Software Engineering Team 1: README.txt

CreateTask

Description

- This function establishes a connection to a database, inserts a new task record into a specific table, and retrieves the ID of the newly created task.

Function

- Establishes a connection to the database using credentials and URL.
- Takes in task name, dates, deadline, ID, status, description, and difficulty as inputs. These are all inserted into the database using MySQL query and prepared statements.
- Retrieves the ID of newly created tasks using generated keys. The ID is then obtained from the result set of prepared statements.

AddBudget Servlet

Installation and Configuration

1. Copy the AddBudget.java file to your project's source code directory.
2. Import the necessary Java libraries:
    a. java.io.IOException
    b. java.io.PrintWriter
    c. java.sql.*
    d. javax.servlet.ServletException
    e. javax.servlet.http.HttpServlet
    f. javax.servlet.http.HttpServletRequest
    g. javax.servlet.http.HttpServletResponse

3. Ensure that the MySQL Connector/J library is included in your project's classpath.
4. Update the DB_URL, DB_USER, and DB_PASSWORD constants in the AddBudget.java file to match your MySQL database configuration.
5. Build and deploy the project on your Apache Tomcat server.

Usage

The AddBudget servlet is accessed through an HTTP POST request. It expects two parameters:

I. '**projectName**': The name of the project for which the budget is being updated.
II. '**budgetValue**': The budget value to be added to the project's start and current budgets.

NOTE: To use the servlet, make an HTTP POST request to the servlet's URL, providing the required parameters. The servlet will update the corresponding project's budget values in the database.

Error Handling

If an error occurs during the database operation, the servlet will print the stack trace and set the HTTP response status to 500 (Internal Server Error). Otherwise, if the operation is successful, the response status will be set to 200 (OK).

Person Class

Installation and Configuration

1. Copy the Person.java file to your project's source code directory.
2. Import the necessary Java libraries:
    a. java.sql.*
3. Ensure that the MySQL Connector/J library is included in your project's classpath.
4. Update the database connection parameters (url, dbName, driver, userName, dbpassword) in the Person.java file to match your MySQL database configuration.

Usage

To insert a person's details into the database, follow these steps:

1. Create a new instance of the Person class, providing the person's details (fname, lname, username, password, email, profilePicturePath) in the constructor.
2. Call the insertPerson() static method of the Person class, passing the Person object as a parameter. Example: Person.insertPerson(personObject);
3. The insertPerson() method will establish a connection to the database and insert the person's details into the "Person" table.
4. If an error occurs during the database operation, the stack trace will be printed.

Note: Make sure to handle exceptions appropriately in your application, such as displaying error messages to the user or taking appropriate recovery actions.

UpdateBudget Servlet

Installation and Configuration

1. Copy the UpdateBudget.java file to your project's source code directory.
2. Import the necessary Java libraries:
    a. java.io.IOException
    b. java.io.PrintWriter
    c. java.sql.*
    d. javax.servlet.ServletException
    e. javax.servlet.http.HttpServlet
    f. javax.servlet.http.HttpServletRequest
    g. javax.servlet.http.HttpServletResponse
3. Ensure that the MySQL Connector/J library is included in your project's classpath.
4. Update the DB_URL, DB_USER, and DB_PASSWORD constants in the UpdateBudget.java file to match your MySQL database configuration.
5. Build and deploy the project on your Apache Tomcat server.

Usage

The UpdateBudget servlet is accessed through an HTTP POST request. It expects two parameters:

I. '**projectName**': The name of the project for which the budget is being updated.
II. '**budgetValue**': The value by which the current budget should be increased.

To use the servlet, make an HTTP POST request to the servlet's URL, providing the required parameters. The servlet will update the corresponding project's current budget in the database.

If the update operation is successful and at least one row is updated in the "Projects" table, the servlet will set the HTTP response status to 200 (OK). If no rows are updated, indicating that the project was not found in the database, the response status will be set to 404 (Not Found). If an error occurs during the database operation, the response status will be set to 500 (Internal Server Error).

Error Handling

If an error occurs during the database operation, the servlet will print the stack trace and set the HTTP response status to 500 (Internal Server Error). If the project is not found in the database, the response status will be set to 404 (Not Found).

ViewTask Class

Dependencies

- None

Usage

1. Make sure you have the appropriate JDBC driver for your database system.
2. Set the JDBC driver name, database URL, database credentials, and other required variables in the code.
3. Compile and run the ViewTasks class.

Code Explanation

1. Import the necessary packages
2. Define the ViewTasks class
3. Declare the JDBC driver and database URL
    a. Replace "driver" with the appropriate JDBC driver name and "url" with the database URL.
4. Declare the database credentials
    a. Replace "username" and "password" with the actual username and password for your database.
5. Define the main method
6. Declare necessary variables for the database connection and query execution
7. Set up the database connection and execute the query
8. Iterate over the result set and print task details
9. Close the resources (result set, statement, and connection) in the finally block
10. Handle any potential exceptions in separate catch blocks
11. Close resources in the finally block

LogoutServletClass

Dependencies

- None

Usage

1. Configure the web application deployment descriptor (e.g., web.xml or annotations) to map the servlet URL pattern to "/logout".

Code Explanation

1. Import the necessary packages
    - import java.io.IOException;
    - import javax.servlet.ServletException;
    - import javax.servlet.annotation.WebServlet;
    - import javax.servlet.http.HttpServlet;
    - import javax.servlet.http.HttpServletRequest;
    - import javax.servlet.http.HttpServletResponse;
    - import javax.servlet.http.HttpSession;
2. Define the LogoutServlet class
3. Override the doGet method to handle the GET request
4. Retrieve the current session and invalidate it
5. Redirect the user to the login page after logout
6. Replace "login.jsp" with the appropriate URL or page name of your login page

LoginServletClass

Dependencies

- '**com.project.db.DatabaseConnection**': Custom class for establishing a database connection.

Usage

1. Configure the web application deployment descriptor (e.g., web.xml or annotations) to map the servlet URL pattern to "/login".

Code Explanation

1. Import the necessary packages
    - import java.io.IOException;
    - import java.io.PrintWriter;
    - import java.sql.Connection;
    - import java.sql.PreparedStatement;
    - import java.sql.ResultSet;
    - import java.sql.SQLException;
    - import javax.servlet.RequestDispatcher;
    - import javax.servlet.ServletException;
    - import javax.servlet.annotation.WebServlet;
    - import javax.servlet.http.HttpServlet;
    - import javax.servlet.http.HttpServletRequest;
    - import javax.servlet.http.HttpServletResponse;
    - import javax.servlet.http.HttpSession;
    - import com.project.db.DatabaseConnection;
2. Define the LoginServlet class
3. Override the doPost method to handle the POST request
4. Retrieve the username and password from the request parameters
5. Initialize the dispatcher and session objects
6. Establish a database connection using the DatabaseConnection class:
    a. Make sure to have the DatabaseConnection class properly implemented with the necessary database configuration
7. Prepare and execute a SQL query to check if the username and password match a record in the database
8. Check the result of the query and handle the appropriate scenarios:
    a. If a matching record is found, set the username in the session and forward the request to the "index.jsp" page
    b. Replace "index.jsp" and "login.jsp" with the appropriate URLs or page names of your home and login pages, respectively