# UNIVERSITY COLLEGE OF ENGINEERING

# TINDIVANAM

(A Constituent college of Anna University, Chennai)

## DEPARTMENT OF INFORMATION TECHNOLOGY

## LABORATORY RECORD

## 2020-2021

This is to certify that this is a bonafide report of the work done by Mr. **T.ELAVARASAN** Register number **422418205013** of **Third Year B.Tech**., Department of Information Technology in the **Web Technology Laboratory (IT8511)** in the **V semester**.

University Examination held on: **04/08/2021**

**Staff in charge** **Head of the Department**

**Internal Examiner** **External Examiner**

| EX NO | DATE | LIST OF EXPERIMENT | MARK | SIGNATURE |
|---|---|---|---|---|
| 1 | 18/08/2020 | Create a web page with the following using HTML.<br>i) To embed an image map in a web page. ii) To fix the hot spots.<br>iii) Show all the related information when the hot spots are clicked | | |
| 2 | 04/09/2020 | Create a web page with all types of Cascading style sheets. | | |
| 3 | 11/09/2020 | Client Side Scripts for Validating Web Form Controls using DHTML. | | |
| 4 | 19/09/2020 | Installation of Apache Tomcat web server. | | |
| 5 | 01/10/2020 | Write programs in Java using Servlets: To invoke servlets from HTML forms. Session Tracking. | | |
| 6 | 08/10/2020 | Write programs in Java to create three-tier applications using JSP and Databases<br>● For conducting on-line examination.<br>● For displaying student mark list. Assume that student information is available in a database which has been stored in a database server. | | |
| 7 | 17/10/2020 | Programs using XML – Schema – XSLT/XSL. | | |
| 8 | 24/10/2020 | Programs using DOM and SAX parsers. | | |
| 9 | 29/10/2020 | Programs using AJAX. | | |
| 10 | 31/10/2020 | Consider a case where we have two web Services- an airline service and a travel agent and the travel agent is searching for an airline. Implement this scenario using Web Services and Data base. | | |

**Ex No: 1**

# CREATION OF IMAGE MAP TO FIX HTOSPOTS

**AIM**

To create a web page with image map to fix the hotspots and show its related information.

**SOFTARE USED**

**Dreamweaver or Notepad and browser.**

**DESCRIPTION**

In Web page development, an image map is a graphic image defined so that a user can click on different areas of the image and be linked to different destinations. You make an image map by defining each of the sensitive areas in terms of their x and y coordinates (that is, a certain horizontal distance and a certain vertical distance from the left-hand corner of the image). With each set of coordinates, you specify a Uniform Resource Locator or Web address that will be linked to when the user clicks on that area.
There are three HTML elements used to create image maps:
<img> specifies the location of the image to be included in the map.
<map> is used to create the map of clickable areas.
<area> is used within the map element to define the clickable areas.

**CODE**
**main.html**

```
<html>
<head>
<BODY bgcolor="#gop6876cgdt5564ss">
<img src ="indiamap.jpeg" usemap="#indiamap" />
<map name=indiamap>
<area shape="circle" coords="500,1092,15" href="tamilnadu.html" alt="Tamilnadu"> <area
shape="rect" coords="376,1076,406,1100" href="karnataka.html" alt="Karnataka"> </map>
</head>
</BODY>
</html>
```

**tamilnadu.html**

```
<html>
<head><title>Tanil Nadu - India</title></head>
<body bgcolor="palegreen">
<h1><center>Tamil Nadu</center></h1>
<h3>is one of the 29 states of India. Its capital and largest city is Chennai.
Tamil Nadu lies in the southernmost part of the Indian Peninsula and
is bordered by the States of puducherry, Kerala, Karnataka, Andha Pradesh.
</h3>
<h3>
<ul>
```

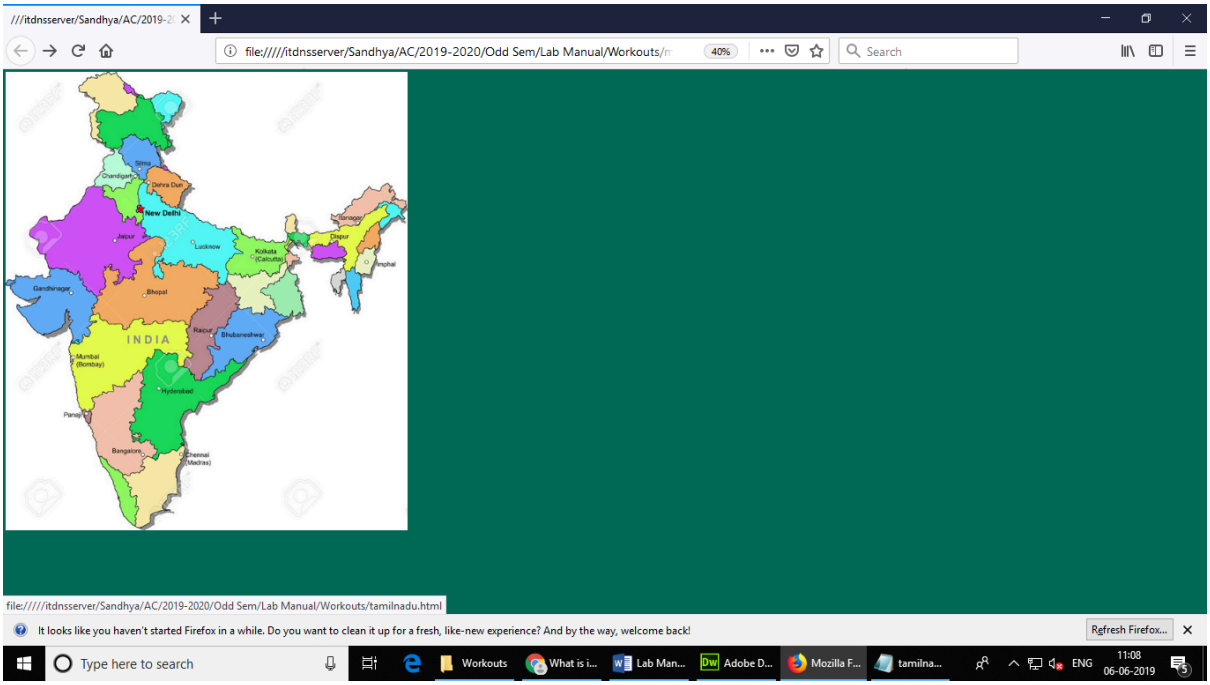
<li>Districts - 33
<li>Capital City - Chennai
<li>Largest City - Chennai
<li>Governor - Banwarilal Purohit
<li>Chief Minister - Edappadi K. Palaniswami 
<li>Population - 72,147,030

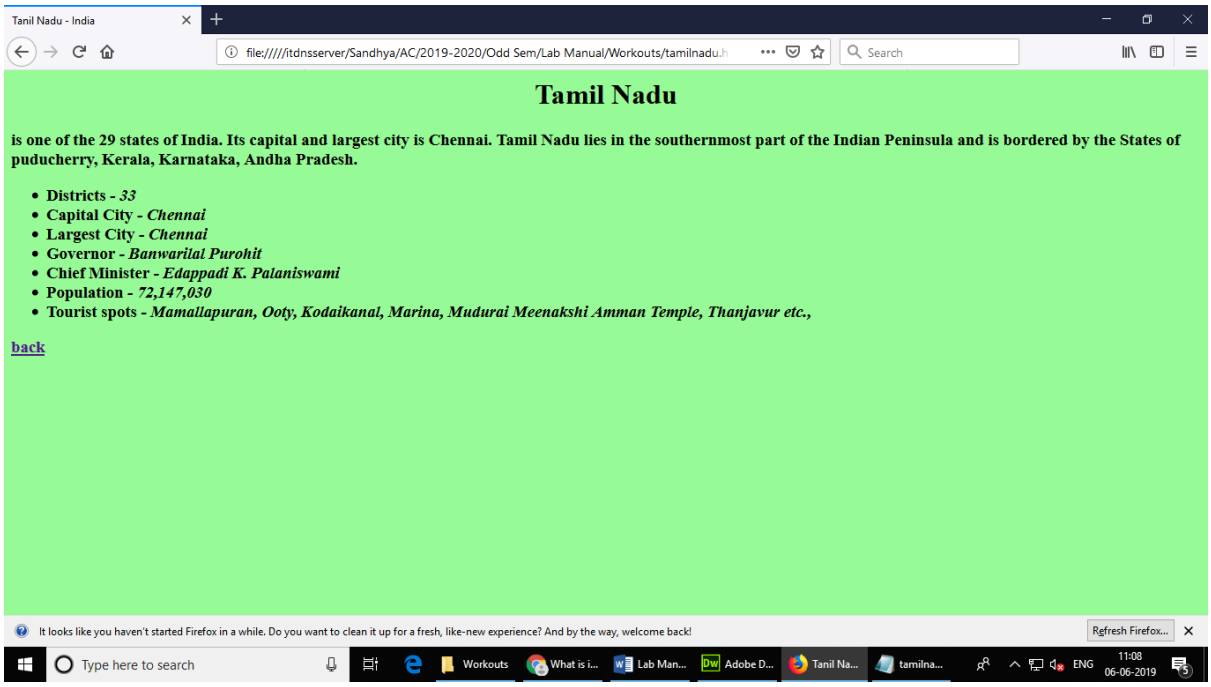
```
<li>Tourist spots<i> - Mamallapuran, Ooty, Kodaikanal, Marina,
Mudurai Meenakshi Amman Temple, Thanjavur etc.,</i>
</ul>
<a href="main.html">back</a>
</body>
</html>
```

**karnataka.html**

```
<html>
<head><title>Karnataka - India</title></head>
<body bgcolor="wheat">
<h1><center>Karnataka</center></h1>
<h3>
<ul>
<li>Districts<i> - 30</i>
<li>Capital City<i> - Bangalore</i>
<li>Largest City<i> - Bangalore</i>
<li>Governor<i>- Vajubhai Vala</i>
<li>Chief Minister<i> - H. D. Kumaraswamy </i>
<li>Population<i> - 61,130,704</i>
<li>Tourist spots<i> - Gol Gumbaz, Mysore Palace, Keshava Temple etc.,</i>
</ul>
</h3>
<a href="main.html">back</a>
</body>
</html>
```

**OUTPUT**


///itdnsserver/Sandhya/AC/2019-2
file:////itdnsserver/Sandhya/AC/2019-2020/Odd Sem/Lab Manual/Workouts/m
40%
Search
Shimla
Chandigarh
Dehra Dun
New Delhi
Jaipur
Lucknow
Kolkata (Calcutta)
Kohima
Dispur
Imphal
Gandhinagar
Bhopal
Raipur
Bhubaneshwar
INDIA
Mumbai (Bombay)
Hyderabad
Panaji
Bangalore
Chennai (Madras)
file:////itdnsserver/Sandhya/AC/2019-2020/Odd Sem/Lab Manual/Workouts/tamilnadu.html
It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!
Refresh Firefox...
Type here to search
Workouts
What is i...
Lab Man...
Adobe D...
Mozilla F...
tamilna...
ENG
11:08
06-06-2019


Tanil Nadu - India
file:////itdnsserver/Sandhya/AC/2019-2020/Odd Sem/Lab Manual/Workouts/tamilnadu.h
Search
Tamil Nadu
is one of the 29 states of India. Its capital and largest city is Chennai. Tamil Nadu lies in the southernmost part of the Indian Peninsula and is bordered by the States of puducherry, Kerala, Karnataka, Andha Pradesh.
Districts - 33
Capital City - Chennai
Largest City - Chennai
Governor - Banwarilal Purohit
Chief Minister - Edappadi K. Palaniswami
Population - 72,147,030
Tourist spots - Mamallapuran, Ooty, Kodaikanal, Marina, Mudurai Meenakshi Amman Temple, Thanjavur etc.,
back
It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!
Refresh Firefox...
Type here to search
Workouts
What is i...
Lab Man...
Adobe D...
Tanil Na...
tamilna...
ENG
11:08
06-06-2019

## RESULT

Thus the Web Page had been created using image map to fix the hotspots and relevant information had been displayed.

**Ex No: 2**

# CREATION OF A WEB PAGE WITH ALL TYPES OF CASCADING STYLE SHEETS

## AIM

To create a webpage with all types of Cascading Style Sheets.

**SOFTARE USED**

**Dreamweaver or Notepad and browser.**

**DESCRIPTION**

Cascading Style Sheet (CSS) is used to set the style in web pages which contain HTML elements. It sets the background color, font-size, font-family, color etc property of elements in a web pages.
There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

## ALGORITHM

**Internal CSS:**

STEP 1: Create a HTML program with <style> tag.

STEP 2: Inside the <style> tag, specify the format required for that web page.

STEP 3: Run the program with a web browser.

**External CSS:**

STEP 4: Open a notepad, type the needed CSS in it and save it with .css extension.

STEP5: Refer this .css file in the HTML using the tag <link>.

STEP 6: Run the program with a web browser.

## CODE

**style.html**

```
<html>
<head>
<title>FLOWERS</title>
<!--Extended Style Sheet -->
<link rel="stylesheet" type="text/css" href="style.css">
<!-- Embed Style Sheet-->
<style type="text/css">
p{
background-color: lightgrey;
text-align: justify;
margin: 2em 7em;
}
</style>
```

```
</head>
<body id="body">
<h1>FLOWER</h1>
<p>
<span style="font: 200 x-large fantasy">Flower</span>
sometimes known as a bloom or blossom,
is the reproductive structure found in flowering plants.
```

```
The flower is God's finest workmanship in the world.
It is his finest gift to the mankind.
We have seen the flowers of many kinds and to many colors.
In India we see the flowers like
</p>
<!-- Inline Sytle Sheet-->
<table style="background-position: center;text-align: center;padding: 3px;">
<tr>
<td align="left">
<div class="div">
<ul>
<li><a href="">Lily</a></li>
<li><a href="">Lotus</a></li>
<li><a href="">Rose</a></li>
<li><a href="">Jasmine</a></li>
</ul>
</div>
</td>
</tr>
</table>
</body>
</html>
```

**style.css**

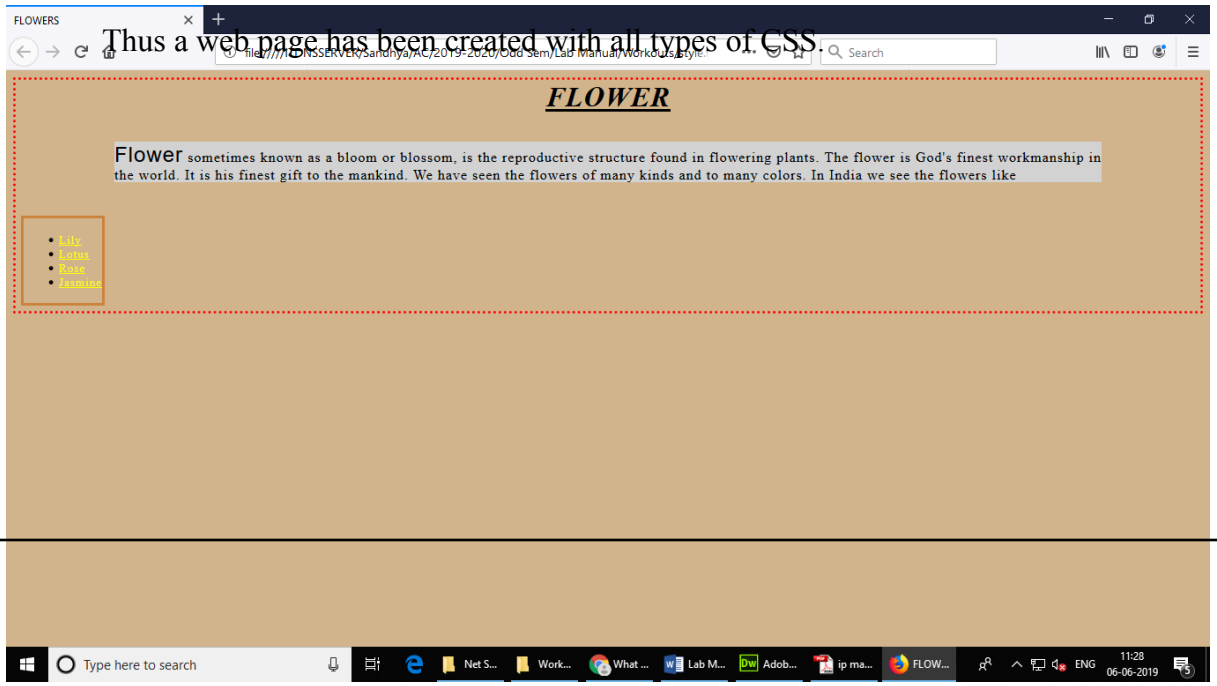
```
h1,h2{
text-decoration: underline;
font-style: italic;
text-align: center;
}
#body{
background-color: tan;
border: red dotted;
text-align: center;
}
.div{
border: peru solid ;
}
*{
letter-spacing: 1px;
}
a:link{
color: black;
}
a:visited{
color: yellow;
}
a:hover{
color: green;
}
a:active{
color: blue;
}
ul li{
```

```
font-size: small;
}
```

**OUTPUT**

# RESULT

Thus a web page has been created with all types of CSS.



**Ex No: 3**

# CLIENT SIDE SCRIPTS FOR VALIDATING WEB FORM CONTROLS USING DHTML

## AIM

To create a DHTML for validating web form controls.

## SOFTARE USED

Dreamweaver or Notepad and browser.

## DESCRIPTION

DHTML stands for Dynamic HTML, it is totally different from HTML. The browsers which support the dynamic HTML are some of the versions of Netscape Navigator and Internet Explorer of version higher than 4.0. The DHTML is based on the properties of the HTML, javascript, CSS, and DOM (Document Object Model which is used to access individual elements of a document) which helps in making dynamic content. It is the combination of HTML, CSS, JS, and DOM. The DHTML make use of Dynamic object model to make changes in settings and also in properties and methods. It also makes uses of Scripting and it is also part of earlier computing trends.

## ALGORITHM

1. The form will include one text field called "Your Name", and a submit button.

2. Validation script will ensure that the user enters their name before the form is sent to the server.

3. Open this page to see it in action.

4. Try pressing the Send Details button without filling anything in the "Your Name" field.

5. You might like to open the source code for this form in a separate window

6. The page consists of a JavaScript function called validate_form() that performs the form validation, followed by the form itself.

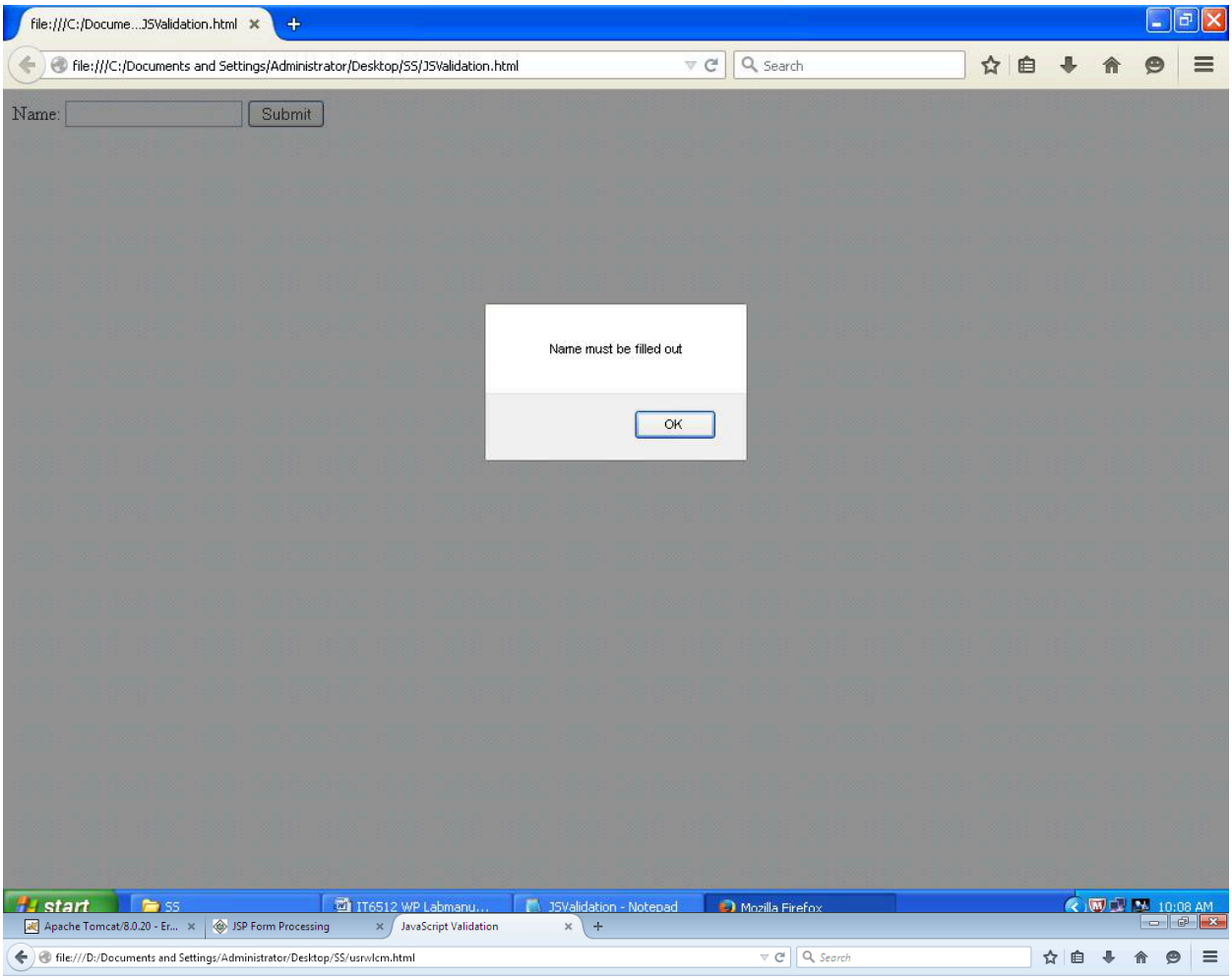
## CODE

```
<!DOCTYPE html>
<html>
<head>
<script>
functionvalidateForm() {
var x = document.forms["myForm"]["fname"].value;
        if (x == null || x == "")
        {
alert("Name must be filled out");
```

```
return false;
        }
}
</script>
</head>
<body>
<form name="myForm" action="usrwlcm.html" onsubmit="return validateForm()"
method="post">
Name: <input type="text" name="fname">
```

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

**OUTPUT**




**Welcome to the page**

## RESULT

Thus the JavaScript to validate the form has been executed.

**Ex No: 4**

# INSTALLATION OF APACHE TOMCAT WEB SERVER

**Apache Tomcat HTTP Server**

Apache Tomcat is a Java-capable HTTP server, which could execute special Java programs known as "Java Servlet" and "Java Server Pages (JSP)". Tomcat is an open-source project, under the "Apache Software Foundation" (which also provides the most use, open-source, industrial-strength Apache HTTP Server). The mother site for Tomcat is http://tomcat.apache.org. Alternatively, you can find tomcat via the Apache mother site @ http://www.apache.org.

Tomcat was originally written by James Duncan Davison (then working in Sun) in 1998, based on an earlier Sun's server called Java Web Server (JWS). It began at version 3.0 after JWS 2.1 it replaced. Sun subsequently made Tomcat open-source and gave it to Apache.

**The various Tomcat releases are:**

Tomcat 3.0 (1999): Reference Implementation (RI) for Servlet 2.2 and JSP 1.1.

Tomcat 4.1 (Sep 2002): RI for Servlet 2.3 and JSP 1.2.

Tomcat 5.0 (Dec 2003): RI for Servlet 2.4 and JSP 2.0.

Tomcat 6.0 (Feb 2007): RI for Servlet 2.5 and JSP 2.1.

Tomcat 7.0 (Jan 2011): RI for Servlet 3.0, JSP 2.2 and EL 2.2.

Tomcat 8.0 (Jun 2014): RI for Servlet 3.1, JSP 2.3, EL 3.0 and WebSocket 1.0. Tomcat 8.5 (June 2016) supports HTTP/2, OpenSSL, TLS virtual hosting and JASPIC 1.1.

Tomcat 9.0 (Jan 2018): RI for Servlet 4.0, JSP 2.3, EL 3.0, WebSocket 1.0, JASPIC 1.1.

Tomcat 10.0 (???):

**How to Install Tomcat and Get Started with Java Servlet Programming**

**2.1 STEP 0: Create a Directory to Keep all your Works**

I shall assume that you have created a directory called "c:\myWebProject" (for Windows) or "~\myWebProject" (for Mac OS X) in your earlier exercises. Do it otherwise. This step is important; otherwise, you will be out-of-sync with this article and will not be able to find your files later.

**2.2 STEP 1: Download and Install Tomcat**

**For Windows**

Goto http://tomcat.apache.org ⇒ Under "Tomcat 9.0.{xx} Released", where {xx} is the latest update number ⇒ Click "Download" ⇒ Under "9.0.{xx}" ⇒ Binary Distributions ⇒ Core ⇒ "zip" (e.g., "apache-tomcat-9.0.{xx}.zip", about 11 MB).

UNZIP the downloaded file into your project directory "c:\myWebProject". Tomcat shall be unzipped into directory "c:\myWebProject\apache-tomcat-9.0.{xx}".

For EASE OF USE, we shall shorten and rename this directory to "c:\myWebProject\tomcat".

Take note of Your Tomcat Installed Directory. Hereafter, I shall refer to the Tomcat installed directory as <TOMCAT_HOME>.

**Tomcat's Sub-Directories**

Take a quick look at the Tomcat installed directory. It contains the these sub-directories:

bin: contains the binaries and scripts (e.g., startup.bat and shutdown.bat for Windows; startup.sh and shutdown.sh for Unixes and Mac OS X).

conf: contains the system-wide configuration files, such as server.xml, web.xml, and context.xml.

webapps: contains the webapps to be deployed. You can also place the WAR (Webapp Archive) file for deployment here.

lib: contains the Tomcat's system-wide library JAR files, accessible by all webapps. You could also place external JAR file (such as MySQL JDBC Driver) here.

logs: contains Tomcat's log files. You may need to check for error messages here.

work: Tomcat's working directory used by JSP, for JSP-to-Servlet conversion.

**STEP 2: Create an Environment Variable JAVA_HOME**

**(For Windows)**

You need to create an environment variable (system variable available to all applications) called "JAVA_HOME", and set it to your JDK installed directory.

Many Java applications (such as Tomcat) require the environment variable JAVA_HOME to be set to the JDK installed directory.

To set the JAVA_HOME environment variable:

First, find your JDK installed directory. For JDK 11, the default is "c:\Program Files\Java\jdk-11.0.{x}", where "{x} is the update number. Use your "File Explorer" to find this directory and take note of your update number {x}.

Check if JAVA_HOME is already set. Start a CMD and issue:

set JAVA_HOME

If you get a message "Environment variable JAVA_HOME not defined", proceed to the next step.

If you get "JAVA_HOME=C:\Program Files\Java\jdk-11.0.{x}", verify that it is set correctly to your JDK directory. If not, proceed to the next step.

To set the environment variable JAVA_HOME in Windows 10:

Launch "Control Panel" ⇒ (Optional) "System and Security" ⇒ "System" ⇒ Click "Advanced system settings" on the left pane.

Switch to "Advanced" tab ⇒ Click "Environment Variables"

Under "System Variables" (the bottom pane) ⇒ Click "New" (or Look for "JAVA_HOME" and "Edit" if it is already set) ⇒ In "Variable Name", enter "JAVA_HOME" ⇒ In "Variable Value", enter your JDK installed directory you noted in Step 1. (In the latest Windows 10: you can push the "Browse Directory" button and navigate to the JDK installed directory to avoid typo error.)

To verify, RE-START a CMD (restart is needed to refresh the environment variables) and issue:

set JAVA_HOME

JAVA_HOME=c:\Program Files\Java\jdk-11.0.{x}  <== Verify that this is YOUR JDK installed directory

Notes: Windows' environment variables (such as JAVA_HOME, PATH) are NOT case-sensitive.

**STEP 3: Configure the Tomcat Server**

The Tomcat configuration files, in XML format, are located in the "conf" sub-directory of your Tomcat installed directory, e.g. "c:\myWebProject\tomcat\conf" (for Windows) or "~/myWebProject/tomcat/conf" (for Mac OS X). The important configuration files are:

server.xml

web.xml

context.xml

Make a BACKUP of the configuration files before you proceed!!!

**Step 3(a) "conf\server.xml" - Set the TCP Port Number**

Use a programming text editor (e.g., Sublime Text, Atom) to open the configuration file "server.xml".

The default TCP port number configured in Tomcat is 8080, you may choose any number between 1024 and 65535, which is not used by existing applications. We shall choose 9999 in this article. (For production server, you should use port 80, which is pre-assigned to HTTP server as the default port number.)

Locate the following lines (around Line 69) that define the HTTP connector, and change port="8080" to port="9999".

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="9999" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

**Step 3(b) "conf\web.xml" - Enable Directory Listing**

Again, use a programming text editor to open the configuration file "web.xml".

We shall enable directory listing by changing "listings" from "false" to "true" for the "default" servlet. This is handy for test system, but not for production system for security.

Locate the following lines (around Line 108) that define the "default" servlet; and change the "listings" 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>true</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
```

**Step 3(c) "conf\context.xml" - Enabling Automatic Reload**

We shall add the attribute reloadable="true" to the <Context> element to enable automatic reload after code changes. Again, this is handy for test system but not recommended for production, due to the overhead of detecting changes.

Locate the <Context> start element (around Line 19), and change it to <Context reloadable="true">.

```
<Context reloadable="true">
   ......
   ......
</Context>
```

**2.5 STEP 4: Start Tomcat Server**

The Tomcat's executable programs and scripts are kept in the "bin" sub-directory of the Tomcat installed directory.

**Step 4(a) Start Server**

For Windows

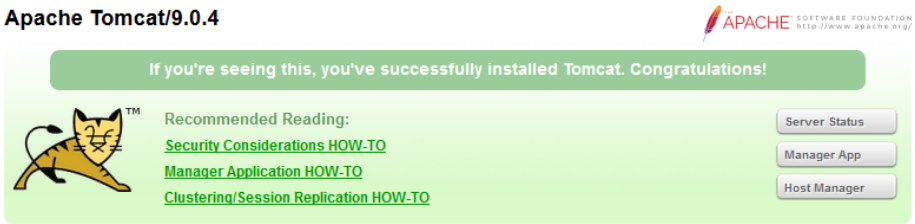
I shall assume that Tomcat is installed in "c:\myWebProject\tomcat". Launch a CMD shell and issue:

```
c:                          // Change drive
cd \myWebProject\tomcat\bin  // Change directory to your Tomcat's binary directory
startup                     // Run startup.bat to start tomcat server
```

**Step 4(b) Start a Client to Access the Server**

Start a browser (Firefox, Chrome) as an HTTP client. Issue URL "http://localhost:9999" to access the Tomcat server's welcome page. The hostname "localhost" (with IP address of 127.0.0.1) is meant for local loop-back testing within the same machine. For users on the other machines over the net, they have to use the server's IP address or DNS domain name in

the form of "http://*serverHostnameOrIPAddress*:9999".
(Optional) Try issuing URL http://localhost:9999/examples to view the servlet and JSP examples. Try running some of the servlet examples.




**Step 4(c) Shutdown Server**

For Windows

You can shutdown the tomcat server by either:

Press Ctrl-C on the Tomcat console; OR

Run "<TOMCAT_HOME>\bin**shutdown.bat**" script. Open a new "cmd" and issue:

**c:**// Change the current drive

**cd \myWebProject\tomcat\bin**// Change directory to your Tomcat's binary directory

**shutdown**// Run shutdown.bat to shutdown the server

**Ex No: 5**

# INVOKING SERVLETS FROM HTML FORMS-SESSION TRACKING

## AIM

To create a simple application to perform session tracking using servlet.

## SOFTWARE REQUIRED

Tomcat Server

## DESCRIPTION

Session simply means a particular interval of time. Session Tracking is a way to maintain state (data) of an user. It is also known as session management in servlet.

### Session Tracking Techniques

There are four techniques used in Session tracking:

- **Cookies**
- **Hidden Form Field**
- **URL Rewriting**
- **HttpSession**

## ALGORITHM

**client.html**

1. Create a web page using HTML form that contains the fields such as text, password and one submit button.
2. Set the URL of the server as the value of form's action attribute.
3. Run the HTML program.
4. Submit the form data to the server.

**server.java**

1. Define the class server that extends the property of the class GenericServlet.
2. Handle the request from the client by using the method service() of GenericServlet class.
3. Get the parameter names from the HTML form by using the method getParameterNames().
4. Get the parameter values from the HTML forms by using the method getParameter().
5. Send the response to the client by using the method of PrintWriter class.

## CODE

**index.html**

```
<!DOCTYPE html>
  <html>
  <head>
  <meta charset="ISO-8859-1">
  <title>Servlet Login Example</title>
  </head>
  <body>
  <h1>Welcome to Login App by Cookie</h1>
```

```
  <a href="login.html">Login</a>|
  <a href="LogoutServlet">Logout</a>|
  <a href="ProfileServlet">Profile</a>
  </body>
  </html>
```

**link.html**

```
  <a href="login.html">Login</a> |
  <a href="LogoutServlet">Logout</a> |
  <a href="ProfileServlet">Profile</a>
  <hr>
```

**login.html**

```
<form action="LoginServlet" method="post">
Name:<input type="text" name="name"><br>
Password:<input type="password" name="password"><br>
<input type="submit" value="login">
</form>
```

**LoginServlet.java**

```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        request.getRequestDispatcher("link.html").include(request, response);

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

        if(password.equals("admin123")){
            out.print("You are successfully logged in!");
            out.print("<br>Welcome, "+name);

            Cookie ck=new Cookie("name",name);
            response.addCookie(ck);
        }else{
            out.print("sorry, username or password error!");
            request.getRequestDispatcher("login.html").include(request, response);
```

```
            }

            out.close();
        }

    }
```

**LogoutServlet.java**

```
package com.javatpoint;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class LogoutServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
          request.getRequestDispatcher("link.html").include(request, response);
        Cookie ck=new Cookie("name","");
        ck.setMaxAge(0);
        response.addCookie(ck);
         out.print("you are successfully logged out!");
    }
}
```

**File: ProfileServlet.java**

```
package com.javatpoint;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ProfileServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        request.getRequestDispatcher("link.html").include(request, response);

        Cookie ck[]=request.getCookies();
        if(ck!=null){
         String name=ck[0].getValue();
        if(!name.equals("")||name!=null){
            out.print("<b>Welcome to Profile</b>");
            out.print("<br>Welcome, "+name);
```

```
        }
        }else{
            out.print("Please login first");
            request.getRequestDispatcher("login.html").include(request, response);
        }
        out.close();
    }
```

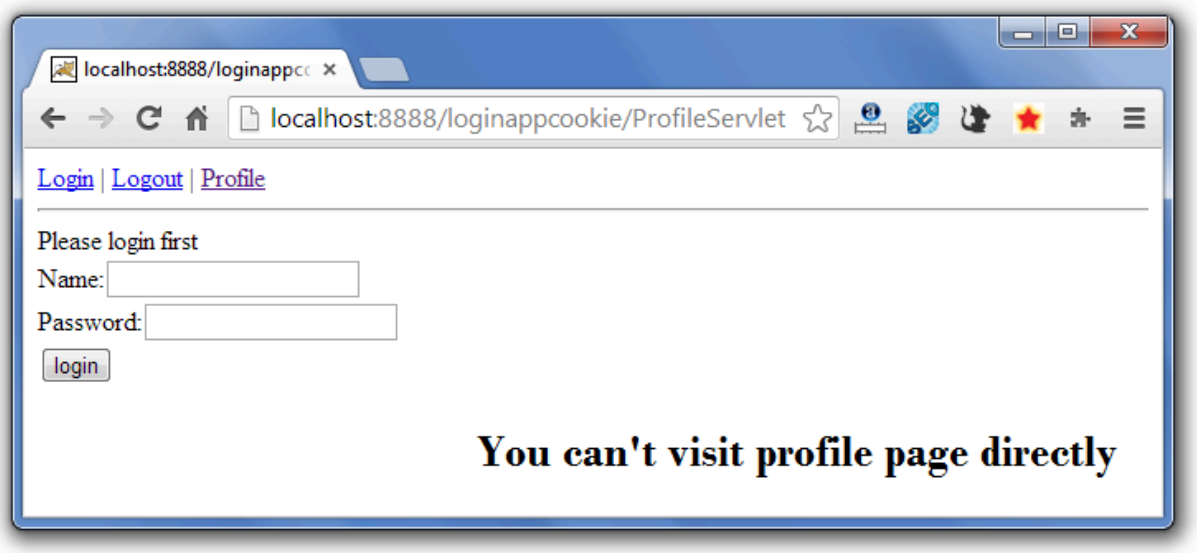
}

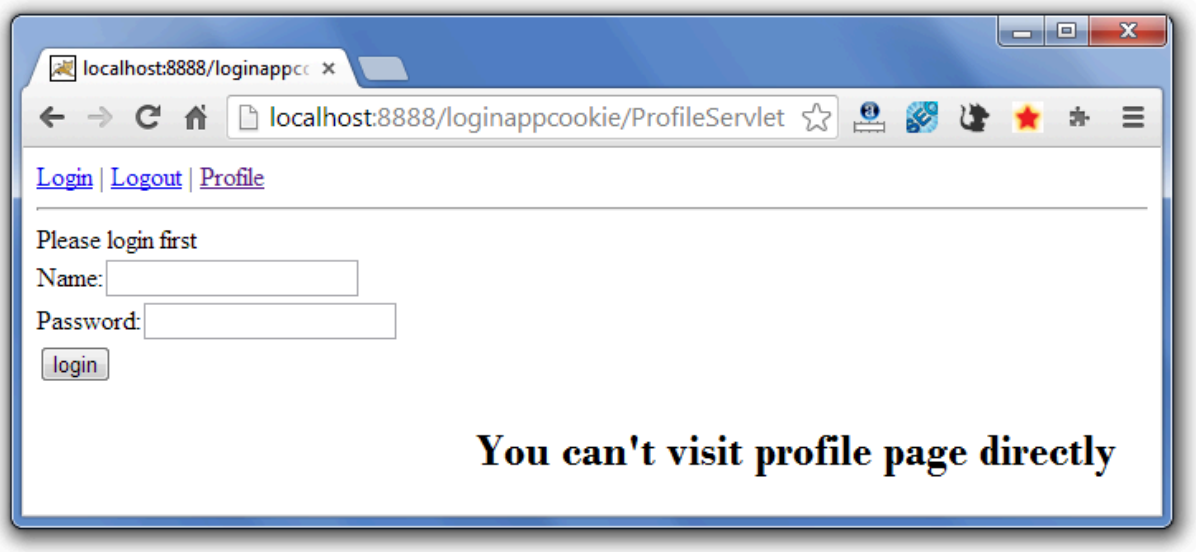
**web.xml**

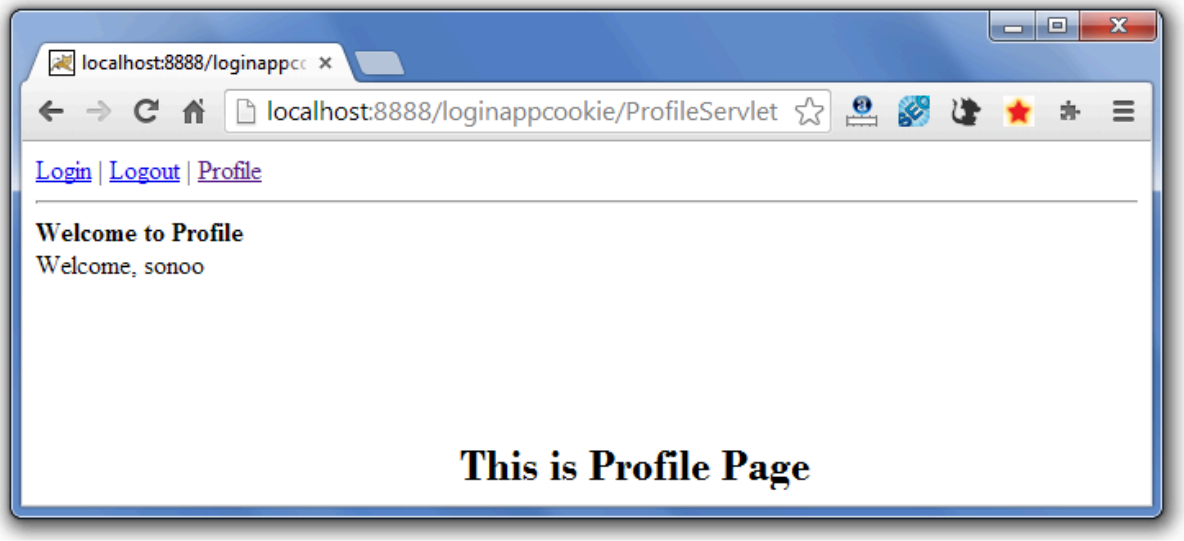
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml /ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

  <servlet>
    <description></description>
    <display-name>LoginServlet</display-name>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>com.javatpoint.LoginServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/LoginServlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <description></description>
    <display-name>ProfileServlet</display-name>
    <servlet-name>ProfileServlet</servlet-name>
    <servlet-class>com.javatpoint.ProfileServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProfileServlet</servlet-name>
    <url-pattern>/ProfileServlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <description></description>
    <display-name>LogoutServlet</display-name>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>com.javatpoint.LogoutServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/LogoutServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

**OUTPUT**

Servlet Login Example
localhost:8888/loginappcookie/
Welcome to Login App by Cookie
Login| Logout| Profile


localhost:8888/loginappc
localhost:8888/loginappcookie/ProfileServlet
Login | Logout | Profile
Please login first
Name:
Password:
login
You can't visit profile page directly


localhost:8888/loginappc
localhost:8888/loginappcookie/ProfileServlet
Login | Logout | Profile
Please login first
Name:
Password:
login
You can't visit profile page directly


localhost:8888/loginappc
localhost:8888/loginappcookie/ProfileServlet
Login | Logout | Profile
Welcome to Profile
Welcome, sonoo
This is Profile Page

## RESULT

Thus a simple application has been created for invoking servlets from HTML forms.

**Ex No: 6a**

# CREATION OF THREE-TIER APPLICATIONS USING JSP AND DATABASES - For conducting on-line examination

## AIM

To create a three-tier application using JSP and Databases for the conduction of online-examination.

## SOFTWARE REQUIRED

Tomcat Server

## DESCRIPTION

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

## ALGORITHM

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.
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.

## CODE

**ExamServer.jsp:**

```
<%@page contentType="text/html" language="java" import="java.sql.*"%>
<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="ExamClient.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;
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:examDS");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("SELECT * FROM examTab");
int i=1;
while(rs.next())
{
 if(i==1)
 {
  String dbans1=rs.getString(1);
  if(str1.equals(dbans1))
  {
   mark=mark+5;
  }
 }
 if(i==2)
 {
  String dbans2=rs.getString(1);
  if(str2.equals(dbans2))
  {
   mark=mark+5;
  }
 }
 if(i==3)
 {
  String dbans3=rs.getString(1);
  if(str3.equals(dbans3))
  {
   mark=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>");
}
%>
</form>
```

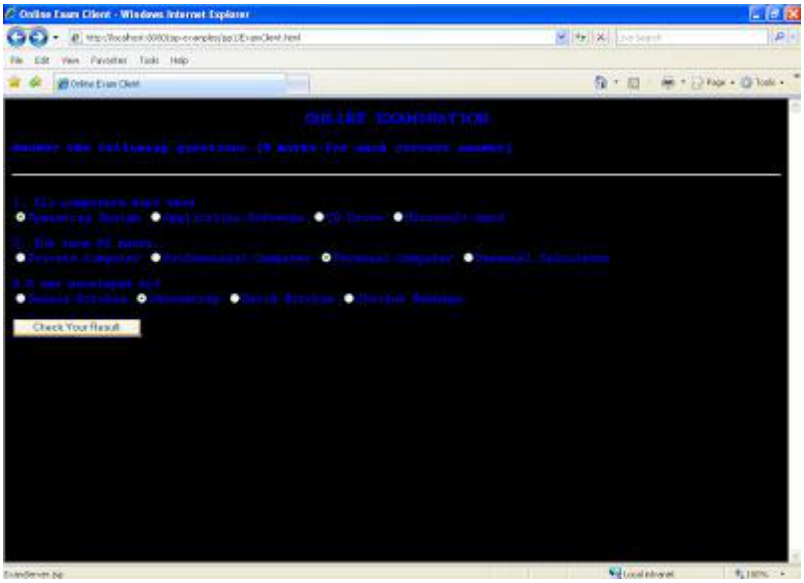
```
</body>
</html>
```

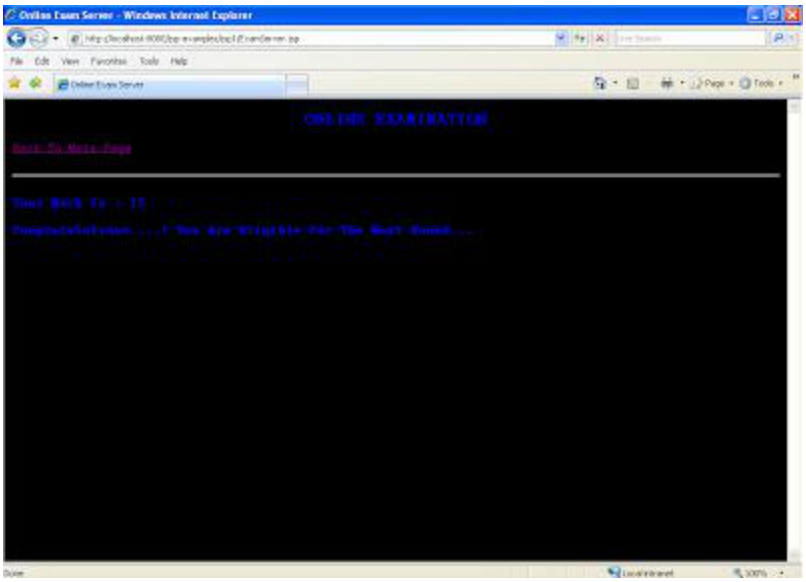
**ExamClient.HTML:**

```
<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="ExamServer.jsp">
1. All computers must have <br/>
<input type="radio" name="ans1" value="Operating System">Operating System
<input type="radio" name="ans1" value="Application Software">Application Software
<input type="radio" name="ans1" value="CD Drive">CD Drive
<input type="radio" name="ans1" value="Microsoft word">Microsoft word
<br/><br/>
2. The term PC means <br/>
<input type="radio" name="ans2" value="Private Computer">Private Computer
<input type="radio" name="ans2" value="Professional Computer">Professional Computer
<input type="radio" name="ans2" value="Personal Computer">Personal Computer
<input type="radio" name="ans2" value="Personal Calculator">Personal Calculator
<br/><br/>
3.C was developed by?<br/>
<input type="radio" 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>
```

## OUTPUT


Online Exam Client - Windows Internet Explorer
File Edit View Favorites Tools Help
Online Exam Client
ONLINE EXAMINATION
Check Your Result


Online Exam Server - Windows Internet Explorer
File Edit View Favorites Tools Help
Online Exam Server
ONLINE EXAMINATION
Back To Main Page
Time Left In : 15
Congratulations....! You are Eligible For The Next Round....
Done
Local intranet
100%

## RESULT

Thus a simple web application using JSP and Database had been created for the conduction of online examination.

**Ex No: 6b**

# CREATION OF THREE-TIER APPLICATIONS USING JSP AND DATABASES - For displaying student mark list

## AIM

To create a three-tier application using JSP and Databases for displaying the student mark list.

## SOFTWARE REQUIRED

Tomcat Server

## DESCRIPTION

Java Server Pages (JSP) is a technology that helps software developers create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language.

## ALGORITHM

1. Design the HTML page (stud.html) with the following
   a. Create a form to get the input (Register Number) from the user.
   b. Set the URL of the server (marklist.jsp) as the value of the action attribute.
   c. Use submit button to invoke the server and send the form data to the server.
2. Create the JSP file with the following
   a. Read the parameter value (Register Number) from the form by using the method getParameter().
   b. Server retrieves the details from the database table with respect to the form input.
   c. Server displays the mark list to the client as the response.

## CODE

**marklist.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/>
<%
```

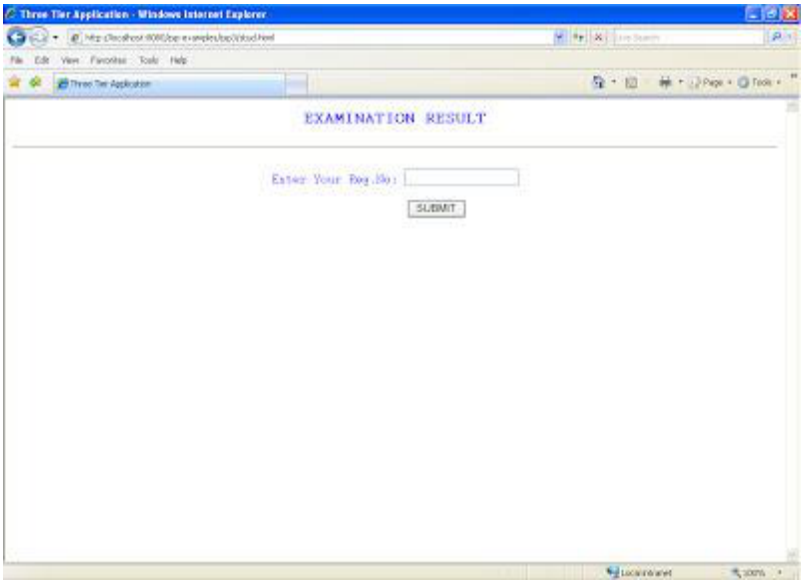
```
String str=request.getParameter("regno");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con=DriverManager.getConnection("jdbc:odbc:markDS");
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("SELECT*FROM markTab WHERE rno="+str);
while(rs.next())
{
%>
Register No:<%=rs.getObject(1)%><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>
<%
}
%>
<br/>
<a href="stud.html">Back</a>
</body>
</html>
```

**stud.html**

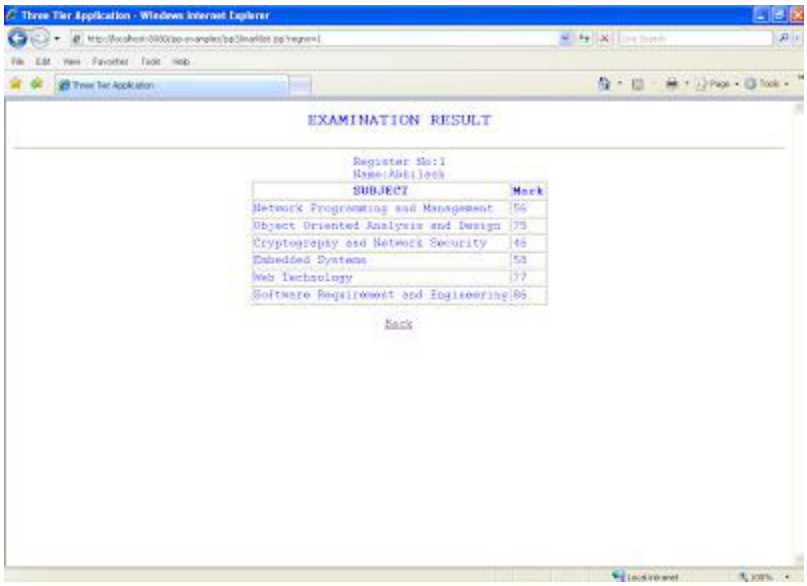
```
<html>
<head>
<title>Three Tier Application</title>
<style type="text/css">
  body{color:blue;font-family:courier;text-align:center}
</style>
</head>
<body>
<h2>EXAMINATION RESULT</h2><hr/>
<form name="f1" method"GET" action="marklist.jsp">
Enter Your Reg.No:
<input type="text" name="regno"/><br/><br/>
      
    
<input type="submit" value="SUBMIT"/>
</form>
</body>
<html>
```

## OUTPUT




## RESULT

Thus a simple web application using JSP and Database had been created displaying student mark list.



**Ex No: 7**

# PROGRAMS USING XML – SCHEMA – XSLT/XSL

## AIM

To write a XML scheme to generate CD Collection details.

## SOFTWARE REQUIRED

Dreamweaver or Notepad and Browser.

## DESCRIPTION

In HTML documents, tags are predefined but in XML documents, tags are not predefined. World Wide Web Consortium (W3C) developed XSL to understand and style an XML document, which can act as XML based Stylesheet Language.

An XSL document specifies how a browser should render an XML document.

**Main parts of XSL Document**

- XSLT: It is a language for transforming XML documents into various other types of documents.
- XPath: It is a language for navigating in XML documents.
- XQuery: It is a language for querying XML documents.
- XSL-FO: It is a language for formatting XML documents.

**How XSLT Works?**

The XSLT stylesheet is written in XML format. It is used to define the transformation rules to be applied on the target XML document. The XSLT processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. At the end it is used by XSLT formatter to generate the actual output and displayed on the end-user.

## ALGORITHM

Step 1: Start the program

Step 2: Use Xml Style Sheet code to define link

```
<?xml-stylesheet type="text/xsl" href="yourxsl.xsl"?>
```

Step 3: Use the catalog tag to define CD collection details.

Step 4: Use the necessary heading for appropriate XML tag.

Step 5: Provide necessary information for CD collection details

Step 6: Stop the program

## CODE

## CDCatalog.xml

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Edited by XMLSpy® -->
<?xml-stylesheet type="text/xsl" href=" CDCatalog.xsl"?>
<catalog>
<cd>
<title>Empire Burlesque</title>
<artist>Bob Dylan</artist>
<country>USA</country>
<company>Columbia</company>
<price>10.90</price>
```

```
<year>1985</year>
</cd>
<cd>
<title>Hide your heart</title>
<artist>Bonnie Tyler</artist>
<country>UK</country>
<company>CBS Records</company>
<price>9.90</price>
<year>1988</year>
</cd>
<cd>
<title>Greatest Hits</title>
<artist>Dolly Parton</artist>
<country>USA</country>
<company>RCA</company>
<price>9.90</price>
<year>1982</year>
</cd>
</catalog>
```

**CDCatalog.xsl**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>My CD Collection</h2>
<table border="1">
<tr bgcolor="#9acd32">
<th>Title</th>
<th>Artist</th>
</tr>
<xsl:for-each select="catalog/cd">
<xsl:sort select="artist"/>
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
```

```
</xsl:stylesheet>
```

**RESULT**

Thus an XML application had been created to display CD Catalog using XSL

**Ex No: 8**

# PROGRAMS USING DOM AND SAX PARSERS

## AIM

To write a java code using DOM and SAX API for parsing an XML document and check whether it is well formatted or not.

## SOFTWARE REQUIRED

Java

## DESCRIPTION

1) DOM parser loads whole XML documents in memory while SAX only loads a small part of the XML file in memory.

2) DOM parser is faster than SAX because it accesses the whole XML document in memory.

3) SAX parser in Java is better suitable for a large XML file than DOM Parser because it doesn't require much memory.

4) DOM parser works on Document Object Model while SAX is an event based XML parser.

## ALGORITHM

Step 1: Start the program.
Step 2: Create an XML document containing student's information.
Step 3: Write a java code that import the various DOM and SAX related classes to parse the XML document.
Step 4: Compile and run the java file which parses the document and displays whether the created XML document is correctly formatted.
Step 5: Stop the program.

## CODE

**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
```

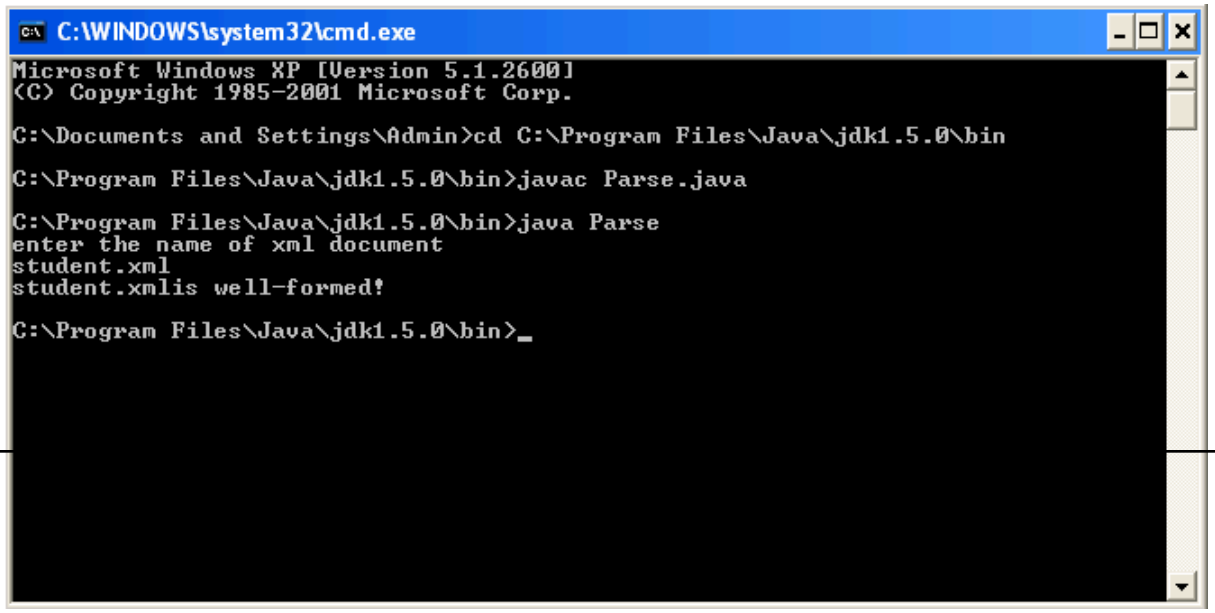
```
{
System.out.print("file not found!");
}
}
catch(IOException ex)
{
ex.printStackTrace();
```

```
}
}
}
```

**OUTPUT**

## RESULT

Thus the program to parse XML document had been created using DOM and SAX API.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Admin>cd C:\Program Files\Java\jdk1.5.0\bin

C:\Program Files\Java\jdk1.5.0\bin>javac Parse.java

C:\Program Files\Java\jdk1.5.0\bin>java Parse
enter the name of xml document
student.xml
student.xmlis well-formed!

C:\Program Files\Java\jdk1.5.0\bin>_
```

**Ex No: 9**

# PROGRAMS USING AJAX

## AIM

To write a java script program for a AJAX.

## SOFTWARE REQUIRED

Dreamweaver or Notepad and Browser.

## DESCRIPTION

AJAX stands for Asynchronous JavaScript and XML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

## ALGORITHM

Step 1 Start the program.

Step 2 A scripting language that is commonly hosted in a browser to add Interactivity to HTML PAGES.

Step 3 Defines the structure of a webpage as a set of programmable objects that can be accessed through JavaScript.

Step 4 Allows a client-side script to perform and httprequest.

Step 5 AJAX applications use xmlhttprequest object to perform asynchronous requests to the server as opposed to performing a full page refresh.

Step 6 Display the result.

Step 7 Stop the program.

**CODE**

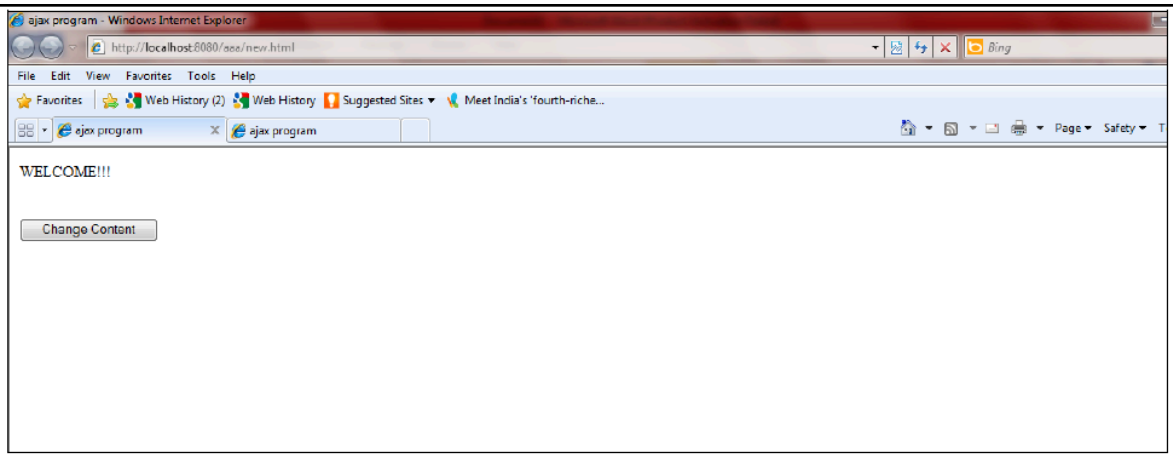
```
<html>
<head>
<script type="text/javascript">
functionloadXMLDoc()
{
if (window.XMLHttpRequest)
xmlhttp=new XMLHttpRequest();
```

```
else
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 &&xmlhttp.status==200)
{
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","new.txt",true);
xmlhttp.send();
}
</script>
<title>ajax program</title>
</head>
<body>
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>
</body>
</html>
```

**New.txt:**

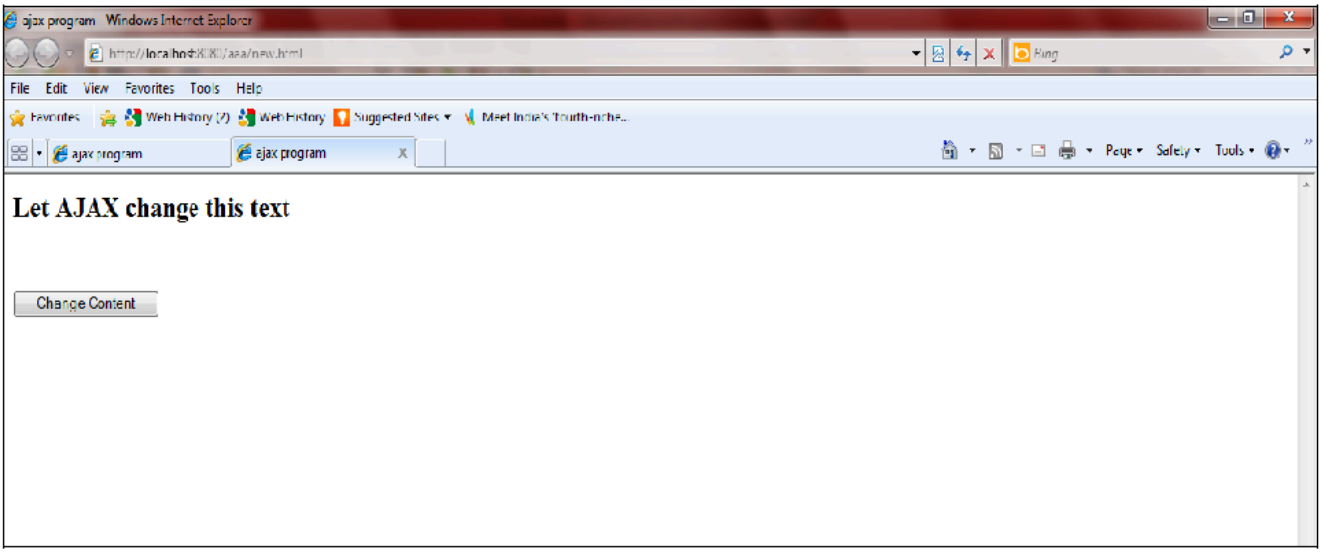
WELCOME!!!

## OUTPUT


ajax program - Windows Internet Explorer
http://localhost:8080/aaa/new.html
File Edit View Favorites Tools Help
Favorites
Web History (2)
Web History
Suggested Sites
Meet India's 'fourth-riche...
ajax program
ajax program
Page
Safety
WELCOME!!!
Change Content

**RESULT**

Thus the program using AJAX had been created.



**Ex No: 10**

# PROGRAM USING WEB SERVICE AND DATABASE

## AIM

To create a web service for travel service using NetBeans.

## SOFTWARE REQUIRED

NetBeans

## DESCRIPTION

A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response.

Web services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program. Web services allow various applications to talk to each other and share data and services among themselves.

## ALGORITHM

1. Using the NetBeans API create a project of the type web application.
2. Establish connectivity with the database along with the necessary authentication parameters.
3. Create a web service in the project.
4. Click on the Design tab and design the prototype of the web service.
5. Click on source tab and modify the application logic of the web service.
6. Save the project.
7. Right click on the project and click on deploy and undeploy.
8. Then test the web service.
9. Create another web application project and create a jsp file.
10. Right click on project and click on create web service client.
11. Browse and choose the web service created i.e wsdl url
12. Drag and drop the web service reference to the source code window.
13. Then pass the appropriate parameters to the web service client and invoke the web service.

## STEPS FOR CREATING TRAVEL WEB SERVICE

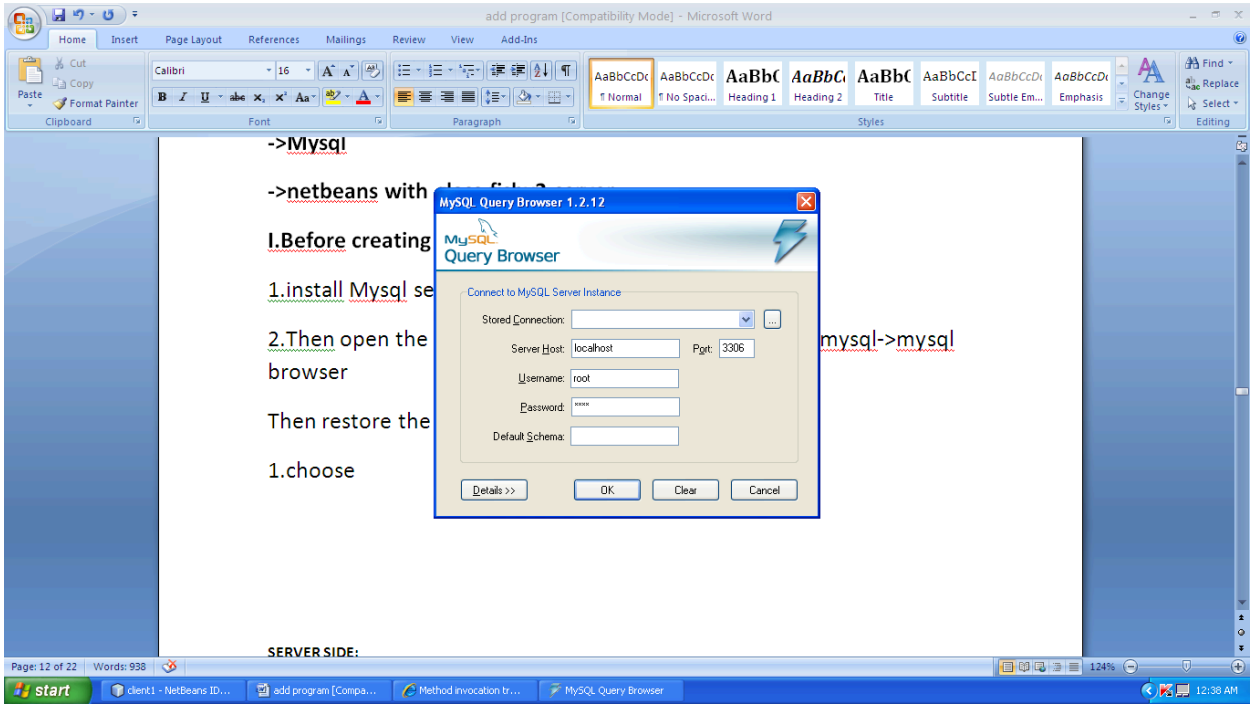
Requirements:

->Mysql

->netbeans with glass fishv2 server

I. Before creating project in net beans do the following:

1. install Mysql setup file and the gui tools .
2. Then open the mysql browser in the start->programs->mysql->mysql browser




Then restore the back up (newproject2011…) as follows

1. choose tools->mysql administrator
2. Then choose restore in that window..Then click open backup in the bottom…open the backup file(newproject201….) which is in the attachment….then click start restore..

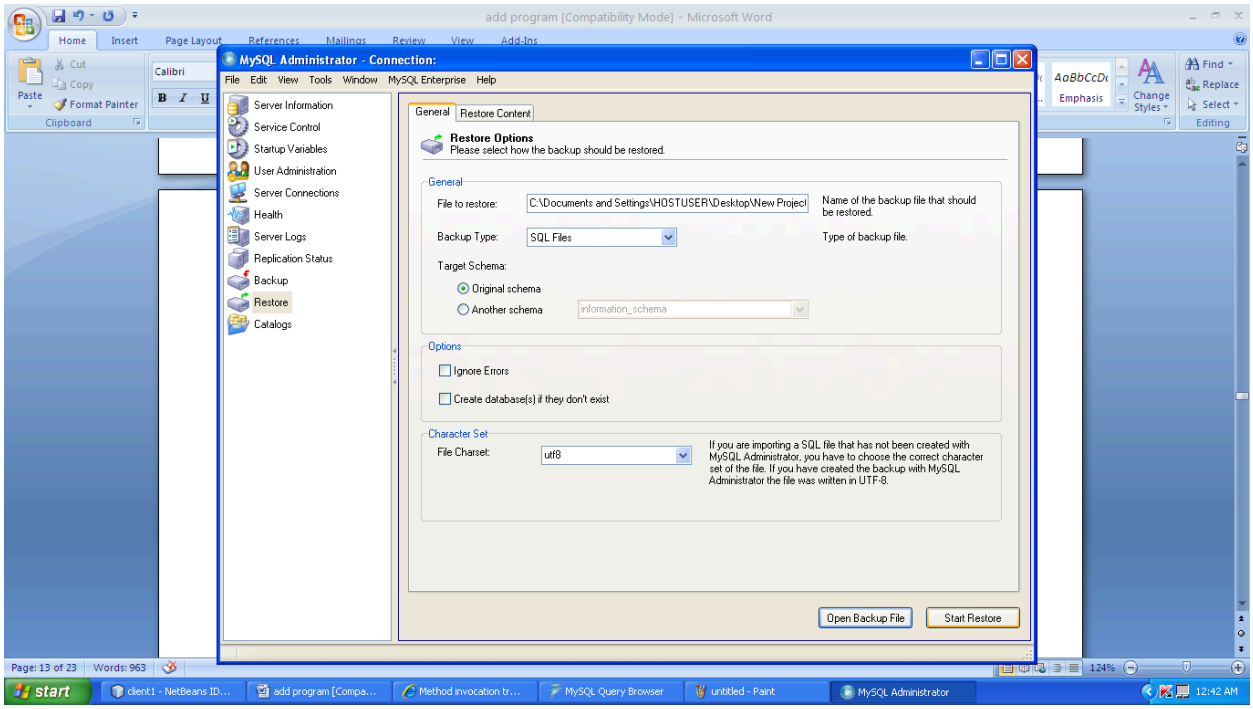
3. now the schema is created in the mysql…..

Now strat the coding in netbeans…

## SERVER SIDE:

1.create the server project as before and with the following changes.

Project name: KPNBusTravels



2.web service name:KPNBusTravels
Package name:kpnpack
3.in design window
Operation name:getBus
Parameters:
Source and destination with the type as String.
4.Add the following code in source

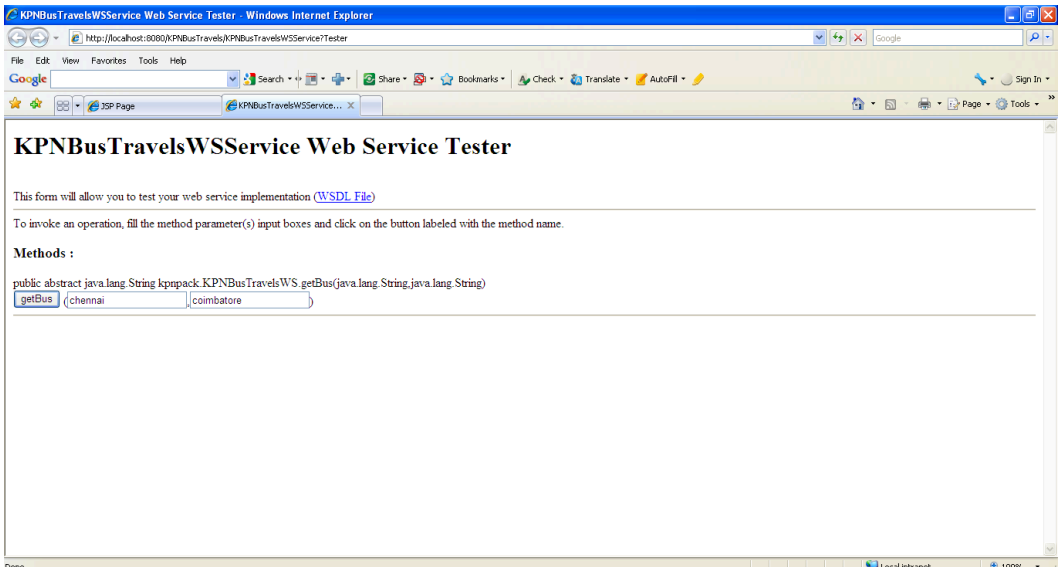
```
package kpnpack;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Hashtable;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
/**
 *
 * @author user
 */
@WebService()
public class KPNBusTravelsWS {
    /**
     * Web service operation
     */
    @WebMethod(operationName = "getBus")
    public String getBus(@WebParam(name = "source")
    String source, @WebParam(name = "destination")
    String destination) {
        //TODO write your implementation code here:
        ArrayList<String> service_name = new ArrayList<String>();
        ArrayList<String> departure_time = new ArrayList<String>();
        ArrayList<String> arrival_time = new ArrayList<String>();
       ArrayList<String> bus_fare = new ArrayList<String>();
      ArrayList<String> bus_type = new ArrayList<String>();
      ArrayList<String> bus_category=new ArrayList<String>();
       try {
       Class.forName("com.mysql.jdbc.Driver");
       Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/qos","root","root");
            Statement st1=con1.createStatement();
              st1.executeUpdate("insert into qostable(webname,
exetime,price,compute,availtime) values('"+wn+"','"+et+"','"+pr+"','"+ct+"','"+at+"')");
       Statement st=con.createStatement();
```

```
ResultSet rs = st.executeQuery(sql);
  int i=0;
while (rs.next()) {
  departure_time.add(rs.getString(3));
  arrival_time.add(rs.getString(4));
  bus_fare.add(rs.getString(5));
   bus_type.add(rs.getString(9));
```

```
            bus_category.add(rs.getString(11));
            service_name.add(rs.getString(12));
          //  service_name.add(rs.getString("Servicename"));
          }
        } catch ( Exception e) {
          e.printStackTrace();
        }
        String concatBus =
service_name.toString()+"&"+departure_time.toString()+"&"+arrival_time.toString()
+"&"+bus_fare.toString() +"&"+
            bus_type.toString() +"&"+bus_category.toString() +",";
        return concatBus;
    }
}
```

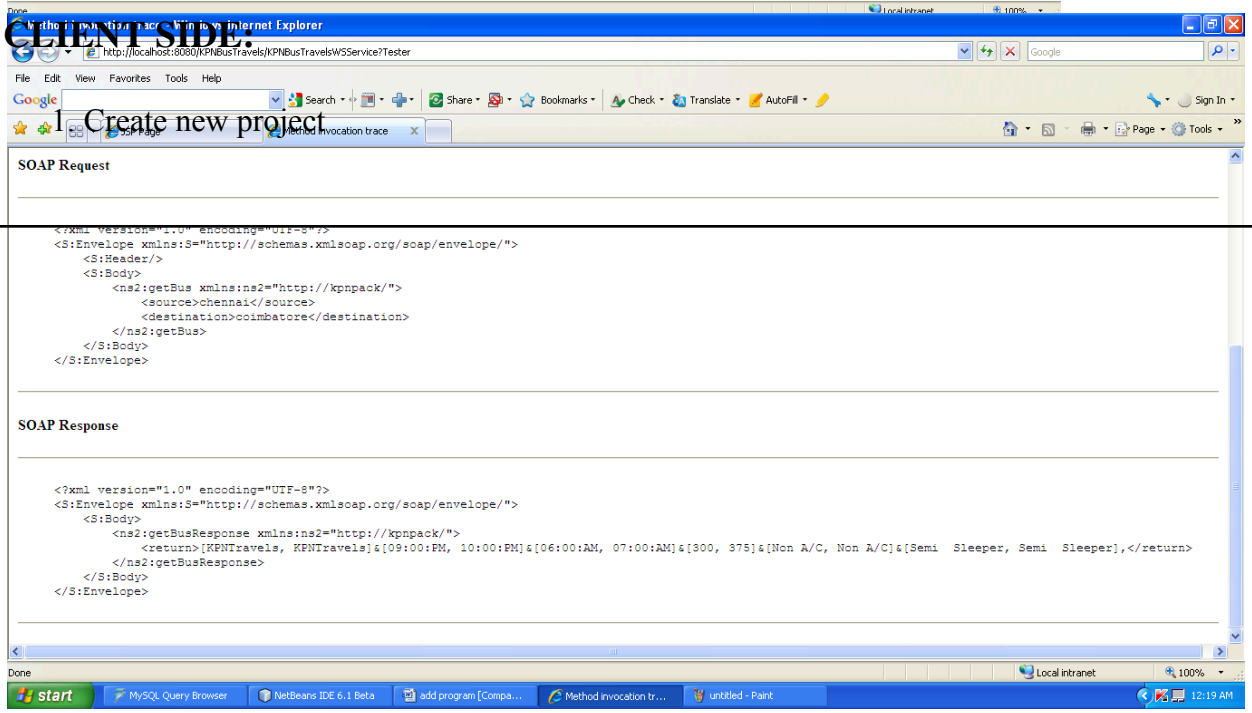
5.Then deploy the service and test the web service as before.

Give input as Chennai and Coimbatore .then click getbus.




# CLIENT SIDE:

1. Create new project



Name:client1

2. As before in addition program create web service reference.
3. The source code for index.jsp and action,jsp are as follow.

## 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>JSP Page</title>
   </head>
   <body>
      <h2>Hello World!</h2>
      <form name="" action="actionn.jsp" method="post">
 Enter Source:<input  name="source" type="text" /><br/>
Enter Destination:<input  name="destination" type="text" /><br/>
 <input  name="ok" type="submit" value="" />
</form>
   </body>
</html>
```

## Action.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>
      <h2>Hello World!</h2>
       <%
 String a1=request.getParameter("source");
  String b1=request.getParameter("destination");
  %>
   <%-- start web service invocation --%><hr/>
   <%-- end web service invocation --%><hr/>
   <%-- start web service invocation --%><hr/>
   <%
      String[] res,commasplit=null,lbrsplit=null,rbrsplit=null;
     String[][] flight_trans=new String[100][100];
      int row=0,col=0;
      int trow=0,tcol=0;
      int midrow=0,midcol=0;
```

```
 // Jagson
String[] jagres,jagcommasplit=null,jaglbrsplit=null,jagrbrsplit=null;
String[][] jagflight_trans=new String[100][100];
 int jagrow=0,jagcol=0;
 //Air India
 String[] airres,aircommasplit=null;
String[][] airflight_trans=new String[100][100];
```

```
 int airrow=0,aircol=0;
 try {
     kpnpack.KPNBusTravelsWSService service = new
kpnpack.KPNBusTravelsWSService();
     kpnpack.KPNBusTravelsWS port = service.getKPNBusTravelsWSPort();
      // TODO initialize WS operation arguments here
     java.lang.String source = "a1";
     java.lang.String destination = "b1";
     // TODO process result here
     java.lang.String result = port.getBus(source, destination);
     out.println("Result = "+result);
     res=result.split("&");
 System.out.println("Jet Split:"+res);
     for(int i=0;i<res.length;i++)
     {
        int j=0;
     row=res.length;
     String lbr=res[i].replace("[", "");
     String rbr=lbr.replace("]", "");
     commasplit=rbr.split(",");
     for( j=0;j<commasplit.length;j++)
         {
         col=commasplit.length;
         System.out.println("flight_trans[i][j]"+i+"/"+j);
          if(commasplit[j]!=null)
         flight_trans[j][i]=commasplit[j];
          }
    // if( flight_trans[i][j]==null)
        // col=0;
     }
     // out.println("Res = "+res[0]);
    System.out.println("TR");
    System.out.println("jetRow:"+row);
     System.out.println("jetColumn:"+col);
   for(int r=0;r<row;r++)
       {
       for(int c=0;c<col;c++)
           {
           if(flight_trans[r][c]!=null)
           System.out.print("tr:"+flight_trans[r][c]);
       }
       System.out.println();
   }

} catch (Exception ex) {
```

```
            // TODO handle custom exceptions here
        }
        %>
        <%-- end web service invocation --%><hr/>
        </body>
    </html>
```

4.Then finally undeploy and deploy the client and run it……..

**RESULT**

Thus the web service for travel services had been created.

**Ex No: 11**

# DEVELOPMENT OF WEB APPLICATION OF ANGULARJS

## AIM

To develop a web application using AngularJS framework.

## SOFTWARE REQUIRED

AngularJS Library, Editor/IDE, Browser and Web server

## DESCRIPTION

AngularJS is a client side JavaScript MVC framework to develop a dynamic web application. AngularJS was originally started as a project in Google but now, it is open source framework.

**AngularJS Extends HTML**

- AngularJS extends HTML with ng-directives.
- The ng-app directive defines an AngularJS application.
- The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
- The ng-bind directive binds application data to the HTML view.

## SETUP ANGULARJS DEVELOPMENT ENVIRONMENT

We need the following tools to setup a development environment for AngularJS:

1. AngularJS Library
2. Editor/IDE
3. Browser
4. Web server

**AngularJS Library**

To download AngularJS library, go to angularjs.org -> click download button, which will open the following popup.

Download AngularJS Library

Select the required version from the popup and click on download button in the popup.

CDN: You can include AngularJS library from CDN url –
https://ajax.googleapis.com/ajax/libs/angularjs/1.3.16/angular.min.js

**Editor**
AngularJS is eventually HTML and JavaScript code. So you can install any good editor/IDE as per your choice.

The following editors are recommended:

- Sublime Text
- Aptana Studio 3
- Ultra Edit
- Eclipse
  Visual Studio

**Online Editor**
You can also use the following online editors for learning purpose.

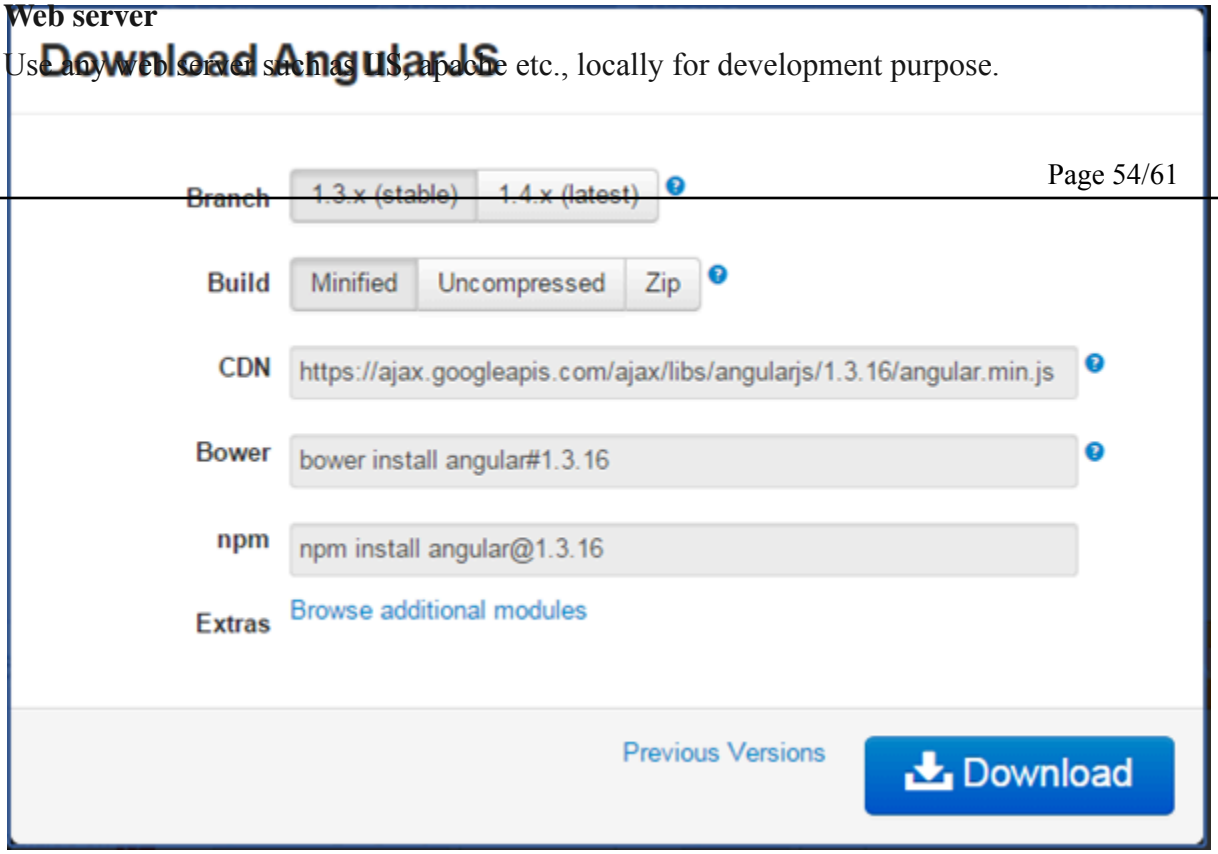
- plnkr.co
- jsbin.com

We are using our own online code editor for all the AngularJS examples in these tutorials.

**Web server**

Use any web server such as IIS, apache etc., locally for development purpose.





**Browser**
You can install any browser of your choice as AngularJS supports cross-browser compatibility. However, it is recommended to use Google Chrome while developing an application.

**Angular Seed**
Use Angular seed project to quickly get started on AngularJS application. The Angular-seed is an application skeleton for a typical AngularJS web application. You can use it to quickly bootstrap your angular webapp projects and development environment for your project.

Let's setup Angular project in Visual Studio 2013 for web.

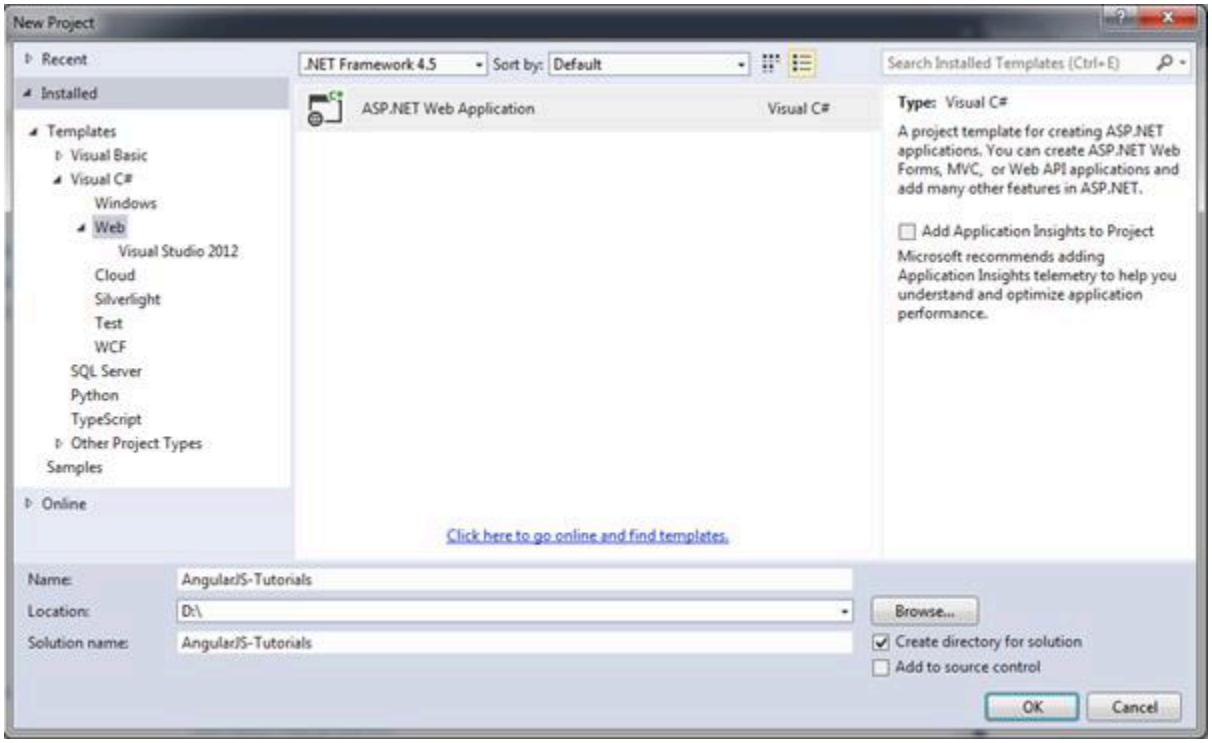
**Setup AngularJS Project in Visual Studio**
You can create AngularJS application in any version of Visual Studio. Here, we will use Visual Studio 2013 for web.

First, create new project by clicking on New Project link on start page. This will open New Project dialog box, as shown below.

AngularJS in Visual Studio

Select Web in the left pane and ASP.NET Web Application in the middle pane and then click OK.

In the New ASP.NET Project dialog box, select Empty template and then click OK.

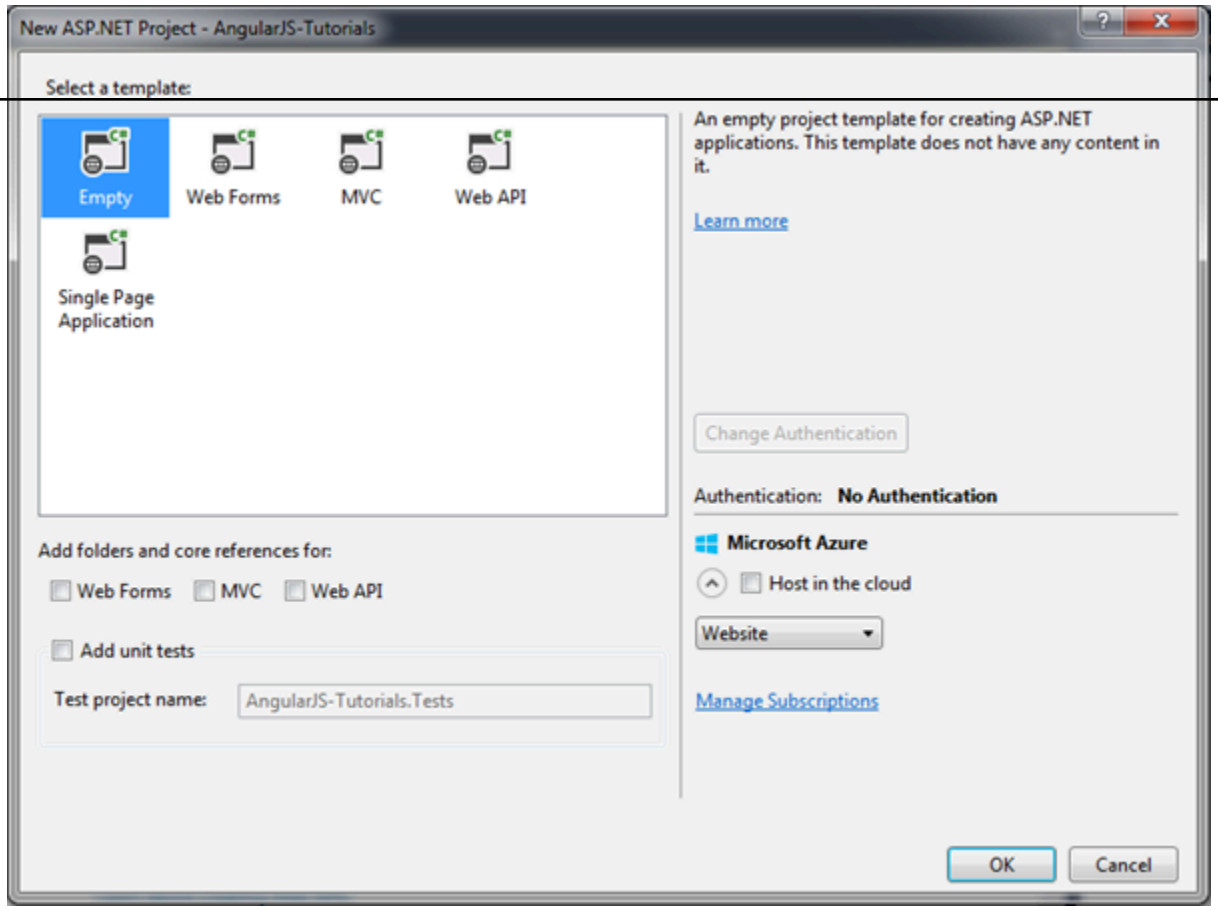

New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows
Web
Visual Studio 2012
Cloud
Silverlight
Test
WCF
SQL Server
Python
TypeScript
Other Project Types
Samples
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
ASP.NET Web Application
Visual C#
Type: Visual C#
A project template for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
Add Application Insights to Project
Microsoft recommends adding Application Insights telemetry to help you understand and optimize application performance.
Click here to go online and find templates.
Name:
AngularJS-Tutorials
Location:
D:\
Solution name:
AngularJS-Tutorials
Browse...
Create directory for solution
Add to source control
OK
Cancel

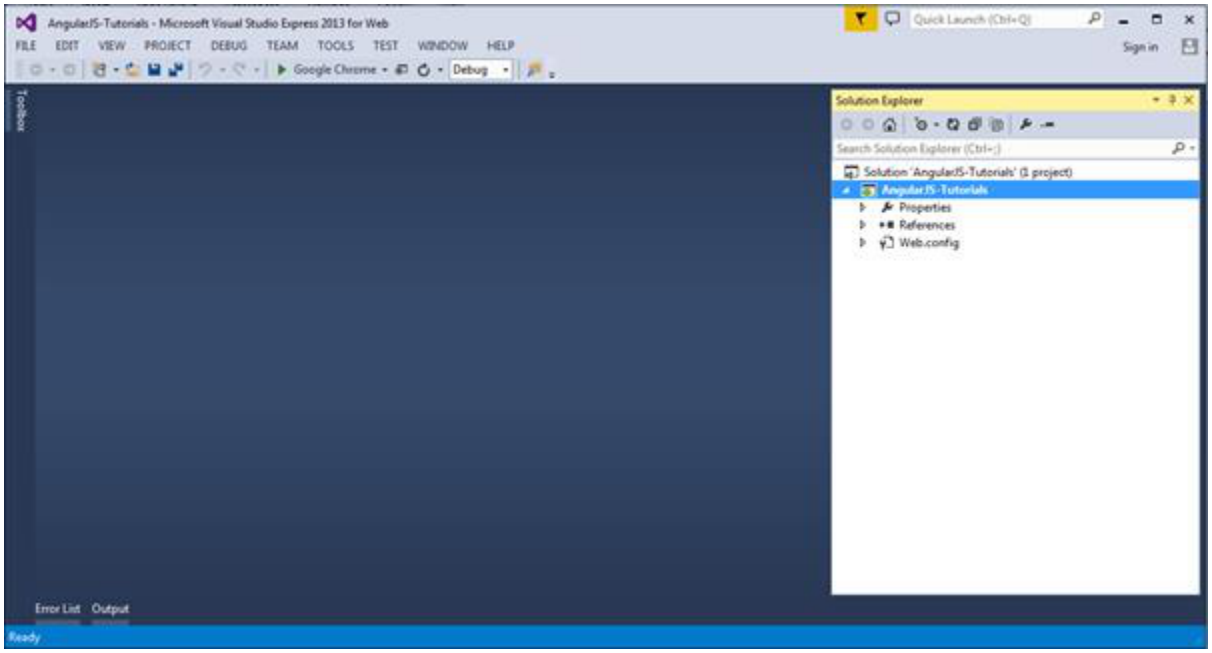
AngularJS in Visual Studio

This will create an empty website project in Visual Studio.

AngularJS in Visual Studio

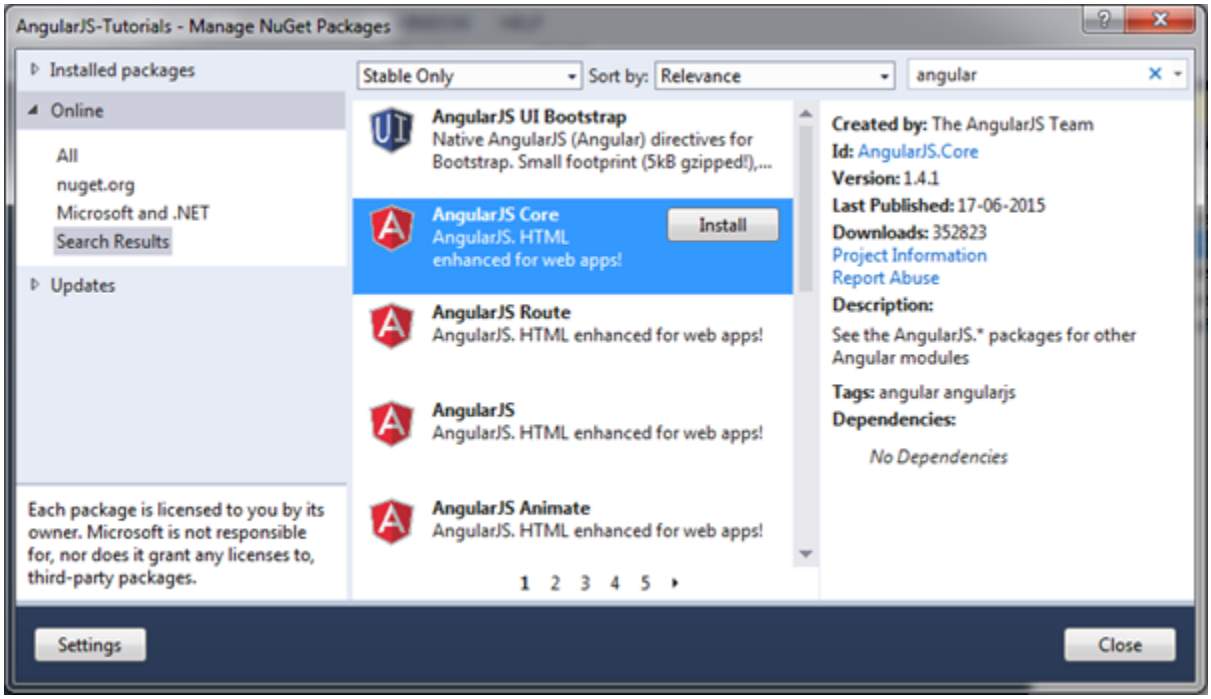
Now, install AngularJS library from NuGet package manager. Right click on the project in Solution Explorer and select Manage NuGet Packages..

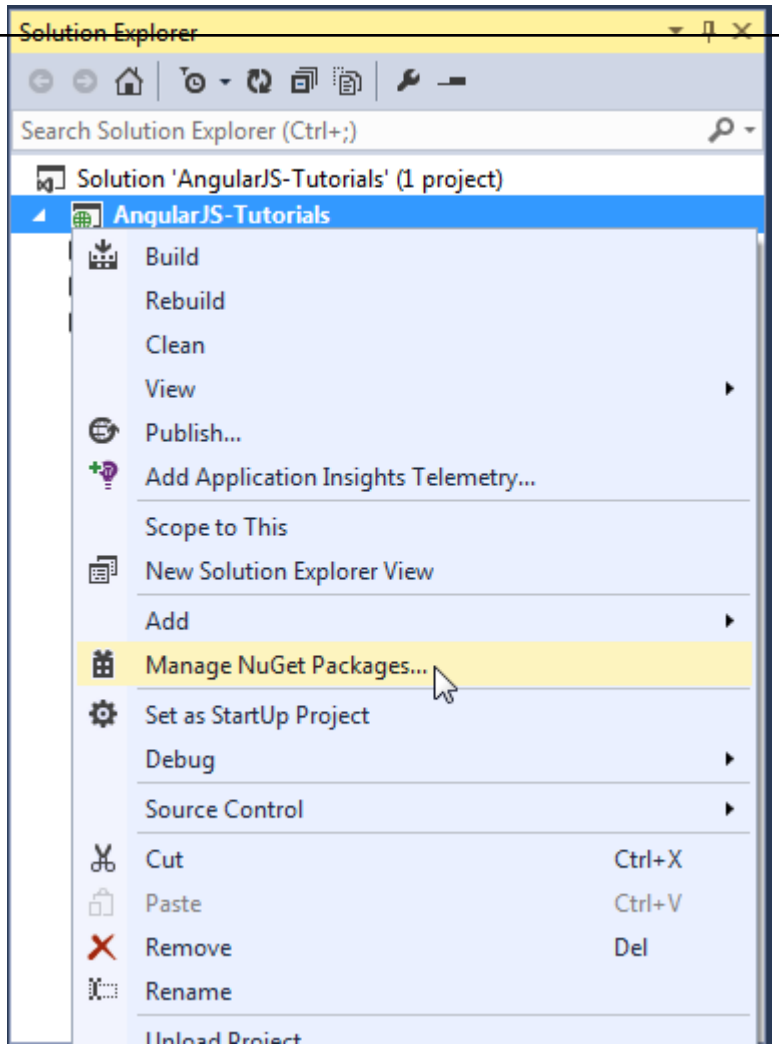

New ASP.NET Project - AngularJS-Tutorials
Select a template:
Empty
Web Forms
MVC
Web API
Single Page Application
An empty project template for creating ASP.NET applications. This template does not have any content in it.
Learn more
Change Authentication
Authentication: No Authentication
Microsoft Azure
Host in the cloud
Website
Manage Subscriptions
Add folders and core references for:
Web Forms
MVC
Web API
Add unit tests
Test project name:
AngularJS-Tutorials.Tests
OK
Cancel


AngularJS-Tutorials - Microsoft Visual Studio Express 2013 for Web
FILE
EDIT
VIEW
PROJECT
DEBUG
TEAM
TOOLS
TEST
WINDOW
HELP
Google Chrome
Debug
Solution Explorer
Solution 'AngularJS-Tutorials' (1 project)
AngularJS-Tutorials
Properties
References
Web.config
Error List
Output
Ready

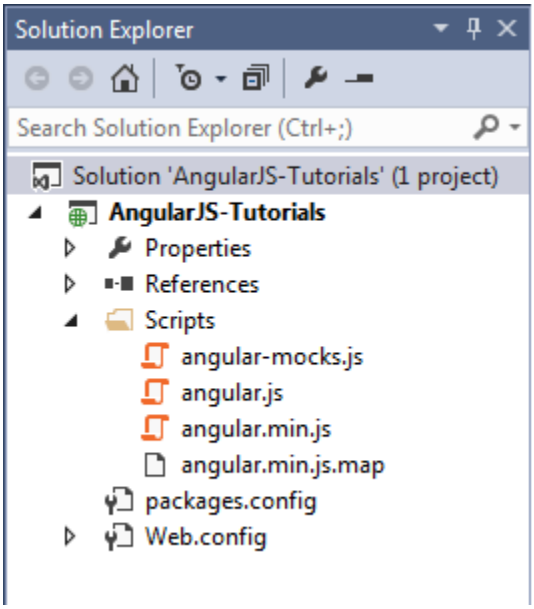
AngularJS in Visual Studio

Search for "angular" in the Manage NuGet Packages dialog box and install AngularJS Core.




Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'AngularJS-Tutorials' (1 project)
AngularJS-Tutorials
Build
Rebuild
Clean
View
Publish...
Add Application Insights Telemetry...
Scope to This
New Solution Explorer View
Add
Manage NuGet Packages...
Set as StartUp Project
Debug
Source Control
Cut Ctrl+X
Paste Ctrl+V
Remove Del
Rename

This will add AngularJS files into Scripts folder such as angular.js, angular.min.js, and angular-mocks.js, as shown below.




AngularJS in Visual Studio

## CODE

Let's create a simple AngularJS web application step by step and understand the basic building blocks of AngularJS.

1. First, create an HTML document with <head> and <body> elements, as show below.

Example: HTML Template

```
<!DOCTYPE html>
<html>
   <head>
   </head>
<body>
  </body>
</html>
```

2. Include angular.js file in the head section (you have learned how to download angular library in the previous section). You can take a reference from the CDN also. (all the examples in this tutorials will use CDN reference.)

Example: Include AngularJS Library

```
<!DOCTYPE html>
<html>
<head>
   <title>First AngularJS Application</title>
   <script src= "~/Scripts/angular.js"></script>
</head>
```

```
<body>
  </body>
</html>
```

3. Here, we will be creating a simple multiplier application which will multiply two numbers and display the result. User will enter two numbers in two separate textboxes and the result will be displayed immediately, as shown below.

First AngularJS Application

The following is the HTML code with AngularJS for the above multiplier example.

Example: First AngularJS Application

```
<!DOCTYPE html>

<html>
<head>
   <title>First AngularJS Application</title>
   <script src= "~/Scripts/angular.js"></script>
</head>
<body ng-app >
   <h1>First AngularJS Application</h1>

   Enter Numbers to Multiply:
   <input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" />
   = <span>{{Num1 * Num2}}</span>
</body>
</html>
```

The above example is looks like HTML code with some strange attributes and braces such as ng-app, ng-model, and {{ }}. These built-in attributes in AngularJS are called directives.

The following figure illustrates the AngularJS building blocks in the above example.

```
<!DOCTYPE html>
<html>
<head>
    <title>First AngularJS Application</title>
    <script src="~/Scripts/angular.js"></script>
</head>
<body ng-app>
    <h1>First AngularJS Application</h1>
    Enter Numbers to Multiply:
    <input type="text" ng-model="Num1" /> x <input type="text" ng-model="Num2" />
    = <span>{{Num1 * Num2}}</span>
</body>
</html>
```

Template

Directives

Expression

© TutorialsTeacher.com

First AngularJS Application

**Template**

In AngularJS, a template is HTML with additional markups. AngularJS compiles templates and renders the resultant HTML.

**Directive**

Directives are markers (attributes) on a DOM element that tell AngularJS to attach a specific behavior to that DOM element or even transform the DOM element and its children.

Most of the directives in AngularJS are starting with **ng**. It stands for Angular. We have applied ng-app and ng-model directive in the above example.

**ng-app**: The ng-app directive is a starting point. If AngularJS framework finds ng-app directive anywhere in the HTML document then it bootstraps (initializes) itself and compiles the HTML template.

**ng-model**: The ng-model directive binds HTML element to a property on the $scope object. You will learn about this model later but for now let us consider this as a model property.

In the above example, we have included ng-model directive to both the textboxes with different names Num1 and Num2. AngularJS framework will create two properties called Num1 and Num2 in the scope and will assign a value that we type into textboxes.

**Expression**

An expression is like JavaScript code which is usually wrapped inside double curly braces such as {{ expression }}. AngularJS framework evaluates the expression and produces a result. In the above example, {{ Num1 * Num2}} will simply display the product of Num1 and Num2.

The following table lists all the important concepts in AngularJS.

| Concept | Description |
| --- | --- |
| Template | HTML with additional markup |
| Directives | Extends the HTML with custom attributes and elements |

| Concept | Description |
| --- | --- |
| Model | The data shown to the user in the view and with which the user interacts |
| Scope | A context where the model is stored so that controllers, directives and expressions can access it |
| Expressions | Executes JavaScript code inside brackets {{ }}. |
| Compiler | Parses the template and instantiates directives and expressions |
| Filter | Formats the value of an expression for display to the user |
| View | what the user sees (the DOM) |
| Data Binding | Sync data between the model and the view |
| Controller | Maintains the application data and business logic |
| Module | a container for different parts of an app including controllers, services, filters, directives which configure the Injector |
| Service | Reusable business logic, independent of views |
| Dependency Injection | Creates and wires objects and functions |
| Injector | Dependency injection container |

**RESULT**

Thus a web application had been created using Angular framework.