SRI VASAVI ENGINEERING COLLEGE

(Autonomous)

# PEDATADEPALLI, TADEPALLIGUDEM-534101.





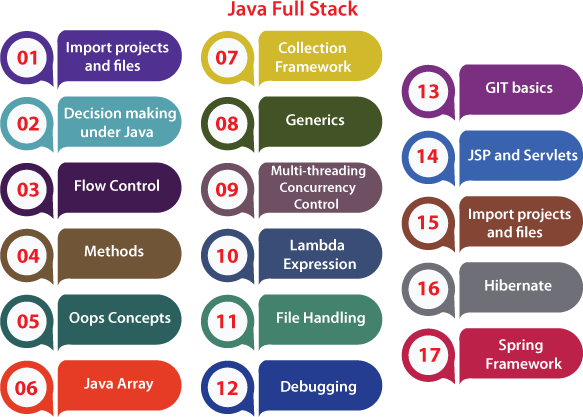
*This is to certify that with a Bonafide record of Practical work done in* ***JAVA****-****Full Stack Technologies Laboratory*** *by Mr./Miss* **LIKHITH KONAKALLA** *bearing Roll No* **21A81A05A1** *of**Branch of* ***VII Semester*** *during the academic year* ***2024-25****.*

**Faculty In-charge of the Laboratory Head of the Department**

**EXTERNAL**

# Basic Installation of IDEs and Development Tools

A full-stack developer is a person who can develop application's backend and frontend. Java full-stack is basically a term used for a web developer that uses Java to develop the entire technology stack is referred to as Java full stack developer.



**Hibernate:**

[Hibernate](https://www.javatpoint.com/hibernate-tutorial) is another concept of Java. A developer should be familiar with the Hibernate and its architecture and should also have implementation knowledge of it. The developer should be experienced in MySQL and SQL workbench. In hibernate, the developer should have familiar with Session factory and session, Adding Entity class, Hibernate in Action, CRUD-Retrieving Record from Database, CRUD-Updating a Record in Database, CRUD-Deleting Record from the database, [HQL(Hibernate](https://www.javatpoint.com/hql) [Query Language)](https://www.javatpoint.com/hql), HQL Where clause, Update record using HQL, Deleting record using HQL, integrate hibernate with JSP and Servlet, hibernate configuration, Hibernate entity class with JSP and Servlet, Hibernate in action with JSP and Servlet, display image files, improve the view of the page, adding update information form, implement update information functionality, update specific column data using hibernate, add view image action, implement view image page, and add delete image action. All the above-mentioned concepts are related to hibernate, and hibernate is the framework that is very important for Java full-stack developers.

## Spring framework:

Spring is a framework that is mostly used for Java. It provides a comprehensive programming and configuration model for modern Java-based enterprise applications. In order to work with the Spring framework, we should have knowledge of the required software, Inversion of control, Dependency injection, Autowire scenarios, Qualifier annotation, Spring Bean, Constructor injection, Spring project on IntelliJ, Adding [SpringMVC](https://www.javatpoint.com/spring-mvc-tutorial) support on Eclipse, Simple dynamic web project, Model in a web project, Spring MVC minimal setup, Basic form with Spring MVC, ModelAndView and foreach on data, Springform elements such as Input and Radio, Radio-buttons and background, Dropdown list, Text area, checkbox, etc., Adding STS 3 support into Eclipse, Add external resource, add style sheet, Form validation, Database connectivity using [JDBC](https://www.javatpoint.com/java-jdbc)(XML configuration and Annotation), Exception handling under Spring MVC, Restful API web service, database connectivity with hibernate, JAX-RS, Spring MVC + Restful web service background, building restful microservice with spring boot, and restful micro-service with database connectivity.

## 7 Best Frameworks For Full Stack Java Developers To Learn:

* Spring
* React
* Angular
* Apache Spark
* Apache Hadoop
* Hibernate
* Struts

## Full Stack Web Developer:

* A full stack web developer is a person who can develop both client and server software.
* In addition to mastering HTML and CSS, he/she also knows how to:
* Program a browser (like using JavaScript, jQuery, Angular, or Vue)
* Program a server (like using PHP, ASP, Python, or Node)
* Program a database (like using SQL, SQLite, or MongoDB)

## Client Software (Front End):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| * [HTML](https://www.w3schools.com/whatis/whatis_html.asp) * [JavaScript](https://www.w3schools.com/whatis/whatis_js.asp) | [CSS](https://www.w3schools.com/whatis/whatis_css.asp)  [ES5](https://www.w3schools.com/whatis/whatis_es5.asp) | | [Bootstrap](https://www.w3schools.com/whatis/whatis_bootstrap.asp)  [HTML DOM](https://www.w3schools.com/whatis/whatis_htmldom.asp) | | [W3.CSS](https://www.w3schools.com/whatis/whatis_w3css.asp)  [JSON](https://www.w3schools.com/whatis/whatis_json.asp) |
| * [XML](https://www.w3schools.com/whatis/whatis_xml.asp) | [jQuery](https://www.w3schools.com/jquery/default.asp) | | [Angular](https://www.w3schools.com/whatis/whatis_angularjs.asp) | | [React](https://www.w3schools.com/whatis/whatis_react.asp) |
| * Backbone.js | Ember.js | | Redux | | [Storybook](https://storybook.js.org/basics/quick-start-guide/) |
| * GraphQL | Meteor.js | | Grunt | | Gulp |
| **Server Software (Back End):** | | | | | |
| * [PHP](https://www.w3schools.com/php/default.asp) | [ASP](https://www.w3schools.com/asp/default.asp) | [C++](https://www.w3schools.com/cpp/default.asp) | | [C#](https://www.w3schools.com/cs/default.asp) | |
| * [Java](https://www.w3schools.com/java/default.asp) | [Python](https://www.w3schools.com/python/default.asp) | [Node.js](https://www.w3schools.com/nodejs/default.asp) | | Express.js | |
| * Ruby | REST | [Go](https://www.w3schools.com/go/index.php) | | [SQL](https://www.w3schools.com/whatis/whatis_sql.asp) | |
| * [MongoDB](https://www.w3schools.com/nodejs/nodejs_mongodb.asp) | [Sass](https://www.w3schools.com/sass/default.php) | Less | | Firebase.com | |

* Parse.com PaaS (Azure and Heroku)

## Hibernate Tutorial:

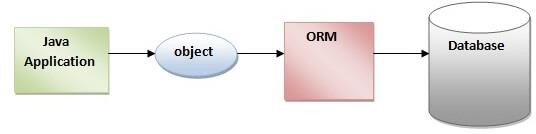
This hibernate tutorial provides in-depth concepts of Hibernate Framework with simplified examples. It was started in 2001 by Gavin King as an alternative to EJB2 style entity bean.

* Hibernate Framework

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

* ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database. The ORM tool internally uses the JDBC API to interact with the database.



## Hibernate Architecture:

* The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.
* The Hibernate architecture is categorized in four layers.
  + Java application layer
  + Hibernate framework layer
  + Backhand api layer
  + Database layer

Hibernate framework uses many objects such as session factory, session, transaction etc. along with existing Java API such as JDBC (Java Database Connectivity), JTA (Java Transaction API) and JNDI (Java Naming Directory Interface)

## First Hibernate Example without IDE:

For creating the first hibernate application, we need to follow the following steps:

1. Create the Persistent class
2. Create the mapping file for Persistent class
3. Create the Configuration file
4. Create the class that retrieves or stores the persistent object
5. Load the jar file
6. Run the first hibernate application by using command prompt

## Hibernate Example using XML in Eclipse:

Here, we are going to create a simple example of hibernate application using eclipse IDE. For creating the first hibernate application in Eclipse IDE, we need to follow the following steps:

1. Create the java project
2. Add jar files for hibernate
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Create the Configuration file
6. Create the class that retrieves or stores the persistent object
7. Run the application

## Example to create the Hibernate Application in MyEclipse:

For creating the first hibernate application in MyEclipse IDE, we need to follow following steps:

1. Create the java project
2. Add hibernate capabilities
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Add mapping of hbm file in configuration file
6. Create the class that retrieves or stores the persistent object
7. Add jar file for oracle
8. Run the application

## Spring and Spring Boot:

Spring framework is an open source Java platform that provides comprehensive infrastructure support for developing robust Java applications very easily and very rapidly. Spring framework was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003. This tutorial has been written based on Spring Framework version 4.1.6 released in Mar 2015.

Applications:

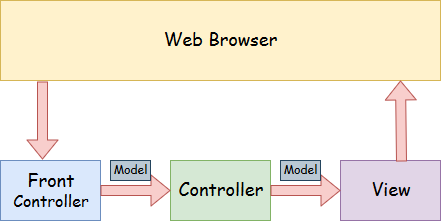
* POJO Based
* Modular
* Integration with existing frameworks
* Web MVC

**MVC**:

* Developing software and web applications requires technical as well as experiential knowledge. The Spring framework is a top choice among developers, students, and researchers to program java-based apps. So, let us look at some instances of Java MVC projects and get familiar with some nuts and bolts of the development process.
* The Model-View-Controller (MVC) architecture is a type of design that separates your application into three logical components with each one handling a specific aspect. The framework enjoys industry-wide acceptance in the development of scalable projects. By studying a Spring MVC example, you would learn how to implement the MVC pattern and create an application with a stellar user interface. So, let’s begin with getting acquainted with the design pattern first.

## Spring MVC:

* A Spring MVC is a Java framework which is used to build web applications. It follows the Model-View-Controller design pattern. It implements all the basic features of a core spring framework like Inversion of Control, Dependency Injection.
* A Spring MVC provides an elegant solution to use MVC in spring framework by the help of DispatcherServlet. Here, DispatcherServlet is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.



## Steps in Eclipse:

* 1. Provide project information and configuration in the pom.xml file.
     + pom.xml
  2. Create the controller class
     + To create the controller class, we are using two annotations @Controller and @RequestMapping.
     + The @Controller annotation marks this class as Controller.
     + The @Requestmapping annotation is used to map the class with the specified URL name.
  3. Provide the entry of controller in the web.xml file
  4. Define the bean in the xml file
  5. Display the message in the JSP page

**Displaying the current date and time and also implementing auto refreshing of the time**

**1.Create new servlet file named Refreshingdate**

**2.**  **Auto-Refreshing Page: The servlet refreshes the displayed current time every second using response.setIntHeader("Refresh", 1), enabling real-time updates.**

**3. Date and Time Display: The SimpleDateFormat class formats and displays the current date and time.**

**DateTime.java:**

package tasks;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class DateTime extends HttpServlet {

private static final long serialVersionUID = 1L;

public DateTime() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

LocalDateTime dt = LocalDateTime.now();

PrintWriter out = response.getWriter();

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss");

String formattedDate = dt.format(formatter);

out.print("<h1>Current Date and Time:</h1>");

System.out.println(formattedDate);

out.println("<h2>"+formattedDate+"</h2>");

}

}



**Displaying the HTTP Request Headers**

**1.Create a new servlet program named httpheaderdetails.**

**2. The method request.getHeaderNames() returns a list of all header names in the HTTP request. You can use it to go through each header name one by one.**

**3. The method request.getHeader(String name) gets the value of a specific header when you provide its name. In the servlet, this method is used inside a loop to get the value of each header name from getHeaderNames() and save them in a Map.**

**HeaderInfo.java:**

package tasks;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Enumeration;

public class HeaderInfo extends HttpServlet {

private static final long ***serialVersionUID*** = 1L;

public HeaderInfo() {

super();

// **TODO** Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// **TODO** Auto-generated method stub

PrintWriter out = response.getWriter();

Enumeration<String> headerNames = request.getHeaderNames();

out.println("<h1>Header Details</h1>");

out.println("<table border = '1'>");

while (headerNames.hasMoreElements()) {

String headerName = headerNames.nextElement();

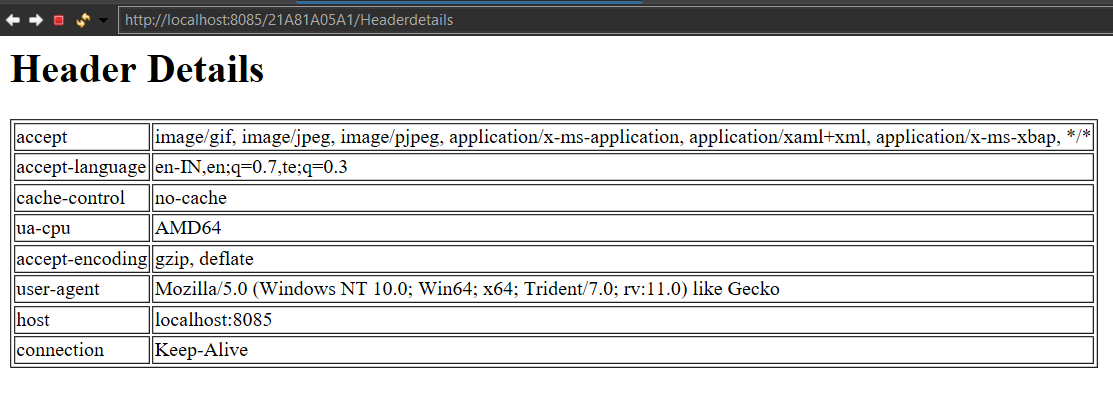
String headerValue = request.getHeader(headerName);

out.println("<tr><td>" + headerName + "</td><td>" + headerValue + "</td></tr>");

}

out.println("</table>");

}

} 

**Displaying the server details**

**1.Create a new servlet named ServerDetails.**

**2.The servlet uses request.getServerName(), request.getServerPort(), and request.getScheme() to get and display the server's name, port number, and protocol.**

**ServerInfo.java:**

package tasks;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

public class ServerInfo extends HttpServlet {

private static final long serialVersionUID = 1L;

public ServerInfo() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

out.println("Server Details");

out.println("<table border = '1'>"

+ "<tr><th>"+"Name"+"</th>"+

"<th> Port</th>"+

"<th>Address</th>"+

"<th>Remote Host</th>"+

"<th>Remote Port</th>"+

"<th>Protocol</th>"+

"</tr>");

out.println("<tr><td>"+request.getServerName()+"</td>"+

"<td>"+request.getServerPort()+"</td>"+

"<td>"+request.getRemoteAddr()+"</td>"+

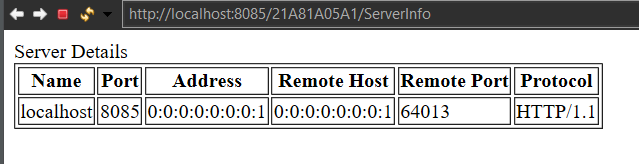
"<td>"+request.getRemoteHost()+"</td>"+

"<td>"+request.getRemotePort()+"</td>"+

"<td>"+request.getProtocol()+"</td>"+

"</tr>");

out.println("</table>");}}



**Calling parameters from web.xml**

**1.Create a New Servlet Program**

* **Create a servlet named callWebxmlParam and write code to call parameters from web.xml using getServletContext().getInitParameter().**

**2.Modify web.xml**

* **Open web.xml and add the following context parameters**

**<context-param>**

**<param-name>name</param-name>**

**<param-value>Likhith Konakalla</param-value>**

**</context-param>**

**<context-param>**

**<param-name>id</param-name>**

**<param-value>21A81A05A1</param-value>**

**</context-param>**

**Params.java:**

package com.tasks;

import jakarta.servlet.ServletConfig;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

public class Params extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* Servlet implementation class SessionAccess

\*/

public Params() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

ServletConfig cfg = getServletConfig();

String a = cfg.getInitParameter("name");

String b = cfg.getInitParameter("rollNo");

String c = cfg.getInitParameter("brn");

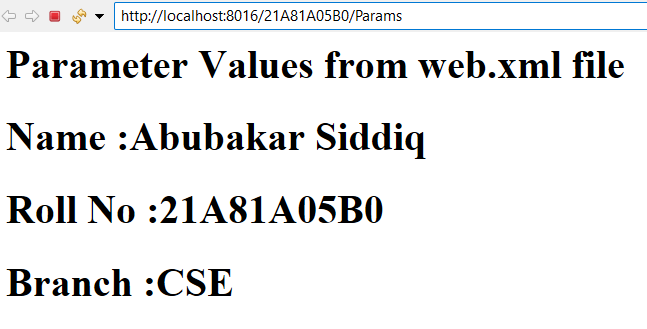
out.println("<h1>Parameter Values from web.xml file");

out.println("<h1>Name :"+a+"</h1>");

out.println("<h1>Roll No :"+b+"</h1>");

out.println("<h1>Branch :"+c+"</h1>");

}}



**Creating Sessions**

**1. Create a New Servlet File**

* **Create a servlet named CreateSessionServlet and use session.setAttribute() to set attributes and session.getAttribute() to retrieve them.**

**2. Run the Application**

* **Execute the application to test the session management functionality.**

**SessionAccess.java:**

package com.tasks;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.servlet.http.HttpSession;

import java.io.IOException;

import java.io.PrintWriter;

/\*\*

\* Servlet implementation class SessionAccess

\*/

public class SessionAccess extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public SessionAccess() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

HttpSession session = request.getSession();

session.setAttribute("Username", "Sasi kiran");

session.setAttribute("Id", "674");

response.setContentType("text/html");

response.getWriter().println("<html><body>");

response.getWriter().println("<h2>Session created successfully</h2>");

response.getWriter().println("<p>Username: " + session.getAttribute("Username") + "</p>");

response.getWriter().println("<p>ID: " + session.getAttribute("Id") + "</p>");

response.getWriter().println("</body></html>");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}



**Checking the session Id for newly created details**

**1. Create a new servlet called CheckSessionServlet.**

**2. Use request.getSession() to get the current session or create a new one if none exists.**

**3. Check if the session is new with session.isNew().**

**4. If new, set sample attributes (username, id) with session.setAttribute() and display them.**

**5. If not new, retrieve and display username, id, session ID (session.getId()), and the last accessed time formatted with SimpleDateFormat**

**ServletExist.java:**

package com.tasks;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.servlet.http.HttpSession;

import java.io.IOException;

import java.io.PrintWriter;

import java.text.SimpleDateFormat;

import java.util.Date;

/\*\*

\* Servlet implementation class ServletExist

\*/

public class ServletExist extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public ServletExist() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

HttpSession session = request.getSession();

boolean isNew = session.isNew();

response.setContentType("text/html");

response.getWriter().println("<html><body>");

if (isNew) {

response.getWriter().println("<h2>New Session Created</h2>");

session.setAttribute("sername", "Hari");

session.setAttribute("id", "677");

response.getWriter().println("<p>Username: " + session.getAttribute("username") + "</p>");

response.getWriter().println("<p>ID: " + session.getAttribute("id") + "</p>");

} else {

response.getWriter().println("<h2>Existing Session Found</h2>");

response.getWriter().println("<p>Username: " + session.getAttribute("Username") + "</p>");

response.getWriter().println("<p>ID: " + session.getAttribute("Id") + "</p>");

}

String sessionId = session.getId();

response.getWriter().println("<p>Session ID: " + sessionId + "</p>");

long lastAccessedTime = session.getLastAccessedTime();

Date lastAccessedDate = new Date(lastAccessedTime);

SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");

String formattedDate = formatter.format(lastAccessedDate);

response.getWriter().println("<p>Last Accessed Time: " + formattedDate + "</p>");

response.getWriter().println("</body></html>");

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

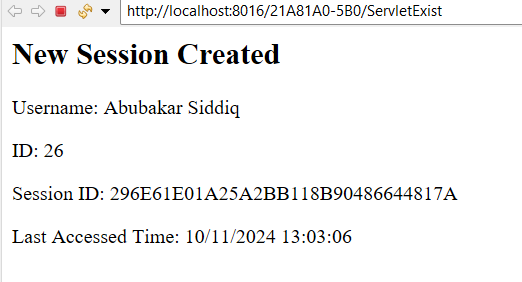
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}



**MySql connection java application**

**Steps to Set Up MySQL Connection in a Java Application**

1. **Download and Install MySQL**
   * **Download MySQL from the official website and complete the installation.**
2. **Create a New Java Application**
   * **Create a new Java file named MySQLConnectionCheck.java.**
3. **Add MySQL Connector JAR to Classpath**
   * **Download the MySQL Connector JAR file if you haven't already.**
   * **In your IDE:**
     + **Right-click on your project.**
     + **Select Build Path → Configure Build Path → Libraries.**
     + **Add the MySQL Connector JAR to the classpath.**
4. **Create a Database in MySQL**
   * **Open MySQL and create the database you want to connect to, using commands like CREATE DATABASE my\_database;.**
5. **Write Code to Check the Connection**
   * **In MySQLConnectionCheck.java, write code to establish a connection to the MySQL database and test if it connects successfully.**

**MySQLConnection.java:**

package database;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class MySQLConnection extends HttpServlet {

private static final long serialVersionUID = 1L;

public DataBaseCon() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

Connection conn = null;

Statement pstmt = null;

ResultSet rs = null;

PrintWriter out = response.getWriter();

try {

Class.forName("com.mysql.cj.jdbc.Driver");

String jdbcUrl = "jdbc:mysql://localhost:3306/CSE\_B";

String username = "root";

String password = "root";

conn = DriverManager.getConnection(jdbcUrl, username, password);

if (conn != null) {

System.out.println("Connected to the database successfully!");

}

conn.close();

} catch (ClassNotFoundException | SQLException e) {

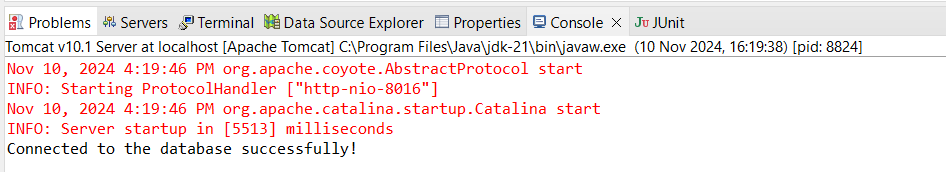
// TODO Auto-generated catch block

e.printStackTrace();

}

}

}



**Retriving data from the table using mysql connection**

**MySQLCon.java:**

package database;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class DataBaseCon extends HttpServlet {

private static final long serialVersionUID = 1L;

public DataBaseCon() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

Connection conn = null;

Statement pstmt = null;

ResultSet rs = null;

PrintWriter out = response.getWriter();

try {

Class.forName("com.mysql.cj.jdbc.Driver");

String jdbcUrl = "jdbc:mysql://localhost:3306/CSE\_B";

String username = "root";

String password = "root";

conn = DriverManager.getConnection(jdbcUrl, username, password);

if (conn != null) {

System.out.println("Connected to the database successfully!");

}

Statement statement = conn.createStatement();

String query = "SELECT \* FROM student";

ResultSet resultSet = statement.executeQuery(query);

while (resultSet.next()) {

int id = resultSet.getInt("rno");

String name = resultSet.getString("name");

String brn = resultSet.getString("brn");

System.out.println("ID: " + id + ", Username: " + name + ", Password: " + brn);

}

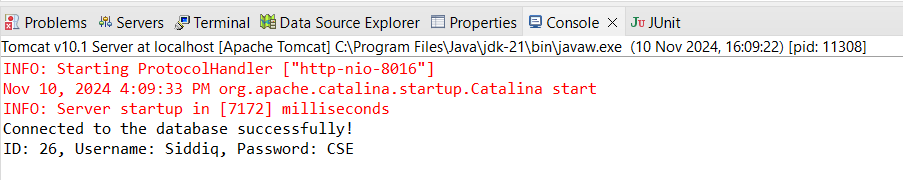
conn.close();

} catch (ClassNotFoundException | SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}}}



**Displaying the greetings for based on the timings**

**1. Create a new servlet called Greetings.java and an HTML file named show.html with the provided code.**

**2. LocalDateTime.now(): This method gets the current date and time.**

**3. DateTimeFormatter.ofPattern(): This method formats the current date and time, or an entered time, into a specific pattern you define.**

**4. LocalTime.parse(): This method converts a time entered as a string into a LocalTime object.**

**5. getHour(): This method retrieves the hour part of the time to check if it’s morning, afternoon, or night.**

**Greetings.java:**

package tasks;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class Greetings extends HttpServlet {

private static final long serialVersionUID = 1L;

public Greetings() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

PrintWriter out = response.getWriter();

LocalDateTime now = LocalDateTime.now();

DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");

String formattedDate = now.format(dateTimeFormatter);

out.println("<html><body>");

int hours = now.getHour();

if(hours < 12) {

out.print("<h1>"+"Good Morning"+"</h1>");

}

else if(hours < 18) {

out.print("<h1>"+"Good Afternoon"+"</h1>");

}

else {

out.print("<h1>"+"Good Evening"+"</h1>");

}

 out.println("<h1>Currrent Time and Date: "+formattedDate+"</h1>");}}

**Creating Employee MVC by using java application**

**🡪Create a Java project consisting of the following files:**

1. **Employee.java (Model Class): This file will serve as the model class for the Employee entity, containing fields such as id, name, and department, along with their getters.**
2. **EmployeeController.java (Controller Class): This file will manage the interaction between the Employee model and the view, ensuring that the data is properly passed and processed for display.**
3. **EmployeeView.java (View Class): This file will handle the presentation logic and display the details of the Employee object to the user.**
4. **MVCDemo.java (Main Class): This file will contain the main method for the program's execution. It will:**
   * **Ask the user to input details to create an Employee.**
   * **Use a factory class inside Main.java to create the Employee object.**
   * **Create EmployeeController and EmployeeView objects to show the employee's details.**

**🡪Finally, run MVCDemo.java to execute the program and see the output.**

**Employee.java**

package com.example.mvc;

public class Employee {

private int eid;

private String ename;

private String edept;

// Constructor

public Employee(int eid, String ename, String edept) {

this.eid = eid;

this.ename = ename;

this.edept = edept;

}

// Getters and Setters

public int getEid() {

return eid;

}

public void setEid(int eid) {

this.eid = eid;

}

public String getEname() {

return ename;

}

public void setEname(String ename) {

this.ename = ename;

}

public String getEdept() {

return edept;

}

public void setEdept(String edept) {

this.edept = edept;

}

}

**EmployeeController.java:**

package com.example.mvc;

public class EmployeeController {

private Employee model;

private EmployeeView view;

// Parameterized Constructor

public EmployeeController(Employee model, EmployeeView view) {

this.model = model;

this.view = view;

}

// Inner Getters and Setters

public void setEmployeeName(String ename) {

model.setEname(ename);

}

public String getEmployeeName() {

return model.getEname();

}

public void setEmployeeID(int eid) {

model.setEid(eid);

}

public int getEmployeeID() {

return model.getEid();

}

public void setEmployeeDept(String edept) {

model.setEdept(edept);

}

public String getEmployeeDept() {

return model.getEdept();

}

// Method to update the view

public void updateView() {

view.printEmployeeDetails(model.getEid(), model.getEname(), model.getEdept());

}

}

**EmployeeView.java:**

package com.example.mvc;

public class EmployeeView {

public void printEmployeeDetails(int eid, String ename, String edept) {

System.out.println("Employee ID: " + eid);

System.out.println("Employee Name: " + ename);

System.out.println("Employee Department: " + edept);

}

}

**MVCDemo.java:**

package com.example.mvc;

import java.util.\*;

**public** **class** MVCDemo {

**public** **static** **void** main(String[] args) {

Employee model = **new** Employee(50, "Abubakar Siddiq", "CSE");

EmployeeView view = **new** EmployeeView();

EmployeeController controller = **new** EmployeeController(model, view);

controller.updateView();

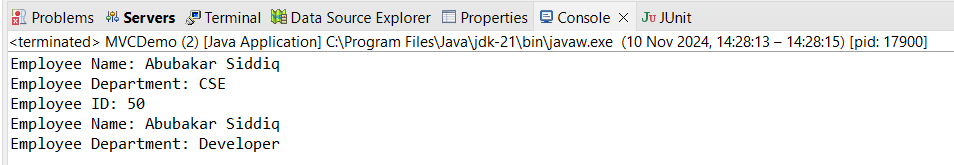
controller.setEmployeeName("Abubakar Siddiq");

controller.setEmployeeDept("Developer");

controller.updateView();

}

}



**Checking the valid password or not**

**1. LoginServlet checks if the user input matches predefined credentials and redirects accordingly.**

**2. ValidServlet displays a "Login Successful!" message if the credentials are correct.**

**3. InvalidServlet shows "Login Failed" if the credentials are incorrect.**

**Initial.java:**

package validator;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

/\*\*

\* Servlet implementation class Initial

\*/

public class Initial extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public Initial() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

String username = request.getParameter("username");

String password = request.getParameter("password");

if(username.equals("Siddiq") && password.equals("Siddiq@1234")) {

response.sendRedirect("ValidServlet");

}

else {

response.sendRedirect("InvalidServlet");

}

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

**ValidServlet.java:**

package validator;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

/\*\*

\* Servlet implementation class ValidServlet

\*/

public class ValidServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public ValidServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.setContentType("text/html");

response.getWriter().println("<html><body><h1>Login Successful!</h1></body></html>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

**InvalidServlet.java:**

package validator;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

public class InvalidServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public InvalidServlet() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.setContentType("text/html");

response.getWriter().println("<html><body><h1>Login Failed. Invalid credentials.</h1></body></html>");

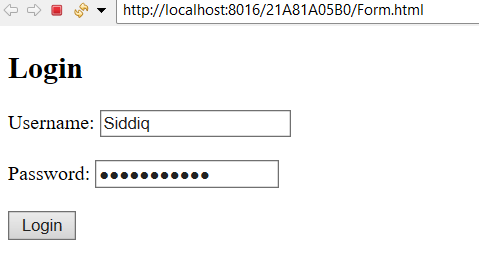
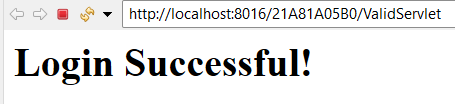
}

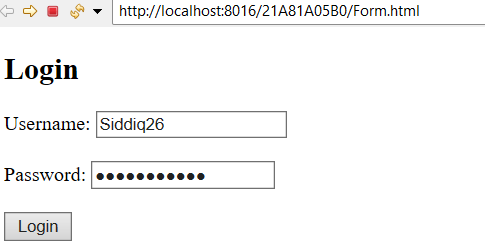
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

doGet(request, response);

}

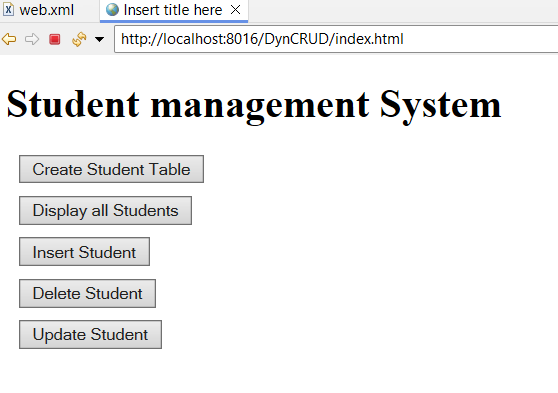
}



**Dynamic Web Project: CRUD using MySQL and Tomcat Server**

1. **Create a new dynamic web project.**
2. **Add the mysql-connector-java.jar file in the lib folder of your Tomcat server.**
3. **Create the Servlets:** 
   * **Create a student table.**
   * **Insert student details into the table.**
   * **Show student details from the table.**
   * **Update student information.**
   * **Delete student records.**
4. **Make the necessary HTML files for user interaction.**
5. **Run the project and check if everything works.**

**CreateServlet.java:**

package com.crud;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

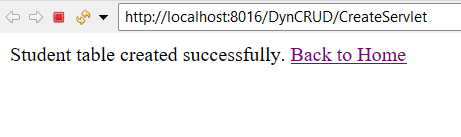
import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

public class CreateServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public CreateServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/vasavi", "root", "root");

Statement stmt = con.createStatement();

String createTableSQL = "CREATE TABLE IF NOT EXISTS students ("

+ "roll\_no VARCHAR(10) PRIMARY KEY, "

+ "name VARCHAR(100), "

+ "branch VARCHAR(50), "

+ "address VARCHAR(255))";

stmt.executeUpdate(createTableSQL);

response.getWriter().println("Student table created successfully.");

stmt.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

response.getWriter().println("<a href='index.html'>Back to Home</a>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

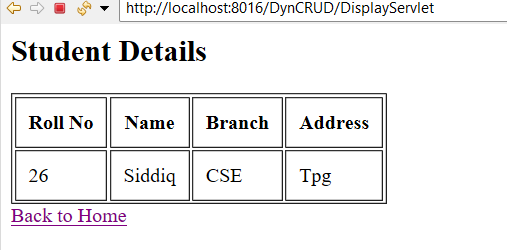
// TODO Auto-generated method stub

doGet(request, response);

}}

**DisplayServlet.java:**

package com.crud;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class DisplayServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public DisplayServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.setContentType("text/html");

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/vasavi", "root", "root");

Statement stmt = con.createStatement();

String query = "SELECT \* FROM students";

ResultSet rs = stmt.executeQuery(query);

response.getWriter().println("<html><body>");

response.getWriter().println("<h2>Student Details</h2>");

response.getWriter().println("<table border='1' cellpadding='10'>");

response.getWriter().println("<tr><th>Roll No</th><th>Name</th><th>Branch</th><th>Address</th></tr>");

while (rs.next()) {

response.getWriter().println("<tr>");

response.getWriter().println("<td>" + rs.getString("roll\_no") + "</td>");

response.getWriter().println("<td>" + rs.getString("name") + "</td>");

response.getWriter().println("<td>" + rs.getString("branch") + "</td>");

response.getWriter().println("<td>" + rs.getString("address") + "</td>");

response.getWriter().println("</tr>");

}

response.getWriter().println("</table>");

response.getWriter().println("</body></html>");

rs.close();

stmt.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

response.getWriter().println("An error occurred while retrieving student details.");

}

response.getWriter().println("<a href='index.html'>Back to Home</a>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

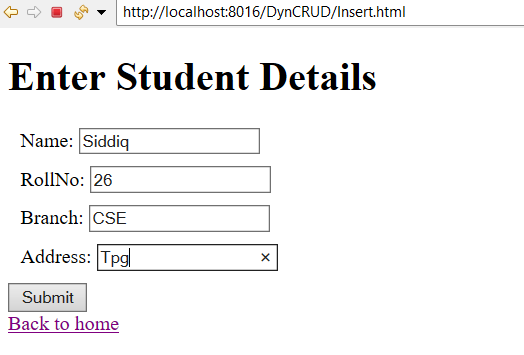
// TODO Auto-generated method stub

doGet(request, response);

}

}

**InsertStudentServlet.java:**

package com.crud;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

public class InsertStudentServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public InsertStudentServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

String rollNo = request.getParameter("roll\_no");

String name = request.getParameter("name");

String branch = request.getParameter("branch");

String address = request.getParameter("address");

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/vasavi", "root", "root");;

String query = "INSERT INTO students (roll\_no, name, branch, address) VALUES (?, ?, ?, ?)";

PreparedStatement pstmt = con.prepareStatement(query);

pstmt.setString(1, rollNo);

pstmt.setString(2, name);

pstmt.setString(3, branch);

pstmt.setString(4, address);

int rowsInserted = pstmt.executeUpdate();

response.getWriter().println(rowsInserted + " student record inserted successfully.");

pstmt.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

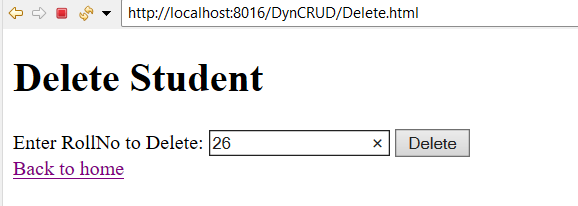
response.getWriter().println("<a href='index.html'>Back to Home</a>");

}

}

**DeleteStudentServlet.java:**

package com.crud;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

public class DeleteStudentServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public DeleteStudentServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

String rollNo = request.getParameter("roll\_no");

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/vasavi", "root", "root");

String query = "DELETE FROM students WHERE roll\_no=?";

PreparedStatement pstmt = con.prepareStatement(query);

pstmt.setString(1, rollNo);

int rowsDeleted = pstmt.executeUpdate();

response.getWriter().println(rowsDeleted + " student record deleted successfully.");

pstmt.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

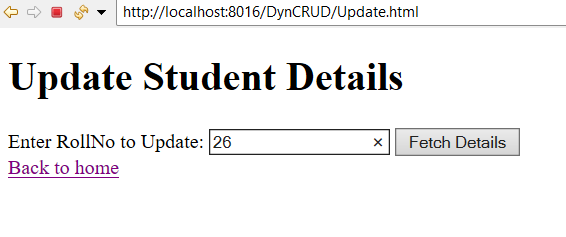
}

response.getWriter().println("<a href='index.html'>Back to Home</a>");

}}

**UpdateStudentServlet.java:**

package com.crud;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

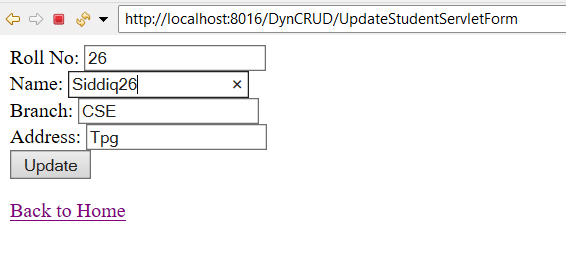
import java.sql.PreparedStatement;

import java.sql.ResultSet;

/\*\*

\* Servlet implementation class UpdateStudentServlet

\*/

public class UpdateStudentServletForm extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public UpdateStudentServletForm() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

String rollNo = request.getParameter("roll\_no");

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/vasavi", "root", "root");

String query = "SELECT \* FROM students WHERE roll\_no=?";

PreparedStatement pstmt = con.prepareStatement(query);

pstmt.setString(1, rollNo);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

response.getWriter().println("<form action='UpdateStudentServlet' method='post'>");

response.getWriter().println("Roll No: <input type='text' name='roll\_no' value='" + rs.getString("roll\_no") + "' readonly><br>");

response.getWriter().println("Name: <input type='text' name='name' value='" + rs.getString("name") + "'><br>");

response.getWriter().println("Branch: <input type='text' name='branch' value='" + rs.getString("branch") + "'><br>");

response.getWriter().println("Address: <input type='text' name='address' value='" + rs.getString("address") + "'><br>");

response.getWriter().println("<input type='submit' value='Update'>");

response.getWriter().println("</form>");

} else {

response.getWriter().println("Invalid Roll No.");

}

rs.close();

pstmt.close();

con.close();

} catch (Exception e) {

e.printStackTrace();

}

response.getWriter().println("<a href='index.html'>Back to Home</a>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

**Hibernate java application**

**Steps to Set Up a Hibernate Java Application**

1. **Download Hibernate** 
   * **Download the necessary Hibernate from the official Hibernate website or a trusted Maven repository.**
2. **Add Hibernate JAR Files to Class path**
   * **Right-click on your project in your IDE.**
   * **Select Build Path → Configure Build Path → Libraries.**
   * **Add the required Hibernate JAR files to the classpath.**
3. **Create Packages and Classes**
   * **Inside the src folder, create a new package: com.cseb.**
   * **In the com.cseb package, create two Java files:**
     + **Main.java (to handle application logic)**
     + **Student.java (to define the Student entity with attributes such as roll No, and name).**
4. **Add Hibernate Configuration File**
   * **In the src folder, add a new file named hibernate.cfg.xml.**
   * **Configure hibernate.cfg.xml with your database connection details and mapping settings for the Student entity.**

**student1.java:**

package com.cseb;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table

public class student1 {

@Id

private int id;

private String name;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

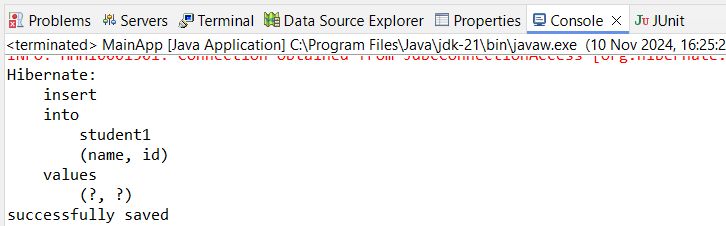
this.name = name; }}

**MainApp.java:**

package com.cseb;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class MainApp {

public static void main(String[] args) {

student1 s = new student1();

s.setId(266);

s.setName("Siddiq");

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession();

Transaction t = session.beginTransaction();

session.persist(s);

t.commit();

session.close();

System.out.println("successfully saved");

}

}**hibernate.cfg.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/cseb</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQL8Dialect</property>

<property name=*"hibernate.current\_session\_context\_class"*>thread</property>

<property name=*"hibernate.show\_sql"*>true</property>

<property name=*"hibernate.format\_sql"*>true</property>

<property name=*"hibernate.hbm2ddl.auto"*>update</property>

<mapping class=*"com.cseb.student1"* />

</session-factory>

</hibernate-configuration>

**Hibernate dynamic project**

**Steps to Create a Hibernate Dynamic Web Project**

1. **Download Hibernate Libraries**
   * **Visit the official** [**Hibernate website**](https://hibernate.org/) **and download the necessary Hibernate libraries**
2. **Add MySQL Connector and Hibernate JARs**
   * **Navigate to the lib folder in the Apache Tomcat directory and add the MySQL Connector JAR along with the downloaded Hibernate JAR files.**
3. **Create a New Dynamic Web Project**
   * **Open Eclipse and create a new dynamic web project named Hibernateser2.**
4. **Set Up the Package Structure**
   * **Create a package named com.cseb.**
5. **Develop Servlet Programs**
   * **Write the required servlet programs within the com.cseb package.**
6. **Create hibernate.cfg.xml**
   * **In the src directory, create hibernate.cfg.xml and configure it with the necessary Hibernate settings.**
7. **Create insert.html**
   * **Create insert.html in the web content folder and include the relevant code.**
8. **Run the Application**
   * **Run insert.html to start the application.**

**StudentNew.java:**

package com.cseb;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table

public class StudentNew {

@Id

private int rno;

private String name;

private String branch;

private String phone;

public int getRno() {

return rno;

}

public void setRno(int rno) {

this.rno = rno;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getBranch() {

return branch;

}

public void setBranch(String branch) {

this.branch = branch;

}

public String getPhone() {

return phone;

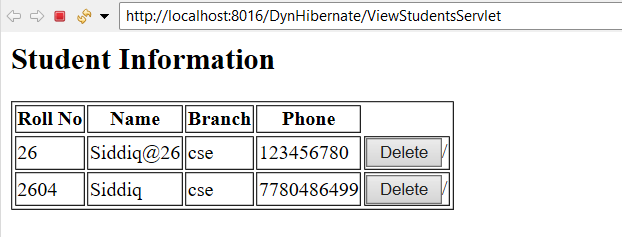
}

public void setPhone(String phone) {

this.phone = phone;

}

}**ViewStudentsServlet.java:**

package com.cseb;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

public class ViewStudentsServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory factory = cfg.buildSessionFactory();

Session session = factory.openSession();

List<StudentNew> students = session.createQuery("from StudentNew", StudentNew.class).list();

response.setContentType("text/html");

PrintWriter out = response.getWriter();

out.println("<h2>Student Information</h2>");

if (students.isEmpty()) {

out.println("<p>No student records found.</p>");

} else {

out.println("<table border='1'>");

out.println("<tr><th>Roll No</th><th>Name</th><th>Branch</th><th>Phone</th></tr>");

for (StudentNew student : students) {

out.println("<tr>");

out.println("<td>" + student.getRno() + "</td>");

out.println("<td>" + student.getName() + "</td>");

out.println("<td>" + student.getBranch() + "</td>"); // Display branch

out.println("<td>" + student.getPhone() + "</td>"); // Display phone

out.println("<td>"

+ "<form method = 'post' action = 'DeleteStudentServlet'><input type = 'hidden' name = 'rno' value="+student.getRno()+" ><button type = 'submit' value = 'Submit'>Delete</button>/<form></td></tr>");

out.println("</tr>");

}

out.println("</table>");

}

session.close();

factory.close();

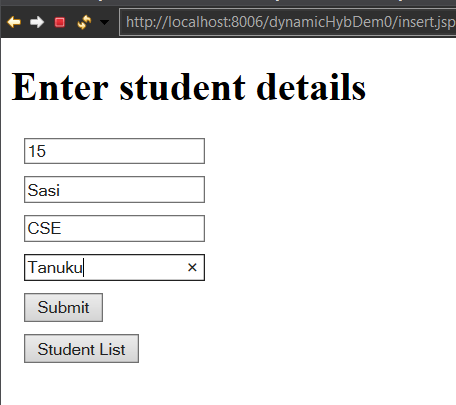
}

}

**StudentService.java:**

package com.cseb;

import jakarta.servlet.RequestDispatcher;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import org.hibernate.query.Query;

public class StudentService extends HttpServlet {

private static final long serialVersionUID = 1L;

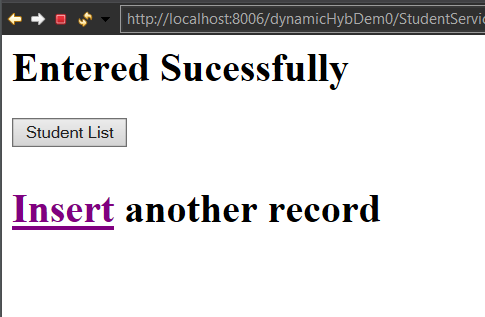
public StudentService() {

super();

// TODO Auto-generated constructor stub

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

 // TODO Auto-generated method stub

PrintWriter out = response.getWriter();

Student s = new Student();

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session session = sf.openSession();

Transaction t = session.beginTransaction();

//Inserting a Record

String name = request.getParameter("name");

String branch = request.getParameter("branch");

String address = request.getParameter("address");

int rno = Integer.parseInt(request.getParameter("rollno"));

s.setRno(rno);

s.setName(name);

s.setBranch(branch);

s.setAddress(address);

session.persist(s);

t.commit();

out.println("<h1>Entered Sucessfully<h1/>");

out.println("<form method = 'get' action = 'StudentList'>"

+ "<input type = 'submit' value = 'Student List'/>"

+ "</form>");

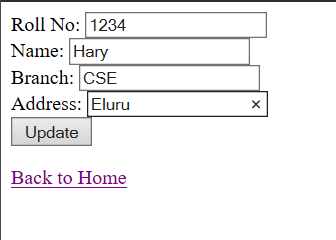
out.println("<a href = 'insert.jsp'>Insert</a> another record");

}

}

**Update.java:**

package com.cseb;



import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class Update extends HttpServlet {

private static final long serialVersionUID = 1L;

public Update() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

int rno = Integer.parseInt(request.getParameter("roll\_no"));

PrintWriter out = response.getWriter();

String name = request.getParameter("name");

String brn = request.getParameter("branch");

String address = request.getParameter("address");

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session session = sf.openSession();

Transaction ts = session.beginTransaction();

try {

Student student = session.get(Student.class, rno);

if (student != null) {

student.setRno(rno);

student.setName(name);

student.setBranch(brn);

student.setAddress(address);

session.update(student);

out.println("Updated");

out.print("<form method = \"get\" action = \"StudentList\" >\r\n"

+ " <input type = \"Submit\" value = \"Student List\"/>\r\n"

+ "</form>");

ts.commit();

}

else {

response.getWriter().println("Student not found.");

}

}

catch(Exception e) {

e.printStackTrace();

}

}

}

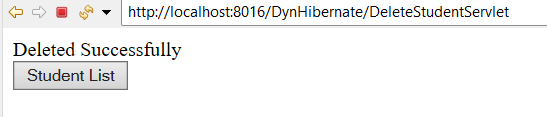
**DeleteStudentServlet.java:**

package com.cseb;

import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class DeleteStudentServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

public DeleteStudentServlet() {

super();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session session = sf.openSession();

Transaction t = session.beginTransaction();

int rno = Integer.parseInt(request.getParameter("rno"));

try {

StudentNew student = session.get(StudentNew.class, rno);

session.remove(student);

t.commit();

out.println("Deleted Successfully");

}

catch(Exception e) {

out.println("Error Occured");

}

out.println("<form method = 'get' action = 'ViewStudentsServlet'>"

+ "<input type = 'submit' value = 'Student List'/>"

+ "</form>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

**Creating simple maven java application**

**1.Click on File, click on new and click on Maven Project**

**2. click on next and select Internal in catalog and type quickstart in filter**

**3. select group id artefact id version(1.1) and click on next**

**4. Enter the group id as com.cseb and artifact id as HibDemo and click on finish**

**5. It will take some time to create a Maven Project and type Y in console**

**whenever it asks you.**

**6. Click on com.cseb.HibDemo, click on App.java**

**7. Run App.java(to check whether Maven Project is running or not)**

**8. Now Right click on com.cseb.HibDemo, click on new, click on class**

**Student2.java:**

package com.cseb;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Student2 {

@Id

private int rno;

private String name;

private String brn;

public int getRno() {

return rno;

}

public void setRno(int rno) {

this.rno = rno;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getBrn() {

return brn;

}

public void setBrn(String brn) {

this.brn = brn;

}

} **App.java:**

package com.cseb;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

public class App

{

public static void main( String[] args )

{

Student2 s = new Student2();

s.setRno(8);

s.setName("Abubakar");

s.setBrn("CSE");

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sa = cfg.buildSessionFactory();

Session session = sa.openSession();

Transaction tx = session.beginTransaction();

session.persist(s);

tx.commit();

session.close(); }}

**Add these dependencies in pom.xml**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.hibernate.orm/hibernate-core -->

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.0.0.Final</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.mysql/mysql-connector-j -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.3.0</version>

</dependency>

**Adding Hibernate Plugin to the Eclipse**

**1. Click on Help menu, click on Eclipse Market place and search for JBossTools 4.28.0 final**

**2. click on install**

**3. Uncheck everything else and check Hibernate and install it**

**(while installing it asks you for Trust, check the trust)**

**4. Next we have to specify hibernate.cfg.xml file**

**(In this we have to specify database and credentials)**

**5. Right click on your project, click on new and click on other**

**6. Search for hibernate**

**7. Select Hibernate Configuration(cfg.xml), click on next**

**8. In Database Dialect select MySQL**

**9. Select Driver class**

**10. Select Connection URL**

**11.Enter Username and Password**

**12. Click on finish**

**13. Add the following tag in the hibernate.cfg.xml**

**<property name="hbm2ddl.auto">update</property>**

**14. Save the file**

**15. Run your Application and check for the Table students in your database**

**hibernate.cfg.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.cj.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/cse\_b?useSSL=false&amp;serverTimezone=UTC</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"hibernate.show\_sql"*>true</property>

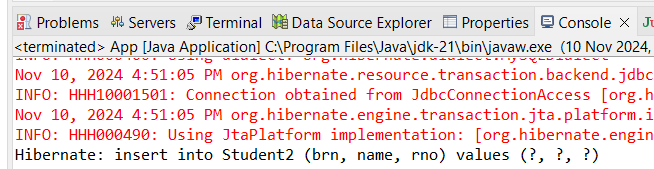
<property name=*"hbm2ddl.auto"*>update</property>

<!-- Map the Student2 class -->

<mapping class=*"com.cseb.Student2"* />

</session-factory>

</hibernate-configuration>



**Creating maven dynamic web project**

**1.Click on File, then New, and select Maven Project.**

**2.Click Next, filter by "apache" in the catalog, and select org.apache.maven.archetypes as GroupId and maven-archetype-webapp as ArtifactId.**

**3. Click Next.**

**4. Enter GroupId as com.cseb and ArtifactId as HibDy, then click Finish.**

**5. Wait for the Maven project creation and type Y in the console when prompted.**

**6. Go to src/main/webapp and click on index.jsp.**

**7. Run index.jsp to check if the Maven project is running.**

**8. Right-click on com.cseb.mavenDynDemo, select New, and click on Class.**

**9.Create Student class and the following code.**

**Student.java:**

package mavenDynDemo;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Student {

@Id

int rno;

String name;

String branch ;

String address;

public int getRno() {

return rno;

}

public void setRno(int rno) {

this.rno = rno;

}

public String getBranch() {

return branch;

}

public void setBranch(String branch) {

this.branch = branch;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;}}

**StudentList.java:**

package mavenDynDemo;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import org.hibernate.query.Query;

import mavenDynDemo.Student;

public class StudentList extends HttpServlet {

private static final long serialVersionUID = 1L;

public StudentList() {

super();

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session session = sf.openSession();

Transaction t = session.beginTransaction();

Query q = session.createQuery("from Student", Student.class);

List<Student> students= q.getResultList();

if(students.isEmpty()) {

out.println("<h1>No Records Found</h1>");

}

else {

out.println("<h2 style = 'margin:10px;'>Student List</h2>");

out.println("<table style = 'margin:10px;' border = 1><tr><th>Name</th><th> Roll No</th><th> Branch</th><th> Address</th><th>Action</th></tr>");

for(Student student : students) {

out.println("<tr><td>"+student.rno+"</td>");

out.println("<td>"+student.name+"</td>");

out.println("<td>"+student.branch+"</td>");

out.println("<td>"+student.address+"</td>");

out.println("<td>"

+ "<form style='display:inline;' method = 'post' action = 'Delete'><input type = 'hidden' name = 'rno' value='"+student.rno+"' ><button type = 'submit' value = 'Submit'>Delete</button></form>");

out.print( "<form style='display:inline;' method = 'post' action = 'UpdateForm'><input type = 'hidden' name = 'rno' value='"+student.rno+"' ><input type = 'submit' value = 'Update' onclick='return confirm(\\\"Are you sure you want to delete?\\\")'/></form></td></tr>");

}

out.println("</table>");

out.println("Go Back to "+"<a href = 'insert.jsp'>insert</a>");

}

t.commit();

session.close();

}

}

**Update.java:**

package mavenDynDemo;

import java.io.IOException;

import java.io.PrintWriter;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import mavenDynDemo.Student;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.\*;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

public class Update extends HttpServlet {

private static final long serialVersionUID = 1L;

public Update() {

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

response.getWriter().append("Served at: ").append(request.getContextPath());

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

int rno = Integer.parseInt(request.getParameter("roll\_no"));

PrintWriter out = response.getWriter();

String name = request.getParameter("name");

String brn = request.getParameter("branch");

String address = request.getParameter("address");

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session session = sf.openSession();

Transaction ts = session.beginTransaction();

try {

Student student = session.get(Student.class, rno);

if (student != null) {

student.setRno(rno);

student.setName(name);

student.setBranch(brn);

student.setAddress(address);

session.update(student);

out.println("Updated");

out.print("<form method = \"get\" action = \"StudentList\" >\r\n"

+ " <input type = \"Submit\" value = \"Student List\"/>\r\n"

+ "</form>");

ts.commit();

}

else {

response.getWriter().println("Student not found.");

}

}

catch(Exception e) {

e.printStackTrace();

}

}

}

**Delete.java:**

package mavenDynDemo;

import java.io.IOException;

import java.io.PrintWriter;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.Transaction;

import org.hibernate.cfg.Configuration;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

public class Delete extends HttpServlet {

private static final long serialVersionUID = 1L;

public Delete() {

// TODO Auto-generated constructor stub

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

PrintWriter out = response.getWriter();

Configuration cfg = new Configuration();

cfg.configure("hibernate.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();

Session session = sf.openSession();

Transaction t = session.beginTransaction();

int rno = Integer.parseInt(request.getParameter("rno"));

try {

Student student = session.get(Student.class, rno);

session.remove(student);

t.commit();

out.println("Deleted Successfully");

}

catch(Exception e) {

out.println("Error Occured");

}

out.println("<form method = 'get' action = 'StudentList'>"

+ "<input type = 'submit' value = 'Student List' />"

+ "</form>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// TODO Auto-generated method stub

doGet(request, response);

}

}

**Insert.jsp:**

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Insert title here</title>

<style>

input {

display:block;

margin:10px;

}

</style>

</head>

<body>

<div style= "display : 'flex';flex-direction: 'column';justify-content: 'center';align-items:'center'">

<h1>Enter student details</h1>

<form method = "POST" action = "StudentService" >

<input type = "number" placeholder = "Enter RollNumber" name = "rollno" required/>

<input type = "text" placeholder = "Enter Name" name = "name" required/>

<input type = "text" placeholder = "Enter Branch" name = "branch" required/>

<input type = "text" placeholder = "Enter Address" name = "address" required/>

<input type = "Submit" value = "Submit">

</form>

<form method = "get" action = "StudentList" >

<input type = "Submit" value = "Student List"/>

</form>

</div>

</body>

</html>

**Add these dependencies to pom.xml file**

**Pom.xml:**

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.0.0.Final</version>

</dependency>

<dependency>

<groupId>jakarta.servlet</groupId>

<artifactId>jakarta.servlet-api</artifactId>

<version>5.0.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>jakarta.servlet.jsp</groupId>

<artifactId>jakarta.servlet.jsp-api</artifactId>

<version>3.0.0</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.3.0</version>

</dependency>

**hibernate.cfg.xml:**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.password"*>root</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/cse\_b</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

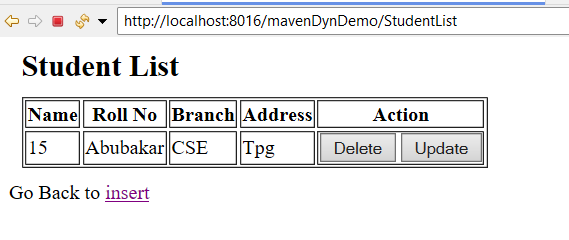
<property name = *"hbm2ddl.auto"*>update</property>

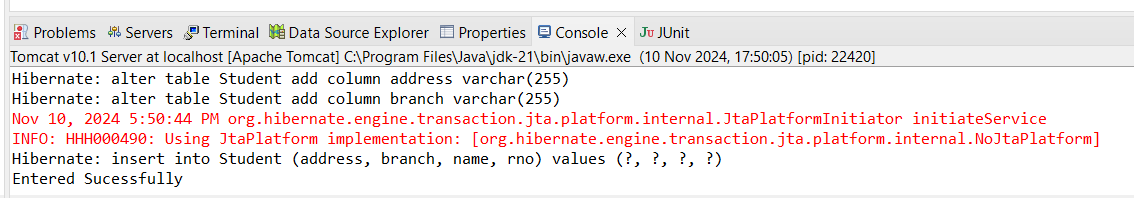
<property name=*"hibernate.show\_sql"*>true</property>

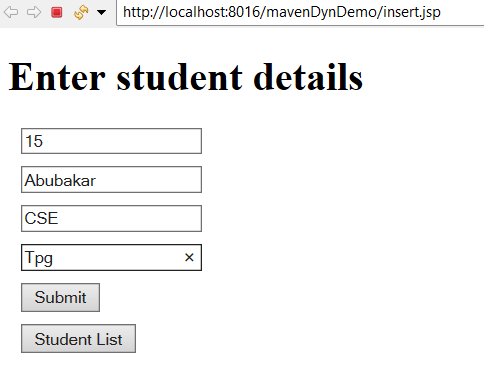
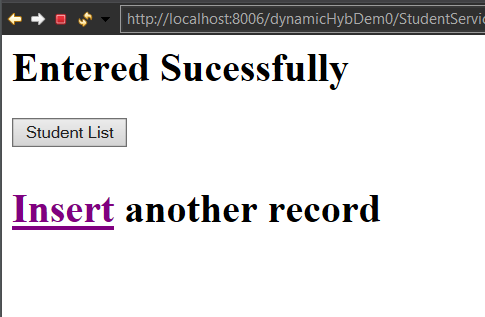
<mapping class=*"mavenDynDemo.Student"* />

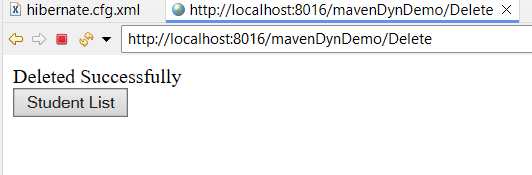
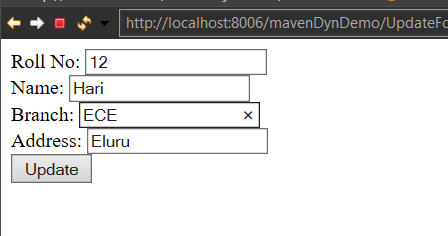
</session-factory>

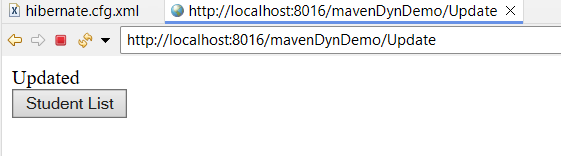
</hibernate-configuration>





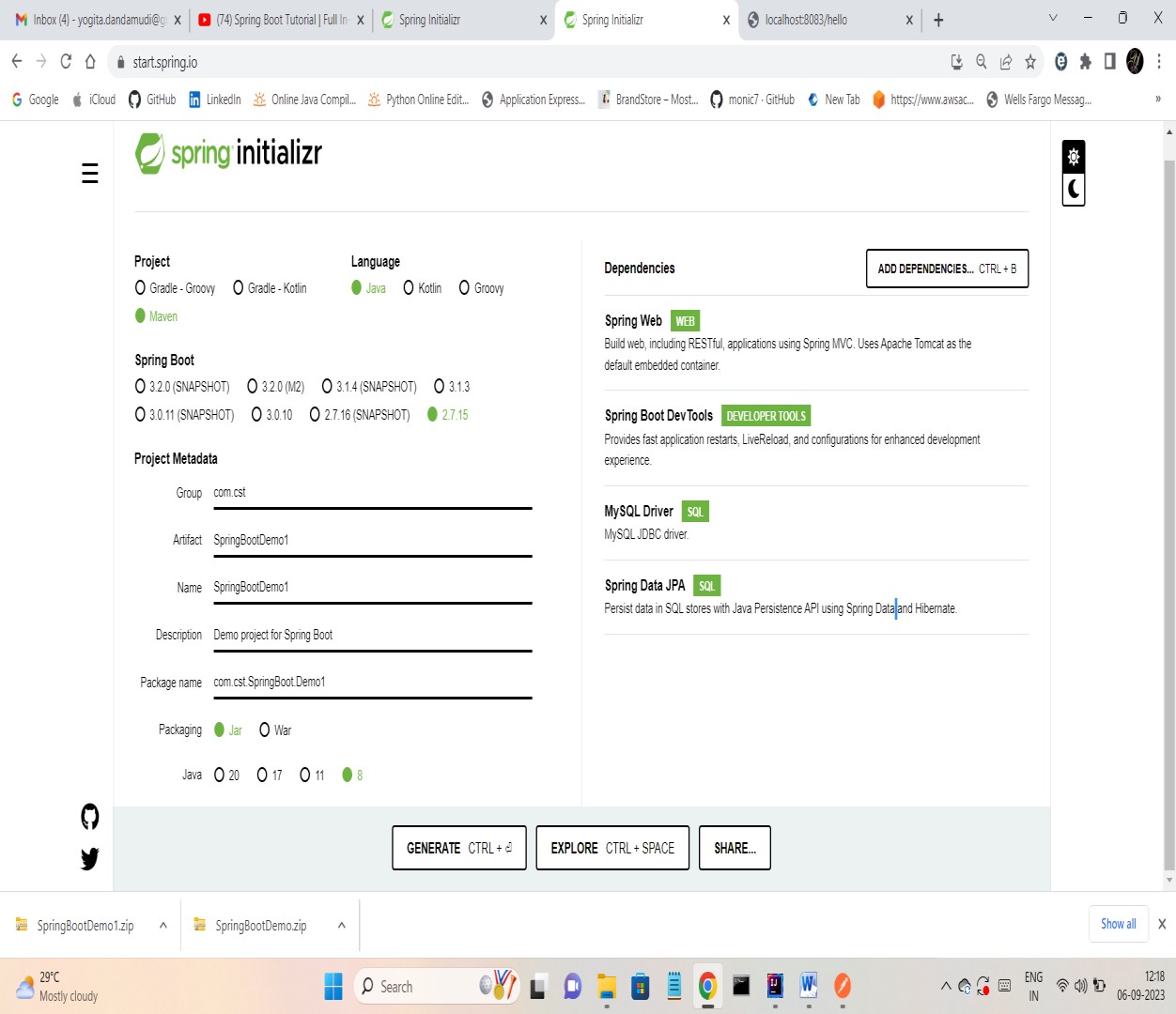




**Create Example programs Using Spring MVC Framework**

**Steps to create a Spring Boot MVC Application:**

1. 
2. Extract the zip file and open it in IntelliJ IDEA
3. Open the application.properties file, add the following content and Make the necessary changes .

server.port=8084

spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/cse

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

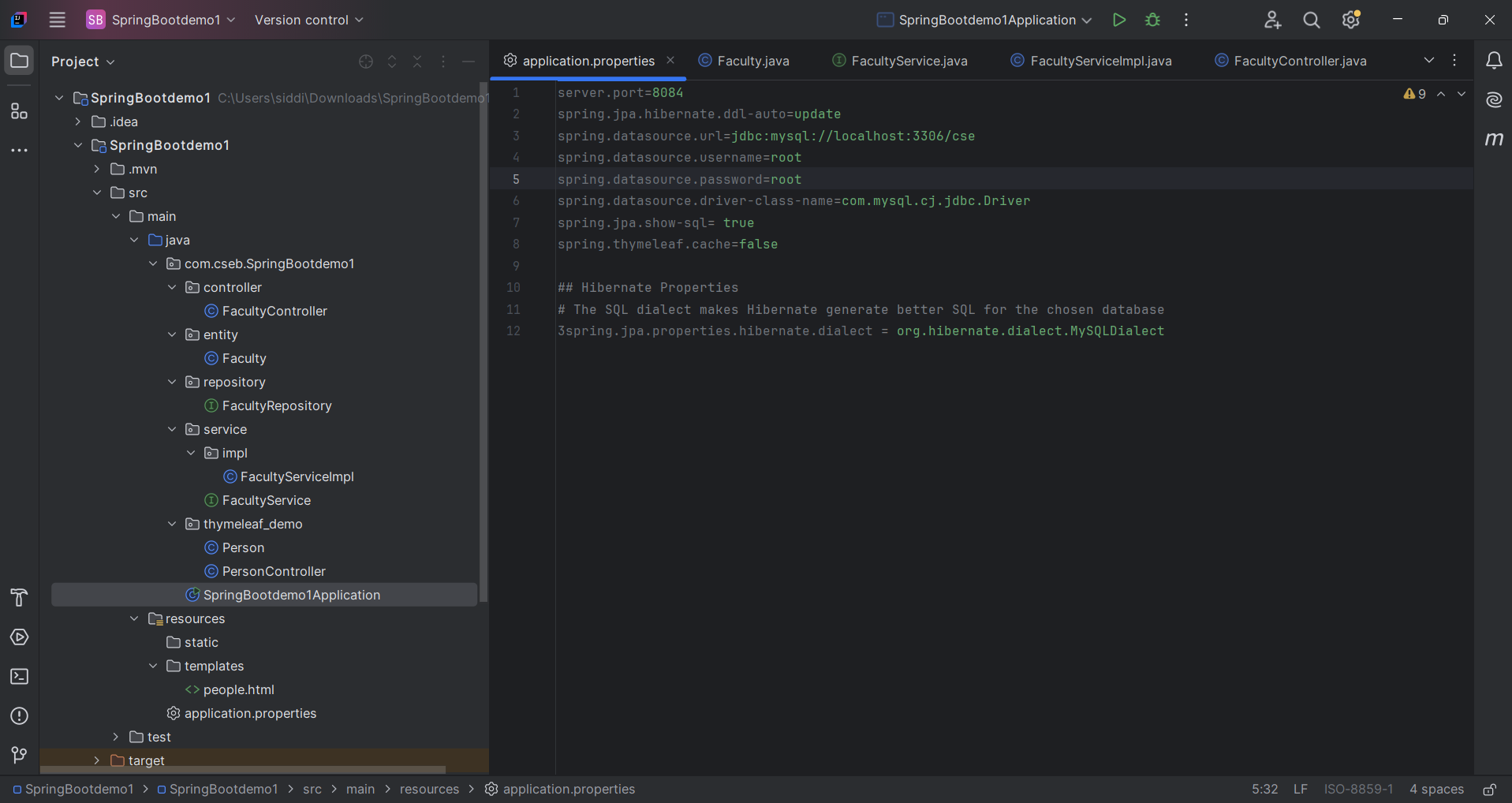
spring.jpa.show-sql= true

spring.thymeleaf.cache=false

## Hibernate Properties

# The SQL dialect makes Hibernate generate better SQL for the chosen database

3spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQLDialect

Create 4 packages controller, entity, repository, service  


1. Create a class Student under entity package  
   **Student.java:**package com.cseb.entity;  
   import jakarta.persistence.Entity;  
   import jakarta.persistence.Id;  
     
   @Entity  
   public class Student {  
    @Id  
    private int rno;  
    private String name;  
    private String brn;  
    public int getRno() {  
    return rno;  
    }  
    public void setRno(int rno) {  
    this.rno = rno;  
    }  
    public String getName() {  
    return name;  
    }  
    public void setName(String name) {  
    this.name = name;  
    }  
    public String getBrn() {  
    return brn;  
    }  
    public void setBrn(String brn) {  
    this.brn = brn;  
    }  
     
   }
2. Create an Interface StudentRepository under the package repository  
   **StudentRepository.java:**  
   package com.cseb.repository;  
   import com.cseb.entity.Student;  
   import org.springframework.data.jpa.repository.JpaRepository;  
     
   import java.util.List;  
     
   public interface StudentRepository extends JpaRepository<Student,Integer> {  
    List<Student> findByBrn(String brn);  
   }
3. Create an Interface under StudentService under the package service  
   **StudentService.java:**package com.cseb.services;  
   import java.util.\*;  
   import com.cseb.entity.Student;  
     
   public interface StudentService {  
    public Student save (Student student);  
    List<Student> fetchAll();  
    Student fetch(Integer id);  
    void deleteById(Integer id);  
    List<Student> fetchByBranch(String name);

}

1. Create a package impl under service
2. Create a Class StudentServiceImpl under impl package  
   **StudentServiceImpl.java:**  
   package com.cseb.services.impl;  
     
   import com.cseb.entity.Student;  
   import com.cseb.repository.StudentRepository;  
   import com.cseb.services.StudentService;  
   import org.springframework.beans.factory.annotation.Autowired;  
   import org.springframework.stereotype.Service;  
   import java.util.\*;  
     
   @Service  
   public class StudentServiceImpl implements StudentService {  
    @Autowired  
    private StudentRepository studentRepository;  
     
    @Override  
    public Student save(Student student) {  
    return studentRepository.save(student);  
    }  
     
    @Override  
    public List<Student> fetchAll() {  
    return studentRepository.findAll();  
    }  
     
    @Override  
    public Student fetch(Integer studentId) {  
    return studentRepository.findById(studentId).get();  
    }  
     
    @Override  
    public void deleteById(Integer studentId) {  
    studentRepository.deleteById(studentId);  
    }  
     
    @Override  
    public List<Student> fetchByBranch (String brn) {  
    return studentRepository.findByBrn(brn);  
    }  
     
   }
3. Create a class StudentController under controller package  
   **StudentController.java:**

package com.cseb.controller;

import com.cseb.entity.Student;

import com.cseb.services.StudentService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@CrossOrigin(origins = "http://localhost:3000")

@Controller

@RestController

public class StudentController {

@Autowired

private StudentService studentService;

@PostMapping("/save")

public Student save(@RequestBody Student student)

{

System.out.println(student);

return studentService.save(student);

}

@GetMapping("/fetchAll")

public List<Student> fetchAll()

{

return studentService.fetchAll();

}

@GetMapping("/fetchById/{id}")

public Student fetchById(@PathVariable("id") Integer studentId)

{

return studentService.fetch(studentId);

}

@DeleteMapping("/deleteById/{id}")

public List<Student> deleteByFacultyId(@PathVariable("id") Integer studentId)

{

studentService.deleteById(studentId);

return studentService.fetchAll();

}

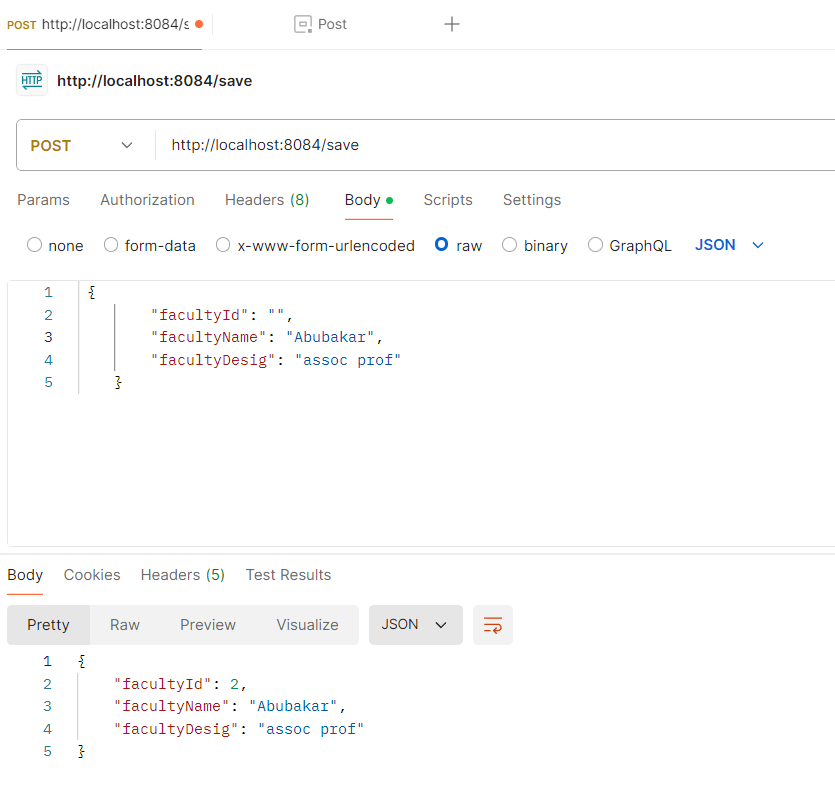
@PostMapping("/fetchByBranch")

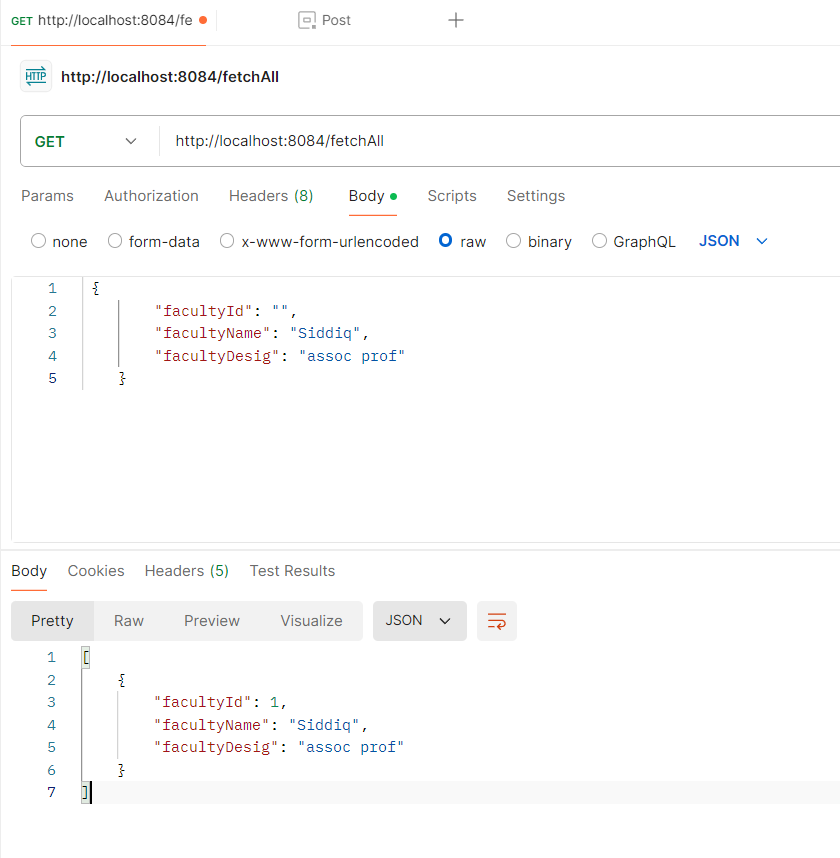
public List<Student> fetchByBranch(@RequestParam String brn){

return studentService.fetchByBranch(brn);

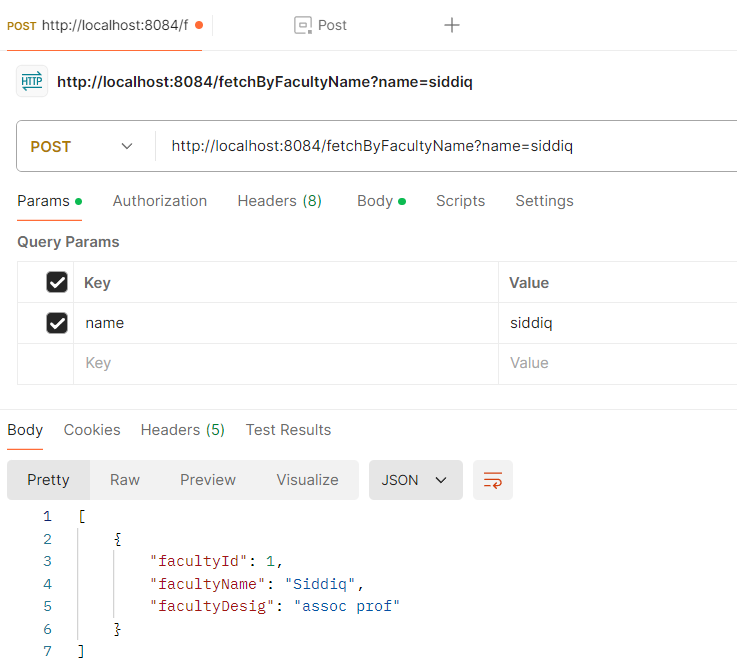
}

}

1. Run the SpringBootDemo1Application
2. Open the Postman app
3. Select the method POST and type the URL <http://localhost:8083/save>
4. Select body ,select raw, select JSON
5. Add the following  
   {  
    "rno" : 12,   
    "name" : "SASI",   
    "brn" : "CSE"  
   }
6. Select the method GET and type the URL http://localhost:8083/fetchAll
7. Test remaining API’s
8.  1.save()  
   2.fetchAll()



3.fetchFacultyName()



**Adding FrontEnd to a spring project using Thymeleaf  
ThymeLeaf:** Thymeleaf is a server-side Java template engine used in Spring Boot for rendering dynamic HTML and other document types. It integrates seamlessly with Spring MVC, allowing easy binding of data from controllers to views. Thymeleaf provides a natural templating syntax, making templates easy to design and preview. It supports automatic HTML escaping, enhancing security, and is ideal for server-side rendering in modern web applications. It simplifies the process of creating dynamic, data-driven web pages.  
**Steps to add thymeleaf to spring application:**  
1. Create a person entity under “com.example” package  
**Person.java**:

package com.example.thymeleaf\_demo;

public class Person {

String name;

int age;

public Person(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}  
**Person.html:**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Title</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">

</head>

<body>

<h1 th:text="${something}" />

<table class="table table-dark table-striped">

<thead>

<tr>

<th scope="col">Name</th>

<th scope="col">Age</th>

</tr>

</thead>

<tbody>

<tr th:each="person: ${people}">

<td th:text="${person.name}" />

<td th:text="${person.age}" />

</tr>

</tbody>

</table>

</body>

</html>  
  
**PersonController.java:**package com.example.thymeleaf\_demo;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import java.util.ArrayList;

import java.util.Arrays;

@Controller

public class PersonController {

@GetMapping

String getPeople(Model model)

{

model.addAttribute("something","this is coming from the controller");

model.addAttribute("people", Arrays.asList(

new Person("abc",28),

new Person("xyz",29),

new Person("mno",23)));

return "people";

}

}

