Project Phase 02

Objectives

- Use PHP headers
- Use cURL (Client URL)
- Add output buffering
- Create utility functions for get and post requests
- Create code for a new record
- Create code to edit a record

Getting Started

Read all of the instructions before starting to code.

Start this phase by using the code you created for phase01 or using the phase02 starter files in Moodle.

You can copy the starter files into your **sas** folder. Git will recognize that these are new files. You will need to stage and commit them.

git stage and commit

Navigate to your folder

When you open VSCode, then open the terminal, you are automatically in your current working directory.

Type the following to "print working directory"

```
pwd
```

You should get something like

```
*[main][/Applications/MAMP/htdocs/web182]
```

The `[main]` is the current Git branch. The `/Applications` is because I'm on a Mac.

If you need to navigate to your `sas` folder you have two choices.

1.) If your your `sas` folder is inside your `web182` folder use.

```
cd sas
pwd
```

This will move you into your `sas` folder and display your current directory. Mine now looks like.

```
*[main][/Applications/MAMP/htdocs/my-web182/sas]
```

2.) If your `sas` folder is outside of your `web182` you will need to go up a level by using `..`

```
cd ../sas
pwd
```

It should look like

```
*[main][/Applications/MAMP/htdocs/sas]
```

Here are the commands to stage and commit now that you are in the correct folder.

`status` - shows your current status `add .` - stages all you untracked files `commit -m"some text"` commits your file and forces you to create a new message about the commit.

```
git status
git add .
git commit -m"starting phase 2 of the project."
```

File structure

This is the file structure you will work with for this assignment.

```
├── phase02
|   ├── index.php
|   ├── private
|   |   ├── functions.php
|   |   ├── initialize.php
|   |   └── shared
|   |       ├── salamander-footer.php
|   |       └── salamander-header.php
|   └── public
|       ├── images
|       ├── index.php
|       ├── salamanders
|       |   ├── create.php
|       |   ├── edit.php
|       |   ├── index.php
|       |   ├── new.php
|       |   └── show.php
|       └── stylesheets
|           └── salamanders.css
```

You can use these commands whenever you are at a point where you want to save your project using Git. Usually, I run them after I've finished a chunk of code I want to make sure I've saved.

What to Watch

We will continue using the tutorial, PHP with MySQL: Essential Training 1.

- Chapter 3: Headers and Redirects
- Chapter 4: Build Forms with PHP

Chapter 3: Headers and Redirects

This chapter shows how PHP uses file headers. At first this may seem a bit dry, but knowing how to redirect files and use output buffering is an essential part of PHP programming.

- Add the functions, error404 and error500.
- Create a file in your **salamanders** folder called **new.php**.
- Add the reference to **initialize.php** and place it at the top of **new.php**.

- Test your code. You should receive "No error".
- Go along with Kevin and try using cURL (Client URL) and run the `curl` command. Here is an example of my code.
- Take a screenshot of the output after you run it and save the file as `curl.png`.

```
❯ curl --head http://localhost:8888/web182/sas/public/salamanders/new.phpcu
HTTP/1.1 200 OKHTTP/1.1 200 OK
Date: Mon, 21 Mar 2022 15:13:12 GMTDate: Mon, 21 Mar 2022 15:13:12 GMT
Server: Apache/2.4.46Server: Apache/2.4.46 ((UnixUnix)) OpenSSL/1.0.2u PHP/
X-Powered-By: PHP/8.0.8X-Powered-By: PHP/8.0.8
Content-Type: text/htmlContent-Type: text/html;; charsetcharset==UTF-8
```

Go through the same process for errors 404 and 500.

NOTE: I could not get mine to work with 404 or 500! It is okay if you don't either.

redirect_to

Create the `redirect_to` function in your **functions.php** file.

Output buffering

Add output buffering to your `initialize.php` file.

Build Forms with PHP

You will need to create the files, `new.php`, `edit.php` and `create.php` from scratch.

Modify your code so you can pull up the `edit.php` file.

Note, you will only be able to see the `id` in the URL. No data is present in the form itself.

Use form parameters

Use the code from `create.php` and modify it for salamanders. The only value we are passing by the `$_POST` superglobal is the salamander name.

form action

Modify the form action in `new.php` so it references `create.php`. Use the `url_for`

utility function.

Detect form submission

Add these two functions to your `funcions.php` file.

```
is_post_request()
is_get_requestis_get_request()
```

Modify your `create.php` file so it uses the two functions you just created.

Single Page Form Processing

This is a new concept for our class and it breaks one of the rules called "separation of concerns"; however it also makes coding easier since everything is in one place. It also shows how you can use `GET` and `POST` together.

Final Checklist

These are the files you should complete while working through the assignment.

Add the following functions to the `functions.php` file

- `error_404`
- `error_500`
- `redirect\_to($location)`
- `is\_post\_request()`
- `is\_get\_request()`

- Add `ob_start()` to `initialize.php`

- Edit the `salamanders/index.php` link

- Add an `sas/index.html` file that lists the two phases. Delete the previous `index.php` file that automatically redirected the user.

- Create the following files in the salamanders folder.

    - `new.php`
    - `create.php`

- `edit.php`

Once you have finished your code, run the staging and commit commands (example above) before pushing your code to GitHub.

```
git status
git add .
git commit -m"Finished phase 2 of the sas project."
git status
```

Git log

You do not need to push your code to GitHub for this assignment.

Instead, submit your git log in Moodle. There are two ways to complete this task.

Copy and paste

Run the following in your terminal to display the log.

```
git log
```

Copy and paste the output in a new file named `phase02-git-log.txt`

OR

Use the command line

You can use the Linux `>` output operator. This operator will write your git log to a new file.

You need to be in your `sas` folder (see above to navigate.)

Now you are ready to run this command that outputs the git command to a new file.

The `--decorate` switch is optional but it makes reading the output easier.

```
git log --decorate > phase02-git-log.txt
```

You can open your new file in VSCode to see it!

Update your webhost

Upload your code to your webhost. The file structure below shows

1. The **index.html** file at the root (top position). This file contains links to phases 1 and 2.
2. The tree shows both phases 1 and 2.

```
├── index.html
├── phase01
|   ├── index.php
|   ├── private
|   |   ├── functions.php
|   |   ├── initialize.php
|   |   └── shared
|   |       ├── salamander-footer.php
|   |       └── salamander-header.php
|   └── public
|       ├── images
|       ├── index.php
|       ├── salamanders
|       |   ├── index.php
|       |   └── show.php
|       └── stylesheets
|           └── salamanders.css
├── phase02
|   ├── index.php
|   ├── private
|   |   ├── functions.php
|   |   ├── initialize.php
|   |   └── shared
|   |       ├── salamander-footer.php
|   |       └── salamander-header.php
|   └── public
|       ├── images
|       ├── index.php
|       ├── salamanders
|       |   ├── create.php
|       |   ├── edit.php
|       |   ├── index.php
|       |   ├── new.php
|       |   └── show.php
|       └── stylesheets
|           └── salamanders.css
```

Test your code on your host and make sure all of your links work.

Submit your work

- Put a link to your website that directs me to phase2 of the project.
- Submit your `phase02-git-log.txt` in Moodle.