CSCE 156 – Lab 10: Using JDBC in a Java Application

0. Prior to the Laboratory

1. Review the laboratory handout
2. Make sure that the Albums database is installed and available in your MySQL instance on CSE
3. Review the SQL and JDBC lecture notes
4. Review a JDBC tutorial from Oracle : http://download.oracle.com/javase/tutorial/jdbc/

1. Lab Objectives& Topics

Upon completion of this lab you should be able to understand how to:

• Write SQL queries for use in JDBC

• Make a JDBC connection, query and process a result set from a database

• 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 some exposure to a multi-tiered application

Note: it would be a good idea to reset your Albums database by rerunning the SQL script from a prior lab.

2. Problem Statement

In this lab you will familiarize yourself with the Java Database Connectivity API (JDBC) by finishing a simple, nearly complete retrieve-and-display Java application. The design of the application is simple: it displays all the albums on the main window, when user select an album and click “Show Songs” button, another window will pop up which shows the song tiles of all the songs in that particular album. It is not necessary to understand the details of the application (Java Swing). The main goal of this lab is to give you some familiarity with JDBC and exposure to a multi-tiered application.

3. Instructions

Importing Your Project

1. Download the zip archive file from Canvas and import it into Eclipse:
   a. File -> Import -> General -> Existing Projects into Workspace
   b. Select "Select archive file" and select the zip file you downloaded
2. The java code is located under the "Java Resources -> src->edu.unl.cse->sql" directory
3. The Java Swing files are located under the "gui" pakage

Modifying Your Application

1. You will first need to make changes to the edu.unl.cse.DatabaseInfo java source file:

   a. Change instances of YOURLOGIN to your cse login

   b. Change instances of YOURSQLPASSWORD to your mysql password (if you have misplaced it, it can be reset by going to http://cse.unl.edu/account)

2. All of the Java code has been completed except for three methods in the AlbumBean class and AlbumAdder class. You can make any modifications to the other classes that you feel will help you, but modifying code in one part may break functionality in another. The two methods that you will need to implement are:

   - public static Album getDetailedAlbum(String albumTitle) – this method will query the database for the specific album with the given information and return an Album object with all of its fields specified.
   - public static List getAlbums() – This method will query the database and get a complete list of all Albums in the database. It will create and populate Album objects and put them in an ArrayList which will then be returned.
   - Public Boolean AddAlbumToDatabase()-this method takes the four pieces of information provided by the user and insert records in the Album database (specifically the Albums and the Bands table. **Note that the AlbumAdder class is instantiated with the four pieces of data that you'll need. Keep in mind however:**

     1. The AlbumAdder has four attributes that are all String data. Your database expects integers for some fields. You will need to handle the validation somehow.

     2. The user should not be expected to know the internal keys to a database—thus the input for asks them to provide a band name. To preserve the integrity of data, you should not insert duplicate band names. 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.

Hints & Tips

1. Refer to the course lecture slides on how to make a JDBC connection and query

2. Refer to the prior lab(s) for the MySQL database schema

3. Refer to the BandBean class for another JDBC example

4. Important: do not forget to close your database resources (especially connections) after you are finished using them.

5. To run the application, just simply choose AlbumTable.Java and run it as a Java application.

6. A main method has been included in the AlbumFactory(AlbumBean) class that you should use to test your method implementations before you test your Java App. Make use of this method and modify it as needed to ensure your code works.

Relations & Keys

You cannot insert an album record into the Albums table before you have inserted a band record into the Bands table. 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 with an auto generated key. You can query the database using JDBC to call this function as follows:

```
PreparedStatement ps = conn.prepareStatement("INSERT INTO Table (…) VALUES (…)");

ps.set…

ps.executeUpdate();

ps = conn.prepareStatement("SELECT LAST_INSERT_ID()");

ResultSet rs = ps.executeQuery();

rs.next();

foreignKeyId = rs.getInt("LAST_INSERT_ID()");
```

4. Completing Your Lab

1. Complete the worksheet and have your lab instructor sign off on it