Railway Crossing Status

Writeup

Railway crossing application.

1. `Add_railway.html`: This is an HTML file containing a form to add a railway crossing. It includes input fields for name, address, landmark, train schedules, person in charge, status, and favorite.

2. `government_login.jsp`: This JSP file displays a form for government user login. It includes input fields for email and password.

3. `Government_users.jsp`: This JSP file displays a form for government user registration. It includes input fields for name, email, and password. It also provides a link to the government login page.

4. `Home.jsp`: This JSP file represents the home page of the railway crossing application. It provides options for the government section and the user section.

5. `Login.jsp`: This JSP file displays a form for user login. It includes input fields for email and password. It also provides a link to the registration page.

6. `Register.jsp`: This JSP file displays a form for user registration. It includes input fields for name, email, and password. It also provides a link to the login page.

7. `Update_railway.jsp`: This JSP file displays a form to update a railway crossing. It retrieves the crossing details from the request parameters and pre-fills the form with the existing values.

8. `Sql queries`: These are SQL queries used to create the necessary database tables and insert sample data. It includes the creation of the "users," "railway_crossings," "user_favourite," and "admin" tables.

The code you provided includes several servlet classes for managing railway crossings. Here's a breakdown of each servlet:

1. DashboardServlet: This servlet handles the main dashboard page for displaying railway crossings. It connects to a MySQL database and retrieves crossing data from the "railway_crossings" table. The servlet generates an HTML table to display the crossing information and allows users to search for crossings and mark them as favorites.

2. DeleteRailwayCrossingServlet: This servlet handles the deletion of a railway crossing. It receives the crossing ID as a parameter, connects to the database, and executes a SQL DELETE statement to remove the crossing with the specified ID from the "railway_crossings" table.

3. FavoriteCrossingsServlet: This servlet displays a list of favorite railway crossings. It retrieves the list of favorite crossings from the user's session and uses the IDs to fetch the corresponding crossing details from the database. The servlet generates an HTML table to display the favorite crossings.

4. GetRailwayDetailsServlet: This servlet retrieves all railway crossings from the database and displays them in an HTML table. It connects to the database, executes a SELECT statement to fetch all rows from the "railway_crossings" table, and generates an HTML table to display the crossing details.

These servlets work together to provide functionality for managing and displaying railway crossings. The DashboardServlet serves as the main entry point, while the other servlets handle specific actions such as deletion and displaying favorites.

5. LoginServlet:
- This servlet handles the login functionality.
- It receives the user's email and password as parameters through a POST request.
- It establishes a connection to the MySQL database using JDBC.
- It executes a SQL SELECT statement to validate the user's credentials.
- If the login is successful, it creates a session for the user and redirects to the "DashboardServlet".
- If the login fails, it displays an error message.

6. RailwayCrossingServlet:
- This servlet handles the addition of a railway crossing.
- It receives various parameters related to a railway crossing through a POST request.
- It establishes a connection to the MySQL database using JDBC.
- It executes a SQL INSERT statement to add the railway crossing to the database.
- If the insertion is successful, it redirects to the "GetRailwayDetailsServlet".
- If the insertion fails, it displays an error message.

7. RegisterServlet:
- This servlet handles the registration functionality.
- It receives the user's name, email, and password as parameters through a POST request.
- It establishes a connection to the MySQL database using JDBC.
- It executes a SQL INSERT statement to add the user to the database.
- If the registration is successful, it redirects to the "login.jsp" page.
- If the registration fails, it displays an error message.

8. UpdateRailwayCrossingServlet:
- This servlet handles the update of a railway crossing.
- It receives various parameters related to a railway crossing through a POST request, including the crossing's ID.
- It establishes a connection to the MySQL database using JDBC.
- It executes a SQL UPDATE statement to update the railway crossing in the database.
- If the update is successful, it redirects to the "GetRailwayDetailsServlet".
- If the update fails, it displays an error message.

.