# Advanced Composite Components

This chapter describes the advanced features of composite components in JavaServer Faces technology.

A composite component is a special type of JavaServer Faces template that acts as a component.

If you are new to composite components, see Composite Components before you proceed with this chapter.

The following topics are addressed here:

- Attributes of a Composite Component
- Invoking a Managed Bean
- Validating Composite Component Values
- The `compositecomponentlogin` Example Application

# Attributes of a Composite Component

You define an attribute of a composite component by using the `composite:attribute` tag.

Table 13-1 lists the commonly used attributes of this tag.

# Table 13-1 Commonly Used Attributes of the `composite:attribute` Tag

| Attribute | Description |
|---|---|
| `name` | Specifies the name of the composite component attribute to be used in the using page.<br><br>Alternatively, the `name` attribute can specify standard event handlers such as action, `actionListener`, and managed bean. |
| `default` | Specifies the default value of the composite component attribute. |
| `required` | Specifies if it is mandatory to provide a value for the attribute. |

| | |
|---|---|
| `method-signature` | Specifies a subclass of `java.lang.Object` as the type of the composite component's attribute.<br><br>The `method-signature` element declares that the composite component attribute is a method expression.<br><br>The `type` attribute and the `method-signature` attribute are mutually exclusive.<br><br>If you specify both, `method-signature` is ignored. |

The default type of an attribute is `java.lang.Object`.

---

Note - Method expressions are similar to value expressions, but rather than supporting the dynamic retrieval and setting of properties, method expressions support the invocation of a method of an arbitrary object, passing a specified set of parameters, and returning the result from the called method (if any).

---

| | |
|---|---|
| `type` | Specifies a fully qualified class name as the type of the attribute.<br><br>The `type` attribute and the `method-signature` attribute are mutually exclusive.<br><br>If you specify both, `method-signature` is ignored.<br><br>The default type of an attribute is `java.lang.Object`. |

The following code snippet defines a composite component attribute and assigns it a default value:

```
<composite:attribute
name="username" default="admin"/>
```

The following code snippet uses the `method-signature` element:

```
<composite:attribute
name="myaction"
method-signature=
"java.lang.String action()"
/>
```

The following code snippet uses the `type` element:

```
<composite:attribute
name="dateofjoining"
type="java.util.Date"
/>
```

# Invoking a Managed Bean

To enable a composite component to handle server-side data, you can invoke a managed bean in one of the following ways:

- Pass the reference of the managed bean to the composite component
- Directly use the properties of the managed bean

The example application described in The `compositecomponentlogin` Example Application shows how to use a managed bean with a composite component by passing the reference of the managed bean to the component.

# Validating Composite Component Values

JavaServer Faces provides the following tags for validating values of input components.

These tags can be used with the `composite:ValueHolder` or with the `composite:EditableValueHolder` tags.

# Table 13-2 lists commonly used validator tags.

Table 13-2 Validator Tags

| Tag Name | Description |
|---|---|
| `f:validateBean` | Delegates the validation of the local value to the Bean Validation API. |
| `f:validateRegex` | Uses the `pattern` attribute to validate the wrapping component.<br>The entire pattern is matched against the `String` value of the component.<br>If it matches, it is valid. |

| | |
|---|---|
| `f:validateRequired` | Enforces the presence of a value.<br><br>Has the same effect as setting the `required` element of a composite component's attribute to true. |

# The `compositecomponentlogin` Example Application

The `compositecomponentlogin` application creates a composite component that accepts a username and password.

The component interacts with a managed bean.

The component stores the username and password in the managed bean, retrieves the values from the bean, and displays these values on the Login page.

The `compositecomponentlogin` application has a composite component file, a using page, and a managed bean.

# The Composite Component File

The composite component file is an XHTML file.

It has a `composite:interface` section that declares the labels for the username, password, and login button.

It also declares a managed bean, which defines properties for the username and password.

```
<composite:interface>
<composite:attribute
name="namePrompt"
default="username"
/>
<composite:attribute
name="passwordPrompt"
default="password"
/>
```

```
<composite:attribute
name="loginButtonText"
default="login"
/>
<composite:attribute
name="loginAction"
method-signature=
"java.lang.String action()"
/>
</composite:interface>
```

# The composite component file accepts input values for the username and password properties of the managed bean.

```
<composite:implementation>
<h:form id="form">
<h:panelGrid columns="2">
#{cc.attrs.namePrompt}
<h:inputText id="name" value=
"#{cc.attrs.myloginBean.name}"
required="true"/>
```

```
#{cc.attrs.passwordPrompt}
<h:inputSecret id="password"
value=
"#{cc.attrs.myloginBean.password}"
required="true"
/>
</h:panelGrid/>
<p>
<h:commandButton id="loginButton"
value="#{cc.attrs.loginButtonText}"
action="#{cc.attrs.loginAction}"/>
```

```
</p>
</h:form>
</composite:implementation>
```

# The Using Page

The using page in this example application is an XHTML file that invokes the login composite component file along with the managed bean.

```
<div id="compositecomponent">
<ez:LoginPanel
myloginBean="#(myloginBean)"
loginAction="#(myloginBean.login)"
>
</ez:LoginPanel>
</div>
```

# The Managed Bean

The managed bean defines a method called `login`, which retrieves the values of the username and password.

```
@ManagedBean
@RequestScoped
public class myloginBean{
private String name;
```

```
private String password;
public myloginBean() {
this.name = "";
this.password = "";
}
public myloginBean
(String name, String password){
this.name = name;
this.password = password;
}
public String getPassword()
```

```
{ return password; }
public void setPassword
(String newValue)
{ password = newValue; }
public String getName()
{ return name; }
public void setName
(String newValue)
{ name = newValue; }

public String login() {
```

```
String msg =
"Your username and password are " +
" username: " + getName() +
" password: " + getPassword();
return msg;
}
}
```

# To Build, Package, Deploy, and Run the `compositecomponentlogin` Example Application Using NetBeans IDE

1. From the File menu, choose Open Project.

2. In the Open Project dialog, navigate to *tut-install*`/examples/web/`.

3. Select the `compositecomponentlogin` folder.

4. Select the Open as Main Project checkbox.

5. Click Open Project.

6. In the Projects tab, right-click `compositecomponentlogin` and select Run.

# The Login page opens in a browser window.