Building a Common Navigator Framework (CNF) Viewer Part III: Configuring Menus

ACTION IS ELOQUENCE.

-- **VOLUMNIA**

CORIOLANUS, ACT 3, SCENE 2

In earlier posts, we walked through how to construct a Common Navigator Framework (CNF) viewer and a basic extension to render property files. Now we will walk through how to configure the menu for a CNF viewer and how to add actions to our viewer to manipulate our content.

Overview

There are two basic options for adding actions to a Common Navigator Framework (CNF) viewer:

- Contribute actions using **org.eclipse.ui.popupMenus** as *objectContribution*s or *viewerContribution*s. The **...popupMenus** extension point allows you to contribute individual action delegates throughout the Eclipse Workbench. CNF viewers can be configured to honor these contributions (which is the default) or to ignore them. In the case of the *Project Explorer* contributed by Platform/UI, object and viewer contributions are honored. (To be covered in detail in Part IV.)
- Contribute actions using **org.eclipse.ui.navigator.navigatorContent** as *actionProvider*s. Sometimes, clients require more programmatic control over exactly what actions are contributed to a given menu in a particular context, as well as what retargetable actions are configured based on the current selection.

CNF *Action Provider*s are only honored by CNF viewers. (To be covered in detail in Part V.)

Configuring Menu Structure

In Part I, we defined our example <viewer /> element, and popup menus were briefly mentioned.

Recall that there are two means to configure a popup menu. The first possibility is simply to specify a value for the *popupMenuId* of the <viewer /> element in **org.eclipse.ui.navigator.viewer**. The second possibility is to take a more active role in configuring the insertion points the menu supports.

Tip: It is an error condition if both the popupMenuId is specified as well as the popupMenu element.

If we were to set the *popupMenuId*, we might have something like the following:

```
<!-- Declare the viewer configuration, and the default content/action bindings -->
<extension
      point="org.eclipse.ui.navigator.viewer">
   <viewer
         popupMenuId="org.eclipse.ui.examples.navigator.view.menu"
         viewerId="org.eclipse.ui.examples.navigator.view"/>
```

The identifier can be used by extenders of **org.eclipse.ui.popupMenus** to add *viewerContributions* specific to the view; however, no identifier is required for *objectContributions*.

When only the *popupMenuId* attribute is specified, the default set of insertion points are automatically configured in the popup menu. These are documented in the schema reference for **org.eclipse.ui.navigator.viewer**, but are duplicated here for convenience:

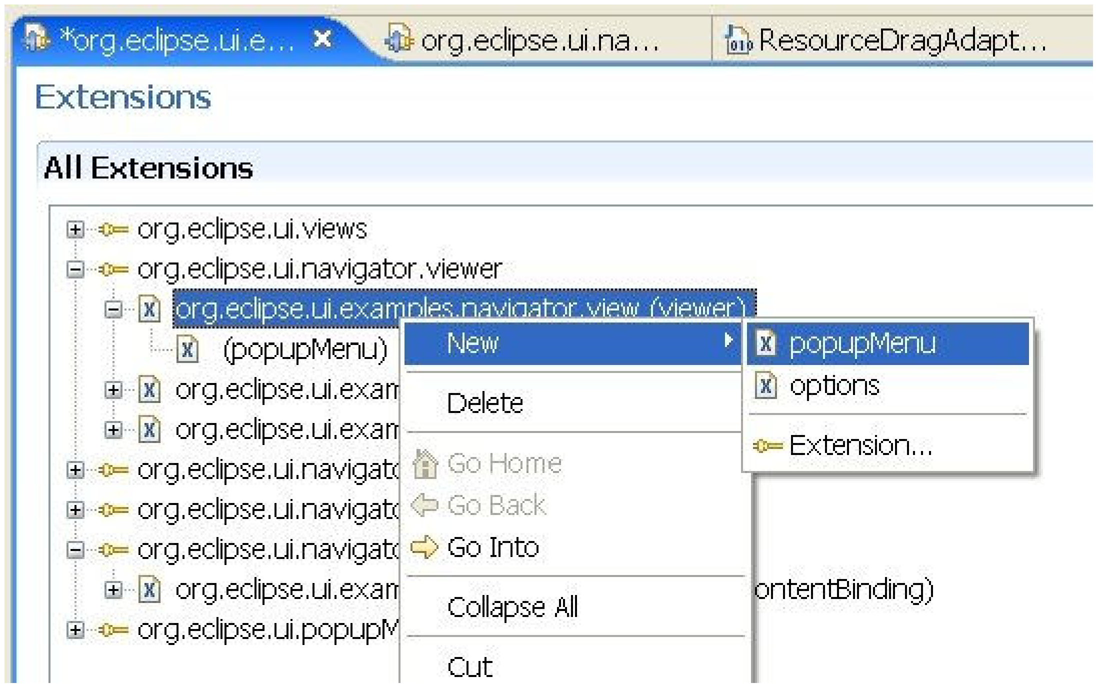
```
"group.new" separator="true"
"group.goto"
"group.open" separator="true"
"group.openWith"
"group.edit" separator="true"
"group.show" separator="true"
"group.reorganize"
"group.port"
"group.generate" separator="true"
"group.search" separator="true"
"group.build" separator="true"
"additions" separator="true"
"group.properties" separator="true"
```

These values can be used by the *menubarPath* attribute of elements in **org.eclipse.ui.popupMenus**.

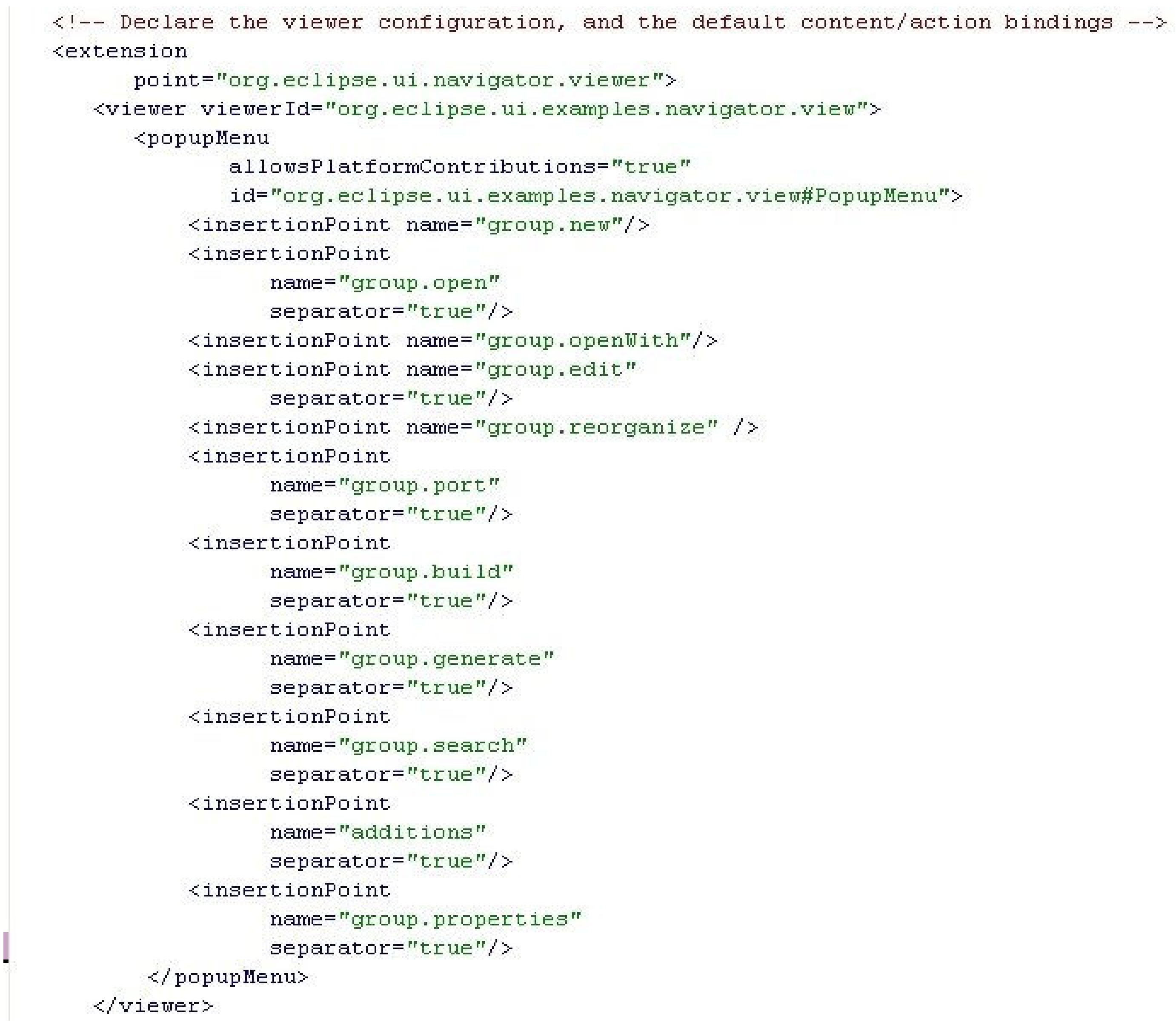
These values can also be used when programmatically using or searching for menu items (see *org.eclipse.jface.action.IContributionManager*, particularly *insertAfter(), insertBefore(), and appendToGroup()*.

All of the string constants ("group.*") are immortalized in *org.eclipse.ui.navigator.ICommonMenuConstants* for programmatic usage.

Alternatively, we could choose to define a <popupMenu /> element as a child of <viewer />, where we could then define our own mix of insertion points. Remember, that you can use the *Extensions* tab of the *Plug-in Manifest Editor* to drive the creation of extension elements from the menu.



For our example, we can use the same menu configuration as the *Project Explorer:*



```
<!-- Declare the viewer configuration, and the default content/action bindings -->
<extension
      point="org.eclipse.ui.navigator.viewer">
   <viewer viewerId="org.eclipse.ui.examples.navigator.view">
      <popupMenu
            allowsPlatformContributions="true"
            id="org.eclipse.ui.examples.navigator.view#PopupMenu">
         <insertionPoint name="group.new"/>
         <insertionPoint
               name="group.open"
               separator="true"/>
         <insertionPoint name="group.openWith"/>
         <insertionPoint name="group.edit"
               separator="true"/>
         <insertionPoint name="group.reorganize" />
         <insertionPoint
               name="group.port"
               separator="true"/>
         <insertionPoint
               name="group.build"
               separator="true"/>
         <insertionPoint
               name="group.generate"
               separator="true"/>
         <insertionPoint
               name="group.search"
               separator="true"/>
         <insertionPoint
               name="additions"
               separator="true"/>
         <insertionPoint
               name="group.properties"
               separator="true"/>
       </popupMenu>
   </viewer>
```

Each <insertionPoint /> element indicates a GroupMarker that is accessible through the *IContributionMenu* API. Where ever the *separator* attribute is set to **true**, the menu will render a horizontal line. The menus in Eclipse are smart enough to only render the line if necessary, so two lines back to back only renders as one horizontal line.

Also, take note of the *allowsPlatformContributions="true"* attribute of <popupMenu /> element. By default, CNF viewers are enabled to honor **..popupMenus** contributions. Setting this attribute to **false** will restrict the menu to only honoring *Action Providers*.

The example above doesn't do anything fancy, but in your own CNF viewer, you could define your own menu insertion point ("group.example") where ever it makes sense for your scenarios. The recommended naming convention is "group.*", as the above insertion points follow, with the exception of the special *additions* group.

Some comments on extending or manipulating the set of insertion points on a CNF menu. It is not possible to declaratively manipulate or extend the set of insertion points for an already configured viewer. It is possible to programmatically add new *GroupMarkers* or *Separators* through *Action Providers*, but this practice should be used with care. We will get into the details of how this might be done in *Part V*, but for now just heed the warning that this practice should be used with caution, and only when taking advantage of the *dependsOn* attribute of the <actionProvider /> element, where the value indicates the identifier of an *Action Provider* that adds extra insertion points to the menu. Insertion points should **never** be programmatically removed as other extensions may depend on them. If your own custom viewer has no use for a particular insertion point, then you can leave it out when you define your own <popupMenu /> element.

Summary

In this article, we have discussed how to configure a menu for a Common Navigator Framework (CNF) viewer. In the next few articles, we will discuss how we can add to this menu using **org.eclipse.ui.popupMenus** and **org.eclipse.ui.navigator.navigatorContent/actionProvider**.

ABOUT THE AUTHOR

MICHAEL ELDER
RESEARCH TRIANGLE PARK, NORTH CAROLINA, UNITED STATES

Michael has been a Java developer since 1999 and currently contributes to Eclipse in open source and commercial venues. Most recently, Michael has contributed the Common Navigator Framework (CNF) to Eclipse 3.2. Prior to that, he helped design several of the frameworks and API in the Web Tools Platform (WTP), and continues to contribute fixes to WTP through the open source process. Michael is currently part of the Services Oriented Architectures (SOA) Tools Team for IBM Rational® based in Research Triangle Park, NC.