JSP

Day 02

Agenda…

- Java beans
- JSP standard actions related to beans
- Other JSP standard actions
- Tag libraries
- Summary

Java beans

"A Java class that represents a piece of information, accessor methods and related actions"

Characteristics of a Bean class

- Is a public class
- Has a public constructor which does not take a parameter
- Has accessor methods
  - Getter methods (get properties)
  - Setter methods (set properties)
- May have any other data / method members

Example of a bean class

```
package naveen.beans;
public class Person{
    private String name;
    private String addr;
    public Person(){
    }
    public String getName(){
        return name;
    }
    public void setName(String name){
        this.name=name;
    }
```

Example continued..

```
    public String getAddress(){
        return addr;
    }
    public void setAddress(String addr){
        this.addr=addr;
    }
}
```

<jsp:useBean> syntax

```
<jsp:useBean
    id="id"
    class="beanClassName"
    type="typeOfTheReference"
    scope="page|request|session|application"/>
```

Example:

```
<jsp:useBean
    id="p1"
    class="Naveen.beans.Person"
    scope="session"/>
```

Same as

```
Naveen.beans.Person p1=new Naveen.beans.Person();
session.setAttribute("p1", p1);
```

<jsp:setProperty>: Syntax

```
<jsp:setProperty
   name="nameOfTheBean"
   property="whichPropertyToBeSet"
   value="valueToBeSet"/>
```

Example:

```
   <jsp:setProperty
   name="p1"
   property="name"
   value="Naveen Kumar"/>
        value="Bangalore"/>
```

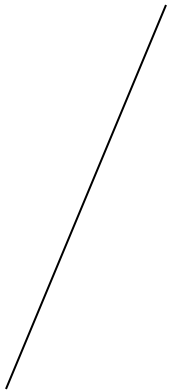
```
<jsp:setProperty
name="p1"
property="address"
```

<jsp:getProperty>: Syntax

```
<jsp:getProperty
   name="nameOfTheBean"
   property="whichPropertyToGet"/>
```

Example:

```
   <jsp:getProperty
   name="p1"
   property="name" />
```



```
   <jsp:getProperty
   name="p1"
   property="address" />
```

<jsp:include/>

- Can be used to include the output of another page
- The request/response objects of the parent are made available to the included resource

Syntax

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

Example

```
<jsp:include page="authorInfo.jsp" />
```

<jsp:forward/>

- Can be used to forward the request and response object to another JSP/servlet
- A method of servlet-chaining using JSP

Syntax

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

Example

```
<jsp:forward page="help.jsp" />
```

<jsp:param/>

- Used to pass name/value pair as parameters to the included/forwarded URI

Syntax

```
<jsp:param name="param_name"
    Value="param_value"/>
```

Example

```
<jsp:param name="eno" value="120"/>
```

Tag Libraries

JSP Custom Tags

- JSP custom tags are merely Java classes that implement special interfaces.
- Once they are developed and deployed, their actions can be called from your JSP using XML syntax.
- They have a start tag and an end tag.

Benefits

- They can reduce or eliminate scriptlets in your JSP applications.
- They have simpler syntax.
- They can improve the productivity of nonprogrammer content developers, by allowing them to perform tasks that cannot be done with HTML.
- They are reusable.

JSP Custom Tags

- A bodyless tag can be expressed as:

  <tagLibraryPrefix:tagName/>

  Ex: <x:getEmpData empid="1001" />

- And, a tag with a body can be expressed as:

  <tagLibraryPrefix:tagName>
  body
  </tagLibraryPrefix:tagName>

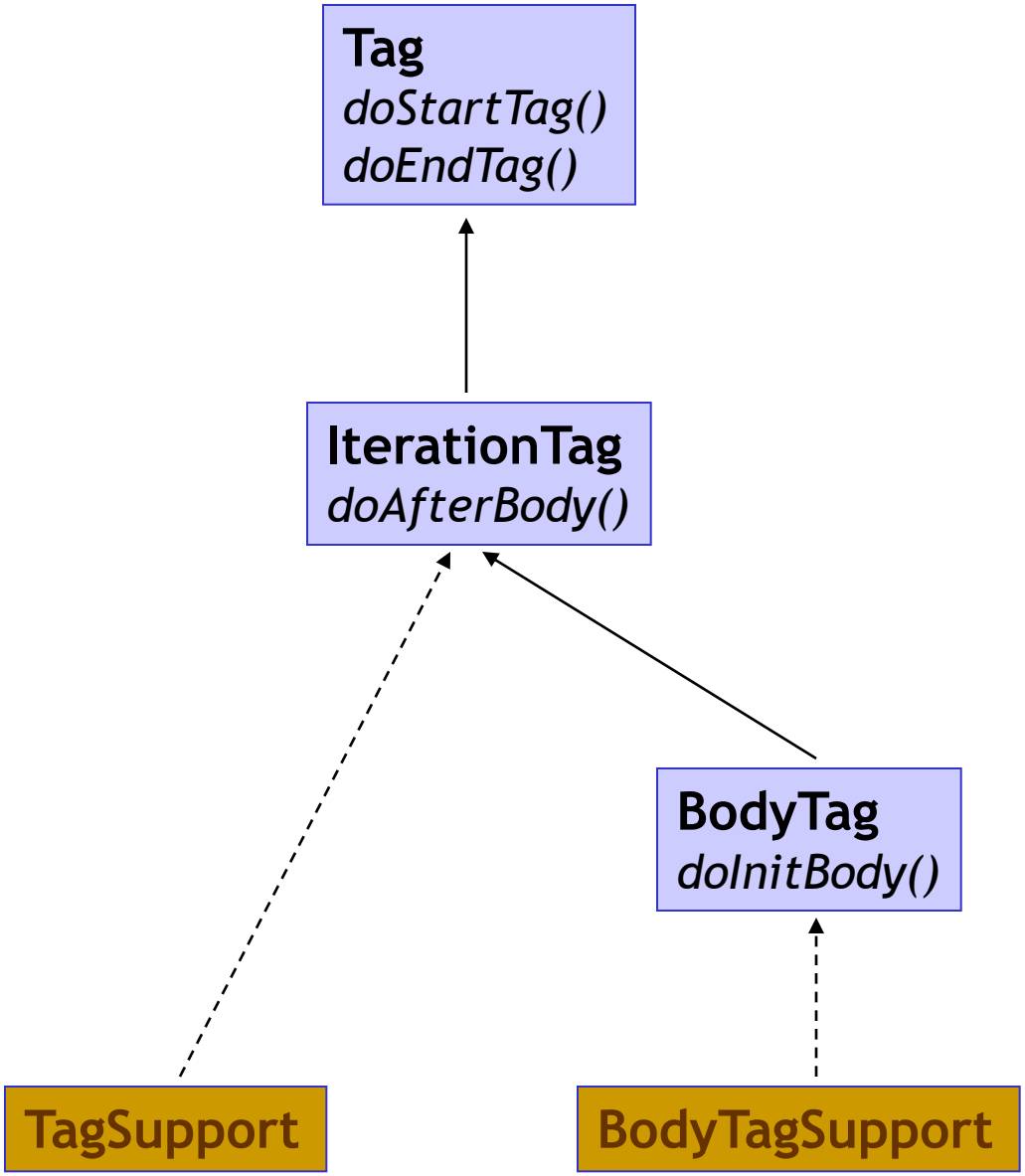
  Ex: <x:getEmpData>1001</x:getEmpData>

Task involved…

1. Develop a tag handler
   - A Java class that implements Tag interface
2. Create a Tag Library Descriptor
   - An XML file with .tld extension
3. Use the tag in a JSP with the help of taglib directive
   - <%@taglib uri="…" prefix="…" %>

1 : Tag Handler

- Is a public java class
- The custom tag's definition is coded here
- Must implement one of the two interfaces
  - Tag or
  - BodyTag
- Normally we extend our Tag handler class to
  - tagSupport or
  - BodyTagSupport


Tag
doStartTag()
doEndTag()
IterationTag
doAfterBody()
BodyTag
doInitBody()
TagSupport
BodyTagSupport

Sample code

```
package naveen;

import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;

public class UcaseTag
extends BodyTagSupport{
      public int doAfterBody(){
                  JspWriter out=getPreviousOut();

                  BodyContent bc=getBodyContent();
                  String txt=bc.getString();

                  try{
                              out.println(txt.toUpperCase());
                  }
                  catch(Exception ex){
                              System.out.println("error in UcaseTag: "+ex);
                  }
                  return 0;
      }
}
```

2: Tag Library Descripters

- These are simple XML documents
- Have .tld extension
- Define a tagname and map the same with the corresponding class (tag handler)

Sample TLD

```
<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.1</jsp-version>
    <short-name>naveen Tags</short-name>

    <tag>
        <name>ucase</name>
        <tag-class>naveen.UcaseTag</tag-class>
    </tag>

    <tag>
        <name>lcase</name>
        <tag-class>naveen.LcaseTag</tag-class>
    </tag>
</taglib>
```

3: Using taglibs

The syntax:

<%@ taglib uri="…" prfix="…" %>

Example:

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


<v:ucase>You are seeing this text in upper case</v:ucase>

<v:lcase>You are seeing this text in lower case</v:lcase>
```

End of

JSP training session