ADVANCED PHP

Contents

- Class
- Sessions
- PHP and MySQL

Class

Object oriented programming in PHP

- PHP, like most modern programming languages (C++, Java, JavaScript, etc.), supports the creation of objects.
- Creating an object requires you to first define an object class (containing variables and/or function definitions) and then using the “new” keyword to create an instance of the object class. (Note that the object must be defined before you instantiate it.)

```
<?php
//  Assume that the "Person" object has been previously defined. . .
$x = new Person();  //  creates an instance of the Person class
$y = new Vehicle('Jaguar','green');  //  creating an object and passing
                                     //  arguments to its constructor
?>
```

Defining (declaring) a class

- Use the “class” keyword which includes the class name (case-insensitive, but otherwise following the rules for PHP identifiers). Note: The name “stdClass” is reserved for use by the PHP interpreter.

```
<?php
class Person
{
    public $name;
    function set_name($new_name)  {
      $this -> name = $new_name;
   }
    function get_name() {
      return $this -> name;
   }
}
```

- Use the “$this” keyword when accessing properties and functions of the current object

Declaring a class (cont.)

- Properties and functions can be declared as “public”, “private”, “protected”
- Default access: public

```
<?php
class Person
{
    protected $name;
    protected $age;
    function set_name($new_name)  {
        $this -> name = $new_name;
    }
    function get_name() {
        return $this -> name;
    }
}
```

Accessing properties and methods

- Once you have an object, you access methods and properties (variables) of the object using the -> notation.

```
<?php
$me = new Person();
$me -> set_name('Russ');
$name = $me -> get_name();
echo $me -> get_name();
$age = 36;
$me -> set_age($age);
?>
```

Constructors and destructors

- Constructors are methods that are (generally) used to initialize the object's properties with values as the object is created.
- Declare a constructor function in an object by writing a function with the two underscores "__"

```
<?php
class Person {
   protected $name;
   protected $age;
   function __construct($new_name, $new_age)  {
          $this -> name = $new_name;
          $this -> age = $new_age;
   }
  // . . . other functions here . . .
}
$p = new Person('Bob Jones', 45);
$q = new Person('Hamilton Lincoln', 67);
?>
```

- Destructors (defined with a function name of __destructor()) are called when an object is destroyed

Inheritance

- Use the “extends” keyword in the class definition to define a new class that inherits from another.

```
<?php
  class Employee extends Person  {
     public $salary;
     function __construct($new_name, $new_age, $new_salary)  {
        $this -> salary = $new_salary;
        parent::__construct($new_name, $new_age);
     }
     function update_salary($new_salary)  {
        $this -> salary = $new_salary;
     }
     $emp = new Employee('Dave Underwood', 25, 25000);
?>
```

Inheritance (cont.)

- The constructor of the parent isn't called unless the child explicitly references it (as in this previous case).
- You could "hard-code" the call to the parent constructor using the function call "Person::__construct($new_name, $new_age);"
- You can use the "self" keyword to ensure that a method is called on the current class (if a method might be subclassed), in this style self::method();
- To check if an object is of a particular class, you can use the instanceof operator.

More on classes

- You can also define interfaces (for which any class that uses that interface must provide implementations of certain methods), and you can define abstract classes or methods (that must be overridden by its children).
- The keyword “final” can be used to denote a method that cannot be overridden by its children.

```
class Person {
    public $name;

    final function get_name() {
        return $this -> name;
    }
}
```

Session

PHP sessions

- By default, HTML and web servers don’t keep track of information entered on a page when the client’s browser opens another page.
- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- Sessions only store information temporarily

PHP sessions (cont.)

- To start a session, use the function session_start() at the beginning of your PHP script before you store or access any data. For the session to work properly, this function needs to execute before any other header calls or other output is sent to the browser.

```
<?php
session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Session example</title>
</head>
<body>
<?php
include_once ('object.php');  //  Includes definition of the Person class
$_SESSION['hello'] = 'Hello world';
echo $_SESSION['hello'] . "<br/><br/>\n";
$_SESSION['one'] = 'one';
$_SESSION['two'] = 'two';
$me = new Person("Russ", 36, 2892700);
$_SESSION['name'] = $me->get_name();
echo "Testing " . $_SESSION['one'] ."," . $_SESSION['two'] . ", " . $me->get_number() . " . . .<br/>\n";
?>
</body></html>
```

view the output page

Using session variables

- Once a session variable has been defined, you can access it from other pages.

```
<?php
session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Session example 2</title>
</head>
<body>
<?php
echo "Welcome to a new page ". $_SESSION['name'] "!<br/>\n";
echo "Hope you enjoy your stay!  <br/>";
?>
<p>Back to regular HTML text...
</p>

</body>
</html>
```

view the output page

More on session variables

- You need to include a call to the session_start() function for each page on which you want to access the session variables.
- A session will end once you quit the browser (unless you've set appropriate cookies that will persist), or you can call the session_destroy() function.
- The function session_unset() removes all session variables. If you want to remove one variable, use the unset($var) function call.
- The default timeout for session is 24 minutes. It's possible to change this timeout.

Deleting all session variables

```
<?php
session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
<title>Session example 3</title>
</head>
<body>
<?php
echo "Deleting all session variables using session_unset(); <br/>\n";
session_unset();
echo "Now the session variables are gone.  <br/>\n";
if (isset($_SESSION['name']))
   {  echo $_SESSION['name'] . "<br/>\n"; }
else
   {  echo "Session variable is not here.";  }
?>

</body>
</html>
```

view the output page

PHP and MySQL

Putting Content into Your Database with PHP5+

- Connect to the database server and login

```
mysqli_connect(”servername","username","password", "database");
```

- Send SQL queries to the server to add, delete, and modify data

```
mysqli_query("query");
```

- Close the connection to the database server (to ensure the information is stored properly)

```
mysqli_close();
```

Student Database: data_in.php

```
<html>
<head>
<title>Putting Data in the DB</title>
</head>
<body>
<?php
/*insert students into DB*/
if(isset($_POST["submit"]))  {
   $conn = mysqli_connect("mysql", "martin", ”martin", "martin");
   if (!$conn) {
       die("Connection failed: " . mysqli_connect_error());
   }
   $date=date("Y-m-d");  /*  Get the current date in the right SQL format  */
   $sql="INSERT INTO students  VALUES(NULL,'" . $_POST["f_name"] . "','" . $_POST["l_name"] . "'," .
     $_POST["student_id"] . ",'" . $_POST["email"] . "','" . $date . "'," . $_POST["gr"] . ")";      /*
     construct the query  */
   mysqli_query($conn, $sql);   /*  execute the query  */
   mysqli_close();
   echo"<h3>Thank you. The data has been entered.</h3> \n";
   echo'<p><a href="data_in.php">Back to registration</a></p>' . "\n";
   echo'<p><a href="data_out.php">View the student lists</a></p>' ."\n";
   }
```

Student Database: data_in.php

```
else  {
?>
<h3>Enter your items into the database</h3>
<form action="data_in.php" method="post">
First Name: <input type="text" name="f_name" /> <br/>
Last Name: <input type="text" name="l_name" /> <br/>
ID: <input type="text" name="student_id" /> <br/>
email: <input type="text" name="email" /> <br/>
Group: <select name="gr">
       <option value ="1">1</option>
       <option value ="2">2</option>
       <option value ="3">3</option>
</select><br/><br/>
<input type="submit" name="submit" /> <input type="reset" />
</form>

<?php
   }     /*  end of "else" block  */
?>

</body>
</html>
```

view the output page

Getting Content out

- Send an SQL query to the server to select data from the database into an array

```
$result=mysqli_query($conn, $query);
while ($row = mysqli_ fetch_assoc($result)) {
//hiển thị dữ liệu
echo ”Id: “ . $row[”id”] . ” - Name:” . $row[”name”] .”<br/>”;
}
```

Student Database: data_out.php

```
<html>
<head>
<title>Getting Data out of the DB</title>
</head>
<body>
<h1> Student Database </h1>
<p> Order the full list of students by
<a href="data_out.php?order=date">date</a>,
<href="data_out.php?order=student_id">id</a>, or
by <a href="data_out.php?order=l_name">surname</a>.
</p>

<p>
<form action="data_out.php" method="post">
Or only see the list of students in group
<select name="gr">
  <option value ="1">1</option>
  <option value ="2">2</option>
  <option value ="3">3</option>
</select>
<br/>
<input type="submit" name="submit" />
</form>
</p>
```

Student Database: data_out.php

```
<?php
$db = mysqli_connect("mysql","martin",”martin", "martin");

switch($_GET["order"]){
case  'date':   $sql = "SELECT * FROM students ORDER BY date"; break;
case  'student_id': $sql = "SELECT * FROM students ORDER BY student_id"; break;
case  'l_name': $sql = "SELECT * FROM students ORDER BY l_name"; break;
default: $sql = "SELECT * FROM students";  break;
}
if(isset($_POST["submit"])){
  $sql = "SELECT * FROM students WHERE gr=" . $_POST["gr"];
}
$result=mysqli_query($db, $sql);      /*  execute the query  */
while($row=mysql_fetch_assoc($result)){
  echo "<h4> Name: " . $row["l_name"] . ', ' . $row["f_name"] . "</h4> \n";
  echo "<h5> ID: " . $row["student_id"] . "<br/> Email: " . $row["email"] . "<br/> Group: "
    . $row["gr"] . "<br/> Posted: " . $row["date"] . "</h5> \n";
}
mysqli_close();
?>
</body>
</html>
```

view the output page

More on PHP and SQL

To increase security of your PHP/SQL setup (and to make it easier to change the database you use), it's recommended that you build an "include" file that will have the information you use to connect to the database.

```
<?php
/* Save this as db_login.php (or whatever you like) and include it
in your php script.  */

//  Here's the information to connect to the database.
$db_host = 'mysql';
$db_database='martin';
$db_username='martin';
$db_password='xxxxx';
?>
```

If someone tries to view this file through their browser, the PHP interpreter will process it and return a blank page to the user (there's no HTML in the file).

Connecting to the database

Now you can build your PHP script as follows (using the commands that we discussed previously):

```
<?php
require_once ('db_login.php');
$connection = mysqli_connect($db_host, $db_username, $db_password, $db_database);
if (!$connection)   /*  check if the connection was actually successful  */
   {
     exit("Could not connect to the database: <br/>" .
        htmlspecialchars(mysqli_connect_error()) );
   }
else  {
  // more statements here. . .
      }
?>
```

Note: The function 'htmlspecialchars()' converts special characters in a string into their HTML escape sequences (like '&' into '&' and so forth).

This can also be used to increase the security of your code by and help thwart attacks on your database by passing it information that your client has submitted before trying to insert it in your database.

MySQL queries inside of PHP

Your mySQL queries from a PHP script are the same as they are as when you're using the mySQL program from the command line with one difference… the queries do not have a semi-colon at the end.

```
<?php
//  Assuming a valid database connection has been established.
//  Build the query string by assigning variables...
$query = $select . $column . $from . $tables . $where;
$result = mysqli_query($connection, $query);
if(!$result) {
   exit("Could not query the database: <br/>" .
                          htmlspecialchars(mysqli_connect_error()) );
            }
else   {
   // process the data
       }
?>
```