CS 328 - Week 4 Lab Exercise - 2026-02-12

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 procedure.

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:** Problems 6-7 are in a *different* directory from your HTML files for Problems 1-5.
- SO: Remember 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** concepts/issues along the way.
- Make sure **BOTH** of your names appear in each file submitted.
- When you are done, before you leave lab, **BOTH** of you should submit appropriate versions of these files using **`~st10/328submit`** on nrs-projects, with a lab number of **84**.
- For this lab exercise's problems, the **only** CSS permitted is 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 set of exact choices** (which of the following free toppings do you want on your sandwich?)
- **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`** .
  - Reminder: assuming the navigator is already in their desired **`public_html`** subdirectory for today's lab, this works for starting off this file:

    `cp ~st10/html-template.html` *your*-**`lab4`**-*file-name*.**`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 **`head`** element's **`title`** element appropriate descriptive content.
- Within the **`body`** element, add an appropriate **`h1`** element indicating this document is for the **CS 328 Week 4 Lab Exercise**.
- Within the **`footer`** element, add a **`p`** element whose content includes **your last names**.

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

- START a **`form`** element.
  - 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)
  - set its **`method`** attribute to **`"get"`**
- In the content of this **`form`** element, add:
  - a top-level **`fieldset`** element that, to start, contains:
    - an appropriate **`legend`** element of your choice
    - a submit button (an **`input`** element with a **`type`** attribute whose value is **`"submit"`**)

- Remember that, for this lab exercise, you may **not** use CSS to format or layout this form, although you **can** tastefully use **`br`** void elements, **`p`** elements, or **`div`** elements if you wish.

validate your document-so-far

Before you go on...

- Create an **`.xhtml`** copy of your document-so-far:

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

- Go to either https://validator.w3.org/nu or https://html5.validator.nu

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

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

  - Correct any issues that are shown before going on to Problem 2.

Problem 2 - add a `fieldset` with logically-grouped radio buttons

Along with this lab exercise handout, there is also a handout "Some additional HTML form widgets". This handout includes a section reviewing radio button basics.

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

- Add *another* **`fieldset`** element containing **at least TWO logically-grouped radio buttons**, **exactly one** of which is initially shown as checked.
  - Be sure to include an appropriate logically-related **`label`** element for *each* radio button.
- Try selecting each radio button, and:
  - Notice what happens when you click on the label for a radio button.
  - Tip: if you can select more than one of your radio buttons at the same time, you have an error!
  - Try submitting your form, and look over the name=value pair that appears at the end of your form's action URL as a result.
  - **Think about**: can you explain why that is the name=value pair submitted?

Validate an `.xhtml` copy of your document-so-far, correcting any issues that are shown, before you go on to Problem 3.

Problem 3 - add a `fieldset` with checkboxes

The handout "Some additional HTML form widgets" also includes a section reviewing checkbox basics.

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

- Add *another* **`fieldset`** element containing **at least TWO checkboxes**, **exactly one** of which is initially shown as checked.
  - Be sure to include an appropriate logically-related **`label`** element for *each* checkbox.
- Try selecting and unselecting each checkbox, and:
  - Notice what happens when you click on the label for a checkbox.
  - Try submitting your form, and look over the name=value pair that appears at the end of your form's action URL as a result.

  - **Think about**: can you explain why that is the name=value pair submitted?

Validate an `.xhtml` copy of your document-so-far, correcting any issues that are shown, before you go on to Problem 4.

Problem 4 - add a `select` element

The handout "Some additional HTML form widgets" includes a section going over drop-down box (**`select`** element) basics.

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

- **ADD** a **`select`** element containing **at least TWO** options, **explicitly** specifying which is initially shown.
  - (Normally, you would have more options! Only two are required for this lab exercise to help you complete the lab in a timely manner.)
  - Be sure to include an appropriate logically-related **`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="multiple"`).
- Try selecting an option from your **`select`** element, and:
  - Notice what happens when you click on the label for the **`select`** element.
  - Look over the name=value pairs that appears at the end of your form's action URL as a result when you submit your form.
  - **Think about**: can you explain why that is the name=value pair submitted for your **`select`** element?

Validate an `.xhtml` copy of your document-so-far, correcting any issues that are shown, before you go on to Problem 5.

Problem 5 - add a `textarea` element

The handout "Some additional HTML form widgets" includes a section going over **`textarea`** element basics.

Within your **`form`** element, *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.
- Try entering more than one line of text in your **`textarea`** and then submit.
  - Notice the name=value submitted for this: you should see special characters in the value part representing the newline character and any spaces you entered.
  - **Think about**: can you explain why that is the name=value pair submitted?

Validate an `.xhtml` copy of your document-so-far, correcting any issues that are shown, before you go on to Problem 6.

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

- On nrs-projects, **CREATE** a new directory **`328lab4-plsql`**, **protect** it, and go to it:

```
mkdir 328lab4-plsql
chmod 700 328lab4-plsql
cd 328lab4-plsql
```

- 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!
```

- Start your initial version of a SQL script **`lab4.sql`** by making a copy of the provided file **`lab4-START.html`**:

```
cp  ~st10/lab4-START.sql  lab4.sql
```

  - In the interests of time, this provides an **opening comment block** and **testing code** for the PL/SQL procedure you will be writing in Problem 6.

- In your file **`lab4.sql`**:
  - Replace **both** instances of **`PUT YOUR NAMES HERE`** (one in the opening comment, and one in a **prompt** command) with both of your first *and* last names.

FUN FACT: you can use `to_char` to print formatted numeric values

- For the argument to **`dbms_output.put_line`**, you can include an expression calling **`to_char`** with a **numeric** expression and a **numeric format** string (built similarly to the numeric format you could use with the SQL*Plus **`col`** command) to print to the screen a number formatted to a specified number of fractional places.
  - For example,

```
dbms_output.put_line( 'LOOKY:' || to_char(3.7, '99.999') );
```

    would print to the screen:

```
LOOKY:  3.700
```

  - (Why the extra space? My guess is that it adds a space for a possible minus sign.)

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

Consider this version of PL/SQL stored procedure **`hello_world`** (slightly added to after class, to include an assignment statement as well as what will be the CS-328-required opening comment block):

```
/*===
    procedure: hello_world: void -> void
    purpose: expects nothing, returns nothing, and,
        IF serveroutput is ON, has the side-effect of
        printing to the screen a cheery greeting, the
        current date, and the current value of local
        variable quantity.
===*/

create or replace procedure hello_world as
```

```
    curr_date date;
    quantity integer := 1;
begin
    select sysdate
    into   curr_date
    from   dual;

    quantity := quantity + 1;

    dbms_output.put_line('Hello, world! on ' || curr_date);
    dbms_output.put_line('quantity is: ' || quantity);
end;
/
show errors
```

(You can see an overly-commented version of this that includes testing calls posted with the Week 4 Lecture 2 class examples.)

To get some practice writing a PL/SQL stored procedure, in your file **`lab4.sql`**:

- **After** the provided opening comment block for this procedure,

  but **before** its provided **`/`** and **`show errors`** and testing code,

  add code to create-or-replace a PL/SQL stored procedure named **`empl_overview`**.

Here are the requirements for **`empl_overview`**:

- It expects nothing (that is, it expects no arguments).
- It returns nothing, but, IF **`serveroutput`** is set to **`on`**, it has the side-effect of printing to the screen the following (in reasonable messages):
  - the current **`sysdate`**
  - the current number of employees
  - the average employee salary for the current employees

If successful, the resulting file **`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*-**lab4**-*file-name*.**html**

  your-**lab4**-*file-name*.**xhtml**

  - that include a correct URL in a comment for successfully executing the .html version of this document from one of your nrs-projects account(s)
  - (that is, I will again leave it up to the navigator to decide if they would like to UPDATE their *your*-**lab4**-*file-name*.**html** and your-**lab4**-*file-name*.**xhtml** so its opening comment includes the URL to *their* copy, or if they want to leave the URL for the driver's copy)
- **lab4.sql** and **lab4-out.txt**
- **prob7-demo.txt**

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

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

(for a driver with username *dr12*, and a navigator with username *na89* - replace these with your *actual* usernames when you actually do this)

These are in a protected directory, **328lab4-plsql**, that is harder for the navigator to make a copy from.

Here is an approach for this:

- The **driver** should ***temporarily*** make their directory **328lab4-plsql** with these files world-executable, and these files world-readable:

  ```
  chmod 711 .   # notice the space and the period!
  ```

  ```
  chmod 644 lab4.sql lab4-out.txt prob7-demo.txt
  ```

- Now the **navigator** can copy these into a directory of their choice (that is **not** the same directory as the one with their copies of *your*-**lab4**-*file-name*.**html** and *your*-**lab4**-*file-name*.**xhtml**).

  For example, assuming that the navigator created their **328lab4-plsql** in their home directory, and that the navigator has **cd**'d to the directory they want to copy the driver's files into:

  ```
  cp ~dr12/328lab4-plsql/lab4.sql .  # notice the space and the period!
  ```

  ```
  cp ~dr12/328lab4-plsql/lab4-out.txt .
  ```

  ```
  cp ~dr12/328lab4-plsql/prob7-demo.txt .
  ```

- After the copies are successfully made, the **driver** and **navigator** should **BOTH protect** these files:

```
chmod 700 .   # notice the space and the period!
chmod 600 lab4.sql lab4-out.txt prob7-demo.txt
```

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

Reminder: how the navigator can get a copy of the .html file

(for a driver with username *dr12*, and a navigator with username *na89* - replace these with your *actual* usernames when you actually do this)

- Because this is a file the nrs-projects web server has to be able to reach, the navigator should be able to get a copy of this file from the driver using an approach like this:
  - Assume the driver has, in their `public_html` directory, a sub-directory `328lab04` containing *your*-**`lab4`**-*file-name*`.html`. (Adapt the following accordingly based on your driver's actual *your*-**`lab4`**-*file-name*`.html` location.)
  - The NAVIGATOR *na89* can now:
    - log in to THEIR (*na89*'s) nrs-projects account, and run these commands:

      ```
      cd public_html
      mkdir 328lab04   # or other name they choose
      chmod 711 328lab04
      cd 328lab04
      cp ~dr12/public_html/328lab04/your-lab4-file-name.html .  # note space & period!
      ```

  - Now the navigator *na89* should have their own copy of *your*-**`lab4`**-*file-name*`.html`
- Either repeat for *your*-**`lab4`**-*file-name*`.xhtml`,

  or (if desired) change the URL within the navigator's *your*-**`lab4`**-*file-name*`.html` to refer to the navigator's copy, and then make the `.xhtml` copy (and double-check that it still validates):

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