ADF Code Corner

020. Expanding an af:tree node by clicking onto the node label

twitter.com/adfcodecorner

Abstract:

Tree nodes in ADF Faces RC are expanded at runtime by clicking onto the plus icon next to the folder name. For improved usability you may want to allow users to click onto the folder name as well to expand the a node; same for closing a folder node. This functionality, which is not provided by default, can be added using the ADF Faces RC client framework using a little bit of JavaScript.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
24-APR-2008

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

ADF Faces RC components have a client and server side component set. Usually the client side is referred to as the client framework because it not only contains the component object but also a set of APIs to work with the ADF Faces RC framework. This how-to explains how to use the client side framework to expand a tree node in ADF Faces RC by the application user clicking onto the tree label.

Code Sample

Starting from an ADF Tree that you built with Oracle ADF Faces, the following changes are needed on the page to get this example to work

1. The af:tree component needs to have a child component af:clientListener to respond to moue clicks

```
<af:tree id="tree1"
    value="#{bindings.DepartmentsView11.treeModel}" var="node"
    selectionListener="
          #{bindings.DepartmentsView11.treeModel.makeCurrent}"
    rowSelection="single">
  <f:facet name="nodeStamp">
    <af:outputText value="#{node}"/>
  </f:facet>
  <af:clientListener method="expandNode" type="selection"/>
</af:tree>
```

The clientListener configures the tree so it calls a local javaScript function "expandNode" when a node in the tree gets selected

2. The JavaScript code that gets added to the page uses the Adf event to determine the event source, which is the handle to the tree

```
</af:document>
  <f:facet name="metaContainer">
       <af:group>
         <![CDATA[
           <script>
               function expandNode(event){
                   var _tree = event.getSource();
                   rwKeySet = event.getAddedSet();
                   var firstRowKey;
                   for(rowKey in rwKeySet){
                      firstRowKey  = rowKey;
                       // we are interested in the first hit,
                       //so break out here
                       break;
                   }
                   if (_tree.isPathExpanded(firstRowKey)){
                        _tree.setDisclosedRowKey(firstRowKey,false);
                   }
                   else{
                       _tree.setDisclosedRowKey(firstRowKey,true);
                   }
              }
       </script> ]]>
       </af:group>
    </f:facet>
 </af:document>
```

Update 06/2010: Use the af:resource tag to add javaScrit instead of CDATA and metaContainer

The three elements highlighted in bold are important to mention: First of all, for better performance, all JavaScript code added to a page should be in the metaContainer of the af:document element. Second,

because the metaContainer cannot have JavaScript directly added, you must have a af:group element as the child of the metaContainer facet.

The expandNode() method has a single input argument, which is the event that is passed into it by the framework. the event source, as mentioned earlier, is the tree, of type AdfRichTree. The event also contains a set of added row keys, which in the case of the tree are the index of the three node.

To determine whether to expand or close the selected node, the isPathExpanded method is called. The isPathExpanded() method is inherited by the AdfRichTree component and comes from its AdfUITree parent class. Not that this information is important, but it gives you an idea of how the ADF Faces client framework is organized in objects and subclasses.

Known Issues

There is one issues with this sample that I am aware of and that I will follow up with. To close a previously expanded node by clicking on the same node label, users first have to click on another node.

Update 06/2010: If you need the tree node to close upon a second mouse click, use the code shown below

```
<af:resource type="javascript">
      function expandDiscloseNode(event){
         var _tree = event.getSource();
         rwKeySet = _tree.getSelectedRowKeys();
         var firstRowKey;
            for(rowKey in rwKeySet){
               firstRowKey  = rowKey;
               // we are interested in the first hit, so break out here
            break;
         }
         if (_tree.isPathExpanded(firstRowKey)){
             _tree.setDisclosedRowKey(firstRowKey,false);
         }
         else{
             _tree.setDisclosedRowKey(firstRowKey,true);
         }
       }
    </af:resource>
```

The page source needs to be changed as shown below.

```
<tree ...>
...
  <af:clientListener method="expandDiscloseNode"
                                          type="click"/>
  </af:tree>
```

What you should be aware of !

Make sure that all JavaScript you add to a page is properly formatted. If you add invalid code, e.g. trying to guess API names or not closing a function properly, then this shows at runtime, not at compile and not at design time. The way it shows is that you see the famous splash screen of death, which is the rotating image you see when starting your web application. Except for that this time the splash doesn't stop spinning. If you see this, go back and check your JavaScript. There exist plans in a next version of JDeveloper 11 to handle this issue more gracefully.

RELATED DOCOMENTATION

⊠	
⊠	
⊠	