Основы PHP

Интерфейс API

- API (Application programming interface)- интерфейс, обеспечивающий соединение и обмен данными между клиентами и сервером

Выбор API

- Среда, в которой выполняется задача
- Производительность
- Простота разработки
- Переносимость

Среда выполнения

- СИ– язык программирования общего назначения. Использование эффективно для автономный задач, а не для web-приложений. С не очень удобен для обработки текстов и управлению памятью. Компилируемая программа.

Язык *Perl (Practical Extraction and Report Language* - язык практических извлечений и отчётов)

- Хорошо приспособлен для обработки текстовой информации.
- Использует общий интерфейс шлюза (CGI).
- Автор языка Perl - Ларри Уолл (Larry Wall), лингвист по образованию.
- подходит для разработки сайтов с помощью модуля CGI.pm. Интерпретируемый тип.

Язык Python

- 1980 Гвидо ван Россум (голландия)
- Скриптовый язык
- Веб-фреймворк Django
- Не предназначен для вычислительных задач, для задач, которые требуют много памяти
- код на Python, зависит от системных библиотек (сложно перенести на другие системы)

Java

- 1991 программист Джеймсон Гослинг
- используется в разработке мобильных приложений, веб-сервисов, программного обеспечения
- Работает с БД Oracle

Среда выполнения

- PHP – разрабатывался специально для web-приложений (для интернет - технологий наиболее предпочтителен). Самое большое преимущество – простой доступ к БД.
- PHP (personal Home Page)– язык написания сценариев на стороне сервера, встроенный в тело web-страницы, таким образом делает страницы динамическими

- Программа на компилируемом языке при помощи специальной программы компилятора преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и далее записывается в исполняемый файл, который может быть запущен на выполнение как отдельная программа
- Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет (интерпретирует) ее текст без предварительного перевода. нтерпретатор переводит на машинный язык прямо во время исполнения программы.
- Программы на интерпретируемых языках можно запускать сразу же после изменения, что облегчает разработку.

Создание сценария

- <HTML>
- <HEAD><title>Первый PHP-сценарий</title></HEA>
- <BODY>
- **<?php**
- **echo "Добро пожаловать, пользователь!";**
- **?>**
- </BODY>
- </HTML>

Конструкции использования сценария

- <? . . . ?> - Сокращенная версия
- <% . . . %> - Стиль ASP
- <SCRIPT LANGUAGE="PHP">
- …..
- </SCRIPT> - Синтаксис, совместимый с редакторами HTML.
- <?php …… ?>

Переменные в PHP

- Имена переменных всегда начинаются с символа **$** и содержат произвольную комбинацию символов,
- В число допустимых символов входят заглавные и прописные латинские буквы, а также символы с ASCII-кодами в диапазоне от 127 до 255 (символы, не используемые в американском английском).
- Переменные в PHP могут быть определены, либо присвоением им значения, либо с помощью **var**

Примеры

- <?php
- $myvar = "foo"; /* Присвоение строки 'foo' */
- badvar = "test"; /* Неверно, нет символа $ */
- $another(test)var = "bad"; /* Неверно, нельзя использовать () */
- Использование PHP для разработки Web-приложений
- $php5 = "is cool"; /* Корректный синтаксис */
- $5php = "is wrong"; /* Неверно, начинается с цифры */
- ?>

Комментарий

- В PHP все, что находится между **/* и */,** трактуется как комментарий
- Для однострочных комментариев могут применяться либо //, либо #, что помещает в комментарий остаток строки:
- <?php
- $var = "foo"; // это игнорируется
- $var = "bar"; # это тоже
- ?>

Уничтожение переменной

- <?php
- $myvar = "Строка";
- unset($myvar); // Уничтожение переменной
- ?>

Хранение данных

- **Целые**
- <?php
- $my_int = 50; /* Стандартная десятичная нотация */
- $my_int = 062; /*То же число в восьмеричной нотации (начинается с цифры ' 0 ') * /
- $my_int = 0x32; /* Шестнадцатеричная нотация */
- ?>

```
<body>

<?php
echo "десятичное число-";
  $mmm=50;
  echo $mmm;
echo "<br>восьмеричное число-";
$my_int = 062;
  echo $my_int;
  echo "<br>шестнадцатиричное число-";
  $kkk=0x32;
  echo $kkk;

?>

</body>
```

Хранение данных

- **С ПЛАВАЮЩЕЙ ТОЧКОЙ** <?php
- /* Стандартная нотация с десятичной точкой */
- $my_float = 5.1;
- /* То же число в экспоненциальном представлении с плавающей точкой */
- $my_float = .051e2;
- ?>

Хранение данных

- **Разбираемые и не разбираемые строки**

```
<?php
$my_int = 50;
$string_one = "Значение переменной равно
$my_int<BR>";
$string_two = 'Значение переменной равно
$my_int<BR>';
echo $string_one;
echo $string_two;
?>
```

Ответ:

Значение переменной равно 50

Управляющие символы PHP

Строка управляющих символов	Результирующий символ
\n	Символ перевода строки
\ r	Символ возврата каретки
\t	Символ горизонтальной табуляции
\\	Символ обратного слэша
\$	Символ $
\ '	' Символ одинарной кавычки
\ "	" Символ двойной кавычки
\###	ASCII-символ (восьмеричный)
\x##	ASCII-символ (шестнадцатеричный)

Примеры использования управляющих символов

- <?php
- /* Неверная строка, не работает в PHP */
- $variable = "Знаете ли вы что такое "управляющие" символы?";
- /* Правильно сформатированная строка */
- $variable = "Знаете ли вы что такое \" управляющие\" символы?";
- ?>

Простые вычисления

- <?php
- $answer = 5 + 4 ; /* $answer теперь равно 9 */
- $answer = $answer - 5; . /* $answer теперь равно 4 */
- $answer = $answer/2; /* $answer теперь равно 2 */
- $answer = 1/3; /* Sanswer теперь равно 0.333333 */
- $answer =((5 + 4)*2)%7; /* $answer теперь равно 4 */
- ?>
- % целочисленное деление с выделением остатка

- **Сокращенная запись математических операций в PHP**
- <?php
- $answer=5; /*Присвоение исходного значения */
- $answer +=2; /*Эквивалент $answer = $answer + 2 ; */
- $answer *=2; */$answer теперь равно 14 */
- $answer %=5; /* $answer теперь равно 4 */
- **Сокращенная запись инкремента и декремента**
- <?php
- $answer++; /* Увеличивает $answer на 1 */
- $answer--; /* Уменьшает $answer на 1 */
- ++$answer; /* Увеличивает $answer на 1 */

- <?php
- $answer = 5;
- echo (++$answer)." ";
- echo "$answer
";
- $answer = 5;
- echo ($answer++)." ";
- echo $answer;
- ?>
- $answer++ увеличивает переменную $answer после выполнения оператора
- ++$answer увеличивает переменную перед выполнением оператора.
- **ОТВЕТ:**
- 6 6
- 5 6

Операция конкатенации строк

. - Эта операция обозначается символом точки и применяется для комбинации двух отдельных переменных (обычно — строковых) в одну строку:

- <?php
- $string = "Спасибо за покупку ";
- $newstring = Sstring . "этой книги!";
- Echo $string. "этой книги!";
- ?>

Конструкции языка ВЕТВЛЕНИЕ

- if(условие) {
- /* Код, выполняемый, если условие истинно */
- } [else {
-
- /* Код, выполняемый, если условие ложно */
- }]

ПРИМЕР

```
1   <?php
2   if (true) echo "Эта строка отображается всегда!<BR>";
3   if (false)
4   {
5   echo "Эта строка не отображается никогда.<BR>";
6   }
7   else {
8   echo "Эта строка тоже отображается всегда!<BR>";
9   }
10  ?>
11
```

Эта строка отображается всегда!
Эта строка тоже отображается всегда!

Таблица 1.2. Операции сравнения

Пример операции	*Действие*
`$foo == $bar`	`true`, если `$foo` равно `$bar`.
`$foo === $bar`	`true`, если `$foo` равно `$bar` и обе переменных относятся к одному типу.
`$foo != $bar`	`true`, если `$foo` не равно `$bar`.
`$foo !== $bar`	`true`, если `$foo` не равно `$bar` и обе переменных не относятся к одному типу.
`$foo < $bar`	`true`, если `$foo` меньше `$bar`.
`$foo > $bar`	`true`, если `$foo` больше `$bar`.
`$foo <= $bar`	`true`, если `$foo` меньше или равно `$bar`.
`$foo >= $bar`	`true`, если `$foo` больше или равно `$bar`.

Пример вложенных условий

```php
<?php
 $value=21;
if ($value > 0) {
if ($value <= 10) {
echo 'Значение переменной $value находится между 1 и 1 0  ' ;
}  else {
if ($value <= 20) {
echo 'Значение переменной $value находится между 1 и 20.';
} else {
echo 'Значение переменной $value больше 20.';
            }
}
}

?>
```

Значение переменной $value больше 20.

Таблица 1.3. Логические операции PHP

Операция	*Действие*
`$foo and $bar`	true, если `$foo` и `$bar` истинны.
`$foo or $bar`	true, если `$foo` или `$bar` истинны.
`$foo xor $bar`	true, если `$foo` или `$bar` истинно (но не оба сразу).
`!$foo`	true, если `$foo` ложно.
`$foo && $bar`	true, если `$foo` и `$bar` истинны.
`$foo || $bar`	true, если `$foo` или `$bar` истинны.

- В PHP операции AND и OR выполняются раньше, чем операции && и ||

ПРИМЕР

```php
<?php
 $value=21;
if (($value > 0)&&($value <= 10))
 {
echo 'Значение переменной $value находится между 1 и 10' ;
}  else {
if ($value <= 20) {
echo 'Значение переменной $value находится между 1 и 20';
} else {
echo 'Значение переменной $value больше 20.';
          }
}


?>
```

Конструкция ELSEIF

- if(условие) {
- /* Блок кода, подлежащий выполнению, если условие истинно */
- } elseif(условие) {
- /* Блок кода, подлежащий выполнению, если первое условие ложно,
- а второе истинно */
- } else {
- /* Блок кода, подлежащий выполнению, если оба условия ложны */
- }

***Можно соединять вместе столько конструкций **elseif**, сколько понадобится.

Конструкция switch (переключатель/ выбор)

switch($variable) {
[case <константа>:]
/* код, выполняющийся, когда $variable равна 1 */
[break;] [continue;]
[case <константа>:]
/* код, выполняющийся, когда $variable равна 2 */
[break;] [continue;]
...другие случаи
[default:]
/*код, выполняющийся, если не было совпадения ни с одним из случаев*/
}

Сравнение конструкций IF и SWITCH

```
<?php
/* Метод с использованием оператора if */
if ($i == 0) echo 'Первый случай';
if ($i == 1) echo 'Второй случай';
?>
```

```
<?php
/* Тот же код с применением оператора switch */
switch($i) {
case 0:
echo 'Первый случай';
break;
case 1:
echo 'Второй случай';
break;
}
?>
```

Конструкция `switch` с использованием оператора

```php
<?php
  $a1 = 10;

  switch ($a1)
  {
    case ($a1>=1) && ($a1<=5):
    echo "число от 1 до 5 равно ".$a1;
    break;
    case ($a1>=6) && ($a1<=10):
      echo "число от 6 до 10 равно  $a1";
    break;
    default:
      echo "\$a1 = $a1";
  }
```

Циклические структуры

Оператор while

- while (условие) {
- /* Код для повторного выполнения, пока указанное условие истинно */

Или в однострочной форме:

- while (условие){ /* Код для повторного выполнения */
- }

ПРИМЕР: Написать сценарий, отображающий каждое число, которое делится на 3, в диапазоне от 1 до 300 и напечатать из них все нечетные

```php
<?php
$count = 1;
while($count <= 300) {
if (($count%3) == 0) {
echo "$count делится на 3!<BR>";
}
$count++;
}
?>
```

3 делится на 3!
6 делится на 3!
9 делится на 3!
12 делится на 3!
15 делится на 3!
18 делится на 3!
21 делится на 3!
24 делится на 3!
27 делится на 3!
30 делится на 3!
33 делится на 3!
36 делится на 3!
39 делится на 3!
42 делится на 3!
45 делится на 3!
48 делится на 3!
51 делится на 3!
54 делится на 3!
57 делится на 3!
60 делится на 3!
63 делится на 3!
66 делится на 3!
69 делится на 3!
72 делится на 3!
75 делится на 3!
78 делится на 3!
81 делится на 3!
84 делится на 3!
87 делится на 3!
90 делится на 3!
93 делится на 3!
96 делится на 3!
99 делится на 3!
102 делится на 3!
105 делится на 3!

- **do** {
- /* Исполняемый код */
- } **while**(условие);
- В отличие от **while**, оператор **do/while** всегда выполняет блок кода, минимум, один раз.

- **for** (инициализация;условие;постобработка)
- {
- /* Код, подлежащий выполнению, пока условие истинно */
- }
- **for** — применяется в случаях, когда нужна **переменная-счетчик**

```
<?php
for ($count = 1; $count <= 300; $count++) {
if (($count%3) == 0) {
echo "$count делится на 3!<BR>";
}
?>
```

Массив

Стандартный способ создания массива

```
/*Создание массива*/
$name[0] = "A";
$name[1] = "B";
$name[2] = "C";
$name[3] = "D";
$name[4] = "E";
/*Вывод значения ячейки массива с индексом 2 на экран*/
echo $name[2];
```

```
$name[] = "A";
$name[] = "B";
$name[] = "C";
$name[] = "D";
$name[] = "E";
```

Способ создания массива в PHP

- $name = array (0 => "A", 1 => "B", 2 => "C", 3 => "D", 4 => "E");
- $name = array("A", "B", "C", "D", "E");

Ассоциативный массив

- Ассоциативные массивы - разновидность массивов PHP.
- Если в простых массивах это были числовые индексы, то ассоциативных эти индексы т

Способы создания ассоциативного массива

```
$color["white"] = "белый";
$color["black"] = "черный";
$color["red"] = "красный";
$color["green"] = "зеленый";
$color["blue"] = "синий";
```

```
$color = array("white" => "белый", "black" =>
"черный", "red" => "красный", "green" =>
"зеленый", "blue" => "синий");
```

Синтаксис массивов

- **$variable[<key expr>] = <expr>;**
- **<key expr>** — это выражение, которое вычисляется как **строка** или любое неотрицательное **целое число**
- <expr> представляет собой выражение, значение которого **ассоциируется** с этим ключом

Цикл при работе с ассоциативными массивами

- foreach(<array> as $key => $value) {
- echo "Значение с индексом $key равно: $val
";
- /* Извлекаем только значения элементов и игнорируем ключи */
- }

```php
<?php
 $myarray =  array ("PHP"=>"Язык", "is" =>"Есть", "cool"=>"Это");
 foreach ( $myarray as $key => $val)
{
   echo "Значение с индексом <font color='red'> $key</font> равно: $val<br>";
}
  foreach ( $myarray as  $val)
{
   echo "Значение   равно:  <font color='#0066FF'>$val</font><br>";
}
?>
</body>
```

Значение с индексом PHP равно: Язык
Значение с индексом is равно: Есть
Значение с индексом cool равно: Это
Значение равно: Язык
Значение равно: Есть
Значение равно: Это

Многомерные массивы

- Многомерный массив – это массив, который содержит в себе еще один массив.

$massiv =

Стационарный ПК			Ноубук			Нетбук		
ОЗУ	HDD	ГЦ	ОЗУ	HDD	ГЦ	ОЗУ	HDD	ГЦ
4096	500	3	3072	320	2	2048	250	1,6

```
$massiv["Стационарный ПК"] = array ("ОЗУ" => "4096", "HDD" => "500", "ГЦ" => "3");
$massiv["Ноутбук"] = array ("ОЗУ" => "3072", "HDD" => "320", "ГЦ" => "2");
$massiv["Нетбук"] = array ("ОЗУ" => "2048", "HDD" => "250", "ГЦ" => "1,6");
```

Пример

```php
<?php
$massiv["Стационарный ПК"] = array ("ОЗУ" => "4096", "HDD" => "500", "ТЦ" => "3");
$massiv["Ноутбук"] = array ("ОЗУ" => "3072", "HDD" => "320", "ТЦ" => "2");
$massiv["Нетбук"] = array ("ОЗУ" => "2048", "HDD" => "250", "ТЦ" => "1,6");
  foreach($massiv as $brand => $mass)
{
foreach($mass  as  $inner_key => $value)
{
echo "[$brand][$inner_key] = $value<br>";
}
}
?>
```

[Стационарный ПК][ОЗУ] = 4096
[Стационарный ПК][HDD] = 500
[Стационарный ПК][ГЦ] = 3
[Ноутбук][ОЗУ] = 3072
[Ноутбук][HDD] = 320
[Ноутбук][ГЦ] = 2
[Нетбук][ОЗУ] = 2048
[Нетбук][HDD] = 250
[Нетбук][ГЦ] = 1,6

```php
<?php
$massiv["Стационарный ПК"] = array ("ОЗУ" => "8","HDD"=>"1000", "ГЦ"=>"8");
$massiv["Ноутбук"] = array ("ОЗУ" => "4072", "HDD" => "320", "ГЦ" => "2");
$massiv["Нетбук"] = array ("ОЗУ" => "2048", "HDD" => "250", "ГЦ" => "1,6");
echo "<table border=2 height=100 bordercolor='green'><tr>";
foreach($massiv as $brand => $mass)
{ echo "<td colspan=3 align=center>$brand </td>";
}
  echo"</tr><tr>";
 foreach($massiv as $brand => $mass)
 {
   foreach($mass  as  $key => $value)
   {
     echo "<td>$key</td>";
   }
 }   echo    "</tr><tr>";
 foreach($massiv as $brand => $mass)
{
   foreach($mass  as  $key => $value)
   {
     echo "<td>$value</td>";
   }
}
echo"</tr></table>";
?>
```

Стационарный ПК			Ноутбук			Нетбук		
ОЗУ	HDD	ГЦ	ОЗУ	HDD	ГЦ	ОЗУ	HDD	ГЦ
8	1000	8	4072	320	2	2048	250	1,6

Работа с файлами

Функции для работы с файлами

Открыть (Создать) файл:

- $fff=fopen($file_name, mode)
  - R открыть только для чтения (по умолчанию)
  - W- только для записи (удаляется содержимое)
  - A -для добавления записи

Чтение из файла:

```
$ppp="mas.txt"; $p= file($ppp);
foreach ($p as $stroca)
    {
        echo $stroca."<br>";
    }
```

Запись в файл:

- fwrite($fff, "Выражение")

Закрыть файл:

- fClose($fff);

Пример на использование файла

- Создание и добавление записей в

```
{
    $kkk=fopen("mas.txt","w");
    for ($i=1; $i<=10; $i++){
    $ppp=$i."элемент\r\n";
    fwrite($kkk,$ppp);
    }
    fclose($kkk);
```

Перевод каретки, перенос строки-\r\n

Пример на использование файла

- Добавление записи в файл

```
<html>

<head>
  <title></title>
</head>

<body>

<?php
   $ppp="massiv.txt";
   $mmm=Fopen($ppp,a);
   $k="Шестой элемент\n";
   $kk=Fwrite($mmm,$k);
   $c=FClose($mmm);
?>

</body>

</html>
```

Перевод каретки, перенос строки-\r\n

- Вывод содержимого файла

```
<html>

<head>
  <title></title>
</head>

<body>

<?php
  $massiv="massiv.txt" ;
  $q=file($massiv);
  foreach($q as $stroca)
  {
    echo $stroca."<br>";
    $r[]=$stroca;
  }
   echo count($r);
?>

</body>

</html>
```

Первый элемент
Второй элемент
Третий элемент
Четвертый элемент
Пятый элемент
Шестой элемент
6