ADF Code Corner

Oracle JDeveloper OTN Harvest 03 / 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-MAR-2012

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

March 2012 Issue – Table of Contents

Remote Task Flow vs. WSRP Portlets 3
What happens when you choose cascade delete on an association ..3
Deploying ADF Security enabled applications to WLS.....5
When to save task flow definitions outside of WEB-INF? 5
Select-one components don't show required field errors 6
Using af:serverListener as a JS client-server proxy.....6
Customizing the af:query default mode 9
Gotcha when using JavaScript in ADF Regions 12
useBindVarsForViewCriteriaLiterals in adf-config.xml 13
Implementing case insensitive sort.....13
OTN Harvest Spotlight - Michael Koniotakis ("Milkbird") 16

Remote Task Flow vs. WSRP Portlets

A remote task flow is bounded task flow that is deployed as a stand-alone Java EE application on a remote server with its **URL Invoke** property set to **url-invoke-allowed**. The remote task flow is accessed either from a direct browser GET request or, when called from another ADF application, through the task flow call activity.

For more information about how to invoke remote task flows from a task flow call activity see chapter **15.6.4 How to Call a Bounded Task Flow Using a URL** of the Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework at

http://docs.oracle.com/cd/E23943_01/web.1111/b31974/taskflows_activities.htm#CHDJDJEF

Compared to WRSP portlets, remote task flows in Oracle JDeveloper 11g R1 and R2 have a functional limitation in that they cannot be embedded as a region on a page but require the calling ADF application to navigate off to another application and page. The difference between a remote task flow call using the task flow call activity and a simple redirect to a remote Java EE application is that the remote task flow has a state token attached that allows to restore the state of the calling application upon task flow return.

A use case for a remote task flow call activity is a "yellow page lookup" scenario in which different ADF applications use an remote task flow to lookup people, products or similar to return a selected value to the calling application.

Note that remote task flow calls need to be performed from a bounded or unbounded top level task flow of the calling application. If called from a region (using the parent call activity) in a page, the region state is not recovered upon task flow return.

ADF developers recently have identified remote task flows as an architecture pattern to partition their ADF applications into independently deployed Java EE applications. While this sounds like a desirable use of the remote task flow feature, it is not possible to achieve for as long as remote task flows don't render as an ADF region.

What happens when you choose cascade delete on an association

You edit ADF Business Components associations that are defined between entities by selecting the association in the JDeveloper Application Navigator and then choosing **Open <Name of Association>** from the right mouse menu.

One of the configuration options in the visual editor is **Implement Cascade Delete** that is located in the **Relationship** menu under the **Behavior** header. Selecting this option indicates that all detail rows that are associated with a parent entity should be deleted when the parent entity is deleted.

However, ADF Business Components does not itself perform the cascade delete, but expects a database constraint to be defined for this. All that the **Implement Cascade Delete** does is to change the delete command issued by ADF Business Components to the database.

To quote the "Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework"

http://docs.oracle.com/cd/E24382_01/web.1112/e16182/bcentities.htm#BABHFJFJ

" … When selected, this option allows the composing entity object to be removed unconditionally together with any composed children entities. If the related Optimize for Database Cascade Delete option is deselected, then the composed entity objects perform their normal `DELETE` statement at transaction commit time to make the changes permanent. If the option is selected, then the composed entities do not perform the `DELETE` statement on the assumption that the database `ON DELETE CASCADE` constraint will handle the deletion of the corresponding rows.

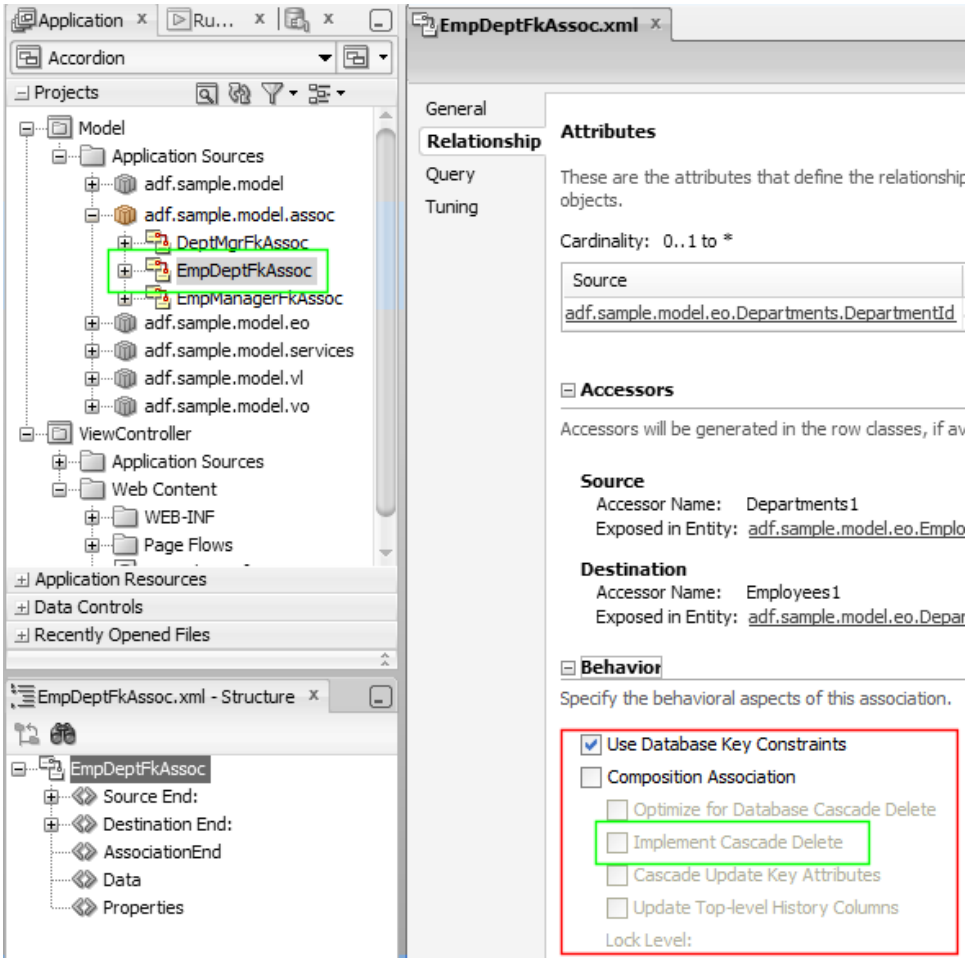


Figure: ADF Business Components association editor

So to make the **Implement Cascade Delete** option work, you need to first ensure the database constraint itself has the cascade delete option set

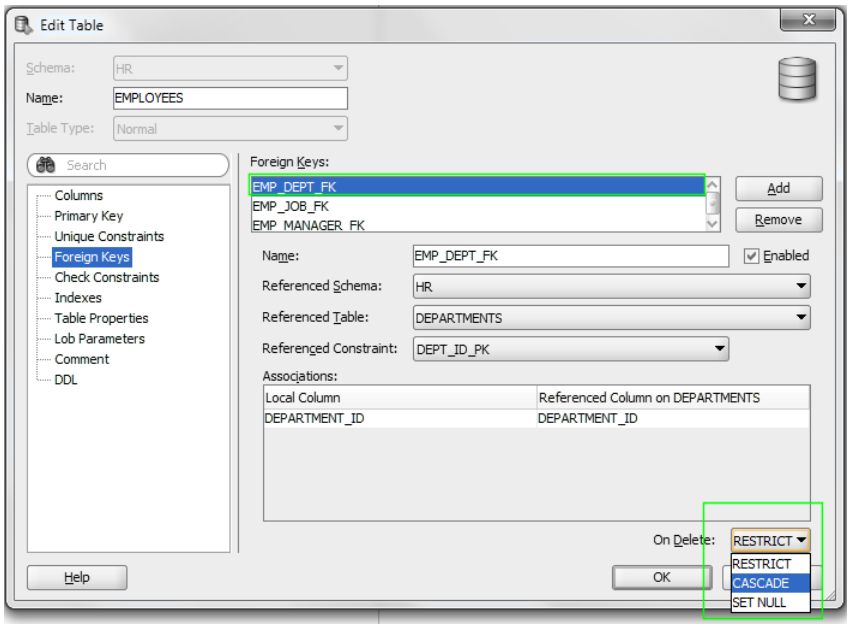


Figure: SQL Developer cascade constraint setting for foreign key

Deploying ADF Security enabled applications to WLS

Dmitry Nefedkin from Oracle produced a video blog entry that puts ADF Security explained in this article (http://www.oracle.com/technetwork/issue-archive/2012/12-jan/o12adf-1364748.html) into action on WebLogic Server and OID. The two videos that you access from here:

https://blogs.oracle.com/imc/entry/secured_your_adf_application_time

explain how to package ADF Security enabled applications up to an EAR file, deploy the application to WebLogic Server, enable Oracle Internet Directory for authentication and use Enterprise Manager to map ADF Security application roles (actually these are OPSS roles) to user groups (aka. enterprise groups or roles) in OID.

Another recording Dmitry did is on implementing Single Sign-On with Oracle Access Manager and is accessible from here: https://blogs.oracle.com/imc/entry/adf_oam

Note that all of the three recordings will also be available on the ADF Insider web page together with a new extended recording of the ADF Security overview session.

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/adfinsider-093342.html

When to save task flow definitions outside of WEB-INF?

To give you the answer to above question upfront: **never**. Task flow definition files are configuration details that are internal to ADF applications. There is no benefit or additional functionality applications gain from the task flow definition to become publicly accessible. In Java EE, all files saved in the WEB-INF directory are not accessible by a GET request. For security reasons you should save all files that are served by the ADF controller and the JSF servlet under the WEB-INF directory. Only image files, CSS and JavaScript make sense to be saved in a public directory structure.

Select-one components don't show required field errors

A problem recently experienced by a number of developers is the missing required field warning for select one components used in forms containing other input field components having its **required** property set to true. The effect at runtime is that required field messages for all input fields are shown but the message for the select one component is missing. Only if the select one component is the only component left with a missing value selection, the message displays.

The defect has been filed as bug 13822582 and **the work around** is to set the **unselectedLabel** property of the select one component.

http://docs.oracle.com/cd/E23943_01/apirefs.1111/e12419/tagdoc/af_selectOneRadio.html

http://docs.oracle.com/cd/E23943_01/apirefs.1111/e12419/tagdoc/af_selectOneListbox.html

http://docs.oracle.com/cd/E23943_01/apirefs.1111/e12419/tagdoc/af_selectOneChoice.html

Using af:serverListener as a JS client-server proxy

Despite of ADF Faces having a client side JavaScript architecture, **JavaScript rule #1** in ADF Faces is to use JavaScript as a fallback option only for development use cases in which there is no native solution to a problem. A built-in feature of the ADF Faces JavaScript client architecture is security that disallows certain component properties like **disabled** and **readOnly** to be changed from JavaScript.

To quote the JavaScript doc for the AdfRichInputText object:

http://docs.oracle.com/cd/E12839_01/apirefs.1111/e12046/oracle/adf/view/js/component/rich/input/AdfRichInputText.html#getReadOnly__

public Boolean ***getReadOnly****()*

Get function for attribute for 'readOnly'. This attribute is secured. You may get it, but you may not set it. Any changes to this attribute will not be transmitted to the server.

Note: The **disabled** property can be enabled for JavaScript modification by setting the component **unsecure** property as explained in the component documentation, which however I don't recommend you to do. If you need to open protected features for modification, do so in a way that you control, for example in that you can check a user permission to do so.

There might be use cases in which you need to change a protected property from JavaScript, and here is how this can and should be done:

The ADF Faces `af:serverListener` component allows developers to invoke server side Java from JavaScript and thus can be used as a proxy to bypass applied client side security.

For example, to be able to change the **readOnly** property on an `af:inputText` component, you define the text field as shown below

```
<af:inputText label="Label 2" id="it2"
              clientComponent="true"
              value="#{SetInputTextFieldReadOnly.textValue}">
```

```
    <af:serverListener type="serverAction"
                       method="#{SetInputTextFieldReadOnly.actionEvent}"/>
</af:inputText>
```

The **clientComponent** property need to be set to **true** on the component to ensure a JavaScript object is created on the client architecture. With no **clientComponent** set to true and no `af:clientListener` attached, the text field component is not accessible from JavaScript at all.

The `af:serverListener` needs to be configured on the component that is passed as a component reference to the server. In the sample, the component reference that is sent to the server is the input text field that you want to switch from updateable to read-only. If the integration is from a non ADF Faces component, like an Applet, you would configure the `af:serverListener` on the `af:document` component. If this is the case, you may not make use of the component reference on the server and instead pass additional payload arguments.

In the Oracle JDeveloper 11.1.1.6 sample to this article the switch between read-only and updateable is simulated by a command button you can press. The sample can be downloaded from here: https://blogs.oracle.com/jdevotnharvest/resource/SetInputTextToReadOnlyJS.zip

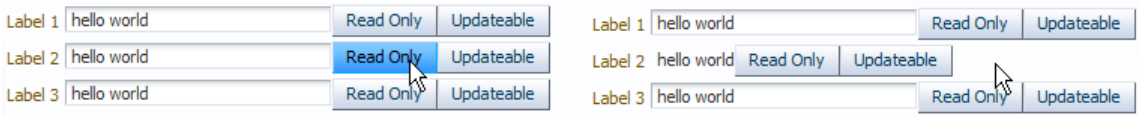


Figure: switch from Updateable to Read-only

Figure: switch from Read-only to Updateable

In a realistic use case, this switch would be triggered from a non ADF Faces component like Applet or another technology you integrate in your ADF Faces page.

Note: The use of a command button in the sample is only for demonstration purpose. If the trigger was a command button then no JavaScript would be needed at all and JavaScript rule #1 would apply

On the server, a managed bean method is invoked by the client JavaScript call shown below:

```
<af:resource type="javascript">
   function setUpdateable(actionEvent){
       actionEvent.cancel();
       var commandButton = actionEvent.getSource();
       var textFieldId = commandButton.getProperty('componentId');
       invokeServerAction(textFieldId,actionEvent,'true');
}
```

```
function setReadOnly(actionEvent){
       actionEvent.cancel();
       var commandButton = actionEvent.getSource();
       var textFieldId = commandButton.getProperty('componentId');
       invokeServerAction(textFieldId,actionEvent,'false');
}

function invokeServerAction(compId,evt,isUpdateable){
  var textField = evt.getSource().findComponent(compId);
  AdfCustomEvent.queue(
         textField,"serverAction",
         {updateable: isUpdateable},false);
   }

</af:resource>
```

Note: payload arguments are surrounded by curly braces. If you want to pass multiple arguments then you use a comma separated list of *key1:value1, key2:value2* …pairs.

The JavaScript is taken from the sample provided for this tip and – as mentioned – is invoked from a command button. The command button has a custom property (`af:clientAttribute`) added that tells the script about the text component Id to modify. The component and the desired switch state "readOnly" or "updateable" is passed to the managed bean method:

```
/**
  * Proxy the client request
  * @param clientEvent The client event gives you access to the
  * component that raises the event as well as to the paraneters
  * passed from the client to the server
*/

public void actionEvent(ClientEvent clientEvent) {
  /*
   * This would be a good place to check an ADF Security resource
   * permission if the user is allowed to perform the change on the
   * attribute. There is no security in JavaScript, but using server
   * side JAAS protects you from unauthorized changes
   */

   RichInputText inputText =
               (RichInputText) clientEvent.getComponent();
   String updateable =
          (String) clientEvent.getParameters().get("updateable");
   inputText.setReadOnly(!Boolean.parseBoolean(updateable));
   AdfFacesContext adfFacesContext =
                           AdfFacesContext.getCurrentInstance();
```

```
    adfFacesContext.addPartialTarget(inputText);
}
```

Sample Download: https://blogs.oracle.com/jdevotnharvest/resource/SetInputTextToReadOnlyJS.zip

Customizing the af:query default mode

To create a search form in ADF, you drag the **All Queriable Attributes** Named Criteria that exists for all View Object entries in the Data Control panel to the page following the steps shown in Figure1 and explained below.

1. Expand the View Object node in the Data Controls panel and drag the **All Queriable Attributes** entry under the **Named Criteria** node onto the page
2. Choose **ADF Query Panel with Table** in the opened menu
3. Configure the table for row selection, filter and sorting behavior as you need it

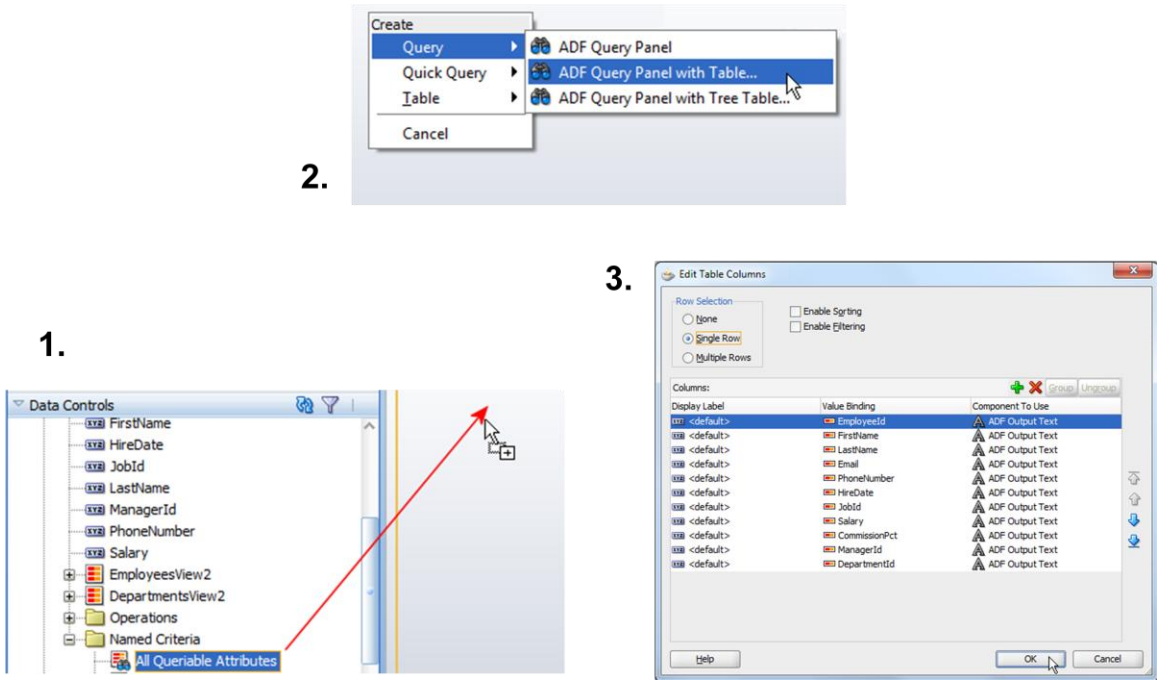


Figure 1: Creating a search form

At runtime, the search form renders in basic search mode as shown to the left in Figure 2.

But what can you do in case you don't like this to start in basic mode but advanced mode as shown to the right in Figure 2? Or, what if you want the search to be case insensitive? As you may guess from the right hand side in Figure 2, there is a solution to this, which is to create a named View Criteria.

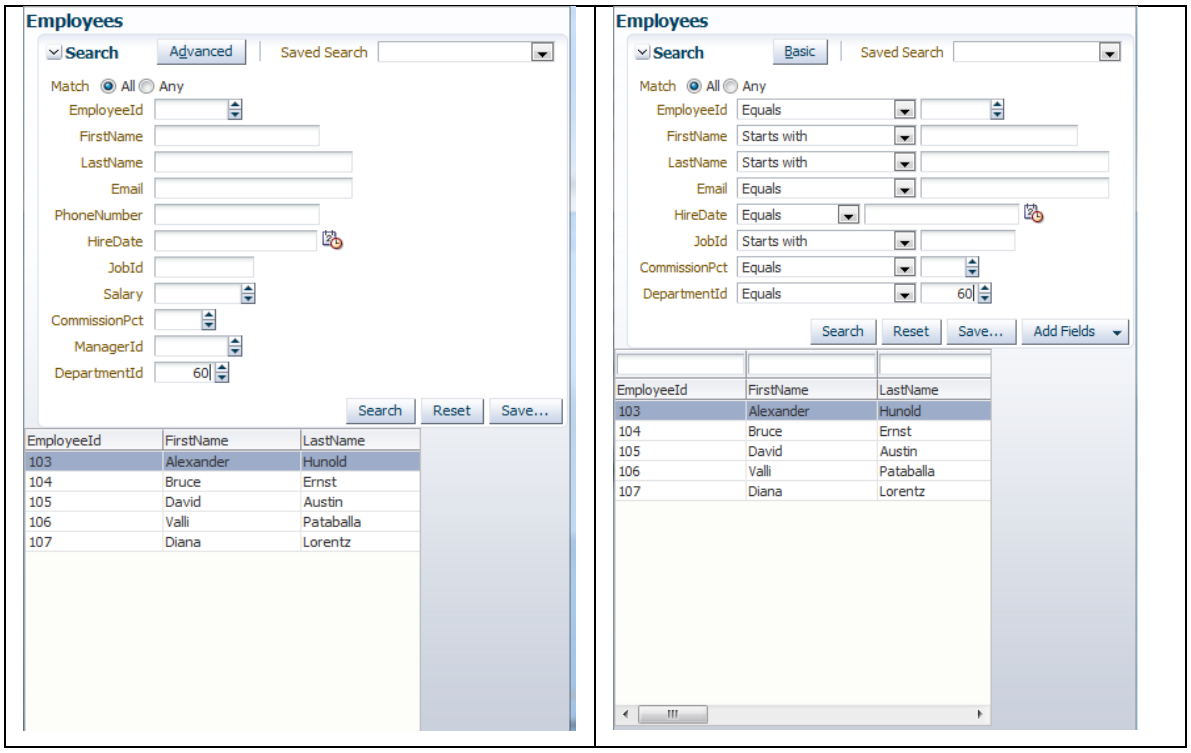


Figure 2: Basic default vs. **Advanced** default

To create a named View Criteria, open the View Object from the JDeveloper Application Navigator with a double click. Select the **Query** menu item and click the green plus icon next to the View Criteria header as shown in Figure 3.

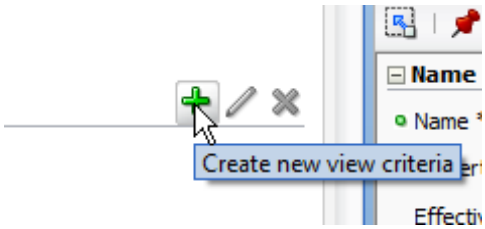


Figure 3: Create a new View Criteria

Shown in Figure 4 is the editor for defining the View Criteria clause. Just select the **Group** node as shown in Figure 4 and press **Add Item** as many times as you have attributes exposed in the View object. If your View Object has a view link defined, you may see other View Object instance. Don't select this instance.

In the same dialog notice the **Ignore Case** option, which you use to filter a query case independent. So if your requirement is to make the search form behave case insensitive in its query, then choose this option.

Also notice the **Query Execution Mode** on the top right that defines on which data the query should be executed. The options are **Database**, **In Memory**, **Both**

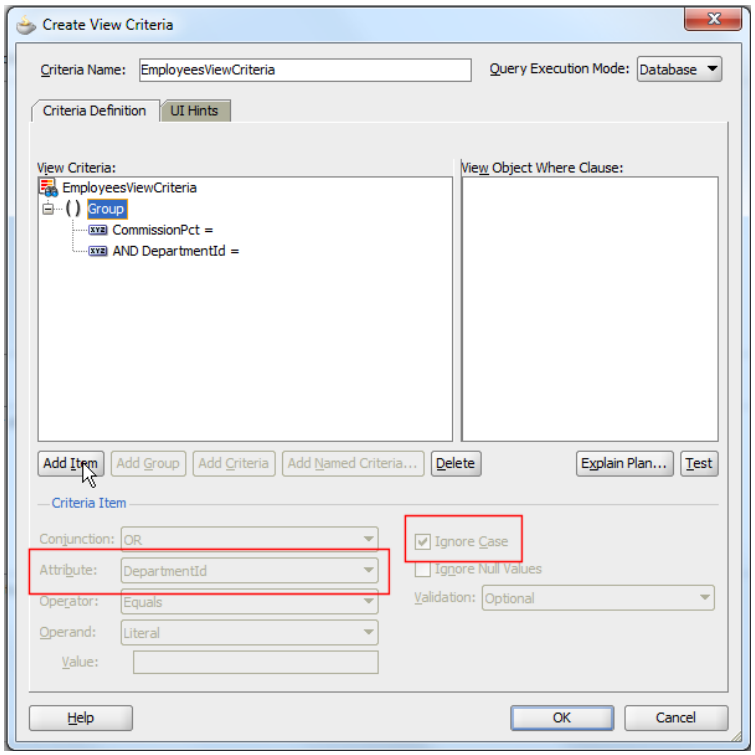


Figure 4: View Criteria definition dialog

Selecting the **UI Hints** tab, you can now define how you want to initially start the search form.

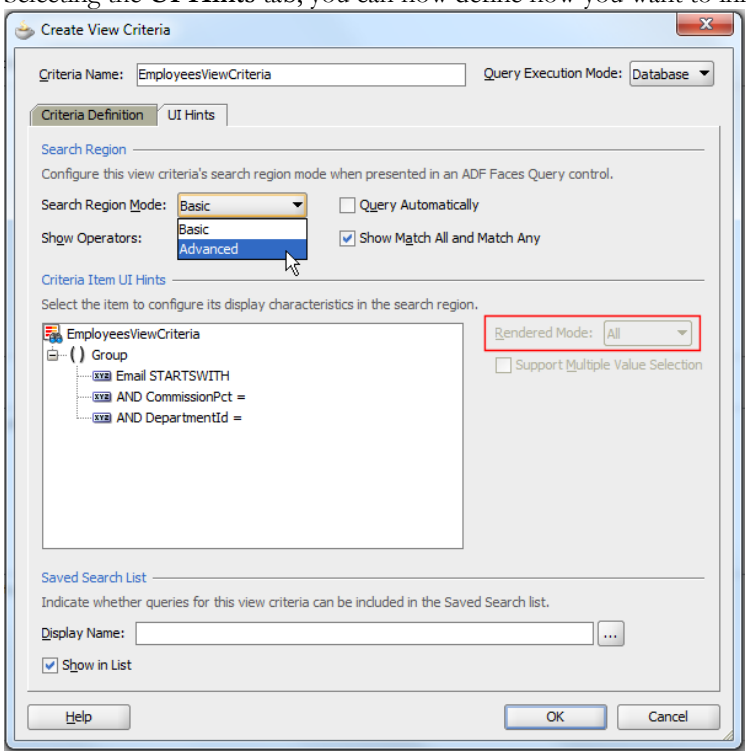


Figure 5: View Criteria UI hints tab

As shown in figure 5, set the **Search Region Mode** property to **Advanced** to always start in advanced mode.

Notice the **Rendered Mode** option in figure 5. The **Rendered Mode** option allows you for each attribute to define whether it is shown in both search modes, in a specific mode only or never.

To create the search form based on the custom View Criteria, expand the View Object node in the Data Controls panel. Expand the **Named Criteria** node and drag the custom View Criteria onto a JSF page. In the opened dialog choose the Query option as shown in Figure 6. Note that if you choose the **ADF**

Query Panel with Table with option and configure the table to show column filters, the column filter will use the same View Criteria.

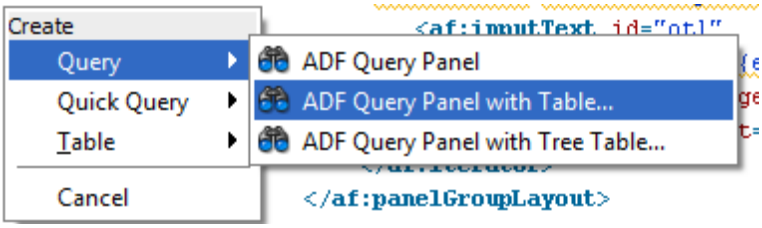


Figure 6: ADF Query panel dialog

Gotcha when using JavaScript in ADF Regions

You use the ADF Faces `af:resource` tag to add or reference JavaScript on a page. However, adding the `af:resource` tag to a page fragment may not produce the desired result if the script is added as shown below

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
 <af:resource type="javascript">
  function yourMethod(evt){ ... }
</af:resource>
```

Adding scripts to a page fragment like this will see the script added for the first page fragment loaded by an ADF region but not for any subsequent fragment navigated to within the context of task flow navigation. The cause of this problem is caching as the af:resource tag is a JSP element and not a lazy loaded JSF component, which makes it a candidate for caching.

To solve the problem, move the `af:resource` tag into a container component like af:panelFormLayout so the script is added when the component is instantiated and added to the page.

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
          xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
<af:panelFormLayout>
 <af:resource type="javascript">
    function yourMethod(evt){ ... }
```

```
 </af:resource>
</af:panelFormLayout>
```

Magically this then works and prevents browser caching of the script when using page fragments.

useBindVarsForViewCriteriaLiterals in adf-config.xml

Found an interesting blog post by Oracle's Jobinesh Purushothaman about a new adf-config setting in Oracle JDeveloper 11g R2. Jobinesh posted it in Summer 2011, but definitively it is worth to be brought up again.

http://jobinesh.blogspot.de/2011/08/what-you-may-need-to-know-about.html

" The 11.1.2 release has introduced a new flag useBindVarsForViewCriteriaLiterals in your application's adf-config.xml.

```
<?xml version="1.0" encoding="US-ASCII" ?>
 <adf-config .... >
   <adf-adfm-config xmlns="http://xmlns.oracle.com/adfm/config">
   <defaults useBindVarsForViewCriteriaLiterals="true"/>
   ...
  </adf-adfm-config>
   ...
```

Idea is to force the run time to generate temporary bind variables instead of directly using literal values while generating WHERE clause for the ViewCriteria. The above said configuration is done at application level which will set ViewCriteria::setUseBindVarsForLiterals(true) for all VC instances.

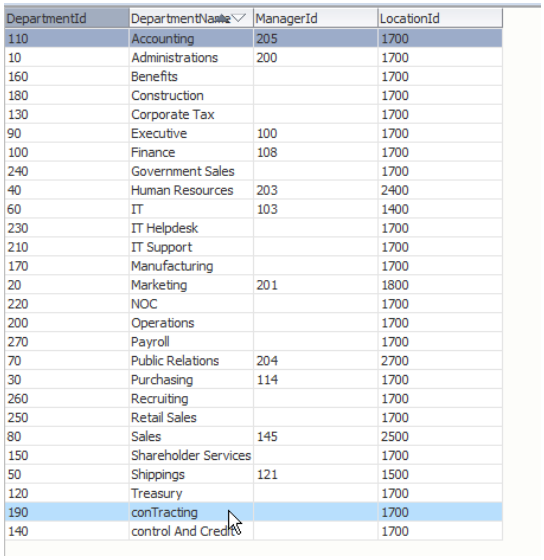
- *This will help to improve performance of query execution by caching SQLs*
- *Reduce/avoid the chance for SQL injection"*

The interesting aspect is security and the prevention of SQL injection attacks, which the use of bind variables is a counter measure for.

Implementing case insensitive sort

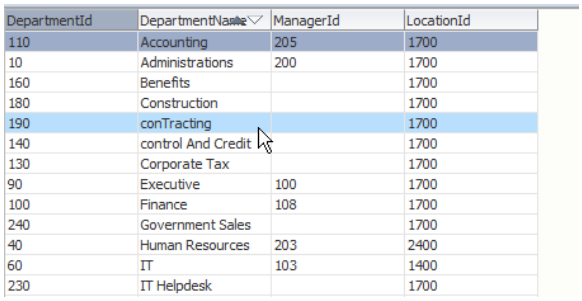
By default, sorting performed on a table using ADF Business Components is not case insensitive. To change the sorting behavior to case insensitive for a field, you can use transient attributes. Here's how to implement this solution:

Figure 1 shows a screen shot of the default case sensitive sorting behavior in Oracle ADF. As you can see the lower case occurrences of the initial "c" words appear later in the sorting than the upper case versions. Figure 2 shows the result after applying the tip in this post. You see that the sorting now shows all words that start with an initial "c" or "C" are ordered next to each other.



DepartmentId	DepartmentName	ManagerId	LocationId
110	Accounting	205	1700
10	Administrations	200	1700
160	Benefits		1700
180	Construction		1700
130	Corporate Tax		1700
90	Executive	100	1700
100	Finance	108	1700
240	Government Sales		1700
40	Human Resources	203	2400
60	IT	103	1400
230	IT Helpdesk		1700
210	IT Support		1700
170	Manufacturing		1700
20	Marketing	201	1800
220	NOC		1700
200	Operations		1700
270	Payroll		1700
70	Public Relations	204	2700
30	Purchasing	114	1700
260	Recruiting		1700
250	Retail Sales		1700
80	Sales	145	2500
150	Shareholder Services		1700
50	Shippings	121	1500
120	Treasury		1700
190	conTracting		1700
140	control And Credit		1700

Figure 1: Default sorting beavior



DepartmentId	DepartmentName	ManagerId	LocationId
110	Accounting	205	1700
10	Administrations	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
40	Human Resources	203	2400
60	IT	103	1400
230	IT Helpdesk		1700

Figure 2: Case insensitive sorting behavior

The implementation of case insensitive queries in this tip is based on transient attributes you create for each attribute you want have sorted case insensitive. To create a transient attribute, press the green plus icon in **Attributes** view of the ADF Business Components entity editor (Figure 3).

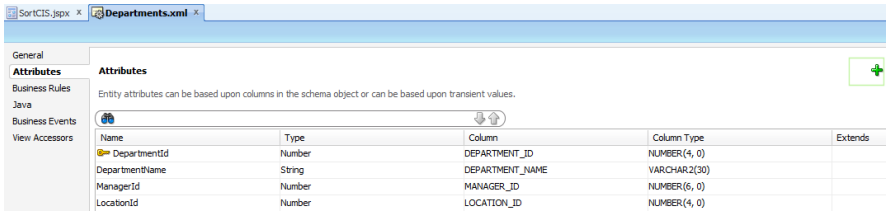


Figure 3: Creating a transient entity attribute

As shown in figure 4, the transient attribute **Value Type** property is set to **Expression** so Groovy can be used to derive its value. In the sample shown in the screen shots, the **DepartmentName** attribute, should be sorted case insensitive. The transient attribute **DepartmentNameCIS** references uses Groovy to reference the **DepartmentName** value.

```
DepartmentName.toUpperCase()
```

Also shown in figure 4, the transient attribute is configured to be dependent on changes to the **DepartmentName** attribute, so the two stay in synch.

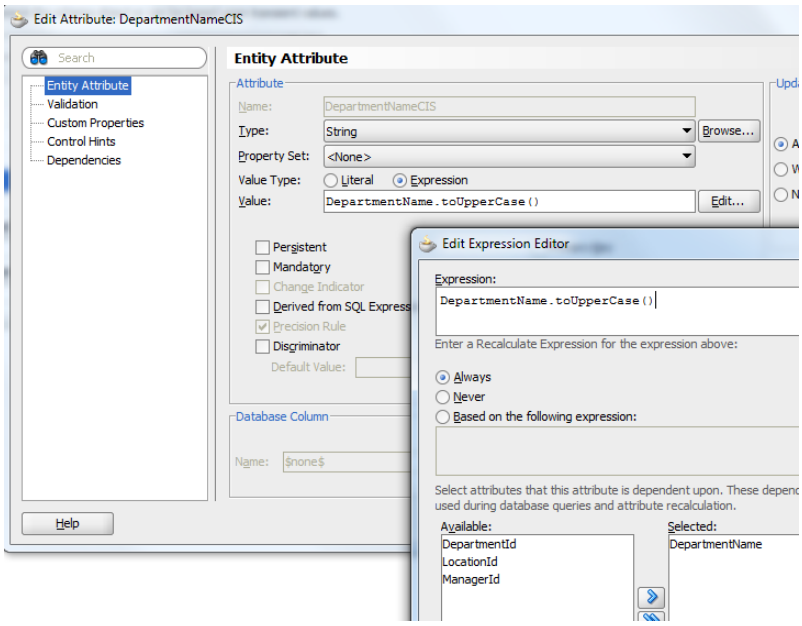


Figure 4: Populating the transient attribute with a Groovy expression

The transient attribute on the entity object then need to be exposed on the View Object by pressing the arrow icon next to the green plus icon and choosing **Add Attribute from Entity**.

You build the ADF bound ADF Faces table as you always do: Select the View Object from the Data Controls panel and choose the table option from the context menu. Configure the table to support sorting (check box).

If you did not remove the transient attribute in the table configuration editor, or have set the transient attribute UI hint to be hidden from the UI, you can delete the table column representing the transient attribute (**DepartmentNameCIS** in the example) from the rendered table at design time.

Then select the column you want to sort case insensitive and set its **SortProperty** property to the name of the transient attribute (**DepartmentCIS** in the example). This then sorts the transient attribute whenever the *Asc* or *Desc* sort icons are pressed on the column you want to sort case insensitive.

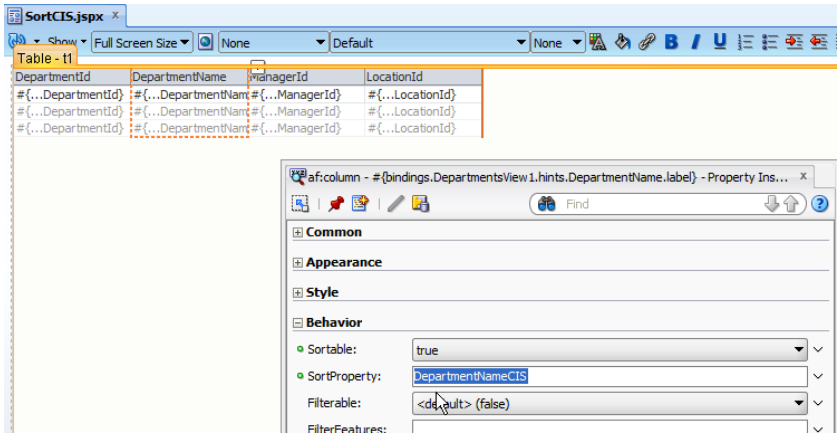


Figure 6: Configuring the sort behavior of the DepartmentName column

ADF Code Corner

OTN Harvest Spotlight

- Michael Koniotakis ("Milkbird")

"It is not enough to know the capabilities of a framework. You really need to know its limitations in order to master it."
- Michael Koniotakis

Blog: http://adfbugs.blogspot.co.uk/

Twitter: https://twitter.com/#!/mkoniotakis

ACC: What is your current role?

MK: I change hats many times even during the day.

My current role is working with Oracle ADF on major software systems, mainly involving healthcare insurance. I have been continuously involved with Oracle products for more than ten years.

My tasks include solution Architecture, technical problem solving, coaching, developing, reviewing, etc.

ACC: What is your IT background?

MK: I got my first PC on my first year at university on 1990 and since then I have spent more time in front on a computer than sleeping. I started with the usual university stuff of the era Fortran, Basic and such, which was to be expected since I was majoring in Chemical Engineering. But the IT bug got me for good, so I went down that career path.

With the millennium change I got the chance to work on large software projects, mainly public sector stuff with Oracle DB, PL/SQL, Forms, Reports, Portal and such.

On 2004 I started working with Java and the newer Oracle product line; JDeveloper and Business Components, first with UIX and later with JSF. I had the chance also to get involved with open source Java frameworks, but I remained mostly an Oracle guy. I guess the data-centric nature of my project work was a natural fit; and I really liked Business Components.

Then come 2007 I had the luck to work with MedNet International Ltd where the decision was taken to move on a mission-critical project to Java using the Oracle technology stack. So we started working on JDeveloper 11g even from the first Technology Preview version.

The MedNet International project was among the first Customer success stories of Oracle and boosted the company's innovation, sales, and my career. I started blogging and communicating with Oracle people very frequently.
Then, at some point, Steve Muench nominated me Oracle ACE; that was in 2010.

	For the last two years I keep working on ADF but I am also getting good exposure to SOA and BPM suites, as things are evolving both on the technology and business fronts. I have to say I really like the way these things are coming together at last. These are very interesting times.
ACC:	How do you currently use Oracle JDeveloper and ADF?
MK:	Well, you will find Oracle JDeveloper open on my PC all the time. I mostly work on standard Fusion Applications with ADF Business Components and ADF Faces. Lately we are also doing more work with Web Services; wrap and expose Business Components as web services, this kind of stuff. And there is the SOA exposure too.
ACC:	So far, what has been your biggest challenge in building Java EE application with Oracle ADF?
MK:	The biggest challenge was at the beginning; we had specific requirements to exactly reproduce Forms functionality in ADF. That was with Oracle JDeveloper 11g technology preview versions, pre-production stuff, the very first releases. It frequently seemed mission impossible to pull off requirements like 'It should work exactly the way Forms worked' and 'We have to use the same database and procedure based logic'. Think procedure-based application modules returning REF CURSORS for that one, and the JDeveloper manual was still in its early drafts. I had experience with JDeveloper9*i* and 10*g*, but it was on new projects with no such constraints. Yet we managed to reach quite similar functionality and to actually prove that some differences can really be enhancements :)
ACC:	Which feature of ADF was the greatest benefit to your project?
MK:	I feel that the greatest benefit was the declarative development. This gave the opportunity to developers with no ADF experience to quickly be productive. Also, custom made add-ons to JDeveloper are a big plus; actually a very well known open source add-on started life as a developer's side work on the MedNet project. The declarative XML development boosted productivity and the adoption of the technology. Yet some of my hard core develop colleagues felt that even though productivity is increased, creativity is reduced. Their point of view is that with code-centric Java you can start writing stuff and given enough creativity you create your masterpiece. I beg to differ; I believe that you can be as creative not only with words and symbols but also with ready-made components and properties, that you can use and transform and change many times until you create whatever you have in mind, so fast that you have more time to create more. JDeveloper is that environment – as long as you don't fight this basic premise.
ACC:	Away from the on line help, what have been your most valuable sources of ADF knowledge?
MK:	At the beginning of JDeveloper 11*g* the only sources were the OTN forums. It is still a valuable

reference since I still search for posts made years ago. Many times you remember that there was an issue but do not remember the solution. There you will find it.

After a while that bloggers started to master the framework and post their findings, many solutions were found in the blogs. Actually I think it's the JDeveloper community that started the how-to blogging thread.

ACC: Are you in any way actively involved in the ADF Community?

MK: I try to share my experience with the ADF community by posting in OTN forums, ADF EMG and in my blog.

I also made some ADF presentation in public groups and attended some Oracle events. I must admit that lately I am not so active as I would like due to heavy work load, yet I hope to be able to devote more time in the near future.

ACC: Your blog is well known in the community. What was the initial idea you had for running a blog mostly on software defects?

MK: The idea started with the large amount of issues that we had at the beginning of our major project, with the environment still being a work in progress.

We could not even distinguish if it was our bad development practice, a documentation issue or if there was something wrong with the framework. It was not easy to describe the problem in the OTN forums I needed screenshots, test cases and my own references to defects. Yet it turned out that other developers were also interested in this.

I believe that software defects are not easy to define. Let's take an end user that gives some requirements. He asked for a black background color and he got red instead. Is this easily defined as bug? What if red is actually better? What if you gave him the ability to change it whenever he wants? Yet when he makes it white he can no longer see the option to change it back; is it a bug? I understand the above example may be naïve, but I only try to make a point.

So even in the simplest cases many issues can arise. Things are even more complex for software development tools. We all create software by using frameworks that others have developed. So what is a bug in a software framework?:

- For sure things that work different than what described in the documentation

What about:

- Things that cannot be done by the framework?
- Things that could be done easier?
- Things that could work faster?
- Functionality that is not documented?

I believe it is not enough to know the capabilities of a framework. You really need to know its limitations in order to master it.

ACC: Are there any follow ups on the issues you find, like bugs filed or Oracle PM Follow ups?

MK: I can say that most of the issues are resolved little by little in newer JDeveloper releases. I had also some good replies and comments from Oracle PMs on posts that helped to resolve the

	issue or find a workaround. I post also the issues as SRs in oracle support, though I cannot say that I am always pleased with the responsiveness or the resolutions
ACC:	ADF Genie grants you a wish, what would you ask for?
MK:	I would ask for better visibility in Oracle Support. I believe it has an enormous knowledge database that should be the most valuable source of ADF knowledge. Yet it has many security restrictions, it is hard to access and hard to use. This would probably make my blog unnecessary, yet I could focus to other subjects.
ACC:	Thank you Michael.