ADF Code Corner

034. How-to pass additional arguments to a JavaScript function using the af:clientAttribute element

Abstract:

In JDeveloper 11, JavaScript events like onClick, onFocus etc. are handled through the ADF Faces RC client framework using the af:clientListener component. The signature of the af:clientListener is defined such that the developer provides the name of a method to call and the JavaScript event to trigger this method call.

The clientListener signature however doesn't foresee a mechanism for the developer to pass additional arguments - like a rowKey - to the JavaScript method. Here client attributes come to the rescue.

twitter.com/adfcodecorner

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
22-JUL-2008

Introduction

A usecase in which a client side JavaScript needs to know about the rowKey of the component instance that triggered the event is for example a new browser window that needs to be opened using the window.open() command, addressing a URL of a taskflow within isolated transaction mode. This would then allow developers to implement context menus that operate similar to the open in new Tab command in the FireFox, Safari and IE browsers. Since the JavaScript function in ADF Faces RC cannot take a second argument in a call from af:clientListener, the rowkey value needs to be passed differently.

Passing the rowKey to a JavaScript function

If you can't pass the additional information with the request, then you need to piggyback on an object that gets passed implicit. The implicit object that is passed by ADF Faces RC with the event is the source object of the request origin. So if the call is issued from an output text component, then the output text component is made available to the JavaScript function. If the event originating component is available to JavaScript then all its properties are available true, at least those that are not suppressed by the client framework for security reasons (Secure Ajax is more important than exposing "all you can eat" to the javaScript developer). Using the af:clientAttribute element, the existing properties of a component can be extended by custom properties - like "rowKey".

```
<af:table value="#{bindings.DepartmentsView1.collectionModel}"
   rows="#{bindings.DepartmentsView1.rangeSize} var="row"
   emptyText="#{bindings.DepartmentsView1.viewable ?
            \'No rows yet.\' : \'Access Denied.\'}"
   fetchSize="#{bindings.DepartmentsView1.rangeSize}"
   selectedRowKeys="#{bindings.DepartmentsView1.
                    collectionModel.selectedRow}"
   selectionListener="#{bindings.DepartmentsView1.
                      collectionModel.makeCurrent}"
   rowSelection="single">
     <af:column sortProperty="DepartmentId" sortable="false"
                headerText="#{bindings.DepartmentsView1.hints
                             .DepartmentId.label}">
        <af:outputText value="#{row.DepartmentId}">
         <af:convertNumber groupingUsed="false"
                           pattern="#{bindings.DepartmentsView1.
```

```
                             hints.DepartmentId.format}"/>
13.           <af:clientAttribute name="rowKey"
                                         value="#{row.rowKeyStr}"/>
14.           <af:clientListener method="handleDepartmentClick"
                                         type="click"/>
15.         </af:outputText>
16.       </af:column>
17.    ...
18.   </af:table>
```

The above markup is a fragment of a table definition using Oracle ADF as the binding layer. The interesting code lines are highlighted in bold.

Line 8 - 10 define the column for the Departmentid, which, on mouse click, will call the JavaScript function. of course in a real application this functionality would be exposed more explicit by a command link, a button or a context menu. However, for this sample, a click on the output text components will do.

In **line 13** the new clientAttribute is defined with a name of "rowKey" and the value of the currently rendered row rowKey. The "row" variable is defined for the table component and used to iterate over during the table rendering to draw the table rows. For each of the instance of the output text component, a new attribute is created with the current rowKey assigned.

Line 14 assigns the clientListener to the output text field so that the **handleDepartmentClick** method is called when the user clicks onto the output text. One important action that happens implicit is that the output text field is rendered as a client component, which means that it has a physical JavaScript object representation at runtime. As a developer you don't need to bother with this information, but this step is needed as otherwise you would have to explicitly set the clientComponent property to true.

The JavaScript method looks as follows

```
01.  <af:document>
02.    <af:messages/>
03.    <f:facet name="metaContainer">
04.        <af:group>
05.            <script>
06.                function handleDepartmentClick(evt){
07.                    rowKey = evt.getSource().getProperty("rowKey");
08.                    alert(rowKey);
09.                }
10.            </script>
11.       </af:group>
12.     </f:facet>
13.    ...
14.  </af:document>
```

Line 7 reads the row key value from the outputText component, which is passed as the source of the JavaScript event. Once the information is available in the JavaScript function, developers can use it as they please e.g. to use it as a request parameter in a call to a new browser window.