# Технология PHP

**PHP** (*Hypertext Preprocessor* – препроцессор гипертекста»; произносится *пи-эйч-пи*) – скриптовый языкпрограммирования, применяемый для разработки веб-приложений и динамических веб-страниц. Во многом PHP ориентирован на работу с базами данных.

PHP – один из популярных [сценарных языков](http://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) благодаря своей простоте, скорости выполнения, богатой функциональности, [кроссплатформенности](http://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C)и и открытому исходному коду (на основе лицензии PHP).

Синтаксис PHP подобен синтаксису языка Си. Некоторые элементы заимствованы из Perl.

PHP – очень гибкий язык. Одну и ту же по сути конструкцию можно записать разными способами – более короткими или длинными.

PHP содержит все основные конструкции и возможности современных языков программирования (ЯП) и множество специфических функций, которые существенно облегчают работу. Поддерживает объекто-ориентированное программирование, начиная с версии 5.

В данном пособии изложены лишь основы языка PHP. Работа с базами данных не рассматривается. Более подробно его стоит изучать на конкретных задачах с помощью огромного числа ресурсов в сети Интернет или учебников.

## Подготовка к работе. Сервер PHP

Код PHP должен обрабатываться сервером, на котором PHP установлен. Сам код можно писать в любом текстовом редакторе, но чтобы этот код заработал – нужен сервер. Поэтому начинать работу с PHP необходимо с сервера.

Основных вариантов здесь два – сразу разместить сайт в сети на хостинге с поддержкой PHP или установить локальный сервер на своем компьютере.

### Размещение сайта на хостинге

Хостинг – это услуга по размещению сайта на стороннем сервере. Сегодня существует большое число вариантов бесплатного хостинга, в том числе с поддержкой PHP, например holm.ru. Условием использования бесплатного хостинга обычно является размещение рекламы на сайте. При этом предоставляется доменное имя третьего уровня, например, mysite.**h17.ru**, где h17.ru – часть адреса, предоставленная хостером, а mysite – выбранное вами имя сайта.

Для размещения сайта необходимо зарегистрироваться и выбрать доменное имя. Вы получите доступ к папкам для размещения сайта. Доступ обычно можно осуществлять как через браузер, так и через ftp-соединение. По ftp файлы передаются так же, как и через обычный файловый менеджер (Проводник, Total Commander), но нужна специальная программа, например, FileZilla.

Сразу после размещения на хостинге сайт будет доступен по указанному адресу.

Преимущества данного способа: необязательно устанавливать дополнительные программы, сайт будет доступен откуда угодно (не получится «забыть дома»).

Недостатки: необходимо наличие Интернета, каждый раз тратится время и трафик на загрузку страниц и других файлов.

### Установка локального сервера

Сервер можно создать и свой, разместив и сайт, и сервер на своем компьютере.

Существует очень удобный набор инструментов для веб-программистов – Denwer («джентльменский набор web-разработчика»). Он включает поддержку PHP, MySQL, Perl и других возможностей и при этом бесплатен. Скачать Denwer последней версии можно с сайта разработчика Д. Котерова: www.denwer.ru.

Описание установки приводится на сайте. Кратко говоря, нужно просто установить Denwer, поместить свой сайт в директорию /home/*mysite*/www./ там, куда вы установили Denwer, и запустить Denwer с помощью Start.exe. После этого он будет доступен в браузере, также как и тестовые страницы, например http://*mysite*/ или http://www.*mysite*/.

Преимущества: возможность отредактировать сайт и мгновенно увидеть внесенные изменения, ничего никуда не нужно загружать.

Недостатки: сайт будет работать только там, где установлен сервер. А для работы сервера могут понадобиться административные права.

Можно использовать сочетание этих подходов: выполнять учебные задания и проверять работу сайта через Denwer, а рабочие версии файлов загружать на хостинг. Это позволит продемонстрировать сайт, а также снизит вероятность потери данных. Кроме того, по заданию, сайт все равно нужно будет разместить на хостинге.

Файл главной страницы сайта должна называться index с любым подходящим расширением (html, htm, php).

## Создание страниц с PHP

Страницы с PHP имеют расширение .php, а в остальном подобны обычным HTML-страницам. PHP встраивается в код аналогично JavaScript. Без сервера остальная часть страницы, кроме кода, будет отображаться как обычно.

Код PHP можно обрамлять разными способами:

* стандартный <?php ?>
* короткий <? ?>
* ASP <% %>
* программный <SCRIPT LANGUAGE="php"> </SCRIPT>

В дальнейшем будем использовать стандартное обрамление.

Скрипт можно разместить в любой части страницы. В отличие от JavaScript, в стандартном оформлении его не нужно заключать в комментарий, в случае сбоя он все равно не отобразится на странице.

### Пример. Hello, world!

Напишем простейший скрипт, который выведет на страницу приветствие Hello,world! и сохраним его как hello\_world.php

<HTML>

<HEAD><TITLE>PHP-страница</TITLE></HEAD>

<BODY>

 <p>Первый PHP-скрипт</p>

 **<?php**

 **print "Hello, world!";**

 **?>**

</BODY>

</HTML>

Как видите, это обычная HTML-страница, кроме выделенных строк. Функция Print выводит текст на страницу. Текст пишется в двойных кавычках.

Если вы просто откроете этот файл через браузер, то увидите только «Первый PHP-скрипт».



А при загрузке с PHP-сервера мы получим:



Посмотрите исходный код полученной страницы.

<HTML>

<HEAD><TITLE>PHP-страница</TITLE></HEAD>

<BODY>

 <p>Первый PHP-скрипт</p>

 Hello, world!</BODY>

</HTML>

Как видите, браузер ничего не знает об исходном PHP-коде. Он получил только результат – страницу с текстом «Hello, world!».

В функции Print можно передать и HTML-теги, браузер обработает их как обычно:

<HTML>

<HEAD><TITLE>PHP-страница</TITLE></HEAD>

<BODY>

 <p>Первый PHP-скрипт</p>

 <?php

 print "<p style='color:red'><b>Hello, world!</b></p>";

 ?>

</BODY>

</HTML>



Обратите внимание на кавычки: поскольку вся строка в print обрамляется двойными кавычками, в тегах HTML использовались одинарные.

Можно было записать то же самое иначе, с помощью символа \:

 print "<p style=**\"**color:red**\"**><b>Hello, world!</b></p>";

Это называется «гашением» специальных символов. Аналогично гасится служебный символ \$ и точка \.. Сочетание \n осуществляет перевод на новую строку (как нажатие Enter).

## Основы синтаксиса PHP

Команды в PHP обязательно отделяются точкой с запятой (;). Команды можно писать как на одной строке, так и на разных, но с точки зрения читаемости лучше каждую команду писать на отдельной строчке и соблюдать отступы.

### Комментарии

Однострочные комментарии указываются после двух символов слеша // или одного символа решетки #. Многострочные, как и в Си, указываются внутри /\* \*/

//это комментарий

# и это комментарий

/\*==========================

 Тут тоже

 комментарий

 \*==========================\*/

### Переменные. Типы данных

Переменные всегда начинаются со знака доллара $, затем идет любая последовательность букв, цифр, знаков подчеркивания, но первый символ не может быть цифрой. Имена переменных чувствительны к регистру, а зарезервированные слова – нет.

При объявлении переменных в РНР не требуется явно указывать тип переменной, при этом одна и та же переменная может иметь на протяжении программы разные типы.

Переменная инициализируется своим значением и остается в памяти до конца работы программы.

 <? php

 $x = 10; //обычная числовая переменная

 $X = $x + 100; //=110 $x и $X – это разные переменные!

 $x\_ = $x \* 2; //=20 $x и $x\_ – это тоже разные переменные

 $x = "текст"; // теперь $x – текстовая переменная

 $666q\_ = 20; //ошибка! нельзя начинать имя переменной с цифры

 ?>

PHP поддерживает следующие типы данных:

* integer (целые числа)
* string (строки)
* boolean (двоичные данные (истина или ложь))
* double (вещественные числа)
* array (массивы)
* object (объекты)
* NULL (пустой тип)
* resource (ресурсы)

Тип переменной определяется автоматически по ее содержимому.

### Вывод данных на страницу

Для вывода данных служат функции print() и echo(). Эти функции можно писать без скобок. Разницы в выводе данных для этих функций нет, но print всегда возвращает 1, а echo ничего не возвращает.

В функции Print можно указать конкретный текст в двойных кавычках или переменные. Нетекстовые переменные автоматически преобразуются к тексту.

 <? php

 $a = 10;

 $b = 3;

 print "a \* b ="; //вывод текста "$a \* $b =";

 print $a\*$b; //вывод значения 30

 ?>

Результат:

Можно было написать, например, так, результат не изменится:

 <? php

 $a = 10; $b = 3;

 print ("a \* b ="); print ($a\*$b);

 ?>

### Арифметические действия

Основные арифметические действия записываются как в Си:

+ сложение

- вычитание

\* умножение

/ деление

% остаток от деления.

Конкатенация строк выполняется с помощью точки:

 <? php

 $a = 50;

 $b = " - полтинник";

 $c = $a.$b; //="50 – полтинник"

 ?>

В предыдущем примере можно было записать:

 <? php

 $a = 10; $b = 3;

 print "a \* b =".$a\*$b;

 ?>

Можно использовать сокращенные записи для присваивания:

|  |  |  |
| --- | --- | --- |
| Оператор присваивания | Пример | Какую запись заменяет |
| += | $a+=1 | $a=$a+1 |
| -= | $a-=1 | $a=$a-1 |
| \*= | $a\*=1 | $a=$a\*1 |
| /= | $a/=1 | $a=$a/1 |
| %= | $a%=1 | $a=$a%1 |
| .= | $a.="слово" | $a=$a."слово" |

Операторы сравнения:

= = равно

!= не равно

= = = равно и совпадает по типу

> больше

>= больше или рано

< меньше

<= меньше или рано

## Проверка условий

### Оператор if

Стандартный оператор сравнения, записывается так же, как в Си:

 if (выражение)

 { операторы, если выражение истинно }

 else

 { операторы, если выражение ложно};

Ветвь else может отсутствовать:

 if (выражение)

 { операторы, если выражение истинно };

Примеры:

<?php

 if ($a == $b)

 { print "$a равно $b";}

 else

 {print "$a не равно $b";}

 if ($a <= 0)

 {

 print "$a должно быть положительным!";

 };

?>

Несколько последовательных сравнений, можно использовать elseif:

 if (выражение1)

 { операторы1 }

 elseif (выражение2)

 { операторы2}

 elseif (выражение3)

 { операторы3}

 else

 { операторы4};

Пример:

<?php

 if ($age < 10)

 { print " ребенок";}

 elseif ($age < 17)

 {

 print "подросток";

 }

 elseif ($age < 60)

 {

 print "взрослый";

 }

 else

 {

 print "старик";

 };

?>

### Задание

Самостоятельно измените предыдущий пример так, чтобы вывести, какое из чисел a и b больше, или они равны.

### Оператор switch

Выбор одного из нескольких вариантов можно сделать с помощью оператора switch.

switch (выражение)

 {

 case значение\_1:

 действия, если выражение равно значению\_1

 break;

 case значение\_2:

 действия, если выражение равно значению\_2

 break;

 default:

 выполняется, если выражение не приняло не одно из значений

 }

Например:

switch ($day)

 {

//дни недели

 case 1: print "понедельник"; break;

 case 2: print "вторник"; break;

 case 3: print "среда"; break;

 case 4: print "четверг"; break;

 case 5: print "пятница"; break;

 case 6: print "суббота"; break;

 case 7: print "воскресенье"; break;

//других дней недели не бывает

 default: print "Такого дня недели нет";

 }

Пример проверки значения, выбранного на форме:

<p>Укажите четвертую от Солнца планету</p>

<form action="script2\_1.php" method="POST">

 <select name=n1 >

 <option value="Земля"> Земля

 <option value="Марс"> Марс

 <option value="Венера"> Венера

 </select>

 <input type=submit value="Готово">

</form>

Скрипт из *script2\_1.php*:

<?php

 switch ($n1)

 {

 case "Земля":

 print "не верно";

 break;

 case "Марс":

 print "Верно !";

 break;

 case "Венера":

 print "не верно";

 break;

 }

?>

### Задание

Напишите скрипт, который по введенному дню и месяцу рождения выводит знак зодиака.

### Оператор ?

Оператор ? используется для более короткой записи if в присваивании, когда нужно выбрать один из двух вариантов значений.

= (условие) ? значение\_если\_истина : значение\_если\_ложь

Например, запись

if ($a == "м")

{

 $hello = "Уважаемый";

}

else

{

 $hello = "Уважаемая";

};

эквивалентна одной строке

$hello = ($a == "м") ? "Уважаемый" : "Уважаемая";

Так можно присвоить меньшее из двух значений:

$c = ($a < $b) ? $a : $b

## Циклы

В PHP используется три основных типа циклов, а также специальный цикл для элементов массивов (он будет рассмотрен в разделе «Массивы»).

### Цикл-счетчик FOR

Обычный счетчик. При первом запуске выполняется *инициализация*. После каждого шага выполняется *приращение*. Если *условие\_работы* истинно, цикл запускается еще раз.

 **for** **(***инициализация***;** *условие\_работы*; *приращение*)

 **{** *тело\_цикла* **}**

Примеры:

//вывод чисел от 1 до 9

for ($i=1;$i<10;++$i)

{

 print"<br>$i";

};

//вывод четных чисел от 20 до 2

for ($i=20;$i>1; $i+=2)

{

 print"<br>$i";

};

### Цикл с предусловием WHILE

Цикл выполняется, пока условие истинно. Цикл может не выполниться ни одного раза.

**while** **(***условие***)**

 **{** *тело\_цикла* **}**

Пример:

//вывод чисел Фибоначчи до 100: 0, 1, 2, 3, 5, 8, 13,...

$i1 = 0;

$i2 = 1;

while ($i2<100)

{

 print"<br>$i2";

 $i2 += $i1;

 $i1 = $i2 - $i1;

};

### Цикл с постусловием DO ... WHILE

Цикл продолжает выполняться, пока условие истинно. Цикл выполняется как минимум раз.

**do**

 **{** *тело\_цикла* **}**

**while** **(***условие***)**

Пример:

//бросаем монетку, пока не выпадет решка (0)

do

{

 //генерируем случайное число 0 или 1

 $coin = rand(0,1);

 $coin\_str = ($coin == 1) ? "орел" : "решка";

 print "<br>$coin\_str";

}

while ($coin == 1);

### Прерывание цикла BREAK

Любой цикл можно принудительно остановить командой **break**.

<?php

 //полный цикл от 1 до 10

 for ($i=1; $i<=10; $i++)

 {

 //на 6 цикл вылетит

 if ($i==6) **break**;

 print "$i <br>";

 };

?>

## Функции

Независимые куски кода удобнее оформлять в виде функций.

Обратите внимание, переменные, объявленные за пределами функций, не являются глобальными по умолчанию, и внутри функций будут не видны. Их нужно передавать как параметры или объявить внутри функции с ключевым словом GLOBAL.

Функция без параметров:

<?php

 function hi ()

 { print "Всем привет"; }

 hi ();

?>

Функция с параметрами:

<?php

 function sum ($a,$b)

 {

 print "$a + $b = ";

 print $a + $b;

 };

sum (3,4);

?>

Функция, возвращающая значение:

<?php

 function sum ($a,$b)

 {

 $c=$a + $b;

 return $c;

 }

 print sum (5,6);

?>

Можно записать короче:

function sum ($a,$b)

 {

 return ($a + $b);

 }

Функция со значением параметра по умолчанию:

<?php

 function textcolor ($a,$b="red")

 {

 print "<font color=$b> $a </font> <br>";

 };

 textcolor ("text 1","blue");

 textcolor ("text 2");

?>

Функция и глобальные переменные:

<?php

 function sum()

 {

 GLOBAL $a, $b;

 print $a + $b;

 };

$a = 3;

$b = 4;

sum ();

?>

### Задание

Напишите функции calc\_power4, print\_tr и fill\_table, которые будут выполнять следующие действия:

* calc\_power4 возвращает число, возведенное в 4 степень;
* fill\_table заполняет таблицу с 4 степенями чисел от 1 до 10;
* print\_tr выводит одну строку такой таблицы с HTML-тегами.

print\_tr должна вызывать calc\_power4, fill\_table должна вызывать две других функции. Дополнительно, закрасьте строки таблицы чередующимися цветами. Результат:



## Строки

Строки пишутся в двойных "" или одинарных '' прямых кавычках. Конкатенация строк выполняется символом точка (не через +).

Если в строке встречается имя переменной, она автоматически заменится на свое значение:

$z=65;

print "z=$z"; //выведется z=65

### Длина строки. STRLEN

Длина строки определяется функцией **strlen**( ):

$string = "hello";

$length = strlen($string); // $length = 5

### Сравнение строк. STRCMP и STRCASECMP

Функции **strcmp**() и **strcasecmp**() проверяют равенство строк, но первая с учетом регистра символов, а вторая без учета.

$a = "hello";

$b = "Hello";

if (strcmp($a,$b)) { // False

 print "Строки записаны одинаково"

}

elseif (strcmp($a,$b)) { // True

 print "Строки одинаковые, но записаны по-разному"

};

### Поиск по строке. STRPOS

Функция **strpos**( ) находит в строке первый экземпляр заданной подстроки и возвращает соответствующий номер символа:

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

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

$a = "Мама мылом мыла Милу, Мила мыло не любила";

print strpos($a, "мыл"); //=6

print strpos($a, "мыл",7); //=12

### Задание

Напишите функцию, с помощью которой можно вывести номера для всех вхождений подстроки в строку.



## Массивы

Массив представляет собой блок однотипных данных. Массивы в PHP не ограничены по размеру.

### Создание, заполнение и вывод массивов. ARRAY, RANGE, SPLIT

Следующие три записи идентичны:

//Перечисление элементов

$languages = array ("Spanish", "English", "Italian");

$languages[] = "Spanish";

$languages[] = "English";

$languages[] = "Italian";

$languages[0] = "Spanish"

$languages[l] = "English";

$languages[2] = "Italian";

Если указывать номера элементов не подряд, то пропущенные будут заполнены пустыми строками или 0, в зависимости от типа данных.

$languages[5] = "Russian";

//$languages = array("Spanish,"English,"Italian","","","Russian");

Числовой массив можно задать диапазоном значений:

$numbers = range(0,9);

//$numbers = array(0,1,2,3,4,5,6,7,8,9)

Наконец, можно получить массив разбиением строки с помощью функции **split**():

array split (string шаблон, string строка [, int порог])

Необязательный параметр «порог» определяет максимальное количество элементов, на которые делится строка слева направо. Функция split( ) работает с учетом регистра символов.

$str = "45;-67;23.7;12;45";

$arr = split(";";$str);

// $arr[0] = 45

// $arr[1] = -67

// $arr[2] = 23.7

// $arr[3] = 12

// $arr[4] = 45;

Следующий пример демонстрирует использование функции split( ) для разбиения десятичного IP-адреса на байты:

$ip = "123.35.0.89"; // целый IP-адрес

$iparr = split ("\.", $ip) // Точку необходимо экранировать

print "$iparr[0]<br>"; // Выводит "123"

print "$iparr[1]<br>"; // Выводит "35"

print "$iparr[2]<br>"; // Выводит "0"

print "$iparr[3]"; // Выводит "89"

### Ассоциативные массивы

В качестве индекса массива может использоваться не число, а строка. Такие массивы называются ассоциативными:

$languages["Spain"] = "Spanish"

$languages["Britain"] = "English"

$languages["Italia"] = "Italian"

С помощью array можно записать:

$languages = array("Spain" => "Spanish", "Britain" => "English", "Italia" => "Italian");

### Цикл-счетчик FOREACH

Существует специальный цикл для перебора всех элементов массива:

foreach (***массив*** as ***имя\_элемента\_массива***)

{

 ***тело\_цикла***

};

Пример использования:

 foreach ($languages as $language)

 {

 print $language;

 };

### Сортировка массивов

Функции для сортировки массивов встроены в PHP:

**sort**() – сортировка по возрастанию (от А до Я);

**rsort**() – сортировка по убыванию (от Я до А).

Пример – сортировка городов по алфавиту:

<? php

 $cities = ("Москва", "Самара", "Владимир", "Казань", "Иркутстк");

 sort($cities);

 print "<p>Города по алфавиту:</p>";

 print "<ol>";

 foreach ($cities as $city)

 {

 print "<li>".$city;

 };

 print "</ol>";

 rsort($cities);

 print "<p>Города в обратном порядке:</p>";

 print "<ol>";

 foreach ($cities as $city)

 {

 print "<li>".$city;

 };

 print "</ol>";

?>

## Взаимодействие с активными формами

Чаще всего в PHP требуется обработать данные, которые пользователь ввел через активную форму. Не останавливаясь подробно на самих элементах форм, рассмотрим подробно методы передачи данных (из формы на PHP-сервер).

### Метод GET

Передает все данные формы через адрес страницы URL. Из-за различных ограничений, связанных с длиной URL, этот метод применяется редко. Переменные и их значения перечисляются в конце адреса, после вопросительного знака:

http://site.domain/?имя1=значение1&...имя2=значение2

http://www.sagmu.ru/index.php?option=com\_content&view=article&id=14&Itemid=6

Зато можно передавать в PHP значения переменных и без форм, просто через адрес страницы. Например, добавив ссылку:

<a href="my\_script.php?x=10&y=20">

### Метод POST

Передает все данные формы через http-запрос пользователя к серверу. Этот метод используется чаще, чем get.

В любом случае в PHP все значения будут доступны через имена элементов и специальные массивы $\_GET и $\_POST, например:

$x = $\_GET['mytext1']; //значение из элемента с name=mytext1

Если для элементов заданы не имена, а идентификаторы id, то можно обращаться к ним напрямую.

$y = $\_GET['mytext2']; //значение из элемента с id=mytext2

### Пример

<form id="myform" action="*script\_1.php*" method="GET">

 <p>Введите число a=<input type=text name='a'>

 <p>Введите число b=<input type=text name='b'>

 <p><input type=submit value='Считать' name=send>

</form>

*script\_1.php* – это имя файла страницы, которая получит введенные значения. В *script\_1.php* добавляем код PHP:

 <?php

 $a = $\_GET['a'];

 $b = $\_GET['b'];

 print "a=$a<br>";

 print "b=$b";

 print "a+b=".($a+$b);

 ?>

Или, поскольку для формы задан id="myform"

 <?php

 print "a=$myform.a<br>";

 print "b=$myform.b";

 print "a+b=".($myform.a+$myform.b);

 ?>

 

Поскольку мы используем метод GET, то при нажатии на кнопку «Считать» браузер перейдет на страницу *script\_1.php?a=8&b=9*, где 8 и 9 – значения a и b.

Заменив GET на POST в обоих скриптах, вы не увидите значений переменных в строке браузера, они передадутся серверу вместе с запросом на загрузку страницы. А результат будет тот же.

Оба скрипта можно расположить и на одной странице.

<HTML>

<HEAD><TITLE>PHP-страница</TITLE></HEAD>

<BODY>

 <form id="myform" action="" method="POST">

 <p>Введите число a=<input type=text name=a>

 <p>Введите число b=<input type=text name=b>

 <p><input type=submit value='Считать' name=send>

 </form>

 <?php

 $a = $\_POST['a'];

 $b = $\_POST['b'];

 print "a=$a<br>";

 print "b=$b<br>";

 print "a+b=".($a+$b);

 ?>

</BODY>

</HTML>

При нажатии на кнопку 'Считать' страница перезагрузится, поля для ввода очистятся и выведется результат. Однако при первой загрузке страницы мы увидим:



Пустые значения a и b смотрятся некрасиво. Хотелось бы, чтобы обработка выполнялась только после нажатия на кнопку. Мы проверим это по нажатию на кнопку (name=send):

 <?php

 if($\_POST['send']) {

 $a = $\_POST['a'];

 $b = $\_POST['b'];

 print "a=$a<br>";

 print "b=$b<br>";

 print "a+b=".($a+$b);

 };

 ?>

### Задание

Самостоятельно измените предыдущий скрипт так, чтобы после нажатия на кнопку и перезагрузки страницы в полях для ввода сохранялись введенные значения. Если хотя бы одно из значений не было введено, сложение не выполнять и вывести сообщение об ошибке.

## Работа с файлами

PHP позволяет считывать и записывать данные в файл, создавать, удалять и перемещать на сервере файлы и каталоги, запускать внешние программы.

### Проверка свойств файла

Прежде чем пытаться работать с файлом, желательно убедиться в том, что он существует. Для решения этой задачи обычно используются две функции:

Функция **filе\_ехists(string filemane)** проверяет, существует ли заданный файл. Если файл существует, функция возвращает TRUE, в противном случае возвращается FALSE.

$filename = "test.dat"

if (! file\_exists ($filename)) {

 print "Файл $filename не найден!";

};

Функция **is\_file(string filemane)** проверяет существование заданного файла и возможность выполнения с ним операций чтения/записи.

$file = "myfile.txt";

if (is\_file($file)) {

 print "Файл $file существует и доступен!";

}

else {

 print "Файл $file не существует или недоступен!";

};

Функция **filesize(string filemane)** возвращает размер (в байтах) файла с заданным именем или FALSE в случае ошибки.

$fs = filesize("test.txt");

print "Размер файла test.txt: $fs байт(-а).";

### Задание

Напишите скрипт, в котором пользователь вводит имя файла в текстовое поле. По нажатию на кнопку проверяется существование файла и, если он существует, выводится его размер. Если размер превышает 10Кб, вывести в Кб, иначе – в байтах, выбрав правильное окончание (байт, байта).

### Открытие и закрытие файлов

Прежде чем выполнять операции с файлом, необходимо открыть его и связать с файловым манипулятором, а после завершения работы с файлом его следует закрыть. Эти темы рассматриваются в следующем разделе.

Функция **fopen(string filename, string mode)** открывает файл (если он существует) и возвращает целое число – файловый манипулятор (file handle).

В качестве *имени файла* (filename) можно указать:

* локальный файл на сервере;
* если параметр задан в виде php://stdin, php://stdout или php://stderr, открывается соответствующий стандартный поток ввода/вывода
* если параметр начинается с префикса http:// или ftp://, функция загружает файл по протоколу HTTP или FTP, соответственно.

*Режимы* (mode) открытия файла:

|  |  |
| --- | --- |
| *Режим*  | *Описание*  |
| r | Только чтение. Указатель текущей позиции устанавливается в начало файла  |
| r+ | Чтение и запись. Указатель текущей позиции устанавливается в начало файла |
| w | Только запись. Указатель текущей позиции устанавливается в начало файла, а все содержимое файла уничтожается. Если файл не существует, функция пытается создать его  |
| w+ | Чтение и запись. Указатель текущей позиции устанавливается в начало файла, а все содержимое файла уничтожается. Если файл не существует, функция пытается создать его   |
| a | Только запись. Указатель текущей позиции устанавливается в конец файла. Если файл не существует, функция пытается создать его |
| a+ | Чтение и запись. Указатель текущей позиции устанавливается в конец файла. Если файл не существует, функция пытается создать его |

После завершения работы файл всегда следует закрывать функцией fclose( ). Функция **fclose(int file\_handle)** закрывает файл с указанным манипулятором. При успешном закрытии возвращается TRUE, при неудаче – FALSE.

Для вывода сообщения об ошибке можно использовать функцию **die(string msg)**. Она прерывает дальнейшее выполнение программы.

Пример:

<?

 $filename = "test.php";

 if is\_file($filename)

 {

 //открываем файл только для чтения

 $f = fopen($filename, "r");

 //...

 //после работы закрываем файл

 fclose($f);

 }

 else

 {

 die("Файл $filename недоступен или не существует");

 };

?>

### Запись в файл

Прежде, чем записывать данные в файл, нужно проверить его доступность для записи с помощью функции **is\_writeable (string file\_name)**. Она возвращает TRUE, если файл с именем file\_name существует и доступен для записи, и FALSE в противном случае.

После этого файл нужно открыть в подходящем режиме (r+, w, w+, a, a+).

Функция **fwrite(int file\_handle, string str[, int length])** записывает содержимое строковой переменной str в файл, заданный файловым манипуляторомfile\_handle. Если при вызове функции передается необязательный параметр длинаlength, запись останавливается либо после записи указанного количества символов, либо при достижении конца строки.

Пример:

<?

 //Информация о посетителе сайта

 $data = "08:13:00|12:37:12|208.247.106.187|Win7";

 $filename = "somefile.txt";

 //Если файл существует и в него возможна запись

 if (is\_writeable($filename)) {

 // Открыть файл для записи в конец файла

 $fh = fopen($filename, "a+");

 // Записать содержимое $data в файл

 $success = fwrite($fh, $data);

 if (!$success)

 {

 die("При записи в файл Sfilename произошла ошибка.");

 };

 // Закрыть файл

 fclose($fh);

 }

 else

 {

 die("Не удалось открыть файл Sfilename.");

 };

?>

Функция **fputs()** является псевдонимом **fwrite()** и используется точно также.

### Чтение из файла

Аналогично записи, сначала файл нужно проверить на доступность для чтения с помощью функции **is\_readable(string file\_name)**. Файл следует открывать в режиме r, r+, w+ или a+.

Функций для чтения существует несколько.

Функция **fread(int file\_handle, int length)** читает из файла, заданного файловым манипулятором, заданное количество байт. Синтаксис функции fwrite( ):

Чтение прекращается после прочтения заданного количества байт или при достижении конца файла.

<?

 $filename = "test.php";

 if is\_readable($filename)

 {

 //открываем файл только для чтения

 $fh = fopen($filename, "r");

 //считываем файл целиком и выводим на страницу

 $file = fread($fh, filesize($fh));

 print "Файл $filename:";

 print $file;

 //после работы закрываем файл

 fclose($fh);

 }

 else

 {

 print "Файл $filename недоступен или не существует";

 };

?>

Функция **fgetc(int file\_handle)** возвращает строку, содержащую ровно один символ из файла в текущей позиции указателя, или FALSE при достижении конца файла.

Еще одна важная функция **feof(int file\_handle)** проверяет, не был ли достигнут конец файла (end of file).

$fh = fopen("test.txt", "r");

//посимвольное считывание файла

while (!feof($fh))

{

 $char = fgetc($fh);

 print $char;

};

fclose($fh);

### Задание

Напишите скрипт, который будет считывать из файла данные до первого пробела. Проверить доступность файла для чтения.

Чтобы считать из файла одну строку (до символа конца строки) можно использовать функцию **fgets (int file\_handle, int length)**. Она считывает строку либо до символа новой строки, либо до конца файла, либо длины length. Если назначить length достаточно большим, то будет выполняться построчное считывание.

$fh = fopen("test.txt", "r");

//построчное считывание файла

while (!feof($fh))

{

 $str = fgets($fh,filesize($fh));

 print $char;

};

fclose($fh);

### Задание

Напишите скрипт, который будет подсчитывать число строк (абзацев) в файле.

Последняя функция чтения, которую мы рассмотрим – **file (string file\_name)** – считывает файл в индексируемый массив. Она не требует открытия и закрытия файла. Каждый элемент выходного массива равен одной строке файла.

<?

 //построчный вывод файла в виде списка HTML

 $file\_array = file("test.txt");

 print "<ol>\n";

 foreach ($file\_array as $file\_line)

 {

 print "<li>$file\_line\n"

 };

 print "</ol>\n";

?>

## Динамическое создание страниц

### Динамическое формирование ссылок

Поскольку с помощью print можно выводить любое содержимое страницы, то можно и создавать ссылки на страницы. Причем куда именно ведет ссылка, будет зависеть от какого-либо условия.

if (CheckPass())

{

 print "Добро пожаловать!

 Перейти <a href=\"authorised.php\">далее</a>";

}

else

{

 print "Здравствуйте, гость!

 Ознакомительная версия <a href=\"guest.php\">здесь</a>";

};

Создание меню из массива:

$menu[] = "index";

$menu[] = "blabla";

$menu[] = "links";

$menu[] = "about";

for ($i=0; $i < sizeof($menu); $i++)

{

 print "<p><a href=\"".$menu[$i].".php\">".$menu[$i]."</a></p>";

};

Или, для вывода вертикального меню в таблицу:

for ($i=0; $i < sizeof($menu); $i++)

{

 print "<tr><td>

 <a href=\"".$menu[$i].".php\">".$menu[$i]."</a>

 </td>

 </tr>";

};

Если имена файлов и названия пунктов меню не совпадают, можно использовать ассоциативный массив:

$menu["index.php"] = "Главная";

$menu["blabla.php"] = "Разное";

$menu["links.php"] = "Ссылки";

$menu["about.php"] = "Об авторе";

foreach($menu as $page\_name => $menu\_item)

{

 print "<p><a href=\"".$page\_name."\">".$menu\_item."</a></p>";

};

### Шаблоны страниц

Шаблоном называется часть web-страниц, которая повторяется на нескольких страницах. Как правило, такие общие фрагменты содержания/кода (то есть шаблоны) сохраняются в отдельных файлах. А при построении web-документа они просто «включаются» в соответствующие места страницы.

Шаблоны нужны, в первую очередь, большим сайтам, где один и тот же код приходится писать по многу раз.

В файл шаблона можно вынести обычный текст, фрагмент HTML, PHP или JavaScript-код.

Обычно шаблоны сохраняются с расширением .tpl (template – шаблон).

В РНР существуют четыре функции для включения файлов в сценарии РНР:

* include(*имя\_файла*);
* include\_once(*имя\_файла*);
* require(*имя\_файла*);
* require\_once(*имя\_файла*).

Функция include() включает содержимое файла в сценарий. include\_once() делает то же самое, но файл включается в страницу только один раз, сколько бы раз не был написан include\_once().

С помощью include() можно включать шаблоны динамически:

// если рейтинг >= 100

if ($rate >= 100)

{

 //вывести поздравление из winner.tpl

 include("winner.tpl");

}

// если рейтинг < 10

else if ($rate < 10)

{

 //вывести предупреждение из looser.tpl

 include("looser.tpl");

};

//в остальных случаях ничего не надо добавлять

В следующем примере поздравление победителю появится только один раз, в скольки бы рейтингах он ни победил:

if ($rate1 >= 100)

{

 include\_once("winner.tpl");

};

if ($rate2 >= 100)

{

 include\_once("winner.tpl");

};

if ($rate3 >= 100)

{

 include\_once("winner.tpl");

};

В отличие от include, require() обязательно включает шаблон в готовый код страницы, даже если он окажется на ветке if-false или в коде с ошибками и, по идее, выполняться не должен. В этом же отличие require\_once() от include\_once(). Используются они значительно реже, чем include.

### Управление шаблонами. Темы (скины)

Используя метод GET для передачи значений переменных и шаблоны, можно фактически все страницы сайта реализовать через один общий файл PHP.

Пример основной страницы сайта index.php:

<HTML>

<? php

 include(Head.tpl);

?>

<BODY>

<? php

 include(PageHeader.tpl);

 include(Menu.tpl);

 switch ($page)

 {

 case "blabla": include(about.tpl); break;

 case "about": include(about.tpl); break;

 case "links": include(links.tpl); break;

 default include(Main.tpl);

 };

 include(PageBottom.tpl);

?>

<BODY>

</HTML>

Соответственно, по адресу index.php или index.php?page=main загрузится содержимое шаблона Main.tpl, а по index.php?page =about – шаблона about.tpl и т.д. А заголовок, меню и нижний колонтитул всегда будут одинаковыми.

Аналогично можно применить к сайту различные темы (шкурки, скины, настройки оформелния).

Для этого необходимо подготовить несколько CSS-файлов с разными настройками оформления, например *mystyles1*.css, *mystyles2*.css, *mystyles3*.css. Выбранный номер стиля будем передавать методом GET через переменную skin, например index.php?skin=1. По умолчанию будет загружаться *mystyles1*.css.

Стили включаются тегом link в разделе HEAD

<HEAD>

...

 <link rel="stylesheet" type="text/css" href="*mystyle1.css*">

</HEAD>

Такую строку тоже можно сформировать через PHP:

<HEAD>

...

 <? php

 //имя файла

 $filename = ($skin > 0) ? " mystyle1.css" : "mystyle".$skin.".css";

 //формируем всю строку

 $str =

 "<link rel=\"stylesheet\" type=\"text/css\" href=\"".$filename."\">";

 print $str;

 ?>

</HEAD>

## Примеры

### Счетчик загрузки страницы

Сегодня существует большое число внешних счетчиков посещений сайта, таких как LiveInternet. Достаточно зарегистрировать свой сайт в системе и добавить готовый код на страницу – и в готовом окошечке будет отображаться число уникальных посетителей, просмотров страниц... В самой системе можно просмотреть динамику посещений и рейтинг относительно других сайтах.

Тем не менее, такой счетчик можно написать и самостоятельно. Самый простой вариант – счетчик числа загрузок страниц (хитов). Он не проверяет уникальность адреса пользователя, был ли переход на страницу или ее просто обновили и т.п.

Принцип работы счетчика очень прост – при каждом запуске PHP, т.е. при каждой загрузке страницы с сервера, из специального файла считывается одно единственное число – счетчик посещений; это число увеличивается на 1 и записывается обратно.

<html>

<head>

<title>Счетчик посещений</title>

</head>

<body>

<p>Счетчик посещений:</p>

<?

 //простой счетчик обращений

 //Назначение: сохранение количества обращений в файле

 //файл, куда будем сохранять посещения

 $access = "hits.txt";

 //если файл существует

 if (file\_exists($access))

 {

 //считать текущее содержимое в массив (из одного элемента)

 $visits = @file($access);

 //увеличть счетчик

 $current\_visitors = $visits[0] + 1;

 }

 //иначе - это первое посещение

 else

 {

 $current\_visitors = 1;

 };

 //вывести счетчик на страницу

 print "<div class='hits'>$current\_visitors</div>";

 //записать в файл новое значение

 // открыть файл hits.txt для чтения, очистив его

 $fh = fopen($access, "w");

 // записать новое значение счетчика

 fwrite($fh, $current\_visitors);

 //закрыть файл

 fclose($fh);

?>

</body>

</html>

Лучше всего сохранить счетчик в отдельный шаблон и встроить в нужное место страницы.

### Голосование

Голосование также очень простое, никак не защищенное от «накрутки». В его работе возможны сбои, когда голосует сразу несколько человек.

Результаты голосования хранятся в файле в отдельных строках.

И вопросы голосования, и результаты будут обрабатываться одной страницей (можно вынести в tpl), но с помощью разных функций.

Вид страницы:



Исходный код:

<html>

<head>

<title>Голосование</title>

</head>

<body>

<p>Голосование</p>

<?

 //Простое голосование

 // ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ -----------------------

 //число вариантов ответа

 $n = 5;

 //вопрос

 $question="Как Вам наш сайт?";

 //варианты ответа

 $variants[1]="Здорово";

 $variants[2]="Нормально";

 $variants[3]="Так себе";

 $variants[4]="Отстой";

 $variants[5]="Никак";

 // ФУНКЦИИ --------------------------------------

 function ShowVote()

 {

 //считываем глобальные переменные

 GLOBAL $question, $variants, $n;

 //создаем форму

 print "<form id='vote' src='' method='POST'>\n";

 print "<p>$question\n";

 //в цикле добавляем варианты ответа

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

 {

 print " <br><input type='radio' name='variant' value='$i'>

 $variants[$i]

 </input>\n";

 };

 //кнопка "голосовать"

 print "<p><input type='submit' name='submit' value='Голосовать'>\n";

 //кнопка "результаты"

 print "<input type='submit' name='submit' value='Результаты'>\n</p>";

 print "</form>\n";

 };

 function DoVote($new\_result)

 {

 GLOBAL $question, $variants, $n;

 //файл, куда будем сохранять результаты

 $access = "vote.dat";

 //если файл существует

 if (file\_exists($access))

 {

 //считать текущее содержимое в массив

 $vote\_results = file($access);

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

 {

 $vote\_results[$i] = trim(str\_replace("\n","",$vote\_results[$i]));

 };

 }

 //иначе - это первый голос

 else

 {

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

 {

 $vote\_results[$i] = 0;

 };

 };

 //увеличиваем нужный вариант на 1

 if (($new\_result > 0) && ($new\_result <= $n))

 {

 $vote\_results[$new\_result] += 1;

 };

 $sum = 0;

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

 {

 $sum += $vote\_results[$i];

 };

 //запись в файл и вывод на экран результатов

 $fh = fopen($access, "w");

 print "<p><b>Результаты голосования</b></p>\n";

 print "<p>$question</p>\n";

 fwrite($fh, "\n");

 //для всех вариантов

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

 {

 $cur\_result=$vote\_results[$i]; //результат по i-тому варианту

 $cur\_variant=$variants[$i]; //текст i-того варианта

 fwrite($fh, $cur\_result."\n");

 print "<br>$cur\_variant: $cur\_result";

 if ($sum > 0) {

 $bar\_width = 100 \* $cur\_result / $sum;

 print "<div class='vote\_bar' style='background-color: #0050F0; height:5px; width: $bar\_width'.'px'></div></p>";

 };

 };

 print "<p>Всего: $sum голосов</p>";

 fclose($fh);

 };

 // ОСНОВНОЙ КОД --------------------------------------

 // отобразить голосование

 $submit = $\_POST["submit"];

 if ($submit != "Голосовать") { //если еще не голосовали

 if ($submit != "Результаты") { //и не просили показать результаты

 ShowVote(); // показать вопросы

 }

 else { //иначе

 DoVote(-1); // показать результаты опроса

 };

 }

 else //иначе

 {

 //проголосовать и показать результаты

 $variant = $\_POST["variant"];

 if ($variant > 0) { //дан голос

 print "<p>Спасибо, Ваш голос принят!</p>";

 DoVote($variant); //сохраням и выводим результаты

 }

 else { //иначе

 DoVote(-1); // показать результаты опроса

 };

 };

?>

</body>

</html>

### Форма для отправки электронной почты

Функция (string получатель, string тема, string сообщение [, string доп\_заголовки]) отправляет сообщение заданному адресату по электронной почте.

Отправка писем должна поддерживаться сервером.

<html>

 <head>

 <title>Отправка письма</title>

 </head>

<body bgcolor="#ffffff" text="#000000" link="#cbda74" vlink="#808040" alink="#808040">

<?php

$form = "

<form id=\"sendmail\" action=\"\" method=\"post\">

<input type=\"hidden\" name=\"seenform\" value=\"y\">

<b>Присылайте Ваши комментарии!</b><br>

Ваше имя:<br>

<input type=\"text\" name=\"name\" size=\"20\" maxlength=\"20\" value=\"\"><br>

Ваш Email:<br>

<input type=\"text\" name-\"email\" size=\"20\" maxlength=\"40\" value=\"\"><br>

Комментарий:<br>

<textarea name=\"comments\" rows=\"3\" cols=\"30\"></textarea><br>

<input type=\"submit\" value=\"Отправить\">

</form>

";

$seenform = $\_POST["seenform"];

// Если форма ранее не отображалась, отобразить ее.

// Для проверки используется значение скрытой переменной $seenform.

if ($seenform != "y") {

 print "$form";

}

else {

 $name = $\_POST["name"];

 $email = $\_POST["email"];

 $comments = $\_POST["comments"];

 // Переменная $recipient определяет получателя данных формы (укажите свой email)

 $recipient = "kornast@yandex.ru";

 // Тема сообщения

 $subject = "Комментарий от $name";

 // Дополнительные заголовки

 $headers = "From: $email";

 // Отправить сообщение или выдать сообщение об ошибке

 mail($recipient, $subject, $comments, $headers) or die("Не удалось отправить сообщение!");

 // Вывести сообщение для пользователя

 print "Спасибо вам, $name, за Ваш комментарий!";

};

?>

</body>

</html>

## Список литературы

1. Учебник PHP. [Электронный ресурс] – Режим доступа: <http://phpclub.ru/manrus/>
2. Учебники и справочники по языку PHP. [Электронный ресурс] – Режим доступа: <http://www.php.su/books/?php>