CS 328 - Week 11 Lab Exercise - 2023-04-06/07

Deadline

Due by the end of lab today.

Purpose

To practice using PHP with OCI and bind variables, calling a stored procedure, and calling a function.

How to submit

Submit your files for this lab using `~st10/328submit` on nrs-projects, with a homework number of **91**.

(**If** you are **re-submitting** an individually-improved or -completed version **after** lab, include a file **`README.txt`** that notes that this is the case, so I and the grader will know that.)

Requirements

- You are required to work in pairs for this lab exercise.
- Make sure BOTH of your names appear in each file submitted!
- When you are done, or before you leave lab, somehow e-mail or copy the lab exercise files so that BOTH/ALL of you have copies, and EACH of you should submit these files using `~st10/328submit` on nrs-projects, with a lab number of **91**.

Your Tasks

Copy over the following SQL scripts and run them in your Oracle account:

```
cp ~st10/count_empl.sql .      # DON'T FORGET the SPACE and PERIOD!!
cp ~st10/terminate_empl.sql .  # DON'T FORGET the SPACE and PERIOD!!
```

- These create and test a PL/SQL stored function `count_empl` and a PL/SQL stored procedure `terminate_empl`.

Create a PHP document named `328lab11.php` that **both** creates a form **and** crafts a response to that form when it is submitted (in postback style, as demonstrated in:

- Week 9 Lecture 2 example `php-form-handling.php`
- Week 10 Lecture 2 example `try-oracle.php`
- Week 11 Lecture 1 example `insert-dept.php`).

This document should meet the following requirements:

- Fill in its opening comment block with the URL I can use to run your version, your names, and the last modified date.
  - Note that there will be a penalty if your document does not display using this URL when I and/or the grader try it.
- In your HTML `body` element *before* your PHP `if` statement:

- include an `h1` element describing this document in general
- and then include an `h2` element including your names

What form?

The PHP function `make_empl_form.php` can be called to create the desired form. (It just has the side-effect of creating the form -- it does NOT create an entire HTML document!)

- Copy this file into an appropriate location on one of your nrs-projects accounts:

```
cp ~st10/make_empl_form.php .  # DON'T FORGET the SPACE and PERIOD!!
```

- Use `require_once` in the `head` element of your `328lab11.php` to make this function available in this PHP, and call it appropriately in its `body` element.

What form response?

When its form is submitted using `method="POST"`, your PHP should craft a response as follows:

- It should call provided PL/SQL function `count_empl` with the *sanitized* user response from the "Last name of employee to terminate" textfield.
  - If that function returns 1, it should then call provided PL/SQL function `terminate_empl` to terminate that employee,

    and then output a `p` element saying that that employee has been terminated (and include their last name IN that message!)
  - If that function returns 0, it should output a `p` element saying that there is no employee with that name (and include their last name IN that message!)
  - If that function returns more than 1, it should output a `p` element saying that there is more than one employee with that name (and include that last name IN that message!)
- It should call provided PL/SQL function `count_empl` with the sanitized user response from the "Last name of employee to get raise".
  - If that function returns >= 1, it should then attempt to increase the salary of all employee(s) with the specified last name in the `empl` table by the specified raise amount, **using bind variables for the employee's last name and the raise amount within a SQL `update` statement**,

    and output a `p` element saying that employee(s) with that last name have been given that raise (and include their last name and the raise percentage IN that message!)
  - If that function returns 0, it should output a `p` element saying that there is no employee with that name (and include their last name IN that message!)
- REMEMBER to call `oci_commit` before you close your OCI connection, to commit any changes made!

- REMEMBER! **Never trust user data!** Appropriately use `htmlspecialchars` and/or `htmlentities` and/or `strip_tags` so that you don't unwittingly execute user-injected inappropriate code or other badness.
- FUN FACT: I don't *think* you'll need it for handling this form, but you can see IF an array KEY exists in an associative array using:

```
array_key_exists($desired_key, $desired_array)
```

- FUN FACT: I don't *think* you'll need it for handling this form, but you can see IF a variable or array reference has a value set for it using:

  ```
  isset($var_or_array_ref)
  ```

How can one strict-validate the HTML resulting from a PHP?

Beware -- if you put the URL of this PHP directly into the validator at https://html5.validator.nu/, it looks like it does regular-HTML5 validation (not strict-validation) of just the response containing its form. This is not a bad start, but it is not strict-validation, nor does it check your PHP's response with a submitted form's values.

Here is one approach I have successfully used to strict-validate a multi-document PHP such as `328lab11.php`:

- Put your `.php`'s URL in a browser and view its source, copy and paste that source into a file with suffix `.xhtml`, (for example, `328lab11-1.xhtml`) and put the URL of that `.xhtml` document into the validator.
- Put your `.php`'s URL in a browser and submit its form, then view that *response*'s source, and copy and paste that *response*'s source into a file with suffix `.xhtml`, (for example, `328lab11-2.xhtml`) and put the URL of that `.xhtml` document into the validator.

Optional additions

- You may add an external CSS formatting your PHP document if you would like; if you do so, also submit that `.css` file.

Submit your resulting `328lab11.php` (and all additional files it uses, if any) with a lab number of **91**.