ADF Code Corner

013. How-to declaratively create new table rows based on existing row content

Abstract:

A frequent requirement posted on the Oracle JDeveloper forum on OTN is to create new rows in a table based on a copy of an existing row. Using the new CreateWithParams operation exposed on the ADF Business Components ViewObject this task becomes fully declarative in Oracle JDeveloper 11. This article provides instructions on how to achieve the goal.

twitter.com/adfcodecorner

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
20-NOV-2008

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.

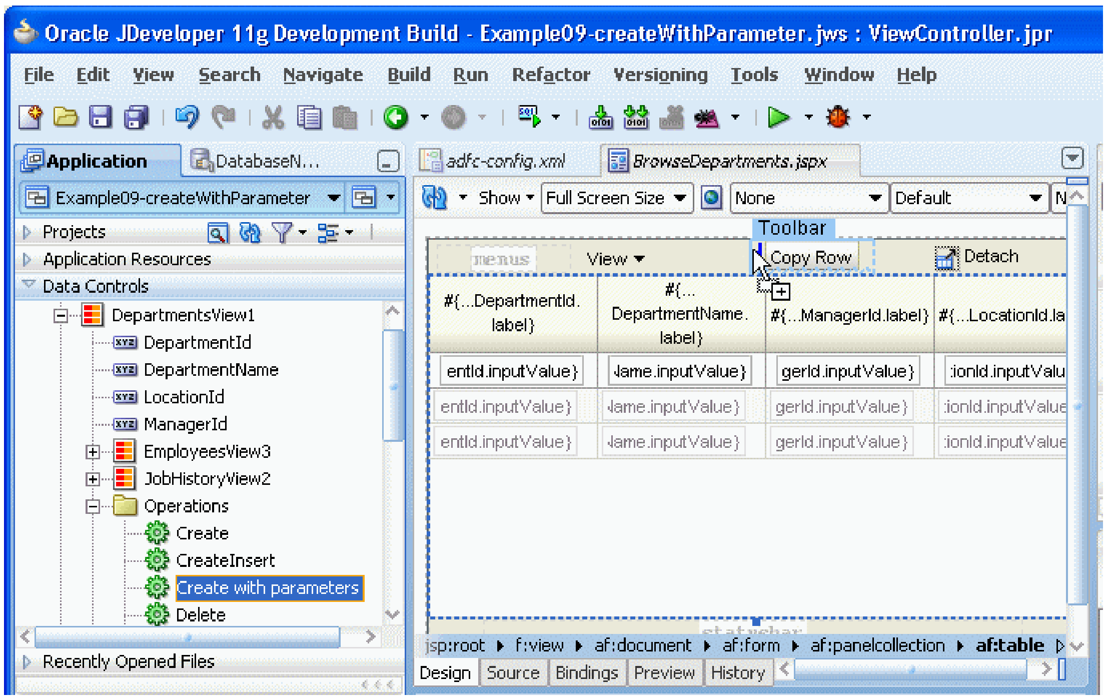
Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

There are many ways to create a new row in an iterator and provision it with the values of an existing row. Its just a matter of what your developer background is and how much you like coding in Java. For those that have a 4GL background, everything that is declarative end-to-end seems to be preferred. Lucky enough, Oracle ADF and ADF Business Components greatly simplify web application development in Java EE and this also includes the usecase mentioned above.

How-to

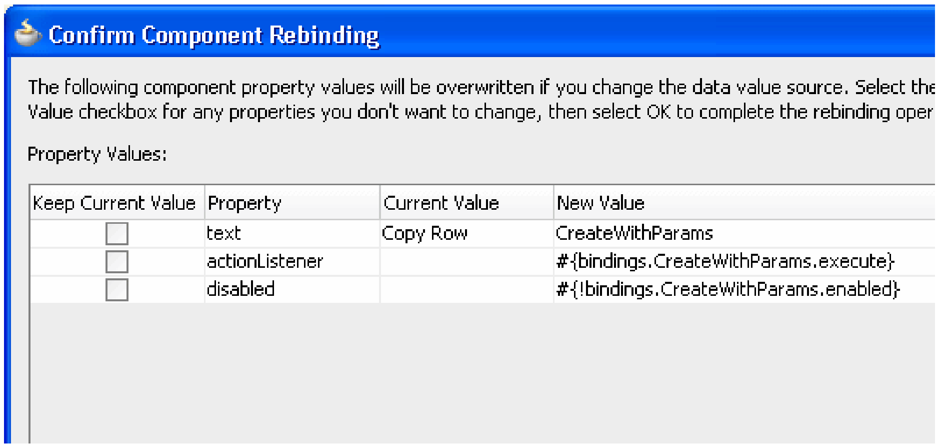
This how-to starts from an existing ADF Faces table, which is created by dragging a ViewObject from the data control palette to the JSF page.



The surrounding `af:panelCollection` component has a toolbar facet to which a a single toolbar button is added to initiate the copy. The button ID is referenced in the table's "partialTrigger" property so that clicking the button initiates a table refresh.

To assign the "Create with parameters" operation to the button, select the "Create with parameters" operation under the ViewObject node and drag it over the toolbar button

In the component rebind dialog, check the "keep current value" checkbox of the "text" property and press **Ok**. This adds an EL expression to the button that references the CreateWithParams operation binding in he page's pagedef file.

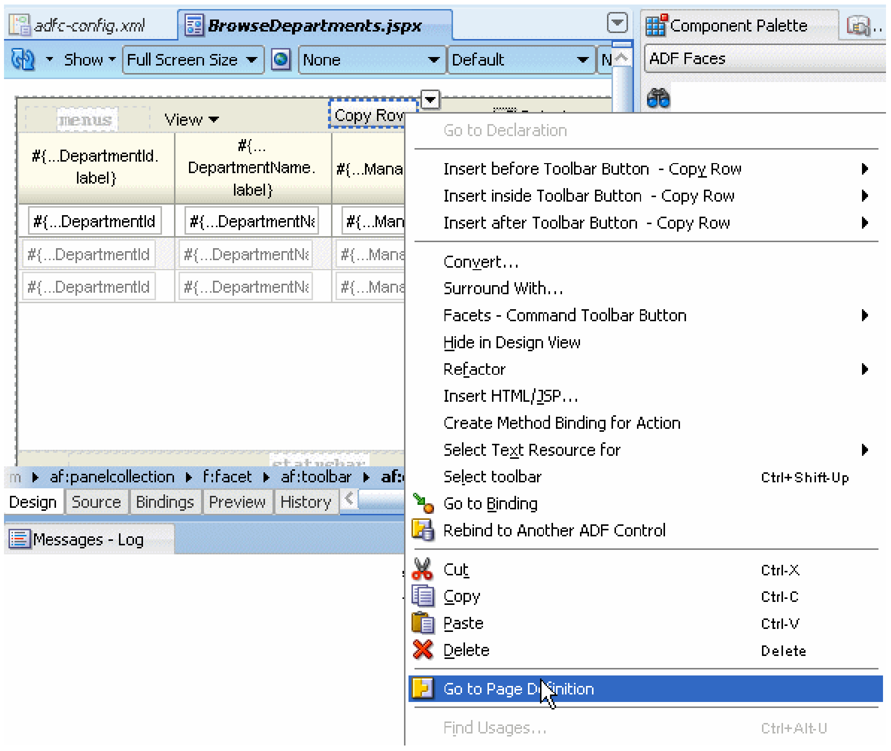

Confirm Component Rebinding

The following component property values will be overwritten if you change the data value source. Select the
Value checkbox for any properties you don't want to change, then select OK to complete the rebinding oper

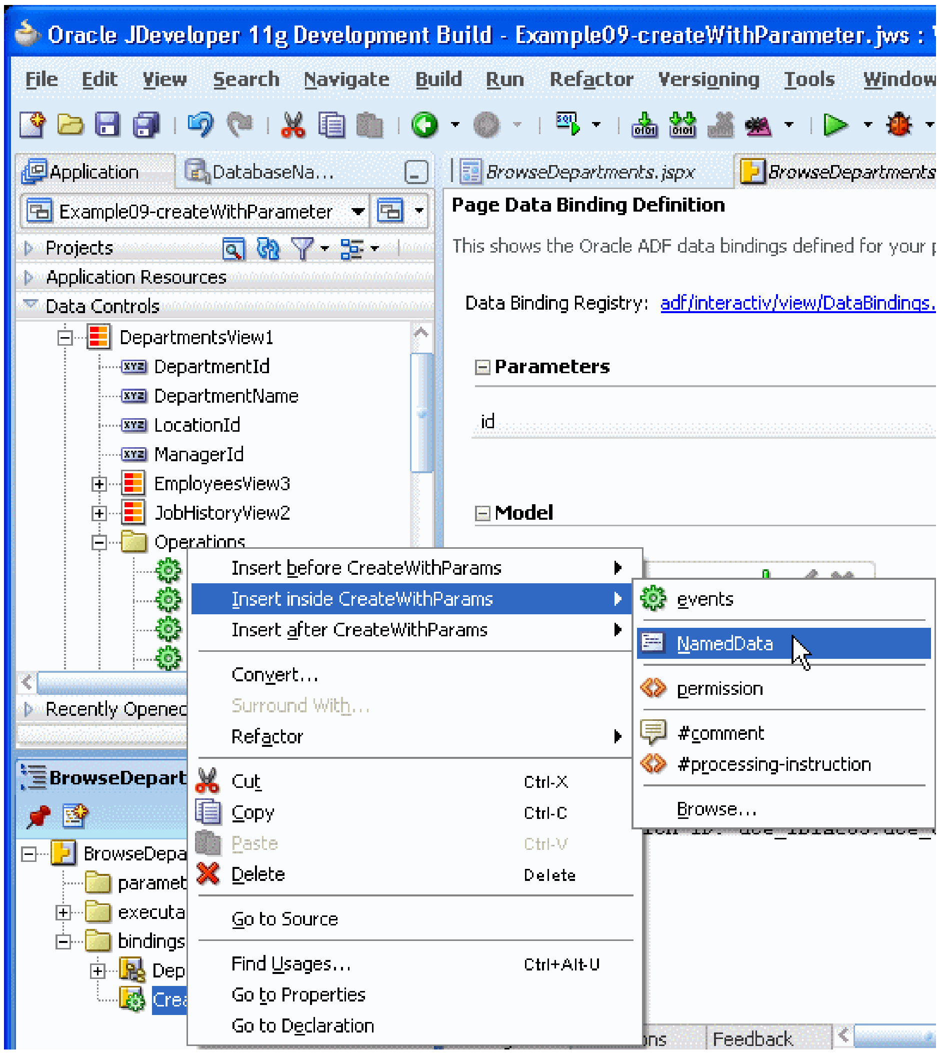
Property Values:

Keep Current Value	Property	Current Value	New Value
☐	text	Copy Row	CreateWithParams
☐	actionListener		#{bindings.CreateWithParams.execute}
☐	disabled		#{!bindings.CreateWithParams.enabled}

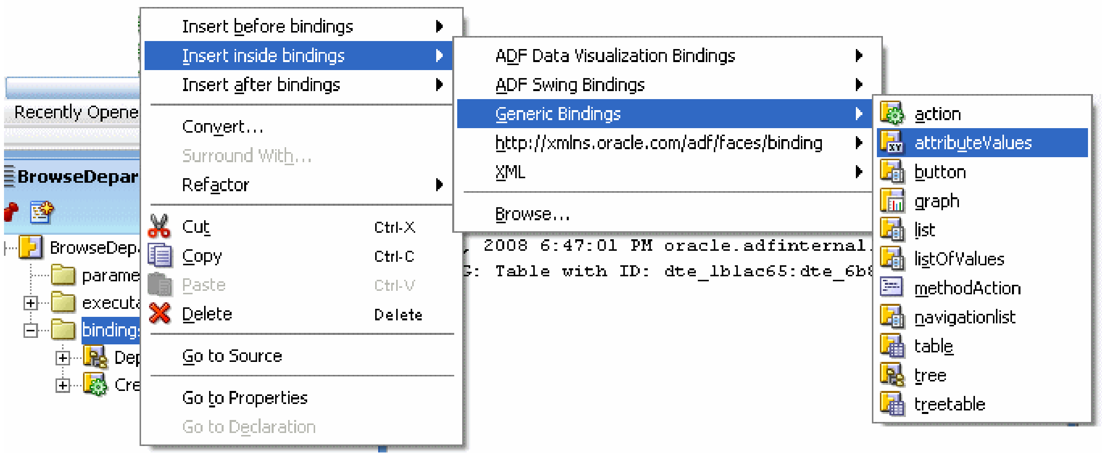
To navigate to the page definition file, for further editing, select the toolbar button and choose "Go to Page definition" from the context menu.



The **pageDef** file has an new entry "CreateWithParams" under its **bindings** node to create a new row. However, it does not yet know which attributes to provision with data when getting invoked. To address this, select **Insert inside CreateWithParams** from the context menu and then click the **NamedData** option.



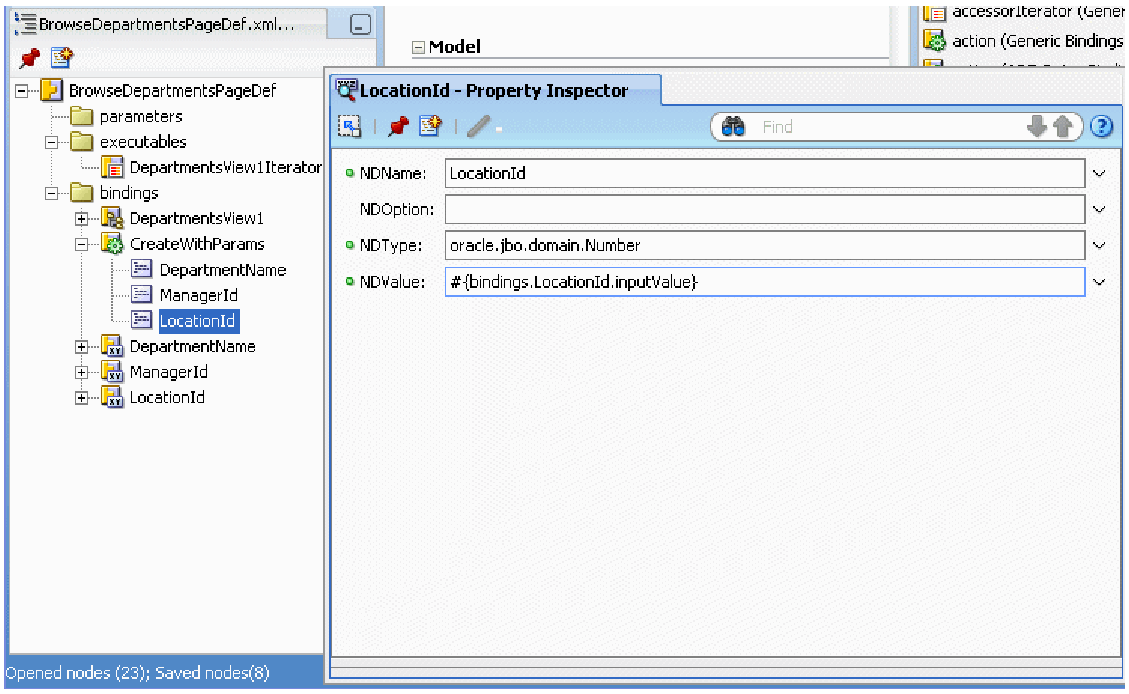
Create a named data item for all attributes that you want to provision data for. In this example, data should be provisioned for DepartmentName, ManagerId and LocationId. The data should be copied from the current selected row in the table. A nice trick in ADF is to create attribute bindings for the attributes that you need to copy the value from. In the image below, you see attribute bindings created for DepartmentName, LocationId and ManagerId. Because ADF synchronizes the selected table row in the UI with the information held in the iterator, the attribute bindings always contain the attribute values of the current selected row, making them EL accessible.



```
, 2008 6:47:01 PM oracle.adfinternal.
G: Table with ID: dte_lblac65:dte_6b
```

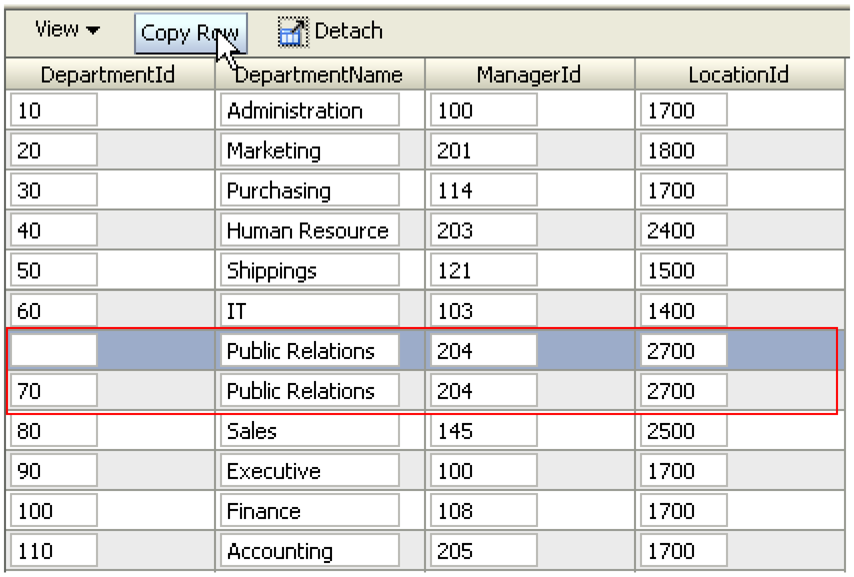
The row attribute values can be copied from the attribute binding to the NDValue property of the **NamedData** item using ExpressionLanguage.

In the example below, the **NamedData** item has a name of LocationId, so the LocationId attribute of the new row gets provisioned, a**NDType** of `oracle.jbo.domain.Number`, which matches the attribute type of the underlying EntityObject and a **NDValue** of #{bindings.LocationId.inputValue} that is referencing the current row's LocationId attribute, exposed by the attribute binding.



NDName:	LocationId
NDOption:	
NDType:	oracle.jbo.domain.Number
NDValue:	#{bindings.LocationId.inputValue}

At runtime, you can now select a table row and press the "Copy Row" button. The button invokes the "CreateWithParams" operation and passes the initial values for the DepartmentName, ManagerId and the LocationId as a copy of the current selected row attributes.



DepartmentId	DepartmentName	ManagerId	LocationId
10	Administration	100	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resource	203	2400
50	Shippings	121	1500
60	IT	103	1400
	Public Relations	204	2700
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700