ADF Code Corner

102. How to dynamically enable or disable list items of an ADF bound select many checkbox component

twitter.com/adfcodecorner

Abstract:

A question on the Oracle JDeveloper forum on OTN was about how to dynamically disable checkbox options in an ADF bound SelectManyCheckbox component. Though asked for a specific component, the use case is generic and applies to all select choice components that read their listdata from the f:selectItems tag.

This article exlains the original posters use case using the SelectManyCheckbox component

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
25-JUN-2012

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

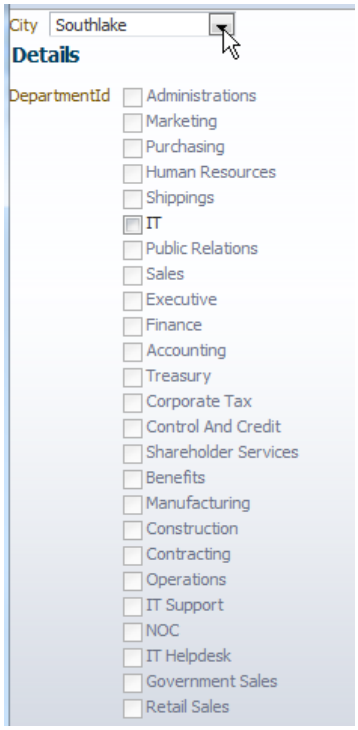
Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

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

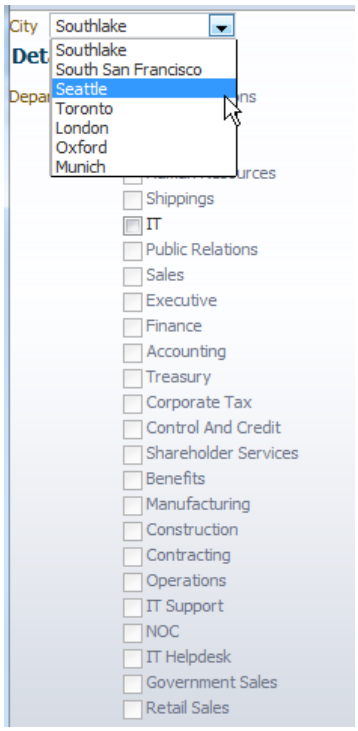
Introduction

The images below show the desired behavior, which is that based on the selected location in the select one choice component, the list of departments show disabled for departments that are not within this location.

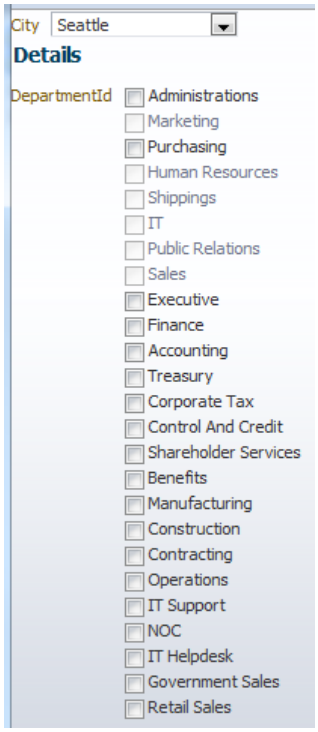
Initially the location choice loads with the city of Southlake as selected. As you will see later, I use a view criteria on the locations view object instance to only show cities that have departments.



Changing the locations choice to Seattle will change the list selected row and – through partial refresh – the list of departments, which now shows all departments in Seattle as enabled.

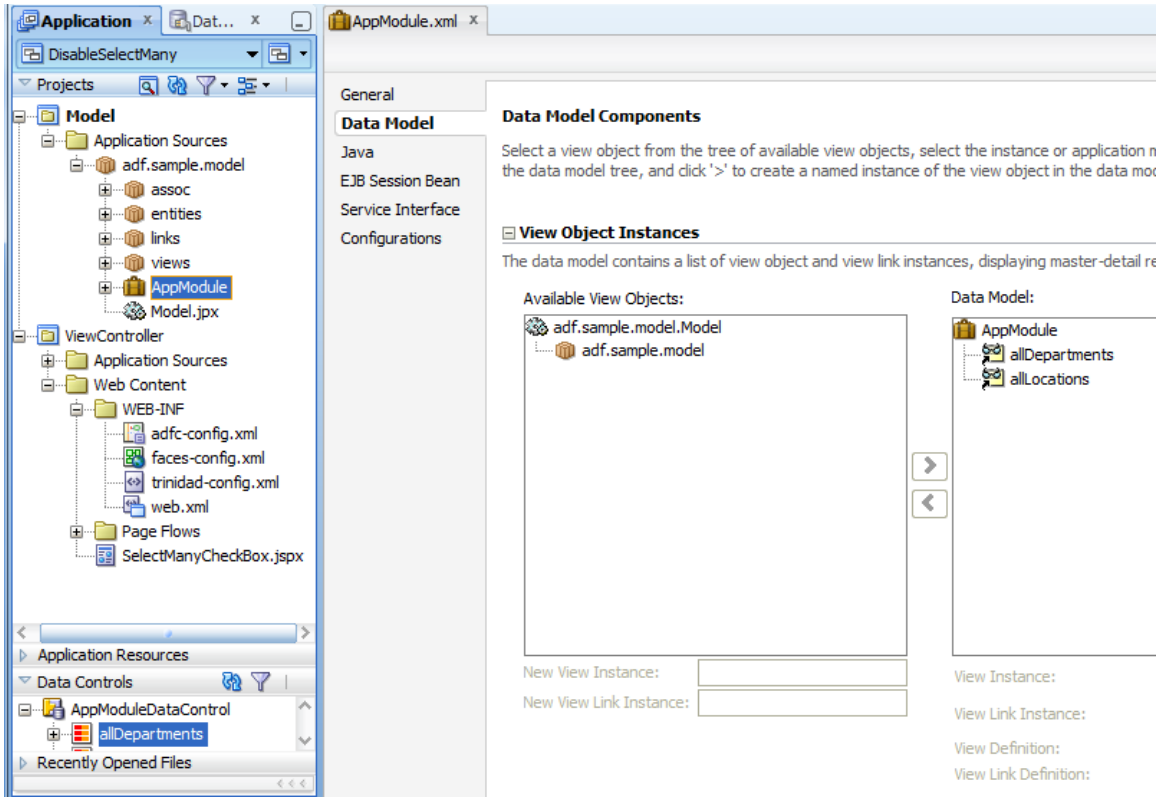


The city list is created as a navigation list as we will see later

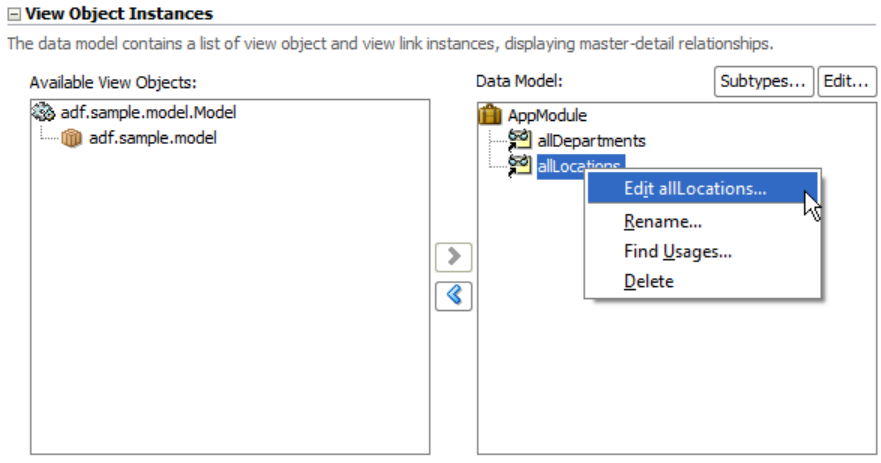


Model

Building the model is not in the focus of this article. So the following screen shots are just for reference. The ADF BC Data Model exposes two view object instances that query the departments and the locations independently.

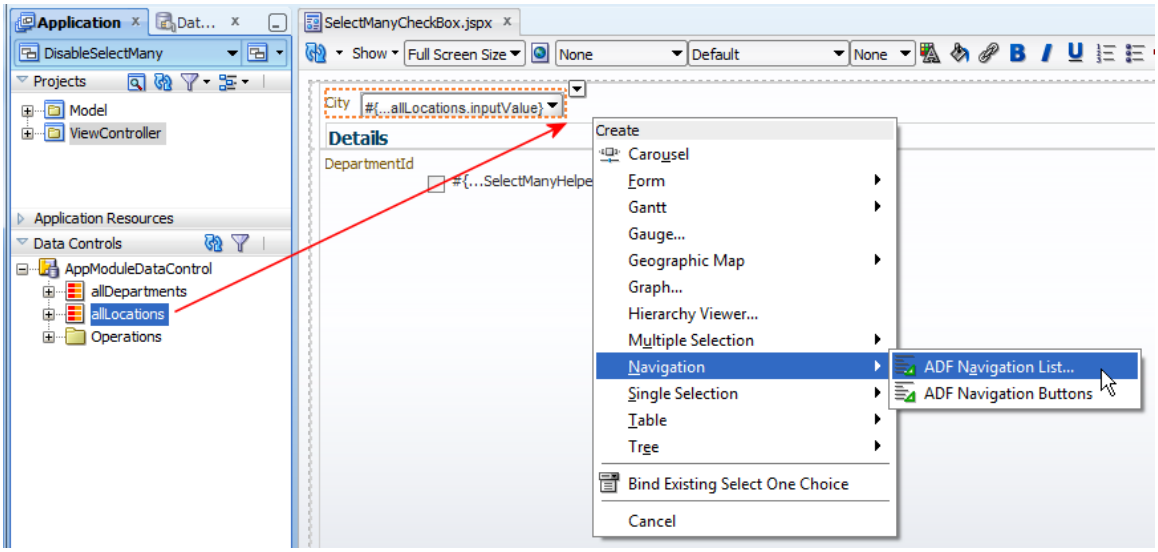


For demonstration purposes, a view criteria is defined on the LocationsView view object and configured on the **allLocations** instance to only query locations (cities) that have associated departments. To see the view criteria configuration, use the right mouse button on the **allLocations** view object instance and choose **Edit allLocations.**

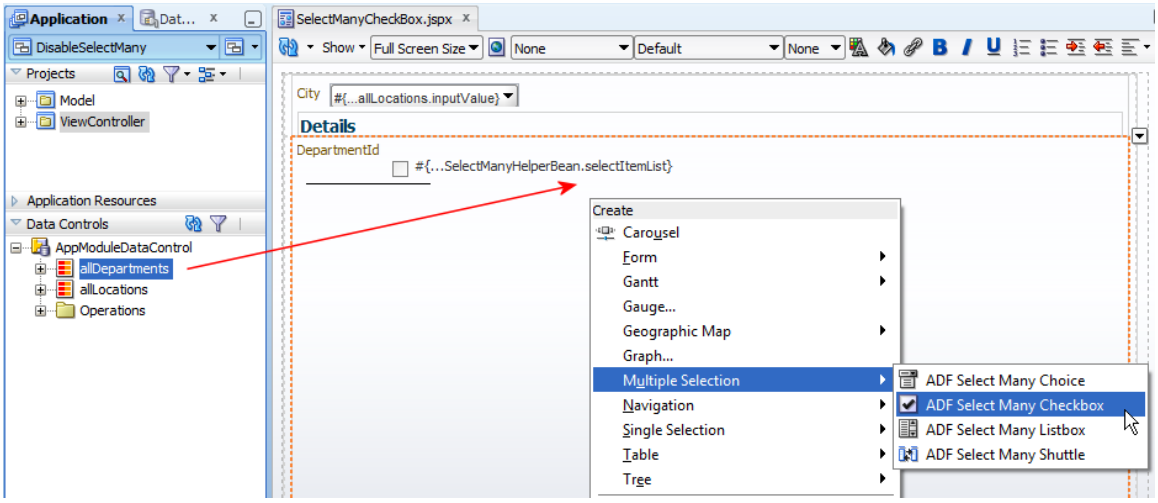


View

The view is built by dragging the **allDepartments** and **allLocations** collections from the data control panel to the view. The **allLocations** collection is added as an ADF Navigation List. Selecting a value in the list would navigate the underlying iterator if submitted. In the sample, I am accessing the list directly and not the iterator.



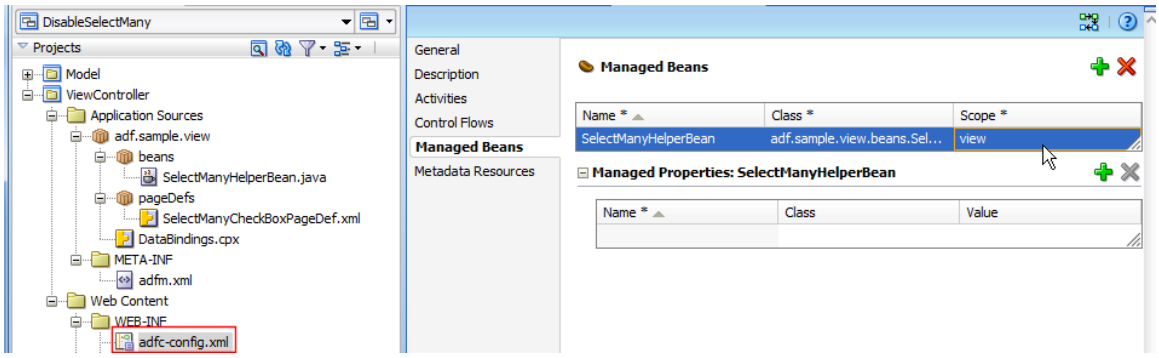
The **allDepartments** collection is dropped as an **ADF Select Many Checkbox** component as per the initial use case on OTN. The PageDef file thus contains two list bindings, one for locations and one for departments.



```
      <af:panelGroupLayout id="pgl2">
        <af:selectOneChoice id="nl1" autoSubmit="true"
                            value="#{bindings.allLocations.inputValue}"
                            label="#{bindings.allLocations.label}">
          <f:selectItems value="#{bindings.allLocations.items}" id="sil"/>
        </af:selectOneChoice>
        <af:panelHeader text="Details" partialTriggers="nl1" id="ph1"/>
      </af:panelGroupLayout>
    </f:facet>
    <f:facet name="center">
      <!-- id="af_one_column_header_scroll"   -->
      <af:panelGroupLayout layout="scroll" id="pgl1">
        <af:selectManyCheckbox value="#{bindings.allDepartments.inputValue}"
                               label="#{bindings.allDepartments.label}"
                               id="smc1" partialTriggers="nl1">
          <f:selectItems value="#{viewScope.SelectManyHelperBean.selectItemList}" id="si2"/>
        </af:selectManyCheckbox>
      </af:panelGroupLayout>
```

The only change that is required in the view to get the use case working is to replace the **f:selectItems** value property to reference a managed bean property returning the list items instead of the binding layer.

Also note that the locations list (af:selectOneChoice) component has its **autosubmit** property set to true so that a change in the list selection causes a submit event that triggers the partial refresh configured on the af:selectManyCheckbox.



The managed bean is configured in view scope to avoid unnecessary data queries during partial refreshes and requests.

The managed bean – and I documented the code using code comments – accesses the ADF binding layer to read the list of all departments. It then gets the selected city code (LocationId) to compare with the list of departments and to enable or disable the `SelectItem` instances returned to the **f:selectItems** tag in the **af:selectManyCheckbox** component. The only know-how that was needed is to not change the list of departments returned by the ADF binding layer with the enabled/disabled state but to create a copy of the list.

However, have a look yourself.

```
import java.util.ArrayList;
import java.util.List;

import javax.el.ELContext;
import javax.el.ExpressionFactory;
import javax.el.ValueExpression;

import javax.faces.context.FacesContext;
import javax.faces.model.SelectItem;

import oracle.adf.model.BindingContext;
import oracle.binding.BindingContainer;

import oracle.jbo.Row;
import oracle.jbo.domain.Number;
import oracle.jbo.uicli.binding.JUCtrlListBinding;


/**
 * Managed bean in view scope to reduce the refresh rate for the list
 * data. A bean in request scope would fetch the data on each partial
 * reresh
 */
 public class SelectManyHelperBean {
   //original list item queried from the ADF binding layer
   List<SelectItem> selectItemList = null;
   //select list with enabled/disabled item state rendered on view
   ArrayList<SelectItem> newSelectItemList = null;
   //to avoid unnecessary refreshes, we check if the selected Location
   //in the location list item has changed. If not then we don't
   //process the deparments list

   Object currentLocationId = null;

   public SelectManyHelperBean() {
        super();
   }

  public void setSelectItemList(List<SelectItem> selectItemList) {}

  /**
   * Queries the list data from the ADF binding layer and returns the
   * list with disabled set to true or false dependent on whether
   * displayed item is in location or not
   * @return List<SelectItem>
  */

    public List<SelectItem> getSelectItemList() {
```

```
//Create Value Expression to access the ADF list for the
//departments

//select many checkbox

FacesContext fctx = FacesContext.getCurrentInstance();

ELContext elctx = fctx.getELContext();

ExpressionFactory exprfact =
   fctx.getApplication().getExpressionFactory();

ValueExpression vexpr = exprfact.createValueExpression(
                           elctx,
                           "#{bindings.allDepartments.items}",
                           Object.class);
selectItemList = (List<SelectItem>) vexpr.getValue(elctx);

/*
 * START: Disable all departments that are not in the selected
 * location
 */

BindingContext bctx = BindingContext.getCurrent();
BindingContainer bindings = bctx.getCurrentBindingsEntry();

//get access to the Locations list to determine the current
//location

JUCtrlListBinding locationListBinding =
              (JUCtrlListBinding) bindings.get("allLocations");

Row selectedListRow = locationListBinding.getCurrentRow();
//get the location ID value to compare with the Departments
//list item LocationId
Object locationId = selectedListRow.getAttribute("LocationId");

//only refresh list if location id has changed
if(locationId != currentLocationId){
  //access the department list items to display in the select
  //many checkbox choice
  JUCtrlListBinding deptListBinding =
          (JUCtrlListBinding) bindings.get("allDepartments");

 //we need to copy the list items into a new list and return
 //this to  the select many component as the list read from the
 //ADF binding layer appears to be immutable

 newSelectItemList = new ArrayList<SelectItem>();
```

```
        //for each item in the list of departments, check if the
        //locationId is the same as the selected locationId. If so,
        //enable check box
       for(SelectItem li : selectItemList){
        //the departments list queried from the ADF binding layer only
        //returns a list of indx that we need to resolve to determine
        //the department select items LocationId
        Integer listIndex = (Integer) li.getValue();
        Row deptListRow =
             deptListBinding.getRowAtRangeIndex(listIndex.intValue());
        //create a new select item to add to the modified list of
        //departments

        SelectItem s = new SelectItem();
        s.setLabel(li.getLabel());
        s.setValue(li.getValue());
       //determine if the item should be enabled or disabled
       if(((Number) deptListRow.getAttribute("LocationId"))
                             .equals((Number)locationId)) {
         s.setDisabled(false);
      }
      else{
         s.setDisabled(true);
      }
      //add item to list
      newSelectItemList.add(s);
    }
   }
   //make sure the current locationId is saved for later comparison.
   //If the location id doesn't change then the list is not refreshed.
   //Remember that the managed bean is in view scope
   locationId = currentLocationId;
   //return the modified list
   return newSelectItemList;
 }
}
```

Sample Download

You can download a JDeveloper 11.1.1.6 workspace with the sample shown in this article as sample #102 from ADF Code Corner

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

You need to configure the dataset connection to point to the HR sample schema of any Oracle database you have in reach.

RELATED DOCOMENTATION

⊠	SelectManyCheckbox component tag http://docs.oracle.com/cd/E23943_01/apirefs.1111/e12419/tagdoc/af_selectManyCheckbox.html
⊠	
⊠	