Shared Flow

Table of contents

1. Introduction

A shared flow (a kind of controller class) provides a place for actions, exception handlers and data that the developer wants to make available to multiple page flows. Shared flows are most useful for accessing shared state, for shared/templated user interface, and when you cannot change your controller class hierarchy.

Shared flows can reside anywhere in your web app, and can be referenced from other controller files or from pages.

> **Note:**
> Both shared flows and Page Flow Inheritance offer ways to share actions and exception handlers. See Shared Flow vs. Inheritance for some guidelines on when to use each one.

2. Shared Flow Basics

The Shared Flow feature has the following basic properties, which are described in more detail below.

- A shared flow is *referenced* by any page flow that wants to share its actions, exception handlers, and data.
- When you hit a page flow, you are guaranteed to have access to a *single instance of each of its referenced shared flows*. These shared flow instances are stored in the user session, and they are not destroyed until the session ends, or until they are destroyed explicitly.
- A page flow or any of its JSPs can raise actions on any referenced shared flow.
- Unhandled exceptions in the page flow are *automatically* tried on all referenced shared flows, until one of them handles the exception.

3. Creating a Shared Flow

To create a shared flow controller class:

1. add the `@Jpf.Controller` annotation to the class, and
2. extend the `SharedFlowController` class.

For example:

```
import org.apache.beehive.netui.pageflow.annotations.Jpf;
import org.apache.beehive.netui.pageflow.SharedFlowController;

@Jpf.Controller
public class MySharedFlow extends SharedFlowController
{
```

```
    ...
}
```

You can add actions and exception handling just like you would in a page flow controller.

4. Referencing Shared Flows from Page Flows

To share actions, exception handlers, and state from a shared flow, a page flow needs to *declare a reference* in its controller class, using the `@Jpf.SharedFlowRef` annotation. The following example shows two shared flow references being declared for page flow controller `SomeController`:

```
@Jpf.Contoller(
    sharedFlowsRefs={
        @Jpf.SharedFlowRef(name="sharedFlowOne",
type=example.SharedFlowClassOne.class),
        @Jpf.SharedFlowRef(name="sharedFlowTwo",
type=example.SharedFlowClassTwo.class)
    }
)
public class SomeController extends PageFlowController
```

Notice that this declaration *assigns a name* to each referenced shared flow. This name will be used throughout the page flow to shared actions and state. Throughout this document, we will refer to this as the **shared flow name**.

5. Shared Actions

You can raise Shared Flow 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. Basic instructions for raising actions can be found here. **The only difference between raising a shared flow action and raising a normal action is that you include the shared flow name along with the action name.** For example, the following JSP tag raises `someAction` on `example.SharedFlowClassOne`:

```
    <netui:anchor action="sharedFlowOne.someAction"/>
```

Notice that the pattern is ***shared-flow-name*.*action-name***. As another example, the following page flow action (part of `SomeController` in the above example) raises action `anotherAction` on `example.ShardFlowClassTwo`:

```
@Jpf.Action(
    forwards={
        @Jpf.Forward(name="sharedAction",
action="sharedFlowTwo.anotherAction")
    }
```

```
)
public Forward doSharedAction()
{
    return new Forward("sharedAction");
}
```

You always use the shared flow *name*, not its classname, to refer to it. This allows page flows to choose their own namespaces for shared flow actions, thus avoiding conflicts in action names.

6. Shared Flow Exception Handling

When an exception occurs and your page flow does not handle it (using `@Jpf.Catch`), *every referenced shared flow gets a chance to handle it.* The list of `@Jpf.SharedFlowRef` annotations is traversed in order, and the first shared flow with a matching `@Jpf.Catch` handles the exception.

For more information on exception handling, see this document.

7. Databinding to Shared Flow Properties

If your shared flow controller exposes properties, you can databind to those properties from JSPs or from JavaServer Faces pages, using the `sharedFlow` implicit object. The basic pattern is: **`sharedFlow.`*shared-flow-name*`.`*property-name***. For example, the following JSP fragment displays the value of the `someProperty` property on `example.SharedFlowClassOne` (for the page flow definition, see the example in Referencing, above):

```
This is the value: ${sharedFlow.sharedFlowOne.someProperty}
```

To be clear, what will be shown is the result of a method `getSomeProperty` on class `example.SharedFlowClassOne`. In another example, the following text box pushes a value into the `writableProperty` property when its form is submitted:

```
Edit this: <netui:textBox
dataSource="sharedFlow.sharedFlowOne.writableProperty"/>
```

Of course, you can do the same kind of binding from JavaServer Faces pages, using the JSF Expression Language:

```
<h:panelGrid rendered="#{sharedFlow.sharedFlowOne.showDetails}">
    ...
```

In that example, a `panelGrid` component decides whether to be rendered based on the value of the `showDetails` property in `example.SharedFlowOneClass`

8. Accessing a Shared Flow Directly

If you want, you can declare a member variable in your page flow controller which will be automatically initialized with a reference to a shared flow controller. This is *optional*; you do not have to do this in order to use a shared flow controller. If you do want to access the shared flow controller object directly, you annotate a member field using the `@Jpf.SharedFlowField` annotation. For example, the following page flow controller gets its `mySharedFlow` field automatically initialized:

```
@Jpf.Contoller(
    sharedFlowsRefs={
        @Jpf.SharedFlowRef(name="sharedFlowOne",
type=example.SharedFlowClassOne.class)
    }
)
public class SomeController extends PageFlowController
{
    @Jpf.SharedFlowField(name="sharedFlowOne")
    private example.SharedFlowClassOne mySharedFlow;
}
```

Note that you use the *name* of the shared flow reference in order to initialize the field. Once you have done this, you can use the shared flow controller object in any way you like (access its state, call methods, etc.).