Project Phase05

Objectives

Setup

You can use the **starter-files** or continue working with the code you have built.

Git

Continue using version control for your project.

If you need to start over with git, you must initialize your project. Navigate to your **sas** folder. Then

```
git init
```

If you continue to use the same files, including git, now is a good time to check if you need to stage and commit. Start with `git status`. If git displays that you are on a clean tree then continue with the assignment. If `git status` shows you have unstaged files, then enter the following to stage and commit.

```
git add .
git commit -m"Starting phase05"
```

Videos

Watch chapter 8: Validation Data with PHP from PHP and MySQL Essential Training: The Basics

Validation

8.1 Common data validation types

This chapter starts by listing some of the standard validation types.

- Presence
- String length
- Data type
- Inclusion in a set
- Format
- Uniqueness

- Copy the **validation_functions.php** file into your **private** folder.

- Read through the functions

- If you have a question about how a function works, post it in Slack.

- Add the line `require('validaton_functions.php')` to your **initialize.php** file.

- Notice that his functions return Boolean values (true or false). He doesn't actually return true or false; rather the expression is evaluated. This is a common way to write a `return` value.

In the `has_valid_email_format` function, he uses a regular expression. These are incredibly powerful, but if you don't use them frequently, you will spend a lot of time looking them up (that's okay!). In most instances, PHP has a similar function to a regular expression and the PHP function executes faster.

8.2 Validate form values

- As programmers, we want to validate data before creating and updating records.
- Put your validation code in a reusable function.
- Reporting all errors at once for the best user experience.

Copy and paste the **snippet1.txt** file that contains the `function validate_subject` in your **query_functions.php** file.

- Modify this code, so it validates salamanders instead of subjects.
- Delete the `position` and `visible` sections. We are not using those.

Here is what the code should look like if you want to copy and paste it.

```
function validate_salamander($salamander) {
    $errors = [];

    if(is_blank($salamander['name'])) {
      $errors[] = "Name cannot be blank.";
    }
    if(!has_length($salamander['name'], ['min' => 2, 'max' => 255])) {
      $errors[] = "Name must be between 2 and 255 characters.";
    }

    if(is_blank($salamander['description'])) {
        $errors[] = "Description cannot be blank.";
      }

    if(is_blank($salamander['habitat'])) {
    $errors[] = "Habitat cannot be blank.";
    }

    return $errors;
}
```

Modify your **edit.php** file using the `if` statement to test whether the `$result` was `true`. If it is not true, then display the errors.

Test your code using `var_dump()`.

I like to add the `exit()` function after `var_dump` so the output is easier to read.

```
vardump($someVariable); exit();
```

8.3 Display validation errors

I modified his `dispaly_errors()` function and deleted the CSS. Use this instead of his code.

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

Make sure to initialize `$errors` with an empty array by adding this line to **initialize.php**; otherwise you will most likely receive an error message.

```
$erros = [];
```

Later in the video, Kevin runs into a problem and has you move three lines outside of an `if...else` statement. I tried using the following code to mimic the same result. Based on my testing, you do not need these lines at all. I left them here as a reference.

```
$salamander_set = find_all_salamanders();
$salamander_count = mysqli_num_rows($salamander_set);
mysqli_free_result($salamander_set);
```

Once everything is working, don't forget to delete **create.php**

8.4 Problems with validation logic

This is a short video that is worth watching for reference.

Upload phase05 to your webhost

NOTE: Make sure to change your **db_credentials.php** file, so the program uses your webhost's credentials.

Test your program on your webhost

Git and GitHub

Time for a final commit and push. `git status` is unnecessary, but it helps me find anything I may have left out.

```
git status
git add .
git commit -m"Phase05 complete."
```

Push your code to your GitHub repo

```
git push
```

Submit your work

Submit your GitHub and webhost addresses in the comments section of Moodle.