Transitions, Security and Updating Databases

In this exercise, we are going to learn how to:

- Enable the transition from one web page to another
- Encrypt passwords with a one way hash and verify encrypted passwords
- Make changes to a database from the web using AJAX and REST

Exercise 1: Simple Transitions Between Web Pages

In all of these exercises, make sure you give your files an extension of .php. The PHP code will not work otherwise.

Create two files called pageone.php and pagetwo.php. Pageone.php has the following code:

```
<!doctype html>
<html>
  <head>

  </head>
  <body>
    <p>This is page one</p>
    <form>
      <button type="button" >Press this to go to page two</button>
    </form>
  </body>
</html>
```

Pagetwo.php is similar:

```
<!doctype html>
<html>
  <head>

  </head>
  <body>
    <p>This is page two</p>
```

```
    <form>
      <button type="button" >Press this to go to page one</button>
    </form>
  </body>
</html>
```

To make the button on pageone.php work as advertised, add the following code to the button:

onclick="window.location='pagetwo.php'"

Likewise, to make pagetwo.php transition to pageone.php, add:

onclick="window.location='pageone.php'"

Exercise 2: Sessions and Transitioning in PHP

Create a new file called mylogin.php and myusercreate.php. Here is the initial code for mylogin.php:

```
<!doctype html>
<html>
  <head>
    <style>
      .leftcol{
        display: inline-block;
        width: 100px;
        text-align: right;
        font-weight: bold;
      }
      .rightcol{
        display: inline-block;
      }

    </style>
    <script>
```

```
      function loginstate(){
         if (user.length==0)
            loginbutt.disabled=true;
         else
            loginbutt.disabled=false;
      }
   </script>
</head>
<body>
  <div id="messages" style="color: red;">

  </div><br />
  <div><a href="myusercreate.php">Create a New User Here</a></div>
  <div class="leftcol">
     <label for="user">Username: </label>
  </div>
  <div class="rightcol">
     <input type="text" id="user" onkeyup="loginstate();"/>
  </div><br />
  <div class="leftcol">
     <label for="pass">Password: </label>
  </div>
  <div class="rightcol">
     <input type="password" id="pass" />
  </div><br />
  <div class="leftcol">
  </div>
  <div class="rightcol">
     <button type="button" id="loginbutt" disabled>Login</button>
```

```
    </div><br />


  </body>
</html>
```

And here is the initial code for myusercreate.php

```
<!doctype html>
<html>
  <head>
    <style>
      .leftcol{
        display: inline-block;
        width: 100px;
        text-align: right;
        font-weight: bold;
      }
      .rightcol{
        display: inline-block;
      }


    </style>
    <script>
      function loginstate(){
        if (user.length==0)
          createbutt.disabled=true;
        else
          createbutt.disabled=false;
      }
    </script>
```

```
  </head>
  <body>
    <div id="messages" style="color: red;">

    </div><br />
    <div class="leftcol">
      <label for="user">Username: </label>
    </div>
    <div class="rightcol">
      <input type="text" id="user" onkeyup="loginstate();"/>
    </div><br />
    <div class="leftcol">
      <label for="pass">Password: </label>
    </div>
    <div class="rightcol">
      <input type="password" id="pass" />
    </div><br />
    <div class="leftcol">
    </div>
    <div class="rightcol">
      <button type="button" id="createbutt" disabled>Create</button>
    </div><br />

  </body>
</html>
```

Add the following code to the very beginning of pageone.php.

```
<?php
 session_start();
 if (!($_SESSION['haslogin']=='Y'))
```

```
    header('location: mylogin.php?from=pageone.php');
?>
```

Do something similar for pagetwo.php, but set the header to from=pagetwo.php

Observe that now, going to pageone.php and pagetwo.php redirects you to mylogin.php.

What do each of these commands do?

Session_start initiates a session. It checks for a session cookie on your machine. If the session cookie does not exist, it creates one. Otherwise, it retrieves the session cookie.

$_SESSION['haslogin'] is a session variable. Every user on the server will have a different $_SESSION['haslogin']. Of course, since we haven't programmed the value, at the moment, $_SESSION['haslogin'] will never be equal to Y.

The header('location') command sends a HTTP header command. There are two special HTTP headers- error messages like 404 (file not found) and location, which redirects the browser to a different webpage.

Exercise 3: Creating Our User

In order for us to get access to pageone.php and pagetwo.php we will need to login. However, to login, we will need to create logins. Create the file douser.php that looks like this:

```
<?php
$user=$_REQUEST['user'];
$pass=password_hash($_REQUEST['pass'],PASSWORD_BCRYPT);
$mydb = new PDO('mysql:host=localhost;dbname=<your database here>', '<username here>', '<password here>');
$stmt=$mydb->prepare("Select username ".
    "from accounts ".
    "where username=:user"
    );
$stmt->execute([":user"=>$user]);
if ($stmt->rowCount()==0)
  $stmt=$mydb->prepare("Insert into accounts (username, userpass) ".
    "values(:user,:pass)"
    );
  else
$stmt=$mydb->prepare("Update accounts ".
    "set userpass=:pass ".
    "where username=:user"
    );
$stmt->execute([":user"=>$user, ":pass"=>$pass]);
echo "executed";
```

13. ?>

Test the page by going to http://<your domain>/<your directory>/ douser.php?user=test&pass=test

Check the accounts table in your database to see if the test user was created.

This is what the code does:

Line 3: We encrypt the password using the Blowfish encryption algorithm.

Line 5-7: Check if the username exists. If the username does not exist, the query will return 0 rows.

Line 8: If the username does not exist, add the user to the database

Line 9-10: If the username exists, change the password.

For the purposes of creating the user, we also want to check if the user exists. Create a file called checkuser.php as follows:

```
<?php
 $user=$_REQUEST['user'];
 $mydb = new PDO('mysql:host=localhost;dbname=<database name>', '<username>', '<password>');
 $stmt=$mydb->prepare("Select username ".
                 "from accounts ".
                 "where username=:user"
                );
 $stmt->execute([":user"=>$user]);
 if ($stmt->rowCount()==0)
  echo "false";
 else
  echo "true";
?>
```

Now that we have done these things, we can now get myusercreate.php to work. The functions in the page should be: (note loginstate already exists and should be modified)

```
function loginstate(){
   if (user.length==0)
      createbutt.disabled=true;
```

```
   else{
      createbutt.disabled=false;
      const xhttp = new XMLHttpRequest();
      xhttp.onload = function() {
        if (this.responseText=="false")
          createbutt.innerHTML="Create";
        else
          createbutt.innerHTML="Update";

      }
      xhttp.open("GET", "checkuser.php?user="+user.value, true);
      xhttp.send();
   }
}
function modifyrecord(){
 const xhttp = new XMLHttpRequest();
 xhttp.onload = function() {
   if (this.responseText=="executed"){
      messages.innerHTML="User "+user.value;
    if (createbutt.innerHTML=="Create")
       messages.innerHTML+=" created!";
    else
       messages.innerHTML+=" updated!";

   }
 }
 xhttp.open("GET", "douser.php?user="+user.value+"&pass="+pass.value, true);
 xhttp.send();
}
```

Link modifyrecord to the button.

Exercise 4: Performing Our Login

Now that we have users, we can actually login. However, before we do that, we need to create code to logout so we can test the login. Create a file called mylogout.php. Use this file as needed to logout.

```
<?php

 session_start();

 $_SESSION['haslogin']='N';


?>
```

Now, create verifylogin.php which actually tests the password.

```
1.  <?php
2.  session_start();
3.  $user=$_REQUEST['user'];
4.  $pass=$_REQUEST['pass'];
5.  $jsonfile = file_get_contents('amazoniainfo.json');
6.  $dbinfo = json_decode($jsonfile, false);
7.  $mydb = new PDO('mysql:host=localhost;dbname='.$dbinfo->dbname, $dbinfo->dbuser,
    $dbinfo->dbpass);
8.  $stmt=$mydb->prepare("Select username, userpass ".
                i.  "from accounts ".
               ii.  "where username=:user"
              iii.  );
9.  $stmt->execute([":user"=>$user]);
10. if ($stmt->rowCount()==0)
11.   echo "false";
12. else {
13.   $result=$stmt->fetch(PDO::FETCH_OBJ);
14.   echo $result->userpass;
15.   if (password_verify($pass,$result->userpass)){
16.     echo "true";
17.     $_SESSION['haslogin']='Y';
18.   }
19.   else
20.     echo "false";
21.   }
22. ?>
```

The only things new about this code are:

Line 13: the fetch command which fetches a single record from the query result.

Line 15: password_verify which checks a plain text password against the encrypted password

Line 17: This is how we login. We change the state of the session variable.

In mylogin.php, we can now create an AJAX function to call verifylogin.

```
function checklogin(){
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    console.log(this.responseText);
    if (this.responseText=="true")
       window.location="<?php echo $_REQUEST['from'];?>";
    else
      messages.innerHTML="Login failure."

  }
  xhttp.open("GET", "verifylogin.php?user="+user.value+"&pass="+pass.value, true);
  xhttp.send();
}
```

Link this to the button and now everything should be working.