CSCE 156 – Computer Science II

Lab 10.0 - JDBC in a Webapp II

Prior to Lab

1. Review this laboratory handout prior to lab.
2. Make sure that the Albums database is installed and available in your MySQL instance on CSE

Lab Objectives & Topics

Following the lab, you should be able to:

- Write SQL queries to insert into a database using JDBC
- Handle data integrity problems arising from bad user input and database integrity constraints (foreign key constraints)
- Have further exposure to a multi-tiered application using Java Servlets and the HTTP request/response protocol.

Peer Programming Pair-Up

To encourage collaboration and a team environment, labs will be structured in a *pair programming* setup. At the start of each lab, you will be randomly paired up with another student (conflicts such as absences will be dealt with by the lab instructor). One of you will be designated the *driver* and the other the *navigator*.

The navigator will be responsible for reading the instructions and telling the driver what to do next. The driver will be in charge of the keyboard and workstation. Both driver and navigator are responsible for suggesting fixes and solutions together. Neither the navigator nor the driver is "in charge." Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each week you should alternate: if you were a driver last week, be a navigator next, etc. Resolve any issues (you were both drivers last week) within your pair. Ask the lab instructor to resolve issues only when you cannot come to a consensus.

Because of the peer programming setup of labs, it is absolutely essential that you complete any pre-lab activities and familiarize yourself with the handouts prior to coming to lab. Failure to do so will negatively impact your ability to collaborate and work with others which may mean that you will not be able to complete the lab.

JDBC in a Web Application II

In this lab you will continue to work with the Java Database Connectivity API (JDBC) by finishing some new functionality to the Album web application from the prior lab. If you need to, you can check out the project for this lab from GitHub at `https://github.com/cbourke/CSCE156-Lab10`.

In this lab you will add functionality to the "Add Album" button/page to allow a user to enter an album by providing the title, band (by name), year, and album number. Again, the focus will be on the JDBC functionality for inserting these records, but you are invited to look into how the HTML web form posts to an XML configured (web.xml) Java Servlet (`AlbumAdderServlet.java`) which hands over the logic of inserting the album to the database using another Plain Old Java Object (POJO), `AlbumAdder.java`.

Your task is to make changes to the `AlbumAdder` class to take the four pieces of information provided by the user and insert records in the Album database (specifically the Albums and the Bands tables). Most of the code has been provided, but you will need to implement the `addAlbumToDatabase()` method as specified in the JavaDocs. Keep the following in mind.

- The Servlet provides all four arguments as strings. If the user does not enter valid data (non-integer values for year or album or empty strings), then you should handle those situations appropriately.
- The user should not be expected to know the internal keys to a database. Thus the page only asks them to provide a band name. To preserve the integrity of data, you should not insert duplicate band records. Therefore, you should check to see if the Band record already exists (according to its name). If it does, use that foreign key; if it doesn't, then you'll also need to insert the Band record.
- You cannot insert an album record into the Albums table without a valid band ID. This is because the Albums table has a foreign key reference to a band. This

is good database design it ensures good data integrity. Unfortunately, it presents a problem: we can insert a band record, but we need to immediately pull the auto-generated key of the inserted record so we can use it in the album record. There are many mechanisms for dealing with this, unfortunately a lot of them are vendor-specific (they work with one database, but not others).

One mechanism in mysql is the `LAST_INSERT_ID()` function which returns the ID of the last inserted record (within the session/connection) with an auto generated key. You can query the database using JDBC to call this function as follows.

```
//after you have inserted a record...
PreparedStatement ps;
ps = conn.prepareStatement("SELECT LAST_INSERT_ID()");
ResultSet rs = ps.executeQuery();
//increment to first (and only) record:
rs.next();
int foreignKeyId = rs.getInt("LAST_INSERT_ID()");
```

Deploying Your App

Refer to the instructions in the previous lab for how to generate a WAR file and deploy to glassfish. Recall that you need to use the JEE version of Eclipse and make changes to the `unl.cse.albums.DatabaseInfo` file to use your MySQL credentials.

Advanced Activities (Optional)

1. Every piece of data coming from a webform is represented with a `String`. The `AlbumAdderServlet` had to do the necessary conversion and handle the situation where invalid (non-numeric) data was provided. We could have also done the validation at the client tier to prevent bad data from being sent to the server. Add some Javascript to your page to validate the data to prevent the user from clicking the "Add" button unless the year and album number are valid.
2. A Cross-Site Scripting (XSS) attack is an attack that is similar to an SQL injection attack that involves an attacker who injects HTML into a webpage by submitting it as legitimate input but the application fails to properly *sanitize* the HTML when it gets served back to the user. We will demonstrate that our webapp is vulnerable to this attack by hacking it.
    a) Modify your Album database and extend the length of the `Band.BandName` column by opening your mysql client and executing the following SQL:

```
alter table Band change column name
  name varchar(2048) not null default '';
```

b) Navigate to the page where you enter albums/bands and enter the following data:

   - Album Title: *Yeah Ghost*
   - Band Name (exactly as shown, but no line breaks):

     ```
     Zero 7</a></td><td>2009</td></tr></table><p>For more,
     click <a href=
     "http://cse.unl.edu/~cbourke/cse156/passwordStealer.html">
     HERE</a><!--
     ```

   - Album Year: 2009
   - Album Number: 4

c) Submit the new Album and view the list–what has happened? Click the new link at the bottom and see what happens.

d) This attack was not that sophisticated. Imagine injecting JavaScript which operates in silence and in the background and sends all browser data to another site and you can get an idea of how severe the problem can be.

e) Read more on how to prevent these kinds of attacks:
`http://en.wikipedia.org/wiki/Cross-site_scripting`
`https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)`
Modify your project to protect against these sort of attacks.

3. Make this application more complete by adding more functionality so that users can insert songs, bands, etc. as well as delete records from the database.