THE POWER OF PERSONALIZATION IN THE E-BUSINESS SUITE

BOB BROWN

DARC CORPORATION

Introduction

Personalization is a feature of the 11.5.10 Oracle E-Business Suite that is the answer to many previously unavailable capabilities ... how to secure individual fields on a form, how to alter the appearance of non-folder enabled forms, present data not available on the forms, make non-required fields required, etc. This presentation will discuss the various capabilities of the Personalization feature in the context of an Order Management implementation.

At Long Last

Since the dawn of time, users of the Oracle E-Business Suite have struggled with how to do the following:

- Remove unused fields, buttons, tabs from the screens
- Re-label fields, buttons, tips to match their terminology
- Change the default value of fields
- Allow easy access from one form to another, passing context
- Do any of the above for only a particular user or responsibility
- Do any of the above only if certain conditions are true.
- Do all of the above without writing code, or "violating" Support

Version 11i of the E-Business Suite introduced the concept of "folders" – a technology that gives the user some degree of control over the appearance of their forms and that addressed some of the issues above. As the applications have matured, some modules have incorporated functionality to address these. However, not until the release of 11.5.10 has there been a suite-wide capability.

Personalization provides a means to achieve the above and more without having to venture into the dreaded "Customization Zone". To be certain, achieving desired functionality will require writing of some SQL. But in general the code can be kept very simple, and integrating it into the Personalization DOES NOT require the use of the Oracle Developer Suite. Personalizations are configured directly in the forms, just as though you were creating transaction types, or entering customers.

A Quick Disclaimer

There are two ways that a user interacts with the Oracle Applications – through the HTML-based forms or through the “back-office” Oracle forms. The Personalization features discussed herein only apply to the back-office forms. HTML-based forms are developed under OA Framework and altering their appearance and behavior is accomplished via other means.

A Personalization Primer

Personalization allows you to alter the appearance and behavior of Oracle forms. Personalizations consist of a set of rules, conditions and actions that are processed by the applications. Among other things, Personalization allows you to:

- **Change an object's properties**. With Personalization, you can exercise access almost unlimited control over what a given user can see, access, what they must enter, value defaulting, etc.

- **Execute Forms built-ins.** These include actions like block and field navigation, forcing commits at rules-driven points in the process, etc.

- **Display messages and additional menu entries.** Informational messages can be displayed to the user as guides to processing transactions. You can alter the “tool tip” text that appears when the cursor hovers over a field. Information can be displayed in the form of message boxes, tool tips, and in the status area at the bottom left of all forms. You can also add new entries on the menu to initiate other functions.

- **Launch processes.** Through form built-ins or other available API’s you can initiate and submit concurrent requests either at a users control, or based on certain conditions that exist in the form.

While Personalization gives you a greater degree of control over the user’s experience with the form, there are certain things that you cannot do:

- **Create new items**. You are limited to operating on fields and objects already known to the form, and are not allowed to create new objects that can be made visible to the user.

- **Move items from one canvas to another**. The location for a given item is fixed to a given “canvas”, or region of the form. While you can use Personalization to move fields around on a given canvas, you cannot move the field to a different canvas.

- **Display an item that is not natively on a given canvas**

- **Display a flexfield on a canvas (darn!)**. Because of the way flexfields are designed and controlled, you are not allowed to display any flexfield as a “native” form field.

A Note on Implementing Personalizations

Implementing Personalizations is truly a system administration function, and should not be given to the general user population. As with any development or configuration effort, best practice would suggest that Personalizations be developed and tested in a non-production environment before moving to production.

The Personalization configuration form is access from the Tools menu via the path Help → Diagnostics → Custom Code → Personalize, and requires the same access as Help → Diagnostics → Examine.

Any queries implemented as part of configuring a given Personalization should be thoroughly tested and tuned prior to production deployment.

Personalization Structure

Personalizations are defined against a given form and function combination and consist of Rules and Conditions, Contexts, and Actions. Rules and Conditions govern "when" the Personalization executes, Contexts help define "who" can execute the personalization, and the Actions define "what" the Personalization will do.

Form versus Function

Each screen that is executed in the Oracle Applications is called a "form". One might think that form and function are equivalent, but that is not the case. The Oracle Applications decouple the notion form from the context in which the form is executed. "Function" refers to the context in which a form is executed.

For example, the Quick Order Entry form in Order Management has the name OEXOETEL. However, this form can be invoked via two means: directly from the Navigator or form the Quick Order Organizer. While in each instance the same form is invoked, each represents a different "function". Therefore, when implementing a Personalization it is important to consider not only the form being personalized, but the context in which the form is invoked, i.e. the function.

Rules and Conditions

Rules consist simply of a sequence number, description, the level (i.e. form or function) of the personalization and an enabled / disabled flag. You are allowed an unlimited number of Personalizations and the sequence numbers can include decimals.

The Enabled flag allows you to turn off an individual Personalization. Alternatively, you can completely disable all Personalizations by executing the menu path Help → Diagnostics → Custom Code → Off. This is useful when the behavior of a form is completely broken due to a Personalization.

The level is where you specify whether a Personalization is to apply to the form, regardless of function, or to a specific function invocation of the form. In this author's experience, all Personalizations have been applied at the form level. It is important to note that the default level is function.

Conditions consist of a Trigger Event, an optional Trigger Object (depending on the event), and an optional additional SQL-based Condition. The triggering event is one of:

- WHEN-NEW-FORM-INSTANCE : when form is first opened
- WHEN-NEW-BLOCK-INSTANCE : when navigating to a block
- WHEN-NEW-RECORD-INSTANCE : when cursor is placed on a record
- WHEN-NEW-ITEM-INSTANCE : when cursor moves to a displayed field
- SPECIALx : used to add entries to the menu

The Trigger Object is dependent on the Trigger Event. For example, if the trigger event is WHEN-NEW-BLOCK-INSTANCE, the Trigger Object is a list of values containing all the blocks on the form. The WHEN-NEW-ITEM-INSTANCE list shows all the individual items (fields, controls, etc.) available on the form.

The SQL-based Condition is an additional field that allows you to put virtually any rule that needs to be enforced. The entry must be a valid SQL statement, and as such should be tested and tuned thoroughly before being placed in production.

Context

The Context allows you to restrict application of the Personalization to any combination of Users and Responsibilities. By default, the Personalization applies at the site level. Using combinations of Context and the Condition allow you to get very specific on when a Personalization applies. For example, to restrict a

Personalization to only users working in a given Operating Unit (say OU 102), you would place the following SQL statement in the Condition field:

fnd_profile.value('org_id') = 102

Any valid SQL statement may be placed in the field, and the condition field may be up to 2000 characters long.

Actions

The Action section of the Personalization is where you define the specific activities that are to occur when the personalization executes. Actions take on one of four types:

- Property
- Message
- Builtin
- Menu

The Property type of action allows you to change the location, appearance, default value, and any number of other features about a specific item. Some of the more frequently altered properties include:

- UPDATE_ALLOWED – defines whether or not the field may be updated. This is the property that would be changed to implement field-level security.
- ENTERABLE – defines whether or not the field may be entered.
- X_POS, Y_POS, WIDTH – define the physical placement of the field on the form along with its size
- VALUE – allows you to change the value of the item
- PROMPT_TEXT – changes the field label
- REQUIRED – makes a field required / not required
- INITIAL_VALUE – can be used as a default value source
- TOOLTIP_TEXT – used to alter the text that is displayed when the cursor hovers over a field.

The Message type of action allows you to display information to the user when certain conditions exist when processing the transaction. Messages can either be displayed in a popup box, or information can be displayed in the status line on the lower left of each form. Message text may be static or dynamic, depending on your requirements.

The Builtin type of action is for controlling screen navigation, for initiating functions and processes, or perhaps for raising a form trigger error. Following is a list of commonly used Builtin types:

- DO_KEY – this will execute a specific action such as a form commit, navigating to the next block, clearing a block, etc.
- GO_ITEM, GO_BLOCK – these will move the cursor to the specified field or block
- RAISE FORM_TRIGGER_FAILURE – this is useful when you wish to add complex validations, say prior to saving a row, and inform the user of failure and prevent saving bad data
- Launch SRS Form – open the standard report submission form
- Launch a Function – open an new form

Each different type of Builtin has different arguments. For example, the Launch SRS Form can take a parameter for the specific report / process you wish to be executed when the Personalization applies. The Launch a Function Builtin allows you to specify the form and function to execute, as well as pass parameters into the function.

The Menu action is where you define additional options available to the user via the menu. Adding this functionality is a two-step process. First, create a Personalization to define the menu entry. Typically the trigger event is WHEN-NEW-FORM-INSTANCE. The action is Menu and you assign a label to one of the SPECIALx options available. The second step is to then create another Personalization whose Trigger Event is the SPECIALx value assigned in the WHEN-NEW-FORM-INSTANCE Personalization.

Examples

Lessons Learned

The Personalization feature is extremely flexible and powerful, but there are specific guidelines that should be followed when using them.

If you are going to personalize a form that is also folder-enabled, it is recommended that you personalize first, and then folderize. The behavior of the Personalizations is much more predictable in this manner. In addition, you may wish to consider the need to lock down the folder capability for users when the forms have been personalized. This is especially true when actions are driven on WHEN-NEW-ITEM-INSTANCE or WHEN-NEW-BLOCK-INSTANCE events. A user changing the sequence of entry via a folder may preclude the proper events firing. In this author's experience, the additional validations and controls afforded by Personalization are worth the trade-off of needing to restrict the folder capability.

Trigger behavior may not always be what you expect. This can be especially true when using WHEN-NEW-ITEM-INSTANCE trigger events. There may be other overriding actions done natively in the form that prevent your Personalization from firing properly. Workarounds are usually available and thorough testing is a must.

From a maintenance standpoint, the Rules defined for a Personalization set should be numbered and grouped according to the sequence in which they occur, and the Trigger Event / Trigger Object combinations. For example, WHEN-NEW-ITEM-INSTANCE Personalizations should be numbered first. All WHEN-VALIDATE-RECORD Trigger Events against a specific block should be numbered in sequence, or using a "." format, e.g. 10.1, 10.2. This is more for maintainability.

Testing is absolutely paramount, especially with Personalizations that alter data and execute form navigation controls. WHEN-VALIDATE-RECORD triggers are typically the best place to enforce rules, but if a WHEN-NEW-ITEM-INSTANCE trigger is called for to execute a specific action be certain of all the different navigation possibilities.

Conclusion

Personalization is an incredibly powerful feature that can go a long way to improving a user's experience with the applications by making the application

- Easier to use
- More friendly with meaningful messages and hints
- More accurate with rule-based value and property setting and rule enforcement
- More efficient with rule-based execution of processes

Metalink Resources