# Java Bean

A Java Bean is a java class that should follow following conventions:

* It should have a no-arg constructor.
* It should be Serializable.
* It should provide methods to set and get the values of the properties, known as getter and setter methods.

## Why use Java Bean?

| According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object, so we can access this object from multiple places. Moreover, it provides the easy maintenance. |
| --- |

### Simple example of java bean class

1. //Employee.java
3. **package** mypack;
4. **public** **class** Employee **implements** java.io.Serializable{
5. **private** **int** id;
6. **private** String name;
8. **public** Employee(){}
10. **public** **void** setId(**int** id){**this**.id=id;}
12. **public** **int** getId(){**return** id;}
14. **public** **void** setName(String name){**this**.name=name;}
16. **public** String getName(){**return** name;}
18. }

JSP Action Tags (Action Elements)

| There are many JSP action tags or elements. Each tag is used to perform some specific tasks. The action tags basically are used to control the flow between pages and to use Java Bean. Jsp action tags are as follows:   * jsp:forward * jsp:include * jsp:useBean * jsp:setProperty * jsp:getProperty * jsp:plugin * jsp:param * jsp:fallback |
| --- |

| The jsp:useBean, jsp:setProperty and jsp:getProperty tags are used for bean development. So we will see these tags in bean developement. |
| --- |

### jsp:forward action tag

| The jsp:forward action tag is used to forward the request to another resource it may be jsp, html or another resource. |
| --- |

### Syntax of jsp:forward action tag without parameter

1. <jsp:forward page="relativeURL | <%= expression %>" />

### Syntax of jsp:forward action tag with parameter

1. <jsp:forward page="relativeURL | <%= expression %>">
2. <jsp:param name="parametername" value="parametervalue | <%=expression%>" />
3. </jsp:forward>

### Example of jsp:forward action tag without parameter

| In this example, we are simply forwarding the request to the printdate.jsp file. |
| --- |

### index.jsp

1. <html>
2. <body>
3. <h2>**this** is index page</h2>
5. <jsp:forward page="printdate.jsp" />
6. </body>
7. </html>

### printdate.jsp

1. <html>
2. <body>
3. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
4. </body>
5. </html>

### Example of jsp:forward action tag with parameter

| In this example, we are forwarding the request to the printdate.jsp file with parameter and printdate.jsp file prints the parameter value with date and time. |
| --- |

### index.jsp

1. <html>
2. <body>
3. <h2>**this** is index page</h2>
5. <jsp:forward page="printdate.jsp" >
6. <jsp:param name="name" value="javatpoint.com" />
7. </jsp:forward>
9. </body>
10. </html>

### printdate.jsp

1. <html>
2. <body>
4. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>
5. <%= request.getParameter("name") %>
7. </body>
8. </html>

# jsp:include action tag

The **jsp:include action tag** is used to include the content of another resource it may be jsp, html or servlet.

The jsp include action tag includes the resource at request time so it is **better for dynamic pages**because there might be changes in future.

### Advantage of jsp:include action tag

code reusability

### Syntax of jsp:include action tag without parameter

1. <jsp:include page="relativeURL | <%= expression %>" />

### Syntax of jsp:include action tag with parameter

1. <jsp:include page="relativeURL | <%= expression %>">
2. <jsp:param name="parametername" value="parametervalue | <%=expression%>" />
3. </jsp:include>

### Example of jsp:include action tag without parameter

| In this example, index.jsp file includes the content of the printdate.jsp file. |
| --- |

*File: index.jsp*

1. <html>
2. <body>
3. <h2>**this** is index page</h2>
5. <jsp:include page="printdate.jsp" />
7. <h2>end section of index page</h2>
8. </body>
9. </html>

*File: printdate.jsp*

1. <% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %>

# Java Bean

| A Java Bean is a java class that should follow following conventions:   * It should have a no-arg constructor. * It should be Serializable. * It should provide methods to set and get the values of the properties, known as getter and setter methods. |
| --- |

## Why use Java Bean?

| According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object, so we can access this object from multiple places. Moreover, it provides the easy maintenance. |
| --- |

### Simple example of java bean class

1. //Employee.java
2. **package** mypack;
3. **public** **class** Employee **implements** java.io.Serializable{
4. **private** **int** id;
5. **private** String name;
6. **public** Employee(){}
7. **public** **void** setId(**int** id){**this**.id=id;}
8. **public** **int** getId(){**return** id;}
9. **public** **void** setName(String name){**this**.name=name;}
10. **public** String getName(){**return** name;}
11. }

### How to access the java bean class?

| To access the java bean class, we should use getter and setter methods. |
| --- |

1. **package** mypack;
2. **public** **class** Test{
3. **public** **static** **void** main(String args[]){
5. Employee e=**new** Employee();//object is created
7. e.setName("Arjun");//setting value to the object
9. System.out.println(e.getName());
11. }}

#### Note: There are two ways to provide values to the object, one way is by constructor and second is by setter method.

# jsp:useBean action tag

| The jsp:useBean action tag is used to locate or instantiate a bean class. If bean object of the Bean class is already created, it doesn't create the bean depending on the scope. But if object of bean is not created, it instantiates the bean. |
| --- |

## Syntax of jsp:useBean action tag

1. <jsp:useBean id= "instanceName" scope= "page | request | session | application"
2. **class**= "packageName.className" type= "packageName.className"
3. beanName="packageName.className | <%= expression >" >
4. </jsp:useBean>

### Attributes and Usage of jsp:useBean action tag

| 1. **id:**is used to identify the bean in the specified scope. 2. **scope:**represents the scope of the bean. It may be page, request, session or application. The default scope is page.    * **page:**specifies that you can use this bean within the JSP page. The default scope is page.    * **request:**specifies that you can use this bean from any JSP page that processes the same request. It has wider scope than page.    * **session:**specifies that you can use this bean from any JSP page in the same session whether processes the same request or not. It has wider scope than request.    * **application:**specifies that you can use this bean from any JSP page in the same application. It has wider scope than session. 3. **class:**instantiates the specified bean class (i.e. creates an object of the bean class) but it must have no-arg or no constructor and must not be abstract. 4. **type:**provides the bean a data type if the bean already exists in the scope. It is mainly used with class or beanName attribute. If you use it without class or beanName, no bean is instantiated. 5. **beanName:**instantiates the bean using the java.beans.Beans.instantiate() method. |
| --- |

### Simple example of jsp:useBean action tag

| In this example, we are simply invoking the method of the Bean class. |
| --- |

### Calculator.java (a simple Bean class)

1. **package** com.javatpoint;
2. **public** **class** Calculator{
4. **public** **int** cube(**int** n){**return** n\*n\*n;}
6. }

### index.jsp file

1. <jsp:useBean id="obj" **class**="com.javatpoint.Calculator"/>
3. <%
4. **int** m=obj.cube(5);
5. out.print("cube of 5 is "+m);
6. %>



# jsp:setProperty and jsp:getProperty action tags

| The setProperty and getProperty action tags are used for developing web application with Java Bean. In web devlopment, bean class is mostly used because it is a reusable software component that represents data. |
| --- |
| The jsp:setProperty action tag sets a property value or values in a bean using the setter method. |

Syntax of jsp:setProperty action tag

1. <jsp:setProperty name="instanceOfBean" property= "\*"   |
2. property="propertyName" param="parameterName"  |
3. property="propertyName" value="{ string | <%= expression %>}"
4. />

### Example of jsp:setProperty action tag if you have to set all the values of incoming request in the bean

1. <jsp:setProperty name="bean" property="\*" />

### Example of jsp:setProperty action tag if you have to set value of the incoming specific property

1. <jsp:setProperty name="bean" property="username" />

### Example of jsp:setProperty action tag if you have to set a specific value in the property

1. <jsp:setProperty name="bean" property="username" value="Kumar" />

## jsp:getProperty action tag

| The jsp:getProperty action tag returns the value of the property. |
| --- |

### Syntax of jsp:getProperty action tag

1. <jsp:getProperty name="instanceOfBean" property="propertyName" />

### Simple example of jsp:getProperty action tag

1. <jsp:getProperty name="obj" property="name" />

## Example of bean development in JSP

| In this example there are 3 pages:   * index.html for input of values * welocme.jsp file that sets the incoming values to the bean object and prints the one value * User.java bean class that have setter and getter methods |
| --- |

#### index.html

1. <form action="process.jsp" method="post">
2. Name:<input type="text" name="name"><br>
3. Password:<input type="password" name="password"><br>
4. Email:<input type="text" name="email"><br>
5. <input type="submit" value="register">
6. </form>

#### process.jsp

1. <jsp:useBean id="u" **class**="org.sssit.User"></jsp:useBean>
2. <jsp:setProperty property="\*" name="u"/>
4. Record:<br>
5. <jsp:getProperty property="name" name="u"/><br>
6. <jsp:getProperty property="password" name="u"/><br>
7. <jsp:getProperty property="email" name="u" /><br>

#### User.java

1. **package** org.sssit;
3. **public** **class** User {
4. **private** String name,password,email;
5. //setters and getters
6. }

#### Reusing Bean in Multiple Jsp Pages

Let's see the simple example, that prints the data of bean object in two jsp pages.

#### index.jsp

Same as above.

#### User.java

Same as above.

#### process.jsp

1. <jsp:useBean id="u" **class**="org.sssit.User" scope="session"></jsp:useBean>
2. <jsp:setProperty property="\*" name="u"/>
4. Record:<br>
5. <jsp:getProperty property="name" name="u"/><br>
6. <jsp:getProperty property="password" name="u"/><br>
7. <jsp:getProperty property="email" name="u" /><br>
8. <a href="second.jsp">Visit Page</a>

#### second.jsp

1. <jsp:useBean id="u" **class**="org.sssit.User" scope="session"></jsp:useBean>
2. Record:<br>
3. <jsp:getProperty property="name" name="u"/><br>
4. <jsp:getProperty property="password" name="u"/><br>
5. <jsp:getProperty property="email" name="u" /><br>

#### Using variable value in setProperty tag

In some case, you may get some value from the database, that is to be set in the bean object, in such case, you need to use expression tag. For example:

#### process.jsp

1. <jsp:useBean id="u" **class**="org.sssit.User"></jsp:useBean>
2. <%
3. String name="arjun";
4. %>
5. <jsp:setProperty property="name" name="u" value="<%=name %>"/>
6. Record:<br>
7. <jsp:getProperty property="name" name="u"/><br>

# Custom Tags in JSP

**Custom tags** are user-defined tags. They eliminate the possibility of scriptlet tag and separates the business logic from the JSP page.

The same business logic can be used many times by the use of custom tags.

### Advantages of Custom Tags

| The key advantages of Custom tags are as follows:   1. **Eliminates the need of scriptlet tag** The custom tags eliminates the need of scriptlet tag which is considered a bad programming approach in JSP. 2. **Separation of business logic from JSP**The custom tags separate the the business logic from the JSP page so that it may be easy to maintain. 3. **Reusability** The custom tags makes the possibility to reuse the same business logic again and again. |
| --- |

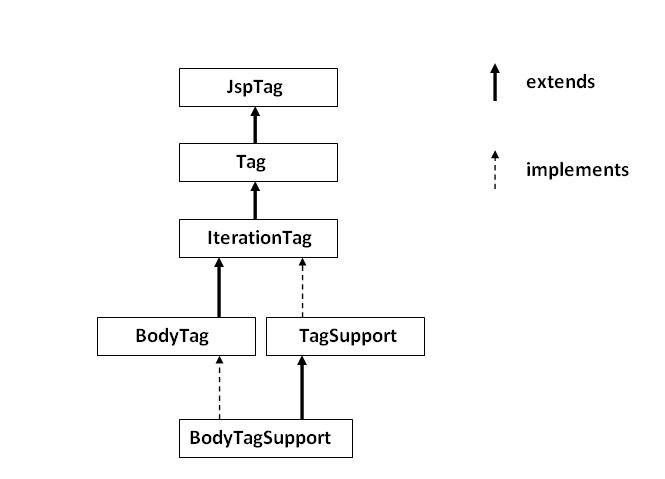
### Syntax to use custom tag

| There are two ways to use the custom tag. They are given below: |
| --- |

1. <prefix:tagname attr1=value1....attrn=valuen />
2. <prefix:tagname attr1=value1....attrn=valuen >
3. body code
4. </prefix:tagname>

### JSP Custom Tag API

| The javax.servlet.jsp.tagext package contains classes and interfaces for JSP custom tag API. The JspTag is the root interface in the Custom Tag hierarchy. |
| --- |



### JspTag interface

| The JspTag is the root interface for all the interfaces and classes used in custom tag. It is a marker interface. |
| --- |

### Tag interface

| The Tag interface is the sub interface of JspTag interface. It provides methods to perform action at the start and end of the tag. |
| --- |

### Fields of Tag interface

There are four fields defined in the Tag interface. They are:

| **Field Name** | **Description** |
| --- | --- |
| **public static int EVAL\_BODY\_INCLUDE** | it evaluates the body content. |
| **public static int EVAL\_PAGE** | it evaluates the JSP page content after the custom tag. |
| **public static int SKIP\_BODY** | it skips the body content of the tag. |
| **public static int SKIP\_PAGE** | it skips the JSP page content after the custom tag. |

### Methods of Tag interface

The methods of the Tag interface are as follows:

| **Method Name** | **Description** |
| --- | --- |
| **public void setPageContext(PageContext pc)** | it sets the given PageContext object. |
| **public void setParent(Tag t)** | it sets the parent of the tag handler. |
| **public Tag getParent()** | it returns the parent tag handler object. |
| **public int doStartTag()throws JspException** | it is invoked by the JSP page implementation object. The JSP programmer should override this method and define the business logic to be performed at the start of the tag. |
| **public int doEndTag()throws JspException** | it is invoked by the JSP page implementation object. The JSP programmer should override this method and define the business logic to be performed at the end of the tag. |
| **public void release()** | it is invoked by the JSP page implementation object to release the state. |

### IterationTag interface

| The IterationTag interface is the sub interface of the Tag interface. It provides an additional method to reevaluate the body. |
| --- |

### Field of IterationTag interface

| There is only one field defined in the IterationTag interface.   * **public static int EVAL\_BODY\_AGAIN**it reevaluates the body content. |
| --- |

### Method of Tag interface

| There is only one method defined in the IterationTag interface.   * **public int doAfterBody()throws JspException**it is invoked by the JSP page implementation object after the evaluation of the body. If this method returns EVAL\_BODY\_INCLUDE, body content will be reevaluated, if it returns SKIP\_BODY, no more body content will be evaluated. |
| --- |

### TagSupport class

| The TagSupport class implements the IterationTag interface. It acts as the base class for new Tag Handlers. It provides some additional methods also. |
| --- |