JSP

JSP

- JSP stands for Java Server Pages.
- It is a dynamic web resource program which is used to created web applications.

Limitations with Servlets

- To work with servlet strong java knowledge is required.
- It is not suitable for non-java programmers.
- It does not give any implicit object.
- (Object which can be used directly without any configuration)
- 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.

Advantages of JSP

- To work with jsp strong java knowlege is not required.
- It is suitable for java and non-java programmers.
- It supports tag based language.
- It allows us to work with custom tags and third party supplied tags.
- It gives 9 implicit objects.
- Configuration of each jsp program in web.xml file is optional.
- Handling exceptions are optional.
- We can maintain HTML code and java code sperately.
- It gives all the features of servlet.

First web application development having JSP program as 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.

ABC.jsp

```
<center>
    <h1>

        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">

</web-app>

Request url

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

Configuration of jsp program in web.xml file

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>
```

```
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>/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 access our web application by using url pattern. It means how can we hide our application accessible by file name

- To hide our application accessible by file name we need to place "ABC.jsp" file inside "Web Content/WEB-INF" folder.

Deployment Directory Structure

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

Note:

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

ABC.jsp

```
<center>
        <h1>

                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

Life cycle methods of JSP

We have following three life cycle methods in jsp.

1) _jspInit()

- It is used for instantiation event.
- This method will execute just before JES class creation.
- JES stands for Java Equivalent Servlet.

2) _jspService()

- It is used for request arrival event.
- This method will execute when ever request goes to jsp program.

3) _jspDestroy()

- It is used for destruction event.
- This method will execute just before JES class destruction.

Phases of JSP

There are two phases in JSP.

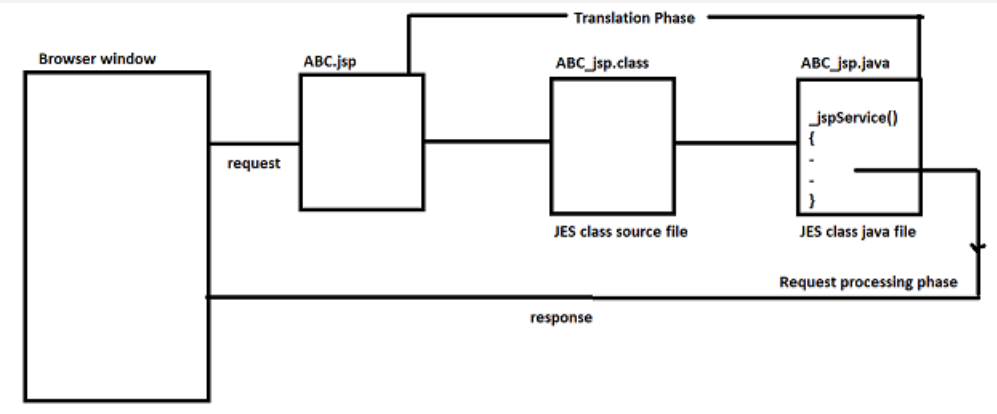
1)Translation phase

- In this phase, our jsp program converts to JES class.

2)Request processing phase

- In this phase ,our JES class will be executed and result will send to browser window/client.

Diagram: jsp1.1



How to enable <load-on-startup> and what happens if we enable <load-on-startup>?

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

ex:

```
<?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 it will perform translation phase during the server startup or during the deployment of web applications.

JSP Tags/Elements

- JSP contains following tags.

1) Scriting 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 --%>

ECJ4.4.2 Download link

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

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/

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 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 :"+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/

Declaration tag

It 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

```
<%!

    int i=100;
%>

<%= "The value is ="+i %>
```

index2.jsp

```
<%!

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

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 error is called exception.
- Exception may comes anywhere in our web application so handling the exceptions is a safer side for the programmer.

There are two ways to handle the exceptions in jsp.

1) Using "errorPage" and "isErrorPage" attributes of page directive tag.

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

1) Using "errorPage" and "isErrorPage" attributes of page directive tag

Deployment Directory structure

```
JspApp4
|
|---Java Resources
|
|---Web Content
        |
        |---index.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.

index.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>
```

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>
```

process.jsp

```
<%@page errorPage="error.jsp" %>
<%
        String sno1=request.getParameter("t1");
        String sno2=request.getParameter("t2");
```

```
        //convert string to int
        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 is occured.
</i></b>
<br>
<%=  exception %>
```

Request url

http://localhost:2525/JspApp4/

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

- This approach is highly 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

|

|---index.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.

index.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>
```

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>


 <error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/error.jsp</location>
 </error-page>


</web-app>
```

process.jsp

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


        //convert string to int
        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 is occured.
</i></b>
<br>
<%=  exception %>
```

Request url

http://localhost:2525/JspApp4/

HTML to 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>
```

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

```
<%@page import="java.sql.*" %>
<%
        String sno=request.getParameter("t1");
        int no=Integer.parseInt(sno);

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

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

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

        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:
XE","system","admin");
                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("Record Not Inserted");
            else
                out.println("Record Inserted");

            ps.close();
            con.close();
        }
        catch(Exception e)
        {
            out.println(e);
        }
%>
```

Request url

http://localhost:2525/JspApp5/

Action Tags

Action tags use servlet API funtionality features.

Action tags executed dynamically at runtime.

In action tags we don't have any xml tags.

Action tags contains only 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:include>

<jsp:forward>

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

and etc.

Action include

- In action include , the output of source jsp program and destination jsp program combinely goes to browser window as dynamic response.
- Here code will not add to source jsp program but output.
- It internally uses servlet 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>Begining of A.jsp program</i></b>
<br>
<jsp:include page="B.jsp"/>
<br>
<b><i>Ending of A.jsp program</i></b>
```

B.jsp

```
<b><i>This is B.jsp program</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 forward

- In action forward, the output of source jsp program will be discarded and output destination jsp program goes to browser window as dynamic response.
- It internally uses servlet functionality called rd.forwared(req,res).

syntax:

```
<jsp:forwared  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>Begining of A.jsp program</i></b>
```

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

B.jsp

```
<b><i>This is B.jsp program</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 JavaBean communication

- JSP to javabean communication is possible by using three tags.

1) <jsp:useBean> tag

- It is used to locate and created bean class object.

2) <jsp:setProperty> tag

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

3) <jsp:getProperty> tag

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

Note:

- All the above tags are independent tags.

ex:1

Deployment Directory structure

```
JspApp7
|
|---Java Resources
|       |
        |------src
                |
                |---com.ihub.www
                        |
                        |----CubeNumber.java

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

Note:

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

CubeNumber.java

```
package com.ihub.www;


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

process.jsp

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


<%= "Cube of a given number is ="+cn.cube(5) %>
```

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/JspApp7/

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>
```

User.java

```
package com.ihub.www;

public class User
{
        private String username;
        private String password;
        private String email;

        //setter and getter methods
        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;
	}

}
```

process.jsp

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

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

Records Are <br>

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

Email : <jsp:getProperty property="email" name="u"/>

Request url

http://localhost:2525/JspApp8/

2) Custom Action tags

- Tags which are created by ther user based on the application requirements are called custom tags.

ex:

tagname attributevalue

| |

<ihub:cube number="5"/>

| |

prefix attributename

To create custom tag in jsp ,we will use taglib directive.

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" %>
Cube of a given number is =<ihub:cube number="5" />
```

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>
```

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 java.io.IOException;

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

public class CubeNumber extends TagSupport
{
        private int number;

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

        public int doStartTag()throws JspException
        {

                JspWriter out=pageContext.getOut();
```

```
            try
            {
                out.println(number*number*number);
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }

            return SKIP_BODY;
        }
}
```

Request url

http://localhost:2525/JspApp9/

How to convert dynamic web project to a war file

- As a part of monolithic architecture , we need to deploy our project in the form of war file.

ex:

right click to project --> export --> war file -->

select destination location --> save --> finish.

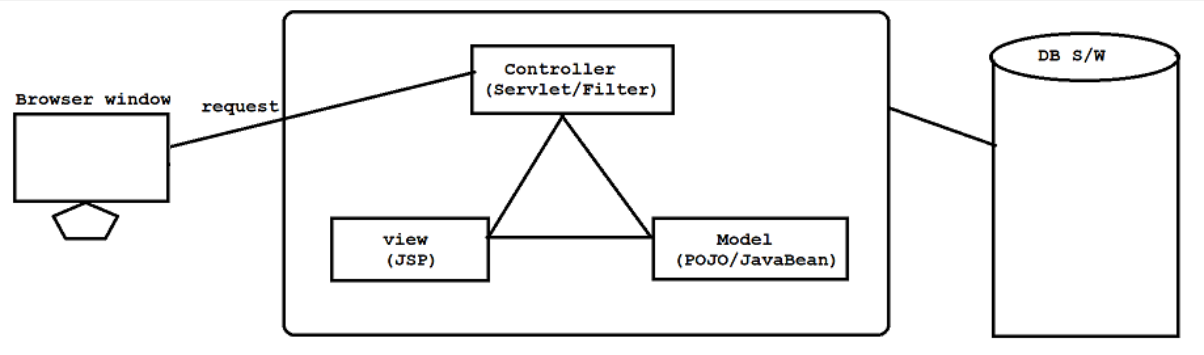
MVC in JSP

- MVC stands for Model View Controller.
- It is a design pattern which seperates business logic, persistence logic and data.
- Controller acts like a interface between model and view.
- Controller is used to intercept all incoming request.
- Model contains bussiness logic and data.
- View contains presentation logic i.e UI.

Note:

- **It is higly recommanded to create web applications by using MVC design pattern.**

Diagram: jsp4.1



Deployment Directory structure

MVCApp

|

|----Java Resources

| |

|------src

|

|---com.ihub.www

|

|---LoginBean.java

|---LoginSrv.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">

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

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

    <input type="submit" value="Login"/>
</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 doGet(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");
```

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

                //add bean object to the 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 :"+lb.getUsername() %> <br>
<%= "Passwrod :"+lb.getPassword() %> <br>
```

error.jsp

```
<b><i>
	<font color="red">
		Sorry! Incorrect username or password
	</font>
</i></b>

<%@include  file="form.html" %>
```

Request url

http://localhost:2525/MVCApp/

Implict objects in JSP

- Object which can be used directly without any configuration is called implicit object.
- Implicit objects created by the web container which are available for every JSP page.

JSP contains 9 implicit objects as follow.

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 Login </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">

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

</web-app>
```

process.jsp

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

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

```
<%
        out.println(config.getInitParameter("driver"));
%>
```

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

```
<%
        out.println(application.getInitParameter("driver"));
%>
```

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"/>

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

first.jsp

```
<%
        String name = request.getParameter("t1");
        session.setAttribute("pname", name);
%>
```

```
<%= "Welcome =  "+name %>


<br>
<a href="second.jsp"> goto to next page </a>
```

second.jsp

```
<%

        String name=(String)session.getAttribute("pname");
%>
<%= "Hey! : "+ 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.

ex: page

request

session

application

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

first.jsp

```
<%
        String name = request.getParameter("t1");
        pageContext.setAttribute("pname",
name,pageContext.SESSION_SCOPE);
%>
<%= "Welcome =  "+name %>
<br>
<a href="second.jsp"> goto to next page </a>
```

second.jsp

```
<%

String
name=(String)pageContext.getAttribute("pname",pageContext.SESSION_SCO
PE);
%>
<%= "Hey! : "+ 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