CSE479
Web Programming

Nishat Tasnim Niloy

Lecturer

Department of Computer Science and Engineering

Faculty of Science and Engineering

Topic 6

PHP Basic

PHP

- PHP is a server scripting language and is a powerful tool for making dynamic and interactive Web pages quickly.
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft’s ASP.
- Where is it used?
  - It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
  - It is deep enough to run the largest social network (Facebook)!
  - It is also easy enough to be a beginner’s first server-side language!

What can PHP do?

- PHP can generate dynamic page content.
- PHP can create, open, read, write, delete, and close files on the server.
- PHP can collect form data.
- PHP can send and receive cookies.
- PHP can add, delete, modify data in your database.
- PHP can restrict users to access some pages on your website.
- PHP can encrypt data.

Why use PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.).
- PHP is compatible with almost all servers used today (Apache, IIS, etc.).
- PHP supports a wide range of databases.
- PHP is free.
- PHP is easy to learn and runs efficiently on the server side.

PHP files and Syntax

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code.
- PHP code are executed on the server, and the result is returned to the browser as plain HTML.
- PHP files have extension “.php”.
- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>
- A PHP file normally contains HTML tags, and some PHP scripting code.

PHP files and Syntax

```
<html>
   <body>
      <h1>My first PHP page </h1>
     <?php
          echo "Hello World!";
      ?>
     </body>
</html>
```

PHP Comments

```
<html>
  <body>
    <h1>My first PHP page </h1>
     <?php
         // This is a single line comment
         # This is also a single line comment
        /*
       This is a multiple lines comment block
      that spans over more than one line
      */
     ?>
    </body>
</html>
```

PHP Case Sensitivity

In PHP all user-defined functions, classes, and keywords not case sensitive.

```
<html>
  <body>
      <h1>My first PHP page </h1>
      <?php
       ECHO "Hello World!<br>";
        echo "Hello World!<br>";
        EcHo "Hello World!<br>";
       ?>
   </body>
</html>
```

Variables

- However; in PHP, all variables are case-sensitive.

```
<html>
   <body>
     <h1>My first PHP page </h1>
     <?php
         $color="red";
         echo "My car is " . $color .     "<br>";
         echo "My house is " . $COLOR . "<br>";
         echo "My boat is " . $coLOR . "<br>";
       ?>
   </body>
</html>
```

Variables

- A variable starts with the $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and _).
- Variable names are case sensitive ($y and $Y are two different variables).

PHP Data Types

- PHP supports the following data types:
  - String
  - Integer
  - Float (floating point numbers (also called double))
  - Boolean
  - Array
  - Object
  - NULL
  - Resource

PHP Constants

- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- Unlike variables, constants are automatically global across the entire script.
- To set a constant, use the define() function - it takes three parameters:
  - The first parameter defines the name of the constant,
  - The second parameter defines the value of the constant
  - The optional third parameter specifies whether the constant name should be case-insensitive. Default is false.

PHP Operators

- Arithmetic: +,-,*,/,**,%
- Assignment: =, +=, -=, *=, /=, %=
- String: .(concatenation), .=
- Increment/decrement: ++ and – (post and pre)
- Relational: ==, ===, !=, !==, <, <=, >, >=, <>
- Logical: and, &&, or, ——, xor, !
- Array: +, ==, ===, !=, <>, !==

Conditional Statements and Loops

- Conditional Statements (branches)
  - ► if ...
  - ► if else
  - ► if elseif else

► switch Loops

- ► while - loops through a block of code as long as the specified condition is true.
- ► do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true.
- ► for - loops through a block of code a specified number of times.
- ► foreach - loops through a block of code for each element in an array.

PHP functions

- The real power of PHP comes from its functions; it has more than 1000 built-in functions.
- Besides the built-in PHP functions, we can create our own functions.
- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function.
- A user defined function declaration starts with the word “function”.

PHP Arrays

- An array can hold many values under a single name, and you can access the values by referring to an index number.
- In PHP, the array() function is used to create an array.
- In PHP, there are three types of arrays:
  - Indexed arrays - Arrays with a numeric index.
  - Associative arrays - Arrays with named keys.
  - Multidimensional arrays - Arrays containing one or more arrays.

PHP Superglobals

- Several predefined variables in PHP are ”superglobals”, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

- The PHP superglobal variables are:
    - $GLOBALS
    - $_SERVER
    - $_REQUEST
    - $_POST
    - $_GET
    - $_FILES
    - $_ENV
    - $_COOKIE
    - $_SESSION

Forms with PHP

- Form data is sent to the server when the user clicks Submit.
- The server can then use this data for various purposes (this is not validation).
- The PHP superglobals $_GET and $_POST are used to collect form-data.
- GET vs POST
  - Both GET and POST create an array (e.g. array(key =>value, key2 =>value2, key3 =>value3, ...)).
  - This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.
  - $_GET is an array of variables passed to the current script via the URL parameters.
  - $_POST is an array of variables passed to the current script via the HTTP POST method.

GET

- Information sent from a form with the GET method is visible to everyone (all variable names and values are displayed in the URL).
- GET also has limits on the amount of information to send. The limitation is about 2000 characters.
- However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.
- GET may be used for sending non-sensitive data.
- GET should NEVER be used for sending passwords or other sensitive information!

POST

- Information sent from a form with the POST method is invisible to others (all names/values are embedded within the body of the HTTP request) and has no limits on the amount of information to send.
- Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.
- However, because the variables are not displayed in the URL, it is not possible to bookmark the page.
- Developers prefer POST for sending form data.

Validation and Database Interaction

- PHP can be used to perform form validation as well.
- However, this validation is performed on the server, which might waste time and server resources.
- JavaScript is always preferred for client side validation.
- PHP 5 and later can work with a MySQL database using:
  - MySQLi extension (the “i” stands for improved)
  - PDO (PHP Data Objects)
- Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

MySQL vs PDO

- Both MySQLi and PDO have their advantages:
- PDO will work on 12 different database systems, where as MySQLi will only work with MySQL databases.
- So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.
- Both are object-oriented, but MySQLi also offers a procedural API. Both support Prepared Statements.
- Prepared Statements protect from SQL injection and are very important for web application security.

MySQL and PHP

- The following procedure has to be used to PHP/ MySQL interaction.
    - ► Open a connection.
    - ► Run SQL statements and process the returns (repeat how many ever times).
    - ► Close the connection.