Rapport Java TP1

Mouad ALoubi

December 12, 2024

Table of Contents

1 Objective

The purpose of this practice session was to conceptualize, design, and implement a fully functional Java-based employee management system. This system incorporates a multi-layered architecture that integrates models, views, controllers, and database interactions. The session's main goal was to develop practical experience in applying these software engineering concepts.

2 Assignment Overview

The task was to create an employee management application with the following key components:

- **Model:** This represents the employee data structure, encompassing attributes such as name, email, phone number, salary, job position, and role.
- **DAO (Data Access Object):** This layer facilitates interactions with a PostgreSQL database through SQL queries.
- **Controller:** This manages the logic for employee operations, including addition, deletion, updating, and data retrieval.
- **View:** A user-friendly graphical interface designed to enable efficient user interactions.

3 Code Implementation

This section provides detailed examples of the implementation of the system's components.

3.1 Main Class

```
import Controllers.EmployeeController;
import DAO.EmployeeDAOImpl;
import Views.EmployeeView;

public class Main {
  public static void main(String[] args) {
    // Initialize the database connection
    EmployeeDAOImpl dao = new EmployeeDAOImpl();

    // Render the View
    EmployeeView ev = new EmployeeView();

    // Add controller for the view
    EmployeeController ec = new EmployeeController();
  }
}
```

3.2 DAO

3.2.1 DAO Interface

```java
package DAO;

interface EmployeeDAOI {
  // Database credentials
  public String url = "jdbc:postgresql://localhost:5432/java_db";
  public String dbuser = "postgres";    // Database username
  public String dbpw = "pg1234";        // Database password

  // Abstract methods
  public boolean addEmployee(Employee em);
  public boolean deleteEmployee(int id);
  public boolean updateEmployee(int id, Employee em);
}
```

3.2.2 DAO Implementation

```java
package DAO;

public class EmployeeDAOImpl implements EmployeeDAOI {
  // Constructor
  public EmployeeDAOImpl();

  // Methods to be overridden here
}
```

3.3 Model

```
package Models;

public class Employee {
  // Constructor
  public Employee(ResultSet rs);

  // Getters
  public int getId();
  public String getLname();
  public String getFname();
  public String getEmail();
  public double getSalary();
  public String getPhone();
  public String getPost();
  public String getRole();

  // Setters
  public void setId(int id);
  public void setLname(String lname);
  public void setFname(String fname);
  public void setEmail(String email);
  public void setSalary(double salary);
  public void setPhone(String phone);
  public void setPost(String post);
  public void setRole(String role);

  // Methods for controller interactions
  public boolean addEmployee();
  public static boolean deleteEmployee(int id);
  public boolean updateEmployee(int id);

  public String toString();
}
```

3.4 Controller

```
package Controllers;

public class EmployeeController {
  // Constructor
  public EmployeeController();

  // Event listener initialization methods
  private void initAddEvent();
  private void initDeleteEvent();
  private void initUpdateEvent();
  private void initShowEvent();

  // Useful View handling methods
  public static void populateTable();
  public static void emptyFields();
}
```

3.5 View

```
package Views;

public class EmployeeView extends JFrame {
  // Constructor
  public EmployeeView();
}
```

4 Results

The application was developed and tested on an Ubuntu system with PostgreSQL as the database management system. It successfully demonstrated the following functionalities:

- Establishing a reliable connection to a PostgreSQL database.
- Performing operations such as adding, deleting, and updating employee records through the GUI.
- Displaying employee data in a user-friendly, tabular format within the interface.

5 Challenges and Solutions

- **Challenge:** Managing SQL exceptions during database operations.
- **Solution:** Integrated comprehensive try-catch blocks and meaningful error messages.
- **Challenge:** Ensuring synchronization between GUI updates and database changes.
- **Solution:** Designed methods to dynamically repopulate the GUI table after each operation.

6 Conclusion

This practice session provided valuable hands-on experience in developing a complete Java application. It emphasized the use of the MVC architecture, effective database interaction, and GUI development, fostering a deeper understanding of software engineering principles.

7 References

- Official Java Documentation: https://docs.oracle.com/en/java/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- pgJDBC Documentation: https://jdbc.postgresql.org/documentation/