Section-2: Web Technologies Lab

Structure **Page No.**

2.0 INTRODUCTION

The section-2 of this lab manual is for Web Technologies Lab, which is based on MCS-220 Web Technologies course. At the end of this section, you will get list of lab sessions. These ten lab sessions consist of several laboratory/ programming exercises which you have to implement/write code. For writing a program, you need to use IDE (Integrated Development Environment) for better and smooth programming experiences. An IDE is an application that provides various application development facilities for Programming Languages. It also provides many useful GUI tools which

are useful for a developer's needs during application development. Some of the most common features of IDEs are as follows:

- **Code Editor:** It is designed for writing and editing the source code.
- **Compiler:** It is used to transform source code written in a programming language (human readable/writable language) into a form that is executable by a computer.
- **Debugger:** It is used during testing the application, which helps to debug application programs during the development.
- **Builder:** This automates common developer tasks to build the application to run/execute the application.
- **Browsers:**
    - **Class Browser** is used to examine and reference the properties of an object-oriented class hierarchy.
    - **Object Browser** is used to examine the objects instantiated in a running application program.
- **Class hierarchy Diagram:** It allows the programmer to visualize the project structure of object-oriented programming code.

Manual development was the trend before the existence of IDEs; developers/programmers used to write their programs manually in text editors ***t*** and save the file with programming extension (like .java, .jsp, .vb, .c, .cc, etc.), then they compile these file through a compiler which is installed and setup in the environment. If program is compiled successfully, then it will be built and run on the applications server. In case of the error messages shown during the compilation, they have to go back to the text editor to revise the source code and recompile. IDE development has made the program writing and compilation process simpler.

Some popular Java IDEs includes Eclipse, IntelliJ IDEA, and NetBeans. In this section, we will focus on these popular open-source IDEs. There are many other IDEs also which support Java Programming Language. In this section of the lab manual, we will talk about the use of the development environment of NetBeans, Eclipse and IntelliJ Idea.

2.1OBJECTIVE

The objectives of this section are:

- To install different IDEs for Java/J2EE programming,
- Develop simple applications using IDEs, and
- Use the given list of exercises for your assigned lab sessions.

2.2 SETUP DEVELOPMENT ENVIRONMENT

For the Setup development environment, we need to set up a compiler, builder, editor (mainly JDK and IDE). Before installation of any Idea, Compiler and builder are required. That is why we need to install the latest version of JDK in our system. The best thing about IDE is that its intelligence provides code completion hints during coding in IDE. It also provides easy code, file and folder navigation. In this unit Windows 10 based installation and environment setup will be shown. Official documents for Linux and Mac-based environment setup are provided in the reference section.

Step 1: Installation of Java Development Kit (JDK)

- First, check the java version if already installed by executing the command "java –version" in the terminal or command prompt as shown in Figure-2.1.

```
C:\Users\Vinay>java --version
java 16.0.1 2021-04-20
Java(TM) SE Runtime Environment (build 16.0.1+9-24)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.1+9-24, mixed mode, sharing)

C:\Users\Vinay>
```

Figure 2.1: Check Java Version

- If Java is not installed in your system, then, first of all, download the latest version of JDK from its official website (https://www.oracle.com/in/java/technologies/javase-downloads.html). For development practice, you can choose any latest version of JDK, but it is strongly recommended to choose the LTS (Long Term Support) version only for the production environment. Choose the JDK package as per your system/platform.
- Install the JDK in your system and recheck the installed version.

Step 2: System Environment variable setup for Java in Windows 10

- For quick open Click on start menu and type System Environment, Control panel Option for Edit the System Environment Variables will show, click on it to open as shown in figure- 2.2
- System Property wizard will open as shown in figure- 2.3.
- Click on Environment Variables as marked in figure- 3. The Wizard for Environment Variables will be open as shown in figure – 5.
- Click on the New button if the classpath for JAVA_HOME is not set or click on the Edit Button to update with your latest JDK version as shown in figure- 2.4.
- Click Ok in all Wizard opens to save the JAVA_HOME variable as shown in figure – 2.5.
- Check the java verion as shown in figure- 1.1.

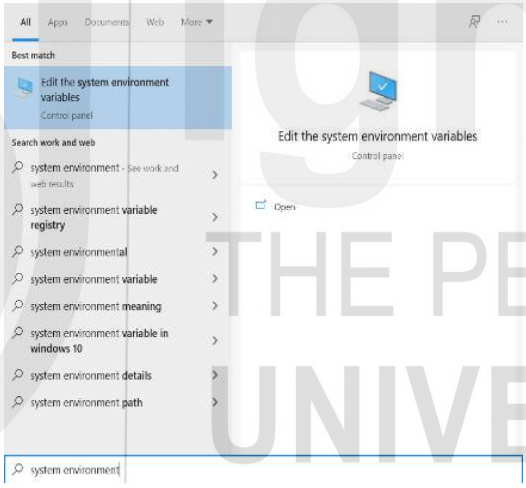


Figure 2.2: Edit the System Environment Variables

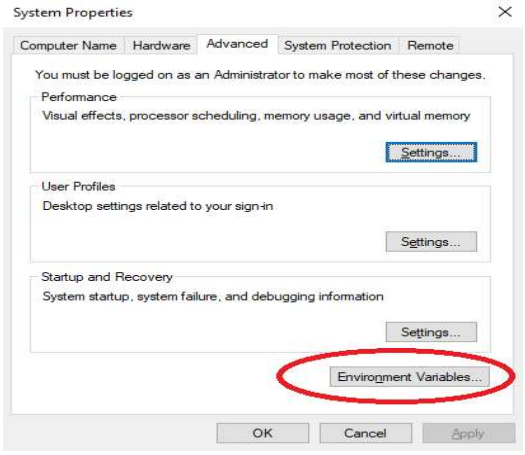


Figure 2.3: System Properties winzard

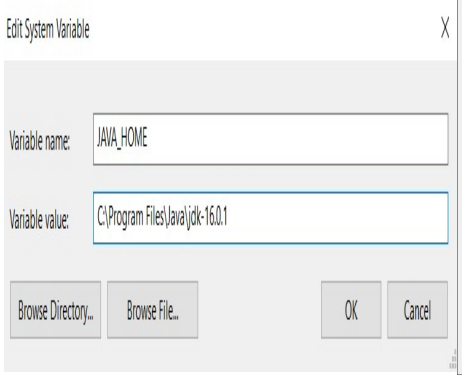


Figure 2.4: System Variable creation or editing

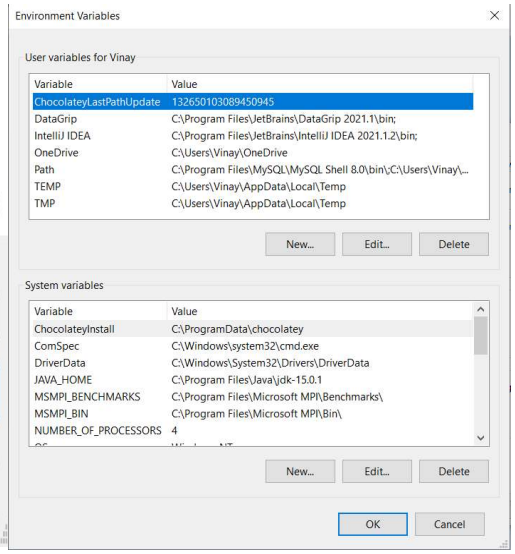


Figure 2.5: Environment Variable

Setup JAVA_HOME, CLASSPATH and PATH for Linux

- Execute following command in terminal to edit /etc/profile file
  vi /etc/profile
- Add following lines in last line in profile file.
  export JAVA_HOME=/usr/jdk16.0.1
  export CLASSPATH=$CLASSPATH:/home/vinay/LOG4J_HOME/log4j-2.2.16.jar
  export PATH=$PATH:/usr/ jdk16.0.1/bin
- Execute the command "/etc/profile" to activate these settings.
- Execute the following command in the terminal to verify whether the settings are correct or not.
  $ java -version

Now the class path for JDK is set successfully now you are ready to install IDE (NetBeans/Eclipse/IntelliJ Idea)

Step 3: Install IDEs: In this section, we will learn how to configure Netbeans and Eclipse IDE.

INSTALL APACHE NETBEANS IN WINDOWS 10

NetBeans is an open-source Java IDE which is one of the biggest and most popular IDE. It is supported for a variety of operating systems like Linux, Windows, macOS, Solaris, etc., along with feature-limited OS-independent versions.

Using NetBeans, it is very easy to create custom software applications. The NetBeans highlights Java code syntactically as well as semantically. Also, there are many tools in NetBeans that help in writing bug-free code. The NetBeans is primarily a Java IDE, but it also has extensions for working in other programming languages, including C, C++, PHP, HTML5, JavaScript, etc.

To download Apache Netbeans Idea, visit the official site of Apache Netbeans; URL is mentioned below:

https://netbeans.apache.org/download/index.html

Download the package as per your platform(Linux/Windows/MacOS). In this section we are using Windows 10.

Locate the Apache-NetBeans-12.4-bin-windows-x64.exe setup file in your system, where it is saved after download and run as administrator. The User access control wizard will be displayed. Click on Yes. Setup will configure the installer as shown in figure – 2.6.

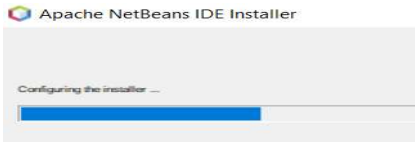


Figure 2.6: Setup Progress bar

The Apache NetBeans IDE installer Wizard will come up. If you want to customize the installation, click on the Customize button as shown in figure- 2.7 then click on Next. After clicking on Next in figure- 2.7 the Licence Agreement wizard will be displayed. Tick on the checkbox for acceptance of terms and conditions, then click on Next for process further, as shown in figure- 2.9. If you want to change the default location of Netbeans-12.4 folder, click on the 1st Browse button as shown in figure – 2.9.
If you want to change the default JDK version, select the desired JDK version click on the 2nd Browse... button as shown in figure – 2.9.

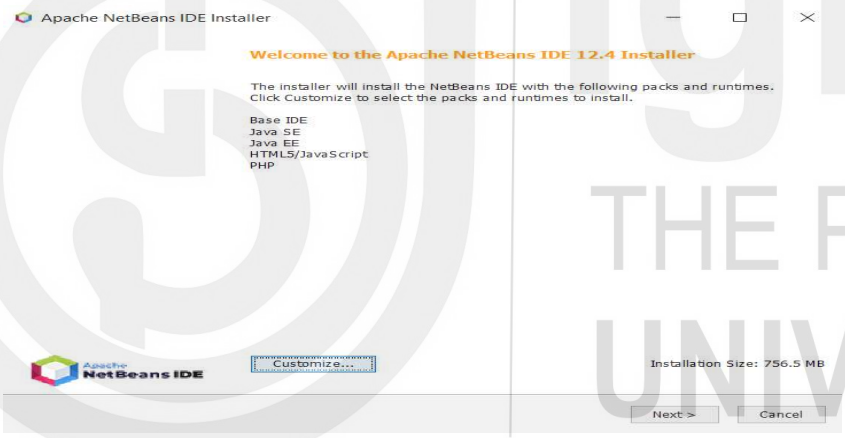


Figure 2.7: Installer Wizard

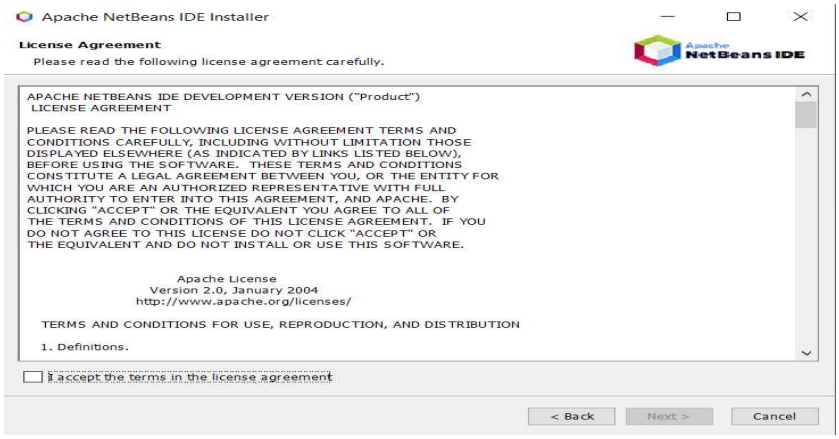


Figure 2.8: License Agrement

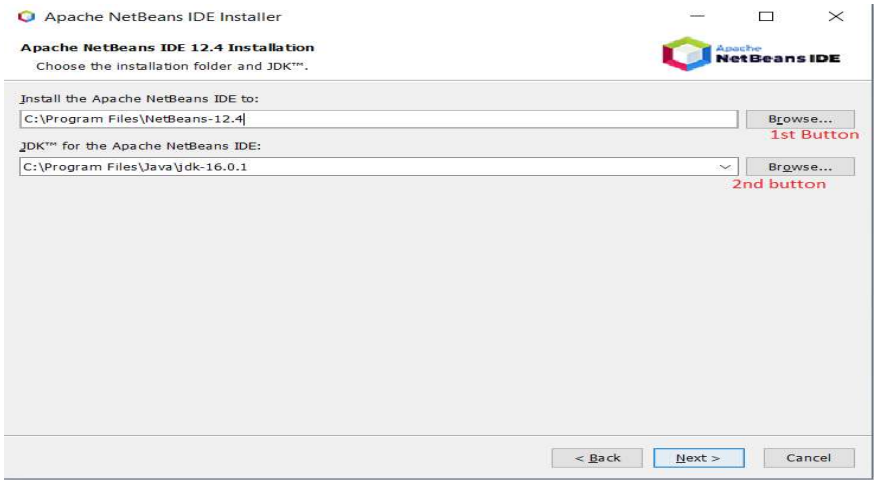


Figure 2.9: Wizard to choose the installation forlder and JDK

Then click on Next > Button (Installation Summary will display as shown in figure – 2.10)

Figure 2.10: Summary of config

Finally, click on Install to install the software; the installation process will begin; wait till the process finishes. Now, click on the Finish Button to close the wizard. Now you are ready to use the NetBeans IDE. Go to the start program, navigate to Apache NetBeans, click on Apache NetBeans IDE 12.4 link to open the IDE, or double click on the Apache NetBeans IDE 12.4 icon created in Desktop as well to open the IDE.

Welcome to the First look of Apache NetBeans IDE 12.4 as shown in figure – 2.11. You can maximize the window size for a full view. Close the startup page.

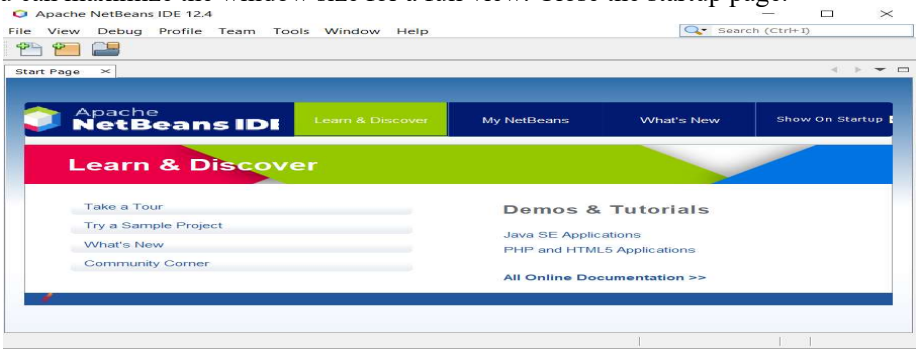


Figure 2.11: First look of Apache Netbeans IDE

Now you are ready to create Java Projects in Apache NetBeans 12.4 IDE.

To Create New Project, navigate as follow:

File→ New Project

The menu bar is self-explanatory now you can navigate the Menu bar and explore the available options provided by NetBeans IDE.

INSTALL ECLIPSE IDE IN WINDOWS 10

Eclipse is an Open Source IDE used for developing applications in Java and other Programming languages using plugins which make it one of the most popular IDE worldwide for Developers. Most of the source code of eclipse is written in Java. Some basic features of Eclipse are:

- PDE (Plugin Development Environment) is available in Eclipse for Java programmers to create specific functionalities in their applications.
- Eclipse provides powerful tools for the various processes in application development such as charting, modeling, reporting, testing, etc. so that Java developers can develop the applications quickly.
- Eclipse can be used on different platforms like Linux, macOS, Solaris and Windows.

Download the latest version of Eclipse IDE software from its official website.

URL: https://www.eclipse.org/downloads/

Click to Download x86_64(it supports both 32bit and 64 bit architecture) to download the setup file as per your platform(OS) requirements.

Similar to Apache NetBeans installation, Locate the setup file in your system where it has been saved after download. Run the setup file "eclipse-inst-jre-win64.exe(for Windows)" as administrator. The Eclipse Installer wizard will be displayed.

Figure 2.12: Eclipse Installer wizard

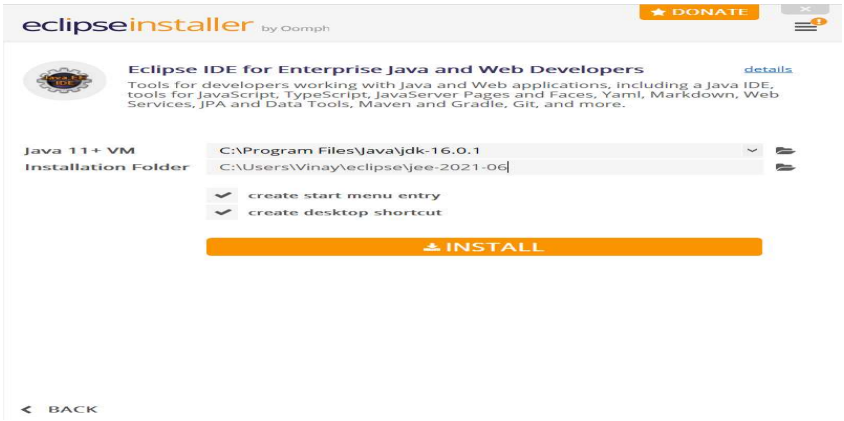


Figure 2.13: Folder Location setting for JVM and IDE

Eclipse IDE provides various types of Development environments for various programming languages, as shown in figure- 2.12. Here we are going to install Eclipse IDE for Enterprises Java and Web Development environment, so click on the link (Eclipse IDE for Enterprises Java and Web Development) highlighted in figure- 12. Next, wizard will come as shown in figure- 2.13, in which we have selected JDK location and installation folder for the eclipse Idea.

If you want to change the default JDK version, select the desired JDK version click on the browse folder icon as shown in figure- 2.13.
If you want to change the default location of the eclipse Installation folder, click on the browse folder icon and choose your location where you want to install the eclipse Idea as shown in figure- 2.13.
Here we are keeping all location and JDK versions as default. Now click the Install Button to start the Installation of the Eclipse IDE.

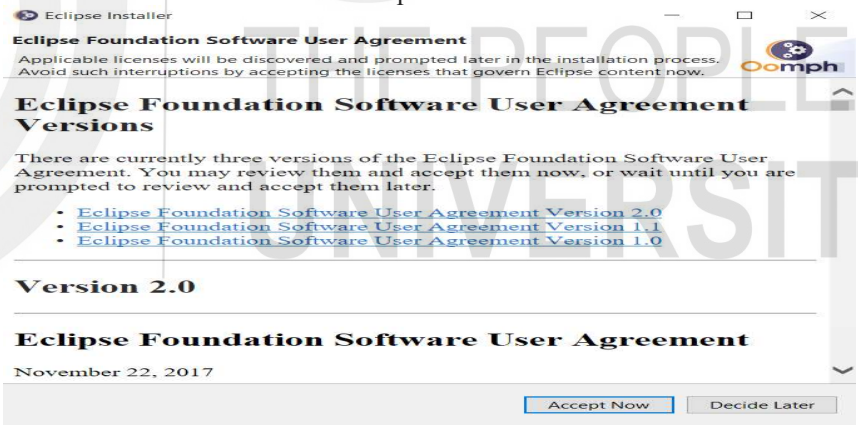


Figure 14: User Agreement

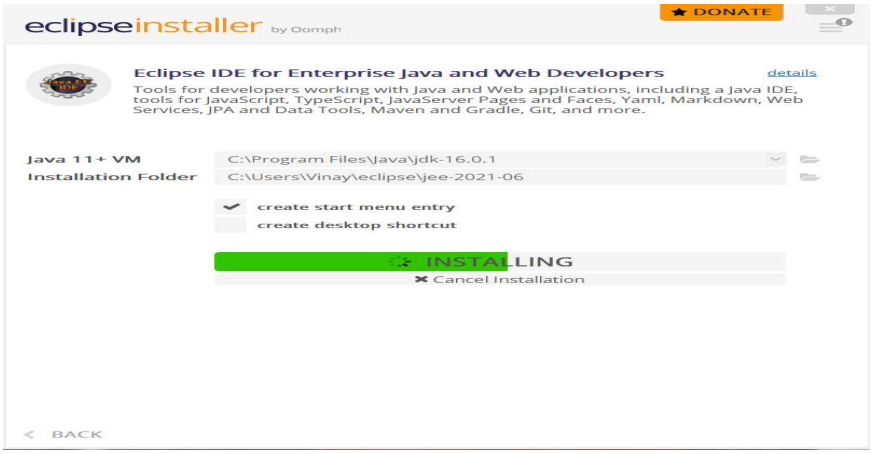


Figure 2.15: Installing in Process

Figure 2.16: Installation complete

Next, it will ask to Accept the Eclipse Foundation Software User Agreement, click on Accept Now to agree to the Agreement to process further, as shown in figure – 2.14. The installation will start if everything is setup well. During the installation process, it will download files from its Repository, so Internet is mandatory during the Eclipse installation. Installation time depends on your Internet and computer speed. Once Installation has been completed, the final screen will display as shown in figure- 16. Now Click on the LAUNCH button to Launch the IDE.

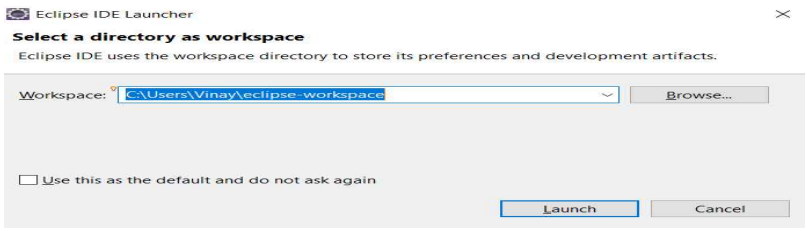


Figure 2.17: Workspace Setup

Before the Open Eclipse IDE, it will ask you to set Workspace location in which Eclipse will configure it's setting and metadata configration. You can set any folder/directory as a workspace, in our case, we are keeping it as default. as shown in figure – 2.17.

After selecting the directory, click on Launch to Eclipse IDE will open as shown in figure – 2.18.

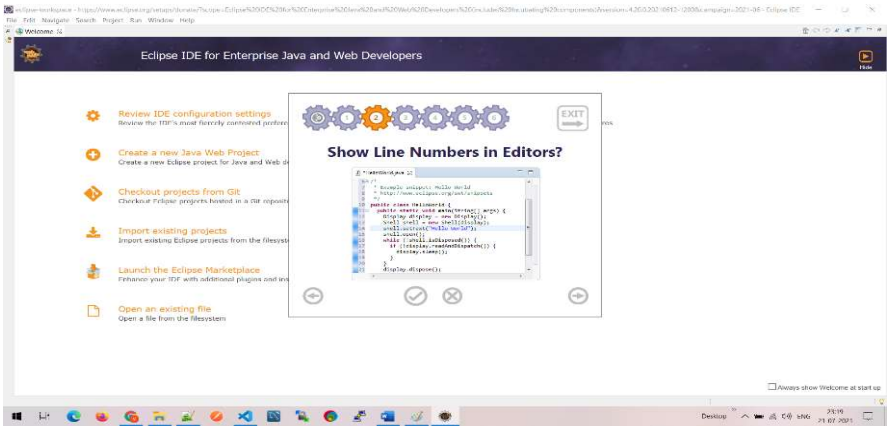


Figure 2.18: First GUI of Eclipse

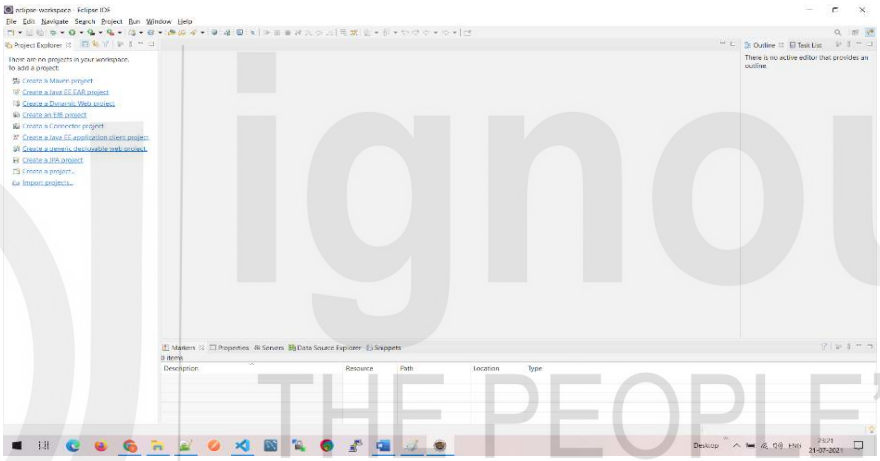


Figure 2.19: Default Workbench of Eclipse IDE

Close the Welcome window and Hints after reading. Default User Interface of Eclipse IDE shown in figure – 2.19:

Now you are ready to create Projects using Eclipse IDE.

2.3 PROJECT CREATION USING IDE

In this section, we will try to create multiple projects either in Apache NetBeans or Eclipse. Before you start with the practical sessions, let us see how to create a java project in Apache NetBeans and Eclipse first.

Create Project in Apache NetBeans:

Step 1: Open the Apache NetBeans 12.4 IDE.
Step 2: New Project: Click on ***File Menu → New Project*** (Wizard shown in figure - 20 will display)
Step 3: Finding Features(One time job)
This option asks us to activate the J2EE features in IDE once after fresh installing of Apache Netbean IDE as shown in figure – 2.20. Click on download and activate/enable the required plugin/features for Java Web Application development.

Once the process is finished, the Wizard for New Web application will be displayed as shown in figure – 2.21. Skip step 3 if already Plugin activated.

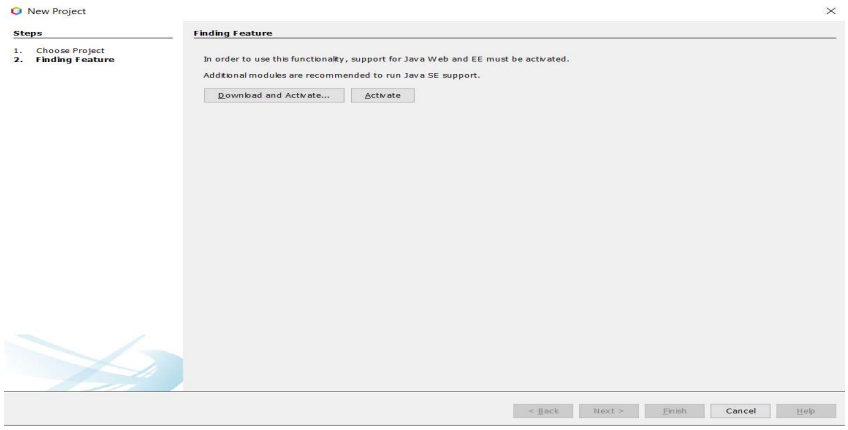


Figure 2.20: Finding Feature(One time if J2EE feature not activated)

Step 4: Click on ***Java with Ant → Java Web*** in Categories and select ***Web Application*** in Projects section then click on Next button as shown in figure -2. 21.

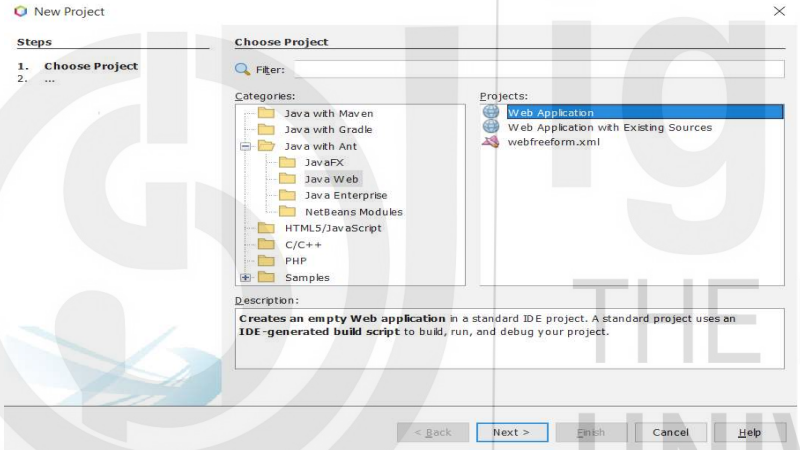


Figure 2.21: New Project Wizard

Step 5: Name and Location or Project

In this wizard we have to enter ***Project Name***, ***Project location*** where project source code to be stored/located, ***Library folder;*** if external dedicated libraries location needs to be added in the Project. In this case Only Project name is changed from the default value, the rest fields are as default, as shown in figire- 2.22 . Then click on Next to procced.

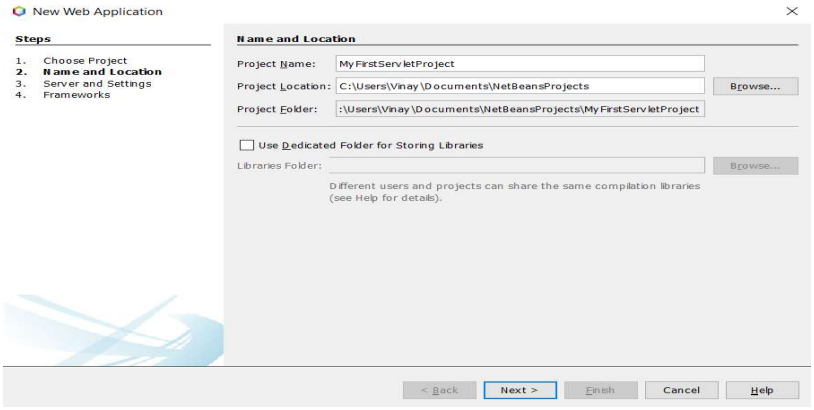


Figure2. 22: Project Name & Location Selection

Step 6: Server and Setting

The Server and Setting wizard will come as shown in figure- 2.23. If the server is configured you may skip Steps 7 to Step 8 and follow Step 9, if Server is not configured, then you have to configure it first. To do that Click on Add Button to add the web server as shown in figure – 2.23.

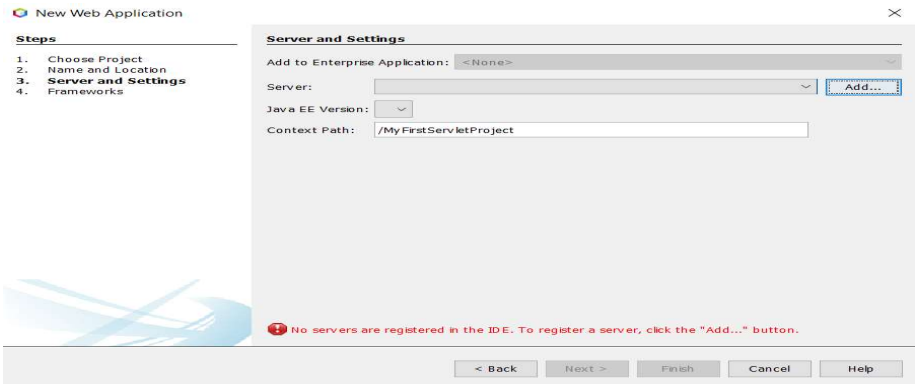


Figure 2.23: Application Server Setting Wizard

Step 7: Choose Server

After clicking on Add Button in figure- 2.23, this Wizard will appear; select the appropriate web server(Apache Tomcat or TomEE) as shown in figure - 24; click on the Next Button and follow the instruction. Next, it will ask you to select the location of the Application Server; in our case tomcat location is *"D:\Apache\apache-tomcat-10.0.8*". If Apache Tomcat is not available in your system, you have to download the Apache Tomcat Web server from it's official websites: https://tomcat.apache.org/download-10.cgi#10.0.8. Choose the package as per your operating system and architecture requirements. Download Apache Tomcat 10.0.8 and extract the zip file to a specific folder location. In our case, we are downloading 64-bit Windows zip and -extracting it to the location mentioned above.

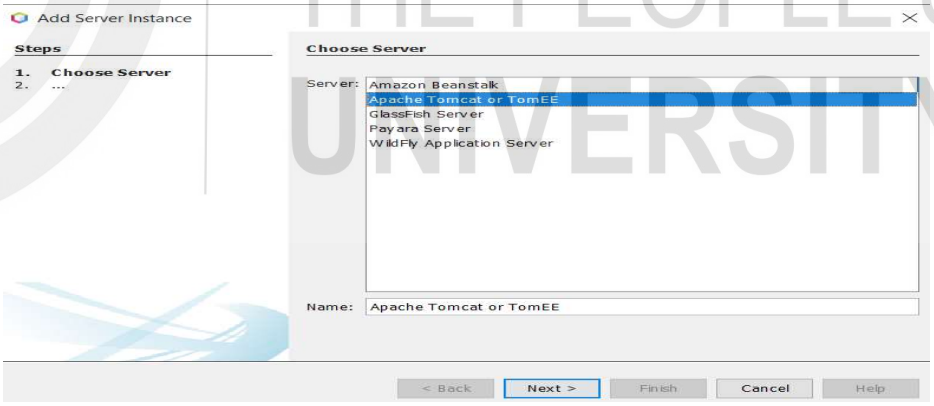


Figure 2.24: Server Selection

Step 8: Installation and Login Details:

Now you need to **s**elect the server location while clicking on the ***Browse button*** to select the location. In this case location is ***D:\Apache\apache-tomcat-10.0.8***

Set the ***username and password*** for Web Server's Manager. Check the ***checkbox*** to create a username and password if it does not exist.

Click Finish to complete the step of installation and login details.

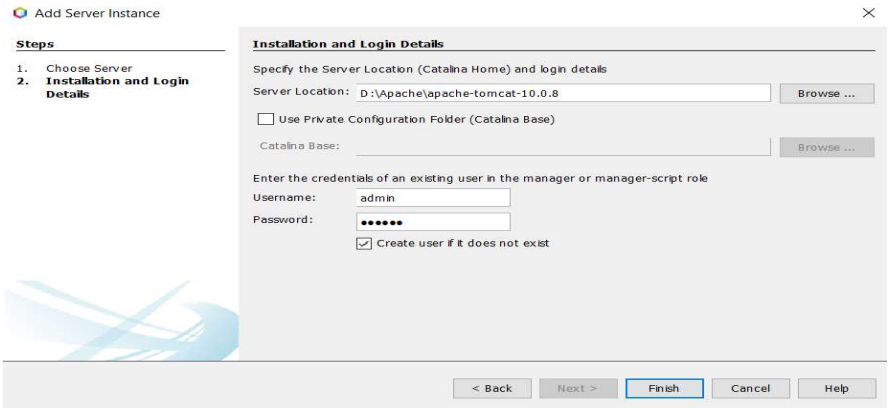


Figure 2.25: Installation and Login details as a Manager of Application Server setup

Step 9: Server and Settings

In this step, Select the server from the available list while clicking on the dropdown and choose one. You can also set the Java EE Version while clicking on the dropdown of the Java EE Version. Keep the context path as default as shown in figure – 2.26.

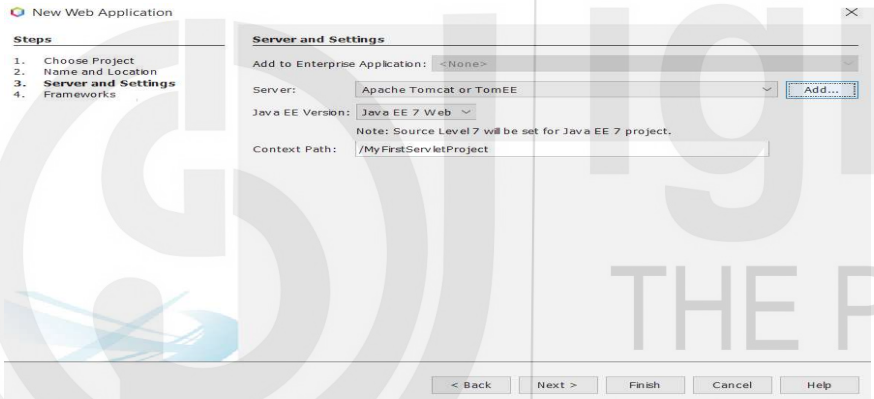


Figure 2.26: Server and JDK setting

Step 10: Select Frameworks

You can select any Framework from the available list to make the Project compliant with that framework. In this case, it will be unchecked because we are creating a servlet project. In the examples in the latter part of this lab section, we will select the Spring Web MVC when we will be creating Spring MVC projects. Click to Finish.

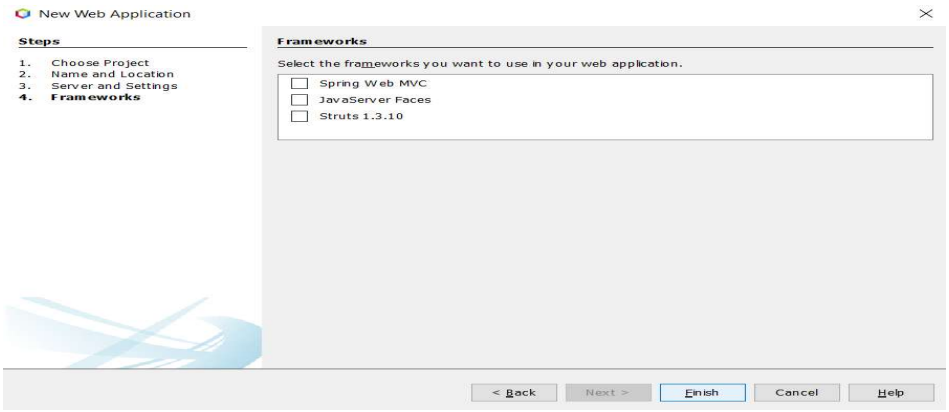


Figure 2.27: Framwork Selection

Step 11: My Project Open in IDE

Now Project Setup is ready for coding using NetBeans IDE. In figure- 2.28, selected parts in index.html are edited before running the Project to display the proper message. To run the Project in NetBeans IDE, you have to right-click on the Project Directory in the left Projects panel, then click on Run.

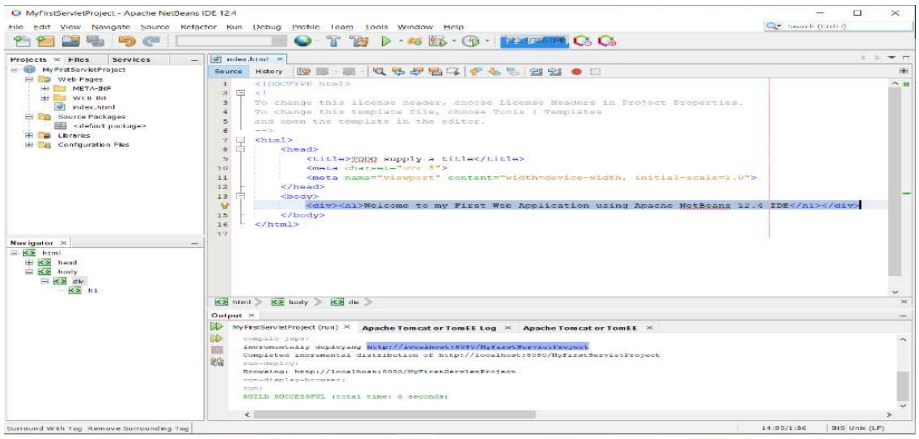


Figure 2.28: Project Structure in IDE

It will automatically open the browser after compilation and running the Project. In this case, the following screen will open or visit http://localhost:8080/MyFirstServletProject to access the application locally in your browser.

Figure 2.29: Output of the Project

2.4 CREATE SERVLET PROJECT

Now you are aware how to create a Servlet Project in Netbeans. In this section, we are creating a Servlet Application using Netbeans. Examples are taken from Block 1 **Unit 2: Basics of Servlet of MCS-220** to show the execution of the programme. The default interface of the IDE while creating a Project using Netbean IDE is shown in figure -2.28.

Example-1: Write a Servlet program to display Servlet Request Information:

Source Code: RequestServlet.java

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Vinay
 */
public class RequestServlet extends HttpServlet {

  /**
   * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
   * methods.
   *
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
          /* TODO output your page here. You may use following sample code. */
          java.util.Date date = new java.util.Date();
          out.println("Current Date & Time: " + date.toString());
          out.println("<h3>Request Information Example</h3>");
          out.println("Method: " + request.getMethod() + "<br>");
          out.println("Request URI: " + request.getRequestURI() + "<br>");
          out.println("Protocol: " + request.getProtocol() + "<br>");
          out.println("PathInfo: " + request.getPathInfo() + "<br>");
          out.println("Remote Address: " + request.getRemoteAddr());
      }
  }

  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
  /**
   * Handles the HTTP <code>GET</code> method.
   *
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      processRequest(request, response);
  }

  /**
   * Handles the HTTP <code>POST</code> method.
   *
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      processRequest(request, response);
```

```
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

Source Code: web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
    <servlet>
        <servlet-name>RequestServlet</servlet-name>
        <servlet-class>RequestServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>RequestServlet</servlet-name>
        <url-pattern>/RequestServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

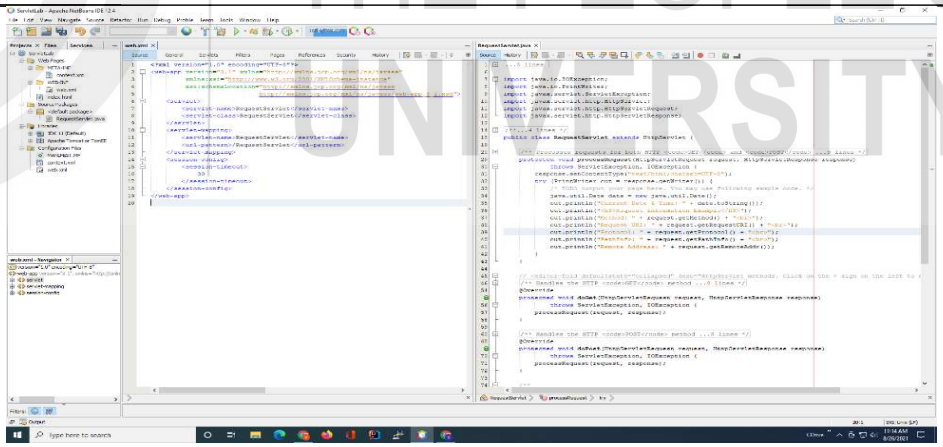


Figure 2.30: Course code in IDE

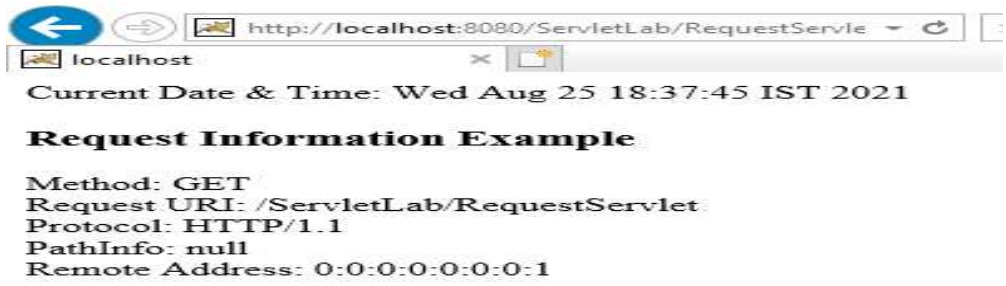


Figure 2.31: Output of Program 1

Example-2: Write a Servlet programme to Create a Form to capture Personal Information and show the values on another Servlet page using Bootstrap.

PersonalInfoFrom.Java

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
*
* @author Vinay
*/
public class PersonalInfoFrom extends HttpServlet {

  /**
   * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
   * methods.
   *
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
          /* TODO output your page here. You may use following sample code. */
          out.println("<!DOCTYPE html>");
          out.println("<html lang=\"en\"><head<meta charset=\"utf-8\"><meta
name=\"viewport\" content=\"width=device-width, initial-scale=1\"><!-- Bootstrap CSS --
><link href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css\"
rel=\"stylesheet\" integrity=\"sha384-
KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We\"
crossorigin=\"anonymous\"><title>Servlet Lab</title></head>");
          out.println("<body>\n"
                  + "        <div class=\"container\">\n"
```

```
                + "          <h1 class=\"text-center\">Personal Information Form</div>\n"
                + "       <div class=\"container\">  \n"
                + "          <form class=\"form row g-3\" action=\"info\" method=\"post\">\n"
                + "             <div class=\"col-12\">\n"
                + "                <label for=\"inputName\" class=\"form-label\">Name</label>\n"
                + "                <input type=\"text\" name=\"name\" class=\"form-control\"
id=\"inputName\" placeholder=\"Enter the full Name\">\n"
                + "             </div>\n"
                + "             <div class=\"col-md-6\">\n"
                + "                <label for=\"inputEmail4\" class=\"form-label\">Email</label>\n"
                + "                <input type=\"email\" name=\"emailId\" class=\"form-control\"
id=\"inputEmail4\" placeholder=\"Enter the Email ID\">\n"
                + "             </div>\n"
                + "             <div class=\"col-md-6\">\n"
                + "                <label for=\"inputPassword4\" class=\"form-
label\">Password</label>\n"
                + "                <input type=\"password\" name=\"pwd\" class=\"form-control\"
id=\"inputPassword4\" placeholder=\"Enter the Password\">\n"
                + "             </div>\n"
                + "             <div class=\"col-12\">\n"
                + "                <label for=\"inputAddress\" class=\"form-
label\">Address</label>\n"
                + "                <input type=\"text\" name=\"address\" class=\"form-control\"
id=\"inputAddress\" >\n"
                + "             </div>\n"
                + "             <div class=\"col-12\">\n"
                + "                <label for=\"inputAddress2\" class=\"form-label\">Address
2</label>\n"
                + "                <input type=\"text\" name=\"address2\" class=\"form-control\"
id=\"inputAddress2\" >\n"
                + "             </div>\n"
                + "             <div class=\"col-md-6\">\n"
                + "                <label for=\"inputCity\" class=\"form-label\">City</label>\n"
                + "                <input type=\"text\" name=\"city\" class=\"form-control\"
id=\"inputCity\">\n"
                + "             </div>\n"
                + "             <div class=\"col-md-4\">\n"
                + "                <label for=\"inputState\" class=\"form-label\">State</label>\n"
                + "                <select id=\"inputState\" name=\"state\" class=\"form-select\">\n"
                + "                   <option selected>Delhi</option>\n"
                + "                   <option>Bihar</option>\n"
                + "                   <option>UP</option>\n"
                + "                </select>\n"
                + "             </div>\n"
                + "             <div class=\"col-md-2\">\n"
                + "                <label for=\"inputZip\" class=\"form-label\">Zip</label>\n"
                + "                <input type=\"text\" name=\"zip\" class=\"form-control\"
id=\"inputZip\">\n"
                + "             </div>\n"
                + "             <div class=\"col-12 text-center\">\n"
                + "                <button type=\"submit\" class=\"btn btn-primary\"
>Submit</button>\n"
                + "             </div>\n"
                + "          </form>\n"
                + "       </div>\n"
                + "       <!-- Bootstrap Bundle with Popper JS-->\n"
                + "       <script
src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js\"
integrity=\"sha384-
U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFIhvj\"
crossorigin=\"anonymous\"></script>\n"
                + "    </body>\n"
```

```
                + "</html>");
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

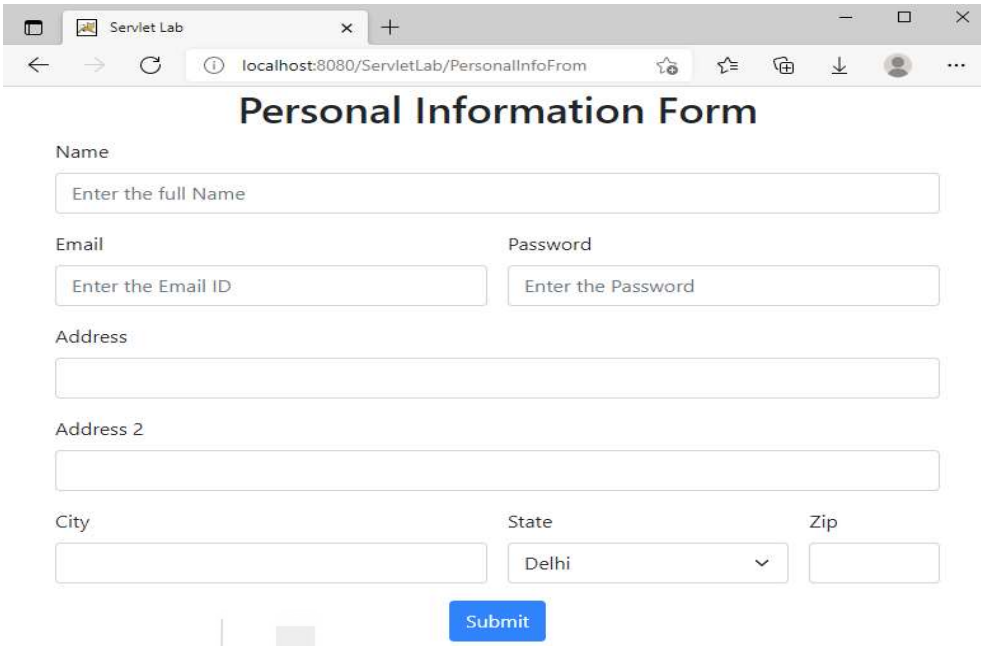


Figure 2.32: Output for Program 2 (PersonalInfoFrom.java)

PersonalInfo.java

```
/*
* To change this license header, choose License Headers in Project Properties.
* To change this template file, choose Tools | Templates
* and open the template in the editor.
*/

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
*
* @author Vinay
*/
public class PersonalInfo extends HttpServlet {

  /**
   * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
   * methods.
   *
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  protected void processRequest(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
          /* TODO output your page here. You may use following sample code. */
          out.println("<!DOCTYPE html>");
          out.println("<html lang=\"en\">\n"
```

```
                + "    <head>\n"
                + "<!-- Required meta tags -->\n"
                + "<meta charset=\"utf-8\">\n"
                + "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n"
                + "<!-- Bootstrap CSS -->\n"
                + "<link
href=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css\"
rel=\"stylesheet\" integrity=\"sha384-
KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We\"
crossorigin=\"anonymous\">\n"
                + "\n <title>Servlet Lab</title>\n"
                + "    </head>\n"
                + "    <body>\n"
                + "<div class=\"container\"><h1 class=\"text-center\">Personal
Informations</div>\n"
                + "<div class=\"container\"> ");
        out.println("<div class=\"col-12\">\n"
                + "<label for=\"inputName\" class=\"form-label text-primary\">Name:</label>\n"
+ request.getParameter("name")
                + "    </div>\n"
                + "    <div class=\"col-md-6\">\n"
                + "<label for=\"inputEmail4\" class=\"form-label text-primary\">Email
Id:</label>\n"+ request.getParameter("emailId")
                + "    </div>\n"
                + "    <div class=\"col-md-6\">\n"
                + "<label for=\"inputPassword4\" class=\"form-label text-
primary\">Password:</label>\n"+ request.getParameter("pwd")
                + "    </div>\n"
                + "    <div class=\"col-12\">\n"
                + "<label for=\"inputAddress\" class=\"form-label text-
primary\">Address:</label>\n"+ request.getParameter("address")
                + "    </div>\n"
                + "    <div class=\"col-12\">\n"
                + "<label for=\"inputAddress2\" class=\"form-label text-primary\">Address
2:</label>\n"+ request.getParameter("address2")
                + "    </div>\n"
                + "    <div class=\"col-md-6\">\n"
                + "<label for=\"inputCity\" class=\"form-label text-primary\">City:</label>\n"+
request.getParameter("city")
                + "    </div>\n"
                + "    <div class=\"col-md-4\">\n"
                + "<label for=\"inputState\" class=\"form-label text-primary\">State:</label>\n"+
request.getParameter("state")
                + "    </div>\n"
                + "    <div class=\"col-md-2\">\n"
                + "<label for=\"inputZip\" class=\"form-label text-primary\">Zip:</label>\n"+
request.getParameter("zip")
                + "    </div>");
        out.println(" </div></div><!-- Bootstrap Bundle with Popper JS-->\n"
                + "<script
src=\"https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js\"
integrity=\"sha384-
U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFIhvj\"
crossorigin=\"anonymous\"></script>\n"
                + "    </body>\n"
                + "</html>");
    }
  }

  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on
the left to edit the code.">
```

```
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Figure 2.33: Output of Program 2(PersonalInfo.java)

web.xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
  <servlet>
    <servlet-name>RequestServlet</servlet-name>
    <servlet-class>RequestServlet</servlet-class>
  </servlet>
```

```
<servlet>
    <servlet-name>PersonalInfo</servlet-name>
    <servlet-class>PersonalInfo</servlet-class>
</servlet>
<servlet>
    <servlet-name>PersonalInfoFrom</servlet-name>
    <servlet-class>PersonalInfoFrom</servlet-class>
</servlet>
<servlet>
    <servlet-name>StudentInfoDisplay</servlet-name>
    <servlet-class>StudentInfoDisplay</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>RequestServlet</servlet-name>
    <url-pattern>/RequestServlet</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>PersonalInfo</servlet-name>
    <url-pattern>/info</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>PersonalInfoFrom</servlet-name>
    <url-pattern>/PersonalInfoFrom</url-pattern>
</servlet-mapping>
<servlet-mapping>
    <servlet-name>StudentInfoDisplay</servlet-name>
    <url-pattern>/StudentInfoDisplay</url-pattern>
</servlet-mapping>
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
</web-app>
```

2.5 CREATE JSP PROJECT

Let us Create a JSP Project with the same example Created in Servlet Project using Apache Netbeans IDE. To do that, open IDE and navigate as follow:

File → New Project → select in categories (Java with Ant → Java Web) → Select Web Application in Project Template → click on Next→ Enter the Project Name and location → Click on Next → Select the Application Server → Next → Finish

Empty Web Application Project Created(For JSP Project)

1. Add **sdtForm.jsp** in Web Pages
2. Add **studentView.jsp** in Web Pages

The directory structure of the Project:

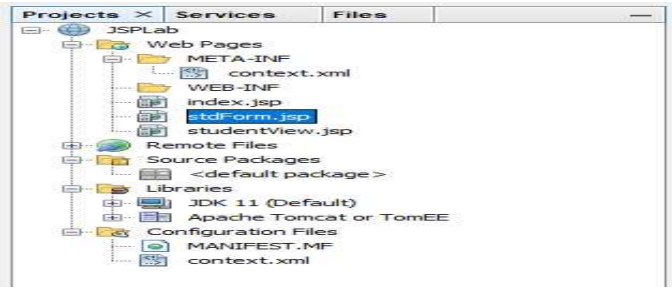


Figure 2.34: Directory Structure of JSP Project

stdForm.jsp

```
<%--
  Document   : stdForm
  Created on : Aug 27, 2021, 2:55:11 PM
  Author     : Vinay
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>JSP Project</title>
  </head>
  <body>
    <div class="container">
      <h1 class="text-center">Personal Information Form</h1>

      <form class="form row g-3" action="studentView.jsp" method="post">
        <div class="col-12">
          <label for="inputName" class="form-label">Name</label>
          <input type="text" name="name" class="form-control" id="inputName"
placeholder="Enter the full Name">
        </div>
        <div class="col-md-6">
          <label for="inputEmail4" class="form-label">Email</label>
          <input type="email" name="emailId" class="form-control" id="inputEmail4"
              placeholder="Enter the Email ID">
        </div>
        <div class="col-md-6">
          <label for="inputPassword4" class="form-label">Password</label>
          <input type="password" name="password" class="form-control"
id="inputPassword4"
              placeholder="Enter the Password">
        </div>
        <div class="col-12">
          <label for="inputAddress" class="form-label">Address</label>
          <input type="text" name="address1" class="form-control" id="inputAddress">
        </div>
        <div class="col-12">
          <label for="inputAddress2" class="form-label">Address 2</label>
          <input type="text" name="address2" class="form-control" id="inputAddress2">
        </div>
        <div class="col-md-6">
          <label for="inputCity" class="form-label">City</label>
          <input

            type="text" name="city" class="form-control" id="inputCity">
        </div>
        <div class="col-md-4">
          <label for="inputState" class="form-label">State</label>
          <select id="inputState" name="state" class="form-select">
            <option selected value="Delhi">Delhi</option>
```

```
                <option value="Bihar">Bihar</option>
                <option value="UP">UP</option>
              </select>
            </div>
            <div class="col-md-2">
              <label for="inputZip" class="form-label">Zip</label> <input
                type="text" name="zipCode" class="form-control" id="inputZip">
            </div>
            <div class="col-12 text-center">
              <button type="submit" class="btn btn-primary"><i class="fas fa-save"></i>
Submit</button>
            </div>
          </form>
      </div>

      <!-- Optional JavaScript; choose one of the two! -->

      <!-- Option 1: Bootstrap Bundle with Popper -->
      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

      <!-- Option 2: Separate Popper and Bootstrap JS -->
      <!--
      <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>
      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>
      -->
  </body>
</html>
```

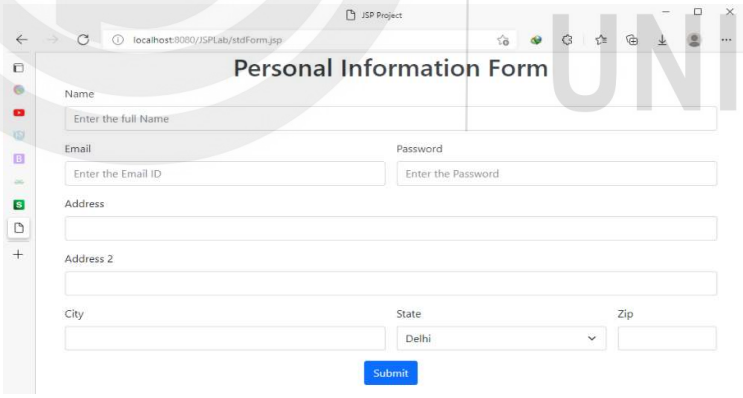


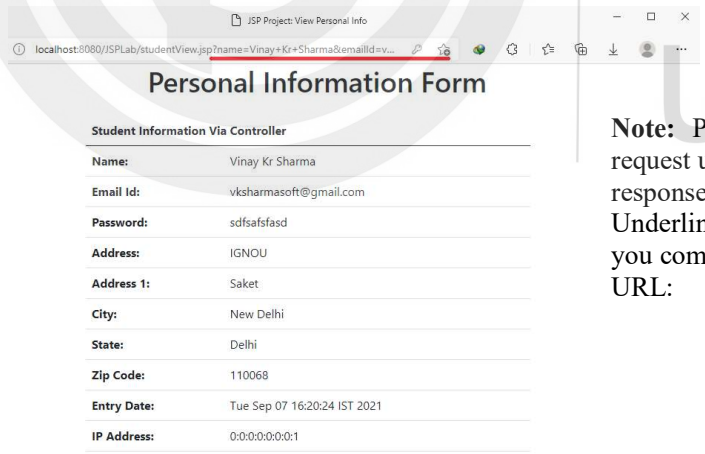
Figure 2.35: Output of sdtForm.jsp

studentView.jsp

```
<%--
  Document   : studentView
  Created on : Sep 7, 2021, 3:34:23 PM
  Author     : Vinay
--%>
<%@page import="java.util.Date"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

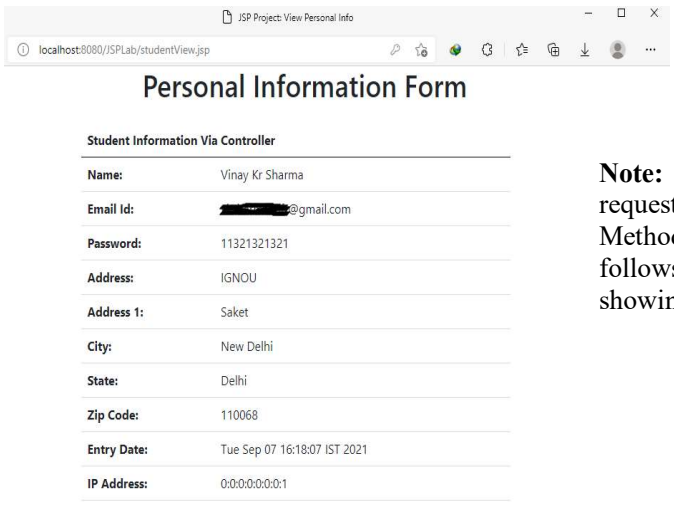
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>JSP Project: View Personal Info</title>
  </head>
  <body>
    <div class="container">
      <h1 class="text-center">Personal Information Form</h1>

     <div class="row col-12 mt-4">
    <div class="col-2"></div>
    <div class="col-8 table-responsive">
      <table class="table">
        <thead>
        <tr>
          <th colspan="2">Student Information Via Controller</th>
        </tr>
        </thead>
        <tbody>
        <tr>
          <th>Name:</th>
          <td><%= request.getParameter("name")%></td>
        </tr>
        <tr>
          <th>Email Id:</th>
          <td><%= request.getParameter("emailId")%></td>
        </tr>
        <tr>
          <th>Password:</th>
          <td><%= request.getParameter("password")%></td>
        </tr>
        <tr>
          <th>Address:</th>
          <td><%= request.getParameter("address1")%></td>
        </tr>
        <tr>
          <th>Address 1:</th>
          <td><%= request.getParameter("address2")%></td>
        </tr>
        <tr>
          <th>City:</th>
          <td><%= request.getParameter("city")%></td>
        </tr>
        <tr>
          <th>State:</th>
          <td><%= request.getParameter("state")%></td>
        </tr>
        <tr>
          <th>Zip Code:</th>
          <td><%= request.getParameter("zipCode")%></td>
        </tr>
        <tr>
```

```
            <th>Entry Date:</th>
            <td><%= new Date()%></td>
          </tr>
          <tr>
            <th>IP Address:</th>
            <td><%= request.getRemoteAddr()%></td>
          </tr>
          </tbody>
        </table>
    </div>
    <div class="col-2"></div>
  </div>
    </div>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"
integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"
integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```



Note: Process the Form request using Get Method, response UI is as follows. Underline in image show you complete form data in URL:

Figure 2.36: Output of studentView.jsp using GET Method



Note: Process the Form request using Post Method, response UI is as follows. Form data is not showing in URL:

Figure 2.37: Output of studentView.jsp using POST Method

2.6 CREATE SPRING MVC, BOOTSTRAP, HIBERNATE PROJECT FOR CRUD

In this section, you will be explained how to create a Spring Boot Project with Dependencies (Spring MVC, Hibernate/JPA) using Spring Initializer (https://start.spring.io/). For the development of Spring Projects, Eclipse IDE is recommended.

To do that, first of all, Spring Tool 4 plugin needs to be configured in default eclipse IDE. If already configured, you mak skip it. This Plugin will provide you with complete intelligence of the Spring Framework for development. To do that following instructions are given:

Open Eclipse IDE → Help → Eclipse Marketplace

Eclipse Marketplace will open. Type Spring in the search box and press the enter to search the Plugin as shown in figure – 2.38. Click on Install in Spring Tool 4 as per the current version then click on the Install button.

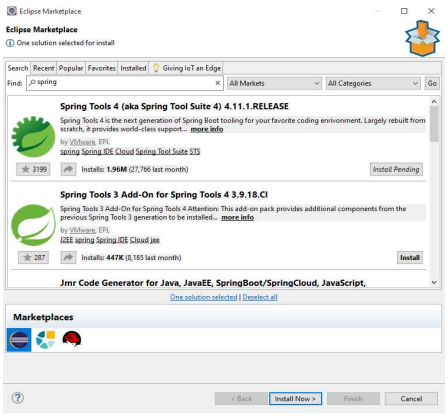


Figure 2.38: Eclipse Marketplace

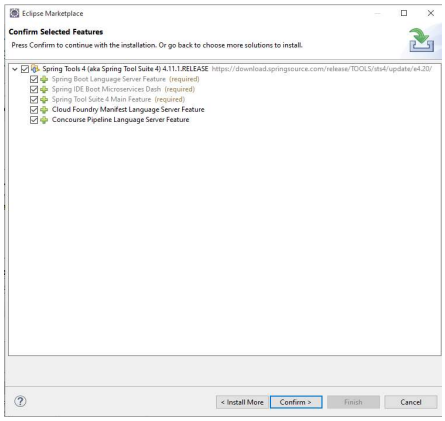


Figure 2.39: Confirm the selected Plugins /Features

Wizard to Configuration feature of Spring Framework will show different options for selection. You can select the options as per your need but selecting all is recommended as shown in figure-2.39. Click to Confirm to proceed further.

Next wizard will ask you to accept the licence, term, and condition, then click on the Finish button to install the Plugin. Once installation is completed, it will ask you to restart the IDE, and you have to restart it.

Now you are ready for Java Application Development using Spring Framework in Eclipse IDE.

Create Spring Project using Spring Initializer

In this section, you will learn how to create all types of maven based Spring project using various dependencies. Using this method, you don't need to manually configure the Project from scratch, and there is no need to add dependencies one by one and configure them correctly. Out of the box, it will provide you complete blank Project with all dependencies along with its configurations. This is the best approach for creating Spring Project without any mistake in Configuration or settings. Let us create a Spring Project using Spring Initializer.

Step 1: Visit the Spring Initializer website (https://start.spring.io/), a page will load as shown in figure- 2.40. On this page, we will make provisioning of Type of Project, languages, Packaging, Metadata of Project and dependencies for our Project.

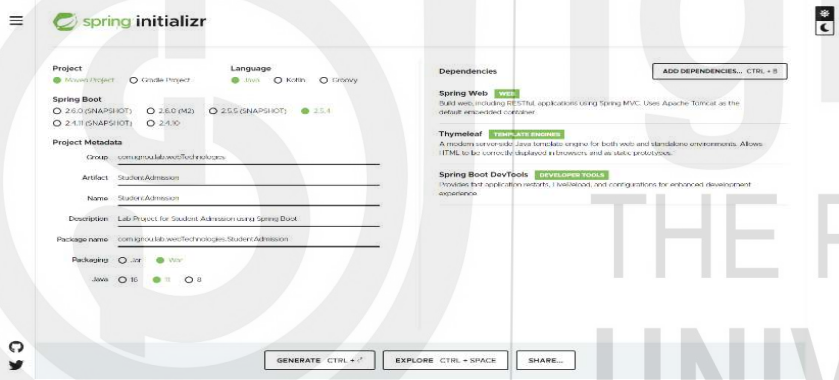


Figure 2.40: Spring Initializer Web Page

Step 2: In this step, we have to set up our requirements for the Spring Project as shown in figure-2.40 and explained in the following bullet points. Here we are creating Maven-based Project using Java Language, so all explanations will be based accordingly.

- **Project:** Maven Project
- **Language:** Java
- **Spring Boot Version:** 2.5.4
- **Project Metadata**
  - **Group:** com.ignou.lab.webTechnologies
  - **Artifact:** StudentAdmission
  - **Name:** StudentAdmission
  - **Description:** Lab Project for Student Admission using Spring Boot
  - **Package name:** com.ignou.lab.webTechnologies.StudentAdmission
  - **Packaging:** War
  - **Java Version:** 11(Current version is 17)
- **Dependencies:** Spring Web, Thymeleaf, Spring Boot DevTool
  - If you want to create a Spring MVC Project, just select Spring web and Thymeleaf (UI Template) dependencies.

- If you want to create a Spring MVC and Hibernate/JPA, just select Spring Data JPA, MySQL Driver dependencies.
- If you want to implement Security add Spring Security dependency.
- If you want to implement a UI Template (Thymeleaf, Apache Framework, Groovy Template, etc) dependencies.

Note: You can also search dependencies from the Maven Repository(https://mvnrepository.com/) and add in pom.xml as per your requirements.

Step 3: Click on Generate to Download the Empty Project with all required Configuration along with directory structure as shown in figure- 40.

Step 4: Now it is time to open the Project in Eclipse IDE. To do that, follow the following instructions.

1. Extract the blank Project which has been downloaded in compress format in step no 3 in your Workspace where your Project will be located.

2. Open Eclipse and navigate to open the Project (***File → Open Projects from file system → Click on Directory and Select the StudentAdmission folder→ click on Finish***) window will open as shown in figure – 2.41.

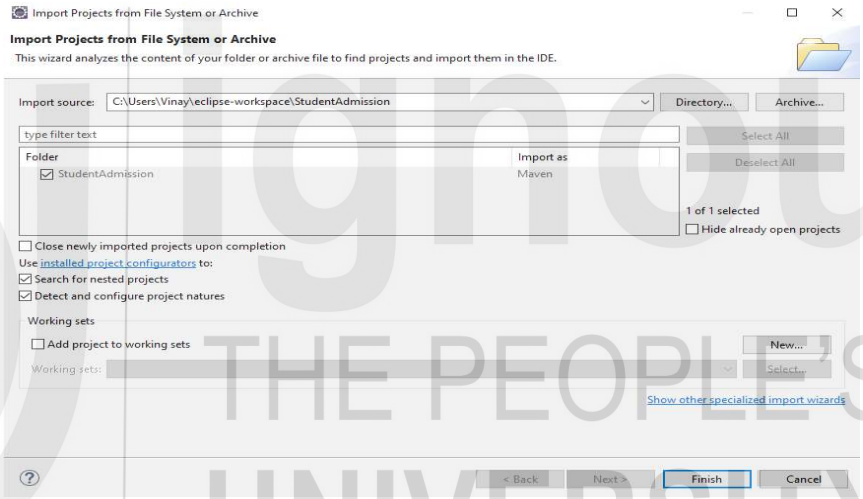


Figure 2.41: Window to Import Project in Eclipse

3. The system will download all dependencies automatically from the Internet. Wait till the process is completed. The default directory structure of the Project is shown in the Project Panel on the left-hand side, as shown in figure - 42. ***Internet is compulsory for working with maven***.

4. Once Processes are completed, the default directory structure of the Project can be seen in figure – 2.42. Now you are ready for coding.

5. Verify the default pom.xml (you can add more dependencies if required in this file at any time during the development. It is mandatory to update the Project after adding dependencies before building and compiling the Project.)

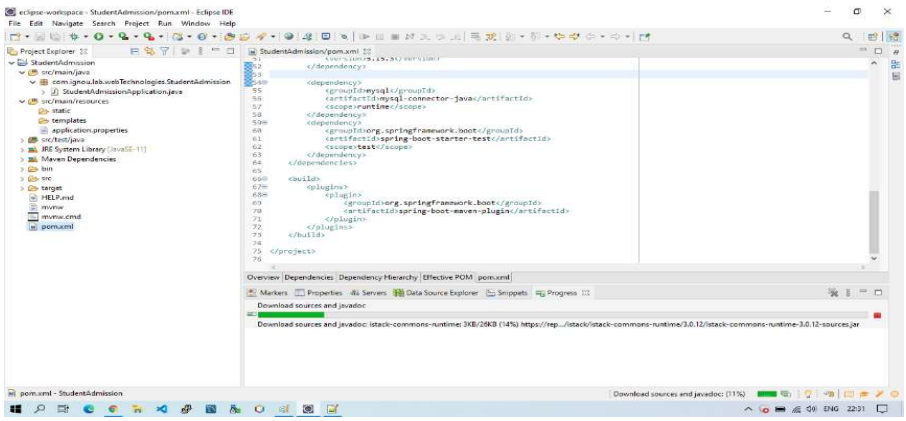


Figure 2.42: Eclipse IDE interface while downloading of dependencies of default project Structure

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.5.4</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.ignou.lab.webTechnologies</groupId>
    <artifactId>StudentAdmission</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>StudentAdmission</name>
    <description>Lab Project for Student Admission using Spring Boot</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
```

```
        <scope>test</scope>
      </dependency>
    </dependencies>

    <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
      </plugins>
    </build>
</project>
```

Note: We can also add dependencies as per our need; for example, to implement bootstrap in this Project, we need to add all required webjars dependencies to support Bootstrap and Font Awesome. You have to add the following Dependencies (you may also choose the version as per your requirements. Visit the dependencies link to see all available versions.)

```
<!-- https://mvnrepository.com/artifact/org.webjars/webjars-locator -->
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>webjars-locator</artifactId>
    <version>0.41</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/jquery -->
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>3.6.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>5.0.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/font-awesome -->
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>font-awesome</artifactId>
    <version>5.15.3</version>
</dependency>
```

6. After adding new dependencies in pom.xml maven will update automatically. If not, just right-click on the Project folder → Maven → Update Project. as shown in figure- 2.43.

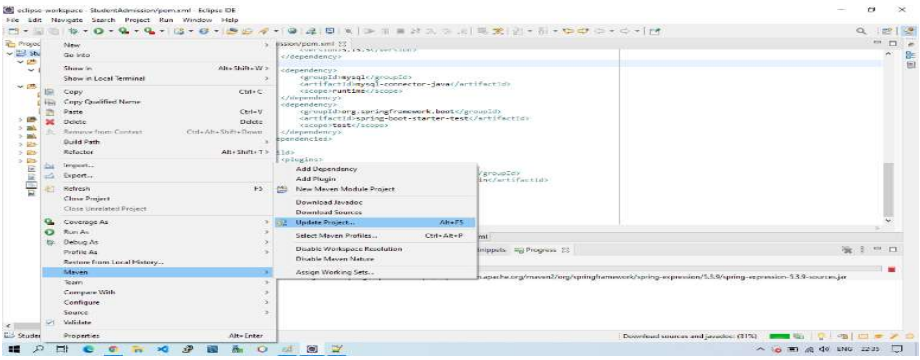


Figure 2.43: How to Update Maven dependencies

Note: Updating the Maven Project takes time. It depends on Internet speed and availability of the Maven repository of all dependencies, so wait till the process is complete.

7. Now you are ready to test the application to do that follow the steps:
    a. Right-Clicking on the Project → Run As→ Spring Boot App
    b. Database connection Error occurred if Database property not set as shown in figure – 2.45. if you are not using hibernate/JPA this error will not occur.

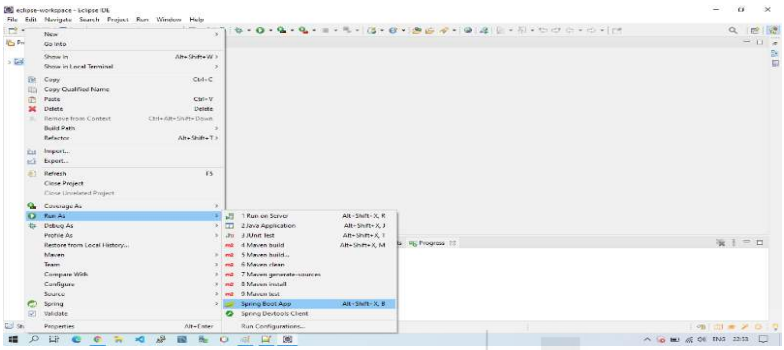


Figure 2.44: How to Run Spring Project in Eclispe

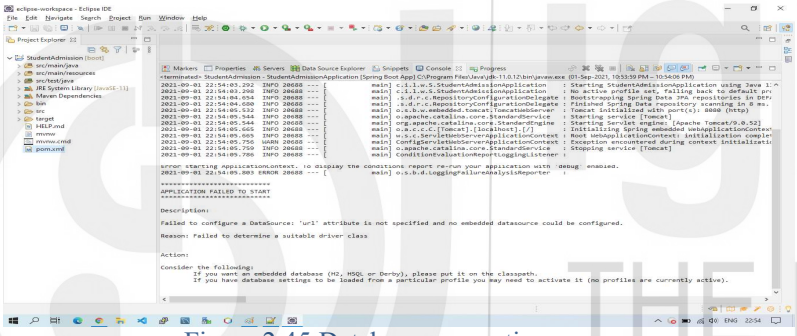


Figure 2.45:Database connection error

a. To resolve the above DB error, you have to add the required database properties in the application.properties file. The following code of snap is the minimum requirement of data source configuration and spring server property.

```
# ==================================
# Server Properties
# ==================================
server.servlet.context-path=/
server.port=8082
# ==================================
# DATABASE
# ==================================
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/Student_Admission
spring.datasource.username=root
spring.datasource.password=Ignou@1234
spring.jpa.hibernate.use-new-id-generator-mappings=false
# ==================================
# JPA / HIBERNATE
# ==================================
spring.jpa.show-sql=true
#spring.jpa.generate-ddl=true
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

d. One more file you should add is index.html in the webapp/template folder in the resource folder. In case of eclipse you should add index.html in the webapp folder.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Insert title here</title>
</head>
<body>
<h1>Welcome to the Lab Project for Student Admission using Spring Boot</h1>
</body>
</html>
```

e. Now re-run the application by following

Right-Clicking on the Project → Run As→ Spring Boot App

Once Project is compiled and run successfully, Eclipse IDE will look as figure -2.46.

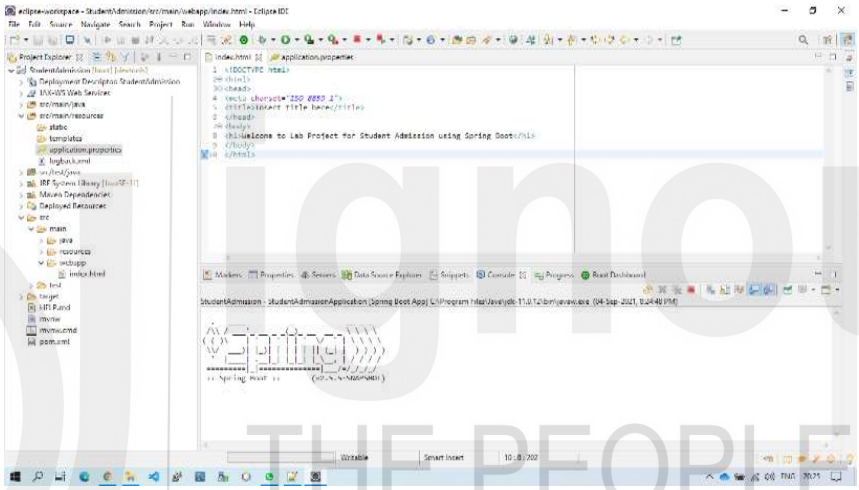


Figure 2.46: Project run Successfully in Eclipse IDE

f. Open http://localhost:8080/ in the Browser following page.

Figure 2.47: Project Output in browser

g. If you want to change port in spring, add the following property in the application.properties

server.port=8084

h. To change the default context path, add the following properties in the application.properties

server.servlet.context-path=/

Now you can start developing in Spring Boot.

2.6.1 ADD BOOTSTRAP, JQUERY, AND FONT AWESOME IN SPRING BOOT

There are 2 ways to add Bootstrap in Spring Boot Project

i. **Maven based:** First of all, we have to add following dependencies in pom.xml, version of dependency may change as per availability and your requirements

```
<!-- https://mvnrepository.com/artifact/org.webjars/webjars-locator -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>webjars-locator</artifactId>
  <version>0.41</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/jquery -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>jquery</artifactId>
  <version>3.6.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>5.0.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.webjars/font-awesome -->
<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>font-awesome</artifactId>
  <version>5.15.3</version>
</dependency>
```

- After adding above dependencies, you have to update maven with following steps
  Right click on project → Maven→ Update Project (Following window will show)
- Check offline and Force Update of Snapshoot/return then click OK button
- Now add the following CSS in Head section in html. Between <head> and </head>

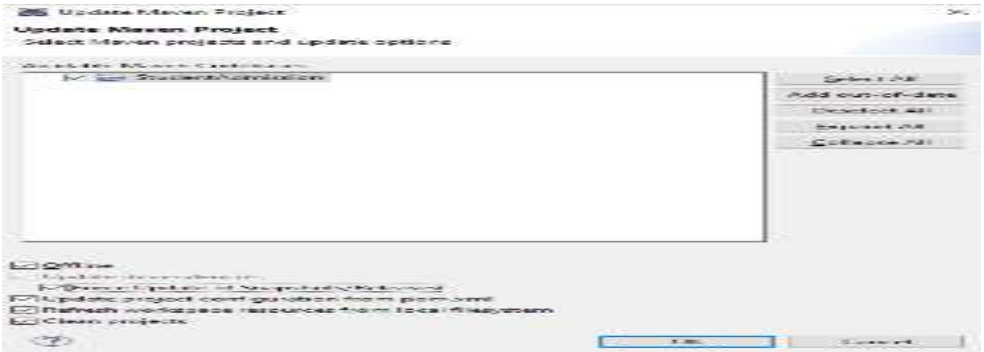


Figure 2.48: Update Maven

<link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet" type="text/css"/>
<link href="/webjars/font-awesome/css/all.css" rel="stylesheet" type="text/css"/>

Note: 1st line of above code is for bootstrap and 2nd line of code is for font awesome

- Now Add the following JavaScript or jQuery just before closing of body tag </body> in html

<script src="/webjars/jquery/jquery.min.js" type="application/javascript"></script>
<script src="/webjars/bootstrap/js/bootstrap.bundle.min.js" type="application/javascript"></script>

Note: 1st line of the above code is for jQuery, and 2nd line of code is for Bootstrap

- Source code of index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
  <meta charset="ISO-8859-1">
  <title>Insert title here</title>
  <link href="/webjars/bootstrap/css/bootstrap.min.css" rel="stylesheet" type="text/css"/>
  <link href="/webjars/font-awesome/css/all.css" rel="stylesheet" type="text/css"/>

</head>
<body>
<h1>Welcome to Lab Project for Student Admission using Spring Boot</h1>
<i class="fas fa-thumbs-up fa-5x"></i>
<script src="/webjars/jquery/jquery.min.js" type="application/javascript"></script>
<script src="/webjars/bootstrap/js/bootstrap.bundle.min.js" type="application/javascript"></script>
</body>
</html>
```

- Now refresh the page (http://localhost:8080/), Page will be load in browser with all default CSS as shown in figure-2.49.

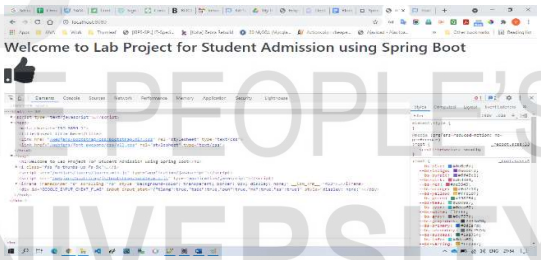


Figure 2.49: Output of Bootstrap in browser

ii. **Direct add in HTML**: To add directly in HTML visit the following link and copy Starter Template

URL: https://getbootstrap.com/docs/5.1/getting-started/introduction/

1. Copy the Starter Template code from above URL and paste into index.html

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We" crossorigin="anonymous">

    <title>Hello, world!</title>
  </head>
  <body>
```

```
    <h1>Hello, world!</h1>
    <!-- Optional JavaScript; choose one of the two! -->
    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFIhvj"
crossorigin="anonymous"></script>
    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"
integrity="sha384-
eMNCOe7tC1doHpGoWe/6oMVemdAVTMs2xqW4mwXrXsW0L84Iytr2wi5v2QjrP/xp"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.min.js"
integrity="sha384-
cn7l7gDp0eyniUwwAZgrzD06kc/tftFf19TOAs2zVinnD/C7E91j9yyk5//jjpt/"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

- Now refresh the page (http://localhost:8080/), Page will be load in browser with all default CSS as shown in figure no 2.50.

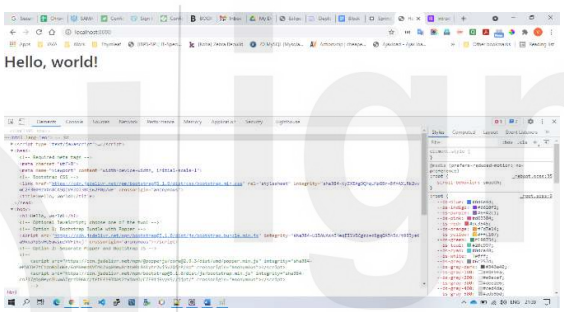


Figure 2.50: Output of bootstrap in Browser

2. To add the font awesome directly in html pages, visit to its official site https://fontawesome.com/start

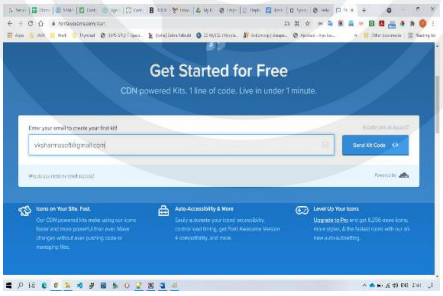


Figure 2.51: Official site of font-awesome

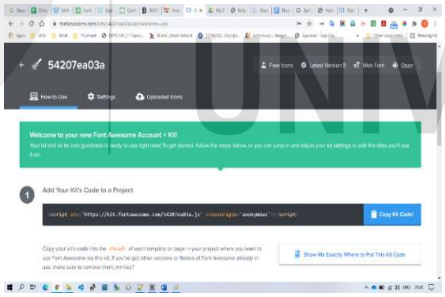


Figure 2.52: Font awesome Kit's Code

3. Enter your email id and click on Send Kid code.
4. You have to register yourself to access it its free to access it.
5. Once you register yourself, you will be able to access it as shown in figure no 52.
6. Click on Copy Kei code or Show me Exactly where to Put this Kit code.
7. Copy the kit's code and icon code and paste it in your HTML.

```
<!doctype html>
<html>
  <head>
    <!-- Place your kit's code here -->
```

```
    <script src="https://kit.fontawesome.com/54207ea03a.js"
crossorigin="anonymous"></script>
  </head>
  <body>
    <!-- Ready to use Font Awesome. Activate interlock. Dynotherms - connected. Infracells -
up. Icons are go! -->
    <i class="fas fa-thumbs-up fa-5x"></i>
  </body>
</html>
```

8. Final HTML will be as follow

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
KyZXEAg3QhqLMpG8r+8fhAXLRk2vvoC2f3B09zVXn8CA5QIVfZOJ3BCsw2P0p/We"
crossorigin="anonymous">
 <script src="https://kit.fontawesome.com/54207ea03a.js" crossorigin="anonymous"></script>
    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
<i class="fas fa-thumbs-up fa-5x"></i>
    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
U1DAWAznBHeqEIlVSCgzq+c9gqGAJn5c/t99JyeKa9xxaYpSvHU5awsuZVVFIhvj"
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js"
integrity="sha384-
eMNCOe7tC1doHpGoWe/6oMVemdAVTMs2xqW4mwXrXsW0L84Iytr2wi5v2QjrP/xp"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/js/bootstrap.min.js"
integrity="sha384-
cn7l7gDp0eyniUwwAZgrzD06kc/tftFf19TOAs2zVinnD/C7E91j9yyk5//jjpt/"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

9. Now Refresh the page and see the output as follow:

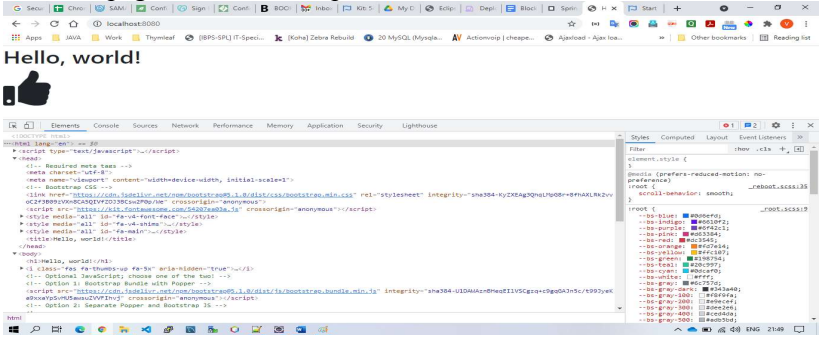


Figure 2.53: Font awesome output using CDN.

2.6.2 CREATE FORM IN SPRING BOOT

Now let us create a form in Spring Boot (MVC) and handle the request through Spring Controller. To achieve the target, we have to follow up the following steps. I am going to use the same example explained in the Servlet Section as above.

a. Generate the blank application using Spring Initializer(https://start.spring.io/) as explained above
b. Extract the Project and open in eclipse/Netbean/Intellij Idea IDE
c. Add the required dependencies in pom.xml then update the maven project to resolve the dependencies.
d. Create all required packages (entity, controller)
e. Create entity class
f. Create the controller for handling requests
g. Create view(add html files in template folder)

Complete Directory Structure of the Project shown in figure no 53

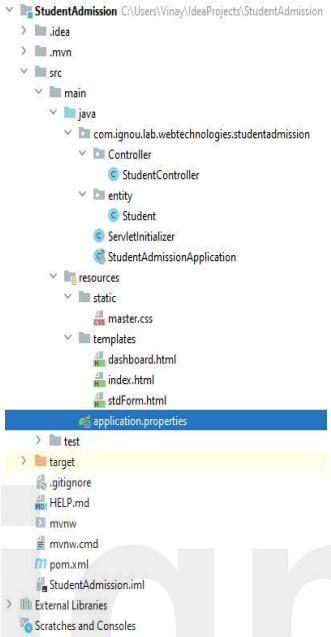


Figure 2.54: Directory Structure of Spring Project

Source Code of Project:

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.5.4</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.ignou.lab.webTechnologies</groupId>
   <artifactId>StudentAdmission</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <packaging>war</packaging>
   <name>StudentAdmission</name>
   <description>Lab Project for Student Admission using Spring Boot</description>
   <properties>
      <java.version>11</java.version>
   </properties>
   <dependencies>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-thymeleaf</artifactId>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-web</artifactId>
      </dependency>
```

```
        <!-- https://mvnrepository.com/artifact/org.webjars/webjars-locator -->
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>webjars-locator</artifactId>
            <version>0.41</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.webjars/jquery -->
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>jquery</artifactId>
            <version>3.6.0</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.webjars/bootstrap -->
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>bootstrap</artifactId>
            <version>5.0.2</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.webjars/font-awesome -->
        <dependency>
            <groupId>org.webjars</groupId>
            <artifactId>font-awesome</artifactId>
            <version>5.15.3</version>
        </dependency>


        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-tomcat</artifactId>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Dashboard.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Student Information Form</title>
  <link th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" th:rel="stylesheet"
type="text/css"/>
```

```
  <link th:href="@{/webjars/font-awesome/css/all.css}" th:rel="stylesheet" type="text/css"/>
  <link rel="stylesheet" th:href="@{/css/master.css}" type="text/css">
</head>
<body>
<div class="container">
  <h1 class="text-center">Welcome to Lab Project for Student Admission using Spring
Boot</h1>
  <div class="row col-12 mt-4">
    <div class="col-3"></div>
    <div class="col-6">
      <a class="btn btn-primary" th:href="@{/stdForm}">Student Form <i class="fas fa-
arrow-right"></i></a>
    </div>
    <div class="col-3"></div>
  </div>
  <div class="row col-12 mt-4">
    <div class="col-2"></div>
    <div class="col-8 table-responsive">
      <table class="table">
        <thead>
        <tr>
          <th colspan="2">Student Information Via Controller</th>
        </tr>
        </thead>
        <tbody>
        <tr>
          <th>Name:</th>
          <td th:text="${student?.name}"/>
        </tr>
        <tr>
          <th>Email Id:</th>
          <td th:text="${student?.emailId}"/>
        </tr>
        <tr>
          <th>Password:</th>
          <td th:text="${student?.password}"/>
        </tr>
        <tr>
          <th>Address:</th>
          <td th:text="${student?.address1}"/>
        </tr>
        <tr>
          <th>Address 1:</th>
          <td th:text="${student?.address2}"/>
        </tr>
        <tr>
          <th>City:</th>
          <td th:text="${student?.city}"/>
        </tr>
        <tr>
          <th>State:</th>
          <td th:text="${student?.state}"/>
        </tr>
        <tr>
          <th>Zip Code:</th>
          <td th:text="${student?.zipCode}"/>
        </tr>
        <tr>
          <th>Entry Date:</th>
          <td th:text="${student?.entryDate}"/>
        </tr>
        <tr>
```

```
                <th>IP Address:</th>
                <td th:text="${student?.ip}"/>
              </tr>
              </tbody>
            </table>
        </div>
        <div class="col-2"></div>
    </div>
</div>

<script th:src="@{/webjars/jquery/jquery.min.js}" type="application/javascript"></script>
<script th:src="@{/webjars/bootstrap/js/bootstrap.bundle.min.js}"
type="application/javascript"></script>
</body>
</html>
```

Important Note: th:text="${student?.name}" here ? is used to check and handle the SpelEvaluationException(Property or field 'name' cannot be found on null) .

stdForm.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Student Information Form</title>
    <link th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" th:rel="stylesheet"
type="text/css"/>
    <link th:href="@{/webjars/font-awesome/css/all.css}" th:rel="stylesheet" type="text/css"/>
    <link rel="stylesheet" th:href="@{/css/master.css}" type="text/css">
</head>
<body>
<div class="container">
    <h1 class="text-center">Personal Information Form</h1>

    <form class="form row g-3" th:action="@{/student/view}" th:object="${student}"
method="post">
        <div class="col-12">
            <label for="inputName" class="form-label">Name</label>
            <input type="text" th:field="*{name}" class="form-control" id="inputName"
placeholder="Enter the full Name">
        </div>
        <div class="col-md-6">
            <label for="inputEmail4" class="form-label">Email</label>
            <input type="email" th:field="*{emailId}" class="form-control" id="inputEmail4"
                   placeholder="Enter the Email ID">
        </div>
        <div class="col-md-6">
            <label for="inputPassword4" class="form-label">Password</label>
            <input type="password" th:field="*{password}" class="form-control"
id="inputPassword4"
                   placeholder="Enter the Password">
        </div>
        <div class="col-12">
            <label for="inputAddress" class="form-label">Address</label>
            <input type="text" th:field="*{address1}" class="form-control" id="inputAddress">
        </div>
        <div class="col-12">
            <label for="inputAddress2" class="form-label">Address 2</label>
            <input type="text" th:field="*{address2}" class="form-control" id="inputAddress2">
        </div>
        <div class="col-md-6">
            <label for="inputCity" class="form-label">City</label>
            <input

                   type="text" th:field="*{city}" class="form-control" id="inputCity">
```

```
      </div>
      <div class="col-md-4">
         <label for="inputState" class="form-label">State</label>
         <select id="inputState" th:field="*{state}" class="form-select">
            <option selected value="Delhi">Delhi</option>
            <option value="Bihar">Bihar</option>
            <option value="UP">UP</option>
         </select>
      </div>
      <div class="col-md-2">
         <label for="inputZip" class="form-label">Zip</label> <input
            type="text" th:field="*{zipCode}" class="form-control" id="inputZip">
      </div>
      <div class="col-12 text-center">
         <button type="submit" class="btn btn-primary"><i class="fas fa-save"></i>
Submit</button>
      </div>
   </form>
</div>

<script th:src="@{/webjars/jquery/jquery.min.js}" type="application/javascript"></script>
<script th:src="@{/webjars/bootstrap/js/bootstrap.bundle.min.js}"
type="application/javascript"></script>
</body>
</html>
```

Student.java (Controller Class)

```
package com.ignou.lab.webtechnologies.studentadmission.Controller;

import com.ignou.lab.webtechnologies.studentadmission.entity.Student;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import javax.servlet.http.HttpServletRequest;
import java.util.Date;

@Controller
public class StudentController {
   @GetMapping("/")
   public String welcome() {
      return "redirect:/dashboard";
   }

   @GetMapping("/dashboard")
   public String maiPage() {
      return "dashboard";
   }

   @GetMapping("/stdForm")
   public String studentInputForm(Model theModel) {
      Student student = new Student();
      theModel.addAttribute("student", student);
      return "stdForm";
   }

   @PostMapping("/student/view")
   public String viewStdInfo(@ModelAttribute("student") Student theStudent, Model
theModel, HttpServletRequest request) {
      theStudent.setEntryDate(new Date());
      theStudent.setIp(request.getRemoteAddr());
```

```
        theModel.addAttribute("student", theStudent);
        return "dashboard";
    }

}
```

Important Note: In this controller class, four functions are added, let us see them one by one

1. **welcome():** this method is mapped with /(default end URL of application where application will land that is http://localhost:8080/). In this function, the response is redirecting to another end URL using "redirect:/dashboard". This mechanism will be used when you want your controller to redirect response to another controller as a request.
2. **maiPage()**: This Method is used to show the default view of your application. End URL of this method is http://localhost:8080/dashboard
3. **studentInputForm()**: In this method, we are binding our Entity object with Form object. The Model represents here a Java object carrying data that's why entity object is added as Model's attribute, which will be used in stdForm.html as Form Object. End URL of this method is http://localhost:8080/stdForm
4. **viewStdInfo()**: This method is handling form request, and response are shown on dashboard.html page End URL of this method is http://localhost:8080/student/view

Student.java(Entity Class)

```
package com.ignou.lab.webtechnologies.studentadmission.entity;

import java.util.Date;

public class Student {
    private int id;
    private String name;
    private String emailId;
    private String password;
    private String address1;
    private String address2;
    private String city;
    private String state;
    private String zipCode;
    private Date entryDate;
    private String ip;

    public Student() {
    }

    public Student(int id, String name, String emailId, String password, String address1, String
address2, String city, String state, String zipCode, Date entryDate, String ip) {
        this.id = id;
        this.name = name;
        this.emailId = emailId;
        this.password = password;
        this.address1 = address1;
        this.address2 = address2;
        this.city = city;
        this.state = state;
        this.zipCode = zipCode;
        this.entryDate = entryDate;
        this.ip = ip;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
```

```
public String getName() {
   return name;
}

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

public String getEmailId() {
   return emailId;
}

public void setEmailId(String emailId) {
   this.emailId = emailId;
}

public String getPassword() {
   return password;
}

public void setPassword(String password) {
   this.password = password;
}

public String getAddress1() {
   return address1;
}

public void setAddress1(String address1) {
   this.address1 = address1;
}

public String getAddress2() {
   return address2;
}

public void setAddress2(String address2) {
   this.address2 = address2;
}

public String getCity() {
   return city;
}

public void setCity(String city) {
   this.city = city;
}

public String getState() {
   return state;
}

public void setState(String state) {
   this.state = state;
}

public String getZipCode() {
   return zipCode;
}

public void setZipCode(String zipCode) {
   this.zipCode = zipCode;
```

```
    }

    public Date getEntryDate() {
        return entryDate;
    }

    public void setEntryDate(Date entryDate) {
        this.entryDate = entryDate;
    }

    public String getIp() {
        return ip;
    }

    public void setIp(String ip) {
        this.ip = ip;
    }
}
```

Figure 2.55: Output of Personal Information Form

Figure 2.56: View Input Details after Submitting the Form

2.7 LIST OF LAB ASSIGNMENTS (SESSION WISE)

In this section List of Problems is given for deep dive into real programming. Unit wise problem/exercise given for your lab. You can use Apache NetBeans IDE or Eclipse IDE for doing the following Lab Exercises:

SESSION 1: BASIC OF SERVLET

1. Write a Servlet Programme to print the current date and time along with the timestamp.
2. Create an HTML form with the input of student information using HTTP Protocol and method, then display the input information using Servlet.
3. Write a servlet program to capture client IPs and display it.
4. Write a servlet program for session management using HTTP Session along with tracking and also use a cookie for session tracking.
5. Write a CRUD (Create/Save, Read, Edit/Update, Delete) application using servlet. Create a Database named IGNOU, create a table named Student which must capture the student information (basics, contact, enrollment details along with courses). Make necessary assumptions required.

SESSION 2: JSP

1. Write JSP Programme to print current date and time along with timestamp, implement auto-refresh of a page.
2. Create a JSP page and implement a Scripting Tag, Expression tag and Declaration tag.
3. Import JSTL library in JSP Page and use its following tags:
    a. out
    b. if
    c. forEach
    d. choice, when and otherwise
    e. url and redirect
4. Create a JSP Page for database connectivity using JDBC and show the students details from the database created during exercise no 5 in session 1.
5. Write a JSP application using following Action Elements
    a. jsp:forward
    b. jsp:include
    c. set & getProperty
    d. jsp:useBean
6. Write a JSP program using the following implicit objects with an example:
    a. Out
    b. request
    c. response
    d. Session
    e. pageContext
    f. exception
7. Create a JSP Project implementing all the above (Session 1 and Session 2) concepts. Login Form, CRUD operation of Student details, Session Management with exception handling using Servlet and JSP. Make necessary assumptions required.

SESSION 3: MAVEN, FRAMEWORK FOR J2EE

1. Create a Maven-based project using Spring Initializr.
2. Create a Maven-based J2EE application for Spring MVC.

3. Create a Maven-based J2EE application for Hibernate and JPA. Use the same database created in Exercise no. 5 of Session 1.
4. Define a new implementation of Teacher Interface for his/her favourite Course in Spring using Inversion of Control and retrieving the information from the new teacher implementation.

SESSION 4: DEPENDENCY INJECTION, CONTROLLER

1. Write a class and implement it using dependency injection.
2. Write the service interface with the getRandom() method, define 3 courses in an array, and inject using dependency injection into the teacher Interface(exercise no 4 in session 3). Test the application and verify the retrieving of random courses.
3. Write a programme using Spring Framework to create a controller and display response in view using @GetMapping() annotation.
4. Create a Form to capture Student Admission information (make usual assumptions about the attributes) using Spring Form tags (must use Text, textarea, dropdown, date picker, true/false and checkbox) and write a controller using @PostMapping() to display the form information.

SESSION 5: FORM VALIDATION, BOOTSTRAP AND CSS

1. Apply the client validation in the form created in the above exercise 4 of session 4, along with server-side validation.
2. Write a programme to bind form objects with entity bean in Spring MVC.
3. Configure Bootstrap in Spring MVC and use default styling classes in the form and view created in the above exercises.
4. Apply custom Styling to your pages in Spring MVC.

SESSION 6: HIBERNATE, JPA

1. Create a database for the student admission lifecycle and configure Hibernate and JPA with the Spring MVC Project along with all table entities.
2. Retrieve Student information using Hibernate and printing the value in the console.
3. Create CRUD (Create/Save, Read/Fetch, Edit/Update, Delete) using Spring MVC and Hibernation.
4. Apply batch update for student's admission approval using Spring MVC and Hibernate.

SESSION 7: SPRING BOOT AND REST CONTROLLER

1. Create a Spring Boot application using Spring Initializer. Add the following dependencies manually:
    a. Spring MVC
    b. Hibernate
    c. JPA
    d. Thymeleaf
    e. DevTool
    f. Actuator
    g. MySQL/MSSQL/Oracle/MongoDB (as per your choice) driver.
2. Configure Database settings through the property file in Spring Boot.
3. Create JPA Repositories for all entities used in the Student Admission lifecycle.

4. Create Rest Controller to fetch Student Information using JPA Repository; the response should display in JSON format.

SESSION 8: REST API, SPRING SECURITY, ACTUATOR

1. Add actuator dependency in Spring boot and Test Actuator (all available Endpoints).
2. Create REST API for CRUD operation using Spring Boot and Hibernate/JPA using annotation.
3. Create a REST API for CRUD operation using Spring Boot and Hibernate/JPA using XML configuration.
4. Add Spring Security dependency in the existing spring boot application created in Exercise 2 of Session 8 and configure it as the default login.

SESSION 9: SPRING SECURITY

1. Create a User table into the database and bind the user entity with Spring Security for Login.
2. Create a Custom Login Page in HTML and authenticate using Spring Security in Spring Boot.
3. Implement logout functionality in Spring Security.
4. Create a User registration form and validate the form. Once information is validated and saved, write functionality to auto-login using Spring Security.

SESSION 10: BOOTSTRAP, ROLE BASED AUTHENTICATION AND CSRF

1. Add Bootstrap styling in the Login and registration page in the above exercises in session 9.
2. Create a Role Table in the database and configure it with Spring Boot, Security and Hibernate.
3. Write code for role-based authentication using Spring Security.
4. Display current logged in User details on the dashboard along with client IP, data time and user's current role.
5. Add CSRF functionality in the authentication.
6. Restrict role-based access to views in Spring Boot Security.

2.8 REFERENCES

- Download JDK: https://www.oracle.com/in/java/technologies/javase-downloads.html
- JDK Installation Document for Multiple Platform: https://docs.oracle.com/en/java/javase/16/install/overview-jdk-installation.html#GUID-8677A77F-231A-40F7-98B9-1FD0B48C346A
- Java Environment PATH AND CLASSPATH Setup for the various platform: https://docs.oracle.com/javase/tutorial/essential/environment/paths.html
- Download NetBeans IDE: https://netbeans.apache.org/download/index.html
- Download Eclipse IDE: https://www.eclipse.org/downloads/
- Download Apache Tomcat Web Server: https://tomcat.apache.org/download-10.cgi#10.0.8