Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283

LEARNING TOPICS

- What is PHP?
- How PHP code looks like?
- PHP Comments, Variables
- PHP Commands (echo, print)
- Operators in PHP
- PHP Concatenation, character Escaping
- Control Structures in PHP
- Looping Structures in PHP

What is PHP?

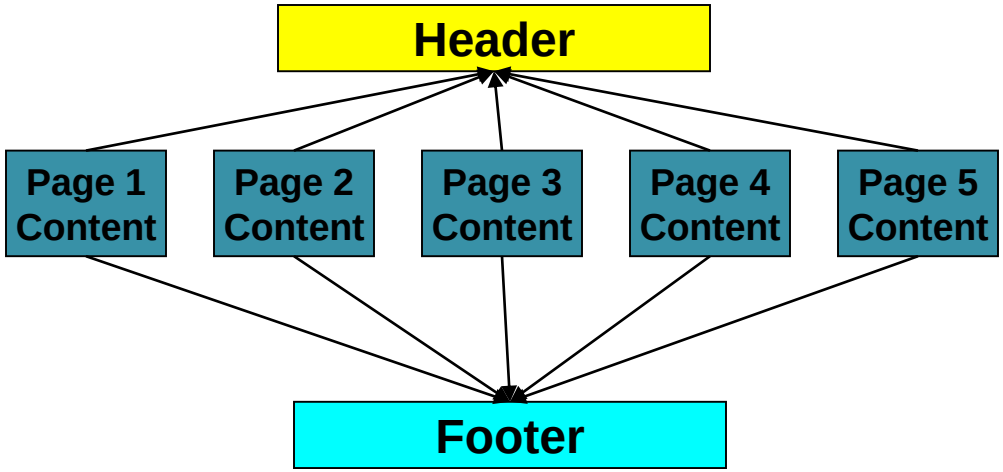
- PHP == 'PHP Hypertext Preprocessor'
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
- This allows the programmer to embed PHP scripts within HTML pages.

What is PHP (cont'd)

- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side
- Source-code not visible by client
  - 'View Source' in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

Advantages of PHP

- Any changes to header or footer only require editing of a single file.
- This reduces the amount of work necessary for site maintenance and redesign.
- Helps separate the content and design for easier maintenance



How PHP code looks?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php

echo"hello
world!";

?>
```

PHP Comments

- Standard C, C++, and shell comment symbols.

```
// C++ and Java-style comment

# Shell-style comments

/* C-style comments
     These can span multiple lines */
```

PHP Variables

- PHP variables must begin with a "$" sign
- Case-sensitive (`$Foo $foo $f0o`)
- Global and locally-scoped variables
  - Global variables can be used anywhere
  - Local variables restricted to a function or class
- Certain variable names reserved by PHP
  - Form variables (`$_POST, $_GET`)
  - Server variables (`$_SERVER`)
  - Etc.

Usage of Variables

```
<?php
$foo = 25;              // Numerical variable
$bar = "Hello";         // String variable

$foo = ($foo * 7);      // Multiplies foo by 7
$bar = ($bar * 7);      // Invalid expression
?>
```

Echo Command

- The PHP command '**`echo`**' is used to output the parameters passed to it
  - The typical usage for this is to send data to the client's web-browser
- Syntax
  - **`echo string arg`*`1`*` [, string arg`*`n`*`...]`**

Examples of Echo Command

```
<?php
$foo = 25;              // Numerical variable
$bar = "Hello";         // String variable

echo $bar;              // Outputs Hello
echo $foo,$bar;         // Outputs 25Hello
echo "5x5=",$foo;       // Outputs 5x5=25
echo "5x5=$foo";        // Outputs 5x5=25
echo '5x5=$foo';        // Outputs 5x5=$foo
?>
```

- **Notice** how `echo '5x5=$foo'` outputs `$foo` rather than replacing it with `25`
- Strings in single quotes (`' '`) are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as "\n" or "\\")

Operators

Arithmetic Operations

```
<?php
        $a=15;
        $b=30;
        $total=$a+$b;
        Print $total;
        Print "<p><h1>$total</h1>";
        // total is 45
?>
```

- $a - $b // subtraction
- $a * $b // multiplication
- $a / $b // division
- $a += 5 // $a = $a+5

Concatenation

- Use a period to join strings into one.

```
<?php
$string1="Hello";
$string2="PHP";
$string3=$string1 . " " . $string2;
Print $string3;
?>
```

```
Hello PHP
```

Character Escaping

❖If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
<?php
$heading="\"Computer Science\"";
Print $heading;
?>
```

```
"Computer Science"
```

Control Structures in PHP

```
if (expr) statement

<?php
    if ($a > $b) {
            echo "a is bigger than b";
            $b = $a;
    }
?>
<?php
    if ($a > $b) {
            echo "a is greater than b";
    } else {
            echo "a is NOT greater than
b";
    }
?>
```

PHP Control Structures

```
if ($link == 0) {
   echo 'The variable link is equal to 0';
}
else if (($link > 0) && ($link <= 5)) {
    echo 'The variable link is between 1
and 5';
}
else {
    echo 'The variable link is equal to
'.$link;
}
```

PHP Control Structures

```
<?php
    switch ($i) {
          case "apple":
                echo "i is apple";
                break;
          case "bar":
                echo "i is bar";
                break;
          case "cake":
                echo "i is cake";
                break;
          default:
                echo 'Enter correct option';

    }
?>
```

Loops

- `while (condition) {statements;}`

```
<?php
$count=0;
While($count<3)
{
        Print “hi PHP. ”;
        $count += 1;
        // $count = $count + 1;
        // or
        // $count++;
?>
```

```
hi PHP. hi PHP. hi PHP.
```

Loops

- `for (expr1; expr2; expr3)`
  `statement`
- ```
  <?php
      $i = 0;
      do {
          echo $i;
      } while ($i > 0);
  ?>
  ```
  ```

Loops

```
<?php
    $arr = array(1, 2, 3, 4);
    foreach ($arr as $value) {
        echo "$value \n";
    }
        foreach ($arr as &$value) {
        $value = $value * 2;
    }
    // $arr is now array(2, 4, 6, 8)
    unset($value);//break the
reference
?>
```

Loops

- `foreach ($arr as $key => $value) { echo "Key:$key; Value:$value<br />\n"; }`

- **`break`** ends execution of the current **`for`**, **`foreach`**, **`while`**, **`do-while`** or **`switch`** structure.

- **`continue`** is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration.

Arrays

- An array in PHP is actually an ordered map which maps values to keys.

- An array can be thought of in many ways:

- Linearly indexed array , list (vector), hash table (which is an implementation of a map), dictionary, collection, stack (LIFO), queue (FIFO)

Arrays

- Kinds of arrays:
  - numeric arrays.
  - associative arrays.
  - multi dimensional arrays.

Arrays

- In numeric arrays each key value corresponds to numeric values.
- They can be divided into two categories
  1. automatic numeric array index.
  2. manual array numeric index.

automatic numeric array index

```
<?php
   $x=array(1,2,3);
   print_r($x);
?>
```

o/p: array(0=>1,1=>2,2=>3)

Arrays

- **Manual array numeric index**

```
<?php
   $x[2]=10;    $x[3]=50;//$x=array(2=>10,3=>50);
   echo $x[2];    echo $x[3];
?>
```

- **Associative arrays**

In associated arrays each ID associated with its value

```
    <?php

$x=array("ab"=>1,"cd"=>2,"xy"=>3);
          print_r($x);
     ?>
```

Arrays

❖ Multidimensional Arrays-An array contains one or more arrays

```
<?php
   $z=array(array(10,20,30),array(40,50,
   60));
   print_r($z);
?>
```

```
Array ( [0] => Array ( [0] => 10 [1] => 20 [2] => 30 )
          [1] => Array ( [0] => 40 [1] => 50 [2] => 60 ) )
```

Arrays

```
<?php
     $x=array("ab"=>1,array(2,3,4),
"cd"=>8);
     print_r($x);
?>
```

Array (["ab"] => 1 [0] => Array ([0] => 2 [1] => 3 [2] => 4) ["cd"] => 8)

Arrays

```
<?php
      $x=array(3=>4,array(2,3,4),5
);
      print_r($x);
?>
```

Array ([3] => 4 [4] => Array ([0] => 2 [1] => 3 [2] => 4) [5] => 5)

Arrays

- Array operations
  - sort
  - ksort
  - rsort
  - krsort
  - array_merge
  - array_combine
  - array_intersect

Date Display

```
$datedisplay=date("yyyy/m/d"
);
print $datedisplay;
```

2009/4/1

```
$datedisplay=date("l, F J,
  Y");
print $datedisplay;
```

Wednesday, April 1, 2009

Month, Day & Date Format Symbols

M	Jan
F	January
m	01
n	1

Day of Month	d	01
Day of Month	J	1
Day of Week	l	Monday
Day of Week	D	Mon

Functions in PHP

- Functions MUST be defined before then can be called
- Function headers are of the format

```
function functionName($arg_1, $arg_2, …, $arg_n)
```

- Unlike variables, function names are not case sensitive

Functions example

```
<?php
        // This is a function
    function ag($arg_1, $arg_2)
     {
       $arg_2 = $arg_1 * $arg_2;
       return $arg_2;
    }

    $result_1 = ag(12, 3);  // Store the function
    echo $result_1;         // Outputs 36
    echo ag(12, 3);         // Outputs 36
?>
```

Include Files

- `include "header.php";`
- `include ("footer.php");`
- This inserts files; the code in files will be inserted into current code.
- **`require`** is identical to **`include`** except upon failure it will also produce a fatal **`E_COMPILE_ERROR`** level error. In other words, it will halt the script whereas **`include`** only emits a warning (**`E_WARNING`**) which allows the script to continue.

Include Files

- The **`include_once`** statement includes and evaluates the specified file during the execution of the script.

- This is a behavior similar to the **`include`** statement, with the only difference being that if the code from a file has already been included, it will not be included again.

- The **`require_once`** statement is identical to **`require`** except PHP will check if the file has already been included, and if so, not include (require) it again

PHP - Forms

```
<?php
        if ($_POST["submit"])
            echo "<h2>You clicked
    Submit!</h2>";
        else if ($_POST["cancel"])
            echo "<h2>You clicked
    Cancel!</h2>";
    ?>
```

```
    <form action="form.php" method="post">
        <input type="submit" name="submit"
    value="Submit">
        <input type="submit" name="cancel"
    value="Cancel">
    </form>
```

PHP – Forms (Contd..)

```
<?php
    …
    $term=$_REQUEST[“sterm”];
    …
?>
```

```
<form action="form.php" method="post">
    <input type=“text" name=“sterm"

    value=“<?= $term ?>">

</form>
```