Mastering Personalization and Customization of Oracle ADF applications with Oracle MDS

Frank Nimphius
Oracle Corporation

Schlüsselworte:

Customization, Personalization, MDS, ADF, Java

Introduction

A quote by Charles Darwin [1809-1882] is that "It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change". This statement also holds true in information technology (IT) where developers need to build long living business applications that are able to adapt to organizational changes, differences in end user preferences and changes in the business they support. Developers who create software for commercial or in house use, have the requirement to tailor their application to specific customer needs in a way that survives the installation of maintenance and error correction patches to the application over time. The ability of applications to adapt to changes is not a luxury in Enterprise 2.0. It is a necessity that needs to be considered in the application design and that should the drive selection of the development platform and architecture.

Customization, Personalization, Rebranding

A good investment is one with longevity. This is also true for custom applications you build. Worldly wisdom says that nothing is as certain as change. Therefore, to protect your investment in enterprise application development, the need to react to change should be considered a part of the application design and not as an optional nice-to-have. The ability of an application to change in response to external conditions, like the needs of the individual user working with it, also helps to ensure good usability and leads to higher rates of end-user adoption.

"It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change" - Charles Darwin [1809-1882]

Before metadata customization, software developers used triggers and events built into an application to ensure that at runtime the application user interface (UI) tailors itself to the needs of the authenticated user, his or her responsibilities and organisational role and vertical. The areas in which customization is used include

- Rebranding
- Personalization
- Customization

Rebranding

The Brand is the visual identity of an application that associates it with a company, industry or market. The visual style of an application may change for various reasons, including acquisition, regional differences, modernization or other, for example seasonal, reasons. Look and feel definitions, like

images and cascading style sheets (CSS) that have hard coded references in the application program code make it diffcult to change the look and feel dynamically and therefore should be avoided.

Personalization

Every end user is unique. Even when sharing the same education, skills and job description, application users tend to have different preferences when it comes to how they like the user interface to be rendered. For example, agents that help customers online may have individual preferences of how the customer call history is displayed. Some agents may prefer a summary table to show the problem statements first, while others want to see the date of previous calls to be shown first. Similar, some agents may prefer list of values when querying or inputing data, while others don't. Personalization, or user customization as it is also referred to allows users to make themselves at home within an application. Throughout this paper, we use the terms personalization and user customization interchangeably.

Customization

Application developers define customizations in cumulative layers that are applied at runtime to change a base application to the needs of a specific installation or user group. The customization layer itself is pre-configured and referred to as "seeded". Customization may adapt an application to regional or industry differences, or just change its behavior and UI according to the responsibility of a user or group.

Figure 1 shows a user interface that is customizable for normal users and power users. Compared to normal mode, the power user mode replaces list of values in the table filters with input fields and wizards with a single input form. A similar, very popular requirement expressed by customers is to show more or fewer attributes on an input form based on the configured user roles within an application. This requirement can be achieved easily using MDS customization.

Another aspect of customization is security, which is not an obvious use case. However, using seeded customization to tailor the UI and navigation for the indvidual user helps to suppress the display of sensitive information if a user is not authorized to see it.

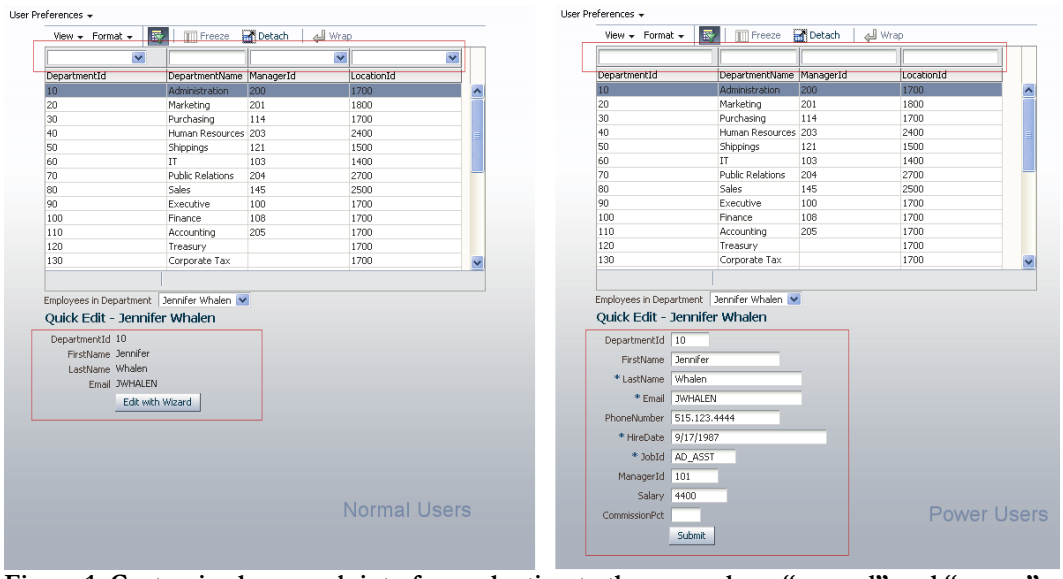

Figure 1: Customized user web interface, adapting to the user role as "normal" and "power" users.

Though all the above can be achieved using conditional statements in Java, such a practice would become unmanageable quickly. Source code, added declaratively or in Java, that does not add to the application core business functionality should not go into an application as it makes maintenance harder and more expensive. To address the requirement for rebranding, personalization and customization efficiently, applications should be designed such that differences between the versions are kept external to the base application.

About Oracle Metadata Services

Oracle Metadata Services (MDS) is the personalization and customization engine within Oracle Fusion Middleware that manages all of this XML metadata of behalf of components such as JDeveloper and ADF.

Metadata is used by the following components

- ADF Faces rich client Java Server Faces components
- ADF Faces Data Visualization Tools (DVT)
- ADF Task Flow
- ADF Binding layer
- ADF Business Components
- Oracle WebCenter

How MDS Manages Customization and Personalization

Customization and personalization are dynamic structure and property changes to the metadata of application documents like views, bindings and task flow definition files. Modifications that, for example, are applied as customization to a page or a page fragment include the addition, removal or property changes of UI components.

An application that is customized with MDS consists of a base application and one or many customization layers that hold the modifications that are applied at runtime. A customization layer is defined by a set of metadata documents that are stored in a metadata store on the file system or the MDS database repository, and a customization layer object, a Java class that determines when to apply the changes. This customization layer class determines the conditions under which a specific customization is applied to the application at runtime. As you will see in this paper, using Java gives you great flexibility and control over how sophisticated customization use cases can be implemented.

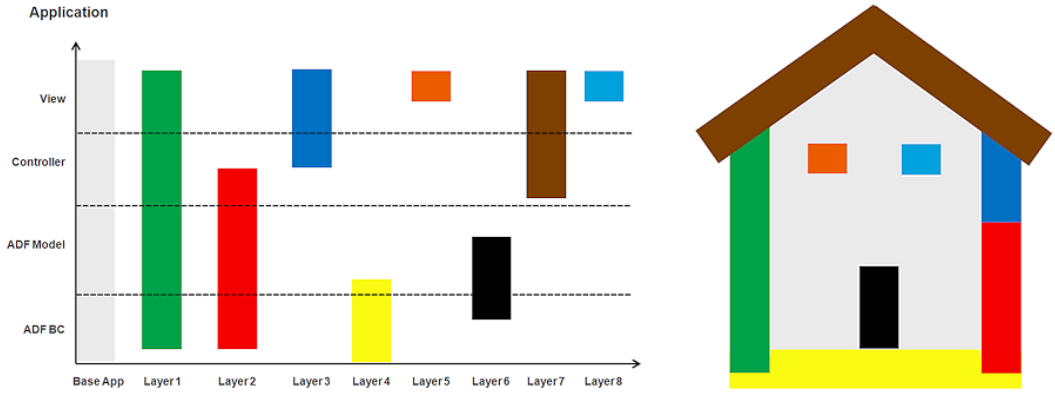

Figure 2: Customization layers are applied to construct the final user view of the application

As shown in figure 2, each document in a customization layer defines changes for a single base application document, like for a page or page fragment. The runtime appearance and behavior of an application then is the outcome of the merged content of a base application with its document change definitions stored in the MDS repository.

Customization in MDS is applied hierarchically in the order customization layers are defined for an application. This way, possible conflicts that may occur when different layers change the same setting of a component, are handled in a predictable way. As you will see later, the order of the layers is not necessarily statically defined but can also be influenced by the customization class object and thus the developer.

MDS stores its customization definitions separately from the base application so that changes applied to the base application, such as a patch, have no impact on customization unless components are removed, in which case no customizations for this component are applied at runtime.

Runtime

The MDS runtime engine applies the document changes defined in the customization layer for the requested application when the view of a page renders. Because customization definitions may be used with multiple instances of an application, customization layers use configurable caching to determine when customization should be dynamically applied and when it can be read from the cache. Customization layers are applied in the order they are configured for an application with the personalization layer added on top. No additional coding is required by the application developer to enable end user personalization of an application, just a configuration of which options should be allowed. Personalizations, such as storing changes of the table column display index, is implemented and managed automatically by the ADF Faces components.

Design Time

To pre-seed customizations into an ADF application, Oracle JDeveloper must be started in the Customization Developer Role. This is an option displayed during the IDE startup, and also accessible from the tools preferences dialog. When the Customization Role in the IDE is enabled, the application opens in such a way that only customizable metadata files are editable. Application developers select the customization layer and layer value for which they want to define changes and start editing the application using the same design time editors and tools that they used for building the base application. Any change that is made, however, does not touch the application definition but rather is stored in a separate metadata file specific to this layer and laver value.

Kontaktadresse:

Frank Nimphius
Oracle Corporation

E-Mail frank.nimphius@oracle.com
Internet: www.oracle.com
www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html