INTERNATIONAL UNIVERSITY OF JAPAN
Public Management and Policy Analysis Program
Graduate School of International Relations

ADC5031 (2 Credits)
Public Management Information Systems
Spring 2019

Start of PHP

PHP Hypertext Preprocessor (PHP) is a general purpose computer language as well as a server-side HTML embedded script language that is used to develop Web applications as CGI programs.[1] Other Web application languages include PERL (Practical Extraction and Report Language), Python (http://www.python.org/), Adobe ColdFusion, Ruby, Java, JSP (JavaServer Pages), C++, and Microsoft Visual Basic and ASP (Active Server Pages).

0. PHP Basics

A PHP file is an ASCII text file to be "interpreted" (as opposed to "compiled" to generate an object file) by PHP processor and then executed. You may use a text editor such as nano (revised pico), Emacs, TexShop, or Mac Tex.

Once you write a PHP script, you may run it at the UNIX prompt by typing,

```
$ php dgGo.php
```

The PHP interpreter in Ubuntu is `/usr/bin/php5` that is linked as `/usr/bin/php`. If you want to execute short PHP scripts, use the `-r` option to run them enclosed by quote at the UNIX prompt. If you want to pass arguments to the PHP script, use the `-e` option. You may list multiple instructions separated by semi-colon (;).

```
$ php -r 'phpinfo(); echo "\n PHP information.\n";'
```

The PHP online manual is available at http://www.php.net/manual/en/ and the function section is at http://www.php.net/manual/en/language.functions.php.

1. Glance at a PHP Script

A PHP file may or may not be combined with HTML/XHTML. If a PHP script produces a series of well-arranged HTML tags, a Web browser can interpret it properly.

Let us take a look at the following example. You will be able to learn,

- PHP scripts need to begin with `<?php` and end with `?>`
- A PHP instruction ends with semi-colon (;)
- PHP is a general purpose computer language.
- Variable names begin with `$` (e.g., `$total`).
- Assignment operator is `=`.

[1] This document is largely based on online manual http://www.php.net/manual/en/index.php and its language reference http://www.php.net/manual/en/langref.php.

- Other arithmetic and relational operators are similar to those of C and Java.
- A comment is enclosed by `/*` and `*/` or follows //

PHP Script without HTML/XHTML

```
<?php

/* Calculate the sum of 1-100 */

$total = 0; // initial value

for ($i=1; $i<=100; $i++) {
     $total = $total + $i;
}

echo "The result is $total\n";

?>
```

PHP Script with HTML/XHTML

```
<html>

<head>
<title>PHP Script with HTML/XHTML</title>
</head>

<body>

<?php
echo "<h3>Calculate the sum of 1-100</h3>";

$total = 0; // initial value

for ($i=1; $i<=100; $i++) {
     $total = $total + $i;
}

echo "<p>The result is $total</p>";

?>

</body>
</html>
```

2. Variable and Array

A *variable* stores a numeric or string value, while an *array* can have a set of related numeric or string values. PHP supports integer, real number (floating point), string, boolean, array, and object. Their names begin with `$`.

```
$credit = 2;

$height = 5.6;

$student = "James";

$yes = FALSE;

$year = array(2010, 2011, 2012, 2013);

$color = array("red", "yellow", "blue");

$std_year = array(1=>'Freshman', 2=>'Sophomore', 3=>'Junior', 4=>'Senior');

$iuj = new dgGo();
```

3. Operators and Functions

PHP's operators and functions are very similar to those of C and Java.

3.1 Operators

PHP supports arithmetic, logical, relative, and other operators.

- Arithmetic operators: +, -, *, /, and ^ (power), %.
- Logical operators: && (and), || (or), ! (not).
- Relational operators: <, <=, >, >=, == (equal), and <> or != (not equal)
- Incrementing: ++, -- (e.g., $i++, $--, ++$i, --$i)
- Concatenation: period(.) (e.g., "IUJ"."PMPP")

3.2 Functions

PHP supports various functions.

String Function	Math Functions	Date Functions
▪ `substr(),strpos ()`	▪ `abs()`	▪ `date()`
▪ `strlen(), strops()`	▪ `sin(), cos(), tan()`	▪ `time()`
▪ `str_repeat(), locate()`	▪ `ceil(), floor(), round()`	▪ `gmdate()`
▪ `explode()`	▪ `exp(), log (), sqrt()`	▪ `getdate()`
▪ `trim(), ltrim()`	▪ `max(), min()`	▪ `localtime()`
▪ `is_string()`	▪ `is_int()`	▪ `mktime()`

Some examples of PHP functions are,

```
echo substr("ABCDEF", 0, 2 );         // AB

echo substr("ABCDEF");                // 6

echo str_repeat("=", 10);             // ==========

echo strpos("ABCDEFB", "B", 2);       // 6 (relative position from the first "B")

$record = explode('\n', $stream);
```

The `echo` command displays output of strings. `explode()` splits a string (a stream of data) into an array using the delimiter specified. The example above splits the data stream `$stream` and stores each line into an element of an array `$record`.

4. Outputting Numbers and Strings

PHP has several ways to output numbers and strings.

4.1 `echo` or `print`

`echo` outputs numbers and/or strings, while `print` outputs strings.

```
echo 'Welcome to PHP';

echo("Welcome to PHP");

print "Welcome to PHP";

echo $total;

echo 'The sum of '.$total;  // Using concatenation

echo 'The sum of 1 through 100 is $total. /n';
```

4.2 `printf`

`printf` outputs a formatted string. `%f` and `%s` respectively indicate a real number and a string, where `%10s` means a string variable ten characters long.

```
printf("The interest rate of last was %f", $rate);

printf("The interest rate of %s last year was %f", $country, $rate);

printf("Today's GRE word is %10s.", $gre_word);
```

4.3 Here Document

Here document syntax can output multiple lines. This syntax is very useful when outputting long HTML/XHTML codes in PHP. It will be irritating and messy to output individual lines without this syntax (e.g., `echo '<a href="#week1">Week 1</a> &#183;';…`)

```
echo <<<OUTPUT
<p style="text-indent:1em;background-color:#dfdfdf; border-top: 1px solid #c7b6a9;">
<a href="#week1">Week 1</a> &#183;
<a href="#week2">Week 2</a> &#183;
<a href="#week3">Week 3</a> &#183;
<a href="#week4">Week 4</a> &#183;
<a href="#week5">Week 5</a> &#183;
<a href="#week6">Week 6</a> &#183;
<a href="#week7">Week 7</a> &#183;
<a href="#week8">Week 8</a> &#183;
<a href="#week9">Week 9</a> &#183;
<a href="#week10">Week 10</a>
</p>
OUTPUT;
```

Since this `echo` command above is treated as a single line, you may not omit the semi-colon at the end of the sentence (i.e., `OUTPUT;`). Alternatively, you may use the following approach (defining the string as a variable).

```
$output = <<<OUTPUT
<p style="text-indent:1em;background-color:#dfdfdf; border-top: 1px solid #c7b6a9;">
<a href="#week1">Week 1</a> &#183;
<a href="#week2">Week 2</a> &#183;
<a href="#week3">Week 3</a> &#183;
<a href="#week4">Week 4</a> &#183;
<a href="#week5">Week 5</a> &#183;
<a href="#week6">Week 6</a> &#183;
<a href="#week7">Week 7</a> &#183;
<a href="#week8">Week 8</a> &#183;
<a href="#week9">Week 9</a> &#183;
<a href="#week10">Week 10</a>
</p>
OUTPUT;

echo $output;
```

5. Control Structure

Like other computer languages, PHP supports various control structures such as if, while, for, and switch.

5.1 if (condition) { … }

If a condition specified is true, then execute command(s) enclosed by { and }. Otherwise, the program skips them. The following *if* statement evaluates the value of a variable `score`; if the value is greater than 95, then assign “A” to a variable `grade`; otherwise, do nothing.

```
if ($score > 95) {

        $grade = “A”;

}
```

5.2 if (condition) { … } else { … }

If a condition specified is true, then execute command(s) enclosed by the first set of { and }. Otherwise, consider another condition specified, decide whether or not to execute command(s) in the second set of { and }. Then keep repeating the similar job until the `else` clause.

```
if ($score > 95) {

        $grade = "A";

} elseif ($score >90) {

        $grade = "A-";
…

} else {

        $grade = "F";
}
```

5.3 while (condition) { … }

The while statement executes a set of commands within { and } as long as the condition specified is satisfied. The following example calculates the sum of 1 through 100.

```
while ($i <=100 ) {

        $total = $total + $i;
        $i++;

}
```

5.4 for (condition) { … }

Like `while`, the `for` statement executes a set of commands with { and } as long as the condition is met. The condition below says, "Repeat the set of commands while the index variable `i` increases from 1 through 100 with a step of 1. Therefore, the following example returns the same result as the `while` example above.

```
for ($i=1; $i<=100; $i++) {

        $total = $total + $i;

}
```

5.5 switch (…) {case …; case …; …}

The `switch` statement evaluates the value of a variable and then executes a set of commands depending on the value. The following example evaluates the value of a variable `grade`; if "A" (string A), assign 4.0 to a variable `gpa` and then escape `switch` to execute next commands after ; if "A-", assign 3.75 to `gpa` and then escape `switch` to execute next commands after `switch`; … If a variable to be evaluated is numeric, quotes are not needed.

```
switch ($grade) {

        case "A": $gpa = 4.0; break;
        case "A-": $gpe = 3.75; break;
            …
}
```

5.6 foreach ($array as $value) { … }

The `foreach` executes a set of commands within { and } for all elements in an array. The following example create an array `$_phone` and assigns 1, 4, 2, 4 ($_phone[0]=1, $_phone[1]=4, $_phone[2]=2, and $_phone[3]=4) and prints an index (`$key`) and the value of the index (`$value`) for all elements (index from 0 through 3).

```
$_phone = array(1, 4, 2, 4);

foreach ($_phone as $key => $value ) {

       echo "$key: $value \n";
}
```

* `\n` is an escape sequence of carriage return to have a new line.

The result will be,

```
0: 1
1: 4
2: 2
3: 4
```

When you run a query on a database, you will get a data stream (as opposed to a well organized data set in a table format) as a result. In order to go through record by record in MySql, use `mysql_fetch_array()`, which breaks the data stream into individual records and store them into an array.

```
$records = mysql_query("SELECT * FROM student", $db_conn);

while($list = mysql_fetch_array($records) ) {

       echo "$list[name] $list[program] \n";

       …

}
```

The above example retrieves all attributes from a table `student` and store such data stream into a variable `$records`, which looks like "1B3039 Kohistani, Muzhdah PMPP IUJ 1… 1B3069 Purwo Wiyatmanto PMPP IUJ 1 … " When you try to print `$records`, you will get such series of data not in a table format; yes, it is difficult to know which one is which.

The `mysql_fetch_array()` arranges the data stream and saves it into an array `$list`. The `while` structure repeats the set of commands within { and } from the first record and the last one. If the first record is of Muzhdah and the second is of Pur, the result will look like,

```
Kohistani, Muzhdah PMPP
Purwo Wiyatmanto PMPP
…
```

This usage is frequently used in LAMP (Linux, Apache, MySql, and PHP). Therefore, you must be familiar with this type of loop.

6. File Handling

This section explains how to read and write a file in PHP.

6.1 Reading Files

The `fopen()` and `fclose()`commands open and close a file. The `fread()` and `fwrite()` read and write contents.

```
$file = “text.txt”;

$fo = fopen($file, ‘a+’);

$content = fread( $fo, filesize($file) );

fclose($fo);

$record = explode(“\t”, $content);
```

6.2 Writing Files

Available are such options as “r” (open a file for read only), “w” (open a file for write only), “a” (open a file for write only), “x” (creates a new file for write only), and “+” (read and write). For instance, “a+” opens a file for read and write.

```
$file = “text.txt”;

$fo = fopen($file, ‘w’);

fwrite($fo, $content);

fclose($fo);
```

6.3 Including Modules

The `require` and `include` commands include and evaluate a specified file. When error occurs, `require` will halt the script, while `include` will leave a warning message and continue.

```
require ‘header.php’;

include ‘menu.php’;
```

7. Function and Class

A *function* is a module or independent subroutine to be called from a program. A *class* in an object-oriented programming (OOP) is a blueprint of objects and have both data and methods.

7.1 Function

A function may be passed over some arguments from the main program and may return some values. The function is called by its name and arguments such as `count_page(“db connection”, “sql”);`.

```
function count_page($db_conn, $sql) {

        global $table, $perPage;
        $query = mysql_query("SELECT COUNT(*) FROM $table ".$sql, $db_conn);
        $count[total] = mysql_result($query, 0, "COUNT(*)");
        $count[page] = ceil($count[total] / $perPage);

return $count;
}
```

7.2 Class

A class is a blueprint of an object that includes data (properties) and methods (functions).

```
class dgGo {

public $CONST_referer   = "http://www.dgGo.org";
public $CONST_userAgent = "Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 5.1)";
public $CONST_cookie    = "user=dgGo; activity=evaluating";
public $CONST_timeout   = 10;

function __construct() {

        $this->curl = curl_init();

} // function __construct()


function __destruct() {

       curl_close($this->curl);

} // function __destruct()


public function get_robots() {

global $url;

       if (@file_get_contents($url."robots.txt") ) {

              $robots = file_get_contents($url."robots.txt");
              echo "Robots: \n $robots \n";
              $this->update_db("robots", $robots);

       } else {

               echo "No robots found. \n\n";

       } // if @file_get_contents

       echo "ROBOTS ".str_repeat("_", 80)."\n\n";

} // function get_robots()

…

} // class dgGo
```

```
class full_site extends dgGo {

…

}
```

An object is an instantiated class that inherits properties and functions from the class.

```
$dgSite = new dgGo();          // instantiate an object

$dgSite->get_robots();         // check and obtain robots.txt
```

8. Working with MySql

You need to first connect the database server and the send SQL to the server. PHP has database related functions that are customized to a specific database server such as Oracle, DBII, MySql, Microsoft SQL Server.

8.1 Connecting to MySql

The following PHP script is to connect a database using the objected oriented preprogramming style.

```
<?php

define("SQL_HOST", "localhost");
define("SQL_USER", "jeeshim");
define("SQL_PASS", "…");
define("SQL_DB", "jeeshim");


$db_conn = new mysqli(SQL_HOST, SQL_USER, SQL_PASS, SQL_DB);

if ($db_conn->connect_errno) {

     echo "Could not connect to MySQL. The error is " . $db_conn->connect_error;

}

?>
```

Alternatively, you may use a traditional procedural style to access a database. The `mysql_connect()` command connects to MySql server using information about host computer, MySql user name, and its password. `mysql_select_db()` selects a database using the database connection established. One you finish using database, run `mysql_close()` to terminate MySql connection.

```
define('SQL_HOST', 'localhost');                    // host name
define('SQL_USER', 'jeeshim');                      // DB account (user)
define('SQL_PASS', '…');                            // DB account's password
define('SQL_DB', 'jeeshim');                        // Database name

$db_conn = mysql_connect(SQL_HOST, SQL_USER, SQL_PASS)

     or die('Could not connect to MySql database' . mysql_error() );

mysql_select_db(SQL_DB, $db_conn);

...

mysql_close($db_conn);
```

8.2 Running Queries

Your SQL statement is executed by `query()` in the objected oriented programming style. The `fetch_arrya()` function fetches the data stream.

```
<?php

require "db_config.php";

$sql = "SELECT * FROM firm WHERE employee > 10";

$record = $db_conn->query($sql); // to run a query

echo "The list of IT firms \n";

while($list = $record->fetch_array(MYSQLI_BOTH )) {

        echo "$list[firm] $list[employee] \n";

}

$record ->free(); // to clear query result

$db_conn->close(); // to close db connection

?>
```

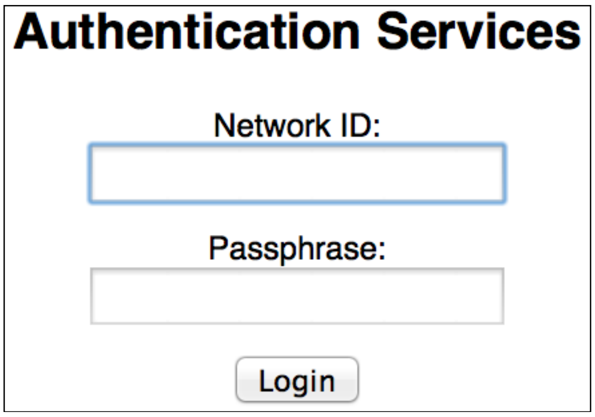
Alternatively, you may use `mysql_query()` in the traditional style. The `mysql_num_rows()` returns the number of records, while `mysql_fetch_array()` fetches the data stream. Compare the usage of `mysql_fetch_array()` with in 5.6.

```
$sql = 'SELECT * FROM firm WHERE employee > 10 ';

$db = mysql_query($sql, $db_conn) or die('Query failure');

$record_num = mysql_num_rows($db);           // to get the total number of records

while ( list($firm, $sic) = mysql_fetch_array($db) ) {

     echo "$firm $sic \n";

}
```

9 Receiving Contents Passed by POST and GET

Once a Web form pass over variables to the CGI program (a PHP script) specified, the program has to receive the contents using `$_POST[]`, `$_GET[]`, or `$_REQUEST[]`. `$_REQUEST[]` contains the contents of `$_POST[]`,`$_GET []`, and `$_COOKIE[]`.

Suppose a Web form (index.php) in a HTML/XHTML file passes over two variables `id` and `pw` to CGI program `login.php`.



```
<?php

echo <<<HTML
<html>
<head>
<title>Login</title>
</head>

<body>

<h3>Authentication Services</h3>
<form method="POST" action="login.php">

<p>
<label for="network_id" class="form">Network ID:</label><br />
<input type="text" name="id" id="network_id" />
</p>

<p>
<label for="passphrase" class="form">Passphrase: </label><br />
<input type="password" name="pass" id="passphrase" />
</p>

<input type="submit" value="Login" />
</form>

</body></html>
HTML;

?>
```

The information (say, jeeshim and jeeshim2017) that a user type in is sent to the CGI program as follows.

```
http:// …. /login.php?id=jeeshim&pass=jeeshim2018
```

Then `login.php` will receive the values of two variables using either `$_GET[]` or `$_REQUEST[]` because the method used in the Web form is GET as opposed to POST.

The following PHP script (login.php) receives two values from the Web form above. The function `isset()` checks if the variable is set or not. The following script says that when both values are set (user typed in some values in both variables), the script prints the network id.

```
<?php

echo <<<HTML
<html>
<head>
<title>Login result</title>
</head>

<body>
<h3>Hello</h3>
HTML;


if (isset($_REQUEST[id]) && isset($_REQUEST[pass]) ) {
        $network_id = $_REQUEST[id];
        $passwd = $_REQUEST[pass];
        echo "<p>Network ID is $network_id and its password is $passwd</p>";

} else {

        echo "<h3>Incorrect network ID or wrong password.</h3>";

}

echo <<<HTML2
</body>
</html>
HTML2;

?>
```

The value of id (i.e., jeeshim) passed from the Web page is received by `$_REQUEST[id]` and then copied into a variable `$network_id`. Similarly, the value of password (i.e., jeeshim2018) is received by `$_REQUEST["pass"]`.

End of this document.