EX NO: 1 DATE:	CREATE A WEBPAGE AND IMPLEMENT WEB MAPPING

AIM:

To Create a web page with the following using HTML.

- To embed an image map in a web page.
- To fix the hot spots.
- Show all the related information when the hot spots are clicked.

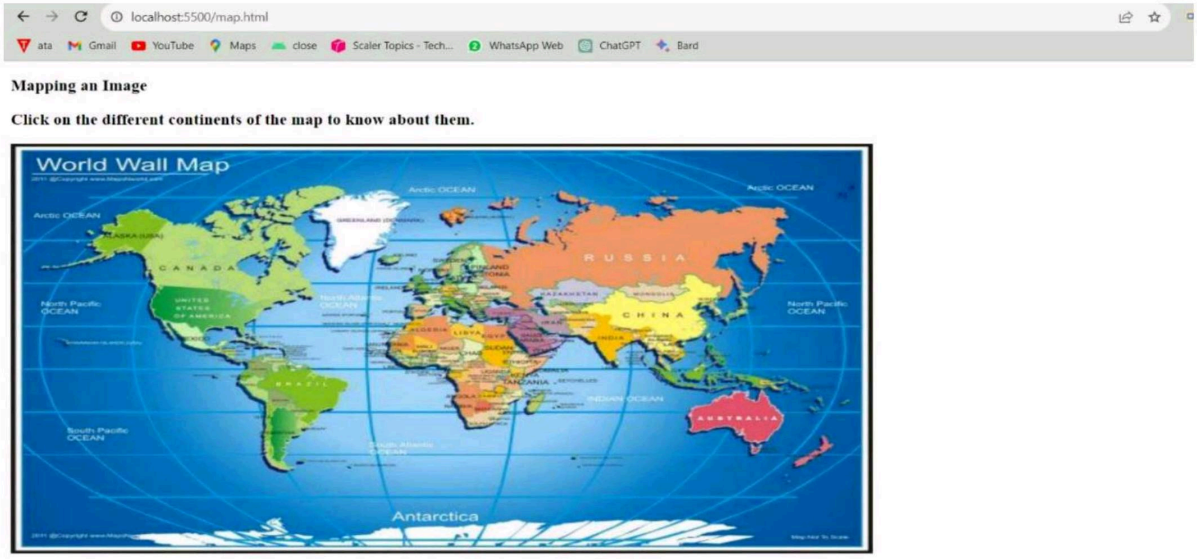
ALGORITHM:

1. Declare HTML structure with language and type.
2. Add world map image with dimensions and alt text.
3. Create clickable continent areas using <area> tags.
4. Define coordinates and link each area to Wikipedia.
5. Complete HTML structure and tag closure.

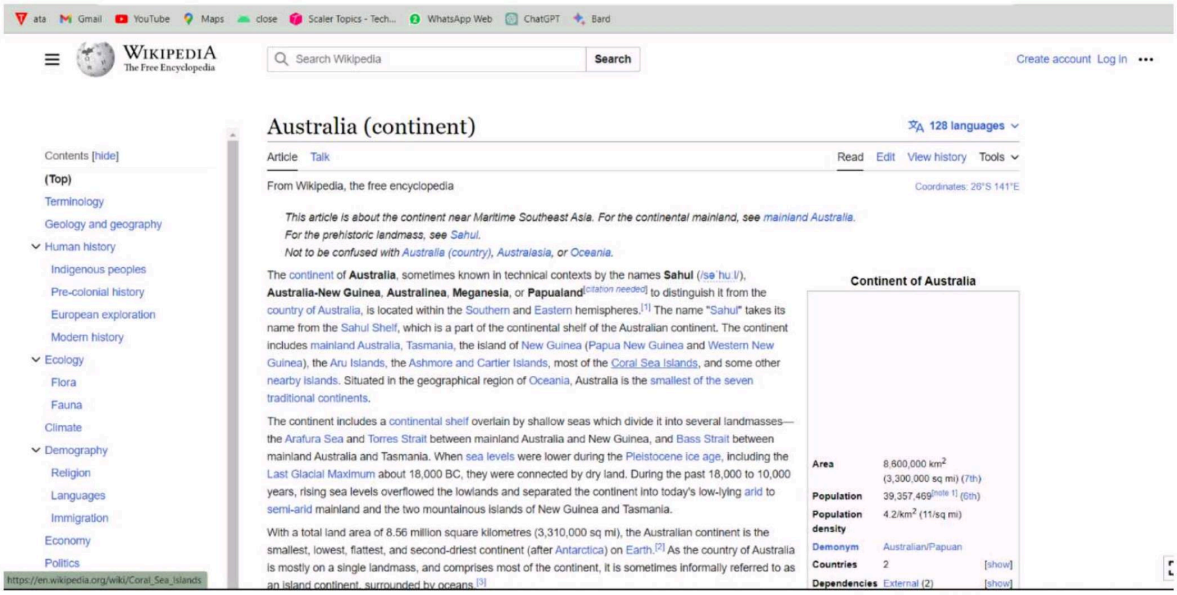
PROGRAM:

```
<!DOCTYPE html>
<html>
<h3>Mapping an Image
<body>
<p>Click on the different continents of the map to know about them.</p>
<img src="worldmap.jpg" width="960" height="492" alt="World Map"
usemap="#worldmap">
<map name="worldmap">
<area shape="rect" coords="184, 36, 272, 158" alt="north america"
                                href="https://en.wikipedia.org/wiki/North_America">
<area shape="rect" coords="282, 215, 354, 367" alt="south america"
                                href="https://en.wikipedia.org/wiki/South_America">
<area shape="rect" coords="506, 151, 570, 333" alt="africa"
                                href="https://en.wikipedia.org/wiki/Africa">
<area shape="rect" coords="618, 42, 791, 162" alt="asia"
                                href="https://en.wikipedia.org/wiki/Asia">
<area shape="rect" coords="509, 44, 593, 110" alt="europe"
                        href="https://en.wikipedia.org/wiki/Europe">
<area shape="rect" coords="786, 288, 862, 341" alt="australia"
                        href="https://en.wikipedia.org/wiki/Australia_(continent)">
<area shape="rect" coords="249, 463, 760, 488" alt="antartica"
                        href="https://en.wikipedia.org/wiki/Antarctica">
</map>
</body>
</html>
```

OUTPUT:



Related information for the hotspot:



RESULT:

The program is executed and Image mapping is done successfully.

EX NO: 2 DATE:	CREATE A WEB PAGE WITH ALL TYPES OF CASCADING STYLE SHEETS.

AIM:

To create a webpage with all types of cascading style sheets.

ALGORITHM:

1. Define HTML document structure and language.
2. Create internal styles for body, h1, and p.
3. Link an external stylesheet (styles.css).
4. Apply inline styles to specific <h1> and <p> elements.
5. Define additional styles for <h1> and <p> within <style> in <head>.

PROGRAM:

Style.html:

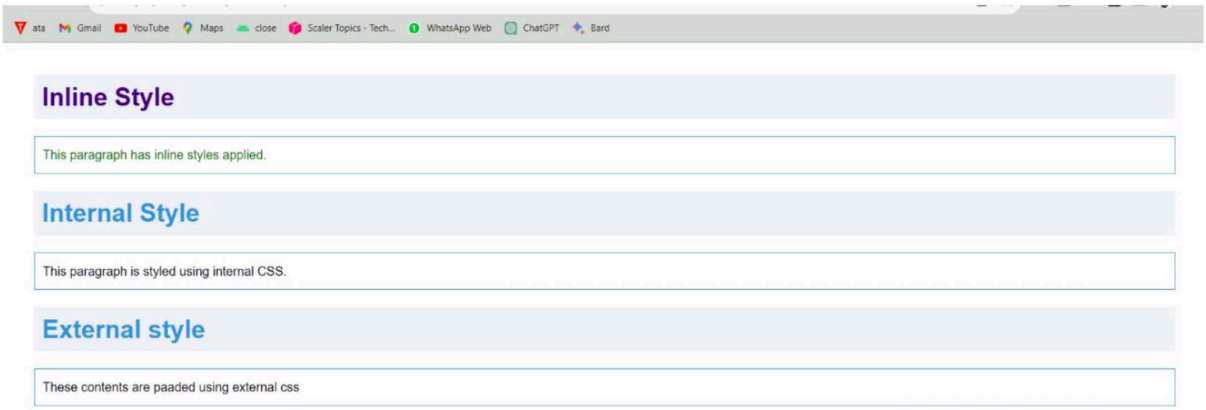
```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>CSS Example</title>
   <!-- Internal Style -->
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 40px;
      }
      h1 {
         color: #3498db;
      }
      p {
         font-size: 16px;
         line-height: 1.6;
      }
   </style>
    <!-- External Style -->
   <link rel="stylesheet" href="styles.css">
```

```
</head>
<body>
    <h1 style="color: indigo;">Inline Style</h1>
    <p style="color: green;">This paragraph has inline styles applied.</p>
    <h1>Internal Style</h1>
    <p>This paragraph is styled using internal CSS.</p>
    <h1> External style</h1>
    <p> These contents are padded using external CSS</p>
</body>
</html>
```

Styles.css:

```
/* External Style */
h1 {
    background-color: #ecf0f1;
    padding: 10px;
}
p {
    border: 1px solid #3498db;
    padding: 10px;
}
```

OUTPUT:



RESULT: Program is executed and the webpage with all types of cascading style sheets is successfully obtained.

EX NO: 3 DATE:	CREATE CLIENT SIDE SCRIPTS FOR VALIDATING WEB FORM CONTROLS USING DHTML

AIM:

To create a simple HTML form with DHTML-based client-side validation to ensure the name and email are filled out correctly, triggering an alert for a successful form submission.

ALGORITHM:

1. Initialize HTML document.
2. Style error messages with red color using CSS.
3. Write a JavaScript function (validateForm) for form validation.
    - Get name and email values.
    - Check if name is empty, display an alert, and return false.
    - Check if email is empty, display an alert, and return false.
    - Use a regular expression to validate email format, display an alert if invalid, and return false.
    - If all validations pass, display a success alert and return true.
4. Create an HTML form (myForm) with onsubmit calling validateForm and method set to "post."
5. Include input fields for name and email with appropriate labels.
6. Add a submit button to trigger form submission.

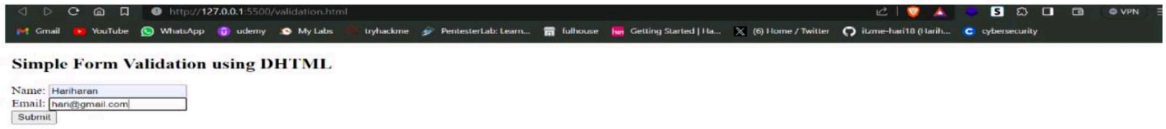
PROGRAM:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
    <style>
        .error {
            color: red;
        }
    </style>
    <script>
        function validateForm() {
```

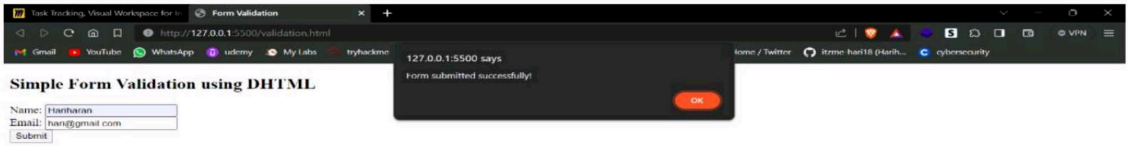
```
        var name = document.forms["myForm"]["name"].value;
        var email = document.forms["myForm"]["email"].value;
        // Simple validation for name and email
        if (name == "") {
          alert("Name must be filled out");
          return false;
        }
        if (email == "") {
          alert("Email must be filled out");
          return false;
        }
        // Additional email format validation using a regular expression
        var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!email.match(emailRegex)) {
          alert("Invalid email format");
          return false;
        }
        // If all validations pass, the form is submitted
        alert("Form submitted successfully!");
        return true;
      }
    </script>
</head>
<body>
    <h2>Simple Form Validation using DHTML</h2>
    <form name="myForm" onsubmit="return validateForm()" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name">
        <br>
 <label for="email">Email:</label>
        <input type="text" id="email" name="email">
        <br>
```

```
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

OUTPUT :



Validation of the form



RESULT :

Thus the program is executed and the validation from implemented successfully using DHTML .

EX NO: 4 DATE:	INSTALLATION OF APACHE TOMCAT WEB SERVER

AIM:

The aim of this lab is to install Apache Tomcat Web Server.

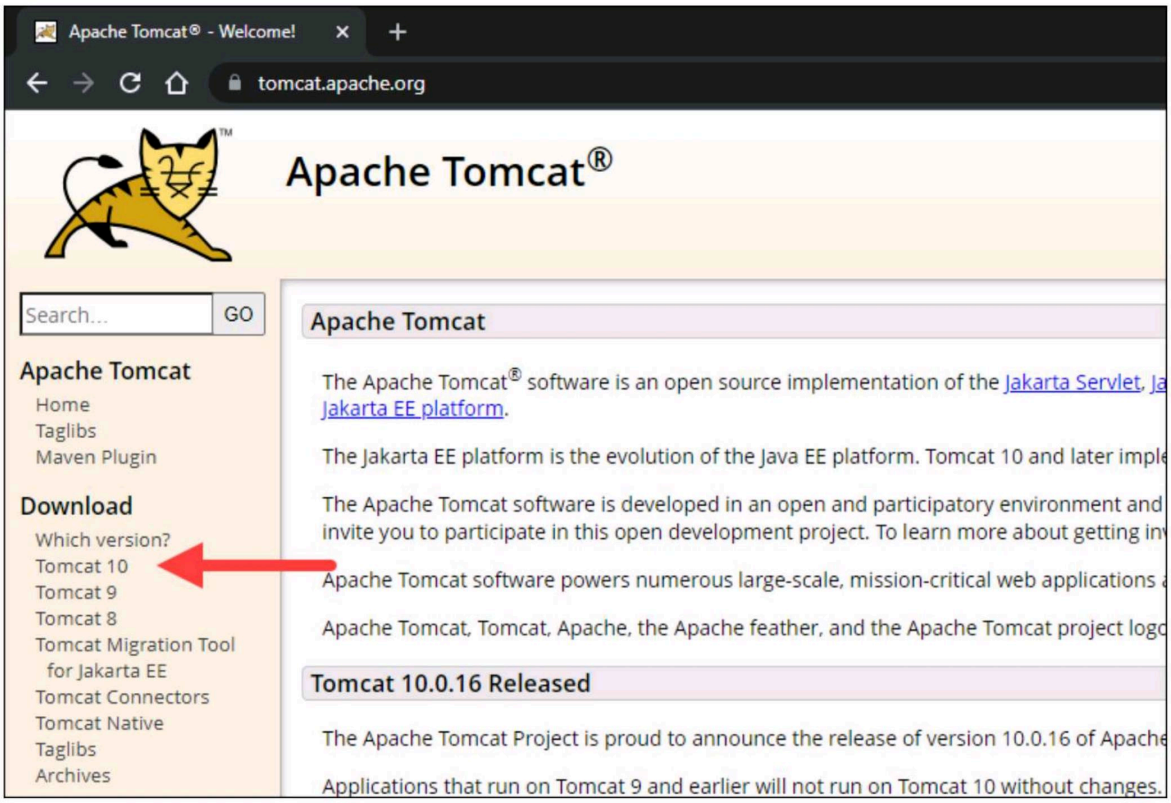
PREREQUISITES:

- Java JRE installed and configured
- Administrator privileges

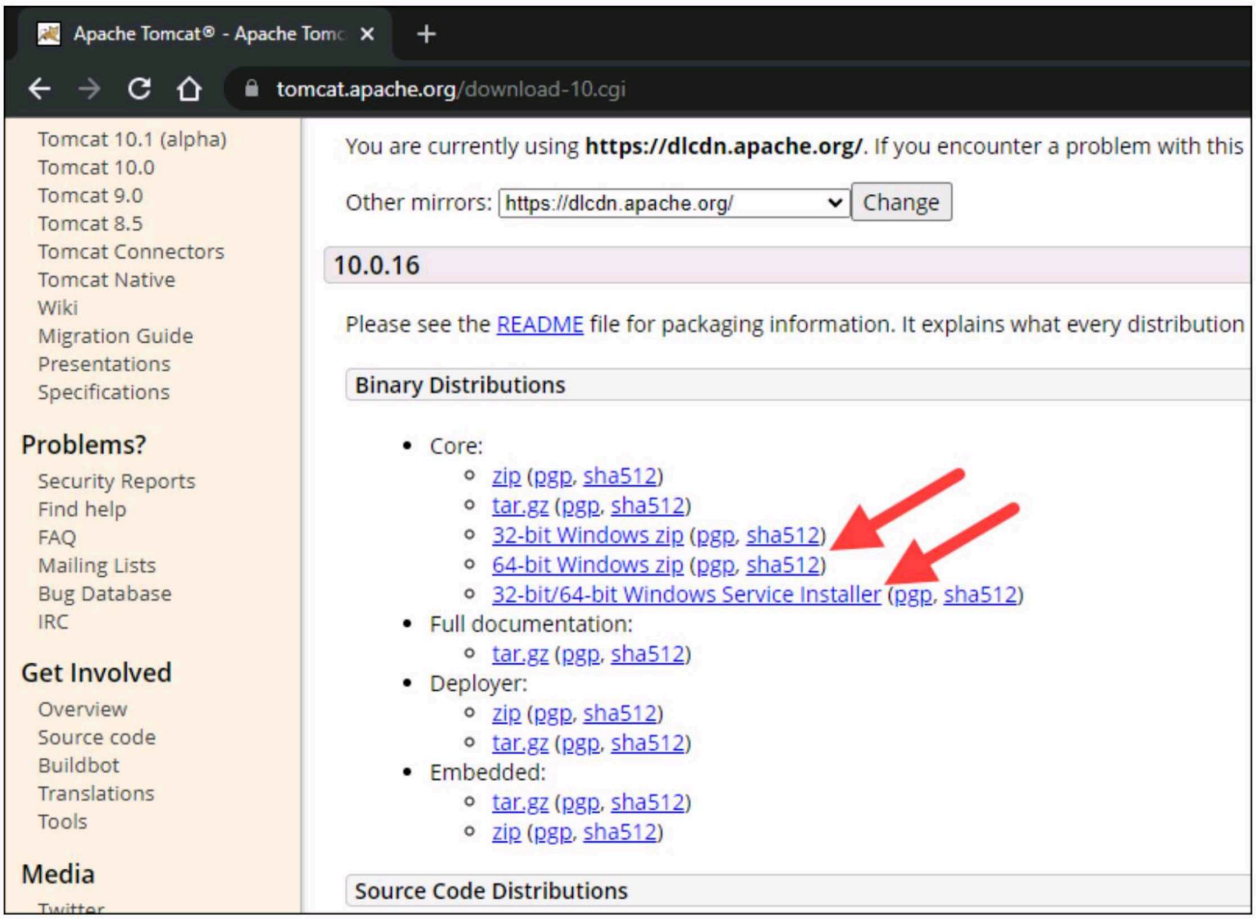
PROCEDURE:

Step 1: Download Tomcat for Windows
To download the Tomcat installation file, follow the steps below:

1. Browse to the official Apache Tomcat website. Locate the *Download* section and click the **latest Tomcat version** available. At the time of writing this article, the latest Tomcat version was version 10.



2. On the *Download* page, scroll down and locate the *Binary Distributions* area.
   - In the *Core* list, depending on the installation type you prefer, click the download link for the **Windows Service Installer** or the **32bit/64bit Windows zip file**.



Step 2: Install Tomcat

Install Tomcat via the **Windows Service Installer** for an automated and wizard-guided experience. The service installer installs the Tomcat service and runs it automatically when the system boots.

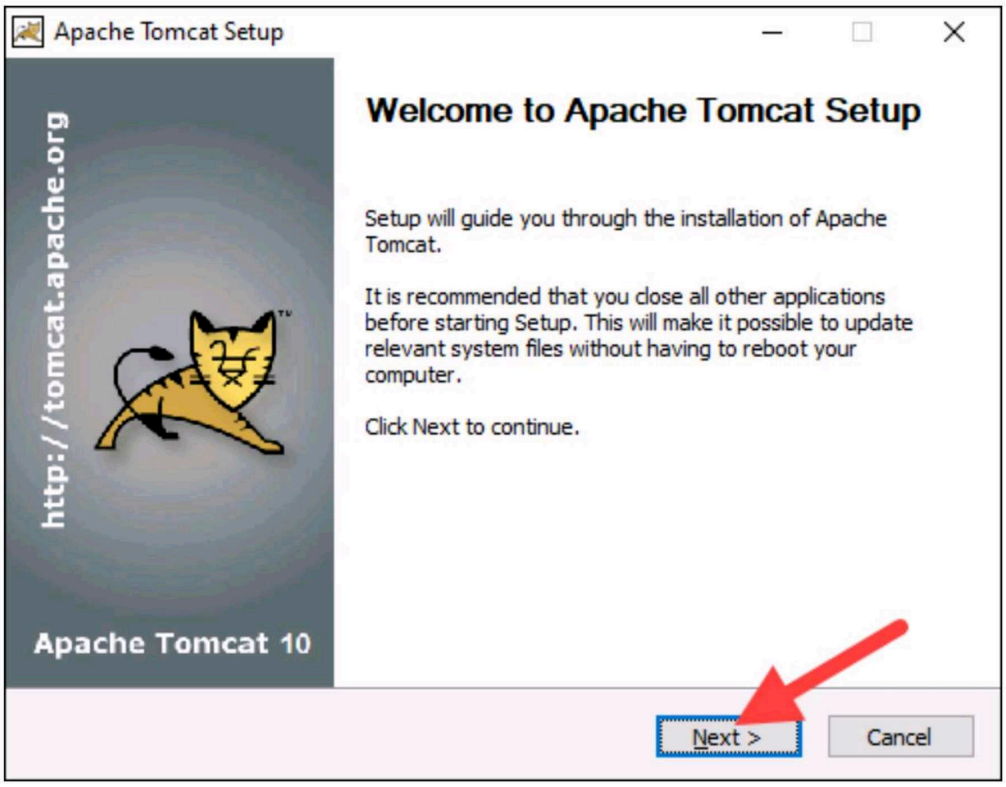
For a portable experience, install Tomcat using the **zip file** and avoid installing the service. Easily uninstall Tomcat when it is no longer needed by deleting the Tomcat directory, or move it around when necessary.
Note: Take a look at our list of 13 best Java IDEs, which help write, debug, and test Java code.

Method 1: Install Tomcat Using the Windows Service Installer

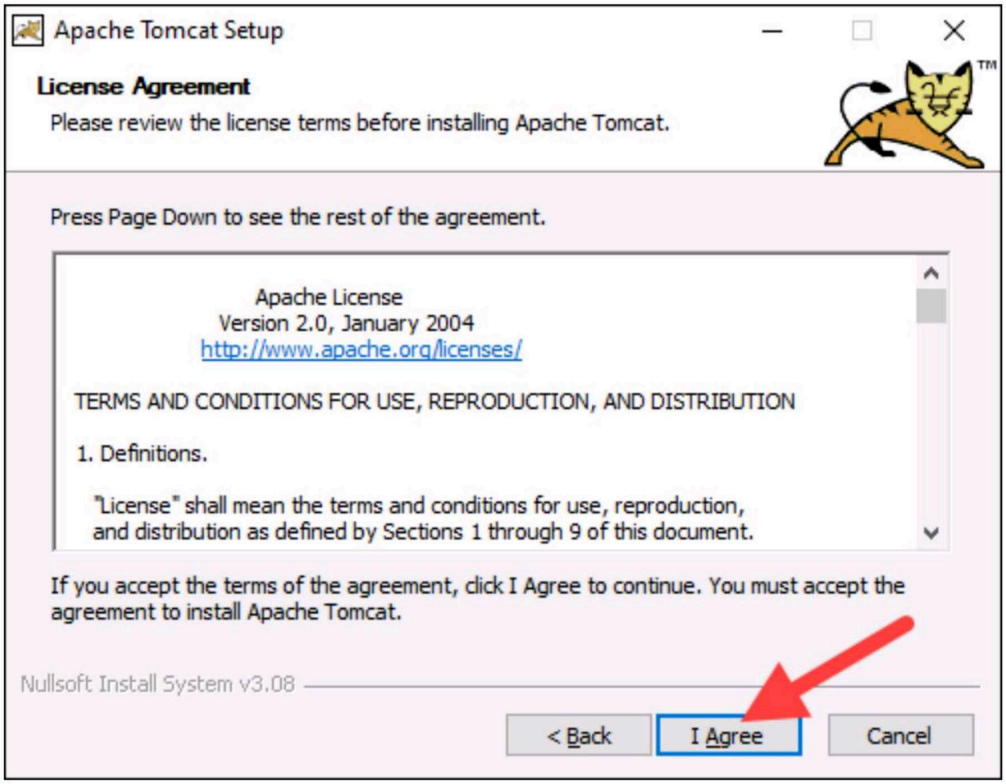
Follow the steps below to install Tomcat using the Windows Service Installer.

1. Open the downloaded **Windows Service Installer** file to start the installation process.

2. In the Tomcat Setup welcome screen, click **Next** to proceed.

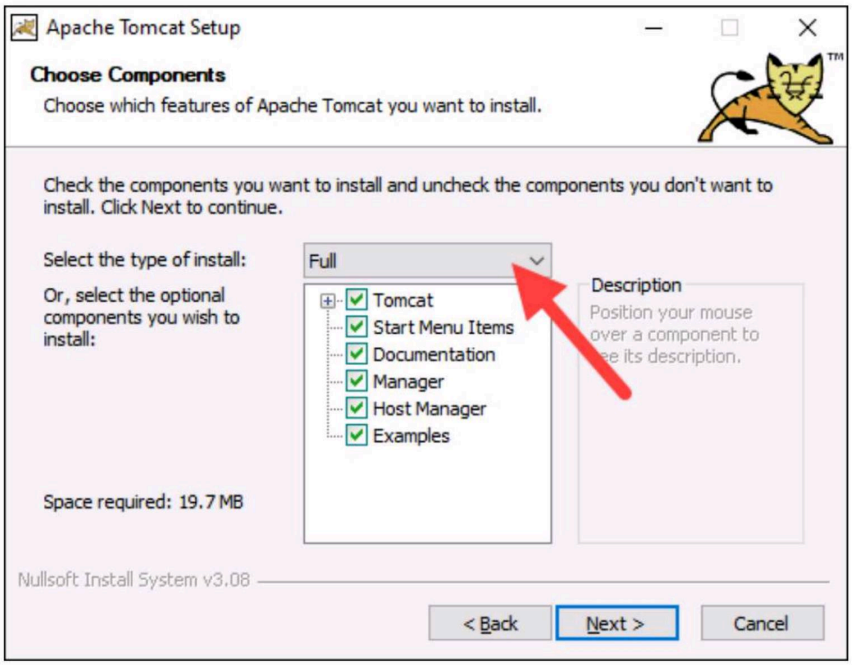


3. Read the License Agreement and if you agree to the terms, click **I Agree** to proceed to the next step.

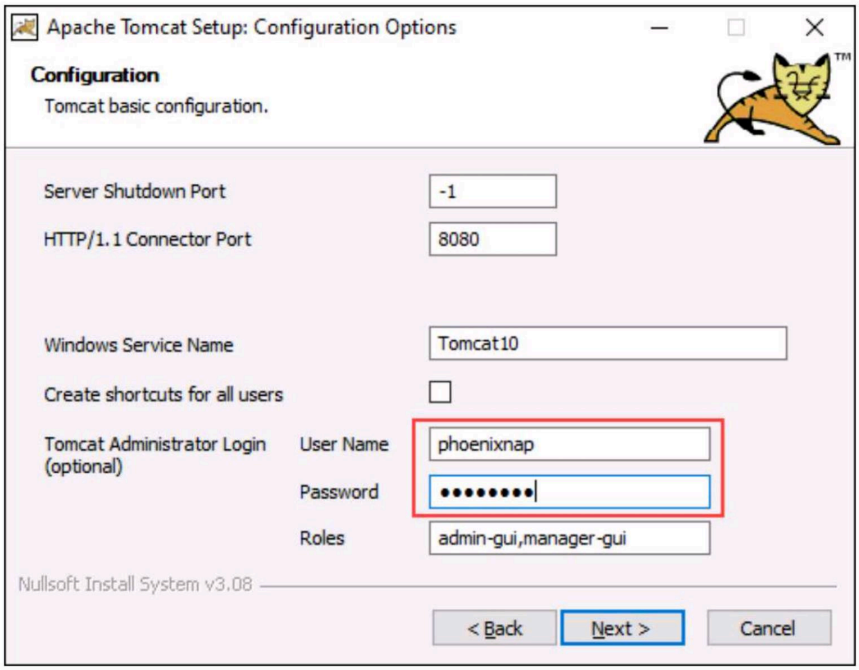


4. In the Tomcat component selection screen, choose **Full** in the dropdown menu to ensure the wizard installs the Tomcat Host Manager and Servlet and JSP examples web applications.

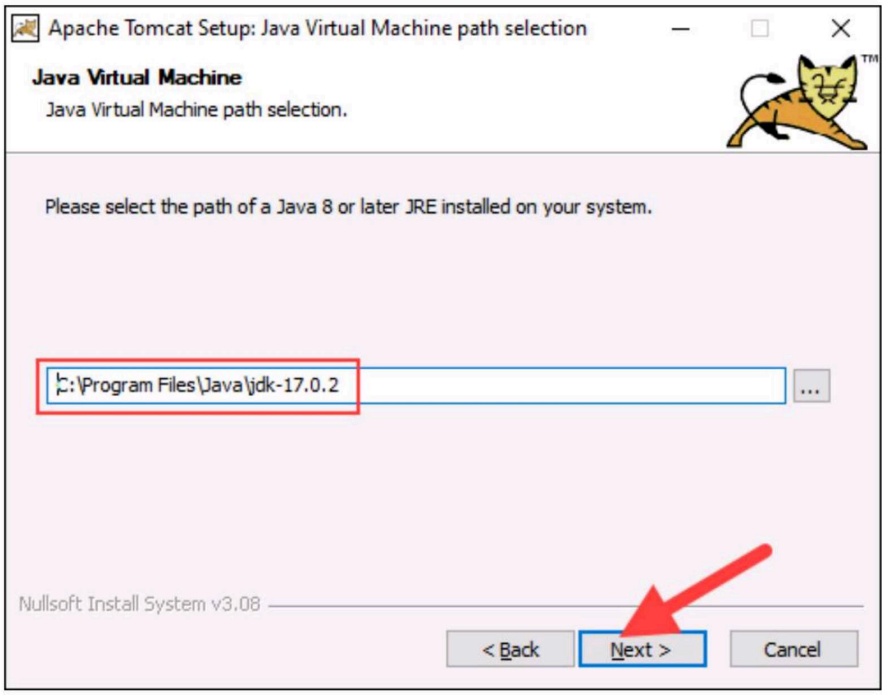
Alternatively, keep the default **Normal** installation type and click **Next**.



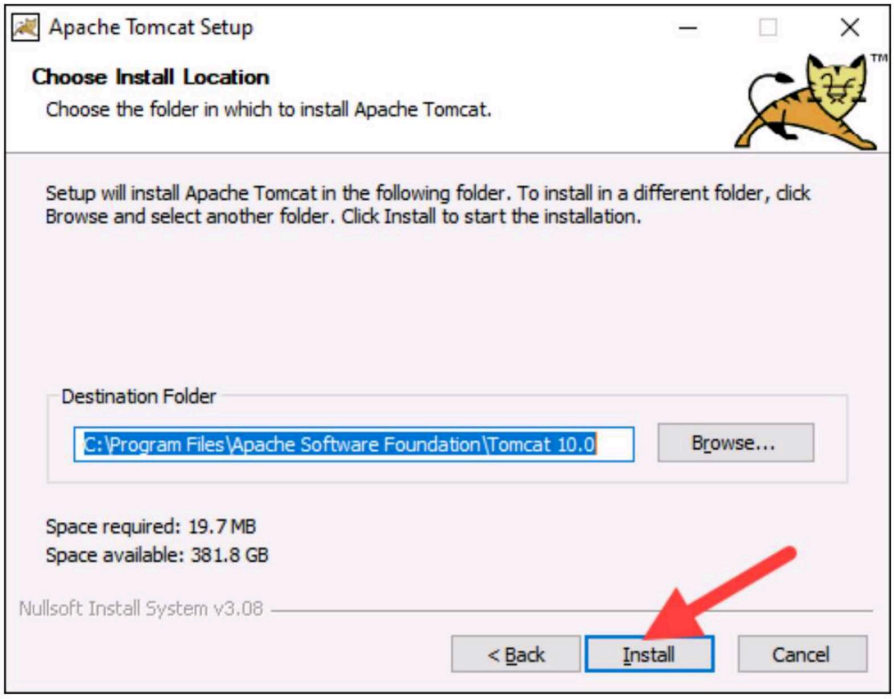
6. The next step configures the Tomcat server. For instance, enter the **Administrator login credentials** or choose a different **connection port**. When finished, click **Next** to proceed to the next step.



7. The next step requires you to enter the full path to the JRE directory on your system. The wizard auto-completes this if you have previously set up the Java environment variables. Click **Next** to proceed to the next step.

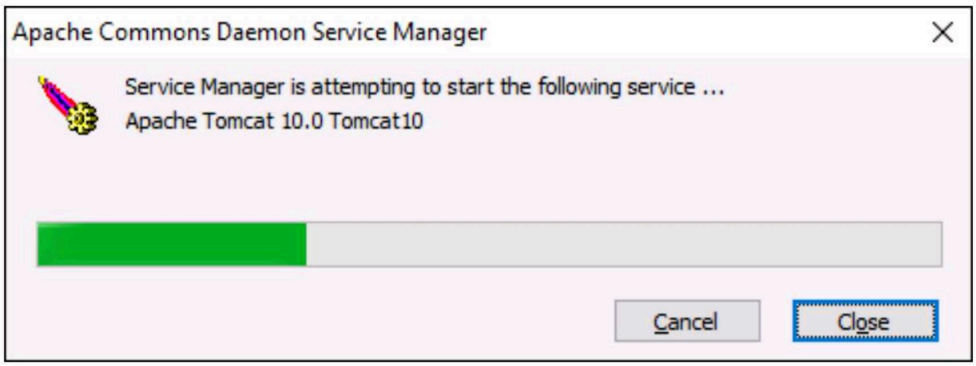


8. Choose the Tomcat server install location or keep the default one and click **Install**.



9. Check the **Run Apache Tomcat** box to start the service after the installation finishes. Optionally, check the **Show Readme** box to see the Readme file. To complete the installation, click **Finish**

10.A popup window appears that starts the Tomcat service. After the process completes, the window closes automatically. The Apache Tomcat web server is now successfully installed .



RESULT:

Thus the installation of Apache Tomcat Web Server done successfully.

EX NO: 5A DATE:	WRITE PROGRAMS IN JAVA USING SERVELTS FOR FORM INVOKING

AIM:

To Write programs in Java using Servlets

- To invoke servlets from HTML forms.

ALGORITHM:

1 : Create a Java servlet (MyServlet) extending HttpServlet.

Override the doPost method to handle form submissions.

2 : Create an HTML form (form.html) with action="MyServlet" to link to the servlet.

Include form fields (name, email, etc.) with appropriate names.

3 : Add a submit button (<input type="submit" value="Submit">) within the form.

4 : In MyServlet, use request.getParameter("parameterName") to retrieve form data.

Process the data (e.g., store in a database or perform other operations).

5 : In MyServlet, use request.getSession() to get or create a session.

Use session.setAttribute("attributeName", attributeValue) to store data in the session.

6 : Utilize session data for session tracking purposes

PROGRAM:

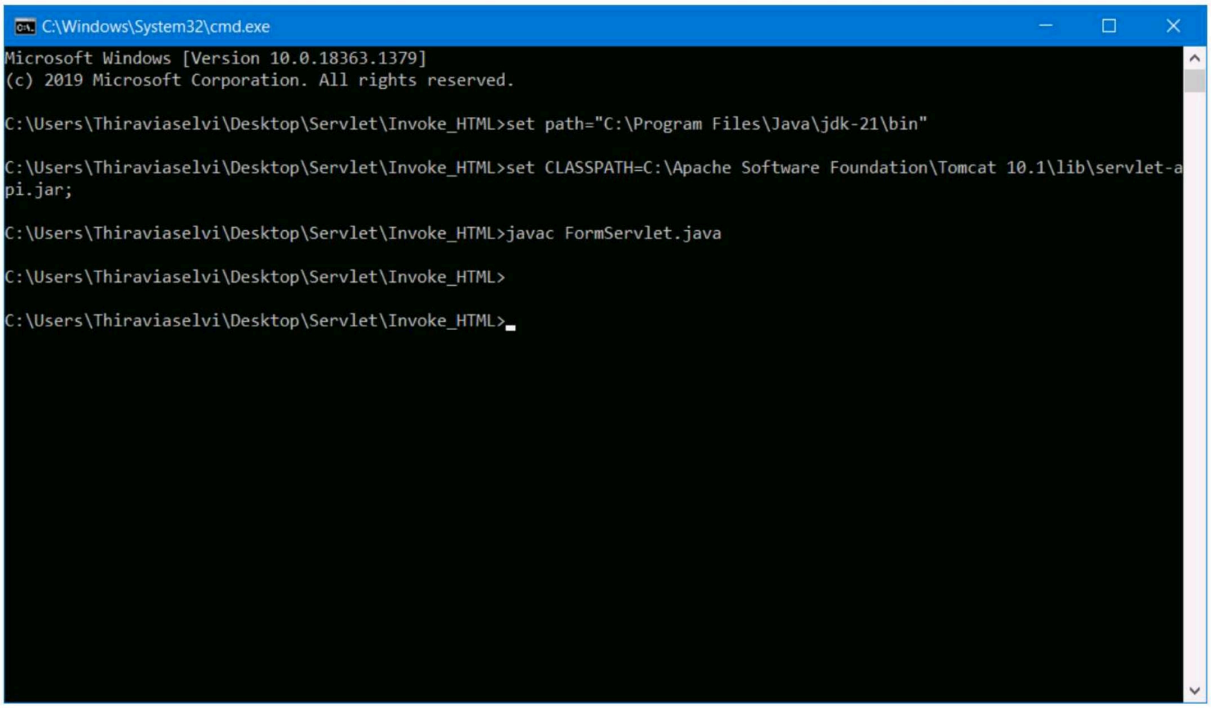
Servlet_Form.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Servlet Form Example</title>
</head>
<body>
   <form action="http://localhost:8080/FormServlet" method="post">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name" required><br><br>

      <label for="email">Email:</label>
      <input type="email" id="email" name="email" required><br><br>

      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

FormServlet.java

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class FormServlet extends HttpServlet {
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

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

      out.println("<html>");
      out.println("<head><title>Form Submission Result</title></head>");
      out.println("<body>");
      out.println("<h2>Form Submission Result</h2>");
      out.println("<p>Name: " + name + "</p>");
      out.println("<p>Email: " + email + "</p>");
      out.println("</body>");
      out.println("</html>");
   }
}
```



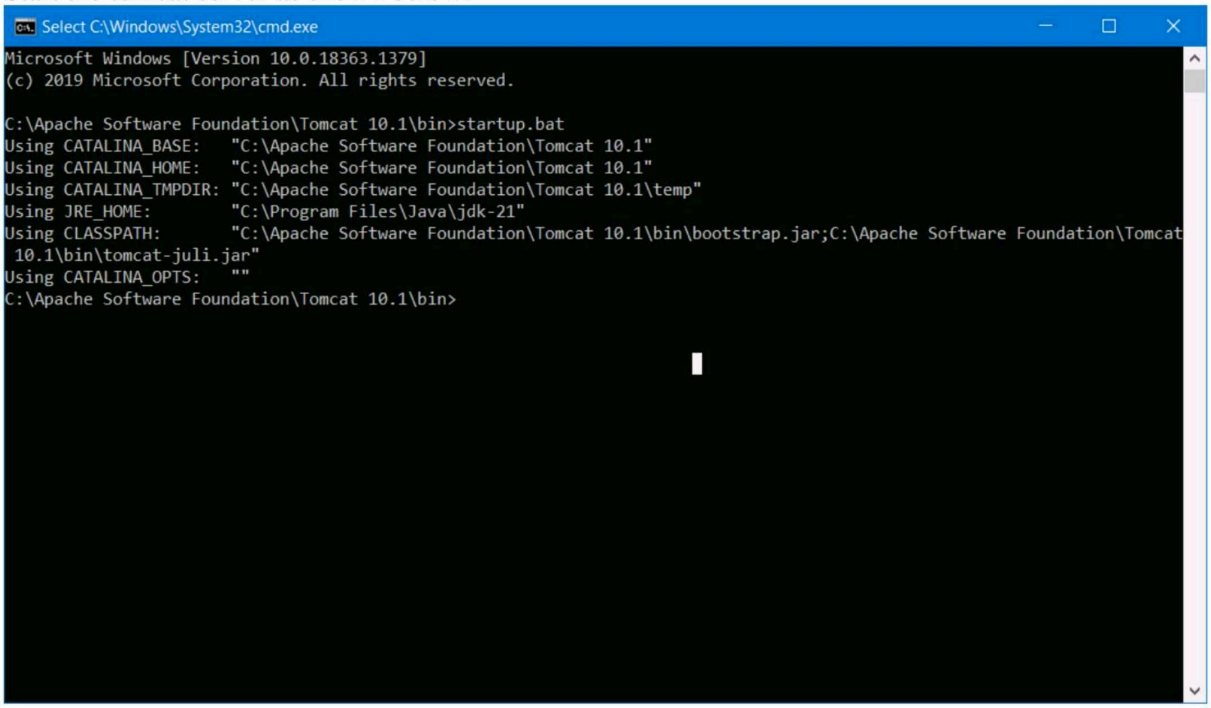
Cut and paste FormServlet.class into the location C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT\WEB-INF\classes

Edit the web.xml file in in the C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT\WEB-INF with the following code:

web.xml

```
<servlet>
<servlet-name>invoke</servlet-name>
<servlet-class>FormServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>invoke</servlet-name>
<url-pattern>/FormServlet</url-pattern>
</servlet-mapping>
```

Start the tomcat server as shown below:



Now doubleclick Servlet_Form.html

RESULT:

Thus the program for Servlets to invoke from HTML forms has written and executed successfully.

EX NO: 5 B DATE:	WRITE PROGRAMS IN JAVA USING SERVELTS FOR SESSION TRACKING

AIM:

To Write programs in Java using Servlets

- Session Tracking

ALGORITHM:

1 : Create a Java servlet (MyServlet) extending HttpServlet.

Override the doPost method to handle form submissions.

2 : Create an HTML form (form.html) with action="MyServlet" to link to the servlet.

Include form fields (name, email, etc.) with appropriate names.

3 : Add a submit button (<input type="submit" value="Submit">) within the form.

4 : In MyServlet, use request.getParameter("parameterName") to retrieve form data.

Process the data (e.g., store in a database or perform other operations).

5 : In MyServlet, use request.getSession() to get or create a session.

Use session.setAttribute("attributeName", attributeValue) to store data in the session.

6 : Utilize session data for session tracking purposes

PROGRAM:

SesionClient.html

```
<html>
<form action="http://localhost:8080/servlet1">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
</html>
```

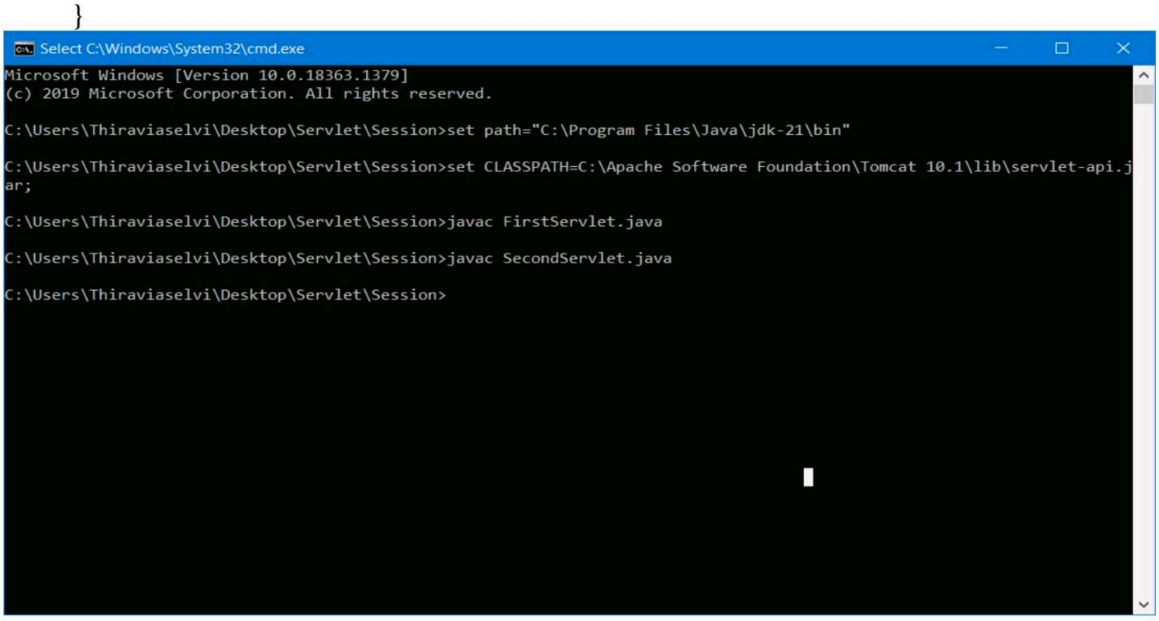
FirstServlet.java

```
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*;
public class FirstServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){
    try{
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String n=request.getParameter("userName");
    out.print("Welcome "+n);
    HttpSession session=request.getSession();
    session.setAttribute("uname",n);
    out.print("<a href='servlet2'>visit</a>");
    out.close();
```

```
    }catch(Exception e){System.out.println(e);}
  }

}
```

SecondServlet.java

```
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
public class SecondServlet extends HttpServlet
{
        public void doGet(HttpServletRequest request, HttpServletResponse response)
        {
            try
              {
              response.setContentType("text/html");
              PrintWriter out = response.getWriter();
              HttpSession session=request.getSession(false);
              String n=(String)session.getAttribute("uname");
              out.print("Hello "+n);
              out.close();
          }
              catch(Exception e)
              {
              System.out.println(e);
              }
        }
}
```



Cut and paste FirstServlet.class and SecondServlet.class into location

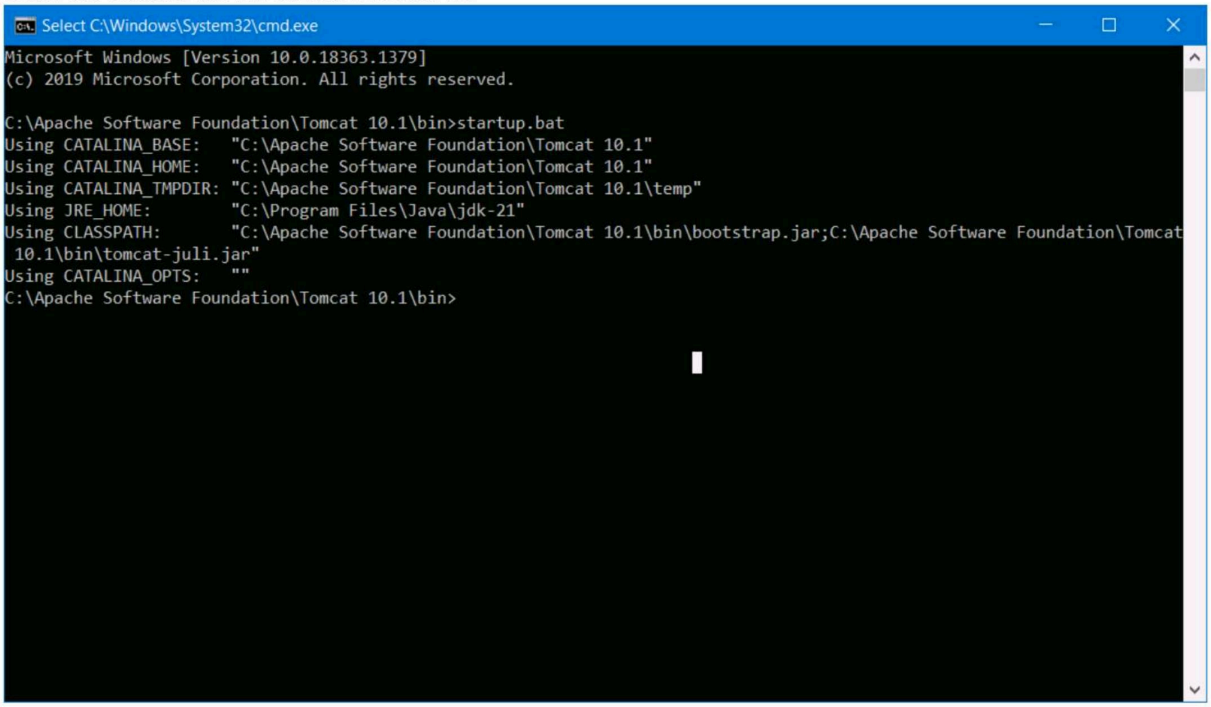
C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT\WEB-INF\classes

Edit the web.xml file in in the C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT\WEB-INF with the following code:

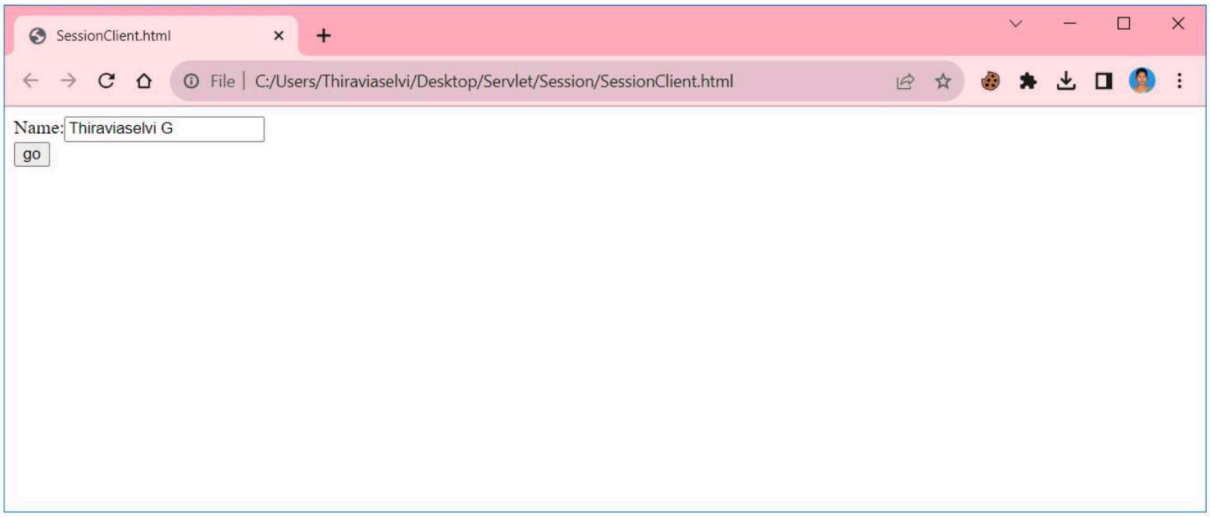
web.xml

```
<servlet>
<servlet-name>s1</servlet-name>
<servlet-class>FirstServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<servlet>
<servlet-name>s2</servlet-name>
<servlet-class>SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s2</servlet-name>
<url-pattern>/servlet2</url-pattern>
</servlet-mapping>
```

Start the tomcat server as shown below:



Now doubleclick SessionClient.html



RESULT:

Thus the program for Servlets of Session tracking has written and executed successfully.

EX NO: 6 A DATE:	WRITE PROGRAMS IN JAVA TO CREATE THREE-TIER APPLICATIONS USING JSP AND DATABASES FOR CONDUCTING ON-LINE EXAMINATION

AIM:

To Write programs in Java to create three-tier applications using JSP and Databases

- For conducting on-line examination

ALGORITHM:

1. Create Exam Interface (Presentation Layer):

Develop JSP pages to create the user interface for the online examination.

Include HTML forms, input elements, and CSS for styling.

2. User Authentication (Presentation and Business Logic Layer):

Implement user authentication using JSP and a corresponding servlet (AuthenticationServlet).

Validate user credentials against the database using a UserDAO.

3. Display Questions (Presentation and Business Logic Layer):

Fetch questions from the database based on the selected exam using an ExamDAO.

Display questions and multiple-choice options using JSP.

4. Handle User Responses (Presentation Layer):

Capture and validate user responses using JavaScript for client-side validation.

5. Submit Exam (Presentation and Business Logic Layer):

Allow users to submit their answers.

Calculate and display the result on the client side using JSP.

Process user responses and calculate the result in the ExamServlet.

6. Database Operations (Data Access Layer):

Design the database schema with tables for users, exams, questions, and user responses.

Create a UserDAO to interact with the user-related database operations (e.g., authentication).

Develop an ExamDAO to handle operations related to exams, questions, and answers.

Establish a database connection using JDBC.

7. SQL Queries (Data Access Layer):

Write SQL queries to retrieve user information, exam details, questions, and store user responses.

PROGRAM:

ExamClient17.html

```
<html>
<head>
<title>Online Exam Client</title>
<style type="text/css">
  body {background-color: black; font-family: courier; color: blue}
</style>
</head>
<body>
<h2 style="text-align: center">ONLINE EXAMINATION</h2>
<h3>Answer the following questions (5 marks for each correct answer)</h3>
<hr/>
<form name="examForm" method="post" action="http://localhost:8080/ExamServer17.jsp">
   <label for="ans1">1. Who is called as the father of the computer?</label><br/>
   <input type="radio" id="ans1" name="ans1" value="Sachin">Sachin
   <input type="radio" name="ans1" value="Stuart">Stuart
   <input type="radio" name="ans1" value="Charles_Babbage">Charles Babbage
   <input type="radio" name="ans1" value="Napier">Napier
   <br/><br/>

   <label for="ans2">2. C was developed by?</label><br/>
   <input type="radio" id="ans2" name="ans2" value="Dennis_Ritchie">Dennis Ritchie
   <input type="radio" name="ans2" value="None">None
   <input type="radio" name="ans2" value="Stroustrup">Stroustrup
   <input type="radio" name="ans2" value="John">John
   <br/><br/>

   <label for="ans3">3. C++ was developed by?</label><br/>
   <input type="radio" id="ans3" name="ans3" value="Dennis_Ritchie">Dennis Ritchie
   <input type="radio" name="ans3" value="Stroustrup">Stroustrup
   <input type="radio" name="ans3" value="David_Ritchie">David Ritchie
   <input type="radio" name="ans3" value="Charles_Babbage">Charles Babbage
   <br/><br/>

   <input type="submit" value="Check Your Result"/>
</form>
</body>
</html>
```

Place the ExamServer17.jsp in the following location C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT

ExamServer17.jsp

```
<%@ page contentType="text/html" language="java" import="java.sql.*" %>
<%@ page import="java.io.*, java.util.*" %>
<html>
<head>
```

```
    <title>Online Exam Server</title>
    <style type="text/css">
        body {
            background-color: black;
            font-family: courier;
            color: blue
        }
    </style>
</head>
<body>
<h2 style="text-align:center">ONLINE EXAMINATION</h2>
<p>
    <a href="ExamClient17.html">Back To Main Page</a>
</p>
<hr/>

<%
String str1 = request.getParameter("ans1");
String str2 = request.getParameter("ans2");
String str3 = request.getParameter("ans3");
int mark = 0;

if (str1 != null && str2 != null && str3 != null) {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        String url = "jdbc:mysql://localhost:3306/online17db";
        String username = "root";
        String password = "12345";
        Connection con = DriverManager.getConnection(url, username, password);

        PreparedStatement stmt = con.prepareStatement("SELECT * FROM online17tab");
        ResultSet rs = stmt.executeQuery();

        int i = 1;
        while (rs.next()) {
            if (i == 1 && str1.equals(rs.getString(1))) {
                mark += 5;
            }
            if (i == 2 && str2.equals(rs.getString(1))) {
                mark += 5;
            }
            if (i == 3 && str3.equals(rs.getString(1))) {
                mark += 5;
            }
            i++;
        }

        if (mark >= 10) {
            out.println("<h4>Your Mark Is : " + mark + "</h4>");
```

```
            out.println("<h3>Congratulations....! You Are Eligible For The Next
Round...</h3>");
        } else {
            out.println("<h4>Your Mark is : " + mark + "</h4>");
            out.println("<h3>Sorry....!! You Are Not Eligible For  The Next Round...</h3>");
        }

        rs.close();
        stmt.close();
        con.close();
    } catch (SQLException | ClassNotFoundException e) {
        out.println("Failed to load MySQL JDBC driver. Please check if the driver JAR is
properly placed.");

    }
} else {
    out.println("<h3>Please answer all questions before submitting.</h3>");
}
%>

</body>
</html>
```

Steps:

Step1: Keep marklist17.html and marklist17.jsp in the location C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT

Step2: Create database in Mysql:

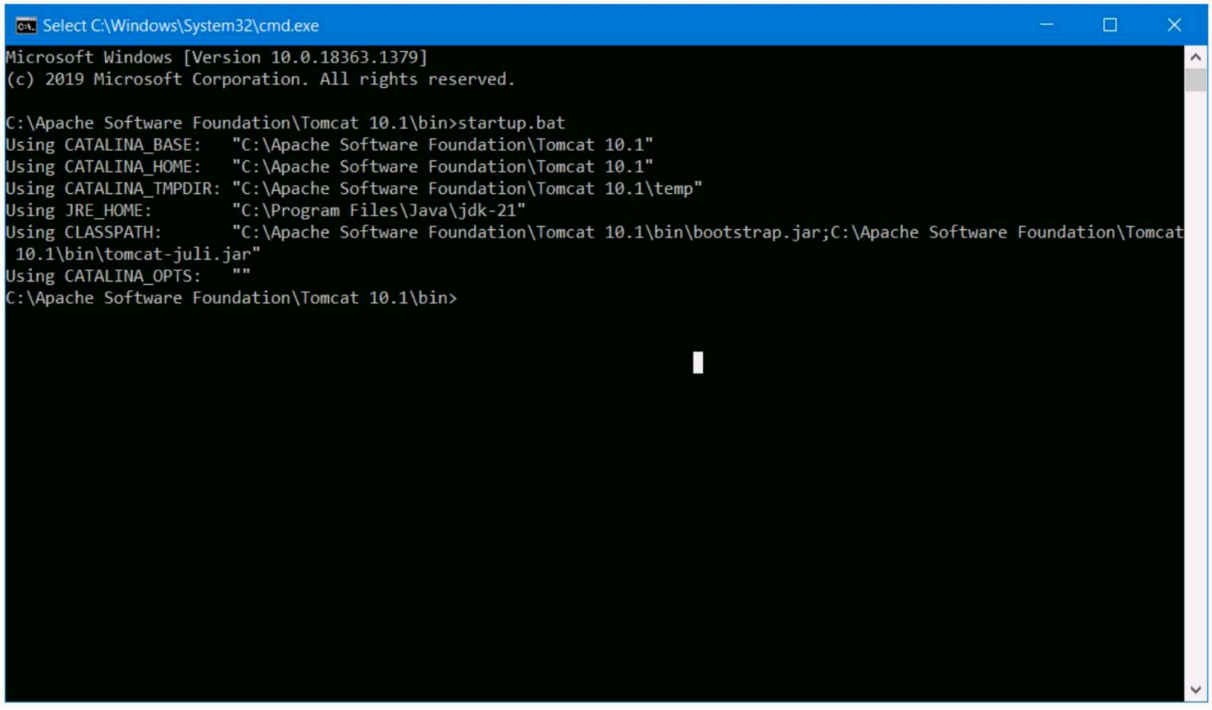
In command prompt do the following steps:

```
C:\Program Files\MySQL\MySQL Server 8.2\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database online17db;
Query OK, 1 row affected (0.44 sec)
mysql> use online17db;
Database changed
mysql> create table online17tab(answer varchar(255));
Query OK, 0 rows affected (1.70 sec)
```

```
mysql>insert into online17tab values('Charles_Babbage');
Query OK, 1 row affected (0.27 sec)
mysql>insert into online17tab values('Dennis_Ritchie');
Query OK, 1 row affected (0.27 sec)
mysql>insert into online17tab values('Stroustrup);
Query OK, 1 row affected (0.27 sec)
ysql> select * from online17tab;
+-----------------+
| answer          |
+-----------------+
| Charles_Babbage |
| Dennis_Ritchie  |
| Stroustrup      |
+-----------------+
3 rows in set (0.00 sec)

mysql> exit
Bye

C:\Program Files\MySQL\MySQL Server 8.2\bin>
```

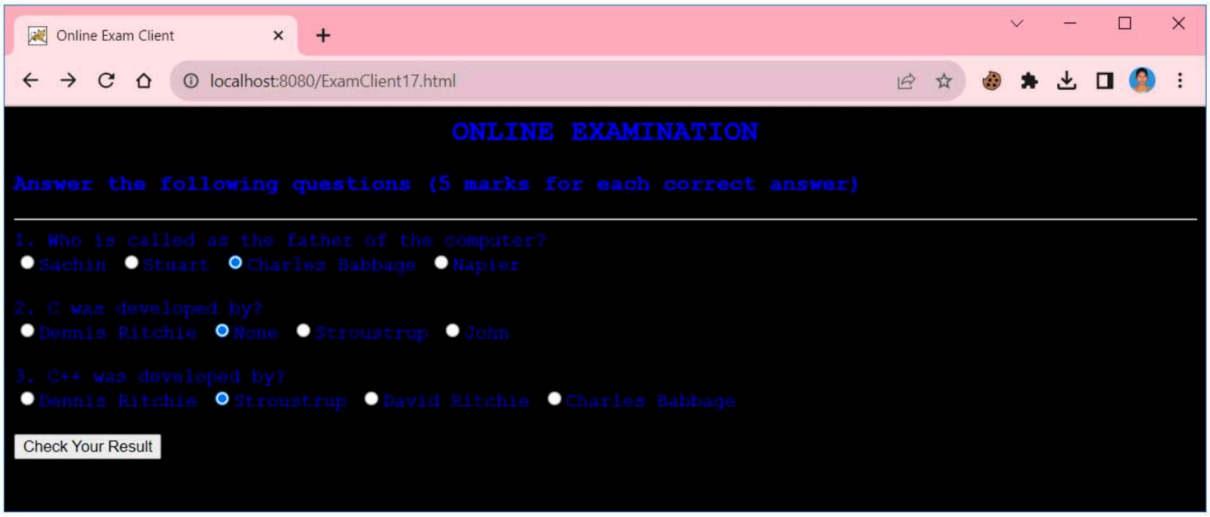
Step3:Make sure the presence of servlet-api.jar, jsp-api.jar and mysql-connector-j-8.2.0 file in C:\Apache Software Foundation\Tomcat 10.1\lib

Step4: Start the tomcat server as shown below:



Run:

Double click the ExamClient17.html file



RESULT:

Thus the java program for Servlets create three-tier applications using JSP and Databases For conducting on-line examination.has written and executed successfully.

EX NO: 6 B DATE:	WRITE PROGRAMS IN JAVA TO CREATE THREE-TIER APPLICATIONS USING JSP AND DATABASES FOR DISPLAYING STUDENT MARK LIST

AIM:

To Write programs in Java to create three-tier applications using JSP and Databases

- FOR DISPLAYING STUDENT MARK LIST

ALGORITHM:

1. **Create JSP Pages:**
   - Develop (marklistfile)**.jsp** for displaying student mark lists.
2. **User Request:**
   - Allow users to access (marklistfile).**jsp** to request the display of student mark lists.
3. **Servlet for Mark List Retrieval:**
   - Create MarkList Servlet to handle requests.
   - In the servlet:
     - Establish JDBC connection.
     - Use **DAO** for students to fetch student mark data based on criteria.
     - Set data as request attributes or use AJAX for dynamic loading.
4. **Database Operations:**
   - Design database schema for students, courses, and marks.
   - Create **DAO** for students database interactions.
   - Write SQL queries to retrieve student mark information.

PROGRAM:

marklist17.html

```
<html>
<head>
<title>Three Tier Application</title>
<style type="text/css">
  body{color:blue;font-family:courier;text-align:center}
</style>
</head>
<body>
```

```
<h2>EXAMINATION RESULT</h2><hr/>
<form name="f1" method"GET" action="http://localhost:8080/marklist17.jsp">
Enter Your Reg.No:
<input type="text" name="regno"/><br/><br/>

<input type="submit" value="SUBMIT"/>
</form>
</body>
<html>
```

Place the marklist17.jsp in the following location C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT

marklist17.jsp

```
<%@ page contentType="text/html" language="java" import="java.sql.*" %>
<html>
<head>
   <title>Three Tier Application</title>
   <style type="text/css">
      body {
         color: blue;
         font-family: courier;
         text-align: center
      }
   </style>
</head>
<body>
   <h2>EXAMINATION RESULT</h2><hr/>
   <%
   String str = request.getParameter("regno");

   // MySQL database connection details
   String jdbcURL = "jdbc:mysql://localhost:3306/markdb";
   String username = "root";
   String password = "12345";

   try {
      Connection con = DriverManager.getConnection(jdbcURL, username, password);
      PreparedStatement pstmt = con.prepareStatement("SELECT * FROM marktab WHERE rno=?");
      pstmt.setString(1, str);
      ResultSet rs = pstmt.executeQuery();

      while (rs.next()) {
   %>
         Register No: <%= rs.getObject(1) %><br/>
         Name: <%= rs.getObject(2) %><br/>
         <table border="1">
            <th>SUBJECT</th><th>Mark</th>
            <tr><td>Network Programming and Management</td><td><%= rs.getObject(3)
%></td></tr>
            <tr><td>Object Oriented Analysis and Design</td><td><%= rs.getObject(4) %></td></tr>
            <tr><td>Cryptography and Network Security</td><td><%= rs.getObject(5) %></td></tr>
            <tr><td>Embedded Systems</td><td><%= rs.getObject(6) %></td></tr>
            <tr><td>Web Technology</td><td><%= rs.getObject(7) %></td></tr>
```

```
        <tr><td>Software Requirement and Engineering</td><td><%= rs.getObject(8) %></td></tr>
          </table>
    <%
        }
        rs.close();
        pstmt.close();
        con.close();
    } catch (SQLException e) {
        out.println("Connection failed! Error: " + e.getMessage());
    }
    %>
    <br/>
    <a href="marklist17.html">Back</a>
</body>
</html>
```

Steps:

Step1: Keep marklist17.html and marklist17.jsp in the location C:\Apache Software Foundation\Tomcat 10.1\webapps\ROOT

Step2: Create database in Mysql:

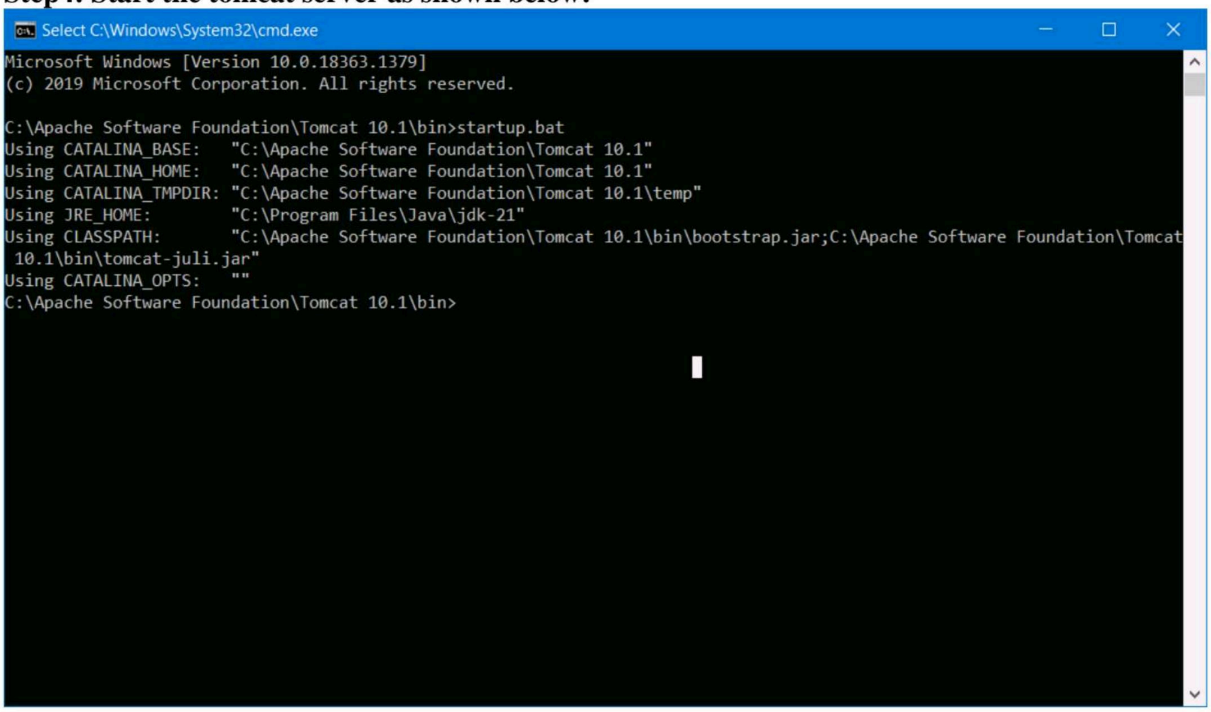
In command prompt do the following steps:

```
C:\Program Files\MySQL\MySQL Server 8.2\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database markdb;
Query OK, 1 row affected (0.44 sec)
mysql> use markdb;
Database changed
mysql> create table marktab(rno int, fname varchar(255), mark1 int, mark2 int, mark3 int, mark4 int,
mark5 int, mark6 int);
Query OK, 0 rows affected (1.70 sec)
mysql> insert into marktab values(10,'aaa',90,90,90,90,90,90);
Query OK, 1 row affected (0.27 sec)
mysql> insert into marktab values(20,'bbb',80,80,80,80,80,80);
Query OK, 1 row affected (0.23 sec)

mysql> select * from marktab;
+------+-------+-------+-------+-------+-------+-------+-------+
| rno  | fname | mark1 | mark2 | mark3 | mark4 | mark5 | mark6 |
```

```
+------+-------+-------+-------+-------+-------+-------+-------+
|  10 | aaa   |    90 |    90 |    90 |    90 |    90 |    90 |
|  20 | bbb   |    80 |    80 |    80 |    80 |    80 |    80 |
+------+-------+-------+-------+-------+-------+-------+-------+

2 rows in set (0.00 sec)
mysql> exit
Bye

C:\Program Files\MySQL\MySQL Server 8.2\bin>
```

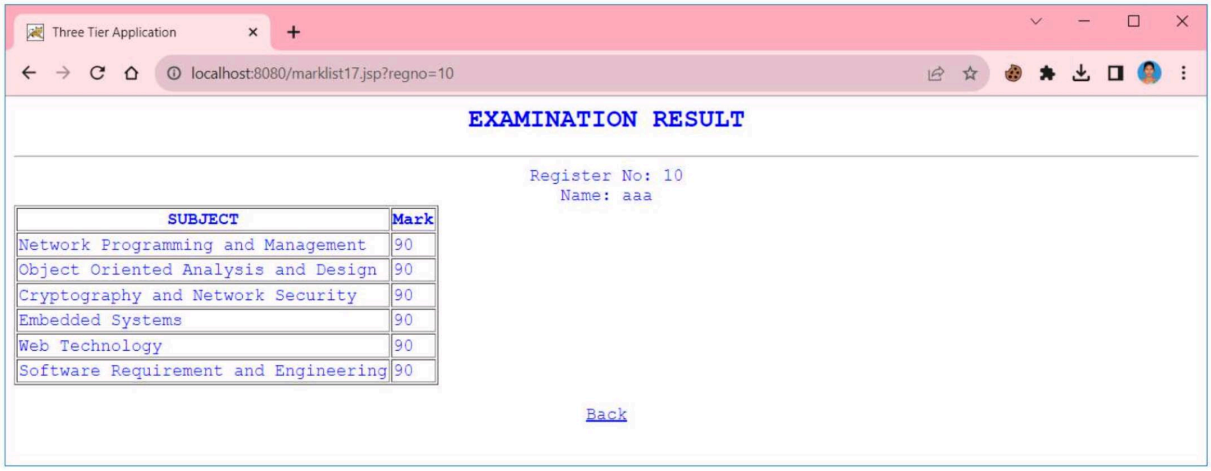
Step3: Make sure the presence of servlet-api.jar, jsp-api.jar and mysql-connector-j-8.2.0 file in C:\Apache Software Foundation\Tomcat 10.1\lib

Step4: Start the tomcat server as shown below:



Run:

Double click the marklist17.html file



RESULT:

Thus the java program for Servlets create three-tier applications using JSP and Databases has for displaying student mark list written and executed successfully.

EX NO: 7 A DATE:	WRITE PROGRAM USING XML – SCHEMA – XSL

AIM:

To write a program using xml – schema – xsl

ALGORITHM:

1. **Create XML Document:**
   - Develop an XML document containing structured data.
   - Include elements and attributes representing the data.
2. **Design XML Schema:**
   - Create an XML Schema Definition (XSD) file to define the structure and data types of the XML document.
   - Specify elements, attributes, and their relationships in the schema.
3. **Validate XML Against Schema:**
   - Implement a program or script (e.g., in Java) to validate the XML document against the XML Schema.
   - Use XML parsing libraries to check conformance.
4. **Create XSLT Stylesheet:**
   - Develop an XSLT stylesheet to define how the XML data should be transformed into another format (e.g., HTML).
   - Specify rules for matching and transforming elements.
5. **Process XML with XSLT:**
   - Implement a program or script (e.g., using an XSLT processor or Java) to apply the XSLT stylesheet to the XML document.
   - Transform the XML data into the desired format (e.g., HTML).
6. **Generate Output:**
   - Output the transformed data to a file or display it in a web browser.
   - The result should reflect the structure and content defined in the XSLT stylesheet

PROGRAM:

Books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
  <book>
```

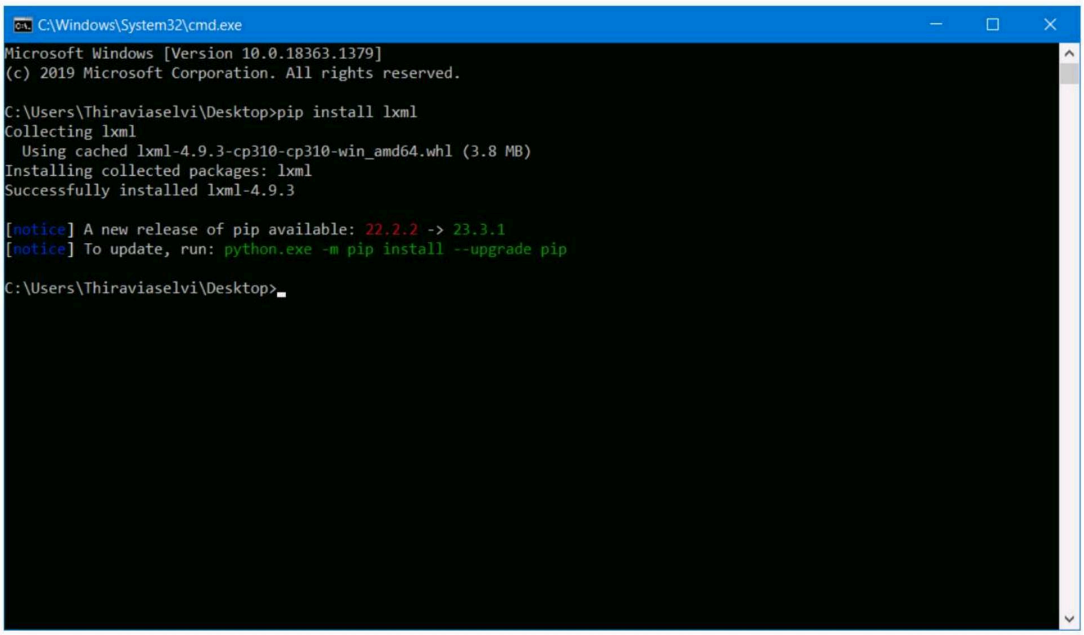
```
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian fiction</genre>
    <publication_date>1949</publication_date>
  </book>
  <book>
    <title>To Kill a Mockingbird</title>
    <author>Harper Lee</author>
    <genre>Novel</genre>
    <publication_date>1960</publication_date>
  </book>
</library>
```

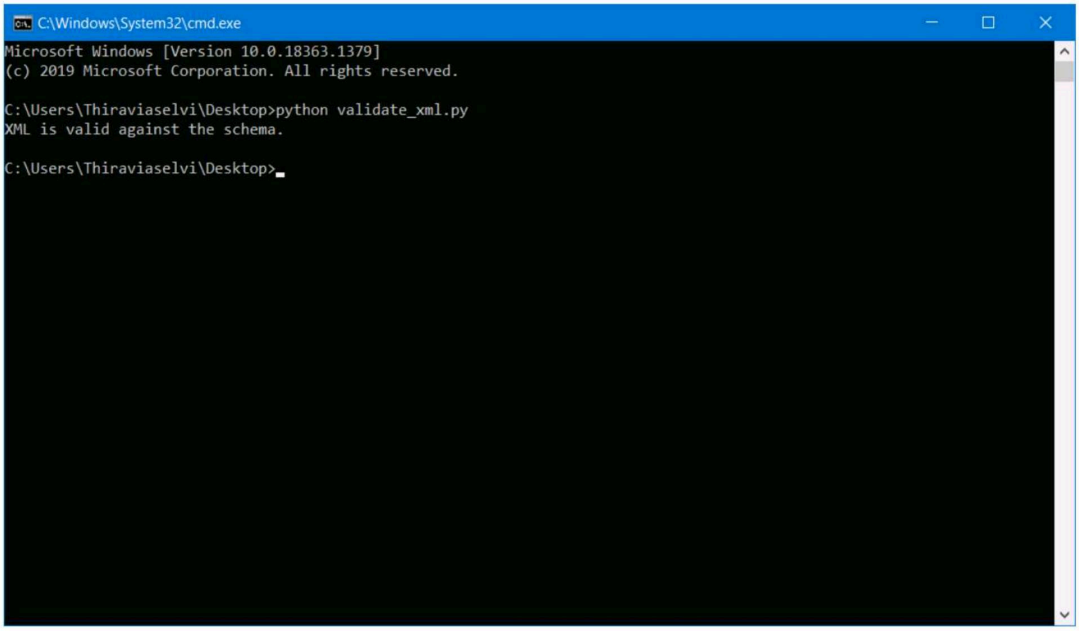
Books.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="library">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="book" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="title" type="xs:string"/>
              <xs:element name="author" type="xs:string"/>
              <xs:element name="genre" type="xs:string"/>
              <xs:element name="publication_date" type="xs:integer"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```

validate_xml.py

```
from lxml import etree
xml_file = 'Books.xml'
xsd_file = 'Books.xsd'
def validate_xml(xml_file, xsd_file):
    xmlschema_doc = etree.parse(xsd_file)
    xmlschema = etree.XMLSchema(xmlschema_doc)
    xml_doc = etree.parse(xml_file)
    if xmlschema.validate(xml_doc):
        print("XML is valid against the schema.")
    else:
        print("XML is not valid against the schema.")
  print(xmlschema.error_log)
validate_xml(xml_file, xsd_file)
```





RESULT:

Thus the program of schema – xsl is written and executed successfully using xml.

EX NO: 7B DATE:	WRITE PROGRAM USING XML – SCHEMA – XSLT

AIM:

To write a program using xml – schema – xslt

ALGORITHM:

1.Create XML Document:

- Develop an XML document containing structured data.
- Include elements and attributes representing the data.

2.Design XML Schema:

- Create an XML Schema Definition (XSD) file to define the structure and data types of the XML document.
- Specify elements, attributes, and their relationships in the schema.

3.Validate XML Against Schema:

- Implement a program or script (e.g., in Java) to validate the XML document against the XML Schema.
- Use XML parsing libraries to check conformance.

4.Create XSLT Stylesheet:

- Develop an XSLT stylesheet to define how the XML data should be transformed into another format (e.g., HTML).
- Specify rules for matching and transforming elements.

5.Process XML with XSLT:

- Implement a program or script (e.g., using an XSLT processor or Java) to apply the XSLT stylesheet to the XML document.
- Transform the XML data into the desired format (e.g., HTML).

6.Generate Output:

- Output the transformed data to a file or display it in a web browser.
- The result should reflect the structure and content defined in the XSLT stylesheet

PROGRAM:

input.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
  <book>
    <title>1984</title>
    <author>George Orwell</author>
    <genre>Dystopian fiction</genre>
```

```
    <publication_date>1949</publication_date>
  </book>
  <book>
    <title>To Kill a Mockingbird</title>
    <author>Harper Lee</author>
    <genre>Novel</genre>
    <publication_date>1960</publication_date>
  </book>
</library>
```

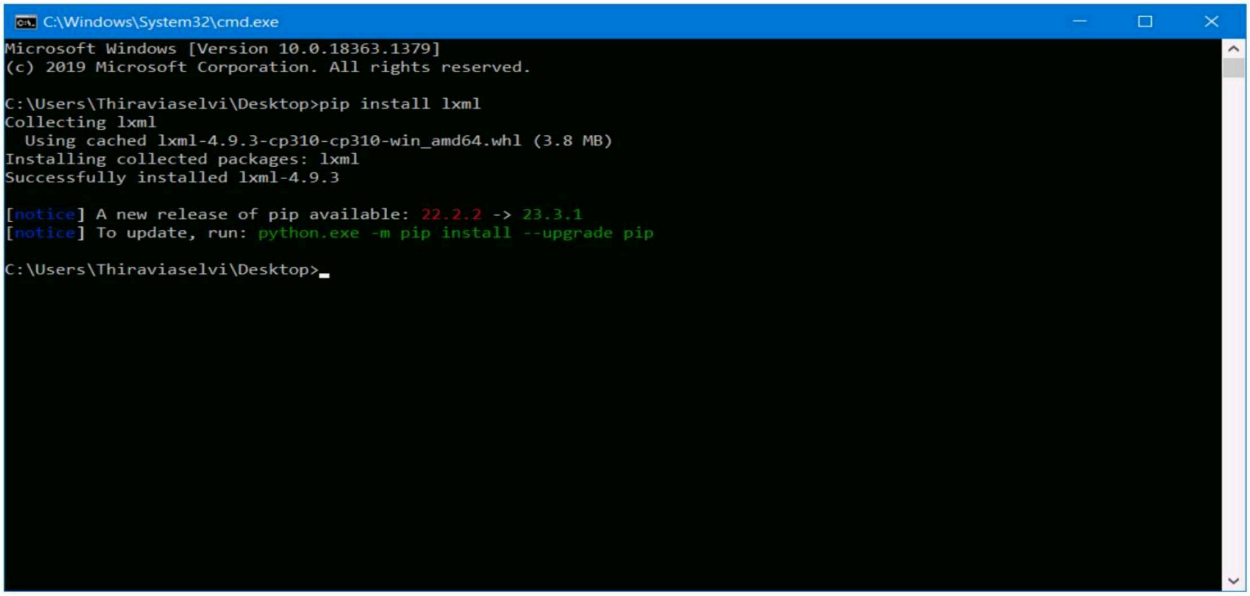
transform.xslt

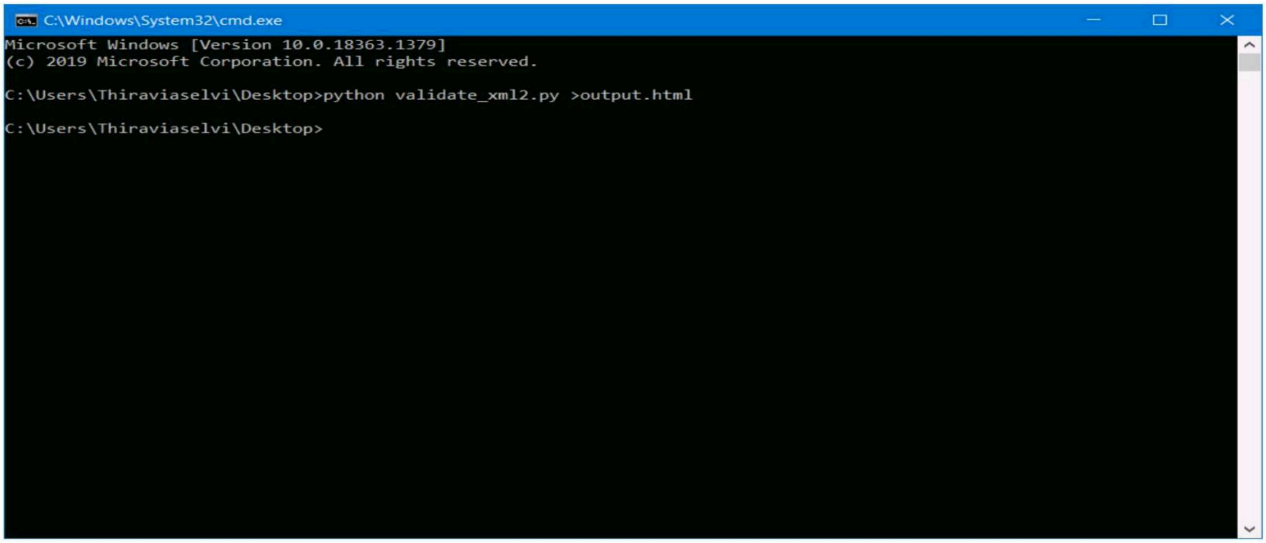
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
    <html>
      <head>
        <title>Book List</title>
      </head>
      <body>
        <h1>Book List</h1>
        <table border="1">
          <tr>
            <th>Title</th>
            <th>Author</th>
            <th>Genre</th>
            <th>Publication Date</th>
          </tr>
          <xsl:for-each select="library/book">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="author"/></td>
              <td><xsl:value-of select="genre"/></td>
              <td><xsl:value-of select="publication_date"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>
```

validate_xml2.py

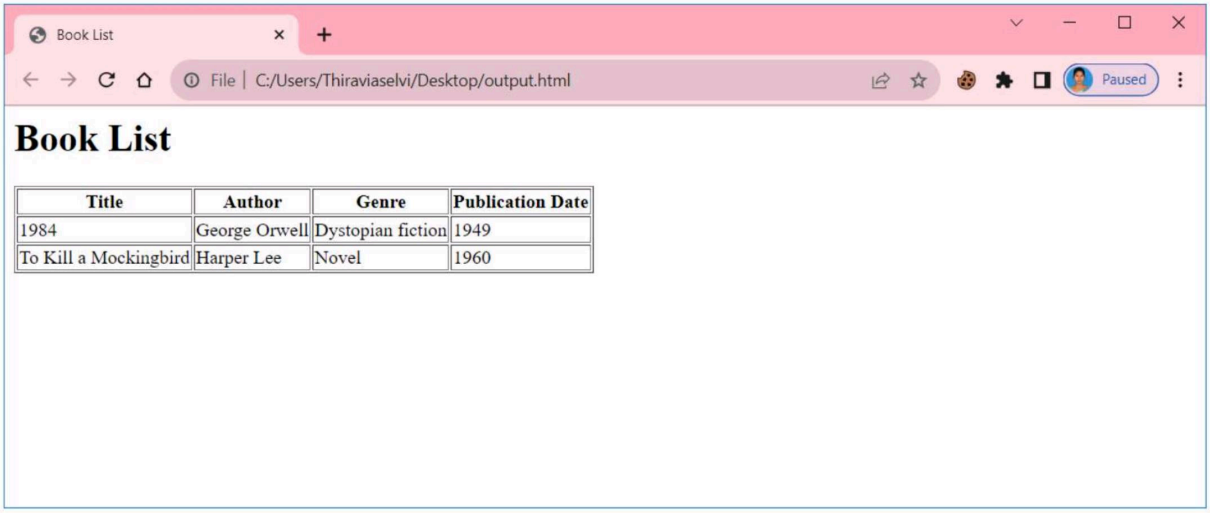
```
from lxml import tree

# Load the XML and XSLT files
xml_file = 'input.xml'
xslt_file = 'transform.xslt'
dom = etree.parse(xml_file)
xslt = etree.parse(xslt_file)

# Create the XSLT transformer
transform = etree.XSLT(xslt)
```

```
# Apply the transformation
result = transform(dom)

# Output the transformed result (HTML)
print(str(result))
```





output.html



RESULT:

Thus the program of schema – xslt is written and executed successfully using xml.