Java JDBC

JDBC Connection

```
import java.sql.Connection; import
java.sql.DriverManager; import
java.sql.SQLException;
/**
 *
 * @author User
 */
public class DatabaseConnection {
   private static final String URL = "jdbc:mysql://localhost:3306/employee_db"; // Database URL private
static final String USER = "root"; // Your MySQL username
private static final String PASSWORD = "sasindu@2002" ;// Your MySQL password

public static Connection getConnection() throws SQLException { try {
// Load the JDBC driver
Class.forName("com.mysql.cj.jdbc.Driver");
// Return the database connection
   return DriverManager.getConnection(URL, USER, PASSWORD);
} catch (ClassNotFoundException | SQLException e) {
System.out.println("Connection failed: " + e.getMessage());        throw
new SQLException("Failed to establish connection.");
   }
}
}
```

CRUD Operations

```
import java.sql.*; import
java.util.ArrayList; import
java.util.List;
/**
 *
 * @author User
 */
public class EmployeeDAO {     // Add an employee to the database     public static
boolean addEmployee(String name, String position, double salary) {        String sql
= "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";        try
(Connection conn = DatabaseConnection.getConnection();
          PreparedStatement stmt = conn.prepareStatement(sql)) {

         stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);

         int rowsAffected = stmt.executeUpdate();           return
rowsAffected > 0; // Return true if insertion was successful

      } catch (SQLException e) {
         System.err.println("Error adding employee: " + e.getMessage());
         return false;

      }

   }
```

```
 // Read all employees     public static
List<Employee> getAllEmployees() {
       List<Employee> employees = new ArrayList<>();
       String sql = "SELECT * FROM employees";

       try (Connection conn = DatabaseConnection.getConnection();
           PreparedStatement stmt = conn.prepareStatement(sql);
           ResultSet rs = stmt.executeQuery()) {

          while (rs.next()) {
             Employee employee = new Employee(
                   rs.getInt("id"),
rs.getString("name"),
rs.getString("position"),
rs.getDouble("salary")
             );
             employees.add(employee);
          }

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

   // Update an employee&#39;s information    public static boolean updateEmployee(int
id, String name, String position, double salary) {        String sql = "UPDATE employees SET
name = ?, position = ?, salary = ? WHERE id = ?";        try (Connection conn =
DatabaseConnection.getConnection();
```

```
            PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setString(1, name);
stmt.setString(2, position);
stmt.setDouble(3, salary);            stmt.setInt(4,
id);

            int rowsAffected = stmt.executeUpdate();

            return rowsAffected > 0; // Returns true if at least one row was updated

        } catch (SQLException e) {
            System.err.println("Error updating employee with ID " + id + ": " + e.getMessage());
            return false;
        }
    }

   // Delete an employee by ID     public static
boolean deleteEmployee(int id) {
        String sql = "DELETE FROM employees WHERE id = ?";

        try (Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement(sql)) {

            stmt.setInt(1, id);
            int rowsAffected = stmt.executeUpdate();

            return rowsAffected > 0; // Returns true if at least one row was deleted
```

```
        } catch (SQLException e) {
            System.err.println("Error deleting employee with ID " + id + ": " + e.getMessage());
            return false;
        }
    }

}
```

Employee.java POJO (Plain Old Java Object) to represent employee data.

```
public class Employee {

 private int id;     private
String name;     private
String position;     private
double salary;

    // Constructor     public Employee(int id, String name, String
position, double salary) {
        this.id = id;
        this.name = name;
this.position = position;
this.salary = salary;
    }

    // Getters and Setters
    public int getId() { return id; }
public void setId(int id) { this.id = id; }
```

```
    public String getName() { return name; }     public void
setName(String name) { this.name = name; }

    public String getPosition() { return position; }     public void
setPosition(String position) { this.position = position; }

    public double getSalary() { return salary; }     public void
setSalary(double salary) { this.salary = salary; }

    // Override toString method for better readability
    @Override     public
String toString() {
        return "Employee{id=" + id + ", name='" + name + "', position='" + position + "', salary=" + salary + "}";
    }
}
```

Main method

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // Add employees
        EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
```

```
        // Update employee (Assuming an employee with ID 1 exists)
        boolean updateSuccess = EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer", 90000);
        System.out.println(updateSuccess ? "Employee updated successfully." : "Employee update failed.");

        // Get all employees
        System.out.println("\n--- Employee List ---");
        List<Employee> employees = EmployeeDAO.getAllEmployees();
employees.forEach(System.out::println);

        // Delete employee (Assuming an employee with ID 2 exists)
boolean deleteSuccess = EmployeeDAO.deleteEmployee(2);
        System.out.println(deleteSuccess ? "Employee deleted successfully." : "Employee deletion failed.");
    }

}
```