CS 328 - Homework 10

Deadline

11:59 pm on **Friday, April 25, 2025**

Purpose

To get more practice writing PHP documents that send requests to Oracle using OCI, now also including using OCI to request calls of PL/SQL stored subroutines.

How to submit

Each time you wish to submit your work-so-far, submit your files using **`~st10/328submit`** on nrs-projects, with a homework number of **`10`**.

Important notes

- **NOTE**: you are **welcome** to use **`require_once`**/**`include_once`**/**`require`**/**`include`** in your PHP documents!
  - BUT when you use them in homework problems' documents, be sure to also **SUBMIT** copies of all files that you are requiring/including!
- **DO NOT USE ANY PHP FRAMEWORKS FOR THESE PROBLEMS** -- one of this course's purposes is to provide you with practice with "plain" PHP.
- Remember: You are required to use **`normalize.css`** for all of your web pages for CS 328, and to add **`link`** elements for additional CSS external stylesheets *after* this (but still **within** the **`head`** element).

  EXCEPT for **`normalize.css`**, **DO NOT USE ANY CSS FRAMEWORKS or PREDEFINED LIBRARIES for this course** (unless you get prior, explicit permission). One of this course's purposes is to provide you with some practice with the basics of "plain" CSS, so you can better make use of frameworks and predefined libraries later.
- Remember: there are now **CS 328 PHP Coding Standards so far** posted on the public course web site, under References -- you are also expected to follow these for all course PHP documents.
- You are expected to follow all course coding standards posted on the public course web site; course documents (including documents generated by your PHP files) are also expected to validate as "strict"-style HTML, and valid CSS.
- Make sure that you have executed the scripts **`create-bks.sql`** and **`pop-bks.sql`**, and that the bookstore tables are successfully created and populated.

Problem 1 - calling a PL/SQL stored subroutine using OCI

On Homework 5 - Problem 6, you wrote a PL/SQL stored function **`get_pub`**, that expects a title's ISBN and returns the **name** of its publisher (**not** its publisher ID).

On Homework 9 - Problem 3, you built a PHP postback document that either creates a **`form`** that included a **`select`**/drop-down widget of titles and ISBNs dynamically built from a query, or crafts a response including queried information for the selected title.

To get some practice calling a PL/SQL stored subroutine from PHP, for this problem, you'll build a PHP postback document that either:

- creates a **`form`** with a **`select`**/drop-down widget of titles and ISBNs dynamically built from a query, or
- crafts a response including the (sanitized) selected title's ISBN and the name of its publisher as returned from a call to PL/SQL stored function **`get_pub`**.

Homework 10 - Problem 1 requirements

Make a **COPY** of your file **`bks.css`** from **Homework 9 - Problem 3** in a **DIFFERENT** directory on nrs-projects (so you will not interfere with your earlier homework's files!).

Create a PHP document **`328hw10-1.php`** that meets the following requirements. (You could use either the posted **`html-template.html`** or your **`328hw9-3.php`** as the initial basis.)

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** the **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** the **`head`** element: **if** you would like, include PHP tag(s) with statement(s):
  - to enable PHP error reporting
  - to include the definition of function **`hum_conn_no_login`**, from a local copy of file **`hum_conn_no_login.php`**
- **Within** the **`head`** element, add a second **`link`** element so that this will be further styled using Homework 10's version of **`bks.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw10-1.php`**.
- Within the **`body`** element, include an **`h1`** element with appropriate content that somehow includes "your" bookstore's name (from previous homeworks' **`about-bks.html`**).
- ***Somewhere*** in the **`body`** element, include an element that **visibly** includes **your name**.
  - (Just in case you'd like to try out using **`328footer-plus-end.html`** for this problem, I am ***not*** requiring that your name be in the **`footer`** element for this problem.)
- Your **`328hw10-1.php`**'s logic should be designed so that its response includes either a form, or a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw10-1.php`** can be the **same** as that from **`328hw9-3.php`** -- that is, it should meet the **same** requirements as that **`form`** element:
  - its **`action`** attribute should have as its action:

`"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`

- instead of being hard-coded, the **`option`** elements of ISBN-title pairs in its **`select`**/drop-down **must** be **dynamically built** based on **querying** for the current ISBNs and titles of books in the bookstore databases' **`title`** table.
- AS in Homework 4 - Problem 4:
  - Set up this **`select`**'s **`option`** elements so that the user **sees**, in the **`select`**/drop-down box, ISBN-title name pairs, **but**...
  - ...when a particular option is selected, the **value** in the resulting name=value pair for this option is **JUST** the ISBN for the user's choice.
  - (for example, if the user selects an option that is displayed as:

    `9780131103627 - The C Programming Language`

    ... the form will submit a name=value pair whose value is just `9780131103627`)
  - ASK ME if you are not sure what I am asking for here.

- When this form is submitted, the response generated by your **`328hw10-1.php`** should include the following:
  - Appropriately **sanitize** the title ISBN submitted by this form.
  - Use the sanitized ISBN as the argument to PL/SQL stored function **`get_pub`**
    - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your PL/SQL function call string -- you are **required** to use a bind variable instead!
  - Output a **`p`** element with content including **both** the sanitized title ISBN and the publisher name as returned by **`get_pub`**.
  - Remember to call **`oci_free_statement`** to free the statement object used to call **`get_pub`** when you are done.
  - Remember to call **`oci_close`** to **close** the connection object used when you are done.
  - For more-convenient re-use, include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-1.php`**.

Your Homework 10 version of **`bks.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets Homework 7 - Problem 5's requirements.
- But, whether it is changed or not, make sure the resulting **`bks.css`** still validates as valid CSS.

Strict-validate the two parts generated by your **`328hw10-1.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw10-1.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw10-1-1.xhtml`**, and put the URL of your **`328hw10-1-1.xhtml`** into the validator.

- Put your **`328hw10-1.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw10-1-2.xhtml`**, and put the URL of your **`328hw10-1-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw10-1.php`** (and all additional files it uses, if any)
- **`328hw10-1-1.xhtml`** and **`328hw10-1-2.xhtml`**
- **`bks.css`**.

Problem 2 - building dynamic form widget(s) involving your "second" database

For this problem, you are going to build another **PHP postback document** that either:

- creates a **`form`** with dynamic form widget(s) **dynamically** built based on a query's results, or
- crafts a response to that form when it is submitted with the help of a dynamic **`select`** statement that uses **bind variables** rather than concatenation (to help thwart SQL injection).

Consider:

Recall that, for Homework 4 - Problem 5, you imagined a **question** that an expected user of your "second" database might want to ask, that could be answered by a `select` statement that has a `where` clause specifying something entered by the user. You then designed a **`form`** element, in **`custom-choice.html`**, to allow a user to make such a choice.

Then, in Homework 7 - Problem 6, you created an external CSS **`custom.css`** to style this form.

Now, finally, it is time to replace the hard-coded form widget element(s) in Homework 7 - Problem 6's **`custom-choice.html`** with dynamically-generated form widget element(s), and use the user's choice of those when that form is submitted to answer such a question.

Homework 10 - Problem2 requirements

Make a **COPY** of your file **`custom.css`** from **Homework 7 - Problem 6** in a **DIFFERENT** directory on nrs-projects (so you will not interfere with your earlier homework's files!).

Using the posted **`html-template.html`** as the initial basis, create a PHP document **`328hw10-2.php`** that meets the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** the **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** the **`head`** element, add a second **`link`** element so that this will be further styled using Homework 10's version of **`custom.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw10-2.php`.**

- Within the **`body`** element, include an **`h1`** element.
- ***Somewhere*** in the **`body`** element, include an element that **visibly** includes **your name**.
  - (Just in case you'd like to try out using **`328footer-plus-end.html`** for this problem, I am ***not*** requiring that your name be in the **`footer`** element for this problem.)
- Your **`328hw10-2.php`**'s logic should be designed so that its response includes either a form, or a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw10-2.php`** should be a variation of your **`form`** from Homework 7 - Problem 6's **`custom-choice.html`**, meeting the following requirements:
  - its **`action`** attribute should have as its action:

    `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`
  - instead of being hard-coded, your form widget element/set of elements for the user's choice **must** be **dynamically built** based on **querying** for the current choices from your "second" database.
- When this form is submitted, the response generated by your **`328hw10-2.php`** should include the following:
  - Appropriately **sanitize** each piece of information submitted by this form.
  - Build a dynamic **`select`** statement that projects something(s) that can be used to answer a question such as that you imagined back in Homework 4 - Problem 5, and uses at least one **bind variable** (instead of concatenation!) in its **`where`** clause based on the user's selection from the form's dynamically-built widget/set of widgets.
  - Add the **`select`**'s results to the response in a pleasing, strict-HTML-style way.
  - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your **`select`** statement string -- you are **required** to use bind variables instead!
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-2.php`**.

Your Homework 10 version of **`custom.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets Homework 7 - Problem 6's requirements.
- But, whether it is changed or not, make sure the resulting **`custom.css`** still validates as valid CSS.

Strict-validate the two parts generated by your **`328hw10-2.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw10-2.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw10-2-1.xhtml`**, and put the URL of your **`328hw10-2-1.xhtml`** into the validator.
- Put your **`328hw10-2.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw10-2-2.xhtml`**, and put the URL of your **`328hw10-2-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw10-2.php`** (and all additional files it uses, if any)
- **`328hw10-2-1.xhtml`** and **`328hw10-2-2.xhtml`**
- **`custom.css`**.

Problem 3 - calling a PL/SQL stored subroutine for your "second" database

One Homework 8 - Problem 4, you wrote a PL/SQL stored function or PL/SQL stored procedure for your "second" database, in **`*second-db*.sql`**.

To get more practice calling a PL/SQL stored subroutine from PHP, for this problem, you'll build a PHP postback document that either:

- creates an appropriate **`form`** allowing the user to enter appropriate argument(s) for your "second" database's stored subroutine, or
- crafts an appropriate response after calling your "second" database's stored subroutine with the (appropriately sanitized) user data from that submitted form.

Homework 10 - Problem 3 requirements

Create a PHP document **`328hw10-3.php`** that meets the following requirements. (You could use either the posted **`html-template.html`** or your **`328hw10-3.php`** as the initial basis.)

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** the **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** the **`head`** element: **if** you would like, include PHP tag(s) with statement(s):
  - to enable PHP error reporting
  - to include the definition of function **`hum_conn_no_login`**, from a local copy of file **`hum_conn_no_login.php`**
- **Within** the **`head`** element, add a second **`link`** element so that this will be further styled using Homework 10's version of **`custom.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw10-3.php`**.
- Within the **`body`** element, include an **`h1`** element with appropriate content.
- ***Somewhere*** in the **`body`** element, include an element that **visibly** includes **your name**.
  - (Just in case you'd like to try out using **`328footer-plus-end.html`** for this problem, I am ***not*** requiring that your name be in the **`footer`** element for this problem.)
- Your **`328hw10-3.php`**'s logic should be designed so that its response includes either a form, or a response to its form -- its response should never include both.

- The initial **`form`** element generated by your **`328hw10-3.php`** should meet following requirements:
  - its **`action`** attribute should have as its action:

    `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`
  - it should include appropriate form widget element(s) (with logically-associated **`label`** element(s)) so that the user can enter/select the needed argument(s) for your "second" database's stored subroutine.
    - ***If*** any of these are based on choosing from an existing table attribute's values, be sure to dynamically build the choices based on querying the current values from your database rather than hard-coding them.
  - include an **`input`** element with **`type="submit"`** (which does *not* need a logically-related **`label`** element)
- When this form is submitted, the response generated by your **`328hw10-3.php`** should include the following:
  - Appropriately **sanitize** the user submission(s) from this form.
  - Appropriately use the sanitized submission(s) to determine the argument(s) to your "second" database's PL/SQL stored subroutine.
    - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your PL/SQL subroutine call string -- you are **required** to use bind variable(s) instead!
  - What should you output in this response? It depends on the nature of your "second" database's PL/SQL stored subroutine:
    - If it is a stored function that returns a desired value, include its results in the response in a pleasing, strict-HTML-style way.
    - If it is a stored procedure (or a stored function that returns a status code) that performs some desired task, note what it did in a pleasing, strict-HTML-style way.
  - ***If*** your "second" databases' PL/SQL stored subroutine changed the database, be sure to call **`oci_commit`** when your logical transaction is complete to commit those changes.
  - Remember to call **`oci_free_statement`** to free the statement object used to call your "second" database's stored subroutine when you are done.
  - Remember to call **`oci_close`** to **close** the connection object used when you are done.
  - For more-convenient re-use, include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-3.php`**.

Your Homework 10 version of **`custom.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets Homework 7 - Problem 6's requirements.
- But, whether it is changed or not, make sure the resulting **`custom.css`** still validates as valid CSS.

Strict-validate the two parts generated by your **`328hw10-3.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw10-3.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw10-3-1.xhtml`**, and put the URL of your **`328hw10-3-1.xhtml`** into the validator.
- Put your **`328hw10-3.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw10-3-2.xhtml`**, and put the URL of your **`328hw10-3-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw10-3.php`** (and all additional files it uses, if any)
- **`328hw10-3-1.xhtml`** and **`328hw10-3-2.xhtml`**
- **`custom.css`**
- your current/latest version of **`*second-db*.sql`**