JSP - JAVABEANS

A JavaBean is a specially constructed Java class written in the Java and coded according to the JavaBeans API specifications.

Following are the unique characteristics that distinguish a JavaBean from other Java classes:

- It provides a default, no-argument constructor.
- It should be serializable and implement the **Serializable** interface.
- It may have a number of properties which can be read or written.
- It may have a number of "getter" and "setter" methods for the properties.

JavaBeans Properties:

A JavaBean property is a named attribute that can be accessed by the user of the object. The attribute can be of any Java data type, including classes that you define.

A JavaBean property may be read, write, read only, or write only. JavaBean properties are accessed through two methods in the JavaBean's implementation class:

Method	Description
get**PropertyName**	For example, if property name is *firstName*, your method name would be getFirstName to read that property. This method is called accessor.
set**PropertyName**	For example, if property name is *firstName*, your method name would be setFirstName to write that property. This method is called mutator.

A read-only attribute will have only a get**PropertyName** method, and a write-only attribute will have only a set**PropertyName** method.

JavaBeans Example:

Consider a student class with few properties:

```
package com.tutorialspoint;

public class StudentsBean implements java.io.Serializable
{
   private String firstName = null;
   private String lastName = null;
   private int age = 0;

   public StudentsBean() {
   }
   public String getFirstName(){
      return firstName;
   }
   public String getLastName(){
      return lastName;
   }
   public int getAge(){
      return age;
   }
   public void setFirstName(String firstName){
      this.firstName = firstName;
```

```
   }
   public void setLastName(String lastName){
      this.lastName = lastName;
   }
   public void setAge(Integer age){
      this.age = age;
   }
}
```

Accessing JavaBeans:

The **useBean** action declares a JavaBean for use in a JSP. Once declared, the bean becomes a scripting variable that can be accessed by both scripting elements and other custom tags used in the JSP. The full syntax for the useBean tag is as follows:

```
<jsp:useBean  typeSpec/>
```

Here values for the scope attribute could be page, request, session or application based on your requirement. The value of the **id** attribute may be any value as a long as it is a unique name among other useBean declarations in the same JSP.

Following example shows its simple usage:

```
<html>
<head>
<title>useBean Example</title>
</head>
<body>

<jsp:useBean   />
<p>The date/time is <%= date %>

</body>
</html>
```

This would produce following result:

```
The date/time is Thu Sep 30 11:18:11 GST 2010
```

Accessing JavaBeans Properties:

Along with <jsp:useBean...>, you can use <jsp:getProperty/> action to access get methods and <jsp:setProperty/> action to access set methods. Here is the full syntax:

```
<jsp:useBean  >
   <jsp:setProperty name="bean's id" property="property name"
                    value="value"/>
   <jsp:getProperty name="bean's id" property="property name"/>
   ...........
</jsp:useBean>
```

The name attribute references the id of a JavaBean previously introduced to the JSP by the useBean action. The property attribute is the name of the get or set methods that should be invoked.

Following is a simple example to access the data using above syntax:

```
<html>
<head>
<title>get and set properties Example</title>
</head>
<body>

<jsp:useBean
                    >
```

```
   <jsp:setProperty name="students" property="firstName"
                    value="Zara"/>
   <jsp:setProperty name="students" property="lastName"
                    value="Ali"/>
   <jsp:setProperty name="students" property="age"
                    value="10"/>
</jsp:useBean>

<p>Student First Name:
   <jsp:getProperty name="students" property="firstName"/>
</p>
<p>Student Last Name:
   <jsp:getProperty name="students" property="lastName"/>
</p>
<p>Student Age:
   <jsp:getProperty name="students" property="age"/>
</p>

</body>
</html>
```

Let us make StudentsBean.class available in CLASSPATH and try to access above JSP. This would produce following result:

```
Student First Name: Zara

Student Last Name: Ali

Student Age: 10
```