A Quick Guide to JavaBeans

The Official Definition of a Bean is: “A Bean is a reusable software component based on Sun’s JavaBeans Specification that can be manipulated visually in a builder tool”. Reusable components are simply pre-built pieces of programming code designed to perform a specific function. While designing an application in a visual environment, controls can be quickly dropped into the design, and modified to fit the task at hand. Most of the controls you'll find are designed to handle such tasks as pushbuttons, menus, text labels, and so forth. As a developer, you only need to write code to "glue" them into your application, and develop the interactions between controls.

A software component (i.e. reusable object) can be plugged into any target software application. You can develop software components using various programming languages, such as C, C++, Java, and Visual Basic.

Once you implement a Bean other can use it in a Builder Environment such as NetBeans or JBuilder to produce GUI applications more efficiently. Builder tool is nothing but an application development tool which lets you both to create new beans or use existing beans to create an application.

To enrich the software systems by adopting component technology JAVA came up with the concept called JavaBeans. Java provides the facility of creating some user defined components by means of Bean programming. We create simple components using JavaBeans. We can embed these beans into the software.

A JavaBean is a Java class that should follow the following conventions:

- It should have a no-arg constructor.
- It should be Serializable (Serialization in Java is a mechanism of writing the state of an object into a byte-stream. It is mainly used in Hibernate, RMI, JPA, EJB and JMS technologies. The reverse operation of serialization is called deserialization where byte-stream is converted into an object. The serialization and deserialization process are platform-independent, it means you can serialize an object in a platform and deserialize in different platform).
- It should provide methods to set and get the values of the properties, known as getter and setter methods.

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

//Employee.java
package mypack;
public class Employee implements java.io.Serializable
{
    private int id;
    private String name;
    public Employee()
    {
    }
    public void setId(int id)
    {
        this.id=id;
    }
    public int getId()
    {
        return id;
    }
    public void setName(String name)
    {
        this.name=name;
    }
    public String getName()
    {
        return name;
    }
}

To access the JavaBean class, we should use getter and setter methods

package mypack;
public class Test
{
    public static void main(String args[])
    {
        Employee e=new Employee(); //object is created
        e.setName("xyz"); //setting value to the object
        System.out.println(e.getName());
    }
}
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.

**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 Property Type:**

- **Simple properties:** To set a simple property is the one that takes a single value as a string or a number, a pair of accessor, i.e. getXXX (), and mutator, i.e setXXX(), methods are employed.

- **Boolean Properties:** A simple property with Boolean values – true or false – set in mutator method.

- **Indexed properties:** An Index property is one that gets or sets an array. It can hold an array of values using public void set propertyName(property_Type[] list) method. With an Index property, you supply two pairs of get and set methods. One for the array and one for the individual entries.

- **Bound properties:** Bound Properties tell interested listeners that their value has changed. A Bean that has a bound property generates an event when the property is changed. The event is of type propertyChangeEvent and is sent to objects that previously registered an interest in receiving such notifications.

- **Constrained properties:** A Bean that has a constrained property generates an event when an attempt is made to change its value. The event is of type propertyChangeEvent. It is sent to objects that previously registered an interest in receiving such.
Advantages of JavaBean

- The advantage of JavaBeans over standard programming controls is that Beans are independent. They are not specific to operating systems or development environments. A Bean created in one development environment can be easily copied and modified by another. This allows JavaBeans greater flexibility in enterprise computing, as components are easily shared between developers.
- It provides an easiness to reuse the software components.
- Persistence is the ability of an object to store its state, for recreation later. Beans use Java’s object serialization capabilities for persistence.
- The JavaBean properties and methods can be exposed to another application.
- The JavaBean has the property of “Write once and run anywhere”.
- The properties, events and methods of the bean can be controlled by the application developer (ex. Add new properties)

Disadvantages of JavaBean

- Need to create the setter and getter method for each property separately.
- JavaBeans are inherently mutable and so lack the advantages offered by immutable objects.