Дмитриев П. О.

ПРАКТИКУМ ПО ВЕБ-ПРОГРАММИРОВАНИЮ

Теоретическое введение в язык PHP

Дмитриев П. О.

Практикум по веб-программированию

Теоретическое введение в язык PHP

Учебное пособие для студентов, обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика»

Саратов
ООО Издательский Центр «Наука»
2016

УДК 004.432(076.5)
ББК 32.973.018я73
Д53

Дмитриев П.О.

Д53 **Практикум по веб-программированию. Теоретическое введение в язык PHP:** Учебное пособие для студентов, обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика» /П.О. Дмитриев. — Саратов: ООО Издательский Центр «Наука», 2016. — 40 с.

ISBN 978-5-9999-2715-6

Данное учебно-методическое пособие представляет собой методическую разработку для обучения основам веб-программирования на языке PHP. Материал разбит на темы и выстроен в последовательности позволяющей наиболее быстро освоить основы программирования на языке PHP.

Для студентов, обучающихся по направлению подготовки бакалавриата 09.03.03 «Прикладная информатика», также может быть полезно для начинающих веб-разработчиков.

Рецензенты:

Кафедра математического обеспечения вычислительных комплексов и информационных систем Саратовского национального исследовательского государственного университета имени Н.Г. Чернышевского

Зав. кафедрой, д.ф.-м.н., профессор ***Д.А. Андрейченко***

УДК 004.432(076.5)
ББК 32.973.018я73

Работа издана в авторской редакции

ISBN 978-5-9999-2715-6

Содержание

Тема 1. Написание скриптов на языке PHP

PHP

Уже известный Вам HTML является языком размётки текста (а CSS — технологией оформления текста, размеченного при помощи HTML), но не является языком программирования, то есть не позволяет организовать циклы, ветвления и прочие алгоритмические конструкции. Вам, вероятно, известно? что некоторую динамику странице можно придать средствами JavaScript. Но программа на JavaScript содержится в самой странице, а значит выполняется на Вашем компьютере как только страница загружена (к слову, обработчик JavaScript на Вашем компьютере может быть и отключен). Что же делать, если необходимо проводить какую-либо обработку на стороне сервера? Например, выборку из базы данных и т.п. Для этих целей используется язык программирования на стороне сервера. Мы рассмотрим один из таких языков — PHP.

Официальное название PHP — hypertext preprocessor (гипертекстовый препроцессор); он является языком сценариев, выполняющихся на сервере. Когда браузер делает запрос к указанному URL, например, http://nto.immpu.sgu.ru/index.php, Web-сервер активизирует интерпретатор PHP, который выполняет PHP-код, расположенный в файле index.php, и возвращает полученный результат Web-серверу. Этот результат Web-сервер встраивает в документ в то место, где был код (то есть код заменяется на результат выполнения этого кода), а затем собранный документ отправляется браузеру для отображения.

Среда разработки

Для написания программ (скриптов) на языке PHP достаточно любого текстового редактора. Даже стандартный для ОС Windows текстовый редактор Блокнот подходит.

Но наиболее удобными являются редакторы с подсветкой синтаксиса и другими сервисными возможностями, помогающими разработчику.

Одними из самых удобных текстовых редакторов считаются следующие:

1. Notepad++ (Windows)
2. Eclipse for PHP Developers (Linux, Windows, MacOS)
3. Kate (Linux)

Эти редакторы удобны для разработки и относятся к программному обеспечению с открытым исходным кодом, распространяемому по лицензии GNU GPL. То есть использование этого программного обеспечения (кроме его продажи) совершенно легально.

Первый скрипт

Вы уже составляли HTML-документы, в которых применяли: теги (и атрибуты), ссылки на сущности (например, ©), комментарии. Кроме этого, в документе могут присутствовать инструкции.

Инструкция имеет вид: <?app instruction ?>. В этой записи app — это приложение, которому будет передана инструкция, а instruction — сама инструкция.

Запишем самую популярную программу "Hello, world!":

```
<?php
   // Используется как пример с 1978 года
   print "Hello, world!";
?>
```

Разберём подробнее, что есть в этом примере. Строка (1) содержит начало инструкции и указание что инструкцию обработает интерпретатор PHP. Строка (2) содержит комментарий. Комментарии в PHP начинаются с двойной наклонной черты, это означает что всё что далее следует до конца строки является комментарием и не анализируется. Строка (3) содержит команду вывода на печать и её аргумент. Строка (4) содержит завершение инструкции.

Сохраните этот файл с именем hello.php. Далее возможны несколько вариантов запуска.

Вариант 1. В дисплейных классах механико-математического факультета установлены ОС Gentoo Linux, в составе которых присутствует интерпретатор PHP. То есть запустить свой скрипт Вы можете набрав в консоли: php hello.php. При этом результат работы скрипта будет распечатан в консоли.

Вариант 2. Вы можете скопировать свой скрипт в папку на студенческом веб-сервере факультета и запустить свой скрипт, сделав запрос в браузере: http:// student.math.sgu.ru /student/hello.php

Для того, чтобы Ваши скрипты не смешивались со скриптами Ваших коллег, создайте в корневой папке веб-сервера папку, назвав её номером Вашей группы и фамилией, например: 351-Ivanov. Тогда скрипт будет вызываться так: http://student.math.sgu.ru/student/351-Ivanov/hello.php

Вернёмся к примеру. В начале было написано, что PHP удобен для создания частей документов. Проиллюстрируем на примере hello.php:

```
<html>
<head>
<title>Prac 1</title>
</head>
<body>
<?php
print "Hello, world"
?>
</body>
</html>
```

В одной странице может быть сколько угодно вставок PHP-скриптов. В том числе, страница целиком может представлять собой скрипт. Покажем на примере вышенаписанного скрипта:

```
<?php
   print "<html>";
   print "<head>";
   print "<title>Prac 1</title>";
   print "</head>";
```

```
    print "<body>";
    print "Hello, world!";
    print "</body>";
    print "</html>";
?>
```

Тема 2. Переменные в PHP

PHP является Си-подобным языком, но обозначение переменных он унаследовал из языка PERL. Поэтому имя переменной в PHP начинается со знака $.

Корректное имя переменной должно начинаться (сразу после знака $) с латинской буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания.

Присваивание в PHP реализуется знаком =. Также как и в языке Си, в PHP есть сокращённые виды присваиваний +=, -=, *= и др.

Типы данных

Целые числа (int)

```
<?php
$a = 1234; // десятичное число
$a = -123; // отрицательное число
$a = 0123; // восьмеричное число (эквивалентно 83 в десятичной системе)
$a = 0x1A; // шестнадцатеричное число (эквивалентно 26 в десятичной системе)
?>
```

Дробные числа (float)

```
<?php
$a = 1.234;
$b = 1.2e3; //   1.2 x 10^3
$c = 7E-10; //   7.0 x 10^-10
?>
```

Строки (string)

Строка – это набор символов. В PHP символ – это то же самое, что и байт, это значит, что возможно ровно 256 различных символов.

Замечание 1. Нет никаких проблем, если строка очень велика. Практически не существует ограничений на размер строк, налагаемых PHP, так что нет абсолютно никаких причин беспокоиться об их длине.

Замечание 2. PHP позволяет использовать многобайтовые кодировки, в которых один символ кодируется двумя байтами. В этом случае для работы со строкой требуется использование строковых функций из специальной библиотеки Multi-byte strings.

Строки в PHP можно определить тремя способами: одинарными кавычками, двойными кавычками и heredoc синтаксисом (многострочная строка).

```
<?php

print 'это простая строка';

print 'Также вы можете вставлять в строки
символ новой строки таким образом,
поскольку это нормально';
// Выведет: Однажды Арнольд сказал: "I'll be back"
print 'Однажды Арнольд сказал: "I\'ll be back"';
// Обратите внимание: в строке присутствует одинарная кавычка.
Для того, чтобы php
// понял, что это не конец строки, а часть ее, нам нужно ему
об этом сообщить.
// Обратный слэш выполняет функцию экранирования.

// Выведет: Это не вставит: \n новую строку
print 'Это не вставит: \n новую строку';
// В операционных системах используются особые специальные
символы в тексте
// которые помогают системе понять, где находится конец строки
и начинается новая,
// где находится табуляция и где вообще заканчивается весь
текст.
// Символ \n обозначает конец строки и переход на новую.
// Слэш является частью символа, а не экранированием.
// Если бы php всё таки вставил этот символ, то мы бы получили
вывод в две строки:
// Это не вставит:
// новую строку

// Выведет: Переменные $expand также $either не подставляются
print 'Переменные $expand также $either не подставляются';
?>
```

Так же как и в одинарных, текст взятый в двойные кавычки это строка.

Если строка определяется в двойных кавычках, либо при помощи heredoc, переменные внутри нее обрабатываются.

Если интерпретатор встречает знак доллара $, он захватывает так много символов, сколько возможно, чтобы сформировать правильное имя переменной. Если вы хотите точно определить конец имени, заключайте имя переменной в фигурные скобки.

```
<?php
$beer = 'Heineken';
print "$beer's taste is great"; // работает, "'" это неверный символ для имени переменной
print "He drank some $beers";   // не работает, 's' это верный символ для имени переменной
print "He drank some ${beer}s"; // работает
print "He drank some {$beer}s"; // работает
?>
```

Heredoc-текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки в heredoc.

```
<?php
print <<<HEREDOC1
Меня зовут "$name". Я печатаю very$fast.
Теперь я вывожу very{$fast}.
Это должно вывести заглавную букву 'A': \x41
HEREDOC1;
?>
```

В данном примере строка "Меня зовут ... ... букву A" образовывается при помощи синтаксиса HereDoc. Обратите внимание что вместо кавычек в начале и конце строки стоит название строки – HEREDOC1. <<

Массив (array)

Массив в PHP это намного более гибкая и сложная конструкция, чем Вы привыкли. С одной стороны Вы можете работать с ним как работали с массивами

в Си. С другой стороны, по сравнению с Си появилось множество дополнительных возможностей.

```
<?php
$array = array('Строка1','Строка2');
?>
```

В примере массив состоит из двух строк. Пока не наблюдается никаких аномалий относительно известных Вам фактов. Продолжим.

```
<?php
$array = array('Строка1','Строка2'); // Определяем массив с двумя строками
print $array[0]; // Обращение к нулевому элементу. Выведет Сыр.
$array[1] = 'Изменённая строка'; // Присвоение первому элементу.
print $array[1]; // Выведет изменённую строку.
?>
```

В нашем примере использовались индексы элементов массива. В PHP понятие индекса расширяется до понятия ключа. То есть индексом может быть любая строка.

```
<?php
$arr = array('key'=>'val' , 'key2'=>'val2'); // Определяем массив с двумя строками
print $arr['key']; // Обращение к элементу с ключом key. Выведет val
print $arr[0]; // Обращение к нулевому элементу. Ошибка. Нет такого элемента
$arr['key2'] = 'Изменённая строка'; // Присвоение элементу с ключом key2 нового значения.
print $arr['key2']; // Выведет изменённую строку.
?>
```

Более того, массив может содержать элементы любых различных типов данных в том числе и другой массив. Массив массивов называется многомерным массивом.

Всего есть два типа массивов – числовые и ассоциативные.

Первый вид представляет из себя массив с ключами в виде чисел как в самом первом примере. Всё остальное является ассоциативным массивом.

```
<?php

$ARRAY = array // числовой. Ключи 0,1 и 2
(
 array('Строка1','Строка2') , // Числовой. Ключами являются 0 и 1
 array('key'=>'val'  ,  'key2'=>'val2')  ,  //  Ассоциативный. Ключи key и key2
 array('key3'=>'val3', 'Строчка') // Тоже ассоциативный. Ключи key3 и 0
);

// Для обращения к какому либо элементу многомерного массива, к примеру к val3
// следует обращаться так:
print $ARRAY[2]['key3'];
// В массиве ARRAY обращаемся сначала к элементу номер 2 [Array('key3'=>'val3', 'Строчка')]
// а там к элементу с ключом key3


$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));
print $arr["somearray"][6];    // 5
print $arr["somearray"][13];   // 9
print $arr["somearray"]["a"];  // 42

?>
```

Как видно, массив может иметь многомерную, причём нерегулярную, структуру. То есть, например, можно построить массив, состоящий из двумерного массива, одномерного массива и скаляра.

Создание и заполнение массивов

Содержание массива может быть задано как это было в примерах — заранее.

```
<?php $arr = Array(1,2,3); // ключи 0,1 и 2 ?>
```

Так же как мы присваивали ранее элементу в массиве другое значение можно создать и новый элемент

```
<?php $arr[3] = 5; // Создаётся элемент с индексом 3 и ему тут же присваивается число (integer) 5. ?>
```

Также можно добавлять элемент со следующим порядковым индексом.

```
$arr[] = 6;
```

В этом случае, создастся элемент в массиве, ему присвоится значение 6, а индекс с которым элемент запишется в массив будет максимальный целочисленные уже существующий индекс + 1.

После $arr[3] = 5; максимальный целочисленный индекс стал 3.

При $arr[] = 6; максимальный индекс + 1 будет 4.

Таким образом, $arr[4] становится 6.

Для вывода сложных структур, в том числе массивов, используется функция print_r().

Объекты (object)

PHP с 5 версии стал поддерживать объектно-ориентированный подход.

```
<?php

// Определяем что такое машины, что она умеет делать и как.
class car
{
    var $color = 'White';
    function tut_tut()    {  print "BIp BIp";  }
}

$honda = new car; // присваиваем переменной honda объект - Машину
```

```
print $honda->color; // Выводим свойство машины - цвет
?>
```

Ресурс (resource)

Ресурс представляет из себя указатель, ссылку, на внешний ресурс. В PHP ресурс — это ссылка на дескриптор (описатель) некоторого внешнего для PHP объекта. Например, при работе с базами данных, Вы будете получать в переменную результат, который вернула СУБД в ответ на ваш запрос. Типом переменной результата будет ресурс.

Null

Чтобы понять, что такое Null, нужно прежде всего узнать, как PHP хранит переменные в памяти. Компьютерная память состоит из большого количества ячеек. В эти ячейки и записываются данные. Обращение к той или иной ячейке происходит при помощи идентификатора ячейки в памяти, её места.

На деле интерпретатор хранит связь между названием переменной и идентификатором ячейки в памяти. Мы же в свою очередь работаем с названием, вместо того чтобы самим запоминать идентификаторы ячеек, которые и без того, известны одному интерпретатору.

Когда мы присваиваем переменной значение в первый раз, интерпретатор выбирает ячейку в памяти, заносит туда наше значение и привязывает название переменной к определенному идентификатору.

Когда к названию переменной соответствует ячейка в памяти, переменная считается определенной. Пустая переменная означает что содержимое ячейки является числом ноль, строкой нулевой длины или вовсе отсутствует.

Значение Null сбрасывает привязку названия переменной к ячейке в памяти и переменной больше не соответствует никакая ячейка. Переменная просто никуда не указывает и ни к чему не ведет. В этом случае переменная считается неопределенной, так как название есть, а привязка к ячейке отсутствует.

Присвоение переменной значения Null можно делать так: $a = Null; Заметьте, что слово Null не взято в кавычки, потому что это не строка с текстом.

Булев тип (boolean)

Тип данных который включает в себя всего два значения: правда и неправда (True и False). Используется в основном в условиях. К примеру выражение ($a > $b) может быть правдой или не правдой.

Мягкость к типам данных

PHP не требует объявлять заранее какого типа будет та или иная переменная. Типы переменных определяются и приводятся в зависимости от контекста.

```
<?php
$a="5"; // Строка содержащая символ 5
$a=$a+10; // Переменная переведена в тип int и участвует в операции сложения
print $a; // Выведет 15
?>
```

Обозначения в документации PHP

mixed –обозначает что функция возвращает разные типы данных, в зависимости от обстоятельств

number – означает что возвращаемый результат представляет из себя integer или float

callback – в PHP функция вызывается как её название и скобочки. К примеру print(). Некоторые функции в PHP (к примеру call_user_func()) принимают в качестве параметров названия функций с которыми они будут работать. В таком случае параметр формата callback которые они требуют представляют из себя строку с названием функции без скобочек. К примеру call_user_func('increment', $a); Данный аспект будет подробнее рассмотрен и применён во второй практической работе.

Полезные функции

isset() – в качестве аргумента функции передаётся переменная. Результат возвращаемый функцией: TRUE — если переменная, переданная в аргументе является определённой, и FALSE в противном случае.

print_r() – распечатывает в удобном для чтения виде значение переменной. Очень удобная для вывода значений массивов и полей объектов.

gettype() – возвращает тип переменной, переданной в аргументе.

is_float() – возвращает TRUE, если переменная в аргументе в настоящий момент имеет тип float.

is_int() – возвращает TRUE, если переменная в аргументе в настоящий момент имеет тип int.

is_string() – возвращает TRUE, если переменная в аргументе в настоящий момент имеет тип string.

is_array() – возвращает TRUE, если переменная в аргументе в настоящий момент имеет тип array.

is_object() – возвращает TRUE, если переменная в аргументе в настоящий момент имеет тип object.

intval() – возвращает аргумент в виде целого числа integer.

floatval() – возвращает аргумент в виде дробного числа float.

strval() – возвращает аргумент в виде строки string.

settype() – преобразует первый аргумент в указанный во втором аргументе тип. Наример, settype($a, 'integer').

Тема 3. Операции

Арифметические операторы

-$a	Отрицание. Смена знака $a.
$a + $b	Сложение. Сумма $a и $b.
$a - $b	Вычитание. Разность $a и $b.
$a * $b	Умножение. Произведение $a и $b.

$a / $b	Деление. Частное от деления $a на $b.
$a % $b	Деление по модулю. Целочисленный остаток от деления $a на $b.

Операторы инкремента и декремента

++$a	Префиксный инкремент. Увеличивает $a на 1 и возвращает значение $a.
$a++	Постфиксный инкремент. Возвращает значение $a, а затем увеличивает $a на 1.
--$a	Префиксный декремент. Уменьшает $a на 1 и возвращает значение $a.
$a--	Постфиксный декремент. Возвращает значение $a, а затем уменьшает $a на 1.

Таким образом $a = 5; print $a++ ; Сначала выведет 5, а потом увеличит 5 на 1.

Логические операторы

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

Строковые операторы

В PHP есть два оператора для работы со строками. Первый — оператор конкатенации ('.'), который возвращает объединение левого и правого аргумента в одну строку. Второй — оператор присвоения вместе с конкатенацией.

```
<?php

$a = "Hello, ";
$b = $a . "world!"; // $b содержит строку "Hello, world!"

$a = "Hello, ";
$a .= "world!";     // $a содержит строку "Hello, world!"

?>
```

Побитовые операторы

Побитовые операторы позволяют устанавливать конкретные биты в 0 или 1 для целочисленных значений. В случае если и левый, и правый операнды строки, побитовые операции будут работать с их ASCII-представлениями.

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

$a >> $b	Сдвиг вправо Все биты переменной $a сдвигаются на $b позиций вправо (каждая позиция подразумевает 'деление на 2')

Операторы сравнения

$a == $b	Проверка равенства. TRUE если $a равно $b.
$a === $b	Проверка на равенство с предварительным принудительным приведением сравниваемых значений к одному типу.
$a != $b	Проверка неравенства. TRUE если $a не равно $b.
$a <> $b	Проверка неравенства. TRUE если $a не равно $b.
$a !== $b	Проверка на неравенство с предварительным принудительным приведением сравниваемых значений к одному типу.
$a < $b	Сравнение. TRUE если $a строго меньше $b.
$a > $b	Сравнение. TRUE если $a строго больше $b.
$a <= $b	Сравнение. TRUE если $a меньше или равно $b.
$a >= $b	Сравнение. TRUE если $a больше или равно $b.

Операторы работы с массивами

$a + $b	Объединение. Объединение массива $a и массива $b.
$a == $b	Равенство. TRUE в случае, если $a и $b содержат одни и те же элементы.
$a === $b	Тождественно равно TRUE в случае, если $ a и $b содержат одни и те же элементы в том же самом порядке.
$a != $b	Неравенство. TRUE если массив $a не равен массиву $b.

$a <> $b	Неравенство. TRUE если массив $a не равен массиву $b.
$a !== $b	Тождественно не равно. TRUE если массив $a не равен тождественно массиву $b.

Тема 4. Ветвления и циклы

Оператор ветвления

В операторе if ставиться условие, если оно выполняется, то выполняется часть кода, записанная в фигурных скобках за ним; если условие не выполняется, то часть кода, идущая после оператора else. Также можно создавать сложные условия с помощью elseif. Схематично выглядит так:

```
if(условие){
...
}elseif(другое условие){
...
}else{
...
}
?>
```

По-русски это звучит так: «Если условие выполняется, то выполняется одно, если нет, то что-то другое». Следующий вопрос — это как создавать условия. Для этого используются переменные и операторы сравнения. Например, так:

```
<?php

$a = 5;
$b = 6;
$c = 5;
if($a != $b && $b > $c) {
  echo "Условие выполняется";
}else{
```

```
echo "Условие не выполнилось";
}
?>
```

Оператор выбора

Этот оператор позволяет делать более сложные ветвления, чем if, более простым способом. На самом деле, при помощи if можно полностью заменить этот оператор, но знать про него полезно.

Итак, этот оператор называется switch. В дословном переводе — «переключатель». Рассмотрим на примере. Пусть нужно вывести день недели по его порядковому номеру.

```
<?php
// $iDayNum получено ранее
switch($iDayNum)
{
   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;
    default:
        print "Какая-то ошибка!";
}
?>
```

При входе в оператор switch, сначала вычисляется выражение, стоящее в скобках. Затем перебираются все значения, указанные при помощи case. В данном примере мы видим 7 разных значений. Соответственно, когда значение, которое имеет вычисленное в скобках выражение, найдено у какого-либо case, начинается исполнение участка когда внутри case. И вот здесь важно понимать, что означает "внутри case". Когда соответствующее значение найдено, произойдет исполнение всего, что есть дальше того case, которому соответствует значение. И это значит, что если после этого case есть другие, они тоже будут выполнены. Этот эффект алгоритмически обоснован и часто нужен. Но часто он и не нужен.

В таком случае можно воспользоваться оператором break. Он позволяет выйти из оператора switch. Поставив его после кода, записанного внутри каждого case, мы обеспечим себе исполнение только одного участка кода для каждого значения.

Кроме этого, у оператора switch может быть необязательная часть default. Код в ней выполнится, если значение выражения не совпало ни с одним из значений в case.

Это очень похоже на часть else для оператора if .. elseif .. else Но не стоит забывать, что switch и if — не одно и то же. Иногда может пригодиться возможность switch исполнять весь участок кода, включая другие case, после того как найдено совпадение.

Циклы

В PHP реализованы стандартные алгоритмические конструкции: цикл с предусловие (while), цикл с постусловием (do...while), цикл for. Кроме того, есть некоторые неклассические циклы, например, цикл по элементам нескалярной переменной (foreach).

Цикл с предусловием

Цикл while — простейший тип цикла. Он действует как и его аналог в Си. Основная форма оператора while:

```
while (условие) {
...
}
```

Смысл оператора while прост. Он предписывает PHP выполнять вложенный(е) оператор(ы) до тех пор пока условие выполняется. Значение выражения проверяется каждый раз при начале цикла, так что если значение выражения изменится внутри цикла, то он не прервется пока не начнется следующий цикл. Иногда, если условие не выполняется с самого начала, цикл не выполняется ни разу.

Если в цикле только один оператор, то фигурные скобки можно опустить.

```
while (условие) оператор;
```

Следующий пример выводит номера с 1 до 10:

```
<?php
$i = 1;

while ($i <= 10) {

print $i;
$i++;
```

```
}
?>
```

Цикл с постусловием

Цикл do..while очень похож на while за исключением того, что значение логического выражения проверяется не до, а после окончания итерации. Основное отличие в том, что do..while гарантировано выполнится хотя бы один раз, что в случае while не обязательно.

Для циклов do..while существует только один вид синтаксиса:

```
<?php
$i = 0;

do {

print $i;

} while ($i > 0);
?>
```

Этот цикл выполнится один раз, так как после окончания условие не выполнится ($i не больше 0), и выполнение цикла завершится.

Кроме того, Вы можете также использовать еще один интересный оператор в циклах — break. С помощью него можно в любой момент оборвать любой цикл, как while, do..while, так и for или foreach. Например:

```
<?php
$i = 1;

while ($i <= 10) {

print $i;
$i++;

if($i > 5) break;

}
?>
```

В данном случае цикл оборвется, когда переменная $i станет равной 6.

Цикл for

Циклы for — наиболее мощные циклы в PHP. Они также работают подобно их аналогам в Си.

Синтаксис цикла for:

```
for (инструкция инициализации; условие; шаговая инструкция){
...
}
```

Инструкция инициализации выполняется в начале цикла. В начале каждого шага цикла проверяется условие. Если оно выполняется, то цикл продолжается и выполняется шаговая инструкция. Если нет, то цикл заканчивается. В конце каждого шага цикла выполняется тело цикла.

Рассмотрим такой пример:

```
<?php
for ($i=1; $i<=10; $i++){
  print $i;
}
?>
```

Пример сводит for к классическому циклу со счётчиком (переменная $i меняется от 1 до 10 с шагом 1). Но на деле цикл for может быть устроен и сложнее. Каждый из его параметров может быть пустым. Если условия нет, то цикл продолжается бесконечно (PHP по умолчанию считает условие выполнившимся (равным TRUE), как и Си). Это не так бесполезно, как могло бы показаться, так как зачастую требуется закончить выполнение цикла используя оператор break в сочетании с логическим условием вместо использования логического выражения в for.

Рассмотрим следующие примеры. Все они выводят номера с 1 по 10 :

```
<?php

// Пример 1
for ($i = 1; $i <= 10; $i++) {
  print $i;
}

// Пример 2
for ($i = 1;;$i++) {
  if ($i > 10) break;
  print $i;
}

// Пример 3
$i = 1;
for (;;) {
  if ($i > 10) break;
  print $i;
  $i++;
}

// Пример 4
for ($i = 1; $i <= 10; print $i, $i++);
?>
```

Цикл по элементам

Цикл по элементам (foreach) очень удобен для обработки массивов (и других структурированных данных). Выглядит он очень просто. Для примера рассмотрим код, который выводит все элементы обычного массива:

```
<?php
// $m что-либо содержит, это массив
foreach($m as $item)
{
   print $item."\n";
}
?>
```

Этот цикл пробегается по всему массиву. А текущий элемент массива доступен в переменной $item.

Но этот цикл умеет больше. Очень часто нам нужен не только сам элемент, или, если быть точным, значение элемента, но и его ключ. В следующем примере вывод делается уже с ключами:

```
<?php
// $m что-либо содержит, это массив
foreach($m as $key=>$value)
{
   print "Элемент массива '".$key."' равен: ".$value."\n";
}
?>
```

Конструкция лишь немного усложнилась — в скобках мы указываем ключ=>значение. И соответствующие ключ и значение текущего элемента массива в теле цикла доступны как переменные $key и $value.

Ещё немного о прерывании циклов

Мы уже пользовались оператором break. Рассмотрим его более подробно.

Оператор break принимает параметр — вложенность. Рассмотрим пример с вложенным циклом.

```
<?php
for($i=1;$i<=9;$i++)
{
   for($j=1;$j<=10000;$j++)
   {
      print ($i*$j)." ";
      if($j==9)
      {
         break(2);
      }
   }
}
?>
```

В данном примере мы сразу же выйдем из обоих циклов, так как их всего 2. По умолчанию break применяется значение 1, то есть break(1) равносильно break.

Кроме break существует еще один управляющий циклом оператор — continue. Он позволяет сразу же переходить к следующей итерации цикла, не выполняя в текущей ничего, что стоит после этого оператора:

```
<?php
for($i=1;$i<=9;$i++)
{
   for($j=1;$j<=11;$j++)
   {
      if($j>9)
      {
         continue;
      }
      print ($i*$j)." ";
   }
   print "<br />\n";
}
?>
```

Тема 5. Функции, определяемые пользователями

Функции существуют во многих языках программирования. Они используются для выделения кода, который выполняет отдельную, четко определенную задачу. Это упрощает чтение кода и позволяет повторно его использовать при каждом выполнении задачи.

Под функцией понимают независимый блок кода, который устанавливает интерфейс вызова, выполняет определенную задачу и возвращает результат (если результат не возвращается, будем считать что возвращается результат типа «пустота»).

Прежде чем вызывать функцию её нужно определить (описать). При попытке вызвать функцию, которая не существует, отобразится сообщение об ошибке.

При отображении сообщения об ошибке необходимо проверить:

1. Правильно ли указано имя функции.
2. Существует ли данная функция в используемой версии PHP.

На имена функций накладываются следующие ограничения:

1. Функция не может иметь то же имя, что у существующей функции.
2. Имя функции может содержать только буквы, цифры и символы подчеркивания.
3. Имя функции не может начинаться с цифры.

Многие языки программирования допускают повторное использование имен функций. Это свойство называется перегрузкой функций. Однако PHP не поддерживает перегрузку функций, поэтому функция не может иметь имя, совпадающее с именем любой встроенной или существующей определяемой пользователем функции.

Пример описания функции, принимающей в аргументе один параметр $x и возвращающей значение sin($x)/cos($x):

```
<?php
function tangens($x)
{
   return sin($x)/cos($x);
}
?>
```

Часто требуется передавать в функции один, два и более параметров. В примере в функцию передаётся один параметр, имеющий формальное имя x. Это означает что внутри описания функции к этому параметру следует обращаться по имени $x. При этом фактический параметр при вызове может иметь иное имя, например print tangens($z), или быть литералом, например print tangens(0). Ниже приведен пример функции, которая требует передачи в нее параметра.

Функции могут иметь несколько параметров и необязательные параметры. Усовершенствуем функцию tangens(), добавив ей необязательный параметр radian, который по умолчанию (если его не указывать), считается равным 1 и означает что расчёты ведутся в радианах; в противном случае расчёты ведутся в градусах.

```
<?php
function tangens($x, $radian=1)
{
   if ($radian == 1){
      return sin($x)/cos($x);
   }else{
      $radian=$x * pi / 180;
      return sin($radian)/cos($radian);
   }
}
?>
```

Первый параметр этой функции обязательный, а второй необязательны.

Из общего курса программирования Вам известно, что параметры могут передаваться по значению и по ссылке. В PHP чтобы для передачи параметра по ссылке используется знак &. Рассмотрим на примере

```
<?php
function f1($x){
   $x++;
}

function f2(&$x){
  $x++;
}

$x = 1;
f1($x);
print $x; // Выведет 1
f2($x);
print $x; // Выведет 2
?>
```

Тема 6. Передача параметров скрипту

Программа на PHP запускается в момент обращения к веб-серверу по адресу страницы, содержащей эту программу. После отработки, PHP-скрипт возвращает результат и прекращает свою деятельность.

Вы уже программировали на Pascal и C++ приложения, работавшие в консоли или в графическом интерфейсе пользователя. Консольное приложение может запросить ввод с клавиатуры и ожидать (не продолжать работу), пока пользователь вводит необходимые данные. Также и приложение графического

интерфейса может «ничего не делать» пока не нажата кнопка или не введены нужные значения.

В случае с выполнением PHP-скрипта на веб-сервере, запросить клавиатурный ввод невозможно. Скрипт должен иметь все необходимые ему для работы параметры уже в момент старта. Для этого используют передачу параметров в запросе веб-серверу.

Спецификация протокола HTTP описывает несколько возможных методов передачи параметров. Мы рассмотри два, доступных при работе с HTML метода: GET и POST.

Запуску любого скрипта предшествует некоторый объём технических операций, выполняемых веб-сервером и интерпретатором PHP. В частности, до того, как исполнится первая команда скрипта, создаются предопределённые переменные.

Набор предопределённых переменных может отличаться в зависимости от версии и типа веб-сервера, а также от версии PHP.

Начиная с версии PHP 4.1.0 введены предопределённые переменные $_GET и $_POST. Они содержат в себе все данные, переданные в запросе по соответствующему методу.

Передача параметров методом GET

Запишем один из наиболее полных видов строки запроса HTTP.

При указании в вызове (URL) строки параметров в виде '?параметр1=значение1&параметр2=значение2&...&параметрN=значениеN' PHP можно заставить создать переменные с именами параметр1, ..., параметрN и значениями значение1, ... значениеN соответственно.

Пример «Вежливый скрипт»

```
<?php
   $name = $_GET['name'];
   if (!isset($name)){
      print "Я не знаю как Вас зовут";
   }else{
      print "Здравствуйте, $name!";
   }
?>
```

Если не помните, найдите в Теме 2 функцию isset() и узнайте её назначение.

Создайте файл 'hi.php', поместите в него текст примера. Затем положите этот файл в папку на веб-сервере (папка веб-сервера доступна в каталоге /mnt/www, но чтобы Ваши файлы не путались с файлами других студентов, разумно создать в этом каталоге папку с именем 'ваш_номер_группы-ваша_фамилия', и помещать файлы туда). Затем обратитесь к Вашему вежливому скрипту, набрав в строке адреса веб-браузера 'http://student.math.sgu.ru/student/hi.php', если Вы создали свою папку, то ссылка будет немного другой:

```
http://student.math.sgu.ru/student/ваша_папка/hi.php').
```

При обращении к скрипту без параметров, он выведет 'Я не знаю как Вас зовут'. При обращении

```
http://honda.immpu.sgu.ru/ваша_папка/hi.php?name=Вася
```

скрипт выведет: 'Здравствуйте, Вася!'

Передача параметров методом POST

Для отправки данных методом POST удобно использовать HTML-формы. Модифицируем вежливый скрипт.

Пример «Вежливый скрипт возвращается»

```
<?php
   $name = $_POST['name'];
   $surname = $_POST['surname'];
   $age = $_POST['age'];
   if (!isset($name)){
      print "<h2>Заполните, пожалуйста анкету</h2>";
      print "<form method='POST' action=''>";
      print     "<p>Ваше     имя:     <input     type='text'
name='name'></p>";
      print    "<p>Ваша    фамилия:    <input    type='text'
name='surname'></p>";
      print    "<p>Ваш    возраст:    <input    type='text'
name='age'></p>";
      print     "<input     type='submit'     value='Отправить
данные'></p>";
      print "</form>";
   }else{
      print "Здравствуйте, $name $surname! Вам $age лет.";
   }
?>
```

В этом скрипте указано что форма будет передавать данные по методу POST. Обратите внимание, что при этом строка вызова скрипта не получает дополнительно строку параметров при отправке данных.

Сравнительная характеристика методов GET и POST

Попробуйте заменить в последнем скрипте POST на GET. Скрипт не потеряет функциональности. Но при отправке данных URL будет получать строку параметров и выглядеть примерно так: hi2.php?name=Vasya&surname=Pupkin&age=100

Таким образом, отправку данных через веб-форму можно производить как методом GET, так и методом POST. Возникает вопрос: в чём разница.

Запросы POST имеют такую же функциональность что и GET, но есть разница в формате.

GET отправляет всю информацию в заголовке запроса, длинна которого ограничена. Если нужно отправить какое-то слово или число на сервер &mdash

он вполне пригоден. Но метод GET оказывается неприменим если нужно отправить текст объёмом больше 255 символов (а вообще говоря даже чуть меньше).

POST отправляет информацию в теле запроса, который не имеет максимального размера. Можно отправлять огромные тексты, файлы и т.п. Именно поэтому этот метод используется при загрузке изображений.

По умолчанию всегда используется GET, и не только в формах, это более простой метод.

Достоинства метода GET

Страницу всегда можно сохранить в закладках (SEO-дружелюбен)

Скорость обработки выше, так как вся информация находится в заголовке

Информация, посылаемая на сервер, всегда видима (в адресной строке)

Недостатки метода GET

Информация, посылаемая на сервер, всегда видима (в адресной строке)

Объем информации, которую можно отправить, ограничен

Достоинства метода POST

Можно отправить много информации на сервер, объем почти неограничен

Отправляемая информация не показывается в адресной строке.

Недостатки метода POST

Медленнее, чем GET, так как анализируются заголовки и тело запроса.

Страницы, сгенерированные как результат запроса POST, нельзя добавить в закладки (CEO-недружелюбен)

Тема 7. Взаимодействие с СУБД

Системы управления базами данных (СУБД) предоставляют универсальный механизм накопления, хранения и обработки больших объёмов

информации. В веб-разработках СУБД применяются достаточно широко. База данных может хранить как сравнительно небольшой объём информации (например, список страниц сайта), так и огромные количества информации (например, записи курсов покупки/продажи валют с посекундной детализацией за несколько лет).

В распоряжении разработчика могут кроме СУБД также имеются другие средства, например, средства работы с файлами. СУБД имеет неоспоримое преимущество — универсальный язык запросов (SQL). Благодаря этому языку, можно строить запросы высокой сложности. При этом все сложности по реализации запроса решаются СУБД.

Соединение с СУБД

СУБД — это приложение, запущенное на некотором компьютере. Также как веб-сервер это приложение имеет архитектуру "Клиент-Сервер". То есть серверная часть этого приложения находится в постоянном ожидании запросов, при получении запроса выполняет этот запрос и отсылает клиенту результат.

Веб-приложение должно иметь возможность соединяться с СУБД, посылать запросы, получать результаты и затем эти результаты обрабатывать.

Далее будем рассматривать взаимодействие с СУБД на примере СУБД MySQL. Это наиболее распространённая среди веб-разработчиков в настоящее время СУБД.

Соединение осуществляется при помощи функции mysql_connect. Документация гласит:

```
resource mysql_connect (
    [string $server = ini_get("mysql.default_host")
    [, string $username = ini_get("mysql.default_user")
    [, string $password = ini_get("mysql.default_password")
    [, bool $new_link = false [, int $client_flags=0]]]]]
)
```

Разберёмся в формальных параметрах и их значениях. Квадратные скобки в описании синтаксиса означают что параметр является необязательным.

server – строка, содержащая имя или адрес хоста, на котором установлена СУБД mysql. Также может содержать номер порта, который прослушивает сервер MySQL. Таким образом, значения этого параметра могут быть, например: localhost, mysql.my_org.ru:1234, 127.0.0.1:5678 и прочие. Чаще всего СУБД MySQL установлена на том же компьютере, где расположен веб-сервер, поэтому в большинстве случаев придётся указывать 'localhost'. Кроме того, в описание синтаксиса показано что этот параметр имеет значение по умолчанию, которое хранится в файле настроек PHP.INI (чтобы оно было верно настроено, нужно самостоятельно это значение туда занести, добавив в нужный раздел строку вида 'default_host="localhost"').

username – имя пользователя, которое будет передано СУБД MySQL для прохождения аутентификации. Также как в случае с параметром server, возможно задание имени пользователя по умолчанию в файле PHP.INI.

password – пароль, необходимый для аутентификации пользователя.

new_link – по умолчанию этот параметр имеет значение false. Это означает, что при вторичном и последующих вызовах mysql_connect с теми же параметрами (сервер, логин, пароль) не будет создаваться новое соединение с сервером. Вместо этого будет возвращена в качестве результата ссылка на уже открытое соединение. Если установить значение true, при каждом вызове mysql_connetn будет открываться новое соединение.

client_flags – этот параметр указывает возможные особые настройки соединения, например, применение сжатия к передаваемым результатам и т.п.

В качестве результата mysql_connect возвращает ссылку на созданное соединение с сервером. Положив эту ссылку в переменную, можно затем указывать ещё в функциях, обращающихся к СУБД. Например, так:

```
<?php
// Соединяется с сервером MySQL, запущенным на localhost
$link = mysql_connect('localhost', 'vasya', '123');
if (!$link) {
    // В случае ошибки выдаём сообщение и прерываем выполнение
скрипта
    die('Не удалось соединиться с БД. Ошибка: ' .
mysql_error());
}
print 'Успешное соединение';
// Закрываем ранее открытое соединение
mysql_close($link);
?>
```

Одна СУБД может хранить множество различных баз данных. Нам же необходимо подключиться к одной из них. Для этого используется функция mysql_select_db.

```
bool mysql_select_db (string $database_name [, resource
$link_identifier])
```

В качестве первого аргумента функции передаётся имя базы данных. Второй аргумент не является обязательным, в нём указывается ссылка на открытое соединение. Если второй аргумент не указан, будет выбрано первое открытое соединение. Рассмотрим необходимость этого аргумента на примере.

Допустим, у есть два сервера, и необходимо осуществить получение данных с одного и отсылку их на другой, тогда соответствующий скрипт будет иметь вид:

```
<?php
// Соединяется с первым сервером
$link1 = mysql_connect('server.ru:3306', 'vasya', '123');
if (!$link1) {
    // В случае ошибки выдаём сообщение и прерываем выполнение
скрипта
    die('Не удалось соединиться с Server.RU. Ошибка: ' .
mysql_error());
}
$link2 = mysql_connect('localhost', 'vasya', '321');
if (!$link2) {
```

```
        // В случае ошибки выдаём сообщение и прерываем выполнение
скрипта
        die('Не удалось соединиться с localhost. Ошибка: ' .
mysql_error());
    }
    print 'Оба соединения успешно установлены.';
    // Подключаемся к базе WorkBase на Server.RU
    mysql_select_db('WordBase', $link1);
    // Подключаемся к базе HomeBase на localhost
    mysql_select_db('HomeBase', $link2);
    // Здесь следует код, реализующий операции с базами данных

    // Закрываем ранее открытые соединения
    mysql_close($link1);
    mysql_close($link2);
    ?>
```

Список литературы

1. Duckett J. Web Design with HTML, CSS, JavaScript and jQuery Set. – Wiley Publishing, 2014.
2. Клименко Р. А. Веб-мастеринг на 100%. 2-е изд. – " Издательский дом"" Питер""", 2015.
3. Квинт И. Создаем сайты с помощью HTML, XHTML и CSS на 100%. 3-е изд. – " Издательский дом"" Питер""", 2014.
4. Warren T. PHP Programming For Beginners: The Simple Guide to Learning PHP Fast!. – 2015.
5. Колисниченко Д. Н. PHP и MySQL. Разработка Web-приложений. 4-е изд. – БХВ-Петербург, 2013.
6. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд //СПб.: Питер. – 2013.
7. Mikowski M. S., Powell J. C. Single Page Web Applications //B and W. – 2013.
8. Лиза Г., Джейсон Г. Разработка веб-сайтов для мобильных устройств. – " Издательский дом"" Питер""", 2013.
9. Бессонов Л. В., Брагина И. Г. Операционные системы. Компьютерные сети: Пособие для студентов, обучающихся по дополнительной специальности «Компьютерная графика и веб-дизайн» // Саратов: Изд-во «Научная книга», 2009. – 44 с. – ISBN 978-5-9758-1063-2
10. Бессонов Л. В., Брагина И. Г. Информационные технологии в профессиональной деятельности преподавателя: Учеб. пособие. – Саратов: Изд-во «Научная книга», 2014. – 28 с. – ISBN 978-5-9758-1524-8
11. Бессонов Л. В. Формирование требований к официальному сайту вуза на примере официального сайта СГУ // Наука и образование в XXI веке. Сборник научных трудов по материалам Международной научно-практической конференции 30 января 2015 г.: в 5 частях. Москва, 2015, С. 105–108.
12. Шаталина А.В., Бессонов Л.В. О подходе к задаче автоматизации проверки корректности и полноты входящих в ООП документов // Фундаментальные и прикладные исследования в современном мире. 2016. № 13-1. С. 68-71.
13. Бессонов Л.В., Сецинская Е.В., Кухарев В.В. Анализ требований к дизайну сайта вуза для лиц с ограниченными физическими возможностями // Фундаментальные и прикладные исследования в современном мире. 2015. №11-4. С. 85-87.
14. Бессонов Л.В., Тышкевич С.В., Панкратов Д.В. О подходе к формированию персональных страниц преподавателей на официальном

сайте вуза // Фундаментальные и прикладные исследования в современном мире. 2015. №11-1. С. 120-123.
15. Дмитриев П.О., Никулин Б.Л. Моделирование системы представления олимпиадных задач на сайте вуза // Новые задачи технических наук и пути их решения: сборник статей Международной научно-практической конференции (10 апреля 2016 г., г. Пермь). Уфа: АЭТЕРНА. 2016. С. 26–28
16. Матершев И.В., Дмитриев П.О. О построении автоматизированной системы управления хостингом для обучающихся по IT-направлениям подготовки // Наука, образование и инновации: сборник статей Международной научно-практической конференции (25 июня 2016 г., г. Томск). В 4 ч. Ч.3. Уфа. 2016. С. 49–51.
17. Бессонов Л.В., Дмитриев О.Ю. Подход к написанию технического задания на разработку сайта поддержки олимпиадного движения по предмету // Фундаментальные и прикладные исследования в современном мире. 2016. №15-1. С. 154-157.
18. Амелин Р. В. Информационные технологии: учебное пособие для студентов механико-математического факультета, обучающегося по специальности 351400 «Прикладная информатика» (по областям). – Саратов: Изд-во Сарат. ун-та, 2006. – 52 с.
19. Амелин Р. В. Мировые информационные ресурсы: учебное пособие для студентов механико-математического факультета, обучающегося по специальности 351400 «Прикладная информатика» (по областям). – Саратов: Изд-во Сарат. ун-та, 2006. – 88 с.
20. Амелин Р.В. Правовой режим государственных информационных систем: монография / под ред. С.Е. Чаннова. М.: ГроссМедиа, 2016. – 338 с.
21. Амелин Р.В. Обязанность по представлению информации в федеральные информационные системы // Административное и муниципальное право. – 2015. – № 10. – С. 1081–1089.
22. Бессонов Л.В. Практикум по веб-программированию. Упражнения и практические задания: Учебно-методическое пособие для студентов, обучающихся по направлению подготовки бакалаврита 09.03.03 «Прикладная информатика» — Саратов: ООО Издательский Центр «Наука», 2016. — 40 с.

Учебное издание

Дмитриев П. О.

Практикум по веб-программированию

Теоретическое введение в язык PHP

Учебное пособие для студентов,
обучающихся по направлению подготовки бакалавриата
09.03.03 «Прикладная информатика»

Подписано в печать 18.11.2016. Формат 60x84 1/16. Бумага офсетная.
Гарнитура Times New Roman. Печать RISO. Объем 2,5 печ. л.
Тираж 100 экз. Заказ № 201.

ООО Издательский Центр «Наука»
410012, г. Саратов, ул. Пугачевская, 117, оф. 50

Отпечатано с готового оригинал-макета
Центр полиграфических и копировальных услуг
Предприниматель Серман Ю.Б. Свидетельство № 3117
410012, Саратов, ул. Московская, д.152, офис 311, тел. 26-18-19