Advanced development with eZ Find

part 2 : Indexing additional fields in Solr

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

Index

1 Goal description

Here is the second 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

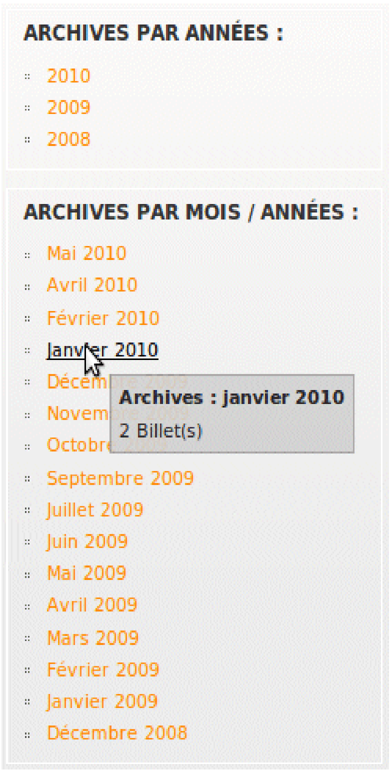
The first part of this tutorial described the low-level mechanisms of eZ Find, and the way eZ Publish attributes (names, types) and Solr fields are mapped. This post describes how eZ Find can dramatically ease development of some functionalities (avoiding complex template operators and heavy SQL queries) by automatically indexing additional fields in Solr when an object is published, which can in turn be leveraged for facetting or advanced filtering.

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.

You should also read and understand the first part of this tutorial : http://share.ez.no/tutorials/ez-publish/advanced-development-with-ez-find-part-1-datatypes-in-solr-and-ez-find

4 Step 1 : Presenting the case


ARCHIVES PAR ANNÉES :

- 2010
- 2009
- 2008

ARCHIVES PAR MOIS / ANNÉES :

- Mai 2010
- Avril 2010
- Février 2010
- Janvier 2010
- Décembre 2009
- Novembre 2009
- Octobre 2009
- Septembre 2009
- Juillet 2009
- Juin 2009
- Mai 2009
- Avril 2009
- Mars 2009
- Février 2009
- Janvier 2009
- Décembre 2008

Archives : janvier 2010
2 Billet(s)

The presented case has been chosen for its high educational value. It is a frequently requested feature, is relatively simple to implement and properly illustrates the underlying concept.

News websites, or blog post lists on blogs often propose to filter the content by year, or by a combination of year and month. Usually in this case, a template operator is developed which builds the appropriate SQL queries. This can quickly become complicated, and often has sharp limitations. Taking a look at the eZArchive operator helps understanding what I mean here.

Frequently, the complex SQL machinery required suffers from functional lacks, like access control propagation, language handling, unability to leverage some subtleties of MySQL or PostGreSql. The developer necessarily has to deal with these problematics, since the API is circumvented to directly write SQL statements. Another limitation of this approach is illustrated by the eZArchive operator : only the 'publication_date' parameter is taken into account, and no room is left for using a content-class-specific date attribute.

Let us see how we can develop a year and year & month filter with eZ Find.

5 Step 2 : Indexing year and year /month in Solr

eZ Find settings (ezfind.ini, to be overridden in the ezfind.ini.append.php file of your extension) allow for delegating the indexing process of an eZ Publish datatype to a given PHP class :

```
[SolrFieldMapSettings]
CustomMap[ezdate]=ezfSolrDocumentFieldDate
```

Our PHP class, randomly named , **ezfSolrDocumentFieldDate** inherits from the **ezfSolrDocumentFieldBase** class, must be added to the **/extension/myextension/classes** folder and must inherit the following skeleton :

```
<?php
class ezfSolrDocumentFieldDate extends ezfSolrDocumentFieldBase
{
    public static function getFieldName( eZContentClassAttribute $classAttribute, $subAttribute = null, $context
= 'search' )
    {
      // return the fieldname like : attr_mydate_d
    }

    public function getData()
    {
      // return the array keys (fieldname => value), like : array('attr_mydate_dt' => '2010-04-30T00:00:00Z')
    }
}
?>
```

6 Step 3 : Understanding the role of the getFieldName() method

This method is invoked the attributes names (within **eZ Find**) to **Solr** field names. For instance, when building a facet using the following syntax : '**mycontentclass/mydateattribute**', this method receives '**mydateattribute**' and should return '**attr_mydateattribute_dt**'. Here is how we are going to implement the body of this method :

- If a subattribute is specified (for instance : '**mycontentclass/mydateattribute/year**'), then return the composed name of the subattribute.
- If no subattribute is specified, then execute the associated parent class code
- **Important** : to make sure the written code is generic enough, and avoid hard-coding the Solr fi eld names, we will use the handy **generateSubattributeFieldName** and **generateAttributeFieldName** methods.

```
const DEFAULT_SUBATTRIBUTE_TYPE = 'date';

public static function getFieldName( eZContentClassAttribute $classAttribute,
$subAttribute = null, $context = 'search' )
{
  switch ( $classAttribute->attribute( 'data_type_string' ) )
  {
    case 'ezdate' :
    {
     if ( $subAttribute and $subAttribute !== '' )
     {
      // A subattribute was passed
      return parent::generateSubattributeFieldName( $classAttribute,
       $subAttribute,
       self::DEFAULT_SUBATTRIBUTE_TYPE );
     }
     else
     {
      // return the default field name here.
      return parent::generateAttributeFieldName( $classAttribute,
self::getClassAttributeType( $classAttribute, null, $context ) );
     }
  } break;

    default:
    {} break;
  }
}
```

7 Step 4 : Understanding the role of the getData() method

This method is invoked to extract data from eZ Publish, and prepare it prior to indexing in Solr. This method is the place to add additional fields like '**year**' et '**yearmonth**'. Once these new fields (subattributes) are indexed, we will leverage them by facetting or filtering on them, using the following, classical syntax:

- '**mycontentclass/mydateattribute/year**', translated in Solr under : '**subattr_date-year_dt**'
- '**mycontentclass/mydateattribute/yearmonth**', translated in Solr under : '**subattr_date-yearmonth_dt**'

```
public function getData()
{
  $contentClassAttribute = $this->ContentObjectAttribute->attribute( 'contentclass_attribute' );

  switch ( $contentClassAttribute->attribute( 'data_type_string' ) )
  {
    case 'ezdate' :
    {
      $returnArray = array();

      // Get timestamp attribute value
      $value = $this->ContentObjectAttribute->metaData();

      // Generate the main filedName attr_XXX_dt
      $fieldName = parent::generateAttributeFieldName( $contentClassAttribute,
      self::DEFAULT_ATTRIBUTE_TYPE );
      $returnArray[$fieldName] = parent::convertTimestampToDate( $value );

      // Generate the yearmonth subattribute filedName subattr_year_dt
      $fieldName = parent::generateSubattributeFieldName( $contentClassAttribute,
      'year',
      self::DEFAULT_SUBATTRIBUTE_TYPE );

      $year = date("Y", $value); // Get Year value : 2010
      $returnArray[$fieldName] = parent::convertTimestampToDate( strtotime($year.'-01-01') );

      // Generate the yearmonth subattribute filedName subattr_yearmonth_dt
      $fieldName = parent::generateSubattributeFieldName( $contentClassAttribute,
      'yearmonth',
      self::DEFAULT_SUBATTRIBUTE_TYPE );

       $month = date("n", $value); // Get Month value : 3
      $returnArray[$fieldName] = parent::convertTimestampToDate( strtotime($year.'-'.$month.'-01')
);

      return $returnArray;

  } break;

  default:
  {} break;
  }
 }
}
```

Note:
$returnArray contains an associative array looking like this (using var_dump) :

```
array(3) {
 ["attr_date_dt"]=>
 string(24) "2008-12-28T00:00:00.000Z"
 ["subattr_date-year_dt"]=>
 string(24) "2008-01-01T00:00:00.000Z"
 ["subattr_date-yearmonth_dt"]=>
 string(24) "2008-12-01T00:00:00.000Z"
}
```

Note :
Solr uses the ISO 8601 date format, of the form : '2010-04-30T00:00:00Z'. The parent class (**ezfSolrDocumentFieldBase**) exposes the **convertTimestampToDate()** methodm used to convert a timestamp to an ISO 8601 formatted date.

8 Step 5 : Building the facet navigation through a template

Our per-year and per year/month data are now available, indexed in Solr. Left is only to build the facet navigation, the usual way :

```
{def $search_yearmonth=fetch( ezfind, search,
           hash( 'query' , '',
                 'facet', array(
                       hash('field', 'billet/date/year',
                            'sort', 'alpha',
                            'limit', 20 ),
                       hash('field', 'billet/date/yearmonth',
                            'sort', 'alpha',
                            'limit', 20 )
                                ),
                 'class_id', array('billet'),
                 'subtree_array', array(2)
 ))}

 {def $search_extras_year=$search_yearmonth['SearchExtras'].facet_fields[0].nameList|
reverse}
 {def
$search_extras_yearmonth=$search_yearmonth['SearchExtras'].facet_fields[1].nameList|
reverse}
 {def $date_count = 0
      $date_ts = 0}

<li id="blog_block_10" class="colonne_block">
 <h1>Archives par années :</h1>
 <ul class="{$current_css} list">
 {foreach $search_extras_year as $facetID =&gt; $datevalue}
  {set $date_count =
$search_yearmonth['SearchExtras'].facet_fields[0].countList[$facetID]}
  {set $date_ts = $datevalue|strtotime}
 <li><a href={concat('/Blogs/(year)/',$date_ts|datetime( 'custom', '%Y' ))|ezurl}
title="Archives : {$date_ts|datetime( 'custom', '%Y' )} // {$date_count}
Billet(s)">{$date_ts|datetime( 'custom', '%Y' )}</a></li>
 {/foreach}
 </ul>
```

```
</li>
<li id="blog_block_11" class="colonne_block">
 <h1>Archives par mois / années :</h1>
 <ul class="{$current_css} list">
 {foreach $search_extras_yearmonth as $facetID =&gt; $datevalue}
  {set $date_count =
$search_yearmonth['SearchExtras'].facet_fields[1].countList[$facetID]}
  {set $date_ts = $datevalue|strtotime}
<li><a href={concat('/Blogs/(year)/',$date_ts|datetime( 'custom', '%Y' ),'/(month)/',
$date_ts|datetime( 'custom', '%n' ))|ezurl} title="Archives : {$date_ts|
datetime( 'custom', '%F %Y' )} //{$date_count} Billet(s)">{$date_ts|datetime( 'custom',
'%F %Y' )|upfirst}</a></li>
 {/foreach}
 </ul>

</li>
{undef $date_ts $date_count $search_yearmonth $search_extras_yearmonth}
```

Note :
The presented fetch is quite basic, for an easier understanding of the mechanism. This code needs to be elaborated to achieve the expected functional result (be able to use fi lter for instance). This is however not this tutorial's purpose, the official documentation already giving all hints for doing this.

2nd Note :
The '**sort**', '**alpha**' statement does not actually specify an alphabetical sort. It rather helps specifying that no '**count**' sort should occur (number of items matching a given facet). In this case, Solr automatically uses an '**increasing**' sort, based on its index and the datatype of the concerned fi eld (this explains the usage of the reverse operator to get an 'increasing' list).

9 Conclusion

This second post out of three describes how to add additional fi elds in Solr/Lucene's index when indexing content objects. This allows for :

- Easing or even making possible the construction of some fi lters, search sort methods or content navigation schemes
- Optimizing development time
- Optimizing response time by alleviating or removing load from the SQL stack

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

10 Resources

- eZ Find 2.2 official documentation :
  http://ez.no/doc/extensions/ez_fi nd/2_2
- eZ Find source code :
  http://pubsvn.ez.no/websvn2/listing.php?repname=ez_extensions&path=%2Fezfi nd %2Fezp4%2Fstable%2F2.2%2Fextension%2Fezfi nd%2F&#ad1cf75aaf3713d19a78efec9843b788a
- Tutorial on eZPedia.org : how to create a template operator :
  http://ezpedia.org/ez/template_operators
- ezarchive template operator source code :
  http://pubsvn.ez.no/websvn2/fi ledetails.php?repname=ez_extensions&path=%2Fezwebin%2Ftrunk %2Fpackages%2Fezwebin_extension%2Fezextension%2Fezwebin%2Fautoloads%2Fezarchive.php
- eZ Find settings files :
  http://pubsvn.ez.no/websvn2/fi ledetails.php?repname=ez_extensions&path=%2Fezfi nd %2Fezp4%2Ftrunk%2Fextension%2Fezfi nd%2Fsettings%2Fezfind.ini
- ezfSolrDocumentFieldBase class source code :
  http://pubsvn.ez.no/websvn2/fi ledetails.php?repname=ez_extensions&path=%2Fezfi nd %2Fezp4%2Ftrunk%2Fextension%2Fezfi nd%2Fclasses%2Fezfsolrdocumentfi eldbase.php
- Apache Solr Wiki :
  http://wiki.apache.org/solr/

11 About the author : Gilles Guirand

Gilles Guirand is a certifi ed 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.

12 License