asgn09: Instructions

Objectives

- Delete a user
- Create inheritable code
- Add an Admin class
- Hashing passwords
- Validate the admins view

Introduction

We will finish up chapter 4 and then start chapter 5 in the LinkedIn/Learning OOP Databases tutorial. Start with the Delete user tutorial and end by watching the Admin Validations video. We will complete the Object-Oriented Authentication next assignment.

Some good news

You've done a lot of hard work to get this far. You started out with objects and then combined them with at database. Now you can start to reap the benefits of that work. The tutorials will seem easier even though you are still learning a lot.

Part 1 - Finish chapter 4

The delete method

Use the starter file or continue working with the file you completed from **asgn08**.

Modify your code so you it will delete a bird. Here are a few lines of code to help you in the delete method. You'll need to write the code before those lines.

```
$stmt = self::$database->prepare($sql);
$stmt->bindValue(':id', $this->id );
$result = $stmt->execute();
```

Creating inheritable code

This is an exciting module! You will learn how making abstract code and inheriting code from a parent class can dramatically improve your code. Remember that one of the goals in software development is writing DRY (Don't Repeat Yourself) code. This is much easier said than done.

Chapter 5 - Admins

Provided files

- `create-table-admins.sql` -- load up this file using the MySQL CLI or a GUI.

- `admins` folder -- Store this folder in your `public/bird-staff/` folder. I have fixed all of the file path problems so it works with the bird's program. Please let me know if I miss anything so I can fix it for the future.

Like the previous assignments, we are going to alter the author's code so we can use them with our bird database. We will use almost all of the author's Admin code. I will alter the author's code so the headers, footers and links work. That way you can concentrate on objects instead of tracking down file path errors. The file you will spend most of your time working on the `admins.class.php` file. Pay close attention when he starts using `static` instead of `self`. This would be a great time to review the difference between the two.

`admin.class.php` - you will add code to this file in the `private/classes/` folder.

Git

Now that your files are in place, it is time to set up git. In your terminal, navigate to your current working folder.

Win

```
c:/xampp/htdocs/web250/asgn09
```

Mac

```
/Applications/MAMP/htdocs/web250/asgn09
```

After that it is the usual git commands

```
git status
git add .
git commit -m "Init commit"
git status
git checkout dev
```

Order matters when using a parent class

When you extend a class from a parent class, the order of the included files is important. The file that contains the parent class **must** come first. Here is a short example where `databaseobject.class.php` comes before `bird.class.php` in the `initialize.php` file.

```
include('classes/databaseobject.class.php');
include('classes/bird.class.php');
include('classes/parsecsv.class.php');
```

Hashing passwords and admin validations

This is a great way to learn about hashing passwords, and as a bonus, it is painless.

The validations are also easy. Again, notice that once you have your classes set up, how nice it is to extend a parent class.

GitHub

Now that you have your code running correctly, it is time to push it to GitHub. You've been through this a few times. Contact your instructor if you run into trouble or have a hard time remembering (it's okay -- we are covering a lot of ground).

Webhost

Push your code to your Webhost. Test it to make sure it works as expected.

Submit work

Submit your GitHub and Webhost address in the Moodle comments section.