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\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_ bearing Roll No.\_\_\_\_\_\_\_\_\_\_\_ of****\_\_\_\_****Branch of* ***VIISemester*** *during the academic year* ***2023-24****.*

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

**EXTERNAL**

**SYLLABUS**

**CO1:** Demonstrate IDE tools Installation. **(K3)**

**CO2:** Develop programs using servlets. **(K3)**

**CO3:** Illustrate MVC architecture. **(K3)**

**CO4:** Demonstrate applications of Hibernate. **(K3)**

**CO5:** Illustrate Spring MVC Framework. **(K3)**

**Exercise 1: Basic Installation of IDEs and Development Tools (use any one of the following IDEs):**

The Student should know about installing IDEs (Integrated Development Environment) in the system such as IntelliJ, Eclipse, NetBeans, Macromedia Dream Viewer and Databases such as My-SQL, Oracle, SQL Server etc.

**Exercise 2:**

**Understanding about Servlets:** Create Example programs Using the below concepts

* + Introduction to Servlets.
  + Write Servlet application to print current date & time.
  + Write Servlet program to link Html & Servlet Communication.
  + Write Servlet program to Auto refresh a page.
  + Demonstrate session tracking using small program.
  + Write Servlet program to insert/delete/update the record into database.
  + Write Servlet program to add cookie to selected value.

**Exercise 3:**

**Understanding about Model View Controller:** Create Example programs Using the below concepts

* + Introduction to MVC in java.
  + Create sample program on Model Layer in MVC Using Java.
  + Create sample program on View Layer in MVC Using Java.
  + Create sample program on Controller Layer in MVC Using Java.
  + Demonstrate MVC Deployment in java.
  + Rules for MVC Mapping in Server Side.
  + How to use Web Server for MVC Deployment.

**Exercise 4:**

**Understanding about Spring MVC Framework:** Create Example programs Using the below concepts

* + Introduction to Spring MVC.
  + Demonstrate the usage of Dispatcher Servlet in Spring MVC.
  + Load the spring jar files or add dependencies in the case of Maven.
  + Create the controller class.
  + Provide the entry of controller in the web.xml file.
  + Define the bean in the separate XML file.
  + Display the message in the JSP page.
  + Start the server and deploy the project.
  + Execute the application on webserver using Spring MVC.

**Exercise 5:**

**Understanding about Hibernate:** Create Example programs Using the below concepts

* + - Introduction to Hibernate.
    - What is ORM
    - Demonstrate the components of Hibernate
    - How to persist objects using Hibernate
    - How to use map using XML and Annotations
    - How to implement Inheritance in Hibernate
    - Working with relationship between entities - association
    - Transactions in Hibernate
    - Querying with HQL (Hibernate Query Language)
    - Various other forms of querying - Criteria, QBE etc.

**Exercise 6:**

**Understanding Some Debugging Tools in Java :**

The Student should know about how to debug the java codes using some debugging tools such as:

* + NetBeans.
  + Eclipse.
  + IntelliJ IDEA.
  + Visual Studio Code.

## Additional Tasks:

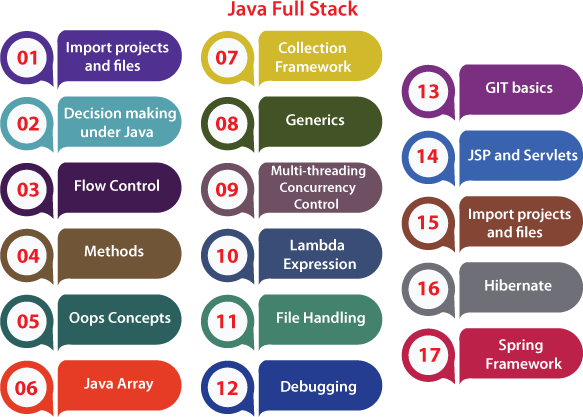
* How we can import project files into IDEs.
* How we can import eclipse (Java IDE) projects.
* How to Create new project in IDEs.
* How to Save the Project using packages.
* How to Compile the Project or Program in IDE.
* How to Build the Project or Program in IDE.
* How to Debug the Errors in IDE.

## Reference Books:

1. Jeffrey C. Jackson, "Web Technologies--A Computer Science Perspective", Pearson Education, 2006.
2. Murach's Java Servlets and JSP, 3rd Edition by (Murach: Training & Reference) 3rd Edition.
3. Spring and Hibernate Paperback – 1 July 2017 by K. Santosh Kumar.
4. Full Stack Java Development with Spring MVC, Hibernate, jQuery, and Bootstrap by Mayur Ramgir,Wiley.

# **Exercise-1** : 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:

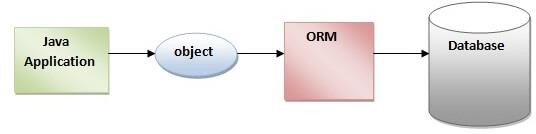
This hibernate tutorial provides in-depth concepts of Hibernate Framework with simplified examples. It was started in 2001 by Gavin King as an alternative to EJB2 style entity bean.

* 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

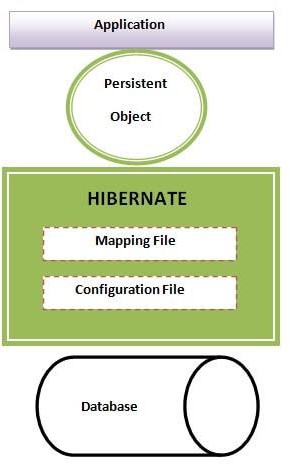
An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database. The ORM tool internally uses the JDBC API to interact with the database.



## 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

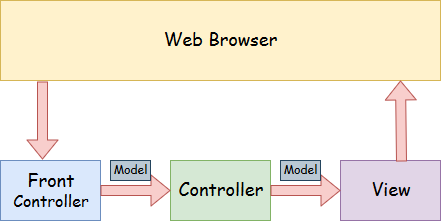
**MVC**:

* 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

**Exercise 2:** Create Example programs Using Servlets

* 1. Write Servlet application to print current date & time:

## Program:

import java.io.IOException; import java.io.PrintWriter;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; public class DateTime extends HttpServlet

{

public void service(HttpServletRequest req,HttpServletResponse res) throws IOException

{

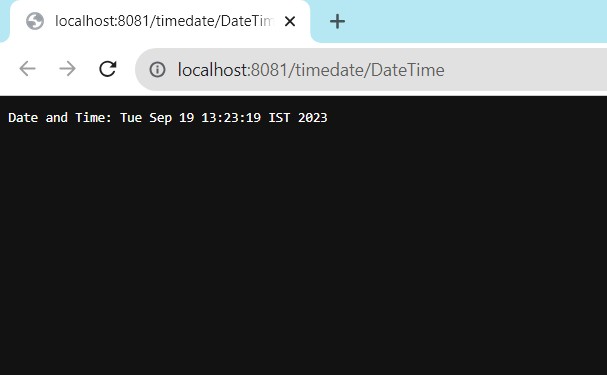
}

}

## Output:

PrintWriter out=res.getWriter();

out.println("Date and Time: "+ (new java.util.Date().toString()));



* 1. Write Servlet program to link Html & Servlet Communication

## Program:

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.Calendar;

public class ServletHtml extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {

res.setContentType("text/html"); PrintWriter pw = res.getWriter(); Calendar cal = Calendar.getInstance();

int hour = cal.get(Calendar.HOUR\_OF\_DAY);//24 hrs format if(hour<12)

pw.println("Good Morning!!"); else if (hour < 16)

pw.println("Good afternoon"); else if(hour<20)

pw.println("Good evening"); else

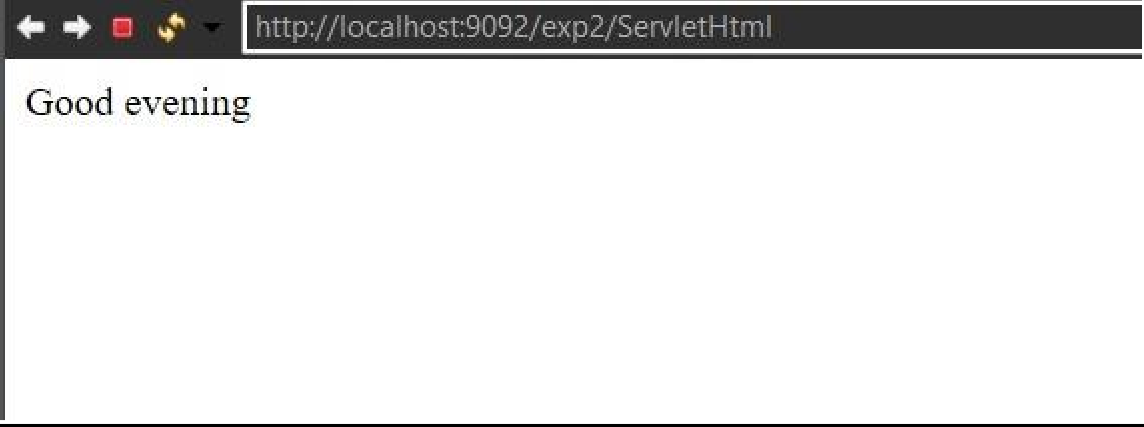
pw.println("Good night");

}

## Output:

pw.close();

}



* 1. Write Servlet program to Auto refresh a page

## Program:

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.Calendar;

import java.util.GregorianCalendar;

public class AutoRefreshPage extends HttpServlet { private static final long serialVersionUID = 1L;

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

response.setIntHeader("Refresh", 1);

}

## Output:

response.setContentType("text/html"); Calendar calendar = new GregorianCalendar(); String am\_pm;

int hour = calendar.get(Calendar.HOUR);

int minute = calendar.get(Calendar.MINUTE); int second = calendar.get(Calendar.SECOND); if(calendar.get(Calendar.AM\_PM) == 0)

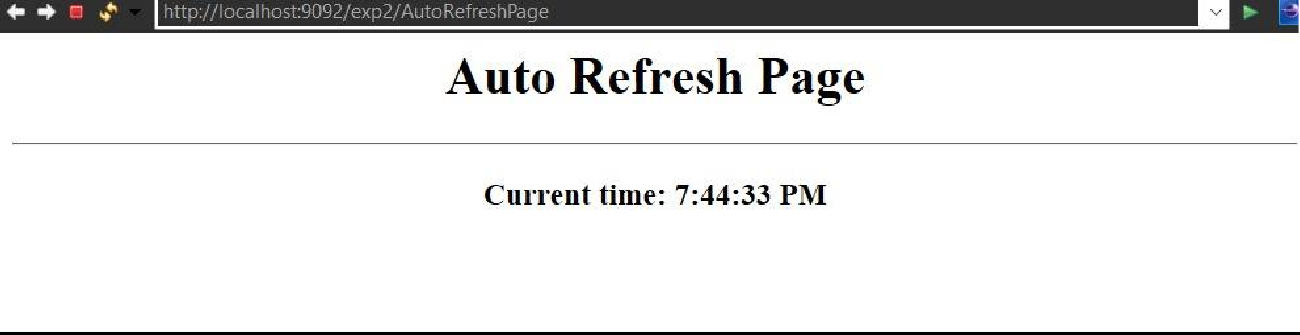
am\_pm = "AM"; else

am\_pm = "PM";

String CT = hour+":"+ minute +":"+ second +" "+ am\_pm; PrintWriter out = response.getWriter();

out.println("<h1 align='center'>Auto Refresh Page</h1><hr>"); out.println("<h3 align='center'>Current time: "+CT+"</h3>");

}



* 1. Demonstrate session tracking using small program

## Program:

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.util.Date;

public class SessionTracking extends HttpServlet { private static final long serialVersionUID = 1L;

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

HttpSession session = request.getSession(true);

// Get session creation time.

Date createTime = new Date(session.getCreationTime());

// Get last access time of this web page.

Date lastAccessTime = new Date(session.getLastAccessedTime()); String title = "Welcome Back to my website";

Integer visitCount = new Integer(0);

String visitCountKey = new String("visitCount"); String userIDKey = new String("userID");

String userID = new String("Shirin");

// Check if this is new comer on your web page. if (session.isNew())

{

title = "Welcome to my website"; session.setAttribute(userIDKey, userID);

}

else

{

visitCount = (Integer)session.getAttribute(visitCountKey); visitCount = visitCount + 1;

userID = (String)session.getAttribute(userIDKey);

}

session.setAttribute(visitCountKey, visitCount);

// Set response content type response.setContentType("text/html"); PrintWriter out = response.getWriter(); String docType =

"<!doctype html public \"-//w3c//dtd html 4.0 " + "transitional//en\">\n";

out.println(docType + "<html>\n" +

"<head><title>" + title + "</title></head>\n" + "<body bgcolor=\"#e5f7c0\">\n" +

"<h1 align=\"center\">" + title + "</h1>\n" +

"<h2 align=\"center\">Session Infomation</h2>\n" + "<table border=\"1\" align=\"center\">\n" +

"<tr bgcolor=\"#eadf8c\">\n" +

"<th>Session info</th><th>value</th></tr>\n" + "<tr>\n" +

" <td>id</td>\n" +

" <td>" + session.getId() + "</td></tr>\n" + "<tr>\n" +

}

### Output:

" <td>Creation Time</td>\n" + " <td>" + createTime +

" </td></tr>\n" + "<tr>\n" +

" <td>Time of Last Access</td>\n" + " <td>" + lastAccessTime +

" </td></tr>\n" + "<tr>\n" +

" <td>User ID</td>\n" + " <td>" + userID +

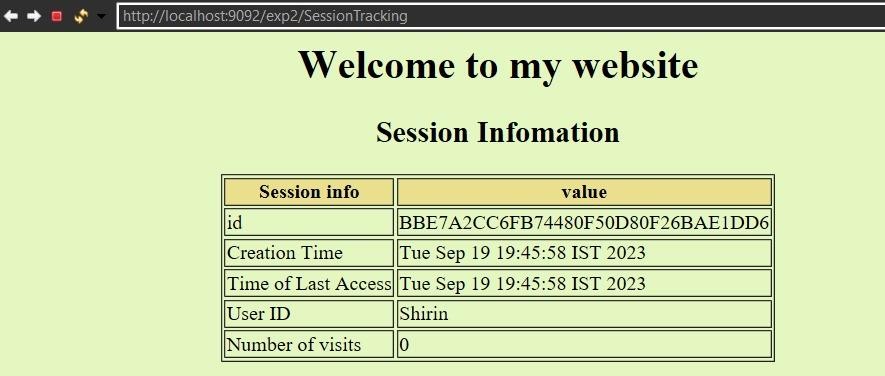
" </td></tr>\n" + "<tr>\n" +

" <td>Number of visits</td>\n" +

" <td>" + visitCount + "</td></tr>\n" + "</table>\n" +

"</body></html>");

}



* 1. Write Servlet program to insert/delete/update the record into database

## Program:

package exp2; import java.sql.\*; import java.util.\*;

public class Databases{

public static void main(String args[])

{

try

{

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

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/db1","root","root"); Scanner sc=new Scanner(System.in);

while(true) {

System.out.println("------MENU ");

System.out.println("1.Insert\n2.Delete\n3.Update\n4.Display\n5.Exit\nEnter Choice ");

int var=sc.nextInt(); switch(var)

{

case 1:

int id,sal;

String name;

System.out.println("Enter id"); id=sc.nextInt(); System.out.println("Enter Salary");

sal=sc.nextInt(); System.out.println("Enter Name");

name=sc.next();

PreparedStatement st=con.prepareStatement("insert into emp

values(?,?,?)");

where id=?");

case 2:

st.setInt(1,id); st.setString(2,name); st.setInt(3,sal);

System.out.println("Data Entered Successfully!!!!!"); st.executeUpdate();

st.close(); break;

System.out.println("Enter id"); id=sc.nextInt();

PreparedStatement st1=con.prepareStatement("delete from emp

st1.setInt(1,id);

int i=st1.executeUpdate(); if(i==1)

{

}

else {

}

System.out.println("Deleted Successfully");

System.out.println("Id is Not Found");

st1.close(); break;

case 3:

System.out.println("Enter field to Update"); System.out.println("\n1.Name\n2.Salary\nEnter Choice..."); int ch=sc.nextInt();

switch(ch)

{

case 1:

name=? where id=?");

int id1; System.out.println("Enter id"); id1=sc.nextInt();

String name1; System.out.println("Enter Name"); name1=sc.next();

PreparedStatement st2=con.prepareStatement("update emp set

st2.setString(1, name1); st2.setInt(2,id1);

int i1=st2.executeUpdate(); if(i1==1)

{

}

else {

}

System.out.println("Updated Successfully");

System.out.println("Id is Not Found");

st2.close(); break;

case 2:

int id2;

sal=? where id=?");

System.out.println("Enter id"); id2=sc.nextInt();

int sal1;

System.out.println("Enter Salary"); sal1=sc.nextInt();

PreparedStatement st3=con.prepareStatement("update emp set

st3.setInt(1, sal1); st3.setInt(2,id2);

int i2=st3.executeUpdate(); if(i2==1)

{

}

else {

}

System.out.println("Updated Successfully");

System.out.println("Id is Not Found");

st3.close(); break;

default:

System.out.println("Invalid Choice!!!");

}

case 4:

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from emp");

"+rs.getInt(3));

while(rs.next())

System.out.println(rs.getInt(1)+" "+rs.getString(2)+"

stmt.close(); break;

case 5:

System.out.println("Thankyou!!!!!!!!!1"); con.close();

System.exit(0);

}

}

}

catch(Exception e)

{

System.out.println(e);

}

}

}

## Output:

------MENU------

1.Insert 2.Delete 3.Update 4.Display 5.Exit

Enter Choice...

1

Enter id 1

Enter Salary 100000

Enter Name cse

Data Entered Successfully!!!!!

|  |  |
| --- | --- |
| ------MENU------ | 2.Delete |
| 1.Insert | 3.Update |
| 2.Delete | 4.Display |
| 3.Update | 5.Exit |
| 4.Display | Enter Choice... |
| 5.Exit | 3 |
| Enter Choice... | Enter field to Update |
| 1 |  |
| Enter id | 1.Name |
| 2 | 2.Salary |
| Enter Salary | Enter Choice... |
| 1000 | 2 |
| Enter Name | Enter id |
| cst | 2 |
| Data Entered Successfully!!!!! | Enter Salary |
| ------MENU------ | 10000 |
| 1.Insert | Updated SuccessFully |
| 2.Delete | 1 cse 100000 |
| 3.Update | 2 cst 10000 |
| 4.Display | ------MENU------ |
| 5.Exit | 1.Insert |
| Enter Choice... | 2.Delete |
| 4 | 3.Update |
| 1 cse 100000 | 4.Display |
| 2 cst 1000 | 5.Exit |
| ------MENU------ | Enter Choice... |
| 1.Insert | 4 |

1 cse 100000

2 cst 10000

------MENU------

1.Insert 2.Delete 3.Update 4.Display 5.Exit

Enter Choice...

2

Enter id 2

Deleted SuccessFully

------MENU------

1.Insert 2.Delete

3.Update 4.Display 5.Exit

Enter Choice... 4

1 cse 100000

------MENU------

1. Insert 2.Delete 3.Update 4.Display 5.Exit

Enter Choice...

5

Thankyou!!!!!!!!

* 1. Write Servlet program to add cookie to selected value

## Program:

### Index.html:

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "[http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"](http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd)>

<head>

<title>CookiesExample</title>

</head>

<body>

<form method ="post" action="Cookie">

<fieldset style="width:14%; background-color:#ccffcc">

<h2>Select Course</h2> <hr>

<input type='radio' name='course' value='FST'>FST<br>

<input type='radio' name='course' value='Cloud'>Cloud<br>

<input type='radio' name='course' value='HCI'>HCI<br>

<input type='radio' name='course' value='SPM'>SPM<br><br>

<input type='submit'> <input type='reset'><br>

</fieldset>

</form>

</body>

</html>

### Cookie.java

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 Cookie extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException {

res.setContentType("text/html"); PrintWriter pw = res.getWriter();

Cookie []c = req.getCookies();

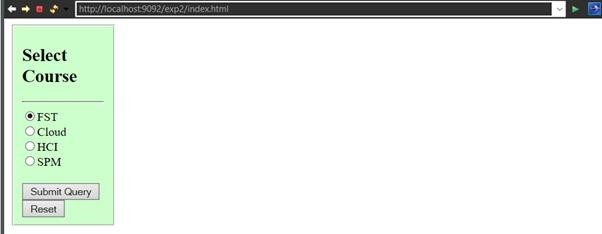
}

### Output:

String value = req.getParameter("course"); Cookie newCookie = new Cookie("Selected",value); res.addCookie(newCookie);

pw.println("<h4>Cookie added with value "+value+"</h4>");

}





# **Exercise 3:** Create Example programs Using Understanding about Model View Controller

### Employee.java:

public class Employee {

// declaring the variables private String EmployeeName; private String EmployeeId;

private String EmployeeDepartment;

// defining getter and setter methods public String getId() {

return EmployeeId;

}

public void setId(String id) { this.EmployeeId = id;

}

public String getName() { return EmployeeName;

}

public void setName(String name) { this.EmployeeName = name;

}

public String getDepartment() { return EmployeeDepartment;

}

public void setDepartment(String Department) { this.EmployeeDepartment = Department;

}

}

### EmployeeView.java:

public class EmployeeView {

public void printEmployeeDetails (String EmployeeName, String EmployeeId, String EmployeeDepartment){

System.out.println("Employee Details: "); System.out.println("Name: " + EmployeeName); System.out.println("Employee ID: " + EmployeeId);

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

}

}

### EmployeeController.java:

public class EmployeeController {

// declaring the variables model and view private Employee model;

private EmployeeView view;

// constructor to initialize

public EmployeeController(Employee model, EmployeeView view) { this.model = model;

this.view = view;

}

// getter and setter methods

public void setEmployeeName(String name){ model.setName(name);

}

public String getEmployeeName(){ return model.getName();

}

public void setEmployeeId(String id){ model.setId(id);

}

public String getEmployeeId(){ return model.getId();

}

public void setEmployeeDepartment(String Department){ model.setDepartment(Department);

}

public String getEmployeeDepartment(){ return model.getDepartment();

}

// method to update view public void updateView() {

view.printEmployeeDetails(model.getName(), model.getId(), model.getDepartment());

}

}

### MVCMain.java:

public class MVCMain {

public static void main(String[] args) {

// fetching the employee record based on the employee\_id from the database Employee model = retriveEmployeeFromDatabase();

// creating a view to write Employee details on console EmployeeView view = new EmployeeView();

EmployeeController controller = new EmployeeController(model, view); controller.updateView();

//updating the model data controller.setEmployeeName("Nirnay"); System.out.println("\n Employee Details after updating: "); controller.updateView();

}

private static Employee retriveEmployeeFromDatabase(){ Employee Employee = new Employee(); Employee.setName("Anu");

Employee.setId("11"); Employee.setDepartment("Salesforce"); return Employee;

}

}

### Output:

Employee Details:

Name: Anu Employee ID: 11

Employee Department: Salesforce

Employee Details after updating: Employee Details:

Name: Nirnay Employee ID: 11

Employee Department: Salesforce

**Web MVC:**

### LoginCridentals.java:

public class LoginCridentals {

public String userName; public String password;

public void setUserName(String name)

{

this.userName=name;

}

public void setPassword(String pass)

{

this.password=pass;

}

public String getUserName()

{

return userName;

}

public String getPassword()

{

return password;

}

public Boolean authorization()

{

if(userName.equalsIgnoreCase("admin"))

{

if(password.equalsIgnoreCase("admin"))

{

return true;

}

return false;

}

return false;

}

}

### errorLogin.jsp:

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<p>Sorry! userName or password error</p>

<a href="index.html" >index</a>

</body>

</html>

### index.html:

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<form action="Controller" method="post">

<label>Username: </label>

<input type="text" name="uname" required><br>

<label>Username: </label>

<input type="password" name="password" required><br>

<input type="submit" value="Login">

</form>

</body>

</html>

### LoginCridentals.jsp:

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

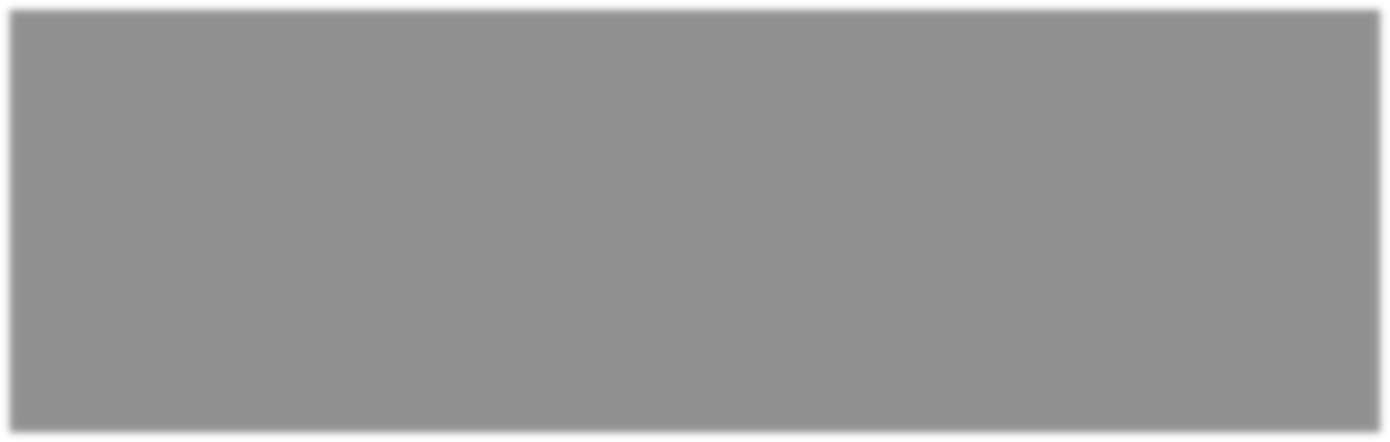
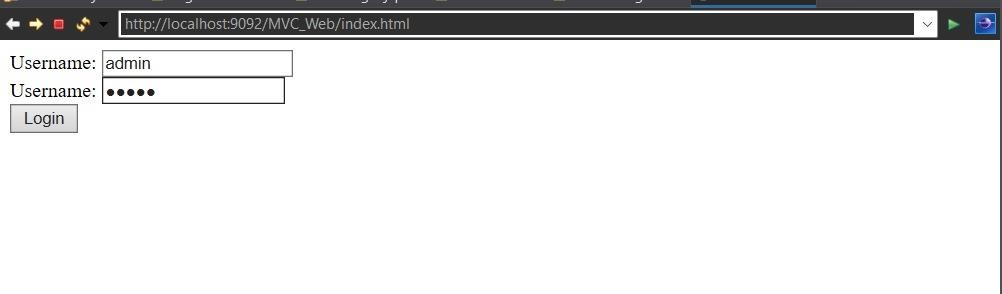
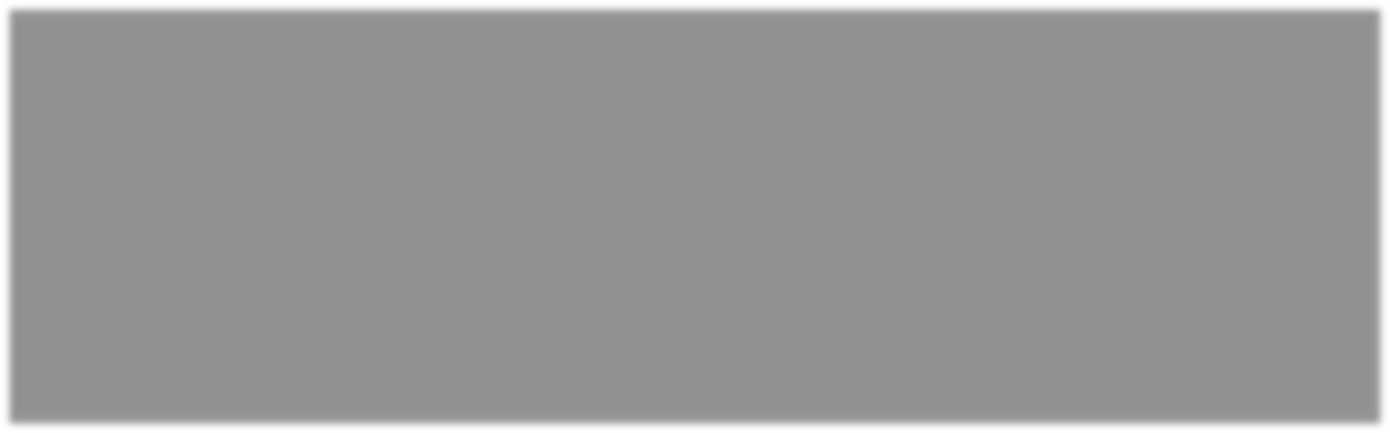
<body>

<p>You are successfully logged in!</p>

</body>

</html>

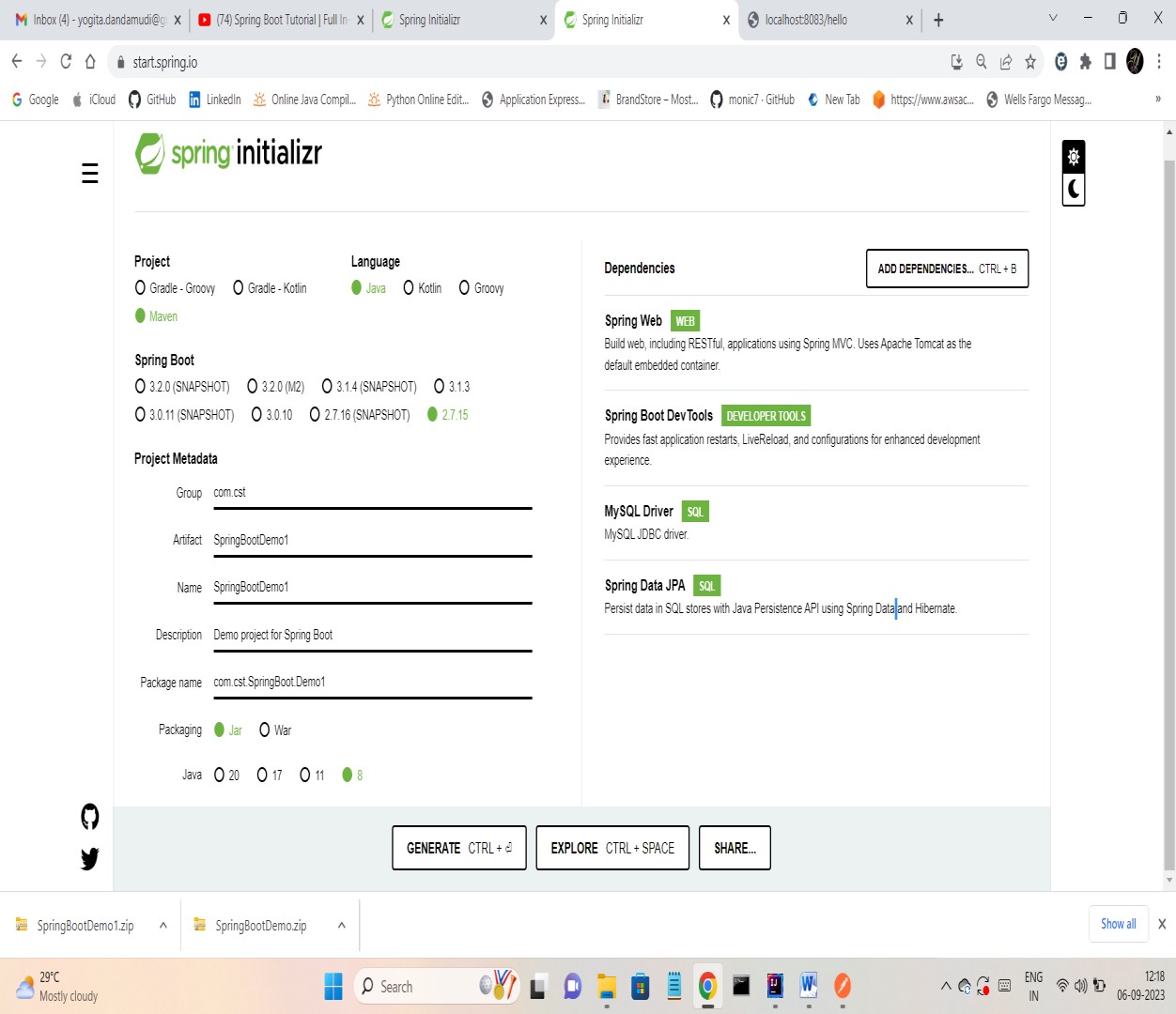
**Output:**



# **Exercise 4:** Create Example programs Using Spring MVC Framework

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

1.



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

server.port=8083 spring.jpa.hibernate.ddl-auto=update

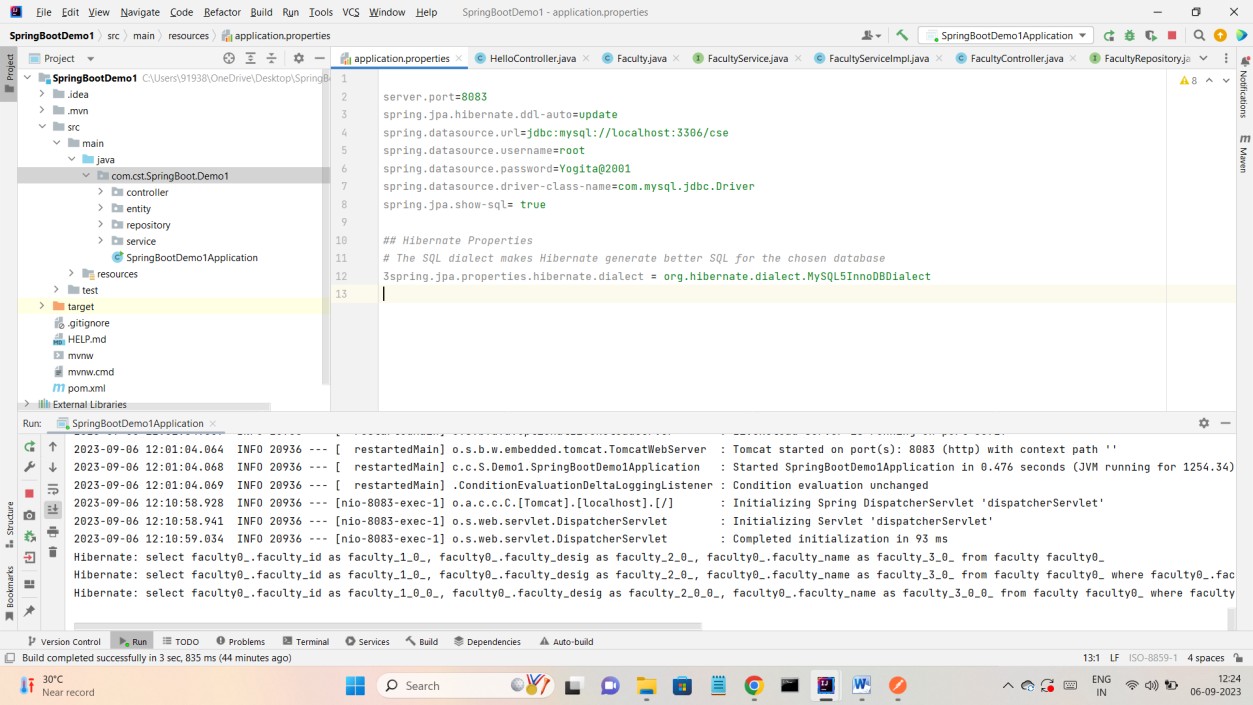
spring.datasource.url=jdbc:mysql://localhost:3306/cse spring.datasource.username=root spring.datasource.password=Yogita@2001 spring.datasource.driver-class-name=com.mysql.jdbc.Driver spring.jpa.show-sql= true

*## Hibernate Properties*

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

3spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect

1. Create 4 packages controller, entity, repository, service



1. Create a class Faculty under entity package Add the following code in Faculty class

package com.cst.SpringBoot.Demo1.entity; import javax.persistence.\*;

@Entity @Table

public class Faculty {

@Id

@GeneratedValue(strategy = GenerationType.*AUTO*) private Long facultyId;

private String facultyName; private String facultyDesig; public Faculty() {

}

public Long getFacultyId() { return facultyId;

}

public void setFacultyId(Long facultyId) { this.facultyId = facultyId;

}

public String getFacultyName() { return facultyName;

}

public void setFacultyName(String facultyName) { this.facultyName = facultyName;

}

public String getFacultyDesig() { return facultyDesig;

}

public void setFacultyDesig(String facultyDesig) { this.facultyDesig = facultyDesig;

}

@Override

public String toString() { return "Faculty{" +

"facultyId=" + facultyId +

", facultyName='" + facultyName + '\'' + ", facultyDesig='" + facultyDesig + '\'' + '}';

}

}

1. Create an Interface FacultyRepository under the package repository Add the following code

package com.cst.SpringBoot.Demo1.repository;

import com.cst.SpringBoot.Demo1.entity.Faculty;

import org.springframework.data.jpa.repository.JpaRepository; import java.util.List;

public interface FacultyRepository extends JpaRepository<Faculty,Long> {

List<Faculty> findByFacultyName(String name);

}

1. Create an Interface under FacultyService under the package service Add the following code

package com.cst.SpringBoot.Demo1.service; import com.cst.SpringBoot.Demo1.entity.Faculty; import java.util.List;

public interface FacultyService { Faculty save(Faculty faculty);

List<Faculty> fetchAll();

Faculty fetchById(Long facultyId);

void deleteByFacultyId(Long facultyId);

List<Faculty> fetchByFacultyName(String name);

}

1. Create a package impl under service
2. Create a Class FacultyServiceImpl under impl package Add the Following code

package com.cst.SpringBoot.Demo1.service.impl;

import com.cst.SpringBoot.Demo1.entity.Faculty;

import com.cst.SpringBoot.Demo1.repository.FacultyRepository; import com.cst.SpringBoot.Demo1.service.FacultyService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service;

import java.util.List; @Service

public class FacultyServiceImpl implements FacultyService { @Autowired

private FacultyRepository facultyRepository;

@Override

public Faculty save(Faculty faculty) { return facultyRepository.save(faculty);

}

@Override

public List<Faculty> fetchAll() { return facultyRepository.findAll();

}

@Override

public Faculty fetchById(Long facultyId) {

return facultyRepository.findById(facultyId).get();

}

@Override

public void deleteByFacultyId(Long facultyId) { facultyRepository.deleteById(facultyId);

}

@Override

public List<Faculty> fetchByFacultyName(String name) { return facultyRepository.findByFacultyName(name);

}

}

1. Create a class FacultyController under controller package Add the following code

package com.cst.SpringBoot.Demo1.controller;

import com.cst.SpringBoot.Demo1.entity.Faculty;

import com.cst.SpringBoot.Demo1.service.FacultyService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.web.bind.annotation.\*;

import java.util.List; @RestController

public class FacultyController { @Autowired

private FacultyService facultyService;

@PostMapping("/save")

public Faculty save(@RequestBody Faculty faculty)

{

return facultyService.save(faculty);

}

@GetMapping("/fetchAll") public List<Faculty> fetchAll()

{

return facultyService.fetchAll();

}

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

public Faculty fetchById(@PathVariable("id") Long facultyId)

{

return facultyService.fetchById(facultyId);

}

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

public String deleteByFacultyId(@PathVariable("id") Long facultyId)

{

facultyService.deleteByFacultyId(facultyId); return "Success";

}

@PostMapping("/fetchByFacultyName")

public List<Faculty> fetchByFacultyName(@RequestBody String name){ return facultyService.fetchByFacultyName(name);

}

}

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

{

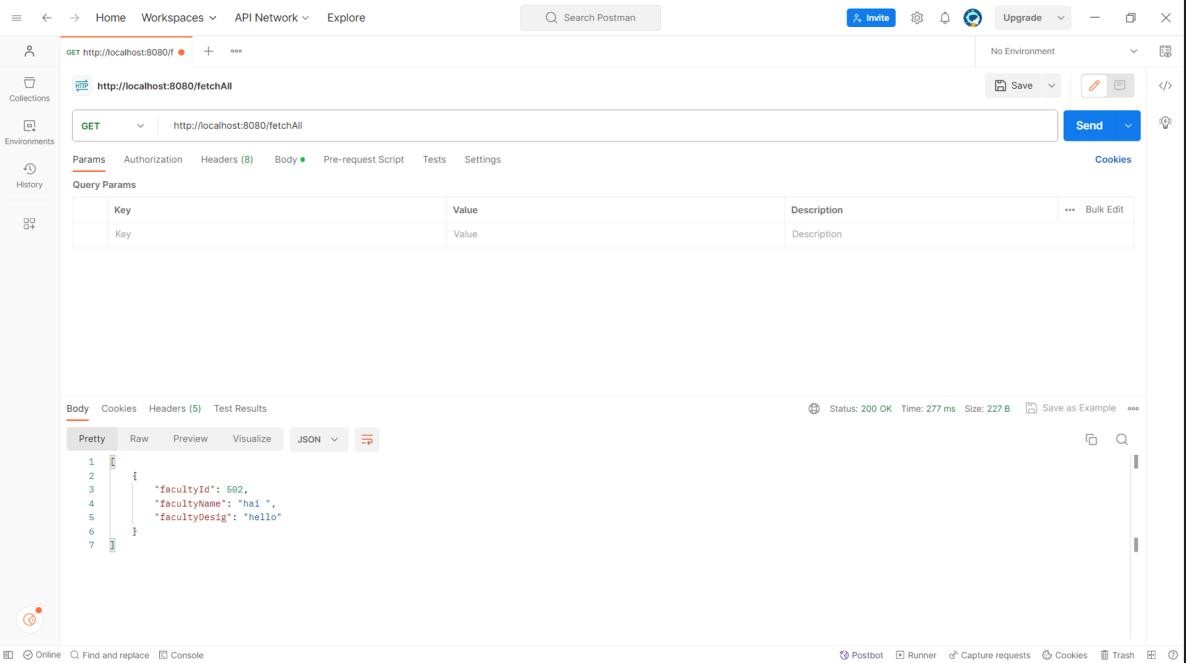
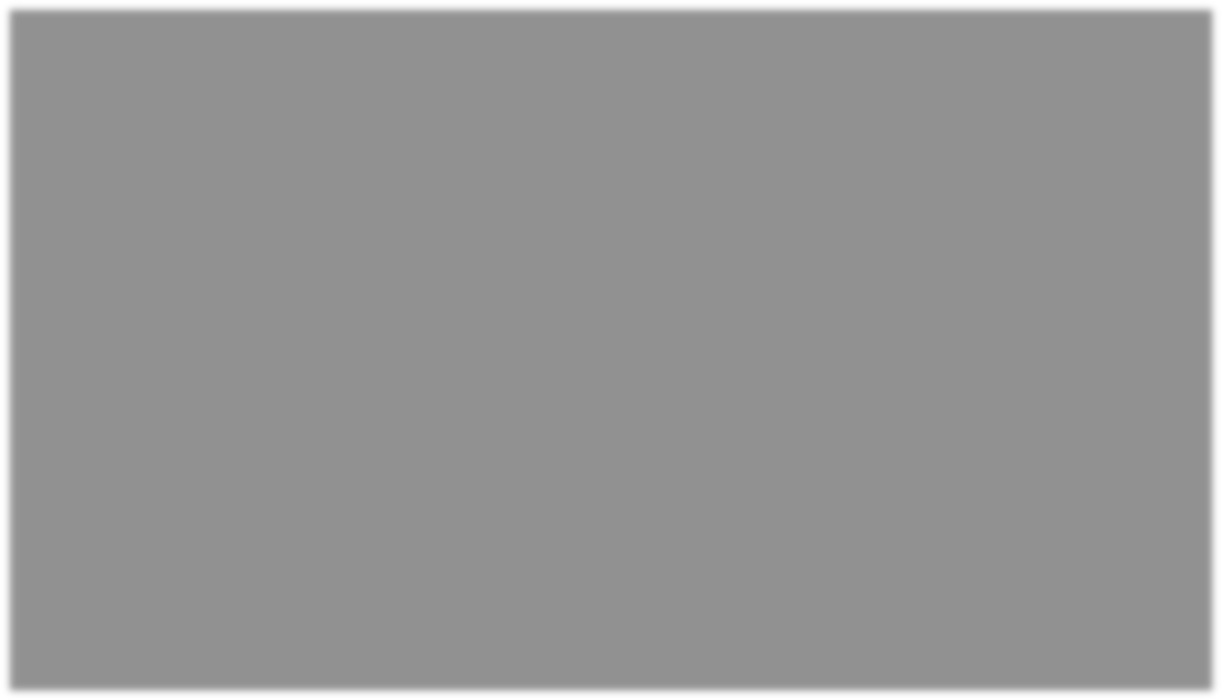
"facultyId" : "502", "facultyName" : "hai", "facultyDesig" : "hello"

}

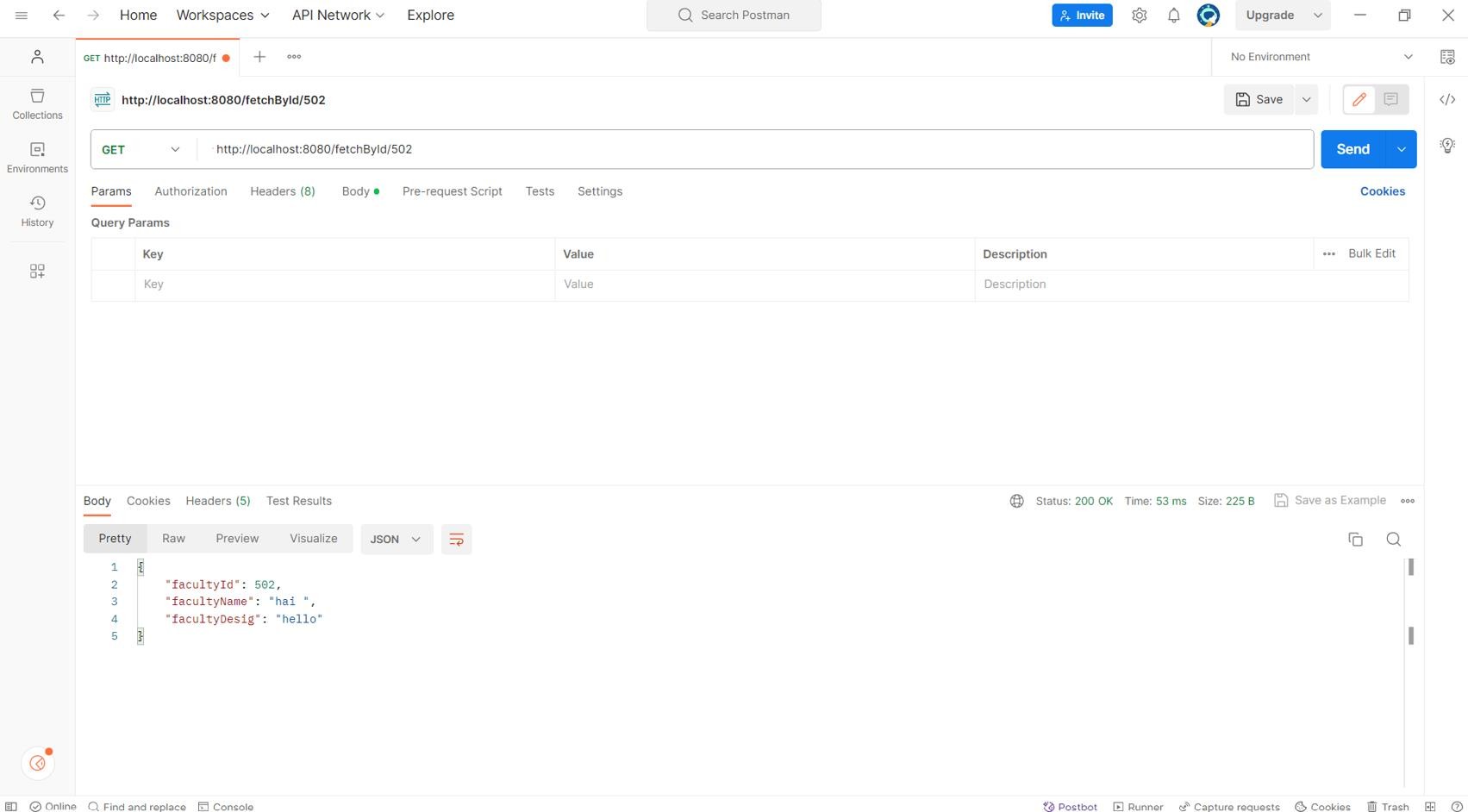
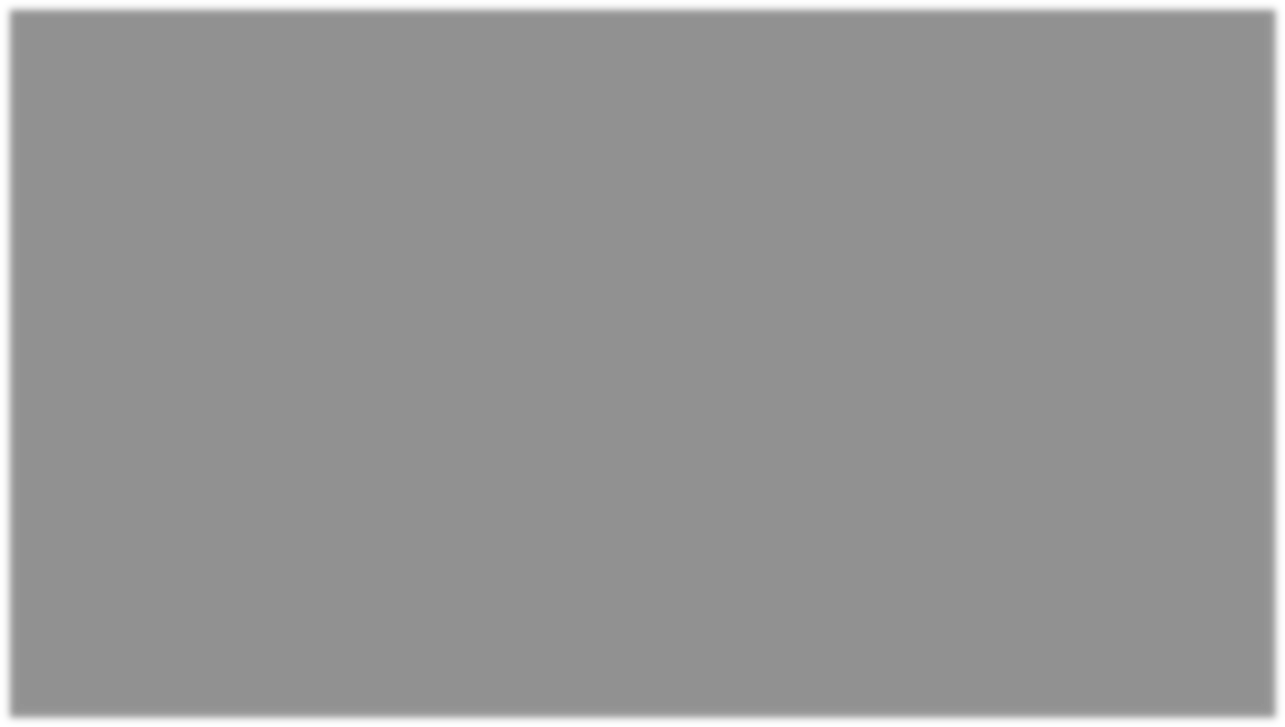
1. Select the method GET and type the URL http://localhost:8083/fetchAll
2. Test remaining API’s

**Output:**

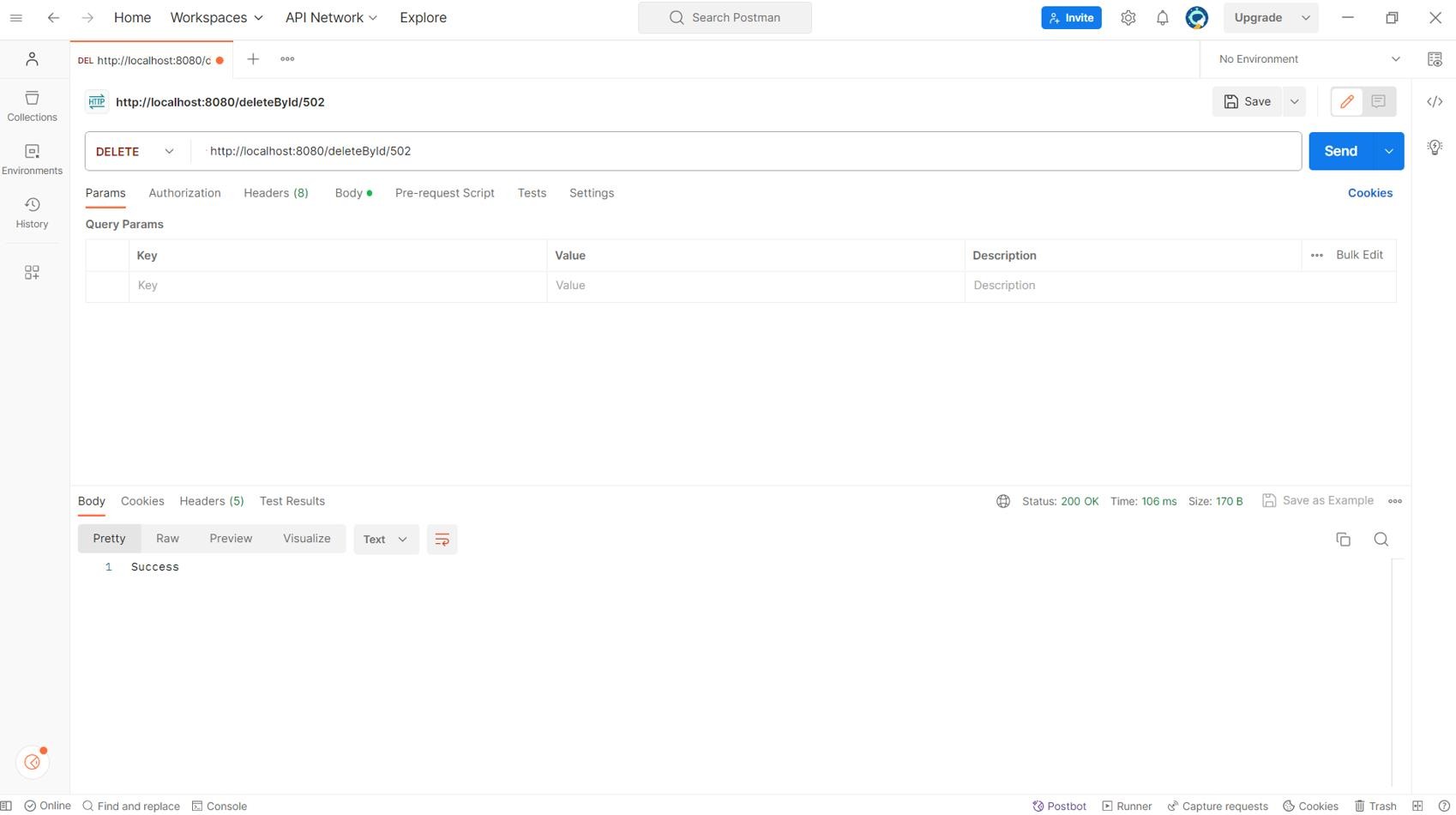
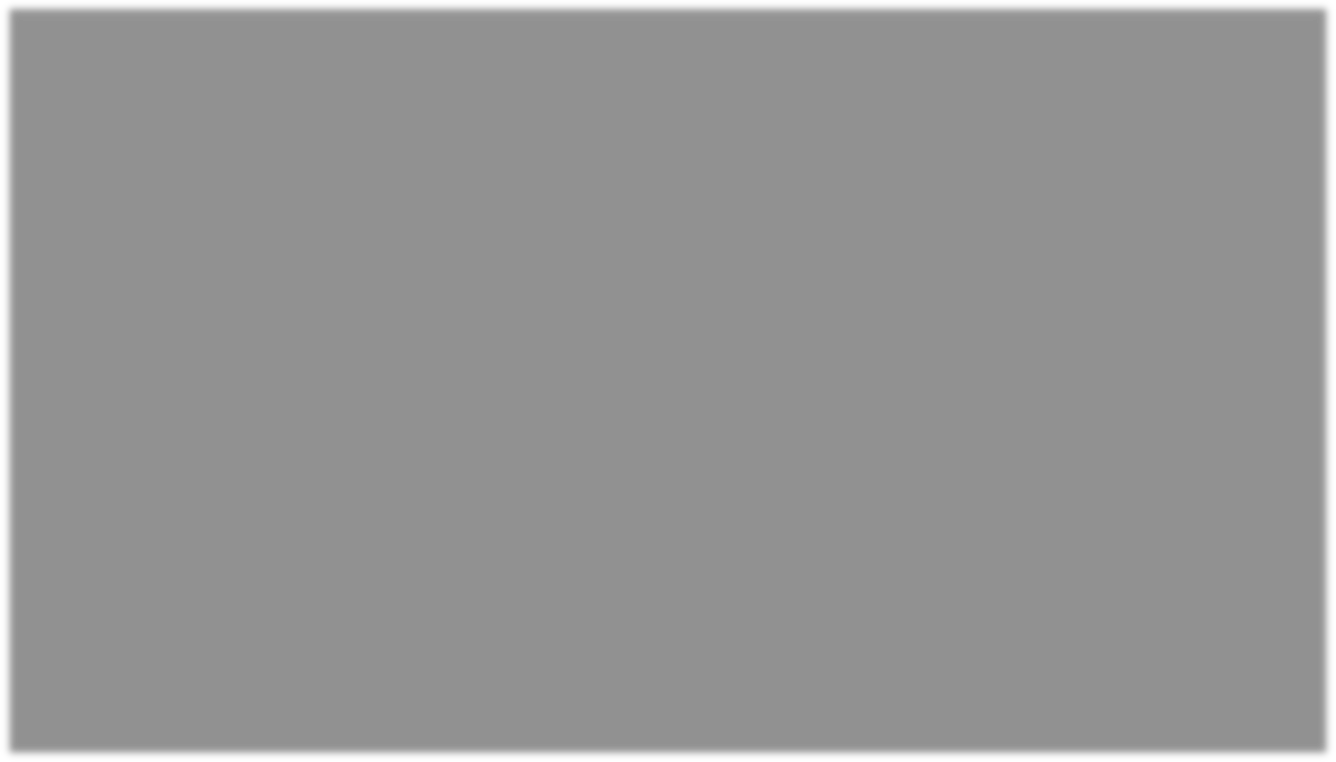
1. fetchAll:



1. fetchById:



1. deleteById:



**Exercise 5:** Create Example programs Using Hibernate

Steps to create Hibernate Maven Project:

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.cst 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.cst.HibDemo, click on App.java
7. Run App.java(to check whether Maven Project is running or not)
8. Now Right click on com.cst.HibDemo, click on new, click on class And name the class as Students
9. Paste the code in Students.java

package com.cst.HibDemo; import org.hibernate.Session;

import org.hibernate.SessionFactory; import org.hibernate.Transaction; import org.hibernate.cfg.Configuration; import org.hibernate.Criteria;

import org.hibernate.criterion.Restrictions;

import java.util.List; import java.util.Scanner;

public class App {

public static void main(String[] args) { System.out.println("Hello World!");

Configuration con = new Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Student.class);

SessionFactory sf = con.buildSessionFactory(); Session se = sf.openSession();

Scanner scanner = new Scanner(System.in); while (true) {

System.out.println("\nChoose an operation: "); System.out.println("1 - Insert new student"); System.out.println("2 - Update existing student"); System.out.println("3 - Delete existing student"); System.out.println("4 - Fetch all student details"); System.out.println("5 - Exit");

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

switch (choice) { case 1:

// Inserting

System.out.println("Enter student roll number: "); int rollNumber = scanner.nextInt(); scanner.nextLine();

System.out.println("Enter student name: "); String name = scanner.nextLine();

System.out.println("Enter student branch: "); String branch = scanner.nextLine();

Transaction tx = se.beginTransaction(); Student newStudent = new Student(); newStudent.setRno(rollNumber); newStudent.setName(name); newStudent.setBrn(branch); se.save(newStudent);

tx.commit();

System.out.println("New student details inserted successfully."); break;

case 2:

// Updating (update existing student)

System.out.println("Enter the Student Roll Number to update: "); int rollNumberToUpdate = scanner.nextInt(); scanner.nextLine();

Student existingStudent = (Student) se.get(Student.class, rollNumberToUpdate); if (existingStudent != null) {

System.out.println("Existing Name: " + existingStudent.getName()); System.out.println("Existing Branch: " + existingStudent.getBrn());

System.out.println("Enter new Name:"); String newNameUpdate = scanner.nextLine();

System.out.println("Enter new Branch: "); String newBranchUpdate = scanner.nextLine();

if (!newNameUpdate.isEmpty()) { existingStudent.setName(newNameUpdate);

}

if (!newBranchUpdate.isEmpty()) { existingStudent.setBrn(newBranchUpdate);

}

Transaction updateTransaction = se.beginTransaction(); se.update(existingStudent); updateTransaction.commit();

System.out.println("Student details updated successfully.");

} else {

System.out.println("No records found for the given Roll Number.");

}

break; case 3:

// Delete (delete existing student)

System.out.println("Enter the Student Roll Number to delete: "); int rollNumberToDelete = scanner.nextInt(); scanner.nextLine();

Student studentToDelete = (Student) se.get(Student.class, rollNumberToDelete); if (studentToDelete != null) {

Transaction deleteTransaction = se.beginTransaction();

se.delete(studentToDelete); deleteTransaction.commit();

System.out.println("Student details deleted successfully.");

} else {

System.out.println("No records found for the given Roll Number.");

}

break; case 4:

// Fetch all student details

Criteria criteria = se.createCriteria(Student.class); List<Student> allStudents = criteria.list();

if (!allStudents.isEmpty()) { System.out.println("\nAll Students in the table:"); for (Student stu : allStudents) {

System.out.println("Roll Number: " + stu.getRno()); System.out.println("Name: " + stu.getName()); System.out.println("Branch: " + stu.getBrn()); System.out.println(" ");

}

} else {

System.out.println("No records found for students.");

}

break; case 5:

System.out.println("Exiting the program..."); se.close();

sf.close(); System.exit(0);

default:

System.out.println("Invalid choice"); break;

}

}

}

}

11.Now open the pom.xml file and add the following code

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>" xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> [http://maven.apache.org/xsd/maven-4.0.0.xsd"](http://maven.apache.org/xsd/maven-4.0.0.xsd)>

<modelVersion>4.0.0</modelVersion>

<groupId>com.cst</groupId>

<artifactId>HibDemo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>HibDemo</name>

<ur[l>http:](http://maven.apache.org/)//[maven.apache.org](http://maven.apache.org/)</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.1.6.Final</version>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

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

<version>8.0.33</version>

</dependency>

</dependencies>

</project>

Note: These dependencies we can get it from Maven Repository

Adding Hibernate Plugin to the Eclipse:

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

4.28.0 final

1. click on install
2. Uncheck everything else and check Hibernate and install it (while installing it asks you for Trust, check the trust)
3. Next we have to specify hibernate.cfg.xml file

(In this we have to specify database and credentials)

1. Right click on your project, click on new and click on other
2. Search for hibernate
3. Select Hibernate Configuration(cfg.xml), click on next
4. In Database Dialect select MySQL
5. Select Driver class
6. Select Connection URL 11.Enter Username and Password
7. Click on finish
8. Add the following tag in the hibernate.cfg.xml

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

1. Save the file
2. Run your Application and check for the Table students in your database.

## Output:

Menu 1.Add Student

2.Delete Student 3.Update student details 4.PrintAll details

5.Exit 1

Enter Id 1

Enter Name cse

Enter Department cse

Enter Total 100

New student details inserted successfully.

Menu 1.Add Student 2.Delete Student

3.Update student details 4.PrintAll details

5.Exit 4

All Students in the table:

Roll Number: 1

Name: cse Branch: cse Branch: 100

Menu 1.Add Student 2.Delete Student

3.Update student details 4.PrintAll details

5.Exit 1

Enter Id 2

Enter Name vasavi

Enter Department CAI

Enter Total 98

New student details inserted successfully.

Menu 1.Add Student

2.Delete Student 3.Update student details 4.PrintAll details

5.Exit 3

Enter Id to Update the Student 2

Enter New Name vasavi

Enter New Department CST

Enter New Total 98

Student details updated successfully.

Menu 1.Add Student 2.Delete Student

3.Update student details 4.PrintAll details

5.Exit 4

All Students in the table Roll Number: 1

Name: cse Branch: cse Branch: 100

Roll Number: 2 Name: vasavi Branch: CST Branch: 98

Menu 1.Add Student 2.Delete Student

3.Update student details 4.PrintAll details

5.Exit 2

Enter Id to delete 2

Student Details Deleted successfully.

Menu 1.Add Student 2.Delete Student

3.Update student details 4.PrintAll details

5.Exit 4

All Students in the table:

Roll Number: 1

Name: cse Branch: cse Branch: 100