Apache Ofbiz
Code Ispection Document

Niccolo' Raspa, Matteo Marinelli

January 27, 2017

Software Engineering 2 Course Project

Contents

1 Apache OFBiz

The class to be inspected belongs to Apache OFBiz®, 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, etc.

It provides a suite of applications that are useful to integrate and automate most business processes of an enterprise.

1.1 Class Assigned

The assigned class is the following:

- SolrProductSearch.java

The class is located in the package:

- `org.apache.ofbiz.solr`

The path to the class is the following:
apache-ofbiz-16.11.01/specialpurpose/solr/src/main/java/org/apache/ofbiz/solr/SolrProductSearch.java

It belongs to the spcial purpose stack together with other secondary functions suchs as eBay Integration, Google Base Integration, Google Checkout Integration, POS System, Project Management and Scrum Management) and it's devoted to search.

The solr component includes an OFBiz service-based wrapper layer to the Apache Solr webapp queries as well as the native Apache Solr web interface itself.

1.2 Documentation

Apache Solr
http://lucene.apache.org/solr/

Apache OfBiz
https://ofbiz.apache.org/documentation.html

Integration of Solr in Ofbiz
https://cwiki.apache.org/confluence/display/OFBIZ/Search+Integration

Solr Overview
https://cwiki.apache.org/confluence/display/solr/Overview+of+Documents%2C+Fields%2C+and+Schema+D

Java Doc
https://ci.apache.org/projects/ofbiz/site/javadocs/org/ofbiz/solr/SolrProductSearch.html

1.3 Apache Solr

This OFBiz component leverages Apache Solr indexing capabilities. Apache Solr is a fast open-source Java search server. Solr enables you to easily create search engines which searches websites, databases and files. Currently, the solr component focuses on Product data. The functions are logically grouped in two operations: indexing and querying. Indexing operations allow a user to create an index document, which allows to perform query to extrapolate informations. The intended use it to create an index of product so that a user or the administrator can query it to find the products he/she needs.

1.3.1 Data Indexing

The solr component indexes data such as Products into the Apache Solr database using services defined in the file: servicesdef/solrservices.xml The initial indexing may need to be performed or scheduled manually, but subsequent indexing may be partially or fully automated, though automated methods are disabled by default and must be enabled.

There are two methods for indexing data:

`* Index rebuilding service (rebuildSolrIndex):`

The rebuildSolrIndex is the most important data import service. It reindexes all OFBiz Products existing in the system into the solr index. It must be run once after installation and also following any data load operation that loads new products.

Once the initial indexing has been performed, one can then use the Job Scheduler to invoke rebuildSolrIndex on a regular basis (every hour, every midnight, etc.) to update the Solr index.

`* ECAs/SECAs (addToSolr, for Product data):`

Although the rebuildSolrIndex is always necessary for the initial data import, one may also use ECAs (Event Condition Action) and SECAs (Service Event Condition Action) to import subsequent data changes automatically at every individual data (e.g. Product) update instead of running rebuildSolrIndex periodically. This is done by defining ECAs or SECAs that trigger the addToSolr service.

1.3.2 Data Querying

Solr queries can be done using two methods:

*** Solr OFBiz services:**

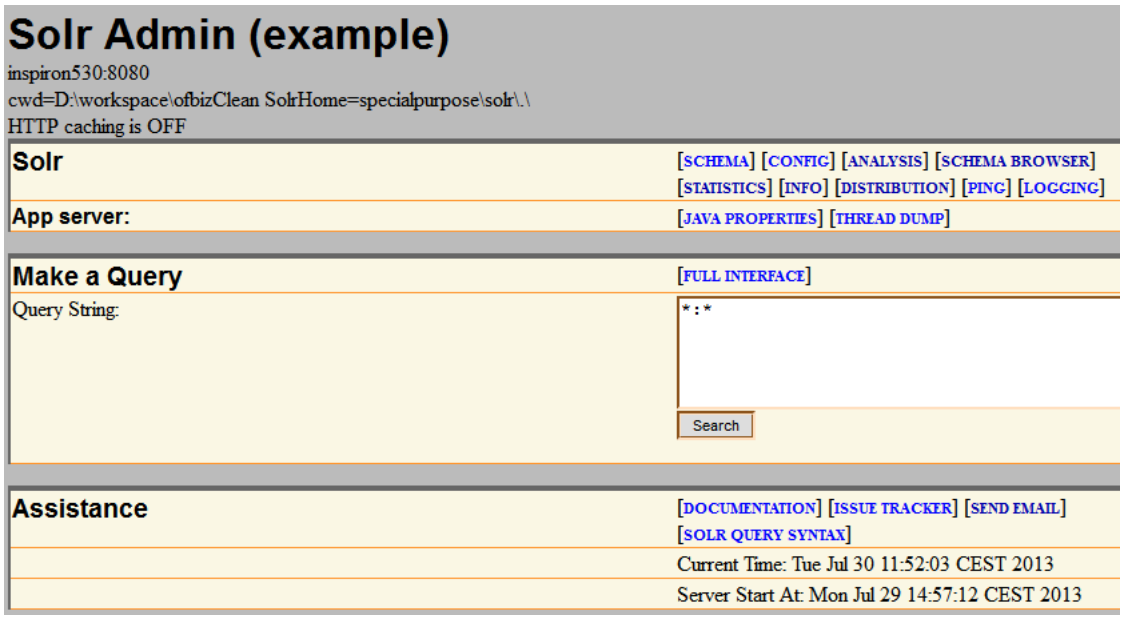
Invoke the functions productsSearch and keyword search.

`* productSearch:` Allows a user to search a product from an ID

`* keywordSearch:` Return a list of products that match a given keywoard.

*** Solr native admin webapp interface:**

One can also perform native Solr queries and diagnostics using the standard admin interface. Please refer to the Apache Solr documentation for usage of this interface.



1.4 Functional role of assigned set of classes

*** Adds product to solr, with product denoted by productId field in instance attribute**

public static Map<String, Object> addToSolr(DispatchContext dctx, Map<String, Object> context)

*** Adds product to solr index**

public static Map<String, Object> addToSolrIndex(DispatchContext dctx, Map<String, Object> context)

*** Adds a List of products to the solr index.**

public static Map<String, Object> addListToSolrIndex(DispatchContext dctx, Map<String, Object> context)

*** Rebuilds the solr index.**

public static Map<String, Object> rebuildSolrIndex(DispatchContext dctx, Map<String, Object> context)

*** Runs a query on the Solr Search Engine and returns the results.**

public static Map<String, Object> runSolrQuery(DispatchContext dctx, Map<String, Object> context)

*** Performs solr products search.**

public static Map<String, Object> productsSearch(DispatchContext dctx, Map<String, Object> context)

*** Performs keyword search.**

public static Map<String, Object> keywordSearch(DispatchContext dctx, Map<String, Object> context)

*** Returns a map of the categories currently available under the root element.**

public static Map<String, Object> getAvailableCategories(DispatchContext dctx, Map<String, Object> context)

*** Return a map of the side deep categories.**

public static Map<String, Object> getSideDeepCategories(DispatchContext dctx, Map<String, Object> context)

2 Code Inspection

The following notations have been used to draw up this document:

- A specific line of code is referred as follows: **L.123**
- An interval of lines of code is referred as follows: **L.123-456**

2.1 Naming Conventions

- Variable name:
  - dctx should be renamed to dispatchContext which is more meaningful (**L.70 L.72 L.73 L.82 L.121 L.196 L.269 L.375 L.378 L.418 L.420 L.498 L.516 L.552 L.560 L.573 L.576 L.628 L.631 L.632 L.655**)
- Variable ambiguity:
  - In method `getAvailableCategories` lines **L.520-528** and in method `getSideDeepCategories`in lines **L.579-589** there is a lot of ambiguity regarding variables since `cat, catL, categories, catList` are simultaneusly used. We propose the following changes:
    - cat to siteCategories
    - catL to categoriesLenght
    - categories resultCategories
    - catList to categoriesList
- Strings rappresenting errors are very long and either should be reduced or managed better using variables.
  - **L.150** "SolrDocumentForProductIdAddedToSolrIndex"

- **L.159** "SolrFailureConnectingToSolrServerToCommitProductId"
- **L222** "SolrAddedDocumentsToSolrIndex"
- **L.231** "SolrFailureConnectingToSolrServerToCommitProductList"
- **L.389** "SolrMissingProductCategoryId"
- **L.679** "SolrClearedSolrIndexAndReindexedDocuments"
- **L.687** "SolrFailureConnectingToSolrServerToRebuildIndex"

2.2 Indentation

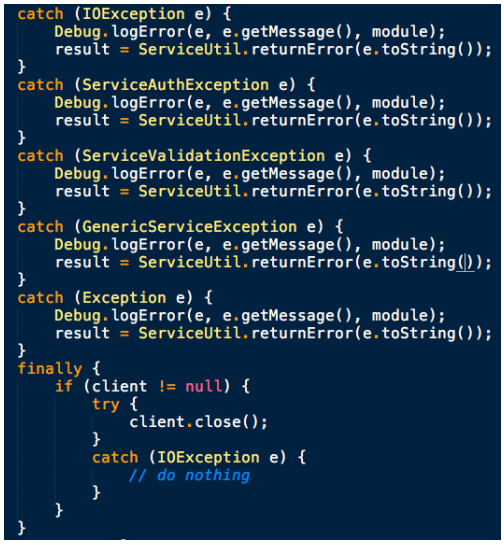
Missing Indentation

- **L.572-612** should be indented
- **L.335** should be indented

2.3 Brackets

Consistent bracing style

There is not consistent bracing style used, it's a mixture of the "Kernighan and Ritchie" style and "one true brace style" (1TBS). We reccomend using the K&R style which makes the code more readable:



The lines where to change the styles are the following:

- catch block

  - **L99**
  - **L.102**
  - **L.105**
  - **L.153**
  - **L.157**
  - **L.175**
  - **L.183**
  - **L.225**
  - **L.229**
  - **L.247**
  - **L.255**
  - **L.357**
  - **L.364**
  - **L.406**
  - **L.488**
  - **L.541**
  - **L.618**
  - **L.681**
  - **L.684**
  - **L.700**
  - **L.703**
  - **L.706**
  - **L.709**
  - **L.712**
  - **L.719**
- finally block
  - **L.179**
  - **l.251**
  - **L.360**
  - **L.715**
- else block
  - **L.604**

All if, while, do-while, try-catch, and for statements that have only one statement to execute are surrounded by curly braces.

- if statement without curly brackets:
  - **L.557**
  - **L.513**
  - **L.503**
  - **L.463**
  - **L.434**
  - **L.432**
  - **L.430**
  - **L.428**
  - **L.424**
  - **L.393**
  - **L.391**
  - **L.389**
  - **L.343**
  - **L.334**
- else statement without curly brackets:
  - **L.345**
  - **L.387**
- for without curly brackets:
  - **L.473**

2.4 File Organization

Line Length

Some lines are over 80 characters but it would be unpratical to break them up but many lines of code are not broken up properly and exceed the indicated caps of 120 characters

- **L.70** could be split before the throws keyword;
- **L.121** could be split before the throws keyword;
- **L.127** comments should be diveded in more rows

- **L.150** could be split before the parameter UtilMisc.toMap(..) and the string "SolrFailureConnectingToSolrServerToCommitProductId" should be reduced to “ConnectionFailedDuringCommit” or using a variable
- **L159** could be split before the parameter UtilMisc.toMap(..) and the string "SolrFailureConnectingToSolrServerToCommitProductId" should be reduced to “ConnectionFailedDuringCommit” or using a variable
- **L196** could be split before the throws keyword;
- **L222** could be split before the parameter UtilMisc.toMap(..)
- **L231** could be split before the parameter UtilMisc.toMap(..) and the string "SolrFailureConnectingToSolrServerToCommitProductList" should be reduced
- **L266** comment should be diveded in more rows
- **L516** could replace the : operator with an if to make the line shorter
- **L518** could be split before the parameter (String) context.get(...)
- **L560** could replace the : operator with an if to make the line shorter
- **L578** could be split after keyword null
- **L628** could be split before the throws keyword;
- **L664** it’s already splitted but not correctly could be splitted after the parameter solrDocs
- **L678** could be split before the parameter UtilMisc.toMap(..) and the string "SolrClearedSolrIndexAndReindexedDocuments" should be reduced or using a variable
- **L686** the string "SolrFailureConnectingToSolrServerToRebuildIndex" should be reduced or using a variable

2.5 Comments

Comments are used to adequately explain what the class

Comments are NOT used to adequately explain what the class, blocks of code and methods do. Comments are usually a single line which briefly explains what a function does, but there is no explanation of the paramaters received in input and the return type.

Commented out code contains a reason for being commented out and a date it can be removed from the source file if determined it is no longer needed.

There are some lines of code which are commented without a proper explanations:

- **L.324-332**
- **L.279**
- **L.341**
- **L.596**

Some lines of code that refers to the verbose mode of the Debus are commented but it would be better to uncomment them and have a VERBOSE_MODE option to select when running the program.

- **L.79**
- **L.134**
- **L.530**
- **L.539**
- **L.578**
- **L.595**

2.6 Java Source Files

The Javadoc is complete but it's very superficial. Every function should be better documented including more details about the input details and the output specification.

2.7 Class and Interface Declarations

Check that the code is free of duplicates

The code for the methods addToSolrIndex and the addListToSolrIndex is quite similar and it could better structured to avoid repetitions. Since the second methods creates a solr document for every element of the list it should be resonable that the second methods calls the first one instead of having repetitions.

2.8 Initialization and Declarations

Declarations appear at the beginning of blocks

Some lines of code are unproperly placed in the middle of a declarative parts.

- **L.84-85**
- **L.132-136**
- **L.276**
- **L.279**
- in function runSolrQuery: **L301,306,334,355,400,401**
- in function keywoardSearch: **L440,441,443,453,461,462,468,469**

2.9 Output Format

Check that error messages are comprehensive and provide guidance as to how to correct the problem.

In the following lines only a line stating the error is printed without any guidance on how to correct the problem:

- **L.157**
- **L.169**
- **L.174**
- **L.179**
- **L.229**
- **L.241**
- **L.246**
- **L.251**

2.10 Computation, Comparisons and Assignments

- **L.288-297** Brutish Programming

2.11 Exceptions

- **L.182** No proper action is performed for the IOException

3 Effort Spent

In this section you will include information about the number of hours each group member has worked towards the fulfillment of this deadline:

- Niccolo' 10 Ore
- Matteo 10 Ore