Quick review on Web-2

By: Ashuqullah Alizai
Herat University
Computer Science Faculty

Alizai.csf@hotmail.com

PHP introduction

- 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.
- Integrated number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP supports a large number of protocols such as POP3, IMAP, and LDAP
- PHP Syntax is C-Like.

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, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

- Five important characteristics make PHP's practical nature possible:
  1. Simplicity
  2. Efficiency
  3. Security
  4. Flexibility
  5. Familiarity

Escaping to PHP:

- PHP is embedded in HTML
- Canonical PHP tags:

  <?php...?>
- Short-open (SGML-style) tags:

  <?...?>
- ASP-style tags:

  <%...%>
- HTML script tags:

  <script language="PHP">...</script>

Commenting PHP Code:

- **Single-line comments:**
  - Shell way
    - # This is a single line comment,
  - C++ way
    - // This is a single line comment too
- **Multi-lines comments:**
  - C way

```
<?
 /* This is a comment with multiline Author : Mohammad
  Mohtashim Purpose: Multiline Comments Demo Subject: PHP */
 print "An example with multi line comments";
?>
```

Variables in PHP

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- In PHP variables automatically converting types from one to another when necessary.
- PHP variables are Perl-like.
- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but not characters like + , - , % , (,) . & , etc

Data types In PHP

- PHP has a total of eight data types which we use to construct our variables:
  - **Integers:** are whole numbers, without a decimal point, like 4195.
  - **Doubles:** are floating-point numbers, like 3.14159.
  - **Booleans:** have only two possible values either true or false.
  - **NULL:** is a special type that only has one value: NULL.
  - **Strings:** are sequences of characters
  - **Arrays:** are named and indexed collections of other values.
  - **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
  - **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

Operators in PHP:

- PHP contains three types of operators:
- **The One and Only Ternary Operator (?:)**
- One of the most elegant operators is the ?: (question mark) operator. Its format is truth_expr ? expr1 : expr2
  - If truth_expr is true assign expr1 and if it's false, assign expr2
- binary operators are used on two operands
  - **Numeric Operators(+,-,*,%,/,.)** All the binary operators (except for the concatenation operator) work only on numeric operands. If one or both of the operands are strings, Booleans, nulls, or resources, they are automatically converted to their numeric equivalents before the calculation is performed
  - **Concatenation Operator (.): concatenates** two strings. This operator works only on strings

Operators in PHP: binary operators

- **Assignment operators(=) enable you to write a value to a variable.**
- valid composite assignment operators: (+=, -=, *=, /=, %=, ^=, .=)
- **Comparison operators(==,!=,>,<,>=,<=) enable you to determine the relationship between** two operands. For these comparison operators, automatic type conversions are performed, if necessary.
- The two === and !== are the same as == and != but the types are not automatically converted while the type and the values are compared.
- **Logical Operators('&&, and', '||, or', 'xor')** first convert their operands to Boolean values and then perform the respective comparison.

Operators in PHP: Unary Operators

- Unary operators act on one operand.
- Negation operators(!) appear before their operand—for example, !$var
- Increment/Decrement Operators($var++,$var--,++$var, --$var)
- The Cast Operators

Operator	Changes Type To
(int), (integer)	Integer
(float), (real), (double)	Floating point
(string)	String
(bool), (boolean)	Boolean
(array)	Array
(object)	Object

- Example:

```
$str = "5";
$num = (int) $str;
```

CONTROL STRUCTURES

- They can be basically divided into two groups:
  - **conditional control structures:** affect the flow of the program and execute or skip certain code according to certain criteria.
  - PHP supports both the if and switch conditional control structures.

Statement

```
switch (expr){
      case expr:
            statement list
      case expr:
            statement list
      ...
      default:
            statement list
}
```

Statement

```
if (expr)
            statement
elseif (expr)
            statement
elseif (expr)
            statement
...
else
            statement
```

CONTROL STRUCTURES

- **loop control structures(for, while, do while):** execute certain code an arbitrary number of times according to specified criteria.
  - such as iterating over a database query result set
- **for loop statement:**

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

- **while loop statement:**

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

- **do...while loop statement:**

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

Loop Control structure

- **break** :Sometimes, you want to terminate the execution of a loop in the middle of an iteration. For this purpose, PHP provides the break statement.
  1. break*;* //the innermost loop is stopped
  2. break *expr;*//will break from the n innermost loops
- **Continue:** In other cases, you may want to stop the execution of a specific loop iteration and begin executing the next one. Complimentary to break, continue provides this functionality.
  1. continue*;* //*stops the execution of the innermost loop iteration and continues executing the next iteration of that loop*
  2. continue *expr;*//*can be used to stop execution of the n innermost loop iterations*

PHP Arrays

- An array is a data structure that stores one or more similar type of values in a single value, and can be created with **array()** function.
- There are three different kind of arrays
- **Numeric array** - An array with a numeric index. Values are stored and accessed in linear fashion and can be(String, numbers, and objects)
- **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.
  - each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

PHP Arrays Examples

- /* First method to create array. */
  - $numbers = array(1, 2, 3, 4, 5);
- /* Second method to create array. */
  - $numbers[0] = "one";
  - $numbers[1] = "two";
  - $numbers[2] = "three";
- /* First method to associate create array. */
  - $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);
- /* Second method to create array. */
  - $salaries['mohammad'] = "high";
  - $salaries['qadir'] = "medium";
  - $salaries['zara'] = "low";

PHP Arrays Examples

- /* First method to create multi dimentional array. */
  - $numbers = array(array(1, 2, 3, 4, 5),array(6,7,8,9,10));
- /* Second method to create array. */
  - $food["pizza"] = array("smal","mediam", "large");
  - $food["pasta"] = array("smal","mediam", "large");
  - $food["kabab"] = array("smal","mediam", "large");

Accessing array elements

- Numeric and associative array
  - echo $nameOfArray['index'] . "
";
  - //traversing array
  - foreach($ nameOfArray as $key=>$value) {
  - echo "Value is $value
";
  - }
- **Multidimensional Arrays**
- echo $NameOfArray['IndexOne']['IndexTwo'] . "
";
- //traversing multidimensional arrays

```
$food['Pizza']= array("small", "large", "mediam");
$food['passta']= array("small", "large", "mediam");
    foreach ($food as $key=>$value) {
     foreach($value as $foodType=> $valu){
          echo " $foodType: $valu $key <br/>";
} }
```

PHP GET and POST Methods

- There are two ways the browser client can send information to the web server.
  - The GET Method
  - The POST Method
- Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.
  - name1=value1&name2=value2&name3=value3

PHP GET method

- 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=value
- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- 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.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

PHP POST Method

- 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.
- The PHP provides **$_POST** associative array to access all the sent information using GET method.

The $_REQUEST variable

- The PHP $_REQUEST variable contains the contents of both $_GET, $_POST, and $_COOKIE.

```
<?php
  if( $_REQUEST["name"] || $_REQUEST["age"] )
  {
     echo "Welcome ". $_REQUEST['name']. "<br />";
     echo "You are ". $_REQUEST['age']. " years old.";
     exit();
  }
?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">

  Name: <input type="text" name="name" />
  Age: <input type="text" name="age" />

  <input type="submit" />
  </form>
</body>
</html>
```

Here $_PHP_SELF variable contains the name of self script in which it is being called.

PHP Form Handling

- Any form element in an HTML page will **automatically** be available to your PHP scripts.
- Example

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

When a user fills out this form and click on the submit button, the form data is sent to a PHP file, called "welcome.php":

welcome.php" looks like this

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br />
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

Output could be something like this:

```
Welcome John!
You are 28 years old.
```

PHP and MySQL

- PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database.
  - **Create a Connection to a MySQL Database**

Function: $con= mysqli_connect(servername,username,password,database);

  - **Closing a Connection**

Function: mysqli_close($con);

  - **Create a Database**

Function: mysqli_query("CREATE DATABASE my_db",$con)

  - **Connection to a database**

Function: mysqli_select_db("my_db", $con);

  - **Create a Table**

```
$sql = "CREATE TABLE Persons( FirstName varchar(15),
   LastName varchar(15) ,Age int)";
```

Function: mysql_query($sql,$con);

Insert Data Into a Database Table

- **Syntax:** t is possible to write the INSERT INTO statement in two forms.
  1. INSERT INTO table_name VALUES (value1, value2, value3,...)
  2. INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,...)
- **Practical**
  - mysqli_select_db("my_db", $con);
    mysqli_query("INSERT INTO Persons (FirstName, LastName, Age) VALUES ('Peter', 'Griffin', '35')");

Insert Data From a Form Into a Database

- When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php".
- The "insert.php" file connects to a database, and retrieves the values from the form with the PHP $_POST variables.

➢Then, the mysqli_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the HTML form:

```
<html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname" />
Lastname: <input type="text" name="lastname" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>

</body>
</html>
```

This is the end for this lecture