Table 14.1 Standard JSP actions for using JavaBeans

Action	Description
`<jsp:useBean>`	Declares the use of a JavaBean instance in a JSP page
`<jsp:setProperty>`	Sets new values to the bean's properties
`<jsp:getProperty>`	Gets the current value of the bean's properties

In the sections that follow, we will describe each of these actions in details.

Declaring JavaBeans using <jsp:useBean>

The `jsp:useBean` action declares a variable in the JSP page and associates an instance of a JavaBean with it. The association is a two-step process. First, the action tries to find an existing instance of the bean. If an instance is not found, a new instance is created and associated with the declared variable. We can customize the behavior of this action using the five attributes shown in table 14.2.

Table 14.2 `<jsp:useBean>` attributes

Attribute Name	Description	Examples
`id`	The name by which the bean is identified in the JSP page.	`id="address"`
`scope`	The scope of the bean's instance: `page`, `request`, `session`, or `application`. The default value is `page`.	`scope="session"`
`class`	The Java class of the bean.	`class="BusinessAddress-Bean"`
`type`	Specifies the type of the variable to be used to refer to the bean.	`type="AddressBean"`
`beanName`	The name of a serialized bean if we are loading from a file or the name of a class if we are creating a new instance.	`beanName="business Data.John"` `beanName="AddressBean"`

Of the five attributes in table 14.2:

- The `id` attribute is mandatory.
- The `scope` attribute is optional.
- The three attributes `class`, `type`, and `beanName` can only be used in one of the following four combinations, and at least one of these attributes or combinations of attributes must be present in a `useBean` action:
  - `class`
  - `type`

- `class` and `type`
- `beanName` and `type`

To make it easier to understand and remember the usage of these attributes, let's first examine their meaning in the following sections. Then, we will look at some examples that will demonstrate how we can use the different combinations.

The id attribute

The `id` attribute uniquely identifies a particular instance of a bean. It is mandatory because its value is required by the other JSP actions, `<jsp:setProperty>` and `<jsp:getProperty>`, to identify the particular instance of the bean. In the generated Java servlet, the value of `id` is treated as a Java language variable; therefore, we can use this variable name in expressions and scriptlets in the JSP page. Note that since the value of the `id` attribute uniquely identifies a particular instance of a bean, we cannot use the same value for the `id` attribute in more than one `<jsp:useBean>` action within a single translation unit.

The scope attribute

The `scope` attribute specifies the scope in which the bean instance resides. Like implicit objects, the existence and accessibility of JavaBeans from JSP pages are determined by the four JSP scopes: *page*, *request*, *session*, and *application*. This attribute is optional, and if not specified, the page scope is used by default.

We cannot use the session scope for a bean in a JSP page if we set the value of the `page` directive attribute `session` to `false`.

The class attribute

The `class` attribute specifies the Java class of the bean instance. If the `<jsp:useBean>` action cannot find an existing bean in the specified scope, it creates a new instance of the bean's class as specified by the value of the `class` attribute using the class's publicly defined no-argument constructor. Therefore, the class specified by the `class` attribute must be a `public` non-abstract class and must have a `public` no-argument constructor. If the class is part of a package, then the fully qualified class name must be specified as `mypackage.MyClass`.

The type attribute

The `type` attribute specifies the type of the variable declared by the `id` attribute. Since the declared variable refers to the actual bean instance at request time, its type must be the same as the bean's class, or a superclass of the bean's class, or an interface implemented by the bean's class. Again, if the class or the interface is part of a package, then the fully qualified name must be specified as `mypackage.MyClass`.

The beanName attribute

The `beanName` attribute specifies the name of a bean as expected by the `instantiate()` method of the `java.beans.Beans` class. For this reason, `beanName` can refer either to a serialized bean or to the name of a class whose instance is to be created.

If the `beanName` attribute refers to a serialized bean, then the bean is loaded from the file that holds the bean. For example, if the attribute is specified as `beanName="businessData.visitorAddresses.John"`, the bean is loaded from the file `businessData/visitorAddresses/John.ser`. Note that we do not use the extension `.ser` in the value for the `beanName` attribute.

If the `beanName` attribute refers to a class, the class is loaded into memory, an instance of the class is created, and the instance is used as a bean. For example, if the attribute is specified as `beanName="chapter14.AddressBean"`, then the class is loaded from the file `chapter14/AddressBean.class`. Note that we do not use the extension `.class` in the value for the `beanName` attribute.

Both `class` and `beanName` can be used to create new instances from the specified class. However, the advantage of using `beanName` over `class` is that the value of the `beanName` attribute can also be specified as a request-time attribute expression, so it can be decided at request time and need not be specified at translation time.