Lab 5: Uploading the Data

Due date: Thursday, May 5, in-class.

This is a **team lab**. Each group submits one set of deliverables.

Lab Overview

While most of your project development will happen outside the lab, we use this lab assignment to start he software development. You will develop the code to address the use cases that load the data into the database, in particular, the bulk-loading use cases.

Data

In support of this lab, and your futher software development activities, we are releasing the test dataset that you will use throughout the course. The dataset consists of five CSV (comma-separated values) files and a collection of XML files. The CSV files provide information about the freezer stock and about the pyrosequencing events. The XML files contain the pyrograms obtained during the pyrosequencing events. The specific files are:

- `HostSpecies.csv`. This file contains the list of host species.
- `FreezerStock.csv`. This file contains the freezer stock log, which specifies the information about each isolate in the freezer and its origin (host, sample).
- `Primers.csv`. This file lists all the primers used during the PCR reactions and pyrosequencing events reflected in the database.
- `DispSequences.csv`. This file contains the list of dispensation sequences used in pyrosequencing.

- `Pyroprints.csv`. This file contains the log of pyroprinting events, which specifies the contents of each well (which isolate was used) and the experiment conditions.
- `PyroPrintsXML.zip`. This file contains a list of XML documents produced by the Qiagen pyrosequencing equipment. Each file stores information about the pyrograms for **all** isolates sequenced during the specific experiment. (This information can be retrieved using the Pyroprint XML parser provided to you as part of this assignment).

Assignment

Each team shall implement complete preliminary versions of the following use cases from the current version of the Use Cases document:

1. **UC-1-1.**
2. **UC-1-2.**
3. **UC-2-1.**
4. **UC-2-2.**

Additionally, teams are strongly encouraged (you'll need to implement them at some point anyway) to implement these use cases:

1. **UC-2-3.**
2. **UC-2-4.**

"Complete preliminary" version means the following:

- **(Complete)** The full functionality of the use case shall be implemented.
- **(Preliminary)** The design of the actual web page/pages implementing each use case does not have to be complete/final.

Implementation notes

What is expected. Use case implementations shall have UI associated with them. Given the nature of the project, I expect that each task specified above can be accomplished by typing a URL on the web server of a team, loading the web form at that URL, providing the necessary information (file/directory names) and hitting the "Upload" button or its analog.

It may be possible to use the same URL/web page as the basis for multiple use cases, but it is not required.

Front page. To make grading/access to your system simpler, each team shall put a preliminary front web page for their application. For this stage of the project, include the links to all web pages implementing the specific use cases identified above on it. Note that neith the front page you are putting up, not the URL for the front page are expected to be permanent - once you start the main software development effort, you will be able to change either or both as you see fit.

Clear All button. To make testing and retesting easy, put on the front page of your web application a "Clear Data" button. When this button is pressed, your program shall execute a series of `DELETE FROM <Table>;` commands for each table in your database[1] to clean up the contents of the database. (You will, of course, remove this functionality later.)

CREATE TABLE statements. Your software shall work under the assumption that the database has been created. That is, you will run your `CREATE TABLE` statements off-line to create the database, before deploying the web-based software.

XML Parser. We have an XML parser designed to parse the Pyromark XML files and extract information needed for the database available for your use in the course[2] The parser is written in Java, so to use it with a PHP application you will probably need to write a Java program doing XML parsing and dumping results in some preliminary files, and then use the parser output within PHP (file I/O) to send data to MySQL (alternatively, if you believe you can insert tuples directly from Java, use JDBC connectivity).

Splitting Pyrogram upload. You may find it to be benefitial to split use case **UC-1-2** even further. Your system should be able to upload the data in the three pyrogram-related CSV files (primers, dispensation sequences, pyrogram log) even without the XML files provided. Similarly, you should be able to upload data in the XML files separately, **as long as the pyroprint log records have already been inserted in the appropriate table/tables**. (This way, you can get CSV file upload working without the need to worry about parsing XML, and deal with XML data in a separate code module.)

Error handling. Obvious errors (missing files/directories) shall be recognized and dealt with graciously. You can assume no errors in the CSV data, but if the formats are wrong (e.g., someone mistakingly put freezer log file name, where the pyroprint log file name was supposed to be), you need to be able to tell that and go back to the "waiting for input" mode.

[1] If your database contains any tables that do not participate in bulk insert procedures, feel free to leave them off the list.

[2] Thanks go to Aldrin Montana who wrote the parser and donated it for the use in the course.

While this is not really "error handling" per se, your code, from the very beginning shall have at least basic protection against SQL injection and buffer overrun attacks.

Deliverables

You will have some hard-copy deliverables and the software deliverable.

Code. Your main deliverable is operational code implementing the specified use cases residing on your team's virtual machine's web server. Put a link to the front page of your application on the team wiki page.

Old hardcopy deliverables. Submit the entire trail of your project documentation from Labs 2,3 and 4. This will yield a numeric grade for your Lab 4 performance.

New hardcopy deliverable. Submit the final version of your database design. A printout of your `CREATE TABLE` statements will suffice.

ChangeLog. Submit a changelog documenting any changes in the database design from the point of your last set of deliverables. If no changes were made, submit a change log specifying that.

On the wiki. To finish off the design stage, post the final version of your E-R diagram and the `CREATE TABLE` statements on the team wiki page. (Don't just attach files - embed the E-R diagram image, and put the SQL code on a wiki page).

Demo. Each team will demonstrate the operation of the data loading functions during the Thursday, May 5 lab period. Be ready to show (a) an empty MySQL database, (b) the operation of your web application, (c) the filled database after information was added to it.