] | Кроме пробельных символов |
| | | (одно|другое) | На этом месте может быть один из перечисленных вариантов, например: (Вася|Петя|Маша). Если Вы не хотите, чтобы это попало в выборку используйте (?: ... ) |

Не пользуйтесь классом символов для обозначения всего лишь одного (вместо "[ ]+" вполне сойдет " +"). Не пишите в классе символов точку — это ведь любой символ, тогда другие символы в классе будут просто лишними (а в негативном классе получится отрицание всех символов).

## **Квантификатор**

Квантификатором можно указать как конкретное значение, так и пределы. Если число заданных подпадает под пределы квантификатора, фрагмент выражения считается совпавшим с разбираемой строкой. Синтаксис:  
  
{*<количество>*}  
  
либо  
  
{*<минимум>*, *<максимум>*}  
  
Если нужно указать только необходимый минимум, а максимума нет, просто ставим запятую и не пишем второе число: "{5,}" ("минимум 5"). Для наиболее часто употребляемых квантификаторов есть специальные обозначения:

|  |  |  |
| --- | --- | --- |
| \* | "звёздочка" или знак умножения | {0,} |
| + | плюс | {1,} |
| ? | вопросительный знак | {0,1} |

На практике такие символы используются чаще, чем фигурные скобки.

## **Якоря**

|  |  |
| --- | --- |
| ^ | привязка к началу строки |
| $ | привязка к концу строки |

Эти символы должны стоять соответственно в самом начале и в самом конце строки.

## **Жадность**

Вопросительный знак выступает еще и как минимизатор квантификатора:  
.\*?

<?

$str = '[b]жирный текст [b]а тут - еще жирнее[/b] вернулись[/b]';

$to = '<b>$1</b>';

$re1 = '|\[b\] (.\*) \[/b\]|ixs';

$re2 = '|\[b\] (.\*?) \[/b\]|ixs';

$result = preg\_replace($re1, $to, $str);

echo "Жадная версия: ".htmlspecialchars($result,null,'windows-1251')."<br />";

$result = preg\_replace($re2, $to, $str);

echo "Ленивая версия: ".htmlspecialchars($result,null,'windows-1251')."<br />";

?>

Результат работы примера:

Жадная версия: <b>жирный текст [b]а тут - еще жирнее[/b] вернулись</b>

Ленивая версия: <b>жирный текст [b]а тут - еще жирнее</b> вернулись[/b]

Строка шаблона, как вы уже заметили, начинается и заканчивается слэшами. После второго идут параметры:

|  |  |
| --- | --- |
| i | регистронезависимый поиск |
| m | многостроковый режим. По умолчанию PCRE ищет совпадения с шаблоном только внутри одной строки, а символы "^" и "$" совпадают только с началом и концом всего текста. Когда этот параметр установлен, "^" и "$" совпадают с началом и концом отдельных строк. |
| s | символ "." (точка) совпадает и с переносом строки (по умолчанию — нет) |
| A | привязка к началу текста |
| E | заставляет символ "$" совпадать только с концом текста. Игнорируется, если установлен парамерт m. |
| U | Инвертирует "жадность" для каждого квантификатора (если же после квантификатора стоит "?", этот квантификатор перестает быть "жадным"). |
| e | Строка замены интерпретитуется как PHP код. |

## **Функции для работы с регулярными выражениями**

* [preg\_grep](https://htmlweb.ru/php/function/preg_grep.php) - Возвращает массив вхождений, которые соответствуют шаблону
* [preg\_match](https://htmlweb.ru/php/function/preg_match.php) - Выполняет проверку на соответствие регулярному выражению. Данная функция ищет только первое совпадение!
* [preg\_match\_all](https://htmlweb.ru/php/function/preg_match_all.php) - Выполняет глобальный поиск шаблона в строке
* [preg\_quote](https://htmlweb.ru/php/function/preg_quote.php) - Экранирует символы в регулярных выражениях. Т.е. вставляет слэши перед всеми служебными символами (например, скобками, квадратными скобками и т.п.), чтобы те воспринимались буквально. Если у вас есть какой-либо ввод информации пользователем, и вы проверяете его с помощью регулярных выражений, то лучше перед этим заэкранировать служебные символы в пришедшей переменной
* [preg\_replace](https://htmlweb.ru/php/function/preg_replace.php) - Выполняет поиск и замену по регулярному выражению
* [preg\_replace\_callback](https://htmlweb.ru/php/function/preg_replace_callback.php) - Выполняет поиск по регулярному выражению и замену
* [preg\_split](https://htmlweb.ru/php/function/preg_split.php) - Разбивает строку по регулярному выражению

#### preg\_grep

Функция **preg\_grep** - Возвращает массив вхождений, которые соответствуют шаблону

Синтаксис

array **preg\_grep** (string pattern, array input [, int flags])

preg\_grep() возвращает массив, состоящий из элементов входящего массива input, которые соответствуют заданному шаблону pattern.   
  
Параметр flags может принимать следующие значения:   
  
PREG\_GREP\_INVERT  
В случае, если этот флаг установлен, функция preg\_grep(), возвращает те элементы массива, которые не соответствуют заданному шаблону pattern.   
Результат, возвращаемый функцией preg\_grep() использует те же индексы, что и массив исходных данных. Если такое поведение вам не подходит, примените array\_values() к массиву, возвращаемому preg\_grep() для реиндексации.   
Пример кода:

// Возвращает все элементы массива,

// содержащие числа с плавающей точкой

$fl\_array = preg\_grep("/^(\d+)?\.\d+$/", $array);

#### preg\_match

Функция **preg\_match** - Выполняет проверку на соответствие регулярному выражению

Синтаксис

int **preg\_match** ( string pattern, string subject [, array matches [, int flags [, int offset]]]) Ищет в заданном тексте subject совпадения с шаблоном pattern

В случае, если дополнительный параметр matches указан, он будет заполнен результатами поиска. Элемент $matches[0] будет содержать часть строки, соответствующую вхождению всего шаблона, $matches[1] - часть строки, соответствующую первой подмаске, и так далее.

flags может принимать следующие значения:

PREG\_OFFSET\_CAPTURE  
В случае, если этот флаг указан, для каждой найденной подстроки будет указана ее позиция в исходной строке. Необходимо помнить, что этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом - смещение.

Поиск осуществляется слева направо, с начала строки. Дополнительный параметр offset может быть использован для указания альтернативной начальной позиции для поиска. Аналогичного результата можно достичь, заменив subject на substr()($subject, $offset).

Функция preg\_match() возвращает количество найденных соответствий. Это может быть 0 (совпадения не найдены) и 1, поскольку preg\_match() прекращает свою работу после первого найденного совпадения. Если необходимо найти либо сосчитать все совпадения, следует воспользоваться функцией preg\_match\_all(). Функция preg\_match() возвращает FALSE в случае, если во время выполнения возникли какие-либо ошибки.

Рекомендация: Не используйте функцию preg\_match(), если необходимо проверить наличие подстроки в заданной строке. Используйте для этого strpos() либо strstr(), поскольку они выполнят эту задачу гораздо быстрее.

#### Пример кода

<?php

// Символ "i" после закрывающего ограничителя шаблона означает

// регистронезависимый поиск.

if (preg\_match("/php/i", "PHP is the web scripting language of choice.")) {

echo "Вхождение найдено.";

} else {

echo "Вхождение не найдено.";

}

?>

#### Пример кода

<?php

/\*

Специальная последовательность \b в шаблоне означает границу слова,

следовательно, только изолированное вхождение слова 'web' будет соответствовать

маске, в отличие от "webbing" или "cobweb".

\*/

if (preg\_match("/\bweb\b/i", "PHP is the web scripting language of choice.")) {

echo "Вхождение найдено.";

} else {

echo "Вхождение не найдено.";

}

if (preg\_match("/\bweb\b/i", "PHP is the website scripting language of choice.")) {

echo "Вхождение найдено.";

} else {

echo "Вхождение не найдено.";

}

?>

#### Пример кода

<?php

// Извлекаем имя хоста из URL

preg\_match("/^(http:\/\/)?([^\/]+)/i",

"http://www.htmlweb.ru/index.html", $matches);

$host = $matches[2];

// извлекаем две последние части имени хоста

preg\_match("/[^\.\/]+\.[^\.\/]+$/", $host, $matches);

echo "domain name is: {$matches[0]}\n";

?>

Результат работы примера:

domain name is: htmlweb.ru

#### preg\_match\_all

Функция **preg\_match\_all** - Выполняет глобальный поиск шаблона в строке

Синтаксис

int **preg\_match\_all** (string pattern, string subject, array matches [, int flags [, int offset]])

Ищет в строке subject все совпадения с шаблоном pattern и помещает результат в массив matches в порядке, определяемом комбинацией флагов flags.  
  
После нахождения первого соответствия последующие поиски будут осуществляться не с начала строки, а от конца последнего найденного вхождения.  
  
Дополнительный параметр flags может комбинировать следующие значения (необходимо понимать, что использование PREG\_PATTERN\_ORDER одновременно с PREG\_SET\_ORDER бессмысленно):

**PREG\_PATTERN\_ORDER**  
Если этот флаг установлен, результат будет упорядочен следующим образом: элемент $matches[0] содержит массив полных вхождений шаблона, элемент $matches[1] содержит массив вхождений первой подмаски, и так далее.

#### Пример кода

<?php

preg\_match\_all("|<[^>]+>(.\*)</[^>]+>|U",

"<b>example: </b><div align=left>this is a test</div>",

$out, PREG\_PATTERN\_ORDER);

echo $out[0][0] . ", " . $out[0][1] . "\n";

echo $out[1][0] . ", " . $out[1][1] . "\n";

?>

Результат работы примера:

<b>example: </b>, <div align="left">this is a test</div>

example: , this is a test

Как мы видим, $out[0] содержит массив полных вхождений шаблона, а элемент $out[1] содержит массив подстрок, содержащихся в тегах.

**PREG\_SET\_ORDER**  
Если этот флаг установлен, результат будет упорядочен следующим образом: элемент $matches[0] содержит первый набор вхождений, элемент $matches[1] содержит второй набор вхождений, и так далее.

#### Пример кода

<?php

preg\_match\_all("|<[^>]+>(.\*)</[^>]+>|U",

"<b>example: </b><div align=\"left\">this is a test</div>",

$out, PREG\_SET\_ORDER);

echo $out[0][0] . ", " . $out[0][1] . "\n";

echo $out[1][0] . ", " . $out[1][1] . "\n";

?>

Результат работы примера:

<b>example: </b>, example:

<div align="left">this is a test</div>, this is a test

В таком случае массив $matches[0] содержит первый набор вхождений, а именно: элемент $matches[0][0] содержит первое вхождение всего шаблона, элемент $matches[0][1] содержит первое вхождение первой подмаски, и так далее. Аналогично массив $matches[1] содержит второй набор вхождений, и так для каждого найденного набора.

**PREG\_OFFSET\_CAPTURE**  
В случае, если этот флаг указан, для каждой найденной подстроки будет указана ее позиция в исходной строке. Необходимо помнить, что этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом - смещение.

В случае, если никакой флаг не используется, по умолчанию используется PREG\_PATTERN\_ORDER.

Поиск осуществляется слева направо, с начала строки. Дополнительный параметр offset может быть использован для указания альтернативной начальной позиции для поиска. Аналогичного результата можно достичь, заменив subject на substr()($subject, $offset).

Возвращает количество найденных вхождений шаблона (может быть нулем) либо FALSE, если во время выполнения возникли какие-либо ошибки.

#### Пример кода

<?php

preg\_match\_all("/\(? (\d{3})? \)? (?(1) [\-\s] ) \d{3}-\d{4}/x",

"Call 555-1212 or 1-800-555-1212", $phones);

?>

#### Пример кода

<?php

// Запись \\2 является примером использования ссылок на подмаски.

// Она означает необходимость соответствия подстроки строке, зафиксированной

// второй подмаской, в нашем примере это ([\w]+).

// Дополнительный слеш необходим, так как используются двойные кавычки.

$html = "<b>bold text</b><a href=howdy.html>click me</a>";

preg\_match\_all("/(<([\w]+)[^>]\*>)(.\*)(<\/\\2>)/", $html, $matches);

for ($i=0; $i< count($matches[0]); $i++) {

echo "matched: " . $matches[0][$i] . "\n";

echo "part 1: " . $matches[1][$i] . "\n";

echo "part 2: " . $matches[3][$i] . "\n";

echo "part 3: " . $matches[4][$i] . "\n\n";

}

?>

Результат работы примера:

matched: <b>bold text</b>

part 1: <b>

part 2: bold text

part 3: </b>

matched: <a href=howdy.html>click me</a>

part 1: <a href=howdy.html>

part 2: click me

part 3: </a>

#### preg\_quote

Функция **preg\_quote** - Экранирует символы в регулярных выражениях

Синтаксис

string **preg\_quote** (string str [, string delimiter])

Функция preg\_quote() принимает строку str и добавляет обратный слеш перед каждым служебным символом. Это бывает полезно, если в составлении шаблона участвуют строковые переменные, значение которых в процессе работы скрипта может меняться.   
  
В случае, если дополнительный параметр delimiter указан, он будет также экранироваться. Это удобно для экранирования ограничителя, который используется в PCRE функциях. Наиболее распространенным ограничителем является символ '/'.   
  
В регулярных выражениях служебными считаются следующие символы: . \\ + \* ? [ ^ ] $ ( ) { } = ! < > | :

#### Пример кода

<?php

$keywords = "$40 for a g3/400";

$keywords = preg\_quote($keywords, "/");

echo $keywords; // возвращает \$40 for a g3\/400

?>

#### Пример кода

<?php

// Выделение курсивом слова в тексте

// В данном примере preg\_quote($word) используется, чтобы

// избежать трактовки символа '\*' как спец. символа.

$textbody = "This book is \*very\* difficult to find.";

$word = "\*very\*";

$textbody = preg\_replace ("/" . preg\_quote($word) . "/",

"<i>" . $word . "</i>",

$textbody);

echo $textbody;

?>

Результат работы примера:

This book is <i>\*very\*</i> difficult to find.

#### preg\_replace

Функция **preg\_replace** - Выполняет поиск и замену по регулярному выражению

Синтаксис

mixed **preg\_replace** ( mixed pattern, mixed replacement, mixed subject [, int limit])

Выполняет поиск в строке subject совпадений с шаблоном pattern и заменяет их на replacement. В случае, если параметр limit указан, будет произведена замена limit вхождений шаблона; в случае, если limit опущен либо равняется -1, будут заменены все вхождения шаблона.   
  
Replacement может содержать ссылки вида \\n либо (начиная с PHP 4.0.4) $n, причем последний вариант предпочтительней. Каждая такая ссылка, будет заменена на подстроку, соответствующую n'нной заключенной в круглые скобки подмаске. n может принимать значения от 0 до 99, причем ссылка \\0 (либо $0) соответствует вхождению всего шаблона. Подмаски нумеруются слева направо, начиная с единицы.   
  
При использовании замены по шаблону с использованием ссылок на подмаски может возникнуть ситуация, когда непосредственно за маской следует цифра. В таком случае нотация вида \\n приводит к ошибке: ссылка на первую подмаску, за которой следует цифра 1, запишется как \\11, что будет интерпретировано как ссылка на одиннадцатую подмаску. Это недоразумение можно устранить, если воспользоваться конструкцией \${1}1, указывающей на изолированную ссылку на первую подмаску, и следующую за ней цифру 1.

Пример кода

<?php

$string = "April 15, 2003";

$pattern = "/(\w+) (\d+), (\d+)/i";

$replacement = "\${1}1,\$3";

echo preg\_replace($pattern, $replacement, $string);

?>

Результатом работы этого примера будет:

April1,2003.

Если во время выполнения функции были обнаружены совпадения с шаблоном, будет возвращено измененное значение subject, в противном случае будет возвращен исходный текст subject.

Первые три параметра функции preg\_replace() могут быть одномерными массивами. В случае, если массив использует ключи, при обработке массива они будут взяты в том порядке, в котором они расположены в массиве. Указание ключей в массиве для pattern и replacement не является обязательным. Если вы все же решили использовать индексы, для сопоставления шаблонов и строк, участвующих в замене, используйте функцию [ksort()](https://htmlweb.ru/php/function/ksort.php) для каждого из массивов.

<?php

// Использование массивов с числовыми индексами в качестве аргументов

// функции preg\_replace()

$string = "The quick brown fox jumped over the lazy dog.";

$patterns[0] = "/quick/";

$patterns[1] = "/brown/";

$patterns[2] = "/fox/";

$replacements[2] = "bear";

$replacements[1] = "black";

$replacements[0] = "slow";

echo preg\_replace($patterns, $replacements, $string);

?>

The bear black slow jumped over the lazy dog.

Используя ksort(), получаем желаемый результат:

<?php

ksort($patterns);

ksort($replacements);

echo preg\_replace($patterns, $replacements, $string);

?>

The slow black bear jumped over the lazy dog.

В случае, если параметр subject является массивом, поиск и замена по шаблону производятся для каждого из его элементов. Возвращаемый результат также будет массивом.   
  
В случае, если параметры pattern и replacement являются массивами, preg\_replace() поочередно извлекает из обоих массивов по паре элементов и использует их для операции поиска и замены. Если массив replacement содержит больше элементов, чем pattern, вместо недостающих элементов для замены будут взяты пустые строки. В случае, если pattern является массивом, а replacement - строкой, по каждому элементу массива pattern будет осущесвтлен поиск и замена на pattern (шаблоном будут поочередно все элементы массива, в то время как строка замены остается фиксированной). Вариант, когда pattern является строкой, а replacement - массивом, не имеет смысла.

Модификатор /e меняет поведение функции preg\_replace() таким образом, что параметр replacement после выполнения необходимых подстановок интерпретируется как PHP-код и только после этого используется для замены. Используя данный модификатор, будьте внимательны: параметр replacement должен содержать корректный PHP-код, в противном случае в строке, содержащей вызов функции preg\_replace(), возникнет ошибка синтаксиса.

#### Пример кода: Замена по нескольким шаблонам

<?php

// Замена по нескольким шаблонам

$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",

"/^\s\*{(\w+)}\s\*=/");

$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");

echo preg\_replace($patterns, $replace, "{startDate} = \"1999-5-27\";");

?>

Этот пример выведет:

$startDate = "5/27/1999";

#### Пример кода: Использование модификатора /e

<?php

// Использование модификатора /e

preg\_replace("/(<\/?)(\w+)([^>]\*>)/e",

"'\\1'.strtoupper('\\2').'\\3'",

$html\_body);

?>

#### Пример кода: Преобразует все HTML-теги к верхнему регистру

<?php

// Конвертор HTML в текст

// $document на выходе должен содержать HTML-документ.

// Необходимо удалить все HTML-теги, секции javascript,

// пробельные символы. Также необходимо заменить некоторые

// HTML-сущности на их эквивалент.

$search = array ("'<script[^>]\*?>.\*?</script>'si", // Вырезает javaScript

"'<[\/\!]\*?[^<>]\*?>'si", // Вырезает HTML-теги

"'([\r\n])[\s]+'", // Вырезает пробельные символы

"'&(quot|#34);'i", // Заменяет HTML-сущности

"'&(amp|#38);'i",

"'&(lt|#60);'i",

"'&(gt|#62);'i",

"'&(nbsp|#160);'i",

"'&(iexcl|#161);'i",

"'&(cent|#162);'i",

"'&(pound|#163);'i",

"'&(copy|#169);'i",

"'&#(\d+);'e"); // интерпретировать как php-код

$replace = array ("",

"",

"\\1",

"\"",

"&",

"<",

">",

" ",

chr(161),

chr(162),

chr(163),

chr(169),

"chr(\\1)");

$text = preg\_replace($search, $replace, $document);

?>

#### preg\_replace\_callback

Функция **preg\_replace\_callback** - Выполняет поиск по регулярному выражению и замену с использованием функции обратного вызова

Синтаксис

mixed **preg\_replace\_callback** (mixed pattern, callback callback, mixed subject [, int limit])

Поведение этой функции во многом напоминает preg\_replace(), за исключением того, что вместо параметра replacement необходимо указывать callback функцию, которой в качестве входящего параметра передается массив найденных вхождений. Ожидаемый результат - строка, которой будет произведена замена.

#### Пример кода

<?php

/\* фильтр, подобный тому, что используется в системах Unix

для преобразования в заглавные начальных букв параграфа \*/

$fp = fopen("php://stdin", "r") or die("can't read stdin");

while (!feof($fp)) {

$line = fgets($fp);

$line = preg\_replace\_callback(

'|<p>\s\*\w|',

create\_function(

// Использование одиночных кавычек в данном случае принципиально,

// альтернатива - экранировать все символы '$'

'$matches',

'return strtolower($matches[0]);'

),

$line

);

echo $line;

}

fclose($fp);

?>

#### preg\_split

Функция **preg\_split** - Разбивает строку по регулярному выражению

Синтаксис

array **preg\_split** (string pattern, string subject [, int limit [, int flags]])

Возвращает массив, состоящий из подстрок заданной строки subject, которая разбита по границам, соответствующим шаблону pattern.   
  
В случае, если параметр limit указан, функция возвращает не более, чем limit подстрок. Специальное значение limit, равное -1, подразумевает отсутствие ограничения, это весьма полезно для указания еще одного опционального параметра flags.   
  
flags может быть произвольной комбинацией следующих флагов (соединение происходит при помощи оператора '|'):   
  
  
**PREG\_SPLIT\_NO\_EMPTY**  
В случае, если этот флаг указан, функция preg\_split() вернет только непустые подстроки.   
  
**PREG\_SPLIT\_DELIM\_CAPTURE**  
В случае, если этот флаг указан, выражение, заключенное в круглые скобки в разделяющем шаблоне, также извлекается из заданной строки и возвращается функцией. Этот флаг был добавлен в PHP 4.0.5.   
  
**PREG\_SPLIT\_OFFSET\_CAPTURE**  
В случае, если этот флаг указан, для каждой найденной подстроки, будет указана ее позиция в исходной строке. Необходимо помнить, что этот флаг меняет формат возвращаемых данных: каждое вхождение возвращается в виде массива, в нулевом элементе которого содержится найденная подстрока, а в первом - смещение.

#### Примеры кода

<?php

// Получение подстрок из заданного текста

// разбиваем строку по произвольному числу запятых и пробельных символов,

// которые включают в себя " ", \r, \t, \n и \f

$keywords = preg\_split("/[\s,]+/", "hypertext language, programming");

?>

<?php

// Разбиваем строку на составляющие символы

$str = 'string';

$chars = preg\_split('//', $str, -1, PREG\_SPLIT\_NO\_EMPTY);

print\_r($chars);

?>

<?php

$str = 'hypertext language programming';

$chars = preg\_split('/ /', $str, -1, PREG\_SPLIT\_OFFSET\_CAPTURE);

print\_r($chars);

?>

В случае, если после открывающей круглой скобки следует "**?:**", захват строки не происходит, и текущая подмаска не нумеруется. Например, если строка "the white queen" сопоставляется с шаблоном the ((?:red|white) (king|queen)), будут захвачены подстроки "white queen" и "queen", и они будут пронумерованы 1 и 2 соответственно:

$r="/the ((?:red|white) (king|queen))/";

0="the white king";

1="white king";

2="king";

# Curl

PHP поддерживает libcurl, библиотеку, созданную Daniel"ом Stenberg"ом, которая даёт возможность соединяться с серверами различных типов и по разным протоколам.  
[libcurl](http://curl.haxx.se/) в настоящее время поддерживает протоколы http, https, ftp, gopher, telnet, dict, file и ldap.  
libcurl также поддерживает сертификаты HTTPS, HTTP POST, HTTP PUT, загрузку по FTP (это можно сделать также РНР-расширением ftp), загрузку на основе форм HTTP, прокси, куки и аутентификацию user+password.

Эти функции были введены в PHP 4.0.2.

## **curl\_init**

curl\_init - инициализирует CURL-сессию.

### **Описание**

resource **curl\_init**([string url])

Функция **curl\_init()** инициализирует новую сессию и возвратит CURL-дескриптор для использования в функциях [**curl\_setopt()**](https://htmlweb.ru/php/php_curl.php#curl_setopt), [**curl\_exec()**](https://htmlweb.ru/php/php_curl.php#curl_exec) и [**curl\_close()**](https://htmlweb.ru/php/php_curl.php#curl_close). Если необязательный параметр ***url*** предоставлен, то опция CURLOPT\_URL получит значение этого параметра. Вы можете вручную устанавливать его с помощью функции [**curl\_setopt()**](https://htmlweb.ru/php/php_curl.php#curl_setopt).

## **curl\_setopt**

curl\_setopt - устанавливает опции для CURL-трансфера/transfer.

### **Описание**

bool **curl\_setopt** (resource ch, string option, mixed value)

Функция **curl\_setopt()** устанавливает опции для CURL-сессии, идентифицируемой параметром ***ch***. Параметр ***option*** является опцией, которую вы хотите установить, а ***value*** это значение опции ***option***.

Параметр ***value*** должен быть long для следующих опций (специфицированных параметром ***option***):

* **CURLOPT\_INFILESIZE**: Если вы выгружаете файл на удалённый сайт, эта опция должна использоваться, для того чтобы сообщит PHP, какой будет ожидаемый размер infile.
* **CURLOPT\_VERBOSE**: Установите эту опцию в ненулевое значение, если вы хотите, чтобы CURL сообщала обо всех действиях.
* **CURLOPT\_HEADER**: Установите эту опцию в ненулевое значение, если вы хотите, чтобы шапка/header включалась в вывод.
* **CURLOPT\_NOPROGRESS**: Установите эту опцию в ненулевое значение, если вы не хотите, чтобы PHP выводил индикатор процесса CURL-трансфера. (PHP автоматически устанавливает эту опцию в ненулевое значение, изменять её необходимо лишь при отладке.)
* **CURLOPT\_NOBODY**: Установите эту опцию в ненулевое значение, если вы не хотите, чтобы тело/body включалось в вывод.
* **CURLOPT\_FAILONERROR**: Установите эту опцию в ненулевое значение, если вы хотите, чтобы PHP завершал работу скрыто, если возвращаемый HTTP-код имеет значение выше 300. По умолчанию страница возвращается нормально с игнорированием кода.
* **CURLOPT\_UPLOAD**: Установите эту опцию в ненулевое значение, если вы хотите, чтобы PHP подготавливал файл к выгрузке.
* **CURLOPT\_POST**: Установите эту опцию в ненулевое значение, если вы хотите, чтобы PHP выполнял регулярный HTTP POST. Этот POST имеет нормальный вид **application/x-www-form-urlencoded**, чаще всего используемый HTML-формами.
* **CURLOPT\_FTPLISTONLY**: Установите эту опцию в ненулевое значение, и PHP будет выводит листинг имён FTP-директории.
* **CURLOPT\_FTPAPPEND**: Установите эту опцию в ненулевое значение, и PHP будет присоединять к удалённому/remote файлу, вместо его перезаписи.
* **CURLOPT\_NETRC**: Установите эту опцию в ненулевое значение, и PHP будет сканировать ваш файл**~./netrc** с целью поиска ваших username и password для удалённого сайта, с которым вы устанавливаете соединение.
* **CURLOPT\_FOLLOWLOCATION**: Установите эту опцию в ненулевое значение, чтобы следовать любому "Location: " header, который сервер высылает как часть HTTP header"а (заметьте, что это рекурсия, PHP будет следовать за всеми "Location: "-header"ами, которые высылаются.)
* **CURLOPT\_PUT**: Установите эту опцию в ненулевое значение, чтобы HTTP PUT файл. Файл для PUT обязан быть установлен с помощью CURLOPT\_INFILE и CURLOPT\_INFILESIZE.
* **CURLOPT\_MUTE**: Установите эту опцию в ненулевое значение, и PHP будет работать скрыто в отношении CURL-функций.
* **CURLOPT\_TIMEOUT**: Передаёт long как параметр, который содержит максимальное время в секундах, которое вы отводите для работы CURL-функций.
* **CURLOPT\_CONNECTTIMEOUT**: Передаёт long как параметр, который содержит максимальное время в секундах, которое вы отводите для ожидания при попытке подключения. Используйте 0 чтобы ждать бесконечно.
* **CURLOPT\_LOW\_SPEED\_LIMIT**: Передаёт long как параметр, который содержит скорость трансфера в байтах в секунду, ниже которого трансфер должен работать в процессе выполнения CURLOPT\_LOW\_SPEED\_TIME, в секундах, чтобы PHP считал его слишком медленным и прерывал его.
* **CURLOPT\_LOW\_SPEED\_TIME**: Передаёт long как параметр, который содержит время в секундах, ниже которого трансфер должен работать в процессе выполнения CURLOPT\_LOW\_SPEED\_LIMIT, чтобы PHP считал его слишком медленным и прерывал его.
* **CURLOPT\_RESUME\_FROM**: Передаёт long как параметр, который содержит смещение в байтах, с которого трансфер должен стартовать.
* **CURLOPT\_SSLVERSION**: Передаёт long как параметр, который содержит используемую версию SSL (2 или 3). По умолчанию PHP пытается определить это сам, хотя в некоторых случаях вы обязаны устанавливать это вручную.
* **CURLOPT\_SSL\_VERIFYHOST**: Передаёт long, если CURL должна проверять Common-имя peer-сертификата в SSL handshake/"рукопожатие". Значение 1 указывает, что мы должны проверить существование общего /common имени, значение 2 указывает, что мы должны убедиться в совпадении с предоставленным hostname.
* **CURLOPT\_TIMECONDITION**: Передаёт long как параметр, который определяет, как рассматривается CURLOPT\_TIMEVALUE. Вы можете установить этот параметр для TIMECOND\_IFMODSINCE или TIMECOND\_ISUNMODSINCE. Это действует только для HTTP.
* **CURLOPT\_TIMEVALUE**: Передаёт long как параметр, который является временем в секундах, прошедшим после 1 января 1970. Это время используется, как специфицировано опцией CURLOPT\_TIMEVALUE, или по умолчанию будет использоваться TIMECOND\_IFMODSINCE.
* **CURLOPT\_RETURNTRANSFER**: Передаёт ненулевое значение, если вы хотите, чтобы CURL непосредственно возвращала полученную информацию, вместо её печати напрямую.

Параметр ***value*** должен быть строкой для следующих значений параметра ***option***:

* **CURLOPT\_URL**: Это URL, который PHP должен получать. Вы можете также устанавливать эту опцию при инициализации сессии функцией [**curl\_init()**](https://htmlweb.ru/php/php_curl.php#curl_init).
* **CURLOPT\_USERPWD**: Передаёт в РНР строку, отформатированную в виде [username]:[password], для использования при соединении.
* **CURLOPT\_PROXYUSERPWD**: Передаёт в РНР строку, отформатированную в виде [username]:[password], для соединения с HTTP-прокси.
* **CURLOPT\_RANGE**: Передаёт специфицированный вами диапазон. Он должен быть в формате "X-Y", где X или Y могут отсутствовать. HTTP-трансферы поддерживают также различные интервалы, разделённые запятыми, как, например, X-Y,N-M.
* **CURLOPT\_POSTFIELDS**: Передаёт строку, содержащую полные данные для передачи операцией HTTP "POST".
* **CURLOPT\_REFERER**: Передаёт строку, содержащую "referer/ссылающийся" header, используемый в HTTP-запросе.
* **CURLOPT\_USERAGENT**: Передаёт строку, содержащую "user-agent" header, используемый в HTTP-запросе.
* **CURLOPT\_FTPPORT**: Передаёт строку, содержащую значение, которое будет использоваться для получения IP-адреса для инструкции ftp "POST". POST-инструкция указывает удалённому серверу: соединиться со специфицированным IP-адресом. Строка может быть обычным IP-адресом, hostname/именем хоста, именем сетевого интерфейса (под UNIX), или просто обычным "-", используемым для системного IP-адреса по умолчанию.
* **CURLOPT\_COOKIE**: Передаёт строку с содержимым куки/cookie, установленным в HTTP header"е.
* **CURLOPT\_SSLCERT**: Передаёт строку, содержащую filename форматированного сертификата PEM.
* **CURLOPT\_SSLCERTPASSWD**: Передаёт строку, содержащую password, необходимый для работы сертификата CURLOPT\_SSLCERT.
* **CURLOPT\_COOKIEFILE**: Передаёт строку, содержащую имя файла с данными куки. Этот cookie-файл может иметь формат Netscape, или содержать обычные шапки/headers в HTTP-стиле, забитые в файл.
* **CURLOPT\_CUSTOMREQUEST**: Передаёт строку, используемую вместо **GET** или **HEAD** при выполнении HTTP-запроса. Это делается для выполнения **DELETE** или других, более скрытых HTTP-запросов. Верными значениями являются **GET**, **POST** и так далее; то есть не вводите здесь полную строку HTTP-запроса. Например, ввод "GET /index.html HTTP/1.0" будет некорректным. (не делайте это, если не уверены, что ваш сервер поддерживает эту команду.)
* **CURLOPT\_PROXY**: Передаёт имя HTTP-прокси туннельным запросам.
* **CURLOPT\_INTERFACE**: Передаёт имя исходящего сетевого интерфейса для использования. Это может быть имя интерфейса, IP-адрес или имя хоста.

( curl\_setopt($ch, CURLOPT\_INTERFACE, $extip) )

* **CURLOPT\_KRB4LEVEL**: Передаёт KRB4 (Kerberos 4) уровень секретности. Это любая из следующих строк (в порядке от менее до более мощной): "clear", "safe", "confidential", "private".  
  Если эта строка не совпадает с какой-либо из указанных, то используется "private". Если вы установите здесь NULL, это отключит KRB4-безопасность. KRB4-безопасность работает в настоящее время только с транзакциями FTP.
* **CURLOPT\_HTTPHEADER**: Передаёт массив полей HTTP-header"а для установки.
* **CURLOPT\_QUOTE**: Передаёт массив FTP-команд для выполнения на сервере до выполнения FTP-запроса.
* **CURLOPT\_POSTQUOTE**: Передаёт массив FTP-команд для выполнения на сервере после выполнения FTP-запроса.

Следующие опции ожидают дескриптора файла, который получается с помощью функции [**fopen()**](https://htmlweb.ru/php/php_file.php#fopen):

* **CURLOPT\_FILE**: Файл, куда должен быть помещён вывод вашего трансфера, по умолчанию это STDOUT.
* **CURLOPT\_INFILE**: Файл, из которого приходит ввод вашего трансфера.
* **CURLOPT\_WRITEHEADER**: Файл для записи header-части вывода.
* **CURLOPT\_STDERR**: Файл для записи ошибок, вместо stderr.

Параметр ***value*** должен быть функцией следующего вида ***long write\_callback (resource ch, string data){ ... return strlen($data);}*** для следующих значений параметра ***option***:

* **CURLOPT\_WRITEFUNCTION**: .
* **CURLOPT\_HEADERFUNCTION**: .

Параметр ***value*** должен быть функцией следующего вида ***string read\_callback (resource ch, resource fd, long length){}*** для следующих значений параметра ***option***:

* **CURLOPT\_READFUNCTION**: .

## **curl\_exec**

curl\_exec - выполняет CURL-сессию.

### **Описание**

bool **curl\_exec** (resource ch)

Эта функция должна вызываться после того, как вы инициализируете сессию CURL и все опции этой сессии уже установлены. Её назначение в том, чтобы просто выполнить предопределённую CURL-сессию (заданную в параметре ***ch***).

## **curl\_close**

curl\_close - закрывает CURL-сессию.

### **Описание**

void **curl\_close** ( resource ch)

Эта функция закрывает сессию CURL и освобождает все ресурсы. CURL-дескриптор ***ch*** также удаляется.

## **curl\_errno**

curl\_errno - возвращает целое число, содержащее номер последней ошибки.

### **Описание**

int **curl\_errno** ( resource ch)

## **curl\_error**

curl\_error - возвращает строку содержащую номер последней ошибки для текущей сессии.

### **Описание**

string **curl\_error** (resource ch)

## **curl\_getinfo**

curl\_getinfo - получает информацию, касающуюся специфической передачи/transfer.

### **Описание**

string **curl\_getinfo** (resource ch, int opt)

Возвращает информацию о последней операции, ***opt*** может быть одной из следующих констант:

* **CURLINFO\_EFFECTIVE\_URL** - Последний использованный URL
* **CURLINFO\_HTTP\_CODE** - Последний полученный код HTTP
* **CURLINFO\_FILETIME** - Дата модификации загруженного документа, если она неизвестна, возвращается -1.
* **CURLINFO\_TOTAL\_TIME** - Полное время выполнения операции в секундах.
* **CURLINFO\_NAMELOOKUP\_TIME** - Время разрешения имени сервера в секундах.
* **CURLINFO\_CONNECT\_TIME** - Время, затраченное на установку соединения, в секундах
* **CURLINFO\_PRETRANSFER\_TIME** - Время, прошедшее от начала операции до готовности к фактической передаче данных, в секундах
* **CURLINFO\_STARTTRANSFER\_TIME** - Время, прошедшее от начала операции до момента передачи первого байта данных, в секундах
* **CURLINFO\_REDIRECT\_TIME** - Общее время, затраченное на перенапрвления, в секундах
* **CURLINFO\_SIZE\_UPLOAD** - Количество байт при закачке
* **CURLINFO\_SIZE\_DOWNLOAD** - Количество байт при загрузке
* **CURLINFO\_SPEED\_DOWNLOAD** - Средняя скорость закачки
* **CURLINFO\_SPEED\_UPLOAD** - Средняя скорость загрузки
* **CURLINFO\_HEADER\_SIZE** - Суммарный размер всех полученных заголовков
* **CURLINFO\_REQUEST\_SIZE** - Суммарный размер всех отправленных запросов, в настоящее время используется только для HTTP запросов
* **CURLINFO\_SSL\_VERIFYRESULT** - Результат проверки SSL сертификата, запрошенной с помощью установки параметра CURLOPT\_SSL\_VERIFYPEER
* **CURLINFO\_CONTENT\_LENGTH\_DOWNLOAD** - размер загруженного документа, прочитанный из заголовка Content-Length
* **CURLINFO\_CONTENT\_LENGTH\_UPLOAD** - Размер закачиваемых данных
* **CURLINFO\_CONTENT\_TYPE** - Содержимое полученного заголовка Content-type, или NULL в случае, когда этот заголовок не был получен

При вызове без необязательного аргумента ***opt*** возвращается ассоциативный массив со следующими индексами, которые соответствуют значениям аргумента ***opt***:

* "url"
* "content\_type"
* ["http\_code"](https://htmlweb.ru/other/http_code.php)
* "header\_size"
* "request\_size"
* "filetime"
* "ssl\_verify\_result"
* "redirect\_count"
* "total\_time"
* "namelookup\_time"
* "connect\_time"
* "pretransfer\_time"
* "size\_upload"
* "size\_download"
* "speed\_download"
* "speed\_upload"
* "download\_content\_length"
* "upload\_content\_length"
* "starttransfer\_time"
* "redirect\_time"

##### Пример использования curl\_getinfo:

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, "http://www.example.com/");

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

curl\_setopt($ch, CURLINFO\_HEADER\_OUT, true); // если этот параметр не указать не работает!

curl\_exec($ch);

var\_dump(curl\_getinfo($ch,CURLINFO\_HEADER\_OUT));

##### Пример использования curl\_getinfo:

$ch = curl\_init(); // create cURL handle (ch)

if (!$ch) {

die("Couldn't initialize a cURL handle");

}

// set some cURL options

$ret = curl\_setopt($ch, CURLOPT\_URL, "http://mail.yahoo.com");

$ret = curl\_setopt($ch, CURLOPT\_HEADER, 1);

$ret = curl\_setopt($ch, CURLOPT\_FOLLOWLOCATION, 1);

$ret = curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 0);

$ret = curl\_setopt($ch, CURLOPT\_TIMEOUT, 30);

// execute

$ret = curl\_exec($ch);

if (empty($ret)) {

// some kind of an error happened

die(curl\_error($ch));

curl\_close($ch); // close cURL handler

} else {

$info = curl\_getinfo($ch);

curl\_close($ch); // close cURL handler

if (empty($info['http\_code'])) {

die("No HTTP code was returned");

} else {

// load the HTTP codes

$http\_codes = parse\_ini\_file("path/to/the/ini/file/I/pasted/above");

// echo results

echo "The server responded: <br />";

echo $info['http\_code'] . " " . $http\_codes[$info['http\_code']];

}

}

## **curl\_version**

curl\_version - возвращает версию текущей CURL.

### **Описание**

string **curl\_version** (void)

Функция **curl\_version()** возвращает строку с текущей версией CURL.

curl\_version=Array  
(  
[version\_number] => 466176  
[age] => 3  
[features] => 34493  
[ssl\_version\_number] => 0  
[version] => 7.29.0  
[host] => x86\_64-redhat-linux-gnu  
[ssl\_version] => NSS/3.19.1 Basic ECC  
[libz\_version] => 1.2.7  
[protocols] => Array  
(  
[0] => dict  
[1] => file  
[2] => ftp  
[3] => ftps  
[4] => gopher  
[5] => http  
[6] => https  
[7] => imap  
[8] => imaps  
[9] => ldap  
[10] => ldaps  
[11] => pop3  
[12] => pop3s  
[13] => rtsp  
[14] => scp  
[15] => sftp  
[16] => smtp  
[17] => smtps  
[18] => telnet  
[19] => tftp  
)  
  
)

## **Предопределённые константы**

Эти константы определены данным расширением и будут доступны только в том случае, если либо вкомпилированы в РНР, либо динамически загружены на этапе прогона.

**CURLOPT\_PORT (integer)**

**CURLOPT\_FILE (integer)**

**CURLOPT\_INFILE (integer)**

**CURLOPT\_INFILESIZE (integer)**

**CURLOPT\_URL (integer)**

**CURLOPT\_PROXY (integer)**

**CURLOPT\_VERBOSE (integer)**

**CURLOPT\_HEADER (integer)**

**CURLOPT\_HTTPHEADER (integer)**

**CURLOPT\_NOPROGRESS (integer)**

**CURLOPT\_NOBODY (integer)**

**CURLOPT\_FAILONERROR (integer)**

**CURLOPT\_UPLOAD (integer)**

**CURLOPT\_POST (integer)**

**CURLOPT\_FTPLISTONLY (integer)**

**CURLOPT\_FTPAPPEND (integer)**

**CURLOPT\_NETRC (integer)**

**CURLOPT\_FOLLOWLOCATION (integer)**

**CURLOPT\_FTPASCII (integer)**

**CURLOPT\_PUT (integer)**

**CURLOPT\_MUTE (integer)**

**CURLOPT\_USERPWD (integer)**

**CURLOPT\_PROXYUSERPWD (integer)**

**CURLOPT\_RANGE (integer)**

**CURLOPT\_TIMEOUT (integer)**

**CURLOPT\_POSTFIELDS (integer)**

**CURLOPT\_REFERER (integer)**

**CURLOPT\_USERAGENT (integer)**

**CURLOPT\_FTPPORT (integer)**

**CURLOPT\_LOW\_SPEED\_LIMIT (integer)**

**CURLOPT\_LOW\_SPEED\_TIME (integer)**

**CURLOPT\_RESUME\_FROM (integer)**

**CURLOPT\_COOKIE (integer)**

**CURLOPT\_SSLCERT (integer)**

**CURLOPT\_SSLCERTPASSWD (integer)**

**CURLOPT\_WRITEHEADER (integer)**

**CURLOPT\_SSL\_VERIFYHOST (integer)**

**CURLOPT\_COOKIEFILE (integer)**

**CURLOPT\_SSLVERSION (integer)**

**CURLOPT\_TIMECONDITION (integer)**

**CURLOPT\_TIMEVALUE (integer)**

**CURLOPT\_CUSTOMREQUEST (integer)**

**CURLOPT\_STDERR (integer)**

**CURLOPT\_TRANSFERTEXT (integer)**

**CURLOPT\_RETURNTRANSFER (integer)**

**CURLOPT\_QUOTE (integer)**

**CURLOPT\_POSTQUOTE (integer)**

**CURLOPT\_INTERFACE (integer)**

**CURLOPT\_KRB4LEVEL (integer)**

**CURLOPT\_HTTPPROXYTUNNEL (integer)**

**CURLOPT\_FILETIME (integer)**

**CURLOPT\_WRITEFUNCTION (integer)**

**CURLOPT\_READFUNCTION (integer)**

**CURLOPT\_PASSWDFUNCTION (integer)**

**CURLOPT\_HEADERFUNCTION (integer)**

**CURLOPT\_MAXREDIRS (integer)**

**CURLOPT\_MAXCONNECTS (integer)**

**CURLOPT\_CLOSEPOLICY (integer)**

**CURLOPT\_FRESH\_CONNECT (integer)**

**CURLOPT\_FORBID\_REUSE (integer)**

**CURLOPT\_RANDOM\_FILE (integer)**

**CURLOPT\_EGDSOCKET (integer)**

**CURLOPT\_CONNECTTIMEOUT (integer)**

**CURLOPT\_SSL\_VERIFYPEER (integer)**

**CURLOPT\_CAINFO (integer)**

**CURLOPT\_COOKIEJAR (integer)**

**CURLOPT\_SSL\_CIPHER\_LIST (integer)**

**CURLOPT\_BINARYTRANSFER (integer)**

**CURLCLOSEPOLICY\_LEAST\_RECENTLY\_USED (integer)**

**CURLCLOSEPOLICY\_LEAST\_TRAFFIC (integer)**

**CURLCLOSEPOLICY\_SLOWEST (integer)**

**CURLCLOSEPOLICY\_CALLBACK (integer)**

**CURLCLOSEPOLICY\_OLDEST (integer)**

**CURLINFO\_EFFECTIVE\_URL (integer)**

**CURLINFO\_HTTP\_CODE (integer)**

**CURLINFO\_HEADER\_SIZE (integer)**

**CURLINFO\_REQUEST\_SIZE (integer)**

**CURLINFO\_TOTAL\_TIME (integer)**

**CURLINFO\_NAMELOOKUP\_TIME (integer)**

**CURLINFO\_CONNECT\_TIME (integer)**

**CURLINFO\_PRETRANSFER\_TIME (integer)**

**CURLINFO\_SIZE\_UPLOAD (integer)**

**CURLINFO\_SIZE\_DOWNLOAD (integer)**

**CURLINFO\_SPEED\_DOWNLOAD (integer)**

**CURLINFO\_SPEED\_UPLOAD (integer)**

**CURLINFO\_FILETIME (integer)**

**CURLINFO\_SSL\_VERIFYRESULT (integer)**

**CURLINFO\_CONTENT\_LENGTH\_DOWNLOAD (integer)**

**CURLINFO\_CONTENT\_LENGTH\_UPLOAD (integer)**

**CURLE\_OK (integer)**

**CURLE\_UNSUPPORTED\_PROTOCOL (integer)**

**CURLE\_FAILED\_INIT (integer)**

**CURLE\_URL\_MALFORMAT (integer)**

**CURLE\_URL\_MALFORMAT\_USER (integer)**

**CURLE\_COULDNT\_RESOLVE\_PROXY (integer)**

**CURLE\_COULDNT\_RESOLVE\_HOST (integer)**

**CURLE\_COULDNT\_CONNECT (integer)**

**CURLE\_FTP\_WEIRD\_SERVER\_REPLY (integer)**

**CURLE\_FTP\_ACCESS\_DENIED (integer)**

**CURLE\_FTP\_USER\_PASSWORD\_INCORRECT (integer)**

**CURLE\_FTP\_WEIRD\_PASS\_REPLY (integer)**

**CURLE\_FTP\_WEIRD\_USER\_REPLY (integer)**

**CURLE\_FTP\_WEIRD\_PASV\_REPLY (integer)**

**CURLE\_FTP\_WEIRD\_227\_FORMAT (integer)**

**CURLE\_FTP\_CANT\_GET\_HOST (integer)**

**CURLE\_FTP\_CANT\_RECONNECT (integer)**

**CURLE\_FTP\_COULDNT\_SET\_BINARY (integer)**

**CURLE\_PARTIAL\_FILE (integer)**

**CURLE\_FTP\_COULDNT\_RETR\_FILE (integer)**

**CURLE\_FTP\_WRITE\_ERROR (integer)**

**CURLE\_FTP\_QUOTE\_ERROR (integer)**

**CURLE\_HTTP\_NOT\_FOUND (integer)**

**CURLE\_WRITE\_ERROR (integer)**

**CURLE\_MALFORMAT\_USER (integer)**

**CURLE\_FTP\_COULDNT\_STOR\_FILE (integer)**

**CURLE\_READ\_ERROR (integer)**

**CURLE\_OUT\_OF\_MEMORY (integer)**

**CURLE\_OPERATION\_TIMEOUTED (integer)**

**CURLE\_FTP\_COULDNT\_SET\_ASCII (integer)**

**CURLE\_FTP\_PORT\_FAILED (integer)**

**CURLE\_FTP\_COULDNT\_USE\_REST (integer)**

**CURLE\_FTP\_COULDNT\_GET\_SIZE (integer)**

**CURLE\_HTTP\_RANGE\_ERROR (integer)**

**CURLE\_HTTP\_POST\_ERROR (integer)**

**CURLE\_SSL\_CONNECT\_ERROR (integer)**

**CURLE\_FTP\_BAD\_DOWNLOAD\_RESUME (integer)**

**CURLE\_FILE\_COULDNT\_READ\_FILE (integer)**

**CURLE\_LDAP\_CANNOT\_BIND (integer)**

**CURLE\_LDAP\_SEARCH\_FAILED (integer)**

**CURLE\_LIBRARY\_NOT\_FOUND (integer)**

**CURLE\_FUNCTION\_NOT\_FOUND (integer)**

**CURLE\_ABORTED\_BY\_CALLBACK (integer)**

**CURLE\_BAD\_FUNCTION\_ARGUMENT (integer)**

**CURLE\_BAD\_CALLING\_ORDER (integer)**

**CURLE\_HTTP\_PORT\_FAILED (integer)**

**CURLE\_BAD\_PASSWORD\_ENTERED (integer)**

**CURLE\_TOO\_MANY\_REDIRECTS (integer)**

**CURLE\_UNKNOWN\_TELNET\_OPTION (integer)**

**CURLE\_TELNET\_OPTION\_SYNTAX (integer)**

**CURLE\_OBSOLETE (integer)**

**CURLE\_SSL\_PEER\_CERTIFICATE (integer)**

## **Примеры**

После того как вы скомпилировали PHP с поддержкой CURL, вы можете начать использовать curl-функций. Основная идея функций CURL - вы инициализируете CURL-сессию с использованием [**curl\_init()**](https://htmlweb.ru/php/php_curl.php#curl_init), затем можно установить все ваши опции для трансфера через использование [**curl\_exec()**](https://htmlweb.ru/php/php_curl.php#curl_exec), а затем закрыть сессию функцией [**curl\_close()**](https://htmlweb.ru/php/php_curl.php#curl_close).

##### Пример 1. Инициализация новой сессии CURL и получение web-страницы.

<?php

$ch = curl\_init();

curl\_setopt ($ch, CURLOPT\_URL, "http://www.example.com/");

curl\_setopt ($ch, CURLOPT\_HEADER, 0);

curl\_exec ($ch);

curl\_close ($ch);

?>

##### Пример 2. Использование модуля CURL РНР для получения example.com

<?php

$ch = curl\_init ("http://www.example.com/");

$fp = fopen ("example\_homepage.txt", "w");

curl\_setopt ($ch, CURLOPT\_FILE, $fp);

curl\_setopt ($ch, CURLOPT\_HEADER, 0);

curl\_exec ($ch);

curl\_close ($ch);

fclose ($fp);

?>

##### Пример 3. Проверка доступности URL с помощью CURL РНР

<?php

function check\_url($url) {

$c = curl\_init();

curl\_setopt($c, CURLOPT\_URL, $url);

curl\_setopt($c, CURLOPT\_HEADER, 1); // читать заголовок

curl\_setopt($c, CURLOPT\_NOBODY, 1); // читать ТОЛЬКО заголовок без тела

curl\_setopt($c, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt($c, CURLOPT\_FRESH\_CONNECT, 1); // не использовать cache

if (!curl\_exec($c)) return false;

$httpcode = curl\_getinfo($c, CURLINFO\_HTTP\_CODE);

return ($httpcode < 400);

}

?>

##### Пример 4. Отделение заголовка от тела, полученного с помощью CURL РНР

<?php

$header\_size = curl\_getinfo($ch, CURLINFO\_HEADER\_SIZE);

$headers = substr($output, 0, $header\_size - 4);

$body = substr($output, $header\_size);

?>

##### Пример 5. Определение адреса перехода URL с помощью CURL РНР

the problem: curl\_setopt($ch,FOLLOW\_LOCATION,1); the error: trouble with open\_basedir and safe\_mode the solution: a function already developed by someone the solution n 2: the same function, modifed, works great for me..

<?php

function curl\_redir\_exec($ch)

{

static $curl\_loops = 0;

static $curl\_max\_loops = 20;

if ($curl\_loops++ >= $curl\_max\_loops)

{

$curl\_loops = 0;

return FALSE;

}

curl\_setopt($ch, CURLOPT\_HEADER, true);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

$data = curl\_exec($ch);

list($header, $data) = explode("\n\n", $data, 2);

$http\_code = curl\_getinfo($ch, CURLINFO\_HTTP\_CODE);

if ($http\_code == 301 || $http\_code == 302)

{

$matches = [];

preg\_match('/Location:(.\*?)\n/', $header, $matches);

$url = @parse\_url(trim(array\_pop($matches)));

if (!$url)

{

//couldn't process the url to redirect to

$curl\_loops = 0;

return $data;

}

$last\_url = parse\_url(curl\_getinfo($ch, CURLINFO\_EFFECTIVE\_URL));

if (!$url['scheme'])

$url['scheme'] = $last\_url['scheme'];

if (!$url['host'])

$url['host'] = $last\_url['host'];

if (!$url['path'])

$url['path'] = $last\_url['path'];

$new\_url = $url['scheme'] . '://' . $url['host'] . $url['path'] . ($url['query']?'?'.$url['query']:'');

curl\_setopt($ch, CURLOPT\_URL, $new\_url);

debug('Redirecting to', $new\_url);

return curl\_redir\_exec($ch);

} else {

$curl\_loops=0;

return $data;

}

}

?>

just use this function without de FOLLOW\_LOCATION and should work. the problem was that when you get to the line where you return the data if http\_code was different than 301 oe 302, $data has obsolete information or none. so $debbbb does the job. 

##### Пример 6. Разбор cookie из заголовка с помощью CURL РНР

Sometimes you can't use CURLOPT\_COOKIEJAR and CURLOPT\_COOKIEFILE becoz of the server php-settings(They say u may grab any files from server using these options). Here is the solution 1)Don't use CURLOPT\_FOLLOWLOCATION 2)Use curl\_setopt($ch, CURLOPT\_HEADER, 1) 3)Grab from the header cookies like this:

preg\_match\_all('|Set-Cookie: (.\*);|U', $content, $results);

$cookies = implode(';', $results[1]);

4)Set them using curl\_setopt($ch, CURLOPT\_COOKIE, $cookies); 

##### Пример 7. Разбор cookie из заголовка с помощью CURL РНР

As Yevgen mentioned earlier sometimes we can't use CURLOPT\_COOKIEJAR and CURLOPT\_COOKIEFILE. Below is a header callback function I wrote back in January that lets you maintain cookies between cURL requests. Cookies are added to $ch during all requests even during redirection, so you can use it together with CURLOPT\_FOLLOWLOCATION. Here is the code:

function read\_header($ch, $string)

{

global $location; #keep track of location/redirects

global $cookiearr; #store cookies here

global $ch;

# ^overrides the function param $ch

# this is okay because we need to

# update the global $ch with

# new cookies

$length = strlen($string);

if(!strncmp($string, "Location:", 9))

{ #keep track of last redirect

$location = trim(substr($string, 9, -1));

}

if(!strncmp($string, "Set-Cookie:", 11))

{ #get the cookie

$cookiestr = trim(substr($string, 11, -1));

$cookie = explode(';', $cookiestr);

$cookie = explode('=', $cookie[0]);

$cookiename = trim(array\_shift($cookie));

$cookiearr[$cookiename] = trim(implode('=', $cookie));

}

$cookie = "";

if(trim($string) == "")

{ #execute only at end of header

foreach ($cookiearr as $key=>$value)

{

$cookie .= "$key=$value; ";

}

curl\_setopt($ch, CURLOPT\_COOKIE, $cookie);

}

return $length;

}

curl\_setopt($ch, CURLOPT\_HEADERFUNCTION, 'read\_header');

This code assumes that you will reuse $ch without initializing it every time (call curl\_init only once, in the beginning). If you need to initialize $ch again at any point in your code you can access the currently stored cookies in $cookiearr and include them in the new $ch. I wrote this function before I had enough experience with regular expressions so you won't find any preg\_match calls here. I have used this code for quite a while and without any problems accessing gmail, yahoo, hotmail, aol etc. where I had to go through login and a few pages before getting to what I was looking for.

##### Пример 8. Установка обработчика загрузки заголовка с помощью CURL РНР

Using cURL, I needed to call a third-party script which was returning binary data as attachment to pass on retrieved data again as attachment. Problem was that the third-party script occassionally returned HTTP errors and I wanted to avoid passing on zero-length attachment in such case. Combination of using CURLOPT\_FAILONERROR and CURLOPT\_HEADERFUNCTION callback helped to process the third-party script HTTP errors neatly:

function curlHeaderCallback($resURL, $strHeader) {

if (preg\_match('/^HTTP/i', $strHeader)) {

header($strHeader);

header('Content-Disposition: attachment; filename="file-name.zip"');

}

return strlen($strHeader);

}

$resURL = curl\_init('http://htmlweb.ru/');

curl\_setopt($resURL, CURLOPT\_BINARYTRANSFER, 1);

curl\_setopt($resURL, CURLOPT\_HEADERFUNCTION, 'curlHeaderCallback');

curl\_setopt($resURL, CURLOPT\_FAILONERROR, 1);

curl\_exec ($resURL);

$intReturnCode = curl\_getinfo($resURL, CURLINFO\_HTTP\_CODE);

curl\_close ($resURL);

if ($intReturnCode != 200) {

print 'Ошибка с кодом: ' . $intReturnCode;

}

##### Пример 9. Сохранение страницы в файл с подсчетом скорости передачи с помощью CURL РНР

WritePageToFile( 'http://es.php.net', 'es.php.net.txt' );

function WritePageToFile( $sHTMLpage, $sTxtfile ) {

$sh = curl\_init( $sHTMLpage );

$hFile = FOpen( $sTxtfile, 'w' );

curl\_setopt( $sh, CURLOPT\_FILE, $hFile );

curl\_setopt( $sh, CURLOPT\_HEADER, 0 );

curl\_exec ( $sh );

$sAverageSpeedDownload = curl\_getInfo( $sh, CURLINFO\_SPEED\_DOWNLOAD );

$sAverageSpeedUpload = curl\_getInfo( $sh, CURLINFO\_SPEED\_UPLOAD );

echo '<pre>';

echo 'Average speed download == ' . $sAverageSpeedDownload . '<br>';

echo 'Average Speed upload == ' . $sAverageSpeedUpload . '<br>';

echo '<br>';

$aCURLinfo = curl\_getInfo( $sh );

print\_r( $aCURLinfo );

echo '</pre>';

curl\_close( $sh );

FClose ( $hFile );

echo '(<b>See the file "'.$sTxtfile.'" in the same path of the hosting'.

' to where this script PHP</b>).<br>';

}

##### Пример 9. Получение страницы через SSL соединение (https)

$ch=curl\_init('https://htmlweb.ru');

curl\_setopt($ch, CURLOPT\_HEADER, 0);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

// Отключить ошибку "SSL certificate problem, verify that the CA cert is OK"

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, 0);

// Отключить ошибку "SSL: certificate subject name 'hostname.ru' does not match target host name '123.123'"

curl\_setopt($ch, CURLOPT\_SSL\_VERIFYHOST, 0);

$out=curl\_exec($ch);

    //$http\_code = curl\_getinfo($ch, CURLINFO\_HTTP\_CODE);

    //echo "http\_code=".$http\_code,', err='.curl\_error($ch);

curl\_close($ch);

##### Пример 10. Использование сессий и cookie в curl

$cookie\_filename=sys\_get\_temp\_dir()+'/cookie.tmp';

$curl=curl\_init('http://htmlweb.ru');

curl\_setopt($curl, CURLOPT\_RETURNTRANSFER, 1);

curl\_setopt($curl, CURLOPT\_COOKIEJAR, $cookie\_filename);//сохранять полученные COOKIE в файл

curl\_setopt($curl, CURLOPT\_COOKIEFILE, $cookie\_filename); //отсылаем серверу COOKIE полученные от него при авторизации

$out=curl\_exec($curl);

##### Пример 11. Отправка файла и многомерный массив в Curl. CURLOPT\_POSTFIELDS + CurlFile

Если Вам нужно будет отправить в POST запросе многомерный массив и файл, то вы столкнетесь с неразрешимой проблемой. Если передавать в CURLOPT\_POSTFIELDS многомерный массив, то второй уровень будет передан как строка "Array". Если преобразовать с помощью http\_build\_query, то Вы не сможете передать файл.

Ниже представлена функция кодирования двумерного массива с подгрузкой файлов для Curl, которая будет работать как в старых версиях PHP 5.3, PHP 5.4, так и в PHP 5.6

/\*\* преобразует многомерный массив в одномерный, используйя сложные индексы и заменяет @ в префиксе на CurlFile для испрользоания в Curl

\* @param $inputArray

\* @param string $inputKey

\* @return array

$requestVars = array(

'id' => array(1, 2,'id'=>1234),

'name' => 'log',

'logfile' => '@/tmp/test.log');

получим:

["id[0]"]=> int(1)

["id[1]"]=> int(2)

["id[id]"]=> int(1234)

["name"]=> string(3) "log"

["logfile"]=> string(13) "/tmp/test.log" }

\*/

function convertToStringArray($inputArray, $inputKey='') {

$resultArray=[];

foreach ($inputArray as $key => $value) {

$tmpKey = (bool)$inputKey ? $inputKey."[$key]" : $key;

if (is\_array($value)) {

$resultArray+=convertToStringArray($value, $tmpKey);

} elseif ($value[0] == '@'){

$resultArray[$tmpKey] = (class\_exists(' CURLFile ', false)) ? new CurlFile(ltrim($value, '@')) : $value;

} else {

$resultArray[$tmpKey] = $value;

}

}

return $resultArray;

}

// проверяем

$requestVars = array(

'id' => array(1, 2,'id'=>1234),

'name' => 'log',

'logfile' => '@/tmp/test.log');

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, 'htmlweb.ru');

curl\_setopt($ch, CURLOPT\_POST, 1);

curl\_setopt($ch, CURLOPT\_POSTFIELDS, convertToStringArray($requestVars));

$res = curl\_exec($ch);

curl\_close($ch);

# IMAP, POP3 и NNTP

В этом разделе описаны функции работы с почтой и почтовыми серверами по протоколам IMAP, POP3 и NNTP

Для большего понимания методики работы этих функций будет полезно ознакомиться с перечисленными ниже документами RFC:

* RFC821: Simple Mail Transfer Protocol (SMTP).
* RFC822: Стандарт текстовых сообщений интернет ARPA.
* RFC2060: Internet Message Access Protocol (IMAP) Version 4rev1.
* RFC1939: Post Office Protocol Version 3 (POP3).
* RFC977: Network News Transfer Protocol (NNTP).
* RFC2076: Common Internet Message Headers.
* RFC2045 , RFC2046 , RFC2047 , RFC2048 & RFC2049: Multipurpose Internet Mail Extensions (MIME).

**imap\_open**. Подключение к серверу и открытие почтового ящика

int imap\_open (string mailbox, string username, string password [, int flags])

Функция возвращает дескриптор открытого почтового ящика IMAP или значение FALSE при возникновении ошибки. Эту функцию можно также использовать для подключения к серверам POP3 и NNTP, но в этом случае некоторые функции пакета будут недоступны.

В аргументе *mailbox* указывается имя сервера и путь к почтовому ящику. Имя сервера заключается в фигурные скобки и состоит из имени сервера или его IP-адреса, необязательного имени протокола, которое надо предварять символом слеша, и необязательного значения номера порта подключения, которое начинается с двоеточия. Имя сервера необходимо указывать всегда. Иногда бывает полезно указывать имя сервера в переменной, но следует учитывать, что фигурные скобки имеют специальное значение для переменных в строках в двойных кавычках, поэтому открывающую скобку следует предварять слешем, например \{$SERVER/ pop3:110}INBOX.

Специальное имя INBOX обозначает основной ящик пользователя и папку входящих сообщений. Если имя ящика включает дополнительные символы помимо стандартных латинских символов, то имя предварительно нужно закодировать функцией imap\_utf7\_encode(). Разделителем иерархии почтовых ящиков обычно является слеш, но для некоторых серверов им также может быть точка.

При помощи аргументов *username* и *password* указываются имя и пароль пользователя.

В аргументе *flags* указываются параметры открытия ящика в виде битовой маски. Эти параметры приведены в следующем списке.

* OP\_READONLY - открытие ящика для чтения
* OP\_ANONYMOUS - не использовать разделы новостей .newsrc. Применяется только для NNTP.
* OP\_HALFOPEN - применяется для имен IMAP и NNTP, если необходимо осуществить подключение без открытия ящика.
* CL\_EXPUNGE - автоматически удалить ящик при его закрытии. См. также imap\_delete() и imap\_expunge().

Для подключения к серверу IMAP, использующему на локальной машине порт 143, можно использовать следующий вызов:

$mbox = imap\_open ("{localhost:143}INBOX", "user\_id", "password");

Для подключения к серверу POP3, использующему на локальной машине порт 110, используется следующий вызов:

$mbox = imap\_open ("{localhost/pop3:110}INBOX", "user\_id", "password");

Для подключения к серверу NNTP на локальной машине, использующему порт 119, используется следующий вызов:

$nntp = imap\_open ("{localhost/nntp:119}comp.test", "", "");

Следующий пример, демонстрирует подключение к серверу IMAP your.imap.host и вывод имеющихся ящиков и сообщений для пользователя username.

$mbox = imap\_open ("{your.imap.host:143}", "username", "password");

echo "<p><h1>Ящики Mailboxes</h1>\n";

$folders = imap\_listmailbox ($mbox, "{your.imap.host:143}", "\*");

if ($folders == FALSE) echo "Ошибка ящиков не обнаружено <br>\n";

else

while (list ($key, $val) = each ($folders))

echo "\t $val <br>\n";

echo "<p><h1>Сообщения в INBOX</h1>\n";

$headers = imap\_headers ($mbox);

if ($headers == FALSE) { echo "Ошибка - сообщений нет <br>\n"; }

else

while (list ($key,$val) = each ($headers))

echo "\t $val <br>\n";

imap\_close($mbox);

После работы этого кода на экран будут выведены следующие строки:

Ящики Mailboxes

{localhost:143}linnn/igf<br>

{localhost:143}linnn<br>

{localhost:143}INBOX<br>

Сообщения в INBOX

1)13-May-2001 Имя отправителя Заголовок (1621 chars)<br>

**imap\_reopen.** Переподключение и выбор другого ящика

int imap\_reopen (int imap\_stream, string mailbox [, string flags])

Функция аналогична imap\_open(), но она не создает новый дескриптор подключения, а использует уже существующий дескриптор *imap\_stream*. Имя пользователя и его пароль не изменяются. В случае успешного выполнения возвращается значение TRUE. При возникновении ошибки возвращается значение FALSE.

**imap\_close.** Отключение от сервера

int imap\_close (int imap\_stream [, int flags])

Функция закрывает поток подключения, связанный с дескриптором *imap\_stream*. Если в аргументе *flags* указывается значение CL\_EXPUNGE, то при закрытии ящика из него автоматически удаляются все сообщения, помеченные на удаление.

**imap\_ping**. Проверка активности подключения

int imap\_ping (int imap\_stream)

Функция возвращает значение TRUE, если подключение все еще действительно. Если оно разорвано, то возвращается значение FALSE.

Вызов функции может использоваться для обнаружения свежих сообщений и для поддержания активности связи с серверами, разрывающими связь через определенный промежуток времени бездействия. Но если учитывать, что время тайм-аута сценария обычно не превышает минуты, а время разрыва связи наступает через сравнительно длительный период, то едва ли эта функция будет полезна при работе веб-сценариев.

**imap\_listmailbox.** Получение списка почтовых ящиков

array imap\_listmailbox (int imap\_stream, string ref, string pattern)

Функция возвращает массив имен почтовых ящиков. Аргументы рассматривались в описании функции imap\_getmailboxes().

$mbox = imap\_open("{your.imap.host}","username","password",OP\_HALFOPEN)

|| die("can't connect: ".imap\_last\_error());

$list = imap\_listmailbox($mbox,"{your.imap.host}","\*");

if(is\_array($list)) {

reset($list);

while (list($key, $val) = each($list))

print imap\_utf7\_decode($val)."<br>\n";

}

else

print "imap\_listmailbox failed: ".imap\_last\_error()."\n";

imap\_close($mbox);

**imap\_getmailboxes.** Получение описания почтовых ящиков

array imap\_getmailboxes (int imap\_stream, string ref, string pattern)

Функция возвращает массив объектов, содержащих описания каждого ящика. Каждый объект имеет три атрибута, в которых указывается полное имя ящика name, разделитель иерархии ящиков delimiter и битовое поле attributes, которое может состоять из компонентов, перечисленных в следующем списке:

* LATT\_NOINFERIORS (1) - ящик не имеет вложенных ящиков.
* LATT\_NOSELECT (2) - контейнер, который нельзя открыть.
* LATT\_MARKED (4) - ящик маркирован. Используется только с параметром UW-IMAPD.
* LATT\_UNMARKED (8) - ящик маркирован. Используется только с параметром UW-IMAPD.

Имена ящиков могут содержать символы национальных алфавитов из набора ASCII, которые могут быть закодированы и затем декодированы функцией imap\_ utf7\_decode().

Аргумент *ref* должен содержать обычную спецификацию сервера, как в функции imap\_open(), а аргумент *pattern* указывает, с какого ящика следует начинать перебор значений, а также шаблон для возвращаемых ящиков.

В аргументе *pattern* могут быть указаны два специальных значения. Символ звездочки означает, что следует возвращать всю иерархию ящиков, а символ процента заставляет функцию возвращать только текущий иерархический уровень без вложенных ящиков.

Пример использования функции:

$mbox = imap\_open("{your.imap.host}","username","password",OP\_HALFOPEN)

|| die("can't connect: ".imap\_last\_error());

$list = imap\_getmailboxes($mbox,"{your.imap.host}","\*");

if(is\_array($list)) {

print\_r ($list);

}

else

print "imap\_getmailboxes failed: ".imap\_last\_error()."\n";

imap\_close($mbox);

См. также imap\_getsubscribed().

**imap\_createmailbox.** Создание почтового ящика

int imap\_createmailbox (int imap\_stream, string mbox)

Функция предназначена для создания новых почтовых ящиков. Имя создаваемого ящика указывается в аргументе *mbox*. Имена, содержащие национальные символы, должны кодироваться функцией imap\_utf7\_encode().

В случае успешного выполнения возвращается значение TRUE. При возникновении ошибки возвращается значение FALSE.

См. также imap\_renamemailbox(), imap\_deletemailbox() и imap\_open().

Пример использования функции:

$mbox = imap\_open("{your.imap.host}","username","password",OP\_HALFOPEN)

|| die("can't connect: ".imap\_last\_error());

$name1 = "newbox";

$name2 = imap\_utf7\_encode("мой личный ящик");

if(@imap\_createmailbox($mbox,

imap\_utf7\_encode("{your.imap.host}INBOX/$name1"))) {

$status = @imap\_status($mbox,"{your.imap.host}INBOX/$name1",SA\_ALL);

if($status) {

echo "Статус нового ящика '$name1': <br>\n",

"Messages: ". $status->messages ."<br>\n",

"Recent: ". $status->recent ."<br>\n",

"Unseen: ". $status->unseen ."<br>\n",

"UIDnext: ". $status->uidnext ."<br>\n",

"UIDvalidity:". $status->uidvalidity."<br>\n";

if(imap\_renamemailbox($mbox,"{your.imap.host}INBOX/$name1",

"{your.imap.host}INBOX/$name2")) {

echo "Ящик '$name1' переименован в '$name2'<br>\n";

}

else{

print "imap\_renamemailbox failed: ".imap\_last\_error()."<br>\n";

}

}

else{

print "imap\_status failed: ".imap\_last\_error()."<br>\n";

}

if(@imap\_deletemailbox($mbox,"{your.imap.host}INBOX/$name2")) {

print "new mailbox removed <br>\n";

}

else {

print "imap\_deletemailbox failed: ".

implode("<br>\n",imap\_errors())."<br>\n";

}

}

else {

print "could not create new mailbox: ".

implode("<br>\n",imap\_errors())."<br>\n";

}

imap\_close($mbox);

**imap\_renamemailboximap\_renamemailbox.** Переименование ящика

int **imap\_renamemailbox** (int *imap\_stream*, string *old\_mbox*, string *new\_mbox*)

Функция позволяет переименовать существующий ящик. Имя ящика задается аргументом *old\_mbox*, а его новое имя указывается в параметре *new\_mbox*. Формат этих аргументов рассматривался в описании функции imap\_open().

В случае успешного выполнения возвращается значение TRUE. При возникновении ошибки возвращается значение FALSE.

См. также imap\_createmailbox(), imap\_deletemailbox(), imap\_open().

**imap\_deletemailbox.** Удаление почтового ящика

int imap\_deletemailbox (int imap\_stream, string mbox)

Функция позволяет удалить существующий ящик. Имя ящика задается аргументом *mbox*. В случае успешного выполнения возвращается значение TRUE. При возникновении ошибки возвращается значение FALSE.

См. также imap\_createmailbox(), imap\_renamemailbox() и imap\_open().

**imap\_check.** Проверка текущего ящика

object imap\_check (int imap\_stream)

Функция возвращает объект, содержащий информацию о текущем почтовом ящике. При возникновении ошибки возвращается значение FALSE.

Свойства возвращаемого объекта перечислены в следующем списке:

* Date - время последнего доступа к содержимому ящика.
* Driver - протокол доступа к ящику.
* Mailbox - полное имя почтового ящика.
* Nmsgs - общее количество сообщений в ящике.
* Recent - количество новых сообщений в ящике.

Пример использования этой функции:

$mbox = imap\_open("{localhost}","igor","igx324879",OP\_HALFOPEN)

|| die("can't connect: ".imap\_last\_error());

print\_r(imap\_check ($mbox));

imap\_close($mbox);

В результате работы этого кода на экран будут выведены следующие строки:

stdClass Object

(

[Date] => Sun, 13 May 2008 14:31:38 +0400 (Московское время (лето))

[Driver] => imap

[Mailbox] => {igor:143/imap/user="igor"}<no\_mailbox>

[Nmsgs] => 0

[Recent] => 0

)

**imap\_status.** Получение статуса указанного ящика

object imap\_status (int imap\_stream, string mailbox, int options)

Функция возвращает объект, содержащий информацию о почтовом ящике с именем *mailbox*. При возникновении ошибки возвращается значение FALSE. Набор свойств возвращаемого объекта варьируется в зависимости от указанного значения аргумента *options*. Возможные значения этого параметра приведены в следующем списке:

* SA\_MESSAGES - возвратить в status->messages общее количество сообщений в ящике.
* SA\_RECENT - возвратить в status->recent количество новых сообщений.
* SA\_UNSEEN - возвратить в status->unseen количество непрочитанных сообщений.
* SA\_UIDNEXT - возвратить в status->uidnext номер uid для следующего нового сообщения.
* SA\_UIDVALIDITY - в status->uidvalidity возвратить константу действительности uid ящика.
* SA\_ALL - возвращаются все перечисленные значения. Объект также содержит свойство status->flags, отражающее в объединенном битовом поле набор параметров создания.

Пример использования функции:

$mbox = imap\_open("{your.imap.host}","username","password",OP\_HALFOPEN)

|| die("can't connect: ".imap\_last\_error());

$status = imap\_status($mbox,"{your.imap.host}INBOX",SA\_ALL);

if($status) {

print\_r($status);

}

else

print "imap\_status failed: ".imap\_lasterror()."\n";

imap\_close($mbox);

**imap\_mailboxmsginfo.** Получение информации о текущем ящике

object imap\_mailboxmsginfo (int imap\_stream)

Функция сходна с imap\_status(), но она также вычисляет объем всех сообщений, находящихся в ящике. Функция возвращает объект, свойства которого перечислены в следующем списке:

* Date - время последней модификации.
* Driver - протокол доступа к ящику.
* Mailbox - имя почтового ящика.
* Nmsgs - количество сообщений.
* Recent - количество полученных сообщений.
* Unread - количество непрочитанных сообщений.
* Deleted - количество удаленных сообщений.
* Size - размер почтового ящика.

Пример использования функции:

<?php

$mbox = imap\_open("{your.imap.host}INBOX","username", "password")

|| die("can't connect: ".imap\_last\_error());

$check = imap\_mailboxmsginfo($mbox);

if($check) {

print "Date: " . $check->Date ."<br>\n" ;

print "Driver: " . $check->Driver ."<br>\n" ;

print "Mailbox: " . $check->Mailbox ."<br>\n" ;

print "Messages: ". $check->Nmsgs ."<br>\n" ;

print "Recent: " . $check->Recent ."<br>\n" ;

print "Unread: " . $check->Unread ."<br>\n" ;

print "Deleted: " . $check->Deleted ."<br>\n" ;

print "Size: " . $check->Size ."<br>\n"

}

else {

print "imap\_check() failed: ".imap\_last\_error(). "<br>\n";

}

imap\_close($mbox);

?>

**imap\_num\_msg**. Получение количества сообщений в текущем ящике

int imap\_num\_msg (int imap\_stream)

Функция возвращает количество сообщений, хранящихся в почтовом ящике.

**imap\_num\_recent**. Получение количества новых сообщений в текущем ящике

int imap\_num\_recent (int imap\_stream)

Функция возвращает количество новых сообщений в почтовом ящике.

**imap\_listsubscribed.** Получение списка активных ящиков

array imap\_listsubscribed (int imap\_stream, string ref, string pattern)

У пользователя может быть сложная структура папок почтового аккаунта, но, как правило, для синхронизации и отображения в программе почтового клиента выбираются только несколько ящиков, а остальные не затрагиваются.

Функция возвращает в массиве имена папок, выбранных для синхронизации. Она аналогична функции imap\_listmailbox().

**imap\_getsubscribed.** Получение описания активных ящиков

array imap\_getsubscribed (int imap\_stream, string ref, string pattern)

Функция аналогична imap\_getmailboxes(), но она возвращает имена только тех ящиков, которые были выбраны для синхронизации.

**imap\_subscribe.** Сделать ящик активным

int imap\_subscribe (int imap\_stream, string mbox)

Указанный ящик в дальнейшем будет использоваться для синхронизации. В случае успешного выполнения возвращается значение TRUE. При возникновении ошибки возвращается значение FALSE.

**imap\_unsubscribe.** Сделать ящик неактивным

int imap\_unsubscribe (int imap\_stream, string mbox)

Функция делает ящик неактивным, и в дальнейшем он не будет использоваться для синхронизации.

**imap\_search.** Поиск в текущем ящике

array imap\_search (int imap\_stream, string criteria, int flags)

Функция возвращает список почтовых ящиков, в которых было обнаружено соответствие критерию, указанному в аргументе *criteria*. В строке критерия указываются искомые слова, разделенные пробелами. Если необходимо искать фразу, то ее следует заключать в двойные кавычки. Перед набором слов указывается имя поля, в котором следует производить поиск. Можно также указать флаги просматриваемых сообщений. Применяемые флаги перечислены в следующем списке:

* ALL - все сообщения, содержащие перечисленные слова.
* ANSWERED - только сообщения, на которые был выслан ответ.
* BCC "string" - искать строку "string" в поле Bcc.
* BEFORE "date" - сообщения с датой, меньшей, чем "date".
* BODY "string" - сообщения, содержащие в своем теле строку "string".
* CC "string" - искать строку "string" в поле Cc.
* DELETED - удаленные сообщения.
* FLAGGED - сообщения со статусом "Важное".
* FROM "string" - искать строку "string" в поле From.
* KEYWORD "string" - искать сообщения, содержащие слово "string".
* NEW - просматривать только новые сообщения.
* OLD - просматривать только старые сообщения.
* ON "date" - просматривать сообщения с датой "date".
* RECENT - просматривать свежие сообщения, имеющие флаг RECENT.
* SEEN - просматривать прочитанные сообщения, имеющие флаг SEEN.
* SINCE "date" - просматривать сообщения, полученные позже "date".
* SUBJECT "string" - искать слово "string" в поле темы Subject.
* TEXT "string" - искать сообщения, содержащие текст "string".
* TO "string" - искать сообщения, содержащие строку "string" в поле адресата To.
* UNANSWERED - просматривать сообщения, на которые не был написан ответ.
* UNDELETED - просматривать не удаленные сообщения.
* UNFLAGGED - просматривать сообщения без флагов.
* UNKEYWORD "string" - искать сообщения, не содержащие слово "string".
* UNSEEN - поиск вести в еще не просмотренных сообщениях.

Чтобы найти все сообщения от отправителя Peter, на которые не был выслан ответ, указывается строка UNANSWERED FROM Peter. Поиск не чувствителен к регистру. Приведенный список критериев взят из исходного текста библиотеки и может быть не полным. Более подробная информация приведена в документе RFC2060, section 6.4.4.

Допустимым значением аргумента flags является SE\_UID. Оно указывает, что вместо относительных последовательных номеров сообщений следует возвращать уникальные идентификаторы UID.

**imap\_scanmailbox.** Поиск текста в сообщениях

array imap\_scanmailbox (int imap\_stream, string ref, string pattern, string content)

Функция подобна imap\_listmailbox(), но она возвращает имена только тех ящиков, которые содержат в себе строку *content*.

**imap\_fetch\_overview.** Получение описания сообщений

array imap\_fetch\_overview (int imap\_stream, string sequence [, int flags])

Функция возвращает в массиве объекты, содержащие информацию о сообщениях, номера которых перечислены в строке *sequence*. Эта строка должна содержать номер сообщения или номера нескольких сообщений через запятую, без пробелов. Можно также указать диапазон номеров, использовав двоеточие. Если в значении аргумента указана константа FT\_UID, то это указывает на то, что номера являются уникальными идентификаторами UID.

Свойства объектов массива перечислены в следующем списке:

* subject - тема сообщения.
* from - отправитель.
* date - дата отправки.
* message\_id - идентификатор сообщения.
* references - ссылки на message\_id.
* size - размер в байтах.
* uid - уникальный идентификатор сообщения UID в ящике.
* msgno - порядковый номер сообщения в ящике.
* recent - флаг "новое сообщение".
* flagged - флаг "важное сообщение".
* answered - на сообщение написан ответ.
* deleted - сообщение выбрано для удаления.
* seen - прочитанное сообщение
* draft - флаг "черновик".

Пример использования функции:

$mbox = imap\_open("{your.imap.host:143}","username","password")

|| die("can't connect: ".imap\_last\_error());

$overview = imap\_fetch\_overview($mbox,"2,4:6",0);

if(is\_array($overview)) { print\_r($overview); }

imap\_close($mbox);

В результате работы этого кода на экран будет выведена следующая информация:

Array

(

[0] => stdClass Object

(

[subject] => Поздравление

[from] => IG <ig@igor.>

[date] => Sun, 13 May 2001 01:41:00 +0400

[message\_id] => <000801c0db2c$3c46a2c0$0100007f@igor>

[size] => 1621

[uid] => 289703661

[msgno] => 1

[recent] => 0

[flagged] => 0

[answered] => 0

[deleted] => 0

[seen] => 1

[draft] => 0

)

**imap\_headers**. Получение списка сообщений текущего ящика

array imap\_headers (int imap\_stream)

Функция возвращает массив, содержащий заголовки имеющихся сообщений. Сообщения вложенных папок не просматриваются.

**imap\_headerinfo.** Просмотр заголовка сообщения

object imap\_headerinfo (int imap\_stream, int msg\_number [, int fromlength [, int subjectlength [, string defaulthost]]])

Функция возвращает объект, содержащий различную информацию о сообщении. В качестве примера можно привести следующий вызов функции:

$mbox = imap\_open ("\{$SERVER/pop3:110}INBOX", "igor7", "igx321");

print\_r(imap\_header($mbox,1));

В результате работы этого кода на экран будут выведены следующие строки:

stdClass Object

(

[date] => Sun, 13 May 2005 01:41:00 +0400

[Date] => Sun, 13 May 2005 01:41:00 +0400

[subject] => Послание

[Subject] => Послание

[message\_id] => <000801c0db2c$3c46a2c0$0100007f@igor>

[toaddress] => igor@localhost

[to] => Array

(

[0] => stdClass Object

(

[mailbox] => igor

[host] => localhost

)

)

[fromaddress] => IG <ig@igor.>

[from] => Array

(

[0] => stdClass Object

(

[personal] => IG

[mailbox] => ig

[host] => igor.

)

)

[reply\_toaddress] => IG <ig@igor.>

[reply\_to] => Array

(

[0] => stdClass Object

(

[personal] => IG

[mailbox] => ig

[host] => igor.

)

)

[senderaddress] => IG <ig@igor.>

[sender] => Array

(

[0] => stdClass Object

(

[personal] => IG

[mailbox] => ig

[host] => igor.

)

)

[Recent] => N

[Unseen] =>

[Flagged] =>

[Answered] =>

[Deleted] =>

[Draft] =>

[Msgno] => 1

[MailDate] => 13-May-2005 01:41:00 +0400

[Size] => 1581

[udate] => 989703660

)

Структура и формат записей определяются реализацией серверного интерфейса и могут сильно различаться для различных систем. В большинстве случаев для сообщений протоколов IMAP, POP3 и NNTP форматы информационных полей идентичны.

Для сообщений news-серверов дополнительно возвращается свойство newsgroup.

Пары полей date, Date и subject, Subject обычно совпадают. В message\_id хранится уникальный хеш сообщения, а в Msgno - относительный порядковый номер. В записях Size и udate возвращаются, соответственно, размер сообщения в байтах и время в формате UNIX Timestamp.

В большинстве случаев адреса fromaddress, reply\_toaddress и senderaddress совпадают. Если у сообщения имеется поле Cc, то также возвращается адрес ccaddress.

**imap\_sort.** Сортировка заголовков

array imap\_sort (int stream, int criteria, int reverse, int options)

Возвращает массив номеров сообщений, сортированных по критерию. Если в аргументе *reverse* указывается значение TRUE, то сортировка проводится в обратном порядке.

Возможные значения аргумента *criteria* приведены в следующем списке:

* SORTDATE - сортировка по дате сообщения.
* SORTARRIVAL - сортировка по дате получения.
* SORTFROM - сортировка по первому адресу в поле From.
* SORTSUBJECT - сортировка по строке темы в поле Subject.
* SORTTO - сортировка по первому адресу в поле To.
* SORTCC - сортировка по первому адресу в поле получателей копии Nn.
* SORTSIZE - сортировка по размеру сообщения.

Флаги options аргумента могут содержать следующие значения:

* SE\_UID - возвращать уникальный идентификатор сообщения UID вместо его порядкового номера.
* SE\_NOPREFETCH - не выполнять с сообщением предварительные манипуляции.

**imap\_fetchstructure.** Получение полной информации о сообщении

object imap\_fetchstructure (int imap\_stream, int msg\_number [, int flags])

Информация возвращается в объекте, который описывает формат и структуру сообщения в текущем ящике, указанного порядковым номером *msg\_number*. Если в аргументе *flags* указывается значение FT\_UID, то это свидетельствует о том, что в аргументе *msg\_number* указан не номер сообщения, а его идентификатор UID.

Поля объекта приведены в следующем списке:

* type - тип основного тела сообщения.
* encoding - кодировка тела сообщения.
* ifsubtype - принимает значение TRUE, если имеется вложенный тип.
* subtype - вложенный тип MIME.
* ifdescription - принимает значение TRUE, если есть строка описания сообщения.
* description - строка описания.
* ifid - принимает значение TRUE, если имеется строка идентификации.
* id - строка идентификации.
* lines - размер в строках.
* bytes - размер в байтах.
* ifdisposition - принимает значение TRUE, если есть строка размещения.
* disposition - строка размещения.
* ifdparameters - принимает значение TRUE, если есть массив дополнительных параметров.
* dparameters - массив дополнительных параметров.
* ifparameters - принимает значение TRUE, если существует массив обычных параметров.
* parameters - массив параметров MIME.
* parts - массив объектов, описывающих каждую часть сообщения.

Значения типа type основного тела перечислены в следующем списке:

* 0 - простой текст.
* 1 - составное сообщение.
* 2 - простое сообщение.
* 3 - приложение.
* 4 - аудиосодержимое.
* 5 - изображение.
* 6 - видео.
* 7 - другое.

Типы кодировки encoding перечислены в следующем списке:

* 0 - 7BIT.
* 1 - 8BIT.
* 2 - BINARY.
* 3 - BASE64.
* 4 - QUOTED-PRINTABLE.
* 5 - другие кодировки.

Для простого сообщения искомый объект может выглядеть следующим образом:

stdClass Object

(

[ifsubtype] => 1

[subtype] => PLAIN

[ifdescription] => 0

[ifid] => 0

[lines] => 2

[bytes] => 41

[ifdisposition] => 0

[ifdparameters] => 0

[ifparameters] => 0

[parameters] => stdClass Object

(

)

)

Для сложного сообщения объект будет выглядеть несколько сложнее:

stdClass Object

(

[type] => 1

[ifsubtype] => 1

[subtype] => ALTERNATIVE

[ifdescription] => 0

[ifid] => 0

[ifdisposition] => 0

[ifdparameters] => 0

[ifparameters] => 1

[parameters] => Array

(

[0] => stdClass Object

(

[attribute] => boundary

[value] => ----=\_NextPart\_000\_0005\_01C0DB4D.C33837A0

)

)

[parts] => Array

(

[0] => stdClass Object

(

[encoding] => 4

[ifsubtype] => 1

[subtype] => PLAIN

[ifdescription] => 0

[ifid] => 0

[lines] => 5

[bytes] => 59

[ifdisposition] => 0

[ifdparameters] => 0

[ifparameters] => 1

[parameters] => Array

(

[0] => stdClass Object

(

[attribute] => charset

[value] => koi8-r

)

)

)

[1] => stdClass Object

(

[encoding] => 4

[ifsubtype] => 1

[subtype] => HTML

[ifdescription] => 0

[ifid] => 0

[lines] => 14

[bytes] => 547

[ifdisposition] => 0

[ifdparameters] => 0

[ifparameters] => 1

[parameters] => Array

(

[0] => stdClass Object

(

[attribute] => charset

[value] => koi8-r

)

)

)

)

)

**imap\_fetchheader.** Получение заголовка сообщения

string imap\_fetchheader (int imap\_stream, int msgno, int flags)

Функция возвращает полный заголовок формата RFC822 сообщения с номером *msgno* из выбранного ящика.

Возможные значения аргумента *flags* указаны в следующем списке:

* FT\_UID - в аргументе *msgno* указан уникальный идентификатор UID.
* FT\_INTERNAL - возвращать текст как есть, а не в формате CRLF.
* FT\_PREFETCHTEXT - (RFC822). Одновременно получается текст сообщения, что сокращает общую длительность подключений IMAP. Полученный текст сохраняется в локальном буфере на время подключения.

**imap\_body.** Получение сообщения

string imap\_body (int imap\_stream, int msg\_number [, int flags])

Функция возвращает тело сообщения с номером *msg\_number* из указанного ящика. Возможные значения аргумента *flags* перечислены в следующем списке:

* FT\_UID - в аргументе *msgno* указан уникальный идентификатор UID.
* FT\_PEEK - не устанавливать флаг сообщения Seen, если он еще не установлен.
* FT\_INTERNAL - возвращать текст как есть, а не в формате CRLF.

Функция возвращает полную точную копию сообщения. С ее помощью можно прочитать простые сообщения. Для составных сообщений функция может вернуть строку This is a multi-part message in MIME format. Чтобы извлечь отдельные части сообщения и обработать их согласно указанному типу MIME, нужно воспользоваться функциями imap\_fetch\_structure() и imap\_fetch\_body().

**imap\_fetchbody.** Получение части сообщения

string imap\_fetchbody (int imap\_stream, int msg\_number, string part\_number [, flags flags])

Функция возвращает часть тела сообщения с номером *msg\_number* из выбранного ящика. Спецификация указания перечисления номеров секций в аргументе *part\_ number* определяется IMAP4.

Возможные значения аргумента *flags* перечислены в следующем списке:

* FT\_UID - в аргументе *msgno* указан уникальный идентификатор UID.
* FT\_PEEK - не устанавливать флаг сообщения Seen, если он еще не установлен.
* FT\_INTERNAL - возвращать текст как есть, а не в формате CRLF.

Для сложного сообщения, структура которого приведена в примере функции imap\_fetchstructure(), содержание частей может быть усложненным.

Основная часть сообщения, которая возвращается вызовом imap\_fetchbody ($m,1,0), выглядит следующим образом:

X-F: <ig@igor.> Sun May 13 01:41:00 2003

Received: from igor [127.0.0.1] by igor.

(SMTPD32-6.06 EVAL) id ADEC10316; Sun, 13 May 2001 01:41:00 +0400

Message-ID: <000801c0db2c$3c46a2c0$0100007f@igor>

From: "IG" <ig@igor.>

To: <igor@localhost>

Subject: Приветствие

Date: Sun, 13 May 2001 01:41:00 +0400

MIME-Version: 1.0

Content-Type: multipart/alternative;

boundary="----=\_NextPart\_000\_0005\_01C0DB4D.C33837A0"

X-Priority: 3

X-MSMail-Priority: Normal

X-Mailer: Microsoft Outlook Express 5.00.2919.6700

X-MimeOLE: Produced By Microsoft MimeOLE V5.00.2919.6700

X-RCPT-TO: <igor@localhost>

X-UIDL: 289703661

Status: U

Первая вложенная часть может быть получена при помощи вызова imap\_ fetchbody($m,1,1). Выглядит она следующим образом:

Привет.

У нас все хорошо.

Пока.

Вторая вложенная часть, возвращенная вызовом imap\_fetchbody($m,1,2), будет выглядеть сложнее:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<HTML><HEAD>

<META content=3D"text/html; charset=3Dkoi8-r" http-equiv=3DContent-Type>

<META content=3D"MSHTML 5.00.2920.0" name=3DGENERATOR>

<STYLE></STYLE>

</HEAD>

<BODY bgColor=3D#ffffff>

<DIV><FONT face=3D"Arial Cyr" size=3D2> Привет.</FONT></DIV>

<DIV><FONT face=3D"Arial CYR" size=3D2> У нас все хорошо.</FONT></DIV>

<DIV><FONT face=3D"Arial CYR" size=3D2> Пока. </FONT></DIV>

</BODY></HTML>

**imap\_uid**. Получение UID сообщения из порядкового номера

int imap\_uid (int imap\_stream, int msgno)

Функция возвращает уникальный идентификатор сообщения UID. Он отличается от обычного порядкового номера, потому что номера перевычисляются при удалении сообщений. Вызов функции предельно прост:

echo imap\_uid($mbox,1); // выводит, например, 289703661

**imap\_msgno**. Получение порядкового номера сообщения из UID

int imap\_msgno (int imap\_stream, int uid)

Действие функции обратно imap\_uid().

**imap\_setflag\_full.** Установка флагов сообщения

string imap\_setflag\_full (int stream, string sequence, string flag, string options)

Функция позволяет установить флаги для выбранного сообщения. В аргументе *sequence* указывается перечисление номеров сообщений, для которых необходимо установить флаги, указанные в аргументе *flag*. Флаги обозначаются, как \\Seen, \\Answered, \\Flagged, \\Deleted, \\Draft и \\Recent.

В аргументе *options* может быть указано значение ST\_UID, указывающее на то, что в аргументе *sequence* указан уникальный идентификатор UID.

Пример использования функции:

$mbox = imap\_open("{your.imap.host:143}","username","password")

|| die("can't connect: ".imap\_last\_error());

if (imap\_setflag\_full($mbox,"2,4,15:19,23","\\Seen \\Flagged"))

print "Флаги установлены\n";

imap\_close($mbox);

**imap\_clearflag\_full.** Сброс флагов сообщения

string imap\_clearflag\_full (int stream, string sequence, string flag, string options)

Действие функции противоположно imap\_setflag\_full(). Она сбрасывает флаги у выбранных сообщений.

**imap\_delete.** Маркировка сообщения как удаленного

int imap\_delete (int imap\_stream, int msg\_number [, int flags])

В аргументе *msg\_number* указывается номер сообщения, а в параметре *flags* при значении FT\_UID указывается, является ли номер идентификатором. Сообщения, помеченные к удалению, остаются в ящике, пока не будет вызвана функция imap\_ expunge() или подключение с установленным флагом CL\_EXPUNGE не будет закрыто функцией imap\_close().

Пример использования функции:

$mbox = imap\_open ("{your.imap.host}INBOX", "username", "password")

|| die ("can't connect: " . imap\_last\_error());

$check = imap\_mailboxmsginfo ($mbox);

print "Сообщений до удаления: ". $check->Nmsgs . "<br>\n" ;

imap\_delete ($mbox, 1);

$check = imap\_mailboxmsginfo ($mbox);

print "Сообщений после удаления: ".$check->Nmsgs."<br>\n" ;

imap\_expunge ($mbox);

$check = imap\_mailboxmsginfo ($mbox);

print " Сообщений после expunge(): ".$check->Nmsgs."<br>\n" ;

imap\_close ($mbox);

**imap\_undelete.** Восстановление удаленного сообщения

int imap\_undelete (int imap\_stream, int msg\_number)

Функция снимает флаг удаления с сообщения с номером *msg\_number*. Этот флаг ранее мог быть установлен функциями imap\_delete(), imap\_mail\_move(), или imap\_ setflag\_full().

**imap\_expunge**. Удаление сообщений, помеченных к удалению

int imap\_expunge (int imap\_stream)

Функция полностью удаляет из ящика сообщения, имеющие флаг \\Deleted, который может быть установлен функциями imap\_delete(), imap\_mail\_move() или imap\_ setflag\_full().

**imap\_mail\_copy.** Копирование сообщения в ящик

int imap\_mail\_copy (int imap\_stream, string msglist, string mbox [, int flags])

Перечисление номеров копируемых сообщений указывается в строковом аргументе *msglist*. Их следует перечислять без пробелов через запятую, диапазоны указываются знаком двоеточия. Функция копирует выбранное сообщение в указанный ящик.

Ящик назначения, указываемый в аргументе *mbox*, указывается без имени сервера.

В аргументе *flags* можно указывать битовую маску из следующих констант:

* CP\_UID - номера являются уникальными идентификаторами UID.
* CP\_MOVE - удалять сообщения после копирования.

Вызов функции выглядит достаточно просто:

imap\_mail\_copy($m, "1,3:5","INBOX/Текущие")

**imap\_mail\_move**. Перемещение сообщения в ящик

int imap\_mail\_move (int imap\_stream, string msglist, string mbox [, int flags])

Функция аналогична следующему вызову:

imap\_mail\_copy(imap\_stream, msglist, mbox, flags | CP\_MOVE )

**imap\_alertsimap\_alerts.** Получение извещений IMAP

array imap\_alerts (void)

Функция возвращает массив, содержащий все извещения, посланные сервером от начала подключения или после прошлого вызова данной функции.

**imap\_mime\_header\_decode.** Декодирование элементов MIME- заголовков

array imap\_header\_decode (string text)

Функция декодирует расширенные заголовки, посланные не в виде текста ASCII. Декодированные элементы возвращаются в массиве объектов, каждый из которых содержит значения "charset" и "text", в которых указываются кодировка и содержимое текста. Если заголовки не были закодированы, то значение элемента "charset" будет равно "default".

Пример использования функции:

$text="=?ISO-8859-1?Q?Keld\_J=F8rn\_Simonsen?= <keld@dkuug.dk>";

print\_r(imap\_mime\_header\_decode($text));

В результате работы этого кода на экран будут выведены следующие строки:

Array(

[0] => stdClass Object (

[charset] => ISO-8859-1

[text] => Keld Jorn Simonsen

)

[1] => stdClass Object (

[charset] => default

[text] => <keld@dkuug.dk>

)

)

**imap\_mail\_compose.** Создание сообщения MIME по шаблону

string imap\_mail\_compose (array envelope, array body)

Функция создает новое сообщение:

<?php

$envelope["from"]="musone@afterfive.com";

$envelope["to"]="musone@darkstar";

$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;

$part1["subtype"]="mixed";

$filename="/tmp/imap.c.gz";

$fp=fopen($filename,"r");

$contents=fread($fp,filesize($filename));

fclose($fp);

$part2["type"]=TYPEAPPLICATION;

$part2["encoding"]=ENCBINARY;

$part2["subtype"]="octet-stream";

$part2["description"]=basename($filename);

$part2["contents.data"]=$contents;

$part3["type"]=TYPETEXT;

$part3["subtype"]="plain";

$part3["description"]="description3";

$part3["contents.data"]="contents.data3\n\n\n\t";

$body[1]=$part1;

$body[2]=$part2;

$body[3]=$part3;

echo nl2br(imap\_mail\_compose($envelope,$body));

?>

**imap\_append.** Создание сообщения в ящике

int imap\_append (int imap\_stream, string mbox, string message [, string flags])

Функция позволяет создавать сообщения в выбранном ящике. В аргументе *message* содержится текст сообщения, включая текст заголовка. Для многих серверов необходимо обозначать конец строк парой символов \r\n.

В аргументе *flags* можно указать флаги, присваиваемые сообщению.

Пример использования функции:

$stream = imap\_open("{your.imap.host}INBOX.Drafts","username", "password");

$check = imap\_check($stream);

print "Сообщений было: ". $check->Nmsgs."\n";

imap\_append($stream,"{your.imap.host}INBOX.Drafts"

,"From: me@my.host\r\n"

."To: you@your.host\r\n"

."Subject: test\r\n"

."\r\n"

."this is a test message, please ignore\r\n",

"\\Flagged \\Draft");

$check = imap\_check($stream);

print "Сообщений стало: ". $check->Nmsgs."\n";

imap\_close($stream);

### **Вспомогательные функции**

**imap\_last\_error.** Получение описания последней ошибки

string imap\_last\_error (void)

Функция возвращает строку, содержащую полное описание ошибки IMAP, произошедшей в ходе исполнения скрипта. Стек сообщений не изменяется, и повторный вызов функции возвратит то же самое значение, если не произошло других ошибок.

**imap\_errors.** Получение всех возникших ошибок

array imap\_errors (void)

Функция возвращает массив, содержащий строки описания ошибок IMAP, произошедших от начала подключения или с момента последнего вызова данной функции. Вызов функции полностью очищает стек ошибок.

**imap\_rfc822\_write\_addressimap\_rfc822\_write\_address.** Форматирование строки адреса

string **imap\_rfc822\_write\_address** (string *mailbox*, string *host*, string *personal*)

Функция возвращает строку электронного адреса, сформированную из заданных значений, согласно правилам, указанным в RFC822. Вызов функции достаточно прост:

print imap\_rfc822\_write\_address("ig","php.net","Igor Grigin");

// выведет: Igor Grigin <ig@php.net>

**imap\_rfc822\_parse\_adrlist.** Интерпретация адреса

array imap\_rfc822\_parse\_adrlist (string address, string default\_host)

Исходная строка с перечислением почтовых адресов, записанных в формате RFC822, указывается в аргументе *address*. Аргумент содержит имя хоста, добавляемое к адресу. Функция возвращает массив объектов, свойства которых указаны в следующем списке:

* mailbox - имя ящика.
* host - имя хоста.
* personal - личное имя.
* adl - доменный разделитель.

Пример использования функции:

$address\_string = "Igor Grigin <ig@my.domain.net>,

postmaster@somedomain.net, root";

$address\_array = imap\_rfc822\_parse\_adrlist($address\_string,"our.net");

if(! is\_array($address\_array)) die("somethings wrong\n");

print\_r($address\_array);

В результате работы этого кода на экран будут выведены следующие строки:

Array(

[0] => stdClass Object (

[mailbox] => ig

[host] => my.domain.net

[personal] => Igor Grigin

)

[1] => stdClass Object (

[mailbox] => postmaster

[host] => somedomain.net

)

[2] => stdClass Object (

[mailbox] => root

[host] => our.net

)

)

**imap\_rfc822\_parse\_headers.** Интерпретация заголовков из строки

object imap\_rfc822\_parse\_headers (string headers [, string defaulthost])

Функция сходна с imap\_header(), но информацию она получает не от сервера IMAP, а из строки *headers*.

**imap\_utf7\_decode.** Декодирование строки UTF-7

string imap\_utf7\_decode (string text)

Функция возвращает оригинальную строку после декодирования. Эта кодировка часто используется для адресов почтовых ящиков, содержащих национальные символы. Функция использует модифицированный алгоритм, описанный в RFC 2060, section 5.1.3. Устаревший алгоритм был описан в RFC1642.

# Ускорение PHP-программ

Как без особых усилий заставить **PHP**-код работать на порядок **быстрее** ? Перед тем как задаваться вопросами кеширования и масштабирования стоит попробовать оптимизировать код. Есть ряд несложных правил:

* **Выносите функции и вычисления из циклов.** Такое:
* $ar=array(1,2,2,3,4,4);
* for($i=0;$i<count($ar);$i++){
* $a=12342+sqrt($b)/12123+$i;

}

Замените на

$ar=array(1,2,2,3,4,4);

$e=12342+sqrt($b)/12123;

$c=count($ar);

for($i=0;$i<$c;$i++){

$a=$e+$i;

}

Альтернативный вариант:

for($i=count($ar)-1;$i>=0;$i--){

}

* **Выносите переменные из строк.** Примерно так:***<?php $x="test=".$test; ?>***, это на 25-40% быстрее.
* **Имена переменных должны быть короче 7 символов.** Это быстрее на 15%.
* **В массивах не стоит обращаться к элементам по имени и без кавычек.** Особенно в многомерных. Сэкономите 40%.
* А если **обращение к элементу массива вынести из строки** (как в пункте 1), то можно сэкономить ещё 25-30%.
* **Используйте Perl-ориентированные регулярные выражения.**Или хотя бы сравнивайте их с POSIX-аналогами по скорости выполнения. Скорость может отличаться до 200% на разных запросах.
* [**SizeOf()**](https://htmlweb.ru/php/function/sizeof.php)**быстрее**[**Count()**](https://htmlweb.ru/php/function/count.php), а в циклах sizeof лучше вообще заменить на переменную, for и while практически не отличимы, **для перебора простых индексных массивов нужно использовать for или while**, а **для ассоциативных массивов - foreach**. Плюс 30% при соблюдении.
* Для чтения файла [**file()**](https://htmlweb.ru/php/function/file.php)**быстрее, чем**[**fopen()**](https://htmlweb.ru/php/function/fopen.php)**+цикл**. Ускорение до 40%.

Прочитать про всё это подробнее, посмотреть графики скорости и загрузить все примеры можно вот тут: "[Оптимизация программ на PHP](http://php.spb.ru/)".

## **Еще про оптимизацию....**

При вставке кусков PHP-кода в HTML страницы всегда используйте полные открывающие и закрывающие скобки <?php и ?>! Это обезопасит Вас от вариаций настройки php.ini short\_open\_tag на разных серверах и возможно сэкономит много времени при переносе или загрузке проектов на разные сервера.

Старайтесь использовать функцию вывода [echo](https://htmlweb.ru/php/function/echo.php) вместо [printf](https://htmlweb.ru/php/function/printf.php) и [sprintf](https://htmlweb.ru/php/function/sprintf.php) там где возможно. Нет надобности использовать эти функции , так как они выполняются медленней потому, что созданы для интерпретации и вывода строки с ее обработкой, подстановкой значений, в отформатированном виде. О чем и говорит буква f в конце названия этих 2-х функций.

Плохо:

sprintf('мама');

printf('папа');

Хорошо:

echo 'мама';

echo 'папа';

По тем же причинам используйте одинарные кавычки там где это возможно и пользуйтесь оператором "." для склейки строк, вместо прямой подстановки переменный в строку, заключенную в кавычки.

Лучший вариант(самый быстрый)

echo 'Вес равен: '.$weight;

Худший вариант(медленный):

echo "Вес равен: $weight";

Если Вам нужно проверить не равно ли возвращенное значение функции нулю (а функция сама по себе возвращает только положительные или только отрицательные значения), то лучше использовать оператор сравнения. Он выполняется быстрей, нежели конкретное сравнение значений.

Плохо:

$i = 0;

if ($i != 0)

{

//Не равно

}

else

{

//Равно

}

Хорошо:

$i = 0;

if ($i > 0)

{

//Не равно

}

else

{

//Равно

}

Нужно также учитывать, что если строка принимает только пустое значения, либо пользовательские строковые данные, то вместо сравнения строки со строкой, для выявления ее пустоты, так же можно использовать сравнение с нулем, которые выполнится быстрее.

Для проверки строки на пустоту используйте функцию [trim($str)](https://htmlweb.ru/php/function/trim.php); Она не только проверит заполнена ли строка, но также обрежет несущественные символы - пробелы (табуляции, white-spaces) и вернет положительное значение, в случае если в строке ей действительно какие то значимые символы.

Плохо:

if ($str != '')

{

//обработка строки

}

Надо:

if (trim($str))

{

//обработка строки

}

Для получения данных из форм методом Get и Post лучше использовать следующий минимальный набор самописных функций:

GetParam ($array, $value, $default = '')

{

return (isset($array[$value])) ? $array[$value] : $default;

}

GetParamSafe ($array, $value, $default = '')

{

return (isset($array[$value])) ? addslashes($array[$value]) : $default;

}

Функция GetParam($\_POST, 'myvar', 'empty') к примеру коректно получит данные из $\_POST['myvar'], и в случае если $\_POST переменная не существует вернет значение по умолчанию, без всяких Waring и Notice. Фунция GetParamSafe($\_POST, 'myvar', 'empty') делает ту же операцию, только возвращает экранированную переменную. Для защиты от SQL инъекций к примеру. А данная конструкция позволяет получить целочисленное число из $\_POST.

intval(GetParam($\_POST, 'myvar', 'empty')):

В случае если в массиве $\_POST лежало совсем не число функия вернет 0;

Для простого сравнения строк не используйте [preg\_match()](https://htmlweb.ru/php/function/preg_match.php) или [preg\_match\_all()](https://htmlweb.ru/php/function/preg_match_all.php). Используйте [strstr()](https://htmlweb.ru/php/function/strstr.php) и [strpos()](https://htmlweb.ru/php/function/strpos.php).

При получении строк из базы данных (MySQL к примеру) старайтесь использовать функцию [mysql\_fetch\_object](https://htmlweb.ru/php/function/mysql_fetch_object.php). К примеру при изменении кода запроса с

$query = "SELECT field7, field3 FROM mytable WHERE id = 5"

на

$query = "SELECT \* FROM mytable WHERE id = 5"

код вывода строки полученной из этих запросов

$row = mysql\_fetch\_array(mysql\_query($query));

echo $row[0].'-->'.$row[1]; //перестанет работать, в то время, как

$row = mysql\_fetch\_object(mysql\_query($query));

echo $row->field7.'-->'.$row->field3; // останется работоспособным.

При использовании сессий для авторизации на сайте, храните в сессии хотя бы IP-адрес, с которого был совершен вход. Так же проверяйте IP входа с текущим IP адресом каждый раз при выполнении закрытого скрипта. Например если злоумышленнику удастся украсть название сессии, то войти он в закрытую часть уже не сможет. Потому что в общем случае у него будет другой IP-адрес.

При формировании больших запросов вставки данных в БД через insert все строчки старайтесь поместить в один-три insert'а. Выполнение каждой строчки отдельно не только загрузит сервер БД, но и задержит работу Вашего скрипта.

В случае если необходимо в разных местах (разных классах) одной системы использовать одни и те же сложно вычисляемые данные (например которые достаются из БД через запрос с последющей обработкой строк), старайтесь их вычислять единожды, хранить глобально для всей системы и передавать в класс(скрипт) один раз, непосредственно при создании класса (подключении скрипта)

При больших нагрузках на Web-сервер задумайтесь над использованием стандартных решений для включения кэша(кэш-технологии). Например бесплатный PHP класс JCache\_Lite\_Function.

При проектировании/разработке больших систем отдавайте предпочтение Объектно-Ориентированному программированию с использование шаблонов проектирования. Наиболее частые шаблоны: MVC, PageController, BodyHandler, Fabric...