

EXNO:01**CREATE THE FOLLOWING USING HTML**

- DATE:**
- a) To embed an image map in a web page
 - b) To fix the hotspots
 - c) Show all the related information when the hot spots are clicked

AIM:

To create a web page using html to embed an image map, fix the hotspots and show all related information when the hotspots are clicked.

ALGORITHM:

Step 1 : Create a html file with map tag.

Step 2: Set the source attribute of the to the location of the image and also set the use map attribute.

Step 3: Specify an area with name, shape and <href> set of the appropriate value.

Step 4: Repeat step3 as many hot spots you want to put in the map.

Step 5: Create html file for each and every hot spots the user will select the particular location it shows information about it.

PROGRAM:

```
<html>
<head>
<title>India Map</title>
</head>
<body bgcolor= "PINK">
<font face = "Monotype Corsiva" color = "Blue" size = "6">
<marquee direction = "left" behavior= "alternate"> INDIAMAP
</marquee>
</font>
<map name= "pagemap">
<area shape="rect" coords="194,151,247,219" href="map1.html">
<area shape="rect" coords="291,268,384,337" href="map2.html">
<area shape="rect" coords="100,337,197,384" href="map3.html">
<area shape="rect" coords="236,543,344,577" href="map4.html">
</map>

</body>
<font color="#ff0000" size="5">
<p><b>Hints:</b><i>Click on the Name of the Cities in the map to now its
description</i></p>
</html>
```

/*map1.html*/

```

<html>
<body bgcolor="SKYBLUE">
<font face="Monotype Corsiva" size="18" color="RED">
<center><b><i><tt>Delhi is the capital of our INDIA<br> and <br>More IT companies
are Camped at Delhi</tt></i></b></center>
<a href="mapping.html">Home Page</a>
</font>
</body>
</html>

```

/*map2.html*/

```

<html>
<body bgcolor="SKYBLUE">
<font face="Times New Roman" size="18" color="RED">
<center><b><i>Calcutta is the wealthy city in WEST BENGAL<br> and <br>it has
Famous "Sunderbans Forests"</i></b></center>
<a href="mapping.html">Home Page</a>
</font>
</body>
</html>

```

/*map3.html*/

```

<html>
<body bgcolor="DARKGREEN">
<font face="Times New Roman" size="12" color="RED">
<center>MUMBAI is the capital of Maharashtra<br> and <br>it has Famous India
Gate</center>
<a href="mapping.html">Home Page</a>
</font>
</body>
</html>

```

/*map4.html*/

```

<html>
<body bgcolor="BLACK">
<font face="Times New Roman" size="12" color="RED">
<center>Chennai is hte capital of Tamil Nadu<br> and <br>More IT companies are
camped at Chennai</center> <a href="mapping.html">Home Page</a>
</font>
</body>
</html>

```

OUTPUT:

The map and area elements

Click on the TAMILNADU ANDHRA PRADESH ANDHRA PRADESH



TAMILNADU

TOURIST SPOTS

brihadeeswara temple,octy,meenakshi temple,marina beach



RESULT:

Thus the program to Create a web page using HTML: To embed an image map in a web page, to fix the hotspots and to show all the related information when the hot spots are clicked has been implemented and executed successfully.

EXNO:02 CREATE A WEB PAGE WITH ALL TYPES OF CASCADING STYLE

DATE: SHEETS USING RWD

AIM:

To create a web page with all types of Cascading style sheets using Reactive Web Design(RWD).

ALGORITHM:

Step 1: Start.

Step 2: There are 3 ways to insert cascading style sheets.

- 1) Inline Style Sheet
- 2) Embedded Style Sheet and
- 3) External Style sheet.

Step 3: To use inline style in a html page, style attribute must be inserted to the relevant tag.

Step 4: The style attribute can contain any CSS property.

Step 5: Syntax:

<p style="color : red">Inline text</p>

Step 6: Embedded style sheet should be used when a single document has a unique style.

Step 7: Styles are created under style tags.

Step 8: Embedded style sheets are inserted in head section of the html page.

Step 9: Syntax:

```
<head>  
<style>  
    body {background-color: powderblue;}  
    h1 {color:blue;}  
    p {color:red;}  
</style>  
</head>
```

Step 10: An external style sheet is used when the style is to be applied for many pages.

Step 11: With an external style sheet, the look of an entire Web site can be changed by changing one file.

Step 12: Each page must link to the style sheet using the tag.

Step 13: The <link> tag goes inside the head section.

Step 14: Syntax:

```
<head>  
    <link rel="stylesheet" href="style.css">  
</head>
```

Step 15: Execute the index.html file

Step 16: Stop.

PROGRAM:

index.html:

```
<!DOCTYPE html>  
<html>  
<head>
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
* {
  box-sizing: border-box;
}

.row::after {
  content: "";
  clear: both;
  display: table;
}

[class*="col-"] {
  float: left;
  padding: 15px;
}

html {
  font-family: "Lucida Sans", sans-serif;
}

.header {
  background-color: #9933cc;
  color: #ffffff;
  padding: 15px;
}

.menu ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

.menu li {
  padding: 8px;
  margin-bottom: 7px;
  background-color: #33b5e5;
  color: #ffffff;
  box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.menu li:hover {
  background-color: #0099cc;
}

.aside {
  background-color: #33b5e5;
  padding: 15px;
  color: #ffffff;
  text-align: center;
}

```

```

    font-size: 14px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24);
}

.footer {
    background-color: #0099cc;
    color: #ffffff;
    text-align: center;
    font-size: 12px;
    padding: 15px;
}

/* For mobile phones: */
[class*="col-"] {
    width: 100%;
}

@media only screen and (min-width: 600px) {
    /* For tablets: */
    .col-s-1 {width: 8.33%;}
    .col-s-2 {width: 16.66%;}
    .col-s-3 {width: 25%;}
    .col-s-4 {width: 33.33%;}
    .col-s-5 {width: 41.66%;}
    .col-s-6 {width: 50%;}
    .col-s-7 {width: 58.33%;}
    .col-s-8 {width: 66.66%;}
    .col-s-9 {width: 75%;}
    .col-s-10 {width: 83.33%;}
    .col-s-11 {width: 91.66%;}
    .col-s-12 {width: 100%;}
}

@media only screen and (min-width: 768px) {
    /* For desktop: */
    .col-1 {width: 8.33%;}
    .col-2 {width: 16.66%;}
    .col-3 {width: 25%;}
    .col-4 {width: 33.33%;}
    .col-5 {width: 41.66%;}
    .col-6 {width: 50%;}
    .col-7 {width: 58.33%;}
    .col-8 {width: 66.66%;}
    .col-9 {width: 75%;}
    .col-10 {width: 83.33%;}
    .col-11 {width: 91.66%;}
    .col-12 {width: 100%;}
}
</style>
</head>
<body>

```

```

<div class="header">
  <h1>Chania</h1>
</div>

<div class="row">
  <div class="col-3 col-s-3 menu">
    <ul>
      <li>The Flight</li>
      <li>The City</li>
      <li>The Island</li>
      <li>The Food</li>
    </ul>
  </div>

  <div class="col-6 col-s-9">
    <h1>The City</h1>
    <p>Chania is the capital of the Chania region on the island of Crete. The city can be divided
in two parts, the old town and the modern city.</p>
  </div>

  <div class="col-3 col-s-12">
    <div class="aside">
      <h2>What?</h2>
      <p>Chania is a city on the island of Crete.</p>
      <h2>Where?</h2>
      <p>Crete is a Greek island in the Mediterranean Sea.</p>
      <h2>How?</h2>
      <p>You can reach Chania airport from all over Europe.</p>
    </div>
  </div>
</div>

<div class="footer">
  <p>Resize the browser window to see how the content respond to the resizing.</p>
</div>

</body>
</html>

```

OUTPUT:



If the size increases, the output will be like:



RESULT:

Thus the program to create a web page with all types of Cascading style sheets using RWD has been implemented and executed successfully.

EXNO:03 CLIENT SIDE SCRIPTS FOR VALIDATING WEB FORM CONTROLS

DATE: USING DHTML

AIM:

To write the HTML codes using JAVA Script and CSS to create Client Side Scripts for Validating Web Form Controls.

ALGORITHM:

Step 1 : Open the notepad.

Step 2 : Create the HTML header and give the title as “WebForm”.

Step 3 : Specify the style type as “text/css”.

Step 4 : Define the properties for the Layout and Layer.

Step 5 : Specify the script type as “text/javascript”.

Step 6 : Create a function as “validate()”.

Step 7 : Specify the conditions for the form to be created using javascript.

Step 8 : Open the body of the HTML.

Step 9 : Create the division as “Layer”.

Step 10: Create a form and name it. Create a table for better alignment of the contents of the form.

Step11: Specify the requirements of the form like Name, Qualification, Date Of Birth, etc.

Step 12: Define the input type, name and id of all the required parameters.

Step 13: Create a “Submit” button and make a link to the function using onclick=“validate()”.

Step 14: Save and run the HTML file to get the output.

PROGRAM:

webform.html:

```
<html>
<head>
<title>WEB FORM</title>
<style type="text/css">
#Layer1 {
position:absolute;
width:1047px;
height:792px;
z-index:1;
left: 103px;
```

```

top: 80px;
}
.style1 {
color: #000099;
font-weight: bold;
background-color: #e06377;
}
.style2 {
color: #000099;
font-weight: bold;
background-color: #e06377;
}
#Layer2 {
position: absolute;
width: 1040px;
height: 49px;
z-index: 2;
left: 59px;
top: 9px;
}
.style3 {
color: #622569;
font-weight: bold;
background-color: #80ced6;
}

body {
background-color: #DADADA;
}
</style>
<script type="text/javascript">
function validation()

```

```
{
if(document.validate.name.value=="")
{
alert("enter the full name")
document.validate.name.focus();
return false
}
if(!isNaN(document.validate.name.value))
{
alert("enter the valid name");
document.validate.name.focus();
document.validate.name.value="";
return false
}
if(document.validate.day.value=="0")
{
alert("enter the day")
document.validate.day.focus();
return false
}
if(document.validate.month.value=="0")
{
alert("enter the month")
document.validate.month.focus();
return false
}
if(document.validate.year.value=="0")
{
alert("enter the year")
document.validate.year.focus();
return false
}
```

```
if(document.validate.gender.value=="0")
{
alert("enter the gender")
document.validate.gender.focus();
return false
}
if(document.validate.address.value=="")
{
alert("enter the address")
document.validate.address.focus();
return false
}
if(document.validate.city.value=="")
{
alert("enter the city")
document.validate.city.focus();
return false
}
if(!isNaN(document.validate.city.value))
{
alert("enter the valid City")
document.validate.city.focus();
document.validate.city.value="";
return false
}
if(document.validate.state.value=="")
{
alert("enter the state")
document.validate.state.focus();
return false
}
if(!isNaN(document.validate.state.value))
```

```

{
alert("enter the valid state")
document.validate.state.focus();
document.validate.state.value="";
return false
}
if(document.validate.pincode.value=="")
{
alert("enter the pincode ")
document.validate.pincode.focus();
return false
}
if( isNaN(document.validate.pincode.value))
{
alert("enter a valid pincode")
document.validate.pincode.focus();
return false
}
if(document.validate.pincode.value.length!=6)
{
alert("please enter the valid pincode")
document.validate.pincode.focus();
return false
}
if(document.validate.address.value=="")
{
alert("enter the address")
document.validate.address.focus();
return false
}
if(document.validate.email.value=="")
{

```

```

alert("enter the email id")
document.validate.email.focus();
return false
}
if(!(document.validate.email.value==""))
{
var x=document.validate.email.value
var atpos=x.indexOf("@");
var dotpos=x.lastIndexOf(".");
if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)
{
alert("Not a valid e-mail address");
document.validate.email.focus();
return false;
}
}
if(document.validate.mobile.value=="")
{
alert("enter the mobile no")
document.validate.mobile.focus();
return false
}
if( isNaN(document.validate.mobile.value))
{
alert("enter a valid mobile no")
document.validate.mobile.focus();
document.validate.mobile.value="";
return false
}
if(document.validate.mobile.value.length !=10)
{
alert("enter the valid mobile no")

```

```

document.validate.mobile.focus();
document.validate.mobile.select();
return false
}
if(document.validate.qualification.value=="0")
{
alert("enter the qualification")
document.validate.qualification.focus();
return false
}
if(document.validate.experience.value=="0")
{
alert("enter the experience")
document.validate.experience.focus();
return false
}
if(document.validate.project.value=="")
{
alert("enter the project description")
document.validate.project.focus();
return false
}
if(document.validate.salary.value=="")
{
alert("enter the salary ")
document.validate.salary.focus();
return false
}
if( isNaN(document.validate.salary.value))
{
alert("enter a valid salary")
document.validate.salary.focus();

```

```

document.validate.salary.value="";
return false
}
}
</script>
</head>
<body>
<div id="Layer1">
<form id="validate" name="validate" method="post" action="">
<p>&nbsp;</p>
<table width="886" border="0" cellspacing="2" cellpadding="2">
<tr>
<td colspan="2"><div align="center" class="style1">Personal Details </div></td>
<td width="30" rowspan="10">&nbsp;</td>
<td colspan="2"> <div align="center" class="style2">Education and Experience </div></td>
</tr>
<tr>
<td width="111">Full Name</td>
<td width="255"><label>
<input name="name" type="text" id="name" />
</label></td>
<td width="152">Qualification</td>
<td width="217"><select name="qualification">
<option value="0">Select</option>
<option value="1">UG</option>
<option value="2">PG</option>
<option value="3">PHD</option>
</select></td>
</tr>
<tr>
<td>Date of Birth </td>
<td><select name="day">

```


<option value="0">Day</option>
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
<option value="7">7</option>
<option value="8">8</option>
<option value="9">9</option>
<option value="10">10</option>
<option value="11">11</option>
<option value="12">12</option>
<option value="13">13</option>
<option value="14">14</option>
<option value="15">15</option>
<option value="16">16</option>
<option value="17">17</option>
<option value="18">18</option>
<option value="19">19</option>
<option value="20">20</option>
<option value="21">21</option>
<option value="22">22</option>
<option value="23">23</option>
<option value="24">24</option>
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>

```

</select>

    <select name="month">
<option value="0">Month</option>
    <option value="1">Janauray</option>
        <option value="2">Feburary</option>
            <option value="3">March</option>
                <option value="4">April</option>
                    <option value="5">May</option>
                        <option value="6">June</option>
                            <option value="7">July</option>
                                <option value="8">August</option>
                                    <option value="9">Sepetember</option>
                                        <option value="10">October</option>
                                            <option value="11">November</option>
                                                <option value="12">December</option>
</select>

    <select name="year">
<option value="0">Year</option>
    <option value="1">2009</option>
        <option value="2">2008</option>
            <option value="3">2007</option>
                <option value="4">2006</option>
                    <option value="5">2005</option>
                        <option value="6">2004</option>
                            <option value="7">2003</option>
                                <option value="8">2002</option>
                                    <option value="9">2001</option>
                                        <option value="10">2000</option>
                                            <option value="11">1999</option>
                                                <option value="12">1998</option>
                                                    <option value="13">1997</option>
                                                        <option value="14">1996</option>

```

```

        <option value="15">1997</option>
        <option value="16">1996</option>
        <option value="17">1995</option>
        <option value="18">1994</option>
        <option value="19">1993</option>
        <option value="20">1992</option>
        <option value="21">1991</option>
        <option value="22">1990</option>
        <option value="23">1989</option>
        <option value="24">1988</option>
        <option value="25">1987</option>
        <option value="26">1986</option>
        <option value="27">1985</option>
        <option value="28">1984</option>
        <option value="29">1983</option>
        <option value="30">1982</option>
        <option value="31">1981</option>
        <option value="32">1980</option>
    </select></td>
    <td>Work Experience </td>
    <td><select name="experience">
        <option value="0">Select</option>
        <option value="1">Fresher</option>
        <option value="2">1 yr</option>
        <option value="3">2 yrs</option>
        <option value="4">3 yrs</option>
        <option value="5">4 yrs</option>
        <option value="6">5 yrs</option>
        <option value="7">>5 yrs</option>
    </select></td>
</tr>
<tr>

```

```

<td height="70">Sex</td>
<td><select name="gender" id="gender" >
  <option value="0">Select Gender</option>
  <option value="1">Male</option>
  <option value="2">Female</option>
</select></td>
<td>Project Description </td>
<td><label>
  <textarea name="project" id="project"></textarea>
</label></td>
</tr>
<tr>
<td>Address</td>
<td><label>
  <textarea name="address" id="address"></textarea>
</label></td>
<td>Expected Salary P.A </td>
<td><label>
  <input name="salary" type="text" id="salary" />
</label></td>
</tr>
<tr>
<td>City</td>
<td><label>
  <input name="city" type="text" id="city" />
</label></td>
<td colspan="2" rowspan="5">&nbsp;</td>
</tr>
<tr>
<td>State</td>
<td><label>
  <input name="state" type="text" id="state" />

```

```

</label></td>
</tr>
<tr>
<td>PinCode</td>
<td><label>
<input name="pincode" type="text" id="pincode" />
</label></td>
</tr>
<tr>
<td>Email ID</td>
<td><label>
<input name="email" type="text" id="email" />
</label></td>
</tr>
<tr>
<td>Mobile Number</td>
<td><label>
<input name="mobile" type="text" id="mobile" />
</label></td>
</tr>
<tr>
<td colspan="5"><label>
<div align="center">
<input type="submit" name="Submit" value="Submit" onclick="return validation()" />
<label>
<input type="reset" name="Submit2" value="Reset" />
</label>
</div>
</label></td>
</tr>
</table>
</form>

```

```

</div>

<div id="Layer2">
<div align="center">
<h1 class="style3">Job Registration Form </h1>
</div>
</div>
</body>
</html>

```

OUTPUT:

Job Registration Form

Personal Details	Education and Experience
Full Name <input type="text"/>	Qualification <input type="text" value="Select"/>
Date of Birth <input type="text" value="Day"/> <input type="text" value="Month"/> <input type="text" value="Year"/>	Work Experience <input type="text" value="Select"/>
Sex <input type="text" value="Select Gender"/>	Project Description <input type="text"/>
Address <input type="text"/>	Expected Salary P.A. <input type="text"/>
City <input type="text"/>	
State <input type="text"/>	
PinCode <input type="text"/>	
Email ID <input type="text"/>	
Mobile Number <input type="text"/>	

Job Registration Form

Personal Details		Education and Experience	
Full Name	<input type="text" value="srinidhi"/>	Qualification	<input type="text" value="Select"/>
Date of Birth	<input type="text" value="6"/> <input type="text" value="November"/> <input type="text" value="2003"/>	Work Experience	<input type="text" value="Select"/>
Sex	<input type="text" value="Female"/>	Project Description	<input type="text"/>
Address	<input type="text" value="abcd"/>	Expected Salary P.A	<input type="text"/>
City	<input type="text" value="tvl"/>		
State	<input type="text" value="tn"/>		
PinCode	<input type="text" value="627358"/>		
Email ID	<input type="text"/>		
Mobile Number	<input type="text"/>		

This page says
enter the email id

Personal Details		Education and Experience	
Full Name	<input type="text" value="srinidhi"/>	Qualification	<input type="text" value="Select"/>
Date of Birth	<input type="text" value="6"/> <input type="text" value="November"/> <input type="text" value="2003"/>	Work Experience	<input type="text" value="Select"/>
Sex	<input type="text" value="Female"/>	Project Description	<input type="text"/>
Address	<input type="text" value="abcd"/>	Expected Salary P.A	<input type="text"/>
City	<input type="text" value="tvl"/>		
State	<input type="text" value="tn"/>		
PinCode	<input type="text" value="627358"/>		
Email ID	<input type="text"/>		
Mobile Number	<input type="text"/>		

RESULT:

Thus the program of Client Side Scripts for Validating Web Form Controls using DHTML has been implemented and executed successfully.

EXP NO:4

INSTALLATION OF APACHE TOMCAT WEB SERVER

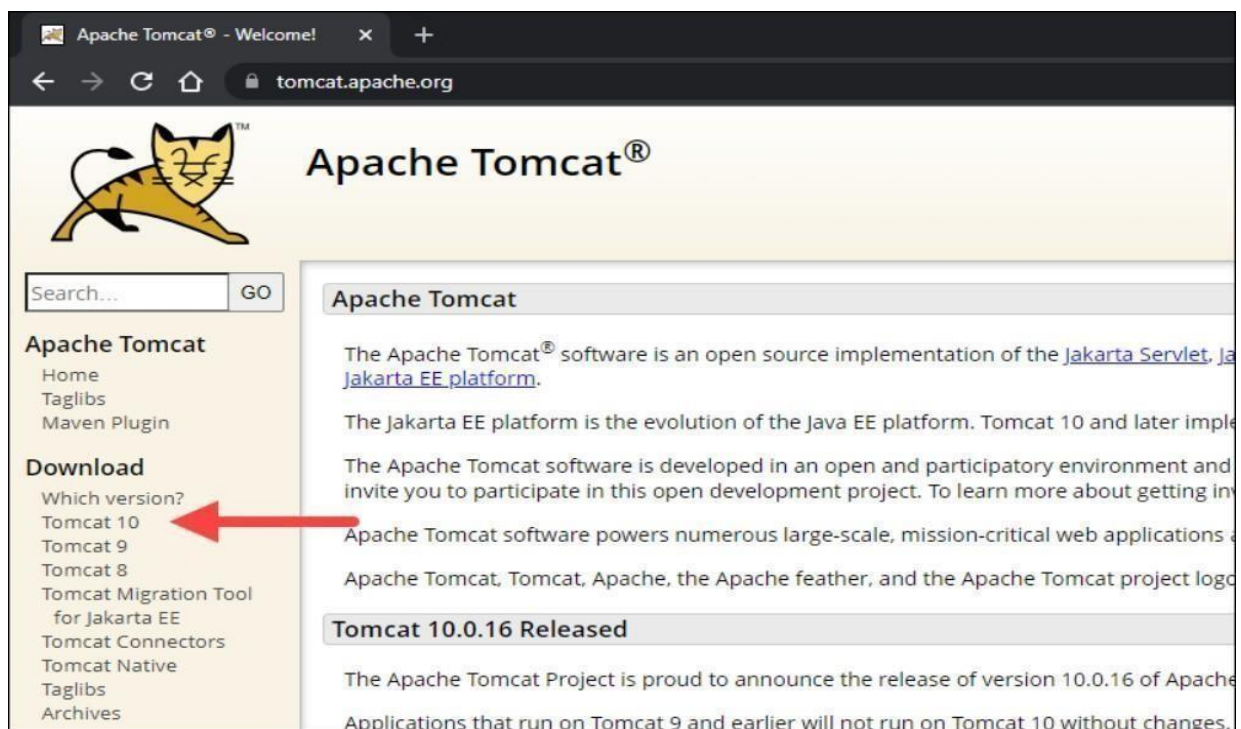
DATE :

AIM:

To study how to install Apache Tomcat Web Server.

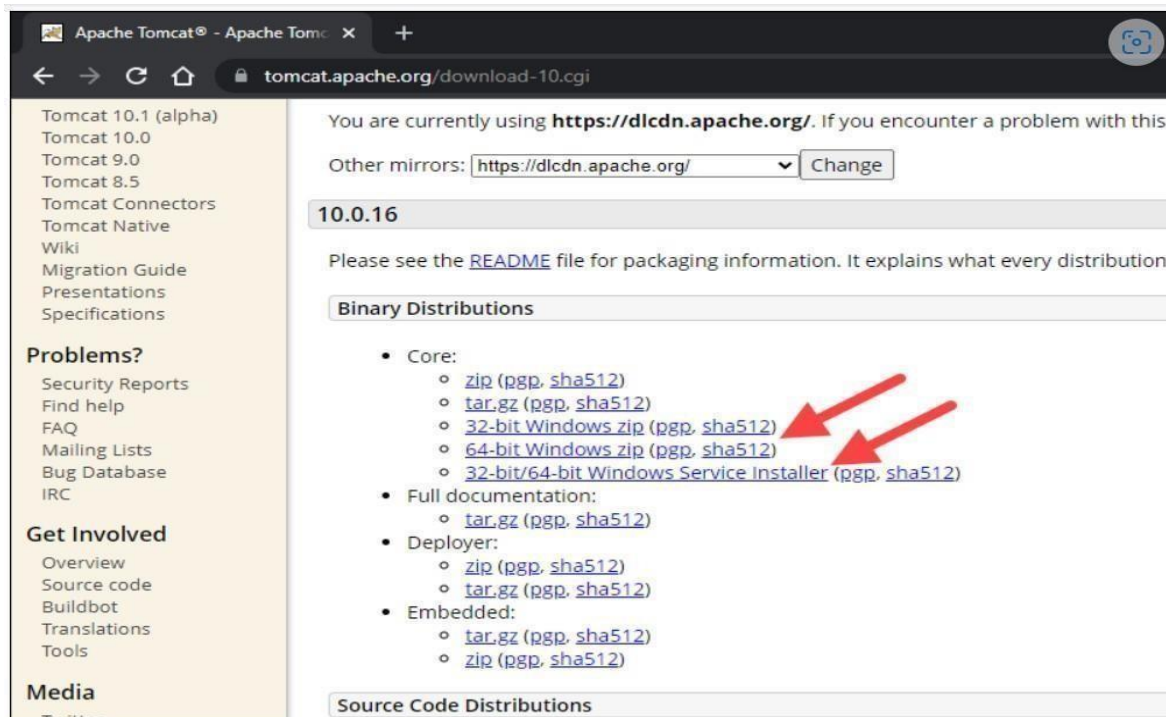
PROCEDURE:

1. Browse to the official Apache Tomcat website. Locate the *Download* section and click the latest Tomcat version available.



2. On the *Download* page, scroll down and locate the *BinaryDistributions* 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.



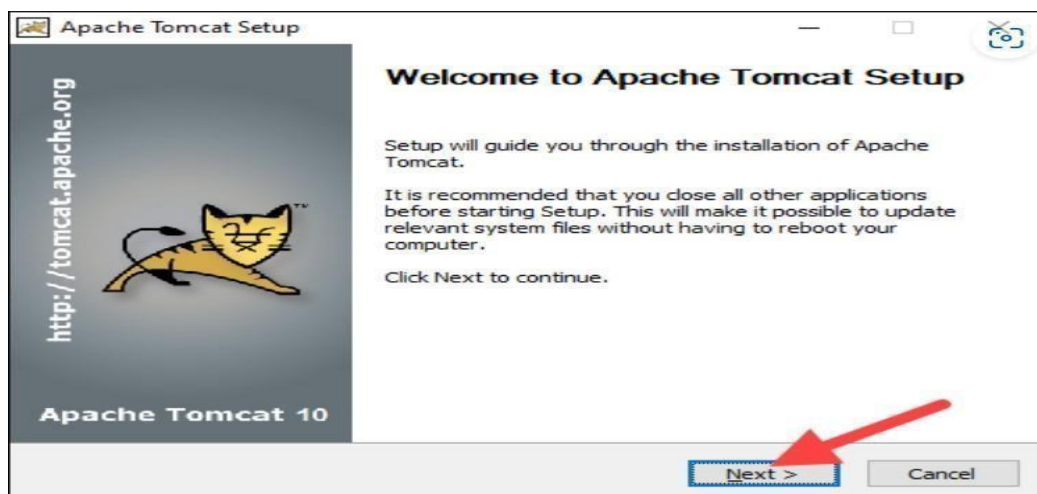
3. Install Tomcat

METHOD 1:

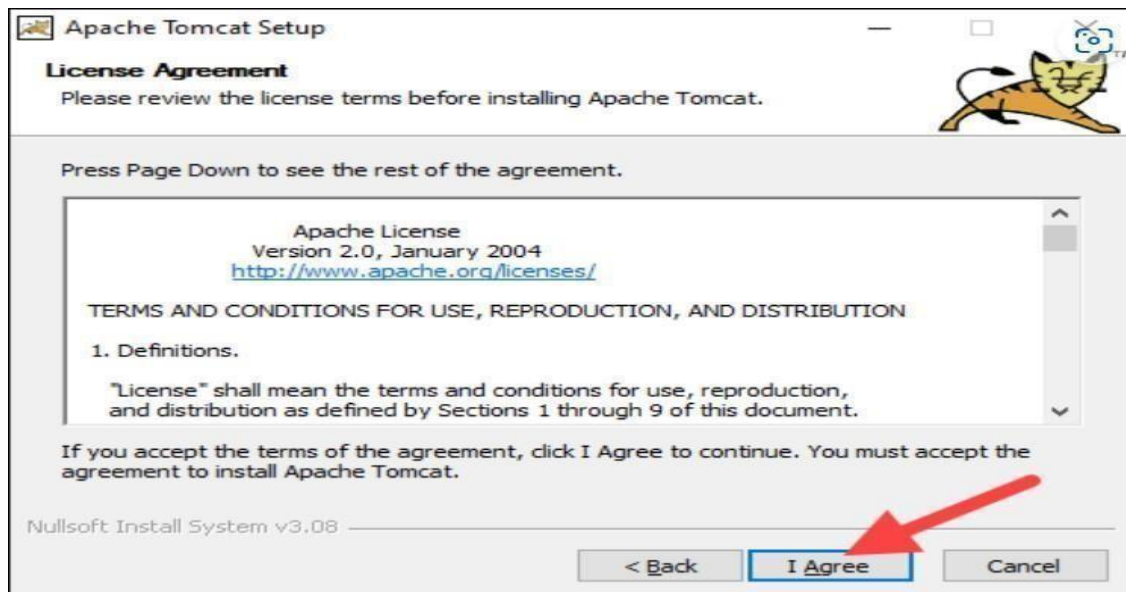
Install Tomcat via the Windows Service Installer for an automated and wizard-guided experience. The service installer installs the Tomcat service and runs it automatically when the system boots.

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

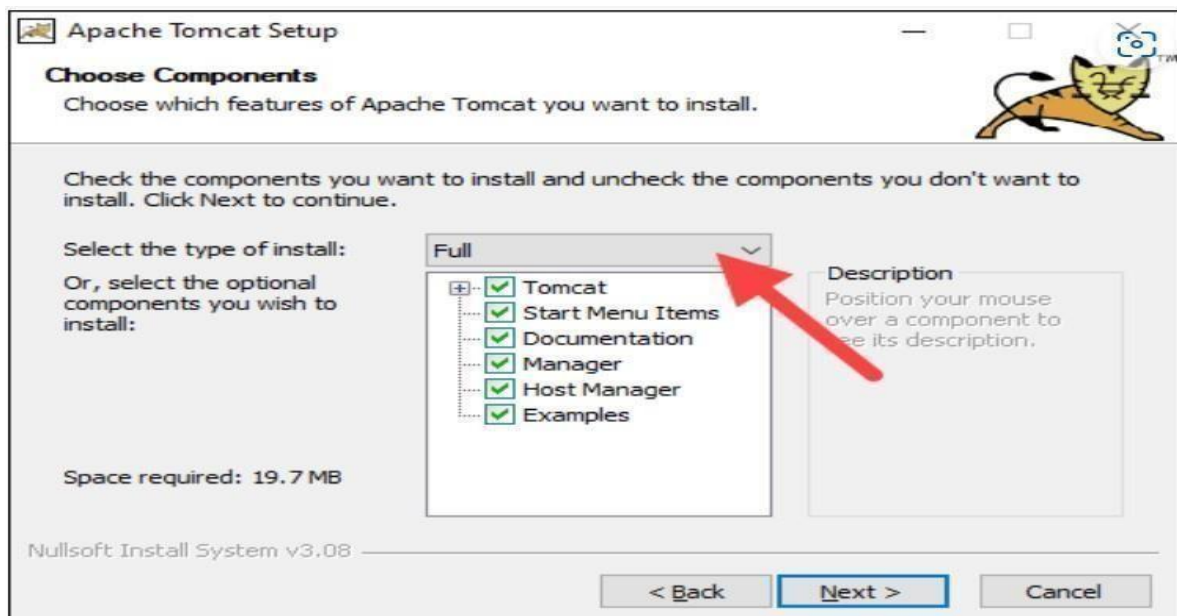
1. Open the downloaded Windows Service Installer file to start the installation process.
2. In the Tomcat Setup welcome screen, click Next to proceed.



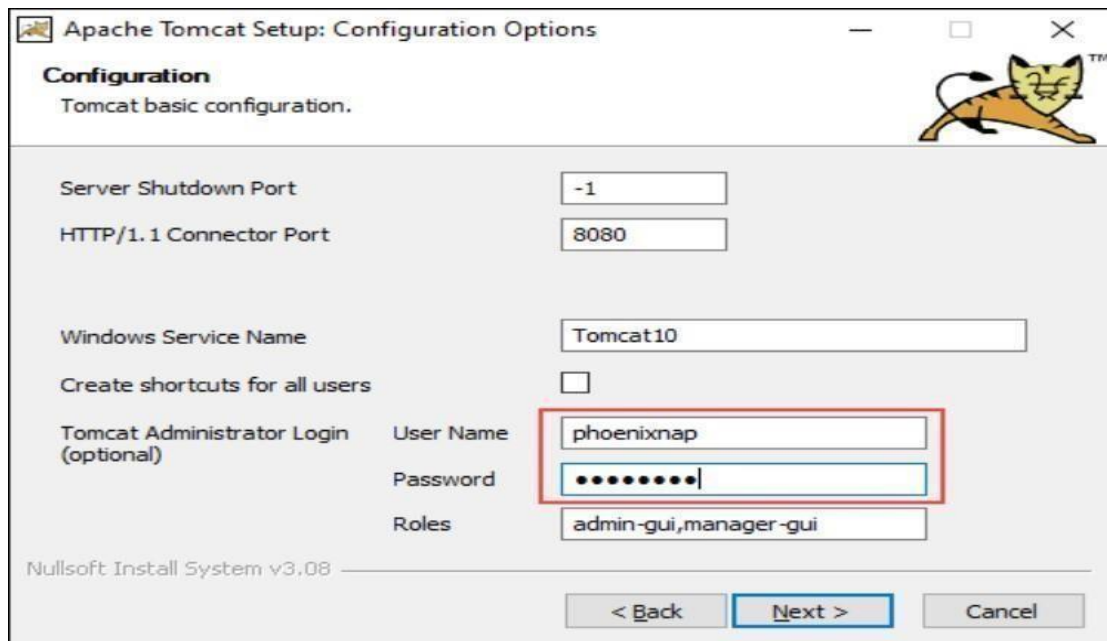
3. Read the License Agreement and if you agree to the terms, click I Agree to proceed to the next step.



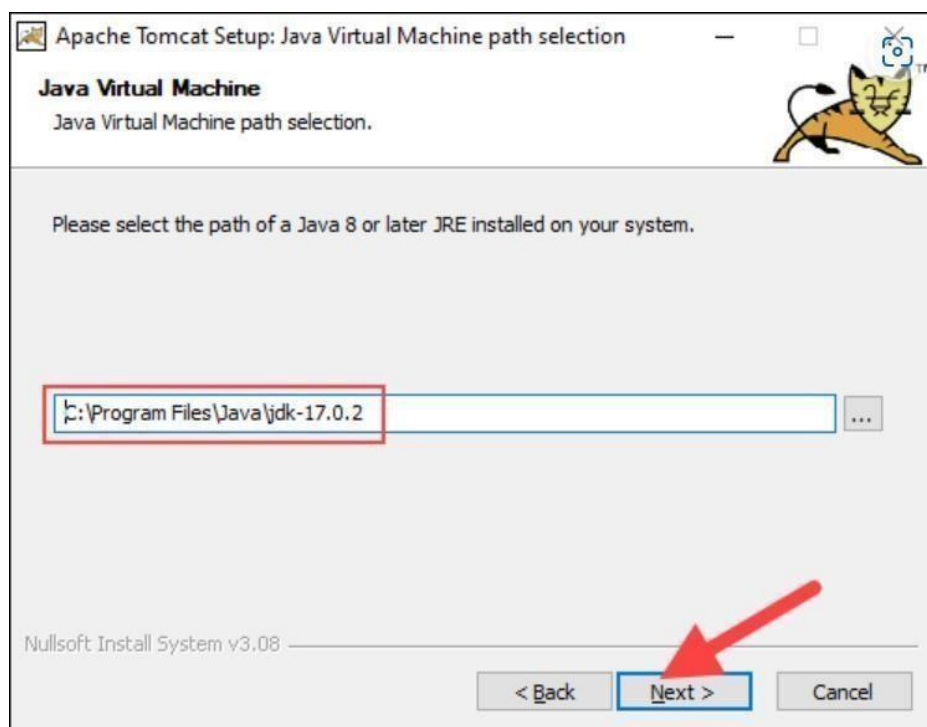
4. In the Tomcat component selection screen, choose Full in the dropdown menu to ensure the wizard installs the Tomcat Host Manager and Servlet and JSP examples web applications. Alternatively, keep the default Normal installation type and click Next.



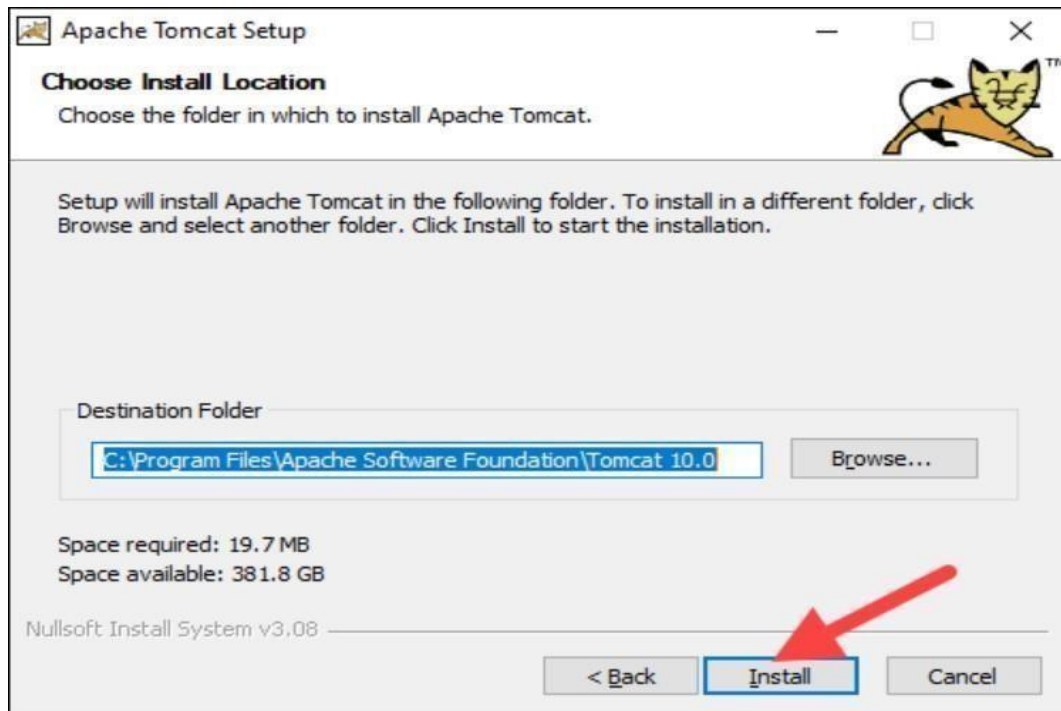
5. The next step configures the Tomcat server. For instance, enter the Administrator login credentials or choose a different connection port. When finished, click Next to proceed to the next step



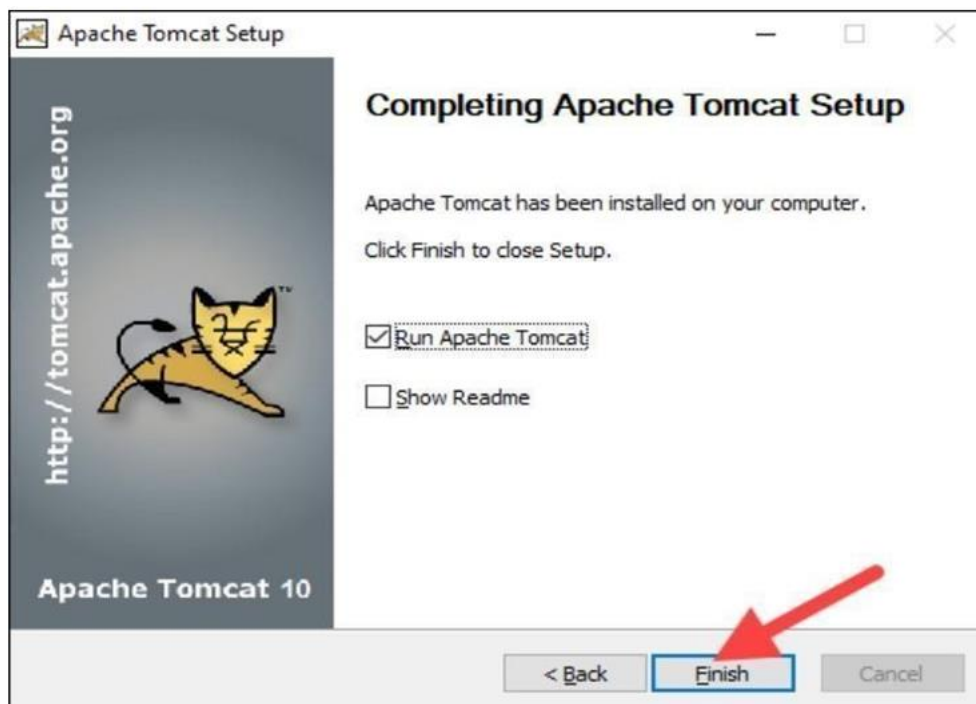
6. The next step requires you to enter the full path to the JRE directory on your system. The wizard auto-completes this if you have previously set up the Java environment variables. Click Next to proceed to the next step.



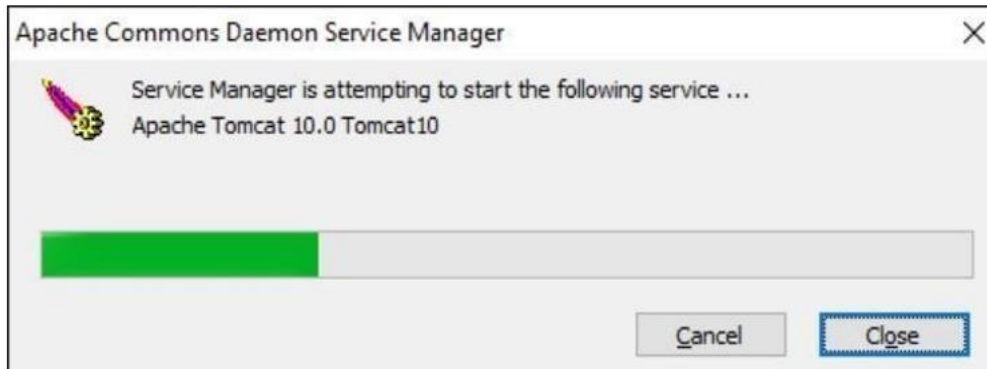
7. Choose the Tomcat server install location or keep the default one and click Install.



8. 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.



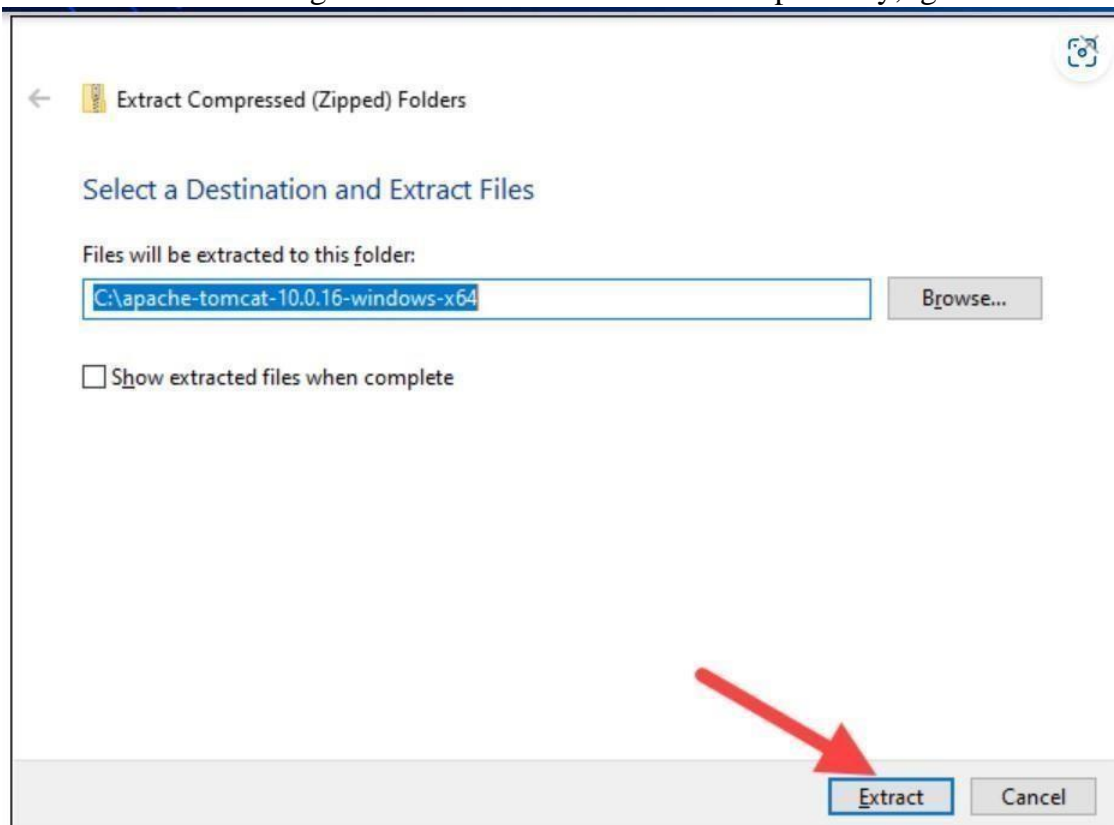
9. A popup window appears that starts the Tomcat service. After the process completes, the window closes automatically. The Apache Tomcat web server is now successfully installed.



METHOD 2:

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.

1. After downloading the 32bit/64bit Windows zip file, depending on your Windows version, unzip the downloaded file. Right-click the file and select Extract all...
2. Choose where to extract the archive contents. For easier navigation, we recommend extracting it to the hard drive's root. Optionally, give the



directory a shorter name to facilitate server configuration later. Click Extract to start the process.

3. Navigate to the *conf* sub-directory within the extracted directory and locate the *server.xml* file.



Important: Back up the .xml files before making any changes.

4. The default connection port is 8080. To choose a different port, edit the *server.xml* file with a text editor, such as Notepad++, and locate the following lines:

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000" redirectPort="8443"/>
```

Change the connector port number to any number between 1024 and 65535.

5. To enable directory browsing, locate the *web.xml* file in the *conf* directory and edit the file with a text editor. Directory browsing helps when testing the system, and sometimes it may be the solution for a 403 forbidden error.

Locate the following lines and change the *listings* value from *false* to *true*:

```
<servlet>
<servlet-name>default</servlet-name>
<servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
<init-param>
<param-name>debug</param-name>
<param-value>0</param-value>
</init-param>
<init-param>
<param-name>listings</param-name>
<param-value>>false</param-value>
</init-param>
```



```
<load-on-startup>1</load-on-startup>
```

```
</servlet>
```

6. Implement an auto-reload feature by editing the context.xml file. Above all, auto-reload is useful in development to prevent restarting the server manually each time a change is made. Using a text editor, open the context.xml file. Locate the following line and change the value from false to true in each instance:

```
<Context                      reloadable="false"                      crossContext="false"
parallelAnnotationScanning="false">
```

```
.....
```

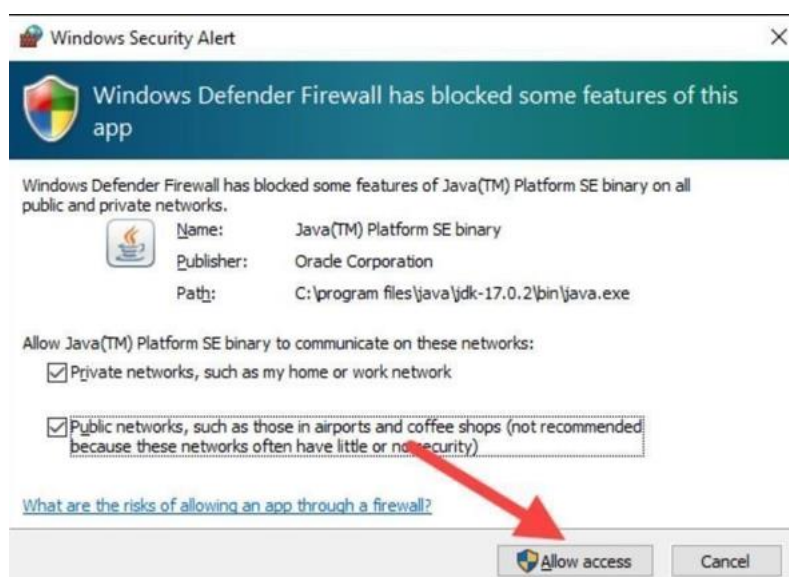
```
.....
```

```
</Context>
```

7. After making the changes, start the server. Press the Windows key and type *cmd*. Press Enter to open a Command Prompt window.
8. Move to the *bin* directory of your Tomcat server and run: startup

```
C:\Users\marij>cd c:\tomcat\bin
c:\tomcat\bin>startup
Using CATALINA_BASE:   "c:\tomcat"
Using CATALINA_HOME:   "c:\tomcat"
Using CATALINA_TMPDIR: "c:\tomcat\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk-17.0.2"
Using CLASSPATH:       "c:\tomcat\bin\bootstrap.jar;c:\tomcat\bin\tomcat-juli.jar"
Using CATALINA_OPTS:   ""
c:\tomcat\bin>
```

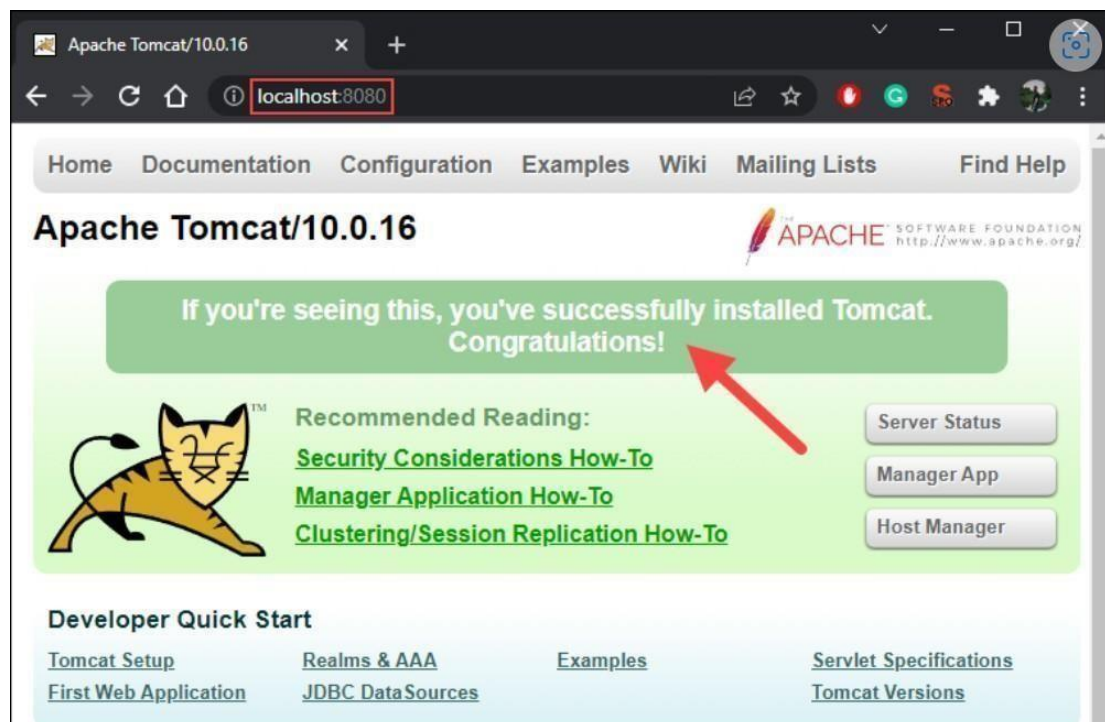
9. Add an exception for Tomcat in the firewall:



10. A new Tomcat console window appears. This console receives error messages and `system.out.println()` messages issued by the Java servlets.

```
Tomcat
10-Feb-2022 16:25:37.580 INFO [main] org.apache.catalina.startup.Catalina.load Server initialization in [490] milliseconds
10-Feb-2022 16:25:37.627 INFO [main] org.apache.catalina.core.StandardService.startInternal Starting service [Catalina]
10-Feb-2022 16:25:37.627 INFO [main] org.apache.catalina.core.StandardEngine.startInternal Starting Servlet engine: [Apache Tomcat/10.0.16]
10-Feb-2022 16:25:37.638 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\tomcat\webapps\docs]
10-Feb-2022 16:25:37.947 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\tomcat\webapps\docs] has finished in [308] ms
10-Feb-2022 16:25:37.947 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\tomcat\webapps\examples]
10-Feb-2022 16:25:38.476 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\tomcat\webapps\manager]
10-Feb-2022 16:25:38.520 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\tomcat\webapps\manager] has finished in [44] ms
10-Feb-2022 16:25:38.521 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deploying web application directory [C:\tomcat\webapps\ROOT]
10-Feb-2022 16:25:38.551 INFO [main] org.apache.catalina.startup.HostConfig.deployDirectory Deployment of web application directory [C:\tomcat\webapps\ROOT] has finished in [30] ms
10-Feb-2022 16:25:38.554 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]
10-Feb-2022 16:25:38.648 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [1068] milliseconds
```

11. Access the server using a browser as an HTTP client. Browse to `http://localhost:8080` and access the Tomcat welcome page to ensure the server works. In addition, use the *Developer Quick Start* links to see more information

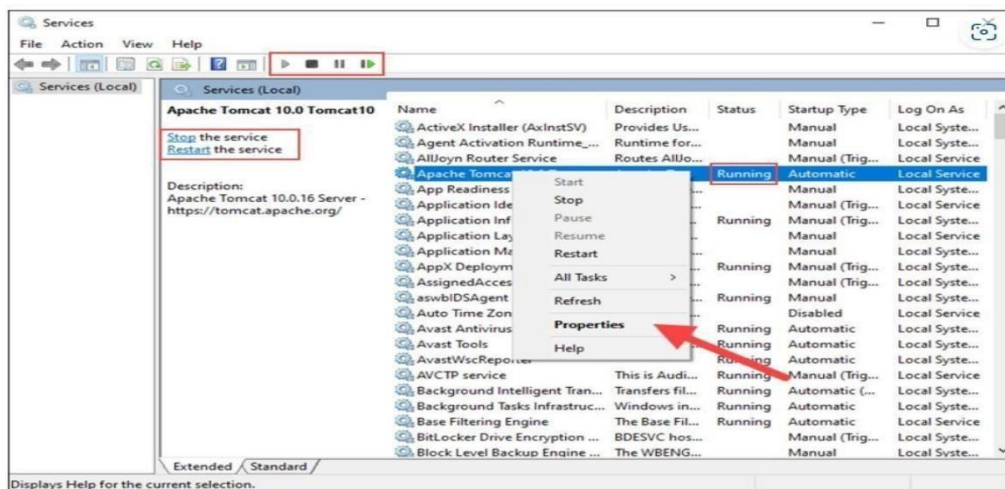


about the server and start using and configuring the server.

12. Shut down the Tomcat server by pressing Ctrl+C on the Tomcat console.
13. Check if Apache Tomcat Service Is Running: Installing Tomcat using the Windows Service Installer installs Tomcat as a Windows service that

automatically runs on boot. Follow the steps below to ensure that Tomcat is started as a Windows service.

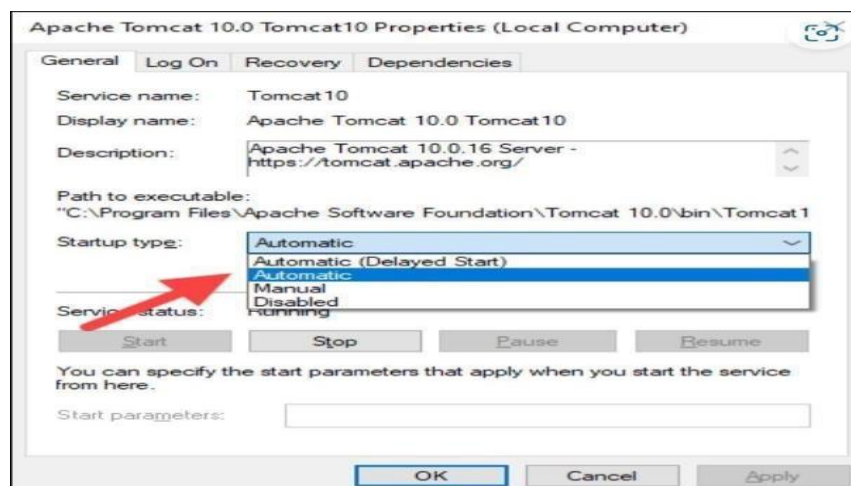
- a) Open the Start menu and search for *Services*.
- b) Select the *Services* result.
- c) In the *Services* window, locate the *Apache Tomcat* service. The *Status* column indicates whether the service is running or not. Start or Stop the service using the buttons in the toolbar or by pressing Stop or Restart on the left side of the service list.



14. Configure the service startup by right-clicking the Tomcat service and selecting Properties.

- i. In the *Properties* window, under the *Startup type* dropdown menu, select how to run the Tomcat service:

Automatic (Delayed Start). Starts the service shortly after boot. A delayed start improves server boot performance and has security benefits. The service starts only when Windows or another service needs it or if invoked. Disables the service startup, even if you try to start it.



15. Click OK to confirm the changes.

RESULT:

Thus, the Apache Tomcat has been installed successfully.

EXP NO: 5a

TO INVOKE JAVA SERVLETS FROM

DATE:

HTML FORMS

AIM :

To write a program in Java Servlets to invoke servlets from HTML forms.

ALGORITHM:

Step 1 : Install a Java development environment and a servlet container .

Step 2 : Develop an HTML form with appropriate input fields, and set the form action attribute to the URL of your servlet.

Step 3 : Write a Java Servlet to handle form submissions. Retrieve form parameters and process it.

Step 4 : Ensure your web.xml or use annotations (like @WebServlet) to map your servlet to a URL.

Step 5 : Deploy your web application to the servlet container and start the server.

Step 6: Open a web browser, navigate to the HTML form (e.g., <http://localhost:8080/YourWebAppName/index.html>), and submit the form.

PROGRAM:

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>NewServlet</servlet-name>

<servlet-class>NewServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>NewServlet</servlet-name>

<url-pattern>/NewServlet</url-pattern>

</servlet-mapping>

<session-config>

<session-timeout>
```

30

```
</session-timeout>
</session-config>
<welcome-file-list>
<welcome-file>index.html</welcome-file>
</welcome-file-list>
</web-app>
```

NewServlet.java:

```
import java.io.*;
import java.util.*;
import javax.servlet.*;

public class NewServlet extends GenericServlet {
    public void service(ServletRequest
    request,ServletResponse response) throws
    ServletException, IOException {
        PrintWriter pw = response.getWriter();
        Enumeration e = request.getParameterNames();
        while(e.hasMoreElements()) {
            String pname = (String)e.nextElement();
            pw.print(pname + " = ");
            String pvalue = request.getParameter(pname);
            pw.println(pvalue);
        }
        pw.close();
    }
}
```

index.html:

```
<!DOCTYPE html>
```

```
<!--
```

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.

```
-->
```

```
<html>
```

```
<body>
```

```
<center>
```

```
<form name = "postparam" method = "post" action="NewServlet">
```

```
<table>
```

```
<tr>
```

```
<td><B>Employee </B> </td>
```

```
<td><input type = "textbox" name="ename" size="25" value=""></td>
```

```
</tr>
```

```
<tr>
```

```
<td><B>Phone </B> </td>
```

```
<td><input type = "textbox" name="phoneno" size="25" value=""></td>
```

```
</tr>
```

```
</table>
```

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

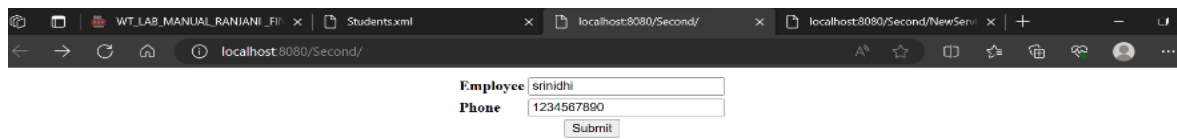
```
</form>
```

```
</center>
```

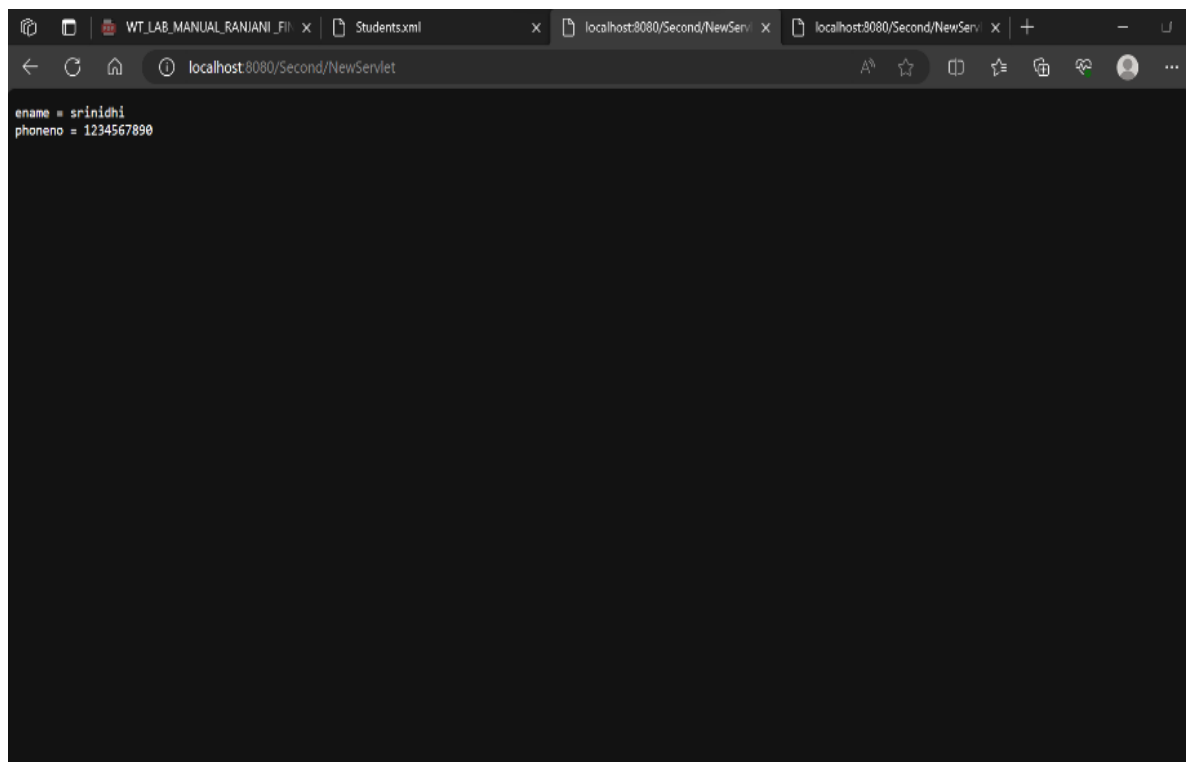
```
</body>
```

```
</html>
```

OUTPUT:



A screenshot of a web browser window. The address bar shows 'localhost:8080/Second/'. The page content includes a form with two input fields: 'Employee' containing 'srinidhi' and 'Phone' containing '1234567890'. Below these fields is a 'Submit' button. The browser's tab bar shows several open tabs, including 'WT_LAB_MANUAL_RANIANI_Fil...', 'Students.xml', 'localhost:8080/Second/', and 'localhost:8080/Second/NewServ...'. The browser's toolbar includes back, forward, and search icons.



A screenshot of a web browser window. The address bar shows 'localhost:8080/Second/NewServlet'. The page content displays the output of a servlet, showing 'ename = srinidhi' and 'phoneno = 1234567890'. The browser's tab bar shows several open tabs, including 'WT_LAB_MANUAL_RANIANI_Fil...', 'Students.xml', 'localhost:8080/Second/NewServlet', and 'localhost:8080/Second/NewServlet'. The browser's toolbar includes back, forward, and search icons.

RESULT:

Thus the program to invoke servlets from HTML forms with Servlets has been implemented and executed successfully.

EXPNO: 5b

SESSION TRACKING

DATE:

AIM:

To create servlet-based application with session and cookies to display hit count.

ALGORITHM:

Step 1: Import required java libraries

Step 2: Extend HttpServlet class

Step 3: Create a session object if it is already not created.

Step 4: Get session creation time.

Step 5: Get last access time of this web page.

Step 6: Set response content type

Step 7: Compile the servlet program SessionTrack and create appropriate entry in web.xml file.

Step 8: Now running <http://localhost:8080/SessionTrack>

PROGRAM:

Index.html :

```
<!DOCTYPE html>

<html>

<head>

<title>Start Page</title>

<meta http-equiv="Content-type" content="text/html; charset=UTF-8">

</head>

<body>

<h1>WT Experiment -5 Session Tracking</h1>

<a href='session'>Invoke session tracking</a>

</body>

</html>
```

Session.java:

```
import jakarta.servlet.ServletException;
```

```

import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

public class session extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)

    throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

    }

```

@Override

```

    protected void doGet(HttpServletRequest request, HttpServletResponse
response)

    throws ServletException, IOException {

        // Create a session object if it is already not created.

        HttpSession session = request.getSession(true);

        // Get session creation time.

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

        // Get last access time of this web page.

        Date lastAccessTime = new Date(session.getLastAccessedTime());

        String title = "Welcome Back to my website";

        Integer visitCount = 0;

        String visitCountKey = "visitCount";

        String userIDKey = "userID";

        String userID = "ABCD";

        // Check if this is new comer on your web page.

```



```

if (session.isNew()){
title = "Welcome to my website";
session.setAttribute(userIDKey, userID);
} else {
visitCount = (Integer)session.getAttribute(visitCountKey);
visitCount = visitCount + 1;
userID = (String)session.getAttribute(userIDKey);
}
session.setAttribute(visitCountKey, visitCount);
// Set response content type
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String docType =
"<!doctype html public "-//w3c//dtd html 4.0 " +
"transitional//en\">\n";
out.println(docType +
"<html>\n" +
"<head><title>" + title + "</title></head>\n" +
"<body bgcolor=\"#f0f0f0\">\n" +
"<h1 align=\"center\">" + title + "</h1>\n" +
"<h2 align=\"center\">Session Infomation</h2>\n" +
"<table border=\"1\" align=\"center\">\n" +
"<tr bgcolor=\"#949494\">\n" +
" <th>Session info</th><th>value</th></tr>\n" +
"<tr>\n" +
" <td>id</td>\n" +
" <td>" + session.getId() + "</td></tr>\n" +
"<tr>\n" +
" <td>Creation Time</td>\n" +

```

```

" <td>" + createTime +
" </td></tr>\n" +
"<tr>\n" +
" <td>Time of Last Access</td>\n" +
" <td>" + lastAccessTime +
" </td></tr>\n" +
"<tr>\n" +
" <td>User ID</td>\n" +
" <td>" + userID +
" </td></tr>\n" +
"<tr>\n" +
" <td>Number of visits</td>\n" +
" <td>" + visitCount + "</td></tr>\n" +
"</table>\n" +
"</body></html>");
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
    processRequest(request, response);
}

@Override
public String getServletInfo() {
    return "Short description";
}
}

```

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>Server</servlet-name>

        <servlet-class>Server</servlet-class>

    </servlet>

    <welcome-file-list><welcome-file>index.html</welcome-file></welcome-file-
list>

    <servlet-mapping>

        <servlet-name>Server</servlet-name>

        <url-pattern>/Server</url-pattern>

    </servlet-mapping>

    <session-config>

        <session-timeout>

            30

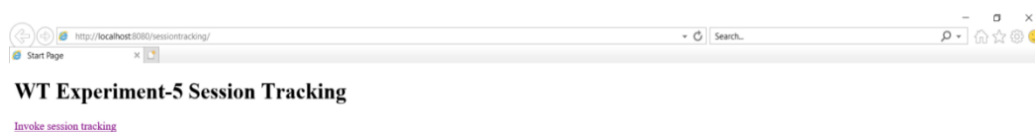
        </session-timeout>

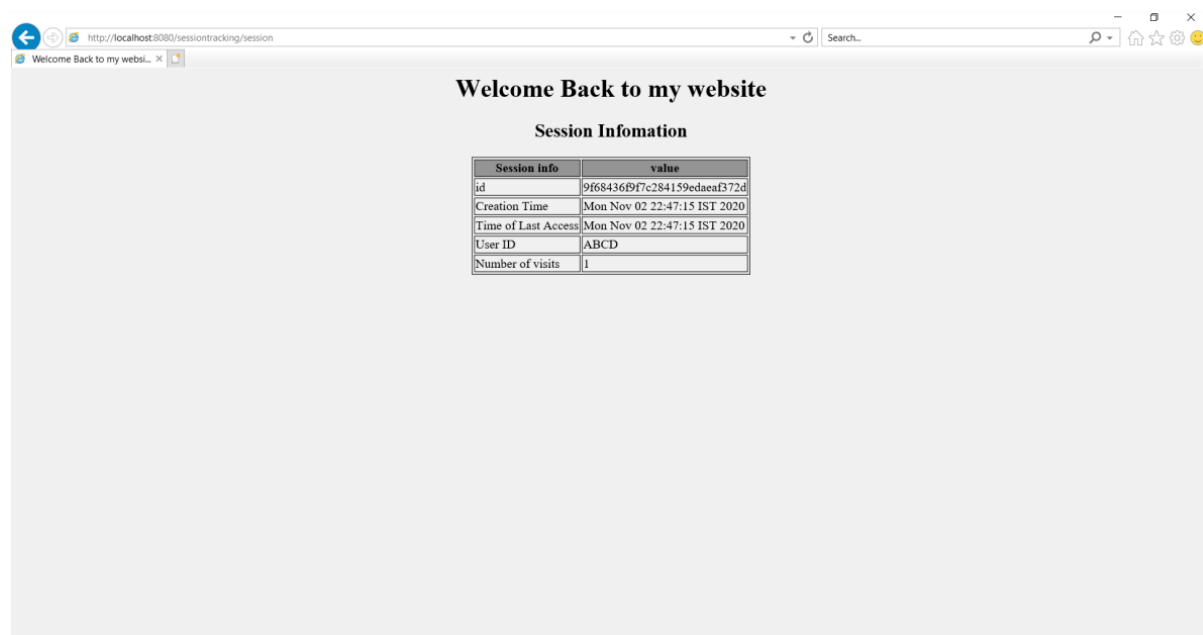
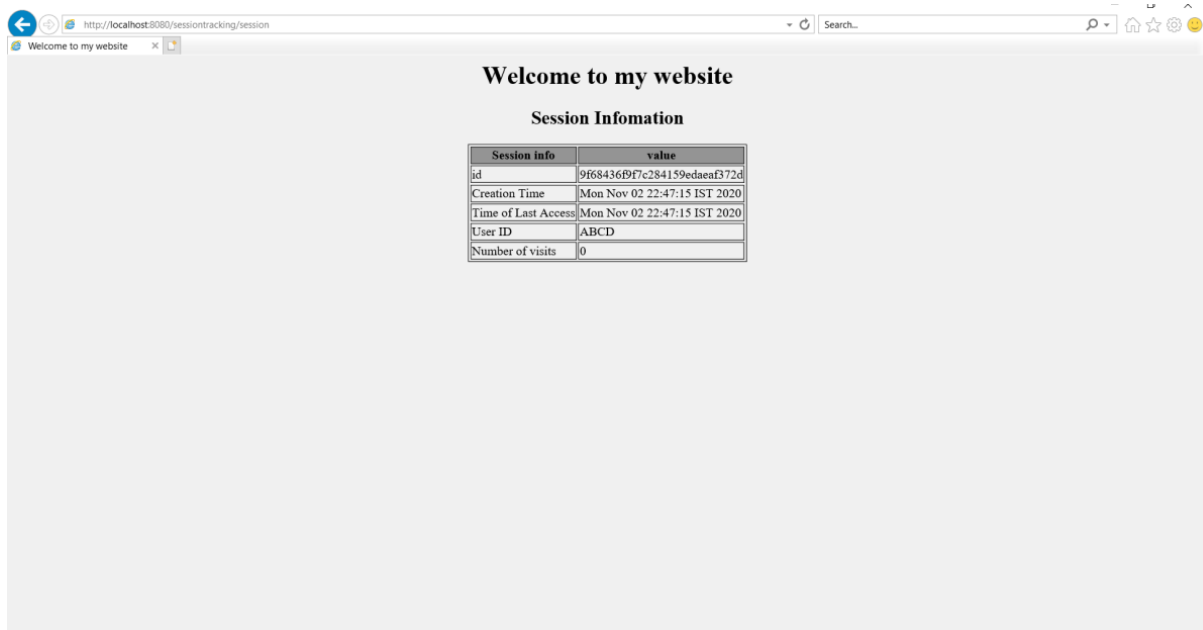
    </session-config>

</web-app>

```

OUTPUT:





RESULT:

Thus the program to create servlet-based application with session and cookies to display hit count has been implemented and executed successfully.

EXNO:6a TO CREATE THREE TIER APPLICATION USING JSP DATABASES

DATE:

AIM:

To write a program for conducting online examinations using JSP databases.

ALGORITHM:

Step 1: Install a Java development environment, a servlet container, and a database server (e.g., MySQL).

Step 2: Design and create a database table that the application will interact with.

Step 3: Develop a servlet that handles interactions with the database. Connect to the database, perform CRUD operations, and store/retrieve data.

Step 4: Develop a JSP page to display data and handle user interactions. Access the data sent by the servlet and generate dynamic content.

Step 5: Ensure your web.xml or use annotations like WebServlet to map your servlet.

Step 6: Deploy your web application to the servlet container and start the server. Access the servlet through a web browser (e.g., <http://localhost:8080/YourWebAppName/UserServlet>) to view the JSP page with data retrieved from the database.

PROGRAM:

Result.jsp:

```
<html>

<%

String[] userans=new String[5];

String[] correctans={ "True","False","True", "True", "True"};

String regno="";

int total=0;

try {

userans[0]=request.getParameter("group1");

userans[1]=request.getParameter("group2");

userans[2]=request.getParameter("group3");

userans[3]=request.getParameter("group4");
```

```

userans[4]=request.getParameter("group5");
for(int i=0;i<=4;i++) {
if(correctans[i].equals(userans[i])) total+=2;
}
regno=request.getParameter("frmregno");
}
catch(Exception e)
{
out.println(e);
}
%>
<b> <center> <h1>Anna university <br/> <br/>
<table border=2>
<th> Register number </th>
<th> Marks </th>
<tr> <td> <%= regno %> </td>
<td> <%= total %> </td>
</table>
</html>

```

index.html:

```

<html>
<form action="Result.jsp" method="get">
<h1>
<center> ONLINE EXAMINATION </center>
<h1> Register no</h1>
<input type="text" name="frmregno"> <br/> <br/>
<b>
1. XML is Extensible Markup Language</b><br/>

```

`<input type="radio" name="group1" value="True">True <input type="radio" name="group1"`

`value="False">False
`

2. Java Servlets is Client Side Scripting

`<input type="radio" name="group2" value="True">True <input type="radio" name="group2"`

`value="False">False
 `

3. CSS is used in style specifications

`<input type="radio" name="group3"`

`value="True">True <input type="radio" name="group3" value="False">False
`

4. Xml Dtd is Document Type Defenition

`<input type="radio" name="group4" value="True">True <input type="radio" name="group4"`

`value="False">False
`

5. HTML is Mark UP LANGUAGE

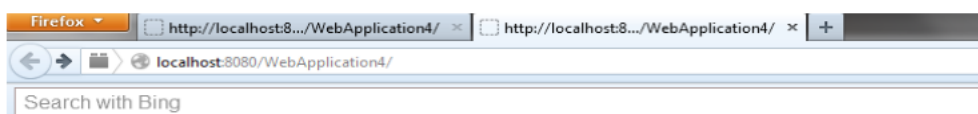
`<input type="radio" name="group5" value="True">True`

`<input type="radio" name="group5" >False`

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

`</html>`

OUTPUT:



ONLINE EXAMINATION

Register no:

1. XML is Extensible Markup Language

☒ True ☐ False

2. Java Servlets is Client Side Scripting

☐ True ☒ False

3. CSS is used in style specifications

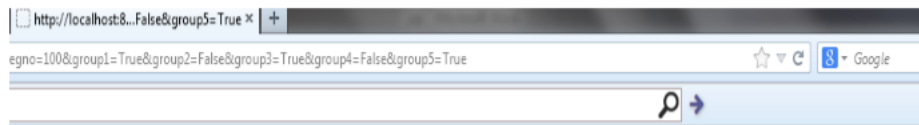
☐ True ☒ False

4. Xml Dtd is Document Type Defenition

☐ True ☒ False

5. HTML is Mark UP LANGUAGE

☐ True ☒ False



Anna university

Register number	Marks
100	4

RESULT:

Thus the program for conducting online examinations using JSP has been written and implemented successfully.

EXNO:6b JSP PROGRAM FOR DISPLAYING STUDENT RESULTS
DATE: USING DATABASE CONNECTIVITY

AIM:

To write a program for displaying student results and assume that student information is available in a database which has been stored in a database server.

ALGORITHM:

Step 1: Design the HTML page (ExamClient.html) with the following

- a) Create a form to get the input from the user.
- b) Use radio buttons to make various options for the questions.
- c) Set the URL of the server (ExamServer.jsp) as the value of the action attribute.
- d) Use submit button to invoke the server and send the form data to the server.

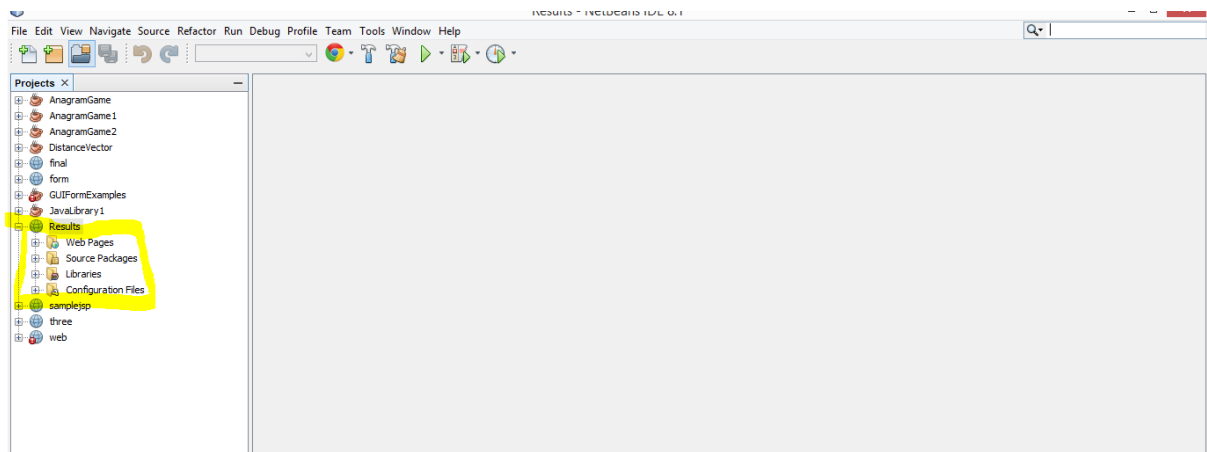
Step 2: Create the JSP file with the following

- a) Read the input from the client.
- b) Retrieve the answers from the database.
- c) Match the answers from the user with the correct answers from the database table.
- d) For each correct answer increment the mark by 5.
- e) Server displays the mark and result to the client as a response.

Step to execute this program

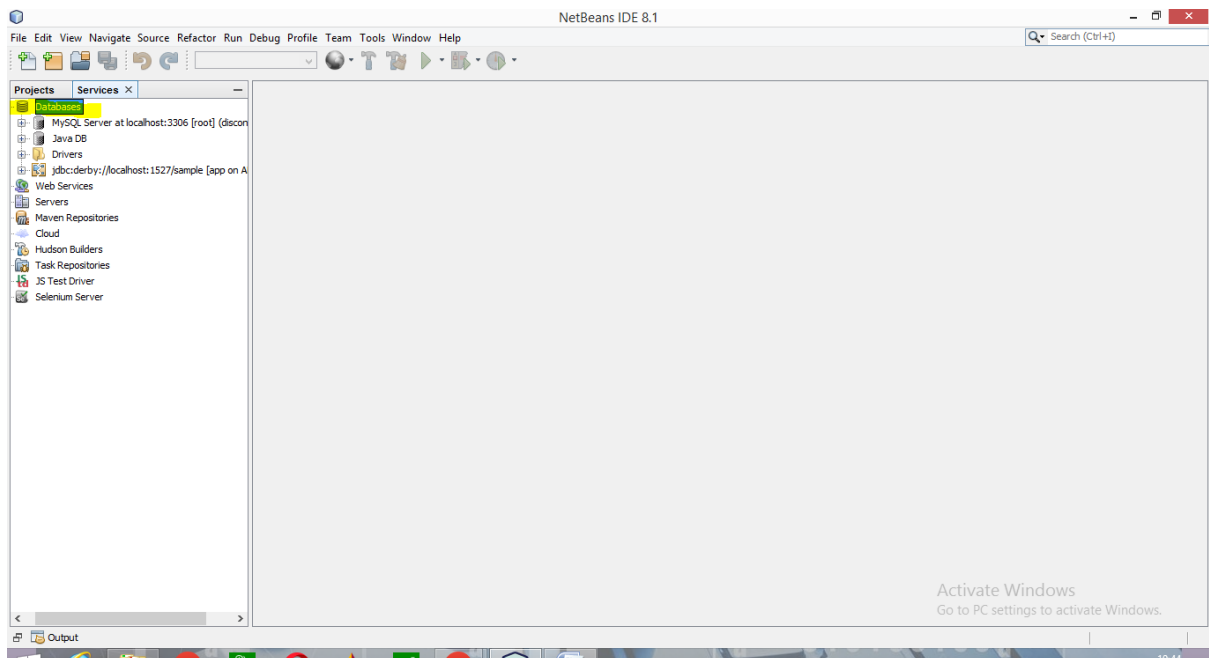
- 1.Create new project
- 2.then create table in derby mysql database
- 3.after creating table create index.html
- 4.create result.jsp
- 5.run and depoly

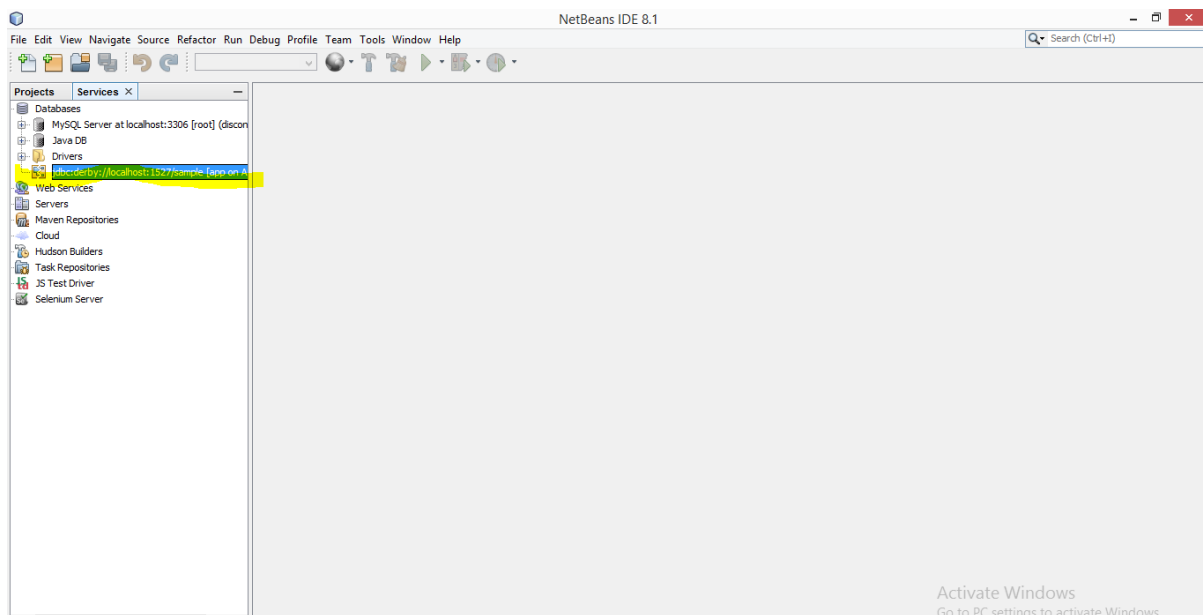
Step 1:



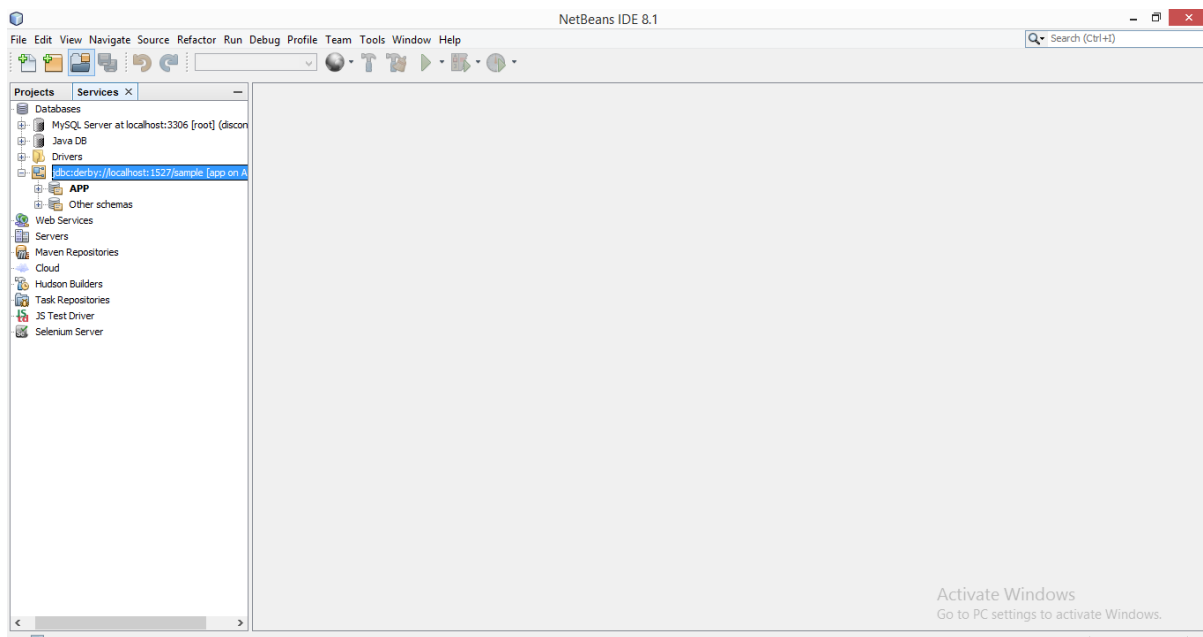
Step 2:

Click ctrl+5



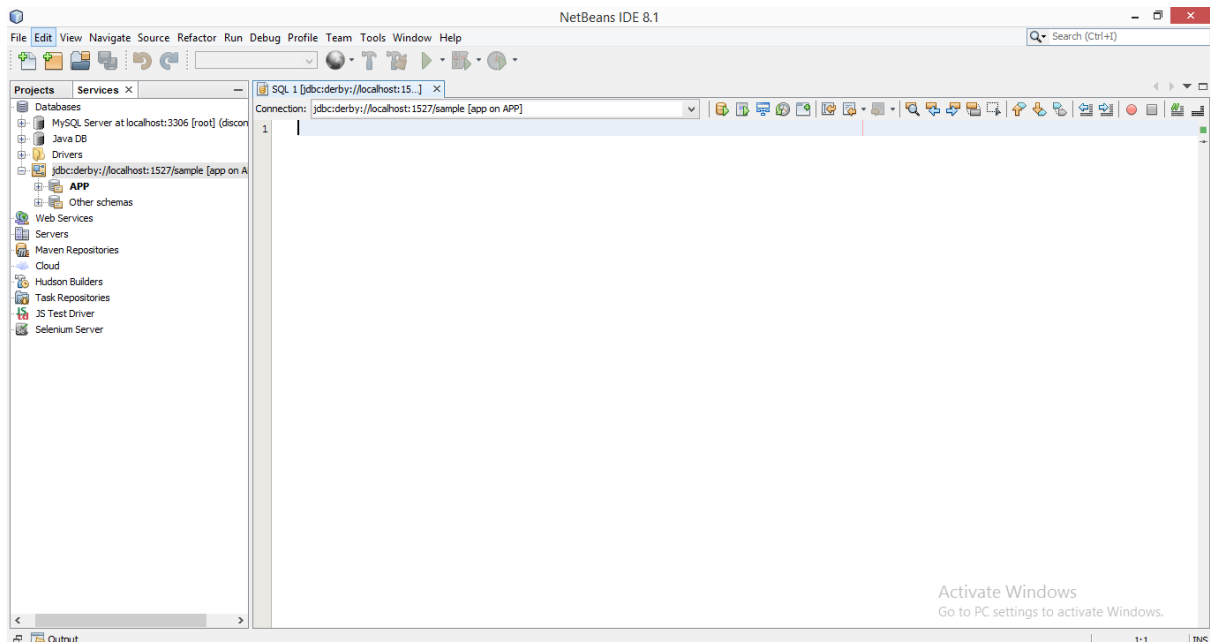


Right click then connect.

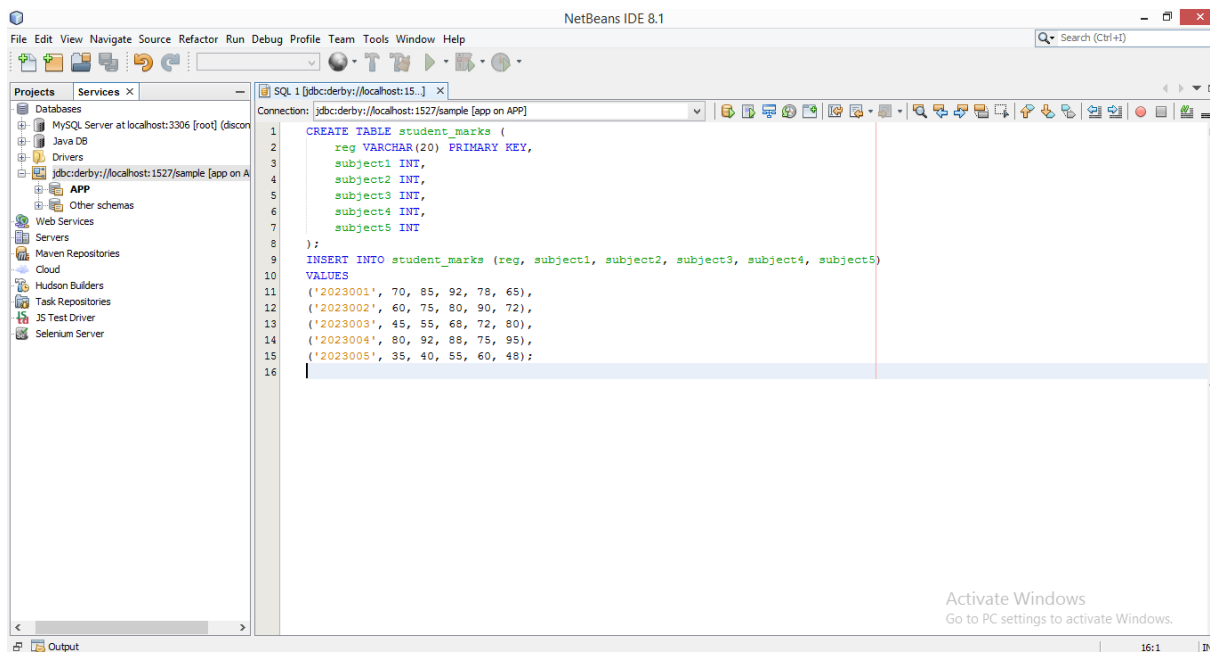


then make it right click on jdbc:localhost

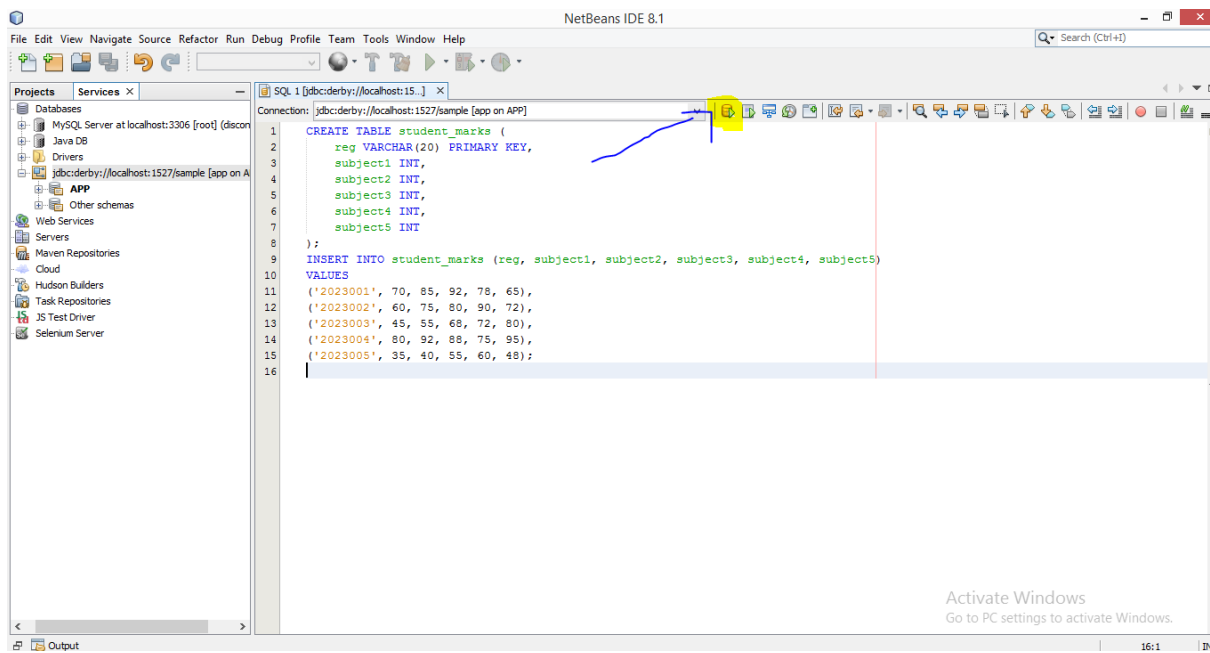
click execute command.



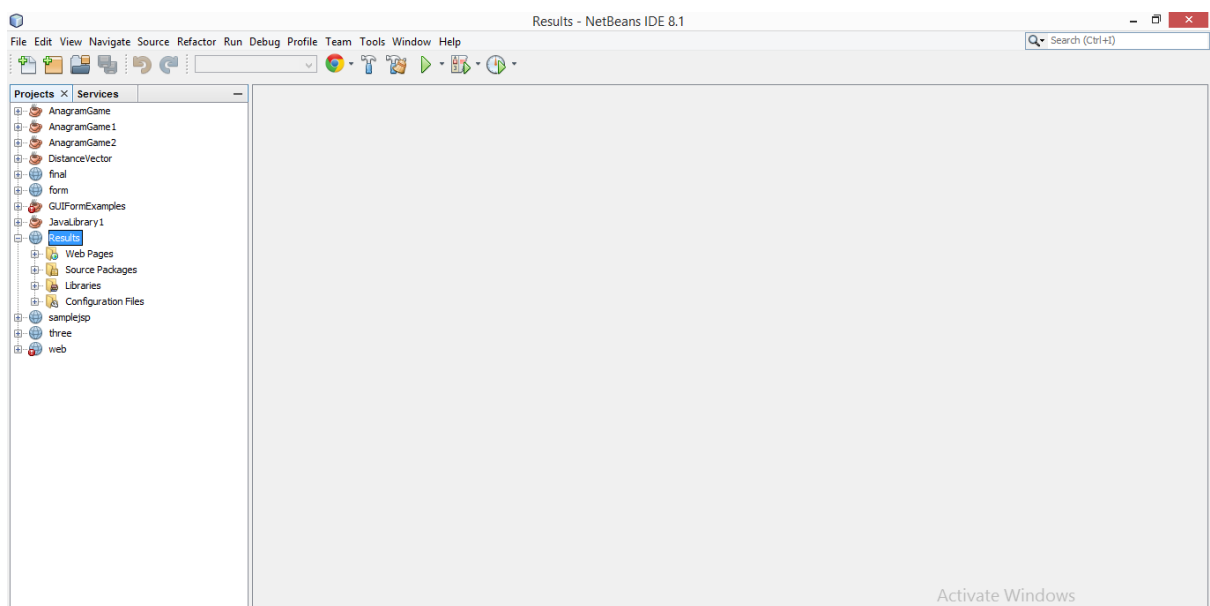
Then type your SQL code.



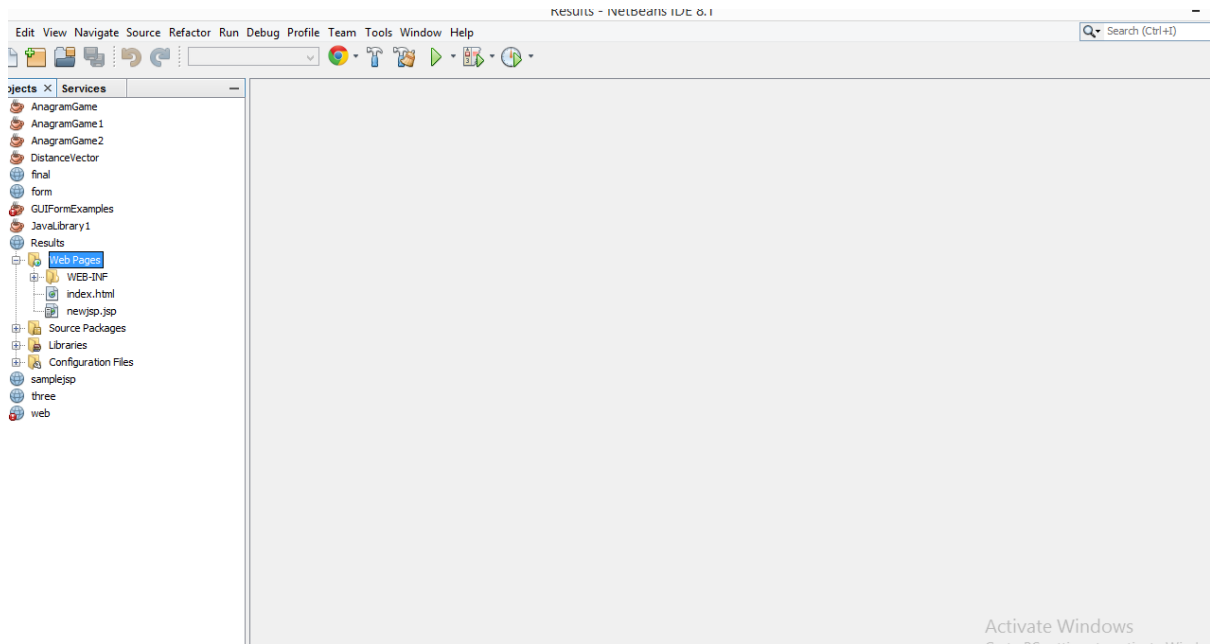
then click this button



now open your project directory



then create index.html and newjsp.jsp



then type your html and jsp code on both file

after step 3 and step 4

now run your project----> click F6

PROGRAM:

SQL code:

```
CREATE TABLE student_marks (  
    reg VARCHAR(20) PRIMARY KEY,  
    subject1 INT,  
    subject2 INT,  
    subject3 INT,  
);  
  
INSERT INTO student_marks (reg, subject1, subject2, subject3)  
VALUES  
( '100', 34,67,45,98),  
( '101', 65,87,45,90),  
( '102', 67,45,34,78);
```

index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Student Details Input</title>

</head>

<body>

<h2>Enter Registration Number:</h2>

<form action="newjsp.jsp" method="post">

<label for="regno">Registration Number:</label>

<input type="text" id="regno" name="regno" required>

<br>

<input type="submit" value="Fetch Details">

</form>

</body>

</html>
```

Result.jsp:

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

<% @ page import="java.sql.*" %>

<% @ page import="java.sql.DriverManager" %>

<% @ page import="java.sql.Connection" %>

<% @ page import="java.sql.PreparedStatement" %>

<% @ page import="java.sql.ResultSet" %>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```

<title>Student Details</title>

</head>

<body>

<%

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

if (regno != null && !regno.isEmpty()) {

Connection connection = null;

try {

// Establish database connection

Class.forName("org.apache.derby.jdbc.ClientDriver");

String url = "jdbc:derby://localhost:1527/sample;create=true";

String username = "app";

String password = "app";

connection = DriverManager.getConnection(url, username, password);

if (connection != null) {

PreparedStatement preparedStatement = connection.prepareStatement("SELECT * FROM
student_marks WHERE reg = ?");

preparedStatement.setString(1, regno);

ResultSet resultSet = preparedStatement.executeQuery();

out.println("<h2>Student Details for Registration Number: " + regno + "</h2>");

if (resultSet.next()) {

int subject1 = resultSet.getInt("subject1");

int subject2 = resultSet.getInt("subject2");

int subject3 = resultSet.getInt("subject3");

int subject4 = resultSet.getInt("subject4");

out.println("Subject 1: " + subject1 + " " +(subject1>40?"P" : "F")+ "<br>");

out.println("Subject 2: " + subject2 + " " +(subject1>40?"P" : "F")+ "<br>");

out.println("Subject 3: " + subject3 + " " +(subject1>40?"P" : "F")+ "<br>");

} else {

out.println("Student not found for Registration Number: " + regno);

```

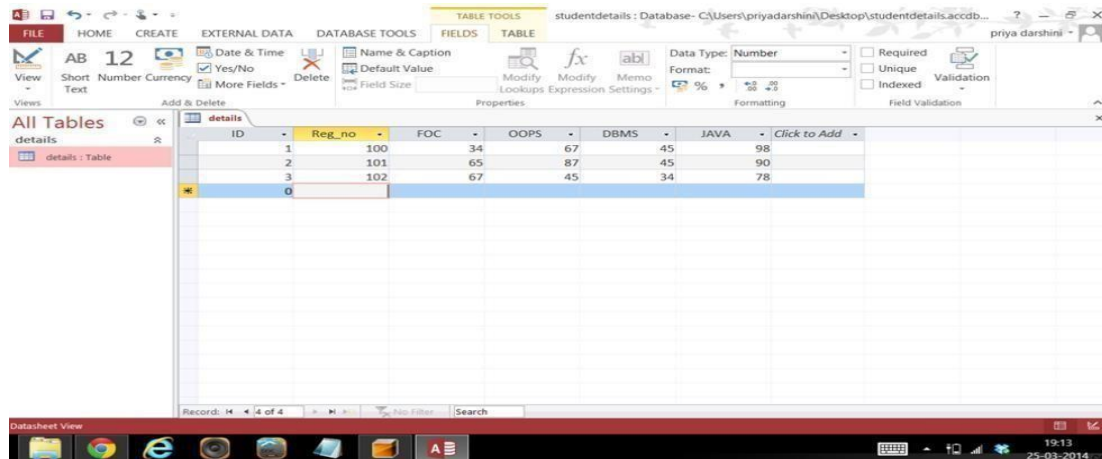


```

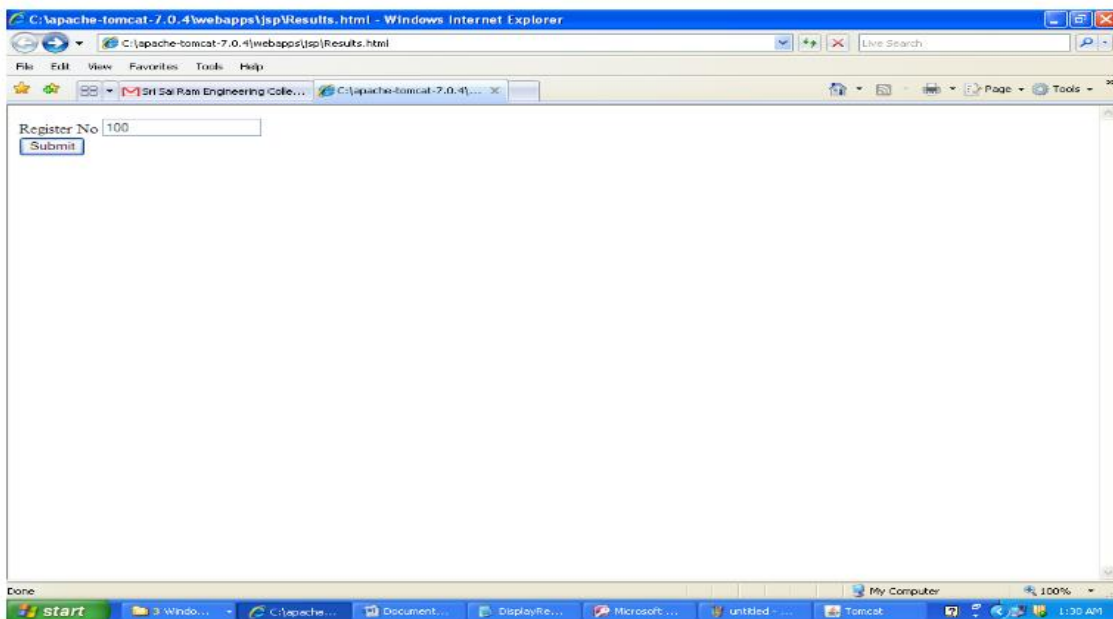
    }
    resultSet.close();
    preparedStatement.close();
    } else {
        out.println("Failed to connect to the database.");
    }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
    } else {
        out.println("<h2>Please enter a Registration Number.</h2>");
    }
    %>
</body>
</html>

```

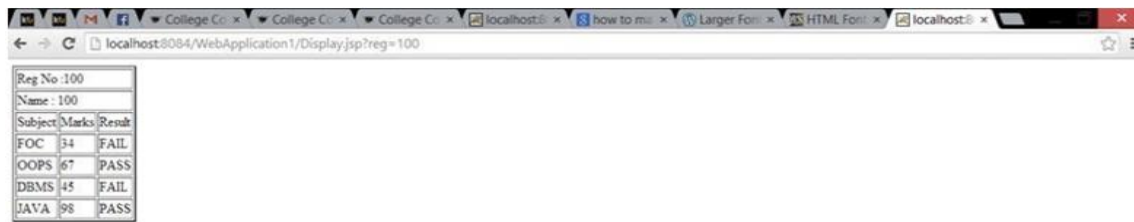
OUTPUT:



ID	Reg_no	FOC	OOPS	DBMS	JAVA	Click to Add
1	100	34	67	45	98	
2	101	65	87	45	90	
3	102	67	45	34	78	
4						



Register No



Reg No :100		
Name : 100		
Subject	Marks	Result
FOC	34	FAIL
OOPS	67	PASS
DBMS	45	FAIL
JAVA	98	PASS

RESULT:

Thus the program for displaying student results and assume that student information is available in a database which has been stored in a database server has been implemented and executed successfully.

EXNO:7 DISPLAYING STUDENT DETAILS USING XML AND XSLT

DATE:

AIM:

To write a program for displaying the student details using XML-XSLT.

ALGORITHM:

Step 1: Create an XML document (stud.xml) that contains the markup tags such as <name>,<dept> and <regno>.

Step 2: Create an XSL document (stud.xsl) that defines the style to display an XML document.

Step 3: Associate style sheet (XSL) with XML using the markup.

Step 4: Load the XML document (stud.xml) in the browser.

PROGRAM:

stud.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="stud.xsl"?>
<studInfo>
<stud>
<name>Abhilash</name>
<dept>IT</dept>
<rno>97709205001</rno>
</stud>
<stud>
<name>Akhila</name>
<dept>IT</dept>
<rno>97709205002</rno>
</stud>
<stud>
<name>Anaswara</name>
<dept>IT</dept>
<rno>97709205003</rno>
</stud>
</studInfo>
```

stud.xsl:

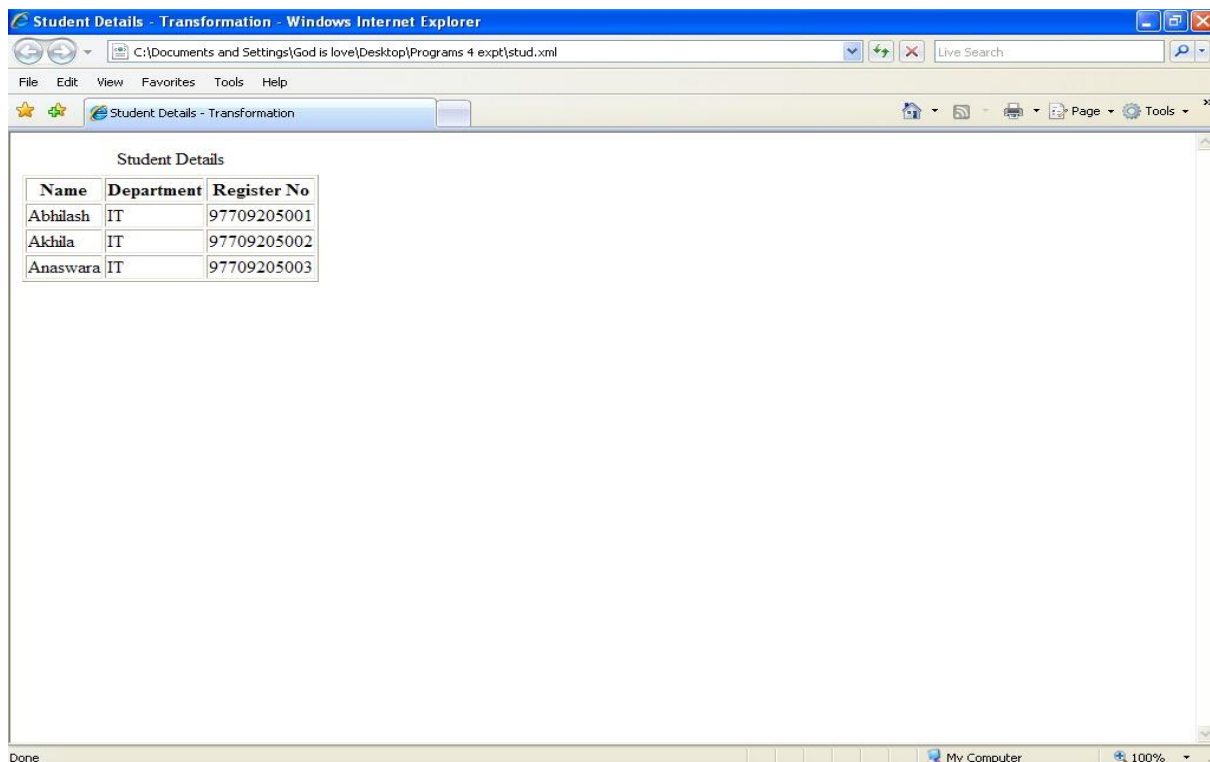
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Student Details - Transformation</title>
```

```

</head>
<body>
<table border="1">
<caption>Student Details</caption>
<tr>
<th>Name</th><th>Department</th><th>Register No</th>
</tr>
<xsl:for-each select="/studInfo/stud">
<tr>
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="dept"/></td>
<td><xsl:value-of select="rno"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:transform>

```

OUTPUT:



RESULT:

Thus the program for displaying the student details using XML-XSLT has been implemented and executed successfully .

EXNO:8a PROGRAM TO PARSE XML FILE USING XML DOM

DATE:

AIM:

To write a program to parse XML file using XML DOM.

ALGORITHM:

Step 1: Start.

Step 2: Create a XML file that has to be parsed.

Step 3: Using DOM parser :

- a) Get a document builder using document builder factory and parse the xml file to create a DOM object.
- b) Get a list of employee elements from the DOM For each employee element get the id, name, age and type.
- c) Create an employee value object and add it to the list.
- d) At the end iterate through the list and print the employees to verify we parsed it right.

Step 4: Using SAX Parser :

- a) Create a Sax parser and parse the xml in the event handler create the employee object.
- b) Print out the data

Step 5: Stop

PROGRAM:

DOM PARSER

Student.xml:

```
<?xml version="1.0"?>
<student>
<Roll_No>10</Roll_No>
<Personal_Info>
<Name>parth</Name>
<Address>pune</Address>
<Phone>1234567890</Phone>
</Personal_Info>
<Class>Second</Class>
```

```
<Subject>Maths</Subject>
<Marks>100</Marks>
<Roll_No>20</Roll_No>
<Personal_Info>
<Name>AnuRadha</Name>
<Address>Bangalore</Address>
<Phone>90901233</Phone>
</Personal_Info>
<Class>Fifth</Class>
<Subject>English</Subject>
<Marks>90</Marks>
<Roll_No>30</Roll_No>
<Personal_Info>
<Name>Anand</Name>
<Address>Mumbai</Address>
<Phone>90901256</Phone>
</Personal_Info>
<Class>Fifth</Class>
<Subject>English</Subject>
<Marks>90</Marks>
</student>
```

Parse.java:

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.*;

public class Parse
{
    public static void main(String[] arg)
```

```

{
try
{
System.out.println("enter the name of xml document");

BufferedReader input=new      BufferedReader(new
InputStreamReader(System.in));

String file_name=input.readLine();

File fp=new File(file_name);

if(fp.exists())
{
try
{

DocumentBuilderFactory Factory_obj=DocumentBuilderFactory.newInstance();
DocumentBuilder

builder=Factory_obj.newDocumentBuilder();

InputSource ip_src=new InputSource(file_name);

Document doc=builder.parse(ip_src);

System.out.println(file_name+" is well-formed!");

}

catch(Exception e)

{

System.out.println(file_name+" isn't well-formed!"); System.exit(1);

}}

else

{

System.out.print("file not found!");

}}

catch(IOException ex)

{

ex.printStackTrace();

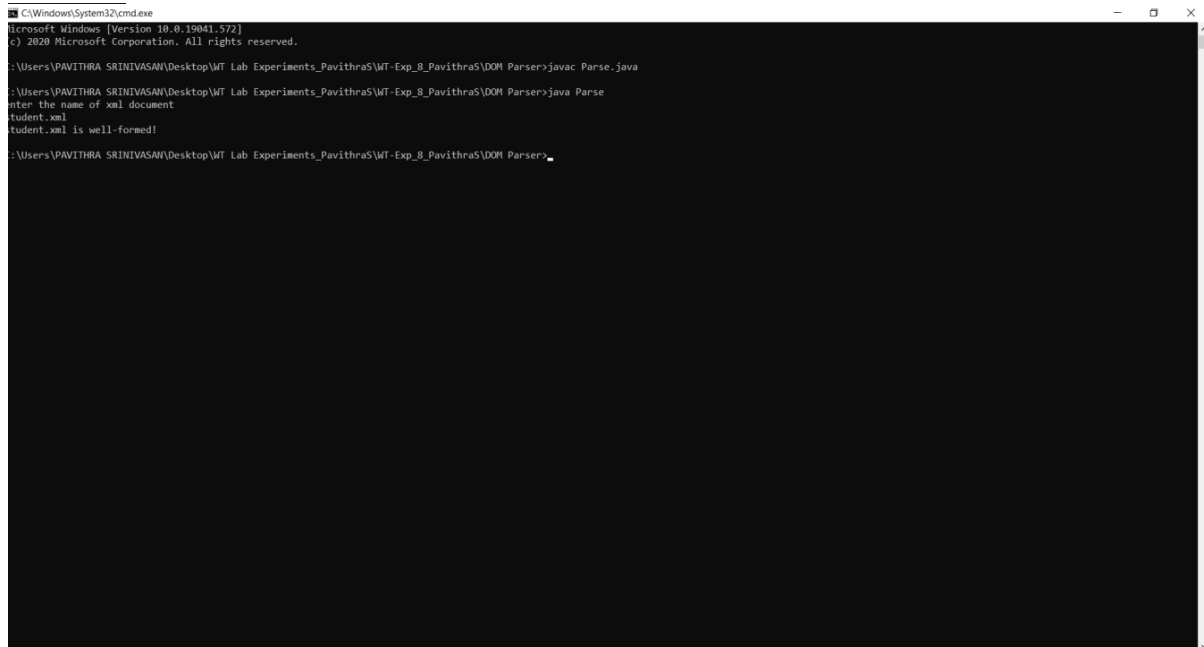
```



```
}}
```

```
}
```

OUTPUT:



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\PAVITHRA SRINIVASAN\Desktop\WT Lab Experiments_Pavithra>javac Parse.java

C:\Users\PAVITHRA SRINIVASAN\Desktop\WT Lab Experiments_Pavithra>java Parse
Enter the name of xml document
student.xml
student.xml is well-formed!

C:\Users\PAVITHRA SRINIVASAN\Desktop\WT Lab Experiments_Pavithra>
```

RESULT:

Thus the program to parse XML file using XML DOM has been implemented and executed successfully.

EXNO:8b PROGRAM TO PARSE XML FILE USING XMLSAX

DATE:

AIM:

To write a program to parse XML file using XMLSAX.

ALGORITHM:

Step 1: Start.

Step 2: Create a XML file that has to be parsed.

Step 3: Using DOM parser :

- e) Get a document builder using document builder factory and parse the xml file to create a DOM object.
- f) Get a list of employee elements from the DOM For each employee element get the id, name, age and type.
- g) Create an employee value object and add it to the list.
- h) At the end iterate through the list and print the employees to verify we parsed it right.

Step 4: Using SAX Parser :

- c) Create a Sax parser and parse the xml in the event handler create the employee object.
- d) Print out the data

Step 5: Stop

PROGRAM:

Employee.xml:

```
<?xml version="1.0"?>
<EmployeeDetail>
<Employee>
<Emp_id>E-001</Emp_id>
<Emp_Name>revathy</Emp_Name>
<Emp_E-mail>revathy@yahoo.com</Emp_E-mail>
</Employee>
<Employee>
<Emp_id>E-002</Emp_id>
```

```

<Emp_Name>vinod</Emp_Name>
<Emp_E-mail>vinod2@yahoo.com</Emp_E-mail>
</Employee>
<Employee>
<Emp_id>E-001</Emp_id>
<Emp_Name>deepak</Emp_Name>
<Emp_E-mail>deepak3@yahoo.com</Emp_E-mail>
</Employee>
</EmployeeDetail>

```

SAXParserCheck.java:

```

import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.io.*;

public class SAXParserCheck
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader bf=new BufferedReader(new InputStreamReader(System.in));
        System.out.print("enter XML file name :"); String xmlfile=bf.readLine();
        SAXParserCheck par=new SAXParserCheck(xmlfile);
    }

    public SAXParserCheck(String str)
    {
        try
        {
            File file=new File(str); if(file.exists())
            {
                XMLReader reader=XMLReaderFactory.createXMLReader(); reader.parse(str);
                System.out.println(str+" is well-formed!");
            }
        }
    }
}

```

```

    }
    else
    {
        System.out.println("File not found:"+str);
    }
}
catch(SAXException sax)
{
    System.out.println(str+" isn't well-formed");
}
catch(IOException io)
{
    System.out.println(io.getMessage());
}
}
}
}

```

Web.xml:

```

<web-app>
<servlet>
<servlet-name>sample</servlet-name>
<servlet-class>sample</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>sample</servlet-name>
<url-pattern>/sample</url-pattern>
    105
</servlet-mapping>
<session-config>

```

<session-timeout>

30

</session-timeout>

</session-config>

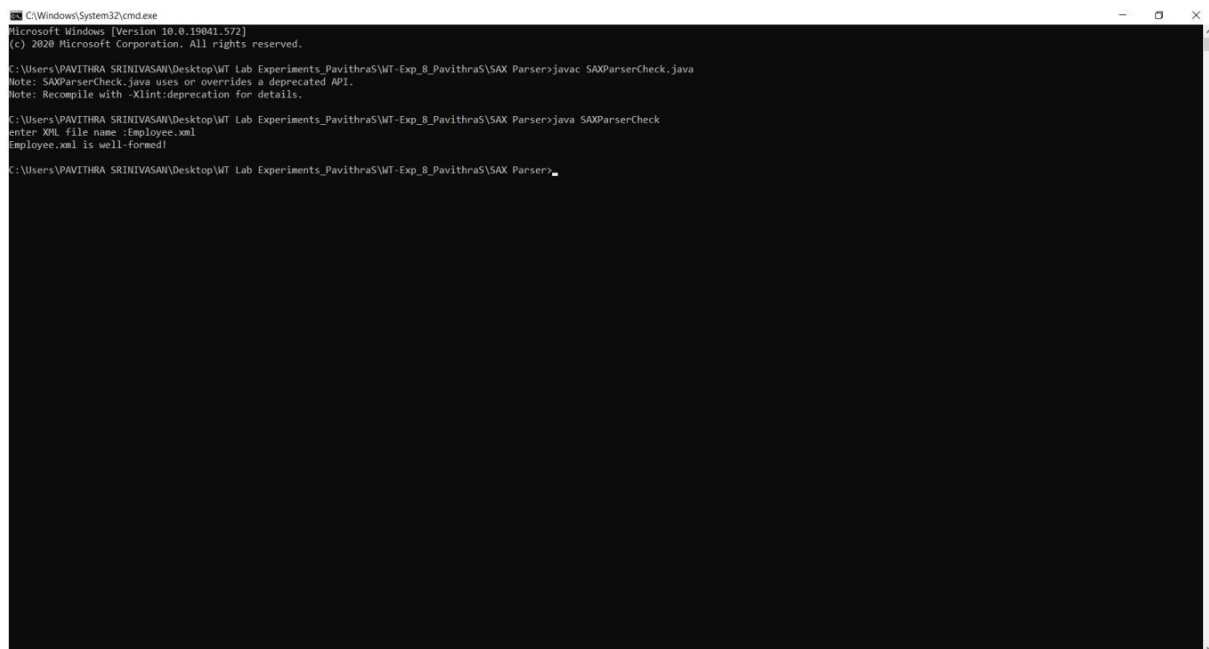
<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

</web-app>

OUTPUT :



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\PAVITHRA SRINIVASAN\Desktop\WT Lab Experiments_PavithraS\WT-Exp_8_PavithraS\SAX Parser>javac SAXParserCheck.java
Note: SAXParserCheck.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\PAVITHRA SRINIVASAN\Desktop\WT Lab Experiments_PavithraS\WT-Exp_8_PavithraS\SAX Parser>java SAXParserCheck
Enter XML file name :Employee.xml
Employee.xml is well-formed!

C:\Users\PAVITHRA SRINIVASAN\Desktop\WT Lab Experiments_PavithraS\WT-Exp_8_PavithraS\SAX Parser>
```

RESULT:

Thus the program to parse XML file using XMLSAX has been implemented and executed successfully.

EXNO:9

CREATE A WEB PAGE USING AJAX

DATE:

AIM:

To create a Web page that updates a page without reloading using AJAX.

ALGORITHM:

Step 1: Start.

Step 2: Create a simple webpage with a textbox and a Button.

Step 3: Pass the input number given in the text box to XMLHttpRequest on Button click.

Step 4: After processing the request the page will be updated with the Multiplicative tables of the given number without reloading.

Step 5: Check the output.

Step 6: Stop.

PROGRAM:

Ajax.html:

```
<!DOCTYPE html>

<html>

<head>

<script>

var request;

function sendInfo()

{

var v=document.vinform.t1.value;

var url="index.jsp?val="+v;

if(window.XMLHttpRequest){

request=new XMLHttpRequest();

}

else if(window.ActiveXObject){

request=new ActiveXObject("Microsoft.XMLHTTP");

}
```

```

try
{
request.onreadystatechange=getInfo;
request.open("GET",url,true);
request.send();
}
catch(e)
{
alert("Unable to connect to server");
}
}

function getInfo(){
if(request.readyState==4){
var val=request.responseText;
document.getElementById('amit').innerHTML=val;
}
}

**Create a Web Page Using AJAX**.

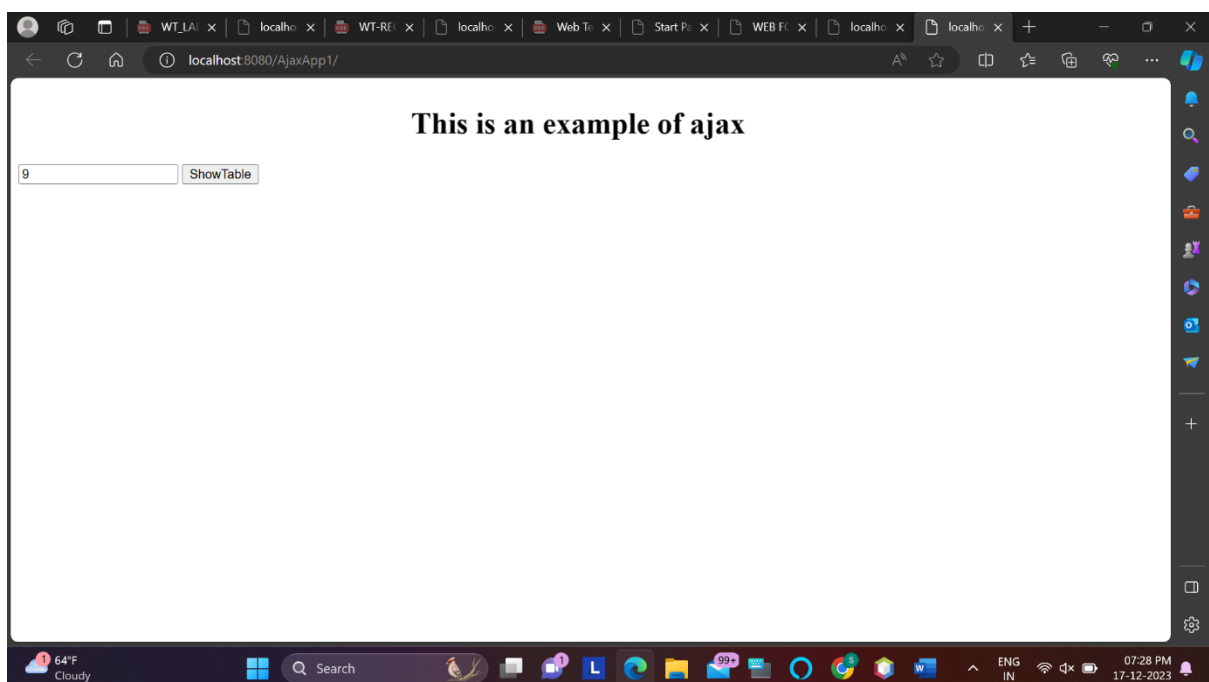
</script>
</head>
<body>
<marquee><h1>This is an example of ajax</h1></marquee>
<form name="vinform">
<input type="text" name="t1">
<input type="button" value="ShowTable" onClick="sendInfo()">
</form>
<span id="amit"> </span>
</body>
</html>

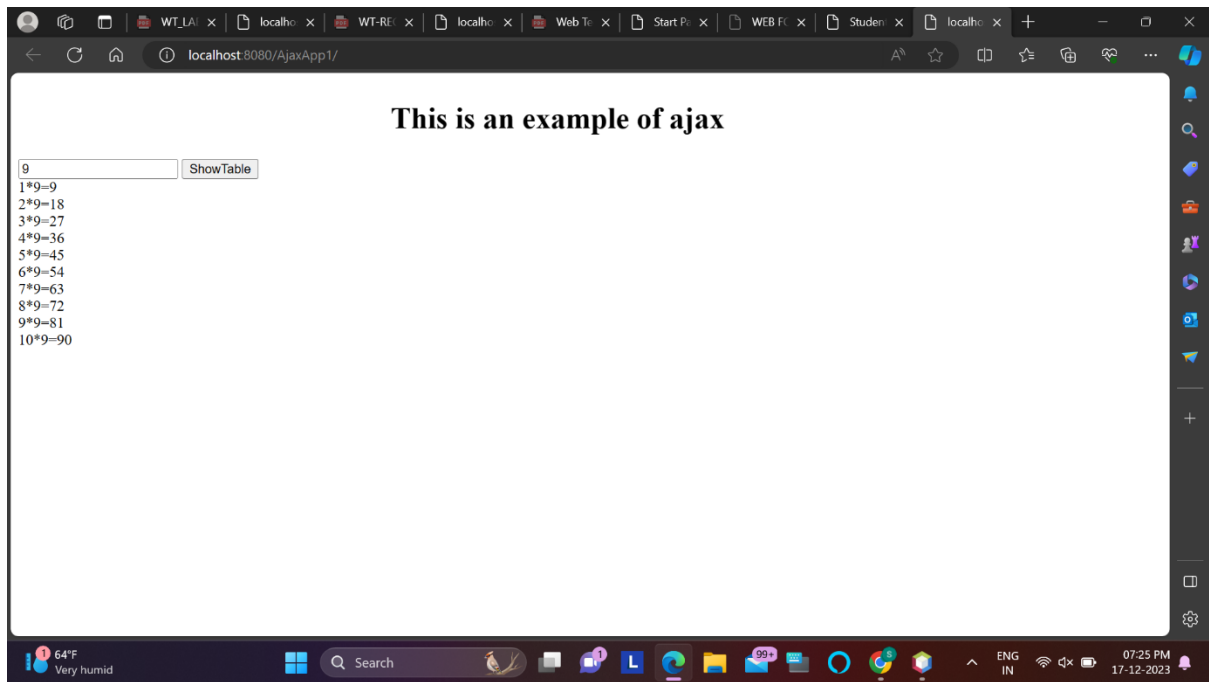
```

Index.jsp:

```
<%  
int n=Integer.parseInt(request.getParameter("val"));  
for(int i=1;i<=10;i++)  
out.print(i*n+"<br>");  
%>
```

OUTPUT:





RESULT:

Thus a Web page that updates a page without reloading using AJAX is created ,executed successfully and output is verified.

EXNO:10

CALCULATOR WEB SERVICE

AIM:

To develop Calculator web service.

ALGORITHM:

SERVER SIDE PROCEDURE:

Step 1: Create new project

- a) New-> Project -> JavaEE -> EJBModule -> Next -> Give Project Name -> Finish.

Step 2: Create new Web service

- b) Right click the created project name -> New -> Web Service -> Give name for Web service and Package -> Finish
- c) A new folder named “Web Services” will be created under the project folder. Inside that, new web services will be created.

Step 3: Creating functions for web service

- a) Under the web services folder, right click the web service created -> Add Operation
 - i. Give appropriate name for the operation.
 - ii. Select the return type of the function.
 - iii. Adding parameters :
 - iv. Click Add -> give name of the parameter and their data type.
 - v. Follow the above step to add more parameters.
 - vi. Click ok.

Step 4: Write the required operation/statements inside the function and also see to that the return type matches. Save the file.

Step 5: Deploying and testing :

- a) Right click the project -> Deploy
- b) In the web services folder, right click on the web service -> Test Web service.

CLIENT SIDE PROCEDURE:

Step 1: Create a new project

- a) File -> New Project -> JavaWeb -> We Application ->Next -> Give a name -> Finish.

Step 2: A file names index.jsp will be created. Modify the code as below:

```
<body>
    <label>a</label><input type="text" name="n1"/></br>
    <label>b</label><input type="text" name="n2"/></br>
<form name="form1" action="add.jsp" method="get">
<input type="submit" value="Add"/>
</form>
<form name="form2" action="sub.jsp" method="get">
<input type="submit" value="Sub"/>
</form>
<form name="form3" action="mul.jsp" method="get">
<input type="submit" value="Multiply"/>
</form>
<form name="form4" action="div.jsp" method="get">
```

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

Step 3: Right click the created project -> New -> JSP -> Give name (add.jsp) -> Finish.

Step 4: Create web service client.

- b) Right click the created project -> New -> Web Service Client.
- c) Enter the WSDL URL that can get form the following steps:
 - i. In the firstly created EJB Module project, right click the Web Service under the Web Services folder -> properties.
 - ii. URL will be displayed. Copy it and paste it in the WSDL URL field.
 - iii. Click Finish.

Step 5: Right click the payroll.jsp file in the editor -> Web Service Client Resources -> Call Web Service Operation. In the dialog box that opens, click the '+' sign until to get the required operation -> click Ok. The code for generating Web Service operation will be generated.

Step 6: Modify the JSP file.

Step 7: Right click the Web Application project -> Deploy.

Step 8: Right click the Web Application project -> Run.

Step 9: Index.jsp file will be displayed. Give appropriate inputs and click "Submit".

Step 10: Result will be displayed.

PROGRAM:

SERVER SIDE CODING:

CalcWebService.java:

package pack;

```
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.ejb.Stateless;
/**
 *
 * @author hari
 */
@WebService()
@Stateless()
public class CalcWebService {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "add")
    public double add(@WebParam(name = "a")
        double a, @WebParam(name = "b")
        double b) {
        //TODO write your implementation code here:
        return a+b;
    }
}
```

```

    * Web service operation
    */
    @WebMethod(operationName = "sub")
    public double sub(@WebParam(name = "a")
    double a, @WebParam(name = "b")
    double b) {
        //TODO write your implementation code here:
        return a-b;
    }
}
/**
    * Web service operation
    */
    @WebMethod(operationName = "mul")
    public double mul(@WebParam(name = "a")
    double a, @WebParam(name = "b")
    double b) {
        //TODO write your implementation code here:
        return a*b;
    }
}
/**
    * Web service operation
    */
    @WebMethod(operationName = "div")
    public double div(@WebParam(name = "a")
    double a, @WebParam(name = "b")
    double b) {
        //TODO write your implementation code here:
        return a/b;
    }
}
}

```

CLIENT SIDE CODING:

Index.jsp:

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Calc JSP</title>
</head>
<body>
    <form name="form1" action="sub.jsp" method="get">
        Number 1<input type="text" name="n1"> <br/>
        Number 2<input type="text" name="n2"> <br/>
        <input type="Submit" value="Subtract"><br/>
    </form>
    <form name="form1" action="add.jsp" method="get">

```

```

        Number 1<input type="text" name="n1"> <br/>
        Number 2<input type="text" name="n2"> <br/>
        <input type="Submit" value="Add"><br/>
    </form>
    <form name="form1" action="div.jsp" method="get">
        Number 1<input type="text" name="n1"> <br/>
        Number 2<input type="text" name="n2"> <br/>
        <input type="Submit" value="Division"><br/>
    </form>
    <form name="form1" action="mul.jsp" method="get">
        Number 1<input type="text" name="n1"> <br/>
        Number 2<input type="text" name="n2"> <br/>
        <input type="Submit" value="Multiplication"><br/>
    </form>
</body>
</html>

```

Sub.jsp:

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
</head>
<body>
<%! double mulres,divres,addres,subres,a,b;%>
<%
    try {
        pack.CalcWebServiceService service = new pack.CalcWebServiceService();
        pack.CalcWebService port = service.getCalcWebServicePort();
        // TODO initialize WS operation arguments here
        a = Double.parseDouble(request.getParameter("n1"));
        b = Double.parseDouble(request.getParameter("n2"));
        // TODO process result here
        subres = port.sub(a, b);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
%>
    Number 1<input type="text" value="<%=a%>"><br/>
    Number 2<input type="text" value="<%=b%>"><br/>
    Subtracted Value<input type="text" value="<%=subres%>">
</body>
</html>

```

Add.jsp:

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%! double addres,a,b;%>
    <%
    try {
      pack.CalcWebServiceService service = new pack.CalcWebServiceService();
      pack.CalcWebService port = service.getCalcWebServicePort();
      // TODO initialize WS operation arguments here
      a = Double.parseDouble(request.getParameter("n1"));
      b = Double.parseDouble(request.getParameter("n2"));
      // TODO process result here
      addres = port.add(a, b);
    } catch (Exception ex) {
      // TODO handle custom exceptions here
    }
    %>
    Number 1<input type="text" value="<%=a%>"><br/>
    Number 2<input type="text" value="<%=b%>"><br/>
    Added Value<input type="text" value="<%=addres%>">
  </body>
</html>

```

Mul.jsp:

```

<% @page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <%! double mulres,a,b;%>
    <%
    try {
      pack.CalcWebServiceService service = new pack.CalcWebServiceService();
      pack.CalcWebService port = service.getCalcWebServicePort();
      // TODO initialize WS operation arguments here
      a = Double.parseDouble(request.getParameter("n1"));
      b = Double.parseDouble(request.getParameter("n2"));
      // TODO process result here
      mulres= port.mul(a, b);
    }
    %>
  </body>
</html>

```

```

    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
%>
Number 1<input type="text" value="<%=a%>"><br/>
Number 2<input type="text" value="<%=b%>"><br/>
Multiplied Value<input type="text" value="<%=mulres%>">
</body>
</html>

```

Div.jsp:

```

<%--
    Document : div
    Created on : Aug 1, 2011, 10:47:02 PM
    Author : hari
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
</head>
<body>
<%! double divres,a,b;%>
<%
    try {
        pack.CalcWebServiceService service = new pack.CalcWebServiceService();
        pack.CalcWebService port = service.getCalcWebServicePort();
        // TODO initialize WS operation arguments here
        a = Double.parseDouble(request.getParameter("n1"));
        b = Double.parseDouble(request.getParameter("n2"));

        divres = port.div(a, b);
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
%>
Number 1<input type="text" value="<%=a%>"><br/>
Number 2<input type="text" value="<%=b%>"><br/>
Quotient Value<input type="text" value="<%=divres%>">
</body>
</html>

```

OUTPUT:

Server

Web

Service:

Method invocation trace

localhost:8080/CalcWebServiceService/CalcWebService?Tester

sub Method invocation

Method parameter(s)

Type	Value
double	10
double	5

Method returned

double : "5.0"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:sub xmlns:ns2="http://pack/">
      <a>10.0</a>
      <b>5.0</b>
    </ns2:sub>
  </S:Body>
</S:Envelope>
```

SOAP Response



SOAP Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:add xmlns:ns2="http://pack/">
      <a>5.0</a>
      <b>5.0</b>
    </ns2:add>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:addResponse xmlns:ns2="http://pack/">
      <return>10.0</return>
    </ns2:addResponse>
  </S:Body>
</S:Envelope>
```

CalcWebService.wsdl:


```

<?xml version='1.0' encoding='UTF-8'?>
<!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.1-
hudson-28-. -->
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.1-
hudson-28-. -->
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://pack/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://pack/"
name="CalcWebServiceService">
<wsp:Policy xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsat"
wsu:Id="CalcWebServicePortBinding_div_WSAT_Policy">
<wsat:ATAlwaysCapability />
<wsat:ATAssertion xmlns:ns1="http://schemas.xmlsoap.org/ws/2002/12/policy"
wsp:Optional="true" ns1:Optional="true" />
</wsp:Policy>
<wsp:Policy xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsat"
wsu:Id="CalcWebServicePortBinding_sub_WSAT_Policy">
<wsat:ATAlwaysCapability />
<wsat:ATAssertion xmlns:ns2="http://schemas.xmlsoap.org/ws/2002/12/policy"
wsp:Optional="true" ns2:Optional="true" />
</wsp:Policy>
<wsp:Policy xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsat"
wsu:Id="CalcWebServicePortBinding_add_WSAT_Policy">
<wsat:ATAlwaysCapability />
<wsat:ATAssertion xmlns:ns3="http://schemas.xmlsoap.org/ws/2002/12/policy"
wsp:Optional="true" ns3:Optional="true" />
</wsp:Policy>
<wsp:Policy xmlns:wsat="http://schemas.xmlsoap.org/ws/2004/10/wsat"
wsu:Id="CalcWebServicePortBinding_mul_WSAT_Policy">
<wsat:ATAlwaysCapability />
<wsat:ATAssertion xmlns:ns4="http://schemas.xmlsoap.org/ws/2002/12/policy"
wsp:Optional="true" ns4:Optional="true" />
</wsp:Policy>
</types>
<xsd:schema>
<xsd:import namespace="http://pack/"
schemaLocation="http://localhost:8080/CalcWebServiceService/CalcWebService?xsd=1" />
</xsd:schema>
</types>
<message name="add">
<part name="parameters" element="tns:add" />
</message>
<message name="addResponse">
<part name="parameters" element="tns:addResponse" />
</message>

```

```

<message name="sub">
<part name="parameters" element="tns:sub" />
</message>
<message name="subResponse">
<part name="parameters" element="tns:subResponse" />
</message>
<message name="mul">
<part name="parameters" element="tns:mul" />
</message>
<message name="mulResponse">
<part name="parameters" element="tns:mulResponse" />
</message>
<message name="div">
<part name="parameters" element="tns:div" />
</message>
<message name="divResponse">
<part name="parameters" element="tns:divResponse" />
</message>
<portType name="CalcWebService">
<operation name="add">
<input wsam:Action="http://pack/CalcWebService/addRequest" message="tns:add" />
<output wsam:Action="http://pack/CalcWebService/addResponse"
message="tns:addResponse" />
</operation>
<operation name="sub">
<input wsam:Action="http://pack/CalcWebService/subRequest" message="tns:sub" />
<output wsam:Action="http://pack/CalcWebService/subResponse"
message="tns:subResponse" />
</operation>
<operation name="mul">
<input wsam:Action="http://pack/CalcWebService/mulRequest" message="tns:mul" />
<output wsam:Action="http://pack/CalcWebService/mulResponse"
message="tns:mulResponse" />
</operation>
<operation name="div">
<input wsam:Action="http://pack/CalcWebService/divRequest" message="tns:div" />
<output wsam:Action="http://pack/CalcWebService/divResponse"
message="tns:divResponse" />
</operation>
</portType>
<binding name="CalcWebServicePortBinding" type="tns:CalcWebService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
<operation name="add">
<wsp:PolicyReference URI="#CalcWebServicePortBinding_add_WSAT_Policy" />
<soap:operation soapAction="" />
<input>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_add_WSAT_Policy" />
<soap:body use="literal" />
</input>
<output>

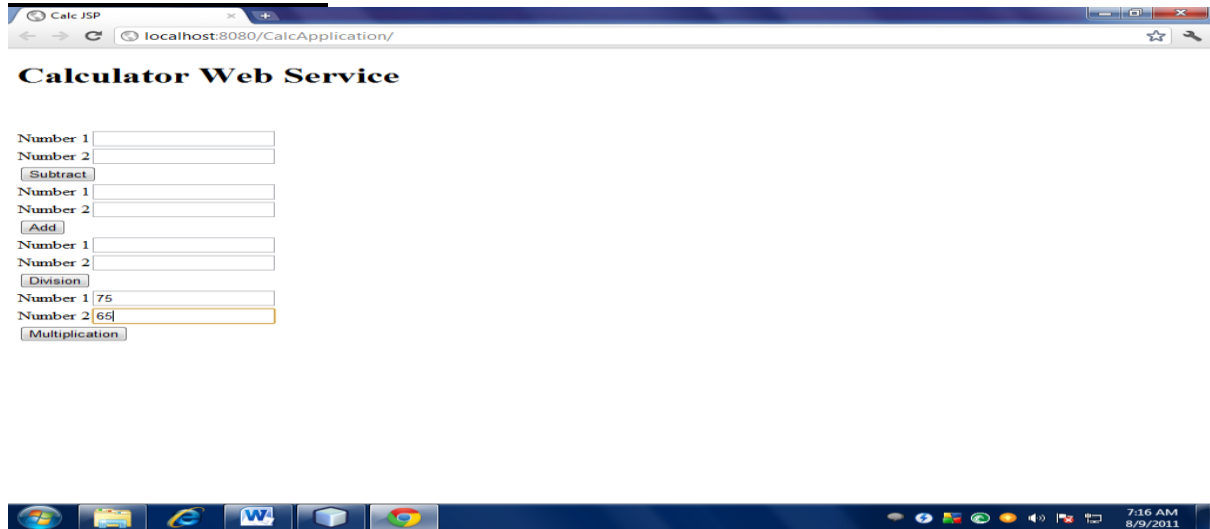
```

```

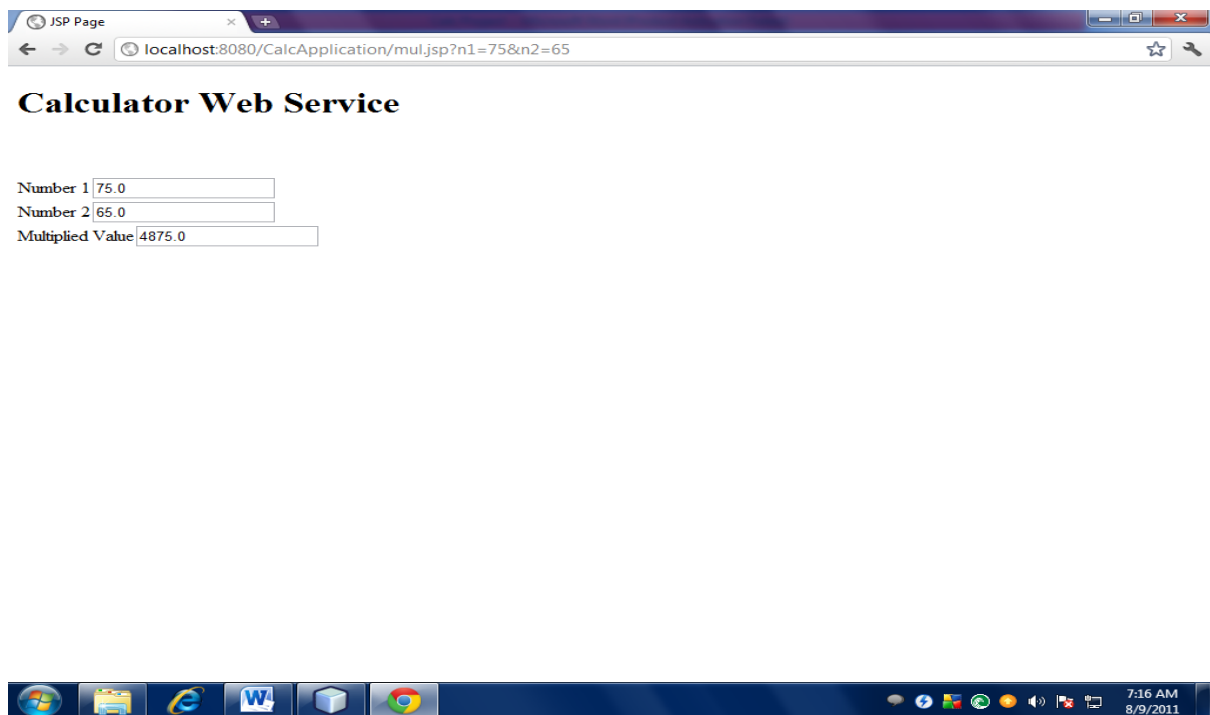
<wsp:PolicyReference URI="#CalcWebServicePortBinding_add_WSAT_Policy" />
<soap:body use="literal" />
</output>
</operation>
<operation name="sub">
<wsp:PolicyReference URI="#CalcWebServicePortBinding_sub_WSAT_Policy" />
<soap:operation soapAction="" />
<input>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_sub_WSAT_Policy" />
<soap:body use="literal" />
</input>
<output>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_sub_WSAT_Policy" />
<soap:body use="literal" />
</output>
</operation>
<operation name="mul">
<wsp:PolicyReference URI="#CalcWebServicePortBinding_mul_WSAT_Policy" />
<soap:operation soapAction="" />
<input>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_mul_WSAT_Policy" />
<soap:body use="literal" />
</input>
<output>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_mul_WSAT_Policy" />
<soap:body use="literal" />
</output>
</operation>
<operation name="div">
<wsp:PolicyReference URI="#CalcWebServicePortBinding_div_WSAT_Policy" />
<soap:operation soapAction="" />
<input>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_div_WSAT_Policy" />
<soap:body use="literal" />
</input>
<output>
<wsp:PolicyReference URI="#CalcWebServicePortBinding_div_WSAT_Policy" />
<soap:body use="literal" />
</output>
</operation>
</binding>
<service name="CalcWebServiceService">
<port name="CalcWebServicePort" binding="tns:CalcWebServicePortBinding">
<soap:address location="http://localhost:8080/CalcWebServiceService/CalcWebService" />
</port>
</service>
</definitions>

```

Client Web Service:



Mul.jsp:



RESULT:

Thus the program to create a calculator with basic operations using EJB module and web service has been implemented and executed successfully.

EXNO:11 DESIGN A LOGIN PAGE USING ANGULAR FRAMEWORK

DATE:

AIM:

To write a program for displaying the login page using angular framework.

ALGORITHM:

REQUIREMENTS:

- 1.Node js
- 2.Angular/cli package
- 3.vs code editor

Step 1: First install node in your system

click to install node js -----> <https://nodejs.org/en>

Step 2: Then install vs code

click this link to install ----> https://code.visualstudio.com/updates/v1_25

Step 3: Then open cmd

then install anugular using this commands

npm install -g @angular/cli

Step 4: Now, create a new Angular project:

ng new angular-demo-login

cd angular-demo-login

Step 5: Next, create a new component for the login page:

ng generate component login

Step 6: Now open your project folder type code . in cmd

Step 7: Now, open the **src/app/login/login.component.html** file and replace its content with the following code.

PROGRAM:

```
<div class="login-container">

  <h2>Login</h2>

  <form (ngSubmit)="login()">

    <div class="form-group">
```

```

    <label for="username">Username:</label>

    <input type="text" id="username" name="username" [(ngModel)]="username" required>
  </div>

  <div class="form-group">

    <label for="password">Password:</label>

    <input type="password" id="password" name="password" [(ngModel)]="password"
required>
  </div>

  <button type="submit">Login</button>

</form>
</div>

```

- Now, open the **src/app/login/login.component.ts** file and replace its content with the following:

```

import { Component } from '@angular/core';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {
  username: string = "";
  password: string = "";
  loggedIn: boolean = false;

  login() {
    // You can add authentication logic here

    console.log(`Username: ${this.username}, Password: ${this.password}`);

    // Add logic to check authentication (e.g., with a service)
  }
}

```

```
// For now, just set loggedIn to true
this.loggedIn = true;

// Clear username and password fields
this.username = "";
this.password = "";
}
}
```

- Open **src/app/app.component.html** and replace its content with:

```
<div style="text-align:center">
  <app-login></app-login>
</div>
```

- create **src/app/app.module.ts** and replace its content with:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms'; // Add this line

import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule, // Add this line
```

```
],  
providers: [],  
bootstrap: [AppComponent]  
}))  
export class AppModule { }
```

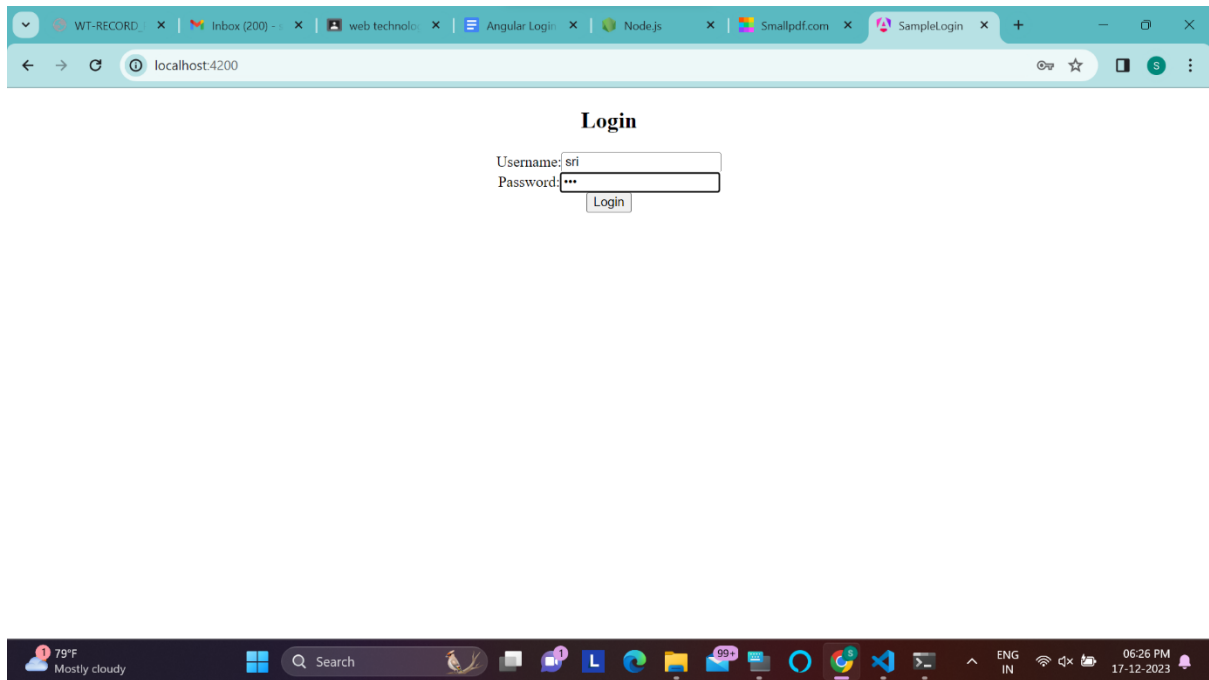
Now, you should be able to run your Angular app type command on terminal

ng serve

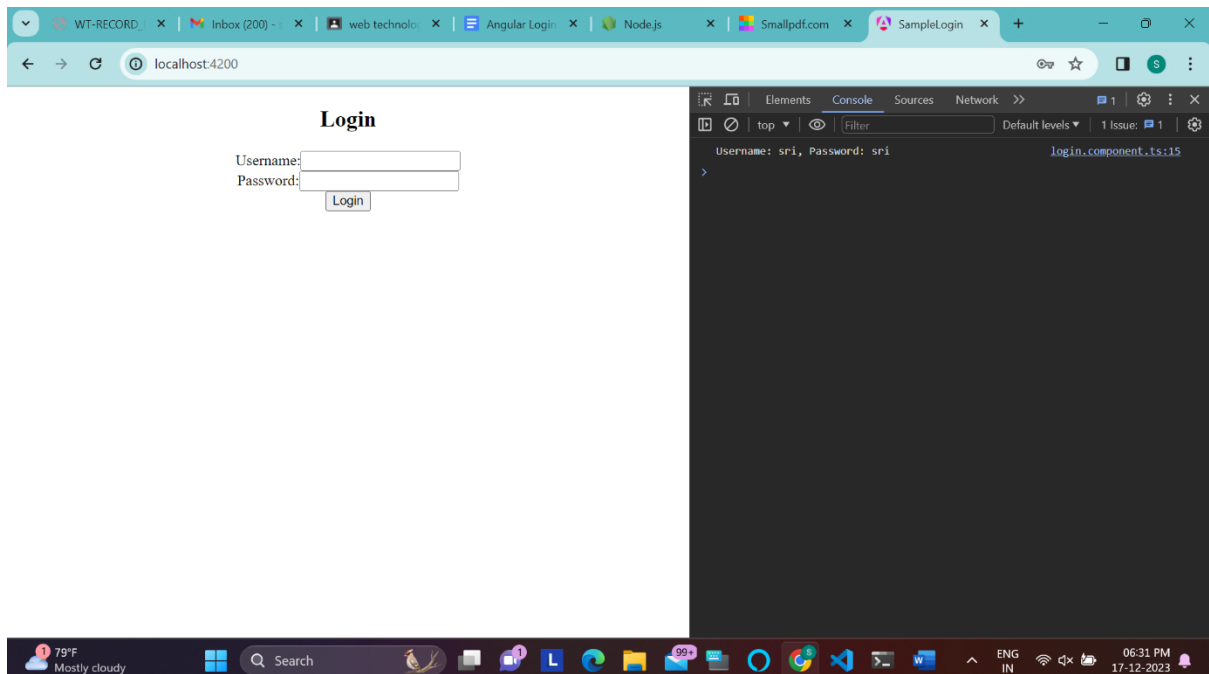
then visit **<http://localhost:4200/>**

OUTPUT:

BEFORE SUBMIT:



AFTER SUBMIT:



RESULT:

Thus the program for displaying the login page using angular framework has been implemented and executed successfully.