ADF Code Corner

78. How-to programmatically expand trees and tree table components upon initial rendering and later

Abstract:

Initially expanding the ADF Faces tree and tree table components is a frequently asked requirements that I cover in sample #20, #21 and #61 on ADF Code Corner. This article is a different approach to sample #21 and allows developers to initially expand ADF bound trees and tree tables down to a defined level of depth. In addition, this solution works for ADF Faces views in JSPX pages and JSFF page fragments as it does not use a phase listener but a managed bean to determine the the tree and tree table disclosed keys.

twitter.com/adfcodecorner

Author:

Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
06-APR-2011

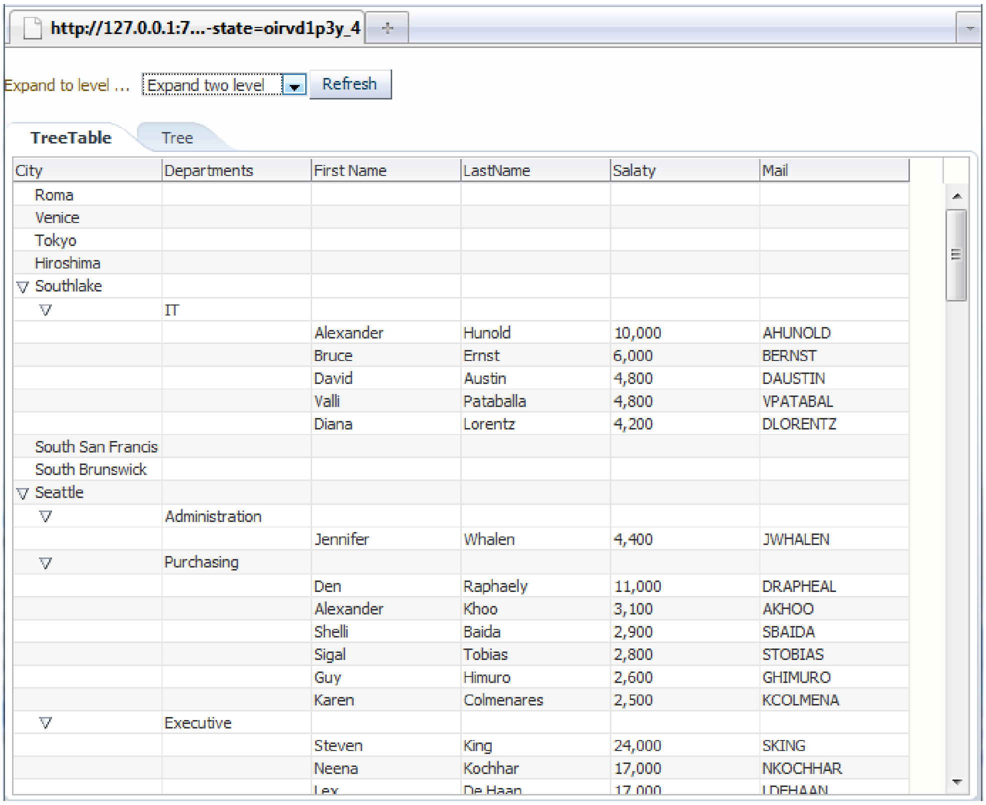
Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

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

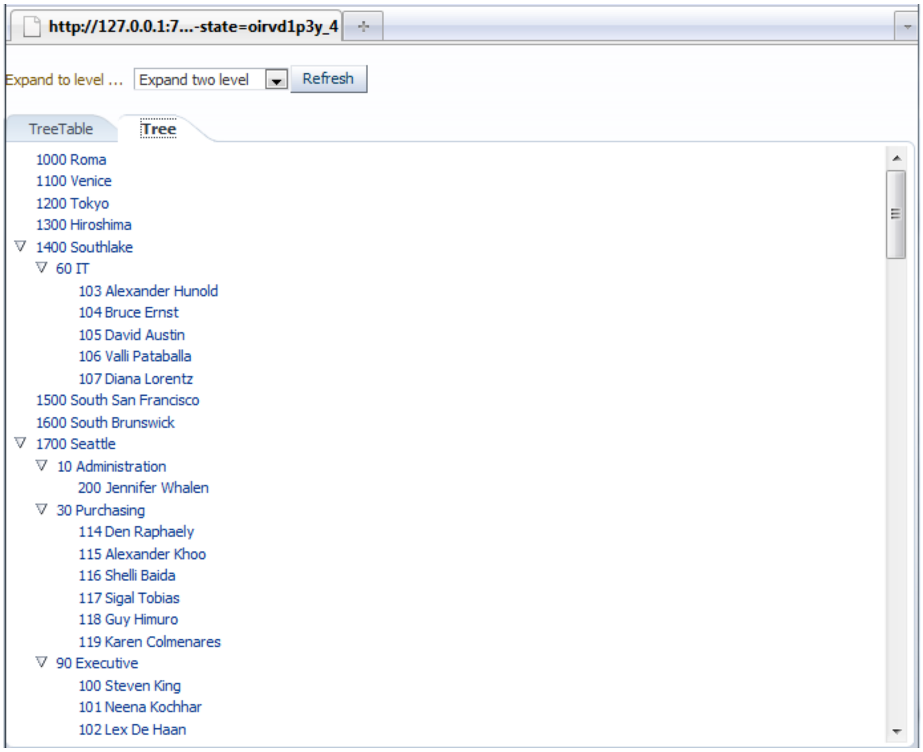
The images below show the sample at runtime. Based on a default value provided for the tree and tree table level to which nodes should be automatically expanded, the tree and tree table renders in expanded mode. Later, using the select one choice component on top of the view in combination with the *Refresh* button, users can change the disclose state dynamically.



Expand to level ... Expand two level

City	Departments	First Name	LastName	Salaty	Mail
Roma					
Venice					
Tokyo					
Hiroshima					
▽ Southlake					
▽	IT				
		Alexander	Hunold	10,000	AHUNOLD
		Bruce	Ernst	6,000	BERNST
		David	Austin	4,800	DAUSTIN
		Valli	Pataballa	4,800	VPATABAL
		Diana	Lorentz	4,200	DLORENTZ
South San Francis					
South Brunswick					
▽ Seattle					
▽	Administration				
		Jennifer	Whalen	4,400	JWHALEN
▽	Purchasing				
		Den	Raphaely	11,000	DRAPHEAL
		Alexander	Khoo	3,100	AKHOO
		Shelli	Baida	2,900	SBAIDA
		Sigal	Tobias	2,800	STOBIAS
		Guy	Himuro	2,600	GHIMURO
		Karen	Colmenares	2,500	KCOLMENA
▽	Executive				
		Steven	King	24,000	SKING
		Neena	Kochhar	17,000	NKOCHHAR
		Lex	De Haan	17,000	LDEHAAN

The same settings applied to a tree component is shown in the image below.

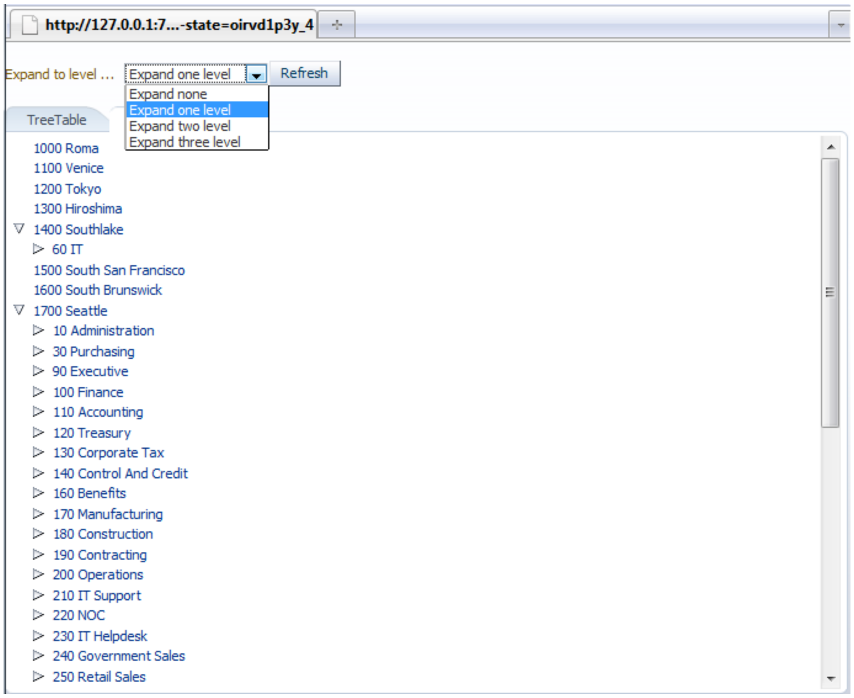


http://127.0.0.1:7...-state=oirvd1p3y_4

Expand to level ... Expand two level

```
    1000 Roma
    1100 Venice
    1200 Tokyo
    1300 Hiroshima
▽   1400 Southlake
    ▽ 60 IT
          103 Alexander Hunold
          104 Bruce Ernst
          105 David Austin
          106 Valli Pataballa
          107 Diana Lorentz
    1500 South San Francisco
    1600 South Brunswick
▽   1700 Seattle
    ▽ 10 Administration
          200 Jennifer Whalen
    ▽ 30 Purchasing
          114 Den Raphaely
          115 Alexander Khoo
          116 Shelli Baida
          117 Sigal Tobias
          118 Guy Himuro
          119 Karen Colmenares
    ▽ 90 Executive
          100 Steven King
          101 Neena Kochhar
          102 Lex De Haan
```

Changing the node level for disclosure then re-renders the tree.



http://127.0.0.1:7...-state=oirvd1p3y_4

Expand to level ... Expand one level

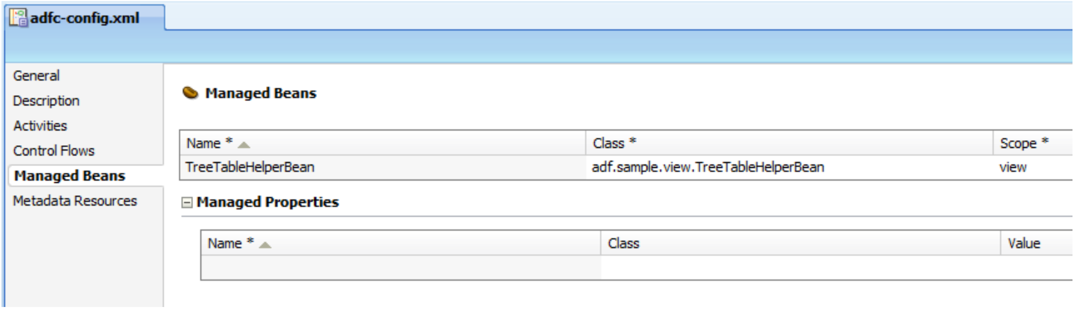
```
    1000 Roma
    1100 Venice
    1200 Tokyo
    1300 Hiroshima
▽   1400 Southlake
    ▷ 60 IT
    1500 South San Francisco
    1600 South Brunswick
▽   1700 Seattle
    ▷ 10 Administration
    ▷ 30 Purchasing
    ▷ 90 Executive
    ▷ 100 Finance
    ▷ 110 Accounting
    ▷ 120 Treasury
    ▷ 130 Corporate Tax
    ▷ 140 Control And Credit
    ▷ 160 Benefits
    ▷ 170 Manufacturing
    ▷ 180 Construction
    ▷ 190 Contracting
    ▷ 200 Operations
    ▷ 210 IT Support
    ▷ 220 NOC
    ▷ 230 IT Helpdesk
    ▷ 240 Government Sales
    ▷ 250 Retail Sales
```

The sample code is provided as an Oracle JDeveloper 11.1.1.4 works pace and can be downloaded as sample #78 from the ADF Code Corner website.

About the sample

The key to this solution is a managed bean that is configured in view scope so its internal variable state survives subsequent requests. The bean must be configured in the task flow definition file that holds the view that shows the tree or tree table component with the functionality introduced in this article. The managed bean exposes two methods:

- A method that returns an instance of `RowKeySetImpl` individually for the tree and the tree table component
- A method that can be called to set the depth of node levels until where the tree and tree table component should be expanded (disclosed)



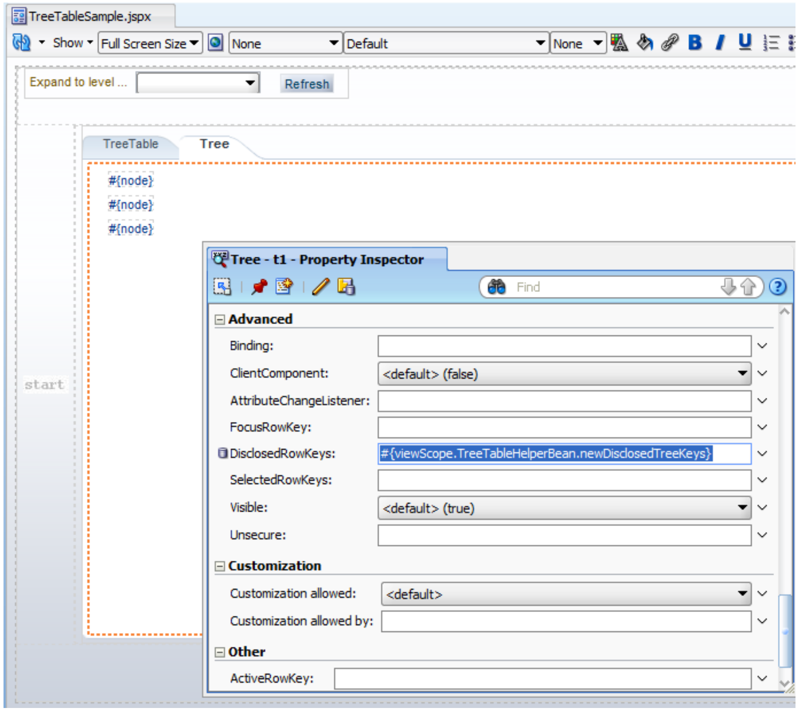
Managed Beans

Name *	Class *	Scope *
TreeTableHelperBean	adf.sample.view.TreeTableHelperBean	view

Managed Properties

Name *	Class	Value

The tree and tree table components have their **DiscloseRowKeys** property configured to point to the managed bean method that returns the instance of `RowKeySetImpl`.



Advanced

Binding:

ClientComponent: <default> (false)

AttributeChangeListener:

FocusRowKey:

DisclosedRowKeys: #{viewScope.TreeTableHelperBean.newDisclosedTreeKeys}

SelectedRowKeys:

Visible: <default> (true)

Unsecure:

Customization

Customization allowed: <default>

Customization allowed by:

Other

ActiveRowKey:

The content of the RowKeySetImpl instance is a collection of **keyPath**, which are lists containing the primary key paths to a node in the tree or tree table hierarchy. For ADF bound components, the keypath is accessible from the JUCtrlHierNodeBinding class that represents individual hierarchical nodes in the ADF binding. For the remainder of this article I expect the reader to understand how to build ADF bound trees and tree tables in Oracle ADF.

Note: Just in case, how to build ADF bound trees and tree tables is briefly covered in ADF Code Corner sample #50. Read page 5 onwards

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/50-synchromize-form-treeselection-169192.pdf

Managed bean Code

Below is the managed bean code that uses a recursive method call to parse the tree hierarchy. The example uses the tree component Id "t1" and the tree table component Id "tt1" to lookup the component instances on the ADF Faces view. The lookup code can be improved as explained in sample #58 on ADF Code Corner:

http://www.oracle.com/technetwork/developer-tools/adf/downloads/58-optimizedadffacescomponentsearch-175858.pdf

```
import java.util.ArrayList;
import java.util.List;


import javax.faces.component.UIViewRoot;
import javax.faces.context.FacesContext;

import oracle.adf.view.rich.component.rich.data.RichTree;
import oracle.adf.view.rich.component.rich.data.RichTreeTable;
import oracle.adf.view.rich.context.AdfFacesContext;

import oracle.jbo.uicli.binding.JUCtrlHierBinding;
import oracle.jbo.uicli.binding.JUCtrlHierNodeBinding;

import org.apache.myfaces.trinidad.model.CollectionModel;
import org.apache.myfaces.trinidad.model.RowKeySetImpl;

public class TreeTableHelperBean {

  //disclose state for tree tables
  private RowKeySetImpl newDisclosedTreeTableKeys = null;
  //disclose state for tree
  private RowKeySetImpl newDisclosedTreeKeys = null;

//allows you to configure the depth of the tree table
private int expandTreeToLevelLevel = 1;
```

```
public TreeTableHelperBean() {
  super();
}

public void setNewDisclosedTreeTableKeys(
                              RowKeySetImpl newDisclosedKeys) {
  this.newDisclosedTreeTableKeys = newDisclosedKeys;
}

public RowKeySetImpl getNewDisclosedTreeTableKeys() {
 if (newDisclosedTreeTableKeys == null) {
   newDisclosedTreeTableKeys = new RowKeySetImpl();
   FacesContext fctx = FacesContext.getCurrentInstance();
   UIViewRoot root = fctx.getViewRoot();
   //lookup the tree table component by its component ID
   RichTreeTable treeTable =
                       (RichTreeTable)root.findComponent("tt1");
   //if tree table is found
   if (treeTable != null) {
     //get the collection model to access the ADF binding layer for
     //the tree binding used
     CollectionModel model = (CollectionModel)treeTable.getValue();
     JUCtrlHierBinding treeBinding =
                  (JUCtrlHierBinding)model.getWrappedData();
     JUCtrlHierNodeBinding nodeBinding =
                          treeBinding.getRootNodeBinding();
     expandAllNodes(nodeBinding, newDisclosedTreeTableKeys, 0,
                    expandTreeToLevelLevel);
     }
   }
   return newDisclosedTreeTableKeys;
 }

  public void setNewDisclosedTreeKeys(
                             RowKeySetImpl newDisclosedTreeKeys) {
     this.newDisclosedTreeKeys = newDisclosedTreeKeys;
   }

   public RowKeySetImpl getNewDisclosedTreeKeys() {
     if (newDisclosedTreeKeys == null) {
       newDisclosedTreeKeys = new RowKeySetImpl();
       FacesContext fctx = FacesContext.getCurrentInstance();
       UIViewRoot root = fctx.getViewRoot();
       //lookup thetree component by its component ID
       RichTree tree = (RichTree)root.findComponent("t1");
      //if tree is found ....
```

```
    if (tree != null) {
      //get the collection model to access the ADF binding
      //layer for the tree binding used. Note that for this
      //sample the bindings used by the tree is different from
      //the binding used for the tree table
      CollectionModel model = (CollectionModel)tree.getValue();
      JUCtrlHierBinding treeBinding =
                      (JUCtrlHierBinding)model.getWrappedData();
      JUCtrlHierNodeBinding nodeBinding =
                              treeBinding.getRootNodeBinding();
      expandAllNodes(nodeBinding, newDisclosedTreeKeys, 0,
                     expandTreeToLevelLevel);
      }
    }
  return newDisclosedTreeKeys;
}


/*
 * Method that allows you to dynamically set the maximum level
 * until where the tree or tree table is disclosed. Note that
 * to use this from a rendered page, you need an additional method
 * that clears the current disclosed row keys
 */
public void setExpandTreeToLevelLevel(int expandTreeToLevelLevel) {
  this.expandTreeToLevelLevel = expandTreeToLevelLevel;
}

public int getExpandTreeToLevelLevel() {
    return expandTreeToLevelLevel;
}

/**
 * Recursive method to expand nodes to a pre-defined level
 *
 * @param nodeBinding the JUCtrlHierNodeBinding representing
 * the current node
 * @param disclosedKeys the RowKeySetImpl instance that holds
 * the keys to disclose
 * @param currentExpandLevel the current depth of the tree node
 * @param maxExpandLevel the max. number of levels to expand nodes
 * for
 */
```

```
    private void expandAllNodes(JUCtrlHierNodeBinding nodeBinding,
                                RowKeySetImpl disclosedKeys,
                                int currentExpandLevel,
                                int maxExpandLevel) {
      if (currentExpandLevel <= maxExpandLevel) {
        List<JUCtrlHierNodeBinding> childNodes =
                (List<JUCtrlHierNodeBinding>)nodeBinding.getChildren();
        ArrayList newKeys = new ArrayList();
        if (childNodes != null) {
          for (JUCtrlHierNodeBinding _node : childNodes) {
            newKeys.add(_node.getKeyPath());
            expandAllNodes(_node, disclosedKeys,
            currentExpandLevel + 1, maxExpandLevel);
         }
       }
        disclosedKeys.addAll(newKeys);
     }
    }

    //handle the case of the Refresh button being pressed. Reset the
    //tree and tree table disclosure state

    public String onRefresh() {
        FacesContext fctx = FacesContext.getCurrentInstance();
        UIViewRoot root = fctx.getViewRoot();
        AdfFacesContext adfFacesContext =
        AdfFacesContext.getCurrentInstance();

        //clear disclosed RowKeys
        newDisclosedTreeTableKeys =null;
        //PPR tree table
        RichTreeTable treeTable =
                      (RichTreeTable)root.findComponent("tt1");
        getNewDisclosedTreeTableKeys();
        adfFacesContext.addPartialTarget(treeTable);

        //reset tree keys
        newDisclosedTreeKeys = null;
        RichTree tree = (RichTree)root.findComponent("t1");
        getNewDisclosedTreeKeys();
        adfFacesContext.addPartialTarget(tree);

        return null;
    }
}
```

Download

The sample workspace can be downloaded from the ADF Code Corner website where it is sample #78.

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

You need to configure the database connection and point it to the HR schema of your local database (Oracle XE or enterprise edition both have the schema installed). The workspace is of Oracle JDeveloper 11.1.1.4, though the code is expected to be backward compatible.

RELATED DOCOMENTATION

☒	
☒	
☒	