ADF Code Corner

010. How-to create a character input counter for text fields

twitter.com/adfcodecorner

Abstract:

Text input areas that allow users to enter free text have a limitation in the maximum length, which usually is determined by the size of the underlying database column. Even for experienced users that know about the input size limitation, it is not convenient to count the number of characters while typing to avoid hitting the server side field validation error. Ideally there is a visual indication for the user that tells him about the number of characters left to finish his free text.

This how-to explains how JavaScript can be used in combination with the ADF Faces Rich Client client framework to meet this requirement.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
15-SEP-2008

Introduction

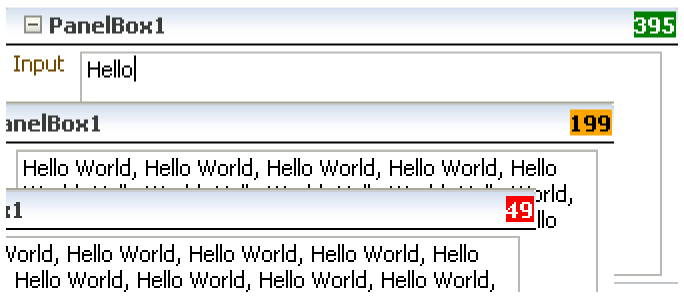
JavaScript should be seen as a programming technique of the last resort in ADF Faces RC. Ideally you find everything you need within ADF Faces and JavaServer Faces native APIs and avoid JavaScript altogether. The reason that makes JavaScript a second best choice is that it is directly added to a page (assuming its not a reusable code that resides in a library), which then increases the download size and time. In addition, JavaScript has a different semantic than Java and - by the time of writing - is not as good to debug and trace in Oracle JDeveloper then Java.

However, there are usecases where it is better to go for a JavaScript solution than for a server side implementation. One of the usecases is the realization of a character input counter. Of course this could be implemented using ADF Faces RC native APIs, but this then would be more expensive at runtime. Before you blindly adopt JavaScript in your ADF Faces RC web application development, think twice. JavaScript that directly accesses the client side document object model should not be used at all and instead you should take the time to familiarize yourself with the client side JavaScript APIs exposed by ADF Faces RC.

This how-to document, beside of giving you a starter code for input character counting and visualization, gives you some hints of how to work with the client side JavaScript framework if ADF Faces RC.

Sample Code explained

The example provided for download in this how-to contains a text field surrounded by a panelBox component. In its header the panelBox component has an output text field to show the current counter value plus a visual indication of green, orange and red.


PanelBox1
395
Input
Hello
anelBox1
199
Hello World, Hello World, Hello World, Hello World, Hello
orld,
llo
t1
49
Vorld, Hello World, Hello World, Hello World, Hello
Hello World, Hello World, Hello World, Hello World,

The page source looks as follows:

```
<af:form>
  <af:panelBox text="PanelBox1" clientComponent="true">
    <f:facet name="toolbar">
      <af:outputText id="out1" clientComponent="true" value="400"/>
     </f:facet>
     <af:inputText label="Input" rows="7" columns="50"
                   clientComponent="true"
                   maximumLength="400">
        <af:clientListener method="charChecker" type="keyPress"/>
      </af:inputText>
    </af:panelBox>
</af:form>
```

For client side components to be accessed with JavaScript, you need to ensure a client component is created. This is done by setting the "clientComponent" property to true. To code the solution generic, the maximumLength property of the input text field is set to the maximum number of characters. Using ADF you may want to use an Expression Language to get this value from the underlying Business Service like ADF Business Components.

The JavaScript function that checks the typed input character is triggered by an af:clientListener component that listens for the keyPress event and that is assigned to the af:inputText field.

```
<af:document>
...

<af:resource tpe="javascript">
  function charChecker(evt){
     textfield = evt.getCurrentTarget();
     textfield_current_content = textfield.getSubmittedValue();
     textfield_current_content_length =
                                 textfield_current_content.length;

     vGREEN               = textfield.getMaximumLength();
     vORANGE              = vGREEN/2;
     vRED                 = vORANGE/4;
     vBACKGROUND_COLOR    ='white';
     vCOLOR               ='black';

     counter          =  AdfPage.PAGE.findComponentByAbsoluteId("out1");
     counter_value        = counter.getValue();

    if (vGREEN - textfield_current_content_length  <= vGREEN &&
        vGREEN - textfield_current_content_length > 0){

        if(vGREEN - textfield_current_content_length <= vRED){
           vBACKGROUND_COLOR   ='red';
           vCOLOR='white';
         }
         else{

           if(vGREEN - textfield_current_content_length <= vORANGE){
             vBACKGROUND_COLOR   ='orange';
             vCOLOR='black';
           }
         else{
              vBACKGROUND_COLOR   ='green';
              vCOLOR='white';
         }

       }

       counter.setInlineStyleProperty(
                    "background-color",vBACKGROUND_COLOR);
       counter.setInlineStyleProperty("color",vCOLOR);
       counter_value = vGREEN - textfield_current_content_length-1;

    }
    else{
      if (   evt.getKeyCode()!= AdfKeyStroke.BACKSPACE_KEY
          && evt.getKeyCode()!= AdfKeyStroke.TAB_KEY
          && evt.getKeyCode()!= AdfKeyStroke.DELETE_KEY
          && evt.getKeyCode()!= AdfKeyStroke.ARROWLEFT_KEY
          && evt.getKeyCode()!= AdfKeyStroke.ARROWUP_KEY
          && evt.getKeyCode()!= AdfKeyStroke.ARROWRIGHT_KEY
          && evt.getKeyCode()!= AdfKeyStroke.ARROWDOWN_KEY)
          {
             evt.cancel();
           }
        }

          counter.setValue(counter_value);
        }
 </af:resource>
…
</af:document>
```

The JavaScript function *charChecker* takes a single argument, which is the keyboard event that is passed to it from the ADF Faces RC client framework. Note that this event is not the native browser event but a wrapped event from the ADF Faces framework. The target of the event is the actual text field receiving the character input. Once you gain access to the text field you also get information about the allowed maximum length and the current field value.

Instead of using the native document.getElementById() call to access a component on the page, you use AdfPage.PAGE.findComponent() instead. The reason why getElementById() is not the best option to use is because it doesn't access the component but the generated HTML output, which means that developers need to have a clear idea of how exactly this output looks like.

The following if/else statement determine the color of the counter background to be green, orange or red. If the maximum length is reached then no more keyboard input is allowed. To handle this, all input keys other than delete and arrow keys are suppressed by canceling the event.

Sample Download

Download the Oracle JDeveloper 11 sample workspace from **ADF Code Corner**:
http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

Note that this sample has been build with a pre-production build of Oracle JDeveloper 11.

RELATED DOCOMENTATION

☒	Oracle Fusion Developer Guide – McGraw Hill Oracle Press, Frank Nimphius, Lynn Munsinger http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543
☒	
☒	