CS 328 - Week 4 Lab Exercise - 2024-02-09

Deadline

Due by the end of lab.

Purpose

To practice with some more HTML form widgets, and to practice writing and running a PL/SQL subroutine.

How to submit

Submit your files for this lab using `~st10/328submit` on nrs-projects, each time entering a lab number of **84**.

- **Important note:** It is quite likely that your SQL files will be in a different directory than your HTML files, especially for the navigator. That's fine, and preferable!
  - Just remember that you need to run `~st10/328submit` from **EACH** directory with files to be submitted for this lab exercise.

Requirements

- You are required to work in **pairs** for this lab exercise.
  - This means **two** people working at **ONE** computer, one typing ("driving"), one saying what to type ("navigating"),

    while **BOTH** are looking at the **shared** computer screen and **discussing** issues along the way.
- Make sure **BOTH** of your names appear in each file submitted.
- When you are done, before you leave lab, somehow e-mail or copy the lab exercise files so that **BOTH** of you have copies, and **BOTH** of you should submit these files using `~st10/328submit` on nrs-projects, with a lab number of **84**.
- For this lab exercise's Problems 1-5, the **only** CSS permitted is **just** the external CSS `normalize.css` included in `html-template.html`.

Set-up for HTML Problems 1-5

Consider before you start:

You can approach today's lab exercise Problems 1-5 in a number of different ways.

You are going to be building a form, trying out several more kinds of form widgets, and seeing what name=value pairs are submitted by those form widgets when a form containing them is submitted.

More-Scattershot approach:

You can just have different "themes" for each example form widget you are practicing with in this problem.

More-Thematic approach:

You can decide on an overall "theme" for your form at the beginning, and add form widgets appropriate for that "theme" for each example form widget you are practicing with in this lab.

...and there are certainly possibilities between these two extremes, also.

If you are interested in being more thematic, note that the elements you will be practicing with are good for the following situations:

- **asking for EXACTLY one choice from a relatively-small number of exact choices** (do you want your sandwich on white, wheat, or gluten-free bread?)
- **asking for ZERO OR MORE choices from a relatively-small number of exact choices** (choose from the following condiments for your sandwich: mayo, mustard, ketchup)
- **asking for EXACTLY one choice from a possibly-larger-number of exact choices** (choose from exactly one of 7-10 sides to be included with your sandwich)
- **asking for possibly-multiple lines of text** (enter your opinion of your sandwich order)

You might want to consider (briefly) before starting your form, then: what is a setting where you might want to ask users for information such as the above?

Problem 1 - start a form

Starting from the **`html-template.html`** posted on the course public site and along with this lab exercise handout, create a strict-style HTML document that meets the class style standards as well as the following requirements:

- Include **`lab4`** somewhere in its file name, and give its file name the suffix **`.html`** .
- Fill in the opening comment block as specified, putting in your **names**, the last modified **date**, and the **URL** that can be used to run your document.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- Give the **`title`** element appropriate descriptive content.
- Include an appropriate **`h1`** element indicating this document is for the **CS 328 Week 4 Lab Exercise**
- Include **your last names** within a **`p`** element that you add to the **`footer`** element.

Add the following after the **`h1`** element you added above:

- START a **`form`** element
  - set its **`method`** attribute to **`"get"`**
  - set its **`action`** attribute to a "real" URL of your choice (because we haven't gotten to writing an actual application program to handle this form yet)
- This `form` element should contain a top-level **`fieldset`** element that contains an appropriate **`legend`** element of your choice, and within this should be:
  - a submit button (an **`input`** element with a **`type`** attribute whose value is **`"submit"`**)
- Remember: For this lab exercise, you may not use CSS to format or layout this form, although you **can** tastefully use **`br`** void elements or **`div`** or **`p`** elements if you wish.

validate your document-so-far

Before you go on...

- REMEMBER that it does not check for all of the CS 328 required style standards -- but you can use an `.xhtml` copy of your document with https://html5.validator.nu/ to assist you in validating and checking over your document.
- SAVE a copy of your page-so-far with your Problem 1 elements as follows -- at the nrs-projects prompt, type the Linux command:

  `cp` *your-file-name*`-lab4.html` *your-file-name*`-lab4.xhtml`

- Go to: **`https://html5.validator.nu/`**

  and paste in the absolute URL of the *your-file-name*`-lab4.xhtml` version of your file

  and click the **"Validate"** button.

  - SAD-BUT-TRUE THING: this validator seems to get overwhelmed frequently -- if you get an error **"Bad Gateway 502"**, that's likely the culprit.
  - IF you get this error, wait a little bit, then try again.

Problem 2 - add a section with logically-grouped radio buttons

Within your your form, within its top-level `fieldset`, before its submit button:

- Add another **`fieldset`** element containing **at least three logically-grouped radio buttons**, **exactly one** of which is initially shown as checked.
  - Be sure to include an appropriate **`label`** element for *each* radio button.
- **MAKE SURE YOU NOTICE**: what gets name=value pair gets submitted from a logically-grouped set of radio buttons!
  - Tip: if you can select more than one of your radio buttons at the same time, you have an error!

Validate an `.xhtml` copy of your document-so-far before you go on (if the gateway for **`https://html5.validator.nu/`** permits).

Problem 3 - add a section with checkboxes

Within your form, within its top-level `fieldset`, **after** Problem 2's `fieldset` but before the form's submit button:

- Add another **`fieldset`** element containing **at least three checkboxes**, **at least one** of which is initially shown as checked.
  - Be sure to include an appropriate **`label`** element for *each* checkbox.
- **MAKE SURE YOU NOTICE**: what gets name=value pair gets submitted from a selected checkbox!
  - Tips:
    - Make sure you can select all of your checkboxes, and that you see name=value pairs for each when you submit your form in that case.
    - Make sure you can un-select all of your checkboxes, and that you see NO name=value pairs for any of your checkboxes when you submit your form in that case.

Validate an `.xhtml` copy of your document-so-far before you go on (if the gateway for **`https://html5.validator.nu/`** permits).

Problem 4 - add a `select` element

Within your form, within its top-level `fieldset`, **after** Problem 3's `fieldset` but before the form's submit button:

- **ADD** a **`select`** element containing **at least five** options, **explicitly** specifying which is initially shown.
  - Be sure to include an appropriate **`label`** element for your `select` element.
  - In this case, write it so that the user can only submit a **single** value at a time (do not include the attribute `multiple`).
- **MAKE SURE YOU NOTICE**: what gets name=value pair gets submitted from a `select` element!
  - Tip: if you can select more than one of your `select` element's options at the same time, you are not following the requirements for this particular problem.

Validate an `.xhtml` copy of your document-so-far before you go on (if the gateway for **`https://html5.validator.nu/`** permits).

Problem 5 - add a `textarea` element

Within your form, within its top-level `fieldset`, **after** Problem 4's `select` element but before the form's submit button:

- Add a **`textarea`** element, with **at least three rows** displayed.
  - Include a **`placeholder`** attribute with an appropriate value.
  - Be sure to include an appropriate **`label`** element for your `textarea` element.
- **MAKE SURE YOU NOTICE**: what gets name=value pair gets submitted from a `textarea` element!
  - Tip: try entering more than one line of text in your `textarea` and then submit -- you will see special characters in the name=value pair representing the newline character and any spaces you entered.

Validate an `.xhtml` copy of your completed document before you go on(if the gateway for **`https://html5.validator.nu/`** permits).

- If necessary, let me know, submit what you have by the end of lab, and if you try later and find there was something you needed to fix, resubmit your fixed file and let me know, so that version will be graded instead.

Lab set-up for PL/SQL Problems 6-7

- On nrs-projects, if the driver has not previously executed **`set-up-ex-tbls.sql`** in their Oracle account, they should do so, so that they have the tables `empl`, `dept`, and `customer` in their database.
  - If needed, they can get a copy of this script using:

    ```
    cp ~st10/set-up-ex-tbls.sql .   # don't forget the blank and period!
    ```

- In a SQL script **`lab4.sql`**:
  - In opening comment(s), FIRST put the script name, both of your names, and today's date/last

modified date.

- Put in the SQL*Plus command:

  `set serveroutput on`

  ...so that you will see output from `dbms_output.put_line` statements you put into today's PL/SQL subroutine.

- Start spooling to a file **`lab4-out.txt`**:

  `spool lab4-out.txt`

  ...(and make sure you **`spool off`** at the script's end!)

- Put both of your names in a **`prompt`** command.

Problem 6 - write a PL/SQL stored procedure `empl_overview`

To get some practice writing a PL/SQL stored procedure, write a stored procedure **`empl_overvie`**`w` that meets the following requirements:

- It expects nothing (that is, it expects no arguments).
- It prints to the screen the following (in reasonable messages):
  - the current `sysdate`
  - the current number of employees
  - the average employee salary for the current employees
- Look in the posted SQL script **`hello.sql`** at the version of the stored procedure **`hello_world`** that we created during class on Wednesday -- after class, I added an opening comment block for that procedure.
  - Create an opening comment block for your procedure that has a `procedure:` part and `purpose:` part in the same style that you see here. (You don't have to give an `examples:` part, but you can if you wish.)
  - Follow that with the PL/SQL code creating your procedure.
- Remember to follow your PL/SQL procedure with:

  `/`

  `show errors`
- Then put a comment saying you are about to **test** your procedure `empl_overview`.
- Follow that with `prompt` command(s) stating that you are about to test `empl_overview` and describe what you should see if it is working properly.
  - (Your description should be specific enough that someone looking just at the spooled output can tell if the test passed or not.)
  - Then write a SQL*Plus command calling your procedure.
- Then, include this `delete` command:

```
delete from empl
where job_title = 'Clerk';
```

  - Follow this with `prompt` command(s) stating that you are about to test `empl_overview` on the case in which all the clerks were removed, and describe what you should see if it is working properly.
  - Then write a SQL*Plus command calling your procedure again.
  - Then call the command:

    `rollback;`

    ...to undo the deletion of the clerks!

- Finally, end your script by turning off spooling:

  `spool off`

If successful, your resulting **`lab4-out.txt`** should show that your procedure successfully compiled, and that its tests passed.

Problem 7 - demo that `empl_overview` IS now stored in the database

If, in Problem 6, your procedure `empl_overview` compiled without errors, this PL/SQL stored procedure is now **stored in your Oracle database** along with your tables, views, etc.

You do **not** have to recompile or recreate it next time you log into `sqlplus` -- it will persist!

To demonstrate this:

- Exit `sqlplus`, and then restart it.
- Type these commands in `sqlplus`:

  `set serveroutput on`

  `spool prob7-demo.txt`

  `prompt` ***include both of your names here***

  `exec empl_overview()`

  `spool off`

- Exit `sqlplus`.

Your resulting file **`prob7-demo.txt`** should contain both of your names, followed by the result of successfully running your stored procedure `empl_overview`.

BEFORE you leave lab:

Make sure that you **both** have copies of the files:

- *your-file-name*-**`lab4.html`**
  - that includes a correct URL in a comment for successfully executing this document from one of your nrs-projects account(s)
- **`lab4.sql`** and **`lab4-out.txt`**
- **`prob7-demo.txt`**

...and you BOTH submit these four files using `~st10/328submit` on nrs-projects, with a lab number of **84**.

How the navigator can get Problem 1-5's `.html` file:

- Because this happen to be a file the nrs-projects web server has to be able to reach, the navigator should be able to get a copy of Problem 1-5's `.html` file from the driver using an approach like this:
  - Assume the driver has username `ab12`, and the navigator has username `yz89`.
  - Also assume the driver created, in their `public_html` directory, a sub-directory named `328lab04`, and this sub-directory contains a lab file *your-file-name*`-lab4.html` to be submitted for the lab exercise.
    - The NAVIGATOR `yz89` can now:
      - log in to THEIR nrs-projects account

```
cd public_html
mkdir 328lab04   # or other name they choose
chmod 711 328lab04
cp ~ab12/public_html/328lab04/your-file-name-lab4.html . # note space & dot!
```

      - And now the navigator `yz89` has their own copy of *your-file-name*`-lab4.html`.
- I will leave it up to the navigator to decide if they would like to UPDATE their *your-file-name*`-lab4.html` so its opening comment includes the URL to *their* copy, or if they want to leave the URL for the driver's copy.
  - HOWEVER: remember that you *will* lose some credit if this URL does not work when I or the grader paste your submitted file's URL into a browser, in either case.

How the navigator can get Problem 6-7's files:

These may be in a directory that is harder for the navigator to make a copy from.

For example -- they might be in a directory `328lab4` that is **not** a sub-directory of `public_html`.

Here is an approach for this:

- The **driver** should *temporarily* make the directory with these files world-executable, and these files world-readable -- assuming the driver is currently in their directory `328lab4`:

```
chmod 711 .   # notice the space and the period!
chmod 644 lab4.sql lab4-out.txt prob7-out.txt
```

- Now the **navigator** can copy these into a directory of their choice -- assuming they are within the directory they want to copy into:

```
cp ~ab12/328lab4/* .  # notice the space and the period!
```

- The **driver** and **navigator** should **BOTH** then **protect** these files:

```
chmod 600 lab4.sql lab4-out.txt prob7-out.txt
```

  ...and **both** can now submit these using `~st10/328submit` from the directory containing these files.