Code Inspection

January 22, 2017

Document Version 0.2

Matteo Frosi (mat. 875393)
Luca Costa (mat. 808109)

Contents

1 Introduction **3**
- 1.1 Purpose . . . 3
- 1.2 Scope . . . 3
- 1.3 Definitions and abbreviations . . . 3

2 Description of the code **3**
- 2.1 Assigned classes . . . 3
- 2.2 Functional roles . . . 4
  - 2.2.1 Accounting package . . . 4
  - 2.2.2 Finaccount subpackage . . . 4
  - 2.2.3 FinAccountProductServices class . . . 7

3 Inspection checklist **8**
- 3.1 Naming conventions . . . 8
- 3.2 Indention . . . 8
- 3.3 Braces . . . 8
- 3.4 File organization . . . 9

4 Bibliography and references **10**

1 Introduction

1.1 Purpose

The purpose of this document is to understand the usage and to find the problems related to a small java class, part of a specific version of Apache OFBiz code.
Code inspection has different goals. First of all, it is needed to improve the quality of the code, finding bugs, missing elements or simply syntactic errors. Moreover, the people that do the inspection improve their skills, augmenting the ability to understand the code. In fact, the inspectors analyze the code made by others, even acquiring new methods in order to accomplish something (maybe in an algorithmic way), while the authors of the code, receiving the report from the inspectors, acknowledge their mistakes, improving their coding skills, too.
Aside from the pure syntactic analysis, that must follow the rules specified in the given document [1], we also have to realize what the given class is used for, and in what is the context in which it is located. To accomplish such goal, in section 2, a brief and high view analysis of the software will be made, including the context, package and class analysis.

1.2 Scope

Apache OFBiz is an open source product for the automation of enterprise processes that includes framework components and business applications for ERP (Enterprise Resource Planning), CRM (Customer Relationship Management), E-Business / E-Commerce, SCM (Supply Chain Management), MRP (Manufacturing Resource Planning), MMS/EAM (Maintenance Management System/Enterprise Asset Management).
Apache OFBiz provides a foundation and starting point for reliable, secure and scalable enterprise solutions.

1.3 Definitions and abbreviations

- Apache: open source company, notorious for its web server
- OMG: is an international, open membership, not-for-profit technology standards consortium. OMG modeling standards enable visual design, execution and maintenance of software and other processes.
- OMG GL (General Ledger): group of standards related to the GLs, that are companies' set of numbered accounts, for their accounting records. The ledgers provide a complete record of financial transactions over the life of the companies.
- K&R style: Indentation style named after Kernighan and Ritchie, who used this style in their book "The C Programming Language".
- Regexp: regular expression, it is a finite automata used to define a search pattern
- Javadoc: is a documentation generator created by Sun Microsystems for the Java language (now owned by Oracle Corporation) for generating API documentation in HTML format from Java source code. The HTML format is used to add the convenience of being able to hyperlink related documents together.
- Sonar: open-source product used to improve code quality via defined metrics.

2 Description of the code

2.1 Assigned classes

There is only one class assigned to our group. The class is:

- FinAccountProductServices.java

This class is located in the org.apache.ofbiz.accounting.finaccount package of Apache OFBiz.

2.2 Functional roles

2.2.1 Accounting package

The class to review is part of the Accounting package of OFBiz. The complete specification of the accounting section is available on the Apache OFBiz Project Overview web page [2], where an overall description for each module is provided. As it is stated in the page:

> "The Accounting entities are organized according to age old and generally accepted principles such as double-entry accounting, a General Ledger with hierarchical accounts, journals and posting of transactions and corresponding entries. The structure is primarily based on the OMG GL standard and the work that was done on an AR/AP extension of the OMG GL standard. This correlates well with other standards such as ebXML and OAGIS. The Accounting entities are structured such that accounts for multiple organizations can be managed. The multiple organizations could be multiple companies, or departments or other organizations within a company. Each Organization can have various GL Accounts associated with it so that it can operate with its own subset of the Master Chart of Accounts. Each Organization can also have its own set of Journals for flexibility, even though the use of Journals should be as minimal as possible in favor of allowing the system to automatically create and post transactions based on business events triggered by standard procedures and documents such as purchase and sales orders, invoices, inventory transfers, payments, receipts, and so forth. There are also entities in place for budgeting and the reconciliation of budgets against actual GL Account balances for a specific fiscal period. Oh yeah, there are also entities used to track custom Fiscal Periods and other entities to keep summary results for accounts in specific periods. Entities to track Fixed Assets are also part of the Accounting entity package. This includes depreciation information in addition to maintenance, scheduling of Fixed Assets (along with the Work Effort entities), and so forth."

Basically, the accounting package deals with all the services such as invoices, payments, taxes, orders, financial accounts, and so on. In particular we deal with the financial accounts sub package.

2.2.2 Finaccount subpackage

The finaccount package, contained in the accounting package of OFBiz, contains three classes (that includes the one we have to inspect), that are:

- FinAccountPaymentServices.java
- FinAccountProductServices.java
- FinAccountServices.java

According to the javadoc of the package [2], the first class deals with the financial accounts used as a payment method, and the second deals with the financial accounts created from product purchases (that are gift certificates). There is no description of the third class (and this is a note of incomplete description), but a quick glance at the code, allows to say that is reserved to furnish utility method to financial accounts. In particular we have all static methods, and they includes:

- creation of a service credit account, that is a type of financial account
- creation of a financial account for a product store
- check the balance of a financial account
- check the status of a financial account

- refund a financial account

So in general these classes concern the usage of a financial account. But what exactly is, w.r.t. the Apache OFBiz software, a financial account? The Apache Wiki [3] states as follows.

A financial account is a tool (similar to bank account statement) that is used for monitoring monetary transactions. Normally they will be linked to a party and the various transactions details (eg payments or receipts) will be shown as entries. The entries for a financial account can be displayed using the 'Financial Account' tab in Accounting or in Party Manager if you enter a party as the owner of the financial account. Currently in OFBiz financial accounts can have the following types:

- Bank Account (by default this type will post to 213500 CUSTOMER DEPOSIT ACCOUNTS)
- Deposit Account (by default this type will post to 213500 CUSTOMER DEPOSIT ACCOUNTS)
- Gift Certificate (by default this type will post to 213200 GIFT CERTIFICATES UNREDEEMED)
- Investment Account (by default this type will post to 213500 CUSTOMER DEPOSIT ACCOUNTS)
- Replenish Account (no default posting account in demo data setup)
- Service Credit Account (no default posting account in demo data setup)

You can also setup each financial account to post to a specific general ledger account for each party. This is done via a specific field during the creation or update of a financial account. This will override the default setting by type.
Financial accounts are used for:

- Managing and Tracking Customer Prepaid Accounts
- Managing and Tracking Customer Credit Limit
- Managing Electronic Gift Certificates / Gift Vouchers/ Gift Card
- Reload of Electronic Gift Card Company

Here follows a screen of the tool that allows the interaction with financial accounts.

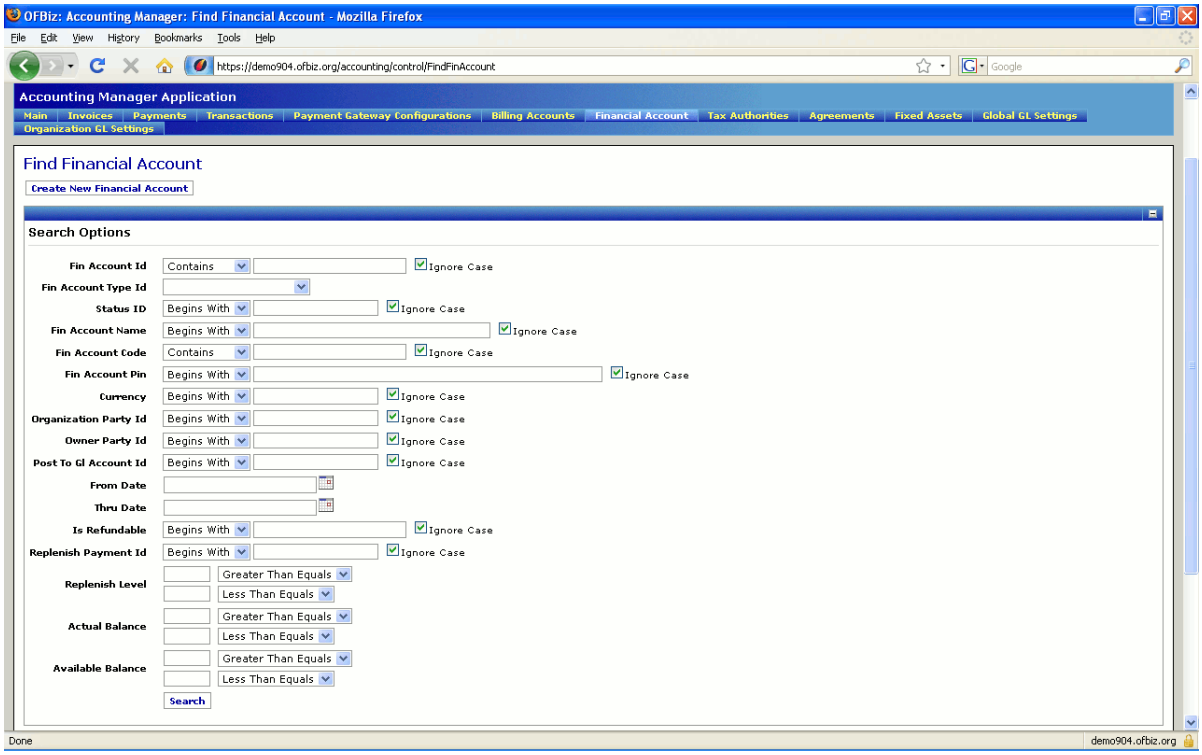

Figure 1: financial account page sample

All the operations related to a financial account are described in the code of the previously named three classes, considering also the imports in each of them. Moreover, from the DataMap [4] of the software we can also realize what entity classes will be involved:

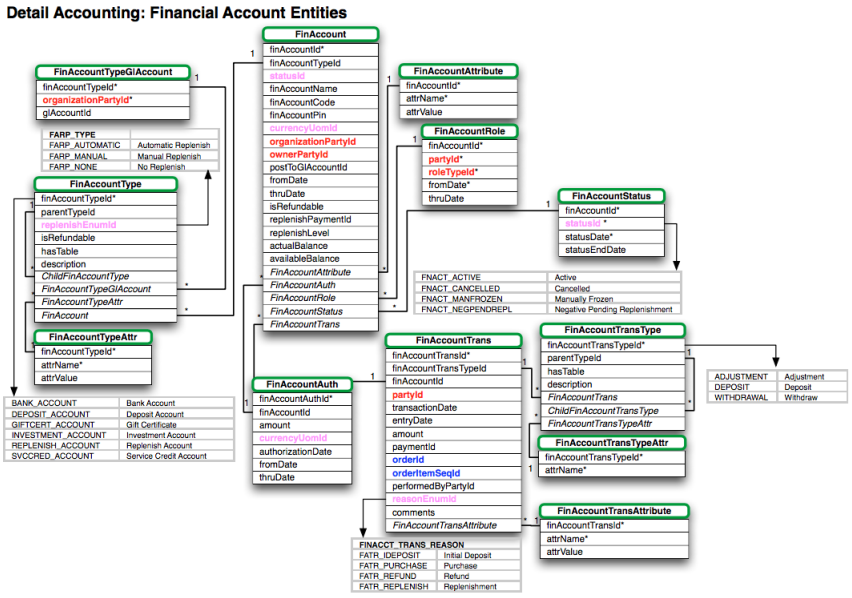

Figure 2: financial account detailed data map

2.2.3 FinAccountProductServices class

3 Inspection checklist

Before proceeding with the inspection of the assigned class, it is good to state the convention used in checking the code.

- L.X indicates the X-th line of code of the class
- L.X-Y indicates the interval of lines of code between X and Y, bounds included

3.1 Naming conventions

- L.81, variable featureAndAppls, of type List<GenericValue>, doesn't suggest its meaning. Moreover the way it is assigned in the next lines, L.81-83, it's even more confusing. The only thing understandable is that it's the result of a query.
- L.79, variable featureAndAppls, of type GenericValue, doesn't suggest its meaning. The only thing that is known is that it is the first element of the featureAndAppls list, filtered by date, L.84 and L.85.
- L.111, variable replenishEnumId, of type String, has an incomplete meaning. Further information can be gathered only at lines L.205 and L.206, that suggests the variable involves the method of replenishing the financial account created.
- L.114, variable orh, of type OrderReadHelper, is understandable only if the type is known and remembered. However it is not used very further in the code (L.117, L.127, L.134, L.137), so that the reader may be aware of the variable meaning.
- L.137, variable billToParty, of type GenericValue, may be misleading. Reading it, it may ring a bell that such variable is the bill that someone, maybe the store who sells the products, has to pay. Looking into further detail in the OrderReadHelper class (org.apache.ofbiz.order/order package), we see that the invoked method getBillToParty invokes the other method getPartyFromRole("BILL_TO_CUSTOMER").
- L.52, static and final variable (constant) module, of type String, is not uppercase, it should be MODULE.
- L.53, static and final variable (constant) resourceOrderError, of type String, is not uppercase and the multiple words forming the name are not separated by an underscore. It should be RESOURCE_ORDER_ERROR.
- L.54, static and final variable (constant) resourceError, of type String, is not uppercase and the multiple words forming the name are not separated by an underscore. It should be RESOURCE_ERROR.

3.2 Indention

The indentation is correct, and always four spaces are used.

3.3 Braces

The K&R (Kernighan and Ritchie) style is adopted, having the first curly brace on the same line as the instruction that opens a new block.
Moreover, every control instruction (if, while, do-while, try-chatch, for) with only one instruction, mostly try-catch, are surrounded by curly braces and not left without them.

3.4 File organization

Blank lines correctly separate the sections of the class, sometimes along with brief comments to describe, sometimes not clearly, the introducing section. From top to bottom we can identify the following sections:

- L.20, package definition
- L.22-26, java imports
- L.28-45, local imports (Apache OFBiz modules)
- L.52-54, constant declaration
- Missing blank line between L.56 and L.57, to separate the name of the method from the comment introducing the firs section of it.
- L.58-62, initialization of the context (utility variables, order item, user login)
- L.65-66, order ID
- L.69-76, order header for store info
- L.78-89, feature of the first product of the order
- L.92-101, financial account data (account type id and account name)
- L.104-111, financial account type (note that L.111 can be moved further in the code, being used only at L.206; such action should also make the variable declared more meaningful).
- L.114, order read helper declaration and initialization
- L.117 and L.120-122, gather and check the currency of the user (the comment refers to "we"). These line should be clustered in one block, due to the fact that they deal with the same thing.
- L.125-134, product store
- L.137-141, party ID (owner of the product)
- L.144-154, payment method info
- Missing blank line between L.154 and L.155
- L.156-174, generation of person/people data
- L.176-182, context for the flexible string expander

4 Bibliography and references

- [2] `https://ofbiz.apache.org/apache-ofbiz-project-overview.html`
- [3]
- https://cwiki.apache.org/confluence/display/OFBIZ/Data+Model+Diagrams