NetUI Control Container

Table of contents

1. Introduction

This document describes how the NetUI page flow runtime implements the container for hosting controls. Controls run within a runtime container which provides services to controls. Within the page flow runtime, the org.apache.beehive.controls.api.context.ControlBeanContext is the base class that provides the extended services to contained controls. The container is managed by the page flow runtime. This document describes the details of the contract that is maintained for controls hosted in page flows by the implementation.

2. Description of the Control Container

This section describes the implementation of the Control container within the page flow runtime.

2.1. Scope of the Control

There are three scopes that the control container implementation provides for controls. The first is at the page flow level. Each page flow, `PageFlowController`, provides its own `ControlBeanContext` to the controls which are defined within the scope of that page flow. The second scope is the shared flows, `SharedFlowController`. All shared flows and the `GlobalApp` share a single `ControlBeanContext`. The final scope is for JSF Backing Beans, `FacesBackingBean`. Just as with page flows, faces backing beans have a `ControlBeanContext` that is scoped to their life time. The result is that the `ControlBeanContext` containing controls has the same lifetime as the object that defines and uses the control instances.

2.2. Single Threaded Page Flow Code

There are three places in the handling of a request where page flow code provides synchronization of multiple threads. Page flows and shared flows are scoped into a `ServletSession`. It is possible to have multiple threads running. This can happen if you have multiple browser windows sharing a session, or if you have multiple requests from a single page containing frames. Multiple requests can be generated through HTML frames or AJAX calls. The page flow runtime insures that multiple threads are not executing code inside of a page flow. A shared flow can potentially have multiple threads running through it which will be described below.

There are three synchronization points in the page flow runtime:

- *onCreate* -- The `onCreate` event is synchronized and only one thread will ever pass through this method during the life of the page flow.

- *beforeAction / action / afterAction* -- These three methods are run in a single synchronization block meaning these method will run together within one thread, without another thread running through the page flow.
- *JSP Rendering* -- The page flow runtime synchronizes on the current page flow during JSP rendering. This prevents a thread from running and action on a page flow while another thread is rendering a JSP which may be accessing page flow state.

Each of these synchronization points will run the resource events on the controls contained inside of the container. This will cause the `onAcquire` and `onRelease` resources events to be triggered on all of the controls within the container. These events will be run on the current page flow and also the shared flow. In reality, the `onAcquire` method is run before the first method invocation is done on a control. `onRelease` will only be run if the `onAcquire` mehtod has been run. The shared flow `ControlBeanContext` has a lock associated with it that must be obtained before user code can be run.

The synchronization point result in the following, all access to a control is single threaded. If a control acquires a resource such as a JDBC connection, it will only be used for a single request. This model also ensures that shared flows are accessed in a single threaded model when there is an instance of a control in any shared flow because of the lock associated with the shared flow `ControlBeanContext`. Finally, if there is a shared flow `ControlBeanContext` we will serialize all threads within a session when they run through user code. This ensures a single threaded model for controls defined in shared flows.

2.3. Programatic Creation of Controls

The ControlBeanContext is lazily created when possible. When a page flow is created, the page flow is searched for fields with `@control` annotations. If any of these are found, the `ControlBeanContext` is created. The shared flow `ControlBeanContext` is created when the first shared flow containing a control annotation is created.

For page flows that want to create a control programmatically using `java.beans.Beans.instantiate`, you must ensure that the context has been created. The following two lines of code will create the `ControlBeanContext` and make sure the `beginContext` method is called correctly.

```
PageFlowControlContainer pfcc =
PageFlowControlContainerFactory.getControlContainer(getRequest(),getServletContext());
pfcc.createAndBeginControlBeanContext(this,getRequest(),getResponse(),getServletContext
```