ADF Code Corner

91. How-to create new lookup data from a list of values select list

twitter.com/adfcodecorner

Abstract:

A requirement voiced on OTN was to have a list of values in form of a select list that provides an option for users to create new lookup entries if the list does not contain what they are looking for. Using model driven list of values and the `af:inputComboboxListOfValues` component, this use case is straight forward to implement, as shown in this article.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
04-OCT-2011

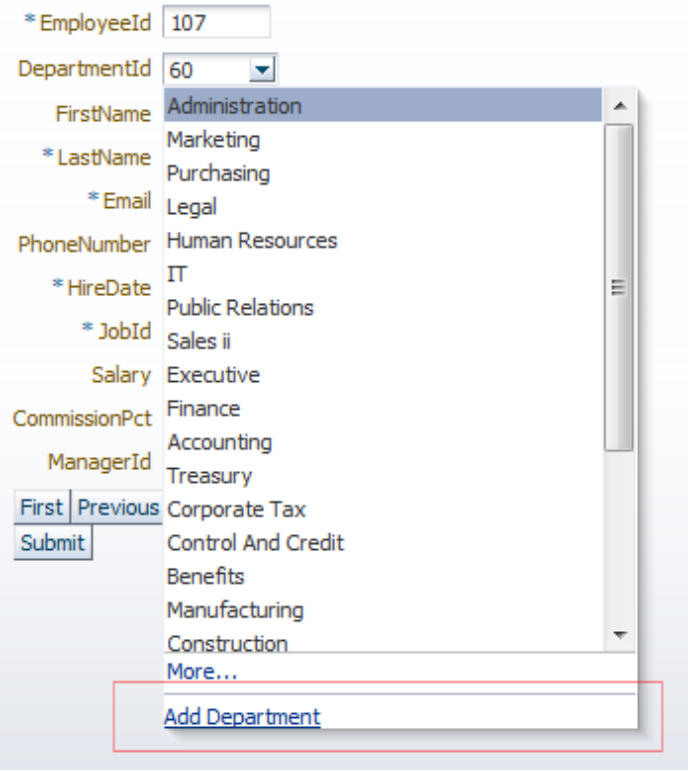
Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

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.

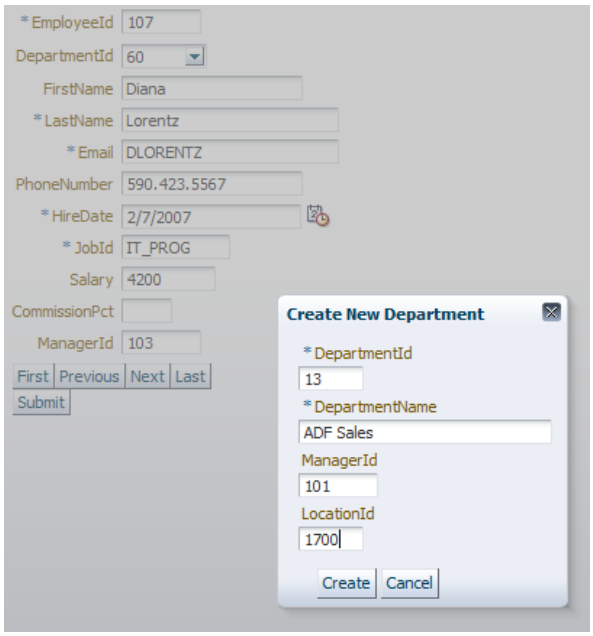
Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

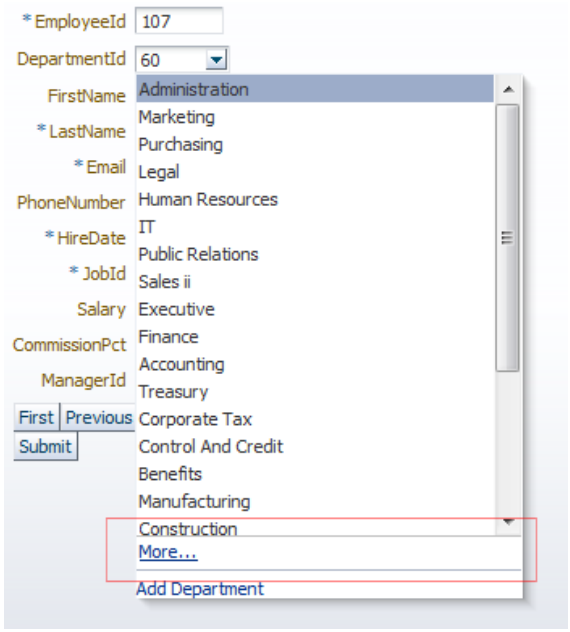
A model driven list of values defined for the **DepartmentId** attribute of the EmployeesView view object is rendered as an `af:inputComboboxListOfValues`. Beside of a default link **More** that allows users to query the for more entries then the initially defined list size (which I set to 25), a command link is provided for users to create a new department and add it to the select list.



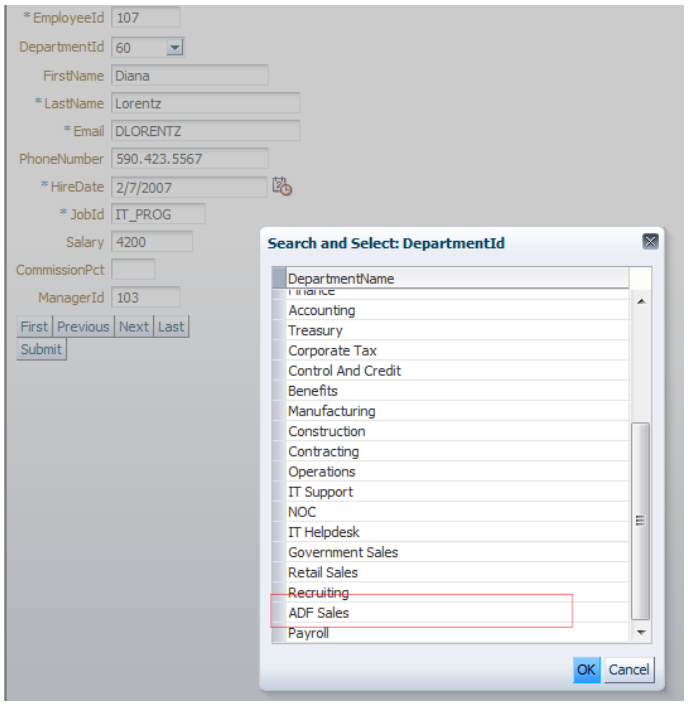
When the **Add department** link is clicked, a dialog opens for the use to define the new department.



Pressing the **Create** button will create the new department and commit the change to the database. Pressing **Cancel** will undo the row creation and return to the form.



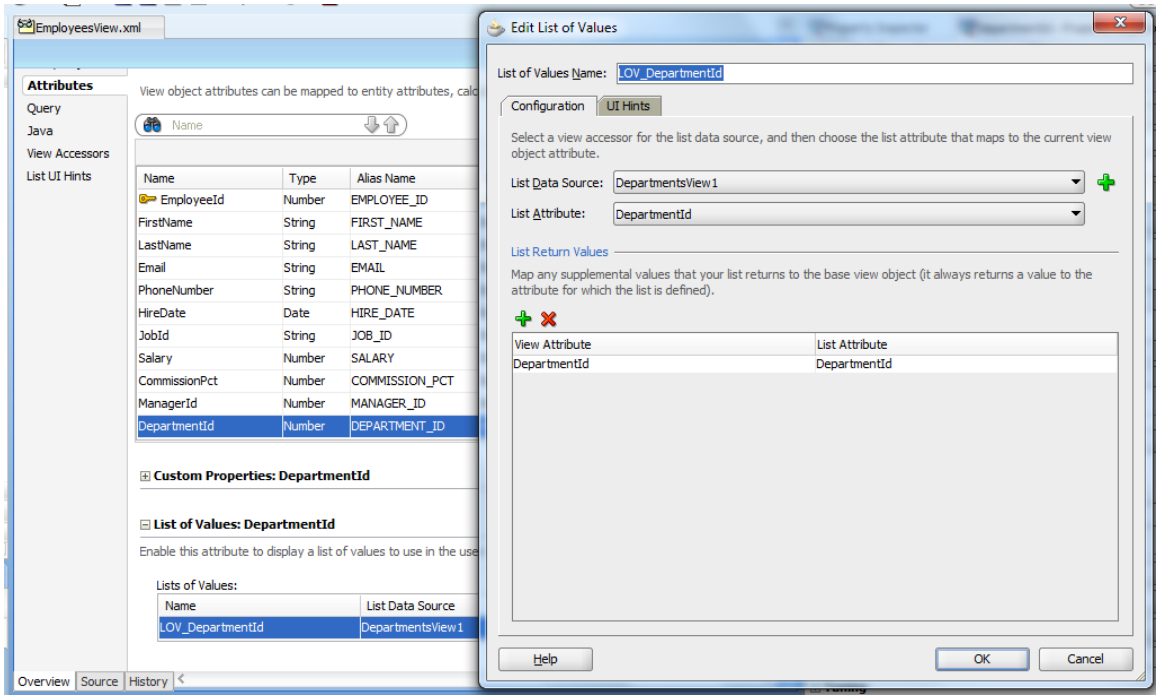
Dependent on where the new record is added, it could be that it doesn't show in the initial list of select values. In this case users would press the **More** button to select the newly created value. Of course, you could disable the list size restriction when creating the model driven task flow, so that all values show at once in the select list.



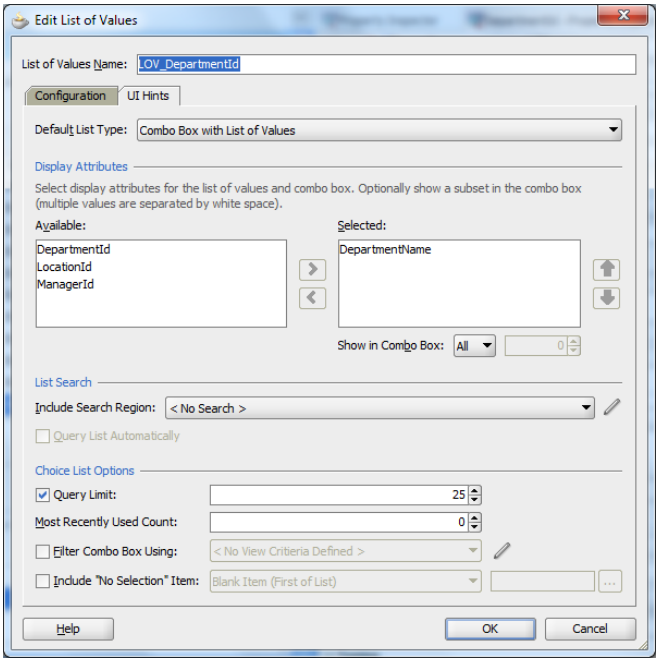
As you can see, the new value has been created.

How-to Implement

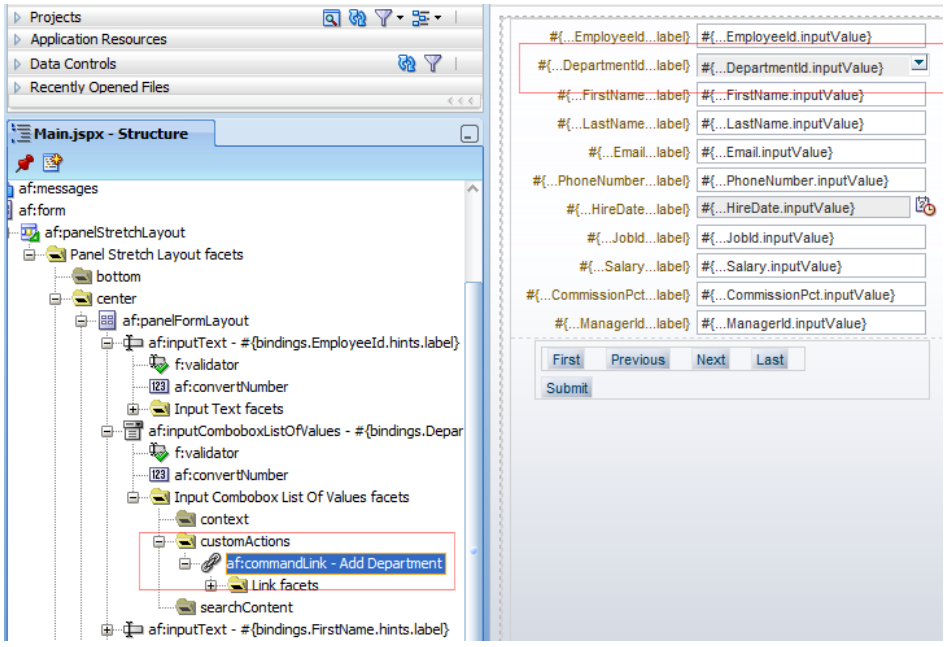
As mentioned earlier, the list is a model driven list of values built for the **DepartmentId** attribute of the Employees table. The DepartmentsView view object is referenced to query the list data.



In the **UI Hints** section of the model driven LOV creation dialog, the **default List Type** is set to **Combo Box with List of values** and the **Query Limit** is set to 25. If you unselect the check box, then no query limit is applied to the initial select list.

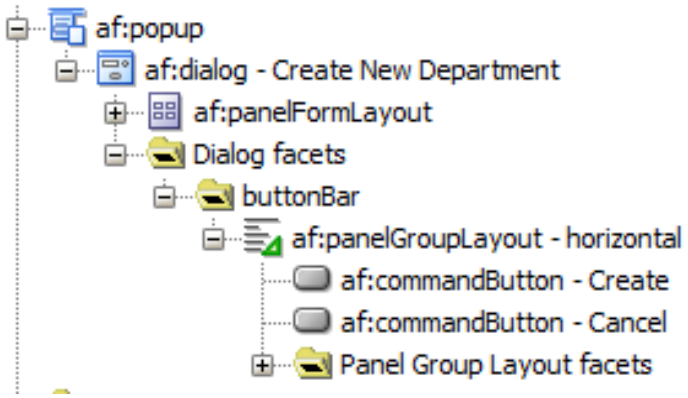


The **DepartmentId** is rendered using the `af:inputComboboxListOfValues` component tag. The `af:inputComboboxListOfValues` tag has a **customActions** facet that can be used to add links to the end of the select list. In this example, an `af:commandlink` is added to create a new row for the DepartmentsView view object and open the popup dialog.

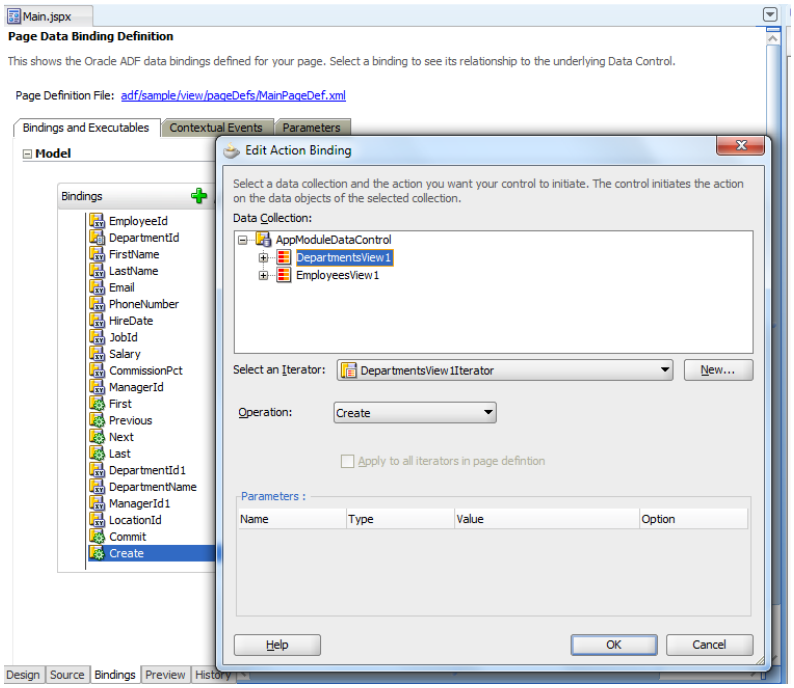


The popup is defined on the same view using the `af:popup` and `af:dialog` components.

Two buttons are added to the `af:dialog` **buttonBar** facet. The `af:dialog` **type property** is set to **none** so that the dialog doesn't show its native ok and cancel buttons.



Note: For creating a new row in the DepartmentsView list, the PageDef file of the view has been manually updated with a method binding that reference the Create operation of the DepartentsView.



The following source code shows the content of the managed bean referenced by the **Add Department** command link, and the two command buttons of the dialog.

```
import oracle.adf.model.BindingContext;
import oracle.adf.model.binding.DCIteratorBinding;
import oracle.adf.view.rich.component.rich.RichPopup;
import oracle.adf.view.rich.component.rich.layout.RichPanelFormLayout;
import oracle.adf.view.rich.context.AdfFacesContext;
import oracle.binding.BindingContainer;
import oracle.binding.OperationBinding;
```

```
public class ComoboxBean {
   //JSF component binding for PanelFormlayout and RichPopup
   private RichPanelFormLayout inputFormLayoutPanel;
   private RichPopup createDepartmentPopup;

  public ComoboxBean() {}

  //method invoked by the Create Button in the department create
  //popup dialog
  public String createNewEntry() {
     BindingContainer bindings = this.getBindings();
     OperationBinding commit =
           (OperationBinding) bindings.get("Commit");
     //adding a list option only makes sense if the change is
     //committed
     commit.execute();
     //refresh form so list is updated in view
     AdfFacesContext adfFacesCtx =
                  AdfFacesContext.getCurrentInstance();
     adfFacesCtx.addPartialTarget(inputFormLayoutPanel);
     return null;
  }

  //method called from the cancel button in the create department
  //dialog
  public String cancelCreation() {
     BindingContainer bindings = this.getBindings();
     DCIteratorBinding dciterator =
         (DCIteratorBinding) bindings.get("DepartmentsView1Iterator");
     //undo new row creation
     dciterator.getCurrentRow().remove();
     return null;
  }

  //helper method to access the bindings container
  private BindingContainer getBindings(){
      BindingContext bctx = BindingContext.getCurrent();
      BindingContainer bindings = bctx.getCurrentBindingsEntry();
      return bindings;
   }

  public void setInputFormLayoutPanel(
          RichPanelFormLayout inputFormLayoutPanel) {
     this.inputFormLayoutPanel = inputFormLayoutPanel;
  }
```

```
  public RichPanelFormLayout getInputFormLayoutPanel() {
        return inputFormLayoutPanel;
  }

  //method called by the Create department link in the list
  public String onCreateNewDepartment() {
    BindingContainer bindings = this.getBindings();
    //method binding in the PageDef file has been manually
    //defined for the Create operation of the DepartmentView
    //collection
    OperationBinding create =
        (OperationBinding) bindings.get("Create");
    create.execute();
    //bring up the popup
    RichPopup.PopupHints hints = new RichPopup.PopupHints();
    createDepartmentPopup.show(hints);
    return null;
  }


   public void setCreateDepartmentPopup(
              RichPopup createDepartmentPopup) {
     this.createDepartmentPopup = createDepartmentPopup;
   }


   public RichPopup getCreateDepartmentPopup() {
     return createDepartmentPopup;
   }
}
```

Download

You can download the JDeveloper 11g 11.1.1.4 sample workspace from the ADF Code Corner website

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

Configure the application database connect to the HR schema of an Oracle database and run Main.jspx from JDeveloper. Open the list for Department ID and create a new department. Reopen the list, search for the new entry and select it.

RELATED DOCOMENTATION

⊠	af:dialog tag document http://download.oracle.com/docs/cd/E21764_01/apirefs.1111/e12419/tagdoc/af_dialog.html
⊠	
⊠	