ADF Code Corner

How-to build a single select one choice component with images in the select item list

twitter.com/adfcodecorner

Abstract:

A frequent requirement is to show images associated with select item values in a single select one choice. The select item component in an af:singleSelectOneChoice component is the JSF RI f:selectItem or f:selectItems, which don't support images nor CSS style sheets to e.g. set the image as a background image.

If you can't find it – build it! This how-to article shows how to build a custom single select one choice list out of an af:inputText field, an af:popup, an af:iterator and some JavaScript

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
03-SEP-2010

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

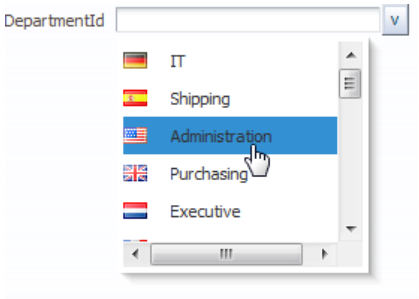
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

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

In this article we explain how to build a custom select one choice component that allows developers to add images to the select item list. The functionality of a single select list with images is not natively available in JavaServer Faces or ADF Faces.The reason is within the f:selectItem and f:selectItems tags, which don't allow the addition of images or CSS style sheet. So if you can't find it, build it with ADF Faces.

The select one choice solution shown in the image below is manually built using Oracle ADF binding for convenient component data binding access and a POJO model to provide the select item values.



Note that for simplicitly, the CSS style sheet used to define the component select list width and height is added inline. You could use skinning instead, which however only makes sense if you add multiple instances of this custom select component in your application – all with the same width and height. A functionality that could not be added as an inline style is the mouse hover effect, for which skinning needed to be used.

Building the custom select one choice component

The custom single select component is based on an af:inputText component with an associated popup.

```
<af:panelGroupLayout id="pgl3" layout="horizontal">
  <af:inputText label="DepartmentId" id="it1" clientComponent="true"
                inlineStyle="width:110.0px;"/>
  <af:commandButton text="v" id="cb1" partialSubmit="true">
    <af:showPopupBehavior popupId="p1" triggerType="action"
                          align="afterEnd" alignId="it1"/>
  </af:commandButton>
</af:panelGroupLayout>
```

The command button that is aligned with the text input field has an associated af:showPopupBehavior that opens a popup below the input component to simulate a select one choice. The popup contains an af:iterator that iterates over an ADF tree binding to render rows of image and input text components.

```
<af:popup id="p1" contentDelivery="lazyUncached" animate="false">
  <af:panelGroupLayout id="pgl1" layout="scroll"
      inlineStyle="width:160px; height:150.0px;">
    <af:iterator id="i1" var="row"
      value="#{bindings.allDepartments.collectionModel}">
      <af:panelGroupLayout id="pgl2" halign="left" styleClass="cust"
         layout="horizontal" clientComponent="true"
         inlineStyle="height:25px; width:150px; cursor:pointer">
        <af:panelGroupLayout id="pgl4" layout="horizontal">
          <af:image source="/flags/#{row.country}.gif" id="i2">
            <af:clientListener method="popupValueSelection('p1','it1')"
                               type="click"/>
            <af:clientAttribute name="departmentId"
                                value="#{row.departmentId}"/>
          </af:image>
          <af:spacer width="10" height="10" id="s1"/>
          <af:inputText id="it2" clientComponent="true"
            value="#{row.departmentName}" readOnly="true">
            <af:clientListener method="popupValueSelection('p1','it1')"
                               type="click"/>
                <af:clientAttribute name="departmentId"
                                    value="#{row.departmentId}"/>
          </af:inputText>
        </af:panelGroupLayout>
        <af:clientListener method="popupValueSelection('p1','it1')"
                           type="click"/>
        <af:clientAttribute name="departmentId"
                            value="#{row.departmentId}"/>
      </af:panelGroupLayout>
    </af:iterator>
  </af:panelGroupLayout>
</af:popup>
```

In the following, the code lines highlighted in bold and red are further discussed:

- contentDelivery – the content delivery is set to lazyUncached, which defers the first launch of the select list a bit but ensures that the opened select item list is refreshed so that the previously selected item is no longer marked.
- inlineStyle – The inline style property is used for simplicity and should be changed to using skinning if the component shall be reused in an application. The inline style property sets the height and the width of the popup list and the select item areas.
- af:iterator – the iterator component loops over the ADF tree binding collection model to read the country Id and department name. Note that you need to set the RangeSize property of the iterator that the ADF tree binding is bound to to -1 as otherwise only 10 values are shown in the select one choice component.
- panelGroupLayout – a nested panelGroupLayout is used to render each selectItem to enforce the wanted rendering with the text label aligned to the left. During testings it showed that adding CSS to the select items changed the text field alignment to middle instead of left aligned. To fix this behavior, a second panelLayoutGroup is used.
- cursor:pointer – CSS artifact that changes the mouse cursor to indicate a selectable field
- af:clientListener – the af:clientListener component is used on the image, the input text field and the panel group layout of the select item. This is to ensure that users can click anywhere in the select list to perform a selection. The selection is handled by a JavaScript function that closes the popup and writes the value back to the input text component of the select choice.
- af:clientAttribute – the af:clientAttribute component is used on the components that have an associated client listener to add a custom property that is accessible from JavaScript. Note that the client attribute cannot handle all data object types (including oracle.jbo.Key) but valid JavaScript types. In the example we use a POJO model in which the departmentId is of type Long.

The JavaScript function

The JavaScript function uses a callback to take two input arguments – the ID of the popup containing the list of select items and the ID of the text input field to write the selected value to. Using a callback like this allows developers to create reusable JavaScript functions that can be stored in external libraries.

```
<af:resource type="javascript">
   function popupValueSelection(popupId,txtId){
     return function(evt){
       //cancel event as there is no server side
       //logic to be executed for this selection
       evt.cancel();
       var popup = AdfPage.PAGE.findComponentByAbsoluteId(popupId);
       var txtField = AdfPage.PAGE.findComponentByAbsoluteId(txtId);
       var selectItem = evt.getSource();
       var depatmentId = selectItem.getProperty('departmentId');
       txtField.setValue(depatmentId);
       popup.hide();
    }
```

```
  }
</af:resource>
```

The code line highlighted in *bold red* reads the custom property from the event source, which is one of the three components with an af:clientListener added. The property value is then set as the value of the input text component that defines the select component. Finally, the popup is closed.

Skinning

It is not possible in CSS to skin the mouse hover event in an inline style definition. For this reason, the hover effect is defined in an ADF Faces skin file.

```
/* show mouse hover effect */
.cust.af|panelGroupLayout:hover {background-color:rgb(50,144,211);}
```

Note that the ".cust" style class is used to uniquely identify the panelGroupLayout component of the custom select one choice.

Known limitation

A limitation is that this custom select choice component is not controlled by the arrow keys on the keyboard. This can be implemented with JavaScript as well but goes beyond the scope of this article.

Downloads

Sample code to this article can be downloaded from the ADF Code Corner site.

RELATED DOCOMENTATION

⊠	ADF Code Corner - http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html
⊠	Client Listener tag - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_clientListener.html
⊠	clientAttribute tag - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12419/tagdoc/af_clientAttribute.html
⊠	AdfRichInputText JavaScript doc - http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e12046/oracle/adf/view/js/component/rich/input/AdfRichInputText.html