ADF Code Corner

66. How-to color-highlight the bar in a graph that represents the current row in the collection

Abstract:

An ADF Faces Data Visualization Tools (DVT) graphs can be configured to change the current row in the underlying collection when a user click into it. This allows developers to implement master-detail behavior between a graph as the master and a form or a table as the detail. However, there is no visual indication for the chart item, for example a bar in a bar chart, the user clicked on. Also, navigating the underlying collection and partially refreshing the graph does not highlight the bar representing the current row.

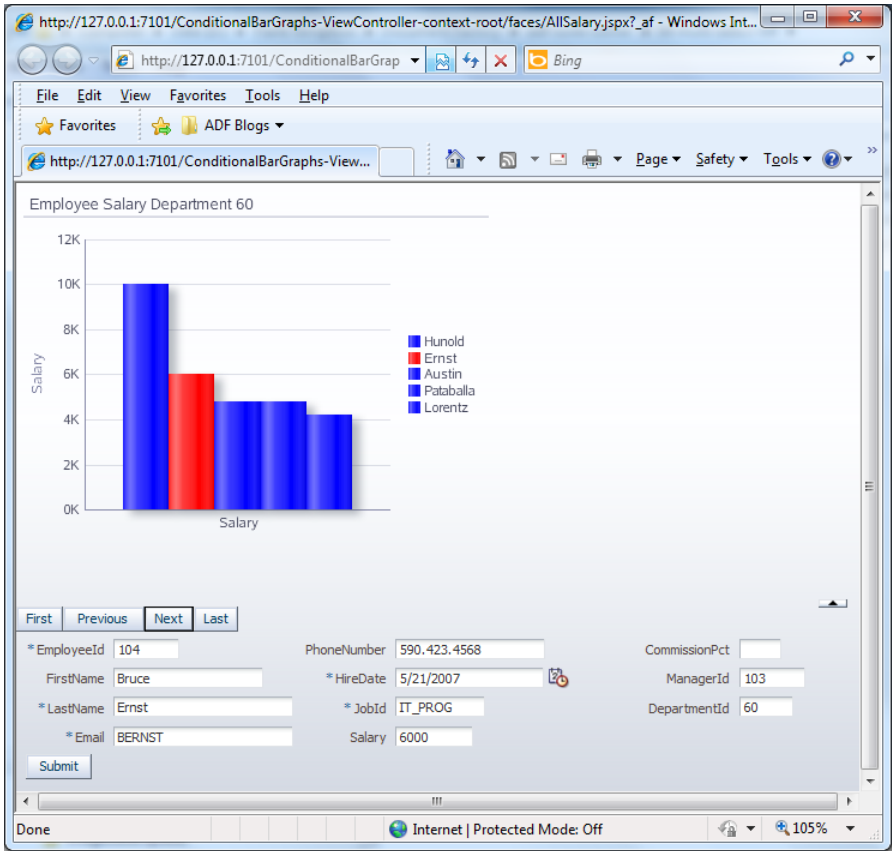
This article explains how to highlight the current row in a bar chart for users to have a visual feedback.

twitter.com/adfcodecorner

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
DD-MON-2010

Introduction

In this sample, a user click on a bar in a graph sets the current row in the underlying ADF iterator. A form that is based on the same iterator is setup to show detail information about the selected bar item.


http://127.0.0.1:7101/ConditionalBarGraphs-ViewController-context-root/faces/AllSalary.jspx?_af - Windows Int...
File Edit View Favorites Tools Help
Employee Salary Department 60
12K
10K
8K
6K
4K
2K
0K
Salary
Salary
Hunold
Ernst
Austin
Pataballa
Lorentz
First Previous Next Last
Submit
Done
Internet | Protected Mode: Off
105%

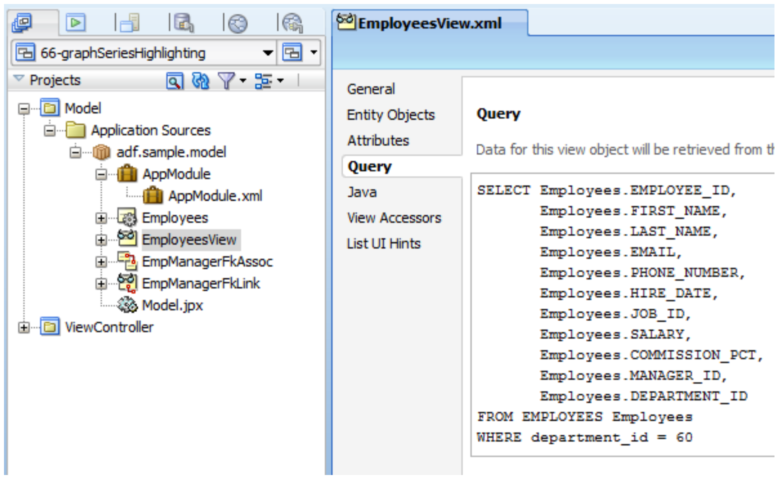
* EmployeeId	104	PhoneNumber	590.423.4568	CommissionPct	
FirstName	Bruce	* HireDate	5/21/2007	ManagerId	103
* LastName	Ernst	* JobId	IT_PROG	DepartmentId	60
* Email	BERNST	Salary	6000		

In addition, a navigation bar on top of the form may be used to scroll within the employee data, which then refreshes the graph, again showing the current employee bar highlighted.

About the ADF Business Component Model

The ADF Business Component model is based on the Employees table of the Oracle HR schema. The View Object query is restricted to return employees in department 60.

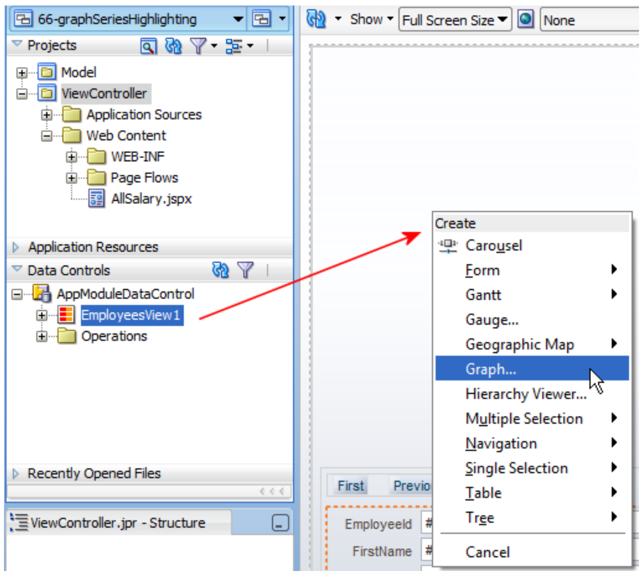
Note The ADF business service is not important for how this sample works and only serves for the purpose of providing sample data for the graph.



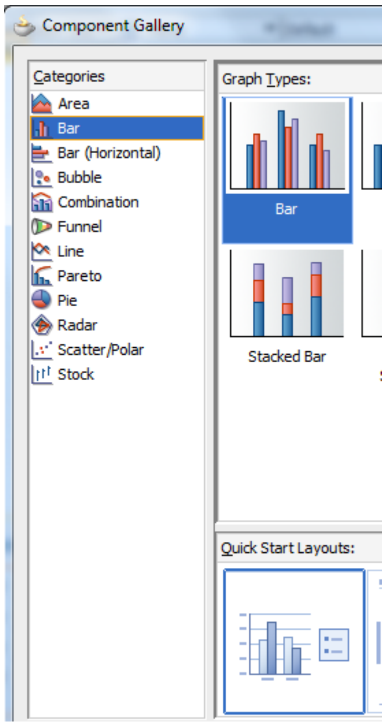
```
SELECT Employees.EMPLOYEE_ID,
       Employees.FIRST_NAME,
       Employees.LAST_NAME,
       Employees.EMAIL,
       Employees.PHONE_NUMBER,
       Employees.HIRE_DATE,
       Employees.JOB_ID,
       Employees.SALARY,
       Employees.COMMISSION_PCT,
       Employees.MANAGER_ID,
       Employees.DEPARTMENT_ID
FROM EMPLOYEES Employees
WHERE department_id = 60
```

Building the Graph

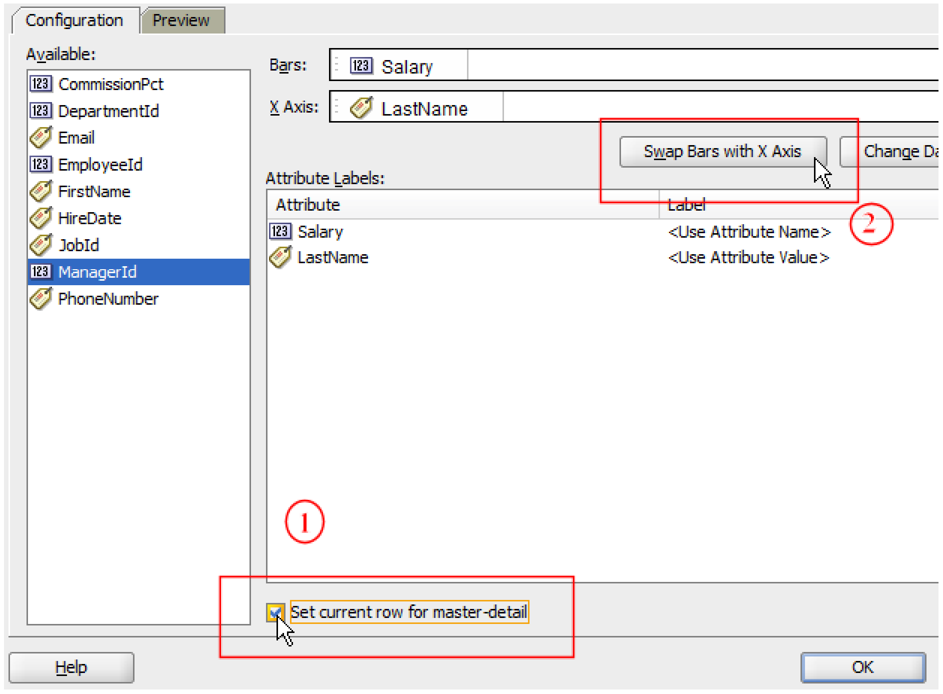
To build the DVT graph in Oracle JDeveloper, drag the collection from the Data Controls palette into the page and choose **Graph** from the context menu.



In the **Component Gallery**, choose the bar graph type.



For the Employees table, drag the **Salary** attribute into the graph field and the **LastName** attribute into the x-Axis field. This displays the employee salary over the last names. To ensure master-detail correlation to happen when users click on a bar, check the **Set current row for master-detail** checkbox (1), which ensures the current row to be set in the ADF binding layer iterator. In the example the current row setting is not used for displaying master-detail data, but to show additional information about the selected row.



Bars: Salary

X Axis: LastName

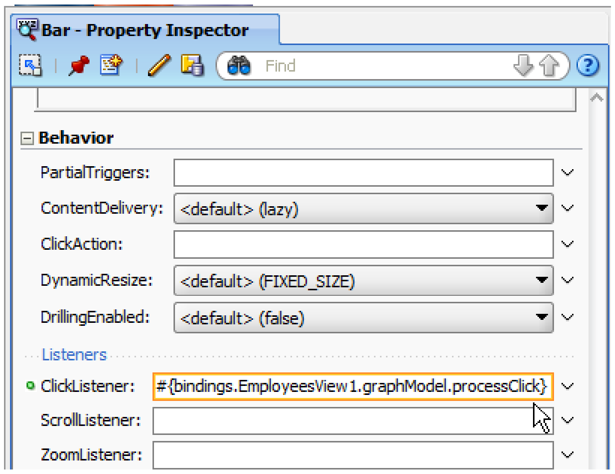
Attribute Labels:

Attribute	Label
Salary	<Use Attribute Name>
LastName	<Use Attribute Value>

[✓] Set current row for master-detail

Press the **Swap Bars with X-Axis** button (2) to turn the last name entries into their own series, which is key for the color highlighting explained in this article. The major change is between setting and not setting this option is that the last names are displayed in the graph's legend than beneath its bar.

To switch the graph color for the bar users click on, the **ClickListener** needs to be overridden. By default the ClickListener property points to the processClick action of the DVT binding in the PageDef file.


Bar - Property Inspector

Behavior

PartialTriggers:

ContentDelivery: <default> (lazy)

ClickAction:

DynamicResize: <default> (FIXED_SIZE)

DrillingEnabled: <default> (false)

Listeners

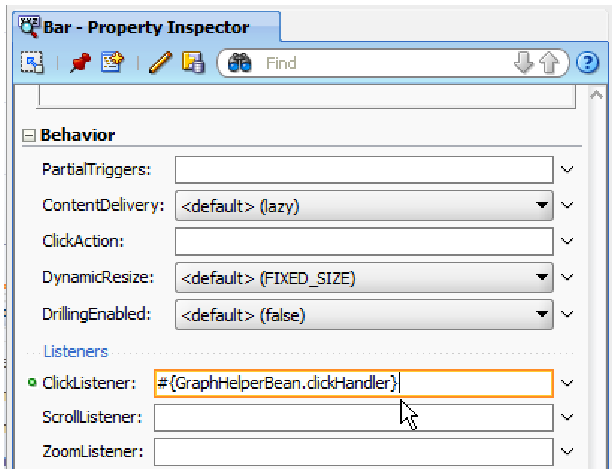
ClickListener: #{bindings.EmployeesView1.graphModel.processClick}

ScrollListener:

ZoomListener:

As shown below, reference a managed bean method that has the click handler method defined.

```
clickHandler(ClickEvent clickEvent)
```


Bar - Property Inspector

Behavior

PartialTriggers:

ContentDelivery: <default> (lazy)

ClickAction:

DynamicResize: <default> (FIXED_SIZE)

DrillingEnabled: <default> (false)

Listeners

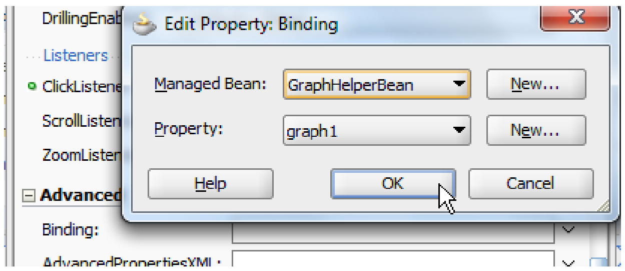
ClickListener: #{GraphHelperBean.clickHandler}

ScrollListener:

ZoomListener:

The click handler method performs the same functionality as the **processClick** action and provides you a great example for how to generically program against the ADF binding layer from a DVT graph instance.

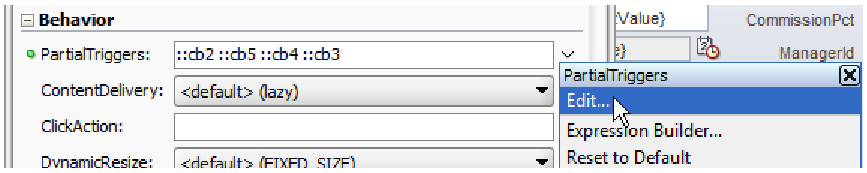
To access the graph instance, in the Property inspector, browse to the graph's **Binding** property and press the **Edit** link in the context menu opened by the arrow down icon to the right.



Managed Bean: GraphHelperBean
Property: graph1

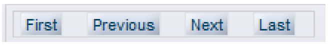
For this example, where the Employee form can also be navigated using navigation buttons, you need to make sure the graph is refreshed when one of the buttons is pressed.

Set the **PartialTriggers** property to point to the navigation button Ids. To set this property, use the **Edit** option from the context menu that opens when clicking the down arrow icon on the right.



PartialTriggers: ::cb2 ::cb5 ::cb4 ::cb3
ContentDelivery: <default> (lazy)
ClickAction:
DynamicResize: <default> (FIXED_SIZE)

Note: The navigation buttons are created by selecting the collection in the ADF Data Controls panel and dragging it onto the page. From the context menu choose **Navigation** and choose the navigation button option.



Shown below is the generated navigation bar page source. In addition to the **actionListener** property, which is automatically set when creating the navigation buttons, the **action** property is set to reference a managed bean method. The managed bean method is used to ensure the row navigation is highlighted in the graph. Also note that the partialSubmit property is set to true, which automatically happens when the **Navigation** context option is selected.

```
 <af:panelGroupLayout layout="horizontal" id="pgl2">
   <af:commandButton actionListener="#{bindings.First.execute}"
                     text="First"
                     disabled="#{!bindings.First.enabled}"
                     partialSubmit="true" id="cb2"
                     action="#{GraphHelperBean.onButtonNavigation}"/>
               …
</af:panelGroupLayout>
```

The **`onButtonNavigation`** method calls a helper function to highlight the new current row bar in the graph.

```
public String onButtonNavigation() {
  highlightSelectedSeriesBar();
  return null;
}
```

Managed Bean

The managed bean doesn't hold state and therefore is configured in request scope. The **clickHandler** method accesses the ADF binding layer through the graph instance

```
GraphDataModel graphModel = (GraphDataModel)graph1.getDataModel();
```

The **graph1** variable is an instance of the graph created through the graph component **Binding** property. It grants access to the ADF binding layer. The **clickEvent** provides the information about the bar (the series) the user clicked on. The bar's row representation is then set as current in the ADF binding layer.

```
import java.awt.Color;
import java.util.List;
import oracle.adf.model.binding.DCIteratorBinding;
import oracle.adf.model.dvt.binding.transform.cube.CubeDataModel;
import oracle.adf.view.faces.bi.component.graph.UIGraph;
import oracle.adf.view.faces.bi.event.ClickEvent;
import oracle.adf.view.faces.bi.model.GraphDataModel;

import oracle.dss.dataView.ComponentHandle;
import oracle.dss.dataView.DataComponentHandle;
import oracle.dss.util.ColumnOutOfRangeException;
import oracle.dss.util.DataAccess;
import oracle.dss.util.DataDirectorException;
import oracle.dss.util.DataMap;
import oracle.dss.util.RowOutOfRangeException;
import oracle.jbo.Row;
import oracle.jbo.uicli.binding.JUCtrlHierBinding;
import oracle.jbo.uicli.binding.JUCtrlHierNodeBinding;

public class GraphHelperBean {
  private UIGraph graph1 = null;
  boolean restoreView = false;

  public GraphHelperBean() {}

  //DVT graph component binding reference defined on the "Binding"
  //property
  public void setGraph1(UIGraph graph1) {
    this.graph1 = graph1;

  }

  public UIGraph getGraph1() {
    return graph1;
  }

  //custom click handler method that accesses the binding layer to set
  //the bar selected by the user as the current row
  public void clickHandler(ClickEvent clickEvent) {
```

```
  //Substitute for:#{bindings.EmployeesView1.graphModel.processClick}
  //get access to the hierarchical binding
 GraphDataModel graphModel  = (GraphDataModel)graph1.getDataModel();
 CubeDataModel cubeDataModel =
                        (CubeDataModel)graphModel.getDataDirector();

 //get access to the graph root binding. This information is needed
 //to the access to the JUCtrlHierbinding that is not exposed on the
 //graph without using internal classes

 JUCtrlHierNodeBinding rootnode =
                cubeDataModel.getCubicBinding().getRootNodeBinding();
 JUCtrlHierBinding hierBinding = rootnode.getHierBinding();

 //resolve the graph component the user clicked on
 ComponentHandle ch = clickEvent.getComponentHandle();
 if (ch != null) {
   DataComponentHandle dch = (DataComponentHandle)ch;
   Object keyPath = dch.getValue(DataMap.DATA_KEYPATH);
   if (keyPath != null) {
    //the code line below is comparable to table and tree selections
    //in where the component keyPath is used to identify the data
    //row in the ADF binding layer
    JUCtrlHierNodeBinding node =
                      hierBinding.findNodeByKeyPath((List)keyPath);
      if (node != null) {
         node.getRowIterator().setCurrentRow(node.getRow());
      }
    }
  }
  //color highlight selected node
  highlightSelectedSeriesBar();
}

public String onButtonNavigation() {
  highlightSelectedSeriesBar();
  return null;
}


/**
 * Highlight the current row in the bar graph. The row highlighting
 * is determined by comparing the current row in the ADF binding with
 * the row represented by the individual bars.
 */
public void highlightSelectedSeriesBar() {
  if (graph1 != null) {
    GraphDataModel graphModel =
                  (GraphDataModel)graph1.getDataModel();
    CubeDataModel cubeDataModel =
                  (CubeDataModel)graphModel.getDataDirector();

    //get the ADF iterator binding from the graph model
    JUCtrlHierNodeBinding rootnode =
                  cubeDataModel.getCubicBinding().getRootNodeBinding();
    JUCtrlHierBinding hierBinding = rootnode.getHierBinding();
    DCIteratorBinding iterator = hierBinding.getDCIteratorBinding();
    int rowCount = graphModel.getRowCount();
    DataAccess da;
    try {
       da = graphModel.getDataDirector().getDataAccess();
       Row currentRow = iterator.getCurrentRow();
       //iterate over the row data
      for (int i = 0; i < rowCount; ++i) {
        List keyPath;
        try {
          keyPath = (List)da.getValue(i, 0, DataMap.DATA_KEYPATH);
         JUCtrlHierNodeBinding node =
                          hierBinding.findNodeByKeyPath(keyPath);
         Row rw = node.getRow();
         if(rw.getKey().equals(currentRow.getKey())){
           node.getRowIterator().setCurrentRow(node.getRow());
           int rwIndex =
             node.getRowIterator().getRangeIndexOf(node.getRow());
             graph1.getSeriesSet()
                   .getSeries(rwIndex, true).setColor(Color.RED);
         }
         else{
             int rwIndex =
               node.getRowIterator().getRangeIndexOf(node.getRow());
             graph1.getSeriesSet()
                      .getSeries(rwIndex, true).setColor(Color.BLUE);
        }
      } catch (RowOutOfRangeException e) {
         e.printStackTrace();
      } catch (ColumnOutOfRangeException e) {
        e.printStackTrace();
      }
    }
  } catch (DataDirectorException e) {
    e.printStackTrace();
  }
 }
```

```
  }
}
```

Note: Initially the first bar in the graph is highlighted. If you set a different than the first bar as current during the initial page load, then you need to also call the **highlightSelectedSeriesBar** method to change the initial bar color coding.

Download

The Oracle JDeveloper 11.1.1.3 workspace can be downloaded from ADF Code Corner:

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

Configure the database connection to reference the HR schema of a local database in your reach and run the JSPX document in the view layer project. Note that for this sample the query of employee records is restricted to employees in department 60.

Using this sample code in your application development project, there is no limitation in the number of graphs to use.

RELATED DOCOMENTATION

⊠	
⊠	
⊠	