MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**1**_____

AIM: Design the static web pages required for an online book store web site.

1) HOME PAGE

DESCRIPTION:

The static home page must contain three **frames**.

- **Top frame** : Logo and the college name and links to Home page, Login page, Registration page,
- **Left frame** : At least four links for navigation, which will display the catalogue of respective links.

For e.g.: When you click the link **"CSE"** the catalogue for **CSE** Books should be displayed in the Right frame.

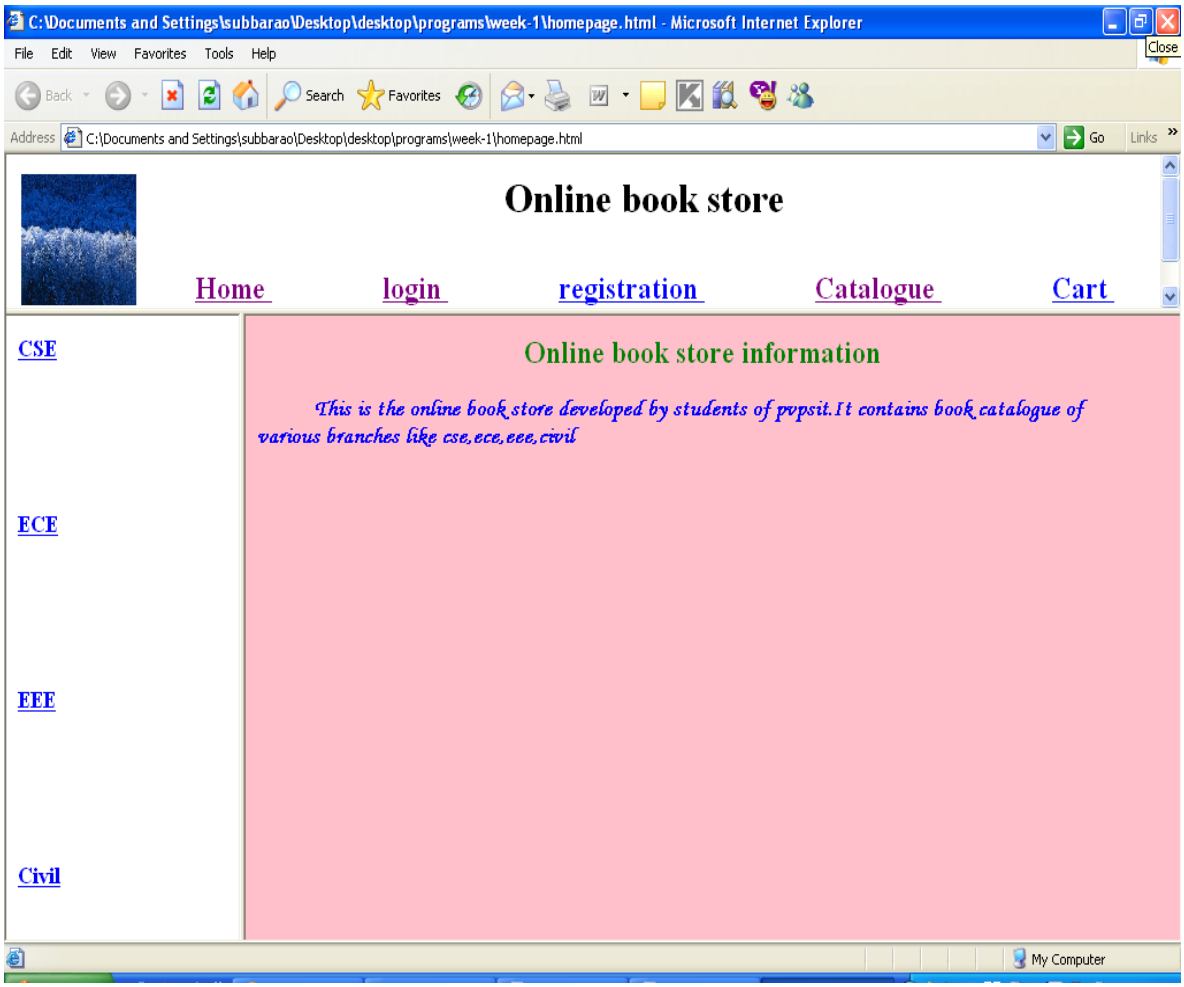
- **Right frame**: The pages to the links in the left frame must be loaded here. Initially this page contains description of the web site.

PROGRAM:

Homepage

```
<head>
 <frameset rows="20%,*">
        <frame src="topframe.html"name="f1">
          <frameset cols="20%,*">
        <frame src="leftframe.html"name="f2">
        <frame src="rightframe.html"name="f3">
           </frameset>
 </frameset>
</head>
```

OUTPUT:

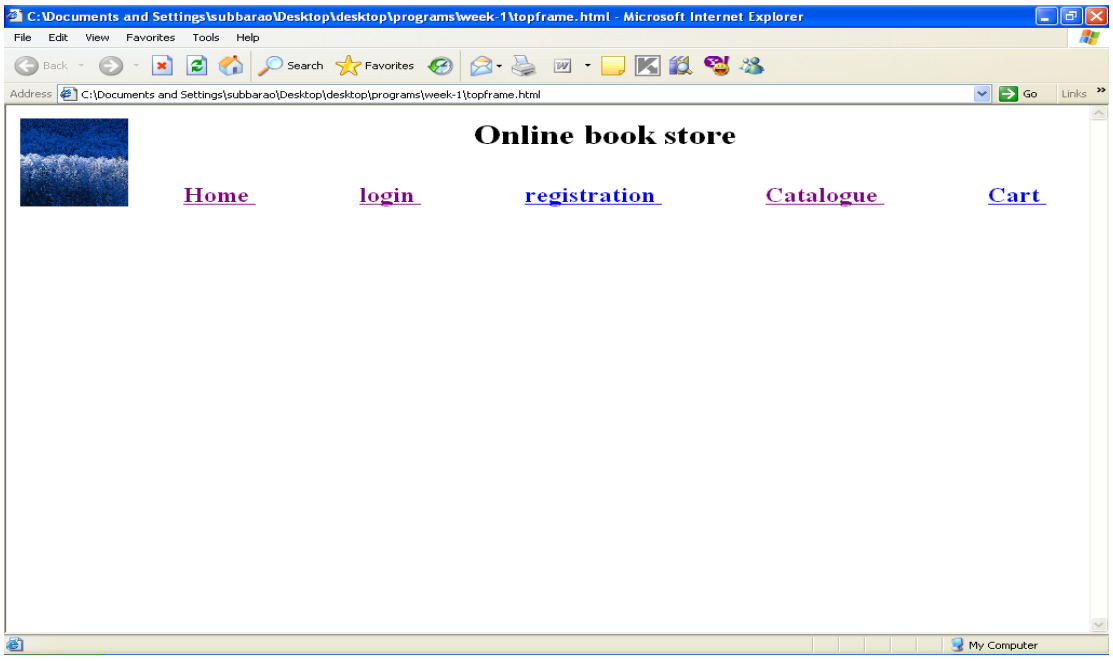


Top frame:

```
<html>
 <body>
  <p>
   <img src="C:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Winter.jpg" align=left width=100 height=100">
    <h1 align=center>Online book store</h1>
  </p>
  <br>
 <p>
   <h2> nbsp;  
  <a href="homepage.html" target=_parent>
    Home
  </a>
```

```
         
<a href="login.html" target="f3">
login
</a>
        
<a href="registration.html" target="f3">
registration
</a>
     nbsp;  
<a href="catalogue.html" target="f3">
Catalogue
</a>
        
<a href="cart.html" target="f3">
Cart
</a>
 
</h2>
</p>
</body>
</html>
```

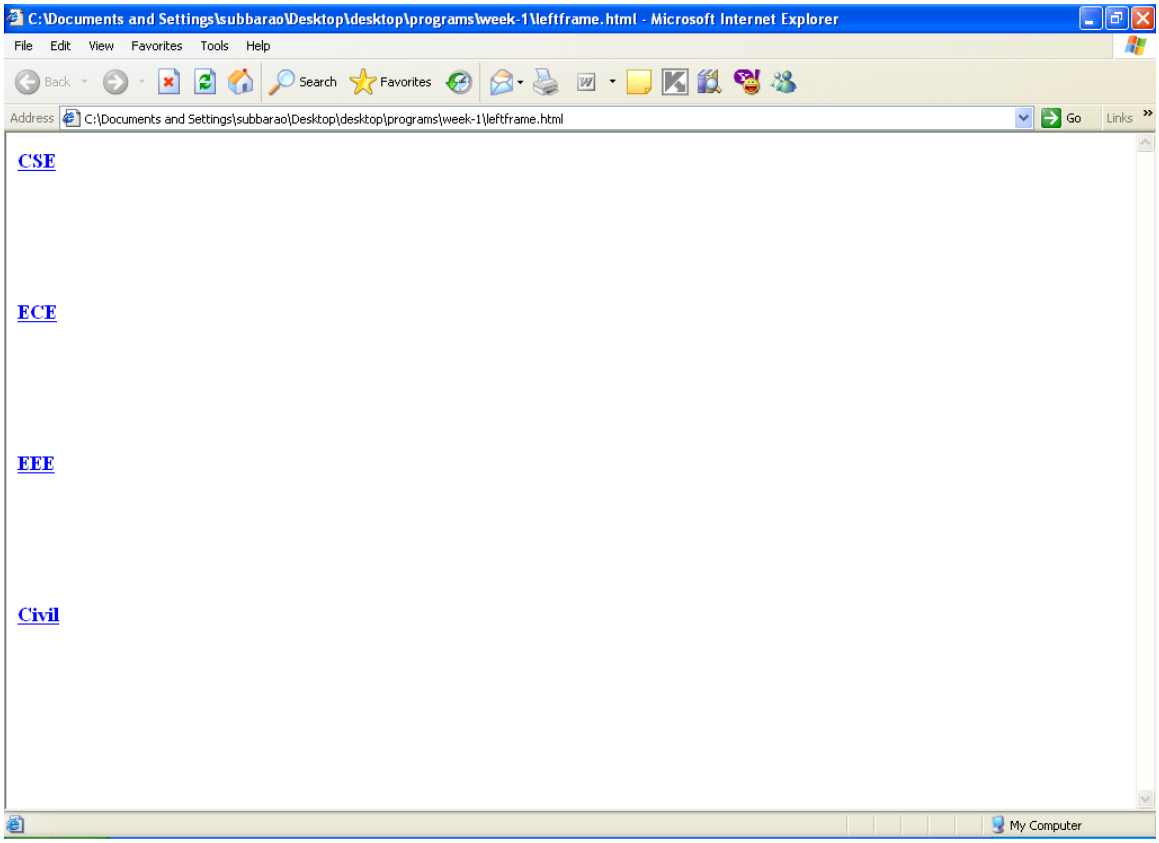
OUTPUT:



Left frame:

```
<html>
  <body>
  <a href=cse.html target="f3"><h3>CSE</h3> </a><br><br><br><br><br>
  <a href=ece.html target="f3"><h3>ECE</h3></a><br><br><br><br><br>
  <a href=eee.html target="f3"><h3>EEE</h3></a><br><br><br><br><br>
  <a href=civil.html target="f3"><h3>Civil</h3></a>
 </body>
</html>
```

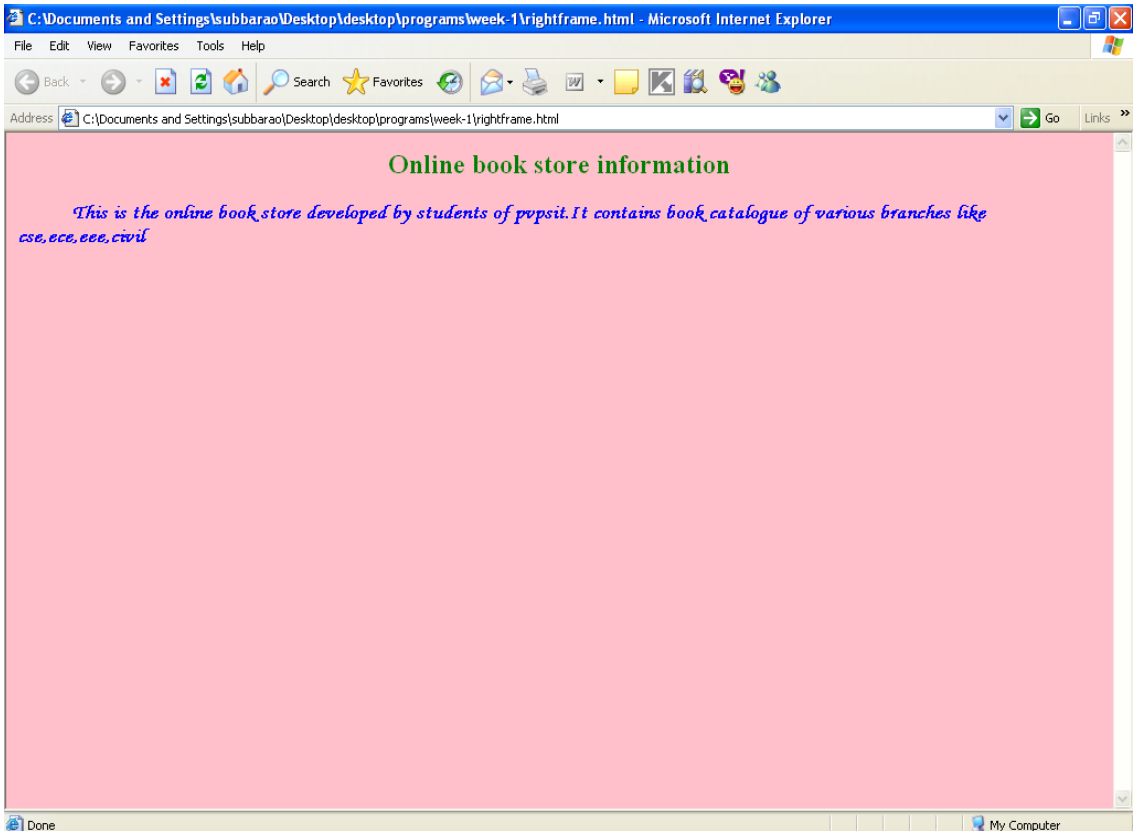
OUTPUT:



Right frame:

```
<html>
 <body bgcolor="pink">
 <p>
  <h2  align="center"> <font face="times new roman" color="green" >Online book store information </font> </h2>

 <h3>           <font face="monotype corsiva" color=blue> This is the online book store developed by students of pvpsit.It contains book catalogue of various branches like cse,ece,eee,civil </font></h3>
 </p>
 </body>
</html>
```

OUTPUT:



2) LOGIN PAGE

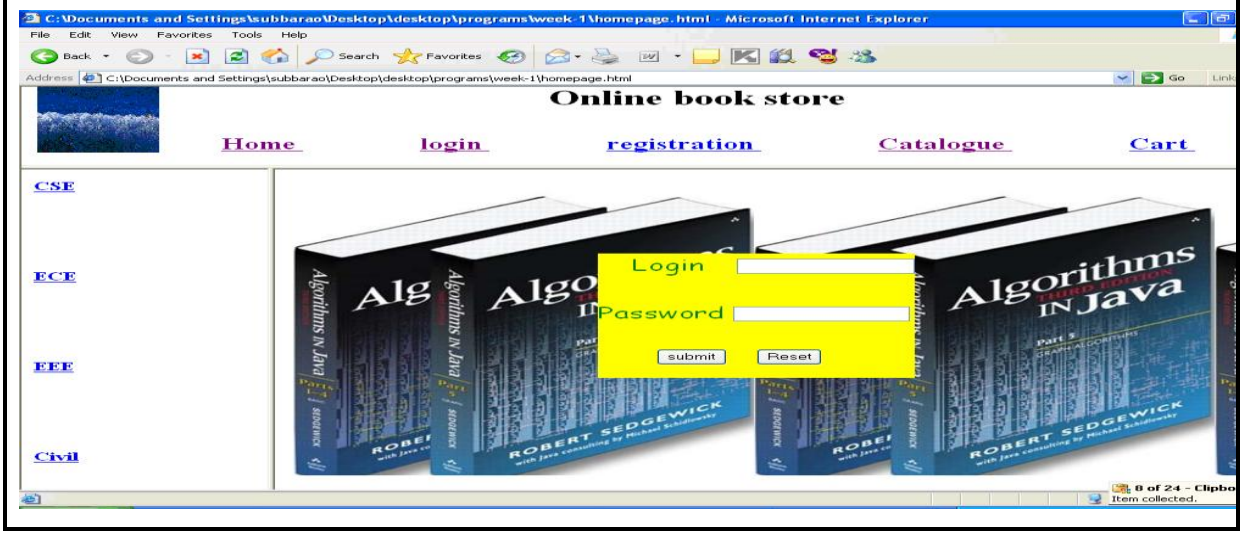
DESCRIPTION:

The login page contains the user name and the password of the user to authenticate.

PROGRAM:

```
<html>
        <head>
          <p style= "background:yellow; top:100px; left:250px; position:absolute; ">
        </head>
        <body background=”E:\1.jpg”>
          <form action="index.jsp" method="get">
          <label>       
              <font face="comic sans ms" color="green" size="5">Login </font>
          </label>     
            <input type="text" size="20" maxlength="20" value=' '><br><br><br>
 <label>
            <font face="comic sans ms" color="green" size="5">Password </font>
        </label>
           <input type="password" size="20" maxlength="20"> <br><br><br>  
         
            <input type="submit" value="submit">     
           <input type="reset">
          </form>
        </body>
</html>
```

OUTPUT:



3) CATOLOGUE PAGE

DESCRIPTION:

The catalogue page should contain the details of all the books available in the web site in a table.

The details should contain the following:

1. Snap shot of Cover Page.
2. Author Name.
3. Publisher.
4. Price.
5. Add to cart button.

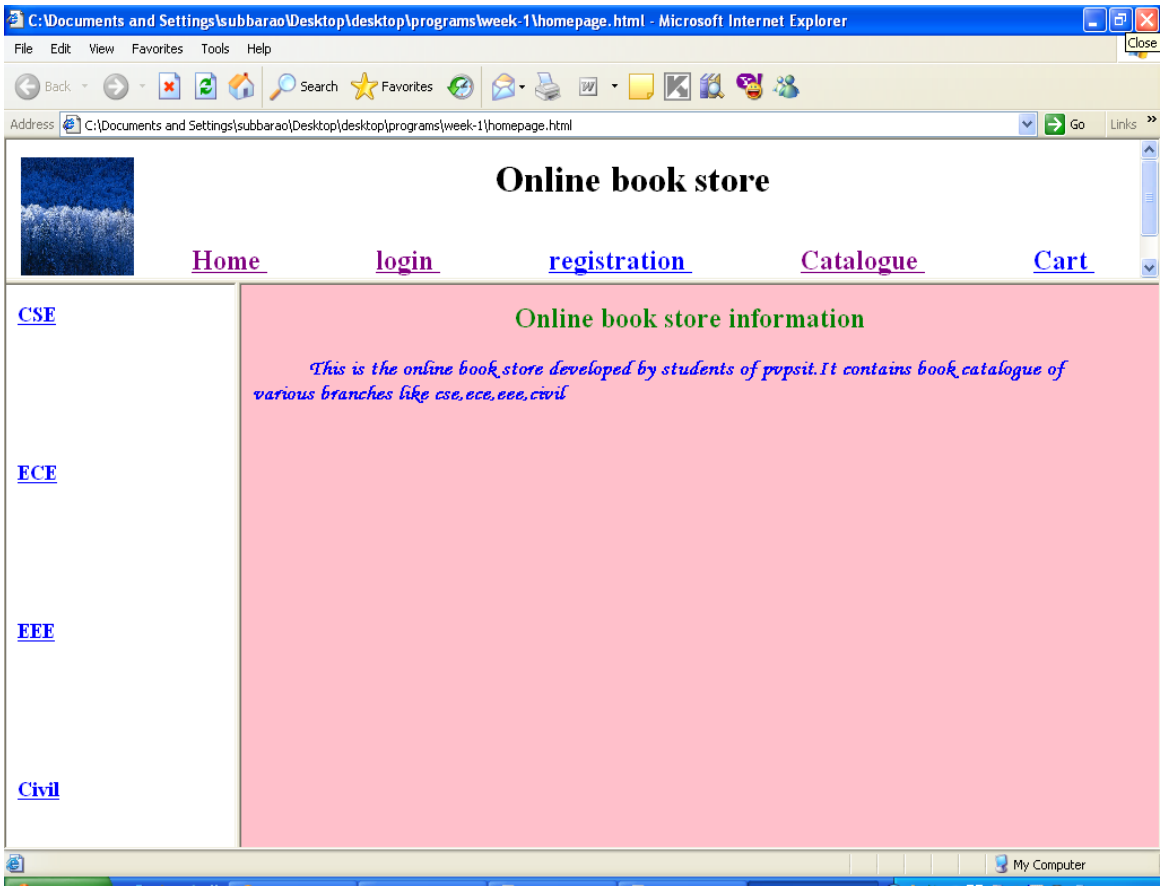
PROGRAM:

```
<html>
<body>
<center>
<table border=1>
<tr>
<th> Book Preview </th>
<th> Book Details </th>
<th> Price </th>
<th> Payment </th>
</tr>
 <tr>
 <td> <img src="C:\Documents and Settings\All users\My Documents\My Pictures\xml.bmp" width=100 height=50>
      </img>
 </td>
 <td>
 <pre>
      <font face="comic sans ms" size=4 color="green" >
      book:XML Bible
      Author:winston
```

```
   Publisher:Wiesley
       </font>
  </pre>
  </td>
<td> $40   </td>
<td>     <a href="cart.html" target="_blank">
     <img src="C:\Documents and Settings\All users\My Documents\My Pictures\cart.bmp" width="150" height="100"></img>
     </a>    
 </td>
</tr>
<tr>
  <td>  <img src="C:\Documents and Settings\All users\My Documents\My Pictures\java.bmp" width=100 height=50>
       </img>
  </td>
<td>
  <pre>
      <font face="comic sans ms" size=4 color="green" >
       book:Java 2
       Author:Watson
       Publisher:BPB publications
      </font>
  </pre>
  </td>
<td>  $40  </td>
<td>     <a href="cart.html" target="_blank">
     <img src="C:\Documents and Settings\All users\My Documents\My Pictures\cart.bmp" width="150" height="100"></img>
     </a>    
 </td>
</tr>
</table> </center>
</body>
</html>
```

OUTPUT:


C:\Documents and Settings\subbarao\Desktop\desktop\programs\week-1\homepage.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Documents and Settings\subbarao\Desktop\desktop\programs\week-1\homepage.html
Go Links
Online book store
Home
login
registration
Catalogue
Cart
CSE
ECE
EEE
Civil
Book Preview
Book Details
Price
Payment
book:XML Bible
Author:winston
Publisher:Wiesley
$40
Add to cart
book:Java 2
Author:Watson
Publisher:BPB publications
$40
Add to cart
file:///C:/Documents%20and%20Settings/subbarao/Desktop/desktop/programs/week-1/catalogue.html
My Computer


C:\Documents and Settings\subbarao\Desktop\desktop\programs\week-1\homepage.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Close
Back Search Favorites
Address C:\Documents and Settings\subbarao\Desktop\desktop\programs\week-1\homepage.html
Go Links
Online book store
Home
login
registration
Catalogue
Cart
CSE
ECE
EEE
Civil
Online book store information
This is the online book store developed by students of pvpsit.It contains book catalogue of various branches like cse,ece,eee,civil
My Computer

RESULT:

Thus the home page, login page, catalogue page for the online book store are created successfully

LAB VIVA QUESTIONS & ANSWERS

1. How can you create an e-mail link?

<mail href="xxx@yyy">

2. How can you open a link in a new browser window?

<a href="url" target="_blank">

3. Who is making the Web standards?

The World Wide Web Consortium

4. What is HTML?

HTML is the Hyper Text Markup Language. It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational elements

5. Define CSS?

Cascading Style Sheets (**CSS**) is a Style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document.

MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **Web Technologies**	Experiment No: **2**

AIM: Design of the cart page and the registration page required for online book store.

4) CART PAGE

DESCRIPTION:

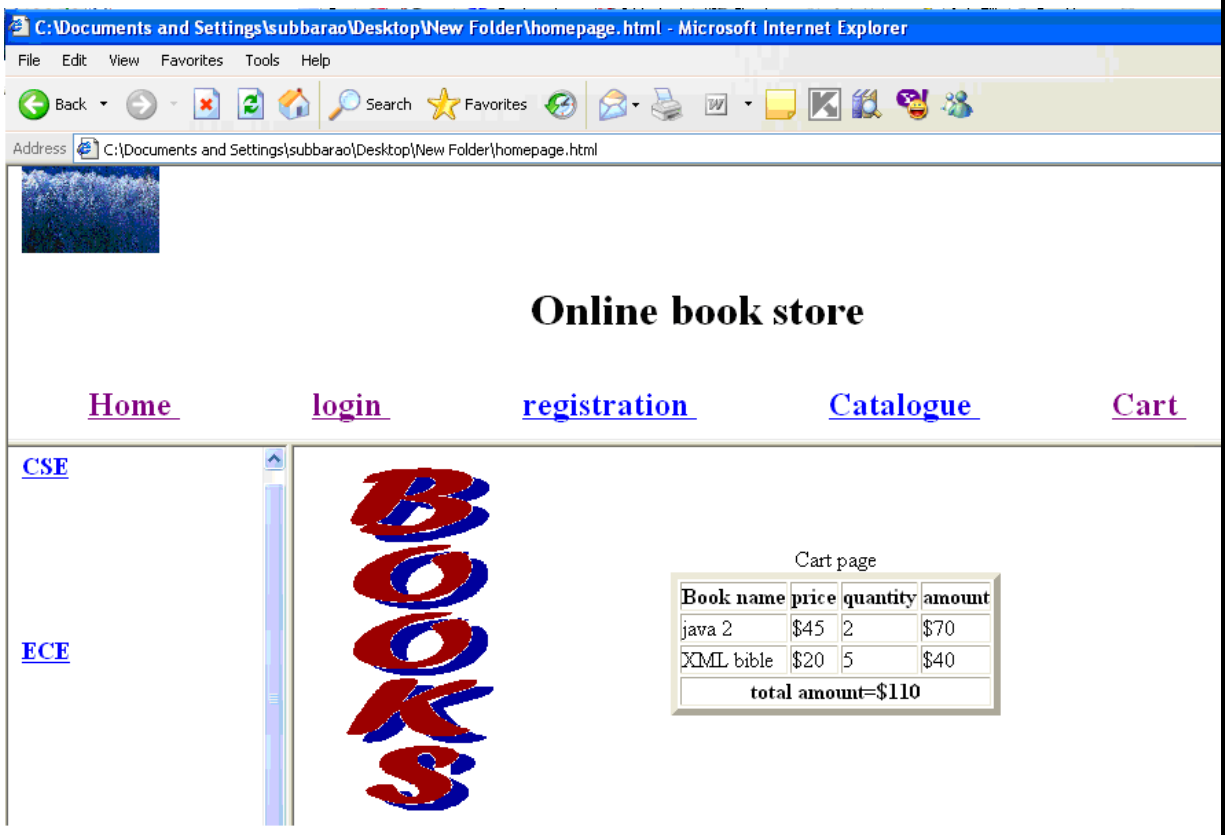
The cart page contains the details about the books which are added to the cart.

PROGRAM:

```
<html>
<body>
<center><br><br><br>
<img src=”E:\aa.jpg”>
<table  border=1 cellpadding=center>
<thead>
<tr>
<th>Book name</th>
<th>price</th>
<th>quantity</th>
<th>amount</th>
</tr>
</thead>
<tr>
<td>java 2</td>
<td>$45</td>
<td>2</td>
<td>$70</td>
</tr>
```

```
<tr>
<td> XML bible</td>
<td> $20</td>
<td> 5</td>
<td> $40</td>
</tr>
<th colspan=4>total amount=$110>
</th>
</tr>
</table>
</center>
</body>
</html>
```

OUTPUT:



5) REGISTRATION PAGE

DESCRIPTION:

Create a "*registration form* "with the following fields

1)Name(Textfield)

2) Password (password field)

3) E-mail id (text field)

4) Phone number (text field)

5) Sex (radio button)

6) Date of birth (3 select boxes)

7) Languages known (check boxes – English, Telugu, Hindi, Tamil)

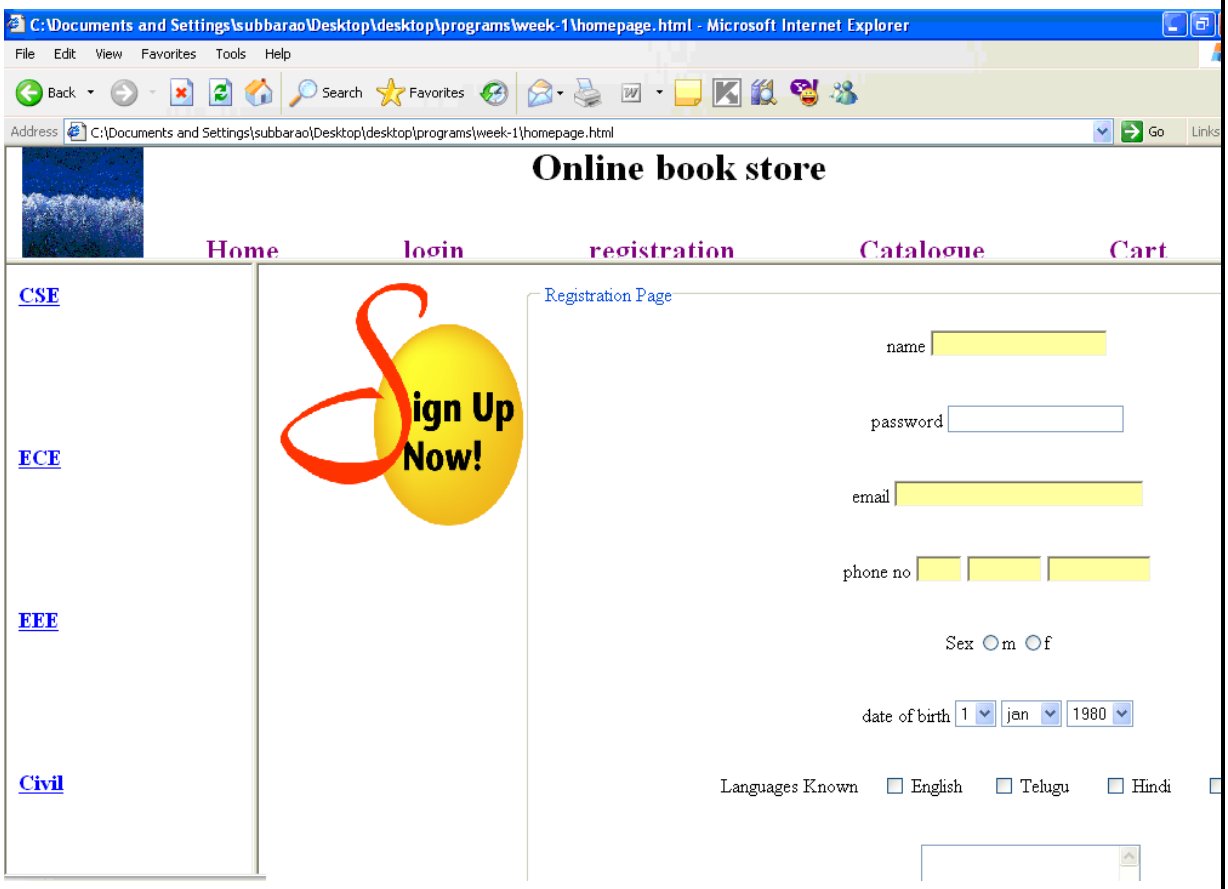
8) Address (text area)

PROGRAM:

```
<html>
<body>
<center>
<img src="E:\aaa.jpg" align="left">
  <form>
<label>name</label>
<input type="text" size="20"><br><br> <br>
<label>password</label>
<input type="password" size="20" maxsize="28"><br> <br> <br>
<label>email</label>
<input type="text" size="30"><br> <br> <br>
<label>phone no</label>
<input type="text" size="2">
<input type="text" size="6">
<input type="text" size="10"><br> <br> <br>
<label>Sex</label>
<input type="radio" name="sex">m
<input type="radio" name="sex">f <br> <br> <br>
<label> date of birth</label>
<select>
  <option>1</option>
```

```
  <option>2</option>
  <option>3</option>
  <option>4</option>
  <option>5</option>
</select>
<select>
   <option>jan</option>
   <option>feb</option>
   <option>mar</option>
   <option>apr</option>
</select>
<select>
   <option>1980</option>
   <option>1981</option>
   <option>1982</option>
   <option>1983</option>
</select> <br> <br> <br>
<label> Languages Known </label>    
<input type="checkbox"> English     
<input type="checkbox"> Telugu      
<input type="checkbox"> Hindi      
<input type="checkbox"> Tamil <br> <br> <br>
<label> Address </label>
<textarea rows=5 cols=20 scrolling="yes"> </textarea>
</center>
</body>
</html>
```

OUTPUT:



RESULT:

Thus the registration and cart pages for online book store pages are created successfully

LAB VIVA QUESTIONS & ANSWERS

1. What is <!DOCTYPE>?

The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in.

**2. What is
 tag?**

The
 tag inserts a single line break. The
 tag is an empty tag which means that it has no end tag.

3. What is <caption> tag?

The <caption> tag defines a table caption. The <caption> tag must be inserted immediately after the <table> tag. We can specify only one caption per table.

4. What is frameset?

The <frameset> tag defines a frameset. The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document. The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

5. Define <PARAM> tag?

The <param> tag is used to define parameters for plugins embedded with an <object> element.

MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **Web Technologies**	Experiment No: 3

AIM: Write *JavaScript* to validate the following fields of the above registration page.

1. Name (Name should contains alphabets and the length should not be less than 6 characters).
2. Password (Password should not be less than 6 characters length).
3. E-mail id (should not contain any invalid and must follow the standard pattern name@domain.com)
4. Phone number (Phone number should contain 10 digits only).

DESCRIPTION:

JavaScript is a simple scripting language invented specifically for use in web browsers to make websites more dynamic. On its own, HTML is capable of outputting more-or-less static pages. Once you load them up your view doesn't change much until you click a link to go to a new page. Adding JavaScript to your code allows you to change how the document looks completely, from changing text, to changing colors, to changing the options available in a drop-down list. JavaScript is a *client-side* language.

JavaScripts are integrated into the browsing environment, which means they can get information about the browser and HTML page, and modify this information, thus changing how things are presented on your screen. This access to information gives JavaScript great power to modify the browsing experience. They can also react to *events*, such as when the user clicks their mouse, or points to a certain page element. This is also a very powerful ability.

Regular Expressions:

One of the most common situations that come up is having an HTML form for users to enter data. Normally, we might be interested in the visitor's name, phone number and email address, and so forth. However, even if we are very careful about putting some hints next to each required field, some visitors are going to get it wrong, either accidentally or for malicious purposes. Here's where regular expressions come in handy. A regular expression is a way of describing a pattern in a piece of text. In fact, it's an easy way of matching a string to a pattern. We could write a simple regular expression and use it to check, quickly, whether or not any given string is a properly formatted user input. This saves us from difficulties and allows us to write clean and tight code.

A regular expression is a JavaScript object. There are multiple ways of creating them. They can be created statically when the script is first parsed or dynamically at run time. A static regular expression is created as follows:

regx=/fish|fow1/;

Dynamic patterns are created using the keyword to create an instance of the RegExp class:

regx=new RegExp("fish|fow1");

Functions:

test(string)- Tests a string for pattern matches. This method returns a Boolean that indicates whether or not the specified pattern exists within the searched string. This is the most commonly used method for validation. It updates some of the properties of the parent RegExp object following a successful search.

exec(string)- Executes a search for a pattern within a string. If the pattern is not found, exec() returns a null value. If it finds one or more matches it returns an array of the match results. It also updates some of the properties of the parent RegExp object

PROGRAM:

style.css

```
<style type="text/css">
body {
background-color:skyblue;
}
.textbox
{
font_family:verdana;
border:px solid black;
color:orange;
}
.fs
{
float:center
border:thin solid #e8e8e8;
color:blue
width:600px;
```

```
}
.lg
{
font-family:verdana;
font-size:20;
color:sea blue;
border-bottom:thin solid red;
}
.button
{
border:thin solid red;
width:100px;
}
.marquee
{
font-family:verdana;
color:red;
font-size:12;
font-weight:bold;
}
.h1
{
float:right;
border-bottom:thin solid green;
font-family:verdana;
color:green;
font-wight:bold;
}
</style>
```

Valid.js

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

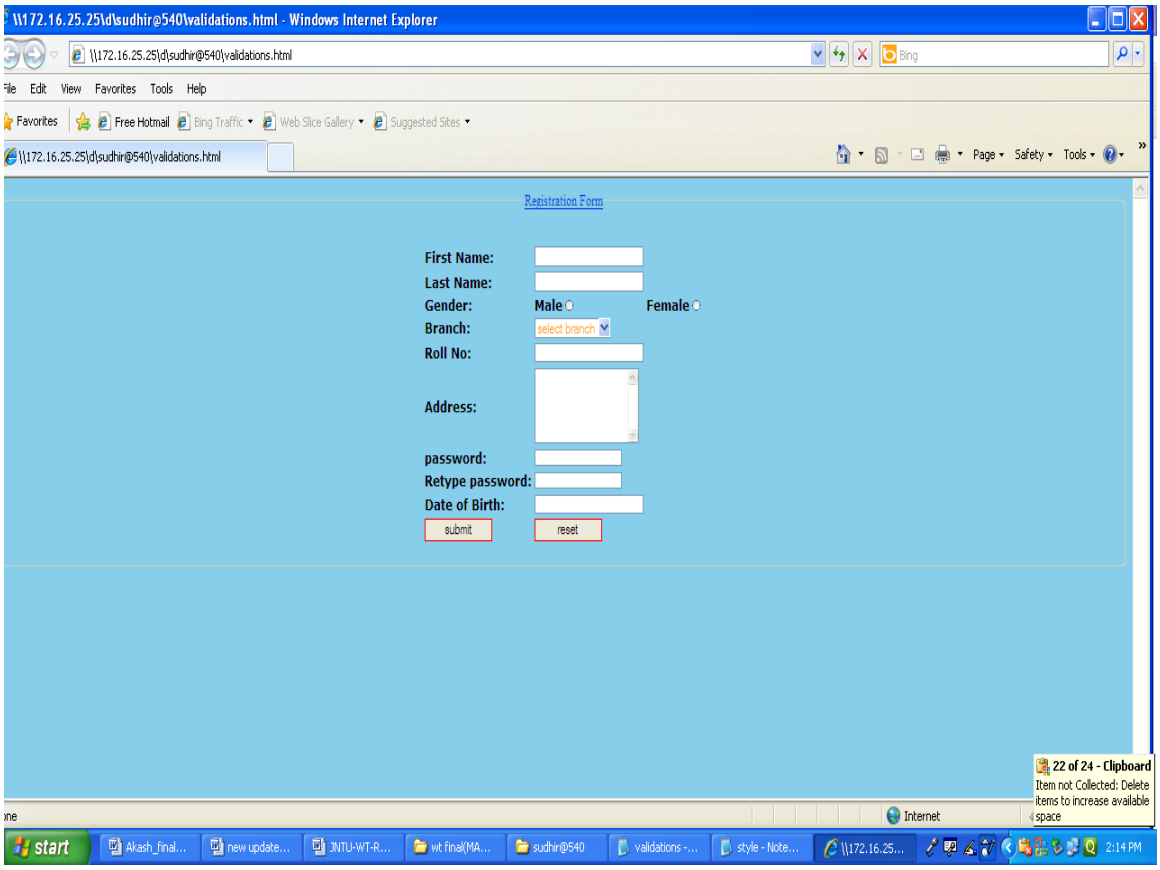
```
body
    {
        background-color:skyblue;
    }
</style>
<script language="javascript">
function fd()
    {
        if (document.fn.fin.value=="")
    {
  alert("please enter your first name");
  document.fn.fin.focus();
  return false;
    }
else
if (document.fn.ln.value=="")
    {
  alert("please enter your last name");
  document.fn.ln.focus();
  return false;
}
else
if(document.fn.gender[0].checked==false&&document.fn.gender[1].checked==false)
   {
        alert("please determine your gender");
            document.fn.gender.focus();
            return false;
    }
else
if(document.fn.branch.value=="")
    {
            alert("please choose your branch");
            document.fn.branch.focus();
            return false;
    }
else
```

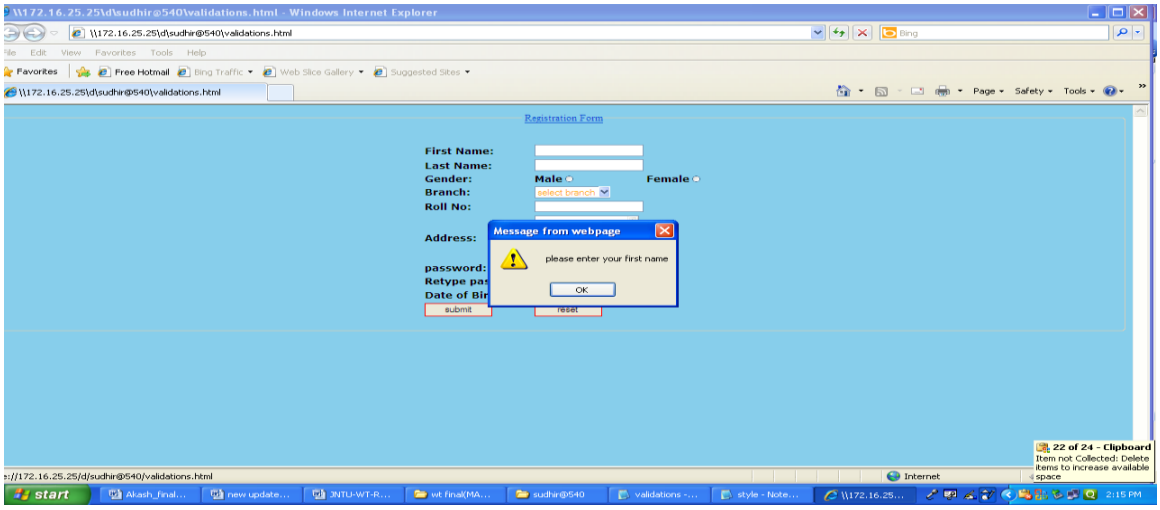
```
if(document.fn.roll.value=="")
    {
            alert("please enter your roll number");
            document.fn.roll.focus();
            return false;
     }
else
if(document.fn.add.value=="")
    {
            alert("please enter your address");
            document.fn.add.focus();
            return false;
    }
else
if(document.fn.pass.value=="")
   {
         alert("please enter your password");
         document.fn.pass.focus();
         return false;
   }
else
if(document.fn.rpass.value=="")
    {
         alert("please enter your retype-password");
         document.fn.rpass.focus();
         return false;
   }
else
if(document.fn.rpass.value!=document.fn.pass.value)
    {
         alert("please check your retype-password");
         document.fn.rpass.focus();
         return false;
     }
else
if(document.fn.dob.value=="")
```

```
    {
        alert("please enter your date of birth");
        document.fn.dob.focus();
        return false;
    }
else
return true;
   }
</script>
</head>
<body>
<fieldset id ="fs">
<legend id="lg"align=center>
<u>Registration Form</u>
</legend>
<font face="verdana">
<form name=fn method=post onsubmit="return fd()">
<table align=center>
<br>
<tr>
<td><b>First Name:</br></td>
<td><input type=text name=fin class="textbox" value=""></td>
</tr>
<tr>
<td><b>Last Name:</b></td>
<td><input type=text name=ln class="textbox" value=""></td>
</tr>
<tr>
<td><b>Gender:</b></td>
<td><b>Male</b><input type=radio name=gender  value=male></td>
<td><b>Female</b><input type=radio name=gender  value=female></td>
</tr>
<tr>
<td><b>Branch:</b></td>
<td><select name=branch class="textbox"><option value=""><b>select branch</b>
</option>
```

```
<option value="cse"><b>CSE</b></option>
<option value="it"><b>IT</b></option>
<option value="eee"><b>EEE</b></option>
<option value="ece"><b>ECE</b></option>
<option value="mech"><b>MECH</b></option>
<option value="civil"><b>CIVIL</b></option></select></td></tr>
<tr>
<td><b>Roll No:</b></td>
<td><input type=text name="roll" class="text box"></td>
</tr>
<tr>
<td><b>Address:</b></td>
<td><textarea rows=5 cols=20 class="textbox" name="add"></textarea></td>
</tr>
<tr>
<td><b>password:</b></td>
<td><input type=password name=pass class="textbox" value=""></td>
</tr>
<tr>
<td><b>Retype password:</b></td>
<td><input type=password name=rpass class="textbox"></td>
</tr>
<tr>
<td><b>Date of Birth:</b></td>
<td><input type=text name=dob class="textbox"></td>
</tr>
<tr>
<td><input type=submit name=submit value=submit class="button"></td>
<td><input type=reset name=reset value=reset class="button"></td>
</tr>
</table>
</form>
</font>
</fieldset>
</body>
</html>
```

OUTPUT:





RESULT:

Thus the home page, login page, catalogue page for the online book store are created successfully

LAB VIVA QUESTIONS & ANSWERS

1. How to define a table and its use?

An HTML table consists of the <table> element and one or more <tr>,<th> and <td> elements. The <tr> element defines a table row, the <th> element defines a table header, and the <td> element defines a table cell. A more complex HTML table may also include <caption>, <col>, <colgroup>, <thead>, <tfoot>, and <tbody> elements.

2. What is use of <thead> tag?

The <thead> tag is used to group header content in an HTML table. The <thead> element is used in conjunction with the <tbody> and <tfoot> elements to specify each part of a table (header, body, footer). Browsers can use these elements to enable scrolling of the table body independently of the header and footer. Also, when printing a large table that spans multiple pages, these elements can enable the table header and footer to be printed at the top and bottom of each page. The <thead> tag must be used in the following context: As a child of a <table> element, after any <caption>, and <colgroup> elements, and before any <tbody>, <tfoot>, and <tr> elements.

3. What is <textarea> tag?

The <textarea> tag defines a multi-line text input control. A text area can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier). The size of a text area can be specified by the cols and rows attributes, or even better; through CSS' height and width properties.

4. What is frameset?

The <frameset> tag defines a frameset. The <frameset> element holds one or more <frame> elements. Each <frame> element can hold a separate document. The <frameset> element specifies HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

5. Define <colgroup> tag?

The <colgroup> tag specifies a group of one or more columns in a table for formatting. The <colgroup> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.

MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **Web Technologies**	Experiment No:_____4_____

AIM: Design a web page using **CSS (C**ascading **S**tyle **S**heets**)** which includes the following:

1) Use different font, styles: In the style definition you define how each selector should work .Then, in the body of your pages, you refer to these selectors to activate the styles.
2) Set a background image for both the page and single elements on the page.
3) Control the repetition of the image with the background-repeat property

DESCRIPTION:

Cascading Style Sheets (**CSS**) is a stylesheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document.

In CSS, *selectors* are used to declare which elements a style applies to, a kind of match expression. Selectors may apply to all elements of a specific type, or only those elements which match a certain attribute; elements may be matched depending on how they are placed relative to each other in the markup code, or on how they are nested within the document object model

A style sheet consists of a list of *rules*. Each rule or rule-set consists of one or more *selectors* and a *declaration block*. A declaration-block consists of a list of semicolon-separated *declarations* in braces. Each declaration itself consists of a *property*, a colon (*:*), a *value*, then a semi-colon (*;*)

Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules:

1. External style sheet
2. Internal style sheet (inside the <head> tag)
3. Inline style (inside an HTML element)

An inline style (inside an HTML element) has the highest priority, which means that it will override a style declared inside the <head> tag, in an external style sheet, or in a browser

(a default value).

Syntax

The CSS syntax is made up of three parts: a selector, a property and a value:

selector {property: value}

The selector is normally the HTML element/tag you wish to define, the property is the attribute you wish to change, and each property can take a value. The property and value are separated by a colon, and surrounded by curly braces:

body {color: black}

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

<head>

<link rel="stylesheet" type="text/css"

href="mystyle.css" />

</head>

The browser will read the style definitions from the file mystyle.css, and format the document according to it.

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag,

<head>

<style>

selector {property:value; property:value;…..}

</style>

</head>

Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly, such as when a style is to be applied to a single occurrence of an element.

To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property.

This is a paragraph

PROGRAM:

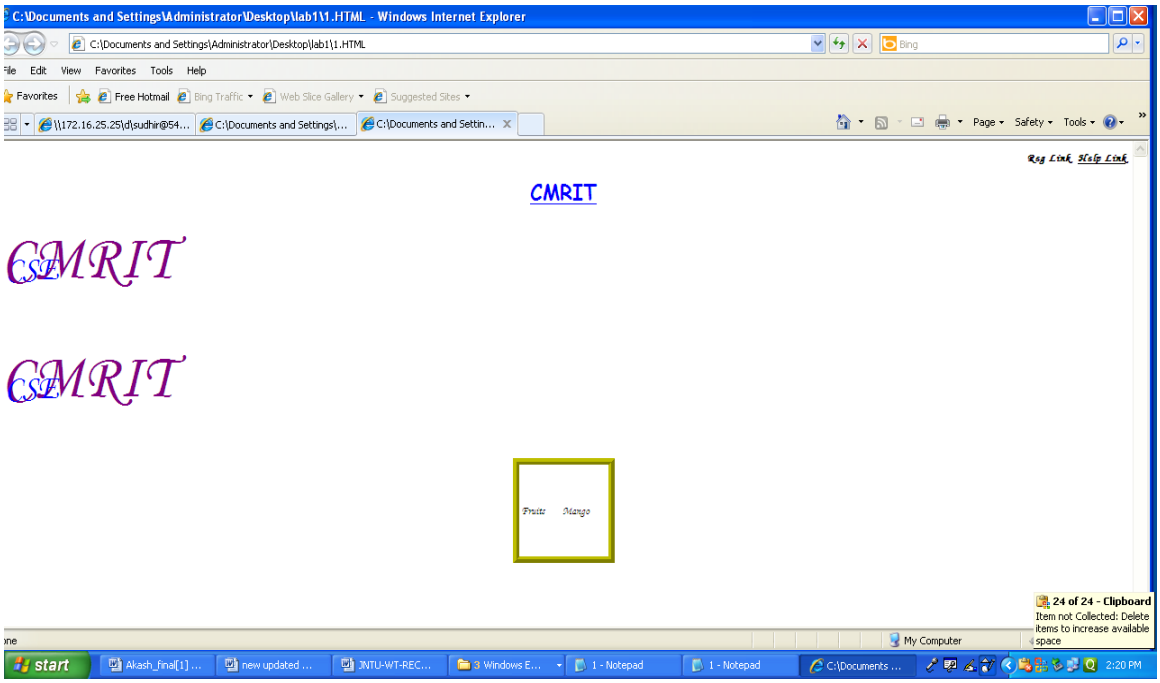
Cas.css:

```
a:link{color:black;}
a:visited{color:pink;}
a:active{color:red;}
a:hover{color:green;}

.right {
        text-align:center;
        text-decoration:underline;
        font-weight:bold;
        color:blue;
        font-family:comic sans ms;
        font-size:30; }
.image {
          text-align:left;
          font-family:"monotype corsiva";
          font-weight:10;
           }

.image1 {
            background-image:url("C:\Documents and Settings\All Users\My Documents\My Pictures\krishna.jpg");
            background-attachment:fixed;
        background-repeat:no-repeat;
            width:150;
            height:150;  }
table { align:center;border:10;
        border-style:ridge;
        border-color:yellow;}
```

htm.html:

```
<html>
<head>
  <link rel="stylesheet" href="cas.css" type="text/css">
  <style>
    .xlink{ text-decoration:none;font-weight:bold;cursor:crosshair;}
    .ylink{text-decoration:underline;font-weight:bold;cursor:help;}
  </style>
</head>
<body class="image">

   <p style="text-align:right;">
     <a href="registration.html" class="xlink"> Reg Link</a>  
     <a href="topframe.html" class="ylink"> Help Link</a>
   </p>
  <p class="right">CMRIT</p>
     <div style="position:relative;font-size:90px;z-index:5;color:purple;">PVPSIT</div>
     <div                              style="position:relative;font-size:50px;z-index:1;top:-70;
left:5;color:blue;">CSE</div>
     <div style="position:relative;font-size:90px;z-index:1;color:purple;">PVPSIT</div>
     <div                              style="position:relative;font-size:50px;z-index:5;top:-70;
left:5;color:blue;">CSE</div>
   <table  align="center" class="image1">
     <tr>
       <td> Fruits</td>
       <td> Mango</td>
     </tr>
  </table>
</body>
</html>
```

OUTPUT:



RESULT: Thus different style of CSS and different type of the properties are applied.

LAB VIVA QUESTIONS & ANSWERS

1.What is the syntax of CSS?

body {color: black}

2. Which HTML tag is used to define an internal style sheet?

<style>

3. What is an External style sheet?

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag.

4. What is an Internal style sheet?

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag.

5. What are the CSS background properties?

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No: **5**

AIM: Write an XML file which will display the Book information.

It includes the following:

1) Title of the book
2) Author Name
3) ISBN number
4) Publisher name
5) Edition
6) Price

Write a Document Type Definition (DTD) to validate the above XML file.

Display the XML file as follows.

The contents should be displayed in a table. The header of the table should be in color GREY. And the Author names column should be displayed in one color and should be capitalized and in bold. Use your own colors for remaining columns.

Use XML schemas XSL and CSS for the above purpose.

DESCRIPTION:

DTD vs XML Schema

The DTD provides a basic grammar for defining an XML Document in terms of the metadata that comprise the shape of the document. An XML Schema provides this, plus a detailed way to define what the data can and cannot contain. It provides far more control for the developer over what is legal, and it provides an Object Oriented approach, with all the benefits this entails.

Many systems interfaces are already defined as a DTD. They are mature definitions, rich and complex. The effort in re-writing the definition may not be worthwhile.

DTD is also established, and examples of common objects defined in a DTD abound on the Internet -- freely available for re-use. A developer may be able to use these to define a DTD more quickly than they would be able to accomplish a complete re-development of the core elements as a new schema.

Finally, you must also consider the fact that the XML Schema is an XML document. It has an XML Namespace to refer to, and an XML DTD to define it. This is all overhead.

When a parser examines the document, it may have to link this all in, interpret the DTD for the Schema, load the namespace, and validate the schema, etc., all *before* it can parse the actual XML document in question. If you're using XML as a protocol between two systems that are in heavy use, and need a quick response, then this overhead may seriously degrade performance.

- Write a Document Type Definition (DTD) to validate the XML file.

PROGRAM:

XML document (bookstore.xml)

```
<bookstore>
	<book>
		<title>web programming</title>
		<author>chrisbates</author>
		<ISBN>123-456-789</ISBN>
		<publisher>wiley</publisher>
		<edition>3</edition>
		<price>350</price>
	</book>
	<book>
		<title>internet worldwideweb</title>
		<author>ditel&amp;amp;ditel</author>
		<ISBN>123-456-781</ISBN>
		<publisher>person</publisher>
		<edition>3</edition>
		<price>450</price>
	</book>
</bookstore>
```

XML document Validation using DTD
DTD document (bookstore.dtd)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT bookstore (book+)>
<!ELEMENT book (title,author,ISBN,publisher,edition,price)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>
```

Bookstore.xml

```
<!DOCTYPE bookstore SYSTEM "C:\Documents and Settings\Administrator\My
Documents\bookstore.dtd">
<bookstore>
	<book>
		<title>web programming</title>
```

```
		<author>chrisbates</author>
		<ISBN>123-456-789</ISBN>
		<publisher>wiley</publisher>
		<edition>3</edition>
		<price>350</price>
	</book>
	<book>
		<title>internet worldwideweb</title>
		<author>ditel&amp;amp;ditel</author>
		<ISBN>123-456-781</ISBN>
		<publisher>person</publisher>
		<edition>3</edition>
		<price>450</price>
	</book>
</bookstore>
```

XML document Validation using DTD
XML Schema (bookstore.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="bookstore">
		<xs:complexType>
		<xs:sequence>
<xs:element name="book" maxOccurs="unbounded">
<xs:complexType>
	<xs:sequence>
<xs:element name="title"	type="xs:string"></xs:element>
<xs:element name="author"	type="xs:string"></xs:element>
<xs:element name="ISBN"	type="xs:string"></xs:element>
<xs:element name="publisher"	type="xs:string"></xs:element>
<xs:element name="edition"	type="xs:int"></xs:element>
<xs:element name="price"	type="xs:decimal"></xs:element>
	</xs:sequence>
</xs:complexType>


</xs:element>

		</xs:sequence>
		</xs:complexType>

	</xs:element>
</xs:schema>
```

Bookstore.xml

```
<bookstore xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="C:\Documents and Settings\Administrator\My
Documents\bookstore.xsd">
	<book>
		<title>web programming</title>
```

```
            <author>chrisbates</author>
            <ISBN>123-456-789</ISBN>
            <publisher>wiley</publisher>
            <edition>3</edition>
            <price>350</price>
        </book>
        <book>
            <title>internet worldwideweb</title>
            <author>ditel&amp;amp;ditel</author>
            <ISBN>123-456-781</ISBN>
            <publisher>person</publisher>
            <edition>3</edition>
            <price>450</price>
        </book>
</bookstore>
```

- **Display the XML file as follows.**

PROGRAM:

XML:

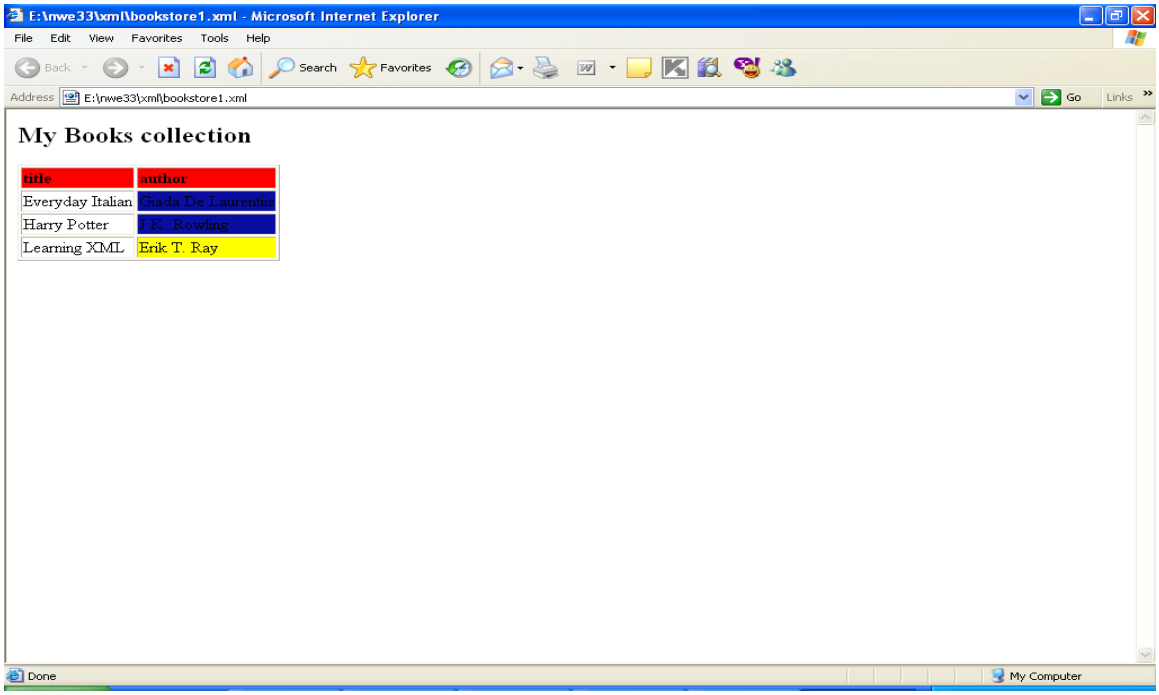
```
</bookstore><?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="bookstore.xsl"?>

<bookstore>
<book>
  <title>Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
<book>
  <title>Learning XML</title>
  <author>Erik T. Ray</author>
```

```
  <year>2003</year>
  <price>39.95</price>
</book>
```

XSL:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2> My Books collection</h2>
<table border="1">
<tr bgcolor="red">
<th align="left">title</th>
<th align="left">author</th>
</tr>
<xsl:for-each select="bookstore/book">
<tr>
<td><xsl:value-of select="title"/></td>
<xsl:choose>
<xsl:when test="price &gt; 30">
<td bgcolor="yellow"><xsl:value-of select="author"/></td>
</xsl:when>
<xsl:when test="price &gt; 10">
<td bgcolor="magneta"><xsl:value-of select="author"/></td>
</xsl:when>
<xsl:otherwise>
<td><xsl:value-of select="author"/></td>
</xsl:otherwise>
</xsl:choose>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

OUTPUT:



RESULT: Thus the XML stylesheets are successfully used to display the content in a table format.

LAB VIVA QUESTIONS & ANSWERS

1. How can XML be used?

XML is used in many aspects of web development, often to simplify data storage and sharing.

2. Define XML?

XML stands for e**X**tensible **M**arkup **L**anguage. XML is designed to transport and store data. XML is important to know, and very easy to learn.

3. How to write the comments in XML?

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment -->

4. What is an XML element?

An XML element is everything from (including) the element's start tag to (including) the element's end tag. An element can contain:

- other elements
- text
- attributes

5. Abbreviate DOM?

Document Object Model.

MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**6**_____

AIM: Create a simple Java bean with a area filled with a color.

The shape of the area depends on the property shape. If it is set to true then the shape of the area is Square and it is Circle, if it is false.

The color of the area should be changed dynamically for every mouse click. The color should also be changed if we change the color in the "property window ".

DESCRIPTION:

A Bean is a JavaBeans component. Beans are independent, reusable software modules. Beans may be visible objects, like AWT components, or invisible objects, like queues and stacks. A builder/integration tool manipulates Beans to create applets and applications.

Beans consist of three things:

- **Events**

An event allows your Beans to communicate when something interesting happens. There are three parts to this communication:

EventObject

Event Listener - (the sink)

An Event Source (the Bean)

The event source defines when and where an event will happen. Classes register themselves as interested in the event, and they receive notification when the event happens. A series of methods patterns represents the registration process:

public synchronized void addListenerType(ListenerType l);

public synchronized void removeListenerType(ListenerType l);

- **Properties**

Properties define the characteristics of the Bean. For instance, when examining an AWT TextField for its properties, you will see properties for the caret position, current text, and the echo character, among others. A property is a public attribute of the Bean, usually represented by a non-public instance variable. It can be read-write, read-only, or write-only. There are four different types of properties:

- **Simple -** As the name implies, simple properties represent the simplest of the four. To

create a property, define a pair of set/get routines. Whatever name used in the pair of routines, becomes the property name

- **Indexed -** An indexed property is for when a single property can hold an array of values. The design pattern for these properties is:

  public void setPropertyName (PropertyType[] list)

  public void setPropertyName (

  PropertyType element, int position)

  public PropertyType[] getPropertyName ()

  public PropertyType getPropertyName (int position)

- **Bound –** A bean that has the bound property generates an event when the property is changed. The event is of type *propertyChangeEvent* and is sent to objects that previously registered an interest in receiveing such notifications. In order for the notification to happen, you need to maintain a watch list for PropertyChangeEvents via the PropertyChangeSupport class. First, you have to create a list of listeners to maintain:

  private PropertyChangeSupport changes =

  new PropertyChangeSupport (this);

  And then, you have to maintain the list:

  public void addPropertyChangeListener (

  PropertyChangeListener p) {

  changes.addPropertyChangeListener (p);

  }

  public void removePropertyChangeListener (

  PropertyChangeListener p) {

  changes.removePropertyChangeListener (p);

  }

- **Constrained -** Constrained properties are similar to bound properties. In addition to maintaining a list of PropertyChangeListeners, the Bean maintains a list of VetoableChangeListeners. Then, prior to the Bean changing a property value, it asks the VetoableChangeListeners if its okay. If it isn't, the listener throws a PropertyVetoException, which you declare the set routine to throw.

➢ **Methods**

Bean methods are available for anyone to call by just making each public. However, you can restrict which methods are visible to the Bean builder/integration tool by providing a getMethodDescriptors method along with your Bean's BeanInfo. Every Bean can provide a supporting BeanInfo class to customize a Bean's appearance to an integration tool.

Procedural Steps to create a Java-Bean:

1) Creating a directory- Create a new directory in C:\beans\demo\sunw\demo with a new folder name colors
2) Create a java source file
3) Compile the java source file
4) Create a manifest file colors.mft in the directory called as C:\beans\demo
5) Create a jar file- to create a jar file type the following command in the command prompt

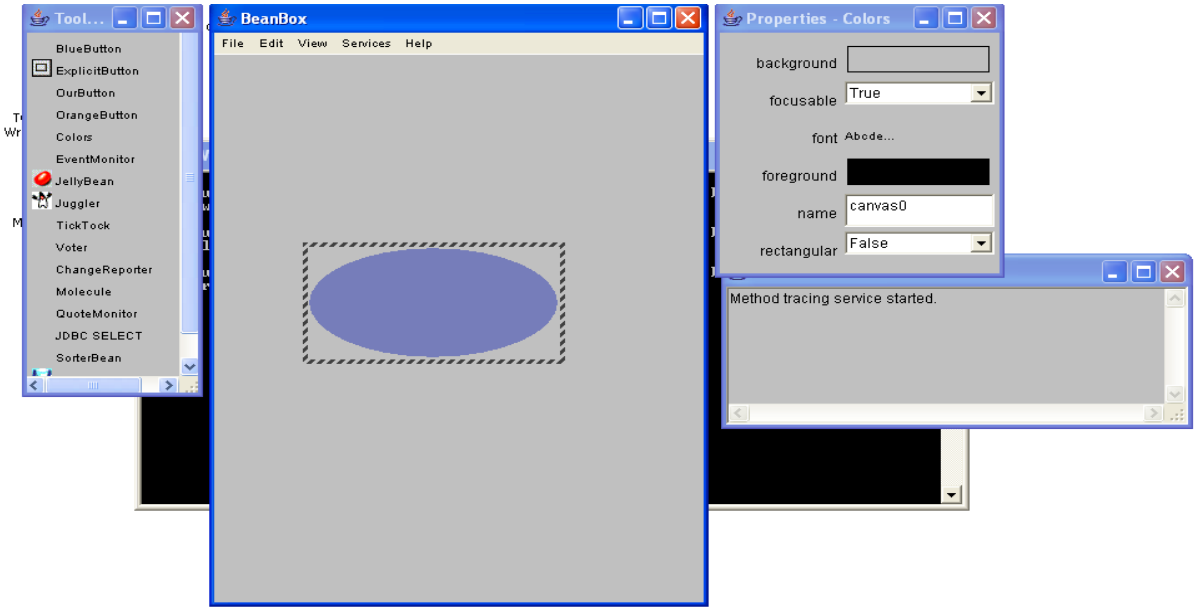
   jar cfm ..\jars\colors.jar colors.mft sunw\demo\colors*.class
6) Start the BDK
7) Check whether the colors bean is placed in toolbox or not.

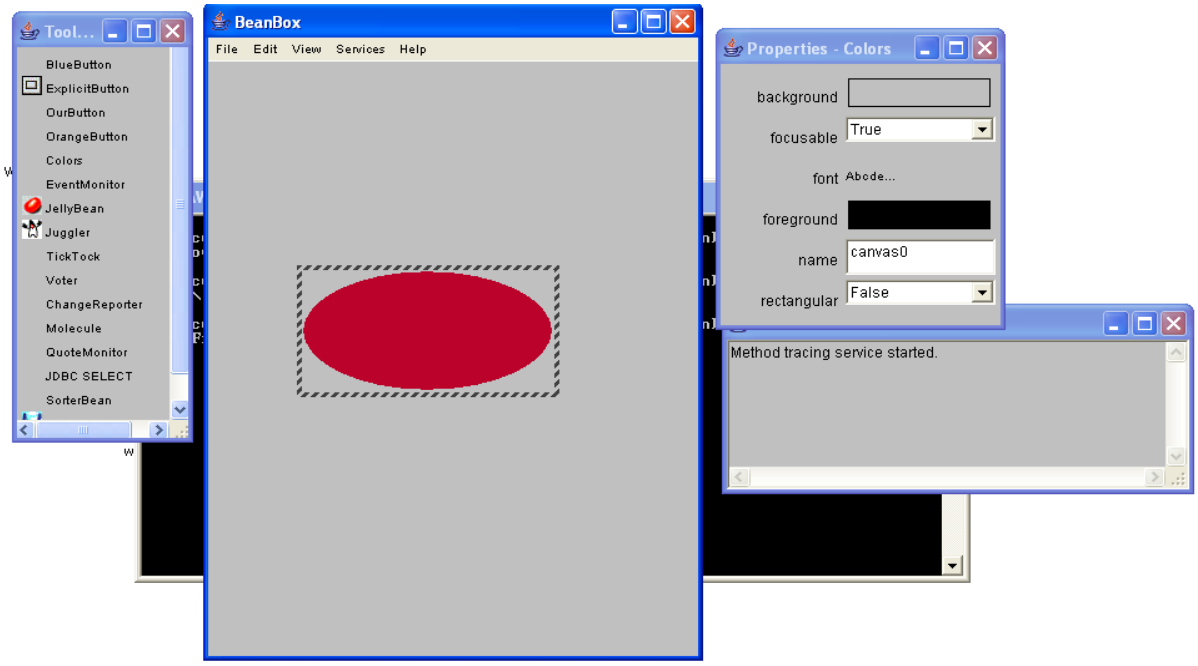
PROGRAM:

```
package sunw.demo.colors;
import java.awt.*;
import java.awt.event.*;
public class Colors extends Canvas
{
 transient private Color color;
private boolean rectangular;
public Colors()
{
 addMouseListener(new MouseAdapter(){
public void mousePressed(MouseEvent me){
change(); }
});
rectangular=false;
setSize(100,100);
change();
}
public boolean getRectangular()
{
 return rectangular;
}
 public void setRectangular(boolean flag)
```

```
{
 this.rectangular=flag;
repaint();
}
public void change()
{
 color=randomColor();
repaint();
}
 private Color randomColor()
{
 int r=(int)(255*Math.random());
 int g=(int)(255*Math.random());
 int b=(int)(255*Math.random());
return new Color(r,g,b);
}
public void paint(Graphics g)
{
Dimension d=getSize();
int h=d.height;
int w=d.width;
g.setColor(color);
if(rectangular)
{
g.fillRect(0,0,w-1,h-1);
}
else
{
g.fillOval(0,0,w-1,h-1);
}
}
}
```

OUTPUT:





RESULT:

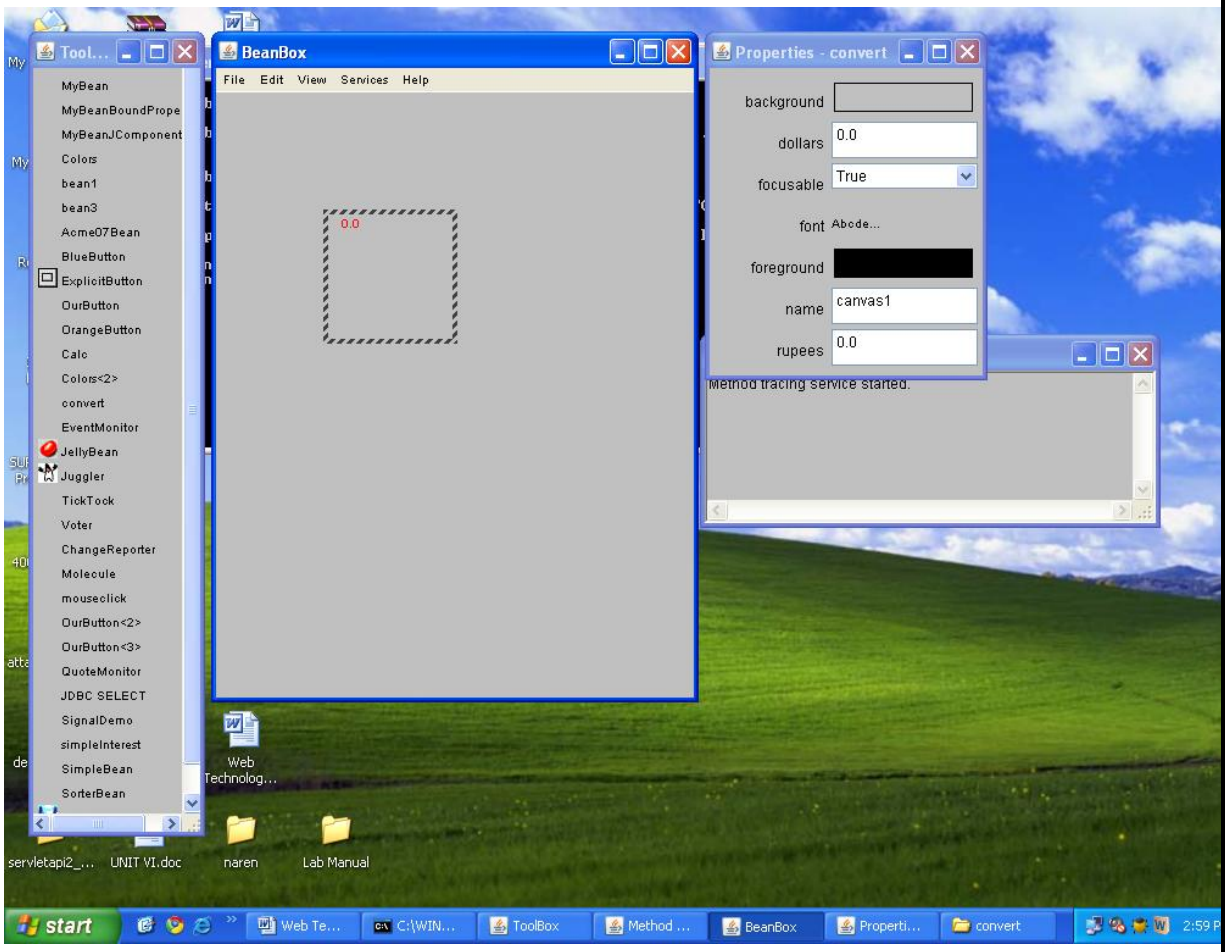
Thus the colors bean is created successfully.

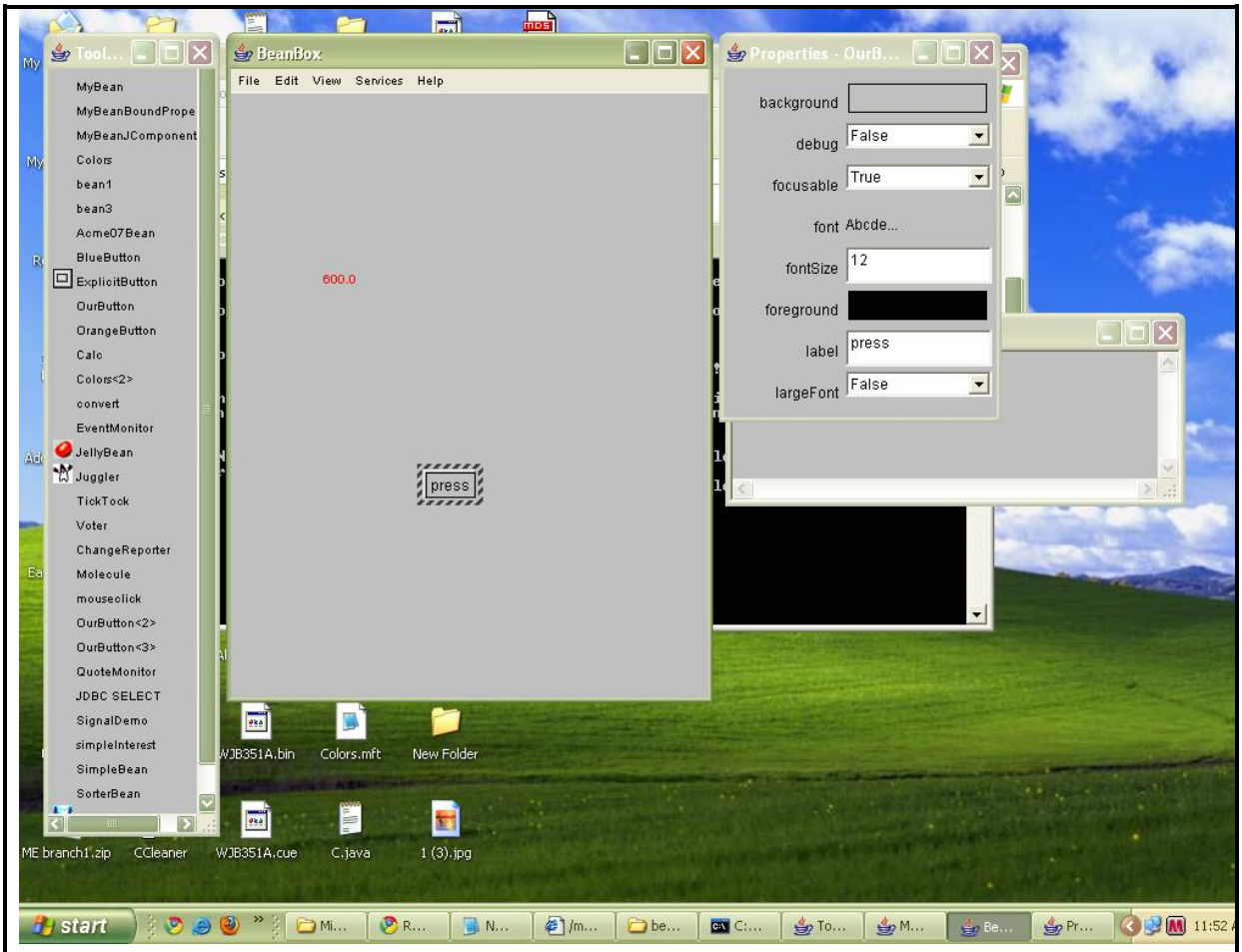
Program 2:

Visual Beans (program 2)
Convert.java

```
package sunw.demo.convert;
import java.awt.*;
import java.awt.event.*;
public class convert extends Canvas
{
        private double dollars=0.0;
        private double rupees=0.0;
        private double dollarvalue=0.0;

        public convert()
        {
        setSize(100,1000);
        }
        public double getDollars()
        {
        return dollars;
        }
        public void setDollars(double value)
        {
        this.dollars=value;
        }
        public void setRupees(double value)
        {
        this.rupees=value;
        }
        public double getRupees()
        {
        return rupees;
        }
        public void change()
        {
         dollarvalue= value();
        repaint();
        }
        private double value()
        {
        return rupees*dollars;
        }
        public void paint(Graphics g)
        {
        g.setColor(Color.red);
        g.drawString(String.valueOf(dollarvalue),10,10);
        }
}
```

Convert.mf
Name: sunw/demo/convert/convert.class
Java-Bean: True (press Enter)
(Carriage return compulsory)





Result:

Thus the convertion bean is created successfully

Program 3:

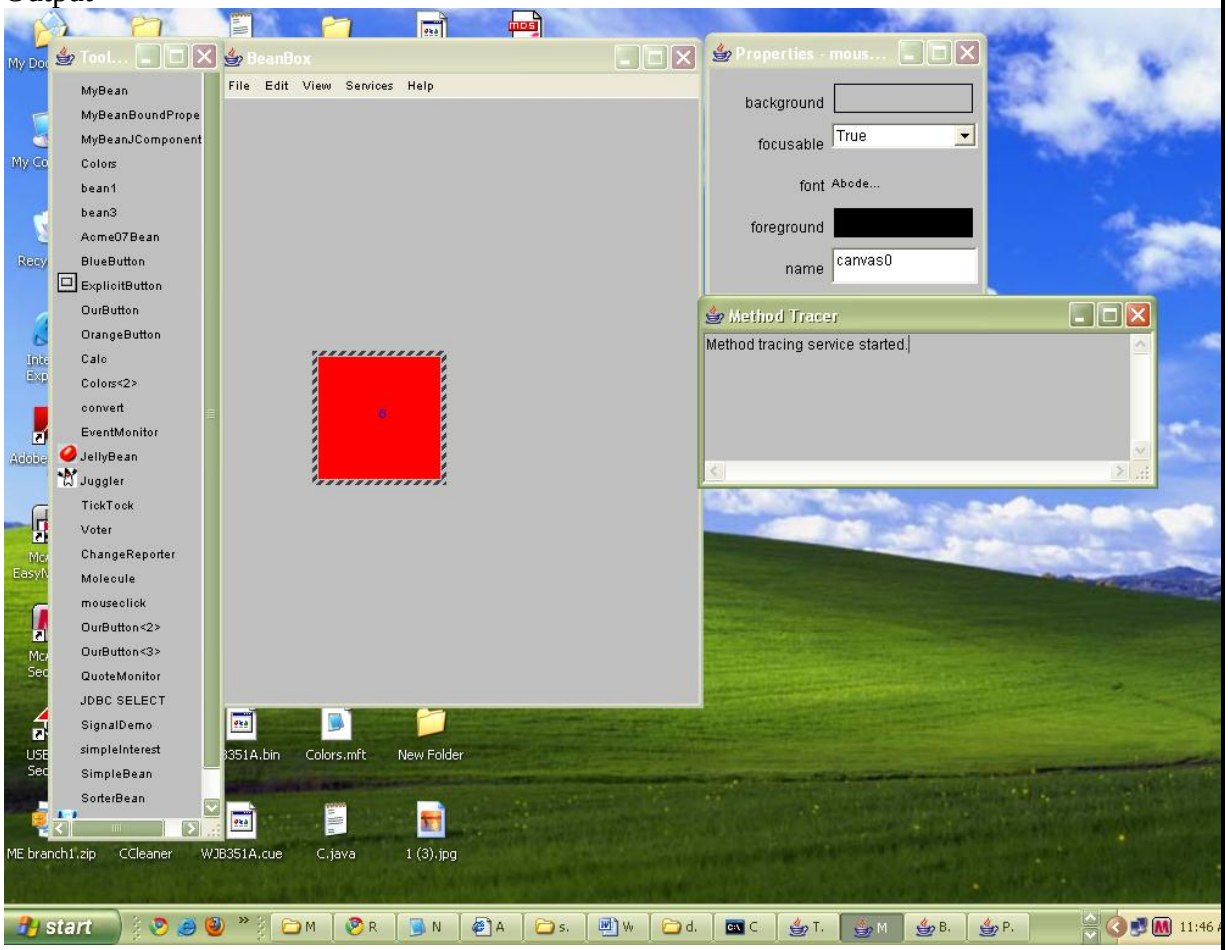
```
package sunw.demo.colors;
import java.awt.*;
import java.awt.event.*;
public class mouseclick extends Canvas {
public int count=0;
public mouseclick() {
addMouseListener(new MouseAdapter() {
public void mousePressed(MouseEvent me) {
change();
}
});
setSize(100,100);
}
public void change() {
count++;
repaint();
}
public void paint(Graphics g) {
```

```
Dimension d = getSize();
int h = d.height;
int w = d.width;
g.setColor(Color.red);
g.fillRect(0,0,100,100);
g.setColor(Color.blue);
g.drawString(String.valueOf(count),50,50);
}
}
```

Mouseclick.mft

Name: sunw/demo/colors/mouseclick.class
Java-Bean: True

Output



Result:

Thus the Mouse Clicks bean is created successfully

LAB VIVA QUESTIONS & ANSWERS

1. Define Java Bean?

JavaBeans are reusable software components for Java. Practically, they are classes that encapsulate many objects into a single object (the bean). They are serializable, have a 0-argument constructor, and allow access to properties using getter and setter methods.

2. What are the advantages of Java Beans?

A bean provides all the benefits of

- The properties, events, and methods of a bean that are exposed to another application can be controlled.
- A bean may register to receive events from other objects and can generate events that are sent to those other objects.
- Auxiliary software can be provided to help configure a java bean.
- The configuration setting of bean can be saved in a persistent storage and restored at a later time.

3. What are the disadvantages of Java Beans?

- A class with a nullary constructor is subject to being instantiated in an invalid state. If such a class is instantiated manually by a developer (rather than automatically by some kind of framework), the developer might not realize that the class has been improperly instantiated. The compiler can't detect such a problem, and even if it's documented, there's no guarantee that the developer will see the documentation.
- Having to create a getter for every property and a setter for many, most, or all of them can lead to an immense quantity of boilerplate code.

4. Explain Bound properties?

Bound properties support the PropertyChangeListener (in the API reference documentation) class. Sometimes when a `Bean` property changes, another object might need to be notified of the change, and react to the change. Whenever a bound property changes, notification of the change is sent to interested listeners.

5. Explain Constrained properties?

A bean property is constrained if the bean supports the VetoableChangeListener (in the API reference documentation) and PropertyChangeEvent (in the API reference documentation) classes, and if the set method for this property throws a PropertyVetoException (in the API reference documentation).

MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**7(a)**_____

AIM: Install TOMCAT web server and APACHE.

While installation assign port number 8080 to APACHE. Make sure that these ports are available i.e., no other process is using this port.

DESCRIPTION:

- ***Set the JAVA_HOME Variable***

You must set the JAVA_HOME environment variable to tell Tomcat where to find Java. Failing to properly set this variable prevents Tomcat from handling JSP pages. This variable should list the base JDK installation directory, not the bin subdirectory.

On Windows XP, you could also go to the Start menu, select Control Panel, choose System, click on the Advanced tab, press the Environment Variables button at the bottom, and enter the JAVA_HOME variable and value directly as:

Name: JAVA_HOME

Value: C:\jdk

- ***Set the CLASSPATH***

Since servlets and JSP are not part of the Java 2 platform, standard edition, you have to identify the servlet classes to the compiler. The server already knows about the servlet classes, but the compiler (i.e., javac) you use for development probably doesn't. So, if you don't set your CLASSPATH, attempts to compile servlets, tag libraries, or other classes that use the servlet and JSP APIs will fail with error messages about unknown classes.

Name: JAVA_HOME

Value: *install_dir/common/lib/servlet-api.jar*

- ***Turn on Servlet Reloading***

The next step is to tell Tomcat to check the modification dates of the class files of requested servlets and reload ones that have changed since they were loaded into the server's memory. This slightly degrades performance in deployment situations, so is turned off by default. However, if you fail to turn it on for your development server, you'll have to restart the server every time you recompile a servlet that has already been loaded into the server's memory.

To turn on servlet reloading, edit *install_dir/conf/server.xml* and add a DefaultContext subelement to the main Host element and supply true for the reloadable attribute. For example, in Tomcat 5.0.27, search for this entry:

```
<Host name="localhost" debug="0" appBase="webapps"  ...>
```

and then insert the following immediately below it:

```
<DefaultContext reloadable="true"/>
```

Be sure to make a backup copy of *server.xml* before making the above change.

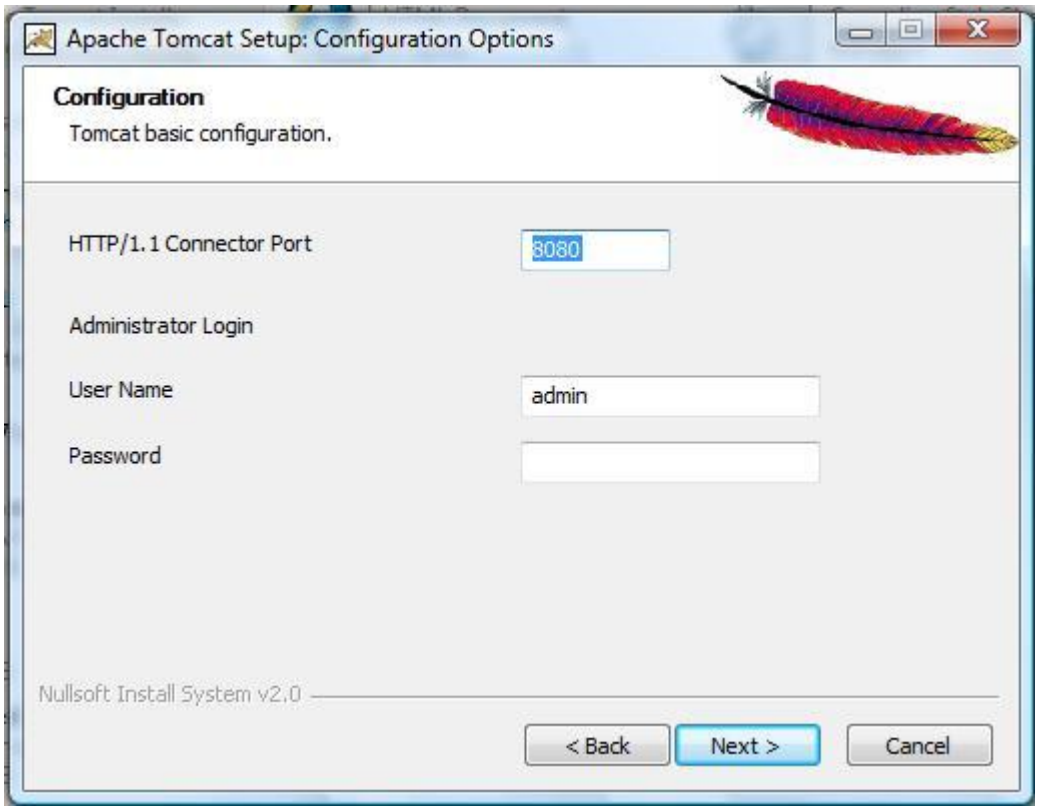
- ***Enable the Invoker Servlet***

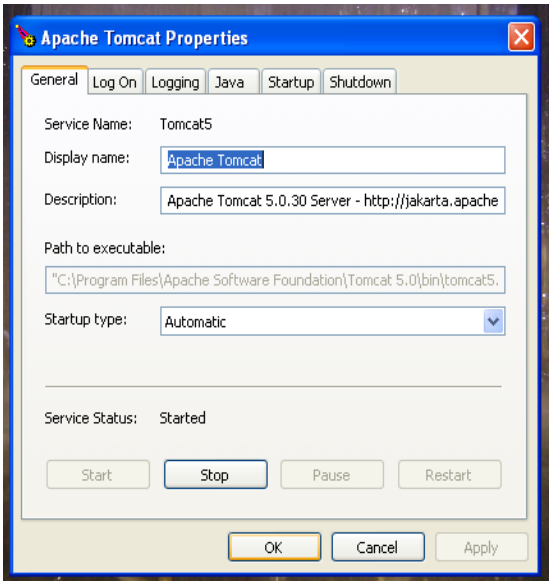
The invoker servlet lets you run servlets without first making changes to your Web application's deployment descriptor. Instead, you just drop your servlet into *WEB-INF/classes* and use the URL *http://host/servlet/ServletName*. The invoker servlet is extremely convenient when you are learning and even when you are doing your initial development.

To enable the invoker servlet, uncomment the following servlet and servlet-mapping elements in *install_dir/conf/web.xml*. Finally, remember to make a backup copy of the original version of this file before you make the changes.

```
<servlet>
    <servlet-name>invoker</servlet-name>
    <servlet-class>
      org.apache.catalina.servlets.InvokerServlet
    </servlet-class>
    ...
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
  </servlet-mapping>
```

OUTPUT:





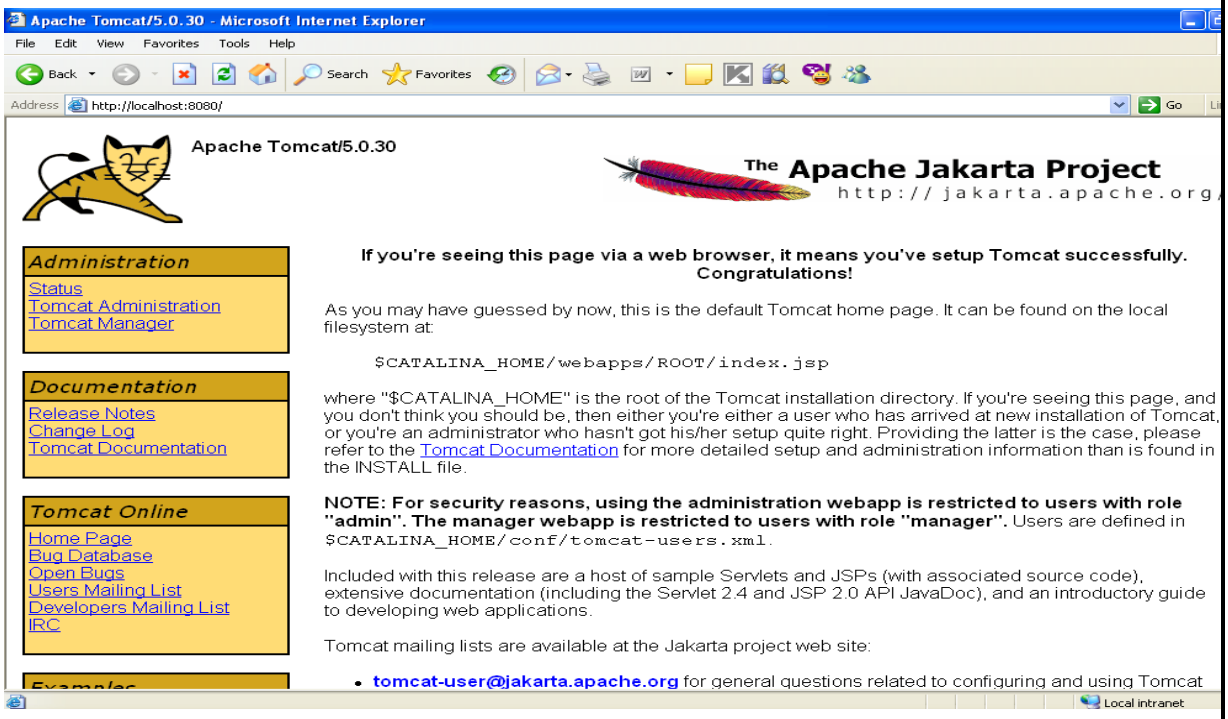
RESULT: Thus TOMCAT web server was installed successfully.

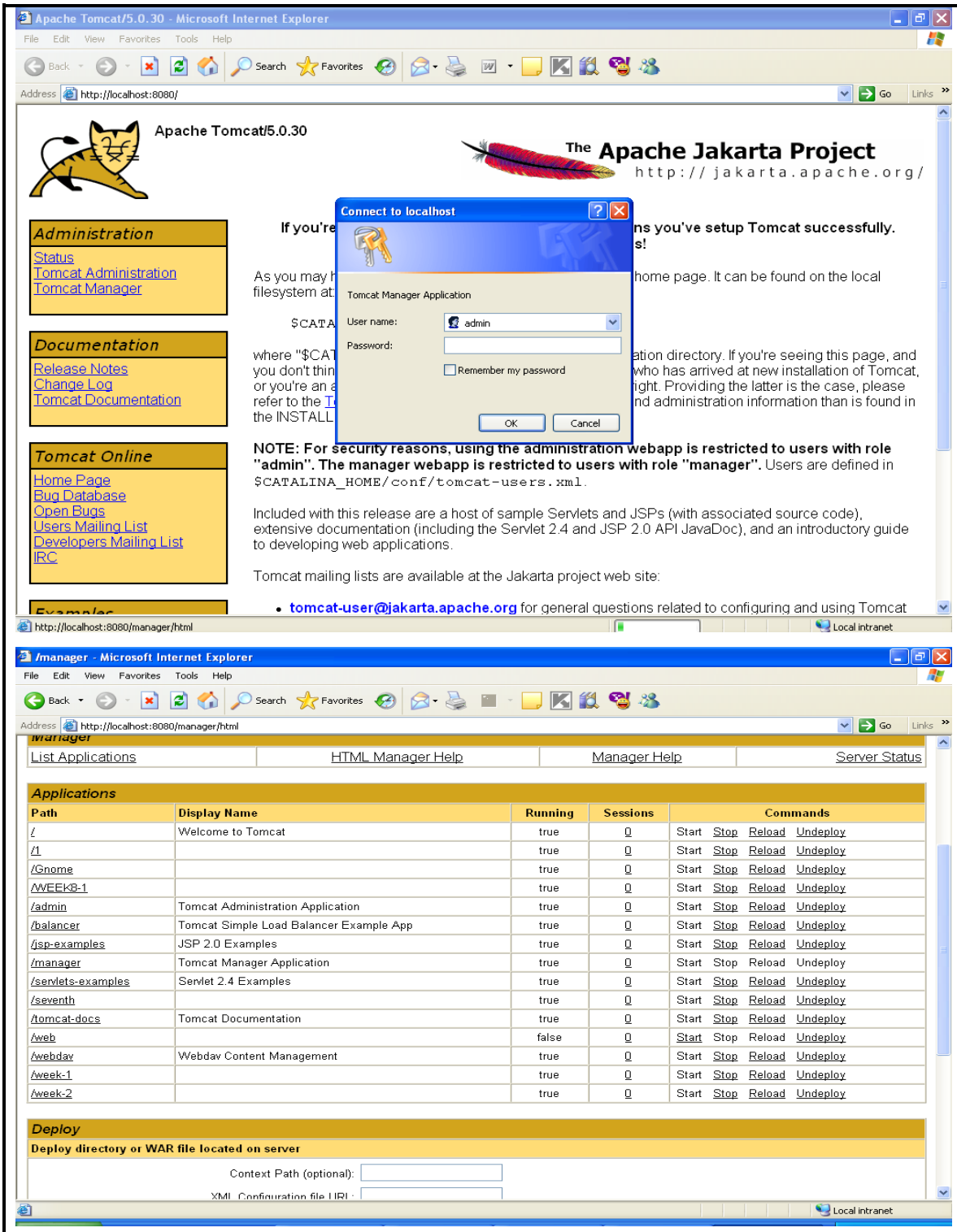
MLR INSTITUTE OF ENG & MANAGEMENT

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No: **7(b)**

AIM: Access the developed static web pages for books web site, using these servers by putting the web pages developed in week-1 and week-2 in the document root.

OUTPUT:





RESULT:

Thus week-1 and week-2 pages are accessed using the TOMCAT web server successfully.

LAB VIVA QUESTIONS & ANSWERS

1. What is Tomcat?

Tomcat is a Java Servlet container and web server from Jakartha project of Apache software foundation. A web server sends web pages as response to the requests sent by the browser client. In addition to the static web pages, dynamic web pages are also sent to the web browsers by the web server. Tomcat is sophisticated in this respect, as it provides both Servlet and JSP technologies. Tomcat provides a good choice as a web server for many web applications and also a free Servlet and JSP engine. Tomcat can be used standalone as well as behind other web servers such as Apache httpd.

2. Explain the concepts of Tomcat Servlet Container.

- Tomcat Servlet Container is a servlet container. The servlets runs in servlet container.
- The implementation of Java Servlet and the Java Server Pages is performed by this container.
- Provides HTTP web server environment in order to run Java code.
- Reduces garbage collection
- Native Windows and Unix wrappers for platform integration

3. Why do I get a ClassNotFoundException when I try to use the ShoppingCart class?

The JSP engine probably can't see the class in its classpath. Tomcat uses the system classpath, so if ShoppingCart.class is visible somewhere in the system classpath, they should see it.

4. Can I set Java system properties differently for each webapp?

No. If you can edit Tomcat's startup scripts, you can add "-D" options to Java. But there is no way to add such properties in web.xml or the webapp's context.

5. Explain Apache Tomcat?

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process.

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**8**_____

AIM: **User Authentication:**

Assume four users user1, user2, user3 and user4 having the passwords pwd1, pwd2, pwd3 and pwd4 respectively. Write a servelet for doing the following.

1. Create a Cookie and add these four user id's and passwords to this Cookie.

2. Read the user id and passwords entered in the Login form (week1) and authenticate with the values (user id and passwords) available in the cookies.

If he is a valid user (i.e., user-name and password match) you should welcome him by name (user-name) else you should display "You are not an authenticated user ". Use init-parameters to do this. Store the user-names and passwords in the web.xml and access them in the servlet by using the getInitParameters() method.

DESCRIPTION:

Servlet Life cycle:

1. *Servlet class loading*
2. *Servlet Instantiation*
3. *call the init method*
4. *call the service method*
5. *call destroy method*

Class loading and instantiation

If you consider a servlet to be just like any other Java program, except that it runs within a servlet container, there has to be a process of loading the class and making it ready for requests. Servlets do not have the exact equivalent of a main method that causes them to start execution.

When a web container starts it searches for the deployment descriptor (WEB.XML) for each of its web applications. When it finds a servlet in the descriptor it will create an instance of the servlet class. At this point the class is considered to be loaded (but not initialized).

The init method

The HttpServlet class inherits the init method from GenericServlet. The init method performs a role slightly similar to a constructor in an "ordinary" Java program in that it

allows initialization of an instance at start up. It is called automatically by the servlet container and as it causes the application context (WEB.XML) to be parsed and any initialization will be performed. It comes in two versions, one with a zero parameter constructor and one that takes a ServletConfig parameter.

The servlet engine creates a request object and a response object. The servlet engine invokes the servlet service() method, passing the request and response objects. Once the init method returns the servlet is said to be placed into service. The process of using init to initialize servlets means that it is possible to change configuration details by modifying the deployment descriptor without having them hard coded in with your Java source and needing a re-compilation.

void init(ServletConfig sc)

Calling the service method

The service() method gets information about the request from the request object, processes the request, and uses methods of the response object to create the client response. The service method can invoke other methods to process the request, such as doGet(), doPost(), or methods you write. The service method is called for each request processed and is not normally overridden by the programmer.

The code that makes a servlet "go" is the. servlet

void service(ServletRequest req,ServletResponse res)

The destroy Method

Two typical reasons for the destroy method being called are if the container is shutting down or if the container is low on resources. This can happen when the container keeps a pool of instances of servlets to ensure adequate performance. If no requests have come in for a particular servlet for a while it may destroy it to ensure resources are available for the servlets that are being requested. The destroy method is called only once, before a servlet is unloaded and thus you cannot be certain when and if it is called.

void destroy()

ServletConfig Class

ServletConfig object is used by the Servlet Container to pass information to the Servlet during it's initialization. Servlet can obtain information regarding initialization parameters and their values using different methods of ServletConfig class initialization parameters are name/value pairs used to provide basic information to the Servlet during it's initialization like JDBC driver name, path to database, username, password etc.

Methods of ServletConfig class

Following are the four methods of this class :

- **getInitParameter(String paramName)** Returns value of the given parameter. If value of parameter could not be found in web.xml file then a null value is returned.
- **getInitParameterNames()** Returns an Enumeration object containing all the names of initialization parameters provided for this Servlet.
- **getServletContext()** Returns reference to the ServletContext object for this Servlet. It is similar to getServletContext() method provided by HttpServlet class.
- **getServletName()** Returns name of the Servlet as provided in the web.xml file or if none is provided then returns complete class path to the Servlet.

PROGRAM:

cologin.html:

```
<html>
<head>
<title> login Page </title>
<p style= "background:yellow; top:100px; left:250px; position:absolute; ">
</head>
<body>
<form  ACTION="clogin">
<label> Login </label>
<input type="text" name="usr" size="20"> <br> <br>
<label> Password </label>
<input type="password" name="pwd" size="20"> <br> <br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

cologin1.html

```
<html>
<head>
<title> login Page </title>
<p style= "background:yellow; top:100px; left:250px; position:absolute; ">
</head>
<body>
```

```
<form  ACTION="clogin1">
<label> Login </label>
<input type="text" name="usr" size="20"> <br> <br>
<label> Password </label>
<input type="password" name="pwd" size="20"> <br> <br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Addcook.java:

```
import javax.servlet.* ;
import javax.servlet.http.*;
import java.io.*;
public class Addcook extends HttpServlet
{
String user,pas;
public void service(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException
{
res.setContentType("text/html");
PrintWriter out=res.getWriter();
Cookie c1=new Cookie("usr1","suni");
Cookie p1=new Cookie("pwd1","ani");
Cookie c2=new Cookie("usr2","abc");
Cookie p2=new Cookie("pwd2","123");
Cookie c3=new Cookie("usr3","def");
Cookie p3=new Cookie("pwd3","456");
Cookie c4=new Cookie("usr4","mno");
Cookie p4=new Cookie("pwd4","789");
res.addCookie(c1);
res.addCookie(p1);
res.addCookie(c2);
res.addCookie(p2);
res.addCookie(c3);
res.addCookie(p3);
```

```
res.addCookie(c4);
res.addCookie(p4);
out.println("COOKIE ADDED");
}
}
```

Clogin.java:

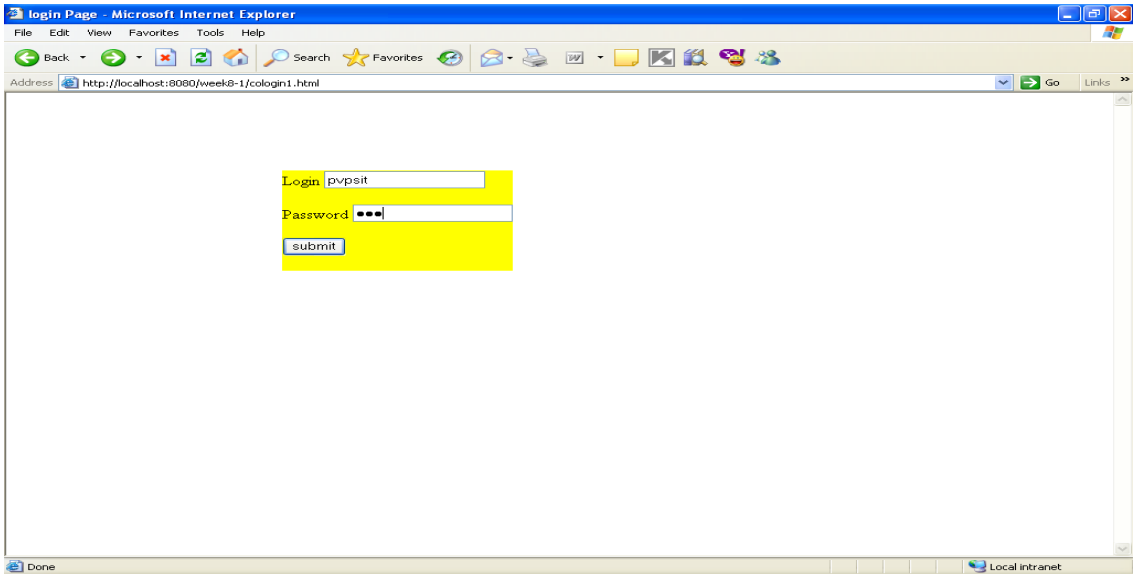
```
import javax.servlet.* ;
import javax.servlet.http.*;
import java.io.*;
public class Clogin extends HttpServlet
{
String user,pas;
public void service(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException
{
res.setContentType("text/html");
PrintWriter out=res.getWriter();
user=req.getParameter("usr");
pas=req.getParameter("pwd");
Cookie[] c=req.getCookies();
for(int i=0;i<c.length;i++)
{
if((c[i].getName().equals("usr1")&&c[i+1].getName().equals("pwd1"))||
c[i].getName().equals("usr2")
&&c[i+1].getName().equals("pwd2"))||(c[i].getName().equals("usr3")&&
c[i+1].getName().equals("pwd3"))||(c[i].getName().equals("usr4")&&
c[i+1].getName().equals("pwd4") ))
        {
            if((user.equals(c[i].getValue()) && pas.equals(c[i+1].getValue())) )
        {
        //RequestDispatcher rd=req.getRequestDispatcher("/cart.html");
        rd.forward(req,res);
        }
        else
        {
        out.println("YOU ARE NOT AUTHORISED USER ");
```

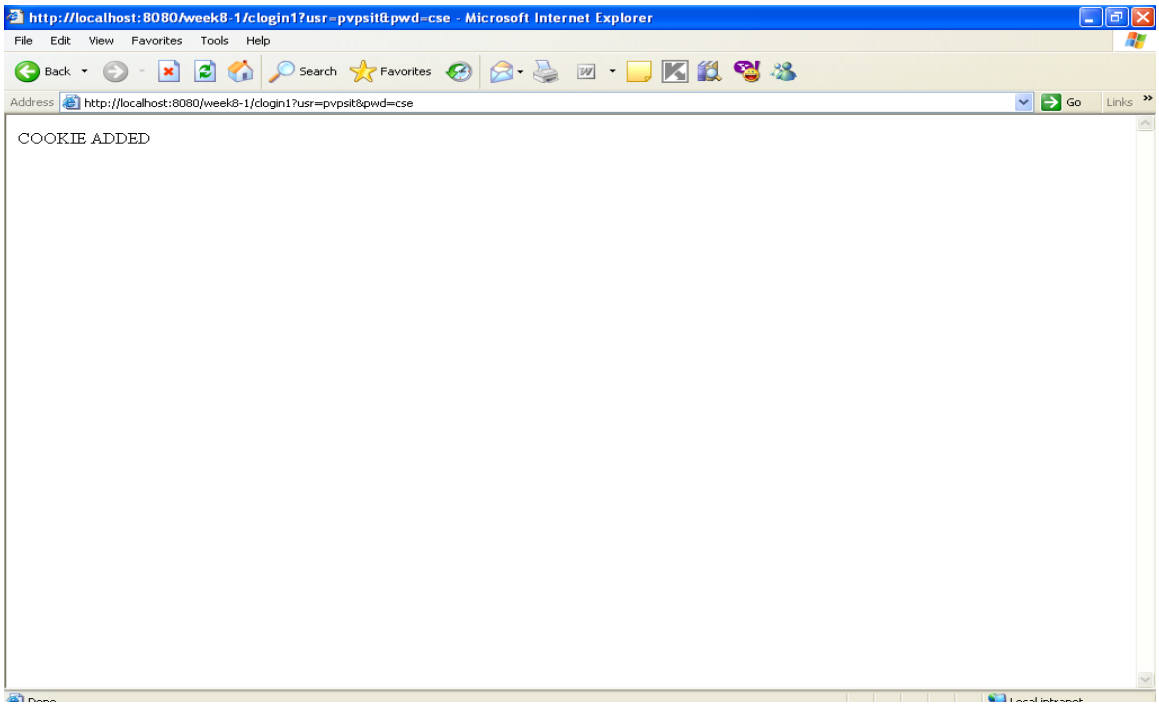
```
            //res.sendRedirect("/cookdemo/cologin.html");
            }
            }
}
}
}
```

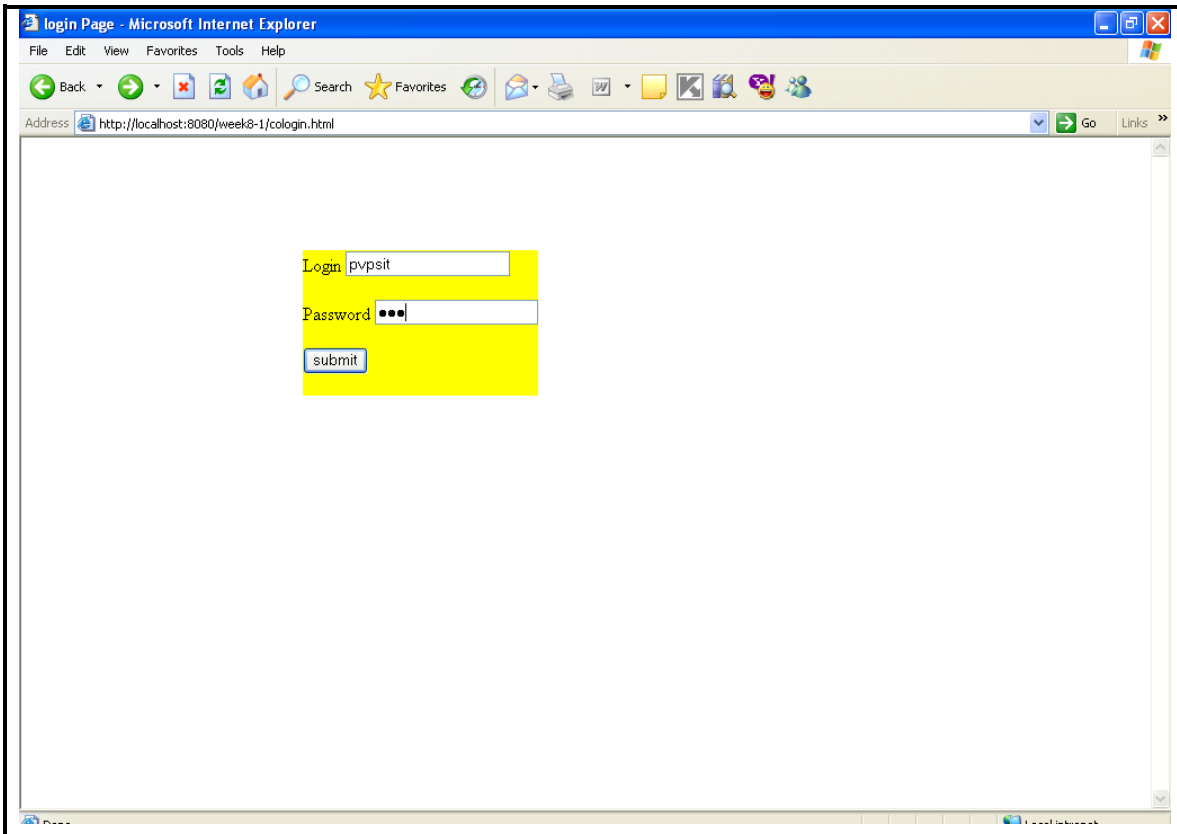
Web.xml:

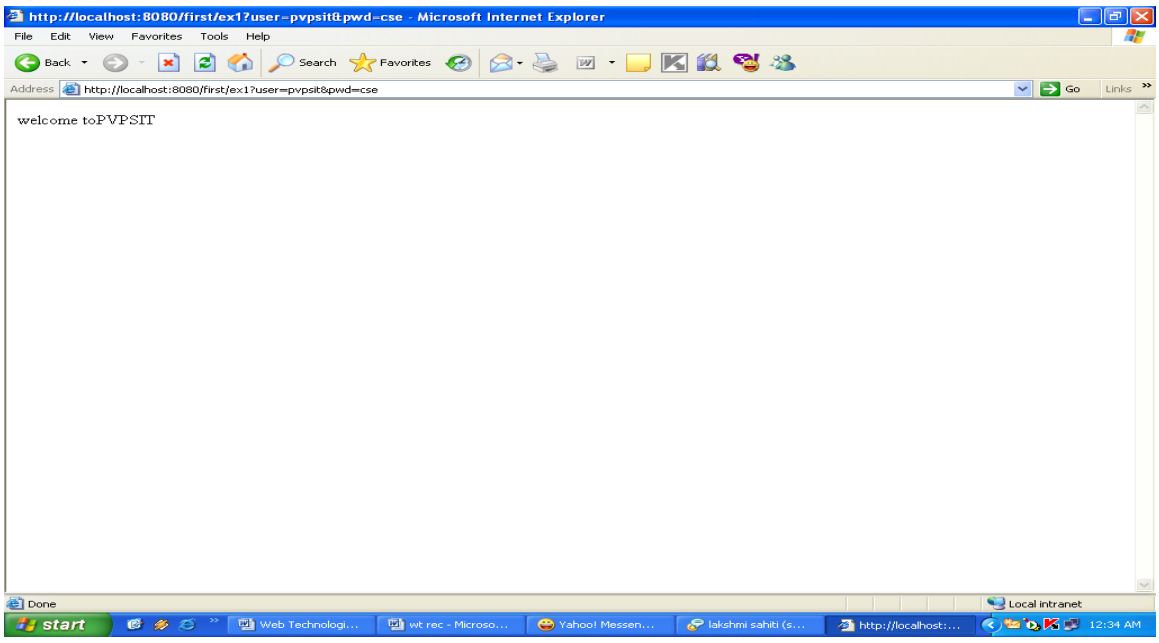
```
<web-app>
<servlet>
<servlet-name>him</servlet-name>
<servlet-class>Clogin</servlet-class>
</servlet>
<servlet>
<servlet-name>him1</servlet-name>
<servlet-class>Addcook</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>him</servlet-name>
<url-pattern>/clogin</url-pattern>
</servlet-mapping>
<servlet-mapping>
<servlet-name>him1</servlet-name>
<url-pattern>/clogin1</url-pattern>
</servlet-mapping>
</web-app>
```

OUTPUT:


login Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week8-1/cologin1.html
Go Links
Login pvpsit
Password
submit
Done
Local intranet


http://localhost:8080/week8-1/clogin1?usr=pvpsit&pwd=cse - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week8-1/clogin1?usr=pvpsit&pwd=cse
Go Links
COOKIE ADDED


login Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week8-1/cologin.html
Go Links
Login pvpsit
Password
submit


http://localhost:8080/first/ex1?user=pvpsit&pwd=cse - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/first/ex1?user=pvpsit&pwd=cse
Go Links
welcome toPVPSIT
Done
Local intranet
start
Web Technologi...
wt rec - Microso...
Yahoo! Messen...
lakshmi sahiti (s...
http://localhost:...
12:34 AM

2. Read the user id and passwords entered in the Login form (week1) and authenticate with the values (user id and passwords) available in the cookies.

If he is a valid user (i.e., user-name and password match) you should welcome him by name (user-name) else you should display “You are not an authenticated user “.

Use init-parameters to do this. Store the user-names and passwords in the webinf.xml and access them in the servlet by using the getInitParameters() method.

home.html:

```
<html>
<head>
  <title>Authentication</title>
</head>
<body>
<form action="ex1">
<label>Username </label>
<input type="text"size="20" name="user"><br><br>
password<input type="text" size="20" name="pwd"><br><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

Example1.java

```
import javax.servlet.*;
import java.io.*;
public class Example1 extends GenericServlet
{
   private String user1,pwd1,user2,pwd2,user3,pwd3,user4,pwd4,user5,pwd5;
   public void init(ServletConfig sc)
   {
     user1=sc.getInitParameter("username1");
     pwd1=sc.getInitParameter("password1");

     user2=sc.getInitParameter("username2");
     pwd2=sc.getInitParameter("password2");

          user3=sc.getInitParameter("username3");
```

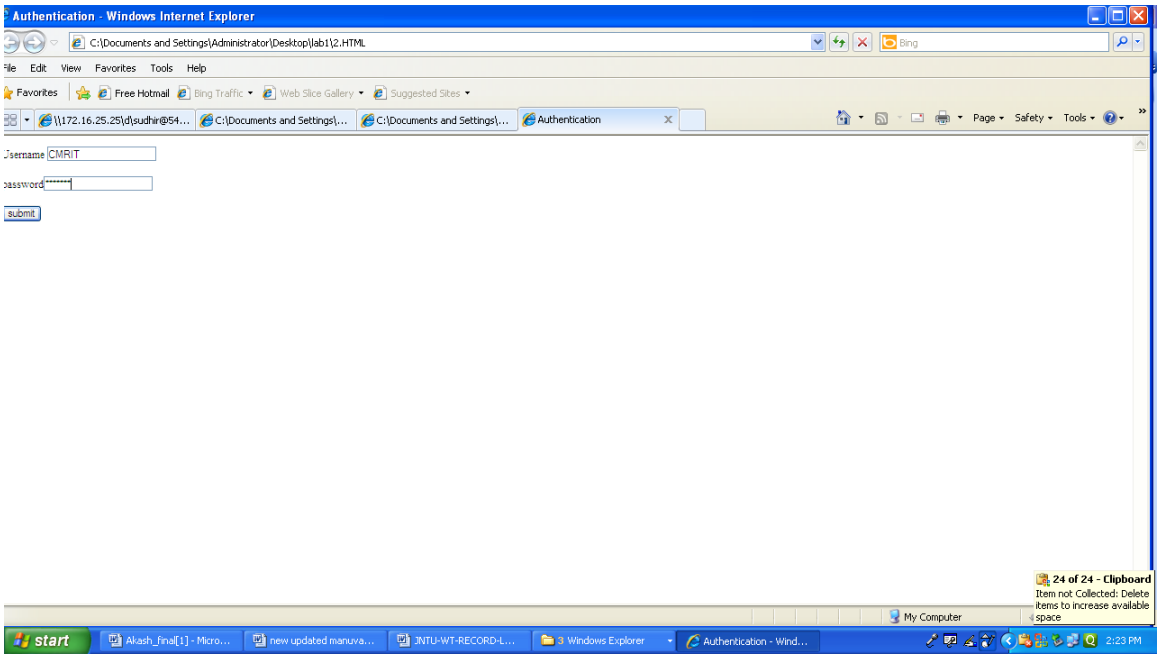
```
    pwd3=sc.getInitParameter("password3");

          user4=sc.getInitParameter("username4");
    pwd4=sc.getInitParameter("password4");
  }
  Public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
 {
    res.setContentType("text/html");
    PrintWriter out=res.getWriter();
    user5=req.getParameter("user");
    pwd5=req.getParameter("pwd");

if((user5.equals(user1)&&pwd5.equals(pwd1))||(user5.equals(user2)&&pwd5.equals(pwd2))|
|(user5.equals(user3)&&pwd5.equals(pwd3))||(user5.equals(user4)&&pwd5.equals(pwd4)))
      out.println("<p> welcome to"+user5.toUpperCase());
    else
       out.println("You are not authorized user");
 }
}
```

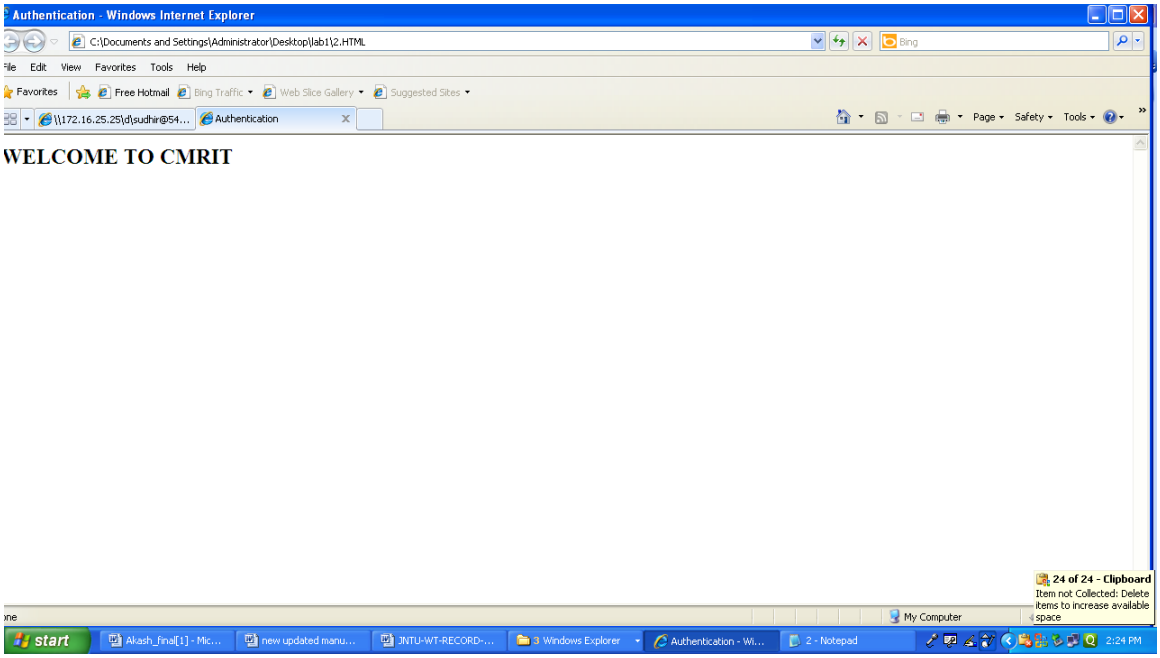
web.xml:

```
<web-app>
<servlet>
<servlet-name>Example</servlet-name>
<servlet-class>Example1</servlet-class>
<init-param>
<param-name>username1</param-name>
<param-value>pvpsit</param-value>
</init-param>
<init-param>
<param-name>password1</param-name>
<param-value>cse</param-value>
</init-param>
<init-param>
<param-name>username2</param-name>
```

```
<param-value>1234</param-value>
</init-param>
<init-param>
<param-name>password2</param-name>
<param-value>4567</param-value>
</init-param>
<init-param>
<param-name>username3</param-name>
<param-value>cse</param-value>
</init-param>
<init-param>
<param-name>password3</param-name>
<param-value>pvpsit</param-value>
</init-param>
<init-param>
<param-name>username4</param-name>
<param-value>wt</param-value>
</init-param>
<init-param>
<param-name>password4</param-name>
<param-value>lab</param-value>
</init-param>
</servlet>
<servlet-mapping>
<servlet-name>Example</servlet-name>
<url-pattern>/ex1</url-pattern>
</servlet-mapping>
</web-app>
```

OUTPUT:





RESULT:

Thus the user authentication is carried out for four users by using both cookies and getInitParameters successfully.

LAB VIVA QUESTIONS & ANSWERS

1. What is servlet?

A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request- response programming model. Before the servlet, CGI scripting language was used as server side programming language.

2. What is the use of servlet?

Uses of servlet includes:

- Processing and storing data submitted by an HTML form.
- Providing dynamic content.
- A Servlet can handle multiple request concurrently and be used to develop high performance system
- Managing state information on top of the stateless HTTP.

3. What is the life cycle of servlet?

Life cycle of Servlet:

- Servlet class loading
- Servlet instantiation
- Initialization (call the init method)
- Request handling (call the service method)
- Removal from service (call the destroy method)

4. How servlet is loaded?

The servlet is loaded by:

- First request is made.
- Server starts up (auto-load).
- There is only a single instance which answers all requests concurrently. This saves memory and allows a Servlet to easily manage persistent data.
- Administrator manually loads.

5. What is servlet interface?

The central abstraction in the Servlet API is the Servlet interface. All servlets implement this interface, either directly or more commonly by extending a class that implements it.

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**9**_____

AIM: Install a database (Mysql or Oracle).

Create a table which should contain at least the following fields: name, password, email-id, phone number (these should hold the data from the registration form).

Practice 'JDBC' connectivity.

Write a java program/servlet/JSP to connect to that database and extract data from the tables and display them. Experiment with various SQL queries.

Insert the details of the users who register with the web site, whenever a new user clicks the submit button in the registration page (week2).

DESCRIPTION:

JDBC Driver Types

There are four types of JDBC drivers in use:

Type 1: JDBC-ODBC Bridge

A Type 1 JDBC-ODBC Bridge provides application developers with a way to access JDBC drivers via the JDBC API. Type 1 JDBC drivers translate the JDBC calls into ODBC calls and then send the calls to the ODBC driver. Type 1 JDBC drivers are generally used when the database client libraries need to be loaded on every client machine.

Type 2: Native API/Partly Java Driver

A Type 2 Native API/Partly Java Driver is a partial Java driver because it converts JDBC calls into database specific calls. Type 2 Native API/Partly Java Driver communicates directly with the database server.

Type 3: Pure Java Driver

A Type 3 Pure Java Driver works in a three tiered architecture. The JDBC calls are passed via the network to the middle tier server. This server translates the calls to the database specific native interface to further request the server. JDBC drivers available from Simba are Type 3 drivers.

Type 4: Native Protocol Java Driver

The type 4 driver is written completely in Java and is hence platform independent. It is installed inside the Java Virtual Machine of the client. It provides better performance over the type 1 and 2 drivers as it does not have the overhead of conversion of calls into ODBC or

database API calls. Unlike the type 3 drivers, it does not need associated software to work.A Type 4 Native Protocol Java Driver converts JDBC calls into the database specific calls so that the client applications can communicate directly with the server.

PROGRAM:

Registration.html:

```
<html>
<head>
<title>Registration page</title>
</head>
<body bgcolor="#00FFFf">
<form METHOD="POST" ACTION="register">
<CENTER>
<table>
<center>
<tr> <td> Username </td>
<td><input type="text" name="usr"> </td> </tr>
<tr><td> Password </td>
<td><input type="password" name="pwd"> </td> </tr>
<tr><td>Age</td>
<td><input type="text" name="age"> </td> </tr>
<tr> <td>Address</td>
<td> <input type="text" name="add"> </td> </tr>
<tr> <td>email</td>
<td> <input type="text" name="mail"> </td> </tr>
<tr> <td>Phone</td>
<td> <input type="text" name="phone"> </td> </tr>
<tr> <td colspan=2 align=center> <input type="submit" value="submit"> </td> </tr>
</center>
</table>
</form>
</body>
```

Login.html

```
<html>
<head>
<title>Registration page</title>
</head>
<body bgcolor=pink> <center> <table>
<form METHOD="POST" ACTION="authent">
<tr> <td> Username </td>
<td><input type="text" name="usr"></td> </tr>
<tr> <td> Password </td>
<td> <input type="password" name="pwd"> </td> </tr>
<tr> <td align=center colspan="2"><input type="submit" value="submit"></td> </tr>
</table> </center>
</form>
</body>
</html>
```

Ini.java:

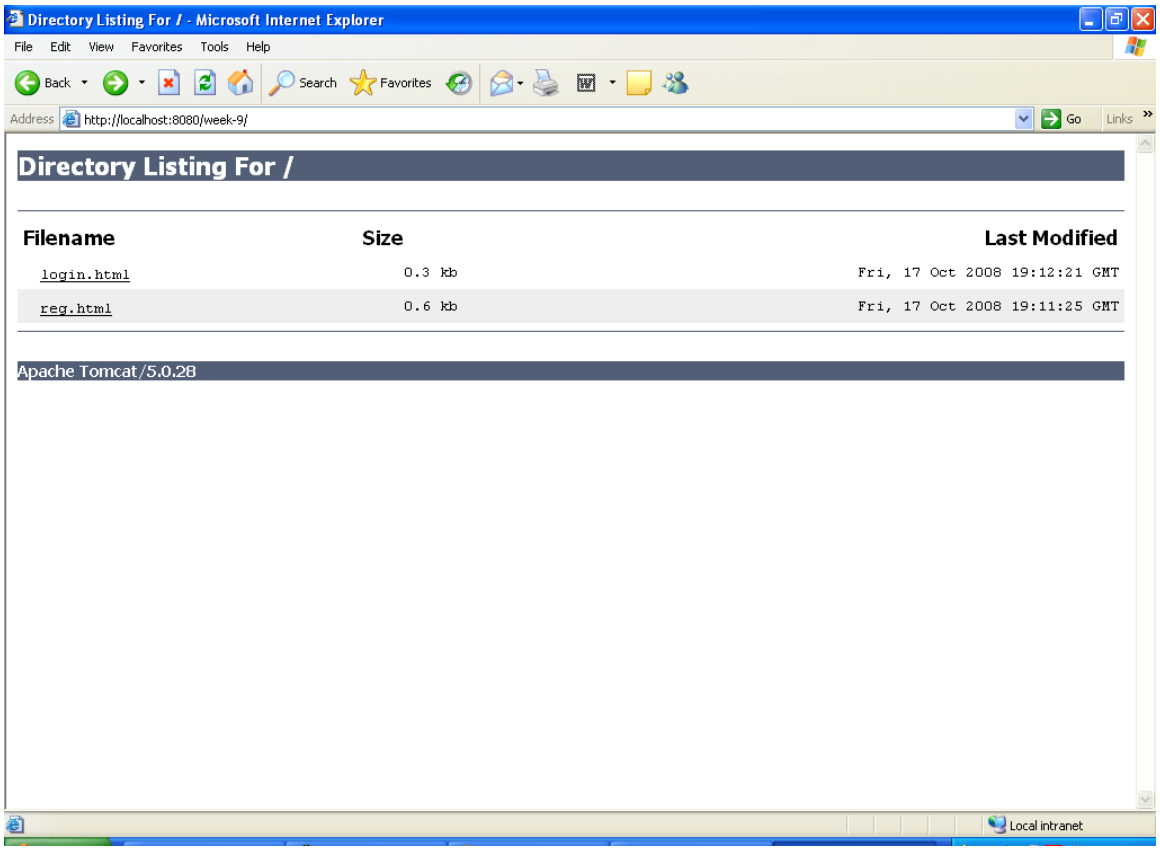
```
import javax.servlet.*;
import java.sql.*;
import java.io.*;
public class Ini extends GenericServlet
{
private String user1,pwd1,email1;

public void service(ServletRequest req,ServletResponse res) throws
ServletException,IOException
{
user1=req.getParameter("user");
pwd1=req.getParameter("pwd");
email1=req.getParameter("email");
res.setContentType("text/html");
PrintWriter out=res.getWriter();
try
  {
   Class.forName("oracle.jdbc.driver.OracleDriver");
```

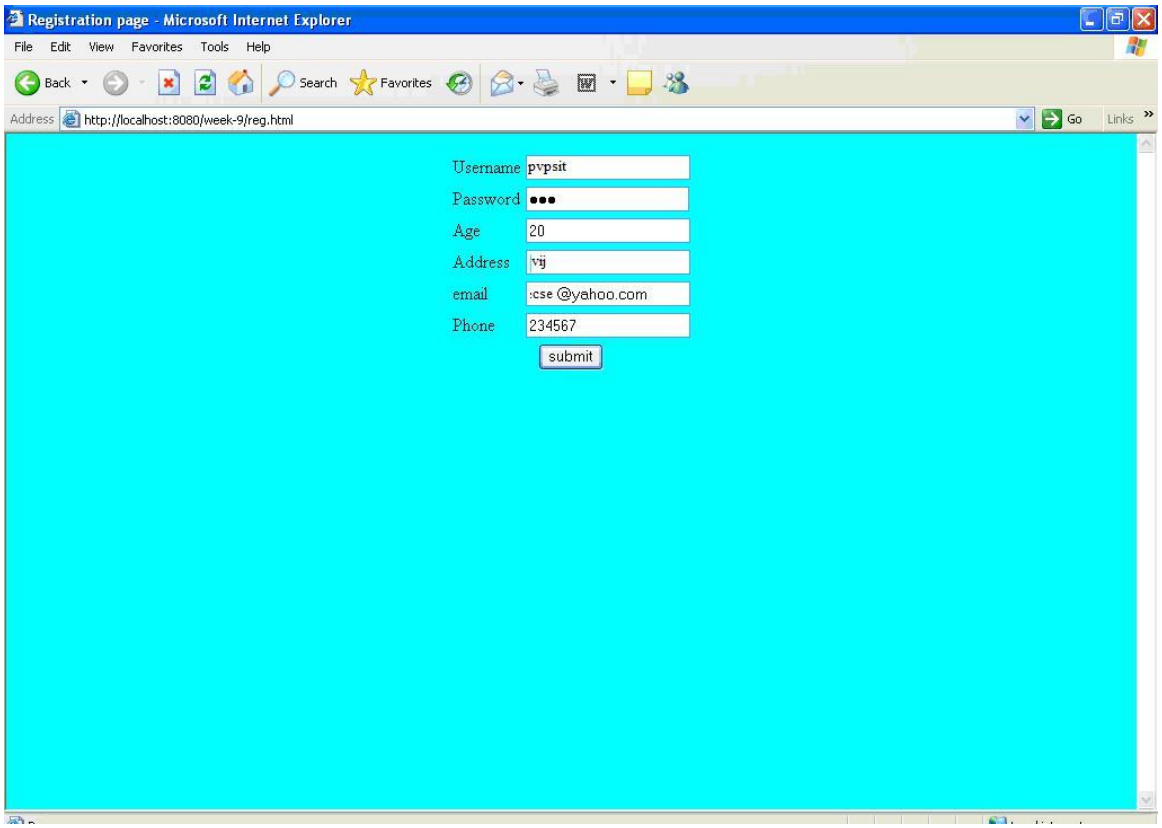
```
   Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@195.100.101.158:1521:cclab","scott",
"tiger");
   PreparedStatement st=con.prepareStatement("insert into personal values(?,?,?,?,?,?)");
 st.setString(1,user1);
st.setString(2,pwd1);
st.setString(3,"25");
st.setString(4,"hyd");
st.setString(5,email1);
st.setString(6,"21234");
st.executeUpdate();
con.close();
  }
catch(SQLException s)
{  out.println("not found "+s);
}
catch(ClassNotFoundException c)
{
   out.println("not found "+c);
}
} }
```

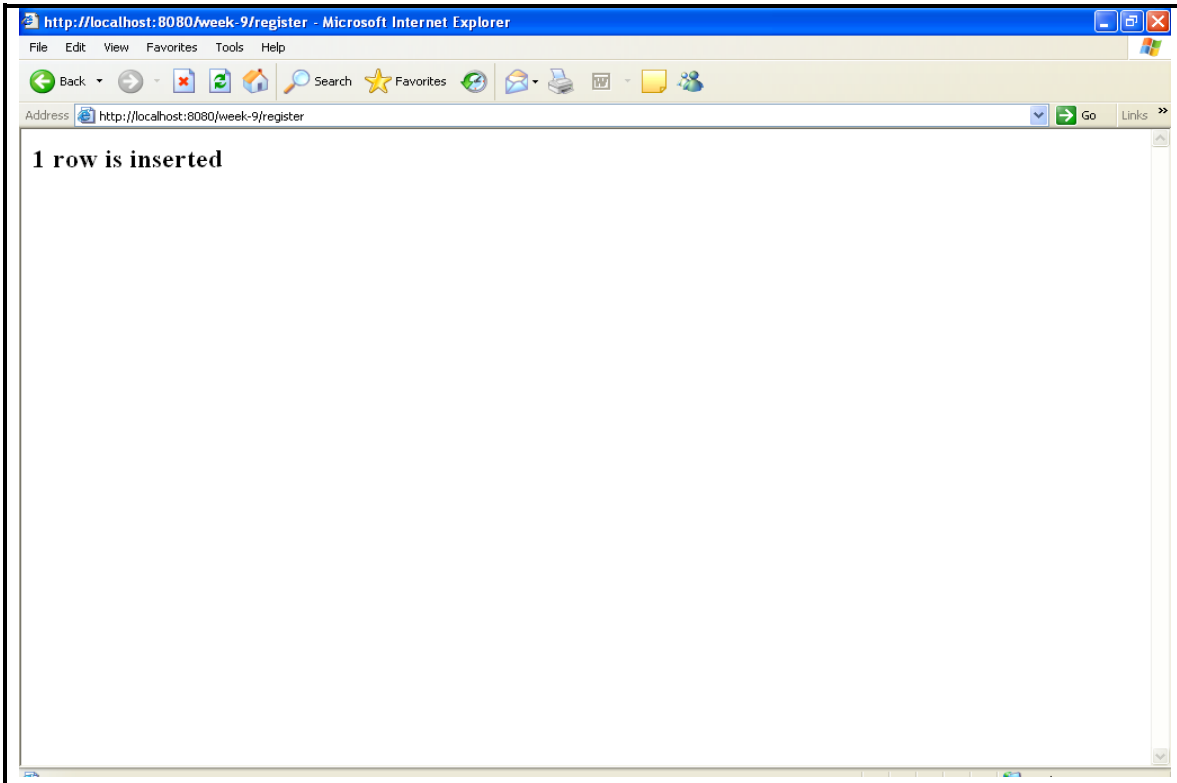
web.xml:

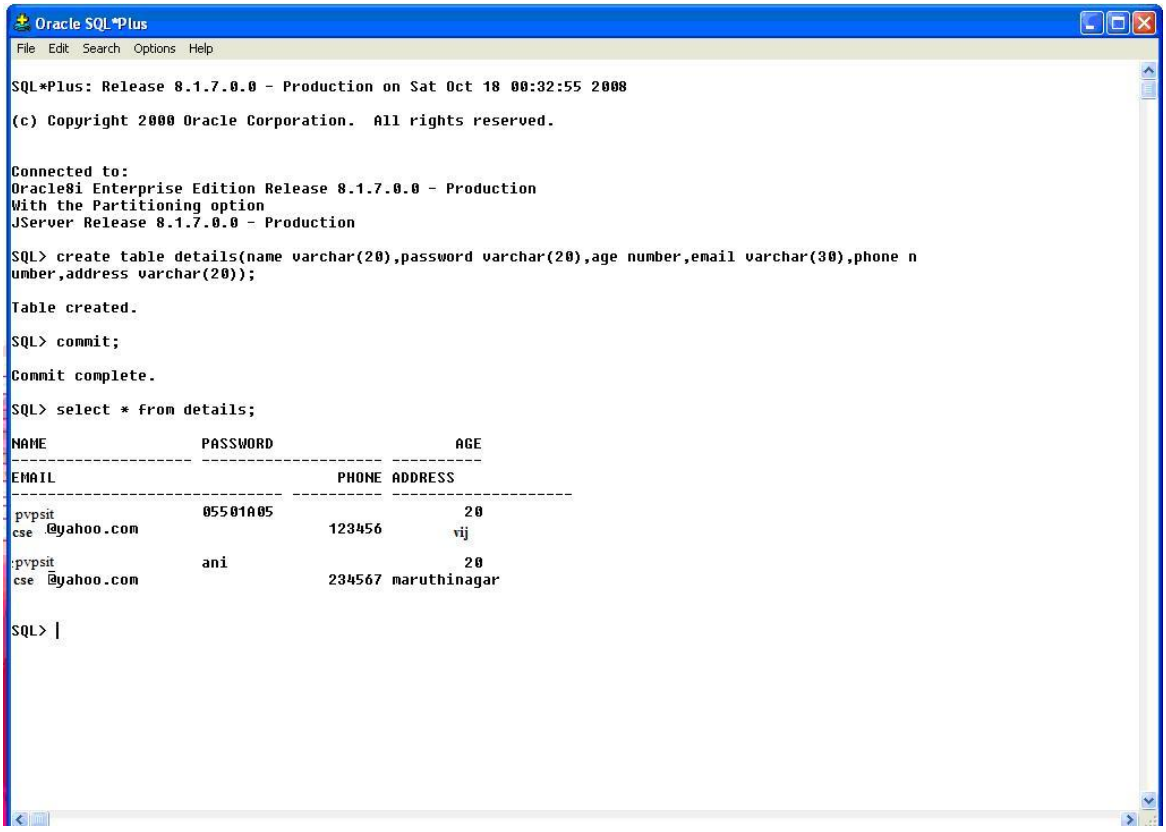
```
<web-app>
<servlet>
<servlet-name>init1</servlet-name>
<servlet-class>Ini</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>init1</servlet-name>
<url-pattern>/regis</url-pattern>
</servlet-mapping>
</web-app>
```

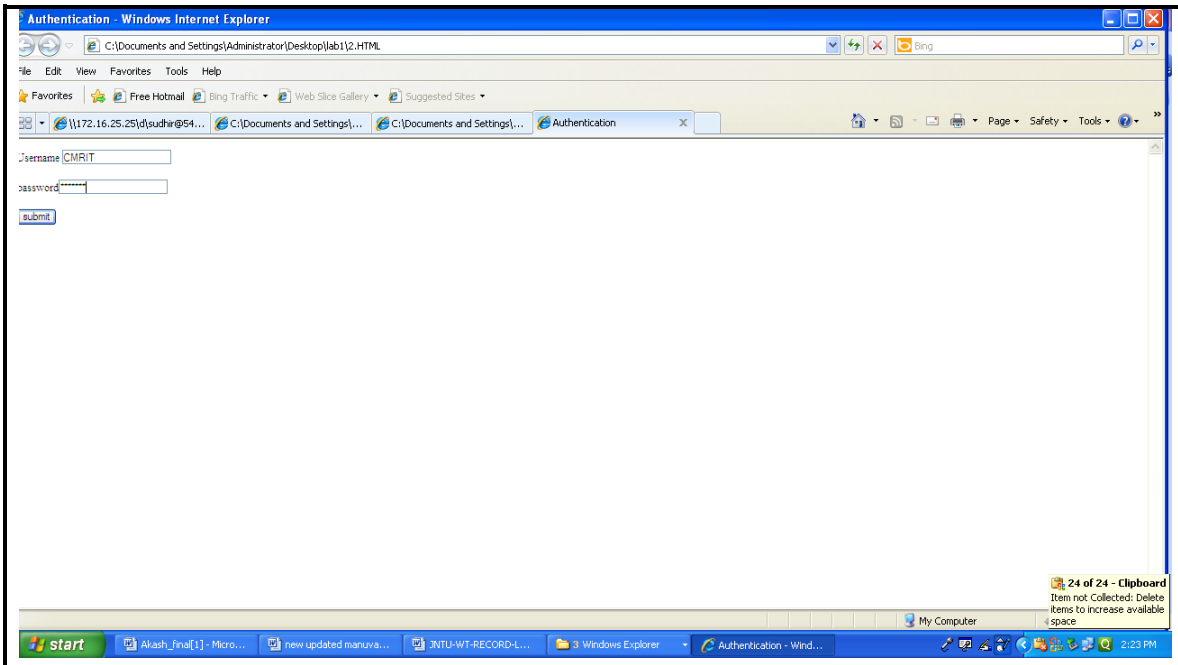
OUTPUT:

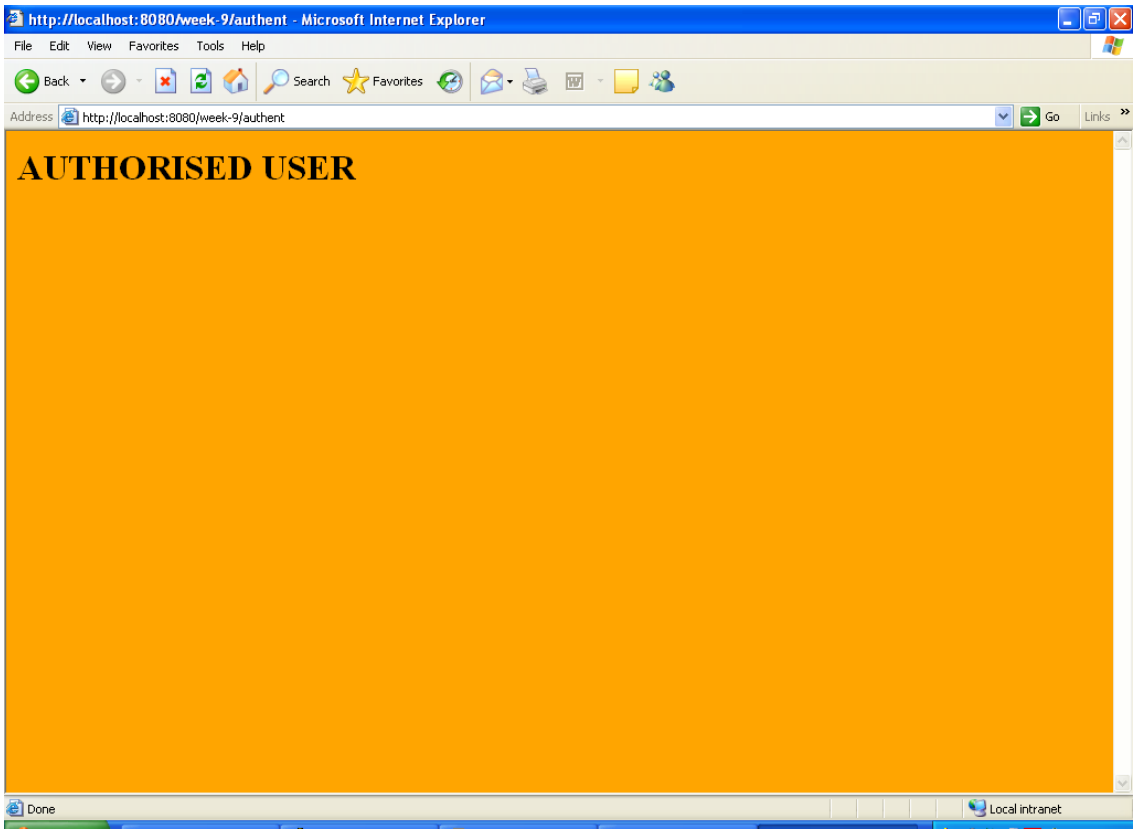

Directory Listing For / - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week-9/
Go Links
Directory Listing For /
Filename Size Last Modified
login.html 0.3 kb Fri, 17 Oct 2008 19:12:21 GMT
reg.html 0.6 kb Fri, 17 Oct 2008 19:11:25 GMT
Apache Tomcat/5.0.28
Local intranet

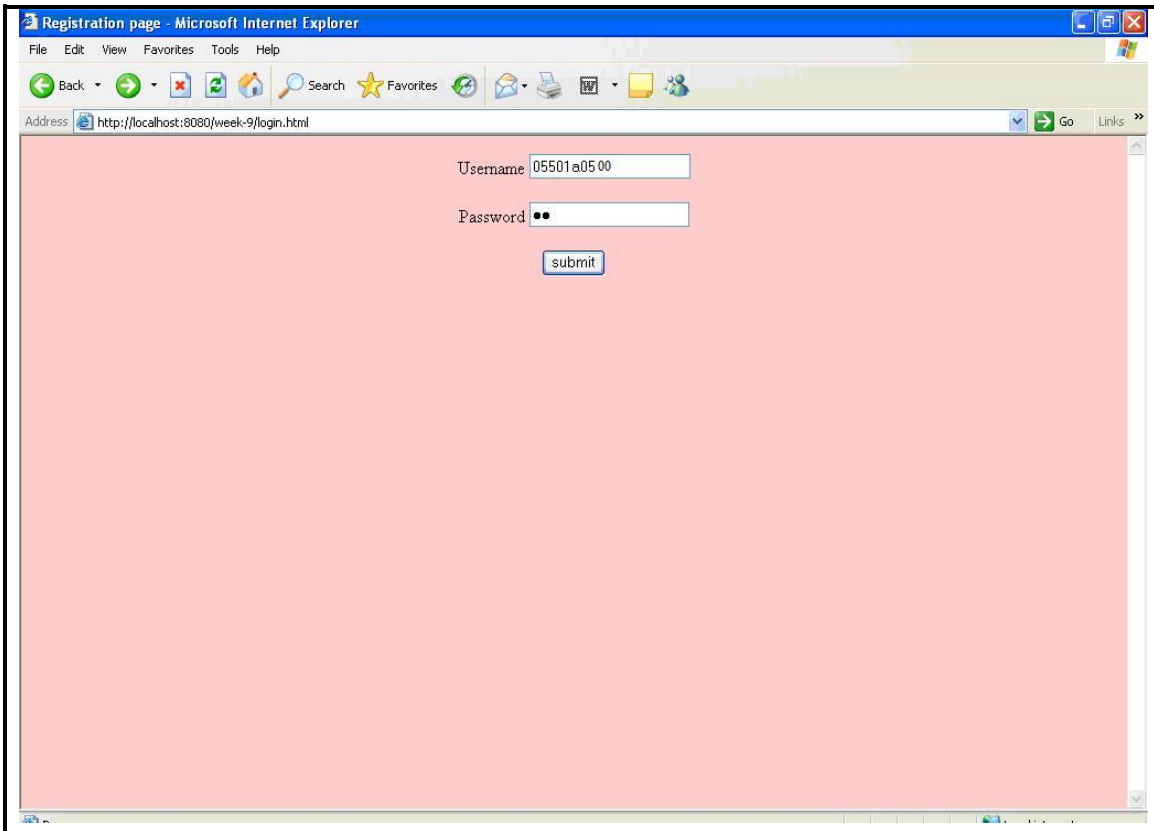

Registration page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week-9/reg.html
Go Links
Username pvpsit
Password
Age 20
Address vij
email :cse @yahoo.com
Phone 234567
submit

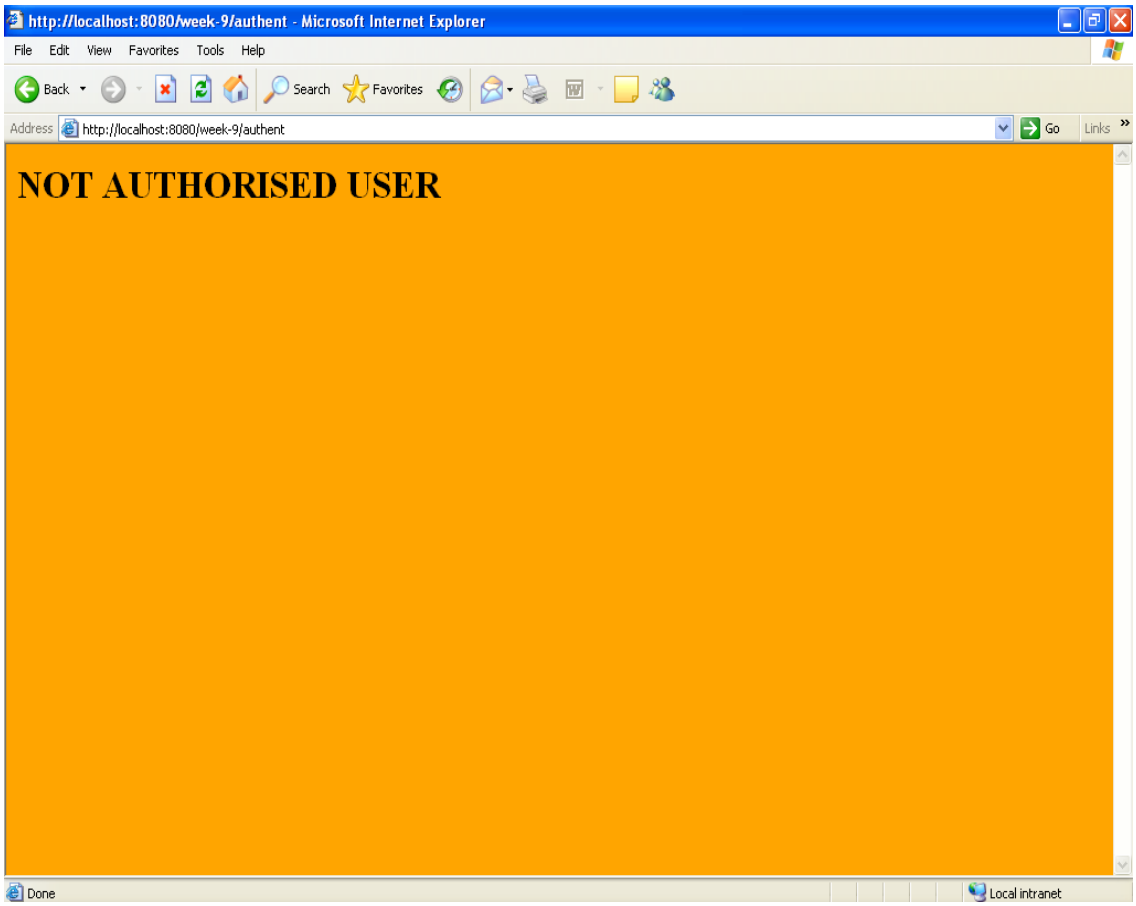

http://localhost:8080/week-9/register - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week-9/register
Go Links
1 row is inserted


Oracle SQL*Plus
File Edit Search Options Help
SQL*Plus: Release 8.1.7.0.0 - Production on Sat Oct 18 00:32:55 2008
(c) Copyright 2000 Oracle Corporation. All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.7.0.0 - Production
With the Partitioning option
JServer Release 8.1.7.0.0 - Production
SQL> create table details(name varchar(20),password varchar(20),age number,email varchar(30),phone n
umber,address varchar(20));
Table created.
SQL> commit;
Commit complete.
SQL> select * from details;
NAME PASSWORD AGE
EMAIL PHONE ADDRESS
pvpsit 05501A05 20
cse @yahoo.com 123456 vij
pvpsit ani 20
cse @yahoo.com 234567 maruthinagar
SQL>


Authentication - Windows Internet Explorer
C:\Documents and Settings\Administrator\Desktop\lab1\2.HTML
Username CMRIT
submit
start


http://localhost:8080/week-9/authent - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:8080/week-9/authent
AUTHORISED USER
Done
Local intranet





RESULT:

Thus a table is created and the details are entered into the table using jdbc from the registration form successfully.

LAB VIVA QUESTIONS & ANSWERS

1. What is JDBC?

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

2. Describe a general JDBC Architecture.

General JDBC Architecture consists of two layers: **JDBC API** (This provides the application-to-JDBC Manager connection) and **JDBC Driver API** (This supports the JDBC Manager-to-Driver Connection).

3. What is a JDBC Driver?

JDBC driver is an interface enabling a Java application to interact with a database. To connect with individual databases, JDBC requires drivers for each database. The JDBC driver gives out the connection to the database and implements the protocol for transferring the query and result between client and database.

4. What is a connection?

Connection interface consists of methods for contacting a database. The connection object represents communication context.

5. What is a statement?

Statement encapsulates an SQL statement which is passed to the database to be parsed, compiled, planned and executed.

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**10**_____

AIM: Write a JSP which does the following job

Insert the details of the 3 or 4 users who register with the web site (week9) by using registration form. Authenticate the user when he submits the login form using the user name and password from the database (similar to week8 instead of cookies).

DESCRIPTION:

JSP Scripting Elements

JSP scripting elements let you insert Java code into the servlet that will be generated from the current JSP page. There are three forms:

1. Expressions of the form <%= expression %> that are evaluated and inserted into the output,
2. Scriptlets of the form <% code %> that are inserted into the servlet's service method, and
3. Declarations of the form <%! code %> that are inserted into the body of the servlet class, outside of any existing methods.

Each of these is described in more detail below.

JSP Expressions

A JSP *expression* is used to insert Java values directly into the output. It has the following form:

<%= Java Expression **%>**

The Java expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at run-time (when the page is requested), and thus has full access to information about the request. For example, the following shows the date/time that the page was requested:

Current time: <%= new java.util.Date() %>

To simplify these expressions, there are a number of predefined variables that you can use. These implicit objects are discussed in more detail later, but for the purpose of expressions, the most important ones are:

- request, the HttpServletRequest;
- response, the HttpServletResponse;
- session, the HttpSession associated with the request (if any); and
- out, the PrintWriter (a buffered version of type JspWriter) used to send output to the

client.

JSP Scriptlets

If you want to do something more complex than insert a simple expression, JSP *scriptlets* let you insert arbitrary code into the servlet method that will be built to generate the page. Scriptlets have the following form:

<% Java Code **%>**

Scriptlets have access to the same automatically defined variables as expressions. So, for example, if you want output to appear in the resultant page, you would use the out variable.

```
<%
String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData);
%>
```

Note that code inside a scriptlet gets inserted *exactly* as written, and any static HTML (template text) before or after a scriptlet gets converted to print statements. This means that scriptlets need not contain complete Java statements, and blocks left open can affect the static HTML outside of the scriptlets.

JSP Declarations

A JSP *declaration* lets you define methods or fields that get inserted into the main body of the servlet class (outside of the service method processing the request). It has the following form:

<%! Java Code **%>**

Since declarations do not generate any output, they are normally used in conjunction with JSP expressions or scriptlets. For example, here is a JSP fragment that prints out the number of times the current page has been requested since the server booted (or the servlet class was changed and reloaded):

```
<%! private int accessCount = 0; %>
```

PROGRAM:

Login.html:

```
<!--Home.html-->
<html> <body>
<center><h1>XYZ Company Ltd.</h1></center>
<table border="1" width="100%" height="100%">
<tr>
        <td valign="top" align="center"><br/>
```

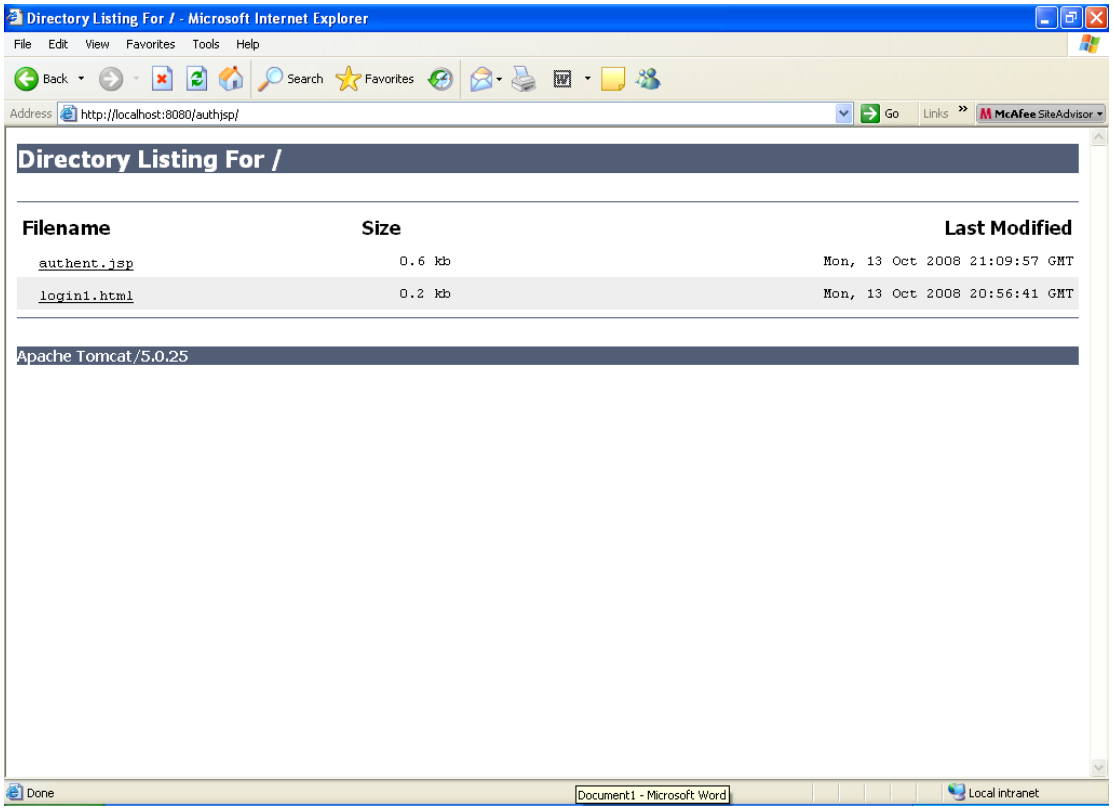
```
                    <form action="auth.jsp"><table>
                    <tr>
                        <td colspan="2" align="center"><b>Login Page</b></td>
                    </tr>
                    <tr>
                        <td colspan="2" align="center"><b> </td>
                    </tr>
                    <tr>
                         <td>User Name</td>
                         <td><input type="text" name="user"/></td>
                    </tr>
                    <tr>
                         <td>Password</td>
                           <td><input type="password" name="pwd"/></td>
                    </tr>
                    <tr>
                            <td> </td>
                            <td> </td>
                    </tr>
                    <tr>
            <td colspan="2" align="center"><input type="submit" value="LogIN"/></td>
                    </tr>
                    </table>
                    </form>
        </td>
</tr>
</table>
</body>
</html>
```

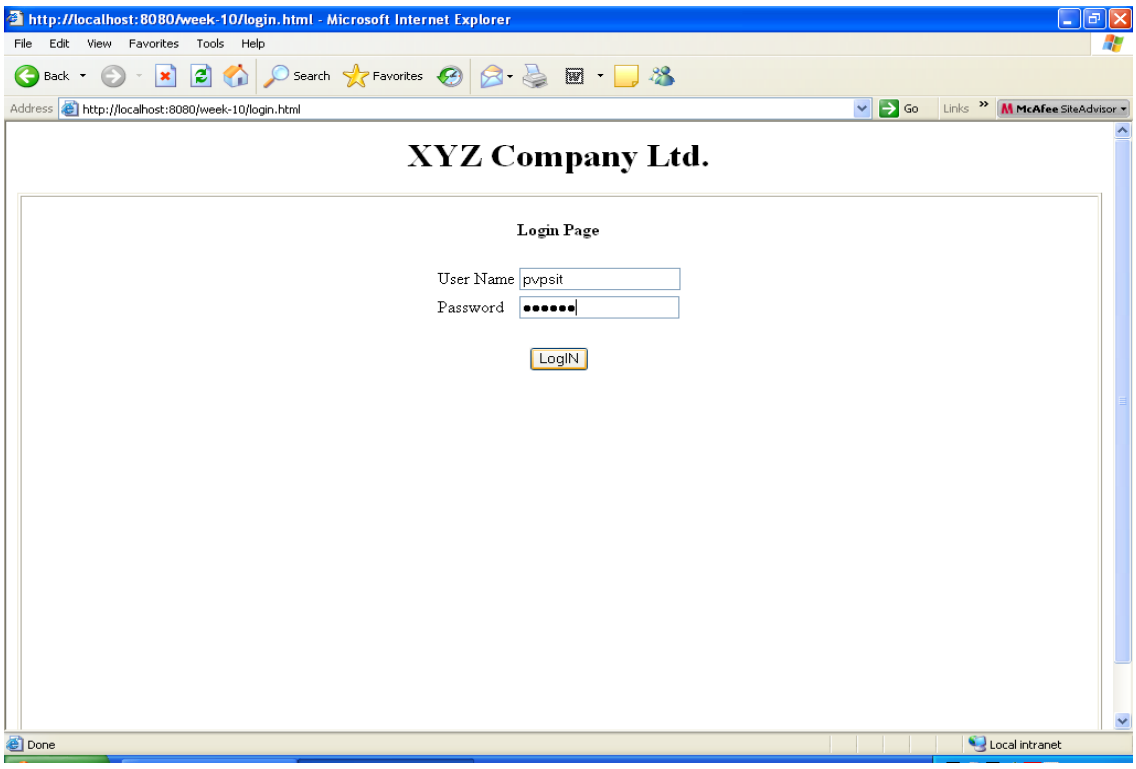
Auth.jsp:

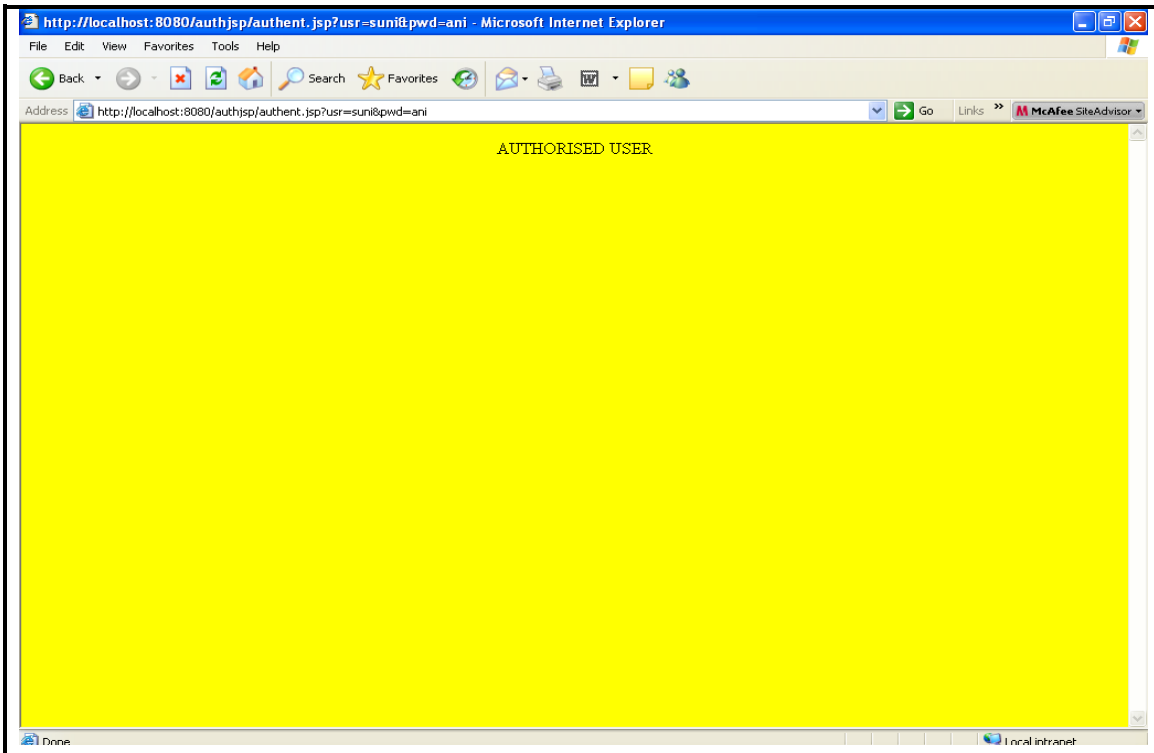
```
<%@page import="java.sql.*;"%>
<html>
<head>
<title>
This is simple data base example in JSP</title>
</title>
```

```
</head>
<body bgcolor="yellow">
<%!String uname,pwd;%>
<%
uname=request.getParameter("user");
pwd=request.getParameter("pwd");
  try
{
  Class.forName("oracle.jdbc.driver.OracleDriver");
  Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@195.100.101.158:1521:CCLAB","sco
tt","tiger");
  Statement st=con.createStatement();
  ResultSet rs=st.executeQuery("select name,password from personal where
name='"+uname+"' and password='"+pwd+"'");
  if(rs.next())
{
out.println("Authorized person");
}
else
{
out.println("UnAuthorized person");
}
con.close();
}
catch(Exception e){out.println(""+e);}
%>
</body>
</html>
```

OUTPUT:


Directory Listing For / - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/authjsp/
Go Links McAfee SiteAdvisor
Directory Listing For /
Filename Size Last Modified
authent.jsp 0.6 kb Mon, 13 Oct 2008 21:09:57 GMT
login1.html 0.2 kb Mon, 13 Oct 2008 20:56:41 GMT
Apache Tomcat/5.0.25
Done
Document1 - Microsoft Word
Local intranet


http://localhost:8080/week-10/login.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:8080/week-10/login.html
Go Links McAfee SiteAdvisor
XYZ Company Ltd.
Login Page
User Name pvpsit
Password
LogIN
Done
Local intranet



RESULT:

The user is authenticated when he submits the login form using the user name and password from the database.

LAB VIVA QUESTIONS & ANSWERS

1. What is JSP?

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, but it uses the Java Programming Language.

2. What is the life cycle of jsp?

Life cyle of jsp:

- Translation
- Compilation
- Loading the class
- Instantiating the class
- jspInit()
- _jspService()
- jspDestroy()

3. What are implicit objects in jsp?

Implicit objects in JSP are the Java objects that the JSP Container makes available to developers in each page. These objects need not be declared or instantiated by the JSP author. They are automatically instantiated by the container and are accessed using standard variables; hence, they are called implicit objects.

4. What is the <jsp: useBean> standard action?

The <jsp: useBean> standard action is used to locate an existing Java Bean or to create a Java Bean if it does not exist. It has attributes to identify the object instance, to specify the lifetime of the bean, and to specify the fully qualified class path and type.

5. What is the jsp declaration?

JSP declarations are used to declare class variables and methods in a JSP page. They are initialized when the class is initialized. Anything defined in a declaration is available for the whole JSP page. A declaration block is enclosed between the <%! and %>tags. A declaration is not included in the service() method when a JSP is translated to a servlet.

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**11**_____

AIM: Extract data from the tables and display them in the catalogue page using JDBC.

DESCRIPTION:

Create tables in the database which contain the details of items (books in our case like Book name, Price, Quantity, Amount)) of each category. Modify your catalogue page (week 2) in such a way that you should connect to the database and extract data from the tables and display them in the catalogue page using JDBC.

PROGRAM:

Retrieve.java:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import java.io.*;
import java.util.*;
public class Retrieve extends HttpServlet
{
public void service(HttpServletRequest req,HttpServletResponse res) throws ServletException,IOException
{
res.setContentType("text/html");
PrintWriter out=res.getWriter();
try{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@195.100.101.158:1521:cclab","scott","tiger");
Statement s=con.createStatement();
ResultSet r=s.executeQuery("select * from cart");
out.println("<center> <table border=1>");
out.println("<thead> <th> Book name </th> <th> Price </th> <th> Quantity </th> <th>
```

```
Amount </th> </thead>");
while(r.next())
{
out.println("<tr> <td> "+r.getString(1)+"</td> ");
out.println("<td> "+r.getString(2)+"</td> ");
out.println("<td> "+r.getInt(3)+"</td> ");
out.println("<td> "+r.getString(4)+"</td> </tr>");
}
out.println("</table></center>");
con.close();
}
catch(SQLException sq)
{
out.println("sql exception"+sq);
}

catch(ClassNotFoundException cl)
{
out.println("class not found"+cl);
}
}
}
```

web.xml:

```
<web-app>
<servlet>
<servlet-name>set</servlet-name>
<servlet-class>Cartenter</servlet-class>
</servlet>
<servlet>
<servlet-name>display</servlet-name>
<servlet-class>Retrieve</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>set</servlet-name>
<url-pattern>/enterdata</url-pattern>
```

</servlet-mapping>

<servlet-mapping>

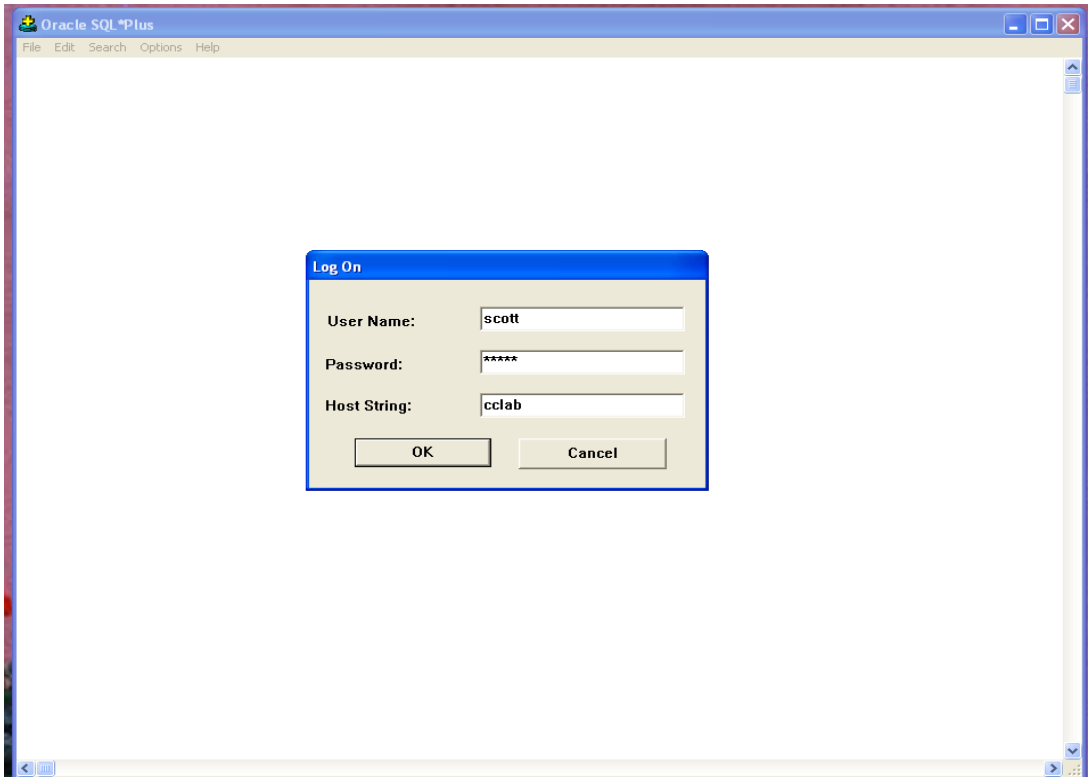
<servlet-name>display</servlet-name>

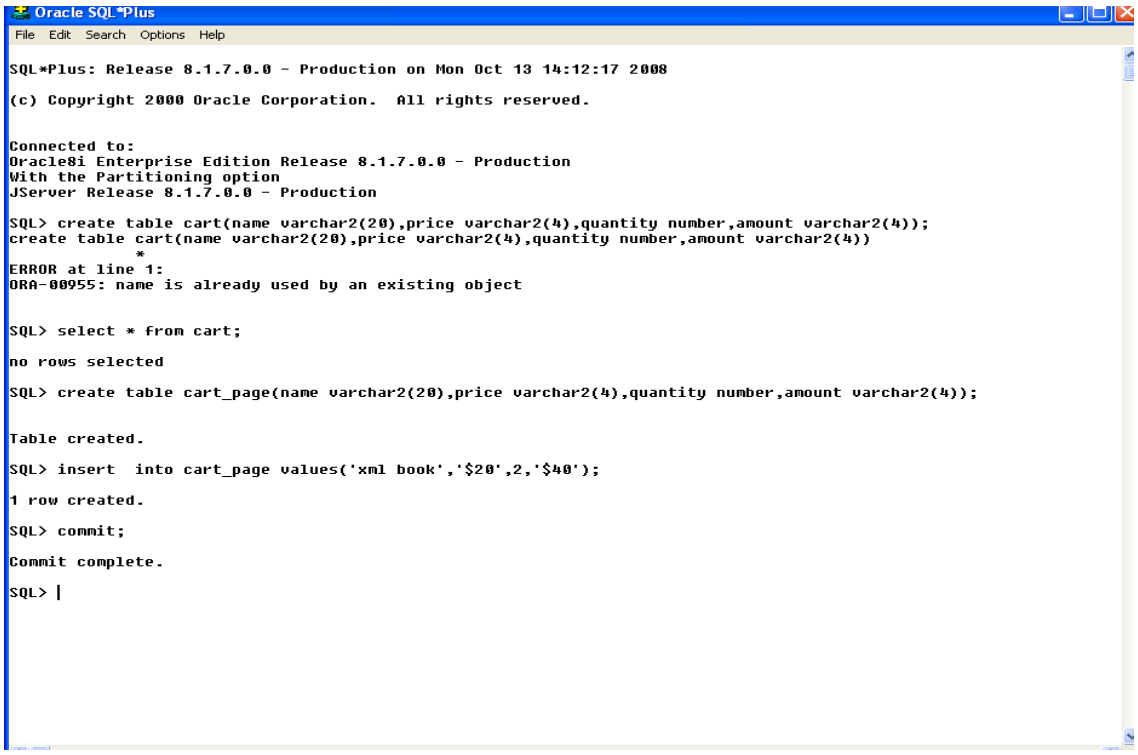
<url-pattern>/display1</url-pattern>

</servlet-mapping>

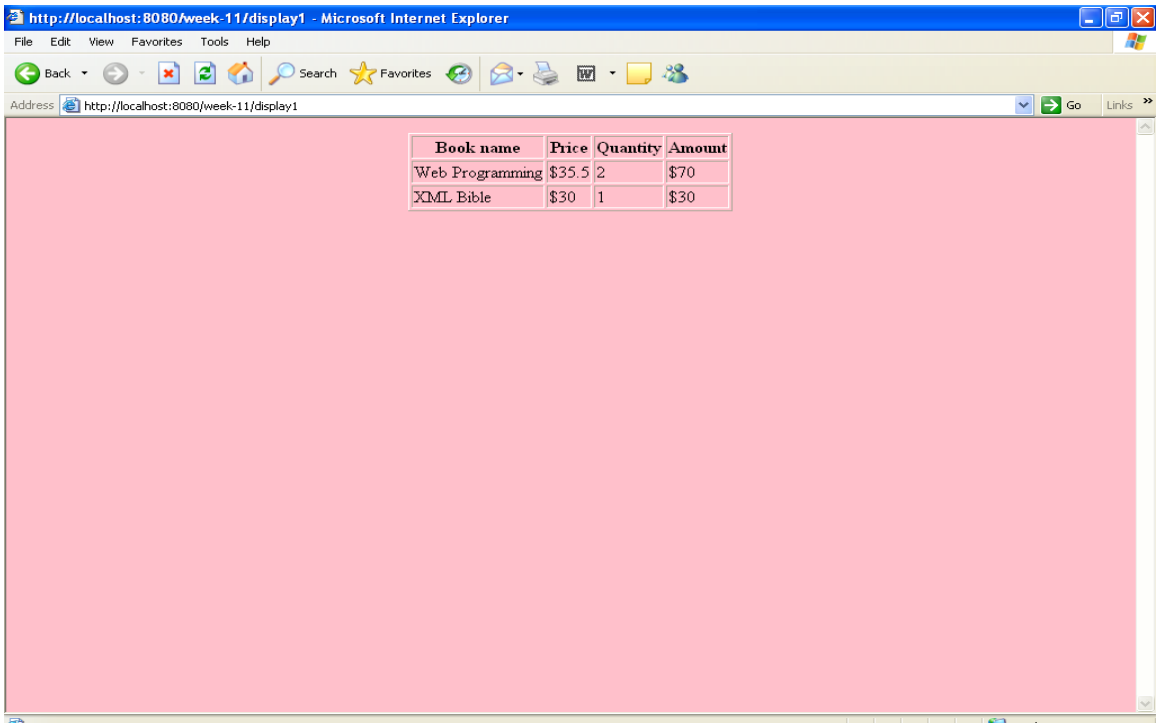
</web-app>

CREATE THE TABLE AND INSERT VALUES INTO THE TABLE:





OUTPUT:



RESULT:

The data is extracted from the tables and displayed in the catalogue page using JDBC

LAB VIVA QUESTIONS & ANSWERS

1. What is a ResultSet?

These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data. The java.sql.ResultSet interface represents the result set of a database query.

2. What are the basic steps to create a JDBC application?

Following are the basic steps to create a JDBC application:

1. Import packages containing the JDBC classes needed for database programming.
2. Register the JDBC driver, so that you can open a communications channel with the database.
3. Open a connection using the DriverManager.getConnection () method.
4. Execute a query using an object of type Statement.
5. Extract data from result set using the appropriate ResultSet.getXXX () method.
6. Clean up the environment by closing all database resources relying on the JVM's garbage collection.

3. What are JDBC driver types?

There are four types of JDBC drivers:

1. JDBC-ODBC Bridge plus ODBC driver, also called Type 1: calls native code of the locally available ODBC driver.
2. Native-API, partly Java driver, also called Type 2: calls database vendor native library on a client side. This code then talks to database over network.
3. JDBC-Net, pure Java driver, also called Type 3 : the pure-java driver that talks with the server-side middleware that then talks to database.
4. Native-protocol, pure Java driver, also called Type 4: the pure-java driver that uses database native protocol.

4. Which type of JDBC driver is the fastest one?

JDBC Net pure Java driver(Type 4) is the fastest driver because it converts the JDBC calls into vendor specific protocol calls and it directly interacts with the database.

5. What are the different types of JDBC Statements?

Types of statements are:

- Statement (regular SQL statement)
- PreparedStatement (more efficient than statement due to pre-compilation of SQL)
- CallableStatement (to call stored procedures on the database)

Laboratory Name : **WEB TECHNOLOGIES**	Experiment No:_____**12**_____

AIM: Modify cart JSPpage to achieve the dynamism with the HTTP protocol and session management

DESCRIPTION:

HTTP is a stateless protocol. Session is required to maintain the state.

Methods of session Object:

There are numerous methods available for session Object. Some are:

- getAttribute(String name)
- getAttributeNames
- isNew()
- getCreationTime
- getId
- invalidate()
- getLastAccessedTime
- getMaxInactiveInterval
- removeAttribute(String name)
- setAttribute(String, object)

The <jsp:include> element allows you to include either static and dynamic files in a JSP file.

Program :

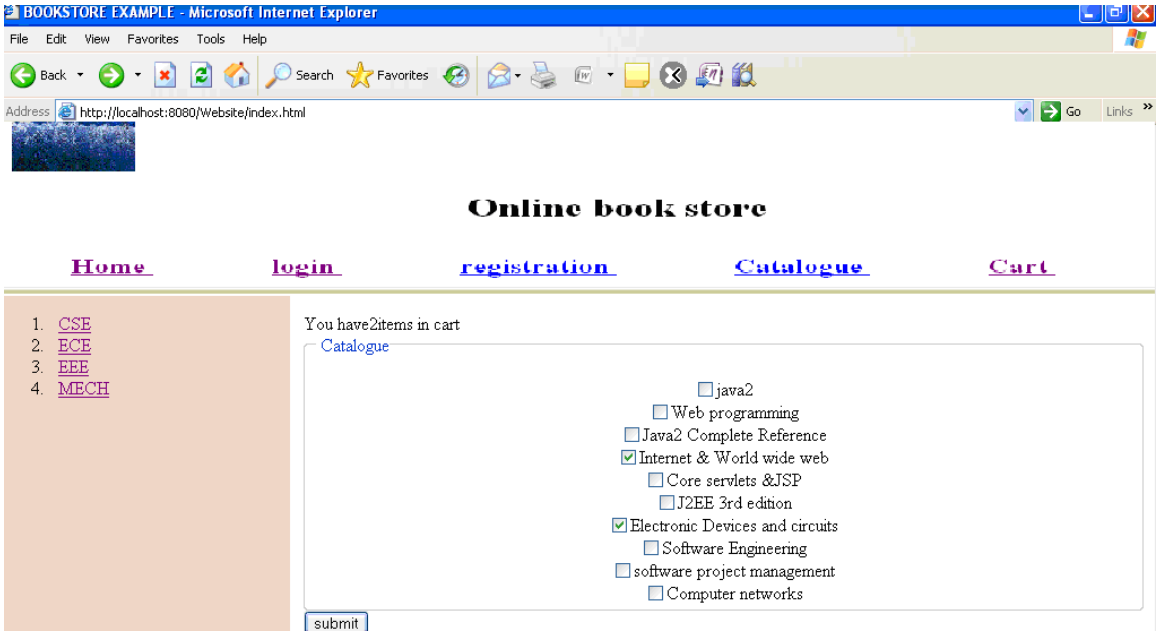
Cart .java

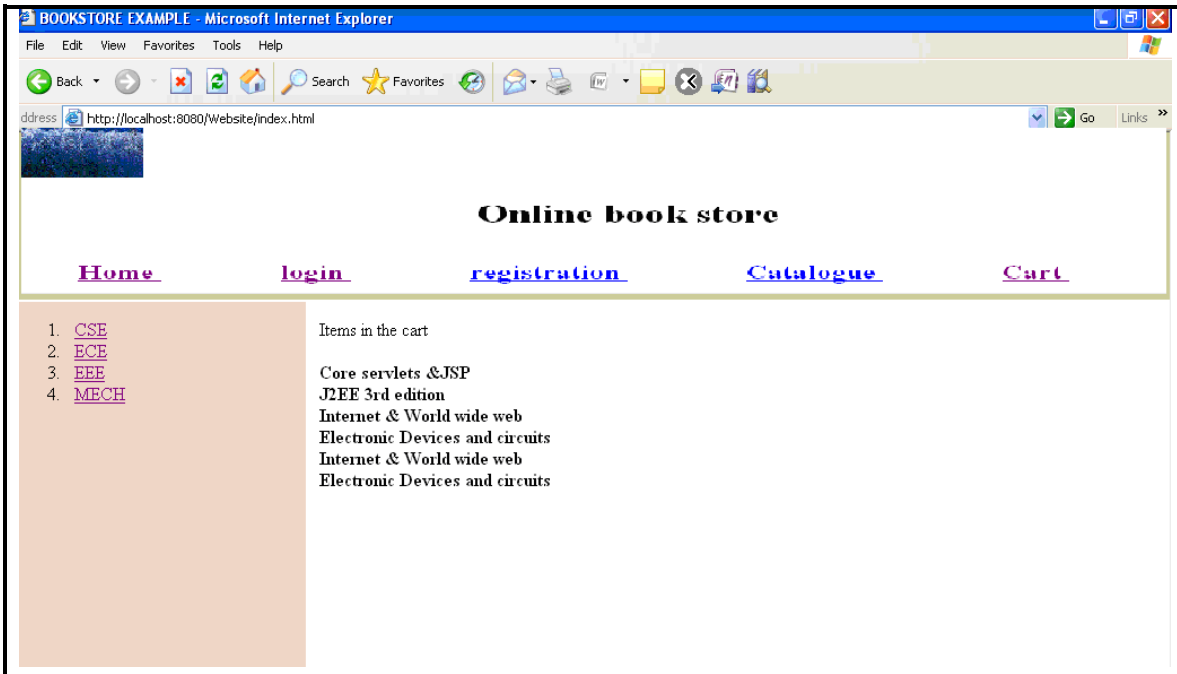
```
import java.util.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class Cart extends HttpServlet
        {

        public void init(ServletConfig config) throws ServletException
        {
        super.init(config);

        }
        public void service(HttpServletRequest req,HttpServletResponse res) throws
IOException,ServletException
        {
        res.setContentType("text/html");
      PrintWriter pw=res.getWriter();
```

```
        HttpSession hs=req.getSession();
        ArrayList cart=(ArrayList)hs.getAttribute("cart");
        if(cart==null)
        {
    pw.println("No items in your cart");
        cart=new ArrayList();
        hs.setAttribute("cart",cart);
        }
        String itemselected[];
        String item;
        itemselected=req.getParameterValues("book");

    if(itemselected!=null)
        {
        for(int i=0;i<itemselected.length;i++)
        {
        item=itemselected[i];
     cart.add(item);

        }}

        pw.println("Items in the cart<br>");
        Iterator it=cart.iterator();
        while(it.hasNext())
        {
        pw.println("<br><b>"+it.next()+"</b>");
        }


}
}

Catalogue.java

import java.util.*;
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class Catalogue extends HttpServlet
        {
        int itemcount=0;
        public void init(ServletConfig config) throws ServletException
        {
        super.init(config);

        }
        public void service(HttpServletRequest req,HttpServletResponse res) throws
IOException,ServletException
        {
        res.setContentType("text/html");
      PrintWriter pw=res.getWriter();
        HttpSession hs=req.getSession();
  ArrayList cart=(ArrayList)hs.getAttribute("cart");
    if(cart!=null)
```

```
        {
        itemcount=cart.size();
}

     pw.println("You have"+itemcount+"items in cart");


       pw.println("<body><center><fieldset><legend>Catalogue</legend><form
action='cart' method='get'><input type='checkbox' value='Java2'
name='book'>java2</input><br>");
        pw.println("<input type='checkbox' name='book' value='web
programming'>Web programming</input><br>");
        pw.println("<input type='checkbox' name='book' value='Java2 Complete
Reference'>Java2 Complete Reference</input><br>");
        pw.println("<input type='checkbox' name='book' value='Internet & World wide
web'>Internet & World wide web</input><br>");
        pw.println("<input type='checkbox' name='book' value='Core servlets
&JSP'>Core servlets &JSP</input><br>");
        pw.println("<input type='checkbox' name='book' value='J2EE 3rd
edition'>J2EE 3rd edition</input><br>");
        pw.println("<input type='checkbox' name='book' value='Electronic Devices and
circuits'>Electronic Devices and circuits</input><br>");
        pw.println("<input type='checkbox' name='book' value='Software
Engineering'>Software Engineering</input><br>");
        pw.println("<input type='checkbox' name='book' value='software project
management'>software project management</input><br>");
        pw.println("<input type='checkbox' name='book' value='Computer
networks'>Computer networks</input><br>");
        pw.println("</fieldset></center>");
        pw.println("<input type='submit' value='submit'>");




}
}

<web-app>

<servlet>
<servlet-name>Login</servlet-name>
<servlet-class>Login</servlet-class>
<init-param>
<param-name>username</param-name>
<param-value>syam</param-value>

</init-param>
<init-param>
<param-name>password</param-name>
<param-value>syam</param-value>

</init-param>

</servlet>
```

```
<servlet-mapping>
<servlet-name>Login</servlet-name>
<url-pattern>/login.do</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>cat</servlet-name>
<servlet-class>Catalogue</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>cat</servlet-name>
<url-pattern>/Cat</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>cart</servlet-name>
<servlet-class>Cart</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>cart</servlet-name>
<url-pattern>/cart</url-pattern>
</servlet-mapping>
</web-app>
```





Result : Thus the session management is successfully created using jsp

LAB VIVA QUESTIONS & ANSWERS

1. What is HTTP?

HTTP is called a stateless protocol because each command is executed independently, without any knowledge of the commands that came before it.

2. What is HTTP session token?

A session token is a unique identifier that is generated and sent from a server to a client to identify the current interaction session. The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries.

3. Abbreviate HTTP?

HTTP is HyperText Transfer Protocol.

4. What are the standards actions available in jsp?

The standards actions include:

- <jsp:include>
- <jsp:forward>
- <jsp:useBean>
- <jsp:setProperty>
- <jsp:getProperty>
- <jsp:param>
- <jsp:plugin>

5. What is the scope available in <jsp: useBean>?

Scope includes:

- Page scope
- Request scope
- application scope
- session scope