Assignment 5

Objectives

Create dynamically generated web pages based on user input.

Overview

This assignment will use the measurements example database as a subject. You will use the modules `db_access` and `db_util` that you created for Assignment #3

The measurements example is described in the Measurements Example page

You will create two modules that provide data from the database and then create a script that uses those modules.

Detailed Requirements

All script files and the database file should be in the web server root cgi-bin directory. All static web resources, such as web pages and style files, should be in the web server root document directory.

The only module that should use the database directly is `db_access`. All data needed by this project can be obtained through those functions. You will need to add two functions to `db_access` to support this assignment:

`def get_area_by_id(area_id):`
Return a list of rows from the area table that have the given area_id. This should never have more than one element. The list may be empty if area_id is not used by an area entity.

`def get_location_by_id(location_id):`
Return a list of rows from the location table that have the given location_id. This should never have more than one element. The list may be empty if location_id is not used by a location entity.

This makes a total of six data access functions in `db_access`. Please restrict yourself to using those, it will make your code easier to follow.

Use `style1.css` to style HTML pages, unless you would rather create your own.

Script `area_table.py`

Script `area_table.py` will generate a report similar to that produced in Assignment #3. The main difference will be that the table is presented using an HTML table so column widths will not be a problem. However, make sure to limit the number of decimal places displayed in the average measurement.

Script `area_selection.py`

Script `area_selection.py` will generate a form with a list of names of areas using the `select` tag with a size of 10. There will be a submit button that will cause Script `location_table.py` to be executed, sending the *id* of the selected area to the script using the name `area_id`.

Script `location_table.py`

Script `location_table.py` will generate a table of locations that belong to an area. The area will be identified by the area id, passed to the script with the name `area_id`. A header on the page should identify the area by name that was chosen. The locations will be listed in a table with the location id, the location name and the location altitude.

Each row will also have a radio button, in the first column. In the initial page, the first button should be checked. There will be a submit button that will cause Script `measurement_table.py` to be executed sending the id of the selected location to the script using the name `location_id`.

If there are no locations for the area, then the displayed page should display a suitable message rather than a table.

Script `measurement_table.py`

Script `measurement_table.py` will generate a table of measurements that belong to a location. The location will be identified by the location id, passed to the script with the name `location_id`. A header on the page should identify the area and location by name. The measurements will be listed in a table with the measurement id and the value of the measurement. The average of the measurements for that location should be listed on the page as well.

If there are no measurements for the location, then the displayed page should display a suitable message rather than a table.

Error Handling

Each script should be able to deal with errors in the submitted data:

- No id submitted
- Submitted id is not an integer
- Submitted id is not the id of an actual entity in the database

When `location_table.py` detects an error, it should redisplay the `area_selection.py` page with an appropriate message. When `measurement_table.py` detectc an error, it should display a generic error page with messages.[1]

Since the scripts that invoke other scripts will avoid these errors (except perhaps the first one), you job is to create pages that will check that the error handling works correctly.

Each such page will have a form with a text field having a value already filled in. There will be a submit button that will submit the value to the appropriate script. The idea is that the page can be loaded and submitted without any typing of values.

There will be six such pages needed: three error possibilities times two scripts (`location_table.py` and `measurement_table.py`). If you prefer, you can have six different forms on one page. Make sure, in either

case, that page names and page content clearly identify the test.

Home Page

Create a single web page with the name `home.html` which will contain links to:

- The script `area_table.py`
- The script `area_selection.py`
- The six error handling test pages

Environment

A database file, named `measures.sqlite` will be available in the directory from which the scripts will be run.

The scripts and unit tests will be run from PyCharm.

Submitting the Assignment

Because this is a complex project with many components: *please do not submit any files not directly related to this project*.

Please archive the entire project.

Sample Output

Assignment #5 Home Page

- area_table.py
- area_selection.py

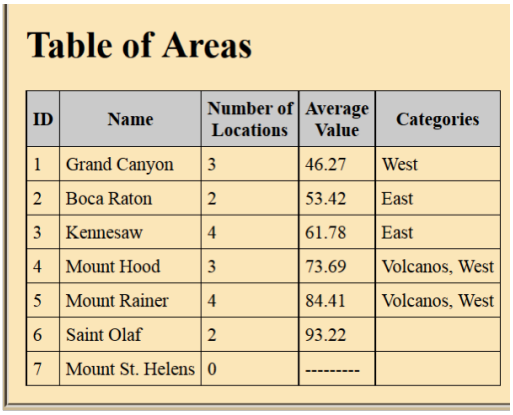
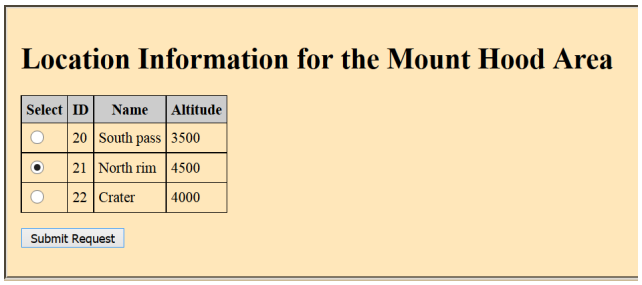
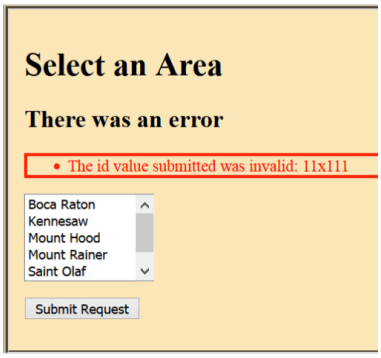
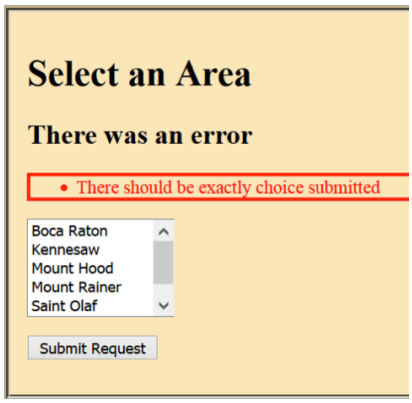


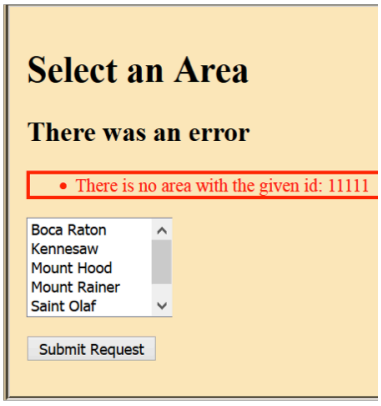
Table of Areas

ID	Name	Number of Locations	Average Value	Categories
1	Grand Canyon	3	46.27	West
2	Boca Raton	2	53.42	East
3	Kennesaw	4	61.78	East
4	Mount Hood	3	73.69	Volcanos, West
5	Mount Rainer	4	84.41	Volcanos, West
6	Saint Olaf	2	93.22	
7	Mount St. Helens	0	---------	









Due on October 26, 2015

1. It is difficult to get the appropriate information to redisplay the `location_table.py` page and get it to the display.
The `area_id` is needed to create location table.
However, that information is not being sent to `measurement_table.py` as it stands.
This is a use for the `hidden` input field. ↩