ADF Code Corner

027. How-to show a glasspane and splash screen for long running queries

Abstract:

Application users are known to be impatient when waiting for a task to be completed. To avoid users pressing a command button twice or changing data input while waiting for a long running query or process to complete, ADF Faces provides the option to suppress any user input for the duration of an action.

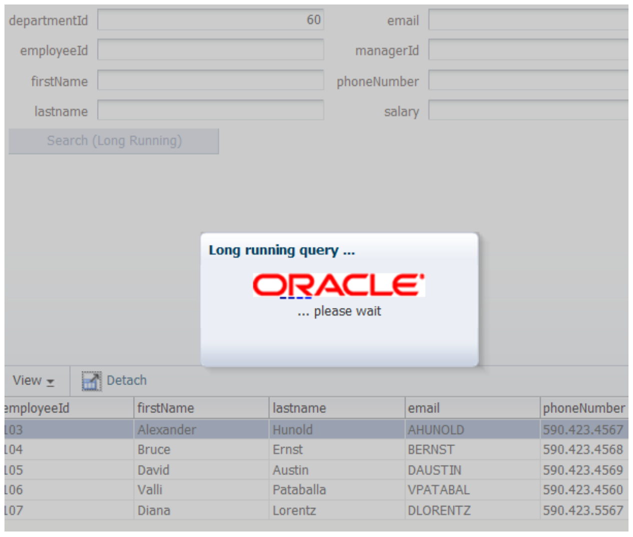
This how-to article explains how developers use JavaScript to build a generic solution that launches a glasspane to prevent user input, displays a splash screen and that can be applied to existing applications with no code changes to the command invoking the query.

twitter.com/adfcodecorner

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
30-JULY-2010

Introduction

This how-to document provides a generic solution that ADF developers use to prevent users from double submitting or changing input data while a long running query executes. The solution implemented in the following consists of a glasspane and a splash screen that are shown to the application user, as shown below



departmentId	60	email	
employeeId		managerId	
firstName		phoneNumber	
lastname		salary	

employeeId	firstName	lastname	email	phoneNumber
103	Alexander	Hunold	AHUNOLD	590.423.4567
104	Bruce	Ernst	BERNST	590.423.4568
105	David	Austin	DAUSTIN	590.423.4569
106	Valli	Pataballa	VPATABAL	590.423.4560
107	Diana	Lorentz	DLORENTZ	590.423.5567

Creating the Splash Screen

The splash screen in this example is build with an af:popup component that contains an af:dialog component. An important configuration on the af:popup is to set its contentDelivery property to "immediate" for the component to render when the page loads.

```
<af:popup id="p1" contentDelivery="immediate">
  <af:dialog id="d2" type="none" title="Long running query ..."
             closeIconVisible="false">
    <af:panelGroupLayout id="pgl1" layout="vertical">
      <af:image source="/images/codecorner.gif" id="i1"/>
        <af:image source="/images/animbar.gif" id="i2"
                  inlineStyle="width:197px;"/>
        <af:outputText value="... please wait" id="ot11"/>
    </af:panelGroupLayout>
  </af:dialog>
</af:popup>
```

Invoking the splash screen

Using JavaScript to launch the splash screen and to disable user input, no change is required to the command component that invokes the long running query or process. This is important as it allows developers to almost declaratively add this functionality to their existing applications.

```
<af:commandButton
   actionListener="#{bindings.findEmployee.execute}"
   text="Search (Long Running)" id="cb1" partialSubmit="true">
     <af:clientListener method="enforcePreventUserInput"
                        type="action"/>
</af:commandButton>
```

In the page source example above, an af:commandButton component has an action listener configured bound to an ADF method binding. The "partialSubmit" attribute is also set to true to issue a partial submit, which is a pre-requisite for the af:popup component to launch.

An af:clientListener component is configured for the button to listen for the button action event, which can be triggered from a key press of mouse click. The listener calls a JavaScript function, "enforcePreventUserInput", to disable the user interaction and to show the splash screen.

```
<af:resource type="javascript">
  function enforcePreventUserInput(evt){
    var popup = AdfPage.PAGE.findComponentByAbsoluteId('p1');
    if (popup != null){
      AdfPage.PAGE.addBusyStateListener(popup,handleBusyState);
      evt.preventUserInput();
    }
  }

  //JavaScript call back handler
  function handleBusyState(evt){
    var popup = AdfPage.PAGE.findComponentByAbsoluteId('p1');
    if(popup!=null){
      if (evt.isBusy()){
        popup.show();
      }
      else if (popup.isPopupVisible()) {
        popup.hide();
        AdfPage.PAGE.removeBusyStateListener(popup,
                                                     handleBusyState);
      }
   }
 }
</af:resource>
```

The JavaScript code above is added to the ADF Faces page source, using the af:resource tag. In a real application development project, this code would better be placed into an external JS file or a page template.

The "enforcePreventUserInput" function is called by the af:clientListener and looks up the af:popup component to launch. Once the handle to the component is found, a callback method is defined that gets called when the application starts becoming busy and when it returns from the task. In the example the busy state information is used to show the popup and, once the task completes, to hide it. After this, the "preventUserInput" method is invoked on the action event to show a glasspane that prevents users from interacting with the application. The glasspane is automatically removed when the application busy state is released.

Download

A sample Oracle JDeveloper workspace can be downloaded from **ADF Code Corner:**

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

The sample is based on a POJO business service that sets the current thread to sleep before executing a search. Run the JSPX document in the view layer project and e.g type in 60 as a value for the departmentId field before pressing the search button or hitting the enter key.