<table>
<thead>
<tr>
<th>Abstract:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A reoccurring question on OTN that for this reason has been answered many times in different blogs is how to allow users to select rows using a checkbox. So time for an ADF Code Corner version and use case. The use case explained in this article is deferred row delete in an ADF table. A checkbox is displayed at the beginning of a table for users to select rows they want to delete with a single button press. Of course, the approach described in here can be used for other use cases as well like row selection for update or similar.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Author:</th>
<th>Frank Nimphius, Oracle Corporation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>twitter.com/fnimphiu</td>
</tr>
<tr>
<td></td>
<td>02-MAR-2012</td>
</tr>
</tbody>
</table>
Introduction

The Oracle JDeveloper 11.1.2.1 sample application you can download at the end of this article allows you to select table rows for delete using a checkbox in front of the table. The table itself can be configured for single or multi row selection. The idea of deferred selection is that users don't have to keep the ctrl key pressed while selecting rows and they can scroll within the table without losing the selection state. Once done, users press a single button for the application logic to process the selected rows.

There are several options in which you can implement the use case. However, striving for best performance and minimal coding the use of transient attributes appears to be the best choice and is explained in the following.
Preparing ADF Business Components

The sample has a single entity and view object created for the Employees table of the Oracle HR sample schema. To add the transient attribute, open the entity object and choose the Attributes menu option. In the attribute dialog, click the green plus icon to provide a name, like MarkedForDelete, type: Boolean and a Display name that is shown as the attribute prompt at runtime.

Most importantly, ensure the Transient radio button value is selected as shown in the image below.

In a next step, open the View Object with a double click onto the file in the Application Navigator and again choose the Attributes menu option.

Click the green plus icon and choose Add Attribute from Entity.

Select the transient attribute from the list of entity attribute in the left side of the selection dialog and press OK to add the attribute to the View Object.
Click the UI Hints tab as shown in the image above and define the Control Type property as Check Box. When you drag the View Object as a table to an ADF page the ADF configuration dialog recognizes the default setting and shows the attribute to be rendered as a checkbox.
All you need to do is to move the checkbox to the top so it appears first in the table.

**Disclaimer:** This sample is built with JDeveloper 11g R2. I didn't test other releases for if the Control Type property is recognized. So in case this step works different, don't blame it on this article.

Back to the work we need to do in the ADF Business Component model. The next step is to expose a public method on the View Object client interface to perform the deferred action on the selected rows.

Still with the View Object editor open, select the Java menu option and press the pencil icon next to the Java Classes header. In the opened dialog, select the Generate View Object Class: <View Object Name>Impl and press Ok.

Open the created Java class and create a public method you want to be called by a button press in the ADF page.
In this sample, the desired action is to iterate over the selected rows and delete them. As you can tell from the code shown below, anything else than delete is possible too and there is no limit is set to developer creativity.

```
/**
 * Sample method that if invoked deletes all rows that have the
 * MarkedForDelete transient attribute set to true
 */

public void deleteSelectedRows(){
    //create a second row set to not impact the row set
    //used in ADF
    RowSet duplicateRowSet = this.createRowSet("duplicateRowSet");
    //set rowset to first row to avoid "attempt to access
    //dead row" exception
    duplicateRowSet.first();
    //get the current row of the table to set it back after
    //re-executing the VO
    Row currentRow = this.getCurrentRow();
    boolean currentRowDeleted = false;
    //get all rows that have the transient attribute
    //"MarkForDelete" set to true
    Row[] rowsToDelete =
        duplicateRowSet.getFilteredRows("MarkedForDelete", true);
    if (rowsToDelete.length>0) {
        //only run through this code if there is something to
        //delete
        for (Row rw : rowsToDelete) {
            //if row is ts marked as the current in VO, set
            //boolean flag
            if (rw.getKey().equals(currentRow.getKey())) {
                currentRowDeleted = true;
            }
            //remove row - don't yet commit
            rw.remove();
        }
        //re-execute VO
        this.executeQuery();
        //reset current row if it hasn't been deleted
        if (!currentRowDeleted) {
            this.setCurrentRow(currentRow);
        }
        duplicateRowSet.closeRowSet();
    }
}
```

Save the file and compile it.
Next, in the View Object editor select the pencil icon next to the Client Interface and move the public method under the EmployeesView node as shown in the image below.

![Image showing the View Object editor with the pencil icon selected and the public method moved to the top.]  

Press Ok to close the dialog and save your work.

**Building the ViewLayer**

To create the sample UI, all that was needed is to drag the Employees view object as a table to the JavaServer Faces page. In the opened dialog, move the MarkedForDelete attribute to the top.

![Image showing the dialog for moving the MarkedForDelete attribute to the top.]
In the sample, the table component is surrounded by an af:panelCollection component.

Drag the deleteSelectedRow method exposed on the View Object client interface from the DataControl panel and drop it as a toolbar button to the PanelCollection toolbar. The af:table PartialTriggers property is configured to point to the toolbar button ID property for the table to refresh in response to a button press.

Sample Download

The sample application to this article can be downloaded as sample #99 from the ADF Code Corner website


Configure the database connection to point to a database with the HR schema unlocked. Run the JSF page and select rows to delete. When you hit the delete button, you delete the selected rows but don’t yet commit this change. So be careful when pressing the commit button ;-)
Optional Sample Extension

Note that the selected table row isn’t automatically added to the selected rows for delete. If you want to implement this, you need to create a custom SelectionListener (af:table property) to set the current selected row and access the current row in the DCIteratorBinding to set the row's MarkedForDelete attribute to true.

Hint: to set a row current in a custom SelectionListener, use Java (MethodExpression) to invoke the EL string that by default is configured for this property when dragging a table from the DataControl panel to the JSF page.

RELATED DOCUMENTATION