ADF Code Corner

Oracle JDeveloper OTN Harvest 02 / 2012

twitter.com/adfcodecorner

Abstract:

The Oracle JDeveloper forum is in the Top 5 of the most active forums on the Oracle Technology Network (OTN). The number of questions and answers published on the forum is steadily increasing with the growing interest in and adoption of the Oracle Application Development Framework (ADF).

The ADF Code Corner "Oracle JDeveloper OTN Harvest" series is a monthly summary of selected topics posted on the OTN Oracle JDeveloper forum. It is an effort to turn knowledge exchange into an interesting read for developers who enjoy harvesting little nuggets of wisdom.

http://blogs.oracle.com/jdevotnharvest/

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
29-FEB-2012

Oracle ADF Code Corner OTN Harvest is a monthly blog series that publishes how-to tips and information around Oracle JDeveloper and Oracle ADF.

Disclaimer: ADF Code Corner OTN Harvest is a blogging effort according to the Oracle blogging policies. It is not an official Oracle publication. All samples and code snippets are provided "as is" with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

If you have questions, please post them to the Oracle OTN JDeveloper forum:
http://forums.oracle.com/forums/forum.jspa?forumID=83

February 2012 Issue – Table of Contents

Oracle JDeveloper 11.1.1.6 New Features

Oracle JDeveloper 11.1.1.6 (aka PatchSet 5) has been released this month with many new features added, mainly focusing on tabloid PC support. A full list of new features has been published on OTN

http://www.oracle.com/technetwork/developer-tools/jdev/index-088099.html

ADF Faces Cheat Sheets

The ADF Faces cheat sheets will help you get started using ADF Faces rich client components. The cheat sheets are grouped by component type and functionality, and work together with the Web User Interface Developer's Guide for ADF and the ADF Faces Tag Library documentation.

Though the document is available for many versions of Oracle JDeveloper it seems to be less known. A good reason to point it out here

http://docs.oracle.com/cd/E18196_01/MenuPage.html

Learn ADF! Online and for Free!

The Oracle JDeveloper and ADF curriculum team has created a 700 minute online advanced ADF training that is accessible for free from the Oracle Learning library.

https://apex.oracle.com/pls/apex/f?p=44785:24:0::NO::P24_CONTENT_ID,P24_PREV_PAGE:6022,1

Quoting from the Oracle Learning library:

This self-paced "eCourse" is the first in a series that addresses Oracle ADF 11g topics through video presentations, quizzes and practices. The course was designed to provide you with the most relevant and performance-based learning experience possible in a self-study environment.

You'll find tips, tricks, and best practices from Oracle experts. You decide the way you want to learn as well as the sequence in which you want to study the modules in the course. Topics in Part 1 include:

- *ADF Overview*
- *ADF Bindings*
- *Project Considerations for Team Development*
- *ADF Task Flows*

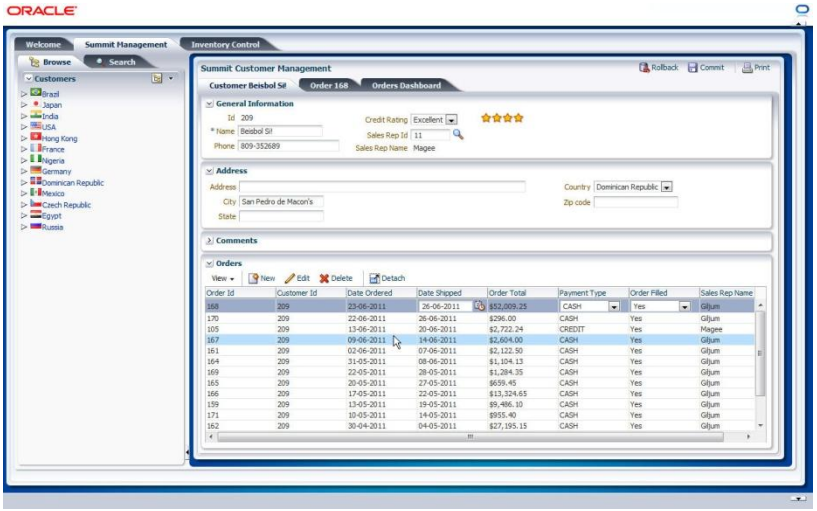
Learn How to skin ADF Faces applications

Recently I observed an increase of questions on OTN and Oracle internal that aim for applying CSS on the generated HTML output of an ADF Faces application. Surely, skinning in ADF is not the same as using CSS in tools like Dreamweaver, but it is the proper way of applying custom images and colors to ADF Faces applications. The biggest risk in styling the generated ADF Faces HTML output with CSS is change in the renderer classes. Oracle constantly works on improving its ADF Faces components, for

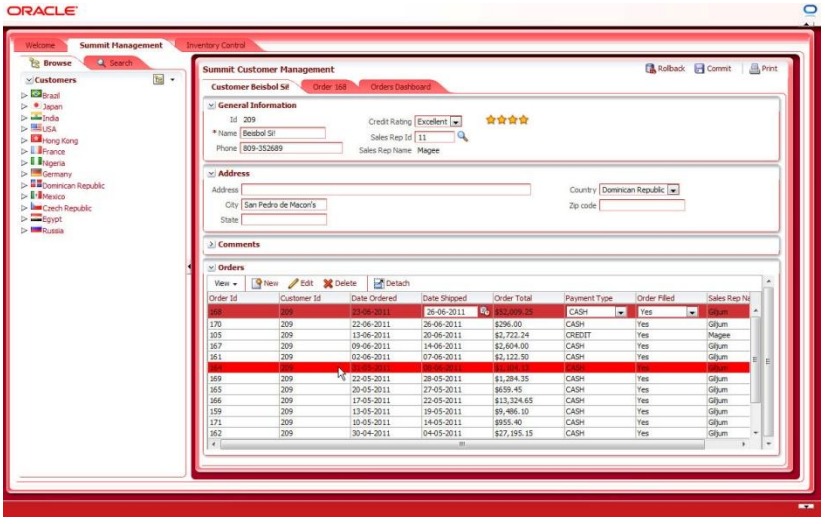
example using HTML 5 to replace Flash and DHTML on some of the ADF Faces components. If you skinned applications on the generated output, then with each of the changes Oracle applies, your custom styles will break.

ADF skinning applies style sheet definitions to style classes at runtime. In contrast to direct output styling, the style classes are dynamically created and derived from the ADF Faces skin selectors. The component developer ensures that the style classes are always set to the correct location in the generated component output, ensuring that changes last across Oracle JDeveloper versions and component changes.

For example, using ADF skinning, you easily get from this …



… to this.



Though I can't save you from learning, I can help you with pointers to sources you want to be aware of:

ADF skinning is documented in the Oracle® Fusion Middleware Web User Interface Developer's Guide for Oracle Application Development Framework

http://docs.oracle.com/cd/E21764_01/web.1111/b31973/af_skin.htm#BAJFEFCJ

An ADF insider recording exists that explains skinning in a 40 minute video. Though this recording doesn't show the new skin editor, you learn a about how skinning works, how you dynamically detect skins at runtime and how you debug skins using FireBug in FireFox.

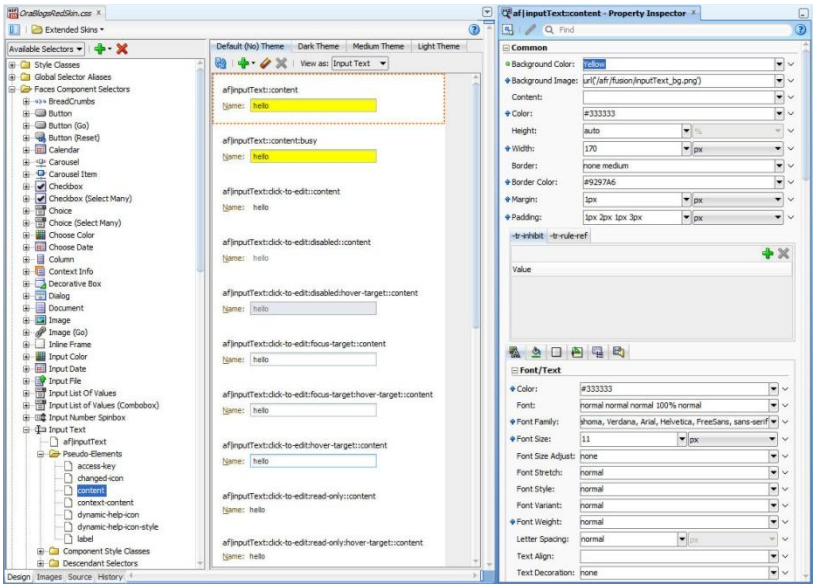
http://download.oracle.com/otn_hosted_doc/jdeveloper/11gdemos/adf-insider-skinning/adf-insider-skinning.html

As mentioned, a visual skin editor exists that you can use in its stand-alone edition for JDeveloper 11g R1 (11.1.1.4, 11.1.1.5) applications and integrated in JDeveloper 11g R2. An article that explains working with the skin editor and a recommend workflow is published here

http://www.oracle.com/technetwork/issue-archive/2011/11-nov/o61adf-512006.html

To download the stand alone skin editor (JDeveloper 11g R2 has it integrated and no extra download is required), use the link below

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

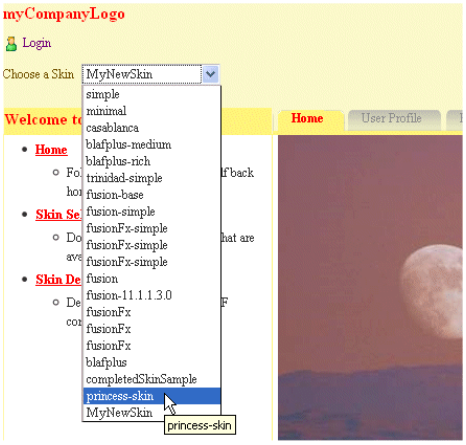


To learn about the stand alone and the integrate skin editor, refer to the Oracle® Fusion Middleware Skin Editor User's Guide for Oracle Application Development Framework, which you can access online from

http://docs.oracle.com/cd/E16162_01/user.1112/e17456/toc.htm

To try the skin editor on a JDeveloper 11g R2 sample, you can run through the hands-on exercise exposed as part of the Oracle learning library.

http://docs.oracle.com/cd/E18941_01/tutorials/jdtut_11r2_83/jdtut_11r2_83.html



A video recording of how to use the skin editor can be found here

https://blogs.oracle.com/shay/entry/adf_faces_skin_editor_how

And finally, a list of all skin selectors and ADF Faces components can be read up in the Oracle Fusion Middleware Tag Reference for Oracle ADF Faces Skin Selectors

http://docs.oracle.com/cd/E21764_01/apirefs.1111/e15862/toc.htm

This document is a well written by the ADF Faces component developers and provides information that you don't find in other documentation.

If you worked through all of this, Skinning should no longer be a problem for you.

Toggle panelBox open state on mouse click in header

A question came up on how to open / close a panelBox component by the user clicking on the panelBox header instead of the close/disclose icon.

The page source below shows the JavaScript code needed for this functionality, as well as the `af:clientListener` configuration on the `af:panelBox` component

```
<f:view xmlns:f="http://java.sun.com/jsf/core"
  xmlns:af="http://xmlns.oracle.com/adf/faces/rich">
    <af:document title="PanelBoxTester.jsf" id="d1">
      <af:resource type="javascript">
         function togglePanelBox(mouseEvent){
           var panelBox = mouseEvent.getSource();
            panelBox.broadcast(new AdfDisclosureEvent(
                                panelBox, !panelBox.getDisclosed()));
        }
   </af:resource>
   <af:form id="f1">
     <af:panelBox text="PanelBox1" id="pb1">
        <f:facet name="toolbar"/>
        <af:clientListener method="togglePanelBox" type="dblClick"/>
     </af:panelBox>
```

```
    </af:form>
  </af:document>
</f:view>
```

Note: The above example toggles the disclosure state when you click anywhere in the panelBox – not just the header. Apparently an ER has been filed to expose just the header for such actions. For the time being this solution will do.

Forms trigger equivalents in Oracle ADF

As more and more Oracle Forms developers approach application development with Oracle ADF, questions for Forms trigger equivalents in Oracle ADF come up frequently.

Oracle Forms trigger equivalents are documented in the Oracle ADF product documentation. The direct link to this is here

http://docs.oracle.com/cd/E15523_01/web.1111/b31974/appendix_formstriggers.htm

Strategies for controlling the af:popup close event

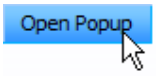
In a previous OTN Harvest summary, I discussed how to handle the af:dialog OK and Cancel event: http://www.oracle.com/technetwork/developer-tools/adf/learnmore/77-ok-cancel-support-in-dialog-351871.pdf . In this post, I get back to this though not covering the "Cancel" case.

There are two options developers have to handle the "Ok" event of an af:dialog component in an af:popup

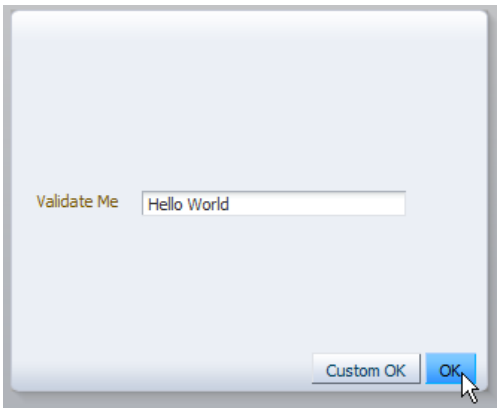
- Using a DialogListener with the default Ok button
- Using a custom command button instead of the default Ok button

The sample use case is quite simple: A command button added to a page opens the popup component that contains a dialog with an input text field in it. Users can provide a value in the input text field and press either a "custom Ok" button or the default "Ok" button to submit the value and close the popup. In both cases, the provided value in the text field is evaluated and if it is not "ADF ROCKS" written in whatever case, an error message is displayed and the popup is not closed.

Example 1: Using the default OK button and a DialogListener



The sample application opens with a single command button visible on the screen. Pressing the command button uses an `af:showPopupBehavior` behavior tag to open the popup.

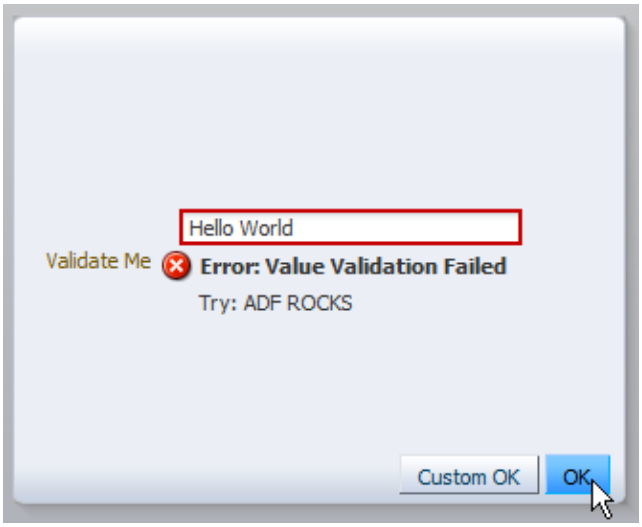


Typing "Hello World" into the text field and pressing the **Ok** button (which is the default Ok button configured in the **Type** property of the `af:dialog` component) invokes a managed bean method that is configured as the DialogListener for the dialog component.

```
<af:dialog …
            binding="#{DialogContentHandler.dialogComponent}"
            contentWidth="300" contentHeight="200"
            dialogListener="#{DialogContentHandler.onDialogAction}">
```

Note that the **binding** property is set to define a JSF component binding of the dialog to the managed bean to allow the lookup of the input text field in Java.

Because the entered text is not "ADF ROCKS", an error message is displayed below the input field



To display error messages in a popup, the `af:message` tag is added to the popup dialog as shown below

```
<af:dialog …>
  <af:panelFormLayout id="pfl1">
      <af:panelLabelAndMessage label="Validate Me" id="plam1">
         <af:panelGroupLayout id="pgl1" layout="vertical">
          <af:inputText id="it1"/>
          <af:message id="m1" for="it1"/>
```

```
        </af:panelGroupLayout>
      </af:panelLabelAndMessage>
  </af:panelFormLayout>
</af:dialog>
```

The dialog listener managed bean code displaying the error or, in the case of the correct entry, dismissing the dialog is shown next:

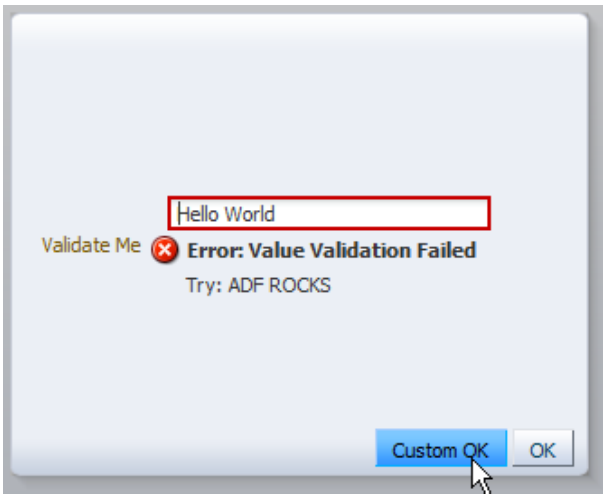
```
/**
 * Dialog Listener that validates the input field for ADF ROCKS.
 * If the string doesn't match, an error message is shown
 *
 * @param dialogEvent
*/
public void onDialogAction(DialogEvent dialogEvent) {
  //lookup the text field starting from the dialog component for
  //which a JSF component binding has been created
   UIComponent inputText = dialogComponent.findComponent("it1");
   String inputTextValue =
         (String) ((RichInputText)inputText).getValue();

   //If the dialog outcome is OK (the OK button has been pressed)
   //validate the entry
   if(dialogEvent.getOutcome() == DialogEvent.Outcome.ok){
     if(inputTextValue != null &&
        inputTextValue.equalsIgnoreCase("ADF ROCKS")){
        //ensure the input text value is reset for a second run
       ((RichInputText) inputText).resetValue();
     }
     else{
       //show error message so that popup doesn't close
       FacesContext fctx = FacesContext.getCurrentInstance();
       FacesMessage fm =
              new FacesMessage(FacesMessage.SEVERITY_ERROR,
                     "Value Validation Failed", "Try: ADF ROCKS");
        fctx.addMessage(inputText.getId(), fm);
     }
  }
}
```

Example 2: Using a custom command button

The `af:dialog` component renders without default buttons when the **Type** property is set to **none.** This is useful if you want to add your own command buttons, which then also are better to customize.

```
<af:dialog id="…" …>
  <f:facet name="buttonBar">
    <af:commandButton text="Custom OK" id="cb2"
        action="#{DialogContentHandler.onOK}"
        partialSubmit="true"/>
</af:dialog>
```

The use case is the same as before and a message displays when the input text field value is not ADF ROCKS.



The only difference to the code executed in a DialogListener is that the command button action listener needs to explicitly close the dialog

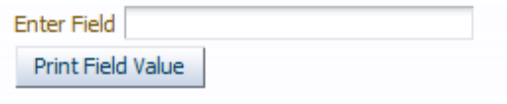
```
public String onOK() {
   UIComponent inputText = dialogComponent.findComponent("it1");
   String inputTextValue = (String)
((RichInputText)inputText).getValue();
    if(inputTextValue != null &&
       inputTextValue.equalsIgnoreCase("ADF ROCKS")){
       RichPopup rp = (RichPopup) dialogComponent.getParent();
       //reset input text component
       ((RichInputText) inputText).resetValue();
        rp.hide();
      }
      else{
          FacesContext fctx = FacesContext.getCurrentInstance();
          FacesMessage fm =
             new FacesMessage(FacesMessage.SEVERITY_ERROR,
                      "Value Validation Failed", "Try: ADF ROCKS");
           fctx.addMessage(inputText.getId(), fm);
        }
      return null;
 }
```

How-to define a default action for page fragments

The `af:form` component has a **DefaultCommand** property that, when set to the component Id of a command component like `af:commandButton` invokes the associated command action when the enter key is pressed anywhere in this form. However, if the form fields are contained in a page fragment exposed in a region then using the **DefaultCommand** property may not be an option as it is difficult to predict the command button id and its surrounding naming containers.

A solution to this is to use JavaScript on the UI input components that, when the **Enter** key is pressed virtually press a button within the page fragment (note that the `af:form` element belongs to the parent page and that you can only have a single `af:form` component per browser page)

Let's assume a page fragment with a single input text component and a command button to press:




```
<af:panelFormLayout id="pfl1">
    <f:facet name="footer">
      <af:commandButton text="Print Field Value" id="cb1" partialSubmit="true"
                        clientComponent="true"
                        action="#{View1Bean.onButtonPressed}"/>
    </f:facet>
    <af:inputText label="Enter Field" id="it1" binding="#{View1Bean.inputTextField}">
      <af:clientListener method="onFieldEnterKey" type="keyUp"/>
    </af:inputText>
  </af:panelFormLayout>
```

The command button is bound to a managed bean action method. By default, the action method is invoked when users press the command button. However, with the JavaScript shown next, this can be simulated and mapped to the **Enter** key press in the text field.

```
<?xml version='1.0' encoding='UTF-8'?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"
                xmlns:af=http://xmlns.oracle.com/adf/faces/rich
                xmlns:f="http://java.sun.com/jsf/core">
   <af:resource type="javaScript">
   //function called by the client listener
     function onFieldEnterKey(inputEvent){
      if (event.getKeyCode() == AdfKeyStroke.ENTER_KEY) {
       //get the input text component from the event
       var inputTextField = inputEvent.getSource();
       //the button is relative to the input text field so
       //relative search will do with no worrying about naming
       //containers
       var defaultButton = inputTextField.findComponent('cb1');
       //perform a partial submot
       var partialSubmit = true;
       AdfActionEvent.queue(defaultButton, partialSubmit);
       //Enter key does not need to go to server as we
       //queued a new event
       event.cancel();
   }
```

```
}
</af:resource>
```

For JavaScript to work, note the use of the `af:clientListener` on the input text field and the use of the **clientComponent="true"** configuration on the button.

Best practice invoking business services methods from JSF beans

Recently there was an increasing interest on OTN in best practices for invoking methods exposed on the ADF Business Components client interface. There exist two options to access methods exposed on ADF Business Components application modules and view objects from a managed bean through ADF.

- Call findDataControl(String) on the BindingContext and access the application module to invoke a method exposed on the Application Module or View Object client interface.
- Create a method binding on a PageDef level that binds to a method exposed on the Application Module or View Object client interface

Speaking of best practices, it's the second option – to use a method binding on the PageDef file – that I recommend for the following reasons

- ADF is about abstracting the view and controller layer from the business service implementation details. A method binding instead exposes an ID (the name o the binding) to the JSF developer, which they use to access the binding from a managed bean using the `OperationBinding` API. Signature or name changes of a method exposed on the business service thus don't require a change in the managed bean(s) referencing it. All changes would be applied in metadata.

- ADF provides the `OperationBinding` class as an abstraction for business services methods. Configuring business service method access on the PageDef file using method bindings exposes a single and consistent API to application developers.

- Programmatic access to a method exposed on the business service require more lines of code than accessing the same method through the binding layer

- Direct business service access violates the recommendation to always work through the ADF binding layer and not to bypass it.

To access a method binding from a managed bean, use the following code

```
BindingContext bctx = BindingContext.getCurrent();
BindingContainer bindings = bctx.getCurrentBindingsEntry();
OperationBinding operationBinding =
       bindings.getOperationBinding("name_of_method_binding");
//optional
operationBinding.getParamsMap().put("argumentName1",value1);
operationBinding.getParamsMap().put("argumentName2",value2);
//invoke method
operationBinding.execute();
if (!operationBinding.getErrors().isEmpty()) {
     //check errors
     List errors = operationBinding.getErrors();
     …
}
```

```
//optional
Object methodReturnValue = operationBinding.getResult();
```

Accessing WebLogic Server JDBC DataSource from Java in JSF

There may be a requirement for you to access a JDBC data source defined on the WebLogic Server (for example to query a database or database schema other than the one the application's business service is connected with.

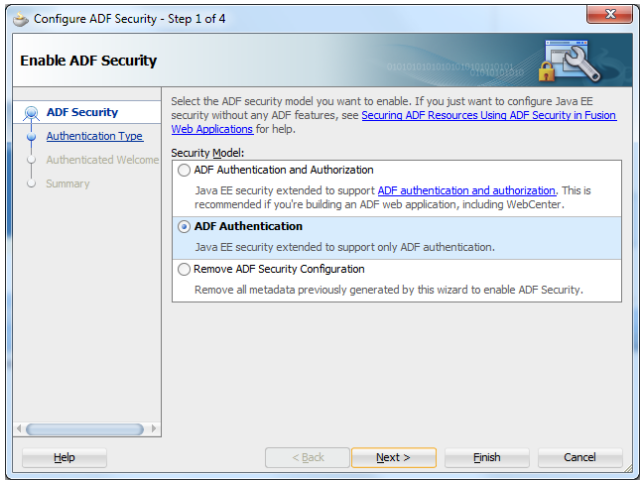
To access the JDBC DataSource, for example from a managed bean in JSF, you code like the following:

```
java.sql.Connection connection = null;
try {
  javax.naming.Context initialContext =
     new javax.naming.InitialContext();
  javax.sql.DataSource dataSource =
          (javax.sql.DataSource)initialContext.lookup(
            "java:comp/env/jdbc/hrconnDS");
  connection = dataSource.getConnection();
  …
 } catch(Exception e){
      e.printStackTrace();
      //or handle more gracefully
 }
```

In the example above, the JDBC DataSource is defined in WLS as "hrconnDS"

Solving JDeveloper 11gR2 issue with ADF Faces login page in IE8

Using Microsoft IE8, forms based authentication in which the login form is built with ADF Faces doesn't load and instead the browser console report complains about invalid or undefined ADF Faces JavaScript objects. The problem that has been reported for JDeveloper 11g R2 using ADF Faces with Facelets seems to occur only for applications using the ADF Security "ADF Authentication" configuration option in combination with an ADF Faces based login form.



Using the ADF Security "ADF Authentication" option configures the web.xml descriptor to protect the application Java EE context root from unauthenticated access.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>allPages</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>valid-users</role-name>
  </auth-constraint>
</security-constraint>
```

The **valid user** role name is mapped in the weblogic.xml file and references the WLS **users** group that all authenticated users are an implicit member of.

```
<security-role-assignment>
   <role-name>valid-users</role-name>
    <principal-name>users</principal-name>
</security-role-assignment>
```

Protecting the application Java EE root path from unauthenticated access however also means that web resources like pictures or styles sheets, or deferred loaded JavaScript libraries, are blocked until the user has authenticated. For the same reason customers often report missing images on their web login pages.

In the case of an ADF Faces login form, deferred resource loading becomes an issue because subsequent JavaScript requests are all blocked by the security constraint defined in the **web.xml** file. Surprisingly, this is only a problem in Microsoft IE8 and not with other browsers, which may indicate differences in the handling of login forms and their resource loading. To this time I am not able to say whether IE is doing it right or wrong. Fact however is that the problem only shows on this browser type and that it needs to be handled without opening a security hole, which you would do if you changed the security url-pattern to /faces/* in which case only JSF page requests would require authentication.

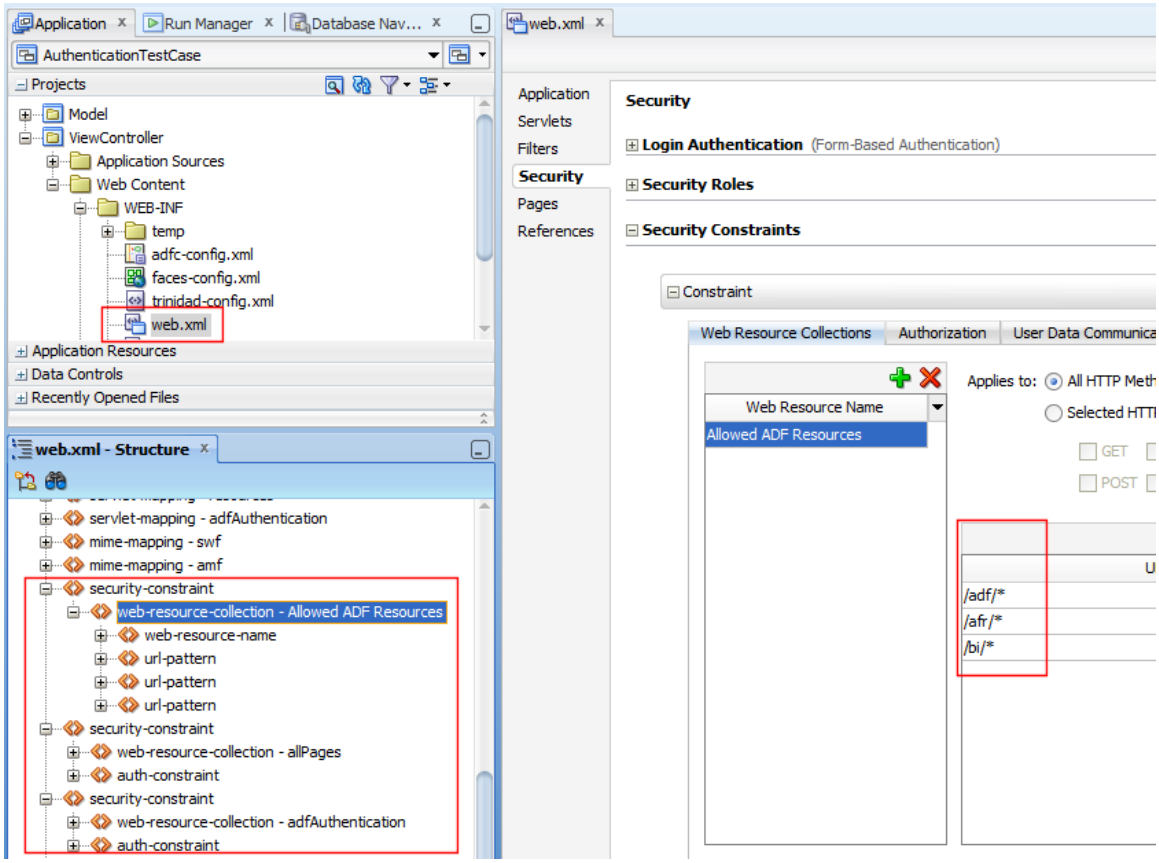
Investigating the problem, the following solution has been found by the ADF Faces team: ADF Faces UIs require additional resources that are loaded through the Trinidad resource loader servlet configured in **web.xml** file.

To allow resources to be loaded when using login forms built with ADF Faces (in which case the JSF page contains a HTML login form) you need to add another security constraint definition to the web.xml file:

```
<security-constraint>
   <web-resource-collection>
     <web-resource-name>Allowed ADF Resources</web-resource-name>
     <url-pattern>/adf/*</url-pattern>
     <url-pattern>/afr/*</url-pattern>
     <url-pattern>/bi/*</url-pattern>
   </web-resource-collection>
 </security-constraint>
```

The above security constraint should do for most of the login pages you would build with ADF Faces. However, if the login screen is more complex and e.g contains DVT graphs or maps, you may have to add more url-patterns for public (anonymous access), like

```
<url-pattern>/servlet/GraphServlet/*</url-pattern>
<url-pattern>/servlet/GaugeServlet/*</url-pattern>
<url-pattern>/mapproxy/*</url-pattern>
<url-pattern>/adflib/</url-pattern>
```



In my JDeveloper 11g R2 test-case project, I ended up with three security-constraint definitions added to the web.xml file: "Allowed ADF Resources", "allPages" and "AdfAuthentication".

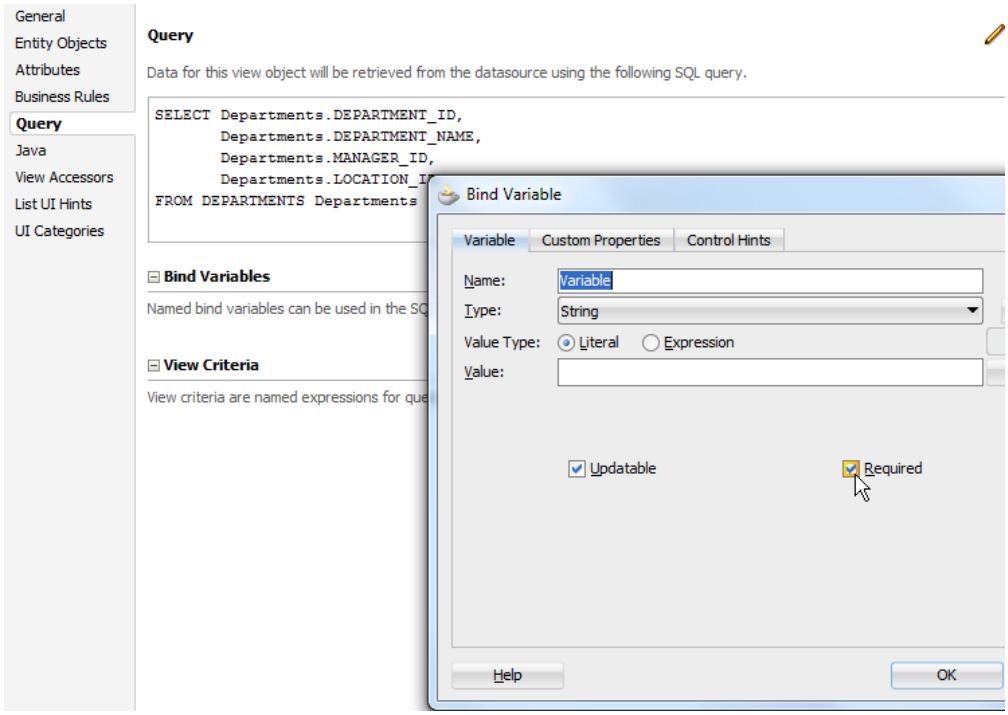
Always place the "Allowed ADF Resources" security constraint definition first in the web.xml file so it is looked at before all the others are so it takes precedence.

Note that both, the web.xml file and the weblogic.xml file open a visual configuration editor when you double click onto the respective file shown in the JDeveloper Application Navigator (Web Content → WEB-INF folder in the ViewController project.

The infamous Missing IN or OUT parameter error

Often error messages like "attempt to set a parameter name that does not occur in the SQL" and "Missing IN or OUT parameter at index" are caused by developers mixing required and optional bind variables in

their SQL queries WHERE clause. If you used bind variables in ADF Business Components SQL queries, make sure the bind variables are set to **required**. Bind variables that are marked as optional are only good to be used in View Criterias.



In a future version of Oracle JDeveloper, the two bind variable types (the optional ones used in View Criteria and the required ones used in the context of SQL queries) are visually separated in the layout of the View Object editor.

ADF Code Corner

OTN Harvest Spotlight

- Lucas Jellema

Lucas Jellema is an Oracle ACE Director for Fusion Middleware, consultant, trainer and instructor on diverse areas including Oracle Database (SQL & PLSQL), Service Oriented Architecture, ADF, Java, Oracle Forms and more.

Lucas authored the Oracle Press book: Oracle SOA Suite 11g Handbook and is a frequent presenter on conferences such as JavaOne, Oracle Open World, ODTUG Kaleidoscope, Devoxx and OBUG.

Blog: http://technology.amis.nl/blog/author/lucas

Twitter: https://twitter.com/#!/lucasjellema

ACC: What is your current role?

LJ: I am CTO at AMIS Services BV, an Oracle and Java specialist based in Nieuwegein, The Netherlands (founded in 1991).

Apart from the internal responsibilities, I act in projects as a solution architect, coach, sometimes developer as well as trainer and 'developer whisperer' (trying to instill confidence in experienced developers and build their enthusiasm for new areas.)

ACC: What is your IT background?

LJ: I started out doing some typing on the Sinclair ZX-81 in 1982; my first attempt at programming was a BASIC program to keep track of matches and results during the World Cup Football 1982 in Spain.

In University I learned Modula 2 and (Turbo) Pascal, briefly encountered Oracle (4 if memory serves me correctly)and SQL and went on to try my luck at Delphi.

In 1994 I started for real at Oracle Consulting with Oracle 6 soon succeeded by Oracle7 and life was good from then on.

Initially I best liked doing ER modeling. I struggled against having to dive into Java (late '90s). I surrendered and survived.

One of the milestones in my career: presenting at the ODTUG 1997 conference - my international debut.

After Forms and Designer, JDeveloper and BC4J, UIX and JHeadstart became my realm, and ADF from the moment that name was first introduced.

From 2006 onwards I have divided my time to SOA in addition to ADF and my two long time friends SQL & PL/SQL.

ACC:	How do you currently use Oracle JDeveloper and ADF?
LJ:	Being a SOA Developer as well as a Java/JEE and ADF developer, I pretty much use JDeveloper all the time - also for XML development and SQL & Database development. ADF is part of many projects in which I am involved. I am typically involved in several projects at the same time, usually working on the architecture, generic application facilities or advanced functionality. So I get to figure out how to best leverage generic features of ADF, how to integrate ADF with for example OPSS (security), Design Time at Run Time for customization or the database or how to implement the more interesting user requirements. Great fun all that. Especially when I can demonstrate how a seemingly impossible requirements, is not that hard to implement in ADF after all. This requires of course a good overview of the nooks and crannies of ADF - which is probably my specialty. We frequently use ADF in conjunction with WebCenter and SOA Suite, BPM and OSB.
ACC:	So far, what has been your biggest challenge in building Java EE application with Oracle ADF?
LJ:	It used to be: convincing non-Oracle Java developers to if not love JDeveloper and ADF to at least give them a serious, open minded chance. That has been getting easier over time though. The biggest technical challenge probably was and still is to find a productive way to have developers work on fairly small portions of a large application - using references to common data services and reusable components - that they can test in their own environment and then bring those portions together at deployment time into a single or preferably even multiple integrated web applications. We have made some headway using ADF Libraries - but it is still a challenge.
ACC:	Which feature of ADF was the greatest benefit to your project?
LJ:	This may come as a surprise, but I would say that the probably the remote debugging facilities (combined with the in-place redefinition of classes and now in 11gR2 ADF configuration files as well) has probably been the biggest productivity boost. I never write flawless code. Even the simplest code fragment will have a bug or two. I simply cannot afford to redeploy my application for every mistake I make. I constantly work in debugging. Being able to analyze the logic of my code (and the ADF framework) that way and inspecting the (ADF) data structures and class hierarchy at run time is of course an added bonus.
ACC:	Away from the on line help, what have been your most valuable sources of ADF knowledge?
LJ:	My fellow ADF Guild members - my ADF developing peers around the world - either in person, or through their blogs or tweets or directly exchanged emails. The willingness to help each other and the experience and knowledge available in the community is spectacular. I do not need to have many original thoughts: Google and access to a number of these community peers is really all I need.

ACC: Are you in any way actively involved in the ADF Community?

LJ: There are many ways in which I make contribution - great and small, valuable and trivial or even frivolous.

The most visible one is probably presentations at conferences such as ODTUG, Oracle Open World and JavaOne, OBUG and UKOUG; including a new session style I have helped to introduce: the 3 hour FMW Live Application Development demo that will run for the 3rd time in April at the OBUG 2012 conference. Most of the slides of these presentations are on Slideshare: http://www.slideshare.net/lucasjellema.

I have written the Oracle SOA Suite 11g Handbook (Oracle Press, 2010) and that contains a few chapters on SOA and ADF integration.

In 2004, we started the AMIS Technology Blog at http://technology.amis.nl/blog that today contains over 2000 articles, a substantial percentage of them on ADF. We also organize public community sessions at AMIS with ADF experts from inside our company or with VIP guests.

Every quarter, I write the Fusion Column in the ODTUG Technical Journal, frequently discussing ADF in combination with other Fusion Middleware components. I also regularly write articles for other media such as Oracle Technology Network, (Dutch) Java Magazine and other Oracle development magazines.

I teach Masterclasses (as part of the Oracle Celebrity Seminars series) as well classroom Training on ADF 11g and associated subjects.

Only for professional purposes, I use Twitter for disseminating tidbits on ADF and other technologies - stuff like interesting articles, tools, events or new software releases. I post the occasional (but very infrequent) comment on ADF EMG or OTN Forums- but I am not really the light of the party there.

I definitely make myself useful to the ADF community through my subtle and highly effective manipulation of Oracle Product Management (although the reverse is probably even more through).

Liaising with product management - channeling feedback from ADF customers and projects back to the development teams and outlining my interpretation of omnipresent needs and requirements is quite useful. There are not that many people that move around and see real world application of ADF in many different settings. I have that privilege and sharing my experiences is one of the ways in which I can be useful.

ACC: You are a SOA and Oracle ADF expert. How do you use ADF in your custom SOA projects?

LJ: When SOA Composite contain Human Tasks that require interaction from a human end user with the system, we will typically use ADF to develop the user interface for that particular task.

With the increasing popularity of BPM, we expect a growth in process oriented applications - that go beyond service oriented composites. These applications will contain many human tasks that require user interfaces, so I expect an increase in the ADF share in custom SOA projects.

This process orientation is typically combined with a requirement for operational insight in process execution. Or, in plainer terms: business dashboards that visualize progress and exceptions or incidents with the execution of business processes.

We can use BAM to collect process metrics and then create either BAM Dashboards or (active, that means auto-refresing) ADF DVT Dashboards based on the BAM Data Control. The latter makes it easier to integrate the dashboard with the user interfaces in which the user can respond to alerts and other run time findings - so that has my preference.

We are contemplating the move from the Database Adapter as a means of integrating the database into SOA Composite application, to ADF BC exposed as a Web Service. Apparently, that is the Fusion Applications does it - so it is at least worth considering. Until now, usually the database adapter (internally based on EclipseLink) is still the favorite.

Of course all run time user interfaces for SOA Suite - such as Enterprise Manager FMW Control, SOA Composer, BAM (Business Activity Monitoring) and BPM Worklist - are created with ADF so regardless of whether we use it explicitly - it is still there.

ACC: What are your top 3 ADF best-practices for SOA developers?

LJ:

i) Work with the framework - and not against it. For example: leverage the ADF Data Controls instead of programmatically accessing traditional data services from the web application. The framework can only help if you let it help you. Otherwise, it gets in the way (or you do).

ii) JMS is a fine and relatively easy to use way to asynchronously exchange information between Java (and ADF) Web Application and the world of the SOA Suite. Use it!

iii) Take a close look at the ADF Data Visualization Tags (DVT) that provide wonderful opportunities to visualize data and in doing so allow quick and intuitive interpretation.

ACC: What should SOA developers avoid when using Oracle ADF?

LJ: SOA Developers - and any ADF developer by the way - should carefully consider how to access data from enterprise resources. Especially for SOA Developers, the first option is likely to be the use of Web Services in combination with the ADF Web Service Data Control.

However, if that is the road they choose, they create a direct and typically undesirable dependency on the WebService definition: the web page may need changing when the XSD under the WebService changes. Additionally, the WebService Data Control does not provide support for caching data.

These, among other reasons, are strong arguments to avoid using the WS Data Control - apart from simple, fine grained use cases. Using a JAX-WS proxy - a generated class that invokes the WebService - that can wrapped in an application friendly POJO that may provide caching and that can be published as a Data Control is a far better approach.

ACC: ADF Genie grants you a wish, what would you ask for?

LJ: When I was just a kid, I had devised a strategy for this question: my first wish would be for three more wishes. That would give me some breathing space. I suppose that is not an option

open to me in this situation?

If you ask me this same question tomorrow, I will give you a different answer. And I believe I can come up with a different wish for each day of the year.

Let me give you my wish for today: the ability to deploy demo ADF applications to a free area in the Oracle Public Cloud. Not for production purposes but only to make running code sample available to the community.