

# CS 328 - Homework 9

## Deadline

Due by 11:59 pm on **Sunday, April 17, 2016**

## How to submit

Submit your files for this homework using `~st10/328submit` on nrs-projects, with a hw number of 9.

## Purpose

To read and think more about security and usability, to practice writing a PL/SQL trigger, to practice more with sessions, and to practice a little with client-side JavaScript.

## Important notes

- **NOTE:** you are welcome to use `require_once` and `include_once` 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!
- Also note: I hope to have you present versions of some of these documents to the class at some point.
- Now there is an introductory set of **CS 328 PHP Coding Standards**, as discussed in class and as given on the public course web site -- still quite subject to change, so keep an eye on it.
- Remember to follow the **CS 328 SQL and PL/SQL Coding Standards** as given in the CS 328 Homework 2 handout and the public course web site for all SQL and PL/SQL code.
- Make sure that you have executed the scripts `create-bks.sql` and `pop-bks.sql`.
- Remember to follow the **CS 328 HTML5 Coding Standards** as discussed in class and as given on the public course web site for all HTML5 documents.
- Remember to follow the **CS 328 CSS3 Coding Standards** as discussed in class and as given on the public course web site for all CSS3 that you write, also.

## Problem 1

Consider the OWASP XSS (Cross Site Scripting) Prevention Cheat Sheet:

[https://www.owasp.org/index.php/XSS\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/XSS_Prevention_Cheat_Sheet)

Read this cheat sheet, especially section 1 - Introduction and section 2 - XSS Prevention Rules through Rule #2.

Then, in a file `hw9-1.txt`:

- include your name
- list at least three important "take-aways" from this cheat sheet, and for each, explain **WHY** you chose it.

Note that these may be posted to the course Canvas site. Submit your file `hw9-1.txt`.

## Problem 2

Consider the "SQL Injection Attacks by Example" article:

<http://unixwiz.net/techtips/sql-injection.html>

Read this article, paying special attention to the "Mitigations" section near the end.

Then, in a file `hw9-2.txt`:

- include your name
- list at least three important "take-aways" from this article, and for each, explain **WHY** you chose it.

Note that these may be posted to the course Canvas site. Submit your file `hw9-2.txt`.

## Problem 3

Consider the course text's Chapter 7, on web design, and consider the slides on "Web Design and Usability" from University of Washington CSE 190 - Spring 2008 posted in the Week 10 - Lecture 2 section of the public course web site, under "In-class examples".

In a file `hw9-3.txt`:

- include your name
- give at least **three** rules-of-thumb or guidelines from Chapter 7 that you want to make sure to remember as you design web sites/web applications OR that you wish others would remember as they do so
- give at least **three** rules-of-thumb or guidelines (different from those you selected from Chapter 7) from the posted University of Washington CSE 190 - Spring 2008 "Web Design and Usability" slides that you want to make sure to remember as you design web sites/web applications OR that you wish others would remember as they do so

Note that these may be posted to the course Canvas site. Submit your file `hw9-3.txt`.

## Problem 4

Consider PL/SQL triggers! We discussed them a bit in the Week 1 Lab, and there is an example trigger posted in the Week 1 Lab section of the posted in-class examples. But you haven't written one in a very long time -- and one would be of use for the bookstore scenario.

### ***FIRST: trigger reminders!***

Here are a few key trigger-related reminders:

- You don't call a trigger -- instead, it is executed/triggered when the specified action occurs to the database. Consider the following trigger headers:
  - This trigger, named `inventory_update`, will be executed/fired BEFORE each insert of a row into a table named `orders`

```
create or replace trigger inventory_update
  before insert
  on orders
  for each row
```

- This trigger, named `add_prof_ct`, will be executed/fired AFTER each insert of a row into a table named `prof`

```
create or replace trigger add_prof_ct
  after insert
  on prof
  for each row
```

- This trigger, named `clear_advisor`, will be executed/fired BEFORE each delete of a row into a table named `prof`

```
create or replace trigger clear_advisor
  before delete
  on prof
  for each row
```

- Oddly, IF you have local declarations for a trigger, you DO put declare after the trigger header and before those declarations:

```
create or replace trigger inventory_update
  before insert
  on orders
  for each row
declare
  -- declare section can be omitted if you do not want
  -- to declare any variables...

  amt_ordered      integer;
  item_ordered     integer;
  amt_in_stock     integer;
begin
```

- ...but just proceed to begin if you don't have any local declarations:

```
create or replace trigger add_prof_ct
  after insert
  on prof
  for each row
begin
```

- NOTE that Oracle is very concerned about preventing potential "circular" rule firings -- so it will NOT permit you to query the table on which the trigger is fired. I would not be allowed to query `prof` within `add_prof_ct`, for example.
- Within the body of a trigger, you can obtain the attribute values in the "new" row being inserted or updated with the syntax `:new.` preceding the name of the attribute whose value you want. Likewise, you can obtain the attribute values in the "old" row being deleted or updated with the syntax `:old.` preceding the name of the attribute whose value you want.
  - For example, the expression

```
:new.prof_id
```

...would be the value of the `prof_id` attribute that was just inserted into `prof` in the trigger `add_prof_ct`

- And as another example, the expression

```
:old.prof_id
```

...would be the value of the `prof_id` attribute that was just deleted from `prof` in the trigger `clear_advisor`

## ***NOW: your actual task***

Now you will write a PL/SQL trigger for the bookstore scenario.

Create a file `328hw9.sql`.

(Make sure that you have a copy of `pop-bks.sql` in the same directory as `328hw9.sql` -- it is called in this problem's testing script, to make sure the tests are run on "fresh", original versions of these tables, before the tests muck with them. Note that `pop-bks.sql` happens to already end with a `commit;` command.)

Start your `328hw9.sql` file with the following:

- comments containing at least your name, CS 328 - Homework 9, and the last-modified date
- include the command to set `serveroutput` on
- followed by a SQL\*Plus `spool` command to spool the results of running this SQL script to a file named `328hw9-out.txt`
- followed by a prompt command including your name

**Be sure** to `spool off` at the end of this script.

Now, consider the bookstore's `order_summary`, `order_line_item`, and `order_needed` tables.

When the stock of some title falls below its `order_point`, a row indicating that an order of this title is needed should be added to the `order_needed` table. (Indeed, `sell_book` makes sure that this happens if necessary when a book is sold.) When a row is added to the `order_needed` table, the current date is inserted for the `date_created` attribute of the new `order_needed` row, and the `date_placed` for this new row is `null` (because the order needed has not been placed yet).

`order_summary` and `order_line_item` hold the details of an order of titles from a publisher. Each row of `order_summary` represents an "overall" order, including such overall details of an order as the publisher that order is from, the unique order number, the date the order was placed, and the date the order is complete. And `order_line_item` gives the details for one of the titles being ordered as part of that order -- it indicates what order is involved, what line-number of that order this represents, which title is being ordered in this line of the order, and how many of that title are involved in this order.

So, consider -- when an order is placed in response to an `order_needed` row, surely the `order_needed` table's `date_placed` attribute ought to be the very value in the `order_summary`'s table's `date_placed` attribute for that order. Also, consider the `on_order` attribute of the `title` table -- this is also the proper time to set this attribute to 'T' for each ordered title, also, since now such a title is indeed on-order.

How might we ensure that these updates are made, if necessary, to the proper rows in the `order_needed`

and title tables?

We said very early in the semester that triggers can be used to enhance database integrity. And, indeed, we can use a trigger here for just that purpose.

What action should trigger a corresponding action? Not an insertion into `order_summary` -- that's overall information for the order, not the individual title for which an order is needed. But it might be handy, after each insert into `order_line_item`, to:

- see IF there is an `order_needed` for that line item's title that has a null value for `date_placed` -- if there is a pending `order_needed` row for that line item's title -- that could now be updated to be the date that the corresponding order was placed;
  - (be careful -- DON'T change the `date_placed` if it is NON-null, that is, for older, previously-handled `order_needed` rows!)
- change the `on_order` attribute for that line item's title to 'T', since it is now on-order.

Important additional information:

- DON'T assume that date is the current date -- someone might be entering in this order information the next day, for example, or on Monday after a Friday order.
- Note that an order for some title might be placed even though there isn't an "open" `order_needed` row for it -- the bookstore manager may choose to simply order more of a title for strategic reasons. So no row in `order_needed` would be updated in that case, although that title's `on_order` attribute should still be set to 'T'.

Within your `328hw9.sql`, design and implement this PL/SQL trigger `order_maint`.

**Separately** (outside of `328hw9.sql`) run the testing script `order_maint_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `order_maint_test_out.txt`, along with your files for this homework.

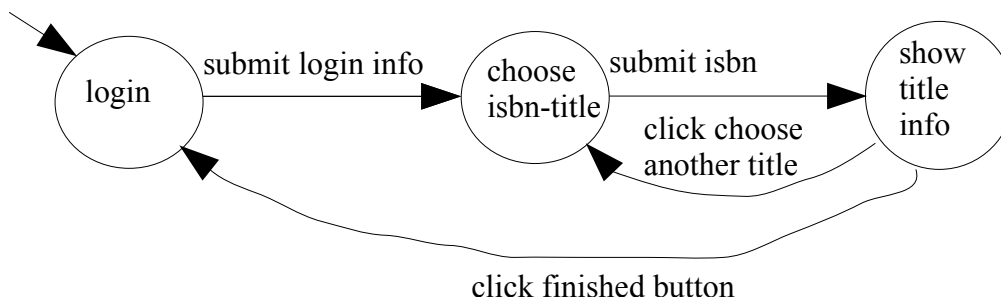
You may also add additional testing calls along with your `order_maint` code in your `328hw9.sql` script if you would like.

Submit your files `328hw9.sql`, `328hw9-out.txt`, and `order_maint_test_out.txt`.

## Problem 5

First: consider your external CSS3 file `bks.css` from the end of Homework 8. Make a **new** copy of this in a **different** directory, since you *might* be modifying it for this Homework 9 and you don't want to accidentally change how it styles Homework 8's files using it (since that could affect your HW 8 grade!)

Consider, for this problem, this finite state machine:



For this problem, you are to create a `title-info.php` document that uses PHP sessions in implementing the above finite state machine.

- All screens should be styled using your Homework 8 version of `bks.css` (and you can modify it as desired/needed).
- All screens should include, in their body element, your name, CS 328, and the name of your bookstore.
- It should use an appropriate "navigational" `if-elseif` statement to determine which screen to display.
  - Within your navigational `if-elseif`, USE an external PHP helper function for each screen of your application, as demonstrated in `try-trio.php` and `dept-fun.php`! (and additional helper functions/require files are encouraged, also!)(and of course submit all of those files also!)
  - Allowing the user the option to ask for the information about multiple titles before the session is complete will require careful thought about how this `if-elseif` should be structured.
- For the choose-isbn-title screen, you are expected to dynamically create, for all current ISBNs, a drop-down/select where each row's displayed content is an ISBN followed by that ISBN's title -- BUT the value is JUST the ISBN (the value submitted is just the ISBN).
  - we're displaying the title for easier human readability, but only submitting the unique ISBN because that's all that's needed to query for all of the rest of the selected title's information
- For the show-title-info screen, using the ISBN selected by the user, you are expected to query for the title, author, publisher **name**, price, and quantity on hand for that ISBN, attractively displaying the resulting information.
  - You should of course use a bind variable for the selected ISBN.
  - This screen will also include form-or-forms-with-submit-buttons to indicate the user's choice of what they wish to do next. I *\*think\** this can be a single form with two submit buttons with different `name` attribute values, or two different forms...
- When will you end/destroy the session? Here, either when the user clicks the "finished" submit button, or if something goes awry (if you end up in a state you shouldn't...) -- you don't want to automatically invalidate the session when you reach the show-title-info screen.
- Submit your resulting `title-info.php`, `bks.css`, and the at-least-three `.php` files containing the functions for your screens and any other PHP functions you decide to implement for this problem.

Do **NOT** add a link to your `title-info.php` from your `index.html` on nrs-projects; everyone is practicing pretty much the same thing here. DO include the URL one can use to run your `title-info.php` on nrs-projects in its opening comment.

Submit your resulting `title-info.php`, `bks.css`, and the at-least-three `.php` files containing the functions for your screens and any other PHP functions you decide to implement for this problem.

## Problem 6

The purpose of this problem is to practice with a little client-side JavaScript.

### 6 part a

Determine at least one type of numeric computation you would like to perform.

Then, design a page `number-fun.php` or `number-fun.html` (your choice) whose opening comment includes:

- the URL for running your version of this page

and whose `body` element visibly includes:

- your name
- CS 328
- at least two textfields (so the user can enter the needed numbers for the computation you chose)
- at least one `button` element (so the user can indicate that they would like for a computation to now be done)
- (you may also include additional elements as you would like)
- how will you show the result? You get to determine this. (You could display it in a textfield, for example, or within a paragraph, or within a textarea, etc.)

## 6 part b

Create an external CSS file `number-fun.css` to style your page `number-fun.{php or html}`.

- Include rule(s) to make sure your page is attractive and nicely laid-out.
- Include rule(s) that will make sure that all numeric textfields' contents will be right-justified (since that works well for numbers).

## 6 part c

Using unobtrusive-style JavaScript, write an external JavaScript `number-fun.js` to now perform the numeric computation(s) you decided upon, using `number-fun.{php or html}`'s textfield's contents when its `button` element is clicked, making sure to somehow show the computation's results to the user. Do what is needed for `number-fun.{php or html}` to use this external JavaScript `number-fun.js`.

Do **NOT** add a link to your `number-fun.{php or html}` from your `index.html` on `nrs-projects`; everyone is practicing pretty much the same thing here. DO include the URL one can use to run your `number-fun.{php or html}` on `nrs-projects` in its opening comment.

Submit your resulting versions of `number-fun.{php or html}`, `number-fun.css`, and `number-fun.js`.

## Problem 7

First: consider your external CSS3 file `custom.css` from Homework 8. Make a **new** copy of this in a **different** directory, since you *might* be modifying it for this Homework 9 and you don't want to accidentally change how it styles Homework 8's problem using it (since that could affect your Homework 8 grade!)

Then: consider the "additional database" you committed to in Homework 7, Problem 1, and your 3-screen application from Homework 8, Problem 4. Decide on an **additional** useful action for someone in its scenario that would involve dynamically querying and creating a form control based on that database, and then letting the user do something useful based on what they choose from that form.

For Homework 9, create PHP document `custom-session2.php`, styled using `custom.css`, which

meets the following specifications:

- This document should be one of those that can generate all of this application's pages, depending on the keys that exist in the `$_POST` and `$_SESSION` arrays when it is reached.
- Each page that this document generates should include, in its body element, your name, CS 328, and the name of your additional database's scenario.
- Its initial form should require that the user enter JUST an Oracle username and password.
- The action for the login form, when submitted, is to give the user a way to choose WHICH action (the one from Homework 8, or the new additional one you are adding for Homework 9) they would like to do.
  - whichever they choose, use the better-have-saved Oracle username and password from the initial form to dynamically-populate-and-build a form control within an appropriate resulting form for the user to respond to (along with other appropriate form controls for your situation).
- Once the user's action is done, they should have at least the options to either quit (end the session, return to the login form) or to return to the choose-which-action screen.
  - Optionally, you may have additional options (perhaps you want to give them the option of returning to the chosen-action's-form, for example).
- USE external PHP helper functions for each page of your application, as demonstrated in `try-trio.php` and `dept-fun.php`! (and additional helper functions/require files are encouraged, also!) (and of course submit all of those files also!)

**DO** add a link to your `custom-session2.php` from your `index.html` on nrs-projects, and **ALSO** include the URL one can use to run your `custom-session2.php` on nrs-projects in its opening comment.

Submit your `custom-session2.php` and Homework 9's `custom.css` and all files they require.