Web Programming

YJ – 2016

PHP: Hypertext Preprocessor

PHP is a recursive acronym for "PHP: Hypertext Preprocessor".

PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

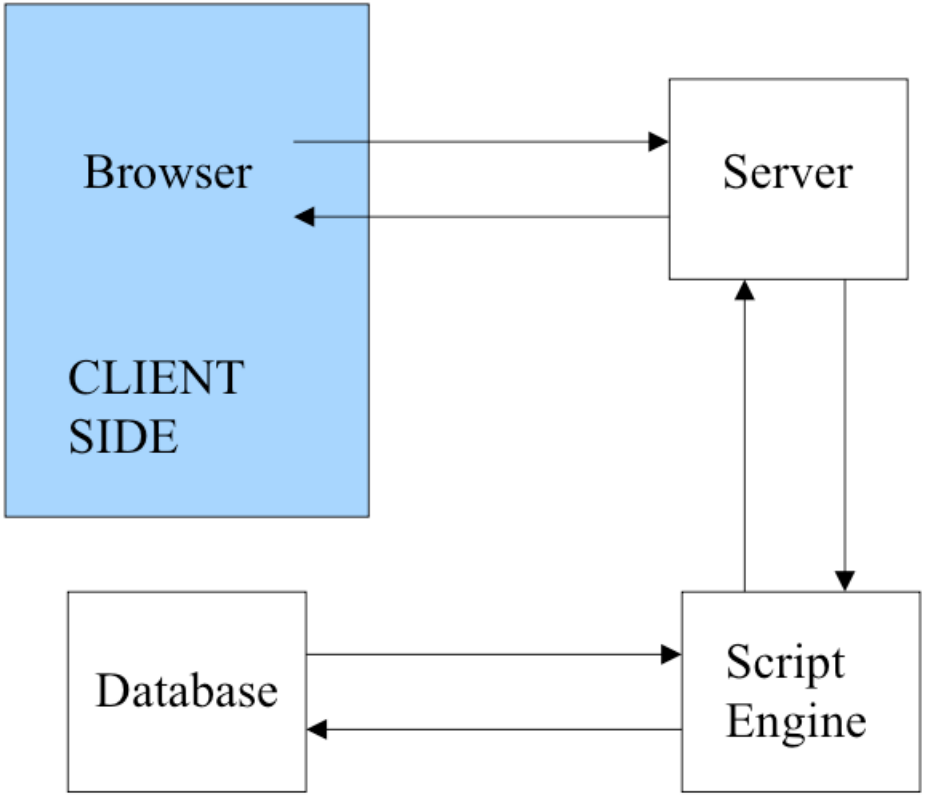
Features:

- Powerful and flexible.
- Easy to learn.
- C-like.
- Extremely portable.
- Cheap.
- Easy to set up.
- Works with lots of databases.
- Availability of source code (you can even contribute!).

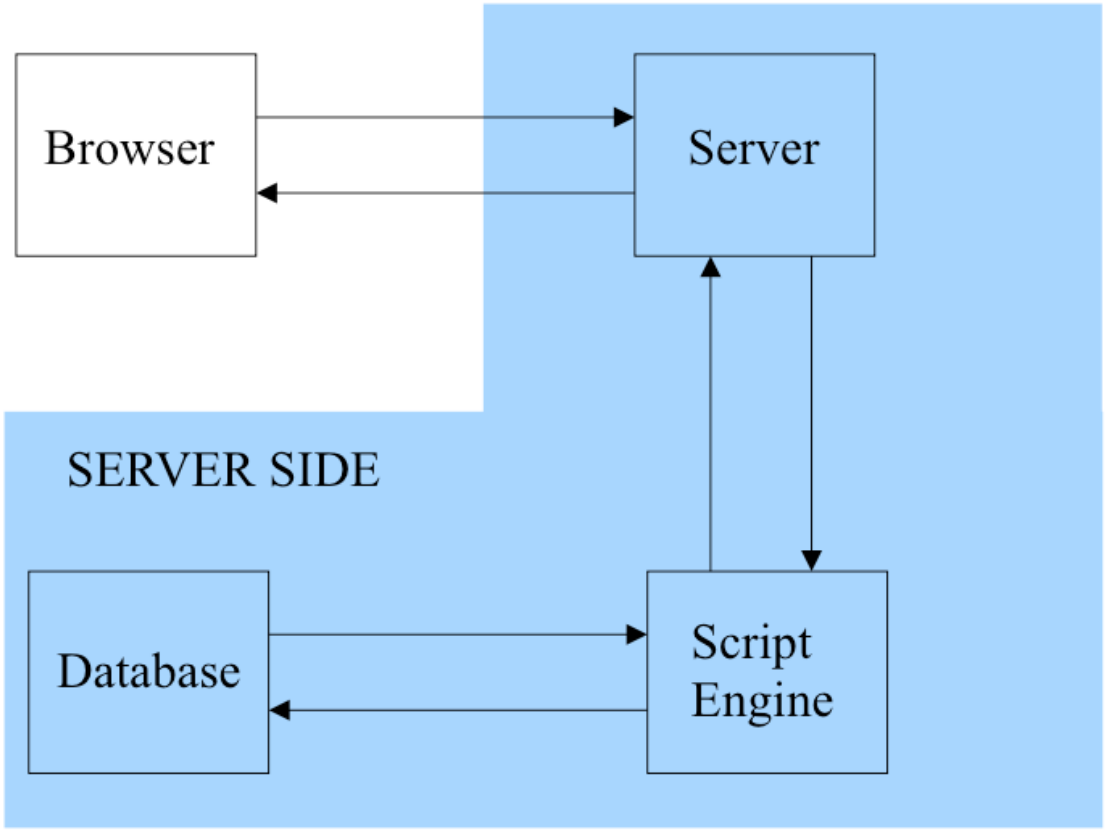
Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

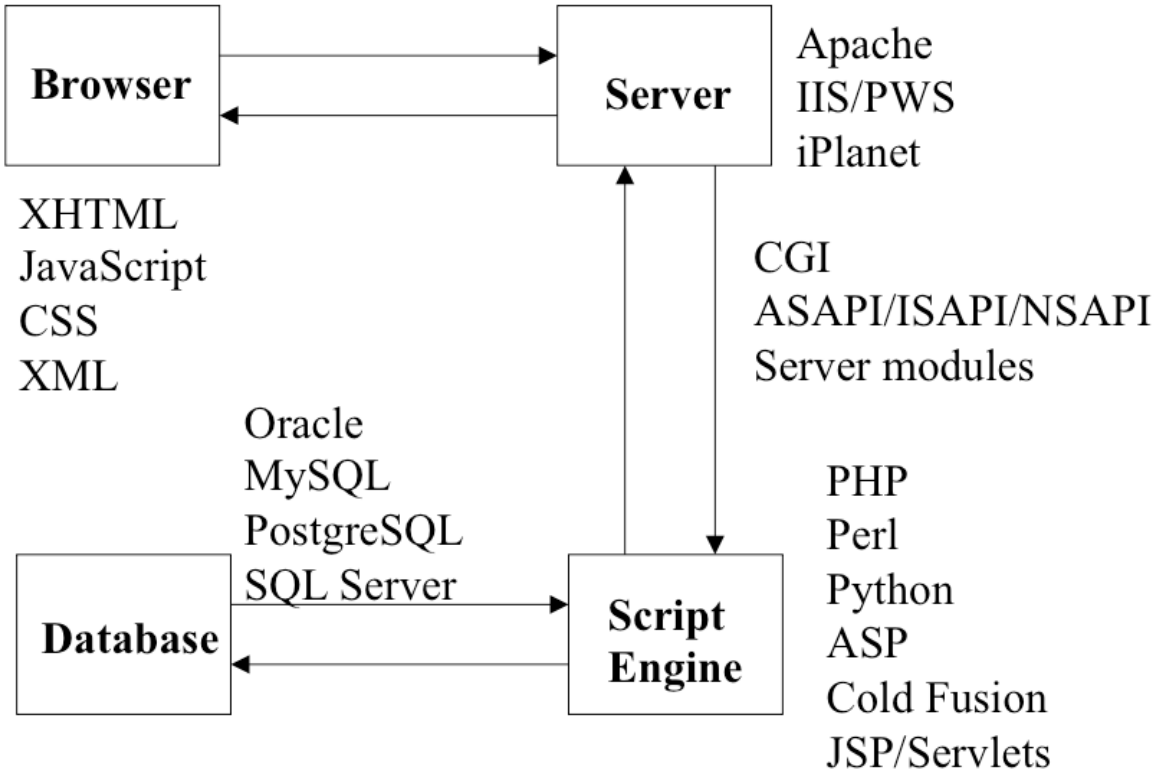
Client side



Server side



Technologies



Hello world

```
<html>
<head>
<title>Hello World</title>
</head>

<body>

<?php
	echo "Hello, World!";
?>

</body>
</html>
```

Variables

- **Integers:** are whole numbers, without a decimal point, like 4195.

```
$int_var = 12345; $another_int = -12345 + 12345;
```

- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.

```
$many = 2.2888800;
```

- **Booleans:** have only two possible values either true or false.

```
if (TRUE)
        print("This will always print<br>");
else
        print("This will never print<br>");
```

- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'

```
$literally = 'My $variable will not print!\n';
print($literally);
```

Variables

- **Arrays:** are named and indexed collections of other values.

```
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
```

- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.

```
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
```

- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

mssql_connect()

Variable Scope

- **Local variables**

  A variable declared in a function is considered local Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function

- **Function parameters**

  Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be

- **Global variables**

  A global variable can be accessed in any part of the program by placing the keyword **GLOBAL** in front of the variable

- **Static variables**

  A static variable will not lose its value when the function exits and will still hold that value should the function be called again

Constant

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script.

```
define("MINSIZE", 50);
echo MINSIZE;
echo constant("MINSIZE"); // same thing as the previous line
```

__LINE__	The current line number of the file.
__FILE__	The full path and filename of the file. If used inside an include,the name of the included file is returned.
__FUNCTION__	The function name. This constant returns the function name as it was declared (case-sensitive.
__CLASS__	The class name. This constant returns the class name as it was declared (case-sensitive).
__METHOD__	The class method name. The method name is returned as it was declared (case-sensitive).

Arithmetic Operators

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by denumerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assigne value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operators

```
$a = 10;
$b = 20;

/* If condition is true then assign a to result otheriwse b */

$result = ($a > $b ) ? $a :$b;

echo "TEST1 : Value of result is $result<br/>";

/* If condition is true then assign a to result otheriwse b */

$result = ($a < $b ) ? $a :$b;

echo "TEST2 : Value of result is $result<br/>";
```

Control

If … else

```php
<?php
$d = date("D");
if ($d=="Fri")
   echo "Have a nice weekend!";
elseif ($d=="Sun")
   echo "Have a nice Sunday!";
else
   echo "Have a nice day!";
?>
```

switch

```php
<?php
$d = date("D");
switch ($d)
{
   case "Mon":
      echo "Today is Monday";
      break;
   case "Tue":
      echo "Today is Tuesday";
      break;
   case "Wed":
      echo "Today is Wednesday";
      break;
   case "Thu":
      echo "Today is Thursday";
      break;
   case "Fri":
      echo "Today is Friday";
      break;
   case "Sat":
      echo "Today is Saturday";
      break;
   case "Sun":
      echo "Today is Sunday";
      break;
   default:
      echo "Wonder which day is this ?";
}?>
```

Loop

For loop

```
for (initialization;
condition; increment)
 {
      code to be executed;
}
```

While loop

```
while (condition)
{
 code to be executed;
}
```

foreach

```
foreach (array as value)
{
code to be executed;
}
```

Array

- An array is a data structure that stores one or more similar type of values in a single value.
- There are three different kind of arrays and each array value is accessed using an ID which is called array index.
  - **Numeric array** - An array with a numeric index. Values are stored and accessed in linear fashion
  - **Associative array** - An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
  - **Multidimensional array** - An array containing one or more arrays and values are accessed using multiple indices

String

- A string is a **set of zero or more characters**.
- In PHP you can use **single or double quotes** to describe a string.

```
$var = "The assignment was easy";
echo '$var'; //Output: $var
echo "$var"; //Output: Assignment was easy
```

- Strings in double quotes are evaluated while strings in single quotes are not.
- Sometimes you will see special characters in strings such as

  \n - newline
  \t - tab
  \s - a space
  \v - vertical tab

- Some String functions

  explode(), implode(), strcmp(), etc.

GET

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.

```php
<?php
if( $_GET["name"] || $_GET["age"] )
{
   echo "Welcome ". $_GET['name']. "<br />";
   echo "You are ". $_GET['age']. " years old.";
   exit();
}
?>
```

POST

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.

```php
<?php
if( $_POST["name"] || $_POST["age"] )
{
   echo "Welcome ". $_POST['name']. "<br />";
   echo "You are ". $_POST['age']. " years old.";
   exit();
}
?>
```

REQUEST

The PHP $_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

```php
<?php
if( $_REQUEST["name"] || $_REQUEST["age"] )
{
   echo "Welcome ". $_REQUEST['name']. "<br />";
   echo "You are ". $_REQUEST['age']. " years old.";
   exit();
}
?>
```

Inclusion

You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to included one PHP file into another PHP file.

- The include() Function

```
<?php
        include("xxmenu.php");
?>
<p>This is an example to show how to include wrong PHP file!</p>
```

- The require() Function

```
<?php
        require("xxmenu.php");
?>
<p>This is an example to show how to include wrong PHP file!</p>
```

Predefined Variables

Sr.No	Variable & Description
1	**$GLOBALS** Contains a reference to every variable which is currently available within the global scope of the script. The keys of this array are the names of the global variables.
2	**$_SERVER** This is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server. There is no guarantee that every web server will provide any of these. See next section for a complete list of all the SERVER variables.
3	**$_GET** An associative array of variables passed to the current script via the HTTP GET method.
4	**$_POST** An associative array of variables passed to the current script via the HTTP POST method.
5	**$_FILES** An associative array of items uploaded to the current script via the HTTP POST method.
6	**$_REQUEST** An associative array consisting of the contents of $_GET, $_POST, and $_COOKIE.
7	**$_COOKIE** An associative array of variables passed to the current script via HTTP cookies.
8	**$_SESSION** An associative array containing session variables available to the current script.
9	**$_PHP_SELF** A string containing PHP script file name in which it is called.