

ADF Code Corner

016-How-to customize the ADF Faces Table Filter



twitter.com/adfcodecorner

Abstract:

ADF Faces tables can be created with a default table filter for users to further filter the result set of a query, which can be in memory or including re-querying the database. By default, table filters are shown as input text fields with no restriction on what users can type in as a filter pattern. The table filter components however can be customized by the developer to add missing user guidance, reducing the surface for user errors when working within ADF bound applications. This how-to demonstrates how to add an `af:selectOnceChoice` as a filter component.

Author:

Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
03-AUG-2010

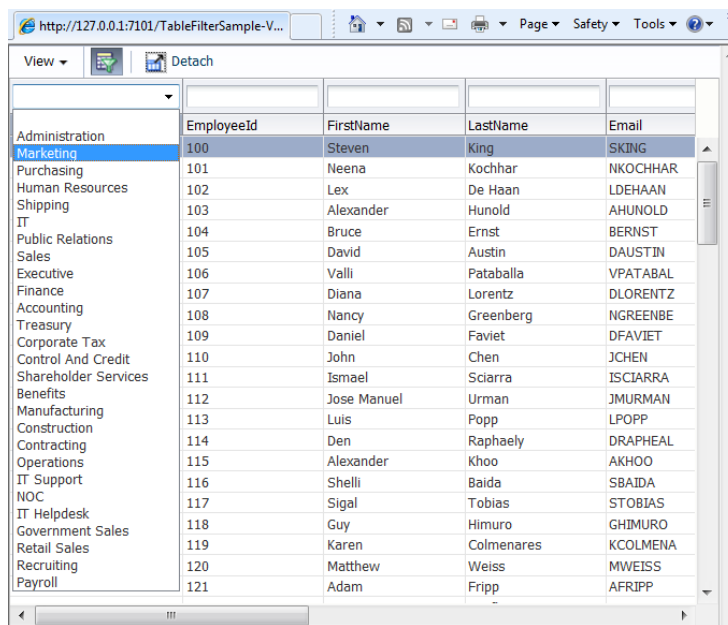
Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: <http://forums.oracle.com/forums/forum.jspa?forumID=83>

Introduction

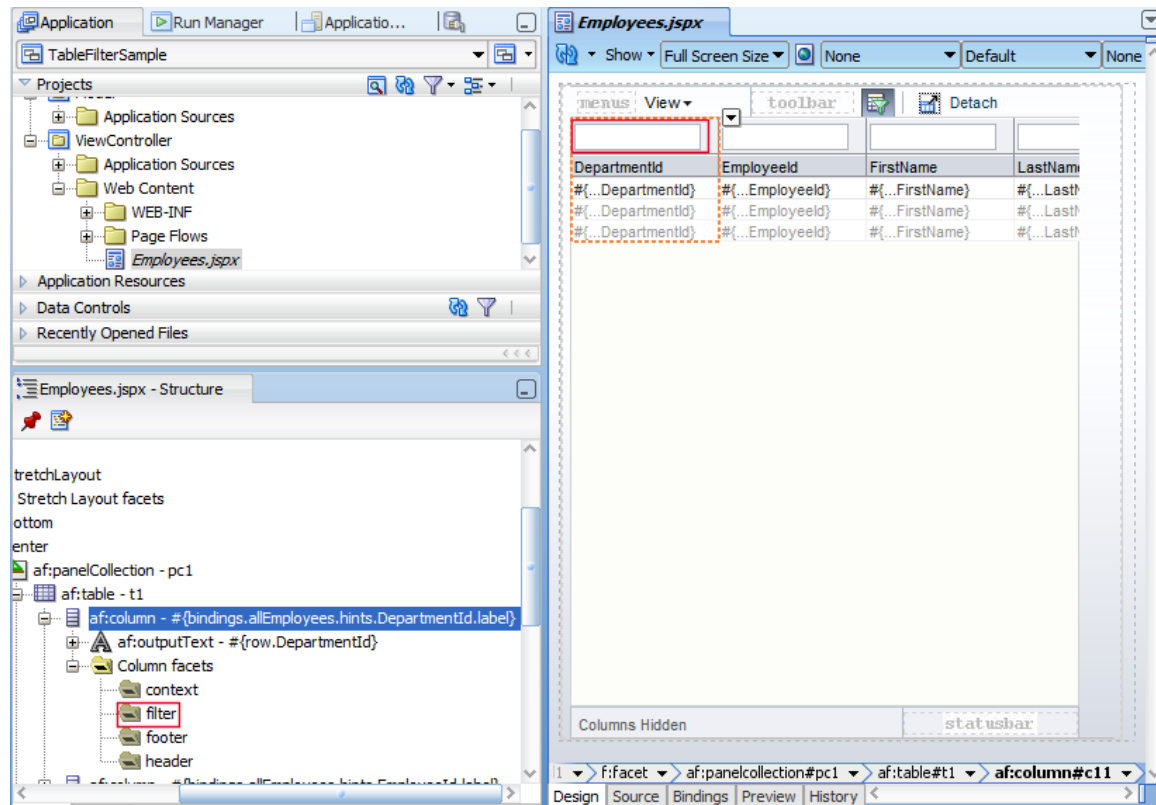
This article explains how to change the default table filter component in ADF Faces to a custom component renderer by the example of an af:selectOneChoice. As shown in the image below, at the end of this how-to, the DepartmentId column can be filtered with a selectOneChoice list, which is less error prone to use than a free text input.



| | EmployeeId | FirstName | LastName | Email |
|----------------------|------------|-------------|------------|----------|
| Administration | 100 | Steven | King | SKING |
| Marketing | 101 | Neena | Kochhar | NKOCHHAR |
| Purchasing | 102 | Lex | De Haan | LDEHAAN |
| Human Resources | 103 | Alexander | Hunold | AHUNOLD |
| Shipping | 104 | Bruce | Ernst | BERNST |
| IT | 105 | David | Austin | DAUSTIN |
| Public Relations | 106 | Valli | Pataballa | VPATABAL |
| Sales | 107 | Diana | Lorentz | DLORENTZ |
| Executive | 108 | Nancy | Greenberg | NGREENBE |
| Finance | 109 | Daniel | Faviet | DFAVIET |
| Accounting | 110 | John | Chen | JCHEN |
| Treasury | 111 | Ismael | Sciarra | ISCIARRA |
| Corporate Tax | 112 | Jose Manuel | Urman | JMURMAN |
| Control And Credit | 113 | Luis | Popp | LPOPP |
| Shareholder Services | 114 | Den | Raphaely | DRAPHEAL |
| Benefits | 115 | Alexander | Khoo | AKHOO |
| Manufacturing | 116 | Shelli | Baida | SBAIDA |
| Construction | 117 | Sigal | Tobias | STOBIAS |
| Contracting | 118 | Guy | Himuro | GHIMURO |
| Operations | 119 | Karen | Colmenares | KCOLMENA |
| IT Support | 120 | Matthew | Weiss | MWEISS |
| NOC | 121 | Adam | Fripp | AFRIPP |
| IT Helpdesk | | | | |
| Government Sales | | | | |
| Retail Sales | | | | |
| Recruiting | | | | |
| Payroll | | | | |

Creating a table

To create a databound table, simply drag a collection – like an ADF BC View Object – from the Data Controls panel to the JavaServer Faces page. In the opened dialog, select table | read-only table and make sure the “filter” option is selected so the table header shows filters in the column headers

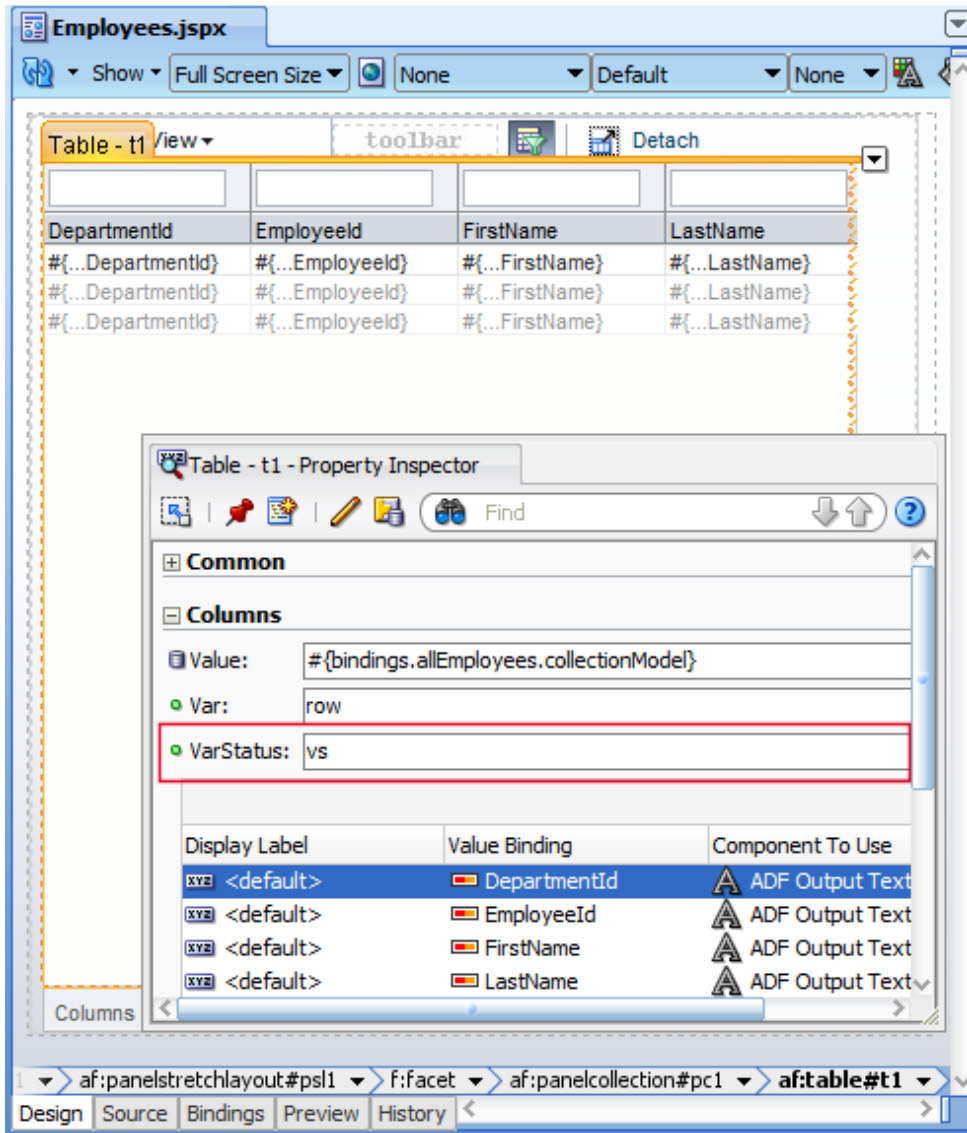


As shown in the image above, the table filter is defined on the `af:column`. The default implementation of the table filter cell renderer is an `af:inputText` field. This allows users to type filter conditions in and query the table by hitting the enter key. To customize the filter, you can add an ADF Faces component to the “filter” facet, as shown in the image above. The component added to the facet then replaces the default filter for the column.

For this article, we will change the default filter renderer with an `af:selectOnce` choice component.

About the table variable status

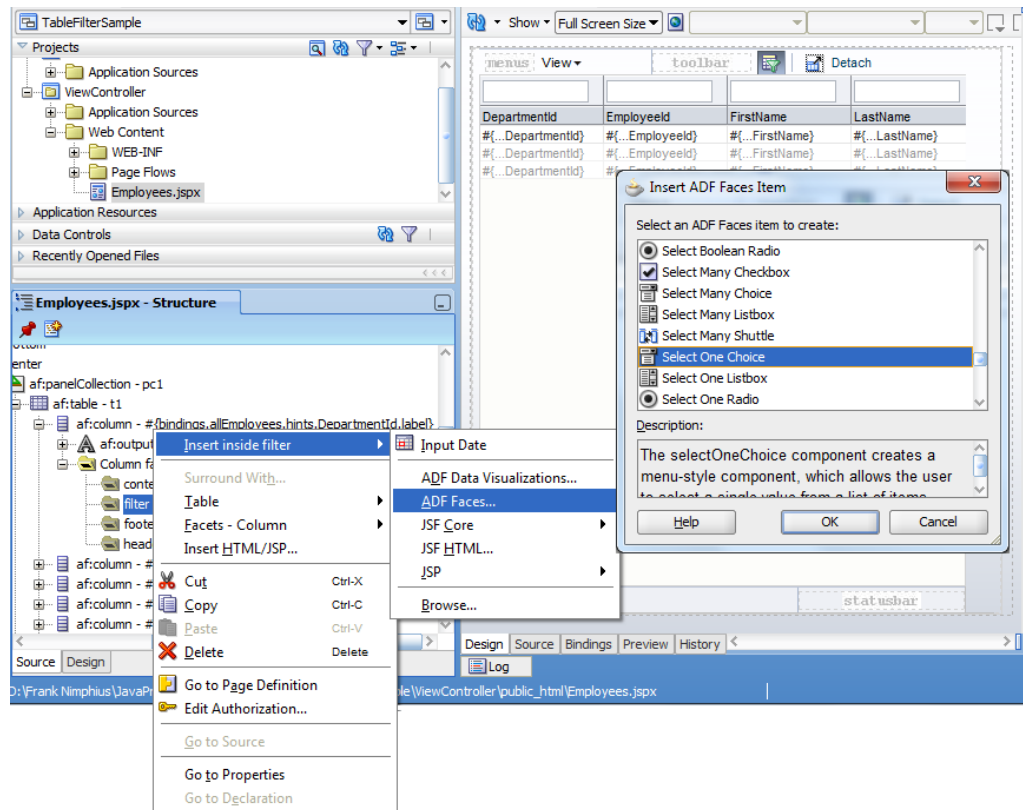
Not many pay attention to a default setting on the ADF bound table, which is the “`varStatus`” attribute that is set to “`vs`” when dragging a collection as a filterable table to a JSF page. The “`vs`” variable is a handle to filter functionality and needs to be referenced from the customized table filter renderer. As shown in the image below, the status variable is displayed in the “Columns” section of the Property Inspector for the selected table.



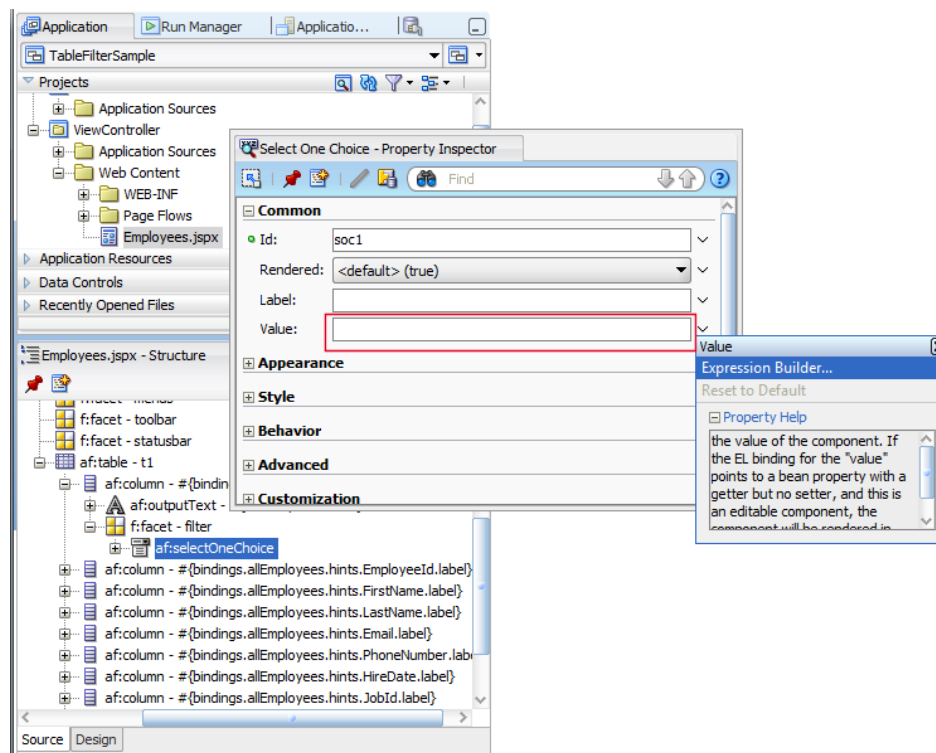
Customizing the table filter

To customize the table filter, select the “filter” facet of an af:column component and choose “Insert inside filter” from the right mouse context menu to add a select one choice component. You can finish the select one choice edit dialog without a configuration, except for clearing the label field.

All configurations to make the select one choice work as a filter component will be done manually in the following.



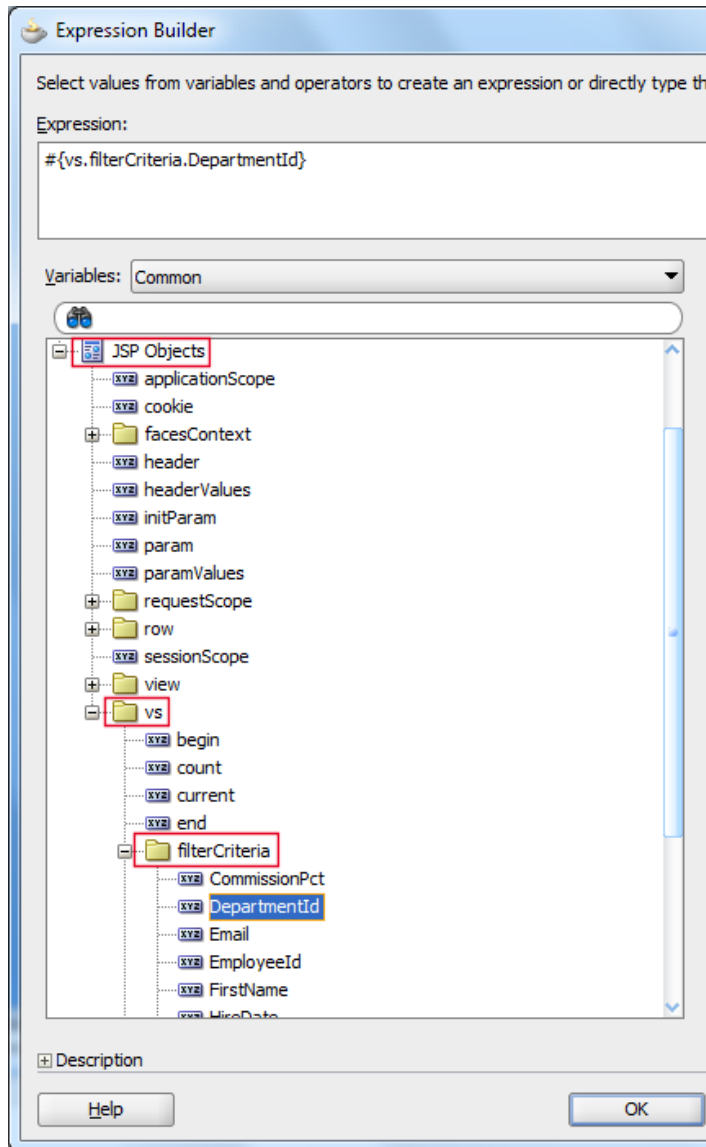
The `af:selectOneChoice` “value” attribute writes the selected list value as the filter criteria to the filter binding. To do so, select the “value” property in the Property Inspector and choose “Expression Builder” from the context menu as shown in the image below.



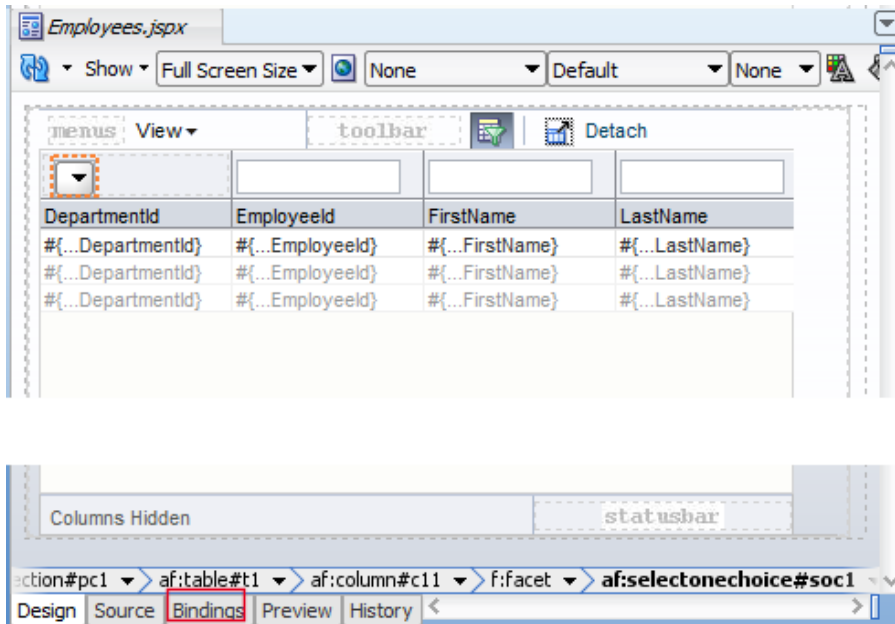
The filter criteria for a column is set using an EL string like `{vs.filterCriteria.<attribute name>}`.

To discover the attributes that are available to filter for a table, in the Expression Builder in Oracle JDeveloper, expand the JSP Objects node to access the “vs” node. Remember that the “vs” note is set as an attribute value on the table. So if the name differs, then you see a different value entry in the EL Builder.

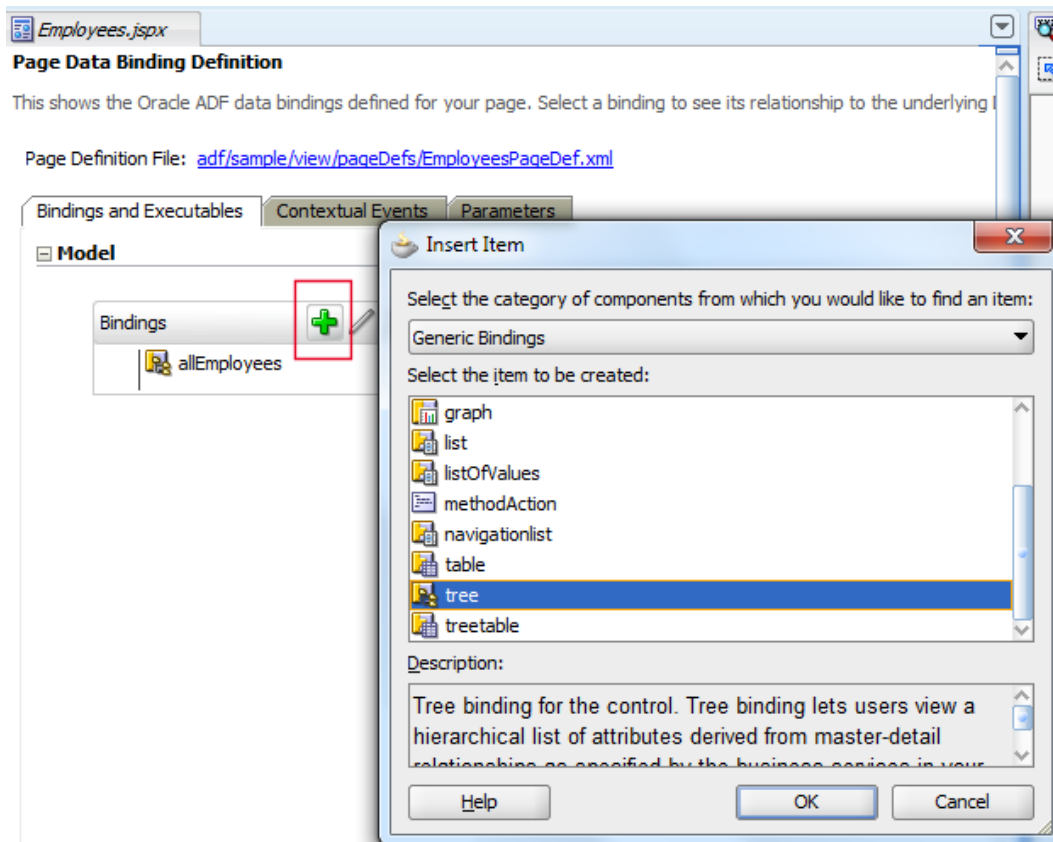
Expand the “filterCriteria” node and select the attribute name of the attribute that is represented by the table column to filter. In the example below, we are going to show a list of departments to help filtering the entries for a specific department.



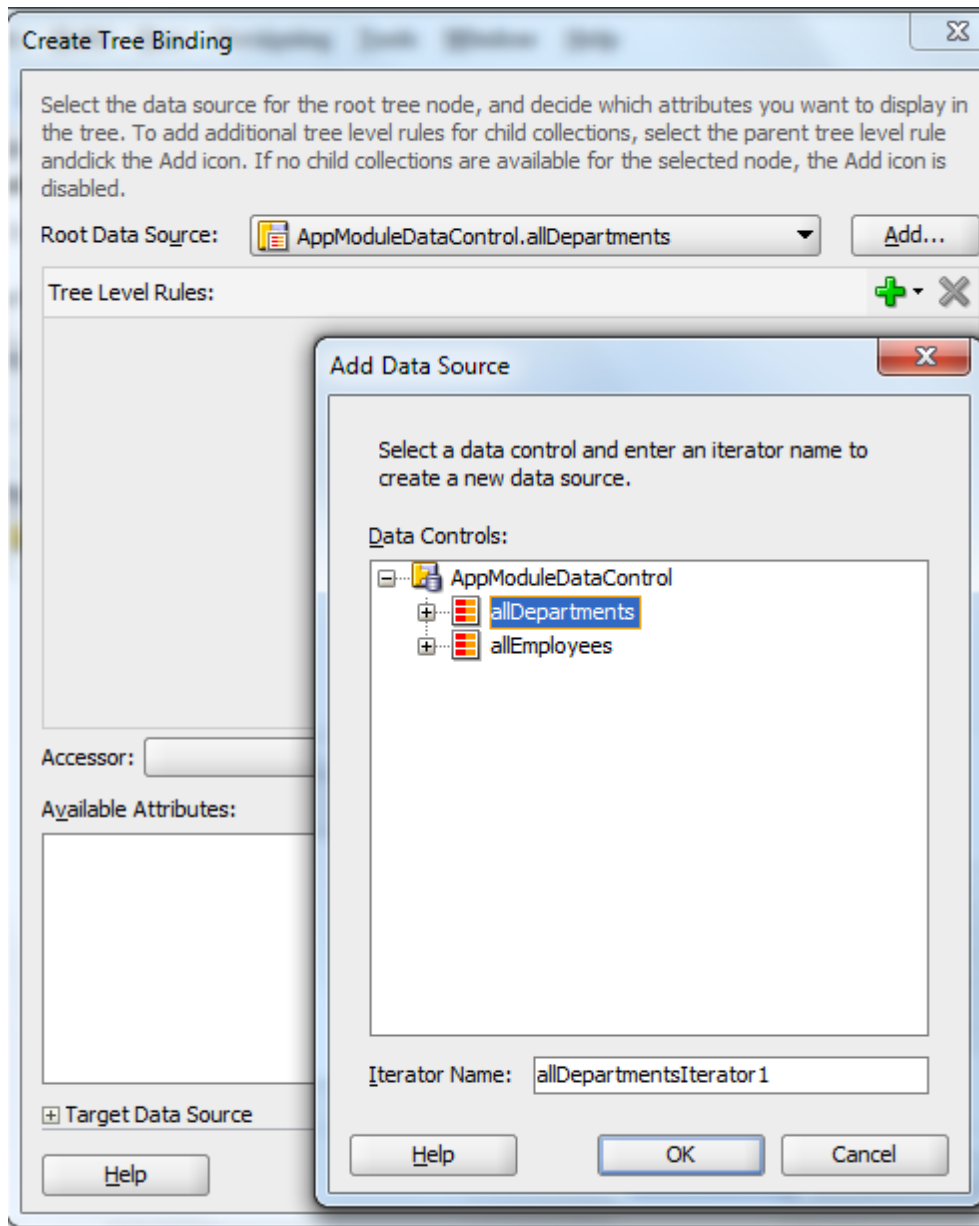
Having done this, the table now shows a selectOneChoice component configured to set the table filter for the DepartmentId attribute. Next is to create the list of values.



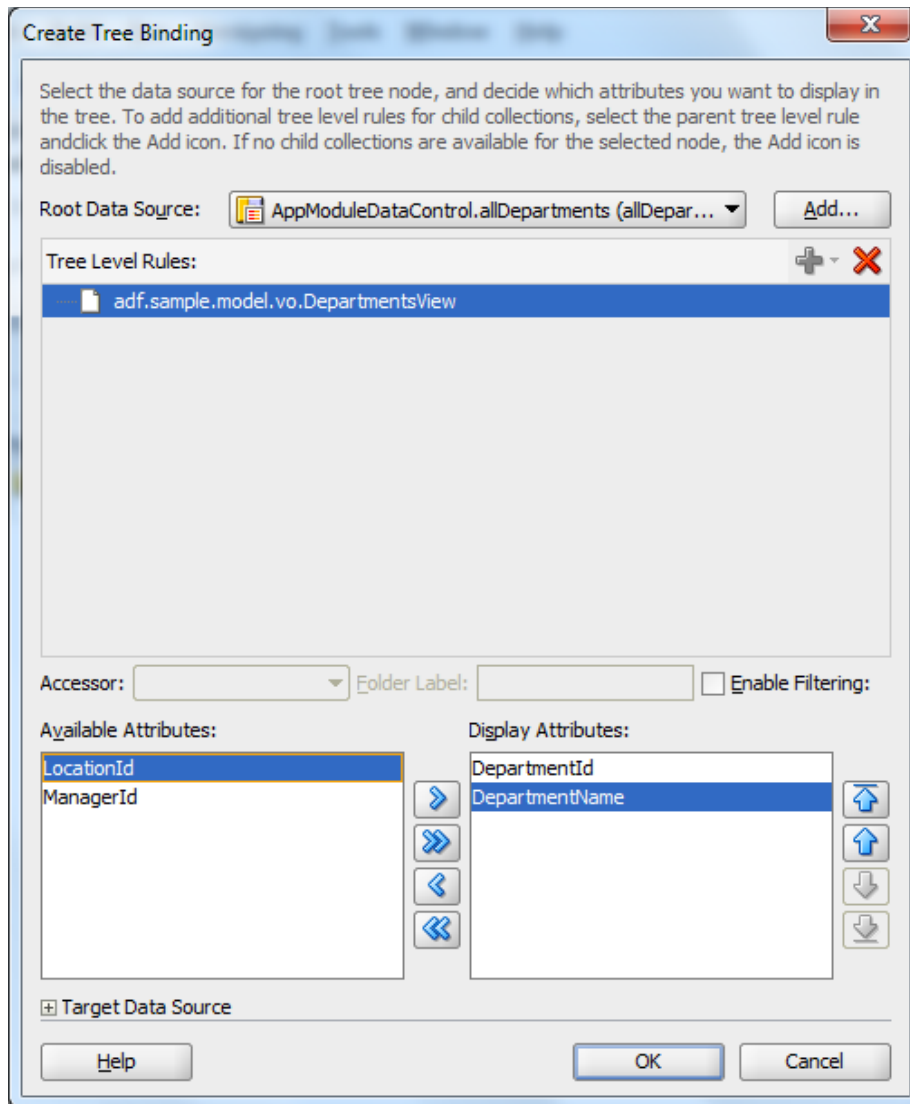
In the example, instead of using a static list of values, we read the department names from an ADF binding so the list is produced dynamically.



To create the ADF binding for the list, select the “Bindings” tab at the bottom of the JSF page. In the binding editor, press the green plus icon in the “Bindings” section and choose “tree” from the list of “Generic Bindings”. The tree binding can be bound to a collection – like Departments View Object in the example – and return a list of values to display in the af:selectOneChoice.



Create a single level tree, as shown in the image above. Press the “Add” button to select a collection, which then implicitly creates an iterator, and then press the green plus icon to create the tree rule.



Form the available list of attributes, select at least one. In the example above, the plan is to show the department name to the user and internally use the department Id when filtering the table.

Note: When the binding is created, navigate to the PageDef file and select the iterator that was created for the tree binding. The iterator is in the “Executable” section of the PageDef file. Select the iterator and open the Property Inspector. Change the “RangeSize” property from “10”, the default value to “-1”, so the list shows all available list values

Note: Be sensible with how many list values you show in the selectOneChoice. If, for example, the departments table had 1 0000 0000 rows then an af:selectOneChoice as the table filtering component is not a good choice. In this case you would use a LOV.

To populate the af:selectOneChoice components with values read from the ADF binding, we use an af:forEach component that iteratively created instance of f:selectItem components for each list entry. For this, you first remove the f:selectItems entry that is created when adding the af:selectOnceChoice component. The page source should look as shown below

```

<af:column sortProperty="DepartmentId" filterable="true"
    sortable="true" id="c11"
    headerText="#{bindings.allEmployees.hints.DepartmentId.label}">
<af:outputText value="#{row.DepartmentId}" id="ot5">
    <af:convertNumber groupingUsed="false"
        pattern="#{bindings.allEmployees.hints.DepartmentId.format}"/>
</af:outputText>
<f:facet name="filter">
    <af:selectOneChoice id="soc1"
        value="#{vs.filterCriteria.DepartmentId}">
        <af:forEach var="listrow"
            items="#{bindings.allDepartments.rangeSet}">
            <f:selectItem id="sil"
                itemValue="#{listrow.DepartmentId}"
                itemLabel="#{listrow.DepartmentName}"/>
        </af:forEach>
    </af:selectOneChoice>
</f:facet>
</af:column>

```

The af:forEach component references the ADF tree binding - #{bindings.allDepartments.rangeSet}, which queries the Departments View Object for the available department Ids and names. The af:forEach component has a “var” attribute value defined as “listrow”, which is used to populate the select item label and value.

At runtime the table shows as in the image below. The af:selectOne Choice displays all departments for the user to choose from.

| | EmployeeId | FirstName | LastName | Email |
|----------------------|------------|-------------|------------|----------|
| Administration | | | | |
| Marketing | 100 | Steven | King | SKING |
| Purchasing | 101 | Neena | Kochhar | NKOCHHAR |
| Human Resources | 102 | Lex | De Haan | LDEHAAN |
| Shipping | 103 | Alexander | Hunold | AHUNOLD |
| IT | 104 | Bruce | Ernst | BERNST |
| Public Relations | 105 | David | Austin | DAUSTIN |
| Sales | 106 | Valli | Pataballa | VPATABAL |
| Executive | 107 | Diana | Lorentz | DLORENTZ |
| Finance | 108 | Nancy | Greenberg | NGREENBE |
| Accounting | 109 | Daniel | Faviet | DFAVIET |
| Treasury | 110 | John | Chen | JCHEN |
| Corporate Tax | 111 | Ismael | Sciarra | ISCIARRA |
| Control And Credit | 112 | Jose Manuel | Urman | JMURMAN |
| Shareholder Services | 113 | Luis | Popp | LPOPP |
| Benefits | 114 | Den | Raphaely | DRAPHEAL |
| Manufacturing | 115 | Alexander | Khoo | AKHOO |
| Construction | 116 | Shelli | Baida | SBAIDA |
| Contracting | 117 | Sigal | Tobias | STOBIAS |
| Operations | 118 | Guy | Himuro | GHIMURO |
| IT Support | 119 | Karen | Colmenares | KCOLMENA |
| NOC | 120 | Matthew | Weiss | MWEISS |
| IT Helpdesk | 121 | Adam | Fripp | AFRIPP |
| Government Sales | | | | |
| Retail Sales | | | | |
| Recruiting | | | | |
| Payroll | | | | |

Download

You can download the Oracle JDeveloper 11g sample for this how-to from the ADF Code Corner site:

<http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html>

Make sure you configure the database connect to point to the HR schema of your Oracle database installation.

RELATED DOCUMENTATION

| | |
|--------------------------|--|
| <input type="checkbox"/> | af:clientListener - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_clientListener.html |
| <input type="checkbox"/> | af:table - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_table.html |
| <input type="checkbox"/> | Oracle Fusion Developer Guide – Oracle Press, McGraw Hill, by Frank Nimphius, Lynn Munsinger http://www.mhprofessional.com/product.php?isbn=0071622543 |