NetUI State Management and Lifecycle

Table of contents

1. Introduction

1.1. State Management

1.1.1. Page Flow State Management

When you hit the URL for a page flow (or any of its actions, or any of its pages) for the first time, an instance of the controller class is created and stored in the user session. By default, it stays in the session as the *current page flow* until you hit another page flow. This means that while you continue to hit URLs in the page flow's URL space, it remains the current page flow. When you do hit another page flow, the original controller instance is destroyed. In other words, by default there is only a *single page flow controller* stored in the session at one time.

Note:

Nested page flows have special rules associated with them: when you hit a nested page flow, the current page flow is pushed aside, and it is restored when you return from the nested page flow. You can also *abnormally exit* a nested page flow by hitting a "regular" (non-nested) page flow while you're still in the nested flow. In that case, the original page flow (the one that was pushed aside) is discarded.

The auto-cleanup of a controller instance is normally helpful in keeping your session small and focused on the task at hand. In some cases, you may want to create a "long-lived" page flow controller that *never* gets destroyed (until the session itself ends). In this case, you simply set the `longLived` attribute to `true` on `@Jpf.Controller`:

```
@Jpf.Controller(longLived=true)
public class MyLongLivedPageFlow extends PageFlowController
{
    ...
}
```

Now, whenever this page flow is hit for the first time, it is stored in the session, and is not removed even when another page flow becomes the current page flow. Each time you hit the URL for this page flow (or any of its actions, or any of its pages), the *same instance* is restored.

You can remove this long-lived controller instance explicitly by calling its `remove()` method.

1.1.2. Shared Flow State Management

Whenever you hit a page flow, each of its referenced shared flow controllers is created and

stored in the session. If a shared flow controller of the right type already exists in the session, that instance is used instead. Once one is created, it is not removed unless you call its `remove()` method, or `PageFlowUtils.removeSharedFlow()`.

1.1.3. State Management for JavaServer Faces "Backing Beans"

When you hit a JSF page (e.g., "/mydir/mypage.faces"), the NetUI runtime looks for a class with the same name and package (e.g., `mydir.mypage`). If this class exists, is annotated with `@Jpf.FacesBacking`, and extends `FacesBackingBean`, then an instance is created and stored in the session. It is removed from the session on the next request that is not for the same page.

See Java Server Faces for more details on JSF integration with NetUI.

1.2. Lifecycle

All NetUI-managed objects (page flow controllers, shared flow controllers, JavaServer Faces "backing beans") are driven through a lifecycle, with callbacks in the appropriate places. This lifecycle always includes:

- **`onCreate`** - when the object is created by the runtime.
- **`onDestroy`** - when the object is "destroyed" (removed) by the runtime.

To run code at either of these points in the lifecycle, you simply override the appropriate method (`onCreate` or `onDestroy`), e.g.,

```
@Jpf.Controller
public class MyPageFlow extends PageFlowController
{
    protected void onCreate()
    {
        // do something to initialize this page flow controller
    }

    ...
}
```

1.2.1. Controller Lifecycle

Flow controllers (page flow controllers and shared flow controllers) have additional methods as part of their lifecycle:

- **`beforeAction`** - before any action is run.
- **`afterAction`** - after any action is run.

Again, to run code at either of these points, override the appropriate method, e.g.,

```
@Jpf.Controller
public class MyPageFlow extends PageFlowController
{
    protected void beforeAction()
    {
        log.debug("before action " + getCurrentActionName() + ", request "
+ getRequest().getRequestURI());
    }

    ...
}
```

Additionally, nested page flows have an additional lifecycle method:

- **`onExitNesting`** - when the page flow is exiting nesting (through a `returnAction` on `@Jpf.Forward` or `@Jpf.SimpleAction`).

1.2.2. JavaServer Faces "Backing Bean" Lifecycle

JSF backing beans (extended from `FacesBackingBean`) have one additional lifecycle method:

- **`onRestore`** - when the backing bean is being restored (along with the page itself) through `navigateTo`=`currentPage` or `navigateTo`=`previousPage` on `@Jpf.Forward` or `@Jpf.SimpleAction`.