asgn08: Instructions

Objectives

- Add comments using docblock.
- Revisit coding standards.
- Continue preventing SQL injection.
- Sanitize values sent to the database with `escape_string`

Setup

- Watch the videos
- Create a new folder named `web250/asgn08-rud
- Download the starter files from Moodle and extract them into your new folder
- Use the existing `wnc_birds` database.

Git

This is how I used git. It is similar to the GitFlow Workflow, but I have eliminated the `dev` branch for now.

- Start on the master branch
- Create a new branch for each of the sections within the chapter. For instance, make a branch for the Dynamic Attribute List. This line combines the checkout with the creation of the branch.

```
git checkout -b asgn08-dynamic-attribute-list
```

- Once you have your code running correctly then do the following. I've thrown in some extra `git status` commands so you will get a better idea of the process.

```
git status
git add .
git commit -m "Finished the Dynamic Attribute List"
git status
git checkout master
git merge asgn08-dynamic-attribute-list
git status
```

Then create a new branch and checkout to it

```
git checkout -b asgn08-sanitize-values
```

Video notes

Watch the following videos from Chapter 4

Dynamic Attribute List

I didn't encounter any problems here

Sanitize values for a database

Ran into a little trouble here. Since we are using PDO, we should not have to sanitize our values using `real_escape_string` ; however, there is a problem where the query will not run if I have add an apostrophe to the input field. I burned too much time here and I will figure out this answer later, or if anyone else figures it out **please** share it with the class.

Here is his code. I found that it did not work (I could be doing something wrong) but I thought you should have the code in case you would like to try it.

Note: If you do not use it, then remember that anywhere he says to add the `sanitized_attributes` you will just add attributes. This sounds weird but it will make sense when you are coding the project.

```
protected function sanitized_attributes() {
       $sanitized = [];
       foreach($this->attributes() as $key => $value) {
           $sanitized[$key] = self::$database->escape_string($value);
       }
       return $sanitized;
   }
```

Find a record to update

Nothing weird that I can recall.

Update a record

I had a hard time using his `$attributes` and `attribute_pairs` arrays so I decided to code it without creating the loop. This is the code I left out

```
$attribute_pairs = [];
   foreach($attributes as $key => $value) {
     $attributes_pairs[] = "{$key} = '{$value}'";
   }
```

This means I am breaking the DRY (Don't Repeat Yourself) rule and my code will bloat a little later because I'll have some duplicate code. This is a good lesson all of us. I need to meet a deadline and I don't have time to write more efficient code. Feel free to share if you figure it out! I'll come back later to fix this but it will work for now.

Here is some sample code to get you going.

```
   $sql = 'UPDATE birds SET ';
   $sql .= 'common_name = :common_name, ';
   ...
   $stmt = self::$database->prepare($sql);
   $stmt->bindValue(':common_name', $this->common_name );
   ...
```

HTML forms for OOP

I think you will like this one! Straight forward and it will make gathering information from HTML forms much easier.

Validations and errors

This code that I had in the `status_error_functions.php` file provided by the author and I think it is incorrect. Our job is to figure it out! He has procedural code here (not a problem) but I could not get it to work correctly. It does not have anything to do with PDO. Here is the code. See if you can fix it.

Original code

```
function display_errors($errors=array()) {
  $output = '';
  if(!empty($errors)) {
    $output .= "<div class=\"errors\">";
    $output .= "Please fix the following errors:";
    $output .= "<ul>";
    foreach($errors as $error) {
      $output .= "<li>" . h($error) . "</li>";
    }
    $output .= "</ul>";
    $output .= "</div>";
  }
  return $output;
}
```

Form Field Folly and the Conservation Level

This is a nifty chunk of code from the `form_fields.php` file and it is important that you understand it; especially the 'selected' part. You can use this code as is. Writing the correct syntax was a pain. After studying this code, run it code and view the HTML source in your browser. Part of the difficulty is the embedded PHP tags but I don't see a more elegant way to write it.

```
<dl>
  <dt>Conservation Level</dt>
  <dd>
    <select name="conservation_id">
    <?php foreach(Bird::CONSERVATION_OPTIONS as $con_id => $con_name) { ?>
      <option value="<?php echo $con_id; ?>" <?php if ($bird->conservation_
    <?php } ?>
    </select>
  </dd>
</dl>
```

PHP Coding Standards

Webhost

After you get the code running locally, upload and test it on your Webhost.

Git and GitHub

Make sure your master branch has the correct version before pushing your code to GitHub

Sumbit

Submit your GitHub and Webhost addresses in Moodle.