ADF Code Corner Oracle JDeveloper OTN Harvest 12 / 2010



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-DEC-2010

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: <u>http://forums.oracle.com/forums/forum.jspa?forumID=83</u>

Table of Content

OTN Harvest – The Blog
How-to find out about ADF application deployment with ojdeploy3
Grid Layouts in ADF Faces using Trinidad
Panel Collection Confusion4
Reading the selected value of an ADF bound Select List in Java5
Formatting the af:inputSpinNumber tick labels6
Beginner Mistake: Adding a String to a value property6
How-to create dependent model-driven LOV7
How-to tell the ViewCriteria a user chose in an af:query component 15
How-to restrict file upload sizes in ADF Faces
Map Viewer doesn't show maps for large parts of the globe19
How-to control user input based on RegEx pattern19
Get social security numbers right21
How-to call server side Java from JavaScript23
How to expose an ADF application in a Portlet?24
How-to query af:quickQuery on page load ?24
How-to create a select one choice list of common time zones25
How-to hide the close icon for task flows opened in dialogs26
How-to populate different select list content per table row27

OTN Harvest - The Blog

The content of this OTN Oracle JDeveloper Harvest summary is also available as a blog:

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

The Oracle "ADF Code Corner Oracle JDeveloper OTN Harvest" blog publishes information as close as possible to their related question on the OTN forum. An extended version of the blog entries, containing screen shots and images, is provided in this OTN Harvest Summary document.

How-to find out about ADF application deployment with ojdeploy

Using the *ojdeploy*, applications or modules can be deployed from a command line window or ANT. To learn about how to use this utility, use the Oracle JDeveloper help

1 - In the search field you see in Oracle JDeveloper (right upper corner), type ojdeploy





Grid Layouts in ADF Faces using Trinidad

ADF Faces does provide a data table component but none to define grid layouts. Grids are common in web design and developers often try HTML table markup wrapped in an fiverbatim tag or directly added the page to build a desired layout. Usually these attempts fail, showing unpredictable results,

However, ADF Faces does not provide a table layout component, but Apache MyFaces Trinidad does. The Trinidad trh:tableLayout component is a thin wrapper around the HTML table element and contains a series of row layout elements, trh:rowLayout. Each trh:rowLayout component may contain one or many trh:cellLayout components to format cells content.

```
<trh:tableLayout id="tl1" halign="left">
<trh:rowLayout id="rl1" valign="top" halign="left">
<trh:rowLayout id="rl1" valign="top" halign="left">
<trh:cellFormat id="cf1" width="100" header="true">
<af:outputLabel value="Label 1" id="ol1"/>
</trh:cellFormat>
<trh:cellFormat id="cf2" header="true"
```

```
width="300">
<af:outputLabel value="Label 2" id="outputLabel1"/>
</trh:cellFormat>
</trh:rowLayout>
<trh:rowLayout id="rowLayout1" valign="top" halign="left">
<trh:rowLayout id="rowLayout1" valign="top" halign="left">
<trh:cellFormat id="cellFormat1" width="100" header="false">
<af:outputLabel value="Label 3" id="outputLabel2"/>
</trh:cellFormat>
</trh:rowLayout>
...
```

</trh:tableLayout>

To add the Trinidad tag library to your ADF Faces projects ...

- · Open the Component Palette and right mouse click into it
- Choose "Edit Tag Libraries" and select the Trinidad components. Move them to the "Selected Libraries" section and Ok the dialog.
- The first time you drag a Trinidad component to a page, the web.xml file is updated with the required filters

Note: The Trinidad tags don't participate in the ADF Faces RC geometry management. However, they are JSF components that are part of the JSF request lifecycle.

ADF Faces RC components work well with Trinidad layout components that don't use PPR. The PPR implementation of Trinidad is different from the one in ADF Faces. However, when you mix ADF Faces components with Trinidad components, avoid Trinidad components that have integrated PPR behavior. Only use passive Trinidad components.

Links:

http://myfaces.apache.org/trinidad/trinidad-api/tagdoc/trh_tableLayout.html

http://myfaces.apache.org/trinidad/trinidad-api/tagdoc/trh_rowLayout.html

http://myfaces.apache.org/trinidad/trinidad-api/tagdoc/trh_cellFormat.html

Panel Collection Confusion

A command button added to the toolbar of a Panel Collection component does not cause field validation in a form when pressed. While this appears confusing it works as designed.

Instead of a full page re-rendering, ADF Faces events and components can trigger partial page refresh, in which only portions of a page are refresh upon a request. In addition, some components – including the af:popup and af:subForm - represent event roots. Event roots don't propagated event notification outside of the component tag boundary, which means that the ADF Faces lifecycle only executed on components that are children of the event root component. The PanelCollection component is an event root and therefore only validates and refreshes data of its child components.

Reading the selected value of an ADF bound Select List in Java

Model driven and dynamic select lists are bound to the JUCtrlListBinding in the associated binding container. To read the user selected list value in a managed bean, the list component **AutoSubmit** property must be set to "true". The **ValueCangeListener** property then references a managed bean method to read the selected value.

🚯 🔹 Show 🕶 Full S	Screen Size 🔻 🔕 None 🔹 D	efaul	it 🔻 None	- • 🖪 🗞 🖉 B 🖌 🖳 🗄 🏛 🕾 🗉 •
Departmentid		<u> </u>		
Departmentid	#{DepartmentId.inputValue}		😋 Select One Choic	e - #{bindings.DepartmentId.label} - Pro
Employeeld	#{Employeeid.inputValue}		🔍 I 🕜 📝 🖉	品 (論 Find · · · · · · · · · · · · · · · · · · ·
FirstName	#{FirstName.inputValue}		Common	
LastName	#{LastName.inputValue}		Common	
Email	#{Email.inputValue}		Appearance	
PhoneNumber	#{PhoneNumber.inputValue}		🗄 Style	
HireDate	#{HireDate.inputValue}		🗆 Behavior	
Jobid	#{Jobld.inputValue}		DRequired:	#{bindings DepartmentId bints mandatory}
Salary	#{Salary.inputValue}		ReadOnlys	
CommissionPct	#{CommissionPct.inputValue}		ReadOnly:	<pre><derault>(raise)</derault></pre>
Managerid	#{Managerld.inputValue}		Disabled:	<default> (false)</default>
			AutoSubmit:	true 🗸 👻
			PartialTriggers:	~ ~
			RefreshCondition:	¥
3			···· Validation ······	
			Immediate:	<default> (false)</default>
			Converter:	<default></default>
			ValuePassThru:	<default> (false)</default>
			Validator:	~
			ValueChangeListene	er: #{EmployeesPage.onEmployeesListChange} ~
1		1	··· Contextual Events	

At design time, the select list is defined in the PageDef file of the page or page fragment containing the list component. The name of the list binding is referenced in the managed bean method shown below.

EditVacationRequestsPageDef.x	nl - Structure	BindingContext bo BindingContainer
1	Contract DepartmentId - Property Inspector	
EditVacationRequestsPageD	强 🏓 🗟 🥒 More - 🍘 Find	↓ ♪ ?
executables findings findings findings	id: DepartmentId	~
EmployeeId	 Advanced 	
arii arii arii arii arii arii arii arii		
B Solid Salary		
🖶 🐨 🔤 ManagerId		

The managed bean method shown below accesses

- the selected list index
- the selected list value

- the selected list row

Formatting the af:inputSpinNumber tick labels



To change the display format of the Input Spin Number tick labels from

,add an af:convertNumber tag as shown below

<af:convertNumber integerOnly="true" minIntegerDigits="2"/>

Beginner Mistake: Adding a String to a value property

A common beginner mistake, e.g. when working with the af:selectOneChoice component or the af:inputText component, is to provide a static value to the component value property. For example:

```
<af:selectOneChoice id="soc1" value="Value One" autoSubmit="true">
<af:selectItem label="..." value="..." />
```

```
ADF CODE CORNER
```

```
</af:selectOneChoice>
```

Or

```
<af:inputText id="it1" value="Hello World"/>
```

Assigning a non-updateable value to an UI input component however renders this component read-only, which come by surprise for developers that are new to JavaServer Faces.

If you experience a problem like this, keep in mind that JSF components expect a value expression to be provided in its value property. If you want to define a default value for a component, define a value for the JavaBean variable which setter/getter method is referenced in the JSF component value property.

How-to create dependent model-driven LOV

Dependent list-of-values can be based on View Objects that have a parent-detail relationship or independent View Objects, as explained in the following.

In the example, a View Object "VacationrequestsView" has two attributes, "DepartmentId" and "EmployeeId" that reference values of the "DepartmentsView" and the "EmployeesView" objects.



To build a dependent model driven list-of-value, open the "VacationrequestsView" view object in the View Object editor. Select the "DepartmentId" attribute and click the green plus icon next to the **List of Values: DepartmentId** header.

Name	Type	Alias Name	Entity Usage	Info	
🖙 VacrequestId	oracle.jb	VACREQUEST_ID	Vacationrequests		
DepartmentId	Number	DEPARTMENT_ID	Vacationrequests		
EmployeeId	Number	EMPLOYEE_ID	Vacationrequests		
FromDate	Date	FROM_DATE	Vacationrequests		
FoDate	Date	TO_DATE	Vacationrequests		
Approved	String	APPROVED	Vacationrequests		
		-			
	es: Department	10			.
🕀 List of Values: De	partmentId				4

In the opened **Create List of Values''** editor, press the green plus icon at the end of the **List Data Source** field.

😔 Create List of Values 🛛 💽
List of Values Name: LOV_DepartmentId
Configuration UI Hints
Select a view accessor for the list data source, and then choose the list attribute that maps to the current view object attribute.
List Data Source: None Specified>
List Attribute:
List Return Values

Select the "DepartmentView" View Object and press the first arrow icon to create a View Accessor instance of it.

🕹 View Accessors	—
Select a view object or shared view inst create a view accessor.	ance and shuttle it to the selected list to
Available View Objects:	View Accessors: Edit
adf.sample.model.Model df.sample.model.vo df.sample.model.vo DepartmentsView 편nloyeesView VacationrequestsView	VacationrequestsView PepartmentsView 1
Na <u>m</u> e: DepartmentsView2	Acce <u>s</u> sor Name: DepartmentsView1 Definition: adf.sample.model.v
Help	OK Cancel

Press OK to close the dialog. In the Create List of Values dialog, select DepartmentId as the List Attribute.

ist of Values <u>N</u> ame:	LOV_DepartmentId	
Select a view acce object attribute.	essor for the list data source, and the	n choose the list a
List <u>D</u> ata Source:	DepartmentsView 1	
List <u>A</u> ttribute:	DepartmentId	
List Return Values		
	antal values that your list returns to th	a haas view ahis
Map any suppleme attribute for which	n the list is defined).	ie base view obje
Map any suppleme attribute for which	n the list is defined).	ie base view obje
Map any suppleme attribute for which	the list is defined).	List Attribute

Select the **UI Hints** tab to define the list as a list-of-value. Choose **Input Text with List of Values** as the **Default List Type**.

ightarrow Create List of Values				
List of Values <u>N</u> ame: LOV_DepartmentId				
Configuration UI Hints				
Defaul <u>t</u> List Type: Input Text with List of Values				
Display Attributes				
Select display attributes for the list of values an (multiple values are separated by white space).	d combo box. Optionally show			
A <u>v</u> ailable:	Selected:			
LocationId	DepartmentId			
ManagerId	DepartmentName			

Select the attributes that should be shown in the LOV result table.

Next, double click the "EmployeesView" view object that provides the dependent list values. In the View Object editor, click the green plus icon next to the **Bind Variables** header.

🚰 EmployeesVie	ew.xml
General Entity Objects Attributes Query Java View Accessors List UI Hints	Query Data for this view object will be retrieved from the datasource using the following SQL query. SELECT Employees.EMPLOYEE_ID, Employees.FIRST_NAME, Employees.LAST_NAME, Employees.EMAIL, Employees.PHONE_NUMBER, Employees.HIRE_DATE, Employees.JOB_ID, Employees.SALARY, Employees.COMMISSION_PCT, Employees.DEPARTMENT_ID FROM EMPLOYEES Employees
	Bind Variables Named bind variables can be used in the SQL query of this view object.

In the **Bind Variable** editor, create a bind variable with the data type of the data passed in from the first LOV selection. In this example, the data type is oracle.jbo.domain.Number. The bind variable should be configured as **updatable** and **not required**. Specifying the bind variable as **not required** allows you to run the View Object also when the bind variable does not have a value. This allows the variable to be used in a View Criteria, which is what is used in this sample to filter the LOV query based on the user selection in the first LOV.

ADF CODE CORNER

🎃 Bind Variable	:		×
Variable C	ustom Properties Control Hints		
<u>N</u> ame:	deptId		
<u>Type</u> :	Number	-	Browse
Value Type:	Literal <u>Expression</u>		Test
<u>V</u> alue:			Edit
	✓ Updatable	<u>R</u> equired	
Help		ок 💦 🗌	Cancel

OK the dialog. After creating the bind variable, press the green plus icon next to the **View Criteria** header of the "EmployeesView" View Object editor.

		+ 4	🖉 🗙 Override
n be used in the SQL query of	this view object.		
J			
Туре	Value	Info	
Number			
for the second			<u></u>
	n be used in the SQL query of Type Number	n be used in the SQL query of this view object. Type Value Number	n be used in the SQL query of this view object. Type Value Info Number

In the **Edit View Criteria** editor, provide a meaningful name for the new criteria and press the **Add Item** button. Select **DepatrtmentId** as the attribute to apply the named where clause (the view criteria) to.

Choose **Bind Variable** as the **Operand**, and select the bind variable created earlier. This step defines a where clause that filters employee data to only match employees of a department Id defined by the bind variable.

ADF CODE CORNER

OTN ORACLE JDEVELOPER FORUM HARVEST 12 / 2010

늴 Edit View C	Criteria		X
<u>C</u> riteria Nam	e: EmployeesByDeptIdViewCriteria		Query Execution Mode: Database 🔻
Criteria Defi	inition UI Hints		
1			
View Criteria	:	Vie	e <u>w</u> Object Where Clause:
🛃 Employe	esByDeptIdViewCriteria	((Employees.DEPARTMENT_ID =
(≟() Grou	μp DepartmentId = :deptId		deptīd))
Add Item	Add Group Add Criteria Add Named Crit	eria Delete	Explain Plan
- Criteria Ite			
Conjunction	AND	✓ Ignore	Case
Attri <u>b</u> ute:	DepartmentId 🔹	🔲 I <u>qn</u> ore	Null Values
Operator:	Equal to 👻	Validation:	Optional 👻
Operand:	Bind Variable 🔹		
<u>P</u> aramet	er: deptid 🔹 🗣		
Help			OK Cancel

EmployeesView.xml Section requests View.xml General Entity Objects Attributes Override Set Source Or Attributes View object attributes can be mapped to entity attributes, calculated or SQL-derived. Query 41) 👸 Name Java 🕂 • 🥖 View Accessors List UI Hints Name Type Alias Name Entity Usage Info 🖙 VacrequestId orade.jb... VACREQUEST_ID Vacationrequests DEPARTMENT_ID DepartmentId Number Vacationrequests mployeeId Number EMPLOYEE_ID Vacationrequests FromDate Date FROM_DATE Vacationrequests ToDate Date TO_DATE Vacationrequests APPROVED Approved String Vacationrequests +• / + Custom Properties: EmployeeId List of Values: EmployeeId Enable this attribute to display a list of values to use in the user interface.

Still in the Edit View Criteriadialog, uncheck the Ignore Null Valueschckbox before pressing OK.

Open the View Object editor for the "VacationsrequestsView" object and select the **EmployeeId** attribute. Press the plus icon next to the **List of Values: EmployeeId** header.

ſ	💩 Create List of Val	ues
	List of Values <u>N</u> ame:	LOV_EmployeeId
	Configuration	JI Hints
	Select a view acce object attribute.	ssor for the list data source, and then choose the list attribute that maps to the current view
	List <u>D</u> ata Source:	<none specified=""></none>
	List <u>A</u> ttribute:	· · · · · · · · · · · · · · · · · · ·
Т		

In the opened **Create List of Values** dialog, press the green plus icon next to the **List Data Source** field. In the opened **View Accessors** dialog, select the "EmployeesView" entry and press the first shuttle button.

👌 View Accessors	•••
Select a view object or shared view inst create a view accessor.	ance and shuttle it to the selected list to
Available View Objects:	View Accessors: Edit
adf.sample.model.Model df.sample.model.vo df.sample.model.vo SepartmentsView SepartmentsView VacationrequestsView	VacationrequestsView DepartmentsView1
Name: EmployeesView2	Accessor Name: EmployeesView1
Help	Definition: adf.sample.model.v OK Cancel

Before closing the dialog, press the Edit button next to the View Accessors label.

🕹 Edit View Accessor: Employees\	/iew1		X
🍘 Search	View Object		
View Object	Configure the view object View Definition: adf.sar View Criteria Select the view criteria to combined with an AND of Available: Bind Parameter Values - Provide values for any b	equery for this accessor. mple.model.vo.EmployeesView that you want to apply to this view object. If you apperator. Selected: S	u select multiple view criteria, they will be oyeesByDeptIdViewCriteria
	Parameter	Туре	Value
	deptId	oracle.jbo.domain.Number	DepartmentId

In the **Edit View Accessor: EmployeesView1** editor, select the View criteria name in the **Available** list field and shuttle it to the **Selected** list. This applies the View criteria to the EmployeesView LOV query. Point the View Criteria bind variable – **deptId** in the example – to the attribute in the View Object that holds the value for the first LOV selection.

🎐 Create List of Values					
List of Values Name: LOV_EmployeeId					
Configuration UI Hints					
Select a view acce object attribute.	ssor for the list data source, and ther	n choose the list attrik			
List <u>D</u> ata Source:	EmployeesView 1				
List <u>A</u> ttribute:	EmployeeId				
List Return Values					
Map any supplemental values that your list returns to the base view object (i attribute for which the list is defined).					
+ ×					
View Attribute		List Attribute			
EmployeeId		EmployeeId			

Select **EmployeeId** as the list attribute that updates the "VacationsrequestView" attribute and press the **UI Hints** tab to define the list as a list-of-value at runtime.

After this, you can run the ADF Business Components tester from the Application Module context menu to try the dependent list-of-values.

Oracle Business Compone File View Create Database	ent Browser (AppModuleLocal-Local) Help			E C		
AppModule	I ✓ > I	ê ✓ 🦻		erride		
	VacrequestId DepartmentId	100 50	LOV_Employ	veeld Image: Constraint of the second seco	LastName Weiss	EmployeeId 120
	Employeeld FromDate ToDate Approved	119 2010-12-14 2010-12-17 N	50 50 50 50 50	Adam Payam Shanta Kevin	Fripp Kaufling Vollman Mourgos	121 122 123 124
			50 50 50 50 50	Julia Irene James Steven	Nayer Mikkilineni Landry Markle Bissot	125 126 127 128 129
Name:AppModule.Vacationreque	estsView1 Definition:adf.sample.model.vo.Va	acationrequestsView	50 FO Help	Mozhe	Atkinson Morlow	125 130 121 OK Cancel

Id the dependent list-of-values work in the ADF Business Components tester, create an ADF Faces page and drag the **VacationsrequestsView** instance as an input form.

OTN ORACLE JDEVELOPER FORUM HARVEST 12 / 2010

ADF CODE CORNER



Because of the model side settings, the DepartmentId and EmployeeId attributes are represented by LOV components.

Edit Form Fields Confermentiation			×
components after you click	COK. You can also add more componer	its directly to the layout later.	
Fields:		🕂 🗙	
Display Label	Value Binding	Component To Use	
💴 <default></default>	 VacrequestId 	🗭 ADF Input Text w/ Label	
🚥 <default></default>	📼 DepartmentId	ADF List of Values Input	
🚾 <default></default>	📼 EmployeeId	ADF List of Values Input	
🚾 <default></default>	FromDate	ADF Input Date w/ Label	
📧 <default></default>	📼 ToDate	ADF Input Date w/ Label	_
🚥 <default></default>	Approved	🛱 ADF Input Text w/ Label	Ŷ
			Û
			л

Select the first list-of-value component – DepartmentId in the sample – and open the Property Inspector. Set the **AutoSubmit** property to **true**.

VacrequestId	#{VacrequestId.inputValue}	😋 Input List Of Valu	ues - #{bindings.DepartmentId.hints.label	
DepartmentId	#{DepartmentId.inputValue}	🖳 I 🏓 📴 I 🥒	🛃 🍘 auto 🕹 🗘	•) 🧿
Employeeld	#{Employeeld.inputValue}	Cara loaci	sucrours (input)	
FromDate	#{FromDate.inputValue}	Disabled:	<default> (false)</default>	~
ToDate	#{ToDate.inputValue}	AutoSubmit:	<default> (false)</default>	~
Approved	#{Approved.inputValue}	AutoTab:	<default> (false)</default>	~
First F	Previous Next Last	PartialTriggers:	false true	~
Submit		RefreshCondition:	V	~
		Validation		

Then select the second list-of-values – EmployeeId in the sample – and point its **PartialTriggers** property to the Id of the first-list-of value component. For this, you can use the **Edit** option of the **PartialTriggers** context menu.

OTN ORACLE JDEVELOPER FORUM HARVEST 12 / 2010

1	Vacrequestid	#{VacrequestId.inputValue}		on - #/bindings EmployeeId bints	Ishall - Property	#
Į.	Departmentid	#{DepartmentId.inputValue}		2 🗔 (🚓 past		Common Components
Į.	Employeeld	#{Employeeld.inputValue}				•>• Bread Crumbs
Į.	FromDate	#{FromDate.inputValue}	Disabled:	<default> (false)</default>	• •	Calendar
Į.	ToDate	#{ToDate.inputValue}	AutoSubmit:	<default> (false)</default>	• •	Carousel
ł.	Approved	#{Approved.inputValue}	AutoTab:	<default> (false)</default>		😐 Carousel Item
Į.	First P	revious Next Last	PartialTriggers:			🔠 Choose Color
81	Submit		PefreshCondition		PartialTrig	jgers 🔰
21			Kenesiicondidon	•	Edit	N
Į.			··· Validation ······		Expressio	on Builder
2.1			Constant	1 C 10	- Deset to	Defects

When you run the page, then the dependent list-of-values behave exactly as when using the ADF Business Components tester

VacrequestId	100				
* DepartmentId	50 🔍	Search and Selec	t: EmployeeId	2	\mathbf{X}
* EmployeeId	103	Search		Advanced	
* FromDate	12/14/2010				_
* ToDate	12/17/2010	DepartmentId	FirstName	LastName	
TODAte		50	Matthew	Weiss	
* Approved	N	50	Adam	Fripp	
		50	Payam	Kaufling	
0.1-11		50	Shanta	Vollman	
Submit		50	Kevin	Mourgos	
		50	Julia	Nayer	=
		50	Irene	Mikkilineni	-
		50	James	Landry	
		50	Steven	Markle	
		50	Laura	Bissot	
		50	Mozhe	Atkinson	
		50	James	Marlow	
		50	L	Olson	
		50	Jason	Mallin	
		50	Michael	Rogers	
		50	Ki	Gee	
		50	Hazel	Philtanker	-
		•	III	4	
				OK Cano	el
					_

ADF CODE CORNER

Note: The LOV initial query and search field can be configured on the model side LOV definition.

Note: The Vacationrequests table is not part of the Oracle HR schema. To reproduce the sample documented in this section, you need to first create the table with a foreign key reference to the "DepartmentId" attribute in the "Departments" table and an "EmployeeId" reference in the "Employees" table.

How-to tell the ViewCriteria a user chose in an af:query component

The af:query component defines a search form for application users to enter search conditions for a selected View Criteria. A View Criteria is a named where clauses that you can create declaratively on the ADF Business Component View Object.

A default View Criteria that allows users to search in all attributes exists by default and exposed in the Data Controls panel.

🚰 EmployeesView.xml								
						3 🔨		
General Entity Objects Attributes	Query	ery						
Query		Data for this view object will be retrieved from the datasource using the following SQL query.						
Java View Accessors List UI Hints	SELECT Employees.EMPLOYEE_ID, Employees.IRST_NAME, Employees.LAST_NAME, Employees.SATL, Employees.OBOBID, Employees.SALARY, Employees.SALARY, Employees.COMMISSION_PCT, Employees.DEPARTMENT_ID FROM EMPLOYEES Employees							
	Nar	ne	Type	Value	Info			
	dept	Id	Number					
	nam	eInstr	String			•		
	first	NameInstr	String					
	View Criteria Image: Criteria are named expressions for queries that are used to further refine the results. Name QueryEmployeesByDepartment QueryEmployeesByFirstNameLastName							
Overview Source	History <					>		

To create an ADF Faces search form, expand the View Object node that contains the View Criteria definition in the Data Controls panel. Drag the View Criteria that should be displayed as the default criteria onto the page and choose **Query** in the opened context menu. One of the options within the **Query** option is to create an **ADF Query Panel with Table**, which displays the result set in a table view, which can have additional column filters defined.

Projects			
Application Resources			
▽ Data Controls	N 25		
🖃 🛃 AppModuleDataControl			
🗐 📲 allEmployees			
CommissionPct			
		1	
·····(IIII) Email		3	
EmployeeId			
FirstName			
HireDate			
JobId			
IastName			
······ XXE PhoneNumber		l	
Salary		Create	-
Operations		Query	🗃 ADF Query Panel
🖻 🛅 Named Criteria		<u>T</u> able	💏 ADF Query Panel with Table
All Queriable Attribu	utes		👸 ADF Query Panel with Tree Table 🕏
QueryEmployeesBy	Department	Cancel	
	FirstNameLastName		
⊕ Operations			
		1	

To intercept the user query for modification, or just to know about the selected View Criteria, you override the **QueryListener** property on the af:query component of the af:table component.

OTN ORACLE JDEVELOPER FORUM HARVEST 12 / 2010

Overriding the **QueryListener** on the table makes sense if the table allows users to further filter the result set using column filters.

ADF CODE CORNER

 Show ▼ Full Screet Employees ✓ Search 	en Size 🔻 J 🙆 [None	• •	Default ▼	None V	Saved Search	2]= =		•	ADF Faces
Employees ⊻ Search			No search fields added.	Basic	Saved Search	[-		Common
Search			No search fields added.	Basic	Saved Search	[•		
✓ Search			No search fields added.	Basic	Saved Search	[•		•>• Bread C
			No search fields added.						
			No search fields added.						Button
									📃 Calenda
									Carous
					Searc	h Reset	Save		Carous
l									Choose
					riteria.HireDa	ate} 🔯			E Choose
Employeeld	FirstName	LastName		<u></u>		-	Ъ		Column
#{Employeeld}	#{FirstName}	#{LastNar	Query - Property Inspecto)r			Jobid}		Contex
#{Employeeld}	#{FirstName}	#{LastNar	🔣 📌 🖻 🥒 👪 (nueryListen	er 🤾		lobid}		Dialog
r(cmpioyeeidy	#() institutiof	#1ยอเพลา	ConjunctionReadOnly:	<default> (false</default>	=)	• ~ ^	lional		
			ResultComponentId:	::resId1		~			Go Butt
			PartialTriggers:			~			A Go Ima
			RefreshCondition:						A Go Link
			• QueryListener:	#{bindings.Quer	yEmployeesByFirst				Go Men
			OuervOperationListener:	#{bindinas.Ouer		Quer	yListener		(
						Edit.			
						Meti Rose	nod Explossio + to Default	on Bui	der
			Binding:			Rese	t to Derault		
			ClientComponent:	<default> (fals</default>	e)		ethod referen	ce to a	
			AttributeChangeListener:			Que	erylistener		
			Visible:	<default> (true</default>	:)	-			

To override the default **QueryListener**, copy the existing string referencing the binding layer to the clipboard and then select **Edit** from the field context menu (press the arrow icon to open it) to selecte or create a new managed bean and method to handle the query event.

The code below is from a managed bean with custom query listener handlers defined for the af:query component and the af:table component. The default listener entry copied to the clipboard was "#{bindings.ImplicitViewCriteriaQuery.processQuery}"

```
public void onQueryList(QueryEvent queryEvent) {
    // The generated QueryListener replaced by this method
    //#{bindings.ImplicitViewCriteriaQuery.processQuery}
    QueryDescriptor qdes = queryEvent.getDescriptor();
    //print or log selected View Criteria
    System.out.println("NAME "+qdes.getName());
    //call default Query Event
    invokeQueryEventMethodExpression("
    #{bindings.ImplicitViewCriteriaQuery.processQuery}",queryEvent);
```

}

```
public void onQueryTable(QueryEvent queryEvent) {
  // The generated QueryListener replaced by this method
  //#{bindings.ImplicitViewCriteriaQuery.processQuery}
  QueryDescriptor qdes = queryEvent.getDescriptor();
  //print or log selected View Criteria
  System.out.println("NAME "+qdes.getName());
  invokeQueryEventMethodExpression(
    "#{bindings.ImplicitViewCriteriaQuery.processQuery}",queryEvent);
}
private void invokeQueryEventMethodExpression(
                       String expression, QueryEvent queryEvent) {
  FacesContext fctx = FacesContext.getCurrentInstance();
 ELContext elctx = fctx.getELContext();
  ExpressionFactory efactory
  fctx.getApplication().getExpressionFactory();
 MethodExpression me =
    efactory.createMethodExpression(elctx,expression,
                                    Object.class,
                                    new Class[]{QueryEvent.class});
   me.invoke(elctx, new Object[]{queryEvent});
}
```

Of course, this code also can be used as a starting point for other query manipulations and also works with saved custom criterias.

To read more about the af:query component, see:

http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_query.html

How-to restrict file upload sizes in ADF Faces

Many of the ADF Faces configuration settings use Apache Trinidad files or context parameters. This is also true when configuring the web.xml file for file upload settings used by the af:inputFile component.

```
<context-param>
<!-- Maximum memory per request (in bytes) -->
<param-name>
org.apache.myfaces.trinidad.UPLOAD_MAX_MEMORY
</param-name>
- <!-- Use 500K -->
<param-value>512000</param-value>
</context-param>
<context-param>
```

For more information about file upload in ADF Faces, see: http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_inputFile.html

Map Viewer doesn't show maps for large parts of the globe

"The Map component provides the ablity to display different maps and enables high performance panning, zooming and display of different layers (aka Themes) of data. Unlike other ADF Faces component, the Map component itself doesn't take a data model via the 'value' attribute. Instead, it only needs a configuration that contains a URL to a Map Viewerer service and optionally a Geo-Coder service if address data will have to be converted to longitutde and latitude."[1]

However, large parts of the globe are not displayed when testing the ADF Faces DVT map viewer component using the Oracle map server accessible from http://elocation.oracle.com/mapviewer. The reason for this is that the exposed map server is for demo purpose only and does not have all the maps installed that exist. There also is no guarantee that the service is up. If you want to try the Oracle map viewer and the ADF Faces DVT map viewer component then the

http://elocation.oracle.com/mapviewer is good to use. If you want to integrate maps in your business applications then you need a license and local installation of the map server. If the demo server is accessible from a browser but not from Oracle JDeveloper, check your proxy settings in Tools | Preferences | Web Browser and Proxy

[1]: http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12418/tagdoc/dvt_map.html

How-to control user input based on RegEx pattern

ADF Faces provides Regular Expression validation component that allow developers to test user input for a specific input pattern like numbers or characters only, value ranges and more. Validation however works after the fact, which means the user already provided a wrong entry. To prevent wrong user entry to an input field, JavaScript can be used to ignore the key press when it doesn't fit a specific pattern.

```
<af:inputText label="RegEx Sample - Values between 1 - 50" id="it2">
<af:clientListener method="applyRegExPatternFilter('^[1-9]{1}$|^[1-4]{1}[0-9]{1}$|^50$')"
type="keyDown"/>
```

</af:inputText>

The Regular expression in the sample code above allows values between 1 and 50 to be entered in a text input field. Any value lower or bigger than this is suppressed and the keyboard entry is not accepted. The JavaScript function for this is shown below:

```
// JavaScript filter that suppresses user input if the defined regular
// expression pattern is not met. Use this pattern if you want to
// enforce specific user input patterns.
```

```
function applyRegExPatternFilter(pattern) {
  return function(evt) {
      var inputField = evt.getCurrentTarget();
      var keyCode = evt.getKeyCode();
      var oldValue = inputField.getSubmittedValue();
       //allowed keys to navigate, delete and tab out
       var controlKeys = new Array(AdfKeyStroke.ARROWRIGHT KEY,
                         AdfKeyStroke.ARROWLEFT KEY,
                         AdfKeyStroke.BACKSPACE KEY,
                         AdfKeyStroke.DELETE KEY,
                         AdfKeyStroke.END KEY,
                         AdfKeyStroke.ESC KEY,
                         AdfKeyStroke.TAB KEY);
      var isControlKey = false;
       //check if the pressed key is a control key
       for (var i=0; i < controlKeys.length; ++i) {</pre>
          if (controlKeys[i] == keyCode) {
               isControlKey = true;
               break;
          }
       }
       if (isControlKey == false)
      {
        var regExp = new RegExp(pattern,"i");
        var hasMatch = false;
        var keyChar =
            AdfKeyStroke.getKeyStroke(keyCode).toMarshalledString();
        hasMatch = regExp.test(oldValue.concat(keyChar));
        if(!hasMatch)
           inputField.setValue(oldValue);
           evt.cancel();
      }
    }
  }
}
```

You add the JavaScript to a page by either adding it to the body area of an af:resource tag, or referencing it in an external JS file from the **source** property of the af:resource tag.

The JavaScript function uses a callback to allow developers to pass additional arguments into it. Using a JavaScript callback like this allows writing generic code that can easily go into external JavaScript files. To change the JavaScript sample to only allow numeric entries, use the following configuration

```
<af:inputText label="RegEx Sample - Values between 1 - 50" id="it2">
<af:clientListener method="applyRegExPatternFilter('^[0-9]*$')"
```

type="keyDown"/>

</af:inputText>

Using the same JavaScript function with the ^[0-9]*\$ expression blocks all user entry of characters.

Note: The Regular Expression used with this JavaScript function must return true for all user entry until the user attempts to add a wrong character. E.g. a RegEx expression to verify mail addresses will not work with this code as it requires a complete mail address to be provided to work.

Note: Using JavaScript like this with af:inputDate doesn't work. The getSubmittedValue does not return a date object but a string, which fails when setting it as a value on the inputDate field.

Get social security numbers right

A common development use case is to guide users when working with input fields that require a specific input format. For example, credit card and social security number fields use character delimiters that you may want to enforce on a field. The following sample uses JavaScript to add a defined delimiter character according to a defined character.

The American social security pattern is defined as xxx-xxx. Users that type 123456789 should have the input automatically corrected to 123-45-6789 while they type. Also, the field should be protected from character input and input length larger than the provided pattern.

With the above configuration, the **handleNumberFormatConversion** method is called for each key stroke in the input field. Additional arguments provided to the function are the input pattern and the character delimiter.

The JavaScript code that is addressed by the clientListener on the InputText is shown below:

```
// JavaScript function that applies a specific format to numeric input.
// The pattern argument defines the input mask, e.g. xxx-xx-xxxx. The
// delimiter defines the delimiter character to add to the user input
// based on the pattern
function handleNumberFormatConversion(pattern, delimiter){
  return function(evt){
    var inputField = evt.getCurrentTarget();
    var keyPressed = evt.getKeyCode();
    var oldValue = inputField.getSubmittedValue();
    //keycode 48-57 are keys 0-9
    //keycode 96-105 are numbpad keys 0-9
```

```
OTN ORACLE JDEVELOPER FORUM HARVEST 12 / 2010
```

```
var validKeys = new Array(48,49,50,51,52,53,54,55,
                           56, 57, 96, 97, 98, 99, 100,
                           101,102,103,104,105,
                          AdfKeyStroke.ARROWRIGHT KEY,
                          AdfKeyStroke.ARROWLEFT KEY,
                          AdfKeyStroke.BACKSPACE KEY,
                          AdfKeyStroke.DELETE KEY,
                          AdfKeyStroke.END KEY,
                          AdfKeyStroke.ESC KEY,
                          AdfKeyStroke.TAB KEY);
  var numberKeys = new Array(48,49,50,51,52,53,54,55,
                              56, 57, 96, 97, 98, 99, 100,
                              101,102,103,104,105);
var isValidKey = false;
for (var i=0; i < validKeys.length; ++i) {</pre>
   if (validKeys[i] == keyPressed) {
        isValidKey = true;
        break;
   }
}
if(isValidKey){
  //key is valid, ensure formatting is correct
  var isNumberKey = false;
  for (var n=0; n < numberKeys.length; ++n) {</pre>
    if(numberKeys[n] == keyPressed){
      isNumberKey = true;
     break;
    }
  }
  if(isNumberKey){
    //if the user provided enough data, cancel
    //the input
    var formatLength = pattern.length;
    if(formatLength == oldValue.length) {
    inputField.setValue(oldValue);
      evt.cancel();
    }
    //more values allowed. Check if delimiter needs to be set
    else{
      //if the date format has a delimiter as the next
      //character, add it
      if(pattern.charAt(oldValue.length) == delimiter) {
        oldValue = oldValue+delimiter;
```



ADF CODE CORNER 12

```
inputField.setValue(oldValue);
        }
    }
    else{
        //key is not valid, so undo entry
        inputField.setValue(oldValue);
        evt.cancel();
    }
}
```

The sample is for number only input. However, changing it for character or mixed input is not difficult to do. Note however that you can't use this with af:inputDate component because this component doesn't work well when setting formatted values as the value property.

How-to call server side Java from JavaScript

The af:serverListener tag in Oracle ADF Faces allows JavaScript to call into server side Java. The example shown below uses an af:clientListener tag to invoke client side JavaScript in response to a key stroke in an Input Text field. The script then call a defined af:serverListener by its name defined in the **type** attribute. The server listener can be defined anywhere on the page, though from a code readability perspective it sounds like a good idea to put it close to from where it is invoked.

```
<af:inputText id="it1" label="...">
<af:clientListener method="handleKeyUp" type="keyUp"/>
<af:serverListener type="MyCustomServerEvent"
method="#{mybean.handleServerEvent}"/>
```

```
</af:inputText>
```

}

The JavaScript function below reads the event source from the event object that gets passed into the called JavaScript function. The call to the server side Java method, which is defined on a managed bean, is issued by a JavaScript call to AdfCustomEvent. The arguments passed to the custom event are the event source, the name of the server listener, a message payload formatted as an array of key:value pairs, and true/false indicating whether or not to make the call immediate in the request lifecycle.

The server side managed bean method uses a single argument signature with the argument type being ClientEvent. The client event provides information about the event source object — as provided in the call to AdfCustomEvent, as well as the payload keys and values. The payload is accessible from a call to getParameters, which returns a HashMap to get the values by its key identifiers.

```
public void handleServerEvent(ClientEvent ce){
   String message = (String) ce.getParameters().get("fvalue");
...
}
```

Find the tag library at:

http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_serverListener.html

How to expose an ADF application in a Portlet?

The Oracle JSF Portlet Bridge allows developers to declaratively expose Oracle ADF applications and task flows as JSR 168 portlets. The configuration requires the definition of the initial view which then becomes the starting point for navigation within the application. Note that applications exposed in a Portlet must be able to run stand alone. This excludes applications that only consist of a bounded task flow using page fragments.

http://download.oracle.com/docs/cd/E14571_01/webcenter.1111/e10148/jpsdg_bridge.htm#CACBAIJD

How-to query af:quickQuery on page load ?

A quick query component doesn't execute the query on page load. Check the "Query Automatically" checkbox in the ViewCriteria definition does not work as it does for the af:query component or list of values. To automatically query the af:quickQuery component, select the page's PageDef.xml file and expand the **Executables** node.

SearchPageDef.xml - Structure	_			
📌 🖹				
🕀 🖓 🕀 🕀 🕀 🕀 🕀 🕀 🕀				
🖮 🛃 SearchPageDef	TimplicitViewCriteriaQuery	- Property Inspector		
parameters	🖳 📌 🔄 🥒 🍈 Find	\$₽?		
variables	🖃 Common			
allEmployeesIterator	• id *:	ImplicitViewCriteriaQuery		
	• Binds:	allEmployeesIterator 🗸 🗸		
	Refresh:	<default> (deferred)</default>		
	RefreshAfter:	v		
	RefreshCondition:	~ ~ •		
	• Criteria:	ImplicitViewCriteria V		
	Customizer:	oracle.jbo.uicli.binding.JUSearc 🗸		
	DefaultQuickSearchAttrName:	~		
	TrackQueryPerformed:	<default> (PageFlow) ▼ ∨</default>		
	InitialQueryOverridden:	tue 🗸 🚽		
	L	Y/		

Select the ImplicitViewCriteriaQuery entry and set the InitialQueryOverriden property to true.

How-to create a select one choice list of common time zones

ADF Faces provides an option to query a list of common timezones for display in a Select One Choice component. The EL expression for this is #{af:getCommonTimeZoneSelectItems()}.

To use this expression in a Single Select One Choice component, drag and drop the component from the Oracle JDeveloper Component Palette into a JSF page. In the opened dialog, copy the expression into the **Value** property below the **Bind to list (select items)** header.

🥧 Insert Select One Choice	- Step 1 of 2		×
Select			
Select Common Properties	Bind to list (select items <u>Value:</u> #{af:getCommonT <u>C</u> reate list (select item)	;) imeZoneSelectItems()}	Bin <u>d</u>
			+ 🗙
	Item Label	Item Value	

Complete the dialog and run the page to see all time zones.



The page source is shown below

<af:selectOneChoice label="TimeZones" id="soc1">

<f:selectItems value="#{af:getCommonTimeZoneSelectItems()}"

</af:selectOneChoice>

For more information about using time zones with the affinputDate component, please read section **9.5.3 What You May Need to Know About Selecting Time Zones without the inputDate Component** of "Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework 11g" :

http://docs.tpu.ru/docs/oracle/en/owl/E14571_01/web.1111/b31973/af_input.htm#BABBJECD

How-to hide the close icon for task flows opened in dialogs

ADF bounded task flows can be opened in an external dialog and return values to the calling application as documented in chapter 19 of Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework11g:

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/taskflows_dialogs.htm#BABBAFJB

Setting the task flow call activity property **Run as Dialog** to **true** and the **Display Type** property to **inline-popup** opens the bounded task flow in an inline popup. To launch the dialog, a command item is used that references the control flow case to the task flow call activity

<af:commandButton text="Lookup" id="cb6" windowEmbedStyle="inlineDocument" useWindow="true" windowHeight="300" windowWidth="300" action="lookup" partialSubmit="true"/>

By default, the dialog that contains the task flow has a close icon defined that if pressed closes the dialog and returns to the calling page. However, no event is sent to the calling page to handle the close case.

To avoid users closing the dialog without the calling application to be notified in a return listener, the close icon shown in the opened dialog can be hidden using ADF Faces skinning.

	DepartmentId	Department Nai			
▷ Roma		A			
Venice					
⊳ Tokyo		19			\frown
Hiroshima				-	\smile
Southlake		DepartmentId	DepartmentName	ManagerId	
▷ South San Fran		357	Sales II		
South Brunswick		10	Administration	200	
▷ Seattle		20	Marketing	201	
> Toronto		30	Purchasing	114	
Whitehorse		40	Human Resources	203	
Beijing		50	Shipping	103	=
⊳ Bombay		60	Π	103	
Sydney		70	Public Relations	204	
Singapore		80	Sales	145	
London		90	Executive	100	
> Oxford		100	Finance	108	
Stretford		110	Accounting	205	
4		120	Treasury		
		130	Corporate Tax		
Lookup		140	Control And Credit		
		150	Sales II		
		160	Benefits		-
		•	III	•	
		Close			

The following skin selector hides the close icon in the dialog

af | panelWindow::close-icon-style{ display:none; }

To learn about skinning, see chapter 20 of Oracle Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework

http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/af_skin.htm#BAJFEFCJ

However, the skin selector that is shown above hides the close icon from all af:panelWindow usages, which may not be intended. To only hide the close icon from dialogs opened by a bounded task flow call activity, the ADF Faces component **styleClass** property can be used.

The af:panelWindow component shown below has a "withCloseWindow" style class property name defined. This name is referenced in the following skin selector, ensuring that the close icon is displayed

af|panelWindow.withCloseIcon::close-icon-style{ display:block; }

In summary, to hide the close icon shown for bounded task flows that are launched in inline popup dialogs, the default display behavior of the close icon of the af:panelWindow needs to be reversed. Instead to always display the close icon, the close icon is always hidden, using the first skin selector. To show the disclosed icon in other usages of the af:panelWindow component, the component is flagged with a **styleClass** property value as shown below

<af:popup id="p1">

```
<af:panelWindow id="pw1" contentWidth="300" contentHeight="300"
styleClass="withCloseIcon"/>
```

</af:popup>

The "withCloseIcon" value is referenced in the second skin definition

af | panelWindow.withCloseIcon::close-icon-style { display:block; }

The complete entry of the skin CSS file looks as shown below:

af|panelWindow::close-icon-style{ display:none; }

af | panelWindow.withCloseIcon::close-icon-style { display:block; }

How-to populate different select list content per table row

A frequent requirement posted on the OTN forum is to render cells of a table column using instances of af:selectOneChoices with each af:selectOneChoice instance showing different list values.

To implement this use case, the select list of the table column is populated dynamically from a managed bean for each row. The table's current rendered row object is accessible in the managed bean using the **#{row}** expression, where "row" is the value added to the table's **var** property.

```
<af:table var="row">
...
<af:column ...>
<af:selectOneChoice ...>
<f:selectItems value="#{browseBean.items}"/>
</af:selectOneChoice>
</af:column
</af:table>
```

The **browseBean** managed bean referenced in the code snippet above has a **setItems** and **getItems** method defined that is accessible from EL using the **#{browseBean.items}** expression.

When the table renders, then the **var** property variable – the #{row} reference – is filled with the data object displayed in the current rendered table row.

The managed bean getItems method returns a List<SelectItem>, which is the model format expected by the f:selectItems tag to populate the af:selectOneChoice list.

```
public void setItems(ArrayList<SelectItem> items) {}
//this method is executed for each table row
public ArrayList<SelectItem> getItems() {
  FacesContext fctx = FacesContext.getCurrentInstance();
 ELContext elctx = fctx.getELContext();
 ExpressionFactory efactory =
         fctx.getApplication().getExpressionFactory();
 ValueExpression ve =
         efactory.createValueExpression(elctx, "#{row}", Object.class);
 Row rw = (Row) ve.getValue(elctx);
  //use one of the row attributes to determine which list to query and
  //show in the current af:selectOneChoice list
  // ...
 ArrayList<SelectItem> alsi = new ArrayList<SelectItem>();
  for( ... ) {
      SelectItem item = new SelectItem();
        item.setLabel(...);
        item.setValue(...);
        alsi.add(item);
  return alsi;
}
```

For better performance, the ADF Faces table stamps it data rows. Stamping means that the cell renderer component – af:selectOneChoice in this example – is instantiated once for the column and then repeatedly used to display the cell data for individual table rows. This however means that you cannot refresh a single select one choice component in a table to change its list values. Instead the whole table needs to be refreshed, rerunning the managed bean list query.

Be aware that having individual list values per table row is an expensive operation that should be used only on small tables for Business Services with low latency data fetching (e.g. ADF Business Components and EJB) and with server side caching strategies for the queried data (e.g. storing queried list data in a managed bean in session scope).

RELATED DOCOMENTATION

Э	