ADF Code Corner

Oracle JDeveloper OTN Harvest

12 / 2011 & 01 / 2012

twitter.com/adfcodecorner

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.

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

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

Disclaimer: ADF Code Corner OTN Harvest is a blogging effort according to the Oracle blogging policies. It is not an official Oracle publication. All samples and code snippets are provided "as is" with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

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

December 2011 & January 2012 Issue – Table of Contents

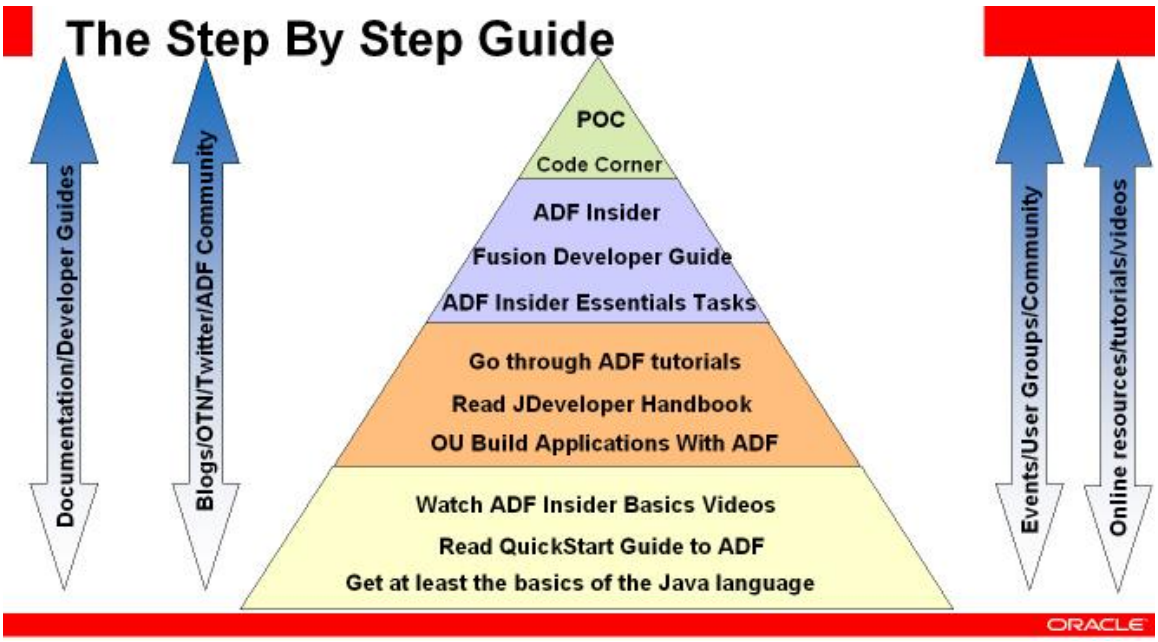
How to start with Oracle ADF

A reoccurring question on the OTN forum is "how do I get started with Oracle ADF". So I decided to frequently post the recommended starting points in different OTN Harvest summary editions.

Shay Shmeltzer from the Oracle JDeveloper and ADF product management team maintains a complete list of starter material you can download for free from OTN and the Internet or buy in book stores

http://blogs.oracle.com/shay/entry/how_do_i_start_learning_oracle_adf_and_jdeveloper

In addition, Grant Ronald worked on a visual learning pyramid that lists resources by complexity and difficulty



The recommended general entry point for you adventuring into Oracle ADF land is the Oracle JDeveloper home page on OTN: http://otn.oracle.com/products/jdev

Task Flows or Portlets – Which one to choose?

Maiko Rocha from the Oracle Architecture Team (aka A-Team) wrote a summary about this topic to lift the confusion.

"If there's a question that we get more often than "When [INSERT_NEXT_PRODUCT_RELEASE_ HERE] is going to be available?", it is "Should I use Task Flows or Portlets?".

I can't remember of a single WebCenter engagement that we on the A-Team have worked on that this question hasn't been asked. Let's see what's currently written up on the internet on the subject and build on top of that."

In his blog article Maiko references the contributions of other great WebCenter minds so you get a complete answer

http://blogs.oracle.com/ATEAM_WEBCENTER/entry/adf_task_flows_versus_portlets

File download and upload handling In Oracle ADF

Uploading and downloading files is an evergreen question on the OTN forum. Timo Hahn, Oracle ACE and OTN forum contributor wrote a three part blog series about this very topic to summarize various threads and answers given. You find Timo's article series here:

Part 1: http://tompeez.wordpress.com/2011/11/26/jdev11-1-2-1-0-handling-imagesfiles-in-adf-part-1/

Part 2: http://tompeez.wordpress.com/2011/11/26/jdev11-1-2-1-0-handling-imagesfiles-in-adf-part-2/

Part 3: http://tompeez.wordpress.com/2011/12/16/jdev11-1-2-1-0-handling-imagesfiles-in-adf-part-3/

How to define the default sort order of an ADF table

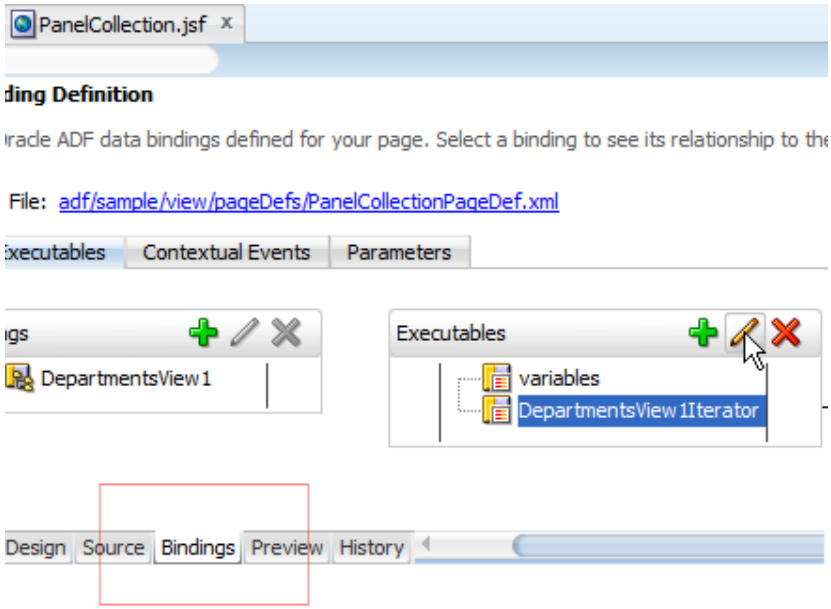
An ADF table is based on a collection and displays data in the order the data shows in the ADF iterator, which actually is how the data is queried by the business service. If, as an ADF developer, you need data to be sorted different from the default, there is a declarative option at the ADF iterator you can use.

The table below shows the data as they are queried from the Departments table of the HR schema. The data is sorted by the **DepartmentId** attribute, which obviously is how the view object query retrieved it.

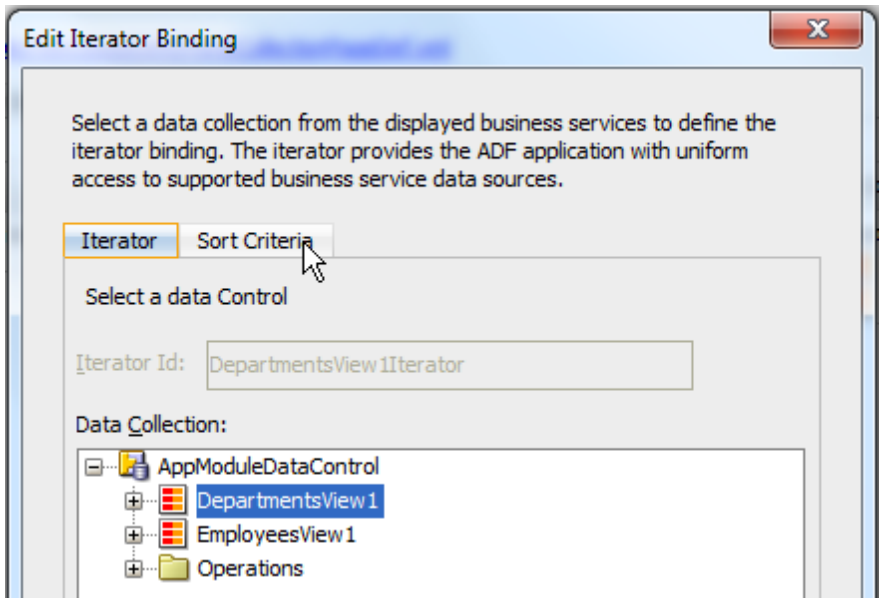
View ▾ Detach

DepartmentId	DepartmentName	ManagerId	LocationId
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Legal	203	2400
50	Human Resources	203	2400
60	IT	103	1400
70	Public Relations	204	2000
80	Sales ii	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700
140	Control And Credit		1700
160	Benefits		1700
170	Manufacturing		1700
180	Construction		1700
190	Contracting		1700
200	Operations		1700
210	IT Support		1700
220	NOC		1700
230	IT Helpdesk		1700
240	Government Sales		1700

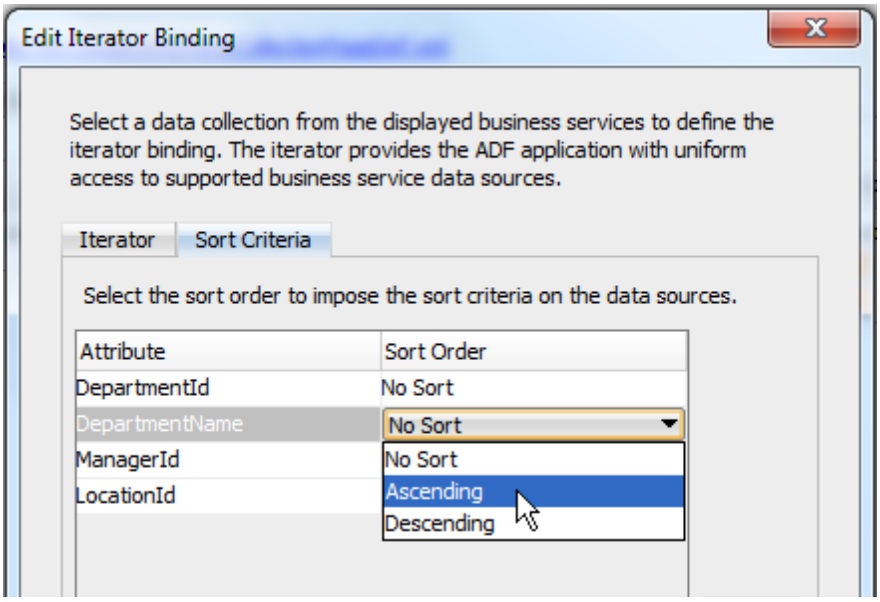
To change the default sorting to sort the table data by the **DepartmentName** attribute, you select the **Binding** tab at the bottom of the ADF bound page or page fragment the ADF Faces table is located in.



Select the Iterator binding, **DepartmentsView1Iterator** in this sample, and press the **pencil**icon to edit the binding configuration.



Switch to the **Sort Criteria** tab to define the new sort order to be applied when the data is displayed in the view.



Choosing **Ascending** as a sort option on the **DepartmentName** attribute, now changes the table initial data ordering so that the **DepartmentName** attribute shows sorted and the **DepartmentId** no longer.

View ▾ Detach

DepartmentId	DepartmentName	ManagerId	LocationId
110	Accounting	205	1700
10	Administration	200	1700
160	Benefits		1700
180	Construction		1700
190	Contracting		1700
140	Control And Credit		1700
130	Corporate Tax		1700
90	Executive	100	1700
100	Finance	108	1700
240	Government Sales		1700
50	Human Resources	203	2400
60	IT	103	1400
230	IT Helpdesk		1700
210	IT Support		1700
40	Legal	203	2400
170	Manufacturing		1700
20	Marketing	201	1800
220	NOC		1700
200	Operations		1700
270	Payroll		1700
70	Public Relations	204	2000
30	Purchasing	114	1700
260	Recruiting		1700

Using JSTL in ADF Faces

The JavaServer Pages Standard Tag Library (JSTL) provides expressions for common web application functionalities. Though JavaServer Faces Expression Language (EL) is different from JSTL expressions, the two can be used in combination for functional for which it makes sense. For example, the table below lists JSTL expressions that operate on String values.

Function	Description
fn:contains()	Tests if an input string contains the specified substring.
fn:containsIgnoreCase()	Tests if an input string contains the specified substring in a case insensitive way.
fn:endsWith()	Tests if an input string ends with the specified suffix.
fn:escapeXml()	Escapes characters that could be interpreted as XML markup.
fn:indexOf()	Returns the index withing a string of the first occurrence of a specified substring.
fn:length()	Returns the number of items in a collection, or the number of characters in a string.
fn:replace()	Returns a string resulting from replacing in an input string all occurrences with a given string.
fn:startsWith()	Tests if an input string starts with the specified prefix.
fn:substring()	Returns a subset of a string.
fn:substringAfter()	Returns a subset of a string following a specific substring.
fn:substringBefore()	Returns a subset of a string before a specific substring.
fn:toLowerCase()	Converts all of the characters of a string to lower case.
fn:toUpperCase()	Converts all of the characters of a string to upper case.
fn:trim()	Removes white spaces from both ends of a string.

Src: http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

To use JSTL in Oracle ADF Faces component EL references, you need to add the JSTL namespace of the tags you want to reference. For the JSTL functions shown in the table above, you need to manually add the name space highlighted in bold to the JSPX document or page fragment.

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:h="http://java.sun.com/jsf/html"
        xmlns:af="http://xmlns.oracle.com/adf/faces/rich"
        xmlns:fn="http://java.sun.com/jsp/jstl/functions">
```

The value expression below, referenced in an af:outputText component, displays the substring (first, second and third character) for a value read from the ADF binding layer.

```
<af:outputText value="#{fn:substring(bindings.employeeName.inputValue,1,3)}" id="ot7"/>
```

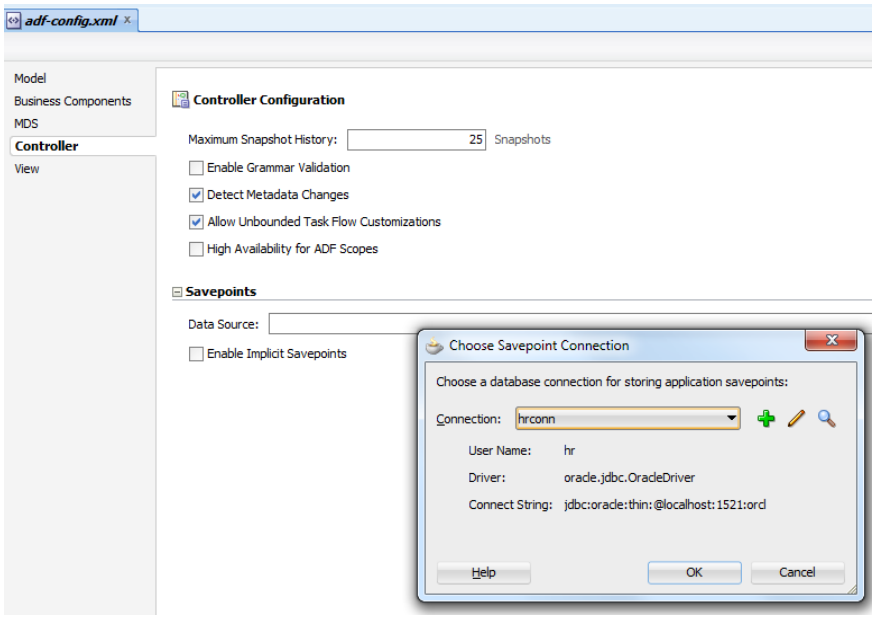
ADF BC bc4jcleanup.sql in Oracle JDeveloper 11g

The ADF Business Component **bc4jcleanup.sql** package contains utility functions to clean temporary ADF BC persistence storage. Searching for the **bc4jcleanup.sql** file in JDeveloper 11g however returns empty because the SQL file name has been changed to **adfbc_purge_statesnapshots.sql**. The file is located in the **oracle_common\common\sql** directory of the JDeveloper 11g installation home.

Oracle JDeveloper 11.1.2 ADFc Savepoint Script

The save point (save for later) feature of the ADF Controller is enabled from **Application Resources | Descriptors | ADF META-INF** in the Application Navigator.

To enable controller save points, double click onto the **adf-config.xml** file and assign a data source name to the **Savepoints | Data Source** property using the magnifying glass.



In JDeveloper 11g R1, make sure the account accessing the schema into which save points should be stored has the create table permission granted. Background for this requirement is that the save point table is automatically created the first time a save point is written by the controller. After this the create table permission can be removed from the account. As you guessed it, this approach is not a DBA's darling, which is why an external script is provided in 11g R2.

In JDeveloper 11g R2, an external script - **adfc_create_save_point_table.sql** - is provided for you t create the save point table in advance. The script is located in the **oracle_common\common\sql** directory of the JDeveloper 11g installation home.

Row currency separation in task flow ADF region Instances

A question on the Oracle JDeveloper and ADF OTN forum in end of 2011 was how to add multiple instances of a bounded task flow as regions to a page and have them all showing a different row data without the Data Control scope being set to "isolated". Duncan Mills blogged a solution to this problem using dynamically created View Object instances. You can access the blog entry from here.

http://blogs.oracle.com/groundside/entry/maintaining_row_currency_separation_in

Please find Duncan's full blog post below (thanks to Duncan for giving me the permission).

By Duncan Mills -

"So here's a quick thought experiment, how would one create a re-usable task flow which should tick the following boxes:

- Multiple copies of the flow are displayed on a single page.
- Each instance of the task flow should maintain it's own row currency and not be affected by navigation in any other instance.
- All instances of the task flow should be part of the same transactional context.

Let's analyse that.

- Well, Requirement (1) is easy, that's one of the core capabilities of task flows. Each has it's own pageFlowScope which is kept private so multiple instances on the page should be fine.
- Requirement (2) can be achieved simply by setting the data control scope to **<isolated/>** in the task flow definition.
- Requirement (3) - oh dear.

So there's the problem, it seems that requirements (2) and (3) are mutually exclusive. If you want everything to use the same transaction you also have to have them sharing the same VO instance and therefore being coordinated from the record currency point of view.

To solve this let's just think a little outside the box and consider what row currency is and therefore how we could approach the problem. Each View Object Instance has a primary RowSetIterator, which identifies the "current row" within the VOs collection.

As all the instances of the task flow are using the same bindings, they are therefore using the same VO instance and therefore all share the same RowSetIterator (RSI). Now a VO instance can have secondary RSIs but these are really only of use in the programmatic sense, and cannot be wired in through the binding layer.

So logically, one approach to this is to define a second VO instance based on the same VO definition and bind one task flow to that and the other to the original.

Well that's fine, but we're painted into a corner here, in that we need to decide up-front how many concurrent views we need and then define explicit VO instances in the AM and separate almost-identical-apart-from-the-bindings task flows. So this may well work when you just have a couple of views but then it get's unworkable, plus it just feels wrong to duplicate all that task flow content just to change the VO instance name.

However, in principle, the idea is along the right lines. We just need to simplify it so that we don't have to do any duplication or pre-definition of VO instances. Is it possible? Yes of course, and surprisingly easy.

Getting Started

The first step here is to create your AM with an initial instance of the VO you need exposed (you can delete it later but having it there will make the design time easier).

Once the VO instance is available in the data control palette you can go ahead and create your re-usable task flows and test them you want to get the basic functionality right after all. Of course at this stage if you use multiple instances of the task flow they will all be coordinated.

Parameterize the Task Flow

At this stage define a task flow parameter which can be passed in to define the unique name for the VO instance that this instance of the task flow will use. You might write some fancy generator method to create unique names in sequence (DeptView2, DeptView3, DeptView4 and so on) or just hardcode a value when you map the task flow into a region, it's up to you.

Just bear in mind that any task flow instances that share the same instance name will also share the same RSI and therefore be coordinated - so you can mix and match both coordinated and uncoordinated instances. For the sake of example let's call that parameter **pVOInstanceName**.

Create a Method to Create New Instances of the VO

Next we need to define a method that will create a VO instance on demand. This is very simple code, just generate up a Java Impl class for your AM and add a method something like this:

```
public void createUniqueVOInstance(String voDefName,
                                    String instanceName){
  ViewObject existingVO = findViewObject(instanceName);
  if (existingVO == null) {
    _logger.info("Creating VO instance for " + instanceName);
    ViewObject newVO = this.createViewObject(instanceName, voDefName);
  }
  else {
    _logger.info("Reusing VO instance for " + instanceName);
  }
}
```

Note how we check to see if that name is already in use first.

Expose this method as part of the client interface so we can bind that into the task flow. Invoke the VO Instance Creation Method from the task flow

Once you've refreshed the data control panel you should be able to drag this new AM method (*createUniqueVOInstance* in my case) into the task flow.

Set that as the default activity on the flow so that it executes first. You'll pass the full name of the ViewObject definition (e.g. **oracle.demo.model.DepartmentsView**) into the voName parameter and the instance name you want will come from the parameter you defined for the taskflow (e.g. #{pageFlowScope.pVOInstanceName}).

So now when you enter the flow you'll create a new VO instance based on the supplied definition with this name.

Naturally you then wire this method activity to the rest of the task flow as normal so that once the instance has been created you can continue to display the page fragments etc.

Rewire the Iterators

The final piece of the puzzle is how to tell all of the bindings that you've created within the task flow to use this newly created VO instance rather than the one that they were created with. This final step is very simple and elegant.

For the views and other activities in the task flow just edit the pageDef XML files and locate all of the iterator bindings that relate to your template VO instance that you used when creating the UI through drag and drop. Now simply change the hard-coded reference to the VO Instance name in the Iterator binding **Binds** attribute to the expression pointing to your new instance name. e.g.

```
<executables>
  <iterator Binds="#{pageFlowScope.pVOInstanceName}" RangeSize="25"
        DataControl="AppModuleDataControl" id="DepartmentsView1Iterator"
        ChangeEventPolicy="ppr"/>
</executables>
```

And there you go, nothing else has to change. Just be sure to make this change throughout all of the pageDefs used by the task flow otherwise you're in for some funny results.

So there you have it, transactionally linked but independently scrolling instances of the same task flow.

And there's more

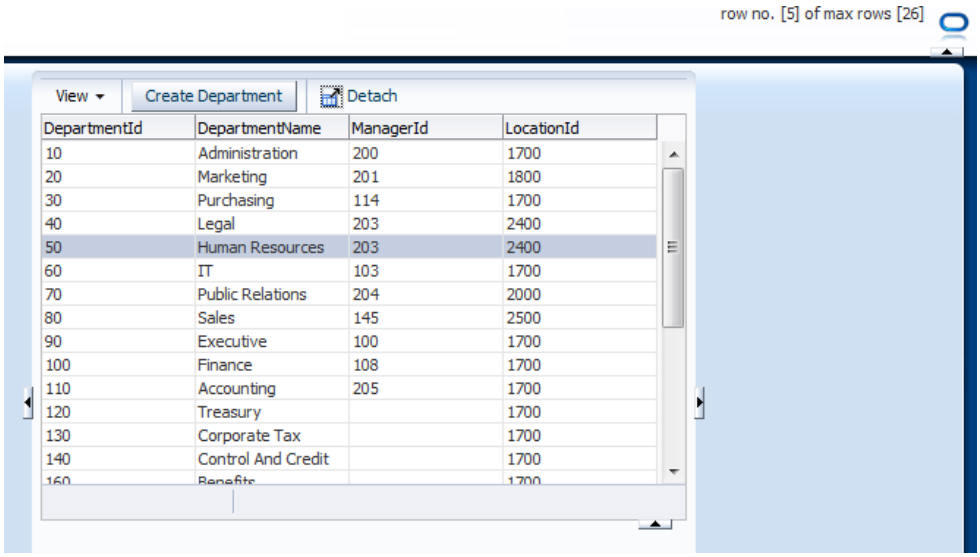
Once you start to think about it, things get even more interesting here. Because each instance of the task flow has it's own VO instance they can also have their own set of view criteria applied and yet all still be visible on the screen at once.

Display selected row number and total rows

A question on OTN was how to display the current row number and the number of totally available rows in a formatted output string. In the example used below, the format is

row no. [<row index>] of max rows [<total numbers of rows>]

As shown in the image below, I used an **af:outputText** component to display the information on the page. The value of the **af:outputText** component is calculated from expression language (EL) accessing the ADF binding layer



The Expression Language in the **af:outputText** component is shown with line breaks for better readability below. In the Expression Builder, you would make this as a single string

row no. [

```
#{(bindings.allDepartmentsIterator.rangeStart < 0 ?
1 :  bindings.allDepartmentsIterator.rangeStart+1) +
( bindings.allDepartmentsIterator.currentRowIndexInRange ==  -1 ?
0 : bindings.allDepartmentsIterator.currentRowIndexInRange)}]
```

of max rows

```
[#{bindings.allDepartmentsIterator.estimatedRowCount}]
```

Note how the expression uses the iterator binding in the ADF binding layer to determine the maximum number of table rows. It then uses the same iterator to determine the current row. Because iterators are zero based, the EL uses additional logic to compensate this.

The **af:outputText** component needs to have its **PartialTriggers** property pointing the **af:table** component ID to ensure the displayed information is refreshed when the table row currency changes.

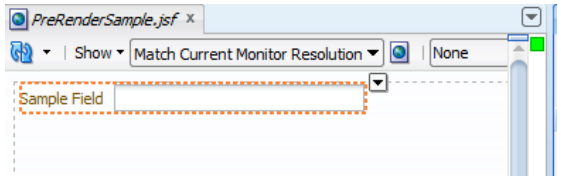
```
<af:outputText   value="row no. [#{(bindings.allDepartmentsIterator.rangeStart &lt; 0 ? 1  :
                 bindings.allDepartmentsIterator.rangeStart+1) +(
                 bindings.allDepartmentsIterator.currentRowIndexInRange ==  -1 ? 0 :
                 bindings.allDepartmentsIterator.currentRowIndexInRange)}] of max rows
                 [#{bindings.allDepartmentsIterator.estimatedRowCount}]"
                 id="ot1" partialTriggers="pc1:t1"/>
```

Using JSF 2.0 component events in JDeveloper 11gR2

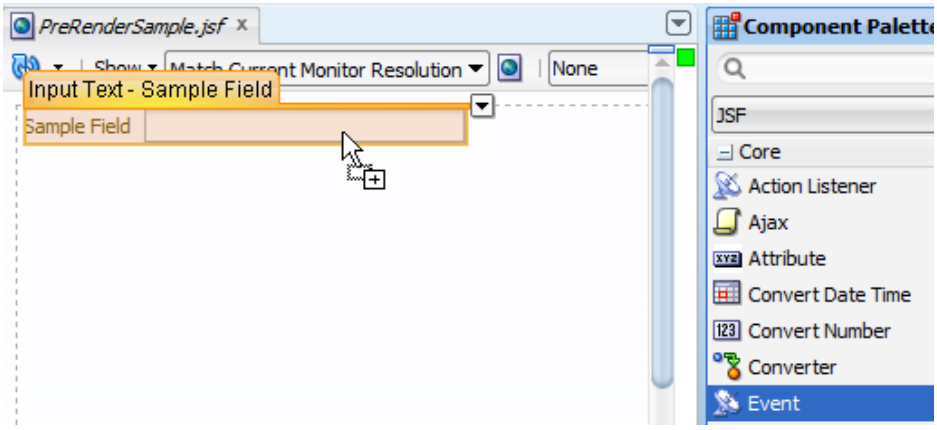
JavaSever Faces 1.x provided Faces and Phase events for developers to respond to user interaction within a form or to the lifecycle. JavaServer Faces 2.0 added a third category of events, system events, to this list, allowing developer to add listeners to a component instance to interact upon validation, or just to respond to when the component is added to a view or rendered.

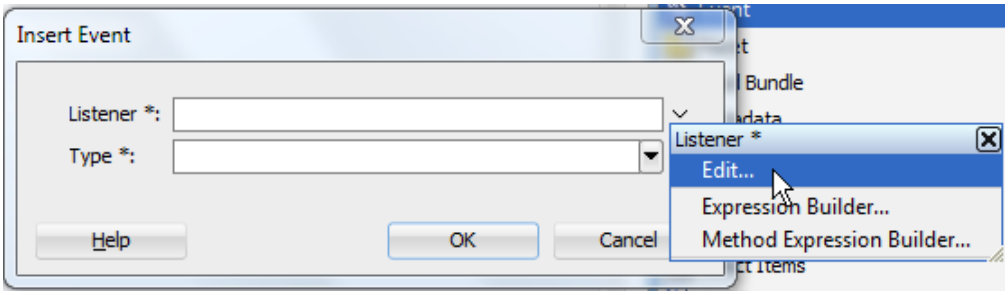
A typical use case for when a component event listener becomes handy is with default values. In this very simple sample I provide, I assume that the objective is to provide a default value of "Hello World" to an input text component whenever the value of this component is not set. For implementing this use case in JSF 1.x, I would need to trick JavaServer Faces and create a component binding or setup a PhaseListener on the application level. In JSF 2.0, implementing this use case has become easier – also for applications built with ADF Faces.

Starting from an ADF Faces `RichInputText` component you add to a page, you open the JDeveloper Component palette.

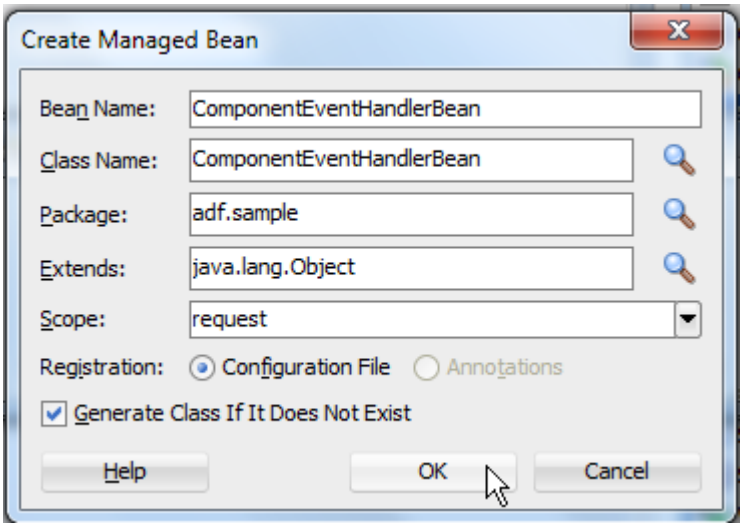


In the Component palette, select the **JSF** component set, not the ADF Faces components, and expand the **Core** component category. In the **Core** component category, select the **Event** listener tag and drag it on top of the input text component on the page.

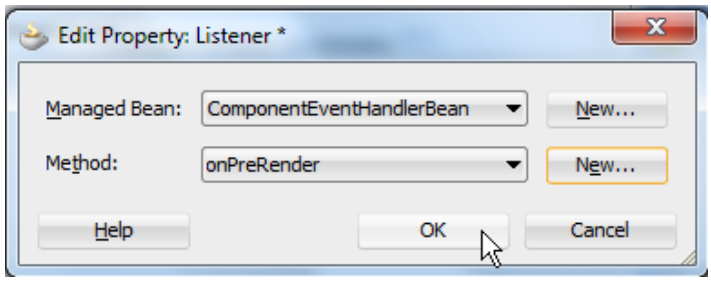




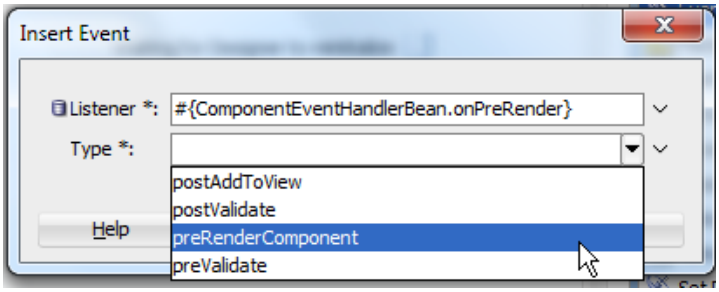
In the opened dialog, create or select a managed bean to define the event listener code in. In the sample I built, the name of the managed bean is **ComponentEventHandlerBean** and the bean is defined in request scope.



I then defined a name for the method to call by this component event. The listener method I chose is **onPreRender.**



A last step is to select the event the component listener should listen for. For this I am selecting the **preRenderComponent** type so my listener method is invoked before the component is rendered in the view.

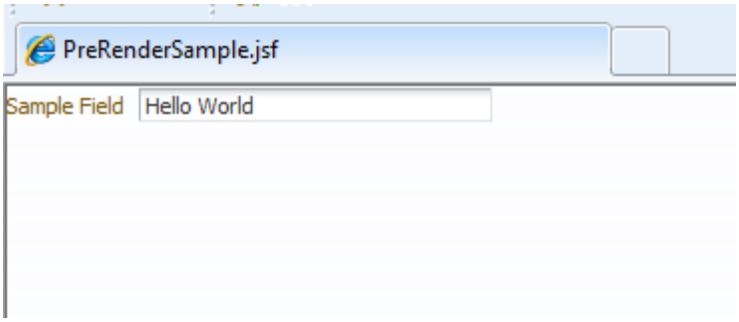


Oracle JDeveloper generated the listener method skeleton, passing the `ComponentSystemEvent` as an argument to it.
The `ComponentSystemEvent` allows accessing the UI component for which the listener is defined. So in the code shown below, I check the value of this component and if it is null, put "Hello World" in.


```
public void onPreRender(ComponentSystemEvent componentSystemEvent) {
 RichInputText richText =
              (RichInputText) componentSystemEvent.getComponent();
 if(richText.getValue()==null){
```

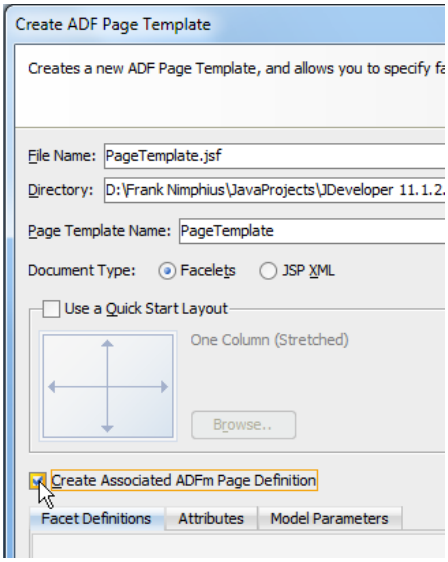
```
    //set new default value
    richText.setValue("Hello World");
  }
}
```

The runtime result is as expected and "Hello World" is shown when the page renders. You may want to try the same with pre-validation or post-validation event. This new feature in JSF 2.0 really makes it easier to get what you want (e.g. setting the disclose state of a tree, which is a common use case in Oracle ADF)

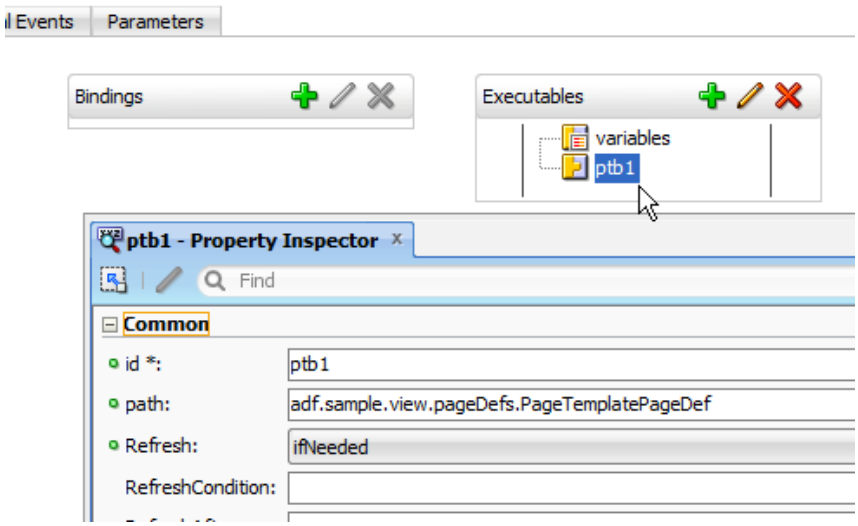


How-to invoke ADF bindings in page templates

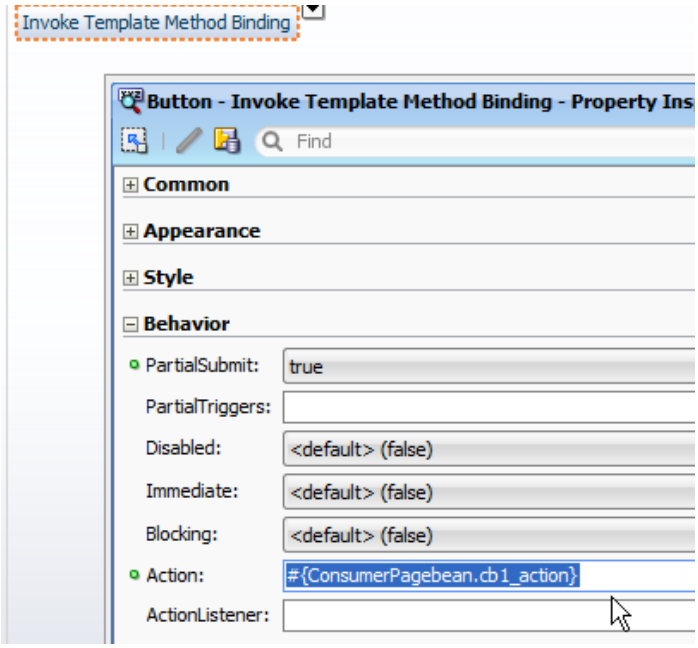
ADF Faces page templates may have their own PageDef file defined to access the ADF binding layer or have ADF bound components added to the template.



ADF Faces pages that use the template will reference the template's PageDef file in the **Executable** section of their own PageDef file. In this sample, the template binding reference in the PageDef file of the template consumer page is called **ptb1**.



The code below invokes a method binding in the template's PageDef file from a command button on the page.



The "trick" for accessing the template's PageDef file is to know that it is another instance of `BindingContainer`.

```
public String cb1_action() {
  BindingContext bctx = BindingContext.getCurrent();
  DCBindingContainer bindings =
        (DCBindingContainer) bctx.getCurrentBindingsEntry();
```

```
  //access the page template Pagedef file reference in the
  //Executable section of the consumer page's Pagedef file
  DCBindingContainer templateBinding =
        (DCBindingContainer) bindings.get("ptb1");
  //get the MethodBinding
  OperationBinding printMethod =
        (OperationBinding) templateBinding .get("printThis");
  printMethod.getParamsMap().put("message", "Hello World");
  printMethod.execute();
  return null;
}
```

How-to determine the ADF tree node type using EL

Creating an ADF tree in ADF produces an entry similar to this in the PageDef file of the view.

```
<tree IterBinding="AllCountriesIterator" id="AllCountries">
  <nodeDefinition
     DefName="oracle.summit.model.views.CountryVO" Name="AllCountries0">
   <AttrNames>
    <Item Value="Country"/>
   </AttrNames>
   <Accessors>
    <Item Value="CustomerVO"/>
   </Accessors>
  </nodeDefinition>
  <nodeDefinition
     DefName="oracle.summit.model.views.CustomerVO" Name="AllCountries1"
     TargetIterator="${bindings.AllCustomersIterator}">
   <AttrNames>
    <Item Value="Id"/>
    <Item Value="Name"/>
   </AttrNames>
  </nodeDefinition>
 </tree>
```

Notice the DefName attribute on each node containing a reference to the actual View Object instance used to render a specific node.

With this information you can now use EL to render the tree nodes differently. For example, the page source below renders the node as a command link if the node presents a customer. For Countries, the node is simply rendered as an output text.

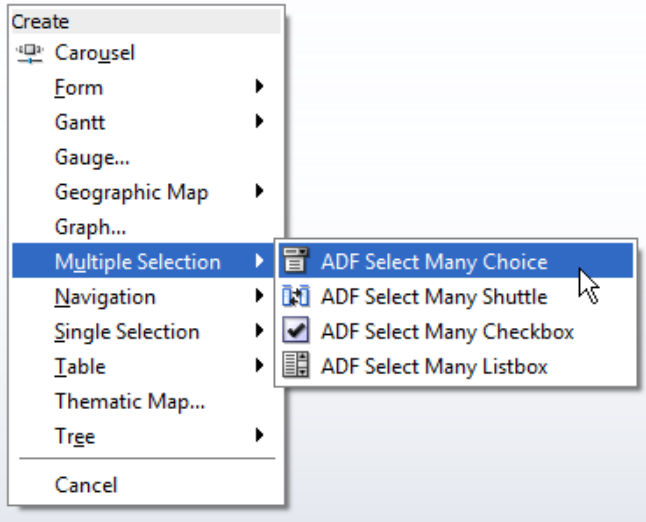
```
<af:tree value="#{bindings.AllCountries.treeModel}" var="node" …>
 <f:facet name="nodeStamp">
    <af:group id="g1">
   <af:commandLink id="cl4" text="#{node}"
         rendered="#{node.hierTypeBinding.viewDefName ==
                     'oracle.summit.model.views.CustomerVO'}"    …/>
```

```
    <af:outputText id="ot4" value="#{node}"
          rendered="#{node.hierTypeBinding.viewDefName ==
                      'oracle.summit.model.views.CountryVO'}"/>
  </af:group>
 </f:facet>
</af:tree>
```

The EL expression **#{node.hierTypeBinding.viewDefName}** returns the name of the node type, which in ADF is the absolute name of the collection instance rendering the node.

How-to access selected rows in af:selectManyChoice

To create an ADF bound select many choice component, drag a collection from the Data Controls panel to a JSF page. In the opened context menu, choose **Multiple Selection | ADF Select Many Choice**.



This generates the page source code similar to this:

```
<af:selectManyChoice value="#{bindings.allDepartments.inputValue}"
                     label="#{bindings.allDepartments.label}"
                     id="smc1">
    <f:selectItems value="#{bindings.allDepartments.items}" id="si1"/>
</af:selectManyChoice>
```

Note that the **value** property of the SelectManyChoice component pints to the sane list binding as the `f:selectItems` tag. To access the user selected values, you use a managed bean to access the ADF binding layer.

If you are only interested in the selected values, then you use code like shown below. Note that the sample I built used JDeveloper 11g R2, which is why the ID type is `Integer`. Doing the same with JDeveloper 11g R1 would require the casting to `oracle.jbo.domain.Number` for ID columns.

```
public String cb1_action() {
  BindingContext bctx = BindingContext.getCurrent();
  BindingContainer bindings = bctx.getCurrentBindingsEntry();
```

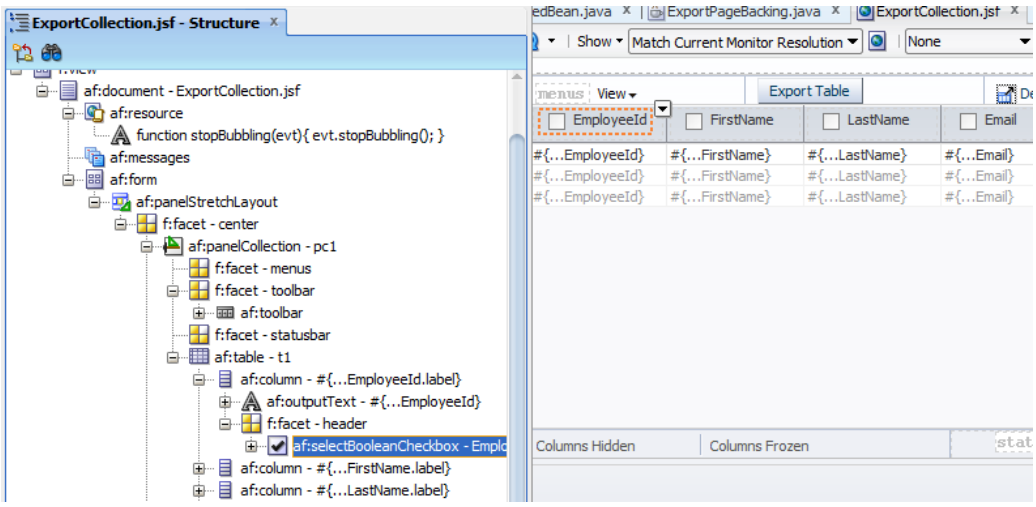
```
  JUCtrlListBinding allDepartsmentList =
           (JUCtrlListBinding) bindings.get("allDepartments");
  Object[] selVals = allDepartsmentList.getSelectedValues();
  for (int i = 0; i < selVals.length; i++) {
    Integer val = (Integer)selVals[i];
    //...
  }
  return null;
}
```

However, what if you wanted to know more about the selected list value; for example the value of another attribute of the selected row? In this case you use similar code, with a little change though. Instead of calling `getSelectedValues()`, the call is to `getSelectedIndices()`.

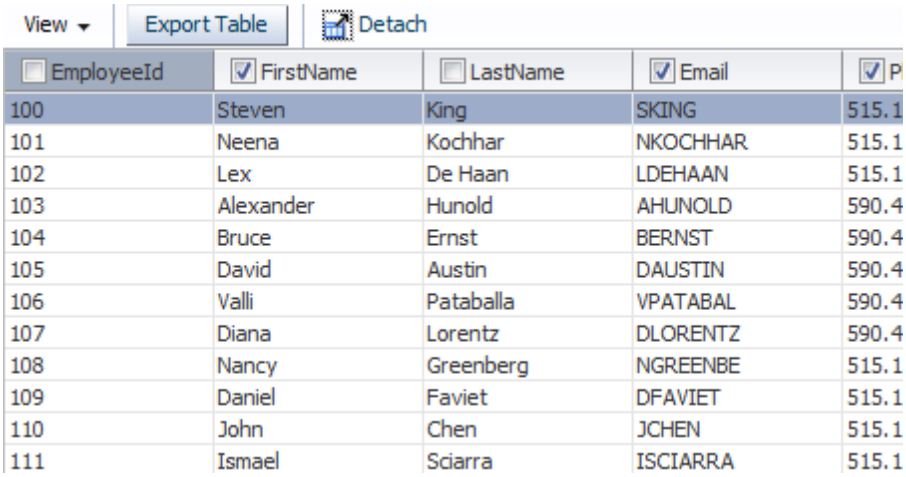
```
public String cb1_action() {
  BindingContext bctx = BindingContext.getCurrent();
  BindingContainer bindings = bctx.getCurrentBindingsEntry();
  JUCtrlListBinding allDepartsmentList =
          (JUCtrlListBinding) bindings.get("allDepartments");
  int[] selVals = allDepartsmentList.getSelectedIndices();
  for (int indx : selVals ) {
    Row rw = allDepartsmentList.getRowAtRangeIndex(indx);
    //... do your stuff
  }
  return null;
}
```

Adding checkboxes to sortable table headers

Using the header facet of the **af:column** tag, developers can add components to the table header, for example to indicate the component to be excluded from printing or export to Excel. The side effect of adding a command item to a table header is when the table column is sortable.

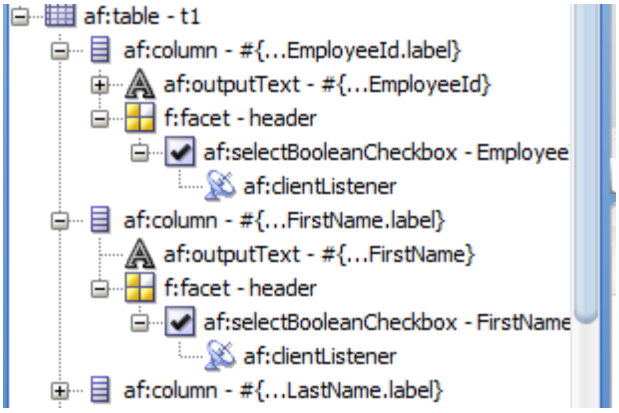


In this case, selecting the component, for example an instance of **af:selectBooleanCheckBox** , will also trigger sorting on the table, which you don't want to happen until users explicitly ask for this.



The solution to this problem is JavaScript added to the checkbox components and the use of stopBubbling() to prevent user the click event to reach the table header and then the sort icons.

```
<af:resource type="javascript">
   function stopBubbling(evt){
      evt.stopBubbling();
  }
</af:resource>
```




```
<af:selectBooleanCheckbox id="sbc2" text="FirstName"
                          value="…" autoSubmit="true">
   <af:clientListener method="stopBubbling" type="click"/>
</af:selectBooleanCheckbox>
```

ADF Code Corner

OTN Harvest Spotlight

- Timo Hahn

Timo Hahn is an ADF consultant with an Oracle partner in Germany and awarded Oracle ACE. Most ADF developers know Timo from his contribution on the OTN forum, where he shares his knowledge and wisdom.

Blog: twitter.com/tompeez

Twitter: http://tompeez.wordpress.com/author/tompeez/

ACC: What is your current role?

I'm a senior consultant with Steria Mummert Consulting AG. I'm working as an architect/developer for JEE applications, specialized in ADF, mostly for customers in the public sector.

ACC: What is your IT background?

I got my Master of Computer Science about 20 years ago and worked as developer since then. I started developing applications using C, C++. My first job, which I started when I was still at university, was to develop printer drivers for special printer hardware like micro fish imagers and various printing presses. This job let me work and travel all over Europe and the North America for about 5 years.

The next 5 years I spend developing an editorial system for daily newspapers including classified and billing system. The system was/is used in some medium sized German daily newspapers.

About 12 years ago I switched to Java when staring a new job in the insurance industry. The project was canceled after 2 years as it was too ambitious for the early stages of Java and J2EE.

Since 2004 I'm working with JDeveloper starting with version 9.0.4 and every version since. I started building Struts based web application using BC4J JBO Tags, BC4J business components and Oracle DB 10g.

Today I'm working with JDeveloper 11.1.1.4.0 using ADF Rich Faces and the full ADF technology stack and web services.

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

Currently I'm migrating and/or upgrading some of the applications written with JDeveloper 10.1.2 and Struts to JDeveloper 11.1.1.4.0. Part of the job is a redesign of the UI to ADF rich face and evaluating the current version of JDeveloper.

	At the same time I try to develop some guidelines and best practices for the other developers in the team. Beside that I'm architect and technical lead for ADF application design for our customers and in house projects.
ACC:	So far, what has been your biggest challenge in building Java EE application with Oracle ADF?
	The transition from 10g using Struts to 11g with ADF rich faces. It was a real challenge to master all the new stuff and the change of application servers in a short time. This was especially challenging as we had developed a wide range of reusable services which all needed to work on the new version to get an application running. An equal challenge is to train all developers and keep them up to date with the latest changes in design patterns and best practices.
ACC:	Which feature of ADF was the greatest benefit to your project?
	Let me name two: 1. The ADF rich faces, as they allowed us to generate nice and modern user interfaces with a great deal of reusability. 2. The binding layer as glue code between the ui and the business layer The main advantage is the great deal of standardization we are able to archive using ADF. In my opinion projects with >10 developers need a great deal of standardization to make them successful and maintainable.
ACC:	Away from the on line help, what have been your most valuable sources of ADF knowledge?
	The OTN forum. When I started using JDeveloper there wasn't really anything else. I spent the first 3-6 month reading many threads. Always looking for some hints on the problem I was working on. Back then most of the terms and the technology used were new to me. Working through the 'Toy Store' sample shed some light into matters, but reading threads I learned a lot about the framework and how it works. Still, I have to mention that it was (and is) a challenge, as you have to filter out the right information. Without the comments and input from the more senior forum users (Oracle employed or not) this wouldn't be possible. Today users can find sample code or even complete mini applications for almost every part of the framework. This is a great help, but users have to be careful not only to cut and paste but to understand what they are copying.
ACC:	Are you in any way actively involved in the ADF Community?
	I spend a couple of hours each day reading threads on the OTN forums. I try to help other developers by passing my experience in the answers to their questions.

	In the German ADF Partner Community I'm giving talks about various topics and I'm participating in the ADF Enterprise Methodology Group. I'm blogging about OFM, JDeveloper and especially on ADF related stuff.
ACC:	What do you recommend as a starting point and path to learn Oracle ADF?
	A good knowledge of Java in general is essential from my experience. Then Shay outlined a good learning path for ADF. Visit the OTN ADF forum and read and search for threads which handle about the problem area you are working on. I know searching the OTN forums is sometimes a hazard too, but it still pays off. Blogs are helpful too, as long as you don't just copy the code. In addition there are a couple of good books out (see Shay's article). If you start with a project it greatly helps to involve someone with experience in ADF to get it all started. Let me quote Sten Vesterli: 'You need a lead programmer and nutcracker…'
ACC:	By your experience and as estimation, how long does it take for new developer to learn ADF?
	This is a hard one to answer as it depends on the individual background and willingness to learn the framework. An experienced Java developer should be productive in 2-4 month. For a total novice in Java there is a steep learning curve from about 1-2 years. Putting a mentor in a team would help to speed up things too. If she/he fights the framework constantly, there is nothing much you can do but hope...
ACC:	You are very active on the OTN forum in helping new developers with ADF issues. What is in for you by doing this?
	Good question. I don't need something out of it actually. I just like helping out other developers and answering questions. When I started learning to work with JDeveloper I got help on the forum, now it's the other way around – payback time. That's OK with me. I still learn a lot reading threads and there are enough questions out there which I can't answer out of the box. These questions make me think how I would solve them. I then try my solution and this way I learn more and more about the framework.
ACC:	ADF Genie grants you a wish, what would you ask for?
	No especially ADF, but as I spend some time on the forum, so my wish would be new better forum software.