CS 328 - Homework 9

Deadline

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

Purpose

To get more practice writing PHP documents that send requests to Oracle using OCI, to read and think about SQL injection, and to practice building a dynamic form widget and using bind variables to help thwart SQL injection.

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 **`9`**.

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 - more practice with PHP and OCI

Putting connection details into a helper function

Consider those steps we used to set up the needed arguments for, and then call, **`oci_connect`** in **`try-oracle.php`**: (I am pasting in the "short form" original of these, rather than the version with comments trying to explain each from **`try-oracle-explained.php`**):

```
    $os_username = substr($_SERVER["CONTEXT_PREFIX"], 2);

    $ora_php_username = "{$os_username}_php";

    $conn_username = "{$ora_php_username}[{$os_username}]";

    $ora_php_password =
        trim(file_get_contents("/home/{$os_username}/.oraauth"));

    $conn = oci_connect(username: $conn_username,
                        password: $ora_php_password,
                        connection_string: null,
                        encoding: "AL32UTF8",
                        session_mode: OCI_DEFAULT);
```

It might be nice to gather these in an easier-to-reuse function:

```
<?php
    /*-----
        function: hum_conn_no_login: void -> connection
        purpose: expects nothing,

            has the side-effect of trying to connect to
            Humboldt's Oracle student database based on where
            the PHP file resides,

            and, if successful, returns the resulting connection
            object,

            but, if NOT successful, ENDS the document and exits the
            current PHP document...! (yes, it can do that...!)

        uses: 328footer-plus-end.html
        last modified: 2025-04-06
    -----*/

    function hum_conn_no_login()
    {
        // get part of the username from where this is installed

        $os_username = substr($_SERVER["CONTEXT_PREFIX"], 2);
```

```
// but the Oracle account you can log into using OCI is your
//     username plus _php

$ora_php_username = "{$os_username}_php";

// but, to ask to use blah_php's password to log in as blah,
//     we need to express it in the form blah_php[blah]

$conn_username = "{$ora_php_username}[{$os_username}]";

// grab the password from this user's .oraauth

$ora_php_password =
    trim(file_get_contents("/home/{$os_username}/.oraauth"));

// now: oci_connect expects:
//    a username,
//    a password,
//    a connection string (can be null in this particular
//                         approach, so PHP can build it from
//                         environment variables),
//    the desired character encoding (we'll use "AL32UTF8"),
//    the desired session mode (we'll use the default, the
//                              PHP constant OCI_DEFAULT)

$connectn = oci_connect($conn_username, $ora_php_password,
                        null, "AL32UTF8", OCI_DEFAULT);

// complain and exit at least somewhat gracefully if
//     oci_connect fails to make a connection

if (! $connectn)
{
    ?>
    <p> Could not log into Oracle, sorry! </p>
    <?php
    require_once("328footer-plus-end.html");

    // exit this PHP now -- this is reasonable
    //     when you have hit an error and there is NO
    //     point in going forward

    exit;
}

// if reach here, I connected!
```

```
        return $connectn;
    }
?>
```

The above function has now also been posted, in a file named **`hum_conn_no_login.php`**, along with this homework. (So is the snippet-of-ending-HTML it uses, **`328footer-plus-end.html`**.) For convenience, you can also make copies of these to your current nrs-projects working directory with this command:

```
cp ~st10/hum_conn_no_login.php .  # remember the space and period!

cp ~st10/328footer-plus-end.html .
```

Posted example using the above function

Finally, there is an example, **`empl-ex.php`**, **using** `hum_conn_no_login` and `328footer-plus-end.html` along the way to querying the `empl` database, also posted along with this homework handout.

Like the Week 10 Lab Exercise, it happens to involve a query that results in multiple rows, and it also happens to build an HTML **`table`** element to tastefully display those rows.

Homework 9 - Problem 1 requirements

Consider the **bookstore** database, and think of a query that:

- projects **at least 3 different columns**, and
- selects **at least 3 different rows**.

Using the posted **`html-template.html`** as the initial basis, create a PHP document **`328hw9-1.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: **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 your local copy of file **`hum_conn_no_login.php`**
- **Within** the **`head`** element, after the `link` element specifying that this document is being styled by `normalize.css`, add a second **`link`** element following course style standards specifying that this document will then be further styled using a file **`328hw9-1.css`** that you are about to create for this problem.
  - **Do not include any inline or internal CSS rules in your `328hw9-1.php`.**

- Within the **`body`** element, include an **`h1`** element 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.)
- Query at least one of the tables **created by `create-bks.sql`**,
  - such that you **project at least 3 different columns**, and
  - such that you **select at least different 3 rows**.
- Display the queried results using a **`table`** element.
- Include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw9-1.php`**.

Your **`328hw9-1.css`** should meet the following requirements:

- Include a comment including at least its file name, *your* **name**, and the last-modified date.
- If you change any of the default foreground and background colors, make sure that, for any text atop a background, the **contrast** between their colors is **at least WCAG 2 AA Compliant** based on the tester at: https://snook.ca/technical/colour_contrast/colour.html.
- Include at least one rule adding an attractive **border** to the **`table`**, **`td`**, and **`th`** elements.
  - You also should make appropriate use of the **`border-collapse`** property to prevent a doubled-border in the resulting displayed table.
- Add additional CSS rules as desired to further attractively format elements of this document.
- Make sure your resulting **`328hw9-1.css`** validates as valid CSS.

Strict-validate your **`328hw9-1.php`**'s result by running it from a browser, viewing its source, copying and pasting that source into a file named **`328hw9-1.xhtml`**, and put the URL of your **`328hw9-1.xhtml`** into the validator.

Submit your resulting files:

- **`328hw9-1.php`** (and all additional files it uses, if any),
- **`328hw9-1.xhtml`**, and
- **`328hw9-1.css`**

Problem 2 - thinking about SQL injection

SQL injection is when someone tries to "inject" additional SQL clauses into a **dynamic** SQL statement -- one built at run time, especially one built based on user input.

READ OVER the handout "**Basics of Oracle/PHP Bind Variables**" that is posted along with this homework handout -- it talks a bit more about SQL injection, and describes one of the important tools for fighting it, **bind variables**.

To add more depth to your knowledge of SQL injection, consider the two articles:

- **"SQL Injection Attacks by Example"** and
- **"How security flaws work: SQL injection"**

...that also are posted along with this homework handout.

Read these articles; pay special attention to the "**Mitigations**" section near the end of article (1), and note that an example of article (2)'s prepared statements are statements using OCI's bind variables.

Then, in a **plain-text file `328hw9-2.txt`**:

- include your name
- list **at least THREE** important "take-aways" from these articles, and **for each**, explain **WHY** you chose it. (So, there are **SIX** PARTS to this -- your three selections, AND **WHY** you chose each.)

Submit your file **`328hw9-2.txt`**.

Problem 3 - practice building a dynamic `select`/drop-down widget and using bind variables

Remember that, posted along with this homework handout, there is a handout: "**Basics of Oracle/PHP Bind Variables**".

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

- creates a **`form`** with a **`select`** drop-down widget **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:

The posted example **`empl-ex.php`** includes an example of how PHP can be used to build a **`table`** element whose contents are the rows resulting from a **`select`** statement.

One could, similarly, use PHP to build a **`select`**/drop-down element whose **`option`** elements are based on the rows resulting from a **`select`** statement.

Then, one could more-safely request information based on the user's selected option from that submitted form with the help of **bind variables**.

Now, finally, it is time to replace the hand-written **`select`**/drop-down element in **`bks-isbn-choice.html`** with a dynamically-generated **`select`**/drop-down element!

Decide on **at least two** pieces of information a user might like to ask about a title available at your bookstore -- as just a *few* of the possible examples:

- What is the price and currently quantity of a particular title?
- Who is the author and publisher of a particular title?
- Who is the author, and what is the price, of a particular title?

Homework 9 - Problem 3 requirements

Make a **COPY** of your file `bks.css` from **Homework 7** - **Problem 5** 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 **`328hw9-3.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 9's version of **`bks.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw9-3.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 **`328hw9-3.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 **`328hw9-3.php`** should be a variation of your **`form`** from Homework 7 - Problem 5's **`bks-isbn-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, 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 **`328hw9-3.php`** should include the following:
  - Appropriately **sanitize** each piece of information submitted by this form.
  - Build a dynamic **`select`** statement that projects **at least two attributes**, and uses at least one **bind variable** (instead of concatenation!) in its **`where`** clause based on the user's selection from the form's **`select`**/drop-down widget.
  - 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 a bind variable instead!
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw9-3.php`**.

Your Homework 9 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 **`328hw9-3.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw9-3.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw9-3-1.xhtml`**, and put the URL of your **`328hw9-3-1.xhtml`** into the validator.
- Put your **`328hw9-3.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 **`328hw9-3-2.xhtml`**, and put the URL of your **`328hw9-3-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw9-3.php`** (and all additional files it uses, if any)
- **`328hw9-3-1.xhtml`** and **`328hw9-3-2.xhtml`**
- **`bks.css`**.