Web Programming

by Fajar Pradana S.ST., M.Eng

01 – PHP Fundamentals and State

PHP

- A programming language devised by Rasmus Lerdorf in 1994 to build a dynamic and interactive web sites.
- Formerly named from Personal Home Page but changed to a recursively named:
  - PHP: Hypertext Preprocessor
- PHP programs (.php) are run on a server—specifically run on a Web server.
  - OR... you may also run it manually through the shell
- PHP often used as a middle-ware to other services on the internet (e.g accessing data on a database or generating documents on the fly)

Why PHP?

- Cross-platform:
  - Most PHP code can be processed without alteration on computers running many different operating systems.
  - For example, a PHP script that runs on Linux generally also runs well on Windows.
- HTML-embedded:
  - PHP code can be written in files containing a mixture of PHP instructions and HTML code.
  - C-based programming language
- Open source
  - You don’t have to pay in using PHP code to build dynamic websites.

System and Software Requirements

- To run PHP code you will need the following software:
  - A computer with an operating system such as Windows, Mac, or Linux
  - A PHP-compatible Web server software
    - Apache, Internet Information Server (IIS), or Nginx
  - PHP software
    - Can be downloaded from php.net
- For database environment
  - MySQL Database Server
    - Can be downloaded from http://mysql.com

System and Software Requirements (2)

- Optional development-related software
  - Any text editor, such as Notepad or Notepad++, Emacs, vi.
    - Or… you may use Adobe Dreamweaver IDE or any other PHP script editor with PHP syntax highlighting feature to aid you in fixing common syntax problems that you may encounter during development. *This will help you code PHP script pretty much easier*.
  - Web browsers
    - IE, Mozilla Firefox, Google Chrome, Opera
    - Browser with Firebug or Web Developer plugin installed is recommended.
  - Helpers script, PHPMyAdmin
    - PHP-based visual database management for MySQL.
  - PHP Manuals
    - Downloadable from PHP documentation downloads page:
    - http://php.net/download-docs.php

Warning!
Dragons Ahead

You may want to turn ON your PHP-enabled web server to test any of the following PHP scripts provided

PHP Syntax

1. A PHP script always starts with **<?php** and ends with **?>**
2. A PHP file must have a .php extension
3. A PHP file normally contains HTML tags, and some PHP scripting code
4. Each code line in PHP must end with a semicolon (;). The semicolon is a separator and is used to distinguish one set of instructions from another.
5. In PHP, we use // to make a one-line comment or /* and */ to make a comment block

Hello, World!

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php echo '<p>Hello World</p>'; ?>
  </body>
</html>
```

Variables

- Issues concerning creating variables:
  - Naming conventions
  - Data type
  - Scope

Variables Naming

- Variable names begin with a dollar sign (**$**).
- The first character after the dollar sign MUST be a letter or an underscore.
- The remaining characters in the name may be letters, numbers, or underscores without a fixed limit
- Variables are CASE SENSITIVE
  - treat two variables with the same name but with different case as two different variables
  - PHP has no command for declaring a variable.

```
<?php
  // both are two different variables
  $myVariable = 0;
  $myvariable = 1;
?>
```

PHP Data Types

Data type	Description
Boolean	Scalar; either True or False
Integer	Scalar; a whole number
Float	Scalar; a number which may have a decimal place
String	Scalar; a series of characters
Array	Compound; an ordered map (contains names mapped to values)
Object	Compound; a type that may contain properties and methods
Resource	Special; contains a reference to an external resource, such as a handler to an open file
NULL	Special; may only contain NULL as a value, meaning the variable; explicitly does not contain any value

Common PHP Operators

- Assignment
  - `=`
- Arithmetic
  - `+, -, /, *, %`
- Concatenation
  - `.`
- Negation
  - `!`
- Logic
  - `||, &&, >, <, ==, >=, <=, !=, ===, !===, and, or`
- Increment
  - `++, --`

Variable Scope

- Local Scope
  - Any variable used from inside function

```
<?php

function send_data() {
     $my_data = "Inside data";
     echo $my_data; // echoes $my_data value
}

// throws an error messages
echo $my_data;

?>
```

Variable Scope: Global

- Global Scope
  - Any variable used from outside a function

```
<?php

  $a = 1;
  $b = 2;

  function Sum()
  {
    global $a, $b;
     $b = $a + $b;
  }

  Sum();    // executing Sum() function
  echo $b; // will echo 3

?>
```

Super Global Arrays

Array	Description
$GLOBALS	Has a reference to every variable that has global scope in a PHP program. Many of the variables in it are also in other superglobal arrays
$_SERVER	Includes everything sent by server in the HTTP response, such as the name of the currently executing script, server name, version of HTTP, remote IP address, and so on. Although most Web server software produces the same server variables, not all do, and not all server variables necessarily have data in them
$_GET	Contains all the querystring variables that were attached to the URL, or produced as a result of using the GET method
$_POST	Contains all the submitted form variables and their data. You use variables from the $_POST or $_REQUEST arrays extensively in most of your PHP programs. For example, to make use of a username or password (or any other data) submitted as part of a form, you'll use PHP variables from the $_REQUEST array

Super Global arrays

Array	Description
$_COOKIE	Contains all cookies sent to the server by the browser. They are turned into variables you can read from this array, and you can write cookies to the user's browser using the setcookie() function. Cookies provide a means of identifying a user across page requests (or beyond, depending upon when the cookie expires) and are often used automatically in session handling
$_FILES	Contains any items uploaded to the server when the POST method is used. It's different from the $_POST array because it specifically contains items uploaded (such as an uploaded image file), not the contents of submitted form fields
$_ENV	Contains data about the environment the server and PHP are operating in, such as the computer name, operating system, and system drive
$_REQUEST	Contains the contents of the $_GET, $_POST, and $COOKIE arrays, all in one

Sending Variables: Request Method

- GET
  - Sending request variables through an URL as a Query String
- POST
  - Sending request variables through the POST body. Variable name and it's value will not be shown on the URL

Attributes in <form> Elements

- Action Attribute
  - Tells to server which page to go to

```
<form action="myprogram.php">
    ...
</form>
```

- Method Attribute
  - The method attribute controls the way that information is sent to the server.

```
<form action="myprogram.php" method="GET">
```

  - or

```
<form action="myprogram.php" method="POST">
```

GET Method

- Browser automatically appends the information to the URL when it sends the page request to the web server

Example:

```
<form action=“test.php" method="GET">
```

- If the form is submitted then the page will be redirected to:

```
http://www.domain.com/test.php?furryanimal=cat&spiky
animal=porcupine
```

Example Get Method

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

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

Note :

- When using method="get" in HTML forms, all variable names and values are displayed in the URL.
- This method should not be used when sending passwords or other sensitive information!

POST Method

- Information in the form is sent in the body of http request and doesn’t appear in the URL

```
<form action="myprogram.php" method="POST">
  <input name="email“ value="name@domain.com"
</form>
```

HTML Standard Form Input Fields

- Text Fields

```
<input type="text" name="text1" />
```

- Password Field

```
<input type="password" name ="pass" />
```

- Radio Buttons

```
<input type="radio" name="radio1" value="Men" />
<input type="radio" name="radio1" value="Women" />
```

- Checkboxes

```
<input type="checkbox" name="vehicle" value="Bike" />
```

- Submit Button

```
<input type="submit" value="Submit" />
```

- Hidden fields

```
<input type="hidden" name="product_id" value="122" />
```

PHP Form Handling

- Get Value

```
<html>
  <body>
    Welcome <?php echo $_GET["text1"]; ?>!<br />
    Your password is <?php echo $_GET["pass"]; ?>.
  </body>
</html>
```

- Post Value

```
<html>
  <body>
    Welcome <?php echo $_POST["text1"]; ?>!<br />
    Your password is <?php echo $_POST["pass"]; ?>.
  </body>
</html>
```

IF Statements

- In PHP we have the following conditional statements:
- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

The IF Statements

- **The if Statement**
- Use the if statement to execute some code only if a specified condition is true.
- **Syntax**
- if (*condition*) *code to be executed if condition is true;*

-
  ```
  <html>
  <body>
  <?php
  $d=date("D");
  if ($d==“Thu") echo "Have a nice weekend!";
  ?>
  </body>
  </html>
  ```

The if...else Statement

- Use the if....else statement to execute some code if a condition is true and another code if a condition is false.
- if (*condition*)
  code to be executed if condition is true;
  else
  code to be executed if condition is false;

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

The if...else Statement

- If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:
-
```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri")
  {
  echo "Hello!<br />";
  echo "Have a nice weekend!";
  echo "See you on Monday!";
  }
?>

</body>
</html>
```

The if...elseif....else Statement

- Use the if....elseif...else statement to select one of several blocks of code to be executed.
- if (*condition*)
  code to be executed if condition is true;
  elseif (*condition*)
  code to be executed if condition is true;
  else
  code to be executed if condition is false;

```
<html>
<body>
<?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!";
?>
</body>
</html>
```

The PHP Switch Statement

- Use the switch statement to select one of many blocks of code to be executed.
- switch (*n*)
  {
  case *label1:*
    code to be executed if n=label1;
    break;
  case *label2:*
    code to be executed if n=label2;
    break;
  default:
    code to be executed if n is different from both label1 and label2;
  }

```
<html>
<body>
<?php
$x=1;
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
</body>
</html>
```

PHP Loops

- In PHP, we have the following looping statements:
- **while** - loops through a block of code while a specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as a 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

The while Loop

- The while loop executes a block of code while a condition is true.

  Syntax
- while (*condition*)
  {
  code to be executed;
  }

```
<html>
<body>

<?php
$i=1;
while($i<=5)
  {
  echo "The number is " . $i . "<br />";
  $i++;
  }
?>

</body>
</html>
```

The do...while Statement

- The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

  Syntax

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

```
<html>
<body>

<?php
$i=1;
do
  {
  $i++;
  echo "The number is " . $i . "<br />";
  }
while ($i<=5);
?>

</body>
</html>
```

The for Loop

- The for loop is used when you know in advance how many times the script should run.

  Syntax
- for (*init; condition; increment*)
  {
  code to be executed;
  }
- Parameters:
- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the iteration)

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
  {
  echo "The number is " . $i . "<br />";
  }
?>

</body>
</html>
```

The foreach Loop

- The foreach loop is used to loop through arrays.

  Syntax
- foreach ($*array* as $*value*)
  {
  code to be executed;
  }
- For every loop iteration, the value of the current array element is assigned to $value (and the array pointer is moved by one) - so on the next loop iteration, you'll be looking at the next array value.

```
<html>
<body>

<?php
$x=array("one","two","three");
foreach ($x as $value)
  {
  echo $value . "<br />";
  }
?>

</body>
</html>
```

Functions

- Function example in PHP

```php
<?php

function sum($a, $b = 2) {
  // define function content here...
  $v = $a + $b + 1;
  // optionally put a return value
  return $v;
}

// calling the function
$x = sum(4);
echo $x; // will prints 7

?>
```

State and Session

- Questions about state:
  - How to keep facebook users keep logged in while browsing friends profiles or other pages?
  - How to keep your shopping cart entries while you are browsing another goods to add?
  - How to keep students previous question answers on an online student examination system?

- How do we keep user state?
  - Cookies
  - Session

COOKIES

- Cookie is often used to identify a user (or user’s session).
- Cookie is a small file that the server embeds on the user's computer.
- Variables stored on a cookie is read when users access a website who own those cookie.
- Web sites can usually only modify their own cookies.

COOKIES

- Sets cookies

```
setcookie(name, [value], [expire], [path], [domain]);

<?php
  setcookie("user", "Alex Porter", time()+3600);
?>
```

- Retrieves cookies

```
$_COOKIE["name of cookie"];

<?php
  echo $_COOKIE["user"];
?>
```

Session

- With session, users are allowed to store information on the server for later use (i.e username, shopping item, question answer, etc)
- Session information is stored temporarily on the server and will be deleted if it is destroyed or after the user has left the website for a specified time.
- Sessions work by creating a unique id (`PHPSESSID`) for each visitor and store variables based on this `PHPSESSID`.
- While variables contained in a session stored securely on the server, this `PHPSESSID` value is stored on the client computer as a cookie in order to be able to keep track with the client, if cookies are disabled, `PHPSESSID` value is stored in the URL as a query string.

Using Sessions

- Starting session

```
<?php session_start(); ?>
```

- Storing session

```
<?php
  session_start();
  $_SESSION['status']=1;
?>
```

- Retrieving a session variable

```
<?php
  session_start();
  echo “Status=“ . $_SESSION[‘status'];
?>
```

Using Sessions

- Removing one session variable

```
<?php
  session_start();
  if(isset($_SESSION['status']))
    unset($_SESSION['status']);
?>
```

- Destroying the whole user’s session

```
<?php
  session_destroy();
?>
```

TUGAS KELOMPOK

1. Di file PHP yang pertama (a.php), buatlah sebuah form yang bisa menerima sebuah nilai masukan berupa angka, dimana nilai masukan tersebut akan dikirimkan ke file kedua (b.php) untuk diproses lebih lanjut.
2. Di file PHP kedua (b.php), hitung nilai faktorial dari angka yang dimasukkan dengan membuat sebuah fungsi. Tampilkan nilai angka yang dimasukkan melalui form tersebut beserta nilai faktorialnya. Selanjutnya, buat sebuah variabel array yang berisi nilai angka yang dimasukkan, hasil kalkulasi nilai faktorial, NIM, dan nama Anda. Simpan variabel array tersebut dalam variabel session. Buat sebuah link di halaman b.php yang mengarah ke file ketiga (c.php).
3. Di file yang ketiga (c.php), tampilkan nilai variabel yang disimpan dalam session. Kemudian hapus seluruh variabel yang disimpan dalam session tersebut.