Introduction to JavaBeans

JavaBeans is one of the reusable class code element often applied on java based software application programming.

Here, Beans can be defined as a way of encompassing multiple objects beneath a single object.

The JavaBeans class components can be labelled as methods, events, properties and persistence.

This type of class is advantageous over others due to its notable reusability characteristics, enables object communication, platform-independent, shorter time consumption, requires lesser manual efforts, etc.

Understanding JavaBeans

As stated, JavaBeans helps in reusing software components in developing an application. With the help of JavaBeans, one can reuse software components that other developers write. This eliminates the need for understanding inner workings.

As Beans consists of numerous objects into a single object, one can directly access this object from various places. Its maintenance is easy. The software component model deals with the creation and reuse of the software components to build an application.

With the help of the builder tool, one can develop new beans or use existing beans to create an application. One can directly use these beans in the software.

If you are still confused, then consider the example of a worker assembling a car, and the radio is one of the components. Instead of manufacturing the radio from scratch, a worker can directly assemble the radio to the car. This eliminates the need for manufacturing a radio for assembly.

Advantages of JavaBeans

A software component architecture provides standard mechanisms to deal with software building blocks. The following list enumerates some of the specific benefits that Java technology provides for a component developer:

- A Bean obtains all the benefits of Java’s “write-once, run-anywhere” paradigm.
- The properties, events, and methods of a Bean that are exposed to an application builder tool can be controlled.
- A Bean may be designed to operate correctly in different locales, which makes it useful in global markets.
- Auxiliary software can be provided to help a person configure a Bean. This software is only needed when the design-time parameters for that component are being set. It does not need to be included in the run-time environment.
- The configuration settings of a Bean can be saved in persistent storage and restored at a later time.
- A Bean may register to receive events from other objects and can generate events that are sent to other objects.

Components of JavaBeans

A JavaBeans component is a Java class with the following features:

A no-argument constructor.

Properties defined with accessors and mutators(getter and setter method).

Class must not define any public instance variables.

The class must implement the **java.io.Serializable** interface.

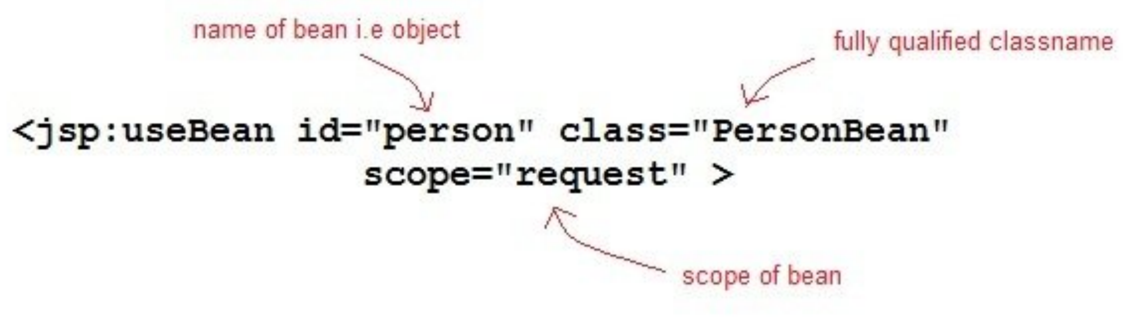
JSP jsp:useBean Tag

If you want to interact with a JavaBeans component using the Action tag in a JSP page, you must first declare a bean. The <jsp:useBean> is a way of declaring and initializing the actual bean object. By bean we mean JavaBean component object. Syntax of <jsp:useBean> tag

<jsp:useBean id = "beanName" class = "className"

scope = "page | request | session | application">

Here the **id** attribute specifies the name of the bean. **Scope** attribute specify where the bean is stored. The **class** attribute specify the fully qualified classname.



```
<jsp:useBean id="person" class="PersonBean"
                  scope="request" >
```

JSP jsp:useBean Tag

Given a useBean declaration of following :

```
<jsp:useBean id="myBean" class="PersonBean" scope="request" />
```

is equivalent to the following java code,

```
PersonBean myBean = (PersonBean)request.getAttribute("myBean");
if(myBean == null)
{
  myBean = new PersonBean();
  request.setAttribute("myBean", myBean);
}
```

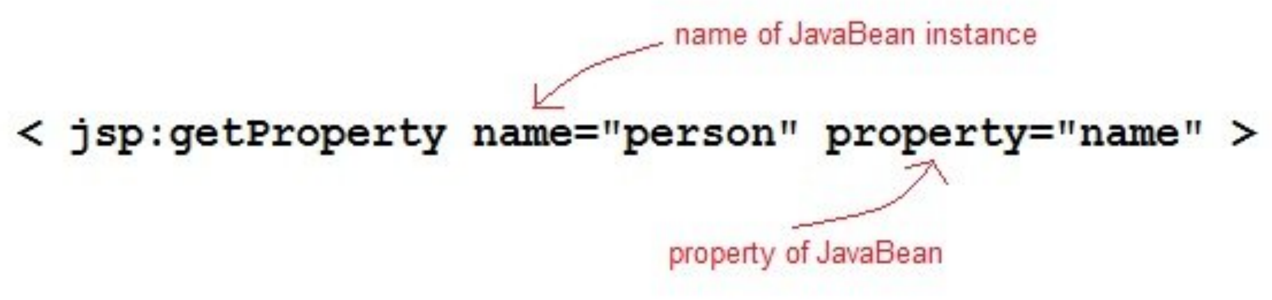
If jsp:useBean tag is used with a body, the content of the body is only executed if the bean is created. If the bean already exists in the named scope, the body is skipped.

JSP jsp:getProperty Tag

The getProperty tag is used to retrieve a property from a JavaBeans instance. The syntax of the getProperty tag is as follows:

<jsp:getProperty name="beanName" property="propertyName" />

The name attribute represents the name of the JavaBean instance. The property attribute represents the property of the JavaBean whose value we want to get.


name of JavaBean instance
< jsp:getProperty name="person" property="name" >
property of JavaBean

JSP jsp:setProperty Tag

The setProperty tag is used to store data in JavaBeans instances. The syntax of setProperty tag is:

<jsp:setProperty name="beanName" property="*">

<!-- or -->

<jsp:setProperty name="beanName" property="propertyName">

<!-- or -->

<jsp:setProperty name="beanName" property="propertyName" param="parameterName">

<!-- or -->

<jsp:setProperty name="beanName" property="propertyName" value="propertyValue">

The name attribute specifies the name of javaBean instances. This must match the id attribute specified in the jsp:useBean tag. The property attribute specifies which property of the bean to access.

Example of JavaBeans

Let's take a simple Java code example to understand what do we mean when we say JavaBean,

```
import java.io.Serializable;
public class StudentBean implements Serializable
{
  private String name;
  private int roll;

  // constructor
  public StudentBean()
  {
    this.name = "";
    this.roll = "";
  }
```

```
// getters and setters
  public void setName(String name)
  {
   this.name = name;
  }
  public String getName()
  {
   return name;
  }
  public int getRoll()
  {
   return roll;
  }
  public void setRoll(int roll)
  {
   this.roll = roll;
  }
}
```

Example of JavaBeans

Using a JavaBean in JSP page

JavaBeans can be used in any JSP page using the <jsp:useBean> tag, For example

```
<jsp:useBean id="bean name" scope="fully qualified path of bean" typeSpec/>
```

Using any JavaBean property in JSP page

JavaBeans can be used in any JSP page using the <jsp:useBean> tag, <jsp:setProperty> tag and <jsp:getProperty> tag , For example

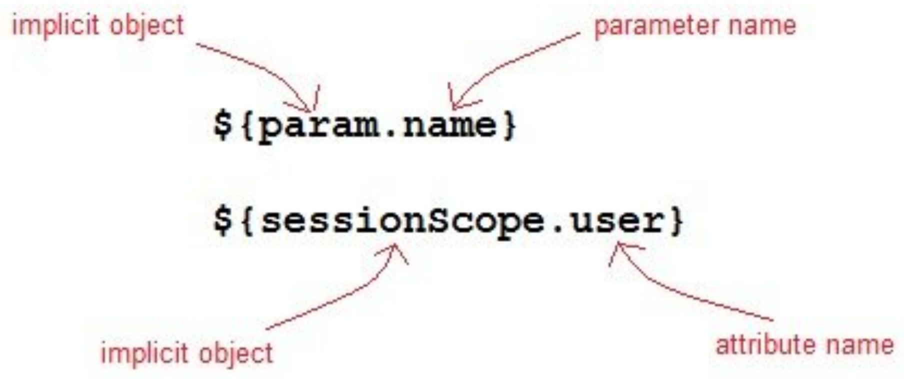
```
<jsp:useBean id="id" class="bean class name" scope="fully qualified path of bean">
   <jsp:setProperty name="beans id" property="property name" value="value"/>
   <jsp:getProperty name="beans id" property="property name"/>
   ...........
</jsp:useBean>
```

JSP Expression Language

Expression Language(EL) was added to JSP 2.0 specification. The purpose of EL is to produce scriptless JSP pages. The syntax of EL in a JSP is as follows:

${expr}

Here expr is a valid EL expression. An expression can be mixed with static text/values and can also be combined with other expressions to form larger expression.



```
${param.name}

${sessionScope.user}
```

How EL expression is used?

EL expression can be used in two ways in a JSP page

As attribute values in standard and custom tags. Example:

<jsp:include page="${location}">

Where location variable is separately defines in the jsp page.

Expressions can also be used in jsp:setProperty to set a properties value, using other bean properties like : If we have a bean named Square with properties length, breadth and area.

<jsp:setProperty name="square" property="area" value="${square.length*square.breadth}" />

To output in HTML tag:

<h1>Welcome ${name}</h1>

To deactivate the evaluation of EL expressions, we specify the isELIgnored attribute of the page directive as below:

<%@ page isELIgnored ="true|false" %>

JSP EL Implicit Objects

Implicit Object	Description
pageContext	It represents the PageContext object.
pageScope	It is used to access the value of any variable which is set in the Page scope
requestScope	It is used to access the value of any variable which is set in the Request scope.
sessionScope	It is used to access the value of any variable which is set in the Session scope
applicationScope	It is used to access the value of any variable which is set in the Application scope
param	Map a request parameter name to a single value
paramValues	Map a request parameter name to corresponding array of string values.
header	Map containing header names and single string values.
headerValues	Map containing header names to corresponding array of string values.
cookie	Map containing cookie names and single string values.

Arithmetic Operations available in EL

Arithmetic Operation	**Operator**
Addition	+
Substraction	-
Multiplication	*
Division	/ and div
Remainder	% and mod

Logical and Relational Operators available in EL

Logical and Relational Operator	Operator
Equals	== and eq
Not equals	!= and ne
Less Than	< and lt
Greater Than	> and gt
Greater Than or Equal	>= and ge
Less Than or Equal	<= and le
and	&& and and
or	|| and or
not	! and not
Logical and Relational Operator	Operator