Introduction to JavaBeans and Bean Builder

A JavaBean is a reusable software component. It encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.

A JavaBean is a specially constructed Java class written in java with a specific set of rules.

- It should implement the serializable interface (this allows java applications and frameworks to save, store and restore the JavaBean state).
- It must have a public, no argument constructor (i.e. a default constructor).
- All properties in java bean must be private with public getters and setter methods (i.e. The JavaBean class attributes must be accessed via accessor and mutator methods that follow a standard naming convention).

Syntax for Setter Methods:

1. It should be public in nature.
2. The return-type should be void.
3. The setter method should be prefixed with set.
4. It should take some argument i.e. it should not be no-argument method.

Example:

```
public void setFirstName (String firstName) {

    this. firstName = firstName;

}
```

Syntax for Getter Methods:

1. It should be public in nature.
2. The return-type should not be void i.e. according to our requirement we have to give return-type.
3. The getter method should be prefixed with get.
4. It should not take any argument.

Example:

```
public String getFirstName () {

    return firstName;

}
```

Syntax for Boolean Methods:

1. It should be public in nature.
2. The return-type should be boolean.
3. The boolean method should be prefixed with is.
4. It should not take any argument.

Example:

```
public String isEmpty () {

    return empty;

}
```

//A JavaBean Class Example

```
import java.io.Serializable;

public class StudentBean implements Serializable {     // implements interface Serializable
   private String firstName;                            // all attributes of class are private
   private String lastName;
   private double grade;

   public StudentBean() {                               // no argument constructor

   }
   public String getFirstName() {                       // setters and getters for private attributes
      return firstName;
   }
   public void setFirstName(String firstName) {
      this.firstName = firstName;
   }
   public String getLastName() {
      return lastName;
   }
   public void setLastName(String lastName) {
      this.lastName = lastName;
   }
   public double getGrade() {
      return grade;
   }
   public void setGrade(double grade) {
      this.grade = grade;
   }
}
```

// Java Program to Access JavaBean Class

```
public class Test {
   public static void main(String[] args) {
      StudentBean sb = new SudentBean();
      sb.setFirstName("Geetha");
      sb.setLastName("Charan");
      System.out.println("Student Name: "+sb.getFirstName()+" "+sb.getLastName());
   }
}
```

Output :- Student Name: Geetha Charan

JavaBean Properties

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

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

1. getPropertyName ()

For example, if the property name is firstName, the method name would be getFirstName() to read that property. This method is called the accessor.

2. setPropertyName ()

For example, if the property name is firstName, the method name would be setFirstName() to write that property. This method is called the mutator.

A read-only attribute will have only a **getPropertyName()** method, and a write-only attribute will have only a **setPropertyName()** method.

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.

Syntax: <jsp:useBean id = "bean's name" scope = "bean's scope" typeSpec/>

Here values for the scope attribute can be a 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.

Example:

```
<html>
    <head>
        <title>useBean Example</title>
    </head>
    <body>
        <jsp:useBean id = “date” class = “java.util.Date” />
        <p>The date/time is <%= date %>
    </body>
</html>
```

Output:

Accessing JavaBeans Properties

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

Syntax:

```
<jsp:useBean id = “id” class = "bean's class" scope = "bean's scope">
    <jsp:setProperty name = “beans’s id” property = "property name"  value=”value”/>
    <jsp:getProperty name = “beans’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 the **set** methods that should be invoked.

Example :

```
<html>
	<head>
		<title>getters and setters example</title>
	</head>
	<body>
		<jsp:useBean id = “students” class = “com.packagename.StudentsName”>
			<jsp:setProperty name = “students” property = “firstName” value = “G”/>
			<jsp:setProperty name = “students” property = “lastName” value = “C”/>
			<jsp:setProperty name = “students” property = “grade” value = “8”/>
			<p>Student First Name:
				<jsp:getProperty name = “students” property = “firstName”/>
			</p>
			<p>Student Last Name:
				<jsp:getProperty name = “students” property = “lastName”/>
			</p>
			<p>Student Grade:
				<jsp:getProperty name = “students” property = “grade”/>
			</p>
	</body>
</html>
```

Output:

Student First Name: G

Student Last Name: C

Student Grade: 8