ADF Code Corner

036. How-to set control hints on POJO entities using the ADF Bean DataControl

Abstract:

A very useful feature that ADF brings to Java EE application development is the ability to specify control hints for enitities in a central place for all UI component to use. A common usecase for this are labels that appear as prompts on a web page, e.g. in a ADF Faces form layout. Users of ADF Business Components know this feature for as long as ADF Business Components is around, but users of EJB, JPA and POJO may not have tried this feature yer.

This how-to document explains how to find an entity for a given UI component and how to set its control property. Its a reverse development as usually you should define labels as soon as you created the data control. However, you will see how this can be done as well.

twitter.com/adfcodecorner

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
25-JUN-2008

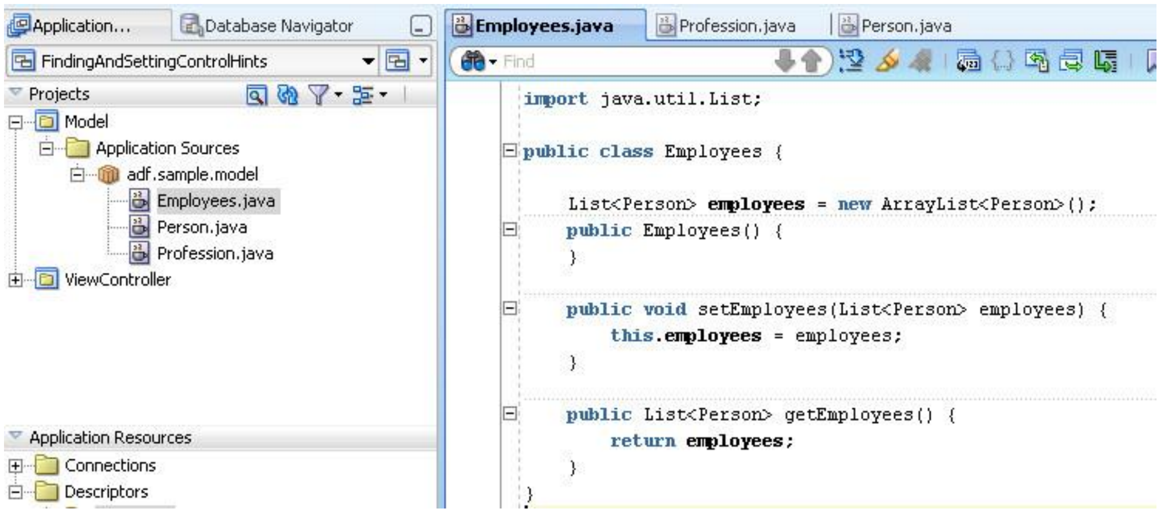
Introduction

Control hints are meta data artifacts in ADF that allow developers to use a central place in their project to define labels, tooltips, as well as formatting rules and validation directives for all usages of an entity attribute.

This not only helps to enforce a consistent naming convention, it also simplifies internationalization of an application because the strings are no longer specific to a JSF page, which means there are less places to search for translatable string.

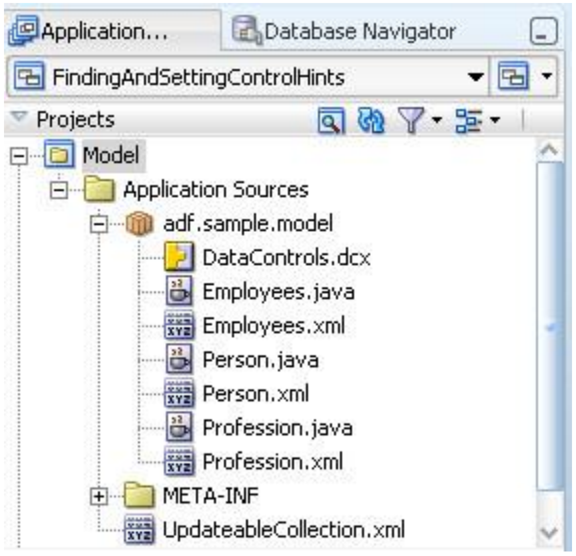
Example

In this section I briefly cover details about the example application. I assume you to be familiar with the concepts of ADF Faces and ADF and will not go into much details about how to use drag and drop to build the web application. The example used for this how-to is a simple POJO Model that shows a list of employees with a history of their past and current professions.

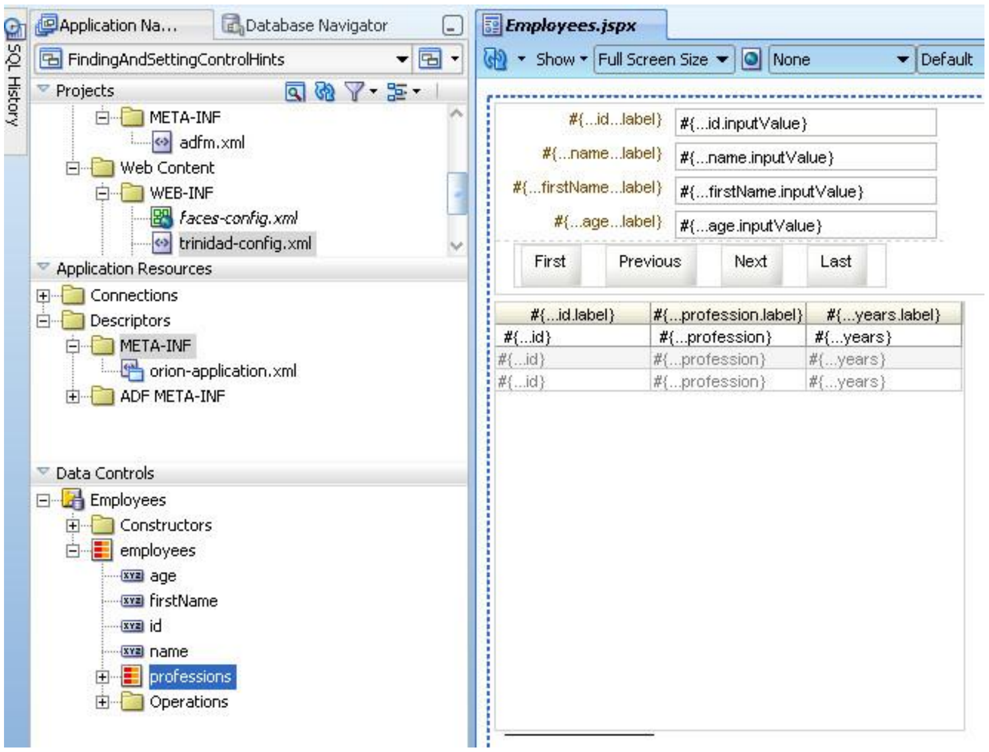


```
import java.util.List;

public class Employees {

    List<Person> employees = new ArrayList<Person>();
    public Employees() {
    }

    public void setEmployees(List<Person> employees) {
        this.employees = employees;
    }

    public List<Person> getEmployees() {
        return employees;
    }
}
```

The ADF Data Control is generated from the Employees class, using the right mouse context menu, which will create the metadata files shown in the image below



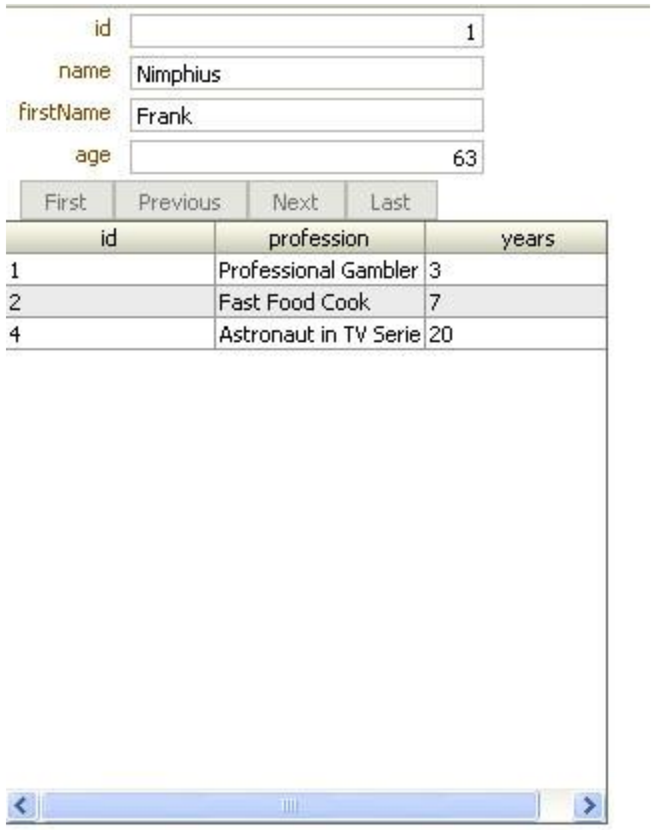
Based on this model, a JSF page is built that shows the employees in a scrollable form with the professional history as detail table. Note the binding reference ".label" in the form prompts and the table headers. ADF attempts to read the attribute label from the generated ADF metadata, the control hints. If no control hints are defined by the developer then the label name is the same as the underlying attribute name.



#{...id...label}	#{...id.inputValue}
#{...name...label}	#{...name.inputValue}
#{...firstName...label}	#{...firstName.inputValue}
#{...age...label}	#{...age.inputValue}

#{...id.label}	#{...profession.label}	#{...years.label}
#{...id}	#{...profession}	#{...years}
#{...id}	#{...profession}	#{...years}
#{...id}	#{...profession}	#{...years}

At runtime, the labels shown in the form and table headers are the names of the attributes. This fine, because the rest of this article will explain how to change this to the better.



id	1
name	Nimphius
firstName	Frank
age	63

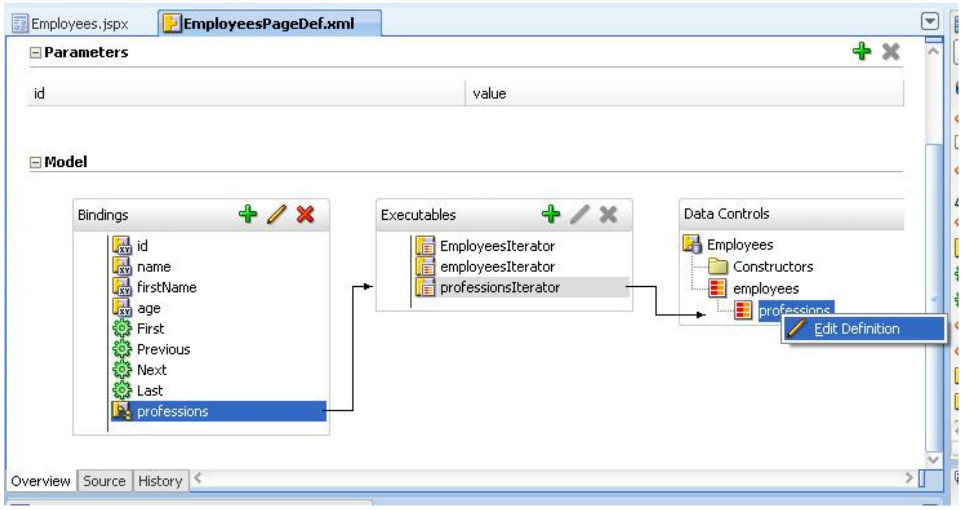
id	profession	years
1	Professional Gambler	3
2	Fast Food Cook	7
4	Astronaut in TV Serie	20

Lets try and change the column header "profession" to "Career", which is much better word to use in this context.

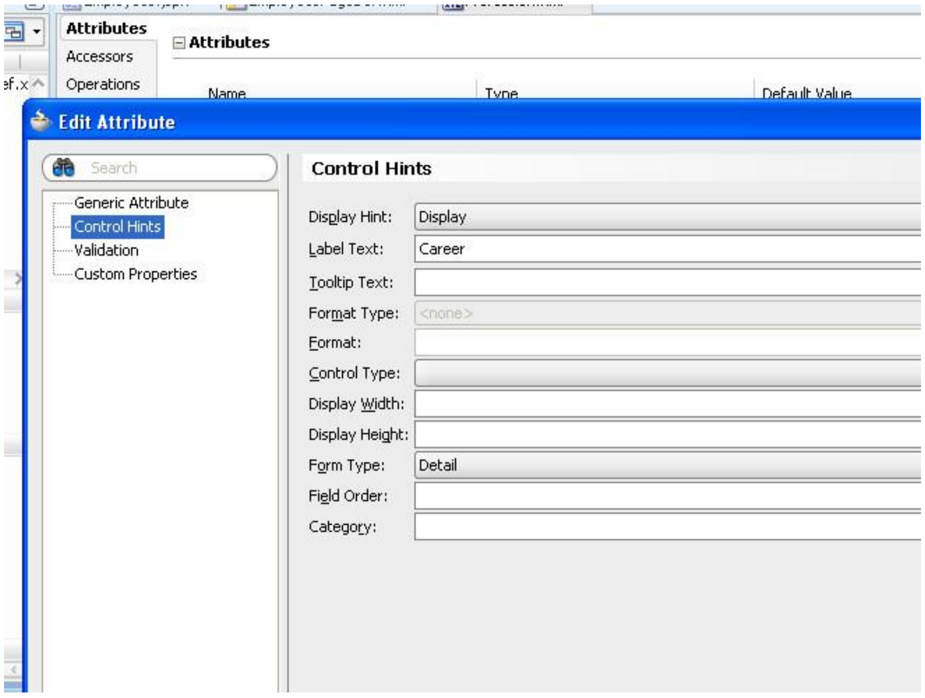
Finding the ADF Metadata of a specific binding

As an ADF developer you get familiar quickly with the concept of bindings. So you know that UI components are bound to named binding artifacts located in a pagedef file, a XML formatted meta data file that is associated with a JSF page. The binding element type may be a table binding, list binding, tree binding, attribute binding or method binding that connects the UI component's value property with the business service attribute of the underlying business model.

The business model in this example is a POJO model. For example, a ExpressionLanguage reference of an ADF Faces input text field #{bindings.name.inutValue} references the "name" attribute binding in the pageDef file of the containing JSF page to read and write data to the inputValue method. In ADF, the #{bindings.name} reference then resolves to an instance of the Person entity within the list of Employees.



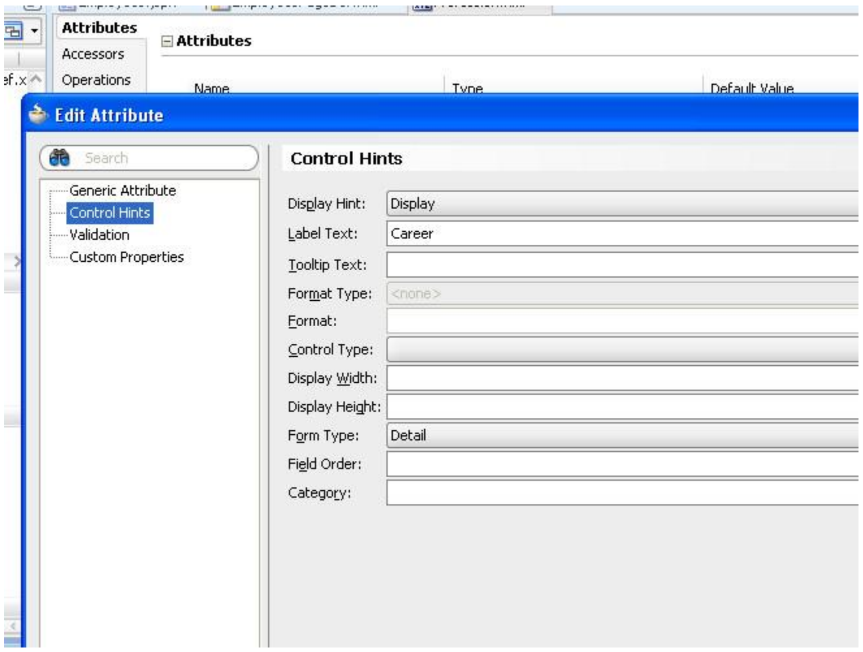
Every ADF bound JavaServer Faces page in JDeveloper 11 has a "Bindings" tab that allows a developer to determine the name of the binding a specific UI component is connected with. In this example, I selected the "profession" column of the detail table in the visual editor and chose "Go to page definition" from the right moue context menu. This opened the associated ADF binding file in the overview editor. The "professions" tree binding used by the table is linked to the "professionsIterator". Clicking onto the "professionsIterator" entry shows the connection to the entity class in the Data Control. Selecting the entity, "professions" in my example, produces the "Edit Definition" context menu on right mouse key usage. Selecting the "Edit Definition" menu option, navigates to the meta data that was generated by ADF for the Profession POJO.



Display Hint:	Display
Label Text:	Career
Tooltip Text:	
Format Type:	<none>
Format:	
Control Type:	
Display Width:	
Display Height:	
Form Type:	Detail
Field Order:	
Category:	

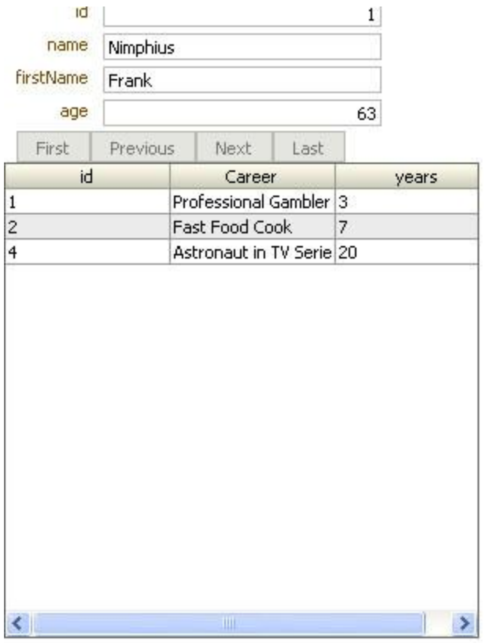
When you select an attribute in this editor, the pencil icon at the upper right corner becomes enabled. Click the pencil icon to launch the editor for the attribute meta data.

The attribute meta data editor allows you to define default values for the attribute, which could be provided as a literal or as an expression, validation like pattern comparison and the control hints. Control hints include properties for the label text, tooltips and format hints that are picked up by the JDeveloper IDE and designtime. For this example the label text is set to "Career".



Display Hint:	Display
Label Text:	Career
Tooltip Text:	
Format Type:	<none>
Format:	
Control Type:	
Display Width:	
Display Height:	
Form Type:	Detail
Field Order:	
Category:	

The string "Career" is added to a project scope properties file so that translated versions of this string can be used easily within an application. Running the same page as before no displays the new label.

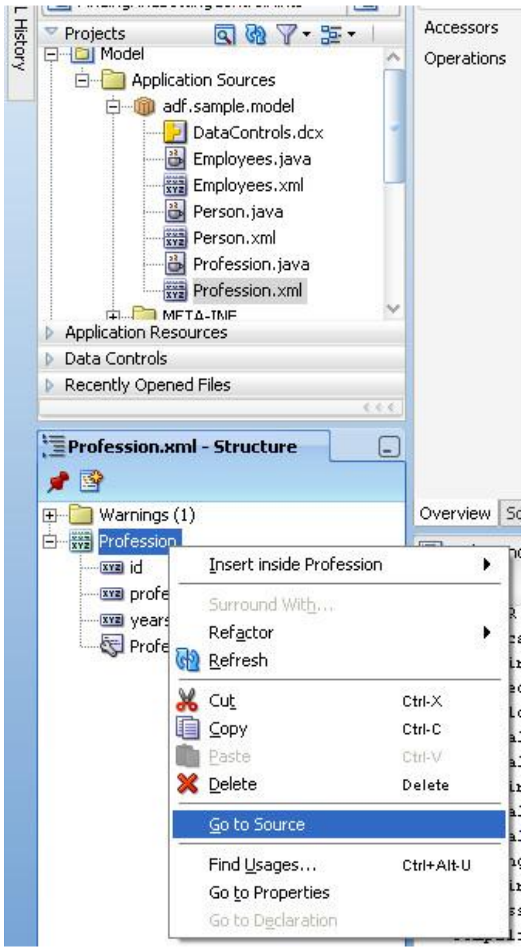


id	1
name	Nimphius
firstName	Frank
age	63

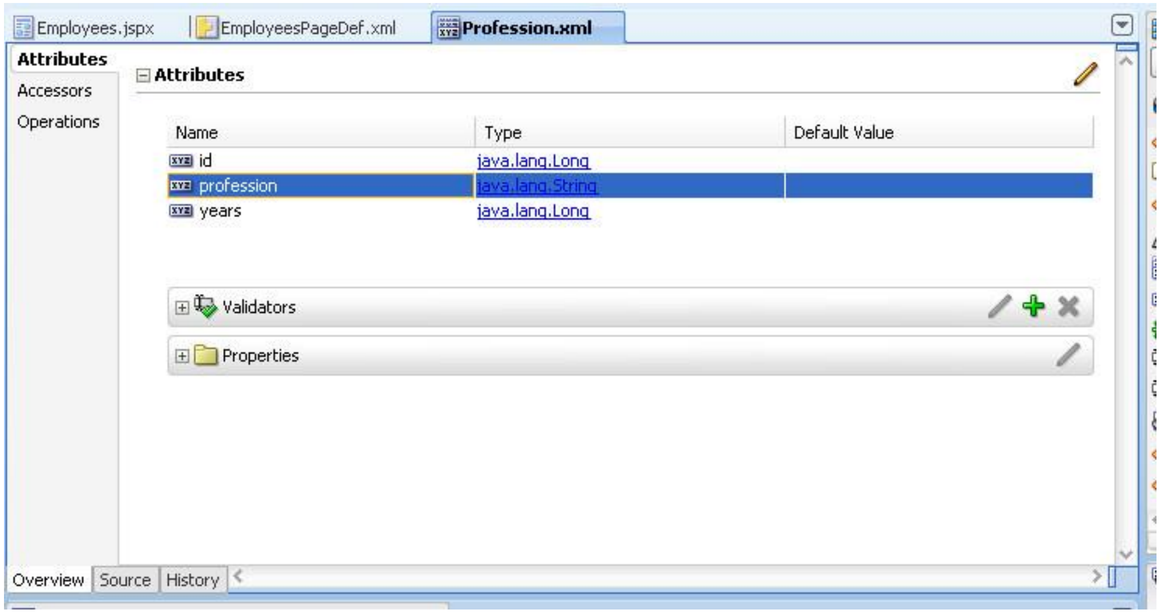
id	Career	years
1	Professional Gambler	3
2	Fast Food Cook	7
4	Astronaut in TV Serie	20

Defining Control hints

As mentioned before, usually you want to define the control hints before you start developing the Web application. In this case, you select the generated XML file of the POJO entity that you want to define labels for and open the Structure Window.



Select the POJO name in the Structure Window and choose "Go to Source" from the context menu. This opens the XML source code view for this file. After clicking the "Overview" tab on the bottom of the source code view, the same editor as above shows



Attributes

Name	Type	Default Value
id	java.lang.Long	
profession	java.lang.String	
years	java.lang.Long	

Download the Example workspace

You can download an Oracle JDeveloper sample workspace from ADF Code Corner:
http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html