JSP

====

JSP stands for Java Server Pages.

JSP is a dynamic web resource program which is used to develop web applications.

JSP is used for presentation layer/logic.

Limitations with Servlets

=========================

> To work with servlets strong java knowledge is required.

> It is not suitable for non-java programmers.

> It does not give any implicit oject.
  (Object which can be used directly without any configuration is called implicit object)

> Configuration of each servlet program in web.xml file is mandatory.

> Handling exceptions are mandatory.

> We can't maintain HTML code and java code seperately.

To overcome this limitations Sun Micro System introduced JSP.

Advantages of JSP

==================

> To work with JSP strong java knowledge is not required.

> It is suitable for java and non-java programmers.

> It gives 9 implicit objects.

> Configuration of jsp program in web.xml file is not mandatory.

> Handling exceptions are optional.

> We can maintain HTML code and Java code seperately.

> It supports tag based language.

> It allows us to work with custom tags.

> It gives all the features of servlets.

First web application development having JSP as dynamic web resource program

==

Deployment Directory structure

```
JspApp1
|
|---Java Resources
|
|---Web Content
          |
          |---ABC.jsp
          |
          |---WEB-INF
                    |
                    |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

Servlet container will execute servlet program.

JSP container will execute jsp program.

But to execute JSP program jsp container will take the support of servlet container.

ABC.jsp

```
<center>
          <h1>
                    Current Date and Time : <br>

                    <%
                              java.util.Date d=new java.util.Date();
                              out.println(d);
```

```
			%>
		</h1>
	</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <welcome-file-list>
		<welcome-file>ABC.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

Request url

----

	http://localhost:2525/JspApp1/

Note:

To see the output in JSP we need to replace ecj-4.2.2.jar file in Tomcat/lib folder.

ex:

http://www.java2s.com/Code/Jar/e/Downloadecj422jar.htm

Configuration of JSP program

============================

Deployment Directory structure

```
JspApp1
|
|---Java Resources
|
|---Web Content
            |
            |---ABC.jsp
            |
            |-----WEB-INF
                        |
                        |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

ABC.jsp

```
<center>
            <h1>
                        Current Date and Time : <br>
```

```
                <%
                        java.util.Date d=new java.util.Date();
                        out.println(d);
                %>
        </h1>
</center>
```

web.xml

```
        <web-app>
                <servlet>
                        <servlet-name>ABC</servlet-name>
                        <jsp-file>/ABC.jsp</jsp-file>
                </servlet>
                <servlet-mapping>
                        <servlet-name>ABC</servlet-name>
                        <url-pattern>/test</url-pattern>
                </servlet-mapping>
        </web-app>
```

Request url

```
        http://localhost:2525/JspApp1/ABC.jsp
        http://localhost:2525/JspApp1/test
```

How can we hide our web application accessible through file name.It means how can we access our web application accessible only by using url pattern

==

In order to access our web application by using url pattern we need to keep our ABC.jsp file inside "WEB-INF" folder.

Deployment Directory structure

```
JspApp1
|
|---Java Resources
|
|---Web Content
            |
            |
            |-----WEB-INF
                        |
                        |----web.xml
                        |
                        |----ABC.jsp
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

ABC.jsp

```
<center>
```

```
        <h1>
                Current Date and Time : <br>

                <%
                        java.util.Date d=new java.util.Date();
                        out.println(d);
                %>
        </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
                        <servlet-name>ABC</servlet-name>
                        <jsp-file>/WEB-INF/ABC.jsp</jsp-file>
                </servlet>
                <servlet-mapping>
                        <servlet-name>ABC</servlet-name>
                        <url-pattern>/test</url-pattern>
                </servlet-mapping>

</web-app>
```

Request url

----

    http://localhost:2525/JspApp1/ABC.jsp --> 404 Error

    http://localhost:2525/JspApp1/test   --> valid

JSP life cycle methods

=====================

We have three life cycle methods in JSP.

1) _jspInit()

It is used for instantitation event.

This method will execute just before JES class object creation.

Here JES stands for Java Equivalent Servlet.

2) _jspService()

It is used for request arrival event.

This method will execute when request goes to JSP program.

3) _jspDestroy()

It is used for destruction event.

This method will execute just before JES class object destruction.

Phases in JSP

==============

We have two phases in JSP.

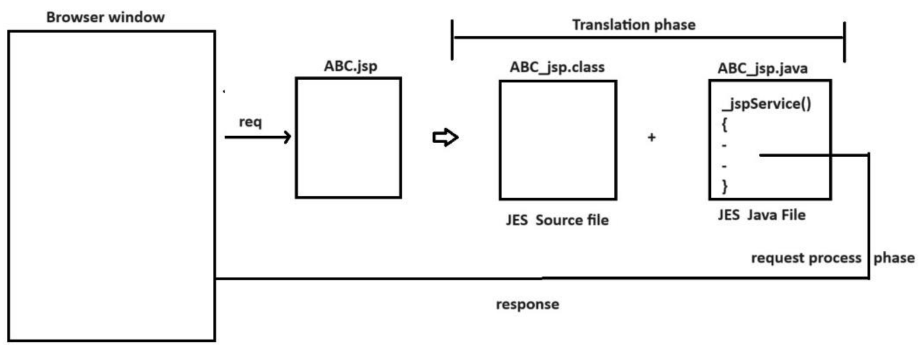
1) Translation phase

In translation phase our JSP program will convert to JES class.

2) Request Processing phase

In request processing phase our JES class will executed and result will send to browser window as dynamic response.

Diagram: jsp2.1



Q)What is <load-on-startup> and what happens if we enable <load-on-startup> ?

We can enable <load-on-startup> inside web.xml file.

web.xml

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


            <servlet>
                  <servlet-name>ABC</servlet-name>
                  <jsp-file>/WEB-INF/ABC.jsp</jsp-file>
                  <load-on-startup>1</load-on-startup>
            </servlet>
            <servlet-mapping>
                  <servlet-name>ABC</servlet-name>
                  <url-pattern>/test</url-pattern>
            </servlet-mapping>


</web-app>
```

If we enable <load-on-startup> then our servlet container performs translation phase during the server startup or during the deployment of web application.

In short, if we enable <load-on-startup> then our servlet container creates JES class object before we give the request.

JSP Tags/Elements

================

We have following important tags in jsp.

1) Scripting tags

----

i) scriptlet tag

ex:

```
<%   code here %>
```

ii) expression tag

ex:

```
<%=  code here %>
```

iii)declaration tag

ex:

```
<%!  code here %>
```

2) Directive tags

----

i) page directive

ex:

```
<%@page  attribute=value %>
```

ii) include directive

ex:

```
<%@include  attribute=value %>
```

3) Standard tags

```
<jsp:include>

<jsp:forward>
```

```
            <jsp:useBean>
            <jsp:setProperty>
            <jsp:getProperty>
            and etc.
```

JSP comments

```
            <%--  comment here --%>
```

i) scriptlet tag

================

It is used to declare java code.

syntax:

```
            <%  code here %>
```

Deployment Directory structure

```
JspApp2
|
|---Java Resources
|
|---Web Content
            |
            |----form.html
            |
            |----process.jsp
            |
            |------WEB-INF
                        |
                        |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">

            Name: <input type="text" name="t1"/> <br>

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

</form>
```

process.jsp

```
<center>
<h1>
<%
            String name=request.getParameter("t1");
            out.println("Welcome :"+name);
```

```
%>
</h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

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

</web-app>
```

Request url

```
        http://localhost:2525/JspApp2/
```

ii) expression tag

==================

The code which is written in expression tag will return to the output stream of a response.It means we don't need to write out.println() to print/display the data.

Expression tag does not support semicolon.

syntax:

```
        <%=  code here %>
```

Deployment Directory structure

```
JspApp2
|
|---Java Resources
|
|---Web Content
        |
        |----form.html
        |
        |----process.jsp
        |
        |------WEB-INF
                |
                |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">

        Name: <input type="text" name="t1"/> <br>

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

</form>
```

process.jsp

```
<center>
<h1>
<%
        String name=request.getParameter("t1");
%>
<%=  "Hello Bro :"+name %>
</h1>
</center>
```

web.xml

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

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

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

</web-app>
```

Request url

```
        http://localhost:2525/JspApp2/
```

iii) Declaration tag

====================

Declaration tag is used to declare fields and methods.

syntax:

```
        <%!  code here %>
```

Deployment Directory structure

```
JspApp3
|
|---Java Resources
|
|---Web Content
      |
      |----index1.jsp
      |----index2.jsp
      |
      |-----WEB-INF
                |
                |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index1.jsp

```
<center>
<h1>
<%!
          int i = 10;
%>

<%=  "The value is ="+i %>
</h1>
</center>
```

index2.jsp

```
<center>
```

```
<h1>
        <%! int cube(int n)
                {
                        return n*n*n;
                }
        %>
        <%= "Cube of a given number is ="+cube(5)  %>
</h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


</web-app>
```

Request url

```
        http://localhost:2525/JspApp3/index1.jsp
        http://localhost:2525/JspApp3/index2.jsp
```

Exception Handling in JSP

=========================

Runtime errors are known as exceptions.

Exception raised at runtime so they are also known as runtime events.

Exception may raise anytime in our web application so handling the exception is a safer side for the programmer.

There are two to handle exceptions in JSP.

1) Using errorPage and isErrorPage attribute of page directive tag.

2) Using <error-page> element in web.xml file.

1) Using errorPage and isErrorPage attribute of page directive tag

----

Deployment Directory structure

```
JspApp4
|
|---Java Resources
|
|---Web Content
        |
        |---form.html
        |
        |---process.jsp
```

```
            |
            |---error.jsp
            |
            |------WEB-INF
                        |
                        |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">

            No1: <input type="text" name="t1"/> <br>

            No2: <input type="text" name="t2"/> <br>

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

</form>
```

process.jsp

```
<%@page errorPage="error.jsp" %>
<%
```

```
        String sno1=request.getParameter("t1");
        String sno2=request.getParameter("t2");

        int a=Integer.parseInt(sno1);
        int b=Integer.parseInt(sno2);

        int c=a/b;
%>
<%= "Division of two numbers is ="+c %>
```

error.jsp

```
<%@page  isErrorPage="true" %>
<b><i>
        Sorry! Exception occured
        <br>
        <%= exception %>
</i></b>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <welcome-file-list>
```

```
            <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>
```

Request url

```
            http://localhost:2525/JspApp4/
```

2) Using <error-page> element in web.xml file

This approach is recommanded to use because we don't need to define errorPage attribute in each JSP file.Defining single entry in web.xml file will handle all types of exceptions.

Deployment Directory structure

```
JspApp4
|
|---Java Resources
|
|---Web Content
            |
            |---form.html
            |
            |---process.jsp
            |
            |---error.jsp
            |
            |------WEB-INF
                      |
                      |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">

          No1: <input type="text" name="t1"/> <br>

          No2: <input type="text" name="t2"/> <br>

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

</form>
```

process.jsp

```
<%
          String sno1=request.getParameter("t1");
          String sno2=request.getParameter("t2");
```

```
        int a=Integer.parseInt(sno1);

        int b=Integer.parseInt(sno2);


        int c=a/b;

%>

<%= "Division of two numbers is ="+c %>
```

error.jsp

```
<%@page  isErrorPage="true" %>

<b><i>

        Sorry! Exception occured

        <br>

        <%= exception %>

</i></b>
```

web.xml

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


  <error-page>

        <exception-type>java.lang.Exception</exception-type>

        <location>/error.jsp</location>
  </error-page>

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

</web-app>
```

Request url

```
        http://localhost:2525/JspApp4/
```

JSP to Database Communication

============================

Deployment Directory Structure

```
JspApp5
|
|---Java Resources
|
|---Web Content
        |
        |----form.html
        |----process.jsp
        |
        |----WEB-INF
```

```
                    |
                    |------web.xml
                    |
                    |-------lib
                              |
                              |---ojdbc14.jar
```

Note:

In above application we need to add "servlet-api.jar" and "ojdbc14.jar" file in project build path.

Copy and paste "ojdbc14.jar" file inside "WEB-INF/lib" folder seperately.

form.html

```
<form action="process.jsp">

        No: <input type="text" name="t1"/> <br>

        Name: <input type="text" name="t2"/> <br>

        Address: <input type="text" name="t3"/> <br>

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

</form>
```

process.jsp

```
<%@page  import="java.sql.*" buffer="8kb"  language="java" %>

<%
	String sno=request.getParameter("t1");
	int no=Integer.parseInt(sno);

	String name=request.getParameter("t2");
	String add=request.getParameter("t3");

	//storing the data in a database
	Connection con=null;
	PreparedStatement ps=null;
	int result=0;
	String qry=null;
	try
	{
		Class.forName("oracle.jdbc.driver.OracleDriver");

	con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","ad
min");
		qry="insert into student values(?,?,?)";
		ps=con.prepareStatement(qry);
		//set the values
		ps.setInt(1,no);
		ps.setString(2,name);
```

```
                ps.setString(3,add);

                //execute

                result=ps.executeUpdate();

                if(result==0)

                        out.println("No Record Inserted");

                else

                        out.println("Record Inserted");


                ps.close();

                con.close();

        }

        catch(Exception e)

        {

                out.println(e);

        }


%>
```

web.xml

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


  <welcome-file-list>

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

</web-app>
```

Request url

----

http://localhost:2525/JspApp5/

Action Tags

===========

Action tags are used to perform perticular task.

Action tags are used to control the flow of web pages and uses java beans.

Action tags are executed dynamically at runtime.

Action tags contains only xml tags and do not have any standard tags.

Action tags are divided into two types.

1) Standard Action Tags

2) Custom Action Tags

1) Standard Action Tags

Built-In tags are called standard action tags.

ex:

    <jsp:forward>

    <jsp:include>

    <jsp:useBean>

    <jsp:setProperty>

    <jsp:getProperty>

    and etc.

Action forward

================

In action forward, Output of source jsp program will be discarded and output of destination jsp program goes to browser window as dynamic response.

It internally uses Servlet API functionality called rd.forward(req,res).

syntax:

```
	<jsp:forward  page="page_name"/>
```

Deployment Directory structure

```
JspApp6
|
|--Java Resources
|
|--Web Content
	|
	|----A.jsp
```

```
        |----B.jsp
        |
        |-----WEB-INF
                |
                |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

A.jsp

```
<b><i> Beging of A.jsp File</i></b>
<br>
<jsp:forward page="B.jsp"/>
<br>
<b><i> Ending of A.jsp File</i></b>
```

B.jsp

```
<b><i> This is B.jsp File </i></b>
```

web.xml

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

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


  <welcome-file-list>
        <welcome-file>A.jsp</welcome-file>
  </welcome-file-list>


</web-app>
```

Request url

```
        http://localhost:2525/JspApp6/
```

Action include

===============

In action include, output of source JSP program and output of destination jsp program combibely goes to browser window as dynamic response.

It internally uses servlet API functionality called rd.include(req,res).

syntax:

```
        <jsp:include page="page_name"/>
```

Deployment Directory structure

JspApp6

```
|
|--Java Resources
|
|--Web Content
            |
            |----A.jsp
            |----B.jsp
            |
            |-----WEB-INF
                        |
                        |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

A.jsp

```
<b><i> Beging of A.jsp File</i></b>
<br>
<jsp:include page="B.jsp"/>
<br>
<b><i> Ending of A.jsp File</i></b>
```

B.jsp

```
<b><i> This is B.jsp File </i></b>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <welcome-file-list>
          <welcome-file>A.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

Request url

```
http://localhost:2525/JspApp6/
```

JSP to Java Bean Communication

==============================

JSP to Java Bean communication is possible by using three tags.

1) <jsp:useBean> tag

It is used to create and locate bean class object.

2) <jsp:setProperty> tag

It is used to set the values to bean object and calls setter methods.

3) <jsp:getProperty> tag

It is used to get the values from bean object and calls getter methods.

Note:

All above tags are independent tags.

ex:1

Deployment Directory Structure

```
JspApp7
|
|---Java Resources
          |
          |------src
                    |
                    |---com.ihub.www
                              |
                              |----CubeNumber.java
|
|---Web Content
          |
          |----index.jsp
          |
          |----WEB-INF
                    |
                    |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.jsp

```
<jsp:useBean id="cn" class="com.ihub.www.CubeNumber"></jsp:useBean>


<center>
<h1>
            <%= "Cube Of a Given Number Is = "+cn.cube(5) %>
</h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


  <welcome-file-list>
```

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

</web-app>
```

CubeNumber.java

```
---------------
package com.ihub.www;

public class CubeNumber
{
        public int cube(int n)
        {
                return n*n*n;
        }
}
```

Request url

```
----------
        http://localhost:2525/JspApp7/
```

2) Custom Action Tags

====================

To create custom tags in JSP we need to use taglib directory.

We can declare taglib directory as follow.

syntax:

```
        <%@taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>
```

Deployment Directory structure

```
------------------------------
JspApp9
|
|---Java Resources
        |
        |------src
                |
                |----com.ihub.www
                        |
                        |----CubeNumber.java
|---Web Content
        |
        |-----process.jsp
        |
        |-----WEB-INF
                |
                |-------web.xml
                |
                |-------mytags.tld
```

```
                    |
                    |------lib
                              |
                              |----jsp-api.jar
```

Note:

In above application we need to add "servlet-api.jar" and "jsp-api.jar" file in project build path.

Copy and paste "jsp-api.jar" file inside "WEB-INF/lib" folder seperately.

process.jsp

```
<%@taglib uri="/WEB-INF/mytags.tld" prefix="ihub" %>

<center>
          <h1>
                    Cube Of a Given number is : <ihub:cube  number="5"/>
          </h1>
</center>
```

mytags.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
      PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
```

```
    "http://java.sun.com/j2ee/dtd/web-jsptaglibrary_1_2.dtd">

<taglib>
 <tlib-version>1.0</tlib-version>
 <jsp-version>1.2</jsp-version>
 <short-name>simple</short-name>
 <uri>mytags</uri>
 <description>A simple tab library for the examples</description>

 <tag>
		<name>cube</name>
		<tag-class>com.ihub.www.CubeNumber</tag-class>
		<attribute>
			<name>number</name>
			<required>true</required>
		</attribute>
 </tag>


</taglib>
```

CubeNumber.java

```
package com.ihub.www;

import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;
```

```
public class CubeNumber extends TagSupport
{
        int number;

        //setter method
        public void setNumber(int number)
        {
                this.number=number;
        }

        public int doStartTag()
        {
                JspWriter out=pageContext.getOut();
                try
                {
                        out.println(number*number*number);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
                return SKIP_BODY;
        }
}
```

web.xml

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

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


  <welcome-file-list>

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

  </welcome-file-list>


</web-app>
```

Request url

```
            http://localhost:2525/JspApp9/
```

Q)What is the difference between HTML and JSP?

HTML	JSP
It stands for Hypertext Markup Language.	It stands for Java Server Pages.
It is a static web resource program.	It is a dynamic web resource program.

It is used to create static web pages.	It is used to create dynamic web pages.
It executes on browser window.	It executes on server.
It is used to create client side components.	It is used to create server side components.
It does not support custom tags.	It supports custom tags.
We need to save html document either with .html or .htm extension.	We need to save jsp file with .jsp extension.

ex:2

Deployment Directory Structure

```
JspApp8
|
|---Java Resources
            |
            |------src
                        |
                        |---com.ihub.www
                                    |
                                    |----User.java
|
|---Web Content
            |
            |----form.html
```

```
          |
          |----process.jsp
          |
          |----WEB-INF
                    |
                    |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="process.jsp">

          UserName: <input type="text" name="username"/> <br>

          Password: <input type="password" name="password"/> <br>

          Email: <input type="text" name="email"/> <br>

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

web.xml

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

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


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


</web-app>
```

process.jsp

```
<jsp:useBean id="u" class="com.ihub.www.User"></jsp:useBean>


<jsp:setProperty property="*" name="u"/>


Records are <br>


<jsp:getProperty property="username" name="u"/> <br>
<jsp:getProperty property="password" name="u"/> <br>
<jsp:getProperty property="email" name="u"/> <br>
```

User.java

```
package com.ihub.www;


public class User
```

```
{
	private String username;
	private String password;
	private String email;

	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
}
```

Request url

----

http://localhost:2525/JspApp8/

MVC In JSP

==========

MVC stands for Model View Controller.

It is one of the design pattern which seperates business logic , persistence logic and data.

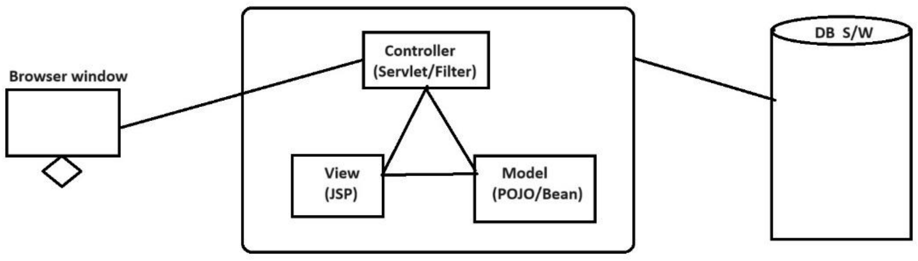
Controller is an interface between Model and View.

Controller will intercept all incoming request.

Model contains business logic and data.

View contains presentation i.e UI.

Diagram: jsp5.1



MVC Architecture

Deployment Directory structure

```
MVCApp
|
|---Java Resources
|          |
           |-------src
                     |
                     |---com.ihub.www
                               |
                               |--LoginSrv.java
                               |--LoginBean.java
|---Web Content
           |
           |----form.html
           |----view.jsp
           |----error.jsp
           |
           |-----WEB-INF
                     |
                     |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test" method="POST">

        <table align="center">
                <tr>
                        <td>Username:</td>
                        <td><input type="text" name="username"/></td>
                </tr>
                <tr>
                        <td>Password:</td>
                        <td><input type="password" name="password"/></td>
                </tr>
                <tr>
                        <td><input type="reset" value="reset"/></td>
                        <td><input type="submit" value="submit"/></td>
                </tr>
        </table>

</form>
```

web.xml

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

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
        <servlet-name>LoginSrv</servlet-name>
        <servlet-class>com.ihub.www.LoginSrv</servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>LoginSrv</servlet-name>
        <url-pattern>/test</url-pattern>
  </servlet-mapping>

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

</web-app>
```

LoginBean.java

```
package com.ihub.www;

public class LoginBean
{
        private String username;
        private String password;
```

```
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}


}

LoginSrv.java
-----------
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginSrv extends HttpServlet
{
	protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//reading form data
		String name=req.getParameter("username");
		String pass=req.getParameter("password");

		//create bean object and set the values
		LoginBean lb=new LoginBean();
		lb.setUsername(name);
		lb.setPassword(pass);

		//add the bean object to request
		req.setAttribute("bean",lb);

		if(pass.equals("admin"))
		{
			RequestDispatcher rd=req.getRequestDispatcher("view.jsp");
			rd.forward(req,res);
```

```
                }
                else
                {
                        RequestDispatcher rd=req.getRequestDispatcher("error.jsp");
                        rd.forward(req,res);
                }

                pw.close();
        }
}
```

view.jsp

```
<%@page  import="com.ihub.www.LoginBean" %>

<%
        LoginBean lb=(LoginBean)request.getAttribute("bean");
%>
<%=  "UserName is = "+lb.getUsername() %> <br>
<%=  "Password is = "+lb.getPassword() %> <br>
```

error.jsp

```
<center>
        <b style="color:red">
```

```
                    Sorry! Incorrect username or password
          </b>
</center>
<%@include file="form.html" %>
```

Request url

----

http://localhost:2525/MVCApp/

Implicit objects in JSP

=======================

Object which can be used directly without any configuration is called implicit object.

Implicit objects are created by the web container which are available for every JSP page.

We have 9 implicit objects in JSP.

ex:

Object	Type
out	JspWriter
request	HttpServletRequest
response	HttpServletResponse
config	ServletConfig
application	ServletContext
session	HttpSession
pageContext	pageContext

page	Object
exception	Throwable

response object

================

In jsp, response is a implicit object of type HttpServletResponse.

It can be used to add or manipulate response such as redirect response or another resources,send error and etc.

Deployment Directory structure

```
JspApp10
|
|--Java Resources
|
|--Web Content
          |
          |---index.html
          |
          |---process.jsp
          |
          |-----WEB-INF
                    |
                    |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
	<h1>
			<a href="process.jsp"> Facebook </a>
	</h1>
</center>
```

process.jsp

```
<%
	response.sendRedirect("https://www.facebook.com/login");
%>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

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

```
  </welcome-file-list>

</web-app>
```

request url

----

	http://localhost:2525/JspApp10/

config object

============

It is an implicit object of type ServletConfig.

The config object is created by web container for each jsp page.

This object is used to read initialized parameters for a perticular jsp page.

Deployment Directory structure

```
JspApp11
|
|--Java Resources
|
|--Web Content
          |
          |---index.html
          |
          |---process.jsp
          |
          |-----WEB-INF
                  |
                  |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
        <h1>
                        <a href="test"> click Here </a>


        </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
 <servlet>
        <servlet-name>ABC</servlet-name>
        <jsp-file>/process.jsp</jsp-file>
        <init-param>
                <param-name>driver</param-name>
                <param-value>oracle.jdbc.driver.OracleDriver</param-value>
        </init-param>
 </servlet>

 <servlet-mapping>
        <servlet-name>ABC</servlet-name>
        <url-pattern>/test</url-pattern>
 </servlet-mapping>



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

</web-app>
```

process.jsp

```
<%
        String value=config.getInitParameter("driver");
%>
```

<%= value %>

Request url

----

    http://localhost:2525/JspApp11

application object

=================

In jsp, application is an implicit object of type ServletContext.

This instance of ServletContext is created only once by the web container.

This object is used to read initialized parameters from configuration file web.xml file.

Deployment Directory structure

```
JspApp12
|
|--Java Resources
|
|--Web Content
          |
          |---index.html
          |
          |---process.jsp
          |
          |-----WEB-INF
                  |
                  |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
        <h1>
                        <a href="test"> click Here </a>


        </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">


  <servlet>
        <servlet-name>ABC</servlet-name>
```

```
            <jsp-file>/process.jsp</jsp-file>

    </servlet>

    <servlet-mapping>
            <servlet-name>ABC</servlet-name>
            <url-pattern>/test</url-pattern>
    </servlet-mapping>

    <context-param>
            <param-name>driver</param-name>
            <param-value>oracle.jdbc.driver.OracleDriver</param-value>
    </context-param>



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

</web-app>

process.jsp
-------------

<%
            String value=application.getInitParameter("driver");
%>
```

<%= value %>

Request url

```
        http://localhost:2525/JspApp12
```

session object

==============

In JSP, session is an implicit object of type HttpSession.

It is used to get or set the session formation.

Deployment Directory structure

```
JspApp13
|
|--Java Resources
|
|--Web Content
        |
        |---form.html
        |
        |---first.jsp
        |
        |---second.jsp
        |
        |-----WEB-INF
                    |
                    |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="first.jsp">

            Name: <input type="text" name="t1"/> <br>

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

first.jsp

```
<%
            String name=request.getParameter("t1");
            out.println("Welcome :"+name);

            session.setAttribute("pname", name);
```

```
%>
<br>
<center>
	<a href="second.jsp"> click for second.jsp </a>
</center>
```

second.jsp

```
<%
	String name=(String)session.getAttribute("pname");
	out.println("Hello :"+name);
%>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

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

</web-app>
```

request url

----

    http://localhost:2525/JspApp13/

pageContext object

====================

In jsp, pageContext is an implicit object of type pageContext class.

The pageContext object can be used to set ,get ,remove attributes from one the following scopes.

page

request

session

application

In JSP, page scope is a default scope.

Deployment Directory structure

JspApp14

|

```
|--Java Resources
|
|--Web Content
          |
          |---form.html
          |
          |---first.jsp
          |
          |---second.jsp
          |
          |-----WEB-INF
                    |
                    |----web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="first.jsp">

          Name: <input type="text" name="t1"/> <br>

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

first.jsp

```
<%
	String name=request.getParameter("t1");
	out.println("Welcome :"+name);

	pageContext.setAttribute("pname", name,pageContext.SESSION_SCOPE);
%>
<br>
<center>
	<a href="second.jsp"> click for second.jsp </a>
</center>
```

second.jsp

```
<%
	String name=(String)pageContext.getAttribute("pname",pageContext.SESSION_SCOPE);
	out.println("Hello :"+name);
%>
```

web.xml

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

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

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

</web-app>
```

request url

----

    http://localhost:2525/JspApp14/

Junit

=======

Junit is a unit testing framework.

Unit testing is important for TDD(Test Driven Development).

Unit testing is a process of checking a piece of code working as per the requirement or not.

To perform unit testing we need to create test suit or test cases.

Steps to perform unit testing

----

step1:

Launch eclipse IDE by choosing workspace location.

step2:

Create a java project i.e JunitProj.

ex:

File --> new --> project --> java project --> next -->

Project Name : JunitProj --> Next --> Finish.

step3:

Create a com.ihub.www package inside "src" folder.

ex:

Right click to src --> new --> package -->

Name: com.ihub.www -->finish

step4:

Create a java program i.e DemoApp.java.

ex:

right click to com.ihub.www --> new --> class -->

class name : DemoApp --> finish.

DemoApp.java

```
package com.ihub.www;
```

```
public class DemoApp
{
	public String concatination(String str1,String str2)
	{
		return str1+str2;
	}

	public int sum(int a,int b)
	{
		return a+b;
```

step5:

	Create Test Cases for java methods.

	ex:

		right click to DemoApp.java --> new --> other --> Junit --> test case

		---> next --> Next --> select methods(concatination and sum) --> finish -->ok.

DemoAppTest.java

```
package com.ihub.www;

import junit.framework.TestCase;

public class DemoAppTest extends TestCase
{
	public void testConcatination() {
		DemoApp da=new DemoApp();
		String result=da.concatination("ihub", "talent");
		assertEquals("ihubtalent", result);
	}

	public void testSum() {

		DemoApp da=new DemoApp();
		int result=da.sum(10, 20);
		assertEquals(30,result);
	}

}
```

step6:

Run the Junit test cases

ex:

right click to DemoAppTest.java --> run as --> junit test .

step7:

Green color indicates test cases are passed.

Brown color indicates test cases are failed.

Maven

=======

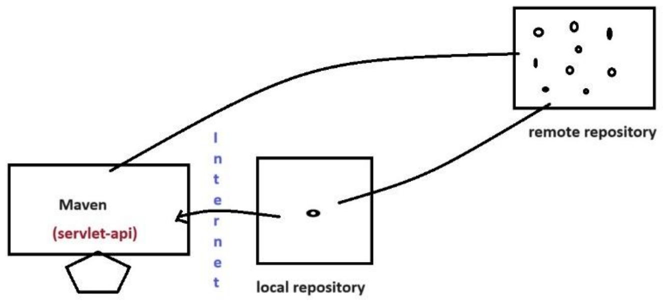
Maven is a project building management tool.

Maven project contains pom.xml file.

POM stands for Project Object Model.

A pom.xml file contains dependencies, plugins, Goals, packaging and etc.

Diagram: jsp6.1



Steps to develop Maven project

----

step1:

Launch eclipse IDE by choosing workspace location.

step2:

-----

create a dynamic web project.

ex:

File --> new --> dynamic project --> Name : MavenProj

---> Next --> Next --> generate web.xml file --> finish.

step3:

Convert dynamic web project to Maven project.

ex:

Right click to dynamic project --> configure -->

convert to maven project -->

Group ID : com.ihub.www

Artifact ID : MavenProj

Name : MavenProj

Description: Demostration on Maven project --> finish.

step4:

Create a "ABC.jsp" file inside "Web Content" folder.

ex:

ABC.jsp

```
<center>
```

```
                <h1>
                        This is Maven Project Demo
                </h1>
        </center>
```

step5:

Add "servlet-api.jar" file manven depedency inside pom.xml file.

ex:

pom.xml

```
-
-
 <dependencies>
        <dependency>
                <groupId>javax.servlet</groupId>
                <artifactId>servlet-api</artifactId>
                <version>2.5</version>
                <scope>provided</scope>
        </dependency>
</dependencies>
<build>
--
--
--
```

step6:

Run the maven project.

ex:

Right click to MavenProject --> run as --> run on server.

step7:

Test the application by using below request url.

ex:

http://localhost:2525/MavenProj/ABC.jsp

How to convert Maven project or Dynamic project to war file

==

step1:

Make sure Dynamic or Maven project is ready in eclipse IDE.

step2:

----

convert Dynamic or Maven project to war file .

ex:

Right click to MavenProj --> export --> war file -->

Destination : Desktop(choose) --> open→finish

Q) What is the difference between GIT and GITHUB ?

GIT	GITHUB
---------	---------
It is a distributed version control system which is used to track the changes in a file of a project.	It is a web-based hosting service for git.
It contains local repository.	It contains remote repository.
It is command based.	It is GUI.
It is installed locally.	It is hosted on web.

Q)Types of stages of Git?

We have three stages in git.

Working Directory :
----

the file exists, but is not part of git's version control.

staging area:
----

the file has been added to git's version control but changes
have not been committed

Repository:

----

the change has been committed

Diagram: git

Remote repository github

==========================

Remote Repository : https://github.com/NiyazulHasan/ih-java-025

Git software : https://git-scm.com/downloads

Q)Write a git command to initialized empty repository?

```
git init
```

Q)Write a git command to check the status?

```
git status
```

Q)Write a git command to check the branch?

```
git branch
```

Q)Write a git command to move from master branch to main branch?

```
git branch --move master main
```

Q)Write a git command to commit the changes?

```
git commit -m "comment here"
```

Q)Write a git command to add remote repository?

```
git remote add origin https://github.com/NiyazulHasan/practice
```

Q)Write a git command to check the remote repository

```
git remote -v
```

Q)Write a git command to push the code to remote origin?

```
git push -f origin main
```

Q)Write a git command to clone the project?

```
git clone <url>
```

Q)Write a git command to pull request?

```
git pull <url>
```