

ADF Code Corner

031. Metadata Services (MDS) Example: Power User vs. Normal User

ORACLE
CODE CORNER



twitter.com/adfcodecorner

Abstract:

The Oracle whitepaper "Building Customizable Oracle ADF Business Applications with Oracle MDS" is based on a simple example application that contains the code sources listed in the document. The application base use case is a seeded customization that allows users to switch the user interface between a normal view with dialogs and wizards and an advanced view that does not expose dialogs and wizards but plain input fields. Other features contained in the sample include the use of URL request parameters to apply seeded customization, the use of ADF Security and how to persist drag and drop operations. This blog article assumes you are familiar with MDS, but at least with the whitepaper mentioned above. If not yet, please follow up with the documentation referenced at the end of this blog article. After downloading and configuring the demo in JDeveloper you have a nice little runtime demo to show developers, customers and prospects.

Author:

Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
21-JUN-2010

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: <http://forums.oracle.com/forums/forum.jspa?forumID=83>

Introduction

This MDS demo opens and runs in Oracle JDeveloper 11g. The intention of releasing this sample on ADF Code Corner is to provide sample code that developers can look at, extend and use in their own application development. Though the intention had never been to release this to public, popular demand on the OTN forum helped changing my mind. The following MDS functionality is implemented and demoed in this sample

- Seeded customization to switch between an advanced user interface and the normal user interface
- End user personalization in that the table column display index can be change using drag and drop
- Programmatic personalization in that the user selection whether column re-ordering is possible or not, is persisted
- Programmatic customization that persist changes of the input field order by the user using drag and drop
- Seeded customization that change the user interface for siteCC=emea added to the request URL
- Customization layer inheritance
- ADF Security integration using OPSS Resource Permission

The following JDeveloper projects and files are contained in the sample workspace

- DynamicSiteCC - Project - Applies layout customization based on the value of the siteCC request parameter. The sample has implementation code defined for siteCC=emea and siteCC=reset
- DynamicSiteCCFilter.java - Servlet filter that reads the "siteCC" request parameter and writes it to the session. A MDS session lasts for the duration of request, which requires request parameter settings to be stored in session, so they are available during subsequent application requests (like used when content delivery is set to lazy on an ADF Faces table) as well. In addition, the servlet filter checks the authenticated user permission to customize the application with the request parameter.
- PowerUserCC - Project - Customization layer object that reads a value from a session attribute to determine whether or not the user interface should be modified by the power user customization layer. In this demo, the session parameter is set from the User Preference | Power User menu option
- PowerUserWithCtxDefaultCC - Project that extends the PowerUserCC layer with the ability to define a default value as a context parameter in the ViewLayer project's web.xml configuration file. The context parameter "adf.sample.cc.PowerUser" can be set to true or false. By default the value is assumed as false. Power user settings in the user session always override the context parameter.
- AdfBindingValidator.java - ViewLayer project class used in normal application user mode to suppress ADF binding validation when running the employee edit wizard
- AllEmployeesBacking.java - ViewLayer project managed bean that contains the programmatic MDS changes persistence, like drag and drop input field reordering or the column reorder allowance.
- employees.js - ViewLayer project file that prevents users from entering numeric values in the DepartmentName table filter

Before you start (Sample setup instructions)

Before you can run the sample, you need to complete 6 easy to follow configuration steps.

1. Unzip the zip file you download at the end of this article to your local file system

2. Copy the CustomizationLayerValues.xml file in <your file system>\MDSPowerUserSample\Infrastructure to jdeveloper/jdev directory of your Oracle JDeveloper 11g installation
3. If you have existing entries defined in your copy of CustomizationLayerValues.xml, merge the site and power user content of the file provided in this directory.
4. Copy the JAR files from <your file system>\MDSPowerUserSample\Infrastructure to the \jdeveloper\jdev\lib\patches directory of your JDeveloper 11g installation.
5. Configure the application database connection to point to a HR schema of yours

When you run the allEmployees.jspx file, you authenticate as sking/welcome1 or ahunold/welcome1. The authentication is mainly used when testing siteCC as a request parameter. If you pass siteCC=emea then the input form that is shown for the power user mode shows in 3 columns. Only sking has the permission (ADF Security) to apply this request parameter to the customization session. A hunold is not.

Hint: The sample was tested with Oracle JDeveloper 11g version 11.1.1.3

Demo Overview

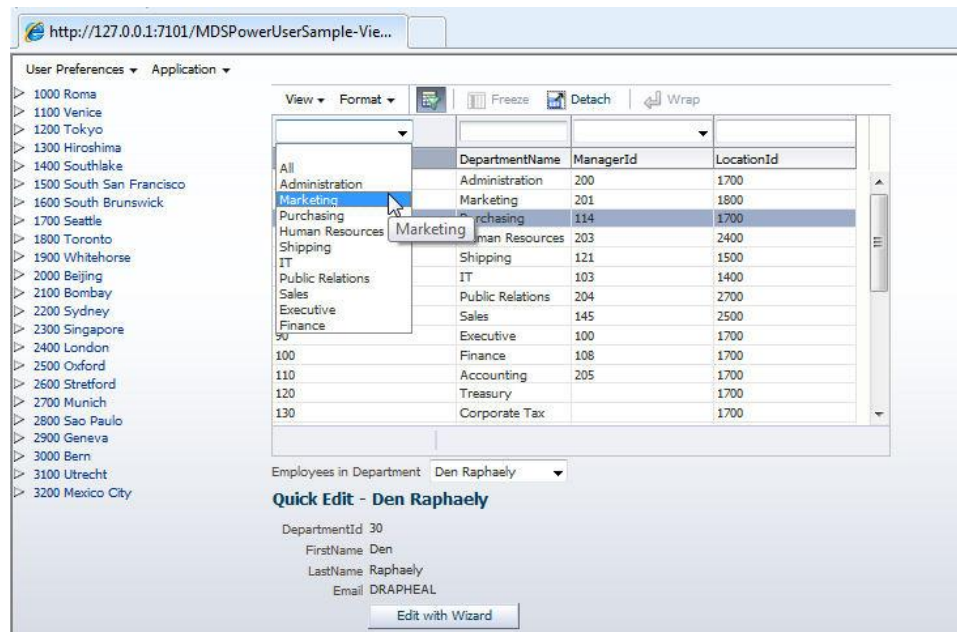
The demo is started from the allEmployees.jspx page and opens in normal user mode (The demo does not persist the user preference for the power user setting. If you wanted to persist this, then a custom user preference table should be used in your business database tier). The start page shows a tree component, a table that lists departments and a form that show dependent employees of the selected department. You can navigate the employee records by choosing the employee name in the select list. Pressing the "Edit with Wizard" opens a wizard to create or edit users in the selected department. As you see in the screen shot below, the normal user model contains a lot of user help and guidance in form of SelectOneChoice components in the table filter fields and a wizard to step users through modifying the selected employee record. The SelectOneChoice components are created in the af:column filter facet. So if you are interested in how this functionality can be implemented in your own applications, have a look at the allEmployees.jspx page.

Hint 1: The "All" option of the SelectOneChoice component in the table filter is not a default functionality. Its worth to study because it provides a way to reset the filter query

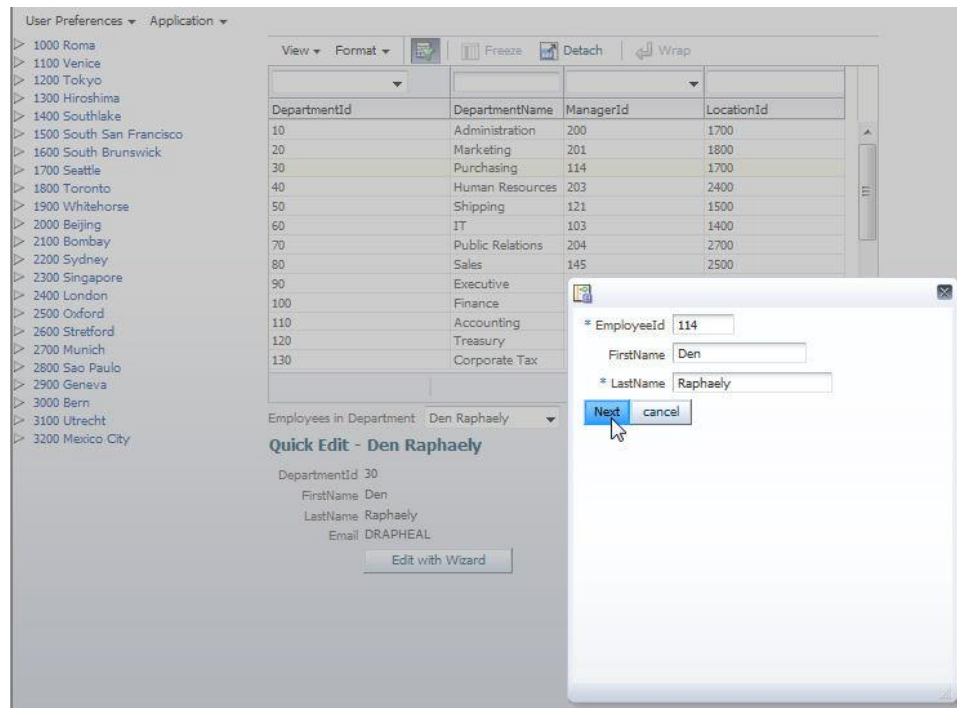
Hint 2: The DepartmentName field filter uses JavaScript to suppress numeric keyboard entries. This little bit of code may be useful for other input components in your application views as well.

Hint 3: Forget the tree component when showing the demo. It does not have any extra functionality (like synchronization with the departments and employees) and is only there to make the UI look better :-)
(now you know how marketing works)

Note: User personalization changes are persisted with the following request. This means that when e.g. you change the order of columns in a table, ensure that a server request occurs afterwards, e.g. in changing the table row currency. In a production application, users would continue working in the application, so that such an extra step is not required.



When you click the "Edit with Wizard" button, a popup opens to guide the user through the steps involved updating an employee or creating a new. Of course this wizard is not impressive at all and could have used train stops for back and forth navigation, but demoing how to build wizards was not the focus of this demo. Worth to mention for the wizard is the use of a custom validation class that is defined in the PageDef file of the wizard views (.jsff files). The AdfBindingValidator helps developers to suppress ADF binding validation until the last page of the wizard is reached. Making ADF validation conditional may be useful in your own application development as well. So please take a look at this.



Now its time to switch Power User mode on. Choose "Power User" from the application menu. The page performs a redirect to itself and, as soon as it comes back, displays without the SelectOneChoice components in the table filter and the "Edit with Wizard" button.

The screenshot shows the application interface in Power User mode. The 'User Preferences' dropdown menu is open, with 'Power User' selected. The main table displays a list of departments with the following data:

DepartmentId	DepartmentName	ManagerId	LocationId
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resources	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relations	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700
110	Accounting	205	1700
120	Treasury		1700
130	Corporate Tax		1700

The 'Employees in Department' dropdown is set to 'Den Raphaely'. Below the table, the 'Quick Edit - Den Raphaely' form is visible, showing fields for DepartmentId (30), FirstName (Den), LastName (Raphaely), and Email (DRAPHEAL), with an 'Edit with Wizard' button.

The Power User UI is the outcome of the normal user base application UI and the MDS metadata customization layer that gets applied at runtime to form the Power User view. If you analyze the application, you find metadata files in the ViewLayer project that contain the diff information that at runtime replaces the select choices with the input text components and that removes the button, replacing it with an editable input form. The MDS diff file is created when starting the JDeveloper IDE in Customization Developer role and changing the allEmployees page for the PowerUser customization layer.

The screenshot shows the application interface in Normal User mode. The 'User Preferences' dropdown menu is open, with 'Normal User' selected. The main table displays a list of departments with the following data:

DepartmentId	DepartmentName	ManagerId	LocationId
10	Administration	200	1700

The 'Employees in Department' dropdown is set to 'Jennifer Whalen'. Below the table, the 'Quick Edit - Jennifer Whalen' form is visible, showing fields for DepartmentId (10), FirstName (Jennifer), LastName (Whalen), Email (JWHALEN), PhoneNumber (515.123.4444), HireDate (9/17/2003), JobId (AD_ASST), ManagerId (101), Salary (4400), and CommissionPct, with a 'Submit' button.

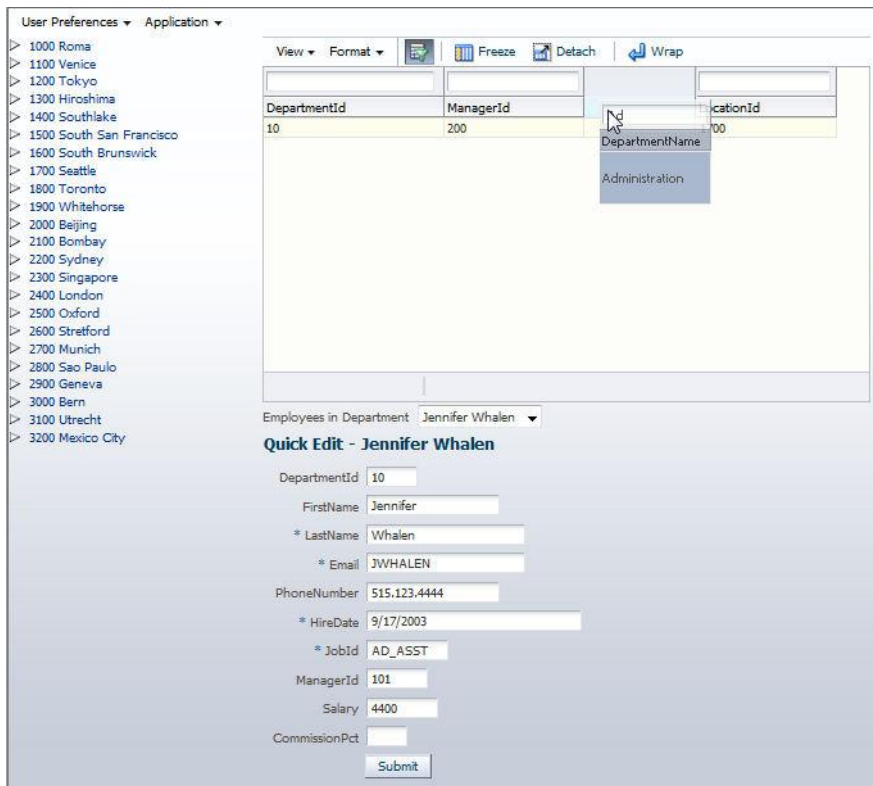
Another example for programmatic user personalization is to persist the ability to change the column order in a table. Though this can be defined as seeded customization in the metadata for a customization layer like the power user layer, this sample shows how to do it dynamically. The usecase is where the user, after changing the display order of the table columns, wants to lock this order so it isn't changed accidentally later. The "Toggle Column Reordering" menu option switches column reordering on and off and persists the current state in MDS. Once you switched off column reordering, you can't move columns with the mouse until you switch it back on.

The screenshot displays the Oracle ADF application interface. On the left, the 'User Preferences' menu is open, showing 'Power User' and 'Normal User' options. The 'Toggle Column Reordering' option is highlighted. Below the menu, a list of departments is visible, including 1600 South Brunswick, 1700 Seattle, 1800 Toronto, 1900 Whitehorse, 2000 Beijing, 2100 Bombay, 2200 Sydney, 2300 Singapore, 2400 London, 2500 Oxford, 2600 Stratford, 2700 Munich, 2800 Sao Paulo, 2900 Geneva, 3000 Bern, 3100 Utrecht, and 3200 Mexico City.

The main area shows a table with columns: DepartmentId, DepartmentName, ManagerId, and LocationId. The table contains one row with values: 10, Administration, 200, and 1700. Above the table, there are buttons for 'View', 'Format', 'Freeze', 'Detach', and 'Wrap'. Below the table, there is a dropdown menu for 'Employees in Department' set to 'Jennifer Whalen'.

Below the dropdown, the 'Quick Edit - Jennifer Whalen' form is visible, containing input fields for: DepartmentId (10), FirstName (Jennifer), * LastName (Whalen), * Email (JWHALEN), PhoneNumber (515.123.4444), * HireDate (9/17/2003), * JobId (AD_ASST), ManagerId (101), Salary (4400), and CommissionPct. A 'Submit' button is located at the bottom of the form.

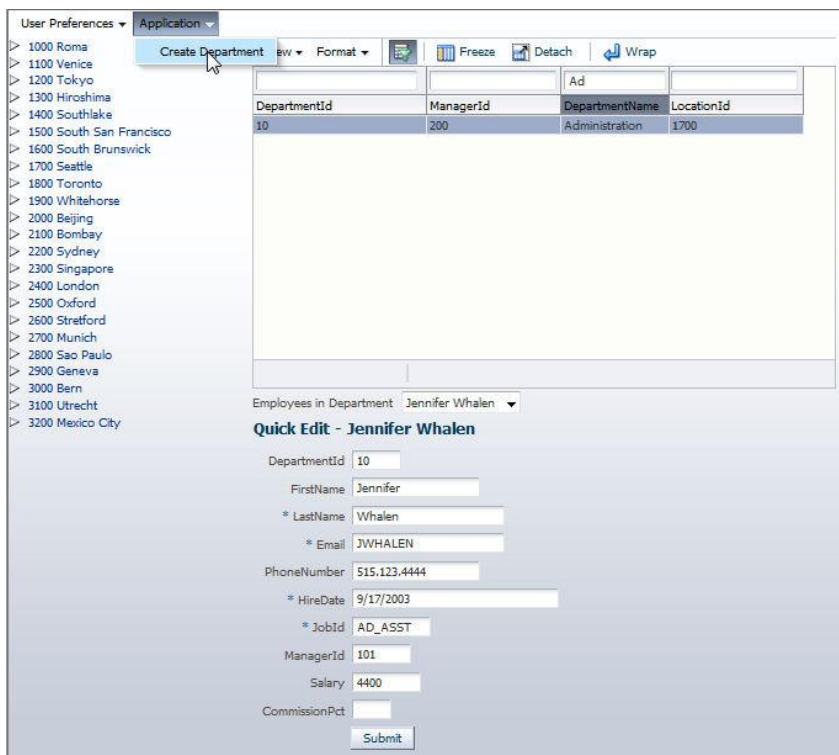
The image below shows what you can do when column reordering is switched on. Selecting a table column with the mouse and moving the mouse with the left mouse button pressed, change the column order as soon as the mouse button is released. This drag and drop functionality is a feature of the ADF Faces table and does not require any additional programming. However, persisting the changed table column order requires MDS (explained in: "Building Customizable Oracle ADF Business Applications with Oracle MDS (Whitepaper)").



The screenshot shows the Oracle ADF application interface. On the left, a tree view lists various departments. The main area displays a table with columns: DepartmentId, ManagerId, Ad, DepartmentName, and LocationId. The first row contains the values: 10, 200, Administration, and 1700. Below the table, there is a dropdown menu for 'Employees in Department' set to 'Jennifer Whalen'. The 'Quick Edit - Jennifer Whalen' form is visible, with fields for DepartmentId (10), FirstName (Jennifer), LastName (Whalen), Email (JWHALEN), PhoneNumber (515.123.4444), HireDate (9/17/2003), JobId (AD_ASST), ManagerId (101), Salary (4400), and CommissionPct. A 'Submit' button is at the bottom of the form.

DepartmentId	ManagerId	Ad	DepartmentName	LocationId
10	200	Administration		1700

To continue with the demo, choose "Create Department" from the "Application" menu.

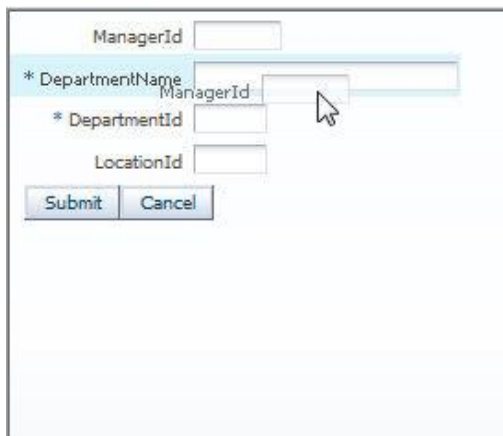


The screenshot shows the Oracle ADF application interface. The 'Application' menu is open, and the 'Create Department' option is highlighted. The table below shows the current state of the department data.

DepartmentId	ManagerId	Ad	DepartmentName	LocationId
10	200	Administration		1700

The createDepartment .jspx page contains an input form that users can reorder the input fields using drag and drop. Unlike column reordering in the ADF Faces table, reordering input fields in a form need to be programmed by the application developer.

In this example, the drop handler is defined in the AllEmployeesBacking.java managed bean. It also contains the code that is required to persist the component change in MDS. The drag handlers, the ADF Faces operation tags are defined in the page source of the createDepartment page. It's not required to provide data in the empty form to see MDS working. Required field validation errors that are shown when moving input components can be ignored. The MDS change is persisted when you press the submit or cancel button of the form.



The last MDS showcase of this demo is the use of request parameters to apply seeded customization. Highlighted by a red border is the siteCC=emea parameter that applies the site level customization for EMEA. The applied customization change is that the input for in power user mode will show in two columns afterwards. To undo this, call siteCC=reset on the request URL. The idea of this usecase is to allow e.g. administrators to simulate a specific deployment environment for testing, without having to set it up. Because everything that is applied from a URL can be tampered with, this functionality is protected with ADF Security. Two users exist in the demo: sking/welcome1 and ahunold/welcome1. Only sking has the privilege to use the siteCC parameter. For ahunold, any attempt to use siteCC is ignored. ADF Security uses the OPSS Resource Permission for this, which is a generic permission class that you need to manually grant in the jazn-data.xml file. This security is explained in the "Building Customizable Oracle ADF Business Applications with Oracle MDS" whitepaper.

The screenshot shows a web browser displaying a page with a table of employees and a 'Quick Edit' form for Jennifer Whalen. The table has columns for DepartmentId, ManagerId, DepartmentName, and LocationId. The 'Quick Edit' form includes fields for DepartmentId, JobId, FirstName, LastName, Email, ManagerId, Salary, CommissionPct, PhoneNumber, and HireDate.

DepartmentId	ManagerId	DepartmentName	LocationId
10	200	Administration	1700
20	201	Marketing	1800
30	114	Purchasing	1700
40	203	Human Resources	2400
50	121	Shipping	1500
60	103	IT	1400
70	204	Public Relations	2700
80	145	Sales	2500
90	100	Executive	1700
100	108	Finance	1700
110	205	Accounting	1700
120		Treasury	1700
130		Corporate Tax	1700

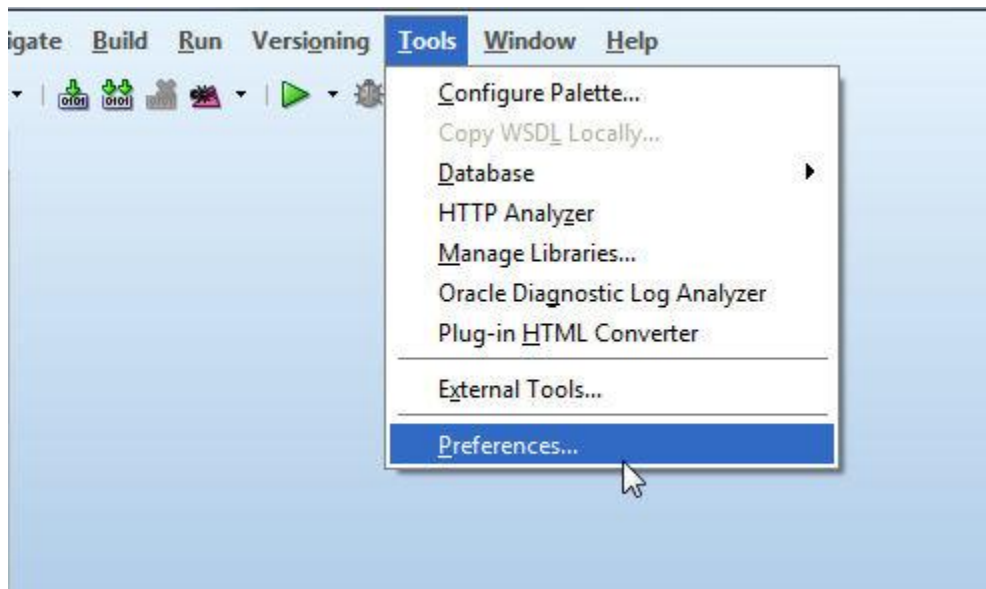
Employees in Department: Jennifer Whalen

Quick Edit - Jennifer Whalen

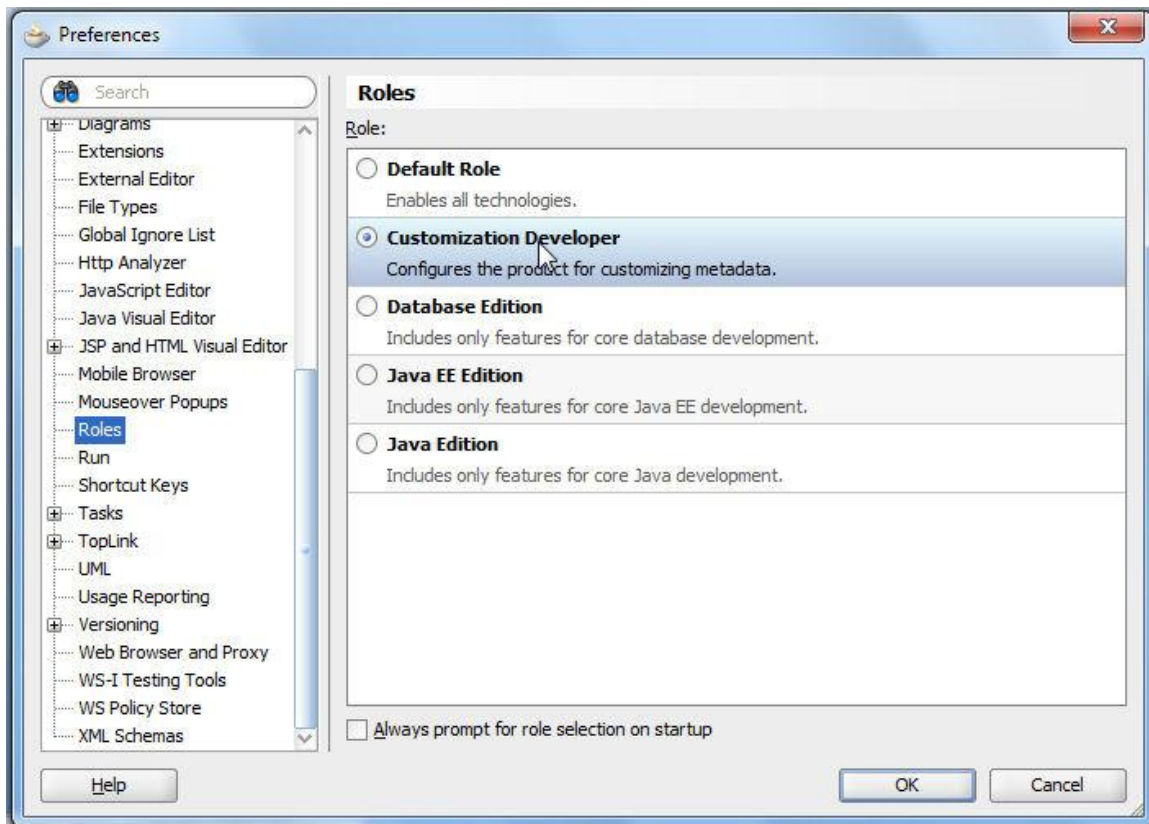
DepartmentId: 10 * JobId: AD_ASST
FirstName: Jennifer ManagerId: 101
* LastName: Whalen Salary: 4400
* Email: JWHALEN CommissionPct:
PhoneNumber: 515.123.4444
* HireDate: 9/17/2003

Submit

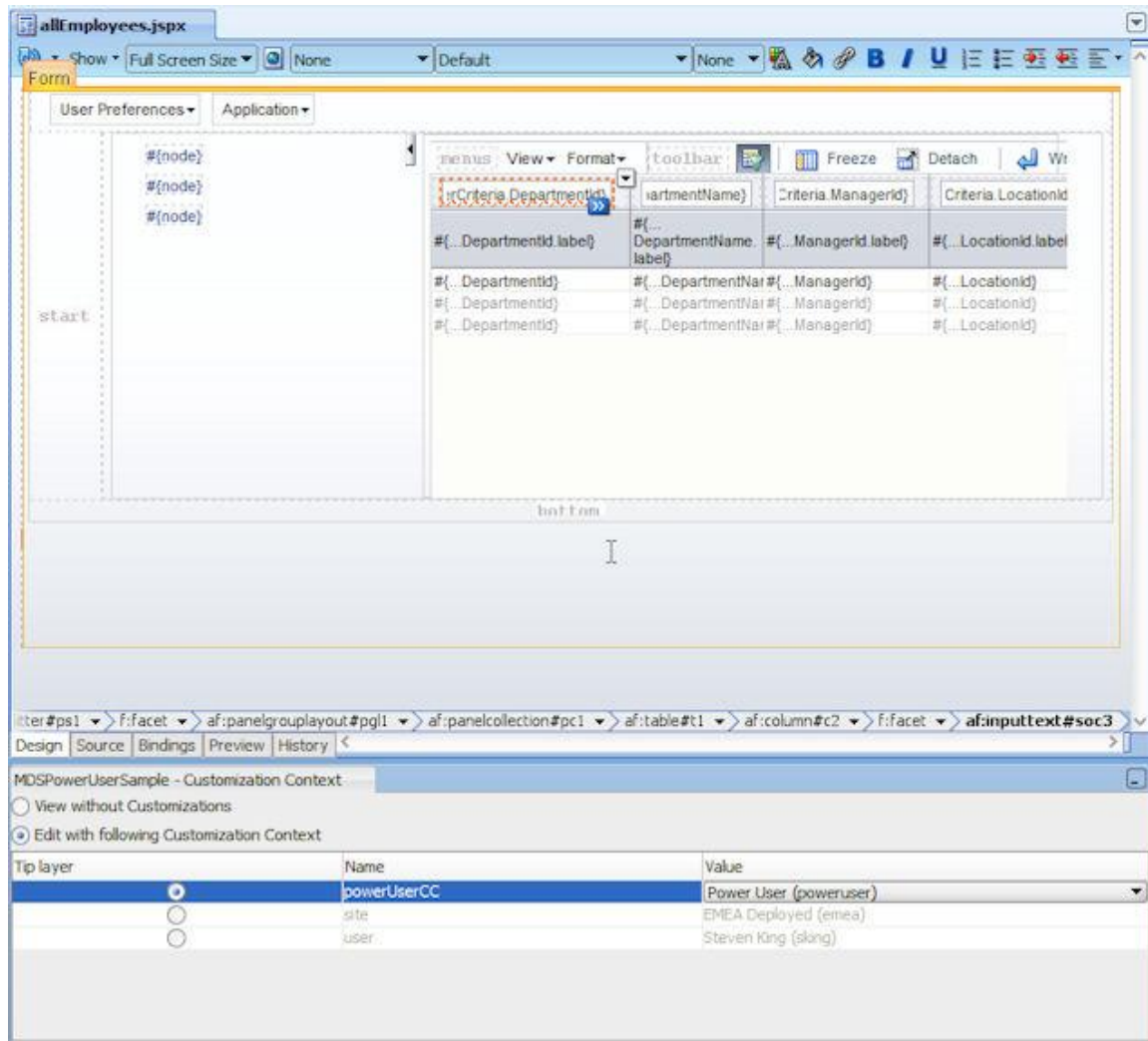
To create seeded customizations, you need to switch the JDeveloper IDE into Customization Developer mode. For this you can use the Tools | Preferences menu



Under "Roles", choose "Customization Developer" and press "Ok". JDeveloper needs to restart to change into the new shaping.



When JDeveloper comes back, make sure the MDS Power User workspace is selected in the Application Navigator. If you configured the sample correctly, then the Power User layer should be shown and enabled in the Customization Context dialog. Select this layer and double click on the `allEmployee.jspx` file to create customizations for the selected layer. JDeveloper always shows you the resulting view that is built from the selected layer and all the layers beneath. All component and attribute changes are protocolled in a metadata file with the extension ".rdf". More details are explained in the "Building Customizable Oracle ADF Business Applications with Oracle MDS" whitepaper.



Download the Sample

The workspace is tested with Oracle JDeveloper 11g version 11.1.1.3 and published as is. You can download the sample zip file from the ADF Code Corner website:

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

Configure the model project to access the HR schema of your Oracle database installation.

RELATED DOCUMENTATION

☒	Metadata Services (MDS) in Fusion Middleware 11g (Whitepaper) http://www.oracle.com/technology/products/webcenter/pdf/metadataservices-fmw_11gr1.pdf
☒	Building Customizable Oracle ADF Business Applications with Oracle MDS (Whitepaper) http://www.oracle.com/technology/products/jdev/11/collateral/adfmds.pdf
☒	Building Customizable Applications Using Oracle Metadata Services (I), by John Stegeman http://www.oracle.com/technology/pub/articles/adf-development-essentials/part8.html
☒	Building Customizable Applications Using Oracle Metadata Services (II), by John Stegeman - http://www.oracle.com/technology/pub/articles/adf-development-essentials/part9.html
☒	Building Customizable Applications Using Oracle Metadata Services (III), by John Stegeman - http://www.oracle.com/technology/pub/articles/adf-development-essentials/part10.html
☒	Application Customization - Cue Card - http://www.oracle.com/technology/products/jdev/11/cuecards/adf_set_18/ccset18_ALL.html
☒	Developer Guide (chapter 34 & 35) - http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/toc.htm
☒	Oracle Fusion Developer Guide – McGraw Hill Oracle Press, Frank Nimphius, Lynn Munsinger http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543