Displaying information from a database

The below link demonstrates displaying information from a database:


Following the above example, using a php loop to populate rows in an HTML table might look like the below code – where pimage, pname, pdesc ... are column names in your SQL table:

```php
while($row = $resultSet->fetch_assoc()){
    ?>
    <tr>
        <td>
            <img src="<?=$row['pimage'] ?>" alt="item" />
            <div class="pname"><?=$row['pname'] ?></div>
        </td>
        <td class="pdesc"><?=$row['pdesc'] ?></td>
        <td class="pinfo">
            <div class="info1"><?=$row['pinfo1'] ?></div>
            <span class="info2"><?=$row['pinfo2'] ?></span>
        </td>
    </tr>
<?php }
```

Using PHP_SELF

The below link discusses security when using PHP_SELF.


Your form tag might look like:

```html
<form name="fauth" class="fauth"
action="<?= htmlentities($_SERVER['PHP_SELF']) ?>"
method="post">
```
Retrieving POST variables

Check if the method is POST (not GET):

```php
if ($_SERVER['REQUEST_METHOD'] === 'POST')
```

NEVER USE UNSANITIZED INPUT IN YOUR SQL QUERY!

Here is an example to safeguard the input:

```php
$username = htmlspecialchars(trim($_POST['siname']));
$password = htmlspecialchars(trim($_POST['sipw']));
```

You might want to place input into an array:

```php
$postvars = array();
$postvars[] = htmlspecialchars(trim($_POST['siname']));
$postvars[] = htmlspecialchars(trim($_POST['sipw']));
```

Or place them into an associative array:

```php
$postvars[] = array("siname" => htmlspecialchars(trim($_POST['siname'])));
$postvars[] = array("sipw" => htmlspecialchars(trim($_POST['sipw'])));
```

Shopping cart

You might use a session shopping cart if signin succeeds, for example:

```php
$_SESSION['cart'] = array();
```

Then you could add items to it:

```php
$_SESSION['cart'][] = $itemID_from_input;
```

Then when the customer is ready to check out, loop through your cart to populate your SQL query with where conditions from your shopping cart so your query might end up looking like:

```sql
SELECT * FROM products WHERE id = '4321' OR id = '3456' OR id = '7578';
```

When your customer checks out (approves the purchase) be sure to update your table, decrementing the number of items available, for example:

```sql
UPDATE products SET pavailable=pavailable-1 WHERE ...
Finally, after checking out, you might destroy your shopping cart:

`$_SESSION['cart'] = null; unset($_SESSION['cart']);`

or just refresh the cart:

`$_SESSION['cart'] = array();`

```php
if / endif
Use a colon : for the if portion, and close it with a semicolon ; after endif. The html in between will only be generated if the expression is true.

```php
<ul if(expression): ?>
    <li class="welcome">Welcome <span><?= $_SESSION['uname']; ?></span></li>
    <li class="cart"><a href="cart.php">view cart</a></li>
    <li class="signout"><a href="signout.php">signout</a></li>
</ul>
```php
endif; ?>

A note about shopping carts

The popular Magento shopping cart has over 300 tables! Zen shopping cart also has dozens of tables. The purpose of lab 5 is to demonstrate using PHP and SQL to manage data.

My demo offers 2 types of products, and allows shoppers to select multiple items as well as choose how many of each item they want to add to their cart. This is beyond what you need to build for lab 5.

Many shopping carts simply have a select button, linking to a page displaying that item, where the shopper can then add one of that item to their cart. I would advise this technique for lab 5. For example:

Display a list of items to purchase, with each item in its own form. All forms can post to the same page (self) but the `$_POST` var will identify which item to link to and display. Use a php loop to generate each/all of the forms as demonstrated above building rows in a table. The forms do not need a name, just the input for the item needs a name, for example:
<?= $row['prod_id'] ?>
</input>
<input type="submit" value="Select" />

So when that form is submitted/selected, the $_POST['item'] var will contain that product id and only that product id, which you can select from your SQL products table and redirect to a page displaying that product.

To remember that product id from one page to another, you might write it to a session variable for example:

$_SESSION['view_item'] = $_POST['item'];

When reading (not writing) a session variable, you should first make sure it exists:

if(isset($_SESSION['view_item'])) ...

To redirect to the display_item_page (should be the same page for any item), you can write, for example:

header("Location: http://" . $_SERVER['HTTP_HOST'] . "/display_item.php");

The display page is where one of that item can be added to the shopping cart, then redirect back to the shopping list page.

Be sure to follow Professor Grogan’s guidelines:

http://analog24.net/pcosci/cs554/cs554Lab5.html

Good luck, and have fun!