5/30/2023

Hansani Peiris

[company name]

[Document title]

[Document subtitle]

# Student Profile

# Introduction

# UML diagrams

## ER Diagram

## Class Diagram

# MVC

# Code Segment

## header.jsp and footer.jsp

header.jsp and footer.jsp pages are formed separately as 2 jsp pages in order to avoid repeating the same code in every page lengthening the size of the code. The header and footer are connected to every page by the below code.

****

****

**A screenshot of a computer

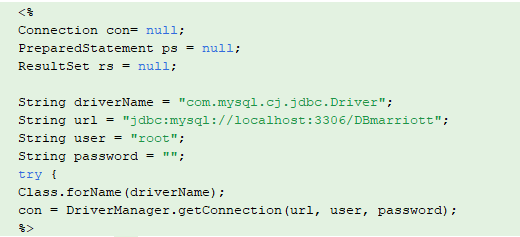
Description automatically generated with low confidence signup.jsp**

**A screen shot of a computer

Description automatically generated with medium confidence**

* This handles the insertion of employee data into the employee table “DBmarriott” MySQL database. It retrieves the employee data from the request parameters, such as id, firstName, lastName, username, passwordInput, position, and email.
* If the update operation is successful (rowsAffected > 0), it redirects the user to the "login.jsp" page and if the update operation fails, it prints a message indicating the failure.
* login.jsp and signup.jsp are connected to the same database table “employee”. Only already inserted ids(in the database table “employee”) can be login through the login page. New users can sign up then login since then their ids get inserted in the database table.

**login.jsp**

****

* This code sets up the database connection using JDBC and initializes variables for the driver name, URL, username, and password.
* login.jsp is connected to the loginbean.jsp. When the login button is clicked it connects to the loginbean.jsp .
* login.jsp" is the presentation layer or the user interface of the login functionality.
* It displays the login form to the user, collects the username and password input, and submits the form to the server for processing.
* This login.jsp JSP file contains , JavaScript, and JSP tags to create the login form and handle user interactions.

**loginbean.jsp**

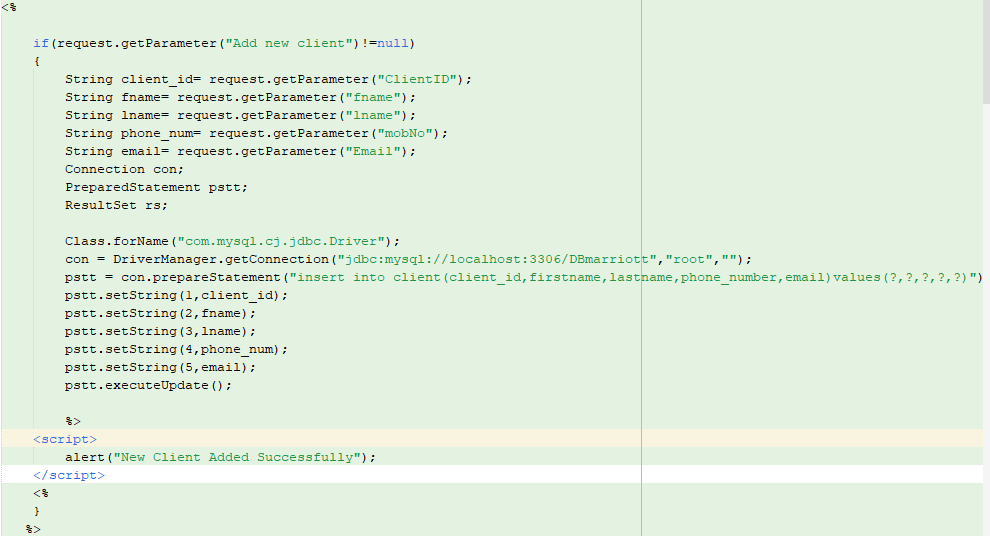
* "loginbean.jsp" is the server-side logic for processing the login functionality.
* It handles the authentication and validation of user credentials against a database.
* It interacts with the database to retrieve user information and validate the credentials.
* It sets session attributes or redirects the user to different pages based on the login outcome (success or failure).





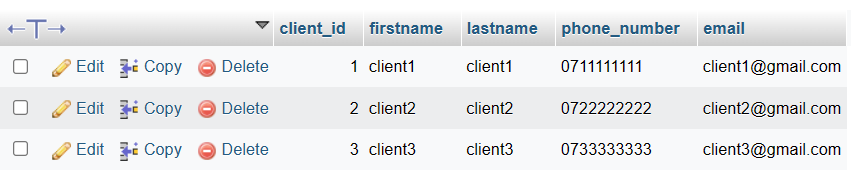
* This code declares and initializes variables for the username (userdbName) and password It defines an SQL query to select records from the "employee" table based on the provided username and password. Also it retrieves the username and password parameters from the request .MoreoverIt checks if the provided username and password are not empty or null.
* If the conditions are met, it connects to the database, constructs the SQL statement, sets the parameters, and runs the query. If the result set contains any rows, it obtains the username and password from the result set and compares them to the credentials provided.
* If the credentials are correct, it stores the username in the session and redirects the user to the "home.jsp" page and if the credentials do not match, the user is redirected to the "error.jsp" page Also if the entered username or password is empty or null, an error message is displayed.

**Client.jsp**



The Manage Client Page is connected to the database from this code. This JSP code block handles the addition of a new client to the MYSQL database "DBmarriott" .

* It checks if a parameter named "Add new client" is present in the request and If the parameter is found, it retrieves the relevant data from other request parameters such as "ClientID", "fname", "lname", "mobNo", and "Email".
* This establishes a database connection using JDBC (Java Database Connectivity) with the MySQL database "DBmarriott" on the local host.
* SQL INSERT statement inserts the client data into the "client" table in the database with columns: "client\_id", "firstname", "lastname", "phone\_number", and "email" and this sets the parameter values for the prepared statement using the retrieved data.
* It executes the INSERT statement using the ‘executeUpdate()’ method, which inserts the new client record into the database. If the insertion is successful, it displays a JavaScript alert message to indicate that the new client was added successfully.



When a new client is added, it shows in the client table in the database as above.

A screenshot of a computer code

Description automatically generated with low confidence

A screen shot of a computer code

Description automatically generated with low confidenceThis code retrieves data from the "client" table in the "DBmarriott" database and displays it in an HTML table format as below.

A screenshot of a phone number

Description automatically generated with medium confidence

In the above code, it establishes a database connection using JDBC (Java Database Connectivity) with the MySQL database "DBmarriott" on the local host and SQL SELECT query string to retrieve all records from the "client" table.

It retrieves the values of columns "client\_id", "firstname", "lastname", "phone\_number", and "email" using getString() method and stores them in respective variables.

Also this code provides links to edit or remove the client record using the client ID. The client ID is passed as a query parameter in the URLs of the edit and remove links.

**Clientupdate.jsp**

**A screenshot of a computer

Description automatically generated with medium confidence**

* When the Edit link is clicked in any row of the table on the Manage Client page and directed to Clientupdate.jsp , this code handles handle the update of client details in the MySQL database. It is triggered when the Update button is clicked.
* It checks if a parameter named "Update" is present in the request and the parameter is found, it retrieves the updated client details from other request parameters such as "ClientID", "fname", "lname", "mobNo", and "Email". It also establishes a database connection using JDBC.
* SQL UPDATE statement updates the client details in the "client" table. The statement sets the values of the "firstname", "lastname", "phone\_number", and "email" columns based on the client ID.
* If the update is successful, it displays a JavaScript alert message to indicate that the client details were updated successfully.
* This code is to retrieve the pre-inserted client details in order to update/ edit the details. It It appears to retrieve client information from the "client" table in the MySQL database based on the "id" parameter passed in the request. In Client.jsp, the client\_id is passed as the “id” in the link to Clientupdate.jsp however in here It retrieves the value of the "id" parameter from the request using request.getParameter("id") and assigns it to the variable client\_id to retrieve the specific record.

A screen shot of a computer

Description automatically generated with medium confidence

**A screenshot of a computer code

Description automatically generated with low confidence**

* This generates an HTML form with input fields for each client attribute, pre-filling the fields with the retrieved values using embedded Java code (<%= %>) to output the values dynamically.

**Clientdelete.jsp**

This code handles the deletion of a client record from the MySQL database. The client \_id is passed as a query parameter.

**A screenshot of a computer

Description automatically generated with medium confidence**

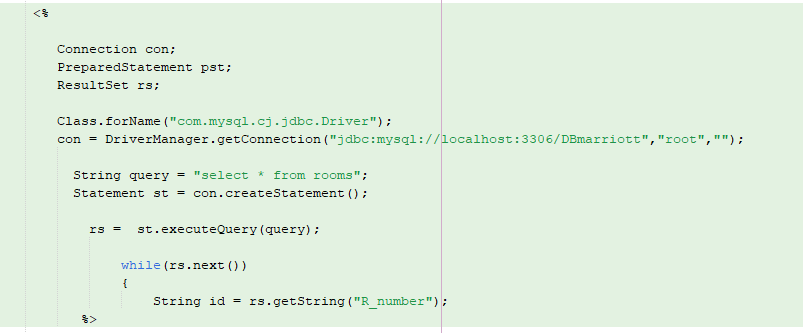
**Rooms.jsp**

****

The Manage Rooms Page is connected to the database from this code. This JSP code block handles the addition of a new room to the rooms table in MYSQL database "DBmarriott" .

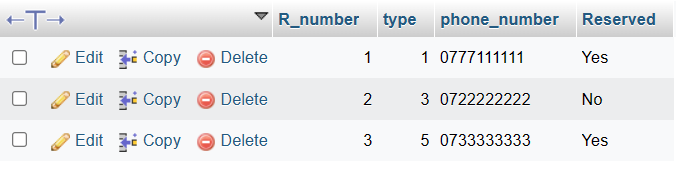
* When the form in the Manage Rooms is submitted by “ Add new room” button it checks the request parameter and retrieves the values of request parameters "RoomNo," "TypeID," "mobNo," and "Reserved."
* This establishes a connection to a MySQL database using the JDBC driver and establishes a connection to a MySQL database using the JDBC driver. SQL statement inserts the room details into the "rooms" table in "DBmarriott" database and also this code sets the values for the SQL statement using the retrieved request parameters.
* ‘executeUpdate()’ method executes the SQL statement. After successfully inserting the data into the table, it displays a JavaScript alert message saying "Room Adddeddddd”.

A screenshot of a computer program

Description automatically generated with low confidence

This code retrieves data from the "rooms" table in the MySQL database and displays it in an HTML table format.

* It establishes a database connection using the same JDBC driver and connection URL as in the previous code and then it establishes a database connection using the same JDBC driver and connection URL as in the previous code.
* After creating a Statement object (st) from the database connection, it executes the query using the executeQuery() method of the Statement object and assigns the result to the ResultSet variable (rs).
* It generates an html table row (<tr>) with the retrieved values using embedded Java code (<%= %>) to output the values dynamically.
* It includes two hyperlinks in the table row, "Edit" and "Remove," which link to separate JSP files (Roomsupdate.jsp and Roomsdelete.jsp) passing the room ID (id) as a query parameter.



When a new room is added , it shows as above in the rooms table in the database.

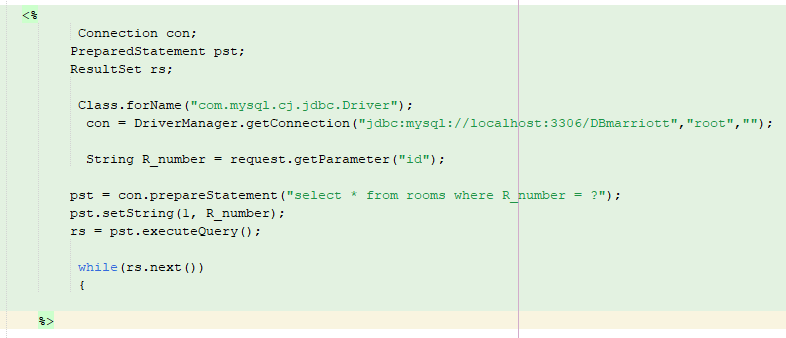
A screenshot of a computer

Description automatically generated with low confidence

This is the list of added rooms and this table is displayed on the Manage Room page. It gets updated when new room is added or details were edited or an existing room record is deleted.

**Roomsupdate.jsp**

****

* This is similar to the Clientupdate.jsp . Here this handles the update functionality for the room details. When the room record is edited and clicked the Update button , it retrieves the updated values for the room details from the request parameters , "RoomNo," "TypeID," "mobNo," and "Reserved." It sets the values for the SQL statement using the updated request parameters and it executes the SQL statement using pst.executeUpdate() to update the data in the database table.
* Similar to Clientupdate.jsp , this code is to retrieve the pre-inserted room details in order to update/ edit the details. It appears to retrieve room details from the "rooms" table in the MySQL database based on the "id" parameter passed in the request. In Room.jsp, the R\_number is passed as the “id” in the link to Roomupdate.jsp then in here it retrieves the value of the "id" parameter from the request using request.getParameter("id") and assigns it to the variable R\_number to retrieve the specific record.
* This generates an HTML form with input fields for each client attribute, pre-filling the fields with the retrieved values using embedded Java code (<%= %>) to output the values dynamically. It helps to display the existing details related to each record before editing, then the user can edit the existing details.

**Roomsdelete.jsp**

****

This code handles the deletion of the records of rooms.