PHP Tips

Columbia University. CS4111 – Introduction to Databases

Basic Setup

All PHP programs are simply text files with a .php extension surrounded by the tags shown below.

```
<?php
        ...
?>
```

Functions

You can define functions in PHP, just as in other programming languages.

```
function getMyTrip($userId, $tripId) {
        ...
}
```

MySQL Connectivity

Refer to the documentation at http://php.net/manual/en/book.mysqli.php to learn about how to access your MySQL database from PHP (in addition to the examples shown below). You may also find the examples at http://www.w3schools.com/php/php_ref_mysqli.asp a helpful introduction.

Require_once

Since most or all of your pages will need a database connection, it is a good idea to define the database connection code in a common PHP file. You can then include this common PHP file in your other PHP files using require_once(). You can also include any other frequently used functions in this common file.

For example, you could create a file "common.php" where you define $conn as follows:

```
<?php
        $dburl = "cs4111.cbnwneimpvpz.us-west-2.rds.amazonaws.com";
        $dbuser = "username";
        $dbpassword = "password";
        $dbname = "cs4111";
        $conn = mysqli_connect($dburl,$dbuser,$dbpassword,$dbname);
?>
```

Then you can include this file at the beginning of each PHP file, as follows:

```
require_once "common.php";
```

Note: Don't forget to close this connection by using mysqli_close($conn) at the end of each PHP file where you include common.php.

Displaying Errors and Debugging

The following PHP statement will ensure that your browser displays PHP error messages if your script generates errors:

```
ini_set('display_errors', 'On');
```

You may find it helpful to include this in your common PHP file, together with the database connection code discussed above.

Input Validation

You may find these filtering capabilities in PHP helpful for validating user input: http://us3.php.net/manual/en/book.filter.php.

Committing Transactions

If you execute updates to your database (INSERT, UPDATE, or DELETE statements), you will need to execute mysqli_commit($conn) for your changes to be committed to your the database.

Error Detection and Handling

mysqli_query() returns false if the SQL statement failed, and mysqli_error($conn) retrieves the last error to occur on the connection. Therefore, you may find it helpful to place your mysqli_query() in an error-checking conditional, to ease your debugging.

```
<?php
        if (!mysqli_query($conn, $stmt)) {
                echo("Error: ", . mysqli_error($conn));
        }
?>
```

Receiving data from Form in HTML

Suppose you have the following HTML form:

```
<form action="menu.php" method="POST">
<table>
        <tr>
                <td align="right">User ID:</td>
                <td><input type="text" name="userid" size="16" maxlength="16"></td>
.........
</table>
</form>
```

In menu.php, you can receive data inputted into this form as follows:

```
$userid = $_POST["userid"];
```

or

```
$userid = $_REQUEST["userid"];
```

Then, you can use $userid as part of your SQL query.

Note: _REQUEST can deal with both GET type and POST type requests.

Creating a Dynamic Page from a Link

If you want to create a dynamic page from a link, you can dynamically create link as follows:

```
while($res = mysqli_fetch_row($stmt)) {
        print "<a href=\"showcity.php?citycode=" . urlencode($res[0]) . "&countrycode=" .
                urlencode($res[1]) . "\">" . $res[2] . "</a>";
}
```

It will generate the following HTML source:

```
<a href="showcity.php?citycode=NYC&countrycode=US">New York City</a>
<a href="showcity.php?citycode=TYO&countrycode=JP">Tokyo</a>
```

Then, showcity.php can receive the citycode as

```
$citycode = $_GET['citycode'];
$countrycode = $_GET['countrycode'];
```

or

```
$citycode = $_REQUEST['citycode'];
$countrycode = $_REQUEST['countrycode'];
```