CA 2 Database Access Using PHP

Nicole Kelly

For this CA, we had to create a program using PHP that would interact with the database. The code should be able to view the contents, insert new data, update the data and delete an object from the database. As far as I see, I had no errors in my code but I do know that the error messages don't work. The error message for 'could not insert on the user' on the createWineForm user comes up but doesn't give you any other options. I would have liked to have fixed that but I could not see a way past it. Many of the errors that I had prior to finishing my code were manly spelling mistakes or missing or wrong code.

Connection

This php file links to the database of the college which is daneel. It checks for the correct username and password. If these are incorrect, an error message appears telling the user they could not connect to the database. I had a problem with this file at the beginning of the CA but it was due to having the wrong password so when I tried to run my project, it failed to run.

```
Connection::$connection = new PDO($dsn, $username, $password);
if (!Connection::$connection) {
    die("Could not connect to database!");
}
```

User

The file was also a little difficult for me to understand. I presume the class user is called on different pages where it's applicable. The private $username is just creating an object called username. The __construct () function creates a new SimpleXMLElement object. This is an easy way of getting element attributes. The function getUsername() , finds the input and stores in the username. Return $this returns the input to the user.

```
public function getUsername() {
    return $this->username;//returns input username
}
```

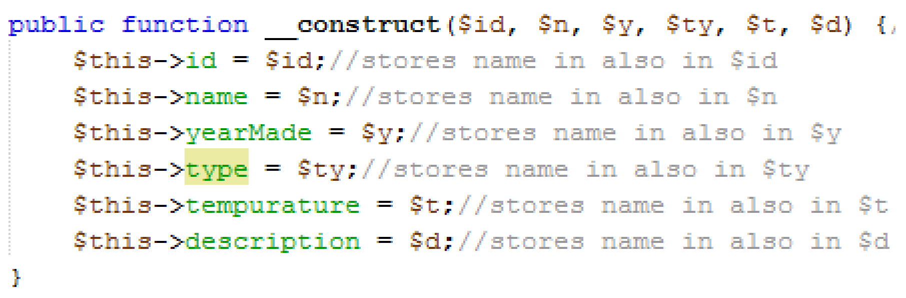
User Table Gateway

This php file creates a new user and checks if that user already exists in the database. This files uses SQL queries to SELECT and INSERT users into the database. It stores their username and password in the database. It also runs if statements that says if the user could not be retrieved from the database or if the user could not be created.

```
if (!$status) {
    die("Could not insert new user");
}
```

Wine

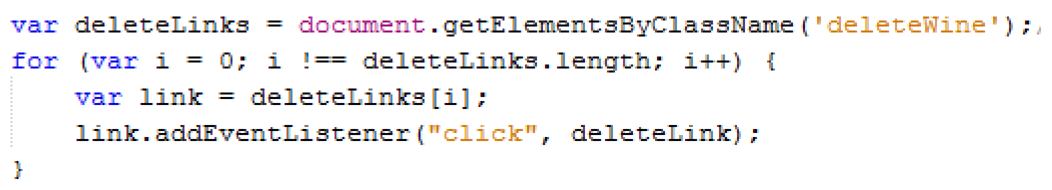
Like the user.php, this page creates a class, this time named wine and stores objects inside that class. . The private $name is just creating an object called name. The __construct () function creates a new SimpleXMLElement object. This is an easy way of getting element attributes.



```
public function __construct($id, $n, $y, $ty, $t, $d) {
    $this->id = $id;//stores name in also in $id
    $this->name = $n;//stores name in also in $n
    $this->yearMade = $y;//stores name in also in $y
    $this->type = $ty;//stores name in also in $ty
    $this->tempurature = $t;//stores name in also in $t
    $this->description = $d;//stores name in also in $d
}
```

Wine.js

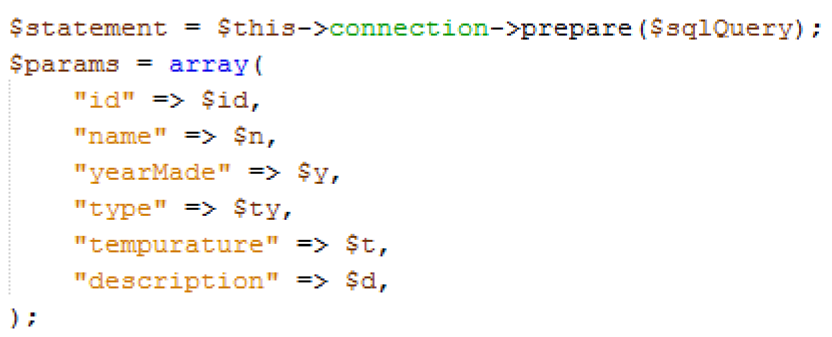
This javaScript file checks that the user has put in the right data on the form for edit and delete wine. It checks with the user to make sure that they want to delete the wine.



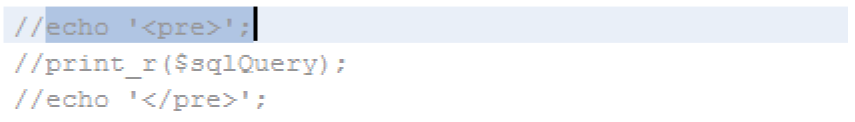
```
var deleteLinks = document.getElementsByClassName('deleteWine');
for (var i = 0; i !== deleteLinks.length; i++) {
    var link = deleteLinks[i];
    link.addEventListener("click", deleteLink);
}
```

Wine Table Gateway

Like user Table gateway, this file holds SQL queries that allow the user to get, insert, update and delete wine from the database. The getWine () function allows the user to view the wines in the database Using the SQL query SELECT. This retrieves all of the wines in the database. Whereas the getWineByWineID () gets a certain wine by its id. INSERT wine allows the user to create a new wine and add it to the database, using the SQL query, INSERT. UPDATE wine allows the user to change the details of a certain wine in the database, using the SQL query, UPDATE. DELETE wine allows the user to delete a wine in the database, using the SQL query, DELETE. These all have error messages that appear if the user was unsuccessful. I had the most problems with the update method as I had updateWines but I wine was the name of the table so when I fixed that it worked. I used the debug method echo '<pre>'; to find my problem after John showed me it.



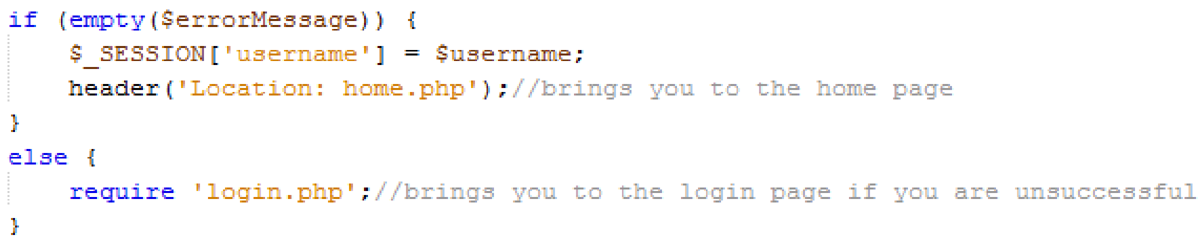
```
$statement = $this->connection->prepare($sqlQuery);
$params = array(
    "id" => $id,
    "name" => $n,
    "yearMade" => $y,
    "type" => $ty,
    "tempurature" => $t,
    "description" => $d,
);
```



```
//echo '<pre>';
//print_r($sqlQuery);
//echo '</pre>';
```

Check Login

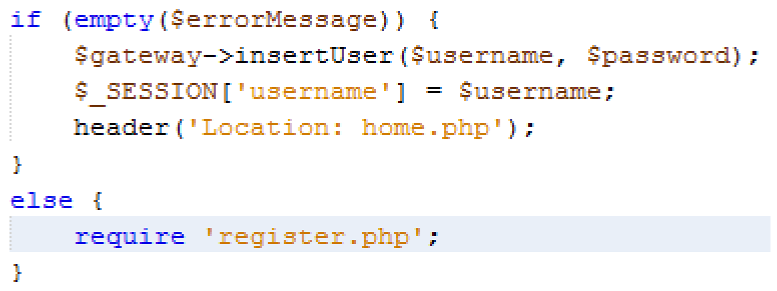
This php file checks the input from the user for the login and outputs text depending on what the user has put in. The $username calls the array that is stored in the username and outputs it. The if and else if, display an error message if the username is already taken and if the password is blank. The header brings you to the home.php file if the form was completed successfully. Filter sanitize protects your code so that users are unable to hack your code. If the login is unsuccessful it will reload the login page with the error messages and to the home page if successful.



```php
if (empty($errorMessage)) {
    $_SESSION['username'] = $username;
    header('Location: home.php');//brings you to the home page
}
else {
    require 'login.php';//brings you to the login page if you are unsuccessful
}
```

Check Register

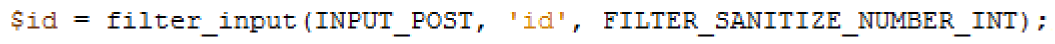
This file allows a new user to register. It stores their username in an array in the database. An error message is shown if the username is already registered or if the password and username fields are blank or if the passwords do not match. It also filters and sanitizes the code also. It also stores the username and password in the database. It also brings the user to the home page if successful and reloads the register page if unsuccessful.



```php
if (empty($errorMessage)) {
    $gateway->insertUser($username, $password);
    $_SESSION['username'] = $username;
    header('Location: home.php');
}
else {
    require 'register.php';
}
```

Create Wine

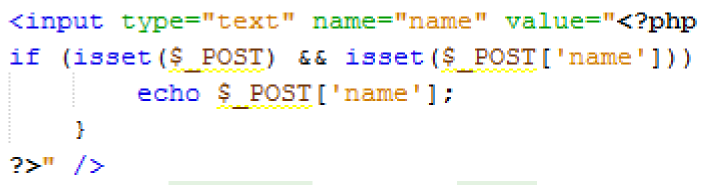
This file lets the user create a wine and stores the input in an array. It uses the filter sanitize to make sure any unwanted characters can’t be put in the input fields. It outputs a message if you created a wine successfully and relocates you to the home page.



```php
$id = filter_input(INPUT_POST, 'id', FILTER_SANITIZE_NUMBER_INT);
```

Create Wine Form

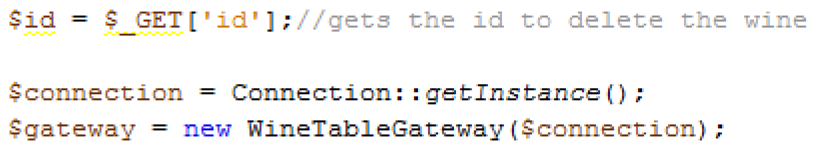
This page was mainly html so I had little or no problems with understanding this page. The html in this page just creates a form for the user to fill out. There is s piece of php code inside of the input box. This code isn't clear to me but I think it gets the input from the user, and keeps it in the input field so that they don't have to retype it.



```
<input type="text" name="name" value="<?php
if (isset($_POST) && isset($_POST['name'])) {
            echo $_POST['name'];
        }
?>" />
```

Delete Wine

This php file allows the user to delete a wine by using the wine id. It connects to the database and the wineTableGateway to delete the wine from both the database and avaya/daneel.

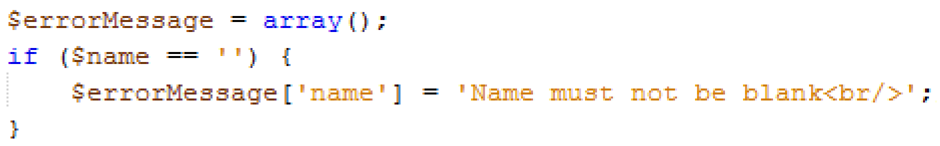


```
$id = $_GET['id'];//gets the id to delete the wine

$connection = Connection::getInstance();
$gateway = new WineTableGateway($connection);
```

Edit Wine

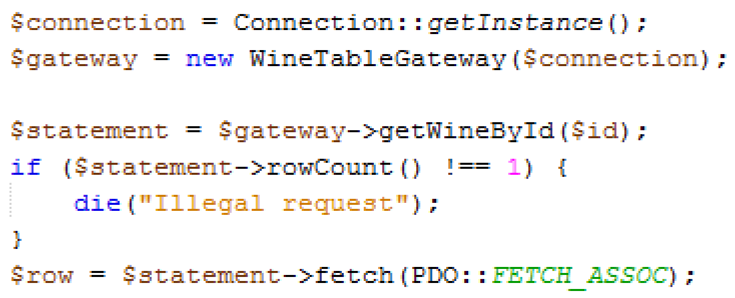
This php file allows the user to edit wine details. It connects to the database and the wineTableGateway to delete the wine from both the database and avaya/daneel. It ensures that the user is logged in. It also displays error messages. It has a validate method which validates a value as an integer, string or float optionally from the specified range.



```
$errorMessage = array();
if ($name == '') {
    $errorMessage['name'] = 'Name must not be blank<br/>';
}
```

Edit Wine Form

This page was mainly html so I had little or no problems with understanding this page. The html in this page just creates a form for the user to fill out to edit the wine details. There is a piece of php code inside of the input box. This code isn't clear to me but I think it gets the input from the user, and keeps it in the input field so that they don't have to retype it. It's very familiar to create wine form apart from the extra php at the beginning which connects to the database and gets the wine by its id. I am unsure of what the if statement below means.

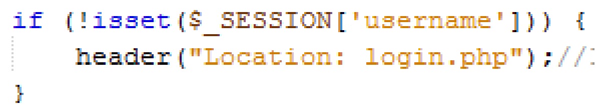


```
$connection = Connection::getInstance();
$gateway = new WineTableGateway($connection);

$statement = $gateway->getWineById($id);
if ($statement->rowCount() !== 1) {
    die("Illegal request");
}
$row = $statement->fetch(PDO::FETCH_ASSOC);
```

Ensure User Logged In

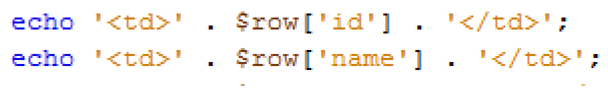
This piece of php code just makes sure the user is logged in before they can edit, view, insert or delete a wine.



```
if (!isset($_SESSION['username'])) {
    header("Location: login.php");//
}
```

Home

The home page is where the user can choose to view, edit, insert or delete a wine. The begining is mainly php code which requires other pages and connects to the database. And calls on the function getWine() from the wine table gateway. The next bit of php code displays the information from the database in a row. And has links to the view, edit and delete beside each wine. I found this page pretty straight forward and had little to no problems here.



```
echo '<td>' . $row['id'] . '</td>';
echo '<td>' . $row['name'] . '</td>';
```

Index

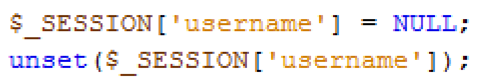
The index page displays the wines in the database. Again I found this page pretty straight forward and had little to no problems here.

Login

Is a page that has a small form that allows the user to log in if they are registered. I had little to no problems with this code and found it straight forward.

Logout

Allows the user to log out and appears as a link on most of the pages. I was unsure of what the code below means.



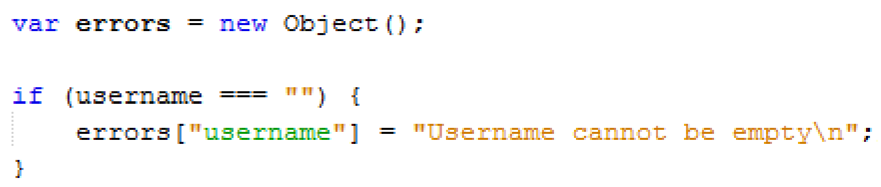
```
$_SESSION['username'] = NULL;
unset($_SESSION['username']);
```

Register

Is a page that has a small form that allows the user to register. I had little to no problems with this code and found it straight forward.

Register.js

This script, the function validateRegistration , gets called when the register button is pressed. Like before the variables are stored in arrays. Again I am not sure about the spanElements code. The if statements which just say that if the input is blank then output a message.

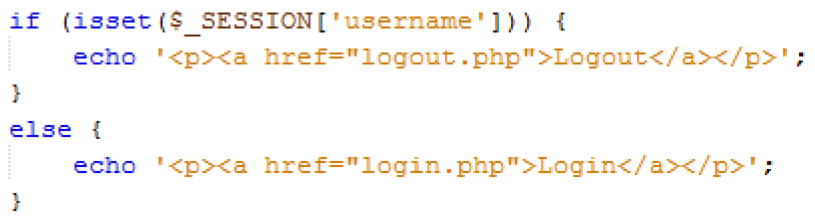


```
var errors = new Object();

if (username === "") {
    errors["username"] = "Username cannot be empty\n";
}
```

Toolbar

I am unsure of what toolbar means. I do know it's important but I'm not sure why. If I was to guess, if you are already logged in it links you to the logout, and if you are not logged in it links to the login.



```
if (isset($_SESSION['username'])) {
    echo '<p><a href="logout.php">Logout</a></p>';
}
else {
    echo '<p><a href="login.php">Login</a></p>';
}
```

View Wine

This file displays the wine that the user wants to view. The php code inside the html displays the wine in a row. It also links to the delete and edit pages. I had little to no problems with this code and found it straight forward.

In Conclusion

In conclusion, I found this CA a lot easier then the last time. When I was writing this report I found I didn't have the if statements in the edit or create wine so I am unsure of whether they work or gave me any errors. This CA I think has improved my debugging skills as I found most of the small bugs like spelling mistakes and such.