ADF Code Corner

005. How-to bind custom declarative components to ADF

Abstract:

Declarative components are reusable UI components that are declarative composites of existing ADF Faces Rich Client components. Using declarative components in JDeveloper 11, application developers can build their own components from one to many existing JavaServer Faces components. Usecases for which declarative components make sense include special layout definitions built out of nested existing layout components or frequently used user dialogs. This how-to document explains how declarative components are build in JDeveloper 11, how the component can be configured to read its data from the ADF binding layer and how partial page refresh (PPR) can be used to conditionally refresh a declarative component.

twitter.com/adfcodecorner

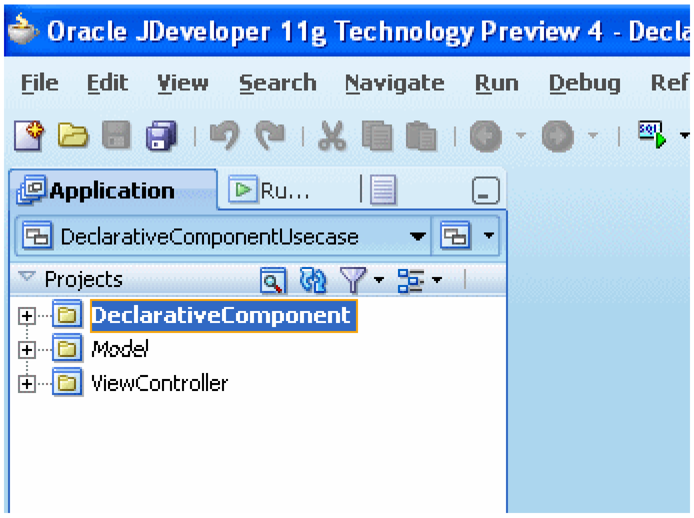
Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
26-MAY-2008

Introduction

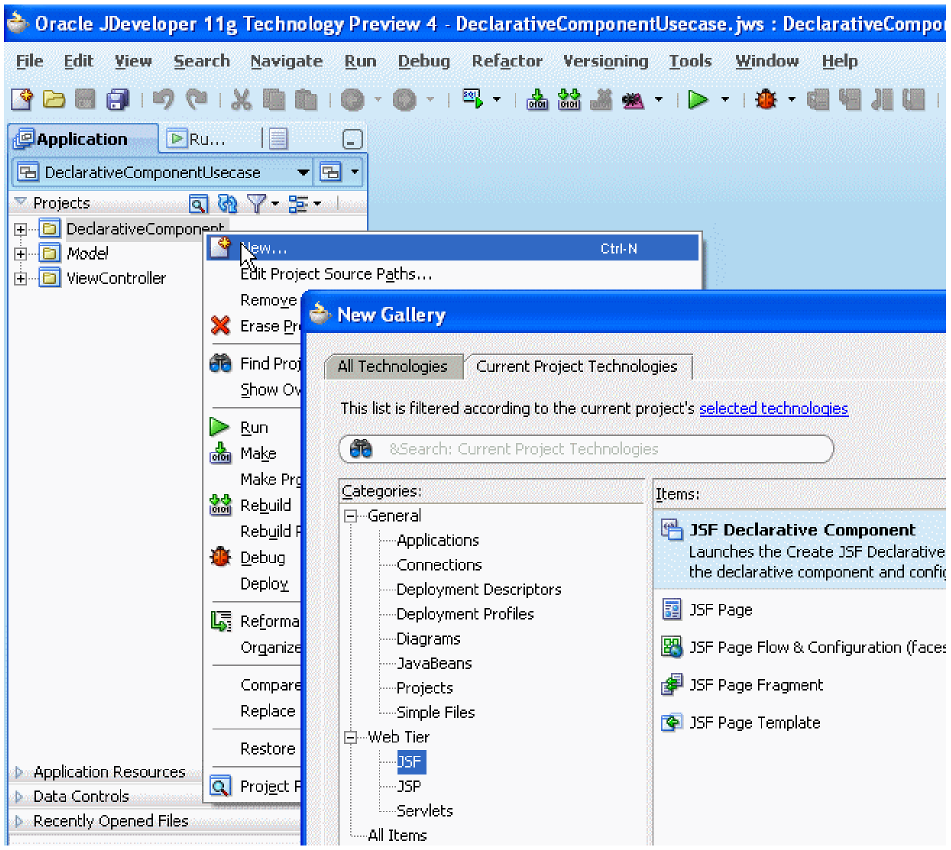
The sample workspace built for this how-to document shows a table that updates a declarative component whenever the row currency changes. The declarative component shows the first name, last name and mail information of the current user. Following this how-to document and downloading the sample workspace you learn how to create and deploy declarative components, how to use attribute bindings to reference the current row of a table and how to enable PPR for declarative components.

Building and deploying a declarative Component

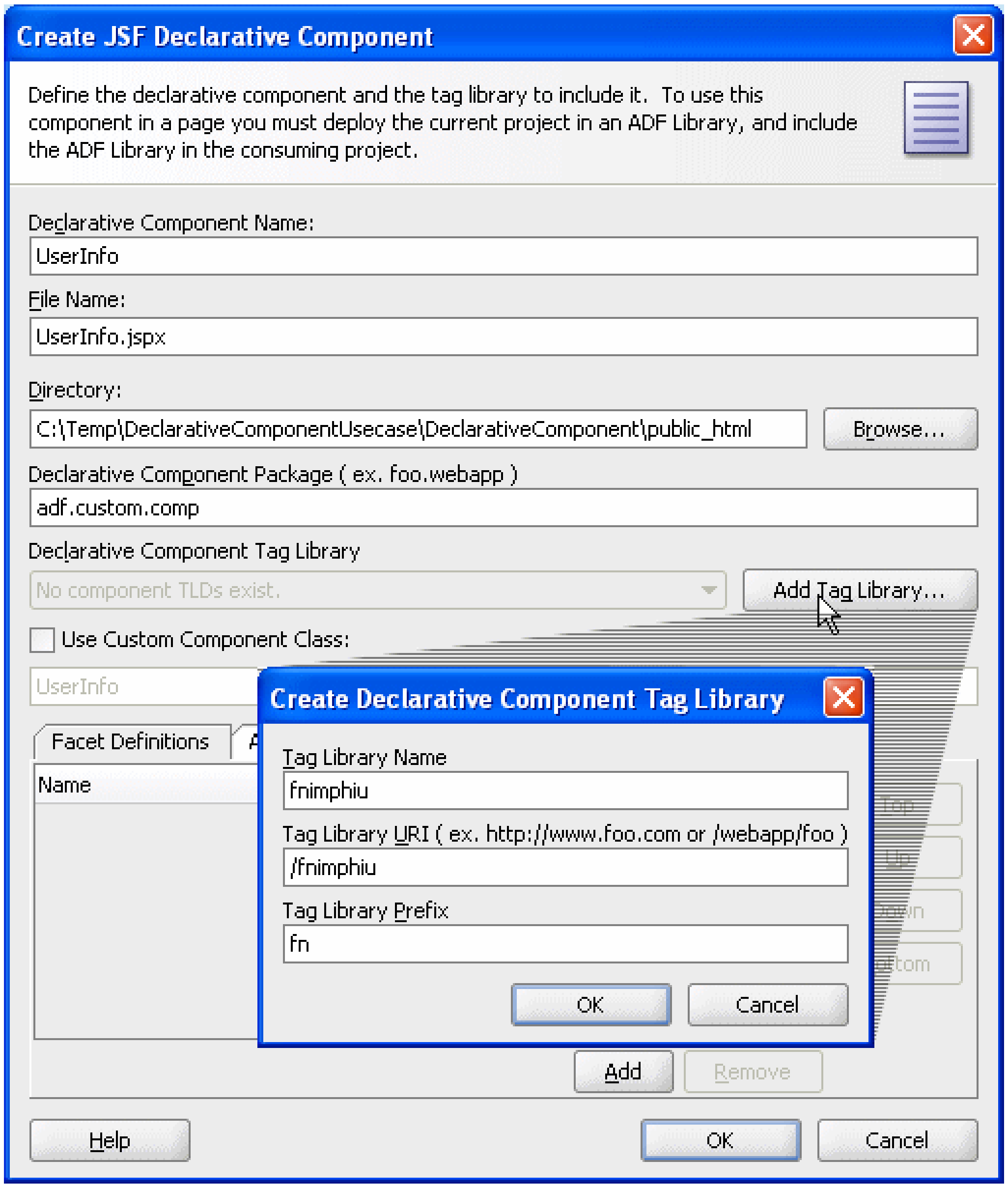
You create a declarative component as described in Shay Shmeltzer's online tutorial. To reuse a declarative component, it needs to be deployed in an ADF Library that is created from its own project in this example. The technology scope of this project is set to "ADF Faces".



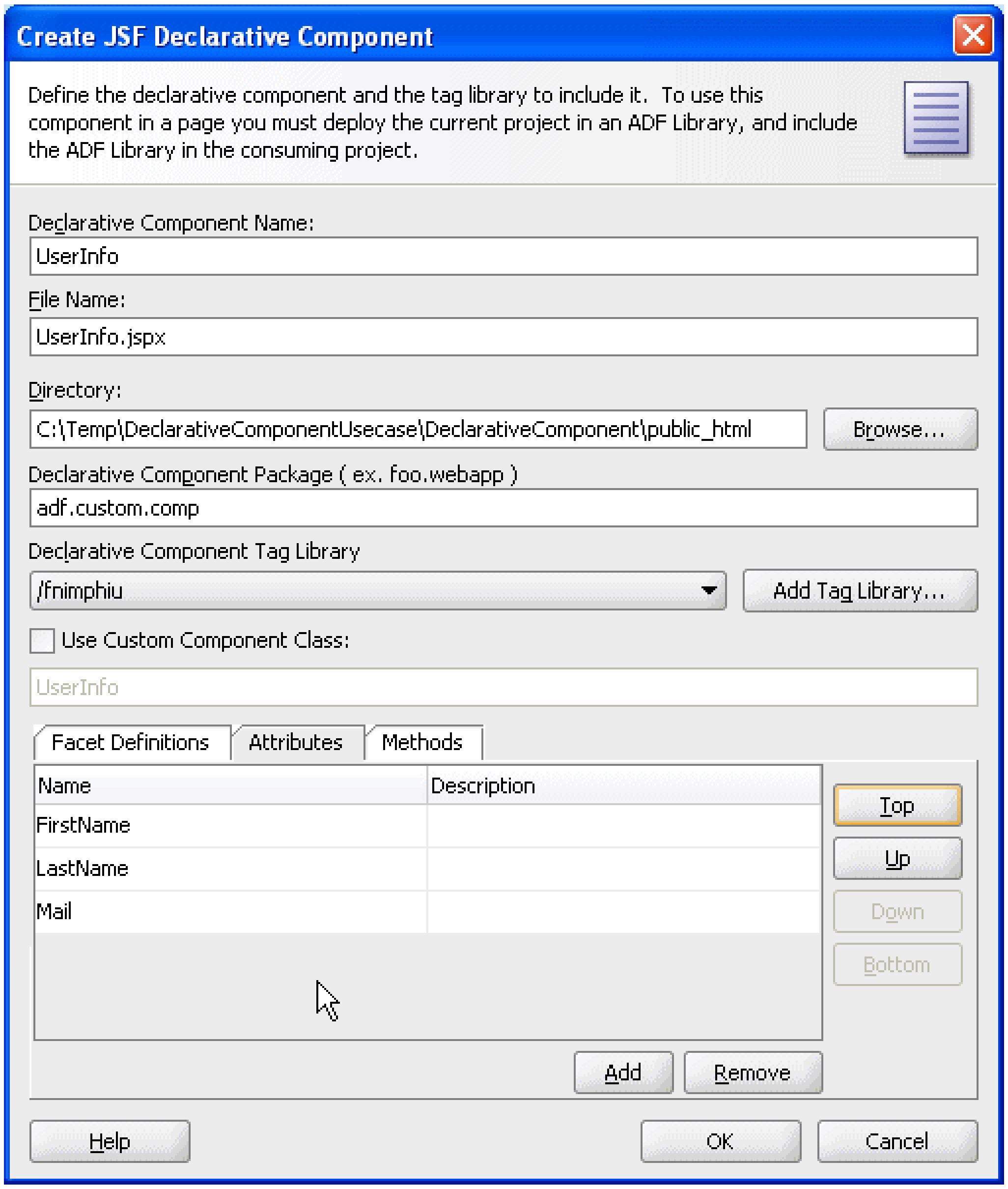
On the project root, choose **New** from the context menu to access the Oracle JDeveloper **New Gallery**. Under the JSF node, select the **JSF Declarative Component** option to start the wizard that leads you through the initial component creation.


Oracle JDeveloper 11g Technology Preview 4 - DeclarativeComponentUsecase.jws : DeclarativeCompo
File Edit View Search Navigate Run Debug Refactor Versioning Tools Window Help
Application
Ru...
DeclarativeComponentUsecase
Projects
DeclarativeComponent
Model
ViewController
New... Ctrl-N
Edit Project Source Paths...
Remove
Erase Pr
Find Proj
Show Ov
Run
Make
Make Pr
Rebuild
Rebuild
Debug
Deploy
Reforma
Organize
Compare
Replace
Restore
Project
Application Resources
Data Controls
Recently Opened Files
New Gallery
All Technologies
Current Project Technologies
This list is filtered according to the current project's selected technologies
&Search: Current Project Technologies
Categories:
General
Applications
Connections
Deployment Descriptors
Deployment Profiles
Diagrams
JavaBeans
Projects
Simple Files
Web Tier
JSF
JSP
Servlets
All Items
Items:
JSF Declarative Component
Launches the Create JSF Declarative
the declarative component and confi
JSF Page
JSF Page Flow & Configuration (faces
JSF Page Fragment
JSF Page Template

Fill in the component template to provide a name for the component, **UserInfo** in this example, a file name and package structure. The custom component is added as a tag library after deployment, which is why you need to use the Add **Tag Library** button to define the library's name, namespace and prefix.


Create JSF Declarative Component
Define the declarative component and the tag library to include it. To use this component in a page you must deploy the current project in an ADF Library, and include the ADF Library in the consuming project.
Declarative Component Name:
UserInfo
File Name:
UserInfo.jspx
Directory:
C:\Temp\DeclarativeComponentUsecase\DeclarativeComponent\public_html
Browse...
Declarative Component Package (ex. foo.webapp)
adf.custom.comp
Declarative Component Tag Library
No component TLDs exist.
Add Tag Library...
Use Custom Component Class:
UserInfo
Facet Definitions
Name
Add
Remove
Help
OK
Cancel
Create Declarative Component Tag Library
Tag Library Name
fnimphiu
Tag Library URI (ex. http://www.foo.com or /webapp/foo)
/fnimphiu
Tag Library Prefix
fn
OK
Cancel

For the declarative component you can create attributes, facets and methods as needed. This can be done later on as well using the property palette. Use attributes to pass data in from the outside to the declarative components, which is what we use within this example. Use a facet for the application developer to be able to add other components as child components to your declarative component. This option is useful if your declarative component is a composite layout container you created to enforce a consistent layout for a specific functionality in a JSF page. Similar, you use methods to map action command contained in the declarative component to a method accessible through Expression Language on the containing page.


Create JSF Declarative Component

Define the declarative component and the tag library to include it. To use this component in a page you must deploy the current project in an ADF Library, and include the ADF Library in the consuming project.

Declarative Component Name: UserInfo

File Name: UserInfo.jspx

Directory: C:\Temp\DeclarativeComponentUsecase\DeclarativeComponent\public_html

Declarative Component Package (ex. foo.webapp): adf.custom.comp

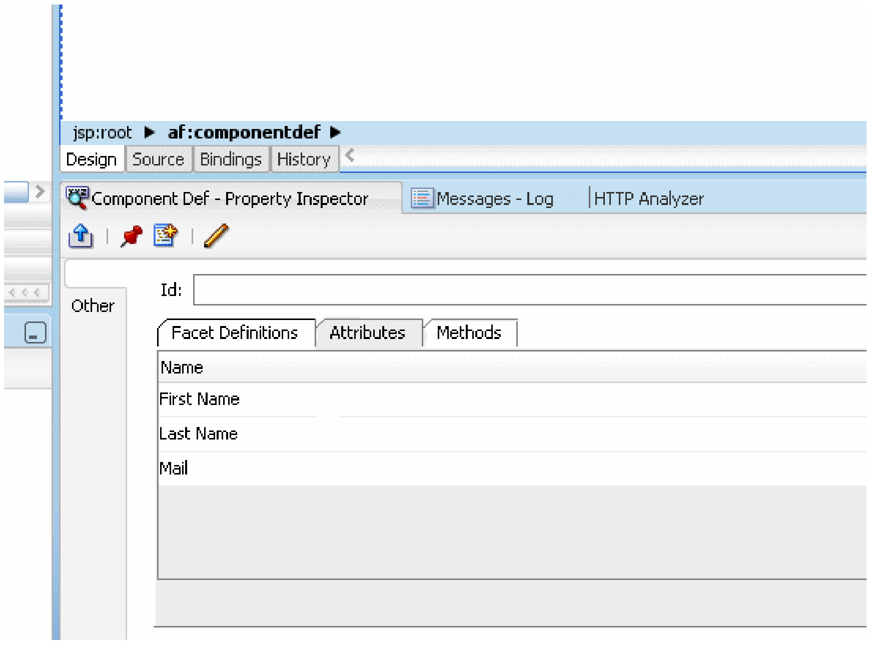
Declarative Component Tag Library: /fnimphiu

[ ] Use Custom Component Class: UserInfo

Facet Definitions | Attributes | Methods

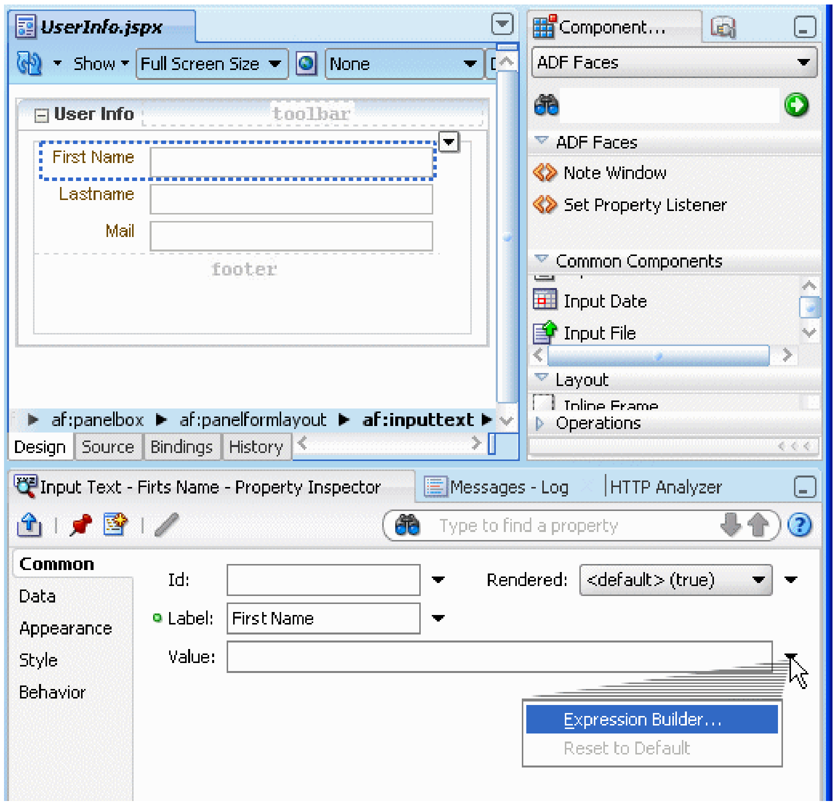
Name	Description
FirstName	
LastName	
Mail	

Once the declarative component is configured and shows in the visual editor of Oracle JDeveloper 11, you can use the property palette to further edit or add facet definitions, attributes and methods. The screenshot below shows the property palette as it is current in Oracle JDeveloper 11 TP4. The property palette's look and feel will change in releases after TP4, though the exposed information remains.

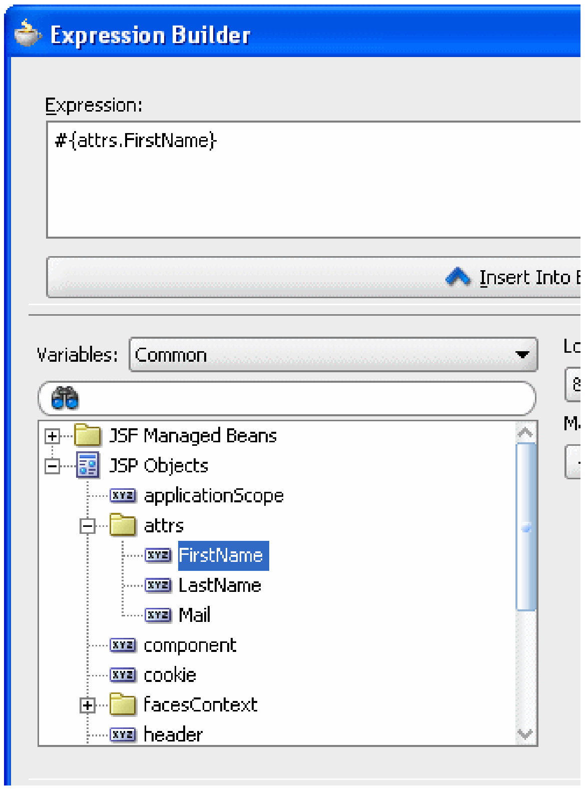

jsp:root ▶ af:componentdef ▶
Design Source Bindings History
Component Def - Property Inspector
Messages - Log
HTTP Analyzer
Other
Id:
Facet Definitions
Attributes
Methods

Name
First Name
Last Name
Mail

For this example, I created a panel box with 3 text fields to show the user information. I could have added a submit button to perform updates to the data as well. Selecting a text field in the visual editor - or the structure window - allows you to use the **Expression Builder** from the property palette to map the component's value property with one of the defined attributes. Any change to the attribute's value from then on shows in the text field as its value.

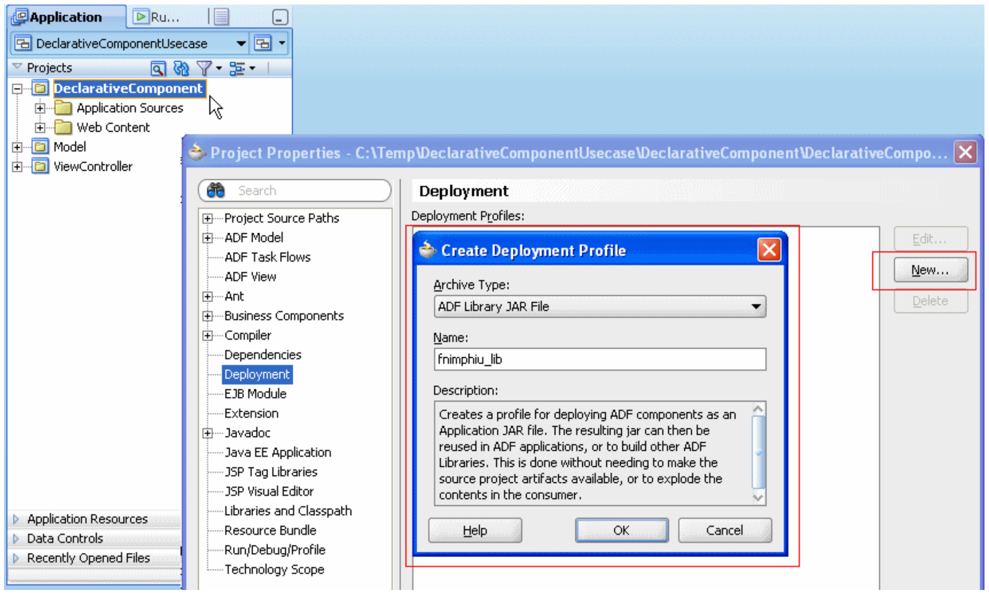


As you see, the defined attributes show under the **attrs** node of the **JSP Object** node for easy access. Select an attribute and press the **Insert** button or double click onto the attribute to build the Expression Language reference to it.



```
#{attrs.FirstName}
```

Once the declarative component is done, **double click** onto the project node, or choose **Properties** from the context menu, to open the project's properties. Choose the **Deployment** node to create a new deployment profile, which you set to be of type **ADF Library JAR file**.

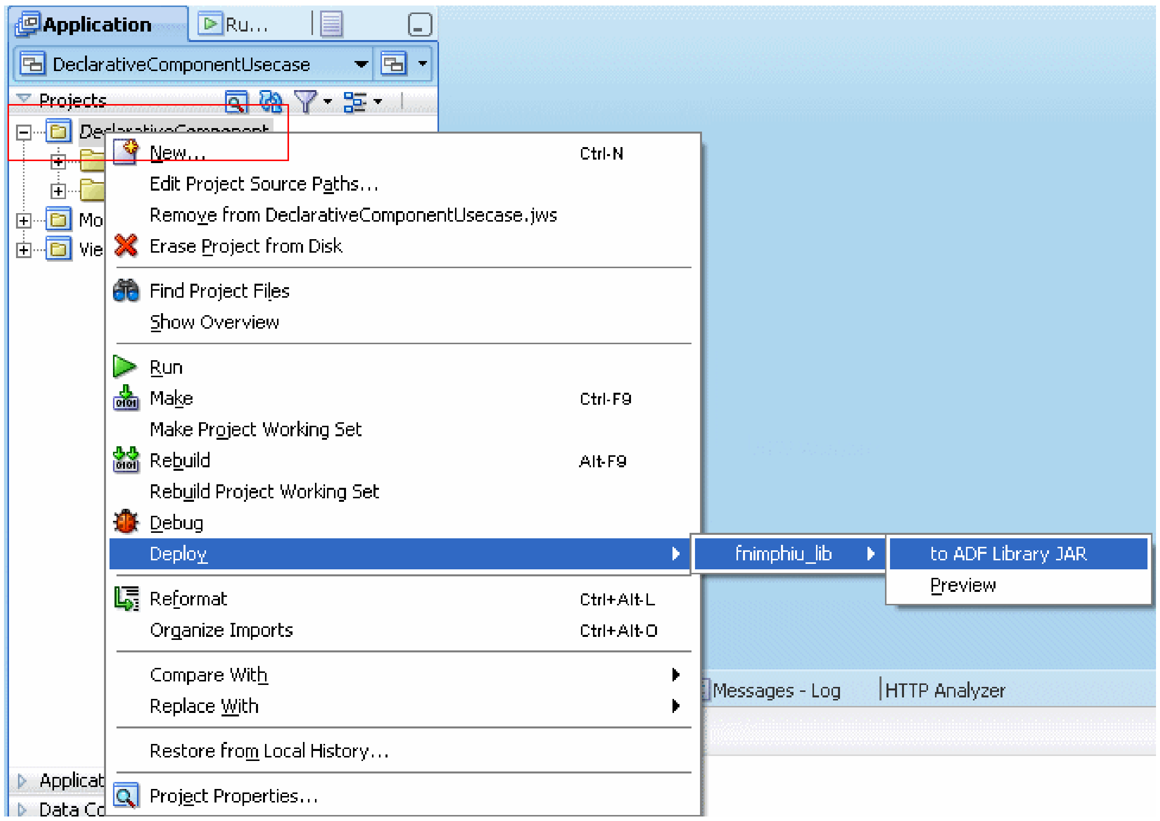


Archive Type: ADF Library JAR File

Name: fnimphiu_lib

Description: Creates a profile for deploying ADF components as an Application JAR file. The resulting jar can then be reused in ADF applications, or to build other ADF Libraries. This is done without needing to make the source project artifacts available, or to explode the contents in the consumer.

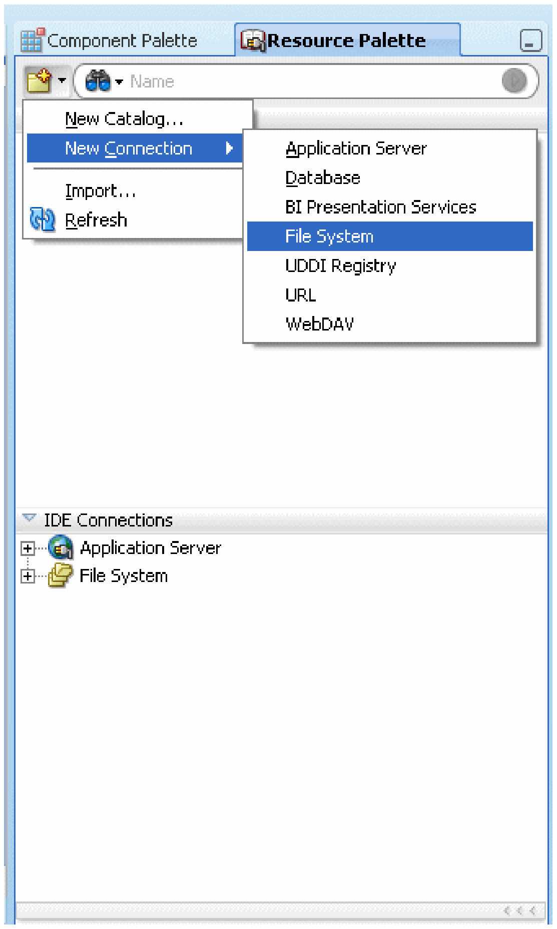
After deploying the library, you'll find the generated JAR in the **deploy** directory of the declarative component project. At least for Oracle JDeveloper 11 TP4, make sure that the deploy directory is within a path **that does not contain a blank** character or copy the generated JAR file to a directory that doesn't contain one. This restriction will be addressed in a later version of Oracle JDeveloper 11.



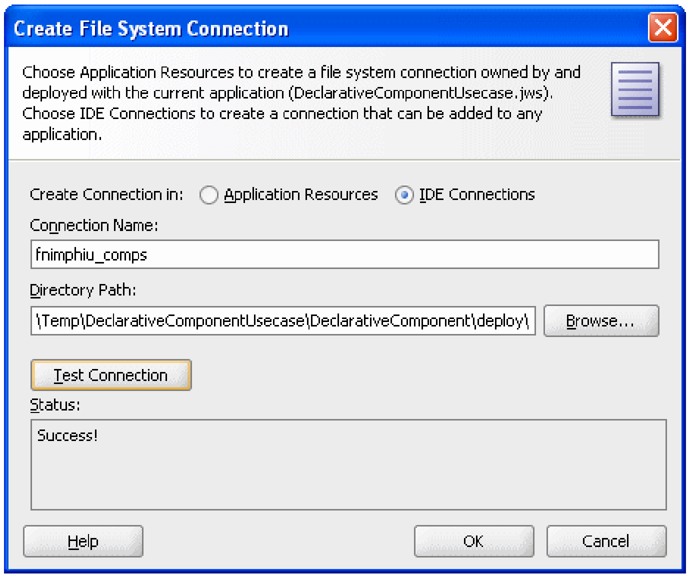
That is all what it takes to build a new declarative JavaServer Faces component, ready to be used in a web application project. The really nice to say about declarative components is that it allows you to create reusable building blocks for your personal application development the easy way without copy and pasting code snippets between projects.

Adding a declarative component to a JSF page

To add a declarative component to the Oracle JDeveloper web project, you need to import the created ADF Library jar file and the contained tag library. While this can be done through the **Manage Libraries** option of the **Tools** menu, there is an even more elegant option, which is to build a new IDE connection to a directory on the file system. Using an IDE connection you create a resource catalog reference that allows you to easily browse and search for existing declarative components to add them to the project.



Following the instructions shown in the image above (Use **View | Resource Palette** to open the resource catalog window), you create a new **File System** connection by clicking onto the **Create folder** icon and choosing **New Connection** from the context menu.


Create File System Connection

Choose Application Resources to create a file system connection owned by and deployed with the current application (DeclarativeComponentUsecase.jws). Choose IDE Connections to create a connection that can be added to any application.

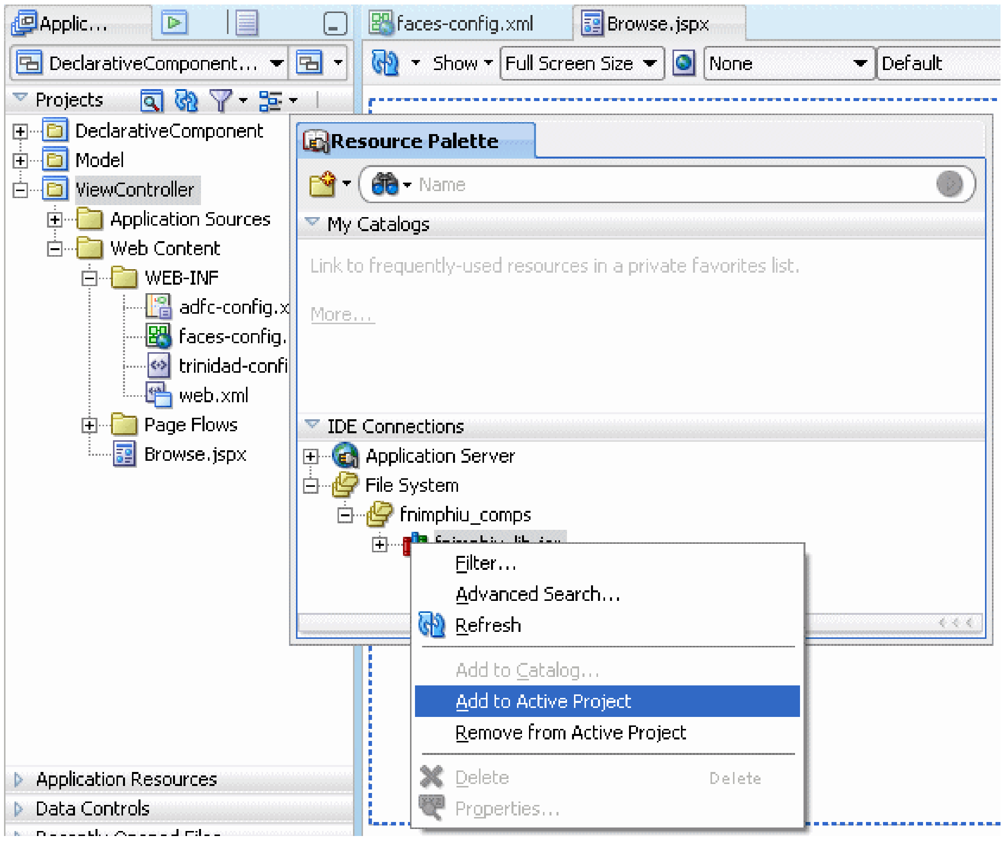
Create Connection in: Application Resources [✓ IDE Connections]

Connection Name: fnimphiu_comps

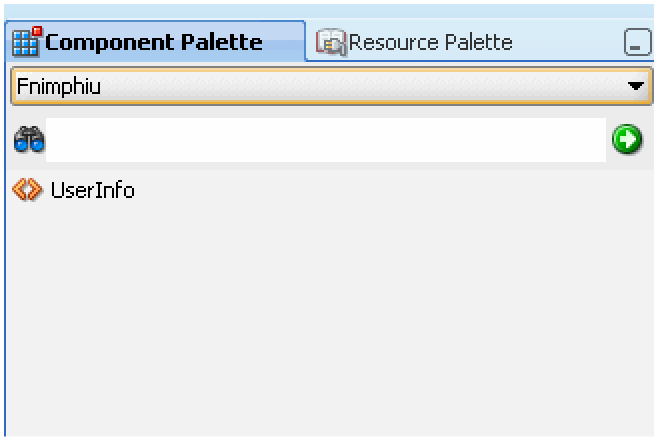
Directory Path: \Temp\DeclarativeComponentUsecase\DeclarativeComponent\deploy\

Status: Success!

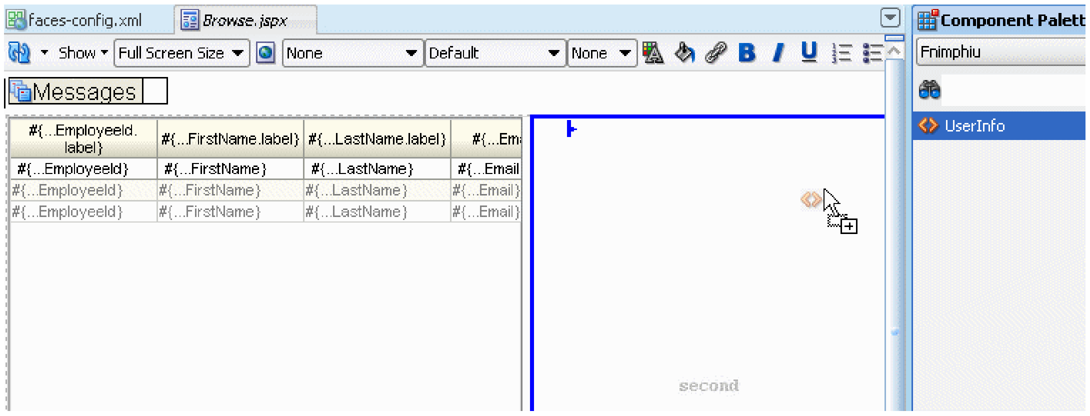
Create a new IDE connection and point the directory path to the directory that contains the ADF library file for the declarative component you just created. You can test the connection to ensure no typo prevents accessing the resources.



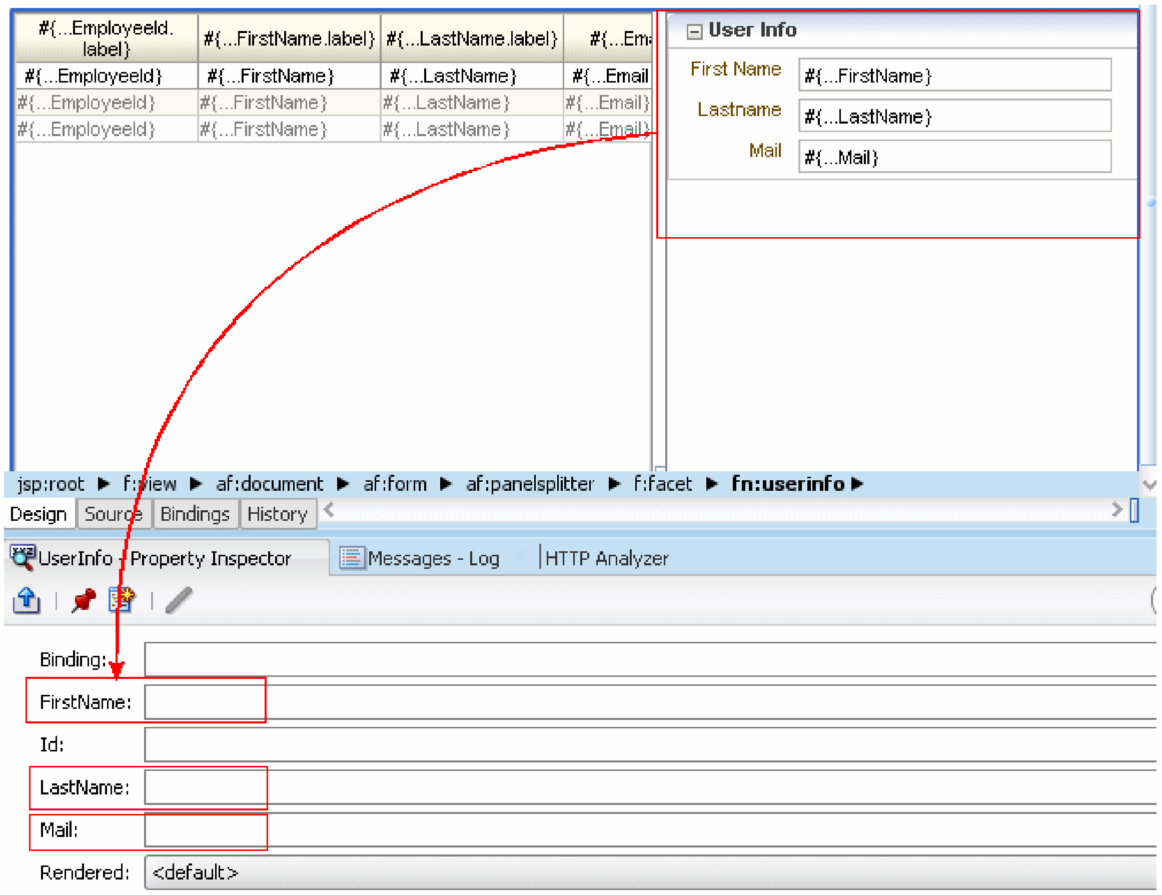
Select the web application project and expand the **File System** node. It shows a sub node with the name of the create IDE connection name. Below this node you find the deployed declarative component JAR file entry. Select it and choose **Add to Active Project** from the context menu to add the library to the current web application project.



Soon after the ADF library gets imported in Oracle JDeveloper, a new node shows in the **Component Palette** for you to add the contained declarative components to the web application project.



Note that in this example, the user info panel appears read only, which means that no additional ADF Faces components can be added to it. This is because the example uses attributes to pass data in, but no facets to add nested components. If you want to allow developers to add ADF Faces components to your declarative component, in which case your declarative component becomes the parent of the added component, you need to add a facet to the declarative component.

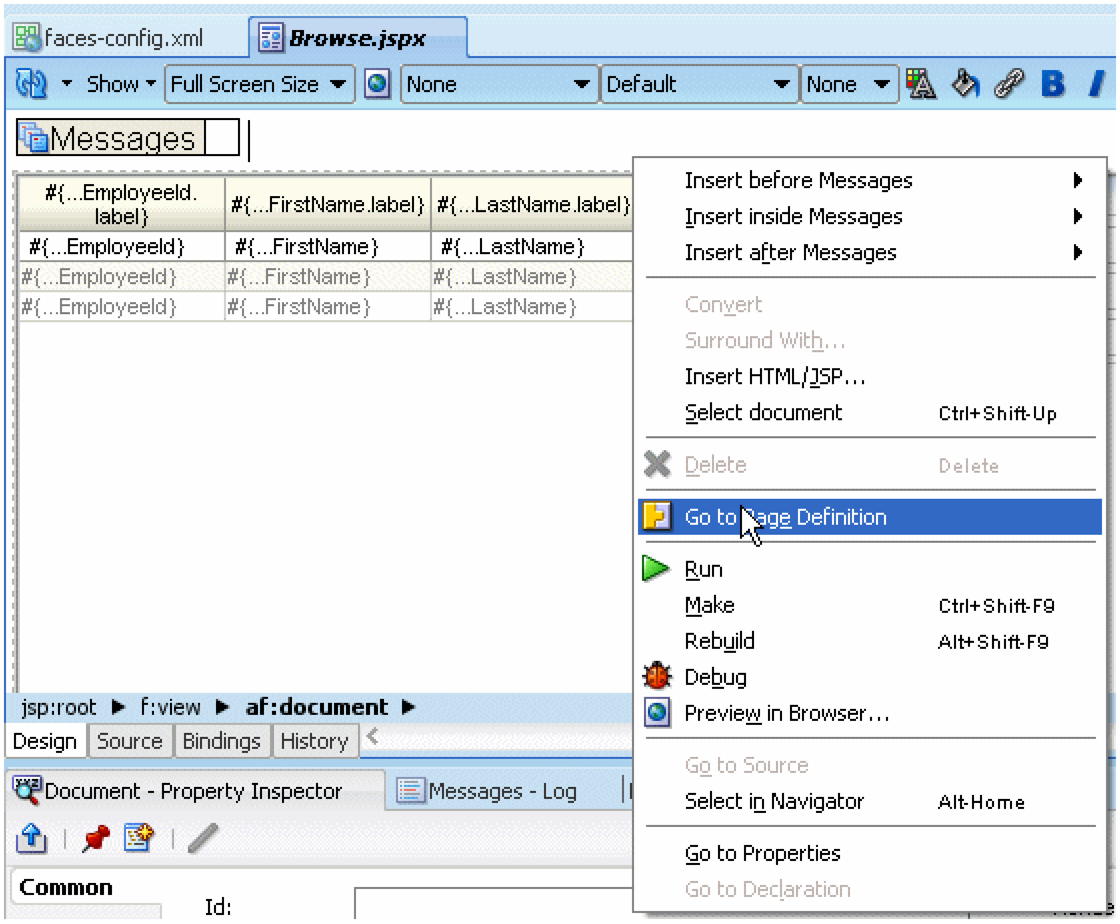


The really cool feature of declarative components is that selecting the declarative component in Oracle JDeveloper, using the visual editor or the Structure window, the property inspector (ctrl+shift+P) exposes the defined attributes and methods as properties. In above image the FirstName, LastName and Mail attributes are exposed in the property inspector.

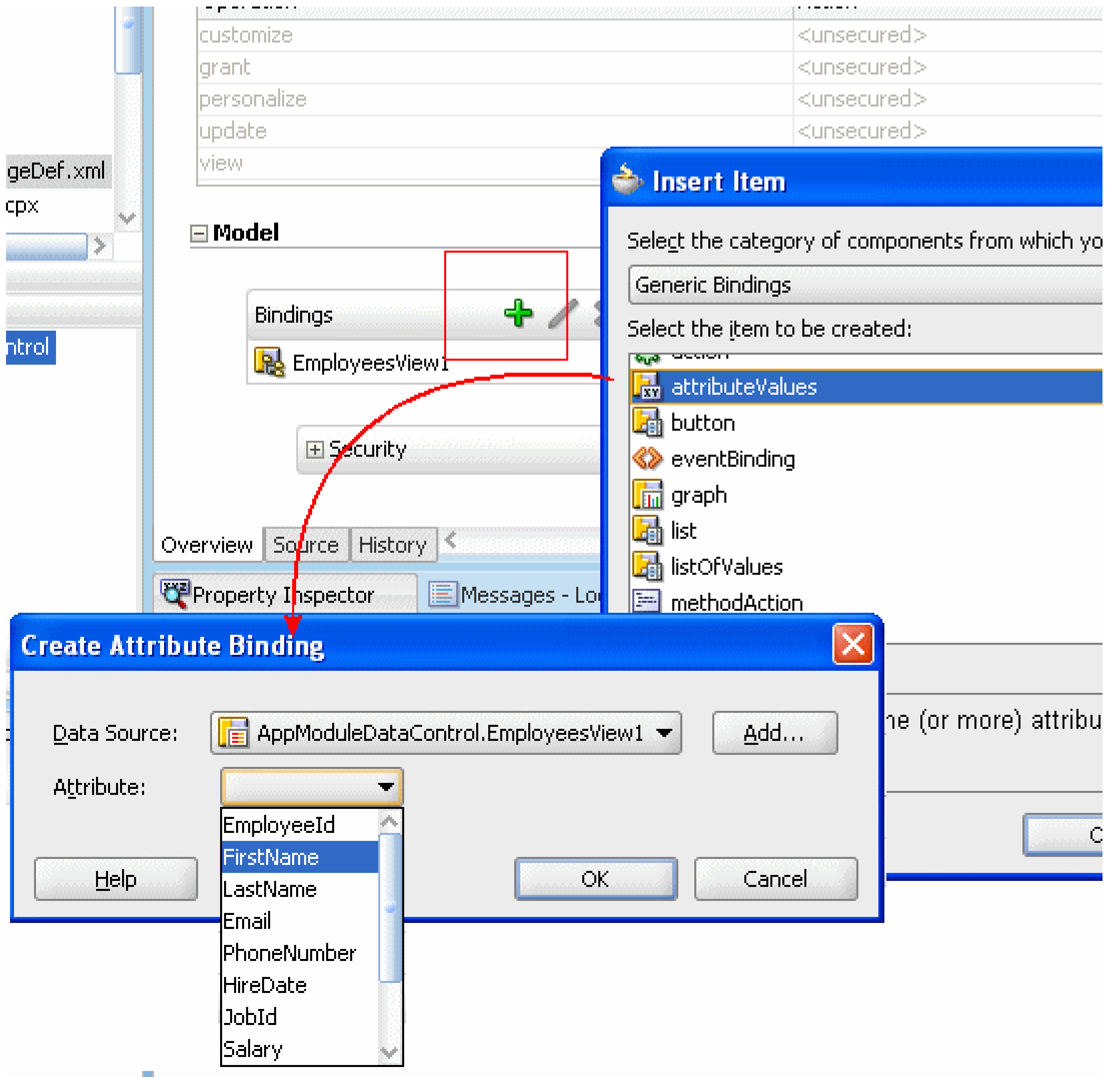
Databinding declarative components

While you could have added hard coded values to the exposed attribute properties of the declarative component, using a Expression Language string to look up the ADF binding layer is closer to what your requirements would be in a production environment.

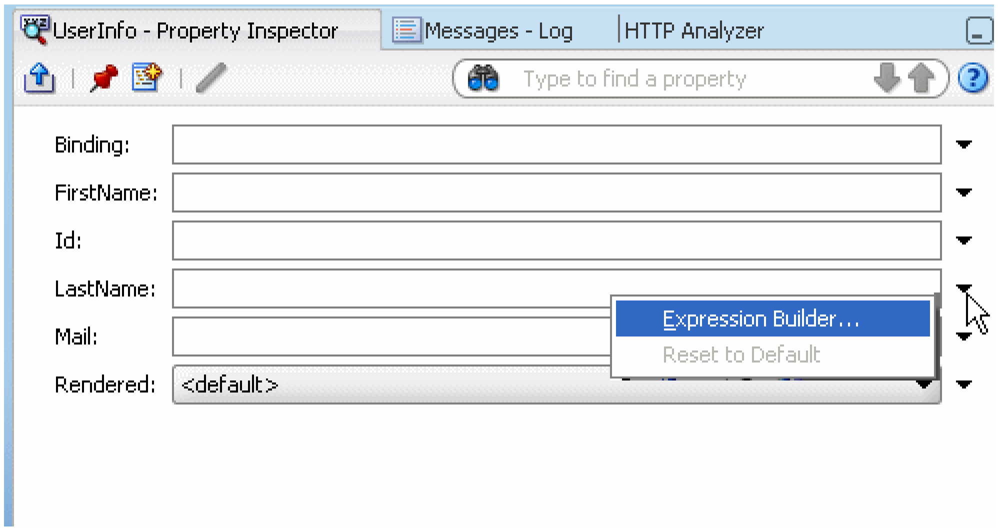
In the example provided with this document, the web page consists of a splitter component, with an ADF bound table added to the first splitter area. A page definition file exists and we navigate to this fie to add additional attribute bindings that are then referenced by the declarative component.



Note that the table binding uses a single node tree binding, which is bound to the EmployesView1 iterator. Selecting a row in the table always sets this row as the current row in the underlying iterator. The action performed for this uses partial page rendering (its doing the Ajax thing if you want to see it this way), which means that no page reload is required for this. All that we need to wire up the declarative component properties to the current row in the iterator is to create a *"window"* to the current row. This *"window"* is built through attribute bindings that expose a single attribute to the binding layer.

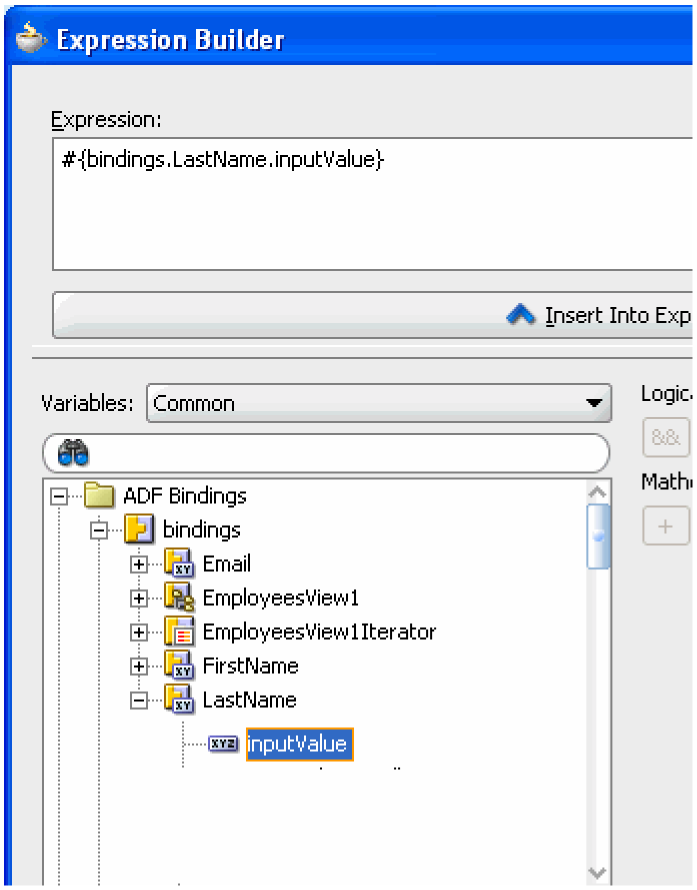


After creating attribute bindings for the the FirstName, LastName and Email attribute or the ViewObject, you are good to bind the declarative component's exposed properties to ADF. For this you use the property palette and choose **Expression Builder** from the context menu (click onto the black triangle icon).



Binding:
FirstName:
Id:
LastName:
Mail:
Rendered: <default>

It is important to node that you bind the attributes to the attribute binding's **inputValue** property. To use the attribute binding itself is a frequent error among application developers that use ADF. Data is accessed through the inputValue property of the attribute binding.



Expression:

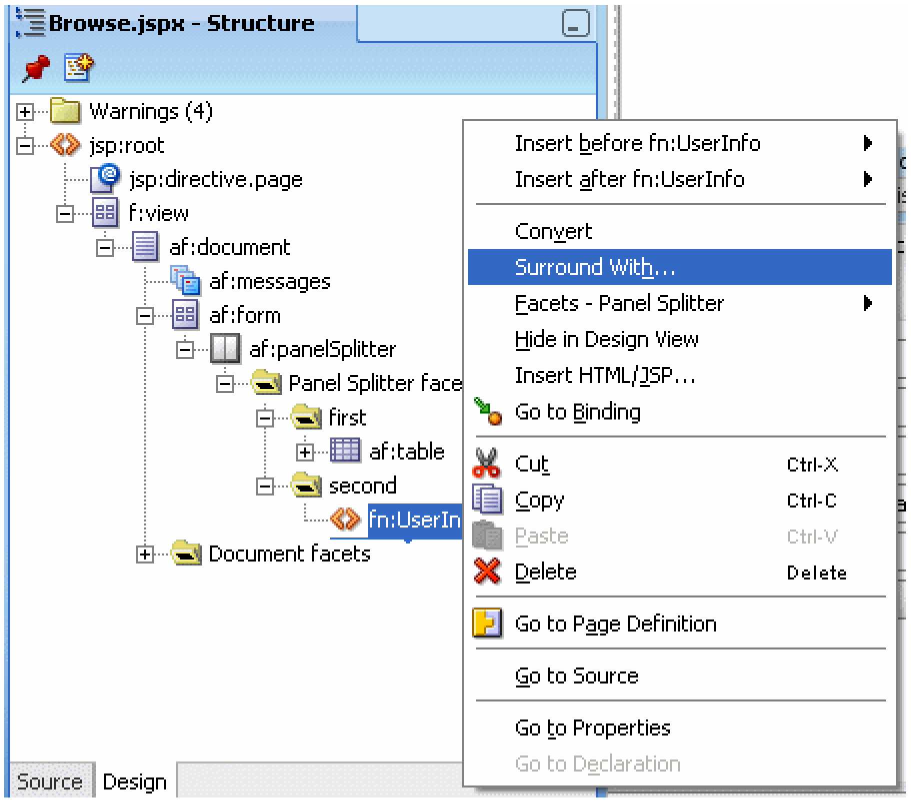
```
#{bindings.LastName.inputValue}
```

Variables: Common

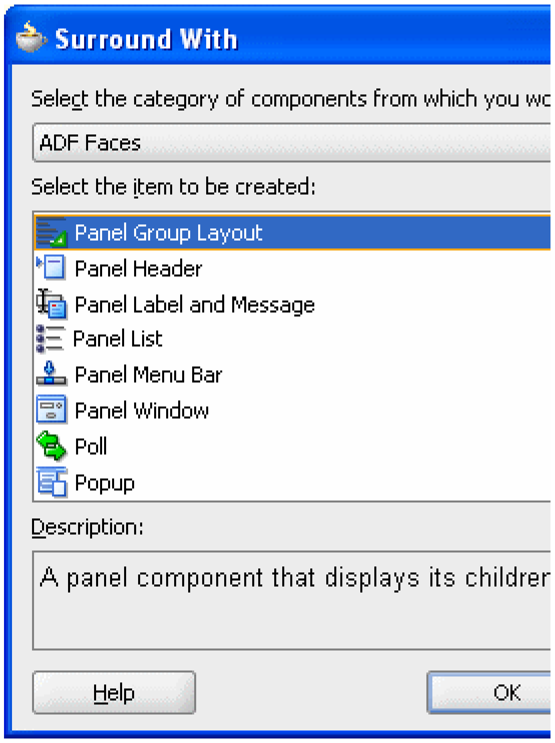
Once all of the declarative component's attribute are bound to attribute bindings in ADF, your are done with this part of the job. All that is missing is to ensure the declarative component gets refreshed as soon as the row currency of the table changes.

Enabling declarative components for Partial Page Rendering

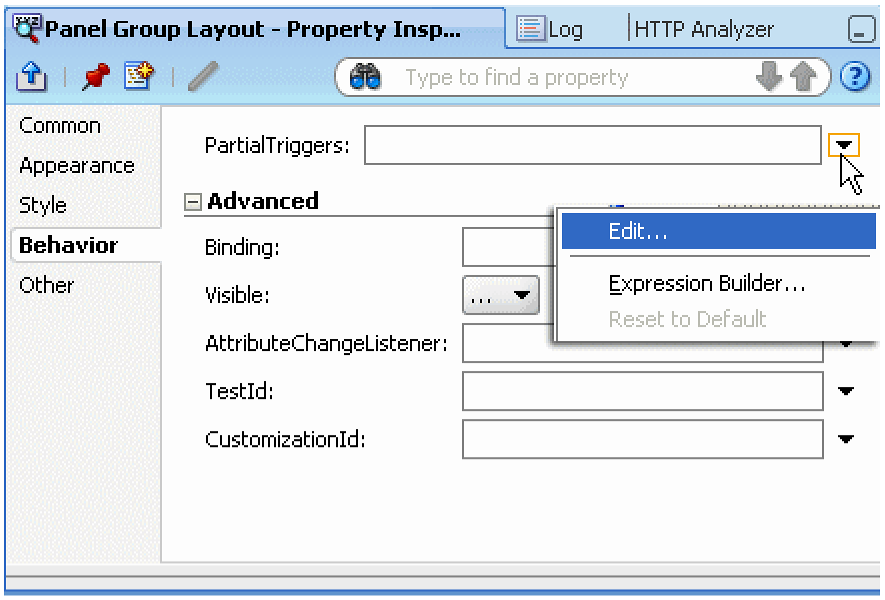
Declarative components themselves don't provide a partial page refresh property that you can set to the ID value of a triggering component. However, you can wrap the declarative component with a component that does support PPR. In this example I surrounded the declarative component with a **PanelGroupLayout** component to then reference the table's ID property form the **Partial Trigger** property.



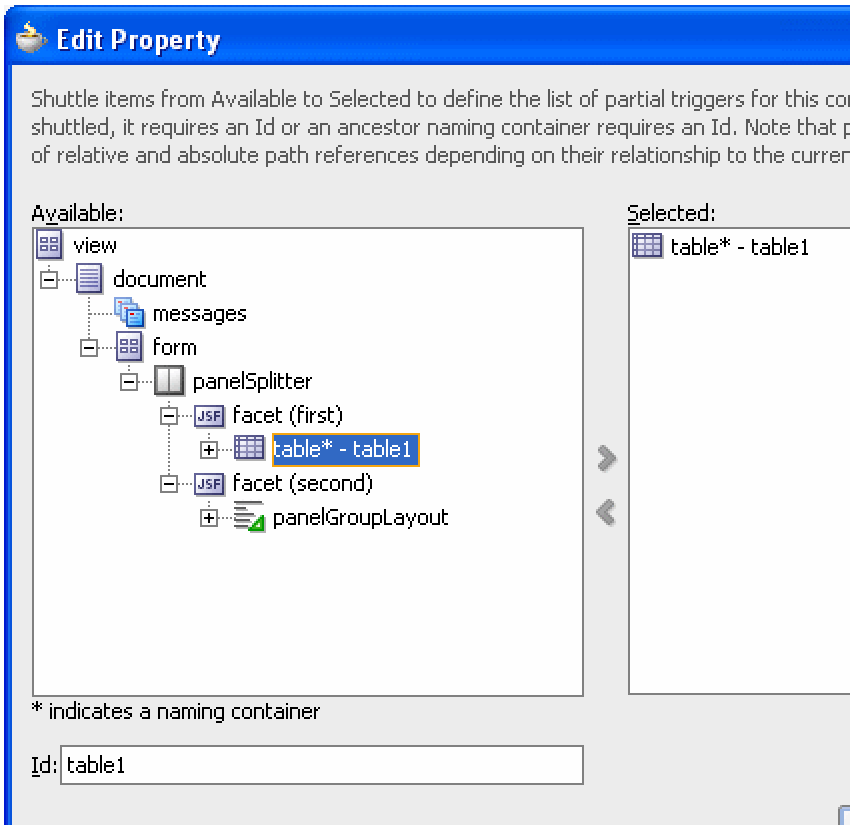
You can choose any layout container as a wrapper for the declarative component.



You can use the **Edit** option of the wrapper component's Partial Trigger Property to browse the page's component tree for the component that triggers the refresh. Note that this approach is less error prone than typing the ID reference in directly because chances are that the triggering component itself is wrapped in a container, like panelCollection in the table case.



While you can define the ID property for the table component within this working step, note that because of a refresh problem in Oracle JDeveloper 11 TP4, you will have to run the dialog twice to get the EL reference created. In this case it is better to define the table's ID property before opening the Expression Language builder on the partial trigger property of the panelGroupLayout.



Shuttle items from Available to Selected to define the list of partial triggers for this co
shuttled, it requires an Id or an ancestor naming container requires an Id. Note that p
of relative and absolute path references depending on their relationship to the curre

Available:

```
view
document
    messages
    form
        panelSplitter
            facet (first)
                table* - table1
            facet (second)
                panelGroupLayout
```

Selected: table* - table1

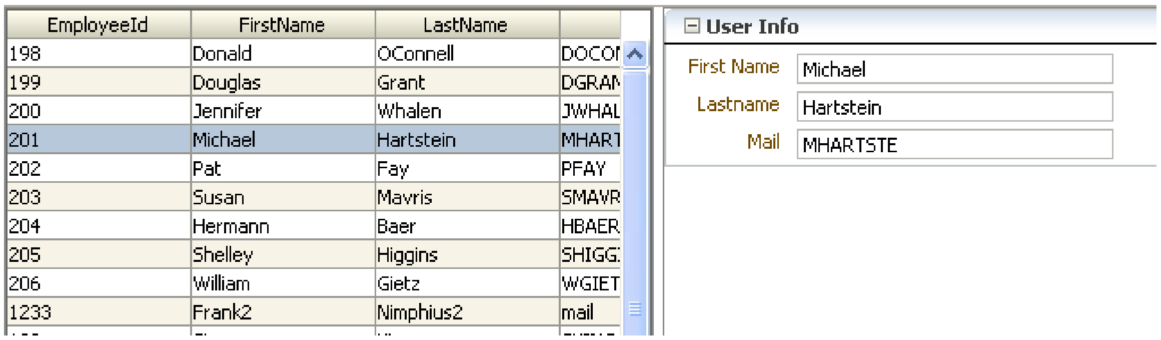
* indicates a naming container

Id: table1

This is it then. At runtime, changing the row selection in the table now will update the declarative component.

Running the Sample

Running the provided example against the HR sample schema shows the following screen.



EmployeeId	FirstName	LastName	
198	Donald	OConnell	DOCOI
199	Douglas	Grant	DGRAN
200	Jennifer	Whalen	JWHAL
201	Michael	Hartstein	MHART
202	Pat	Fay	PFAY
203	Susan	Mavris	SMAVR
204	Hermann	Baer	HBAER
205	Shelley	Higgins	SHIGG
206	William	Gietz	WGIET
1233	Frank2	Nimphius2	mail

User Info

First Name	Michael
Lastname	Hartstein
Mail	MHARTSTE

Download Sample Workspace

The example workspace used in this how-to document is provided on the ADF Code Corner website: http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

Edit the connection property to meet your HR data schema configuration: Download Here

RELATED DOCOMENTATION

☒	Declarative Components – Product Documentation http://download.oracle.com/docs/cd/E15523_01/web.1111/b31973/af_reuse.htm#CACBFGFC
☒	Oracle Fusion Developer Guide – McGraw Hill Oracle Press, Frank Nimphius, Lynn Munsinger http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543
☒	