CSE 305: Principles of Database Systems
Guidelines for Project Assignment No. 3: Online Movie Rental System
How to Use the TA's API to the Project GUI

Section 1: Introduction

The aim of the CSE 305 programming project is to create an Online Movie Rental System. In Project Assignment No. 1, you designed the relational database system to support the activities of an online movie rental system. In Project Assignment No. 3, you will use HTML for the user interface, JDBC for MySQL database connectivity, and a mechanism to map the results from JDBC queries to the HTML pages.

The HTML user interface is provided by the TA as part of the project. It contains all the required screens for performing activities related to the online movie rental system, including those tailored for Manager, Customer Representative and Customer Level transactions (based on the login information). The HTML section is contained in JSP files, which handle the logic to render the data in a readable manner.

The JDBC section, which connects with a MySQL database, has not been implemented in the provided project code. It has been replaced with *Sample Data*. For each of the database transactions -- create, read, update and delete -- sample data is sent to the HTML pages for rendering. This data is encapsulated in Java Beans (Java Objects) and sent to the HTML pages. Mapping of this Sample Data with the HTML pages is handled by the Servlets.

Students have written all required MySQL queries for Project Assignment No. 2. You must use this knowledge to implement JDBC connectivity with appropriate databases, fetch/update the data and encapsulate the results in the form of Java Objects. The application logic of rendering the data

from the Java Objects to the GUI is handled by the existing project code provided by the TA. This document explains the various sections of the project and their importance. Sample JDBC code can be found in Section 4, so that you can get acquainted with the portion of the project you have to implement.

Section 2: Installing and Running the Project

1. Installation instructions:
    a. Download and install Eclipse (https://www.eclipse.org/downloads/) - version Oxygen recommended
    b. In Eclipse, click on **File** -> **Import** from the Eclipse Main Menu
    c. Expand **General**, select **Existing Projects into Workspace**, and click **Next**
    d. Make sure that **Select Archive File** is checked and browse for the ZIP file provided with this project
    e. Click **Finish**
    f. Download and install Apache Tomcat Server 9 (https://tomcat.apache.org/download-90.cgi) in Eclipse
2. Running the project:
    a. Go to **src/main/webapp/index.jsp**, right click and click on **Run As** -> **Run on Server** to start the project
3. Project guidelines:
    As shown in Figure 1, the project is logically divided into packages (in **src/main/java** folder). The details about each package are as follows:
    a. *resources package*:
    It contains controllers required to transfer data to and from the JSP pages (UI) and the database code. The controllers use "Get" and "Post" methods to collect data and they send the data using request.setAttribute method. The UI redirection is done using a Request Dispatcher.

b. *dao package:*

   This Data Access Object package contains application logic to fetch data from the database and return the data by encapsulating it in an object. A framework for all the functions the students need to implement is provided. Sample data is used with the current framework and the students are expected to make changes in the applicable code to fetch data from the database or update the data in the database. Each method has “Sample code begins” and “Sample code ends” demarcation. The students are expected to delete the code between these demarcations and write their own implementation with database connectivity. **Students are required to make changes in this package.**

c. *model package:*

   This package contains the structure of all the tables in the database along with the columns. Each column is represented as an instance variable of the corresponding class. Each instance variable has a getter and setter function which is useful for fetching/storing the values of the objects. These functions are to be used in the corresponding dao package classes. **Students are not required to make changes in this package.**

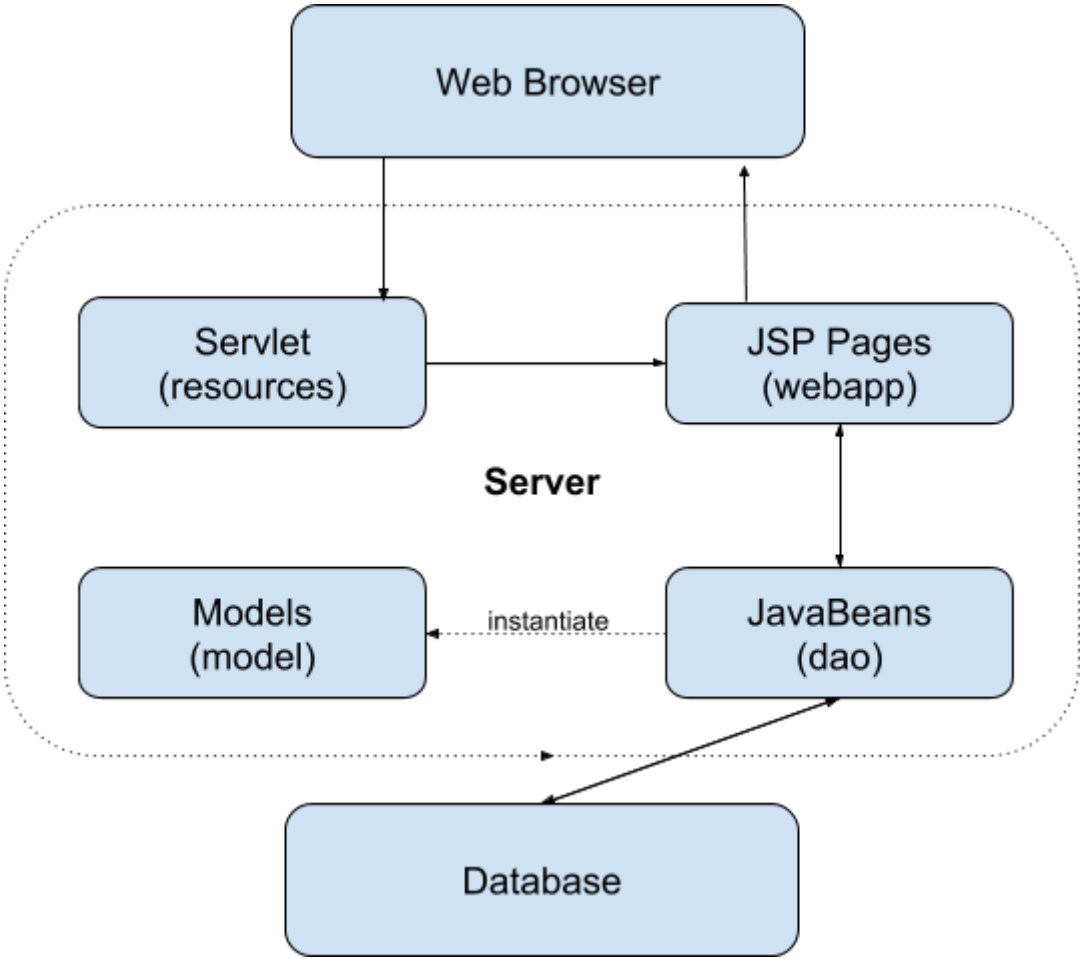

Figure 1: Architectural Diagram of the CSE 305 Database Project

Section 3: Architectural Diagram Overview

An architectural overview of Project Assignment 3 using the TA's UIis given in Figure 1. The following points are worth noting.

1. The JSP pages are rendered on the Web Browser. These pages are hosted on a Tomcat server. The JSP pages receive data in the form of *request attributes* and send data as *response attributes*. Iterating over the data in a result set is implemented using JSTL tags.

2. Any request/response to/from the web browser is directed towards the servlets. The servlets are responsible for parsing the data and further redirecting it to the JSP/JavaBeans for processing.
3. JavaBeans instantiate the models, which are class representations of the SQL tables. The JDBC connectivity and application logic related to the database access are performed here. All data is encapsulated in model instances. The details are mentioned in Section 2: 3(b).

Section 4: Example dao Code

We present dao code to display the contents of the Customer table. This code can be found in the **CustomerDao** class within the *dao* package. We first show the existing dao code for this, which simply uses sample data (not retrieved from the database):

```
public List<Customer> getCustomers(String searchKeyword) {
    /*
     * This method fetches one or more customers based on the searchKeyword and returns it as an ArrayList
     */

    List<Customer> customers = new ArrayList<Customer>();

    /*
     * The students code to fetch data from the database based on searchKeyword will be written here
     * Each record is required to be encapsulated as a "Customer" class object and added to the "customers" ArrayList
     *
     */

    /*Sample data begins*/
    for (int i = 0; i < 10; i++) {
        Customer customer = new Customer();
        customer.setCustomerID("123");
        customer.setAddress("123 Success Street");
        customer.setLastName("Lu");
        customer.setFirstName("Shiyong");
        customer.setCity("Stony Brook");
        customer.setState("NY");
        customer.setEmail("shiyong@cs.sunysb.edu");
        customer.setZipCode(11790);
        customer.setTelephone("5166328959");
        customer.setCreditCard("1234567812345678");
        customer.setRating(1);
        customers.add(customer);
    }
    /*Sample data ends*/

    return customers;
}
```

As can be seen, we are creating Customer model objects and setting values to the variables. We then append the data to a customers List and return the list. We are using Sample Data to set the variables.

The following snippet shows how to replace this Sample Data with the necessary Java/JDBC code:

```
public List<Customer> getCustomers(String searchKeyword) {
    /*
     * This method fetches one or more customers based on the searchKeyword and returns it as an ArrayList
     */

    List<Customer> customers = new ArrayList<Customer>();

    /*
     * The students code to fetch data from the database based on searchKeyword will be written here
     * Each record is required to be encapsulated as a "Customer" class object and added to the "customers" ArrayList
     *
     */

    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/demo", "root", "root");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select * from customer where FirstName like \'%" + searchKeyword + "%\'"
                + "or lastName like \'%" + searchKeyword + "%\'");
        while(rs.next()) {
            Customer customer = new Customer();
            customer.setCustomerID(rs.getString("CustomerId"));
            customer.setAddress(rs.getString("address"));
            customer.setLastName(rs.getString("LastName"));
            customer.setFirstName(rs.getString("FirstName"));
            customer.setCity(rs.getString("City"));
            customer.setState(rs.getString("State"));
            customer.setEmail(rs.getString("Email"));
            customer.setZipCode(rs.getInt("ZipCode"));
            customer.setTelephone(rs.getString("Telephone"));
            customer.setEmail(rs.getString("Email"));
            customer.setCreditCard(rs.getString("CreditCard"));
            customer.setRating(rs.getInt("Rating"));
            customers.add(customer);
        }
    } catch(Exception e) {
        System.out.println(e);
    }

    return customers;
}
```

In this code snippet, we have replaced the contents between “Sample data begins” and “Sample data ends” with JDBC code that connects with a database called “demo”, using username “root” and password “root”. Using the `searchKeyword` given as a function parameter, a query to search the “customer” table to find records having a similar First or Last Name is executed. The result of this query in ResultSet `rs` is iterated and

each record's data is stored in the `customer` object. The `customer` object is then appended to the `customers` List.

Both of the above code snippets are similar in that they store data in objects. In the first code snippet, we store Sample Data; in the second code snippet, we store the data retrieved from customer table. The rest of the code remains the same and the students are required to change only the code between "Sample data begins" and "Sample data ends".

Section 5: Project Assignment 3 Submission Instructions

The students are expected to submit the completed project with implementations of their JDBC code in the project, electronically. Below are the steps to create a WAR file of the project and uploading it:

1. In Eclipse, select the project in **Project Explorer**. Click on **File -> Export**. Type "war" in the search box and select **WAR file** and click on Next. Check the 'Export source files' checkbox. Complete the Export process and the WAR file will be created.
2. If you are using MySQL Workbench for browsing the MySQL tables, then use this link (https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html) for steps to export the database as a **Self-Contained File**.
3. On GitHub (www.github.com), create an account, if not already created, using the CS email ID.
4. Create a **private repository**, add a **README** file. In the README file, include the project member details.
5. Add the generated WAR file in the repository.
6. Add the database dump in the repository.
7. Add the TA as a **Collaborator** in the project repository. The GitHub username of the TA is *gaofengdeng*.

8. Send an email to the TA (gaofeng.deng@stonybrook.edu) after completing this process.
9. Unlike the version of PA3 where you develop your own GUI, you do not need to demo your assignment to the TA. You are only required to submit it electronically as described in steps 1-7.

One member of each project team is expected to perform the above steps to submit the completed project for the team.

Section 6: Extra Credit for Improving the TA's GUI

For those of you who want to delve deeper into the GUI development aspects of the project, extra credit will be given to those teams that improve upon the TA's GUI, the JSP code for which can be found in module *webapp* (see Figure 1). **If you improve the GUI, please mention the improvement details in your project README (see Section 5).**

There will not be any demos for Project Assignment No. 3, assuming you are using the TA's API for the project GUI. If you are making improvements to the TA's GUI, the TA (Gaofeng Deng) will run your code on his local machine to view the results of your improvements. See Gaofeng during his office hours if you have any questions.