Web Programming 2
Packet #5: A Couple of Projects

Objectives: Apply the material that we learned in the previous packets to common tasks.

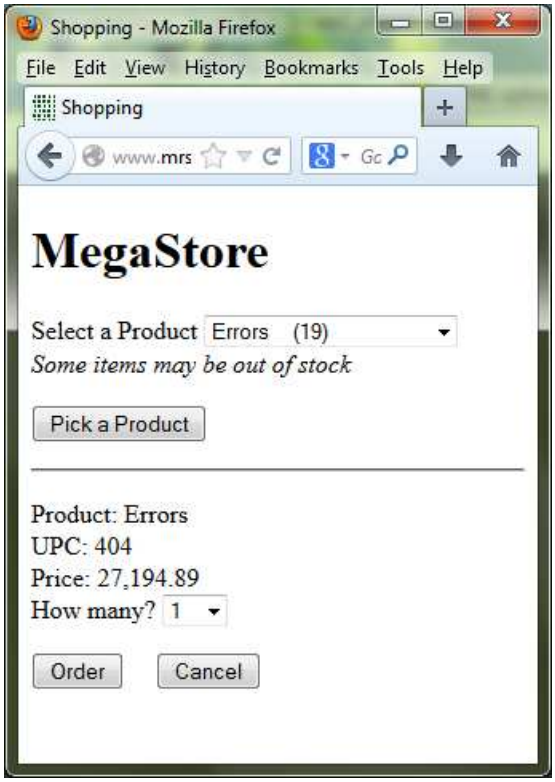
Project 1. Simple shopping site. Visit Johnny Demo's site to see how a great guy like Johnny did this exercise.

Step 1. When a user first visits the site, they should see a drop down list that has been "populated" by the information in the products table.

The number next to each item is the current inventory. If the number is zero then that item should be disabled so that the user cannot select it. The value (in the option tag) associated with each item should be the appropriate UPC number. No price information is available.

The form should use the GET method because it is (1) this action does not change anything in the database and (2) it is a small, non-confidential amount of data.

Code and test this portion before moving on. Clicking on the button should simply refresh the page and do nothing else.

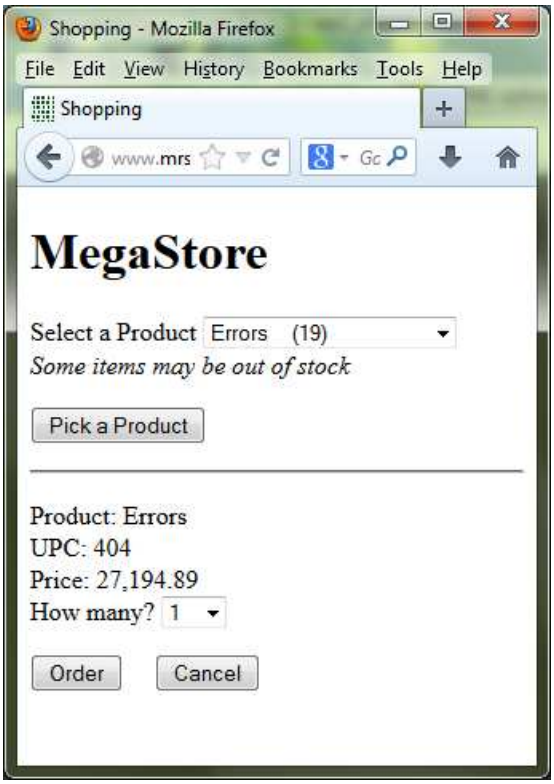


Step 2. If the user clicks the button then another form should appear below the first (as shown in the second figure). In this example the user had selected "Errors." The drop-down list next to "How many?" will normally contain numbers from 1 to 20 unless the number of items in inventory is less than 20. In this example the number of errors is 19 so the drop down list goes from 1 to 19.

The second form should use the POST method because if the user places an order, the contents of the database will change. The action is to simple refresh this page.

Code and test this portion. The bottom form should only appear if the "Pick a Product" button has been clicked. The product information displayed in the bottom should reflect what the user selected. The Order and Cancel buttons simply refresh the page and cause the bottom form to disappear.

Step 3. When the user clicks the Order button, we want to display a screen like the one shown below.



Notice:

- the number of items in the drop down list has decreased.
- bottom form has disappeared.
- the message displays all the relevant information.

When the user clicks the Order button, we will send the following information back to the server:

- the upc value of the product because this is the primary key for the PRODUCTS table. This will allow us to identify and update one unique record in the database.
- the number of items they want to purchase.
- the unit price of the item being ordered.
- the current number in inventory (before processing the order). We will need this value when we update the number in the database.

To send the data back to the server we will use input elements of type hidden. For example, the following statement adds an element to the form but it is not displayed in the form (though it can easily be seen if you view the source code).

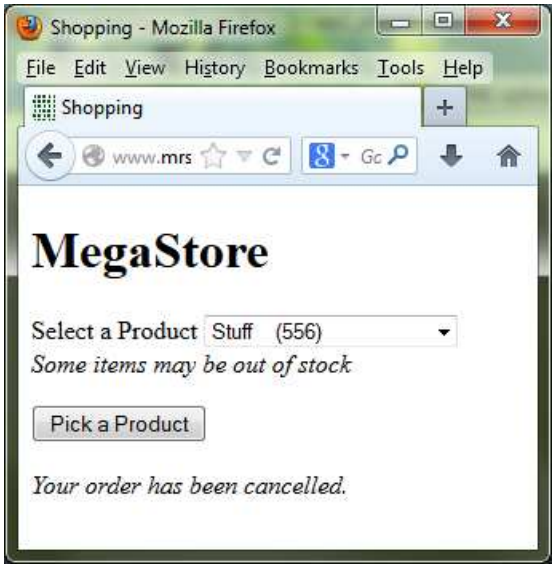
```
<input type= "hidden"  value= "123"  name="upc">
```

The php code to update the table should come before the code that generates the product list. It can follow this format.

```
if ( isset ( $_POST["btnOrder"] ) ){
        Read the data from the second form
        connect to the database and update the table.
        close the connection to the database
}
```

Continued on the next page.

Add some php code so that if the user clicks the Cancel button, the bottom form disappears the message "Your order has been cancelled." appears.



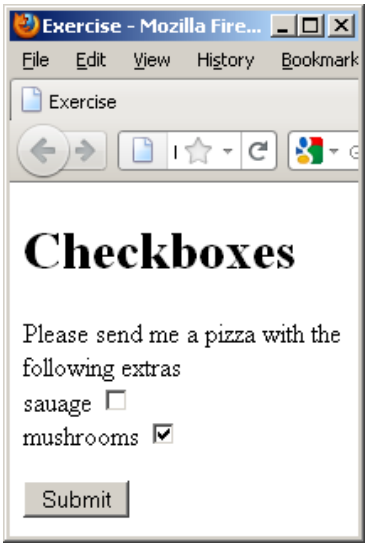
General reminders:

- Use number_format function to properly format the numbers.
- Disable any products in the product list where the inventory is zero.
- When the user selects a product and clicks the "Pick a Product" button, the selected product should be the displayed item in the drop down list. If they cancel a purchase then the first item will be displayed.
- Check that the source code (as viewed from the client's browser) does not contain any HTML errors.

IMPORTANT DISCLAIMER! This is just a simple exercise. We have not done any input checking on the server side. There are other important details that we are not addressing, besides security issues.
****

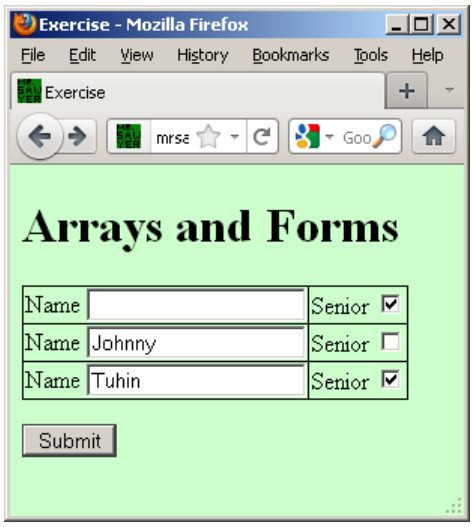
Checkboxes in HTML. A checkbox is an simple element to include in a form. Here is an example:

```
<!DOCTYPE HTML><html><head><meta charset="utf-8">
<title>Exercise</title></head><body>
<h1>Checkboxes</h1>
<form method="get" action="some_page.php">
<p>Please send me a pizza with the following extras<br>
sauage <input type="checkbox" name="chk_sausage"
                value="sausage"><br>
mushrooms <input type="checkbox" name="chk_mush"
                value="mushrooms"></p>
<p><input type="submit" value="Submit"></p>
</form>
</body></html>
```



The user can check both boxes, only one box or not click either of them. However, the value of a checkbox is only sent to the server if the user selected it.

Arrays in HTML. Well, actually there are no arrays in HTML. However, we can name elements of a form in such a way that PHP will treat those elements as an array.



Exercise 2. Notice that the names of the form elements include "[]" at the end. This allows PHP to treat like arrays. Copy, upload, and run this exercise.

```
<!DOCTYPE HTML><html><head>
<meta charset="utf-8"><title>Exercise</title>
<style type="text/css">
        table{border-collapse:collapse}
        td{border: solid 1px black }
</style></head><body>
<h1>Arrays and Forms</h1>
<form method="get" action="ex2.php">
<table><tr>
<td>Name <input type="text" name="name[]" ></td>
<td>Senior <input type="checkbox" name="senior[]" value="12"></td>
</tr><tr>
<td>Name <input type="text" name="name[]" ></td>
<td>Senior <input type="checkbox" name="senior[]" value="12"></td>
</tr><tr>
<td>Name <input type="text" name="name[]" ></td>
<td>Senior <input type="checkbox" name="senior[]" value="12"></td>
</tr></table>
<p><input type="submit" value="Submit"></p>
</form>
</body></html>
```

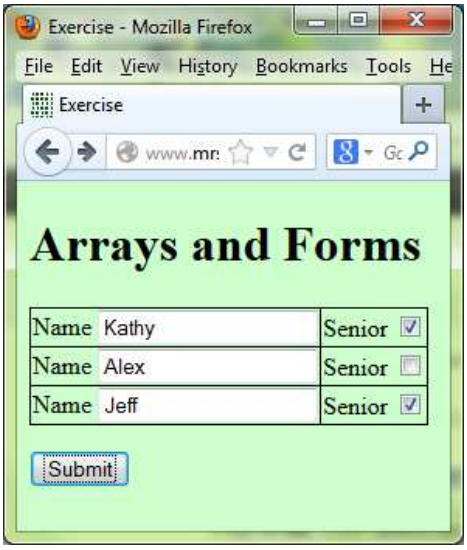
Here is the code for ex2.php.


```
<!DOCTYPE HTML><html><head><meta charset="utf-8">
<title>Exercise</title></head><body><h1>Results</h1>
<?php
$names = $_GET[ "name" ];
$seniors = $_GET[ "senior" ];
for ( $i = 0; $i < count( $names ); $i++ ){
        print "Name #$i is ".$names[$i]."<br>";
}
print "<br>";
for ( $i = 0; $i < count( $seniors ); $i++ ){
        print "Senior #$i is ".$seniors[$i]."<br>";
}
if ( count( $seniors ) == 0 )
        print "<p>There were no seniors.</p>";
?>
</body></html>
```

Things to notice about the exercise/code on the previous page.

- We can use just one $_GET statement to read in all the names and another to read in all checkbox values.
- If a textbox is left empty in the form, it has no effect on the $names array - its length is still three.
- There's a problem. In the form the first and last students were seniors. The PHP page implies that the first two students are seniors. The reason is that the array contains only those values that were selected. The next exercise shows a way to solve this.

Exercise 3. Here is the code for the html page. The page looks exactly like the one in exercise 2 but the source code for the form contains a significant difference - the names of the checkboxes include index numbers.




```
<h1>Arrays and Forms</h1>
<form method="get" action="ex3.php">
<table>
<tr>
<td>Name <input type="text" name="name[]" ></td>
<td>Senior <input type="checkbox" name="senior[0]"
                    value="12"></td>
</tr>
<tr>
<td>Name <input type="text" name="name[]" ></td>
<td>Senior <input type="checkbox" name="senior[1]" value="12"></td>
</tr>
<tr>
<td>Name <input type="text" name="name[]" ></td>
<td>Senior <input type="checkbox" name="senior[2]" value="12"></td>
</tr>
</table>
<p><input type="submit" value="Submit"></p>
</form>
```

Write the PHP page so that results similar to this can be displayed.

Exercise 4. This is a rather lengthy assignment. There is a table, named ACCOUNTS, with three fields: username, password, bio, and admin. The bio field allows the user to enter text up to 65,000 characters. The admin indicates if the user has admin privileges (i.e. the ability to create and delete new records).

The name of the database is mrsawye1_comments. For the login and members pages use

_____ for the user name and _____ for the password.

You will write four php pages. You will also be using sessions so be sure to start each page with

```
session_start();
```

Step 1. The first is the login page and it should look something like the one shown. When the user clicks the Login button, the page should call itself.

If the login button was clicked you should read the username, password, and whether admin was checked or not. Use code similar to this:

```
$sql = "SELECT username, password, admin
        FROM ACCOUNTS
        WHERE  username='$user' and
               password='$pass'";
$result = mysqli_query($link, $sql  ) or exit( mysqli_error($link) );
$num = mysqli_affected_rows($link);
$row = mysqli_fetch_array( $result, MYSQLI_NUM );
```

If the number of affected rows equals zero then either the username and/or the password are invalid. Stay on the page and display an error message.

If their username and password are in the database and they did not check the Admin box then save the username and password in $_SESSION and redirect them to the members page.

If they checked the Admin box but do not have admin privileges, display an error message and stay on the login page.

If they checked the Admin box and do have admin privileges, then redirect them to the admin page.

I have already added two records to the table. Ask me for the passwords.

username	**password**	**bio**	**admin**
doctor			1
nigel			0

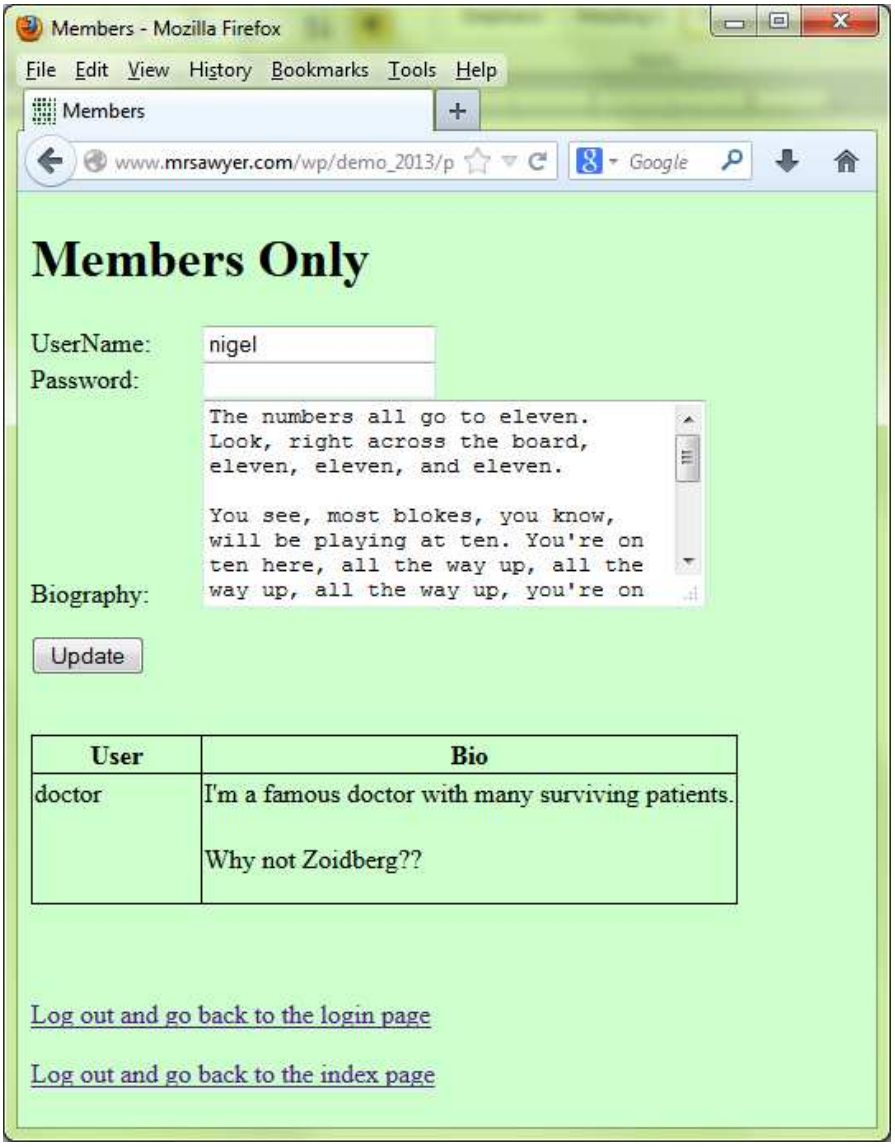
Important. Do NOT confuse these usernames and passwords with the usernames and passwords used to connect to the database.

Step 2. Write the members page. It should look something like the one shown. It will allow the user to edit their username, password, and bio fields. The user will also be able to see the names and bios for all other users (but not be able to change them).

The first thing this page should do is check that the $_SESSION variables were set (indicating that the user logged in). If not, redirect them back to the login page.

You must populate the form with the data from the database. Use a WHERE condition on the SELECT statement to get just the data you need.

If the user clicks the Update button you need to update username, password, and bio fields where the username and passwords fields equal the old values. Be sure to update the session variables as well.



Check out Johnny's site for formatting and html help.

Here is the code for the two links at the bottom.

```
<p><a href="logout.php?login=yes">Log out and go back to the login page</a></p>
<p><a href="logout.php?login=no">Log out and go back to the index page</a></p>
```

The logout page contains no html. It simply unsets and destroys the session and then redirects you to either your login page for this exercise or your index page.

Step 3. Write the admin page.

For this page use _____ for the user name and _____ for the password. This user is allowed to select, update, insert, and delete records. The other user is only allowed to select and update records.

The first thing this page should do is check that the $_SESSION variables were set (indicating that the user logged in). If not, redirect them back to the login page.

There are two forms. The first one obviously allows you to add new users with or without admin privileges.

The code for this should be fairly straightforward. Remember to store only a 1 or a 0 in the admin field.

The second form has a table that lists all the accounts except for the person who has logged in - can't have you deleting yourself, now can we?

The values for each checkbox should match the user name next to that checkbox. All user names will be unique and you should use the user name to identify which record you want to delete.

Name the checkboxes something like ck[] so that in your php code you can easily loop through the selected users. A portion of your php code should look something like this:


```
for ( $i = 0; $i < count( ....
        $sql = "DELETE FROM ACCOUNTS WHERE username = ...
        mysqli_query($link, $sql  ) or exit( mysqli_error($link) );
}
```