ADF Code Corner

64. How-to implement a Select Many Shuttle with pre- selected values

twitter.com/adfcodecorner

Abstract:

The ADF binding layer currently only supports a single current row which works for single select lists updating he model. For multi select lists, developers need to manually perform the model update for user selected values.

In this article, we show how to build an ADF bound Select Many Shuttle component that displays employees within a selected department as pre-selected values, allowing users to relocate other employees from the list of all employees to this department.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
23-NOV-2010

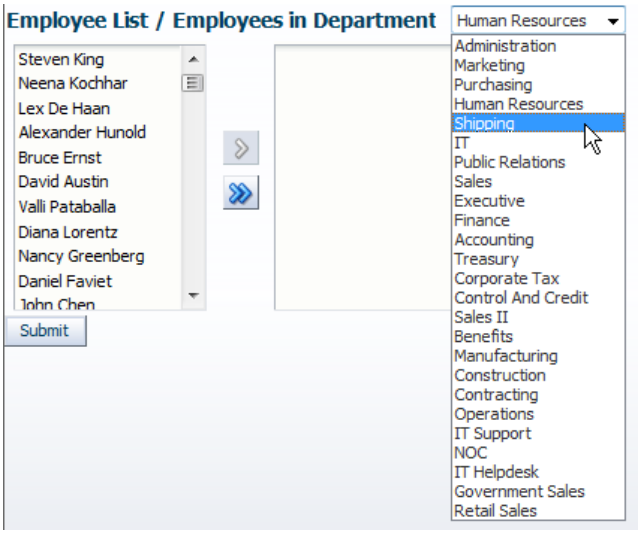
Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

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

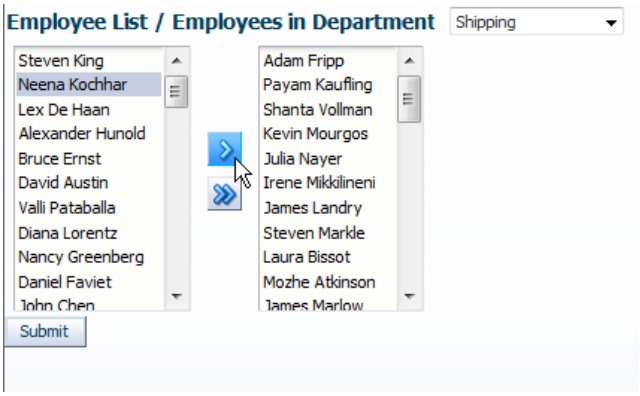
Introduction

The sample provided with this ADF Code Corner article shows all employees of the Oracle HR database sample schema on the left hand side of an ADF Faces Select Many Shuttle component. Users select a department name from a navigation list to choose a department to which selected employees should be relocated. If the department contains employees, then this employees are added to the list of selected employees. Note that skinning is used to only allow the relocation of employees to a new department.The remove list buttons are hidden as it is not allowed to have an employee that is not assigned to a department.



The image below shows how the selected employee list is pre-populated with employees already associated with a selected department.

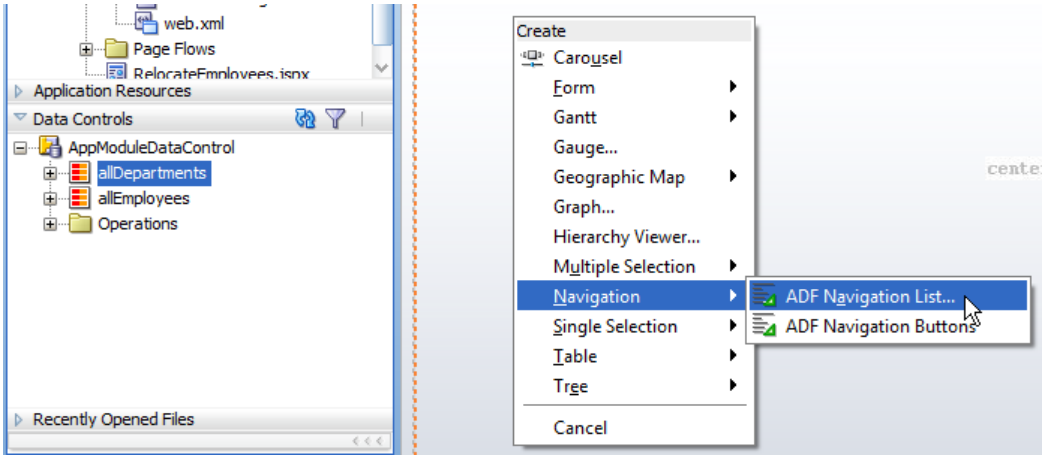
Adding another employee to the selected values list and submitting the form invokes a value change listener that updates the selected employee's department Id.



Building the Sample

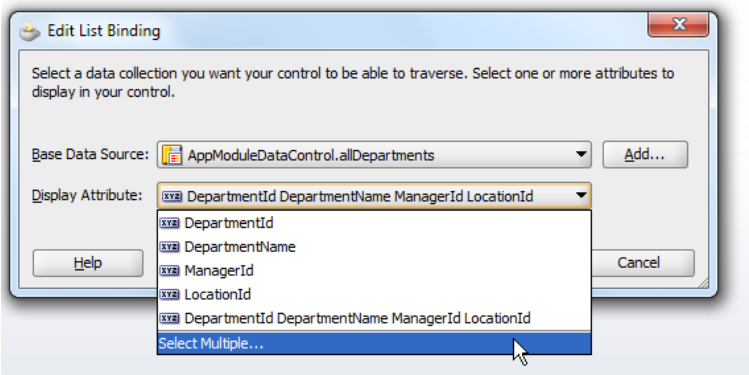
The ADF Business Component model exposes three View Object instances. The "allDepartments" View Object instance lists all departments of the HR schema. It has a dependent View Object instance that shows employees associated with the current department. The "allEmployees" View Object instance lists all employees in the HR schema.

To build the sample shown in the image above, we start with the department list, which is implemented as an ADF Navigation list. This way, the dependent View Object detail is also changed when users select a new department.

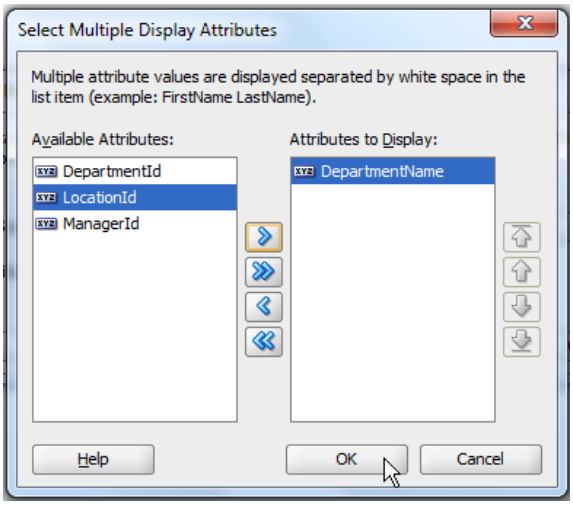


The Base Data Source is automatically set to the "allDepartments" View Object instance that is dragged and dropped from the Data Controls panel, as shown in the image above. Expand the "Display Attribute" select list and scroll to the bottom, to define multiple attributes as the display item.

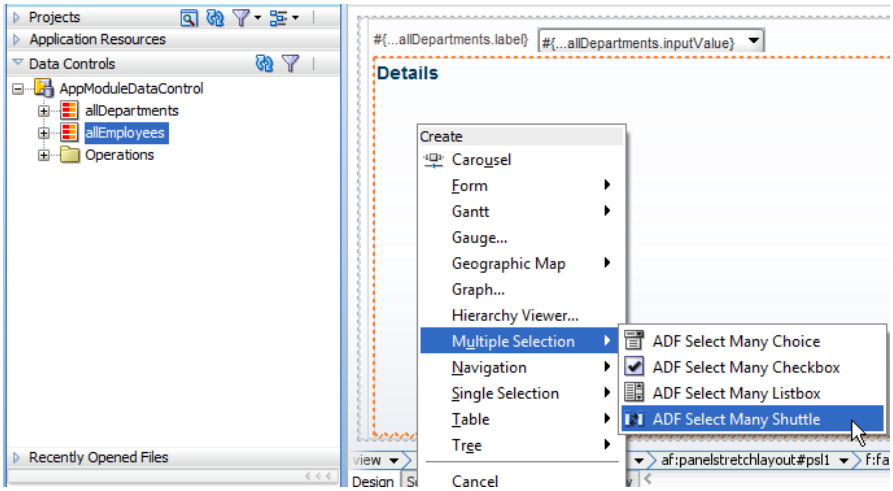
Note: In this sample, only a single display value is used. However, we used the "Select Multiple" entry to make you aware of this option



Choose "DepartmentName" as the display value. The navigation list on the page does two things: First, it defines the department to relocate employees to. Second, it queries the dependent employees View Object instance to show employees that already belong to this department. The latter information is needed to show pre-selected values.

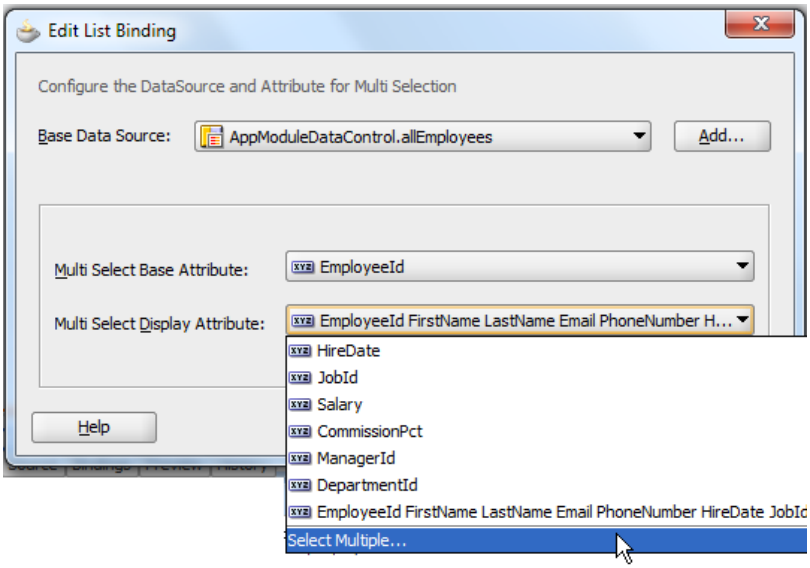


Next, drag and drop the "allEmployees" View Object instance from the Data Controls panel to the page and select the Multiple Selection | ADF Select Many Shuttle.

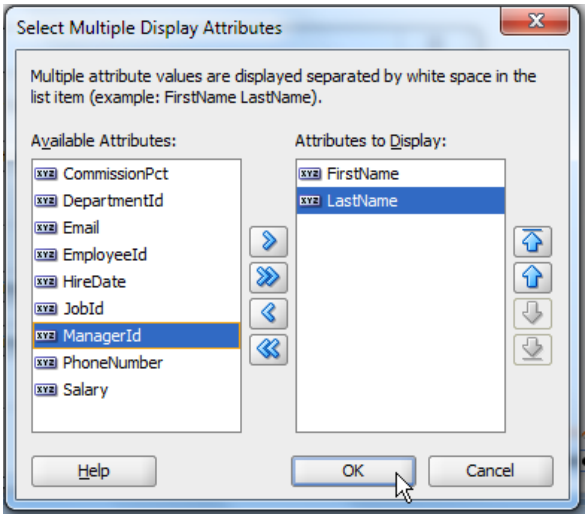


Again, the "Base Data Source" is set to the "allEmployees" View Object instance, represented by an ADF iterator.

In the "Multi Select Display Attribute" list, choose "Select Multi" …



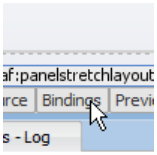
… and define the "FirstName" and "LastName" attributes so that employees are shown with first and last name



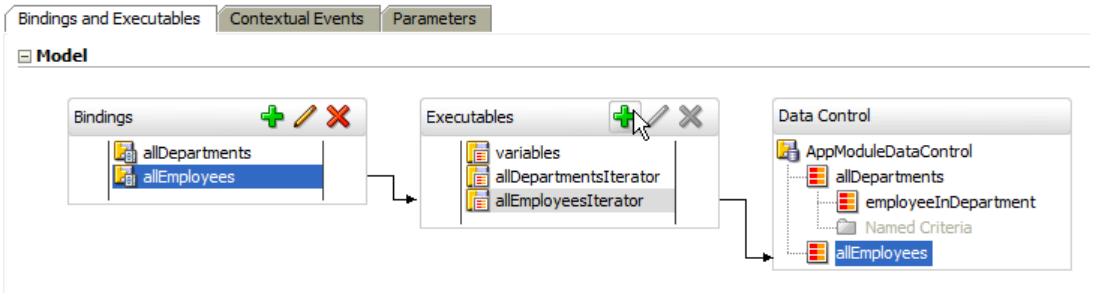
The Select Many Shuttle is created when you "Ok" the dialog. However, to this time the ADF list binding that is created in the PageDef file is used to display the list element and also to track user selections.

To implement the use case of pre-selected values, the Shuttle component value property must point to a managed bean method that returns a list on indices of existing employees in the selected department. To determine the selected values, you need to create an ADF iterator binding for the View Object instance that is dependent on the selected department.

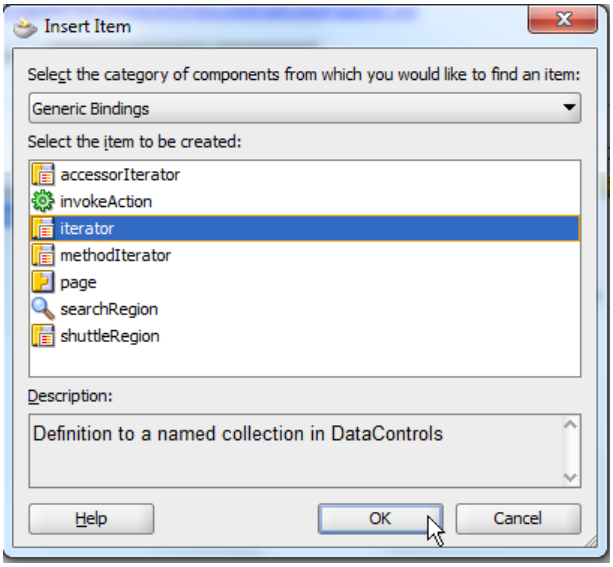
To configure this, select the "Binding" tab at the bottom of the visual page editor, as shown in the image below.



In the page's ADF binding editor, press the green plus icon in the "Executables" section to create a new iterator.

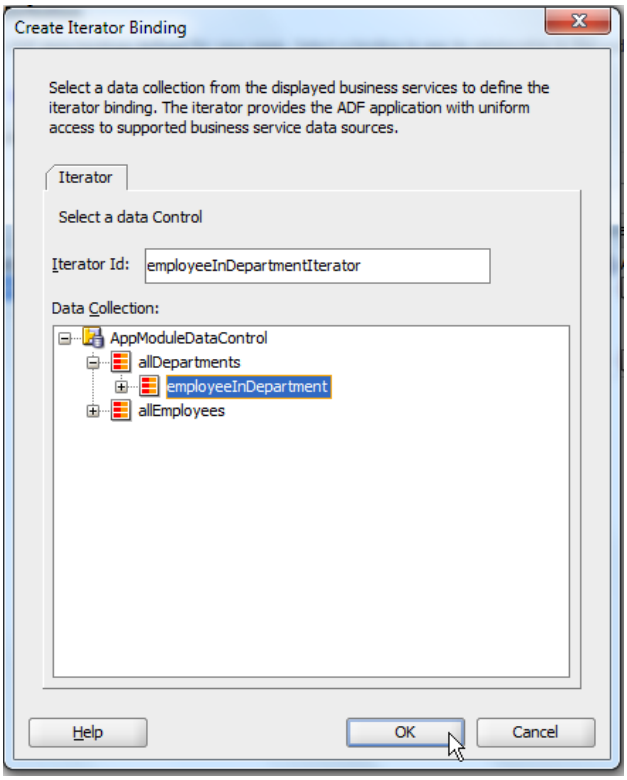


In the opened "Insert Item" dialog, select "iterator" in the "Generic Bindings" category.

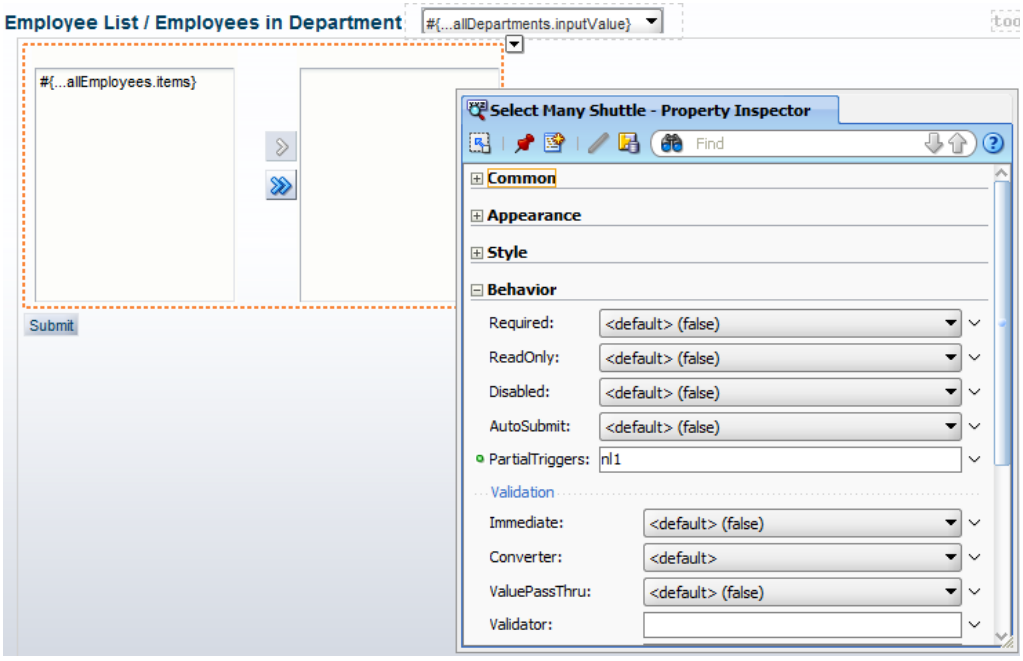


In the list of View Object instances exposed by the Data Control, expand the "allDepartments" node and select the "employeeInDepartment" View Object instance.

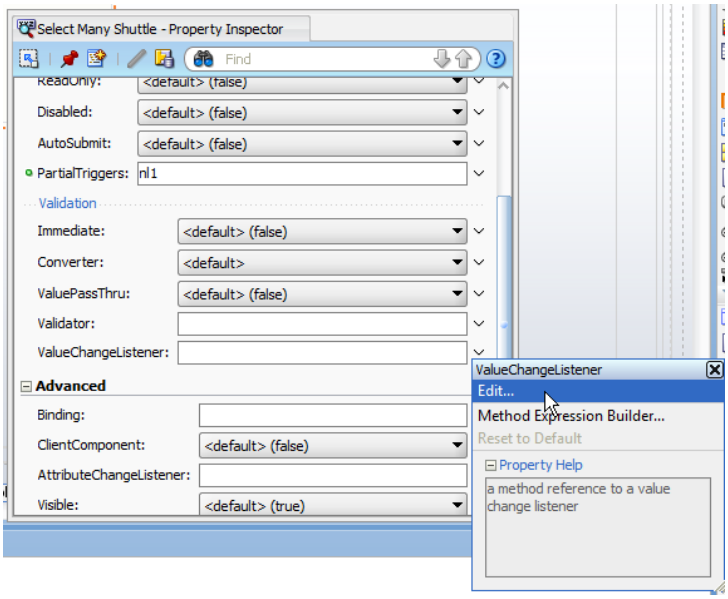
Note: If the business service is not ADF Business Components, then the entries in the ADF Data Control palette are collections (and this also is what View Objects are)



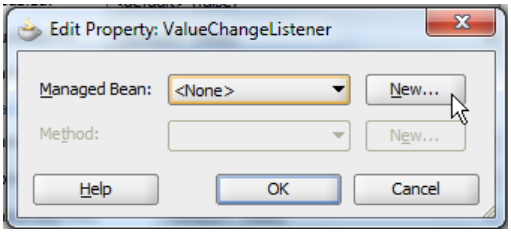
Ok the dialog to create the binding and switch back to the visual page editor. In there, select the Select Many Shuttle component and open the Property Inspector. Browse to the "PartialTriggers" property and click on the arrow icon on its right. From the select opened menu, choose Edit and browse to the list component that shows the departments. This way, a change in the department selection updates the shuttle component using PPR.



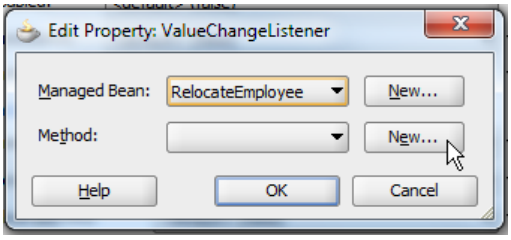
On the same component, select the "ValueChangeListener" property and click the "arrow icon" on the right to open the context menu.



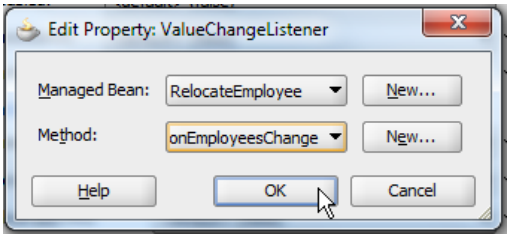
Click the "Edit" menu option to open the "ValueChangeListener" creation dialog. Using this dialog, you can create a new managed bean – or select an existing bean – and create the value change handler method.



Create a new managed bean with the name RelocateEmployee and define it to be in Request Scope.



When the managed bean is created, press "New" for the handler method. Name the new method "onEmployeeChange".



The managed bean shall contain the logic to query the list of existing employees for a department, populate the Select Many Shuttle value property with an array of "Integer" to identify selected value indices. Note however that the indices need to be read from the "allEmployees" iterator.

For this, theRangeSize of the allEmployeesIterator that is created when building the Select Many Shuttle must be set to -1. Otherwise not all employees would be displayed in the Select Many Shuttle, which will lead to some employees not being found for the selected department.

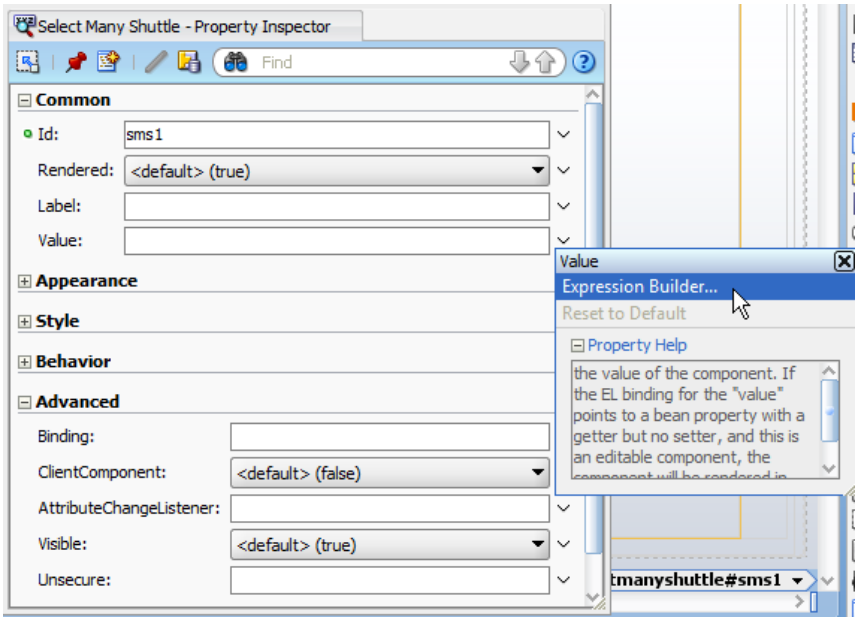
The manage bean code is shown below:

```
import java.util.ArrayList;
import javax.faces.event.ValueChangeEvent;
import oracle.adf.model.BindingContext;
import oracle.adf.model.binding.DCIteratorBinding;
import oracle.binding.BindingContainer;
import oracle.jbo.Row;
import oracle.jbo.RowSetIterator;
import oracle.jbo.domain.Number;
import oracle.jbo.uicli.binding.JUCtrlListBinding;

public class RelocateEmployee {
  private Integer[] selectedIndxs;
  private ArrayList<Integer> selectedIndxList;

  public RelocateEmployee() {
  }

  public void setSelectedIndxs(Integer[] selectedIndxs) {
    this.selectedIndxs = selectedIndxs;
  }

  //method referenced from the value property of the Select
  //Many Shuttle component. Note that a related setter method is
  //contained in this managed bean as well
  public Integer[] getSelectedIndxs() {
    selectedIndxs = null;
    selectedIndxList = new ArrayList<Integer>();
    BindingContext bctx = BindingContext.getCurrent();
    BindingContainer bindings = bctx.getCurrentBindingsEntry();
    //get all employees for the current selected department
    DCIteratorBinding employeesInDepartmentDciter =
       (DCIteratorBinding)bindings.get("employeeInDepartmentIterator");
    RowSetIterator rsi =
                 employeesInDepartmentDciter.getRowSetIterator();
    //get access to the "allEmployees" list binding
    JUCtrlListBinding allEmployeesList  =
                  (JUCtrlListBinding)bindings.get("allEmployees");
    DCIteratorBinding allEmployeesIter =
                          allEmployeesList.getDCIteratorBinding();
```

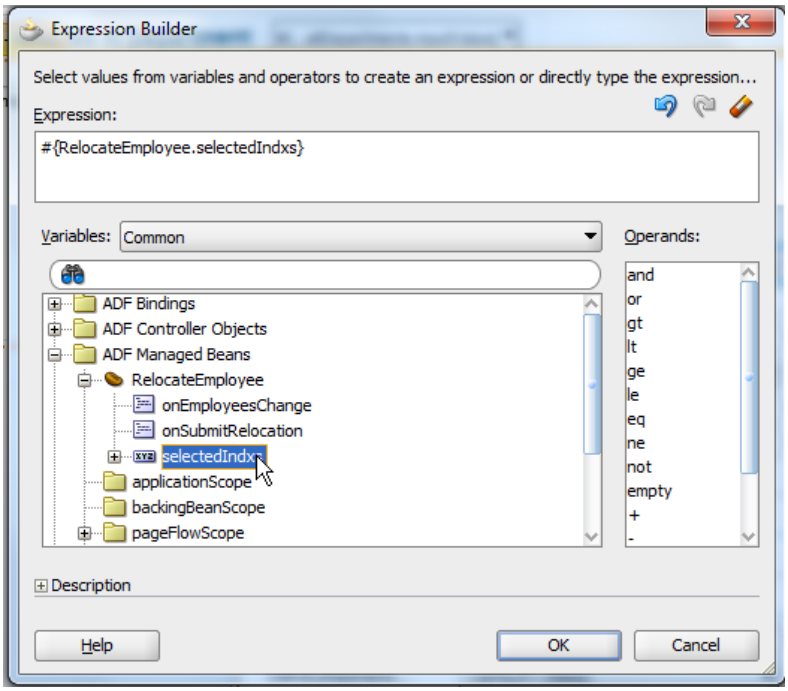
```
  while(rsi.hasNext()){
    //iterate over the employees in the selected department and find
    //the employee record index in the allEmployees iterator. Note
    //that the employee in the allEmployees iterator has a different
    //index number than the same employee record in the department's
    //dependent View Object
    Row rw = (Row) rsi.next();
    allEmployeesIter.setCurrentRowWithKey(rw.getKey()
                                          .toStringFormat(true));
    int indx = allEmployeesIter.getCurrentRowIndexInRange();
    selectedIndxList.add(indx);
  }
  selectedIndxs = selectedIndxList.toArray(new
  Integer[selectedIndxList.size()]);
  return selectedIndxs;
}

//to-do in this method: get the DepartmentsId and update the
//employees added to the department
public void onEmployeesChange(ValueChangeEvent valueChangeEvent) {
  //get all items left as selected after a change
  Integer[] employeesInDepartment =
                        (Integer[]) valueChangeEvent.getNewValue();
//get the DepartmentId
BindingContext bctx = BindingContext.getCurrent();
BindingContainer bindings = bctx.getCurrentBindingsEntry();
DCIteratorBinding allDepartmentsIterator =
          (DCIteratorBinding)bindings.get("allDepartmentsIterator");
//get current row
Row currentRow = allDepartmentsIterator.getCurrentRow();
Number departmentId =
                  (Number) currentRow.getAttribute("DepartmentId");

//compare the selected values in the list and update all rows that
//don't yet have the department Id of the selected department

JUCtrlListBinding allEmployeesList  =
                    (JUCtrlListBinding)bindings.get("allEmployees");
DCIteratorBinding allEmployeesIter =
                            allEmployeesList.getDCIteratorBinding();

if (employeesInDepartment.length > 0) {
  for(int index : employeesInDepartment){
     Row rowToUpdate = allEmployeesIter.getRowAtRangeIndex(index);

     //check if the department Id of the selected employee is the
     //same as the selected department. If not, update the selected
```

```
        //employee record
        if (!((Number)rowToUpdate.getAttribute("DepartmentId"))
              .equals(departmentId)){
            rowToUpdate.setAttribute("DepartmentId", departmentId);
         }
      }
      //Note that this sample does not commit the changes. I do this on
      //purpose not to mess with your database.
    }
  }
}
```

After creating the managed bean, go back to the Select Many Shuttle component and choose the "Value" property. Click the arrow icon on the right and choose "Expression Builder" from the context menu.



Select the "selectedIndxs" entry of the RelocateEmployees managed bean, as shown in the image below.

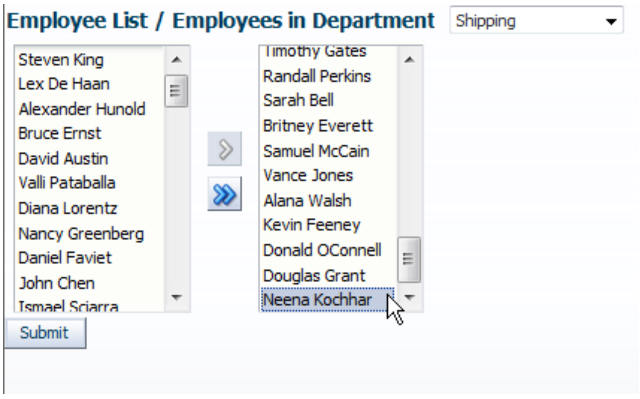


The Select Many Shuttle component's value property is now connected to the managed bean's list of selected employee record indices. Adding employees and submitting the form invokes the value change event handler method shown in the managed bean code above. The value change event newValue entry contains all values selected in the list as an array of Integer. The array list represents indices of employee records in the allEmployees iterator.

Skinning

To hide the shuttle buttons to remove selected values, the following skin definition is used

```
af|selectManyShuttle::remove-horizontal,
af|selectManyShuttle::remove-vertical,
af|selectManyShuttle::removeall-horizontal,
af|selectManyShuttle::removeall-vertical{display:none}
```



Download and Run the Sample

The sample described in this article can be downloaded from the ADF Code Corner website as sample 64.

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

Download the Oracle JDeveloper 11.1.1.3 workspace and configure the database connection (View | Database | Database Connection) to point to a HR schema in a local database of yours. Run the JSPX page, select a department and add employees to it. Press the submit button to persist the change. Change the department in the list and then go back to the department you updated. Just to prove the employee you added now shows in the list of selected values.

Note: Changes are not committed in the sample as it is not good practice for samples to mess with customer databases-

RELATED DOCOMENTATION

☒	Skinning in the Oracle Fusion Web Developer Product Documentation - http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/af_skin.htm#BAJFEFCJ
☒	Skinning in chapter 16 of the Fusion Developer Guide book - http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543
☒	