PHP & Database Connectivity

CMPUT 410: Group Presentation Report

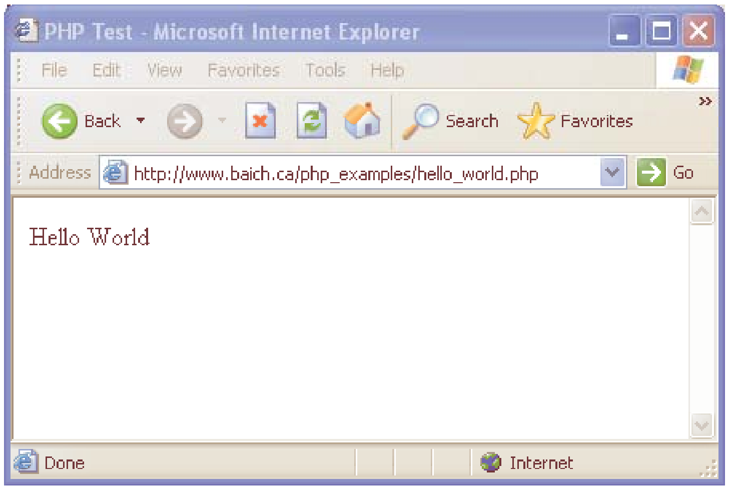
Submitted To :	Dr. Osmar Zaiane
Submitted By:	Leah Denney
	Theresa Baich
	Todd James
	Vinoth Sabanadesan
	Tim Yuan

Table of Contents

Section 1 - What is PHP???

PHP is a FREE server side scripting language for creating dynamic web pages. Most operating systems will support PHP, and most web servers will support it. Once feature of PHP is that it can be embedded into HTML. To embed a php script in a html file you can simply place it in <?php *code goes here* ?>. See example below

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


PHP Test - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.baich.ca/php_examples/hello_world.php Go
Done Internet

Hello World

PHP is a scripting language that borrows its syntax from C, PERL and JAVA. It can be programmed in various styles from procedure to object oriented programming. It runs on nearly every web server and operates with very minimal changes required to the PHP code. It also uses ODBC and has native drivers for MySql, Oracle, Postgres which take advantage of each database's unique features.

PHP History

PHP originated in 1995 by Rasmus Lerdorf, and was initially a simple set of perl scripts. Later, a C implementation with added functionality such as database connectivity and simple dynamic web applications gave PHP1 and PHP2 it's own cult following. However it was not until 1997 that PHP truly found its legs. Zeev Suraski and Andi Gutmans, worked together with Rasmus Lerdorf to create PHP 3. PHP 3 was very successful for the following reasons:

-A solid infrastructure for connecting to a variety of different databases, protocols, and APIs.

-An extensibility feature that attracted developers to add their own extension modules.

-Object oriented Syntax support and more consistent language syntax.

In May of 2000 PHP 4 was born and offered a complete rewrite of PHP's core, now known as the Zend engine. It improved the performance of complex applications and improved the modularity of PHP's code base. In addition to improved performance, it had support for many more web servers, HTTP sessions, output buffering, a more secure ways of handling user input and several new language constructs. PHP 5, released recently, offers another significant performance improvement over php 4 with the new ZEND 2 engine. Although PHP 4 was powerful, it still had scalability problems when dealing with serious high end use. As well PHP 5 adds additional features such as exception handling, and a stronger object oriented model, all the while being highly backward compatible.

Lets get started…

Section 2 - Variables and Expressions

As with other programming languages, PHP allows you to define variables. In PHP there are several variable types, but the most common is called a String. It can hold text and numbers. All strings begin with a **$** sign. To assign some text to a string you would use the following code:

```
$welcome text = "Hello and welcome to my website.";
```

This is quite a simple line to understand, everything inside the quotation marks will be assigned to the string. You must remember a few rules about strings though:
Strings are case sensitive so **$Welcome_Text** is not the same as **$welcome_text**
String names can contain letters, numbers and underscores but cannot begin with a number or underscore When assigning numbers to strings you do not need to include the quotes so: $user_id = 987 would be allowed.

Section 3: Functions and Classes

You can create functions in php:
Syntax:

Function functionName(arguments){
}

Example

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n){
      echo "Example function.\n";
      return $retval;
}?>
```

Classes can also be defined

```
<?php
class Cart {
      var $items; // Global variable

      function add_item($artnr, $num) {
            $this->items[$artnr] += $num;
      }
}
?>
```

Section 4 - Control Structures

In PHP Control Structures in PHP are very similar to other languages.

Conditional Statements

In PHP we have two conditional statements:

if (...else) statement - use this statement if you want to execute a set of code when a condition is true (and another if the condition is not true)

switch statement - use this statement if you want to select one of many sets of lines to execute

The If Statement

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement.

Syntax:

```
if (condition)
  code to be executed if condition is true;
else
  code to be executed if condition is false;
```

Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

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

If more than one line should be executed the lines should be enclosed within curly braces {}

The Switch Statement

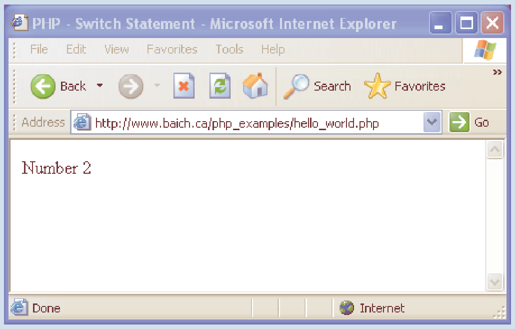
If you want to select one of many blocks of code to be executed, use the Switch statement.

Syntax

```
switch (expression){
      case label1:
            code to be executed if expression = label1;
            break;
      case label2:
            code to be executed if expression = label2;
            break;
      default:
            code to be executed if expression is different from both label1 and label2;
}
```

Example
This is how it works: First we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if none of the cases are true.

```
<html>
      <body>
      <?php
            $x = 2;
            switch ($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 - Switch Statement - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.baich.ca/php_examples/hello_world.php
Go
Done
Internet

```
Number 2
```

Looping
Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to perform this.
In PHP we have the following looping statements: **while** - loops through a block of code as long as a specified condition is true. **do...while** loops loop through a block of code once, and then repeats the loop as long as a special condition is true, **for** - loops through a block of code a specified number of times and **foreach** - loops through a block of code for each element in an array.

The while Statement
The while statement will execute a block of code if and as long a condition is true.
Syntax
while (condition)
code to be executed;
Example
The following example demonstrates a loop that will continue to run as long as the variable i is less than, or equal to 5. i will increase by 1 each time the loop runs:

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

The do...while Statement
The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.
Syntax

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

Example

```
<html>
<body>
      <?php
      $i=0;
      do{
            $i++;
            echo "The number is " . $i . "<br />";
      }while ($i<5);
      ?>
</body>
</html>
```

This example will increment the value of i at least once, and it will continue incrementing the variable i while it has a value of less than 5.

The for Statement
The for statement is used when you know how many times you want to execute a statement or a list of statements.
Syntax

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

Note: The for statement has three parameters. The first parameter is for initializing variables, the second parameter holds the condition, and the third parameter contains any increments required to implement the loop. If more than one variable is included in either the initialization or the increment section, then they should be separated by commas. The condition must evaluate to true or false.

```
<html>
<body>
      <?php
            for ($i=1; $i<=5; $i++)
            {
            echo "Hello World!<br />";
            }
      ?>
</body>
</html>
```

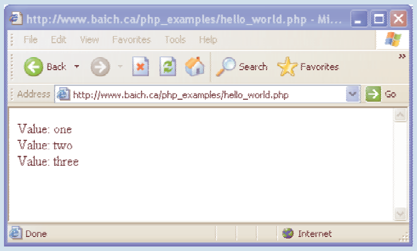
Example The above example prints the text "Hello World!" five times.

The foreach Statement
Loops over the array given by the parameter. On each loop, the value of the current element is assigned to $value and the array pointer is advanced by one - so on the next loop, you'll be looking at the next element.
Syntax

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

Example

```
<html>
<body>
<?php
      $arr=array("one","two","three");
      foreach ($arr as $value)
      {
      echo "Value: " . $value . "<br />";
      }
      ?>
</body>
</html>
```


http://www.baich.ca/php_examples/hello_world.php - Mi...
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.baich.ca/php_examples/hello_world.php Go
Done
Internet

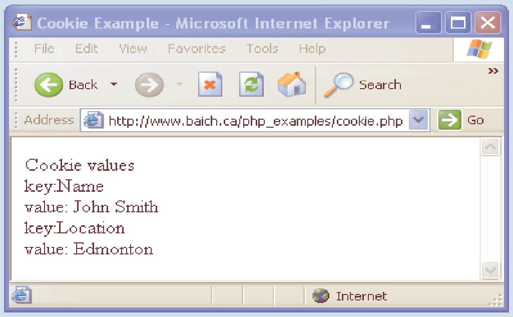
```
Value: one
Value: two
Value: three
```

The above example demonstrates a loop that will print the values of the given array.

<u>Section 5</u> - Cookies in PHP

Setting Cookies You can set a cookie with the **setcookie** function which is described below.
setcookie(string CookieName, string CookieValue, int CookieExpireTime, path, domain, int secure);
Retrieving Cookies You can use the $_COOKIE superglobal to retrieve your cookie.

```
<html>
        <head>
        <title>Cookie Example</title>
        </head>
<body>
<?php
        //set the cookie variables
        setCookie('Name','John Smith', time() + 60 * 60 * 24 * 5);
        setCookie('Location','Edmonton', time() + 60 * 60 * 24 * 5);
        //post the cookie variables
        extract( $POST );
        echo('Cookie values');
        //ouput each cookie and value
        foreach ($_COOKIE as $key=>$value){
                echo('<br>key:'.$key);
                echo('<br>value: '.$value);
        }
?>
</body>
</html>
```

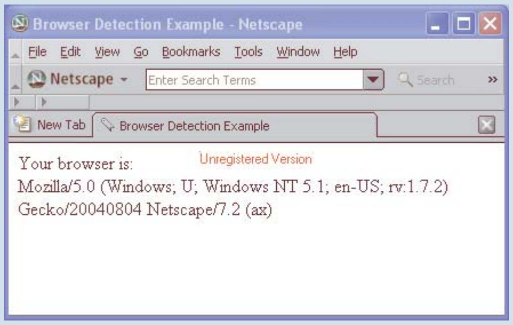

Cookie Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search
Address http://www.baich.ca/php_examples/cookie.php Go
Internet

```
Cookie values
key:Name
value: John Smith
key:Location
value: Edmonton
```

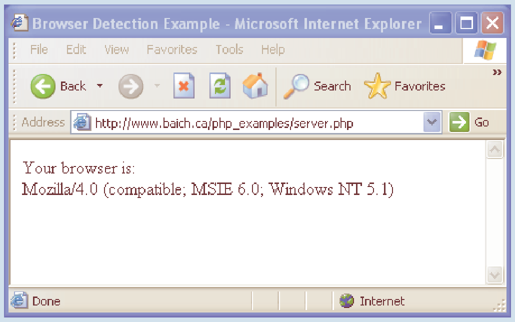
Section 6 - Client/Server Variables

$_ENV - Contains system environment variables
$_GET - Contains variables in the query string, including from GET forms
$_POST - Contains variables submitted from POST forms
$_COOKIE - Contains all cookie variables
$_SERVER - Contains server variables, such as HTTP_USER_AGENT
$_REQUEST - Contains everything in $_GET, $_POST, and $_COOKIE
$_SESSION -- Contains all registered session variables
These variables hold useful information about server configuration, request information, and session information. They are all super-global associative arrays. A useful example is browser detection as follows:

```
<html>
<head>
<title>Browser Detection Example</title>
</head>
<body>
<?php
      echo("Your browser is: <br>");
      echo($_SERVER["HTTP_USER_AGENT"]);
?>
</body>
</html>
```


Browser Detection Example - Netscape
File Edit View Go Bookmarks Tools Window Help
Netscape
Enter Search Terms
Search
New Tab
Browser Detection Example
Unregistered Version

```
Your browser is:
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2)
Gecko/20040804 Netscape/7.2 (ax)
```

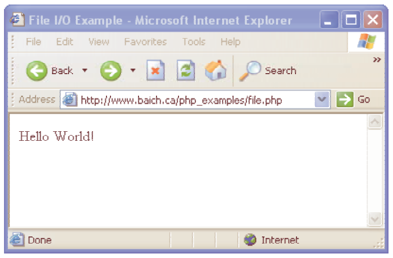

Browser Detection Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://www.baich.ca/php_examples/server.php
Go
Done
Internet

```
Your browser is:
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

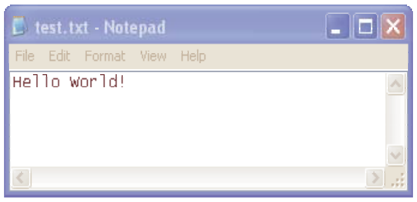
Section 7 - File I/O

The **fopen()** function is used to open files in PHP–If the fopen() function is unable to open the specified file, it returns 0 . The fclose() function is used to close a file and the the feof() function is used to determine if the end of file is true. Keep in mind that y ou cannot read from files opened in w, a, and x mode!

```
<html>
<head>
      <title>File I/O Example</title>
</head>
<body>
<?php
      if (!($f=fopen("test.txt","r")))
            exit("Unable to open file.");
      while (!feof($f)){
            $x=fgetc($f);
            echo $x;
      }
      fclose($f);
?>
</body>
</html>
```


File I/O Example - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Address
http://www.baich.ca/php_examples/file.php
Go
Done
Internet

```
Hello World!
```



```
Hello World!
```

Section 8 - Database Connectivity

The ability for a scripting language to connect to a database is essential for it to be functional and to become popular amongst programmers. It needs to have a simple interface to perform database functions but still be powerful enough to perform the most complex queries. This is achieved in PHP through the use of includes statements, the ability to embed queries in the middle of pages and through the use of different built in packages for connecting to databases. Heavily used code such as those variables storing information about the variables can be placed in a separate file and simply included at the top of each PHP page requiring a database connection rather than having to retype them each time.

For example:

dbConnect.php

```
<?php
/* declare some relevant variables */
$DBhost = "Your-MySQL-servers-IP-or-
domainname";
$DBuser = "Your user name";
$DBpass = "Your Password";
$DBName = "The Name of the Database";
$table = "Table Name";
?>
```

index.php

```
<?php

include(dbconnect.php');

?>
```

PHP allows for connections to multiple types of databases. However, each database uses slight different functions. The most commonly used functions are highlighted in the following tables with examples for each database type.

	mySQL	Oracle	ODBC
Connect	mysql_connect	Ora_Logon	$ odbc_connect
Open DB	mysql_select_db	Ora_Open	
Parse Query		Ora_Parse	
Execute Query	mysql_query	Ora_Exec	odbc_exec
Retrieve Data	mysql_result	Ora_Fetch_Into	odbc_fetch_into
Array for Results	mysql_fetch_array		odbc_fetch_array

Count Results	mysql_numrows	ora_numcols()
		ora_numrows()
			odbc_num_rows

Close Connection	mysql_free_result mysql_close	Ora_Close Ora_Logoff	Odbc_free_result odbc_close

MySQL – Using arrays to store data

Database is first connected using mysql_connect() command and the database is opened using @mysql_select_db(). Queries can then be executed using the mysql_query() function and results can be stored in an array using mysql_fetch_array().

```
<?php
        /* declare some relevant variables */
        $DBhost = "Your-MySQL-servers-IP-or-domainname";
        $DBuser = "Your user name";
        $DBpass = "Your Password";
        $DBName = "The Name of the Database";
        mysql_connect($DBhost,$DBuser,$DBpass) or die("Unable to connect to database");
        @mysql_select_db("$DBName") or die("Unable to select database $DBName");
        $sqlquery = "SELECT * FROM $table WHERE course = 'cmput410'";
        $result = mysql_query($sqlquery);

        while($row = mysql_fetch_array($result)) {
echo "<font face=arial size=-1><p><b>Student Name: </b>" . row[first_name] .
"<br><b>Grade:</b>". $row[grade]. "</p>";
        }
        if(mysql_num_rows($result)<1){
                echo "<font face=Arial size=+0><b>No Results!</b></font><br>";
        }
        mysql free result($result);
```

Oracle
To use an Oracle database PutEnv("ORACLE_SID=ORASID") is first used to set up the environment. The function Ora_Logon() connects to the database and Ora_Open ($connection) opens the connection and is sets a cursor that will be used for further operations. Using Ora_Parse() will parse the query to ensure that it is a valid SQL statemtent and then can be executed using Ora_Exec(). Results can be obtained with Ora_Fetch_Into() if a select query is made. Otherwise the Ora_commit() function must be called to lock in any changes from update or insert statements.

Connecting to a database using PHP is very similar to connecting to a database using any other programming language. A connection needs to be established, a query is executed, results are parsed and the connection is closed. PHP, through different functions for different databases is able to exploit the benefits of each database type to make connections efficient, quick and powerful if used correctly.

```
<?php
        PutEnv("ORACLE_SID=ORASID");

        $connection = Ora_Logon ("username","password");
        if ($connection == false){
                echo Ora_ErrorCode($connection).": ".Ora_Error($connection)."<BR>";
        exit;
        }

        $cursor = Ora_Open ($connection);
        if ($cursor == false){
                echo Ora_ErrorCode($connection).": ".Ora_Error($connection)."<BR>";
                exit;
        }

        $query = “SELECT * FROM table WHERE course = 'cmput410'";
        $result = Ora_Parse ($cursor, $query);
        if ($result == false){
                echo Ora_ErrorCode($cursor).": ".Ora_Error($cursor)."<BR>";
                exit;
        }

        $result = Ora_Exec ($cursor);
        if ($result == false){
                echo Ora_ErrorCode($cursor).": ".Ora_Error($cursor)."<BR>";
                exit;
        }

        echo "<table border=1>";
        echo "<tr><td><b>Student Name</b></td><td> <b>Grade</b></td></tr>";

        while (Ora_Fetch_Into ($cursor, &$values)){
                $name = $values[0];
                $grade = $values[1];

                echo "<tr><td>$name</td><td>$grade</td></tr>";
        }
        echo "</table>";
        Ora_Close ($cursor);
        Ora_Logoff ($connection);
?>
```

Section 9 - PHP vs ASP.NET

Although the ASP.NET framework provides for true object-oriented programming (OOP), and true inheritance, polymorphism, and encapsulation, this is also a weakness for web development. What is gained in robustness, is paid for in efficiency. ASP.NET is expensive with respect to memory usage and execution time when compared with PHP, due to its longer code path. For web-based applications, these limitations can be a serious problem, since for high end use, web applications must scale to thousands of users per second. Also, ASP.NET uses ODBC for integration with databases. Although this provides a consistent set of calling functions to access the target database, it does not allow you to take advantage of a database's unique features like PHP does. Furthermore, ASP.NET officially requires that you use the IIS web server and although APS.NET is free, its platform, IIS, is not.

PHP vs CGI

Unlike Perl, a general purpose scripting language, PHP was designed from the ground up to be used for scripting web pages. As a result, it has a lot of built in facilities that simplify the process. PHP code is, also, embedded directly into XHTML documents. This allows for clear concise XHTML code without having to use multiple print statements, as is necessary with other CGI-based languages.

Section 10 - **Handy References**:

www.phpbuilder.com

php.resourceindex.com

www.php.net

www.w3schools.com/php/default.asp

www.phpfreaks.com

www.vend.com