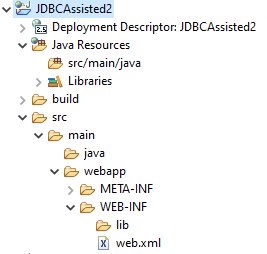
**1.** Creating a dynamic web project

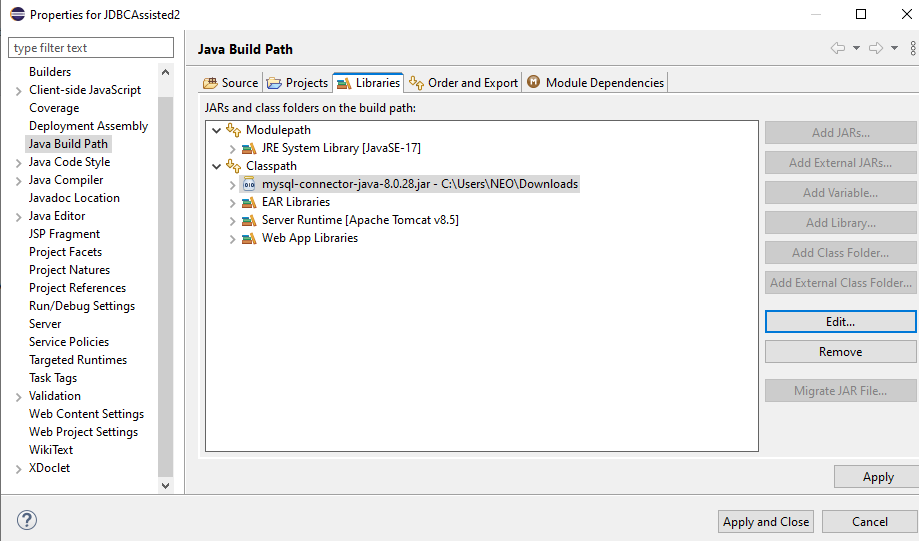
* Open Eclipse
* Go the **File** menu. Choose **New->Dynamic Web Project**
* Enter the project name as JDBC**Assisted2**. Click on **Next**
* Enter nothing in the next screen and click on **Next**
* Check the checkbox **Generate web.xml deployment descriptor** and click on **Finish**
* This will create the project files in the Project Explorer

****

**2.** Adding the jar files for MySQL connection for Java

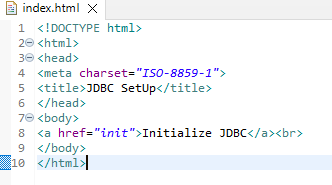
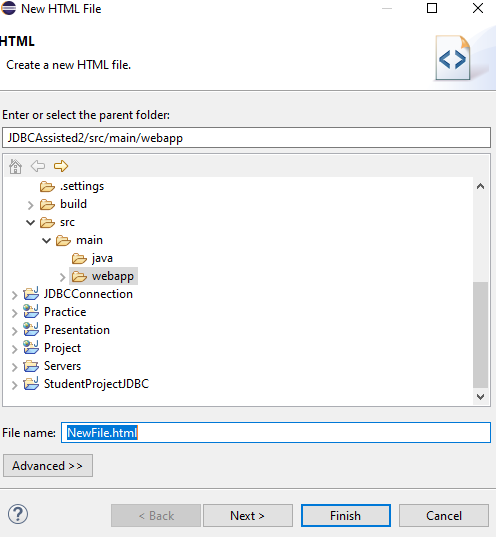
Right Click on Created Project🡪BuildPath🡪Configure BuildPath🡪

Add External Jars🡪 Apply and Close



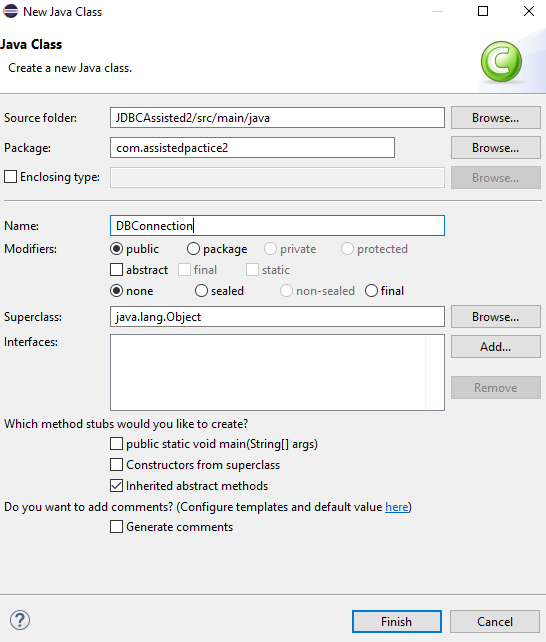
**3.**Creating an HTML page index.html

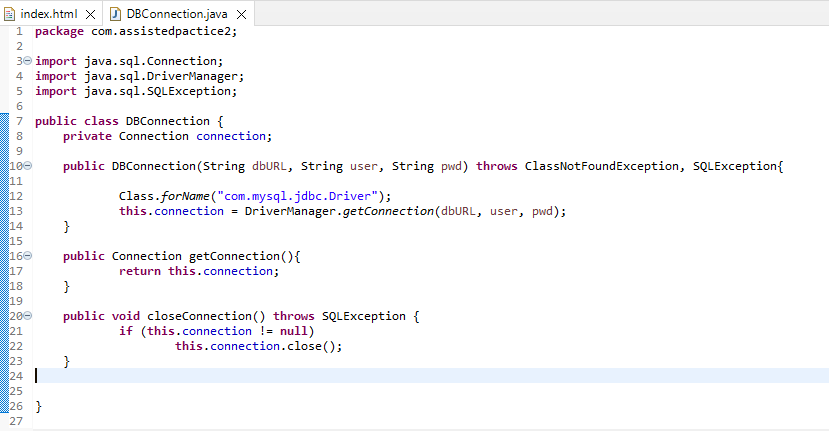
* In the Project Explorer, expand the project **JDBCSetup**
* Expand **WebContent**. Right click on **WebContent**. Choose **New->HTML File**
* Enter the filename as index.html and click on **Finish**
* Enter the following code:



**4.** Creating a DBConnection class to initiate a JDBC connection in code

* In the Project Explorer, expand **JDBCAssisted2->Java Resources**
* Right click on **src** and choose **New->Class**
* In **Package**, enter **com.assistedpactice2** and in **Name** enter **DBConnection** and click on **Finish**
* Enter the following code:





**5. Create a Java class for Database Connection:**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydb";

private static final String USER = "root";

private static final String PASSWORD = "Tagore@12345";

public static Connection getConnection() {

try {

return DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

} catch (SQLException e) {

e.printStackTrace();

throw new RuntimeException("Failed to connect to the database");

}

}

}

**6:** **Create a Java class for CRUD operations:**

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class EmployeeDAO {

public void createEmployee(String firstName, String lastName, double salary) {

String sql = "INSERT INTO employees (first\_name, last\_name, salary) VALUES (?, ?, ?)";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setString(1, firstName);

statement.setString(2, lastName);

statement.setDouble(3, salary);

statement.executeUpdate();

System.out.println("Employee created successfully");

} catch (SQLException e) {

e.printStackTrace();

}

}

public void readEmployees() {

String sql = "SELECT \* FROM employees";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(sql);

ResultSet resultSet = statement.executeQuery()) {

while (resultSet.next()) {

System.out.println(

"ID: " + resultSet.getInt("id") +

", Name: " + resultSet.getString("first\_name") + " " + resultSet.getString("last\_name") +

", Salary: " + resultSet.getDouble("salary")

);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public void updateEmployeeSalary(int employeeId, double newSalary) {

String sql = "UPDATE employees SET salary = ? WHERE id = ?";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setDouble(1, newSalary);

statement.setInt(2, employeeId);

int rowsUpdated = statement.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("Salary updated successfully");

} else {

System.out.println("Employee not found");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public void deleteEmployee(int employeeId) {

String sql = "DELETE FROM employees WHERE id = ?";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(sql)) {

statement.setInt(1, employeeId);

int rowsDeleted = statement.executeUpdate();

if (rowsDeleted > 0) {

System.out.println("Employee deleted successfully");

} else {

System.out.println("Employee not found");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**7:**. **Create a Main Class to Test:**

public class MainApp {

public static void main(String[] args) {

EmployeeDAO employeeDAO = new EmployeeDAO();

// Create employee

employeeDAO.createEmployee("John", "Doe", 50000);

// Read employees

System.out.println("List of Employees:");

employeeDAO.readEmployees();

// Update employee salary

employeeDAO.updateEmployeeSalary(1, 55000);

// Read employees after update

System.out.println("List of Employees after Update:");

employeeDAO.readEmployees();

// Delete employee

employeeDAO.deleteEmployee(1);

// Read employees after delete

System.out.println("List of Employees after Delete:");

employeeDAO.readEmployees();

}

}

9.Run Project🡪Run as Server

This simple project demonstrates the basic setup of a JDBC environment for database interaction in a Java application.