

Actions in NetUI

Table of contents

1 Introduction.....	2
2 Creating Actions.....	2
2.1 Creating an action through an annotation.....	2
2.2 Creating an action through a method.....	3
3 Raising Actions.....	4
3.1 Raising actions through NetUI JSP tags.....	4
3.2 Raising actions from JavaServer Faces components/command-handlers.....	4
3.3 Raising actions from other actions.....	5
3.4 Raising actions through URLs.....	5

1. Introduction

Actions are a core piece of the NetUI framework. They are Page Flow methods or annotations which *make navigational decisions* and *execute controller logic*. This document shows how to create and invoke actions.

2. Creating Actions

There are two ways to create an action in a Page Flow (or Shared Flow) [controller](#):

- through an annotation, or
- through a method.

2.1. Creating an action through an annotation

The simplest way to create an action is through the [@Jpf.SimpleAction](#) annotation at the class level of a Page Flow or Shared Flow controller:

```
@Jpf.Controller(
    simpleActions={
        @Jpf.SimpleAction(name="someAction", path="somePage.jsp")
    }
)
```

In the above example, the action `someAction` will navigate to `somePage.jsp` when it is invoked. As you can imagine, this simple form of the annotation is useful when an action only needs to navigate somewhere, without running complex logic.

The [@Jpf.SimpleAction](#) annotation also has a more powerful form that lets you define *conditions* that cause navigation to different places. The conditions are defined using JSP 2.0-style expressions. Consider the following annotation:

```
@Jpf.SimpleAction(
    name="someAction",
    path="default.jsp",
    conditionalForwards={
        @Jpf.ConditionalForward(condition="{pageFlow.advancedMode}",
    path="advanced.jsp"),
        @Jpf.ConditionalForward(condition="{param.alternate=='yes'}",
    path="alternate.jsp")
    }
)
```

Here, if the page flow's `advancedMode` property is `true` (i.e., it has a public method `getAdvancedMode` which returns `true`), then `someAction` will navigate to `advanced.jsp`. If there is a request URL parameter "alternate" set to "yes", then

`alternate.jsp` will be the destination. Finally, if neither of these conditions is true, then the default destination `default.jsp` is used.

For a list of implicit objects available for these expressions, see [JSP Implicit Objects](#) and [NetUI Implicit Objects](#). Note that page-related implicit objects `pageContext`, `pageScope`, and `pageInput` are not available here.

2.2. Creating an action through a method

To create an action through a method, you make a method that returns [Forward](#) and which is annotated with [@Jpf.Action](#), e.g.,

```
@Jpf.Action(  
    forwards={  
        @Jpf.Forward(name="somePage", path="somePage.jsp")  
    }  
)  
public Forward someAction()  
{  
    return new Forward("somePage");  
}
```

This action `someAction` forwards to `somePage.jsp`. The method has access to any member state in the controller class, and it can of course perform any other logic it needs to perform.

Note:

You do not *have* to return a [Forward](#) object. If your action returns `null`, then no navigation will occur. As an example, the following action method writes out "hello" to the response without navigating to another page:

```
@Jpf.Action  
public Forward writeResponse()  
{  
    getResponse().getWriter().print("hello");  
    return null;  
}
```

To create an action that accepts a *form bean*, simply add a single form bean argument:

```
@Jpf.Action(  
    forwards={  
        @Jpf.Forward(name="somePage", path="somePage.jsp")  
    }  
)  
public Forward submitForm(MyFormBean bean)  
{  
    // perform logic that uses the form bean  
    return new Forward("somePage");  
}
```

Here, `MyFormBean` is just any JavaBean class; there are no particular requirements on it. For information on how to post data to a form bean from a JSP, see [NetUI JSP Overview](#).

3. Raising Actions

You can raise actions through NetUI JSP tags, through components/command-handlers in JavaServer Faces pages, from other actions in the same page flow, or, in general, through URLs.

3.1. Raising actions through NetUI JSP tags

The following tags all support the `action` attribute:

- [Anchor](#)
- [AnchorCell](#)
- [Area](#)
- [Button](#)
- [Form](#)
- [ImageAnchor](#)
- [ImageAnchorCell](#)
- [TreeItem](#)

In general, you set the `action` attribute to the desired action, which will be raised when you click the link, submit the form, etc.

```
<netui:anchor action="someAction">Click me to run someAction</netui:anchor>
```

3.2. Raising actions from JavaServer Faces components/command-handlers

If you have NetUI/JSF integration enabled according to instructions [here](#), you can simply raise actions using the `action` attribute on JSF `command*` components, e.g.,

```
<h:commandLink action="someAction" value="Raise action someAction"/>
```

You can even raise a Page Flow action from a *command handler*. Say you have a JSF `commandLink` component that binds to method `myCommandHandler` in the page's backing bean:

```
<h:commandLink action="#{backing.myCommandHandler}" value="Raise an action"/>
```

Now, your command handler method can choose which Page Flow action to raise:

```
private SearchForm searchForm = ...;
```

```
@Jpf.CommandHandler(  
    raiseActions = {  
        @Jpf.RaiseAction(action="actionOne"),  
        @Jpf.RaiseAction(action="actionTwo", outputFormBean="searchForm")  
    }  
)  
public String myCommandHandler()  
{  
    if (...)  
        return "actionOne";  
    else  
        return "actionTwo";  
}
```

For more information on NetUI/JSF integration, see the [documentation](#) and the [sample](#).

3.3. Raising actions from other actions

To raise an action from another action in the *same* controller class ("action chaining"), simply use the `action` attribute on a [@Jpf.Forward](#) or a [@Jpf.SimpleAction](#) annotation, e.g.,

```
@Jpf.Forward(name="toAnotherAction", action="anotherAction")
```

or,

```
@Jpf.SimpleAction(name="chain", action="anotherAction")
```

3.4. Raising actions through URLs

In general, you can raise an action through a URL of the following pattern:

```
http://myserver/mywebapp/page-flow-directory/action-name.do
```

For example, if your page flow is in directory `/foo/bar` within the webapp, an action URL for `someAction` would look like this:

```
http://myserver/mywebapp/foo/bar/someAction.do
```

Of course, if your browser URL bar is already on a page in `/foo/bar`, then an action URL for `someAction` would simply be:

```
someAction.do
```