Java Developer Internship
Task 7: Java JDBC - Employee Database App

Soumen Das

November 24, 2025

1 Objective

The objective of this task is to create a Java application that connects to a Relational Database Management System (MySQL/PostgreSQL) using JDBC (Java Database Connectivity). The application performs standard CRUD (Create, Read, Update, Delete) operations on an Employee database.

2 System Requirements & Tools

- **Language:** Java
- **Database:** MySQL
- **Driver:** MySQL Connector/J
- **IDE:** VS Code / IntelliJ IDEA

3 Database Setup

Before running the Java application, the database schema was initialized using the following SQL commands:

```sql
CREATE DATABASE company_db;

USE company_db;

CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    department VARCHAR(100),
    salary DOUBLE
);
```

Listing 1: Database Initialization

4 Implementation Details

The application is a console-based tool implemented in a single class `EmployeeDBApp`. It utilizes the `java.sql` package to handle database interactions.

4.1 Key Features

- **Connection Management:** Uses `DriverManager.getConnection()` to establish a secure link to the local MySQL instance.
- **Prepared Statements:** All SQL queries use `PreparedStatement` to prevent SQL injection and ensure efficient execution.
- **Menu-Driven Interface:** A loop-based menu allows the user to perform multiple operations in a single session.

5 Source Code

Below is the complete source code for the `EmployeeDBApp.java` file.

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

public class EmployeeDBApp {

    // Database Credentials
    private static final String URL = "jdbc:mysql://localhost:3306/company_db";
    private static final String USER = "root";
    private static final String PASSWORD = "password";

    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Connection to Database Successful!");

            Scanner scanner = new Scanner(System.in);
            boolean running = true;

            while (running) {
                System.out.println("\n--- EMPLOYEE MANAGEMENT SYSTEM ---");
                System.out.println("1. Add Employee");
                System.out.println("2. View All Employees");
                System.out.println("3. Update Employee");
                System.out.println("4. Delete Employee");
                System.out.println("5. Exit");
                System.out.print("Enter choice: ");

                int choice = scanner.nextInt();
                scanner.nextLine();

                switch (choice) {
                    case 1: addEmployee(conn, scanner); break;
                    case 2: viewEmployees(conn); break;
                    case 3: updateEmployee(conn, scanner); break;
                    case 4: deleteEmployee(conn, scanner); break;
                    case 5: running = false; break;
                    default: System.out.println("Invalid choice.");
                }
            }
            conn.close();
            scanner.close();

        } catch (SQLException e) {
            System.err.println("Connection Error: " + e.getMessage());
        }
    }

    // Methods for CRUD operations omitted for brevity
    // (See full attached code file for implementation details of
    // addEmployee, viewEmployees, updateEmployee, and deleteEmployee)
}
```

Listing 2: EmployeeDBApp.java

6 Execution Output

When the application is executed, the following interaction occurs in the console:

```
> Connection to Database Successful!

--- EMPLOYEE MANAGEMENT SYSTEM ---
1. Add Employee
2. View All Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter choice: 1
Enter Name: Soumen Das
Enter Department: Development
Enter Salary: 50000
> Employee added successfully!

--- EMPLOYEE MANAGEMENT SYSTEM ---
...
```

7 Conclusion

This task successfully demonstrated how to integrate Java with a backend database. The use of JDBC provided a robust way to manage persistent data, fulfilling the requirements of the internship task.