ADF Code Corner

Oracle JDeveloper OTN Harvest 04 / 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-APR-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

April 2012 Issue – Table of Contents

How to activate and passivate ADF BC user session data

User session specific information, like a queried user name or account data, can be saved in the Oracle ADF Business Components session using a call to getUserData() that returns a java.util.`HashMap` as a data store. The user data map can be accessed from Java for read and write by calling …

```
getDBTransaction().getSession().getUserData()
```

on an ApplicationModule or Entity impl class. Alternative, if user data needs to be referenced from an attribute or validator in ADF Business Components, you can use Groovy too.

```
adf.userSession.userData.<the map key>
```

However, the user specific information in the userData `HashMap` is not persisted by default when activation / passivation occurs for application modules that have Application Module pooling enabled, which means that custom session data may be lost between requests.

Note: Read http://docs.oracle.com/cd/E23943_01/web.1111/b31974/bcstatemgmt.htm#sm0318 to learn more about application state management in Oracle ADF Business Components. This document also explains which information automatically gets passivated and activated by the framework.

Avoiding user session data losses: In 2011, Timo Hahn wrote a blog entry in which he explains the problem of lost user session data and how to solve the problem using explicitly passivation in Java.

http://tompeez.wordpress.com/2011/07/08/jdev-always-test-your-app-with-applicationmodule-pooling-turned-off/

As a general recommendation, ADF Business Component applications should always be tested for activation / passivation safety as explained in this document

http://docs.oracle.com/cd/E23943_01/web.1111/b31974/bcstatemgmt.htm#CHDGAIFA

Which option to choose for accessing Web Services in ADF

Oracle ADF provides three options for integrating Web Service:

- Web Service Data Control
- JAX-WS proxy client and POJO Data Control
- JAX-WS proxy client and programmatic View Objects

Note: In the above, I exclude REST services in my recommendation because this is what ADF will address much better using the URL data control in JDeveloper 11g R2 (available) and with improved functionality in the upcoming Oracle JDeveloper 12c release.

For deciding which option to use for integrating Web Service, here is what I consider "a good rule of thumb"

1. Use **Web Service Data Control** only for simple service like weather reports or stock quotes

2. Use **JAX-WS proxy client** for all more complex services and **access them from**
    a. Programmatic view object and entity if your business service is ADF Business Components as this allows for better integration with database queried views
        i. Use View Objects only for read only access
        ii. Use View Objects and Entities for CRUD Web Service integration
    b. POJO Data Control
        i. If your business service is not ADF BC.
        ii. If the WS doesn't require integration into ADF business components
        iii. If the WS access should be from a bounded task flow in an ADF library for maximum reuse

    As a general hint of best practice: Never use the JAX-WS generated proxy client directly. Always access it from a wrapper bean to avoid code losses or problem in cases where the proxy client needs to be re-generated.

    Note: ADF Code Corner published an article explaining how to cache Web Services results when using JAX-WS proxy clients to avoid unnecessary round trips.

    See: http://www.oracle.com/technetwork/developer-tools/adf/learnmore/92-cache-ws-queries-523136.pdf

Required vs. showRequired: and what could go wrong

The `af:inputText` component has two properties that you can use to show a icon indicating the field to be required.

The **required** property, when set to **true** always shows an icon next to the field that indicates this field to be required. If the **required** property is set to **false** then, if the **showRequired** property is set to **true**, the required icon also is shown though a required value is not enforced upon field validation.

This said, the **showRequired** property is a switch to show or hide the required icon in case that an input field doesn't enforce users to provide a value A use case for when to use the **showRequired** property is provided in the tag documentation

" An example of when it can be desirable to use the showRequired property is if you have a field that is initially empty and is required only if some other field on the page is touched."

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

And here's where things can get ugly

One of the ADF Faces layout components, the `af:panelLabelAndMessage` component also has a **showRequired** attribute you set to show a required icon.

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

The `af:panelLabelAndMessage` layout container is often used to add multiple UI components, like and input text component and a checkbox or command button, to a cell of an `af:panelFormLayout` container.


```
<af:panelFormLayout>
  <af:panelLabelAndMessage label="Label" id="plam2" showRequired="true">
     <af:panelGroupLayout id="pgl0" layout="horizontal">
      <af:inputText id="it6" required="false" showRequired="false"/>
      <af:commandButton text="LOV" id="cb1"/>
  </af:panelGroupLayout>
 </af:panelLabelAndMessage>
</af:panelFormLayout>
```

If the `af:inputText` component **required** property is set to **true**, the rendered output is as shown below. Now there are two required field icons, which most likely is not what you want. Same is the case if the **required** property is **false** and the **showRequired** property is **true**.

How to remove one of the required field icons? The answer to this question is "skinning". The following skin definition hides the required icon from all input text components

```
af|inputText::label .AFRequiredIconStyle{display:none}
```

To only remove the required field icon from some input text fields, you can use a **styleClass** property as shown below

```
<af:inputText id="it6" required="true" showRequired="false" styleClass="noRequiredIcon"/>
```

The skin definition needs to change to the one shown below to remove the required field icon next to the text field

```
.noRequiredIcon af|inputText::label .AFRequiredIconStyle{display:none}
```

However, you may wonder why the `af|inputText ::label` selector is still in the statement though the style class reference probably would be sufficient. In CSS the longest match rules so that adding the component selector, other components that also have required field icons are not impacted even if they also have the **styleClass="noRequiredIcon"** set. This way you can handle the setting/hiding of the required field icon dependent on the UI component.

Note: the **styleClass** property can be set dynamically using EL. Using EL you can dynamically set the style class and thus show/hide the redundant required icon.

Navigate regions using "queueActionEventInRegion"

A common requirement in Oracle ADF is to perform navigation within an ADF region triggered by the parent page. While contextual event is one option to perform this task, the **queueActionEventInRegion** method exposed on the RichRegion instance is another.

The **queueActionEventInRegion** performs navigation following control flow cases defined for the current view exposed in the region. Control flow cases can be flows that are defined from the view activity to a next activity or wild card control flow cases.

The easiest way for developers to access the RichRegion instance of an af:region tag is to use its **Binding** property and point it to a managed bean. Once you have a handle to the RichRegion, you can perform navigation from any event raised on the parent view (e.g. menu items action, command button action, value change events etc.)

The sample code below is invoked from the action property on a command button and performs navigation to the control flow case name defined in "navigationCase" variable. In a sample application you can download from here, the **navigationCase** variable is populated from a **af:selectOneChoice** component selection.

```
//process navigation
public String navPickerAction() {
  //get selected navigation option
  String navigationCase = this.currentNavOption;
  RichRegion region = this.findRegionById("adfRegion1");
  region.queueActionEventInRegion(
       createMethodExpressionFromString(navigationCase),
                                        null, null,false, 0, 0,
                                        PhaseId.INVOKE_APPLICATION);
  return null;
}

//Create Method expression
private MethodExpression createMethodExpressionFromString(String s){
  FacesContext fctx = FacesContext.getCurrentInstance();
  ELContext elctx = fctx.getELContext();
  ExpressionFactory exprFactory =
     fctx.getApplication().getExpressionFactory();
     MethodExpression methodExpr = exprFactory.createMethodExpression(
                 elctx,
                 s,
                 null,
                 new Class[]{});
  return methodExpr;
}
```

Another useful method on the RichRegion instance is the ability to peek into the region for control flow cases defined for the current view.

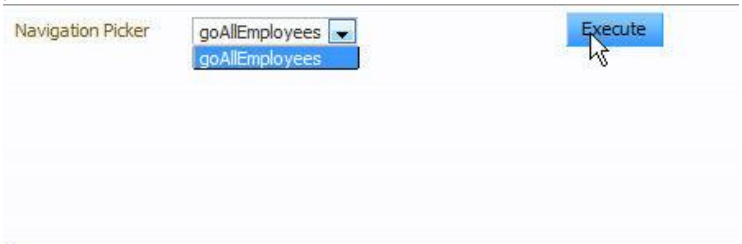
The following code is from the downloadable sample and reads the control flow cases for a view into a list of `SelectItem` used in an af:selectOneChoice

```
//read the navigation case list from the region capabilities
public ArrayList<SelectItem> getNavlist() {
   RichRegion region = this.findRegionById("adfRegion1");
```

```
    Set<String> capabilities =
       region.getRegionModel().getCapabilities();
    navlist = new ArrayList<SelectItem>();

    for(String navcase : capabilities){
      SelectItem item = new SelectItem();
      item.setLabel(navcase);
      item.setValue(navcase);
      navlist.add(item);
    }

    return navlist;
}
```



The select one choice in component in the sample always shows the control flow cases that are defined for the current view in the ADF Region (just for fun, you may want to add some wild card navigation flows in the task flow and then re-run the application to see that the control flow cases are dynamically looked up).

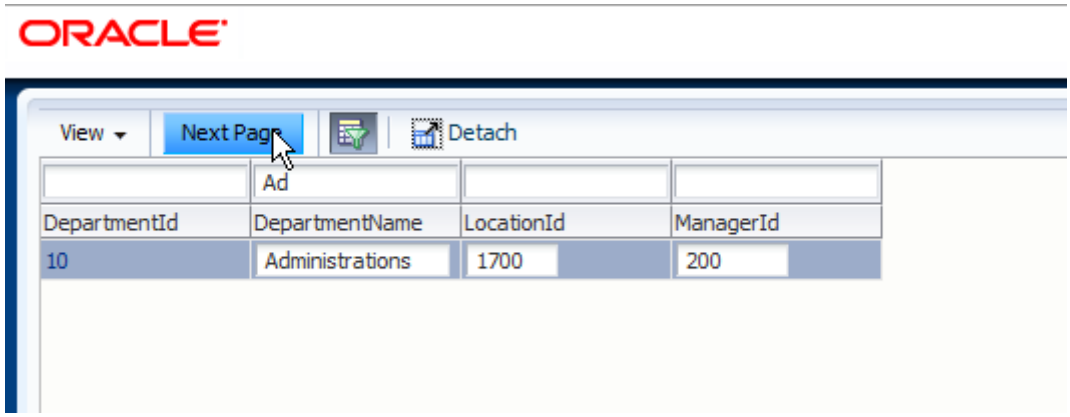
Get the sample application from here and make sure you configure the database connect to point to the HR schema of your local Oracle database.

[Download Oracle JDeveloper 11.1.1.6 sample]

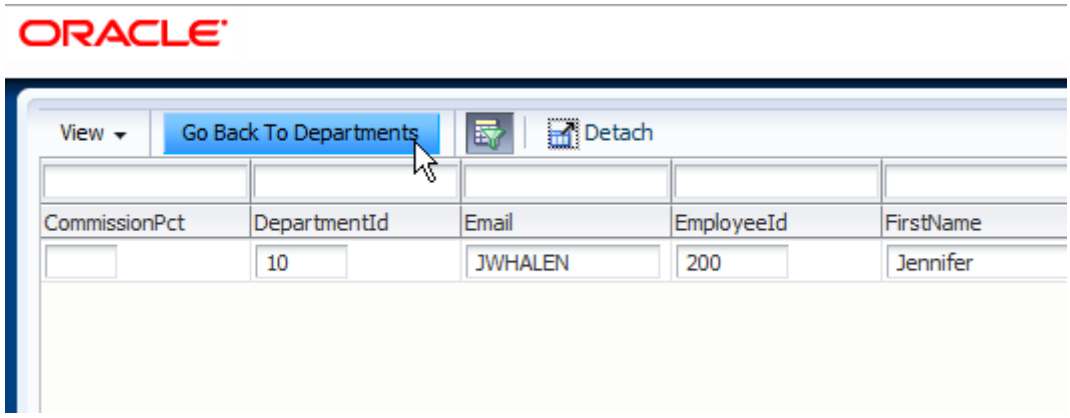
See also: "Reference : Initiate Control Flow Within A Region From Its Parent Page Functional Pattern" http://www.oracle.com/technetwork/developer-tools/adf/queueactioneventinregion-155252.html

Clear Table Filter upon Navigation

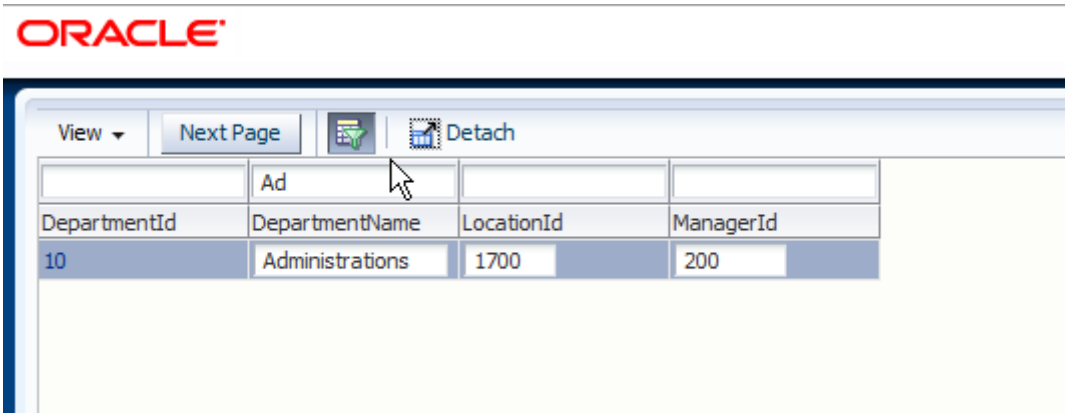
The following behavior can be observed in Oracle ADF for dependent view objects displayed on different ADF Faces views.



A user filters the result set displayed in an ADF bound table using the table filter feature. In the screen shot above, the table shows a single department, which also is marked as the current. To view and work on the dependent detail collection for the selected row, the application user presses a navigation button.



As expected, the dependent collection shows employees for the selected department. The screen shot also shows a command button for the user to press for navigating back to the previous page, which …



Still shows the parent table as it was left, including the table filter settings. Good if this is the way you want it. However, what if you want the table and filter to be reset upon back navigation.

To implement the reset, there are two things that need to be done independently from each other

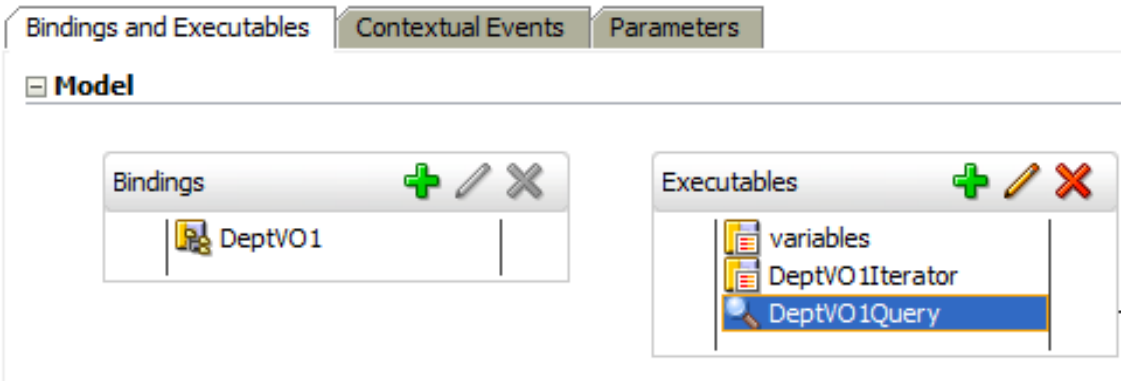
- The table filter needs to be cleared
- The filtered table needs to be reset

To start with the first task, clearing the table filter can be handled easily in a managed bean referenced from the "Next Page" button action property. For this, the table **Binding** property too needs to reference the managed bean so that the code has access to the `RichTable` instance.

```
public String onGoEmp() {
  //access the Rich Tableinstance created for this managed bean
  //through the table Bindings property
  RichTable table = this.getDepartmentsTable();
  //get the table filter model
  FilterableQueryDescriptor filterDescriptor =
          (FilterableQueryDescriptor)table.getFilterModel();
  //sweap the filter
  filterDescriptor.getFilterCriteria().clear();
  //perform navigation to the detail page
  return "goEmp";
}
```

The code above does take care for the table filter to no longer show the search criteria the user entered. The next task is to undo the filter, which can be done declaratively after creating and exposing a client method on the View Object.

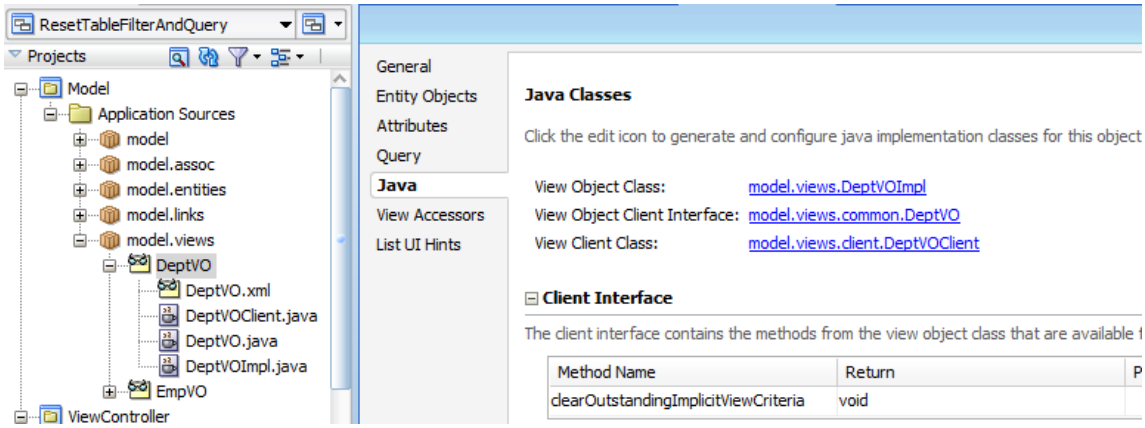
When you create an ADF bound table with the filter feature enabled, Oracle JDeveloper creates a search binding in the executable section of the PageDef file.



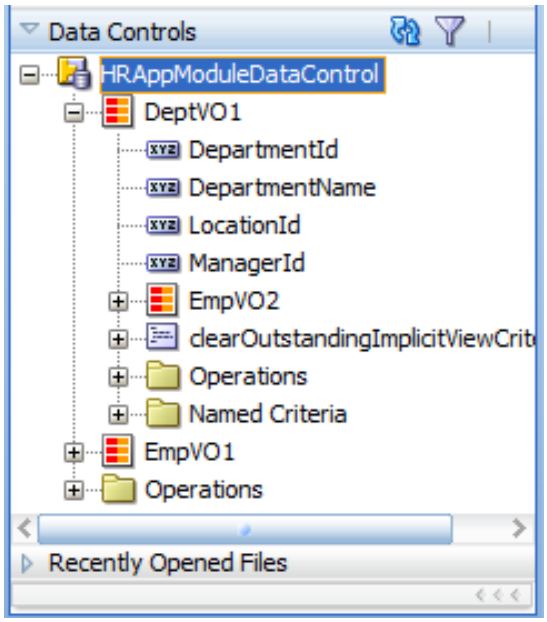
The search binding is referenced by the query listener that is automatically configured in the ADF Faces table properties to implicitly apply a default view criteria to the underlying collection. This view criteria needs to be removed for the table before re-executing the query.

As the view criteria is added to the query (and thus the View Object), it makes sense to expose the client method to unset the default view criteria on the View Object impl class and not the Application Module. In the sample used for this tip, the DeptVOImpl class is built and used to expose the method shown below.

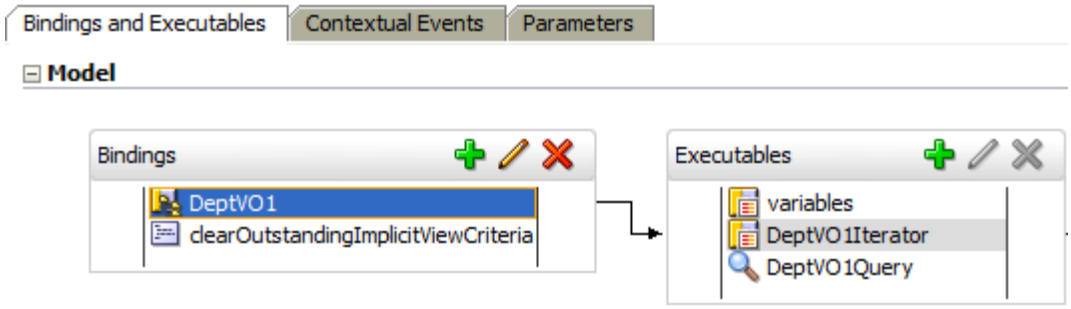
```
public void clearOutstandingImplicitViewCriteria() {
  // we only want to remove the stuff that was added though the table
  //filter (or a default search form)
  // "__ImplicitViewCriteria__" is the magic name for this VC
  ViewCriteria vcDefault =  this.getViewCriteria(
              ViewCriteriaManager.IMPLICIT_VIEW_CRITERIA_NAME);
  if (vcDefault != null) {
    //Clear the stored values
    vcDefault.clear();
    //And refresh the collection
    this.executeQuery();
  }
}
```



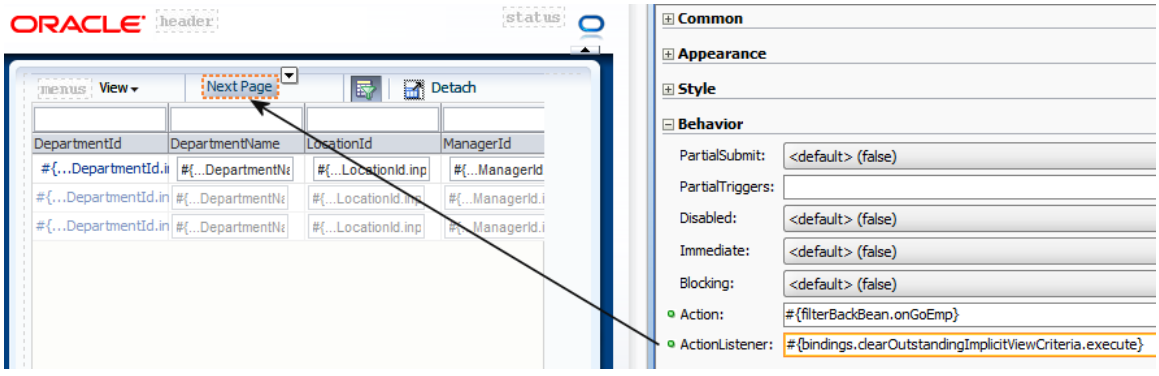
Once the method is exposed on the ADF Business Components client interface from where it displays in the ADF Data Controls panel.



You can now drag the method from the Data Control panel to the "Next Button" command button so it is referenced in its **ActionListener** property. The PageDef file is updated as shown below.



The command button is updated as shown below so that upon navigation the filtered query is always reset



Note that the code doesn't need to be called on navigation to the detail page but could also be called from a command button on the page so that after PPR of the table, the filter is reset.

How-to invoke a method once upon application start

A requirement on the OTN forum was to execute a method only once upon application start either for the application as a whole (all user instances) or once per application user session. In addition, the method to be executed was exposed as an Operation binding on the ADF binding layer.

One way to provide a solution to this requirement is to within the combination of a phase listener on the JSPX of JSF document level and a managed bean in application or session scope (dependent on whether the method should be executed once per application start or once per application user session).

The phase listener can be configured on a JSF document, as mentioned or in the `faces-config.xml` file if there is no single entry to an application. For this example, we assume a single point of entry so that the phase listener can be configured on the f:view attribute.

<f:view beforePhase="#{ManagedBean.onBeforePhase}">

The event logic is configured in a bean in request scope so it could also hold component references for the page if required.

```
public void onBeforePhase(PhaseEvent phaseEvent) {
 //render response is called on an initial page request
 if(phaseEvent.getPhaseId() == PhaseId.RENDER_RESPONSE){
  FacesContext fctx = FacesContext.getCurrentInstance();
  ELContext elctx = fctx.getELContext();
  ExpressionFactory exprFactory =
             fctx.getApplication().getExpressionFactory();
 //call the managed bean in application or session scope. If the
 //bean instance already exists, then no new instance of it will be
 //created,in which case the "initial load" method is not executed
 ValueExpression ve = exprFactory.createValueExpression(
     elctx,
     "#{your_managed_bean_in_application_or-session_scope",
     Object.class);
     ve.getValue(elctx);
 }
}
```

The idea for this phase listener is to reference a managed bean in application scope or session scope base on your requirement. The managed bean in session or application scope invokes the method you want to invoke once per application or user session in its post construct method

```
//Managed Bean in application scope
import javax.annotation.PostConstruct;
...

@PostConstruct
 public void methodInvokeOncedOnPageLoad(){
   //access the methods you want to invoke. If they are exposed in the
   //PageDef file, access the BindingContext --> BindingContainer -->
   //OperationBinding. Alternatively you can call BindingContext -->
   //findDataControl("Name as in DataBindings.cpx") -->
   //getApplicationModule -->  findViewObject/Execute methods
 }
```

Note that the Java EE @PostConstruct bean is called once for each bean instantiation. In the managed bean case, the bean is instantiated once per application or session and so is the method executed once.

Add or remove custom operators in af:query

Under the same header in the Oracle® Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework product documentation you find information about hiding operators from a View Criteria so they don't show in a query component

http://docs.oracle.com/cd/E14571_01/web.1111/b31974/web_search_bc.htm#ADFFD2457

With the information provided, you can create custom operators for each view criteria item by adding code to the view object XML file or remove default operator, for example

```
<ViewCriteriaRow
    Name="vcrow0"
    UpperColumns="1">
  <ViewCriteriaItem
    Name="JobId"
    ViewAttribute="JobId"
    …
    Required="Optional">
   <CompOper
      Name="LessThan"
      ToDo="-1"
      Oper=" <= ">
   </CompOper>
  </ViewCriteriaItem>
 …
</ViewCriteriaRow>
```

A complete list of **Oper** property values is available from here:

http://docs.oracle.com/cd/E15051_01/apirefs.1111/e10653/constant-values.html

Hint: Search for the **oracle.jbo.common.JboCompOper** header

Jang Vijay Singh who originally posted the question on the OTN forum wrote a comprehensive blog summary about this topic, which I recommend you to read from here:

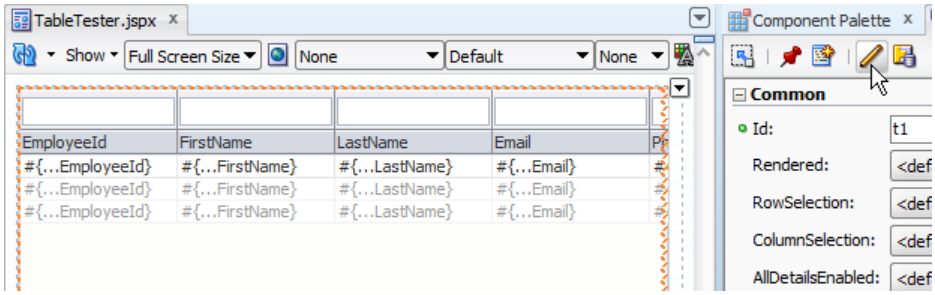
http://weblog.singhpora.com/2012/04/how-to-showhide-operators-in-adfquery.html

Associating a table with a filter and custom View Criteria

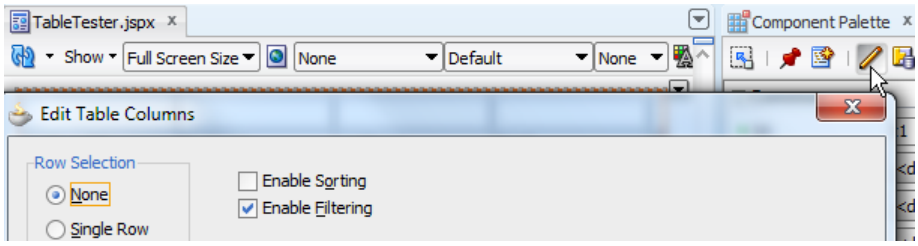
Often developers initially create ADF table without the table filter option. To implement table filters later, follow the steps outlined below

1. In the visual editor, select the table and open the Property Inspector (make sure the table and not a column is selected.

2. In the property inspector header, press the "pencil icon" (tooltip says "Edit Component Definition")



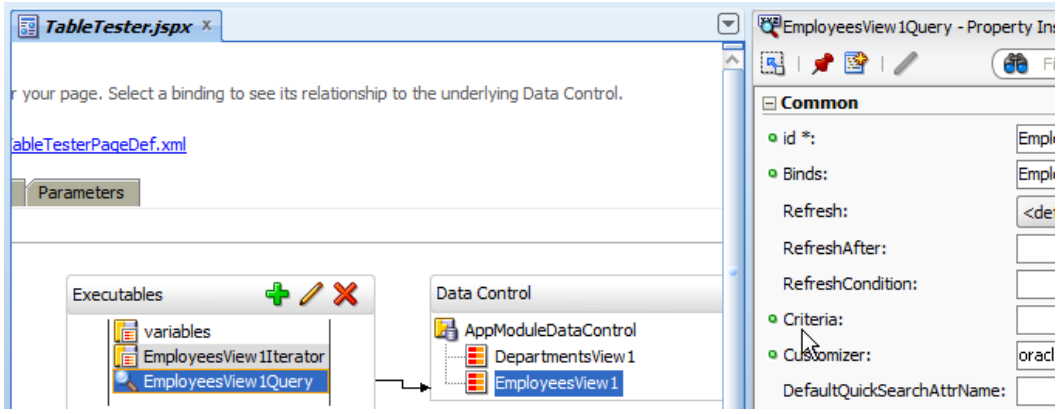
3. In the opened dialog, enable the filter option



4. Press ok

This configures the table with a filter header and default view criteria condition. To filter the table with custom view criteria defined on the underlying View Object.

5. Select the "Bindings" tab
6. In the Executables section, select the search binding (magnifier icon)
7. In the Property Inspector set the "Criteria" property to the name of a named where clause defined on the View Object



ADF Code Corner

OTN Harvest Spotlight

- Donovan Sherriffs

"Development should not be difficult"
-Donovan Sherriffs

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

Twitter:

ACC: What is your current role?

DS: I am currently a ADF consultant working at ITQ in South Africa specializing in Oracle Forms modernization with a side serving of service integration. My responsibilities include Oracle ADF and Java EE development, team leadership, architecture and training.

ACC: What is your IT background?

DS: After obtaining my Information Technology qualification I started off (back in the 90's) developing web based C++ and ASP internet/ intranet sites.

After a brief flirtation with these technologies I move into the Java space where I stayed for a good number of years (working with most databases, application servers, frameworks and buzz words along the way) from Java 1.1 until today.

I have had the privilege of working on all types of systems in various industries from large scale integration projects in the public sector, a BI solution for a small corporate, a high volume transactional equities settlement system, client focused online banking system to name a few.

The last few years have been working with ADF mostly kicking off Oracle Forms modernization projects.

I have worked as a developer, project manager, and architect and have played most roles in the software development life cycle. But my passion is for solving complex technical problems and improving developer productivity through the use of frameworks, standards and patterns.

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

DS: JDeveloper is the standard IDE for all development done in our environment (Java EE, ADF and SOA suite). It is what I live and breathe on a day to day basis. My current focus is rewriting an internal Oracle Forms system of over a thousand forms (ADF using Business Components).

ACC:	So far, what has been your biggest challenge in building Java EE application with Oracle ADF?
DS:	I think this really a matter of understanding best practice. When first approaching an ADF implementation we made a few fundamental mistakes on our first try. Also overcoming the learning curve for Java developers can be steep but steeper for Oracle Forms developers. With a framework of this magnitude you need to know how to use it to best suite you.
ACC:	Which feature of ADF was the greatest benefit to your project?
DS:	Definitely the ease of development features and rich set of UI components with PPR.
ACC:	Away from the on line help, what have been your most valuable sources of ADF knowledge?
DS:	I have to give big round of applause to the community! They are very active and very helpful. The OTN forum and blogs out there are filled with helpful hints and tips.
ACC:	By your experience and as estimation, how long does it take for new developer to learn ADF?
DS:	Depends on the developer. Really, it's something that takes a short amount of time to be productive and a long time to master. So we have had developers who embrace the technology with lots of JSF experience become very productive in a month.
ACC:	What do you recommend as a starting point and path to learn Oracle ADF?
DS:	If you have no Java experience I suggest a Java and Java EE set of tutorials. For an individual with Java background doing some of the online tutorials really helps. After that I would say "build something throw it away then build it better (there is no replacement for experience with these tools)". If you are embarking on a large scale project get expert advice early.
ACC:	What are your top 3 ADF best-practices for ADF developers?
DS:	1. Use a proxy for all service calls 2. Work with the framework, things work better if you use what ADF gives you in the way it was intended 3. Consider clustering, multiple languages, error handling and plumbing related issues early in the process
ACC:	Name 3 ADF anti patterns or gotchas to avoid?
DS:	1. ADF is a vast framework there is always a temptation to use everything it gives you, try to keep it simple. 2. Repeating yourself: Use templates and declarative components to avoid doing the same thing over and over etc. 3. God-like objects in the usage of application module classes

ACC:	How do you see the market for ADF developing for the local area you live and work in?
DS:	ADF has been slow starter in South Africa, but I have recently seen a bit more momentum in the space. South Africa is a price sensitive environment and most of the Java work is in open source technologies but there is a lot of interest in the productivity gains ADF provides. So I think ADF could make some good headway if it is seen as a Java development platform rather than something you rewrite Oracle Forms in or use with SOA suite etc. Also I have noticed a reluctance in Java developers to use ADF as a tool for developing Java based systems. My opinion is that ADF allows you to focus on solving the exciting technical issues instead of mundane tasks by giving you significant set of productivity tools. Once you have mastered ADF it will be a hard task to drag yourself back to building with any other framework. Give it a try.
ACC:	ADF Genie grants you a wish, what would you ask for?
DS:	This may be controversial but I wish for a more open and lightweight ADF environment. This would include • rich Maven support • list-of-values based on Pojo's or services, enhancing the Pojo data control • allowing development on multiple IDEs . In this way the ADF framework based approach could promulgate through the Java development community and allow other developers to contribute and enjoy the benefits of building application rapidly. This could pave the way forward for ADF becoming the de facto standard for Java development.
ACC:	Thank you Donovan

RELATED DOCOMENTATION

☒	
☒	
☒	