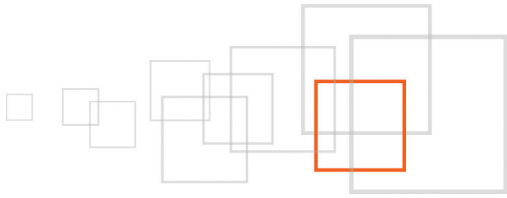
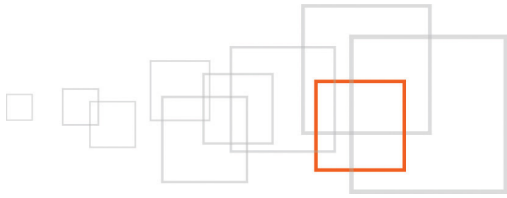
Advanced development with eZ Find

part 1 : Datatypes in Solr and eZ Find

By Gilles Guirand
http://www.gandbox.fr



Index



1 Goal description

Here is the first part of a series of tutorials about eZ Find. At the end of this series, you will have been exposed all details on how it works, and be able to make an advanced usage, in various context, of this enterprise-grade search plug-in. This series will serve as a base for a talk on the subject at the eZ Conference 2010, in Berlin. As well as the Libre Software meeting.

2 Introduction

This tutorial describes how eZ Find transforms and adapts eZ Publish content, and the respective datatypes, to index them in Solr. It by the way presents a few of the 2.2 version's novelties. The understanding of these low-level mechanisms are essential pre-requisites for development and debugging phases, be it only to know where to search for the key information, or be able to read code snippets that help understanding the exact role of a given configuration directive, parameter or filter.

3 Pre-requisites and target population

This tutorial requires to know how to set up eZ Find. The online documentation describes the required operation in details, there : http://ez.no/doc/extensions/ez_find/2_2.

4 Step 1 : how to speed-up eZ Find development tasks : tips & tricks

The following tips will make development on eZ Find more comfortable, and also significantly increase your speed as a developer, and the quality of the output work.

4.1 Re-index some content after a modification

Re-indexing all content to merely test one single, minor modification's impact on one's application can quickly become a drawn-out process. A few, hidden, life-saver arguments exist in the concerned script ***/bin/php/updatesearchindexsolr.php*** , allowing for pointing to :

- a root node
- an offset
- a limit

It is mandatory to use the 3 parameters simultaneously :

```
php extension/ezfind/bin/php/updatesearchindexsolr.php --siteaccess=mysiteaccess
--topNodeID=2546 --offset=0 --limit=10
```

4.2 Check what Solr actually indexed

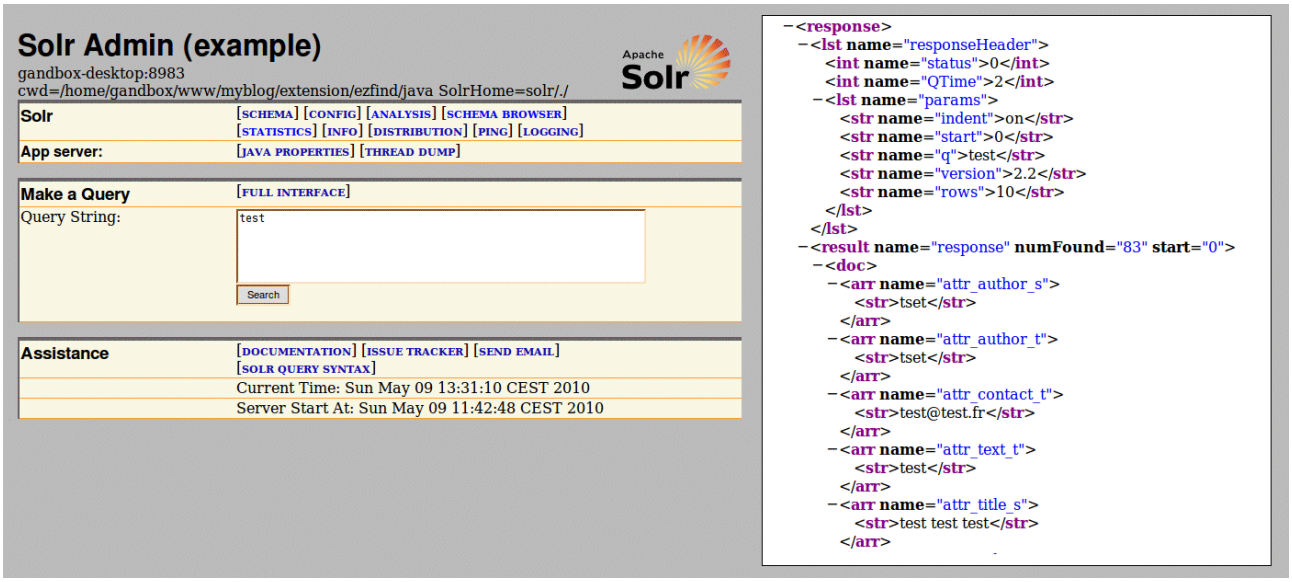


Illustration 1: Solr Web administration

In order to check whether the content and its attributes were properly indexed, simply search for them in Solr's Web administration : **http://localhost:8983/solr/admin/ .** This interface will also help you **figure how the Solr fields were named**. For instance, when using the 2.2 version of eZ Find, one can observe two fi elds looking like duplicates when it comes to articles' titles :

- **attr_title_s**
- **attr_title_t**

The rest of the tutorial explains this behaviour.

4.3 Debugging directly in Solr

By leaving an active console opened, you will be able to view all messages sent to Solr, of the following form :

```
INFO: [] webapp=/solr path=/select params={ ... MESSAGE ... } status=400 QTime=5
```

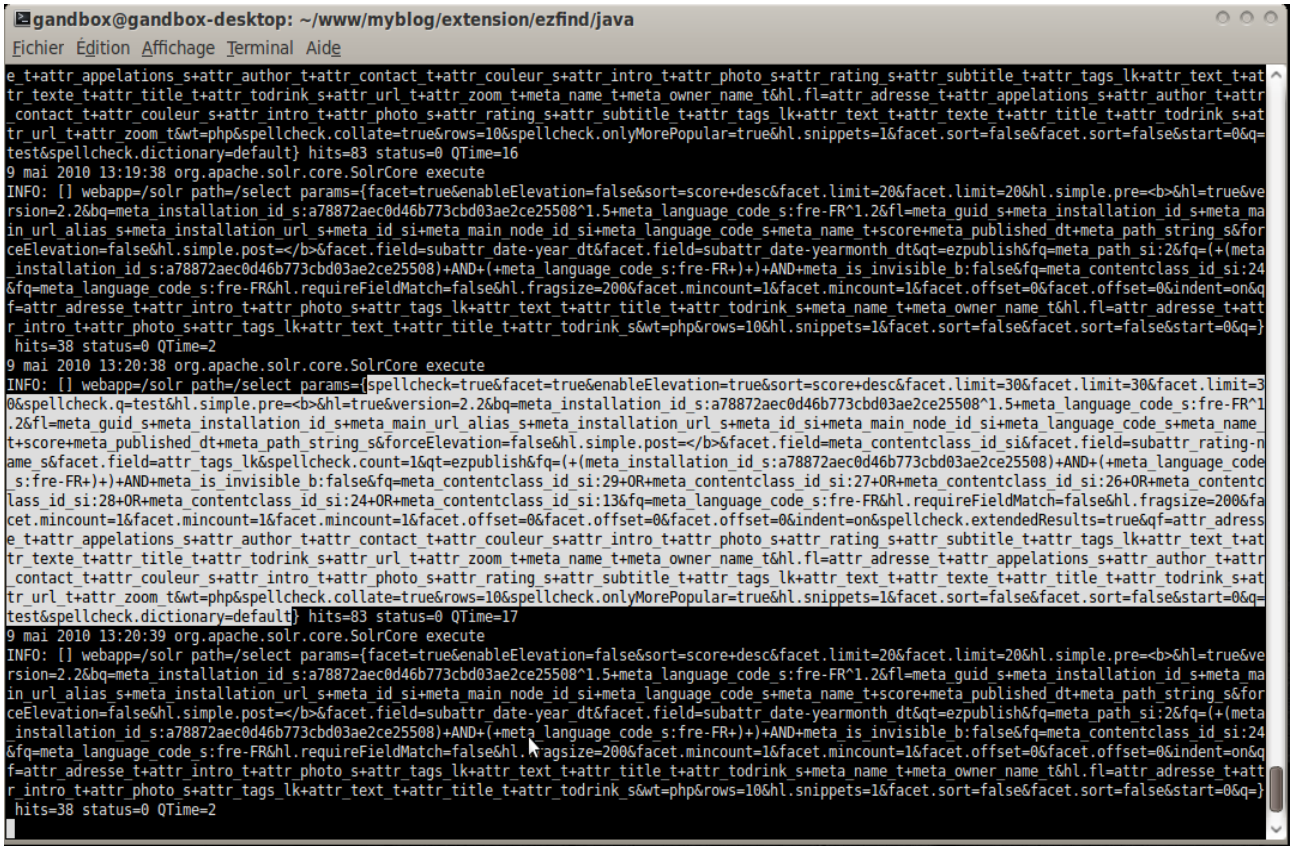


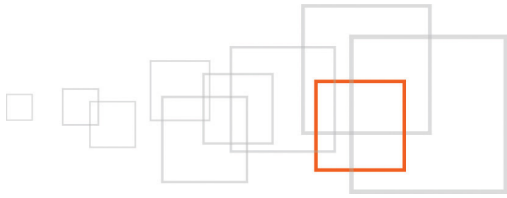
Illustration 2: Solr output in an active console

This message can be copied and pasted from the console to the end of the URL, in Solr's administration interface : **http://localhost:8983/solr/select/?MESSAGE** . The obtained result is **the exact output sent by Solr to eZ Find before transformation** and display of the results. Using this trick is pretty useful when debugging, by, for example, directly manipulating the messages to retrieve the expected result.

5 Step 2 : How eZ Find pushes content from eZ Publish to Solr

Here is the code execution flow when an eZ Publish content object is added/updated:

- Execution of the **registerSearchObject** method, from the eZContentOperationCollection class. If **DelayedIndexing** is enabled (see details below) : insertion of the objectID in the '**ezpending_actions**' table for later indexing.
- If **DelayedIndexing** is disabled, the concerned content object is fetched, and transmitted to the **eZSearch::addObject($object, $needCommit)** method.
- If **eZ Find is installed and enabled**, call of the **eZSolr::addObject($contentObject, $commit = true)** method.
- Finally, once the content has been prepared and modeled, call of the **addDocs** method in the **/extension/ezfind/classes/ezsolrbase.php** class, **POST**ing the data through **HTTP**, using cURL.



5.1 eZ Find, a search plug-in

eZ Find was built-in as an **eZ Publish search plug-in**, system in which the ***/content/search*** view was natively built to be externalized under the form of a search plug-in. Thus, in the eZ Find extension, the /ezfind/settings/site.ini.append.php file declares eZ Find as a search plug-in :

```
[SearchSettings]
SearchEngine=ezsolr
```

The getEngine() method (*kernel/classes/ezsearch.php*) then takes care of retrieving the associated PHP class, to interface the search plug-in, namely /search/plugins/ezsolr/ezsolr.php in our case.

5.2 eZ Find and the native DelayedIndexing mechanism

For every content update operation, the **addObject** method of the **eZSolr** class is invoked. This plug-in-based way of functioning allows to inherit from other native features of eZ Publish's search, like the **DelayedIndexing** one, which gives the possibility to make asynchronous content indexing (globally or per content class). This mechanism relies on a scheduled task (cronjob) : indexcontent.php :

```
[SearchSettings]
DelayedIndexing=disabled|enabled|classbased
DelayedIndexingClassList[]
DelayedIndexingClassList[]=mycontentclass
```

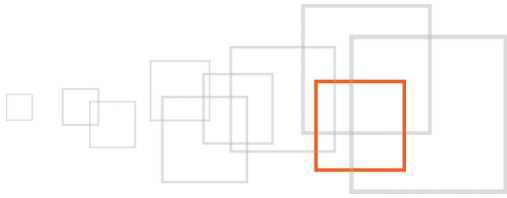
Note :

This technique is particularly efficient when it comes to optimizing the back office response time (and consequently : the user experience), or regular content imports. However, the **DelayedIndexing** technique suffers from a few limitations : it is generic (read : not specifically optimized for eZ Find), and merely loops over the entries of the '**ezpending_actions**' table to index the objects, without using Solr finesses like batch-indexing of content, followed by a single '*commit*'.

6 Step 3 : Understanding the addObject method

The **addObject** (/search/plugins/ezsolr/ezsolr.php) method takes the following parameters :

- One single content object (eZContentObject)
- An optional parameter, indicating whether the freshly added content should be followed by a 'commit' (this is, by the way, a very good way to optimize batch-indexings of content)



In brief, this methods acts as follows :

- Checks whether the content object should be indexed or not, according to the list of excluded content classes (settings : [**IndexExclude**])
- Retrieval of the **objects's metadata** and creation of names and values for the associated Solr fields
- Retrieval of the **node's metadata** and creation of names and values for the associated Solr fields
- Retrieval of the **object attributes' content**, for each language, and creation of names and values for the associated Solr fields
- Push of the collected data to Solr's index, in single or multi-core mode, depending on the settings
- **Final commit**, conditioned by the second, optional parameter of the **addObject** method

Note : since **eZ Find 2.2**, it is possible to use language-specific 'cores' (*/extension/ezfind/java/solr.multicore*), allowing for having per-language indexes and configuration files (spellings.txt, synonmys.txt, stopwords.txt, etc.).

7 Step 4 : Understanding the Solr-side naming of fields

For this, we will have to brush-up **the way eZ Find names the fields transmitted to Solr**. Advanced usages and techniques for eZ Find require an in-depth understanding of the fields semantics and naming mechanism.

The /ezfind/classes/ezfsolrdocumentfieldbase.php class, or potentially inheriting classes when some complex datatypes are in use (see my contribution for instance : ezfsolrdocumentfieldobjectrelation), takes care of creating the Solr fied names, following precise semantics, detailed here :

7.1 Attributes names

Solr name :

attr_[contentattributename]_[contentattributetype], example : **'attr_title_s'.** Note the absence of the content class identifier, opening for nice perspectives like **filtering on several content classes having identical names** (will be covered in a another post).

Mapping in eZ Find's fetch function (in 'filter' for example) :

[contentclassname]/[contentattributename]/[contentsubattributename] , example **'article/title'**. The content class name, **'article'**, is used as an additional filter when building the query. The **'title'** is transformed into **'attr_title_s'**, the **_s** being infered from eZ Find's settings (see below)

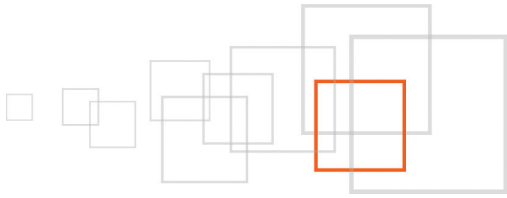
7.2 Metadata names

Solr name :

meta_[metadataname]_[metadatatype], example : **meta_class_identifier_s**

Mapping in eZ Find's fetch function (for sorting for example) :

Internal usage, to sort on 'class_identifier' for instance.



7.3 Sub-attribute names (not used by default)

Solr name :

subattr_[contentattributename]-[contentsubattributename]_[contentsubattributetype] , example : **subattr_relatedimage-alttext_s**

Natively, the subattribute concept is not or little used, because the standard state and features of eZ Publish does not require it massively. It however is here as an opening for advanced usages, and it is, for instance, a great tool to extend eZ Find and index additional fields.

As an example, check my contribution : ezfsolrdocumentfieldobjectrelation, indexing all attributes of an object's related objects, storing them as subattributes. This then opens for applying all sorts of operations to there subattributes (search, filtering, facetting), using the **'myclass/myattribute/mysubattribute'** syntax.

The next posts in this series about eZ Find will explain how to leverage this mechanism, presenting the full portfolio of possible applications.

8 Step 5 : Field type management on the eZ Find side

In Solr field names, the last information signifies the field type, consequently how Solr will process the information (string, text, date, array, etc..). On the eZ Find side, this definition is made the standard way, through settings. **Since eZ Find 2.2, it is possible to define a field type er usage context** :

- **DatatypeMap[]** : global setting, applied to all contexts by default
- **DatatypeMapSort[] :** sort-specific directive
- **DatatypeMapFacet[] :** facet-specific directive
- **DatatypeMapFilter[] :** filter-specific directive

For example, by default in eZ Find 2.2, the “Text line” attributes (ezstring) are assigned a different solr field type for the sorting :

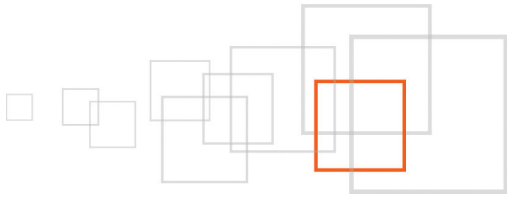
- **DatatypeMap[ezstring]=text**
- **DatatypeMapSort[ezstring]=string**

The consequence on the Solr side is that the “Text line” attributes will systematically be mapped on two different fields :

- **attr_title_t** (used for standard search, facetting and filtering)
- **attr_title_s** (used for sorting)

Note :

It is recommended to also use the '**string**' type for facetting, in order for Solr to consider the whitespace character as a “normal” character, and get fully qualified facets like '**my facet**' instead of '**my**' and '**facet**'.



9 Step 6 : Field type management on the Solr side

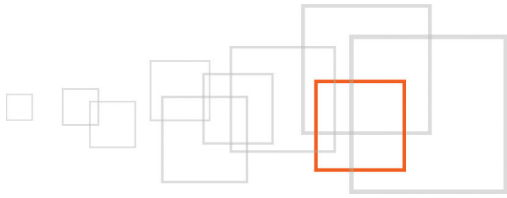
Solr relies on several configuration files, one of them is used to tell him that a **'_s'** at the end of a field name means **string**, **'_t'** for **text**, etc. Here is the file : /ezfind/java/solr/conf/schema.xml.

This configuration file contains the hard-coded definition for a certain amount of fields (metadata fields for instance), but also defines the so-called dynamic fields, each of them associating the dynamic fields names (content attributes ones, coming from eZ Publish) to Solr types :

```
<dynamicField name="*_i" type="int" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_f" type="float" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_d" type="double" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_si" type="sint" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_sf" type="sfloat" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_sd" type="sdouble" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_s" type="string" indexed="true" stored="true" multiValued="true"
termVectors="true"/>

<dynamicField name="*_sl" type="slong" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_l" type="long" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_t" type="text" indexed="true" stored="true" multiValued="true"
termVectors="true"/>

<dynamicField name="*_b" type="boolean" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_dt" type="date" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_random" type="random" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_k" type="keyword" indexed="true" stored="true" multiValued="true"/>
<dynamicField name="*_lk" type="lckeyword" indexed="true" stored="true" multiValued="true"/>
<!-- some trie-coded dynamic fields for faster range queries →
<dynamicField name="*_ti" type="tint" indexed="true" stored="true"/>
<dynamicField name="*_tl" type="tlong" indexed="true" stored="true"/>
<dynamicField name="*_tf" type="tfloat" indexed="true" stored="true"/>
<dynamicField name="*_td" type="tdouble" indexed="true" stored="true"/>
<dynamicField name="*_tdt" type="tdate" indexed="true" stored="true"/>
<!-- geopoint for geospatial/location searches, boosting, ... →
<dynamicField name="*_gpt" type="geopoint" indexed="true" stored="true"/>
```

This files can also be used to define more complex behaviours for given eZ Publish datatypes , like the keywords datatype (ezkeyword). Two different field types definitions can be found, related to the keywords datatype ('**keyword**' for case-sensitive cases, and '**lckeyword**' for lower-case cases), to be used equally at the eZ Find level (**DatatypeMap**) :

```
<fieldtype name="lckeyword" class="solr.TextField" positionIncrementGap="100">
 <analyzer type="index">
 <tokenizer class="solr.PatternTokenizerFactory" pattern=", *" />
 <filter class="solr.TrimFilterFactory" />
 <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
 <filter class="solr.LowerCaseFilterFactory"/>
 <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
 </analyzer>
 <analyzer type="query">
 <tokenizer class="solr.PatternTokenizerFactory" pattern=", *" />
 <filter class="solr.TrimFilterFactory" />
 <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true"
expand="true"/>


 <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"/>
 <filter class="solr.LowerCaseFilterFactory"/>
 <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
 </analyzer>
</fieldtype>
```



This example, keywords fields management, teaches a lot about Solr configuration. One can note the way **Solr filters** are called, how coma-based word separations are handled (**PatternTokenizerFactory**), case-sensitivity management (**LowerCaseFilterFactory**), duplicate removal (**RemoveDuplicatesTokenFilterFactory**), etc.

10 Conclusion

This first part of the series describes fundamental elements in understanding eZ Find, namely :

- How eZ Find interfaces with eZ Publish as a search plug-in
- The two-ways interface between the eZ Publish content object attributes (and datatypes) and Solr field types

These points are essential pre-requisites before envisaging any advanced development on eZ Find. I will continue covering this subject in the two forthcoming parts of this series. Stay tuned !

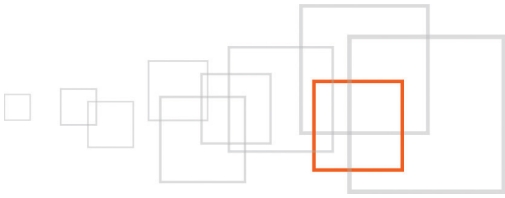
I would like to thank Nicolas Pastorino for translating this tutorial to english, and Paul Borgermans for his availability.

11 Resources

This tutorial is the english version of the originally french one, published on the author's blog :http://www.gandbox.fr/Blogs/Technologies-Web/Developpement-avance-avec-eZ-Find-partie-1-La-gestion-des-datatypes-entre-eZ-Find-Solr

The following resources were referred to during this tutorial :

- eZ Find 2.2 online documentation : http://ez.no/doc/extensions/ez_find/2_2
- eZ Find source code: http://pubsvn.ez.no/websvn2/listing.php?repname=ez_extensions&path=%2Fezfind%2Fezp4%2Fstable%2F2.2%2Fextension%2Fezfind%2F&#ad1cf75aaf3713d19a78efec9843b788a
- Source code for the indexcontent.php script : http://pubsvn.ez.no/websvn2/filedetails.php?repname=nextgen&path=%2Ftrunk%2Fcronjobs%2Findexcontent.php
- eZ Find's Solr schema definition file : http://pubsvn.ez.no/websvn2/filedetails.php?repname=ez_extensions&path=%2Fezfind%2Fezp4%2Fstable%2F2.2%2Fextension%2Fezfind%2Fjava%2Fsolr%2Fconf%2Fschema.xml
- Apache Solr Wiki : http://wiki.apache.org/solr/



12 About the author : Gilles Guirand

Gilles Guirand is a certified eZ Publish Developer. He is widely acknowledged by the community to be one of the national experts on highly technical and complex eZ Publish issues. With over 12 years experience in designing complex web architectures, he has been the driving force behind some of the most ambitious eZ Publish Projects: Web Site Generators, HighAvailability, Widgets, SOA, eZ Find, SSO, Web Accessibility and IT systems Integrations.

13 License