OFBiz Framework Introduction

Framework Diagram Overview

The “Artifact Reference Diagram” is one of the diagrams in the OFBiz Framework Quick Reference Book from Undersun Consulting. It shows how different artifacts in the OFBiz framework refer to each other, starting with a web browser and going through the 3 tiers (user interface, logic and data) down to the database.

Basic Bottom to Top Implementation

When designing based on artifacts like use cases it is good to design all 3 tiers before implementation. For implementation I recommend creating artifacts bottom to top. This helps you keep the structure of the application in mind and allows you to take advantage of the tools in OFBiz to re-use lower level artifacts in the higher level ones. A visual representation of this is available on the first page of the OFBiz Framework Quick Reference Book in the Development Flow diagram.

Step 1: Define Entities

NOTE: Entity and Service Definition, classpath entries, and webapps are referred in the **ofbiz-component.xml** file in each component

In: **entitydef/entitymodel.xml**

Example, ExampleType, ExampleStatus, and ExampleItem: Shows master-detail, and type and status patterns

ExampleFeature, ExampleFeatureAppl, ExampleFeatureApplType: Shows related entities with flexible many-to-many join pattern

Existing OFBiz Entities Used: StatusItem, Uom, Enumeration

Don’t forget the entity-groups: **entitydef/entitygroup.xml**

Step 2: Define Services

In: **servicedef/services.xml**

createExample, updateExample, deleteExample, createExampleStatus, createExampleItem, updateExampleItem, deleteExampleItem

createExampleFeature, updateExampleFeature, deleteExampleFeature, createExampleFeatureAppl, updateExampleFeatureAppl, deleteExampleFeatureAppl

Step 3: Implement Services

In: **script/org/ofbiz/example/example/ExampleServices.xml** & **script/org/ofbiz/example/feature/ExampleFeatureServices.xml**

Step 4: Define Request & View Maps

In: **webapp/example/WEB-INF/controller.xml**

FindExamples, *EditExample* (createExample, updateExample), *EditExampleItems* (createExampleItem, updateExampleItem, deleteExampleItem), *EditExampleFeatureAppls* (example_createExampleFeatureAppl, example_updateExampleFeatureAppl, example_deleteExampleFeatureAppl)

FindExampleFeatures, *EditExampleFeature* (createExampleFeature, updateExampleFeature), *EditExampleFeatureExampleAppls* (feature_createExampleFeatureAppl, feature_updateExampleFeatureAppl, feature_deleteExampleFeatureAppl)

Step 5: Define Screens

In: **widget/ExampleScreens.xml** & **widget/ExampleFeatureScreens.xml**

FindExamples, EditExample, EditExampleItems, EditExampleFeatureAppls; FindExampleFeatures, EditExampleFeature. EditExampleFeatureExampleAppls

Step 5.1: Screen Actions

In: **widget/ExampleScreens.xml** & **widget/ExampleFeatureScreens.xml**

Step 5.2: Screen Widgets (visual Elements)

Screens In: **widget/ExampleScreens.xml** & **widget/ExampleFeatureScreens.xml** Forms In: **widget/ExampleForms.xml** & **widget/ExampleFeatureForms.xml** Menus In: **widget/ExampleMenus.xml** & **widget/ExampleFeatureMenus.xml**

Example Forms: ListExamples, EditExample, ListExampleItems, AddExampleItem, ListExampleFeatureAppls, AddExampleFeatureAppl

Feature Forms: ListExampleFeatures, EditExampleFeature, ListExampleFeaturesAppls, AddExampleFeatureAppl

What are the main things that you need in an enterprise application?

1. Model Business Information Structures
2. Model Business Processes
3. Implement Automated Business Processes
4. Model User Interactions (includes mapping of user input to a Business Process)
5. Presentation of Information to Users
    1. Preparing and Retrieving Information
    2. Layout and Formatting of Information

What is the purpose of a framework? A framework should provide a consistent way of modeling each of these pieces and tying them together.

The OFBiz Framework goes one step further and provides “best practices” tools for each of these needs. These best practices tools follow somewhat of an 80/20 intent in that they should be applicable about 80% of the time and when they do apply should require about 20% of the work that alternatives would require.

In addition to the primary best practices tools the OFBiz Framework also has recommendations for various secondary tools such as Java Methods, FreeMarker Templates, and BeanShell scripts that can be used when more flexibility or control are needed. The framework has touch points that allow the use of other tools in any place, and combined with the best practice and secondary tools.

As a general rule for OFBiz (and a nice principle for life in general) the less code the better and minor compromises in functionality are okay for the sake of less code (easier and cheaper to maintain, customize, etc, etc). Often these compromises in functionality are really a good thing for the sake of consistency, it is easier for both end-users and customizers to understand the system as a whole.