Project Phase04 - CRUD

Objectives

- Find a single record and displays its contents
- Create a new record
- Update an existing record
- Pass an array to a function
- Delete a record
- Add a README.md file

Setup

Continue working in your `sas` folder. You may use your existing code or download the starter files.

Create a folder named `web182/sas/phase04`.

Download and place the starter code in the `phase04` folder or you may use the code you have already written from the previous phase.

Watch Chapter 7 - CRUD from the InLearning video, PHP with MySQL Essential Training: 1 The Basics.

We are replacing **subject** with **salamander**.

I modifed the author's `delete.php` file and put it in the starter files.

I did not include the HTML `form` section in the code for the `new.php` and `edit.php` files. The forms change a lot from phase03, so it is better to write the code yourself.

Use Relative Paths

When the author is using the `redirect_to/url_for()` function, he uses a absolute path. Notice the front slash before the word salamander. This works only because of the way he wrote his function. A better idea is to use relative paths (the function still works).

Your code should look like this when using the `redirect_to` function.

```
redirect_to(url_for('salamanders/index.php'));
```

or if you have modified his code to use camelCase

```
redirectTo(urlFor('salamanders/index.php'));
```

Forms and Labels

In this video, the author uses XHTML syntax (version 4). Here are a couple of items to take note. Time to update his code. As you write your code, be sure to

- Remove the trailing `.../>` at the end of the `<input type=...` tags.
- Add HTML labels to forms.

Git

Continue using version control for your project.

Commit

This is a good time to commit your code.

```
git status
git add .
git commit -m"Starting phase04 of the sas project."
```

7.1 Find a Single Record

Note how the `id` is passed using `GET` and not `POST`. It is typical in PHP coding to use both `GET` and `POST` requests.

Use this information to create a `salamander` array instead of `subjects`.

7.2 Use Form Data to Create Records

This is an example of 2-page form submission. As he demonstrates, you can remove the code

at the top of `new.php`.

Another important point is that `new.php` gathers the new salamander information from the user. The `create.php` file is the one that will actually create the salamander with an INSERT statement. In short `new.php -> create.php`.

Notice that he leaves out `id` in the query. MySQL takes care of this when you run the query becuase we are using MySQL's `AUTO_INCREMENT`.

Remember that you are using the field names from the database.

- name
- habitat
- description

Make sure to use the HTML `textarea` for habitat and description.

Commit

Now is a good time to commit your progress.

7.3 Use Form Data to Update Records

This is an example of single-page form submission.

Copy and paste the form you created in `new.php`.

Continue working through the code until you are able to successfully edit your data.

Commit

Use Git to commit your progress.

7.4 Form Options from Database Data

The video shows how to create a drop-down list using a for loop. In addition, he shows you how to automatically make the list larger. We are not using any of these features.

Create an Array in create.php

In the second half of the video, he shows how to make an array for your `create.php` file. He

references this in the previous video, now is a good time to do the same thing when creating a new record.

Be sure to test the function by creating a new salamander after modifying your code.

Commit

Use Git to commit your progress.

7.5 Delete a Record

This is another example of single-page processing. Another way to think of this is a page calls itself in the HTML form action.

If it is a `GET` request, then display the form, otherwise, it is a `POST` request and you will handle the form data and delete the record.

You shouldn't need to create a link from the salamanders/index.php page since you wrote it in phase03.

Test your code.

Commit

Use Git to commit your progress.

Uplaod to your webhost

Remember that the database you created on your webhost is different than the database on your localhost. You can choose not to upload the `db_credentials.php` file and keep using the one you have on the server.

Pushing to GitHub

Temporarily remove the db-credentials.php file

You `db_credentials.php` file has your login information in it and GitHub is a public repository. It is a good idea to temporarily remove the file from your `sas` folder, push to GitHub, then put the file back.

We will learn a better way to do this in the future.

Watch the video on Creating a GitHub repo

What to submit

NOTE: I have left a place in Moodle to submit your file in the case you can't create a GitHub repository.

Add the following to the comments section in Moodle.

- Link to your phase04 webhost
- Link to your GitHub repository