**EX.NO:**

**DATE:**

**1. WEB PAGE CREATION USING HTML**

**Aim:**

To create a web page which includes a map and display the related information when a hot spot is clicked in the map.

**Procedure:**

* Create a html file with map tag
* Set the source attribute of the img tag to the location of the image and also set the use map attribute
* Specify an area with name, shape and href set to the appropriate values
* Repeat step 3 as many hot spots you want to put in the map
* Create html files for each and every hot spot the user will select

**Program:**

**ImageMap.html**

<HTML>

<HEAD>

<TITLE>Image Map</TITLE> </HEAD>

<BODY>

<img src="india\_map.jpg" usemap="#metroid" ismap="ismap" > <map name="metroid" id="metroid">

<area href='TamilNadu.html' shape='circle' coords='175,495,30' title='TamilNadu'/>

<area href = "Karnataka.html" shape = "rect" coords = "100,400,150,450" title = "Karnataka" /> <area href = "AndhraPradesh.html" shape = "poly" coords = "150, 415, 175,348,265,360,190,420,190,440" title = "Andhra Pradesh" />

<area href = "Kerala.html" shape = "poly" coords = "108,455,150,515,115,490,148,495,110,448,155,501" title = "Kerala" /> </map>

</BODY>

</HTML>

**TamilNadu.html**

<HTML>

<HEAD>

<TITLE>About Tamil Nadu</TITLE>

</HEAD>

<BODY>

<CENTER><H1>Tamil Nadu</H1></CENTER> <HR>

<UL>

<LI>Area : 1,30,058 Sq. Kms.</LI>

<LI>Capital : Chennai</LI>

<LI>Language : Tamil</LI>

<LI>Population : 6,21,10,839</LI> </UL><hr>

<a href='ImageMap.html'>India Map</a>

</BODY>

</HTML>

**Karnataka.html**

<HTML>

<HEAD>

<TITLE>About Karnataka</TITLE> </HEAD>

<BODY>

<CENTER><H1>Karnataka</H1></CENTER>

<HR>

<UL>

<LI>Area : 1,91,791 Sq. Kms</LI>

<LI>Capital : Bangalore</LI>

<LI>Language : Kannada</LI>

<LI>Population : 5,27,33,958</LI>

</UL>

<hr>

<a href='ImageMap.html'>India Map</a>

</BODY>

</HTML>

**AndhraPradesh.html**

<HTML>

<HEAD>

<TITLE>About Andhra Pradesh</TITLE> </HEAD>

<BODY>

<CENTER><H1>Andhra Pradesh</H1></CENTER> <HR>

<UL>

<LI>Area : 2,75,068 Sq. Kms</LI>

<LI>Capital : Hyderabad</LI>

<LI>Language : Telugu</LI>

<LI>Population : 7,57,27,541</LI>

</UL>

<hr>

<a href='ImageMap.html'>India Map</a>

</BODY>

</HTML>

**Kerala.html**

<HTML>

<HEAD>

<TITLE>About Kerala</TITLE>

</HEAD>

<BODY>

<CENTER>

<H1>Kerala</H1></CENTER>

<HR>

<UL>

<LI>Area : 38,863 Sq. Kms.</LI>

<LI>Capital : Thiruvananthapuram</LI>

<LI>Language : Malayalam</LI>

<LI>Population : 3,18,38,619</LI>

</UL>

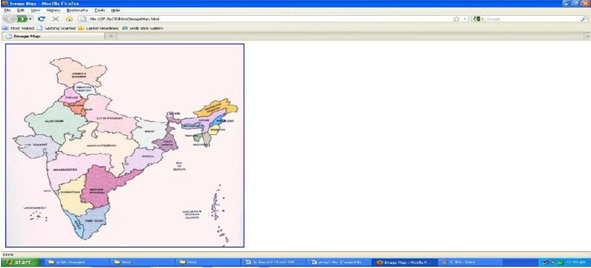
<hr>

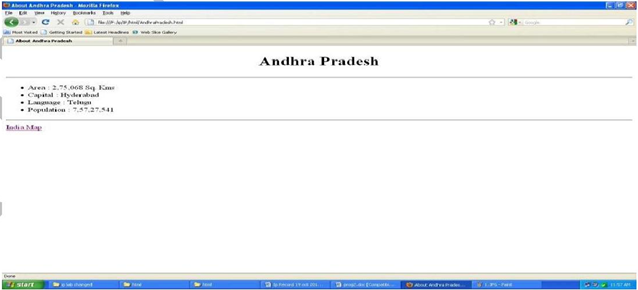
<a href='ImageMap.html'>India Map</a>

</BODY>

</HTML>

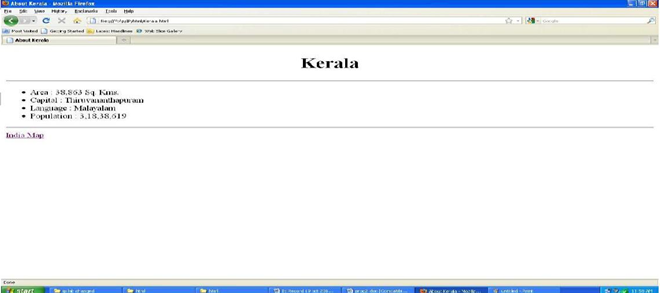
**Output:**











**Result:**

Thus the creation of a web page which includes a map and display the related in-formation when a hot spot is clicked in the map was executed successfully.

**EX.NO:**

**DATE:**

**2. CASCADING STYLE SHEETS**

**Aim:**

To create a web page that displays college information using various style sheet

**Procedure:**

* Create a web page with frame sets consisting two frames
* In the first frame include the links
* In the second frame set display the web page of the link
* Create a external style sheets
* Create a embedded style sheets
* Create a inline and internal style sheets and make it link to the external style sheets

**Program:**

**CSS CODE:**

**External.css**

h3{font-family:arial;font-size:20;color:cyan}

table{border-color:green}

td{font-size:20pt;color:magenta}

**HTML CODE:**

**Style.html**

<html>

<head><h1><center>ALL STYLE SHEETS</center></h1>

<title>USE of STYLESHEETS

</title>

<link rel="stylesheet" href="External.css" type="text/css"> <!-- External Style Sheet -->

<style type="text/css"> <!-- Internal Style Sheet -->

.S1{font-family:verdana; font-style:italic; color:red; text-align:center}

.S2{font-family:tahoma; font-style:italic; font-size:20; text-align:center;}

font{font-family:georgia; color:blue; font-size:20}

ul{list-style-type:circle}

</style>

</head>

<body>

<ol style="list-style-type:lower-alpha">

<b> Anna University </b><br><br><br>

<li> University College of Engineering, Nagercoil

<li> University College of Engineering, Nellai

<li> University College of Engineering, Tuticorin

</ol>

<p style="font-size:20pt;color:purple">

University College of Engineering</p> <!-- InlineStyle Sheet -->

<p class="S2"> Run by Anna University, Chennai<br>

It is approved by AICTE.

<br>

</p>

<h2 class="S1"> University College of Engineering</h2>

<br>

<font>Located in Konam, Nagercoil</font><br>

<br>

<font>

<h2>List of Courses offered</h2>

<ul>

<li>CSE</li>

<li>IT</li>

<li>ECE</li>

<li>EEE</li>

<li>MECH</li>

<li>Civil</li>

</ul>

</font>

<h3>Pass percentage in year 2015</h3>

<table width="100%" cellspacing="2" cellpadding="2" border="5">

<tr>

<th>Sl.No</th>

<th>Dept</th>

<th>Pass Percentage</th>

</tr>

<tr>

<td align="center">1</td>

<td align="center">CSE</td>

<td align="center">80</td>

</tr>

<tr>

<td align="center">2</td>

<td align="center">ECE</td>

<td align="center">78</td>

</tr>

<tr>

<td align="center">3</td>

<td align="center">Mech</td>

<td align="center">75</td>

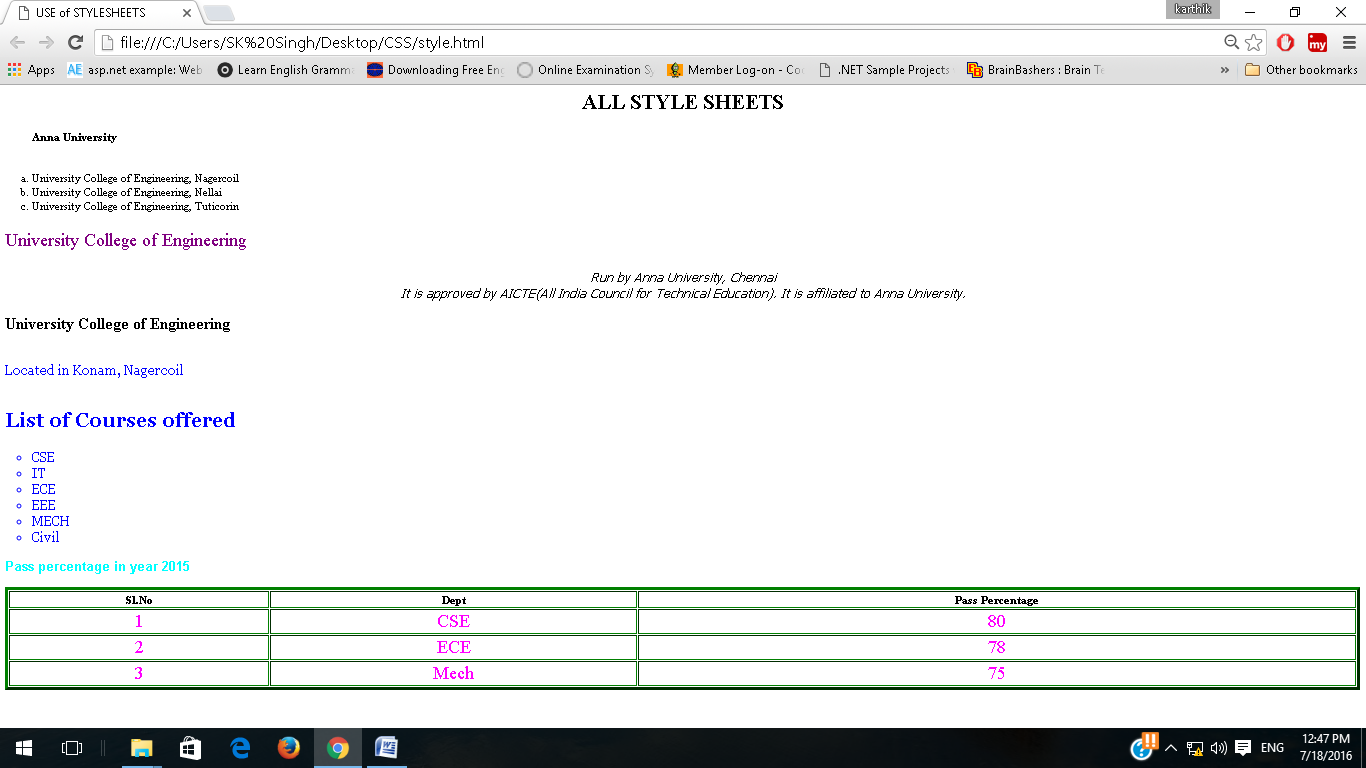
</tr>

</table>

</body>

</html>

**Output:**

****

**Result:**

Thus the creation of a web page that displays college information using various style sheet was successfully executed and verified.

**EX.NO:**

**DATE:**

**3. RETRIEVING USER INFORMATION FROM XML DOCUMENT**

**Aim:**

To create and save an XML document at the server, which contain ten users information. To write a program which takes user id as an input and returns the user details by taking the user information from the XML document.

**Procedure:**

* Save Students information in the XML file on the specific location.
* Create and establish the connection between html file and XML file.
* Get the user ID as input
* Display the student’s information.

**StudentDetails.xml :**

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

<Student>

<PersonDetails>

<id>101</id>

<name>Anand</name>

<city>Madurai</city>

<Branch>CSE</Branch>

<Year>I</Year>

</PersonDetails>

<PersonDetails>

<id>102</id>

<name>Anu</name>

<city>Konam</city>

<Branch>CSE</Branch>

<Year>II</Year>

</PersonDetails>

<PersonDetails>

<id>103</id>

<name>Archana</name>

<city>Madurai</city>

<Branch>CSE</Branch>

<Year>I</Year>

</PersonDetails>

<PersonDetails>

<id>104</id>

<name>Monica</name>

<city>Nellai</city>

<Branch>CSE</Branch>

<Year>III</Year>

</PersonDetails>

</Student>

**LogIn.html :**

<!DOCTYPE html>

<html>

<head>

</head>

<body>

<script type="text/javascript">

function Display()

{

if(window.XMLHttpRequest)

{

xmlhttp=new XMLHttpRequest();

}

xmlhttp.open("GET","UserInfo.xml",false);

xmlhttp.send();

xmlDoc=xmlhttp.responseXML;

var x=xmlDoc.getElementsByTagName("PersonDetails");

var key\_id=document.getElementById("key").value;

for(i=0;i<x.length;i++)

{

if(key\_id.match(x[i].getElementsByTagName("id")[0].childNodes[0].nodeValue))

j=i;

}

document.write("<h3>User Details are...</h3> <hr> Registeration ID=");

document.write(x[j].getElementsByTagName("id")[0].childNodes[0].nodeValue);

document.write("</br> Name=");

document.write(x[j].getElementsByTagName("name")[0].childNodes[0].nodeValue);

document.write("</br> City=");

document.write(x[j].getElementsByTagName("city")[0].childNodes[0].nodeValue);

document.write("</br> Branch=");

document.write(x[j].getElementsByTagName("Branch")[0].childNodes[0].nodeValue);

document.write("</br> Year=");

document.write(x[j].getElementsByTagName("Year")[0].childNodes[0].nodeValue);

document.write("</br> ");

}

</script>

<form name='myform'>

Enter ID:

<input type='text' id='key'/><br/>

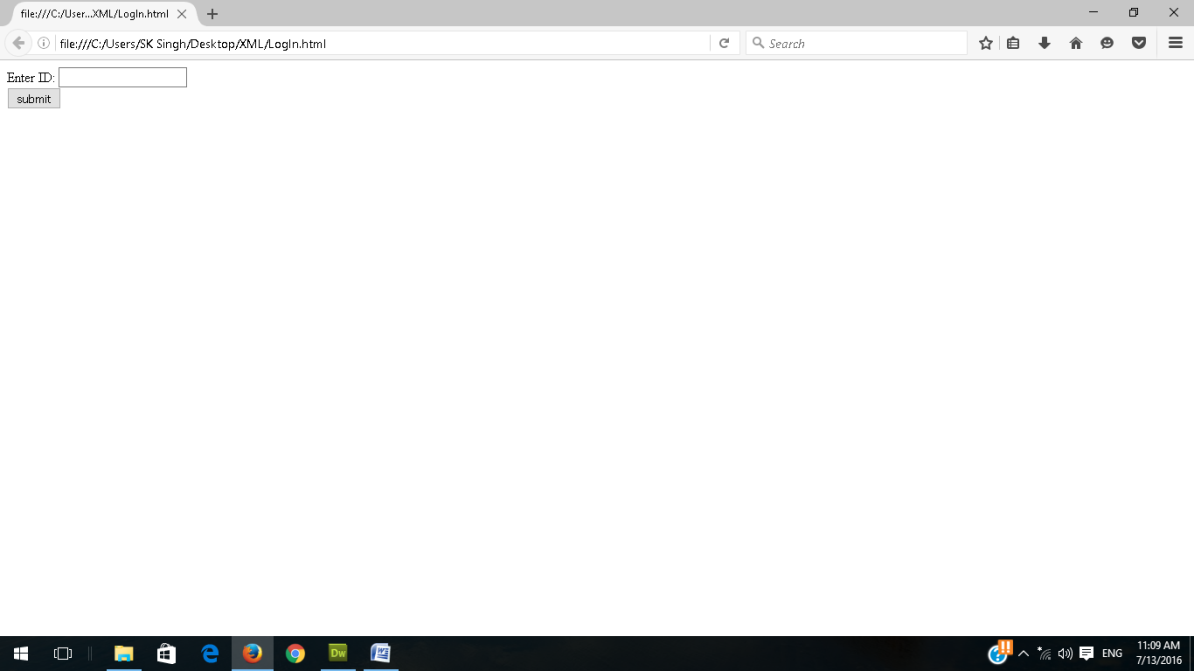
<input type='button' value='submit' onclick='Display()'/>

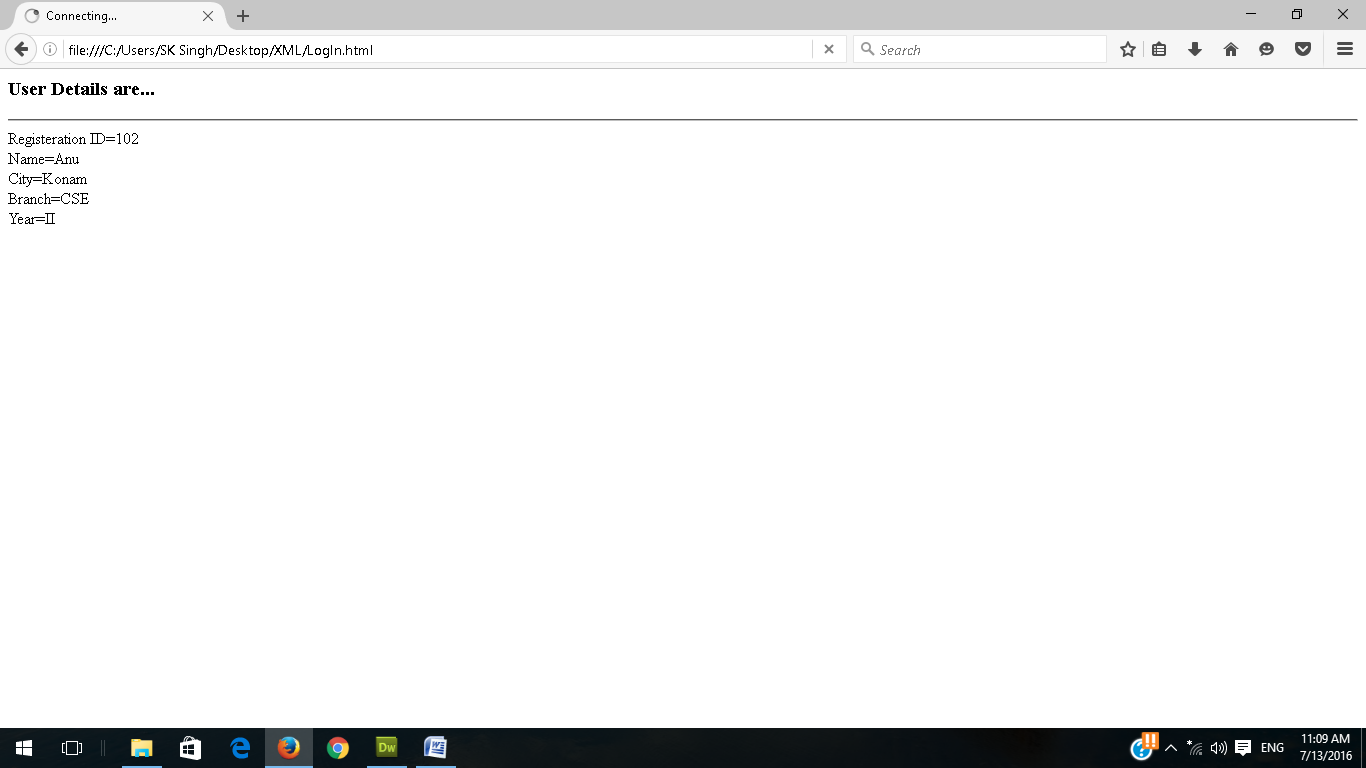
</form>

</body>

</html>

**Output:**





**Result:**

Thus the Program takes user id as an input and returns the user details by taking the user information from the XML document has been executed successfully.

**EX.NO:**

**DATE:**

**4.(a) INVOKING SERVLETS FROM HTML FORM**

**Aim:**

To write a java program to invoke servlets from HTML form.

**Procedure:**

***client.html:***

(1) Create a web page using HTML form that contains the fields such as label,text 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.

**Program:**

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServletHtmlDemo extends HttpServlet

{

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

{

PrintWriter out = res.getWriter();

Enumeration en = req.getParameterNames();

while(en.hasMoreElements())

{

String name=(String)en.nextElement();

out.print(name+"=");

String value=req.getParameter(name);

out.println(value);

out.println(" ");

}

}

}

***web.xml:***

<web-app>

<servlet>

<servlet-name>MyServletHtmlDemo</servlet-name>

<servlet-class>MyServletHtmlDemo</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MyServletHtmlDemo</servlet-name>

<url-pattern>/MyServletHtmlDemo</url-pattern>

</servlet-mapping>

</web-app>

**MyServletHtmlDemo*.HTML:***

<html>

<head>

<title>Student Information Form</title>

</head>

<body>

<center>

<form name="form1" action="MyServletHtmlDemo">

<h3>Enter student information in following fields</h3>

<table>

<tr>

<td><b>Roll Number</b></td>

<td><input type="text" name="Roll Number" size="25" value=""></td>

</tr>

<tr>

<td><b>Student Name</b></td>

<td><input type="text" name="Student Name" size="25" value=""></td>

</tr>

<tr>

<td><b>Student Address</b></td>

<td><input type="text" name="Address" size="25" value=""></td>

</tr>

<tr>

<td><b>Phone</b></td>

<td><input type="text" name="Phone" size="25" value=""></td>

</tr>

<tr>

<td><b>Total Marks</b></td>

<td><input type="text" name="Total Marks " size="25" value=""></td>

</tr>

</table>

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

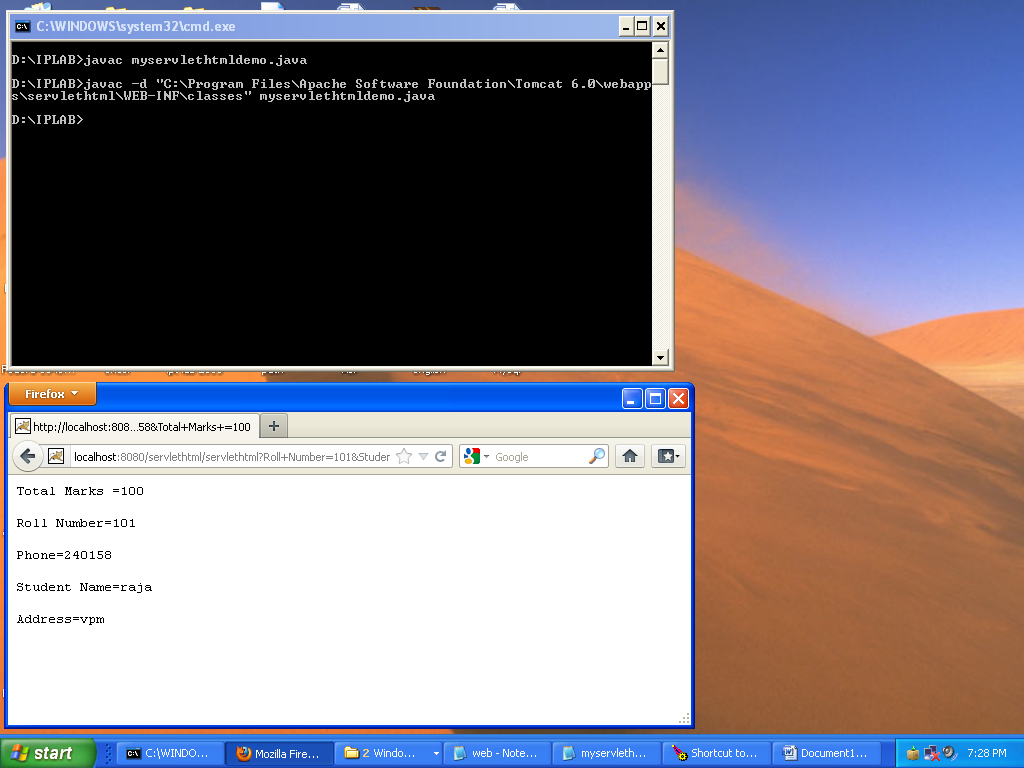
</form>

</center>

</body>

</html>

**Output:**



**Result:**

Thus the java program to invoke servlets from HTML form has been executed successfully.

**EX.NO:**

**DATE:**

**4.(b) SESSION TRACKING A HIT COUNT**

**Aim:**

To write a Java Program for Session tracking a hit count. This servlet uses session tracking to count the number of times a client has accessed it.

**Procedure:**

1. Servlet program to keep track of user visiting the page.
2. The count is incremented by one when user visits.
3. The output displays the greeting message.
4. The number of previous access is also displayed.

**Program:**

import java.io.\*;

import java.util.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class SessionServletDemo extends HttpServlet

{

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

{

res.setContentType("text/html");

HttpSession session = req.getSession();

String heading;

Integer cnt=(Integer)session.getAttribute("cnt");

if(cnt==null)

{

cnt=new Integer(0);

heading="Welcome you are accessing the page for the first time";

}

else

{

heading="Welcome once again";

cnt=new Integer(cnt.intValue()+1);

}

session.setAttribute("cnt",cnt);

PrintWriter out = res.getWriter();

out.println("<html>");

out.println("<head>");

out.println("</head>");

out.println("<body>");

out.println("<center>");

out.println("<h1>"+heading);

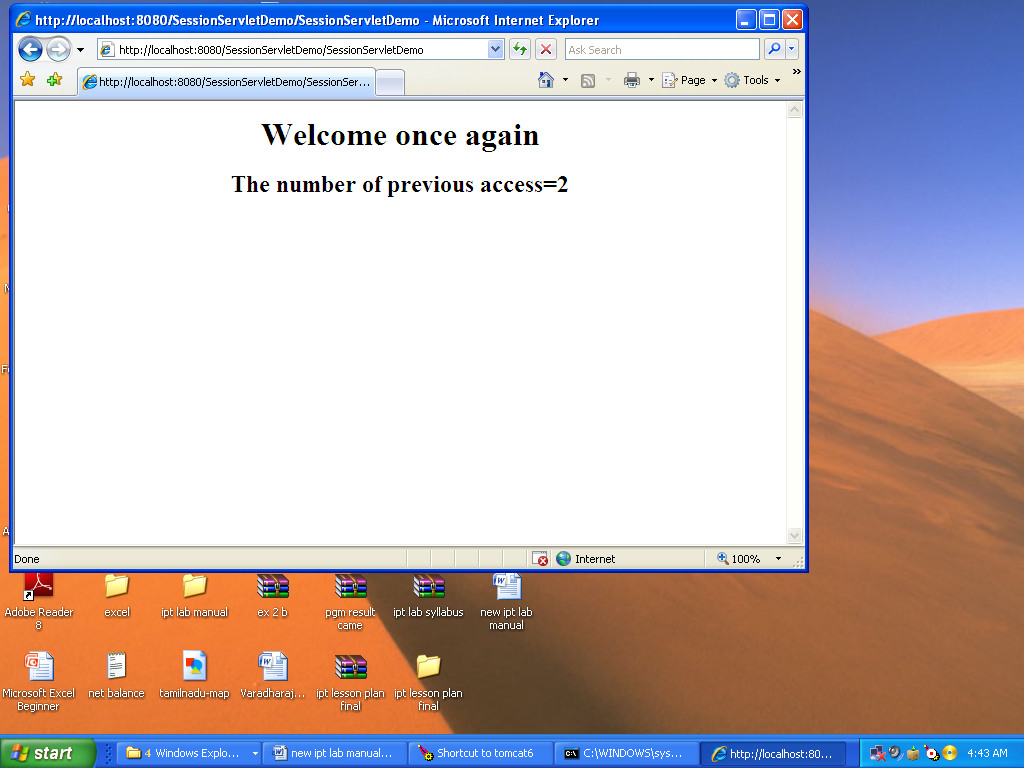
out.println("<h2>The number of previous access="+cnt);

out.println("</center>");

out.println("</body>"); out.println("</html>"); }

}

**Output:**



**Result:**

Thus the Java program for Session tracking a hit count has been executed successfully.

**EX.NO:**

**DATE:**

**5. THREE-TIER APPLICATIONS USING SERVLETS**

**Aim:**

To write java servlet programs to conduct online examination and to display student mark list available in a database.

**Procedure:**

**Client:**

* In index.html on the client side declare the contents that you like to transfer to the server using html form and input type tags.
* Create a submit button and close all the included tags.

**Server:**

* Import all necessary packages
* Define a class that extends servlet
* In the doPost() method, do the following:

i) Set the content type of the response to "text/html"

ii) Create a writer to the response

iii) Get a paratmeter from the request

iv) If its value is equal to right answer then add 5 to mark variable

v) Similarly repeat step

vi) for all parameters

vii) Display the result in an html format using the writer

**Student Mark List Database:**

* Import necessary to java packages and javax packages and classes
* Create a class that extends HttpServlet and implements ServletException
* and IOException
* In the doGet() method, do the following:

i) Create a PrintWriter object

ii) Open a connection with the data source name

iii) Write a sql query and execute to get the resultset

iv) Display the resultset information in html form

**Program:**

**SERVLET CODE:**

import java.io.\*; import java.sql.\*; import javax.servlet.\*;

import javax.servlet.http.\*;

public class StudentServlet3 extends HttpServlet

{

String message,Seat\_no,Name,ans1,ans2,ans3,ans4,ans5; int Total=0;

Connection connect; Statement stmt=null; ResultSet rs=null;

public void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletExcep-tion,IOException

{

try

{

String url="jdbc:odbc:NEO"; Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); connect=DriverManager.getConnection(url," "," "); message="Thank you for participating in online Exam";

}

catch(ClassNotFoundException cnfex){ cnfex.printStackTrace();

}

catch(SQLException sqlex){ sqlex.printStackTrace();

}

catch(Exception excp){ excp.printStackTrace();

}

Seat\_no=request.getParameter("Seat\_no"); Name=request.getParameter("Name"); ans1=request.getParameter("group1"); ans2=request.getParameter("group2"); ans3=request.getParameter("group3"); ans4=request.getParameter("group4"); ans5=request.getParameter("group5"); if(ans1.equals("True"))

Total+=2;

if(ans2.equals("False"))

Total+=2;

if(ans3.equals("True"))

Total+=2;

if(ans4.equals("False"))

Total+=2;

if(ans5.equals("False"))

Total+=2; try

{

Statement stmt=connect.createStatement();

String query="INSERT INTO student("+"Seat\_no,Name,Total"+") VAL-UES('"+Seat\_no+"','"+Name+"','"+Total+"')";

int result=stmt.executeUpdate(query); stmt.close();

}catch(SQLException ex){

}

response.setContentType("text/html"); PrintWriter out=response.getWriter(); out.println("<html>"); out.println("<head>"); out.println("</head>"); out.println("<body bgcolor=cyan>"); out.println("<center>"); out.println("<h1>"+message+"</h1>\n");

out.println("<h3>Yours results stored in our database</h3>"); out.print("<br><br>");

out.println("<b>"+"Participants and their Marks"+"</b>"); out.println("<table border=5>");

try

{

Statement stmt=connect.createStatement(); String query="SELECT \* FROM student"; rs=stmt.executeQuery(query); out.println("<th>"+"Seat\_no"+"</th>"); out.println("<th>"+"Name"+"</th>"); out.println("<th>"+"Marks"+"</th>"); while(rs.next())

{

out.println("<tr>");

out.print("<td>"+rs.getInt(1)+"</td>");

out.print("<td>"+rs.getString(2)+"</td>");

out.print("<td>"+rs.getString(3)+"</td>");

out.println("</tr>");

}

out.println("</table>");

}

catch(SQLException ex){ } finally

{

try

{

if(rs!=null)

rs.close();

if(stmt!=null)

stmt.close();

if(connect!=null)

connect.close();

}

catch(SQLException e){ }

}

out.println("</center>");

out.println("</body></html>");

Total=0;

} }

**HTML CODE:**

<html><head><title>Database Test</title></head> <body>

<center><h1>Online Examination</h1> </center>

<form action="StudentServlet3.view" method="POST"> <div align="left"><br></div>

<b>Seat Number:</b> <input type="text" name="Seat\_no"> <div align="Right">

<b>Name:</b> <input type="text" name="Name" size="50"><br> </div>

<br><br>

<b>1. Every host implements transport layer.</b><br/> <input type="radio" name="group1" value="True">True <input type="radio" name="group1" value="False">False<br>

<b>2. It is a network layer's responsibility to forward packets reliably from source to destina-tion</b><br/>

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

<input type="radio" name="group2" value="False">False<br>

<b>3. Packet switching is more useful in bursty traffic</b><br/> <input type="radio" name="group3" value="True">True<input type="radio" name="group3" value="False">False<br> <b>4. A phone network uses packet switching</b><br/> <input type="radio" name="group4" value="True">True <input type="radio" name="group4" value="False">False<br>

<b>5. HTML is a Protocol for describing web contents</b><br/> <input type="radio" name="group5" value="True">True

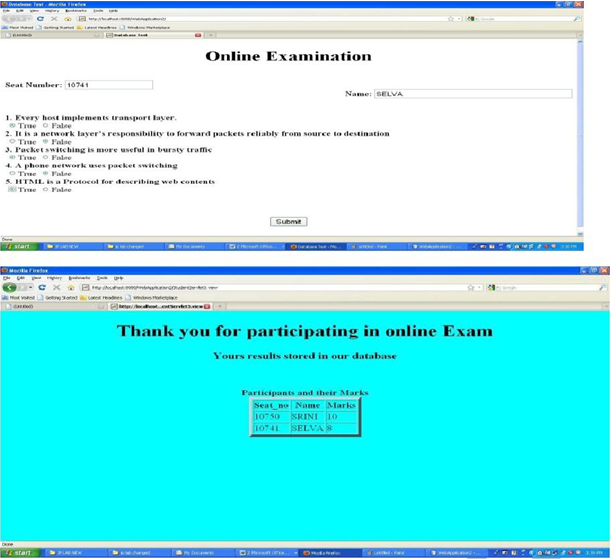
<input type="radio" name="group5" value="False">False<br> <br><br><br>

<center>

<input type="submit" value="Submit"><br><br> </center>

</form></body></html>

**Output:**



**Result:**

Thus to write java servlet programs to conduct online examination and to display student mark list available in a database was successfully executed and verified.

**EX.NO:**

**DATE:**

**6. CONVERTING STATIC WEB PAGES TO DYNAMIC WEB PAGES**

**Aim:**

To convert the static web pages into dynamic web pages using servlets (or JSP) and cookies.

**Procedure:**

**step1:** we will create ahtml form for entering the user name,password and card ID.

**Step2:**From the above HTML form, the servlet program is invoked in which the validity of the user name,password and card id is checked.if it is a valid user then the welcome message will be displayed otherwise the “invalid user” message will be displayed. In this servlet we set the cookies in which the current user name is stored.

**Step3:** compile the above servlet Login servlet.java and copy its class file in tomcats folder at c:\tomcatdirectory\webapps\examples\WEB-INF\classes.

Then edit the web.xml in WEB-INF folder.We must store he user information such as user name,password and card id in the web.xml using init-param.

**Step4:** On successful login , the information from the cookie is checked and shopping cart page for corressponding user can be displayed.

**Step5:**Compile the above servlet LoginSuccess.java and copy its class file in the tomcat's folder at c:\tomcatdirectory\webapps\examples\WEB-INF\classes.

Then edit the web.xml in WEB-INF folder.

**Step6:**Start tomcat web server.Open the web browser and display the login form created in step1.

**Index.jsp**

<html>

<head>

<body>

<form action="http://localhost:8084/ddd/LoginServlet" method="post">

Enter username:

<input type="text" value""name="user">

<br>

Enter Password:

<input type="password" value""name="password">

<br>

Enter Card ID:

<input type="text" value""name="cardID">

<br>

<br> <br> <br>

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

</form>

</body>

**Loginservlet.html**

import java.io.\*;

import java.net.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class LoginServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

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

PrintWriter out = response.getWriter();

try {

String usr=request.getParameter("user");

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

String card=request.getParameter("cardID");

boolean flag=true;

String[] userID=getInitParameter("usernames").split(",");

String[] password=getInitParameter("passwords").split(",");

String[] cardids=getInitParameter("cardIDs").split(",");

int i;

for(i=0;i<userID.length;i++)

{

if(userID[i].equals(usr)&&password[i].equals(pwd)&&cardids[i].equals(card))

{

flag=false;

Cookie MyCookie=new Cookie("CurrentUser", usr);

MyCookie.setMaxAge(60\*60);

response.addCookie(MyCookie);

response.sendRedirect("http://localhost:8084/ddd/LoginSuccess");

}

}

if(flag==true)

{

out.print("Error");

out.println("<h4>Invalid user,please try again by clicking following link</h4>");

out.println("<a href='http://localhost:8084/ddd/'>"+"LoginForm.html");

}

}

finally {

out.close();

}

}

**LoginSuccess.java**

import java.io.\*;

import java.net.\*;

import javax.servlet.\*;

import javax.servlet.http.\***;**

public class LoginSuccess extends HttpServlet {protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

Cookie[] my\_cookies=request.getCookies();

response.setContentType("text/html");

PrintWriter out=response.getWriter();

out.print("Login Success");

out.println("<b>");

String userName=null;

if(my\_cookies!=null)

{

for(Cookie cookie:my\_cookies)

{

if(cookie.getName().equals("currentUser"))

userName=cookie.getValue();

}

}

out.print("<h3>Login Success!!!Welcome</h3>");

out.print("<h2>This is a Shopping cart for"+userName+"</h2>");

out.close();

}

}

**Web.xml**

<servlet>

<servlet-name>LoginServlet</servlet-name>

<servlet-class>LoginServlet</servlet-class>

<init-param>

<param-name>usernames</param-name>

<param-value>user1,user2,user3</param-value>

</init-param>

<init-param>

<param-name>passwords</param-name>

<param-value>pwd1,pwd2,pwd3</param-value>

</init-param>

<init-param>

<param-name>cardIDs</param-name>

<param-value>111,222,333</param-value>

</init-param>

</servlet>

<servlet>

<servlet-name>LoginSuccess</servlet-name>

<servlet-class>LoginSuccess</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>LoginServlet</servlet-name>

<url-pattern>/LoginServlet</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>LoginSuccess</servlet-name>

<url-pattern>/LoginSuccess</url-pattern>

</servlet-mapping> <session-config> <session-timeout>

30

</session-timeout>

</session-config>

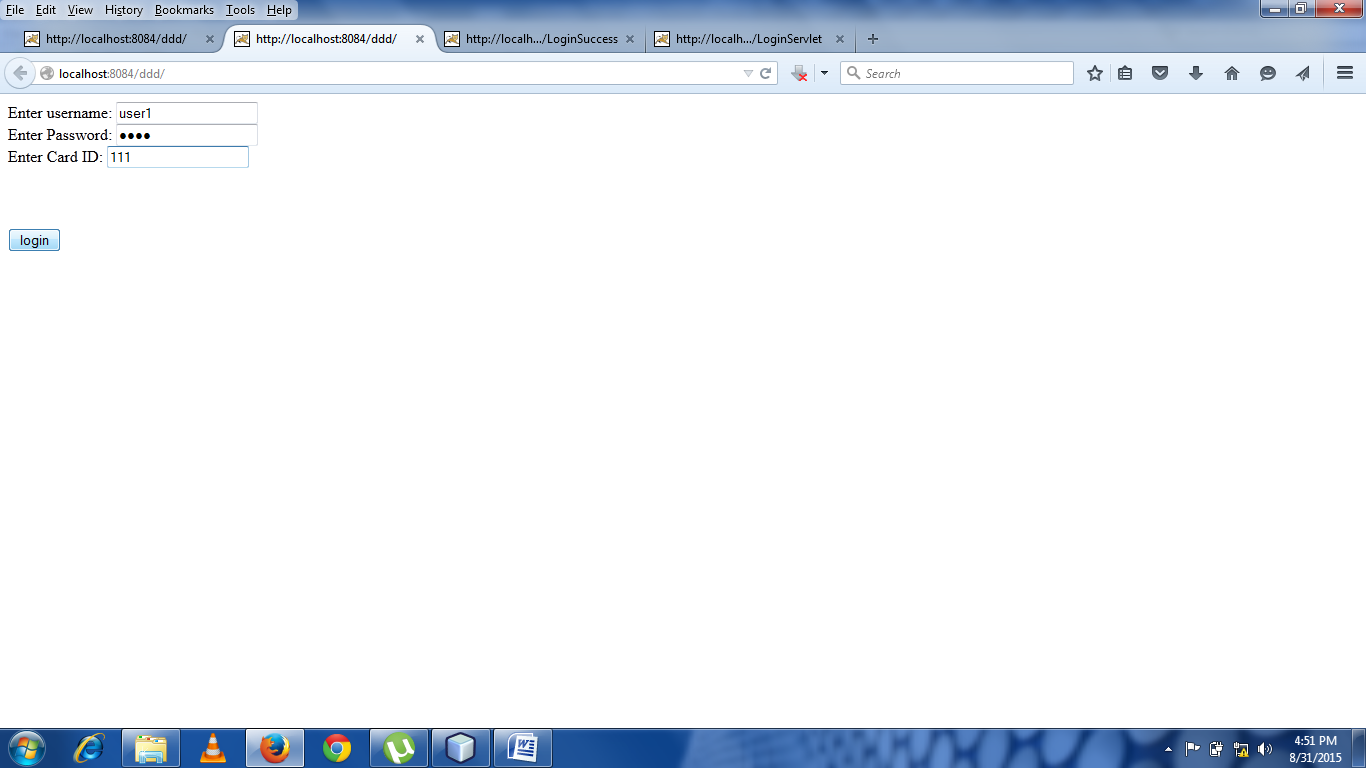
<welcome-file-list>

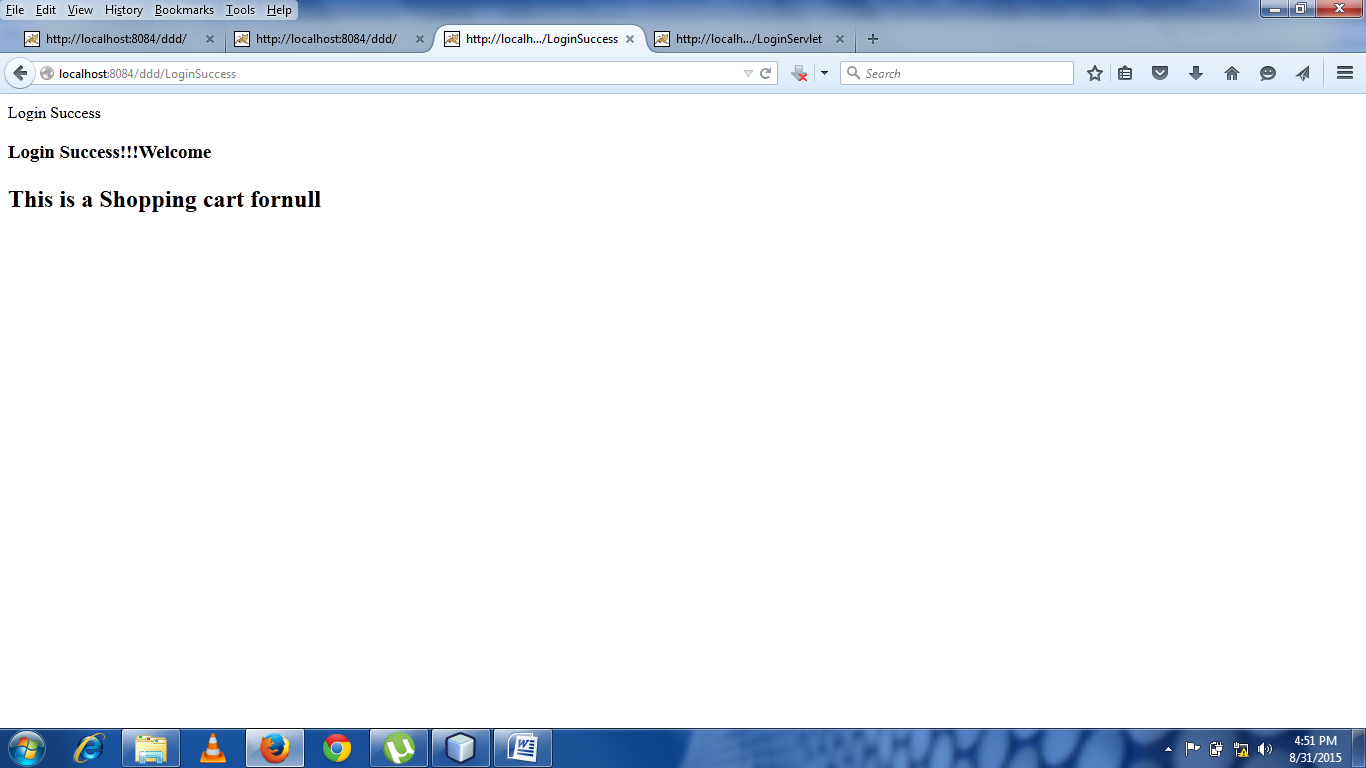
<welcome-file>index.jsp</welcome-file>

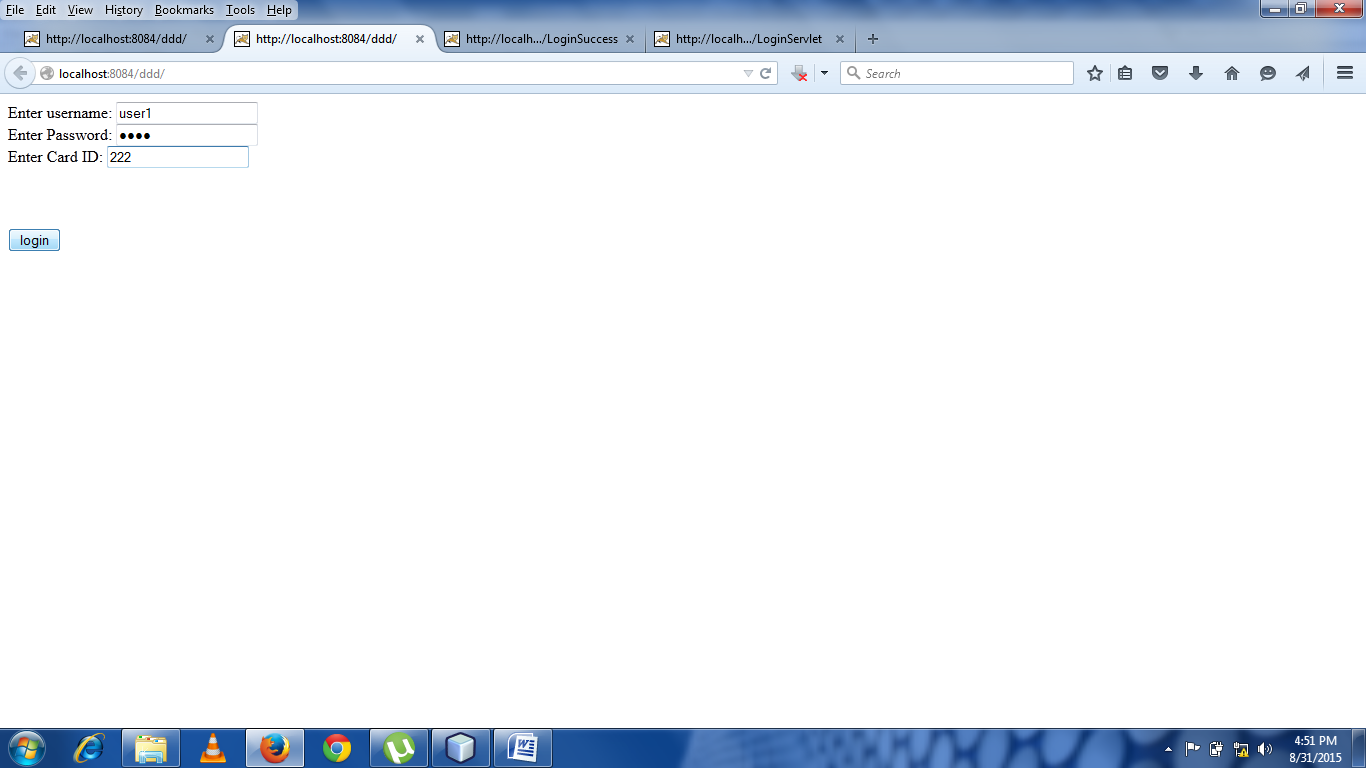
</welcome-file-list>

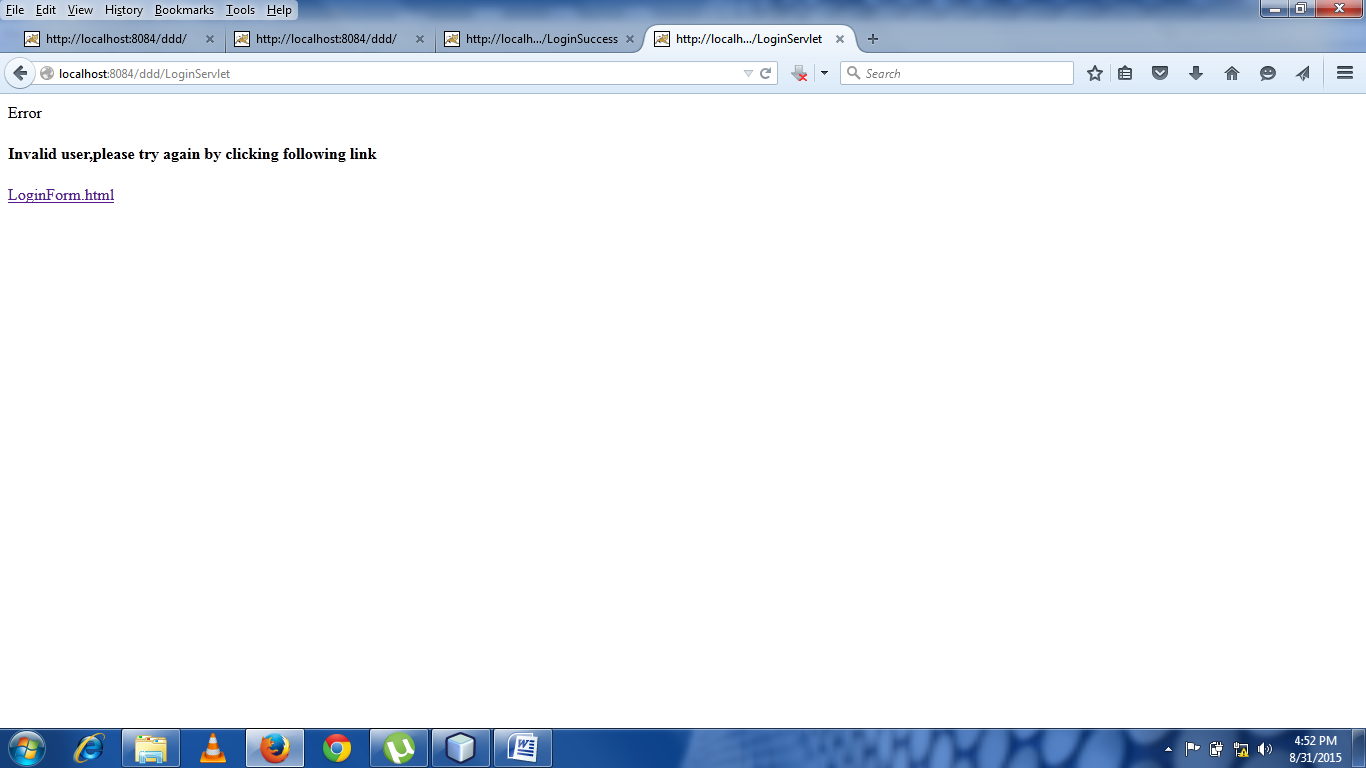
</web-app>

**Output:**









**Result:**

Thus the conversion of the static web pages into dynamic web pages using servlets cookies has been executed successfully.

**EX.NO:**

**DATE:**

**7.WRITE A WEB SERVICES FOR PREDICTING FOR ANY PRODUCT SALES**

**Aim:**

To write a web services for predicting for any product sales

**Procedure:**

* Open the Home page.
* View the 2 products.
* Put the rating by user.
* Display the high rating product on the top of the page.

**Program:**

***Page.html***

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd"

>

<html lang="en">

<head>

<title>AJAX 5 Star Rating</title>

<script src="http://code.jquery.com/jquery-latest.js"></script>

<script>

// This is the first thing we add ------------------------------------------

$(document).ready(function() {

$('.rate\_widget').each(function(i) {

var widget = this;

var out\_data = {

widget\_id : $(widget).attr('id'),

fetch: 1

};

$.post(

'ratings.php',

out\_data,

function(INFO) {

$(widget).data( 'fsr', INFO );

set\_votes(widget);

},

'json'

);

});

$('.ratings\_stars').hover(

// Handles the mouseover

function() {

$(this).prevAll().andSelf().addClass('ratings\_over');

$(this).nextAll().removeClass('ratings\_vote');

},

// Handles the mouseout

function() {

$(this).prevAll().andSelf().removeClass('ratings\_over');

// can't use 'this' because it wont contain the updated data

set\_votes($(this).parent());

}

);

// This actually records the vote

$('.ratings\_stars').bind('click', function() {

var star = this;

var widget = $(this).parent();

var clicked\_data = {

clicked\_on : $(star).attr('class'),

widget\_id : $(star).parent().attr('id')

};

$.post(

'ratings.php',

clicked\_data,

function(INFO) {

widget.data( 'fsr', INFO );

set\_votes(widget);

},

'json'

);

});

});

function set\_votes(widget) {

var avg = $(widget).data('fsr').whole\_avg;

var votes = $(widget).data('fsr').number\_votes;

var exact = $(widget).data('fsr').dec\_avg;

window.console && console.log('and now in set\_votes, it thinks the fsr is ' + $(widget).data('fsr').number\_votes);

$(widget).find('.star\_' + avg).prevAll().andSelf().addClass('ratings\_vote');

$(widget).find('.star\_' + avg).nextAll().removeClass('ratings\_vote');

$(widget).find('.total\_votes').text( votes + ' votes recorded (' + exact + ' rating)' );

}

// END FIRST THING

</script>

<style>

.rate\_widget {

border: 1px solid #CCC;

overflow: visible;

padding: 10px;

position: relative;

width: 180px;

height: 32px;

}

.ratings\_stars {

background: url('star\_empty.png') no-repeat;

float: left;

height: 28px;

padding: 2px;

width: 32px;

}

.ratings\_vote {

background: url('star\_full.png') no-repeat;

}

.ratings\_over {

background: url('star\_highlight.png') no-repeat;

}

.total\_votes {

background: #eaeaea;

top: 58px;

left: 0;

padding: 5px;

position: absolute;

}

.movie\_choice {

font: 10px verdana, sans-serif;

margin: 0 auto 40px auto;

width: 180px;

}

h1 {

text-align: center;

width: 400px;

margin: 20px auto;

}

</style>

</head>

<body>

<h1> Rate the following movies! </h1>

<div class='movie\_choice'>

Rate: Raiders of the Lost Ark

<div id="r1" class="rate\_widget">

<div class="star\_1 ratings\_stars"></div>

<div class="star\_2 ratings\_stars"></div>

<div class="star\_3 ratings\_stars"></div>

<div class="star\_4 ratings\_stars"></div>

<div class="star\_5 ratings\_stars"></div>

<div class="total\_votes">vote data</div>

</div>

</div>

<div class='movie\_choice'>

Rate: The Hunt for Red October

<div id="r2" class="rate\_widget">

<div class="star\_1 ratings\_stars"></div>

<div class="star\_2 ratings\_stars"></div>

<div class="star\_3 ratings\_stars"></div>

<div class="star\_4 ratings\_stars"></div>

<div class="star\_5 ratings\_stars"></div>

<div class="total\_votes">vote data</div>

</div>

</div>

</body>

</html>

***Ratings.php***

<?php

$rating = new ratings($\_POST['widget\_id']);

isset($\_POST['fetch']) ? $rating->get\_ratings() : $rating->vote();

class ratings {

var $data\_file = './ratings.data.txt';

private $widget\_id;

private $data = array();

function \_\_construct($wid) {

$this->widget\_id = $wid;

$all = file\_get\_contents($this->data\_file);

if($all) {

$this->data = unserialize($all);

}

}

public function get\_ratings() {

if($this->data[$this->widget\_id]) {

echo json\_encode($this->data[$this->widget\_id]);

}

else {

$data['widget\_id'] = $this->widget\_id;

$data['number\_votes'] = 0;

$data['total\_points'] = 0;

$data['dec\_avg'] = 0;

$data['whole\_avg'] = 0;

echo json\_encode($data);

}

}

public function vote() {

# Get the value of the vote

preg\_match('/star\_([1-5]{1})/', $\_POST['clicked\_on'], $match);

$vote = $match[1];

$ID = $this->widget\_id;

# Update the record if it exists

if($this->data[$ID]) {

$this->data[$ID]['number\_votes'] += 1;

$this->data[$ID]['total\_points'] += $vote;

}

# Create a new one if it doesn't

else {

$this->data[$ID]['number\_votes'] = 1;

$this->data[$ID]['total\_points'] = $vote;

}

$this->data[$ID]['dec\_avg'] = round( $this->data[$ID]['total\_points'] / $this->data[$ID]['number\_votes'], 1 );

$this->data[$ID]['whole\_avg'] = round( $this->data[$ID]['dec\_avg'] );

file\_put\_contents($this->data\_file, serialize($this->data));

$this->get\_ratings();

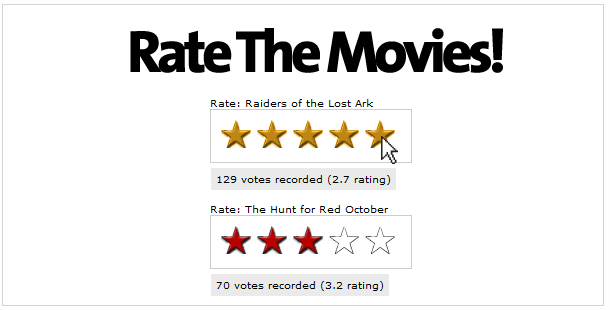
}

# ---

# end class

}

**Output:**

****

**Result:**

Thus the web services for predicting for any product sales has been executed successfully.

**EX.NO:**

**DATE:**

**8. STRUTS WEB FRAMEWORK(XML)**

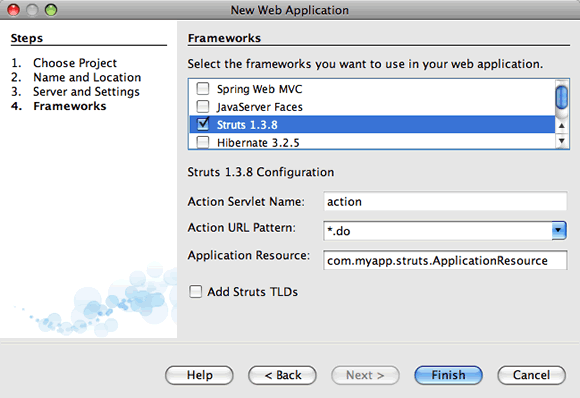
**Aim:**

To build a simple MVC application that displays a login page and returns a success page upon submitting data that passes validation.

**Procedure:**

* [Overview of the Application](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#overview)
* [Setting Up a Struts Application](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#set)
* [Creating JSP Pages](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#jsp)
* [Creating a Login Page](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#login)
* [Creating a Success Page](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#success)
* [Creating an ActionForm Bean](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#actionForm)
* [Creating an Action Class](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#actionClass)
* [Implementing Validation](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#validate)
* [Accessing Bean Data and Preparing a Forwarding Condition](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#beanData)
* [Setting Up an Error Message](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#errorMsg)
* [Adding forward Entries to struts-config.xml](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#forward)
* [Configuring and Running the Application](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#configure)
* [Setting the Welcome Page](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#welcome)
* [Attaching a Stylesheet](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#style)
* [Running the Application](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#run)

**Setting Up a Struts Application**

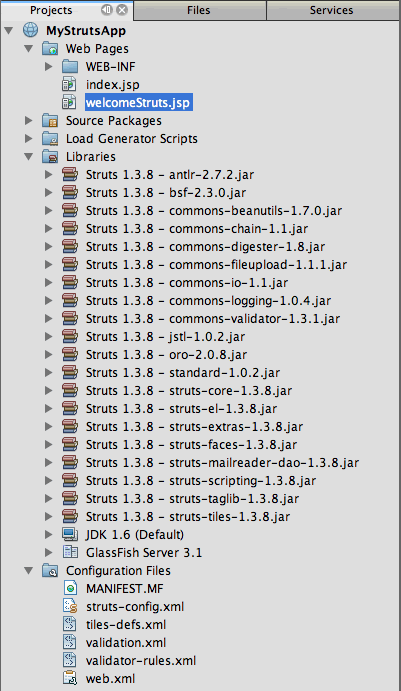
1. Choose File > New Project (Ctrl-Shift-N) from the main menu. Select Java Web in the list of Categories and then select Web Application in the list of Projects. Click Next.
2. In the Name and Location panel, enter MyStrutsApp for Project Name and click Next.
3. In the Server and Settings panel, select the server to which you want to deploy your application. Only servers that are registered with the IDE are listed. (To register a server, click Add next to the Server drop-down list.) Also, note that the Context Path to your deployed application becomes /MyStrutsApp. Click Next.
4. Select Struts in the Frameworks panel.   
   

For purposes of this execution, do not change any of the configuration values in the lower region of this panel. The wizard displays the following configuration options.

* + **Action Servlet Name**: The name of the Struts action servlet used in the application. The web.xml deployment descriptor contains an entry for the action servlet and specifies the appropriate Struts-specific parameters, such as the path to the servlet class within the Struts library and to the struts-config.xmlconfiguration file within the application.
  + **Action URL Pattern**: Specifies the patterns of incoming requests which are mapped to the Struts action controller. This generates a mapping entry in the deployment descriptor. By default, only the \*.do pattern is mapped.
  + **Application Resource**: Lets you specify the resource bundle which will be used in the struts-config.xml file for localizing messages. By default, this iscom.myapp.struts.ApplicationResource.
  + **Add Struts TLDs**: Lets you generate tag library descriptors for the Struts tag libraries. A tag library descriptor is an XML document which contains additional information about the entire tag library as well as each individual tag. In general this is not necessary, because you can refer to on-line URIs rather than local TLD files.

1. Click Finish. The IDE creates the project folder in your file system. As with any web application in the IDE, the project folder contains all of your sources and the IDE's project metadata, such as the Ant build script. However, your web application in addition has all of the Struts libraries on its classpath. Not only are they on the application's classpath, but they are included in the project and will be packaged with it later when you build the project.

The project opens in the IDE. The Projects window is the main entry point to your project sources. It shows a logical view of important project contents. For example, if you expand several nodes within the new project, it may appear as follows:



**Note:**Use the Files window (Window > Files) to see all of your project contents in a directory-based view.

The Struts-specific configuration files, as well as the application's deployment descriptor, are conveniently placed within the Configuration Files folder. Open the deployment descriptor (double-click the web.xml file node to have it display in the Source Editor). In order to handle Struts processing, a mapping is provided for the Struts controller servlet.

<servlet>

<servlet-name>action</servlet-name>

<servlet-class>org.apache.struts.action.ActionServlet</servlet-class>

<init-param>

<param-name>config</param-name>

<param-value>/WEB-INF/struts-config.xml</param-value>

</init-param>

<init-param>

<param-name>debug</param-name>

<param-value>2</param-value>

</init-param>

<init-param>

<param-name>detail</param-name>

<param-value>2</param-value>

</init-param>

<load-on-startup>2</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>action</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

Above, the Struts controller servlet is named action and is defined in the Struts library (org.apache.struts.action.ActionServlet). It is set to handle all requests that satisfy the \*.do mapping. In addition, initialization parameters for the servlet are specified by means of the struts-config.xml file, also contained in the WEB-INFfolder.

**Creating JSP Pages**

Begin by creating two JSP pages for the application. The first displays a form. The second is the view returned when login is successful.

* [**Creating a Login Page**](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#login)
* [**Creating a Success Page**](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#success)

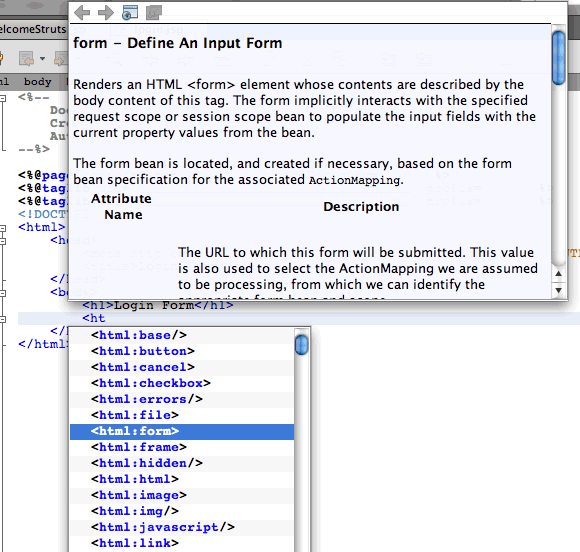
**Creating a Login Page**

1. Right-click the MyStrutsApp project node, choose New > JSP, and name the new file login. Click Finish. The login.jsp file opens in the Source Editor.
2. In the Source Editor, change the content of both the <title> and <h1> tags (or <h2> tags, depending on the IDE version you are using) to Login Form.
3. Add the following two taglib directives to the top of the file:

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>

<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>

Many web applications use JSP pages for views in the MVC paradigm, so Struts provides custom tag libraries which facilitate interaction with HTML forms. These can be easily applied to a JSP file using the IDE's support for code completion. When you type in the Source Editor, the IDE provides you with code completion for Struts tags, as well as the Struts Javadoc. You can also invoke code completion manually by pressing Ctrl-Space:



The [bean taglib](http://struts.apache.org/release/1.3.x/struts-taglib/dev_bean.html) provides you with numerous tags that are helpful when associating a form bean (i.e., an ActionForm bean) with the data collected from the form. The [html taglib](http://struts.apache.org/release/1.3.x/struts-taglib/dev_html.html) offers an interface between the view and other components necessary to a web application. For example, below you replace common html form tags with Struts' <html:form> tags. One benefit this provides is that it causes the server to locate or create a bean object that corresponds to the value provided forhtml:form's action element.

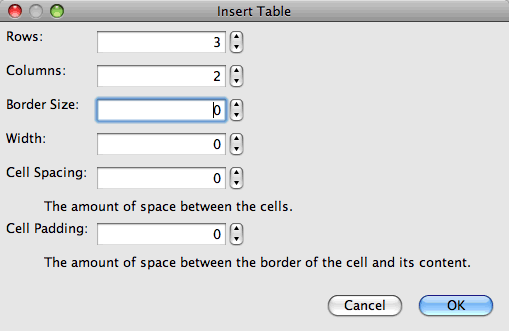
1. Below the <h1> (or <h2>) tags, add the following:

<html:form action="/login">

<html:submit value="Login" />

</html:form>

Whenever you finish typing in the Source Editor, you can tidy up the code by right-clicking and choosing Format (Alt-Shift-F).

1. In the Palette (Window > Palette) in the right region of the IDE, drag a Table item from the HTML category to a point just above the <html:submit value="Login" /> line. The Insert Table dialog box displays. Set the rows to 3, columns to 2, and leave all other settings at 0. Later, you will [attach a stylesheet](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#style) to affect the table display.   
      
   Click OK, then optionally reformat the code (Alt-Shift-F). The form in login.jsp now looks as follows:

<html:form action="/login">

<table border="0">

<thead>

<tr>

<th></th>

<th></th>

</tr>

</thead>

<tbody>

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

<tr>

<td></td>

<td></td>

</tr>

</tbody>

</table>

<html:submit value="Login" />

</html:form>

**Note:**You can safely delete the <thead> table row.

1. In the first table row, enter the following (changes in **bold**):

<tr>

<td>**Enter your name:**</td>

<td>**<html:text property="name" />**</td>

</tr>

1. In the second table row, enter the following (changes in **bold**):

<tr>

<td>**Enter your email:**</td>

<td>**<html:text property="email" />**</td>

</tr>

The html:text element enables you to match the input fields from the form with properties in the form bean that will be created in the next step. So for example, the value of property must match a field declared in the form bean associated with this form.

1. Move the <html:submit value="Login" /> element into the second column of the third table row, so that the third table row appears as follows (changes in **bold**):

<tr>

<td></td>

<td>**<html:submit value="Login" />**</td>

</tr>

At this stage, your login form should look as follows:

<html:form action="/login">

<table border="0">

<tbody>

<tr>

<td>Enter your name:</td>

<td><html:text property="name" /></td>

</tr>

<tr>

<td>Enter your email:</td>

<td><html:text property="email" /></td>

</tr>

<tr>

<td></td>

<td><html:submit value="Login" /></td>

</tr>

</tbody>

</table>

</html:form>

Creating a Success Page

1. Right-click the MyStrutsApp project node, choose New > JSP, and name the new file success. In the Folder field, click the adjacent Browse button and selectWEB-INF from the dialog that displays. Click Select Folder to enter WEB-INF in the Folder field. Any files contained in the WEB-INF folder are not directly accessible to client requests. In order for success.jsp to be properly displayed, it must contain processed data. Click Finish.
2. In the Source Editor, change the content of the newly created page to the following:

<head>

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

<title>Login Success</title>

</head>

<body>

<h1>Congratulations!</h1>

<p>You have successfully logged in.</p>

<p>Your name is: .</p>

<p>Your email address is: .</p>

</body>

1. Add a [bean taglib](http://struts.apache.org/release/1.3.x/struts-taglib/dev_bean.html) directive to the top of the file:

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>

1. Add the following <bean:write> tags (changes in **bold**):

<p>Your name is: **<bean:write name="LoginForm" property="name" />**.</p>

<p>Your email address is: **<bean:write name="LoginForm" property="email" />**.</p>

By employing the <bean:write> tags, you make use of the bean taglib to locate the ActionForm bean you are about to create, and display the user data saved for name and email.

**Creating an ActionForm Bean**

A Struts ActionForm bean is used to persist data between requests. For example, if a user submits a form, the data is temporarily stored in the form bean so that it can either be redisplayed in the form page (if the data is in an invalid format or if login fails) or displayed in a login success page (if data passes validation).

1. Right-click the MyStrutsApp project node and choose New > Other. Under Categories choose Struts, then under File Types choose Struts ActionForm Bean. Click Next.
2. Type in LoginForm for the Class Name. Then select com.myapp.struts in the Package drop-down list and click Finish.

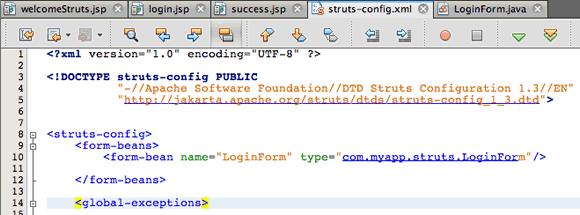
The IDE creates the LoginForm bean and opens it in the Source Editor. By default, the IDE provides it with a String called name and an int called number. Both fields have accessor methods defined for them. Also, the IDE adds a bean declaration to the struts-config.xml file. If you open the struts-config.xml file in the Source Editor, you can see the following declaration, which was added by the wizard:

<form-beans>

**<form-bean name="LoginForm" type="com.myapp.struts.LoginForm" />**

</form-beans>

The IDE provides navigation support in the struts-config.xml file. Hold down the Ctrl key and hover your mouse over the LoginForm bean's fully qualified class name. The name becomes a link, enabling you to navigate directly to the class in the Source Editor:



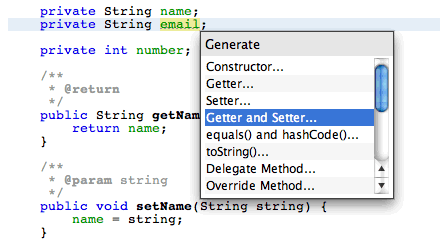
1. In the LoginForm bean in the Source Editor, create fields and accompanying accessor methods that correspond to the name and email text input fields that you created in login.jsp. Because name has already been created in the LoginForm skeleton, you only need to implement email.

Add the following declaration beneath name (changes in **bold**):

private String name;

**private String email;**

To create accessor methods, place your cursor on email and press Alt-Insert.

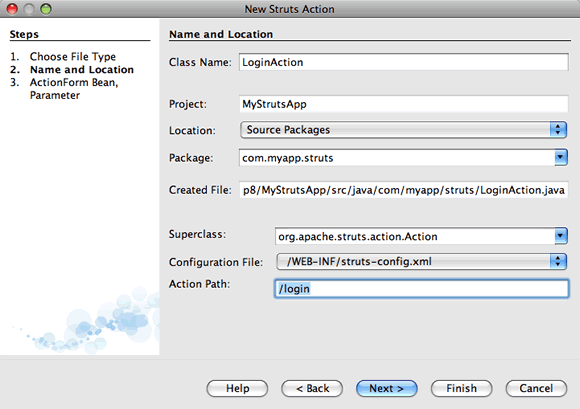


Select Getter and Setter, then in the dialog that displays, select email : String and click Generate. Accessor methods are generated for the email field.

**Note:**You can delete the declaration and accessor methods for number.

**Creating an Action Class**

The Action class contains the business logic in the application. When form data is received, it is the execute method of an Action object that processes the data and determines which view to forward the processed data to. Because the Action class is integral to the Struts framework, NetBeans IDE provides you with a wizard.

1. In the Projects window, right-click the MyStrutsApp project node and choose New > Other. From the Struts category choose Struts Action and click Next.
2. In the Name and Location panel, change the name to LoginAction.
3. Select com.myapp.struts in the Package drop-down list.
4. Type /login in Action Path. This value must match the value you set for the action attribute of the <html:form> tags in login.jsp. Make sure settings appear as in the screenshot below, then click Next.   
   
5. In the third step of the wizard, you are given the opportunity to associate the Action class with a form bean. Notice that the LoginForm bean you previously created is listed as an option for ActionForm Bean Name. Make the following adjustments to the panel:
   * Delete the forward slash for the Input Resource field
   * Set Scope to Request (Session is the default scope setting in Struts.)
   * Deselect the Validate ActionForm Bean option

Click Finish. The LoginAction class is generated, and the file opens in the Source Editor. Also note that the following action entry is added to the struts-config.xml file:

<action-mappings>

**<action name="LoginForm" path="/login" scope="request" type="com.myapp.struts.LoginAction" validate="false"/>**

<action path="/Welcome" forward="/welcomeStruts.jsp"/>

</action-mappings>

The name and scope attributes apply to the form bean that is associated with the action. Specifically, when an incoming request matches /login, the Struts framework automatically instantiates a LoginForm object and populates it with the form data sent in the request. The default value of validate is set to true. This tells the framework to call the validate method of the form bean. You deselected this option in the wizard however because you will hand-code simple validation in the next step, which does not require the validate method.

**Implementing Validation**

In the Source Editor, browse through the LoginAction class and look at the execute method:

public ActionForward execute(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response)

throws Exception {

return mapping.findForward(SUCCESS);

}

Notice the definition of SUCCESS, listed beneath the LoginAction class declaration:

private final static String SUCCESS = "success";

Currently, the mapping.findForward method is set to unconditionally forward any request to an output view called success. This is not really desirable; you want to first perform some sort of validation on the incoming data to determine whether to send the success view, or any different view.

* [**Accessing Bean Data and Preparing a Forwarding Condition**](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#beanData)
* [**Setting Up an Error Message**](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#errorMsg)

**Accessing Bean Data and Preparing a Forwarding Condition**

1. Type in the following code within the body of the execute method:

// extract user data

LoginForm formBean = (LoginForm)form;

String name = formBean.getName();

String email = formBean.getEmail();

In order to use the incoming form data, you need to take execute's ActionForm argument and cast it as LoginForm, then apply the getter methods that you created earlier.

1. Type in the following conditional clause to perform validation on the incoming data:

// perform validation

if ((name == null) || // name parameter does not exist

email == null || // email parameter does not exist

name.equals("") || // name parameter is empty

email.indexOf("@") == -1) { // email lacks '@'

return mapping.findForward(FAILURE);

}

At this stage, the execute method should look as follows:

public ActionForward execute(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response)

throws Exception {

// extract user data

LoginForm formBean = (LoginForm) form;

String name = formBean.getName();

String email = formBean.getEmail();

// perform validation

if ((name == null) || // name parameter does not exist

email == null || // email parameter does not exist

name.equals("") || // name parameter is empty

email.indexOf("@") == -1) { // email lacks '@'

return mapping.findForward(FAILURE);

}

return mapping.findForward(SUCCESS);

}

1. Add a declaration for FAILURE to the LoginAction class (changes in **bold**):

private final static String SUCCESS = "success";

**private final static String FAILURE = "failure";**

Using the above logic, the execute method forwards the request to the success view if the user provides an entry for both name and email fields, and the email entered contains an '@' sign. Otherwise, the failure view is forwarded. As will be demonstrated below in [Adding forward Entries to struts-config.xml](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#forward), you can set thefailure view to point back to the form page, so that the user has another chance to enter data in the correct format.

**Setting Up an Error Message**

If the login form is returned, it would be good to inform the user that validation failed. You can accomplish this by adding an error field in the form bean, and an appropriate<bean:write> tag to the form in login.jsp. Finally, in the Action object, set the error message to be displayed in the event that the failure view is chosen.

1. Open LoginForm and add an error field to the class:

// error message

private String error;

1. Add a getter method and a setter method for error, as [demonstrated above](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#accessors).
2. Modify the setter method so that it appears as follows:

public void setError() {

this.error =

"<span style='color:red'>Please provide valid entries for both fields</span>";

}

1. Open login.jsp and make the following changes:

<html:form action="/login">

<table border="0">

<tbody>

**<tr>**

**<td colspan="2">**

**<bean:write name="LoginForm" property="error" filter="false"/>**

**&nbsp;</td>**

**</tr>**

<tr>

<td>Enter your name:</td>

<td><html:text property="name" /></td>

</tr>

1. In LoginAction, within the if conditional clause, add a statement to set the error message before forwarding the failure condition (changes in **bold**):

if ((name == null) || // name parameter does not exist

email == null || // email parameter does not exist

name.equals("") || // name parameter is empty

email.indexOf("@") == -1) { // email lacks '@'

**formBean.setError();**

return mapping.findForward(FAILURE);

}

Your completed LoginAction class should now appear as follows:

public class LoginAction extends org.apache.struts.action.Action {

private final static String SUCCESS = "success";

private final static String FAILURE = "failure";

public ActionForward execute(ActionMapping mapping, ActionForm form,

HttpServletRequest request, HttpServletResponse response)

throws Exception {

// extract user data

LoginForm formBean = (LoginForm)form;

String name = formBean.getName();

String email = formBean.getEmail();

// perform validation

if ((name == null) || // name parameter does not exist

email == null || // email parameter does not exist

name.equals("") || // name parameter is empty

email.indexOf("@") == -1) { // email lacks '@'

formBean.setError();

return mapping.findForward(FAILURE);

}

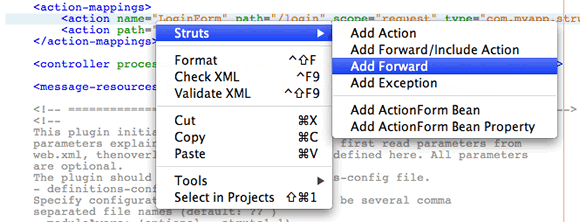
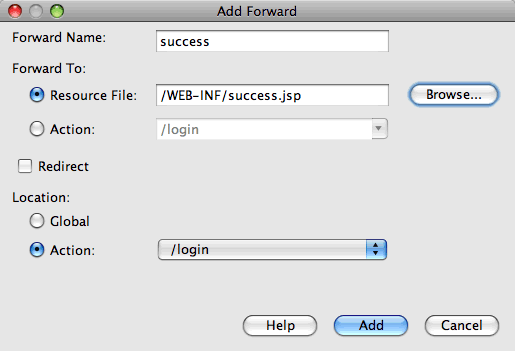
return mapping.findForward(SUCCESS);

}

}

**Adding forward Entries to struts-config.xml**

In order for the application to match JSP pages with forwarding conditions returned by LoginAction's execute method, you need to add forward entries to the struts-config.xml file.

1. Open struts-config.xml in the Source Editor, right-click anywhere in the action entry for LoginForm, and choose Struts > Add Forward.   
   
2. In the Add Forward dialog box, type success in Forward Name. Enter the path to success.jsp in the Resource File field (i.e., /WEB-INF/success.jsp). The dialog box should now look as follows:   
      
   Click Add. Note that the following forward entry was added to struts-config.xml (changes in **bold**):

<action name="LoginForm" path="/login" scope="request" type="com.myapp.struts.LoginAction" validate="false">

**<forward name="success" path="/WEB-INF/success.jsp"/>**

</action>

1. Perform the same action to add a forward entry for failure. Set the Resource File path to /login.jsp. The following forward entry is added to struts-config.xml (changes in **bold**):

<forward name="success" path="/WEB-INF/success.jsp"/>

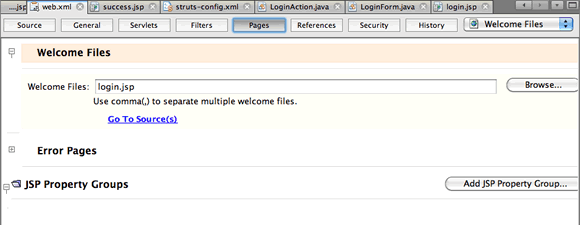
**<forward name="failure" path="/login.jsp"/>**

**Configuring and Running the Application**

The IDE uses an Ant build script to build and run your web application. The IDE generated the build script when you created the project, basing it on the options you entered in the New Project wizard. Before you build and run the application, you need to set the application's default entry point to login.jsp. Optionally, you can also add a simple stylesheet to the project.

* [Setting the Welcome Page](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#welcome)
* [Attaching a Stylesheet](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#style)
* [Running the Application](https://netbeans.org/kb/docs/web/quickstart-webapps-struts.html#run)

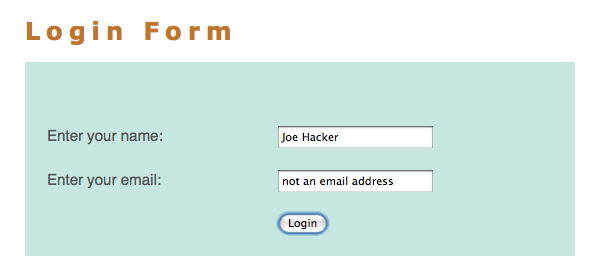
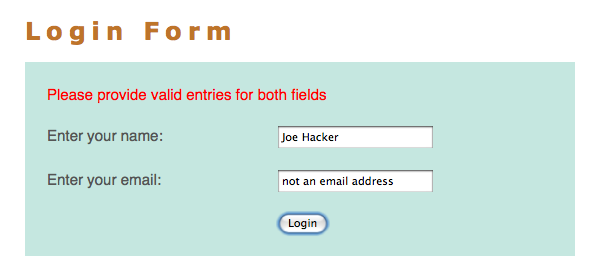
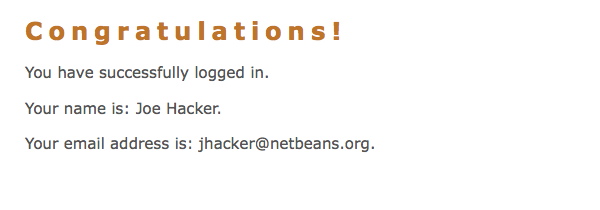
**Setting the Welcome Page**

1. In the Projects window, double-click the web.xml deployment descriptor. The tabs listed along the top of the Source Editor provide you with an interface to the web.xmlfile. Click on the Pages tab. In the Welcome Files field, enter login.jsp.   
      
   Now click on the Source tab to view the file. Note that login.jsp is now listed in the welcome-file entry:
2. <welcome-file>login.jsp</welcome-file>

**Attaching a Stylesheet**

1. Add a simple stylesheet to the project. One easy way to do this is by saving [this sample stylesheet](https://netbeans.org/files/documents/4/2228/stylesheet.css) to your computer. Copy the file (Ctrl-C), then in the IDE, select the Web Pages node in the Projects window and press Ctrl-V). The file is added to your project.
2. Link the stylesheet to your JSP pages by adding a reference between the <head> tags of both login.jsp and success.jsp:
3. <link rel="stylesheet" type="text/css" href="stylesheet.css">

**Running the Application**

1. In the Projects window, right-click the project node and choose Run. The IDE builds the web application and deploys it, using the server you specified when creating the project. The browser opens and displays the login.jsp page. Type in some data that should fail validation, i.e., either leave either field blank, or enter an email address with a missing '@' sign:   
      
     
   When you click Login, the login form page redisplays, containing an error message:   
   
2. Try entering data that should pass validation. Upon clicking Login, you are presented with the success page:   
   

**Result:**

Thus the simple MVC application that displays a login page and returns a success page upon submitting data that passes validation executed successfully.

**EX.NO:**

**DATE:**

**9(A). SEARCHING IN REAL TIME WITH LIVE SEARCHES AND GETTING THE ANSWER WITH AUTO COMPLETE**

**Aim:**

To write AJAX program for Searching in Real Time with Live Searches and getting the answer with auto complete.

**Procedure:**

* Enter a term to search for.
* Ajax contacts Google behind the scenes, looks up words that might match in a dictionary on the server and displays them.
* See a drop-down menu that displays common search terms from Google that might match what you’re typing.
* To select one of those terms, just click it in the menu. That’s all there is to it.
* Ajax application that connects to Google in this way behind the scenes.

**Program:**

***Index.html***

<!DOCTYPE html>

<html lang="en">

<head>

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

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>jQuery Ajax Live Search - DEMO</title>

<!-- Bootstrap core CSS -->

<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">

<!-- Custom styles for this template -->

<link href="starter-template.css" rel="stylesheet">

<!-- HTML5 shim and Respond.js IE8 support of HTML5 elements and media queries -->

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>

<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top" role="navigation">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="sr-only">Toggle navigation</span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

<a class="navbar-brand" href="#">Brand</a>

</div>

<div class="collapse navbar-collapse">

<ul class="nav navbar-nav">

<li class="active"><a href="#">Link 1</a></li>

<li><a href="#">Link 2</a></li>

</ul>

</div><!--/.nav-collapse -->

</div>

</div>

<div class="container">

<div class="starter-template">

<div class="page-header">

<h1>jQuery Ajax Live Search <small>DEMO</small></h1>

</div>

<form role="form" method="post">

<div class="form-group">

<input type="text" class="form-control" id="keyword" placeholder="Enter keyword">

</div>

</form>

<ul id="content"></ul>

</div>

</div><!-- /.container -->

<!-- Bootstrap core JavaScript

================================================== -->

<!-- Placed at the end of the document so the pages load faster -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>

<script src="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/js/bootstrap.min.js"></script>

<script type="text/javascript">

$(document).ready(function() {

$('#keyword').on('input', function() {

var searchKeyword = $(this).val();

if (searchKeyword.length >= 3) {

$.post('search.php', { keywords: searchKeyword }, function(data) {

$('ul#content').empty()

$.each(data, function() {

$('ul#content').append('<li><a href="example.php?id=' + this.id + '">' + this.title + '</a></li>');

});

}, "json");

}

});

});

</script>

</body>

</html>

***Search.php***

<?php

define('DB\_USER', 'ENTER\_USER');

define('DB\_PASSWORD', 'ENTER\_PASSWORD');

define('DB\_SERVER', 'localhost');

define('DB\_NAME', 'ENTER\_DBNAME');

if (!$db = new mysqli(DB\_SERVER, DB\_USER, DB\_PASSWORD, DB\_NAME)) {

die($db->connect\_errno.' - '.$db->connect\_error);

}

$arr = array();

if (!empty($\_POST['keywords'])) {

$keywords = $db->real\_escape\_string($\_POST['keywords']);

$sql = "SELECT ID, post\_title FROM wp\_posts WHERE post\_content LIKE '%".$keywords."%' AND post\_status = 'publish'";

$result = $db->query($sql) or die($mysqli->error);

if ($result->num\_rows > 0) {

while ($obj = $result->fetch\_object()) {

$arr[] = array('id' => $obj->ID, 'title' => $obj->post\_title);

}

}

}

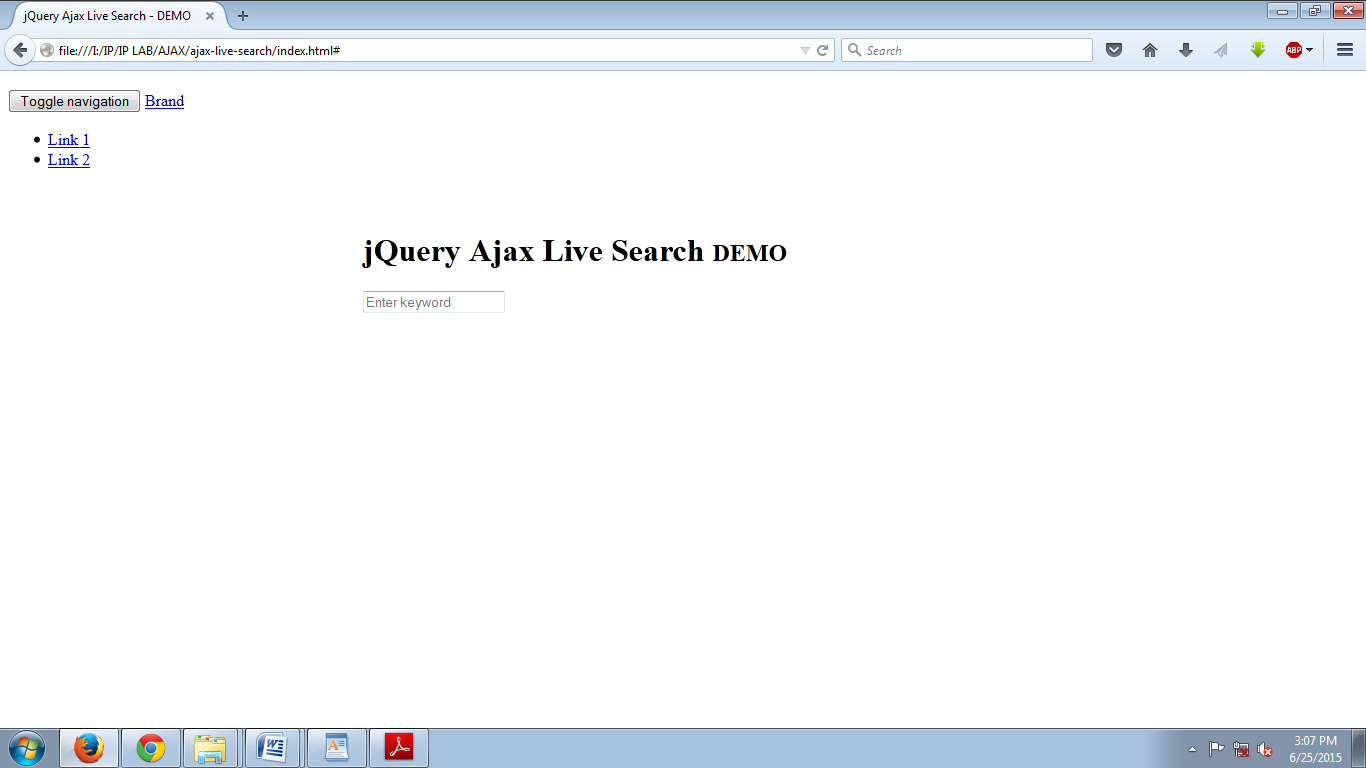
echo json\_encode($arr);

***starter-template.css***

body { padding-top: 10px; }

.starter-template { max-width:640px;margin:0 auto;padding: 25px 15px; }

**Output:**



**Result:**

Thus the AJAX program for Searching in Real Time with Live Searches and getting the answer with auto complete has been executed successfully.

**9(B). AJAX PROGRAM FOR GETTING INSTANT LOGIN FEEDBACK**

**Aim:**

To write Ajax Program for Getting instant login feedback.

**Procedure:**

* Type the wrong login name, get a new page explaining the problem, have to log in on another page.
* Enter an incorrect username and password, and the application says immediately that user name or password Error.

**Program:**

***Login.php***

<?php

if ($\_GET[“qu”] == “steve”){

echo “taken”;

}

else {

echo “ok”;

}

?>

***Login.html***

<body>

<H1>Choose a username</H1>

Enter your new username <input id = “textField” type = “text”

name = “textField” onkeyup = “checkUsername(event)”>

<div id = “targetDiv”><div></div></div>

</body>

function checkUsername(keyEvent)

{

keyEvent = (keyEvent) ? keyEvent: window.event;

input = (keyEvent.target) ? keyEvent.target :

keyEvent.srcElement;

if (keyEvent.type == “keyup”) {

var targetDiv = document.getElementById(“targetDiv”);

targetDiv.innerHTML = “<div></div>”;

if (input.value) {

getData(“login.php?qu=” +

input.value);

}

}

}

function getData(dataSource)

{

if(XMLHttpRequestObject) {

XMLHttpRequestObject.open(“GET”, dataSource);

XMLHttpRequestObject.onreadystatechange = function()

{

if (XMLHttpRequestObject.readyState == 4 &&

XMLHttpRequestObject.status == 200) {

if(XMLHttpRequestObject.responseText == “taken”){

var targetDiv = document.getElementById(“targetDiv”);

targetDiv.innerHTML = “<div>That username is taken.</div>”;

}

}

}

XMLHttpRequestObject.send(null);

}

}

**Output:**

****

**Result:**

Thus the Ajax Program for Getting instant login feedback has been executed successfully.

**EX.NO:**

**DATE:**

**10. WRITE A WEB SERVICES FOR FINDING WHAT PEOPLE THINK BY ASKING 500 PEOPLE’S OPINION FOR ANY CONSUMER PRODUCT**

**Aim:**

To Write a web services for finding what people think by asking 500 people’s opinion for any consumer product

**Procedure:**

* Open the home page.
* Enter the login ID and type the comments then submit.
* Retrieve comments with post id
* Display the comments.

**Program:**

***Index.php***

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<title>jQuery Ajax Comment System - Demo</title>

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

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>

<script src="js/script.js"></script>

</head>

<body>

<div class="wrap">

<h1> Maggy Noodles Comment System</h1>

<?php

// retrive post

include('config.php');

include ('function.php');

dbConnect();

$query = mysql\_query(

'SELECT \*

FROM post

WHERE post\_id = 1');

$row = mysql\_fetch\_array($query);

?>

<div class="post">

<h2><?php echo $row['post\_title']?></h2>

<p><?php echo $row['post\_body']?></p>

</div>

<?php

// retrive comments with post id

$comment\_query = mysql\_query(

"SELECT \*

FROM comment

WHERE post\_id = {$row['post\_id']}

ORDER BY comment\_id DESC

LIMIT 15");

?>

<h2>Comments.....</h2>

<div class="comment-block">

<?php while($comment = mysql\_fetch\_array($comment\_query)): ?>

<div class="comment-item">

<div class="comment-avatar">

<img src="<?php echo avatar($comment['mail']) ?>" alt="avatar">

</div>

<div class="comment-post">

<h3><?php echo $comment['name'] ?> <span>said....</span></h3>

<p><?php echo $comment['comment']?></p>

</div>

</div>

<?php endwhile?>

</div>

<h2>Submit new comment</h2>

<!--comment form -->

<form id="form" method="post">

<!-- need to supply post id with hidden fild -->

<input type="hidden" name="postid" value="<?php echo $row['post\_id']?>">

<label>

<span>Name \*</span>

<input type="text" name="name" id="comment-name" placeholder="Your name here...." required>

</label>

<label>

<span>Email \*</span>

<input type="email" name="mail" id="comment-mail" placeholder="Your mail here...." required>

</label>

<label>

<span>Your comment \*</span>

<textarea name="comment" id="comment" cols="30" rows="10" placeholder="Type your comment here...." required></textarea>

</label>

<input type="submit" id="submit" value="Submit Comment">

</form>

</div>

</body>

</html>

***Ajax\_Comment.php***

<?php

if (isset( $\_SERVER['HTTP\_X\_REQUESTED\_WITH'] )):

include('config.php');

include('function.php');

dbConnect();

if (!empty($\_POST['name']) AND !empty($\_POST['mail']) AND !empty($\_POST['comment']) AND !empty($\_POST['postid'])) {

$name = mysql\_real\_escape\_string($\_POST['name']);

$mail = mysql\_real\_escape\_string($\_POST['mail']);

$comment = mysql\_real\_escape\_string($\_POST['comment']);

$postId = mysql\_real\_escape\_string($\_POST['postid']);

mysql\_query("

INSERT INTO comment

(name, mail, comment, post\_id)

VALUES('{$name}', '{$mail}', '{$comment}', '{$postId}')");

}

?>

<div class="comment-item">

<div class="comment-avatar">

<img src="<?php echo avatar($mail) ?>" alt="avatar">

</div>

<div class="comment-post">

<h3><?php echo $name ?> <span>said....</span></h3>

<p><?php echo $comment?></p>

</div>

</div>

<?php

dbConnect(0);

endif?>

***Config.php***

<?php

# db configuration

define('DB\_HOST', 'localhost');

define('DB\_USER', 'root');

define('DB\_PASS', 'root');

define('DB\_NAME', 'dbname');

?>

***Function.php***

<?php

/\*\*

\* Connect to mysql server

\* @param bool

\* @use true to connect false to close

\*/

function dbConnect($close=true){

if (!$close) {

mysql\_close($link);

return true;

}

$link = mysql\_connect(DB\_HOST, DB\_USER, DB\_PASS) or die('Could not connect to MySQL DB ') . mysql\_error();

if (!mysql\_select\_db(DB\_NAME, $link))

return false;

}

/\*\*

\* gravatar Image

\* @see http://en.gravatar.com/site/implement/images/

\*/

function avatar($mail, $size = 60){

$url = "http://www.gravatar.com/avatar/";

$url .= md5( strtolower( trim( $mail ) ) );

// $url .= "?d=" . urlencode( $default );

$url .= "&s=" . $size;

return $url;

}

?>

***Style.CSS***

/\* general styling \*/

\*{

margin: 0;

padding: 0;

box-sizing: border-box;

-webkit-box-sizing: border-box;

-moz-box-sizing: border-box;

-webkit-font-smoothing: antialiased;

-moz-font-smoothing: antialiased;

-o-font-smoothing: antialiased;

font-smoothing: antialiased;

text-rendering: optimizeLegibility;

}

body{

font: 12px Arial,Tahoma,Helvetica,FreeSans,sans-serif;

text-transform: inherit;

color: #333;

background: #e7edee;

width: 100%;

text-shadow: 0 1px 1px rgba(0, 0, 0, 0.2)

}

.wrap{

width: 720px;

margin: 15px auto;

padding: 15px 20px;

background: white;

border: 2px solid #DBDBDB;

-webkit-border-radius: 5px;

-moz-border-radius: 5px;

border-radius: 5px;

overflow: hidden;

}

a{ text-decoration: none; color: #333}

h1{

font-family: Georgia, "Times New Roman", Times, serif;

font-size: 2.8em;

text-align: center;

margin: 25px 0;

}

h2{font-size: 1.5em; margin: 8px 0}

h3{

font-size: 1.2em;

margin: 5px 0;

}

h3 span{

font-weight: normal;

font-size: 1em;

}

.item{

clear: both;

margin:0;

padding: 10px;

overflow: hidden;

border-top: 1px solid #DBDBDB;

}

.item:last-child{border-bottom:1px solid #DBDBDB}

.item:hover{background: #f9f9f9}

.post{

padding: 10px 0;

border-bottom: 1px solid #E6E6E6;

}

.comment-block{

margin: 20px 0 20px 20px;

}

.comment-item{

overflow: hidden;

width: 500px;

clear: both;

padding: 10px;

border: 1px solid #E6E6E6;

border-radius: 5px;

margin: 5px;

}

.comment-avatar{

width: 60px;

float: left;

}

.comment-avatar img{

width: 60px;

height: 60px;

border-radius: 5px;

}

.comment-post{

width: 400px;

float: left;

padding: 0 5px 0 10px;

}

#form{

clear: both;

margin: 10px;

width: 500px;

}

/\* form styling \*/

input[type="text"],

input[type="email"],

input[type="tel"],

input[type="url"],

textarea {

width:100%;

background: #fff;

border: 1px solid #ddd;

font-size: 13px;

line-height: 20px;

margin: 0;

padding: 7px 10px;

box-shadow: inset 0 1px 2px #eee;

border:1px solid #CCC;

margin:0 0 5px;

border-radius:5px;

}

textarea {

height:100px;

max-width:100%;

}

input[type="submit"] {

cursor:pointer;

width:100%;

border:none;

background:#991D57;

background-image:linear-gradient(bottom, #8C1C50 0%, #991D57 52%);

background-image:-moz-linear-gradient(bottom, #8C1C50 0%, #991D57 52%);

background-image:-webkit-linear-gradient(bottom, #8C1C50 0%, #991D57 52%);

color:#FFF;

margin:0 0 5px;

padding:10px;

border-radius:5px;

}

input[type="submit"]:hover {

background-image:linear-gradient(bottom, #9C215A 0%, #A82767 52%);

background-image:-moz-linear-gradient(bottom, #9C215A 0%, #A82767 52%);

background-image:-webkit-linear-gradient(bottom, #9C215A 0%, #A82767 52%);

-webkit-transition:background 0.3s ease-in-out;

-moz-transition:background 0.3s ease-in-out;

transition:background-color 0.3s ease-in-out;

}

input[type="submit"]:active {

box-shadow:inset 0 1px 3px rgba(0,0,0,0.5);

}

input:focus,

textarea:focus {

outline:0;

border:1px solid #999;

}

label{

display: block;

margin: 5px 0;

font-weight: 900;

cursor: pointer;

}

.alert{

display: none;

padding: 8px 35px 8px 14px;

margin: 20px 0;

text-shadow: 0 1px 0 rgba(255, 255, 255, 0.5);

color: #468847;

background-color: #dff0d8;

border-color: #d6e9c6;

-webkit-border-radius: 4px;

-moz-border-radius: 4px;

border-radius: 4px;

}

***Script.js***

$(document).ready(function(){

var form = $('form');

var submit = $('#submit');

form.on('submit', function(e) {

// prevent default action

e.preventDefault();

// send ajax request

$.ajax({

url: 'ajax\_comment.php',

type: 'POST',

cache: false,

data: form.serialize(), //form serizlize data

beforeSend: function(){

// change submit button value text and disabled it

submit.val('Submitting...').attr('disabled', 'disabled');

},

success: function(data){

// Append with fadeIn see http://stackoverflow.com/a/978731

var item = $(data).hide().fadeIn(800);

$('.comment-block').append(item);

// reset form and button

form.trigger('reset');

submit.val('Submit Comment').removeAttr('disabled');

},

error: function(e){

alert(e);

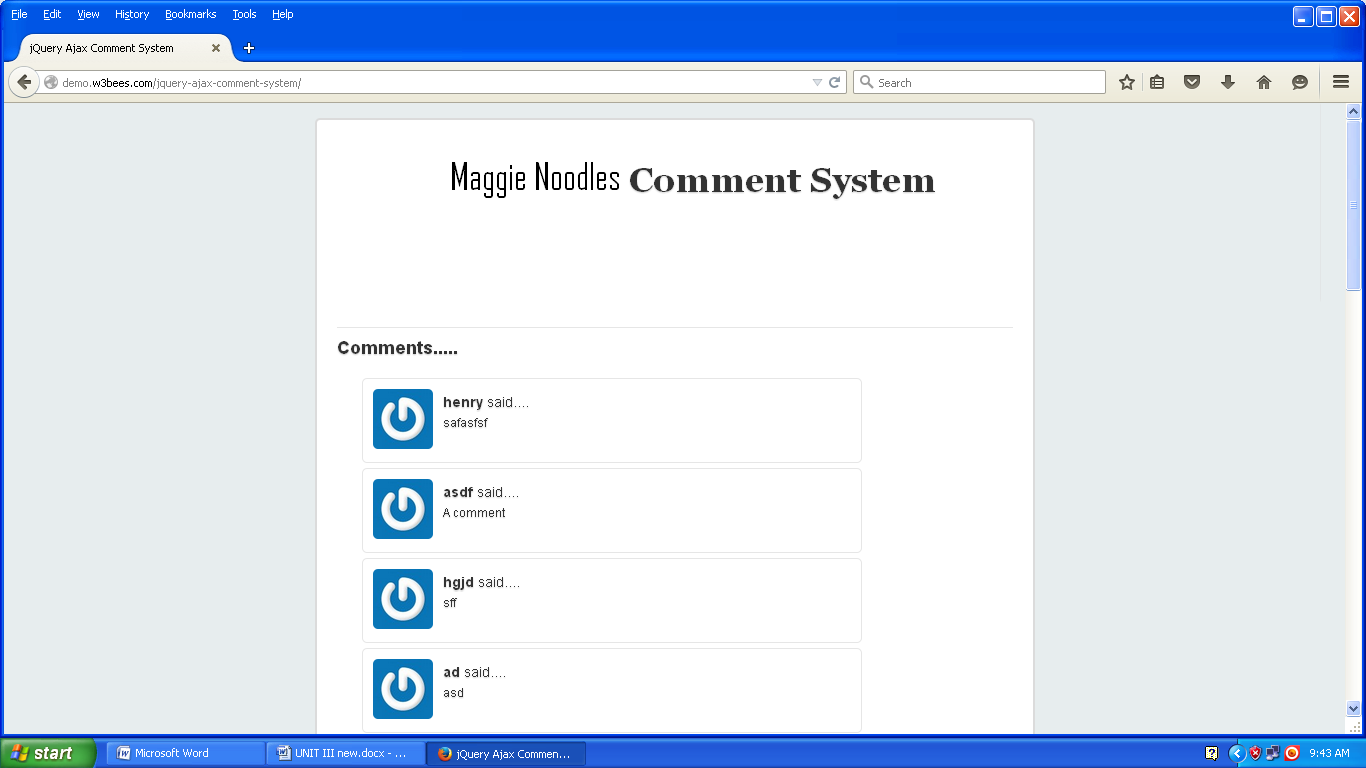
}

});

});

});

**Output:**



**Result:**

Thus a web services for finding what people think by asking 500 people’s opinion for any consumer product has been executed successfully.