ADF Code Corner
Oracle JDeveloper OTN Harvest 10 / 2011

Abstract:

The Oracle JDeveloper forum is in the Top 5 of the most active forums on the Oracle Technology Network (OTN). The number of questions and answers published on the forum is steadily increasing with the growing interest in and adoption of the Oracle Application Development Framework (ADF).

The ADF Code Corner "Oracle JDeveloper OTN Harvest" series is a monthly summary of selected topics posted on the OTN Oracle JDeveloper forum. It is an effort to turn knowledge exchange into an interesting read for developers who enjoy harvesting little nuggets of wisdom.

twitter.com/adfcodecorner

http://blogs.oracle.com/jdevotnharvest/

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
30-OCT-2011

Oracle ADF Code Corner OTN Harvest is a monthly blog series that publishes how-to tips and information around Oracle JDeveloper and Oracle ADF.

If you have questions, please post them to the Oracle OTN JDeveloper forum: http://forums.oracle.com/forums/forum.jspa?forumID=83

October 2011 Issue – Table of Contents

New Oracle ADF Mobile Strategy

At OOW 2011, Oracle's new ADF mobile strategy has been publicly announced. The press release summarizes ADF mobiles as follows:

"

- *Simplified mobile application development: enables mobile developers to develop once and deploy to multiple device platforms, Oracle ADF Mobile prevents mobile applications from having to be re-developed from the ground up whenever a new mobile platform needs to be supported.*
- *Strong native device services integration: allows mobile developers to create rich enterprise mobile applications that leverage the full capabilities of smart phones, including calendars, cameras and GPS.*
- *Tight integration with Oracle Fusion Middleware and Oracle Applications: enables Oracle customers to easily extend enterprise processes and data to mobile users.*
- *Re-use existing Oracle developer skills: Oracle developers can leverage existing Oracle ADF application development expertise to declaratively create mobile applications without having to learn new languages or development paradigms.*
- *Consistent performance and availability: Oracle ADF Mobile applications can operate regardless of network condition or availability.*

"

http://www.oracle.com/us/corporate/press/513001

A more detailed paper is available in PDF from

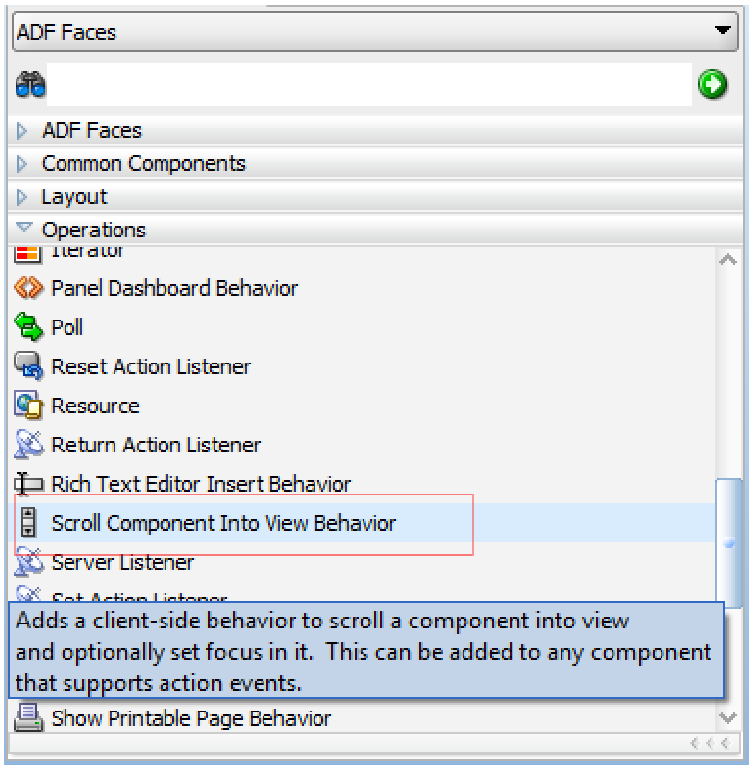
http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/513078.pdf

Setting an anchor in Oracle ADF views

Complex views often become big as well. In a desktop-style user interface, growing view sizes are usually handled through the use of collapsible panels, like `af:panelTabbed` or `af:panelAccordion`. In traditional page layouts, growing page sizes are handled by users scrolling down the browser. For better usability and orientation in a page, developers add anchor links that, when clicked scroll a specific area of a page into the view.

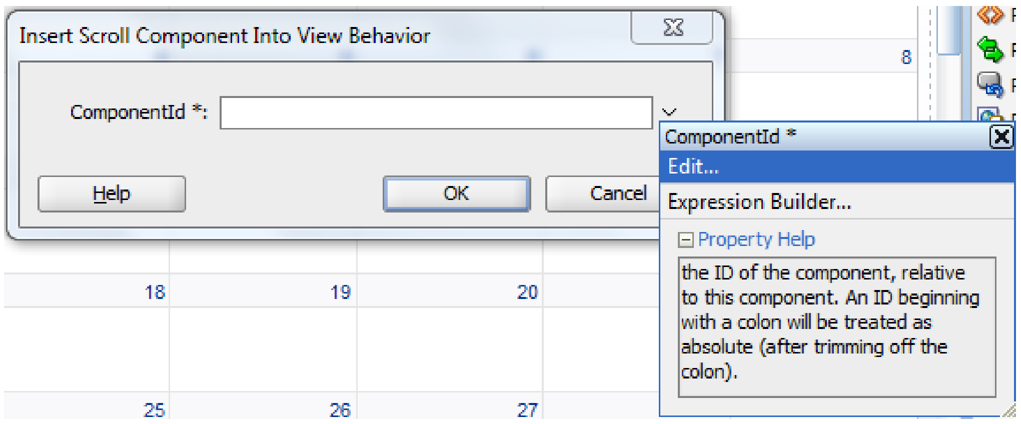
In Oracle ADF Faces, the `af:scrollComponentIntoViewBehavior` tag can be used to navigate to a specific UI component, e.g. a layout container.

To add the `af:scrollComponentIntoViewBehavior` to a command link on a page, drag it from the ADF Faces Component Palette's **Operation** category on top of the component.



Adds a client-side behavior to scroll a component into view and optionally set focus in it. This can be added to any component that supports action events.

After adding the behavior component to the `af:commandLink` link component, a dialog opens for you to select the component that should be moved into view when the link is pressed (make sure you set the **partialSubmit** property of the link to **true**)



the ID of the component, relative to this component. An ID beginning with a colon will be treated as absolute (after trimming off the colon).

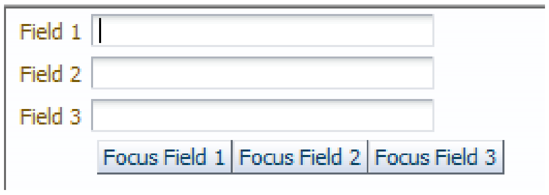
More information is provided in the ADF Faces tag documentation

http://download.oracle.com/docs/cd/E16162_01/apirefs.1112/e17491/tagdoc/af_scrollComponentIntoViewBehavior.html

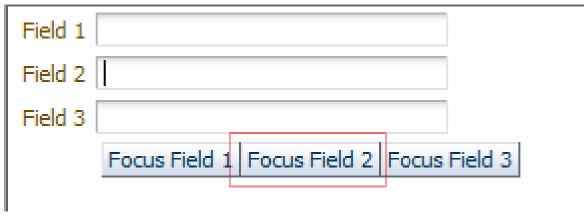
How to programmatically set focus on an input component

The `af:document` component has a property **InitialFocusId** that allows you to define the Id of an input component that should take focus when a page initially loads. But how do you set the component focus for pages that are already rendered? A use case for this is a partial submit that executes logic on the server, after which the cursor focus in the UI needs to be set to a specific input component of a page or page fragment. The solution to this use case is JavaScript that is generated on the server and executed on the client.

The simplified sample I wrote contains of three input text fields and three command buttons. Initially, using the `af:document` **InitialFocusId** property, the cursor focus is set to the first input text field.



The command button send a partial submit to a managed bean on the server. The managed bean reads the component Id of the input text field to put the focus to from a client attribute that is defined on each command button. Using the MyFaces Trinidad `ExtendedRenderKitService` class, the managed bean sends a JavaScript call to the client to change the current component focus.



Important: By default, ADF Faces does not render all its user interface components as JavaScript objects. It only renders those components as JavaScript objects that have behavior, like a table that can have its columns moved. Input text components, for example, are rendered in HTML only, which means that the ADF client side JavaScript framework will not find a handle to the component unless you tell ADF Faces to create one. For this, you set the **ClientComponent** property of the input text fields to **true**. This also is required for the `af:document` **InitialFocusId** property to work.

Below is the managed bean code that handles the command button action and that composes and executes the JavaScript to set the client side UI focus.

```
import javax.faces.component.UIViewRoot;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;

import oracle.adf.view.rich.component.rich.input.RichInputText;
import oracle.adf.view.rich.component.rich.nav.RichCommandButton;

import org.apache.myfaces.trinidad.render.ExtendedRenderKitService;
import org.apache.myfaces.trinidad.util.Service;

public class FocusBean {
    public FocusBean() {
  }


 public String onSetFocus(ActionEvent event) {
        RichCommandButton rcb = (RichCommandButton)event.getSource();
        String focusOn = (String)rcb.getAttributes().get("focusField");
        FacesContext fctx = FacesContext.getCurrentInstance();
        UIViewRoot viewRoot = fctx.getViewRoot();


        //search can be improved to include naming containers
        RichInputText rit =
             (RichInputText)viewRoot.findComponent(focusOn);

        if (rit != null) {
            String clientId = rit.getClientId(fctx);
            //compose JavaSCript to be executed on the client
            StringBuilder script = new StringBuilder();
            //use client id to ensure component is found if located in
            //naming container

            script.append("var textInput = ");
            script.append("AdfPage.PAGE.findComponentByAbsoluteId");
            script.append ("('"+clientId+"');");

            script.append("if(textInput != null){");
            script.append("textInput.focus();");
            script.append("}");
            //invoke JavaScript
            writeJavaScriptToClient(script.toString());
        }
    }

    //generic, reusable helper method to call JavaScript on a client
    private void writeJavaScriptToClient(String script) {
        FacesContext fctx = FacesContext.getCurrentInstance();
        ExtendedRenderKitService erks = null;
        erks = Service.getRenderKitService(
                             fctx, ExtendedRenderKitService.class);
        erks.addScript(fctx, script);
    }
}
```

As mentioned, the command buttons in the sample have a client attribute attached that I use to determine which input component should receive focus. The managed bean calls the client attribute here:

```
RichCommandButton rcb = (RichCommandButton)event.getSource();
String focusOn = (String)rcb.getAttributes().get("focusField");
```

In the page definition, the client attribute is defined as

```
<af:commandButton text="Focus Field 2" id="commandButton1"
                  partialSubmit="true"
                  actionListener="#{FocusBean.onSetFocus}">
  <af:clientAttribute name="focusField" value="it2"/>
</af:commandButton>
<af:commandButton text="Focus Field 3" id="cb1"
                  partialSubmit="true"
                  actionListener="#{FocusBean.onSetFocus}">
  <af:clientAttribute name="focusField" value="it3"/>
</af:commandButton>
```

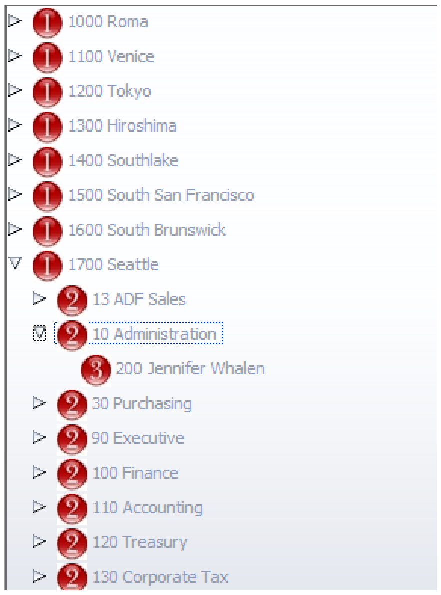
In your custom implementation of this use case, you for sure will have your own way of determining the next focus target. The server side code however doesn't change much.

Download:

The sample can be downloaded from the OTN Harvest blog (http://blogs.oracle.com/jdevotnharvest/):
Direct link: http://blogs.oracle.com/jdevotnharvest/resource/SetFocusToField.zip

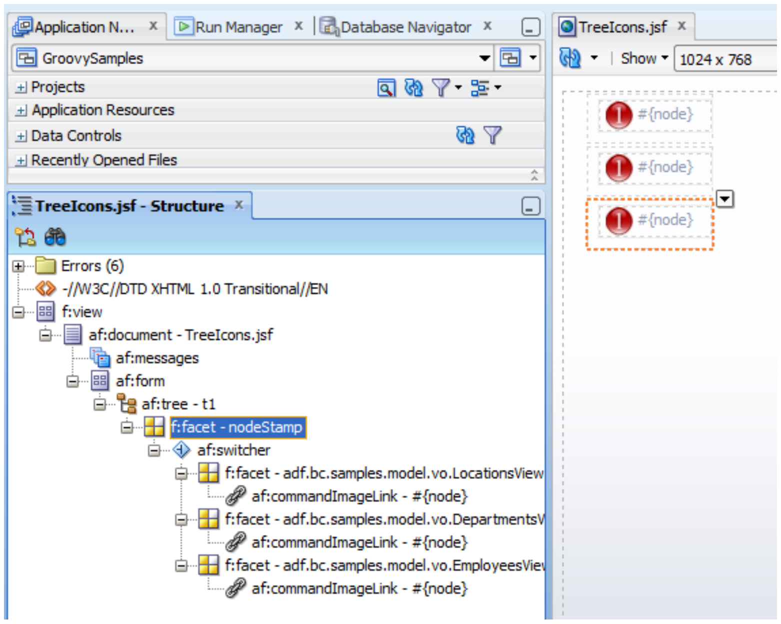
How to render tree node icon based on the tree level they in

Nodes of the `af:tree` component are stamped, which means that upon rendering, the UI component defined in the **nodeStamp** facet is printed repeatedly with different data values. If the requirement is to icons differently on each node level, then EL and the `af:switcher` component help as explained in the following.



The sample uses the Oracle HR sample schema Locations, Departments and Employees hierarchy. The tree nodes are printed as `af:commandImageLinks` because this gives me the option to print an icon and text within the same component, avoiding the need for a surrounding container (the **nodeStamp** facet can only have a single child component) that may mess with the tree layout. Because I have no use of the command link action, I set its **disabled** property to **false**.

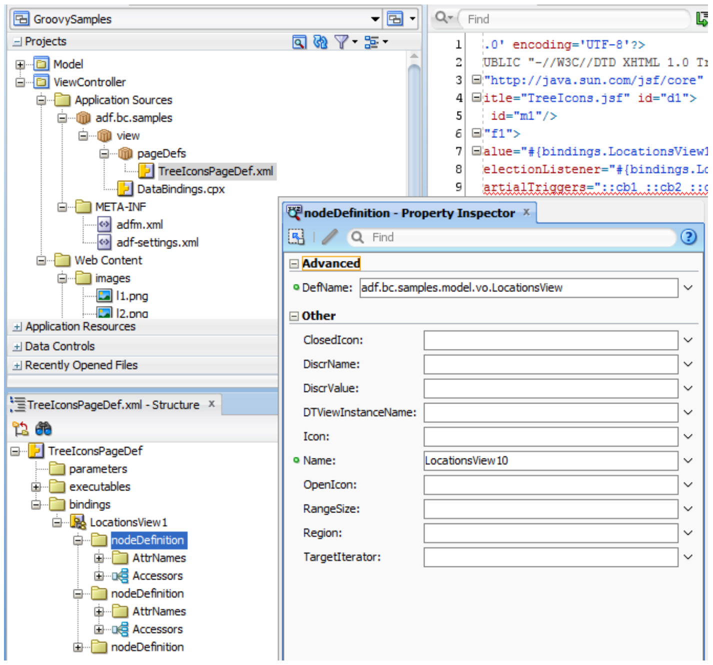
Note: The command image link labels are rendered in gray, which can be corrected to black color using skinning. Alternatively you can look for finding a different UI component to render the node (as said command image link is what worked for me the best).



```
TreeIcons.jsf - Structure
Errors (6)
-//W3C//DTD XHTML 1.0 Transitional//EN
f:view
  af:document - TreeIcons.jsf
    af:messages
    af:form
      af:tree - t1
        f:facet - nodeStamp
          af:switcher
            f:facet - adf.bc.samples.model.vo.LocationsView
              af:commandImageLink - #{node}
            f:facet - adf.bc.samples.model.vo.DepartmentsV
              af:commandImageLink - #{node}
            f:facet - adf.bc.samples.model.vo.EmployeesVie
              af:commandImageLink - #{node}
```

The an `af:switcher` component is used to switch between different command link definitions. To distinguish the tree node level, I used the view definition name of the View Object rendering the node.

You can read the node view definitions from the tree binding in the PageDef file.



```
.0' encoding='UTF-8'?>
UBLIC "-//W3C//DTD XHTML 1.0 Tr
"http://java.sun.com/jsf/core"
itle="TreeIcons.jsf" id="d1">
 id="m1"/>
"f1">
alue="#{bindings.LocationsView1
electionListener="#{bindings.Lc
artialTriggers="::cb1 ::cb2 ::c
```

```
TreeIconsPageDef.xml - Structure
TreeIconsPageDef
  parameters
  executables
  bindings
    LocationsView1
      nodeDefinition
        AttrNames
        Accessors
      nodeDefinition
        AttrNames
        Accessors
      nodeDefinition
```

nodeDefinition - Property Inspector

Advanced

DefName:	adf.bc.samples.model.vo.LocationsView

Other

ClosedIcon:	
DiscrName:	
DiscrValue:	
DTViewInstanceName:	
Icon:	
Name:	LocationsView10
OpenIcon:	
RangeSize:	
Region:	
TargetIterator:	

For this, select the **PageDef** file in the Application navigator and open the Structure Window. Expand the tree node binding and select a **nodeDefinition**. The **DefName** of this node is shown in the Property Inspector.

You use the **DefName** as a facet name in the `af:switcher` component, as shown in the page code snippet below:

```
 <af:tree value="#{bindings.LocationsView1.treeModel}" var="node"
   selectionListener="#{bindings.LocationsView1.treeModel.makeCurrent}"
   rowSelection="single" id="t1">
   <f:facet name="nodeStamp">
     <af:switcher id="s1"
                  facetName="#{node.hierTypeBinding.viewDefName}">
       <f:facet name="adf.bc.samples.model.vo.LocationsView">
         <af:commandImageLink text="#{node}"
               icon="/images/l1.png" id="cil4"
               inlineStyle="height:16px; width:17px;" disabled="true"/>
       </f:facet>
       <f:facet name="adf.bc.samples.model.vo.DepartmentsView">
          <af:commandImageLink text="#{node}" icon="/images/l2.png"
                id="cil6"
                inlineStyle="height:16px; width:17px;" disabled="true"/>
       </f:facet>
       <f:facet name="adf.bc.samples.model.vo.EmployeesView">
         <af:commandImageLink text="#{node}" icon="/images/l3.png"
              id="cil8"
              inlineStyle="height:16px; width:17px;" disabled="true"/>
       </f:facet>
    </af:switcher>
   </f:facet>
</af:tree>
```

The `af:switcher` component uses the **facetName** property to determine which of the contained facet to display upon rendering, using the following EL: **#{node.hierTypeBinding.viewDefName}**

The facets defined in the switcher component are named after the View Object definition names they represent.

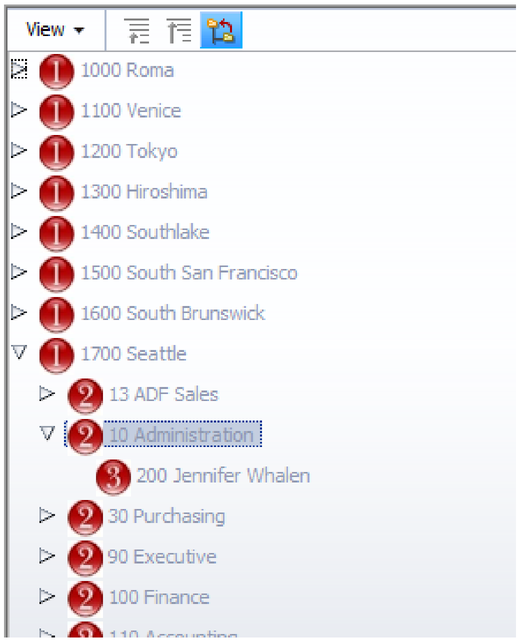
- `adf.bc.samples.model.vo.LocationsView`
- `adf.bc.samples.model.vo.DepartmentsView`
- `adf.bc.samples.model.vo.EmployeesView`

The EL expression used in the switcher **facetName** property returns one of these strings in which case the facet renders, thus showing different icons.

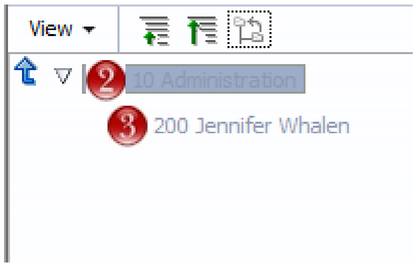
Note that this technique can be used for any use case requiring a difference in the tree node handling or behavior

How to enable the PanelCollection "Show as Top" function

The `af:panelCollection` tag surrounding an `af:tree` component allows users to select a sub-node of the tree to make it the new root node so trees are easier to read for users. See the example below



To make Department 10 the new root node, select it in the tree and click the **Show as Top** icon highlighted in the image above



To return to the previous tree view, you click the blue arrow icon to the left of the root node and select a parent node to make the root node.

When you try this on the tree component you create by dragging a View Object as a tree to a page, it doesn't work as shown above by default. To make this work, you need to add the **pathStamp** facet to your tree. After this, surround the `af:tree` component with an `af:panelCollection` tag and the "Show as Top" functionality becomes available.

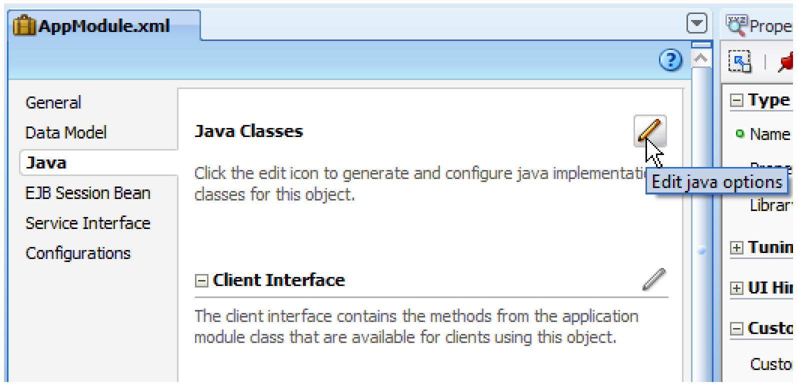
```
<af:tree>
   <f:facet name="nodeStamp">
        <af:outputText value="#{node}" id="ot1"/>
   </f:facet>
   <f:facet name="pathStamp">
       <af:outputText value="#{node}" id="ot2"/>
```

```
    </f:facet>
  </af:tree>
```

How-to access a DB sequence number from Java in ADF

Using ADF Business Components, you can declaratively base an entity object attribute on a database Sequence, which you usually consider for primary key fields that require unique numbering. However, you can do the same from Java, using the ADF BC `SequenceImpl` class, which is a wrapper for a database sequence.

Open the Application Module editor by double clicking the AM file in the Oracle JDeveloper Application Navigator. Select the **Java** menu option and click the **pencil** icon



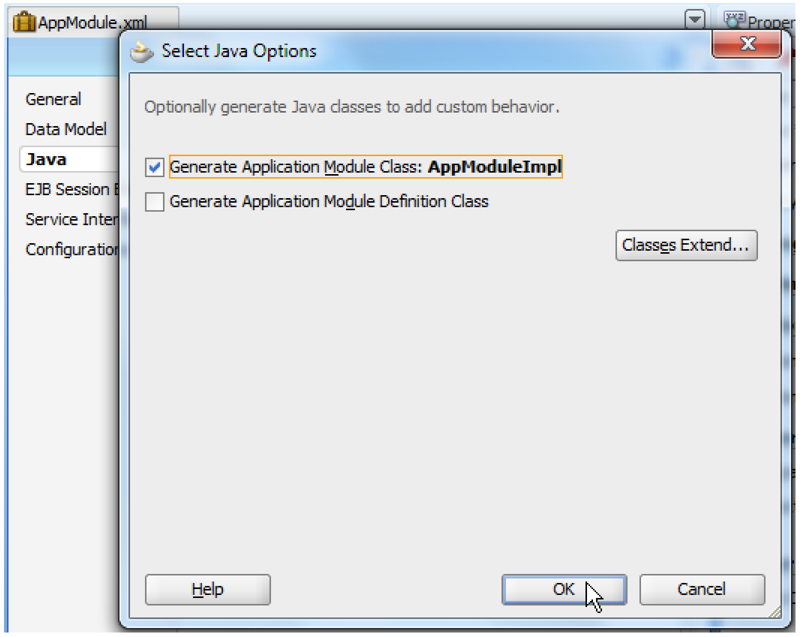
Java Classes

Click the edit icon to generate and configure java implementati... classes for this object.

Client Interface

The client interface contains the methods from the application module class that are available for clients using this object.

In the **Java Options** dialog, check the checkbox to create an implementation file for the Application



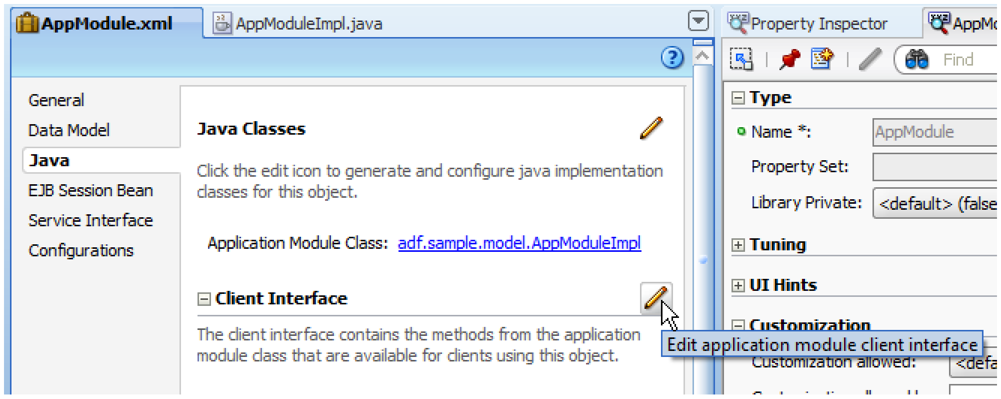
Select Java Options

Optionally generate Java classes to add custom behavior.

[✓] Generate Application Module Class: **AppModuleImpl**

[ ] Generate Application Module Definition Class

Still in the **Java** menu, click the **pencil icon** next to the **Client Interface** label to expose the sequence method on the client interface so it becomes visible on the Data Control.



Java Classes

Click the edit icon to generate and configure java implementation classes for this object.

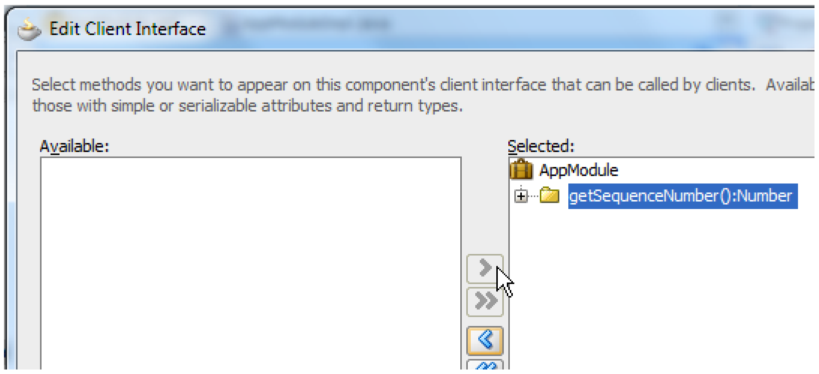
Application Module Class: adf.sample.model.AppModuleImpl

Client Interface

The client interface contains the methods from the application module class that are available for clients using this object.

Name *: AppModule
Property Set:
Library Private: <default> (false
Customization allowed: <defa

Select the methods you want to expose on the Data Control.



Select methods you want to appear on this component's client interface that can be called by clients. Availab those with simple or serializable attributes and return types.

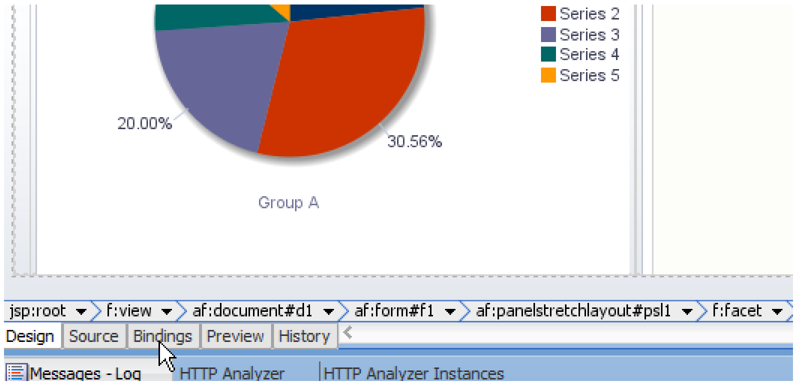
getSequenceNumber():Number

The following code needs to be added to the Application Module Impl class to obtain a sequence programmatically in ADF BC

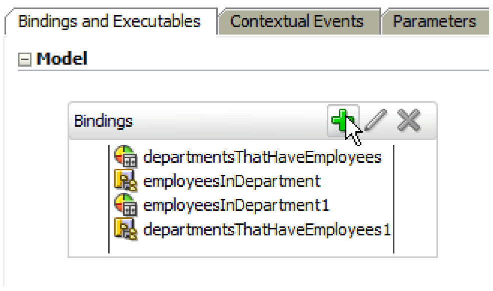
```
public oracle.jbo.domain.Number getSequenceNumber()    {
  try{
   String seq = "your_db_sequence_name";
   oracle.jbo.server.SequenceImpl theSeq =
      new oracle.jbo.server.SequenceImpl(seq, getDBTransaction());
   oracle.jbo.domain.Number seqNextNumber = theSeq.getSequenceNumber();
   return seqNextNumber;
  }
  catch (Exception e){
    //handle exceptions
  }
 return null;
}
```

To access the sequence number from Java in the view layer, for example a managed bean, you expose this method on the Application Module client interface. You then create a method binding in the PageDef file used by the view.

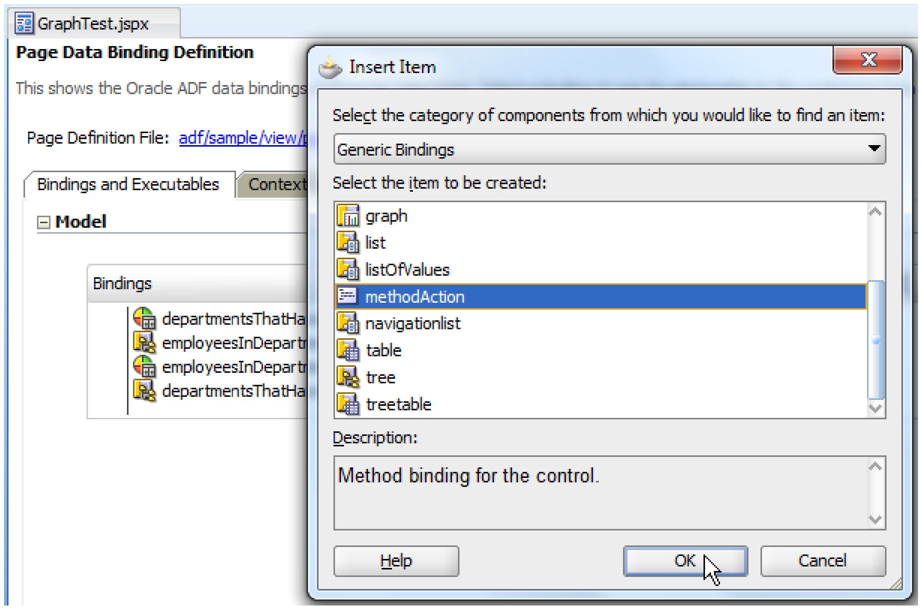
For this, open the ADF Faces page or view to configure the method for and select the **Binding** tab to switch to the PageDef configuration dialog.



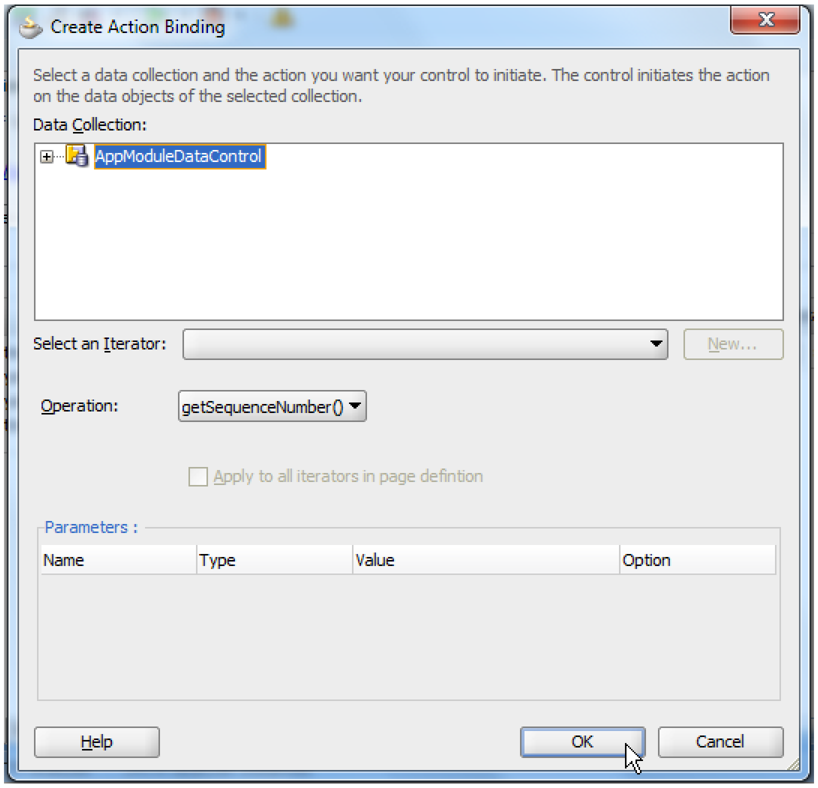
In the bindings editor, press the green plus icon in the **Bindings and Executable** tab.



Select **methodAction** to create a binding to the get sequence method exposed on the Application Module.


GraphTest.jspx
Page Data Binding Definition
This shows the Oracle ADF data bindings
Page Definition File: adf/sample/view/
Bindings and Executables
Context
Model
Bindings
departmentsThatHa
employeesInDepart
employeesInDepart
departmentsThatHa
Insert Item
Select the category of components from which you would like to find an item:
Generic Bindings
Select the item to be created:
graph
list
listOfValues
methodAction
navigationlist
table
tree
treetable
Description:
Method binding for the control.
Help
OK
Cancel

Select the Application Module entry in the **Create Action Binding** dialog and select the method you want to expose in the PageDef file.



Select a data collection and the action you want your control to initiate. The control initiates the action on the data objects of the selected collection.

Data Collection: AppModuleDataControl

Select an Iterator:

Operation: getSequenceNumber()

Apply to all iterators in page defintion

Parameters :

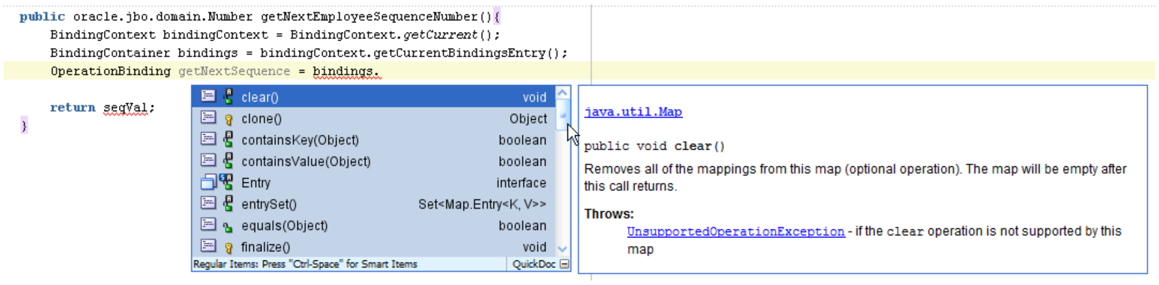
Name	Type	Value	Option

Once the binding is created, add the following method to a managed bean to access the sequence next value exposed on the Application Module.

```
public oracle.jbo.domain.Number getNextEmployeeSequenceNumber(){
  BindingContext bindingContext = BindingContext.getCurrent();
  BindingContainer bindings = bindingContext.getCurrentBindingsEntry();
  OperationBinding getNextSequence =
                   bindings.getOperationBinding("getSequenceNumber");
  oracle.jbo.domain.Number seqVal =
                 (oracle.jbo.domain.Number) getNextSequence.execute();
  return seqVal;
}
```

Smart code insight in Oracle JDeveloper

Oracle JDeveloper provides code insight support when typing the name of an object instance variable followed by a dot – '.' – character and a pausing for half a second.



```
public oracle.jbo.domain.Number getNextEmployeeSequenceNumber(){
    BindingContext bindingContext = BindingContext.getCurrent();
    BindingContainer bindings = bindingContext.getCurrentBindingsEntry();
    OperationBinding getNextSequence = bindings.

    return seqVal;
}
```

clear()	void
clone()	Object
containsKey(Object)	boolean
containsValue(Object)	boolean
Entry	interface
entrySet()	Set<Map.Entry<K, V>>
equals(Object)	boolean
finalize()	void

Regular Items: Press "Ctrl-Space" for Smart Items QuickDoc

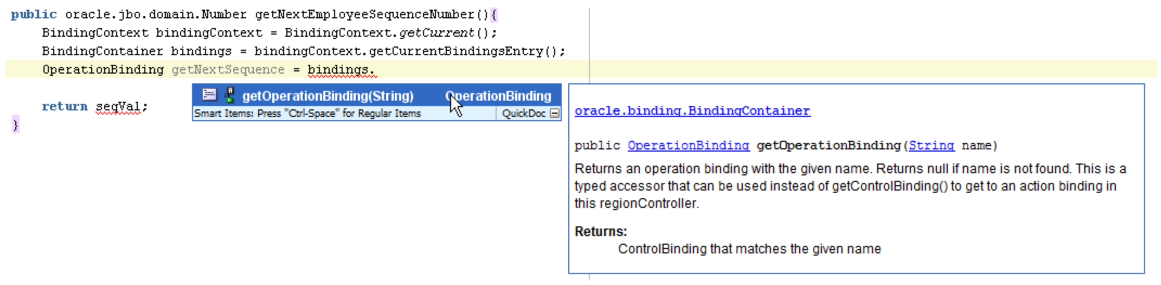
java.util.Map

public void clear()

Removes all of the mappings from this map (optional operation). The map will be empty after this call returns.

Throws:
UnsupportedOperationException - if the clear operation is not supported by this map

To find matching methods easier, hold the ctrl key and press the space key twice. This then only shows methods matching the defined return type (OperationBinding in this case).



```
public oracle.jbo.domain.Number getNextEmployeeSequenceNumber(){
    BindingContext bindingContext = BindingContext.getCurrent();
    BindingContainer bindings = bindingContext.getCurrentBindingsEntry();
    OperationBinding getNextSequence = bindings.

    return seqVal;
}
```

getOperationBinding(String) OperationBinding
Smart Items: Press "Ctrl-Space" for Regular Items
QuickDoc

oracle.binding.BindingContainer

public OperationBinding getOperationBinding(String name)

Returns an operation binding with the given name. Returns null if name is not found. This is a typed accessor that can be used instead of getControlBinding() to get to an action binding in this regionController.

Returns:
ControlBinding that matches the given name

Alternatively you can press **ctrl+alt+space** to show a filtered list of methods. If there is only one method that meets the defined return type, like in getOperationBinding above, then no dialog is shown and the method is added directly to the code.

ADF Code Corner

OTN Harvest Spotlight

- Jan Vervecken

ORACLE CODE CORNER ADF OTN HARVEST

Jan Vervecken is a consultant at Contribute in Belgium, where he helps customers implementing solutions based on Oracle Middleware and Oracle development technologies.

Developers who follow the Oracle JDeveloper and ADF forum on OTN know Jan from is very analytic and well thought through questions and straight to the point answers. Jan also is an active member of the Oracle ADF Enterprise Methodology (EMG).

ACC: What is your current role?

JV: Working for Contribute, a Belgian company focusing on Oracle technology, I assist our customers with their projects in different ways. Depending on the customer, their project, technology, knowledge, etc. I try to help them reach their goals, often with a pragmatic approach towards qualitative and maintainable results.

For quite a few years this has also involved working with Oracle ADF and guiding people in using such flexible framework and making it fit into their context.

ACC: What is your IT background?

JV: At the university of Antwerp I studied computer science. After that, in 1998 I started working at Cronos, a Belgian system integrator, which has grown to over 2400 employees in different companies, one of them Contribute.

For me there has often been some kind of Oracle technology involved. From building web applications using PL/SQL, over generating them with Oracle Designer, toward Java based approaches. It was the sometimes cumbersome and time consuming techniques for Java development several years ago that got met intrigued by the productivity potential of Oracle ADF.

ACC: How do you currently use Oracle JDeveloper and ADF?

JV: Currently it is mainly focusing on specific problems reported by other developers working with JDeveloper on ADF projects. Sometimes these have a simple answer but more often they tend to require further investigation before we can conclude to a future proof solution.

It is important to try to indentify and avoid the "trickery" that seems to work, and find solutions that work with the framework, not against it. For this it is important to ask the question "why" often enough and make sure that the answers to that question are reflected in the design, structure, naming, etc. of what you are building. At different levels trying to find what varies,

and encapsulate it.
There are also the aspects of what some would call application lifecycle management that sometimes requires insights that tend to fall into a kind of no man's land between developers and system administrators.

Stuff related to deploying, testing, continuous integration, etc. which does not always have easy answers. As in many aspects of application development it is important to keep an eye on what your dependencies are. Also keeping an eye on introducing sufficient feedback points in a preferably iterative approach.

ACC: So far, what has been your biggest challenge in building Java EE application with Oracle ADF?

JV: Educating people. The background of people using Oracle ADF can be very different which can sometimes make it a challenge to explain some aspects of Oracle ADF.

Some with a Java EE background tend to do more themselves in code, where sometimes the framework has a declarative solution.

Some with an Oracle Forms background tend to get confused by the many "moving parts", the layering in the framework and the underlying architecture.

ACC: Which feature of ADF was the greatest benefit to your project?

JV: The productivity potential. Many features of Oracle ADF itself and the combination with Oracle JDeveloper have been created with productivity in mind.

It does take some effort to learn Oracle ADF and to make it fit your needs, but after that there could be a real productivity gain. Where for other frameworks or tools, creating something similar today, or again after you have been using it for a while, will take a similar amount of time and effort, so without increasing productivity.

ACC: Which feature do you use the least?

JV: For example working with a Business Components Diagram is something I have wondered about, who does that in a real project and gets some real benefit from it. At the same time I wonder about the effort Oracle has put into such a feature and where this effort could have been better spent on other framework aspects.

ACC: Away from the on line help, what have been your most valuable sources of ADF knowledge?

JV: In the first place the OTN JDeveloper and ADF forum. It allows benefiting and learning from real experience with Oracle ADF, real problems and real (hints to) solutions.

And if you can't find what you are looking for, it allows you to "describe" your own problem. Sure, you would say, what else is a forum for? But, if you think about it, trying to properly "describe" your own problem with sufficient relevant details is really reflecting on what you have been doing and trying to isolate what your problem is really about.

Keeping in mind your intention is to post on the forum, so to explain to someone external to the context you're working in. Such process more often than not results in finding a solution for your problem even before you really post on the forum.

It can lead to finding the proper question to ask. Or if the problem ends up isolated in an example application, this will allow easily reviewing or verifying feedback you get on the forum.

ACC:	Are you in any way actively involved in the ADF Community?
JV:	In the ADF community I only modestly contribute to some online discussions. Typically those contributions tend to be pragmatic and with concrete references to related resources. I get so much more out of the ADF community than I can give back. Maybe I should take this opportunity to just thank everyone else who allowed me to learn from their contributions to the ADF community.
ACC:	What would be your recommendation for people starting with Oracle JDeveloper and Oracle ADF in terms of how to start and pitfalls to avoid?
JV:	There are online resources to be found that recommend on where to start with Oracle JDeveloper and Oracle ADF. But in addition it would probably be a good idea to get someone with experience on your team. Depending on the context, maybe only to get you started or to help guide you over a longer term. Be it through coaching, reviewing, teaching or whatever fits your needs. In many different ways this can help you to evolve in a direction that will allow you reach your goals for your Oracle ADF endeavour.
ACC:	ADF Genie grants you a wish, what would you ask for?
JV:	Recursion, two more wishes, what else? Well, maybe I would ask for more time to learn more about Oracle ADF.
ACC:	Thank you

RELATED DOCOMENTATION

☒	
☒	
☒	