**Exercise 02\_09\_01 – Step 1**

In this Exercise, we will learn how to maintain state, or store persistent data about a Web site visit, using hidden form fields, query strings, cookies, and sessions.



1. Create a folder named Exercise 02\_09\_01 and open it with your IDE. Open up a clean server console on your LAMPP stack. Login to MySQL as root to open up the command line for MySQL Monitor as follows:  
   ***# mysql -h localhost -u root –p***
2. At the MySQL Monitor prompt, enter the following command to create a database and verify it was created:  
   ***> CREATE DATABASE internships;  
   > USE internships;***  
   ***> SHOW DATABASES;***
3. At the MySQL Monitor prompt, enter the following command to create the ***interns*** table:  
   ***> CREATE TABLE interns (internID SMALLINT NOT NULL   
    AUTO\_INCREMENT PRIMARY KEY,   
    email VARCHAR(40), password\_md5 VARCHAR(32),  
    -> first VARCHAR(40), last VARCHAR(40));***
4. At the MySQL Monitor prompt, enter the following command to view the structure of the new table:  
   ***> DESCRIBE interns;***
5. At the MySQL Monitor prompt, enter the following command to create the ***opportunities*** table:  
   ***> CREATE TABLE opportunities*** ***(opportunityID SMALLINT   
    NOT NULL AUTO\_INCREMENT PRIMARY KEY,   
    company VARCHAR(40), city VARCHAR(25),  
    start\_date DATE, end\_date DATE,  
    -> position VARCHAR(30), description VARCHAR(250));***
6. At the MySQL Monitor prompt, enter the following command to view the structure of the new table:  
   ***> DESCRIBE opportunities;***
7. At the MySQL Monitor prompt, enter the following command to create the ***assigned\_opportunities***: ***> CREATE TABLE assigned\_opportunities***   
    ***(opportunityID SMALLINT, internID SMALLINT   
    date\_selected DATE, date\_approved DATE);***
8. At the MySQL Monitor prompt, enter the following command to view the structure of the new table:  
   ***> DESCRIBE assigned\_opportunities;***
9. Download the file ***opportunities.txt*** from Google Classroom. Place the file into the server ***mysql/data/internships*** folder. At the MySQL Monitor prompt, enter the following command to load the data into the ***opportunities*** table:  
   ***> LOAD DATA INFILE ‘opportunities.txt’   
    INTO TABLE opportunities;***
10. At the MySQL Monitor prompt, enter the following command to view the data in the table:  
    ***> SELECT \* FROM opportunities;***Type exit or quit to logout of MySQL Monitor and shut down the server console.

**Exercise 02\_09\_01 – Step 2**



1. Let’s begin to build a Registration/Login page. Create a new file called ***InternLogin.php***. Scaffold a basic HTML code layout into it. Complete our standard opening documentation in the ***<head>*** element. Make sure to have the ***modernizr*** <script> linked in. Set the <title> content to ***College Internships***:  
   ***<!doctype html>  
   <html>  
   <head>  
    <title>College Internships</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
   <body>  
   </body>  
   </html>***
2. Add the following text and elements to the document body:  
   <body>  
    ***<h1>College Internships</h1>  
    <h2>Register / Login</h2>  
    <p>New interns, please complete the top form to register as   
    a user. Returning users, please complete the second form to   
    login.</p>***</body>  
   Give this a browser/server test.
3. Add the following Web form for user registration:  
    ***<h3>New Intern Registration</h3>  
    <form action="RegisterIntern.php" method="post">  
    <p>Enter your name: First  
    <input type="text" name="first">  
    Last:   
    <input type="text" name="last">  
    </p>  
    <p>  
    Enter your e-mail address:   
    <input type="text" name="email">  
    </p>  
    <p>  
    Enter a password for your account:   
    <input type="password" name="password">  
    </p>  
    <p>  
    Confirm your password:   
    <input type="password" name="password2">  
    </p>  
    <p><em>(Passwords are case-sensitive and must be at   
    least 6 characters long)</em></p>  
    <input type="reset" name="reset" value="Reset   
    Registration Form">  
    <input type="submit" name="register"   
    value="Register">  
    </form>***Give this a browser/server test.
4. Add a second Web form for returning user login:  
    ***<h3>Returning Intern Login</h3>  
    <form action="VerifyLogin.php" method="post">  
    <p>  
    Enter your e-mail address:   
    <input type="text" name="email">  
    </p>  
    <p>  
    Enter your password:   
    <input type="password" name="password">  
    <p><em>(Passwords are case-sensitive and must be at   
    least 6 characters long)</em></p>  
    <input type="reset" name="reset" value="Reset   
    Login Form">  
    <input type="submit" name="login"   
    value="Log In">  
    </p>  
    </form>***Give this a browser/server test.

**Exercise 02\_09\_01 – Step 3**



1. Let’s begin to build a New Intern Registration page. Create a new file called ***RegisterIntern.php***. Scaffold a basic HTML code layout into it. Complete our standard opening documentation in the ***<head>*** element. Make sure to have the ***modernizr*** <script> linked in. Set the <title> content to ***Internship Registration***:  
   ***<!doctype html>  
   <html>  
   <head>  
    <title>Internship Registration</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
   <body>  
   </body>  
   </html>***
2. Add the following text and elements to the document body:  
   <body>  
    ***<h1>College Internship</h1>  
    <h2>Intern Registration</h2>  
    <?php  
      
    ?>***</body>  
   Give this a browser/server test.
3. Add the following code to validate the email submitted. The ***preg\_match()*** function is the same one used in ***Exercise 02\_03\_01 PHPEmail4.php***:  
    ***<?php  
    $errors = 0;  
    $email = "";  
    if (empty($\_POST['email'])) {  
    ++$errors  
    echo "<p>You need to enter an e-mail address.</p>\n";  
    }  
    else {  
    $email = stripslashes($\_POST['email']);  
    if (preg\_match("/^[\w-]+(\.[\w-]+)\*@[\w-]+  
    (\.[w-]+)\*(\.[A-Za-z]{2,})$/i", $email) == 0) {  
    ++$errors;  
    echo "<p>You need to enter a valid e-mail   
    address.</p>\n";  
    $email = "";  
    }  
    }  
    ?>***Give this a browser/server test. Test for no email entered, bad email, and a good email.
4. Enter the following code to validate that the first password was entered:  
    ***if (empty($\_POST['password'])) {  
    ++$errors;  
    echo "<p>You need to enter a password.</p>\n";  
    $password = "";  
    }  
    else {  
    $password = stripslashes($\_POST['password']);  
    }***Give this a browser/server test.
5. Enter the following code to validate that the confirmation password was entered:  
    ***if (empty($\_POST['password2'])) {  
    ++$errors;  
    echo "<p>You need to enter a confirmation   
    password.</p>\n";  
    $password2 = "";  
    }  
    else {  
    $password2 = stripslashes($\_POST['password2']);  
    }***Give this a browser/server test.
6. Enter the following code to validate that the password length is at least 6, and that the passwords match:  
    ***if (!empty($password) && !empty($password2)) {  
    if (strlen($password) < 6) {  
    ++$errors;  
    echo "<p>The password is too short.</p>\n";  
    $password = "";  
    $password2 = "";  
    }  
    if ($password <> $password2) {  
    ++$errors;  
    echo "<p>The passwords do not match.</p>\n";  
    $password = "";  
    $password2 = "";  
    }   
    }***Give this a browser/server test.
7. Finally, let’s add some user instructions if there were errors:  
    ***if ($errors > 0) {  
    echo "<p>Please use your browser's BACK button to   
    return " .   
    "to the form and fix the errors indicated.</p>\n";  
    }***Give this a browser/server test.

**Exercise 02\_09\_01 – Step 4**



1. Go back to ***RegisterIntern.php*** and let’s establish our database connection. Enter the following code above the last code that traps the errors as follows:  
    ***$hostname = "localhost";  
    $username = "root";  
    $passwd = "";  
    $DBConnect = false;  
    $DBName = "internships";  
    if ($errors == 0) {  
    $DBConnect = mysqli\_connect($hostname, $username,   
    $passwd);  
    if (!$DBConnect) {  
    ++$errors;  
    echo "<p>Unable to connect to the database server   
    error code: " . mysqli\_connect\_error() . "</p>\n";  
    }  
    }*** if ($errors > 0) {  
   Give this a browser/server test for a connection to a bad host and then a good host.
2. We should close down the database connection, so let’s add some code at the bottom of the script:  
    ***if ($errors == 0) {  
    echo "<p>Closing database \"$DBName\"   
    connection.</p>\n"  
    mysqli\_close($DBConnect);  
    }***Give this a browser/server test.
3. Let’s add an ***else*** clause to the above connect code and ***select*** the database as follows:  
    ***else {  
    $result = mysqli\_select\_db($DBConnect, $DBName);  
    if (!$result) {  
    ++$errors;  
    echo "<p>Unable to select the database   
    \"$DBName\" error code: "   
    . mysqli\_error($DBConnect) . "</p>\n";  
    }  
    }***Give this a browser/server test for a connection to a bad database and then a good database.

**Exercise 02\_09\_01 – Step 5**



1. Go back to ***RegisterIntern.php*** and let’s add a new user into the database. We will ***encrypt*** the password for security. Also notice the ***mysqli\_insert\_id()*** function, which returns us the ID generated by a query on an INSERT query to a table with an ***AUTO\_INCREMENT*** ID:  
    ***$TableName = "interns";*** if ($errors == 0) {  
    ***$first = stripslashes($\_POST['first']);  
    $last = stripslashes($\_POST['last']);  
    $SQLstring = "INSERT INTO $TableName" .  
    " (first, last, email, password\_md5)" .   
    "VALUES('$first', '$last', '$email', " .   
    "'" . md5($password) . "')";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (!$queryResult) {  
    ++$errors;  
    echo "<p>Unable to save your registration information   
    " . "error code: " . mysqli\_error($DBConnect) .   
    "</p>\n";  
    }  
    else {  
    $internID = mysqli\_insert\_id($DBConnect);  
    }*** echo "<p>Closing database \"$DBName\"   
    connection.</p>\n";
2. At the bottom of the script let’s add a positive ***response*** to the user on a successful Registration:  
    ***if ($errors == 0) {  
    $internName = $first . " " . $last;  
    echo "<p>Thank you, $internName. ";  
    echo "Your new Intern ID is <strong>" .   
    $internID . "</strong>.</p>\n";  
    }*** ?>  
   Give this a browser/server test and check to see if the record was inserted with MySQL Monitor and/or phpMyAdmin.
3. We should add code to guarantee that we cannot enter a ***duplicate*** user into the database. We can use the ***$email*** field to help with this. Directly below the ***$TableName*** declaration and before the ***INSERT*** code, enter code as follows:  
    $TableName = "interns";  
    ***if ($errors == 0) {  
    $SQLstring = "SELECT count(\*) FROM $TableName" .   
    " WHERE email='$email'";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if ($queryResult) {  
    $row = mysqli\_fetch\_row($queryResult);  
    if ($row[0] > 0) {  
    ++$errors;  
    echo "<p>The email address entered (" .   
    htmlentities($email) .  
    ") is already registered.</p>\n";  
    }  
    }  
    }***Give this a browser/server test for a connection to a bad database and then a good database.

**Exercise 02\_09\_01 – Step 6**



1. Let’s begin to build a Verify Login page that will make use of a ***hidden*** field to transmit ***state*** data. Create a new file called ***VerifyLogin.php***. Scaffold a basic HTML code layout into it. Complete our standard opening documentation in the ***<head>*** element. Make sure to have the ***modernizr*** <script> linked in. Set the <title> content to ***Verify Intern Login***:  
   ***<!doctype html>  
   <html>  
   <head>  
    <title>Verify Intern Login</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
   <body>  
   </body>  
   </html>***
2. Add the following text and elements to the document body:  
   <body>  
    ***<h1>College Internship</h1>  
    <h2>Verify Intern Login</h2>  
    <?php  
      
    ?>***</body>  
   Give this a browser/server test by submitting the ***Returning Intern Login*** form.
3. Let’s add some user instructions to handle any errors:  
    <?php  
    ***$errors = 0;  
    if ($errors > 0) {  
    echo "<p>Please use your browser's BACK button to   
    return to the form and fix the errors indicated.</p>\n";  
    }*** ?>  
   Give this a browser/server test with ***$errors*** set to 1.
4. Let’s establish our database connection. Enter the following code above the last code that traps the errors as follows:  
    $errors = 0;  
    ***$hostname = "localhost";  
    $username = "root";  
    $passwd = "";  
    $DBConnect = false;  
    $DBName = "internships";  
    if ($errors == 0) {  
    $DBConnect = mysqli\_connect($hostname, $username,   
    $passwd);  
    if (!$DBConnect) {  
    ++$errors;  
    echo "<p>Unable to connect to the database server   
    error code: " . mysqli\_connect\_error() . "</p>\n";  
    }  
    }*** if ($errors > 0) {  
   Give this a browser/server test for a connection to a bad host and then a good host.
5. We should close down the database connection, so let’s add some code at the bottom of the script:  
    ***if ($errors == 0) {  
    echo "<p>Closing database \"$DBName\"   
    connection.</p>\n"  
    mysqli\_close($DBConnect);  
    }***Give this a browser/server test.
6. Let’s add an ***else*** clause to the above connect code and ***select*** the database as follows:  
    ***else {  
    $result = mysqli\_select\_db($DBConnect, $DBName);  
    if (!$result) {  
    ++$errors;  
    echo "<p>Unable to select the database   
    \"$DBName\" error code: "   
    . mysqli\_error($DBConnect) . "</p>\n";  
    }  
    }***Give this a browser/server test for a connection to a bad database and then a good database.
7. Now we can verify if the ***email*** and ***password*** are in the ***interns*** table. Enter the following code above the ***errors*** message:  
    ***$TableName = "interns";  
    if ($errors == 0) {  
    $SQLstring = "SELECT internID, first, last FROM   
    $TableName" .   
    " WHERE email='" . stripslashes($\_POST['email']) . "'   
    and password\_md5='" .   
    md5(stripslashes($\_POST['password'])) . "'";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (mysqli\_num\_rows($queryResult) == 0) {  
    ++$errors;  
    echo "<p>The email address/password combination   
    entered is not valid.</p>\n";  
    }  
    }*** if ($errors > 0) {  
   Give this a browser/server test on invalid and valid combinations.
8. If we have a valid email/password pair, let’s retrieve the data from the ***query*** ***result*** ***set*** as an associative array. We need to pick off the unique internID, and then release the ***memory*** for the result set as follows:  
    if (mysqli\_num\_rows($queryResult) == 0) {  
    ++$errors;  
    echo "<p>The email address/password combination entered is   
    not valid.</p>\n";  
    }  
    ***else {  
    $row = mysqli\_fetch\_assoc($queryResult);  
    $internID = $row['internID'];  
    $internName = $row['first'] . " " . $row['last'];  
    mysqli\_free\_result($queryResult);  
    echo "<p>Welcome back, $internName!</p>\n";  
    }***Give this a browser/server test.
9. Now let’s build a submit form that will show if there were no errors. It will pass the internID on to an ***AvailableOpportunities.php*** app. The code should be right after we close the database:  
    if ($errors == 0) {  
    echo "<p>Closing database \"$DBName\" connection.</p>\n";  
    mysqli\_close($DBConnect);  
    ***echo "<form action='AvailableOpportunities.php'   
    method='post'>\n";  
    echo "<input type='hidden' name='internID'   
    value='$internID'>\n";  
    echo "<input type='submit' name='submit'   
    value='View Available Opportunities'>\n";  
    echo "</form>\n";*** }  
   Give this a browser/server test.
10. We will need to add the form to the ***RegisterIntern.php*** app. Enter the code into the closing part of the script as follows:  
     if ($errors == 0) {  
     $internName = $first . " " . $last  
     echo "<p>Thank you, $internName. ";  
     echo "Your new Intern ID is <strong>" .   
     $internID . "</strong>.</p>\n";  
     ***echo "<form action='AvailableOpportunities.php'   
     method='post'>\n";  
     echo "<input type='hidden' name='internID'   
     value= '$internID'>\n";  
     echo "<input type='submit' name='submit'   
     value='View Available Opportunities'>\n";  
     echo "</form>\n";*** }  
    Give this a browser/server test.

**Exercise 02\_09\_01 – Step 7**



1. Let’s begin to build an Available Opportunities page that will make use of the ***state*** data transmitted by the ***hidden*** field. Create a new file called ***AvailableOpportunities.php***. Scaffold a basic HTML code layout into it. Complete our standard opening documentation in the ***<head>*** element. Make sure to have the ***modernizr*** <script> linked in. Set the <title> content to ***Available Opportunities***:  
   ***<!doctype html>  
   <html>  
   <head>  
    <title>Available Opportunities</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
   <body>  
   </body>  
   </html>***
2. Add the following text and elements to the document body:  
   <body>  
    ***<h1>College Internship</h1>  
    <h2>Available Opportunities</h2>  
    <?php  
      
    ?>***</body>  
   Give this a browser/server test by submitting the ***Verify Intern Login*** form.
3. At the top of the script, let’s use the ***$\_REQUEST*** superglobal to see if we got the state information, the ***internID***, with some debug code:  
    <?php  
    ***if (isset($\_REQUEST['internID'])) {  
    $internID = $\_REQUEST['internID'];  
    }  
    else {  
    $internID = -1;  
    }  
    // debug  
    echo "\$internID: $internID\n";*** ?>  
   Give this a browser/server test by entering the app directly, then by entering thru a valid login. When it works, remove the debug code.
4. At the bottom of the script, let’s copy in the code from ***VerifyLogin.php*** that connects to the database and selects the ***internships*** db. We will also put in the code that closes down the db:  
    ***$errors = 0;  
    $hostname = "localhost";  
    $username = "root";  
    $passwd = "";  
    $DBConnect = false;  
    $DBName = "internships";  
    if ($errors == 0) {  
    $DBConnect = mysqli\_connect($hostname, $username,   
    $passwd);  
    if (!$DBConnect) {  
    ++$errors;  
    echo "<p>Unable to connect to the database server   
    error code: " . mysqli\_connect\_error() . "</p>\n";  
    }  
    else {  
    $result = mysqli\_select\_db($DBConnect, $DBName);  
    if (!$result) {  
    ++$errors;  
    echo "<p>Unable to select the database   
    \"$DBName\" error code: " .   
    mysqli\_error($DBConnect) . "</p>\n";  
    }  
    }  
    }  
    if ($errors == 0) {  
    echo "<p>Closing database \"$DBName\"   
    connection.</p>\n";  
    mysqli\_close($DBConnect);  
    }*** ?>  
   Give this a browser/server test.
5. We can now add the code that checks to see if the user is in the ***interns*** table, before the ***if*** that closes the connection:  
    ***$TableName = "interns";  
    if ($errors == 0) {  
    $SQLstring = "SELECT \* FROM $TableName " .   
    " WHERE internID='$internID'";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (!$queryResult) {  
    ++$errors;  
    echo "<p>Unable to execute the query, error code: " .   
    mysqli\_errno($DBConnect) . ": " .   
    mysqli\_error($DBConnect) . "</p>\n";  
    }  
    else {  
    if (mysqli\_num\_rows($queryResult) == 0) {  
    ++$errors;  
    echo "<p>Invalid Intern ID!</p>\n";  
    }  
    }  
    }*** if ($errors == 0) {  
   Give this a browser/server test by entering the app directly, then by entering through a valid login.
6. We can now add the code that returns the user’s information from the ***interns*** table, directly below the last code, and before the ***if*** that closes the connection. Add some debug code:  
    ***if ($errors == 0) {  
    $row = mysqli\_fetch\_assoc($queryResult);  
    $internName = $row['first'] . " " . $row['last'];  
    }  
    else {  
    $internName = "";  
    }  
    // debug  
    echo "\$internName: $internName\n";***  
    if ($errors == 0) {  
   Give this a browser/server test. When it works, remove the debug code.

**Exercise 02\_09\_01 – Step 8**



1. Return to the ***AvailableOpportunities.php*** file. Before we can display the available opportunities for a user, we must first ascertain if the user has been approved for any opportunities, so they will not be displayed as available. Add the following code before the ***if*** that closes the connection. Other than syntax, we will debug this later because we have not yet built the code to assign opportunities to users:  
    ***$TableName = "assigned\_opportunities";  
    if ($errors == 0) {  
    $SQLstring = "SELECT COUNT(opportunityID) FROM   
    $TableName" .   
    " WHERE internID='$internID'" .  
    " AND date\_approved IS NOT NULL";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (mysqli\_num\_rows($queryResult) > 0) {  
    $row = mysqli\_fetch\_row($queryResult);  
    $approvedOpportunities = $row[0];  
    mysqli\_free\_result($queryResult);  
    }  
    }*** if ($errors == 0) {  
   Give this a browser/server test for syntax.
2. Now we will retrieve a list of the assigned opportunities that has been ***selected*** for the ***current*** user. Add the following code before the ***if*** that closes the connection. Other than syntax, we will debug this later because we have not yet built the code to assign opportunities to users:  
    ***$selectedOpportunities = array();  
    $SQLstring = "SELECT opportunityID FROM $TableName" .   
    " WHERE internID='$internID'";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (mysqli\_num\_rows($queryResult) > 0) {  
    while (($row = mysqli\_fetch\_row($queryResult)) !=   
    false) {  
    $selectedOpportunities[] = $row[0];  
    }  
    mysqli\_free\_result($queryResult);  
    }*** if ($errors == 0) {  
   Give this a browser/server test for syntax.
3. Now we will retrieve a list of the assigned opportunities that has been ***approved*** for ***any*** user. Add the following code before the ***if*** that closes the connection. Other than syntax, we will debug this later because we have not yet built the code to assign opportunities to users:  
    ***$assignedOpportunities = array();  
    $SQLstring = "SELECT opportunityID FROM $TableName" .   
    " WHERE date\_approved IS NOT NULL";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (mysqli\_num\_rows($queryResult) > 0) {  
    while (($row = mysqli\_fetch\_row($queryResult)) !==   
    false) {  
    $assignedOpportunities[] = $row[0];  
    mysqli\_free\_result($queryResult);  
    }  
    }*** if ($errors == 0) {  
   Give this a browser/server test for syntax.
4. We will retrieve a list of opportunities from the ***opportunities*** table. Add the following code before the ***if*** that closes the connection, with some debug to see the number of rows inn the result set:  
    ***$TableName = "opportunities";  
    $opportunities = array();  
    $SQLstring = "SELECT opportunityID, company, city," .   
    " start\_date, end\_date, position, description" .   
    " FROM $TableName";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    //debug  
    echo mysqli\_num\_rows($queryResult) . "<br>\n";*** if ($errors == 0) {  
   Give this a browser/server test, then remove the debug code.
5. Now we will retrieve all of the rows in the opportunities table and explode each record into an associative array as follows. Add debug code to examine the resulting arrays:  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    ***if (mysqli\_num\_rows($queryResult) > 0) {  
    while (($row = mysqli\_fetch\_assoc($queryResult)) !=   
    false) {  
    $opportunities[] = $row;  
    }  
    }  
    mysqli\_free\_result($queryResult);  
    echo "<pre>\n";  
    echo print\_r($opportunities);  
    echo "</pre>\n";*** if ($errors == 0) {  
   Give this a browser/server test, then remove the debug code.
6. Let’s start to build out a table for the display. Scaffold the following code at the ***end*** of the script. We can also add a Log Out hyperlink at the end:  
    ***echo "<table border='1' width='100%'>\n";  
    echo "</table>\n";  
    echo "<p><a href='InternLogin.php'>Log   
    Out</a></p>\n";*** ?>  
   Give this a browser/server test.
7. Now let’s build the table headings:  
    echo "<table border='1' width='100%'>\n";  
    ***echo "<tr>\n";  
    echo "<th style='background-color: cyan'>Company</th>\n";  
    echo "<th style='background-color: cyan'>City</th>\n";  
    echo "<th style='background-color: cyan'>Start   
    Date</th>\n";  
    echo "<th style='background-color: cyan'>End   
    Date</th>\n";  
    echo "<th style='background-color:   
    cyan'>Position</th>\n";  
    echo "<th style='background-color:   
    cyan'>Description</th>\n";  
    echo "<th style='background-color: cyan'>Status</th>\n";  
    echo "</tr>\n";*** echo "</table>\n";  
   Give this a browser/server test.
8. Now let’s ***foreach*** through the ***$opportunities*** array and create our table data. We will skip any that are in the ***$assignedOpportunities*** array. We will also build a hyperlink to ***RequestOpportunity.php***, which we will construct later:  
    echo "</tr>\n";  
    ***foreach ($opportunities as $opportunity) {  
    if (!in\_array($opportunity['opportunityID'],   
    $assignedOpportunities)) {  
    echo "<tr>\n";  
    echo "<td>" . htmlentities($opportunity['company']) .   
    "</td>\n";  
    echo "<td>" . htmlentities($opportunity['city']) .   
    "</td>\n";  
    echo "<td>" . htmlentities($opportunity['start\_date']) .   
    "</td>\n";  
    echo "<td>" . htmlentities($opportunity['end\_date']) .   
    "</td>\n";  
    echo "<td>" . htmlentities($opportunity['position']) .   
    "</td>\n";  
    echo "<td>" . htmlentities($opportunity['description'])   
    . "</td>\n";  
    echo "</tr>\n";  
    }  
    }*** echo "</table>\n";  
   Give this a browser/server test.
9. The last task on the display is to figure out the status of the opportunity and to provide a hyperlink to it if it is available:  
    ***echo "<td>";  
    if (in\_array($opportunity['opportunityID'],   
    $selectedOpportunities)) {  
    echo "Selected";  
    }  
    else if ($approvedOpportunities > 0) {  
    echo "Open";  
    }  
    else {  
    echo "<a href='RequestOpportunity.php?" .   
    "internID=$internID&" .   
    "opportunityID=" .   
    $opportunity['opportunityID'] .   
    "'>Available</a>";  
    }  
    echo "</td>\n";*** echo "</tr>\n";  
   Give this a browser/server test.
10. Return to the VerifyLogin.php file and comment out all of the form code:  
    ***// echo "<p>Closing database \"$DBName\"   
     connection.</p>\n";  
    // mysqli\_close($DBConnect);  
    // echo "<form action='AvailableOpportunities.php'   
     method='post'>\n";  
    // echo "<input type='hidden' name='internID' value=   
     '$internID'>\n";  
    // echo "<input type='submit' name='submit' value='View   
     Available Opportunities'>\n";  
    // echo "</form>\n";***
11. After the commented code, enter the following code to replace the form with a hyperlink containing a ***query*** ***string***:  
    // echo "</form>\n";  
     ***echo "<p><a href='AvailableOpportunities.php?" .   
     "internID=$internID'>Available" .   
     " Opportunities</a></p>\n";***

**Exercise 02\_09\_01 – Step 9**



1. Return to the ***RegisterIntern.php*** file. We will be setting a temporary cookie in this application. Therefore, we need to make sure that it will happen before any output to the Web browser. Cut and paste the entire existing PHP script to above the <!DOCTYPE> declaration.
2. As the first line in the script, code the following variable declaration:  
   <?php  
   ***$body = "";***
3. Replace all of the ***echo*** statements with statements as follows:  
    ***echo*** "<p>You need to enter an e-mail address.</p>\n";  
   Becomes:  
    ***$body .=*** "<p>You need to enter an e-mail address.</p>\n";
4. Within the ***<body>*** tags, code the following PHP script:  
   <body>  
    <h1>College Internship</h1>  
    <h2>Intern Registrations</h2>  
    ***<?php  
    echo $body;  
    ?>***</body>  
   Give this a browser/server test.
5. Just before the **mysqli\_close()** statement, set the cookie as follows:  
    ***setcookie("internID", $internID);*** mysqli\_close($DBConnect);
6. Open ***AvailableOpportunities.php*** and place the following debug code at the top of the script:  
    <?php  
    ***echo "<pre>\n";  
    print\_r($\_COOKIE);  
    echo "</pre>\n";***Give this a browser/server test. Register a new intern, then got to View Available Opportunities. We should see our new temporary cookie in the ***$\_COOKIE*** superglobal. Remove the debug code.

**Exercise 02\_09\_01 – Step 10**



1. Let’s begin to build a Request Opportunity page. Create a new file called ***RequestOpportunity.php***. Scaffold a basic HTML code layout into it. Complete our standard opening documentation in the ***<head>*** element. Make sure to have the ***modernizr*** <script> linked in. Set the <title> content to ***Request Opportunity***:  
   ***<!doctype html>  
   <html>  
   <head>  
    <title>Request Opportunity</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
   <body>  
   </body>  
   </html>***
2. Add the following code to the document ***<body>***:  
   <body>  
    ***<h1>College Internship</h1>  
    <h2>Opportunity Requested</h2>  
    <?php  
    echo $body;  
    ?>***</body>
3. Because we will be setting cookies, as the first line in the script, above the <!DOCTYPE>, code the following script and variable declarations:  
   ***<?php  
   $body = "";  
   $errors = 0;  
   $internID = 0;  
   ?>***<!doctype html>  
   Give this a browser/server test.
4. At the bottom of the script, enter the following code to validate the submitted ***internID***:  
   $internID = 0;  
   ***if (isset($\_GET['internID'])) {  
    $internID = $\_GET['internID'];  
   }  
   else {  
    ++$errors;  
    $body .= "<p>You have not logged in or registered." .   
    " Please return to the " .   
    " <a href='InternLogin.php'>" .   
    "Registration / Login Page</a>.</p>";  
   }***Give this a browser/server test. Enter the application directly to cause an error. Then use the hyperlink to get back and go through a normal login.
5. At the bottom of the script, enter the following code to validate the submitted ***opportunityID***: To debug, comment out the previous validation code  
   ***//}  
   if ($errors == 0) {  
    if (isset($\_GET['opportunityID'])) {  
    $opportunityID = $\_GET['opportunityID'];  
    }  
    else {  
    ++$errors;  
    $body .= "<p>You have not selected an opportunity." .   
    " Please return to the " .   
    " <a href='AvailableOpportunities.php?" .   
    "internID=$internID'>" .   
    "Opportunities Page</a>.</p>";  
    }  
   }***?>To debug, comment out the previous validation code. Give this a browser/server test. Enter the application directly to cause an error. Then use the hyperlink to get back and go through an entry from the AvailableOpportunities.php page. Uncomment the code, and go through a normal login sequence.
6. At the bottom of the script, let’s copy in the code from ***VerifyLogin.php*** that connects to the database and selects the ***internships*** db. We will also put in the code that closes down the db:  
   ***$hostname = "localhost";  
   $username = "root";  
   $passwd = "";  
   $DBConnect = false;  
   $DBName = "internships";  
   if ($errors == 0) {  
    $DBConnect = mysqli\_connect($hostname, $username,   
    $passwd);  
    if (!$DBConnect) {  
    ++$errors;  
    echo "<p>Unable to connect to the database server error   
    code: " . mysqli\_connect\_error() . "</p>\n";  
    }  
    else {  
    $result = mysqli\_select\_db($DBConnect, $DBName);  
    if (!$result) {  
    ++$errors;  
    echo "<p>Unable to select the database \"$DBName\"   
    error code: " . mysqli\_error($DBConnect) . "</p>\n";  
    }  
    }  
   }  
   if ($DBConnect) {  
    echo "<p>Closing database \"$DBName\"   
    connection.</p>\n";  
    mysqli\_close($DBConnect);  
   }****?>*Give this a browser/server test.
7. We will now add the code at the end of the script, before the ***if*** that closes the connection, which will mark the opportunity as selected in the ***assigned\_opportunities*** table. First, we will use the PHP date() function to format a couple of date strings. The first one will be user friendly for display. The second one will build a string in MySQL format. Put in some debug code to display the strings:  
   ***$displayDate = date("l, F j, Y, g:i A");  
   echo "\$displayDate: $displayDate<br>";  
   $dbDate = date("Y-m-d H:i:s");  
   echo "\$dbDate: $dbDate<br>";***if ($errors == 0) {  
   Give this a browser/server test and remove the debug
8. Now let’s do the database work directly below the previous code and before the ***if*** that closes the connection:  
   $dbDate = date("Y-m-d H:i:s");  
   ***if ($errors == 0) {  
    $TableName = "assigned\_opportunities";  
    $SQLstring = "INSERT INTO $TableName" .   
    " (opportunityID, internID, date\_selected)" .   
    " VALUES ($opportunityID, $internID, '$dbDate')";  
    $queryResult = mysqli\_query($DBConnect, $SQLstring);  
    if (!$queryResult) {  
    ++$errors;  
    $body .= "<p>Unable to execute the query, " .   
    "error code: " . mysqli\_errno($DBConnect) . ": " .   
    mysqli\_error($DBConnect) . "</p>\n";  
    }  
    else {  
    $body .= "<p>Your results for opportunity #" .   
    " $opportunityID have been entered on" .   
    " $displayDate.</p>\n";  
    }  
   }***Give this a browser/server test. When this is working, check the data in ***phpMyAdmin***. Delete the new record.
9. Add the following code to the end of the script section to provide a link back to the Available Opportunities page if the ***internID*** is valid, and the Registration/Login page if not valid:  
   ***if ($internID > 0) {  
    $body .= "<p>Return to the " .   
    "<a href='AvailableOpportunities.php?" .   
    "internID=$internID'>Available Opportunities" .   
    "</a> page.</p>\n";  
   }  
   else {  
    $body .= "<p>Please " .   
    "<a href='InternLogin.php'>" .   
    "Register or Log In" .   
    "</a> to use this page.</p>\n";  
   }***?>  
   Give this a browser/server test. Enter the application directly to cause an error. Then use the hyperlink to get back to Login / Registration. Now go through a normal sequence.
10. Now we will set a persistent cookie that will last for 1 week. Note the use of the ***urlencode()*** function which will convert anything in the ***$displayDate*** string to valid URL characters. Enter the following code at the bottom of the script:  
    ***if ($errors == 0) {  
     echo "Setting cookie<br>";  
     setcookie("LastRequestDate",   
     urlencode($displayDate),   
     time()+60\*60\*24\*7);  
    }***  
    ?>  
    Turn on Developer tools in the browser and go to the ***Application*** tab. Open up the ***Cookies*** area. Click on the host, and the persistent cookie should be there.

**Exercise 02\_09\_01 – Step 11**



1. Return to the ***AvailableOpportunities.php*** document. Add the following code to read the persistent cookie. It should be right after the ***internID*** is read from $\_REQUEST:  
    if (isset($\_REQUEST['internID'])) {  
    $internID = $\_REQUEST['internID'];  
    }  
    else {  
    $internID = -1;  
    }  
    ***if (isset($\_COOKIE['LastRequestDate'])) {  
    $lastRequestDate =urldecode(  
    $\_COOKIE['LastRequestDate']);  
    }  
    else {  
    $lastRequestDate = "";  
    }***
2. Add the following statements directly above the code that displays the opportunities ***<table>***:  
    ***if (!empty($lastRequestDate)) {  
    echo "<p>You last requested an internship" .   
    " opportunity on $lastRequestDate.</p>\n";  
    }*** echo "<table border='1' width='100%'>\n";  
   Go through a normal Login, select an available opportunity, and use the link back to the ***Available Opportunities*** page. You should see the message if the cookie was read. Also notice that the Opportunity now has a Status of ***Selected***.

**Exercise 02\_09\_01 – Step 12**



1. Return to the ***InternLogin.php*** document. Insert PHP script delimiters with the following code above the ***<!DOCTYPE>*** declaration to start a session. Use some debug to check the session ID:  
   ***<?php  
   session\_start();  
   echo "Session id: " . session\_id() . "<br>\n";  
   ?>***<!doctype html>  
   Give this a browser/server test, then remove the debug code.
2. Modify the ***action*** attribute of the two forms so they pass the session ID in a ***query*** ***string***:  
    ***<form action="RegisterIntern.php?PHPSESSID=  
    <?php echo session\_id(); ?>" method="post">  
     
    <form action="VerifyLogin.php?PHPSESSID=  
    <?php echo session\_id(); ?>" method="post">***Go through a normal Login, and check the URL on the Verify Intern Login page to make sure the query string is correct.

**Exercise 02\_09\_01 – Step 13**



1. Return to the ***RegisterIntern.php*** document. Add a session\_start() call to the beginning of the top PHP script:  
   <?php ***session\_start();***
2. Modify the ***$internID*** assignment so it stores the value returned by the ***mysqli\_insert\_id()*** function into the ***$\_SESSION*** superglobal. Comment out the ***setcookie()*** function call:  
   ***//*** $internID = mysqli\_insert\_id($DBConnect); ***$\_SESSION['internID'] = mysqli\_insert\_id($DBConnect);  
     
   // setcookie("internID", $internID);***
3. Modify the ***<p>*** element that displays the ***Intern ID*** to use the ***$\_SESSION*** superglobal:  
    $body .= "Your new Intern ID is <strong>" .   
    ***$\_SESSION['internID']*** . "</strong>.</p>\n";
4. Comment out the entire <form> code that uses the ***hidden*** input to store the intern ID:  
   ***//*** $body .= "<form action='AvailableOpportunities.php'   
    method='post'>\n";  
   ***//*** $body .= "<input type='hidden' name='internID'   
    value='$internID'>\n";  
   ***//*** $body .= "<input type='submit' name='submit' value='View   
    Available Opportunities'>\n";  
   ***//*** $body .= "</form>\n";
5. Replace the commented code with a ***hyperlink*** that uses the ***session*** ***ID***, as follows:  
    $body .= "Your new Intern ID is <strong>" .   
    $\_SESSION['internID'] . "</strong>.</p>\n";  
    ***$body .= "<p><a href='AvailableOpportunities.php?" .   
    "PHPSESSID=" . session\_id() . "'>" .   
    "View Available Opportunities</a></p>\n";***Give this a browser/server test. Register a new intern. Make sure that the ***RegisterIntern.php*** page has the proper ***query*** ***string*** and the new ***hyperlink***. Click the link and go to ***AvailableOpportunities.php***. The query string should be correct in the URL. There will be an error about ***invalid*** internID, which we will fix.
6. Return to the ***VerifyLogin.php*** page. Insert PHP script delimiters with session\_start() code above the ***<!DOCTYPE>*** declaration to start a session:  
   ***<?php  
   session\_start();  
   ?>***<!doctype html>
7. Comment out the ***$internID*** assignment statement and replace it with code that places it into the ***$\_SESSION*** superglobal, as follows:  
   **//** $internID = $row['internID'];  
    ***$\_SESSION['internID'] = $row['internID']***
8. Comment out the hyperlink that passes the ***$internID*** with a hyperlink that passes the ***session ID***, as follows  
   ***//*** echo "<p><a href='AvailableOpportunities.php?" .   
   ***//*** "internID=$internID'>Available" .   
   ***//*** " Opportunities</a></p>\n";  
    ***echo "<p><a href='AvailableOpportunities.php?" .   
    "PHPSESSID=" . session\_id() . "'>" .  
    "Available Opportunities</a></p>\n";***  
   Give this a browser/server test. Login an intern. Make sure that the ***VerifyLogin.php*** page has the proper ***query*** ***string*** and the new ***hyperlink***. Click the link and go to ***AvailableOpportunities.php***. The query string should be correct in the URL. There will be an error about ***invalid*** internID, which we will fix.
9. Now let’s return to the ***AvailableOpportunities.php*** page and code it to use the Intern ID from the ***$\_SESSION*** superglobal. Insert PHP script delimiters with session\_start() code above the ***<!DOCTYPE>*** declaration to start a session:  
   ***<?php  
   session\_start();  
   ?>***<!doctype html>
10. Comment out the code the uses the $\_REQUEST superglobal to retrieve the Intern ID as follows:  
    ***//*** if (isset($\_REQUEST['internID'])) {  
    ***//*** $internID = $\_REQUEST['internID'];  
    ***//*** }  
    ***//*** else {  
    ***//*** $internID = -1;  
    ***//*** }
11. Modify the first SQL query which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***$SQLstring = "SELECT \* FROM $TableName" .   
     " WHERE internID='" . $\_SESSION['internID'] .   
     "'";***Use a browser/server test to insure that there are no syntax errors and the first invalid variable error has gone away.
12. Modify the second SQL query which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***$SQLstring = "SELECT COUNT(opportunityID)" .   
     " FROM $TableName" .   
     " WHERE internID='" . $\_SESSION['internID'] . "'" .   
     " AND date\_approved IS NOT NULL";***Use a browser/server test to insure that there are no syntax errors and the second invalid variable error has gone away.
13. Modify the third SQL query which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***$SQLstring = "SELECT opportunityID FROM $TableName" .   
     " WHERE internID='" . $\_SESSION['internID'] . "'";***Use a browser/server test to insure that there are no syntax errors and the third invalid variable error has gone away.
14. Modify the hyperlink to ***RequestOpportunity.php*** which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***echo "<a href='RequestOpportunity.php?" .   
     "PHPSESSID=" . session\_id() .   
     "&opportunityID=" .   
     $opportunity['opportunityID'] .   
     "'>Available</a>";***Use a browser/server test to insure that there are no syntax errors and the hyperlink errors have gone away.
15. Now we can modify the ***RequestOpportunity.php*** page to use the Session ID. Begin by placing a session\_start() call at the top of the first PHP script:  
    <?php  
    ***session\_start();***
16. Comment out the code the uses the $\_GET superglobal to retrieve the Intern ID, and insert code to use the $\_SESSION superglobal, as follows:  
    ***//***$internID = 0; ***//***if (isset($\_GET['internID'])) {  
    ***//*** $internID = $\_GET['internID'];  
    ***//***}  
    ***//***else {  
    ***if (!isset($\_SESSION['internID'])) {***Do a browser/server test. Login and go all the way to selecting an opportunity. The code should now throw undefined variable errors and not insert a record into the database. We will fix that.
17. Modify the first hyperlink to ***AvailableOpportunities.php*** which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***$body .= "<p>You have not selected an opportunity." .   
     " Please return to the " .   
     " <a href='AvailableOpportunities.php?" .   
     "PHPSESSID=" . session\_id() . "’>" .   
     "Opportunities Page</a>.</p>";***Use a browser/server test to insure that no additional errors have been generated.
18. Modify the second hyperlink to ***AvailableOpportunities.php*** which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***$body .= "<p>Return to the " .   
     "<a href='AvailableOpportunities.php?" .   
     "PHPSESSID=" . session\_id() . "’>" .   
     "Available Opportunities</a> page.</p>\n";***Use a browser/server test to insure that no additional errors have been generated.
19. Modify the **INSERT** query which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
     ***$SQLstring = "INSERT INTO $TableName" .   
     " (opportunityID, internID, date\_selected)" .   
     " VALUES ($opportunityID, " .   
     $\_SESSION['internID'] . ", '$dbDate')";***Use a browser/server test to insure that the first undefined variable error has gone away, and a record is inserted. Confirm that in phpMyAdmin, and delete the record.
20. Modify the ***if*** statement which uses the ***$internID*** variable to use the ***$\_SESSION*** superglobal as follows:  
    ***if ($\_SESSION['internID'] > 0) {*** $body .= "<p>Return to the " .   
     "<a href='AvailableOpportunities.php?" .   
     "PHPSESSID=" . session\_id() . "'>" .   
     "Available Opportunities</a> page.</p>\n";  
    }  
    Use a browser/server test to insure that the all undefined variable errors have gone away, a record is inserted and there is a link back to the AvailableOpportunities.php page that works. Confirm that in phpMyAdmin, and delete the record.
21. Now let’s clean up the session. Return to the ***InternLogin.php*** code. After the session\_start() call, place in the following code to initialize the $\_SESSION superglobal and to delete the session, as follows:  
    <?php  
    session\_start();  
    ***$\_SESSION = array();  
    session\_destroy();***?>  
    Login as a user. At the Verify Login page, check the URL to make sure that the session ID has been wiped out.