Advanced eZ Find

September 5-9, Bol, Croatia

Paul Borgermans

About me

- 11+ years in the eZ ecosystem
  - eZ Lucene → eZ Solr → eZ Find
- Fancying :
  - Apache Lucene family of projects (mainly Solr)
  - NoSQL (Not only SQL) and scalable architectures
  - eZ Publish & CMS systems in general
  - Semantic aspects
  - PHPBenelux Community & Conference

- Contact

Gmail paul.borgermans@gmail.com

@paulborgermans

Agenda & Focus points

- Helicopter view
- Apache Solr setup and configuration
- Apache Solr internals: text analysis
- Using eZ Find in templates
  - New powerful range facets
  - New boost parameters for search result tuning
- Extension points in eZ Find
  - Datatype plugins
  - Index time plugins

Helicopter view of eZ Find and Apache Solr

eZ Find main search features

- Tuneable relevancy ranking on top of internal similarity algorithms (and sorting)
- Highlighting of keywords
- Filtering
- Facets (drill down navigation)
- Automatic related content
- Multilingual text processing
- Performance
- Adaptive to your domain data models
- Leverages Apache Solr/Lucene

Apache Solr Curriculum Vitae

- Open source Apache Lucene project, started by Yonik Seeley
- Standalone, enterprise grade search server built on top of Lucene
- Lives in a Java servlet container
- Access through a REST-ful API
  - HTTP
  - Primary payload in requests: XML
  - Other response formats: PHP, JSON, …

Solr in a nutshell

- State of the art, advanced full text search and information retrieval
- Fast, scalable with native replication features
- Multitenant
- Flexible configuration
- Document oriented storage
- Geospatial search
- Native cloud features
  under active development, almost complete, Apache Solr 4.0 (beta)

Solr Architecture

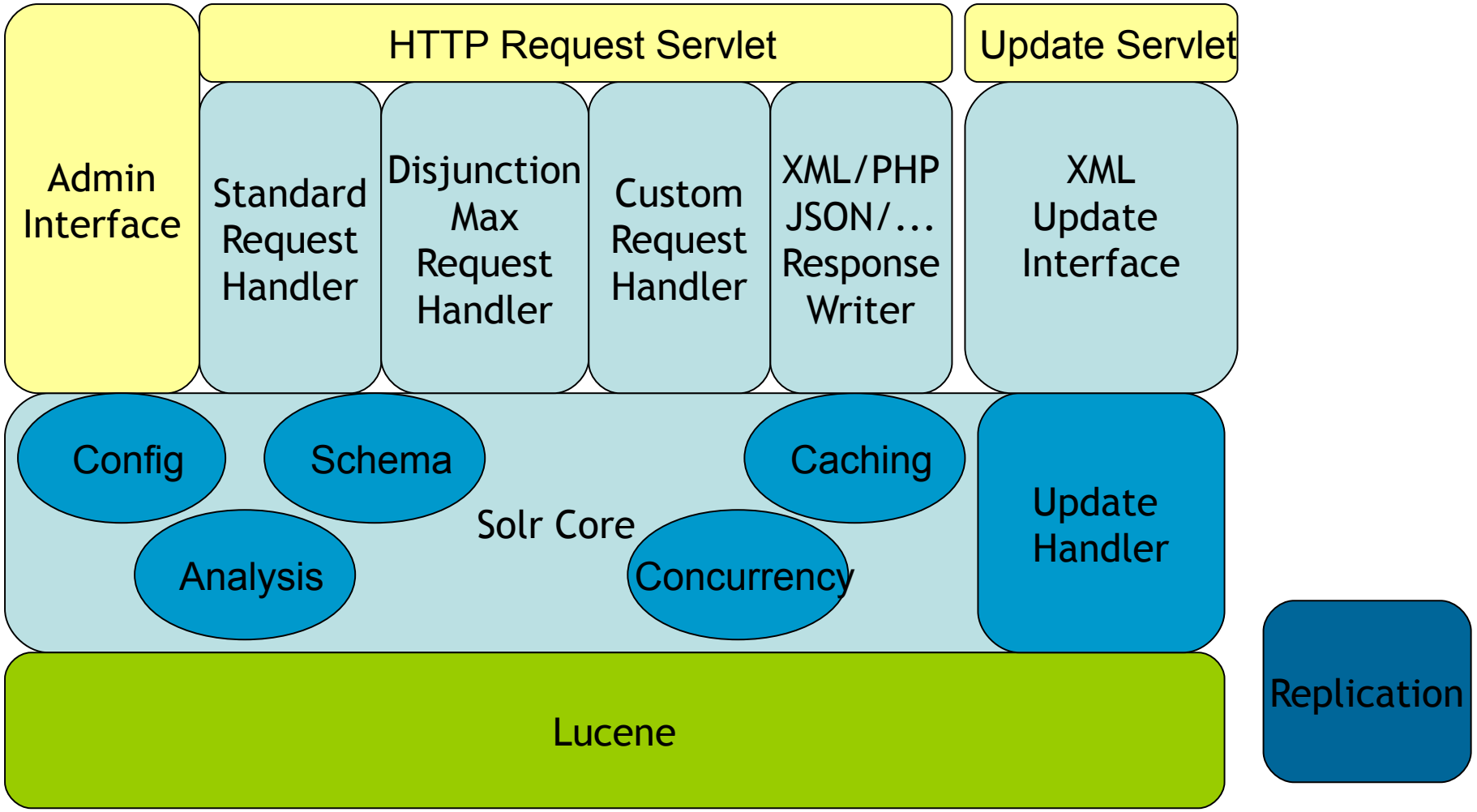


Figure credit: Yonik Seeley

Using eZ Find/Solr beyond search

eZ Find with two main additional roles

- eZ Find/Solr as IR engine/layer
  - Speed up template rendering, especially with complex dynamic pages
- eZ Find/Solr as a content and integration engine
  - Document oriented storage system (hello NoSQL)
  - Archive use-case
  - External content

Integrate legacy/other data with Solr

- Use the Solr Data Import Handler
  - Able to index DB's & XML files directly
  - fire simple requests to Solr to actually index/update using eZ Find low level API
- Alternative: Apache Connector framework (incubator project, includes http crawler)
- DIY: eZ Find API (extension/ezfind/classes)
  - eZSolrBase
  - eZSolrDoc
  - ezfSolrUtils

eZ Tika: indexing binary files

- Based on Apache Tika
  - Text and meta-data extraction for a large variety of file types

- Extension provides
  - Standalone binary (yet another Java .jar)
  - Configuration settings
  - A stub binary file handler
  - A wrapper shell script

And You?
Special Use Cases?

Installation and configuration

- Requirements
- Installing the extension
- Basic installation/activation of Solr
- Multi-lingual/multicore setups
- Basic configuration options of eZ Find
- The Solr web administration interface

Hands on!

Solr backend-requirements

- Java VM
  - JRE 6 or 7 (OpenJDK, Oracle/Sun)
- Servlet container
  - Jetty shipped by default, Tomcat, ...
  - Security to be configured (by default: open)
  - See also http://wiki.apache.org/solr/SolrInstall
- For larger sites/indexes: enough RAM
  - Yet leave enough for the OS/file caching

Extension installation and activation

- eZ Find extension activated the usual way
  - `ActiveExtensions[]=ezfind`
  - (!) Regenerate autoloads if using direct editing of ini settings
- Execute the DB upgrade script
  - Used for elevation
  - See `extension/ezfind/sql/<db>`

Starting the Solr backend

- Direct
  - `java -Dezfind -Xms512M -Xmx512M -jar start.jar`
- Direct, alternate port
  - `java -Djetty.port=8985 -Dezfind -jar start.jar`
- As a service: unix
  - See: `extension/ezfind/bin/scripts/<dist>/`
- As a service: windows
  - Use Tomcat
  - http://wiki.apache.org/solr/SolrTomcat
- Make sure Java runs in server mode!

Indexing

- Initial indexing: use dedicated eZ Find provided script
  - `php extension/ezfind/bin/php/updatesearchindexsolr.php -s <admin siteaccess> --php-exec=php –conc=2`
  - `typical speed: 5-25 objects /sec`
- Re-indexing with important changes
  - Schema changes in the backend Solr
  - ezfind.ini changes related to field mapping
  - Switching from single to multi-core setups
  - Upgrades of eZ Find and/or Solr

Multilingual features?

- Quite some Solr configuration options deal with language specific features
- Default installation: single “core”
  - Good for single language eZ Publish sites
  - Bad for multilingual setups
    - Shares synonyms, stop words, configuration, …
    - Index pollution decreases relevancy ranking

Multi-core setup

- Every language / tenant has its own
  - Index
  - Tunable analyzer options
  - Spell checker dictionary
  - Synonyms
  - Stop word list
  - Elevate configuration
- Additional bonuses:
  - slight increase in performance
  - core admin features

Multilingual / multi-core setup ...

- What is a core actually?
  - A dedicated Solr instance than can co-exist with other cores
  - Shares same servlet, port
  - Each core is independent
    - Own URI part, identifier (ex localhost
    - Own index
    - Own config files

How to configure multicore setups ...

- Easy way
  - Create a new Solr home directory under the java subdir
  - Put a config file solr.xml which specifies the cores
  - Copy the conf and data directories
  - Specify the solr home when starting the servlet container

```
sudo java -jar -Dsolr.solr.home=solr.multicore -jar start.jar
```

How to configure multicore setups ...

- Adapt ini settings for eZ Find
  - solr.ini
    - Cores definition (URI's)
  - ezfind.ini in case of core per language
    - Activate multi core
    - Language to core mappings

More basic configuration options

- Enable delayed indexation of objects (site.ini)
  - Editors will be happier (“faster publishing”)
  - Can be done globally or per class (recommended for binary file indexing)
  - Downside: objects will only be in search results after the configured cronjob has run

More configuration options (...)

- Disable optimize on commit
  - Configure cronjob to do it once per day/week
  - Makes files compact
  - If many delete operations happen, optimize accordingly

More configuration options (...)

- Enable commitWithin (ezfind.ini)
  - Use case: large sites, where commits can also take some time
  - Specified in milliseconds
  - No cronjobs needed
- Only in special cases: disable direct commits
  - Indexing
  - Delete operations

More basic configuration options (...)

- Search handler (if upgrading from older versions)
  - Defaults to “ezpublish” now (Apache Solr 3.6.1)
  - For end users: best use “ezpublish”, you can override in template fetch functions
  - Supports Lucene syntax (wildcards)
  - Does partial language analysis in presence of wildcards

The Solr administration interface

- http://localhost:8983/solr/<core>/admin
- Basic check
  - if everything is configured correctly
  - Statistics (documents indexed, JVM, …)
- Advanced use: see later

Solr Admin (ezfind)

192.168.0.163:8983
cwd=/srv/www/ezp201202/extension/ezfind/java SolrHome=solr.multicore/eng-GB/
HTTP caching is OFF

Apache Solr

Solr	[SCHEMA] [CONFIG] [ANALYSIS] [SCHEMA BROWSER] [REPLICATION] [STATISTICS] [INFO] [DISTRIBUTION] [PING] [LOGGING]
Cores:	[eng-GB][FRE-FR][NOR-NO]
App server:	[JAVA PROPERTIES] [THREAD DUMP]

Make a Query	[FULL INTERFACE]
Query String:	*:*
	Search

Assistance	[DOCUMENTATION] [ISSUE TRACKER] [SEND EMAIL] [SOLR QUERY SYNTAX]
	Current Time: Sat Sep 15 22:22:34 CEST 2012
	Server Start At: Sat Sep 15 22:22:23 CEST 2012

Installation exercise

- Fire up your eZ Summer Camp VM
- Pull the latest master version from https://github.com/ezsystems/ezfind
- Install / activate eZ Find in multi-core setup
- Install / activate Solr (included in eZ Find)
- Re-index from the command line
- Try searching
- Tweak the basic options to see their effects
- Try the solr admin interface

A deeper dive into Apache Solr

- From index → document → field
- Schema.xml
- What happens under the hood
- Request handlers

The Solr/Lucene index

- Inverted index
- Holds a collection of “documents” (hello NoSQL)
- Document
  - Collection of fields
  - Flexible schema!
  - Unique ID (user defined)
- Solr uses a XML based config file:

  `schema.xml`

Field types and fields

- Various field types, derived from base classes
- Indexed (optional)
  - usually analyzed & tokenized
  - makes it searchable and sortable
- Stored (optional)
  - contains also the original submitted content
  - content can be part of the request response
- Can be multi-valued!
  - opens possibilities beyond full text search

Field definitions: schema.xml

- Field types
  - text
  - numerical
  - dates
  - location
  - … (about 30 in total)
- Actual fields (name, definition, properties)
- Dynamic fields
- Copy fields (as aggregators)

schema.xml: simple field type examples

```
    <fieldType name="string" class="solr.StrField"
 sortMissingLast="true" omitNorms="true"/>

    <!-- boolean type: "true" or "false" -->
    <fieldType name="boolean" class="solr.BoolField"
 sortMissingLast="true" omitNorms="true"/>

    <!-- A Trie based date field for faster date range
 queries and date faceting. -->
    <fieldType name="tdate" class="solr.TrieDateField"
 omitNorms="true" precisionStep="6"
 positionIncrementGap="0"/>

  <!-- A text field that only splits on whitespace for exact matching
of words -->
    <fieldType name="text_ws" class="solr.TextField"
positionIncrementGap="100">
      <analyzer>
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
      </analyzer>
    </fieldType>
```

schema.xml: more complex field type

```
  <!-- A general unstemmed text field - good if one does not know the language of the field -->
    <fieldType name="textgen" class="solr.TextField" positionIncrementGap="100">
      <analyzer type="index">
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt"
enablePositionIncrements="false" />
        <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1"
generateNumberParts="1" catenateWords="1" catenateNumbers="1" catenateAll="0"
splitOnCaseChange="0"/>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
      <analyzer type="query">
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true"
expand="true"/>
        <filter class="solr.StopFilterFactory"
                ignoreCase="true"
                words="stopwords.txt"
                enablePositionIncrements="true"
                />
        <filter class="solr.WordDelimiterFilterFactory" generateWordParts="1"
generateNumberParts="1" catenateWords="0" catenateNumbers="0" catenateAll="0"
splitOnCaseChange="0"/>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
    </fieldType>
```

Analysis

- Solr does not really search your text, but rather the terms that result from the analysis of text
- Typically a chain of
  - Character filter(s)
  - Tokenisation
  - Filter A
  - Filter B
  - …

Solr comes with many tokenizers and filters

- Some are language specific
- Others are very specialised
- It is very important to get this right

otherwise, you may not get what you expect!

Text analysis examples

String	Field type “text”	term position 1	term position 2
iPad	=>	i	pad ipad
élève.	=>	elev	
Viele Grüße	=>	viel	gruss

Character filters

- Used to cleanup text before tokenizing
  - HTMLStripCharFilter (strips html, xml, js, css)
  - MappingCharFilter (normalisation of characters, removing accents)
  - Regular expression filter

Tokenizers

- Convert text to tokens (terms)
- You can define only one per field/analyzer
- Examples
  - WhitespaceTokenizer (splits on white space)
  - StandardTokenizer
  - CJK variants

Additional filters

- Many possible per field/analyzer
- Many delivered with Solr out of the box
- If not enough, write a tiny bit of Java or look for contributions

- Examples ...

Phonetic filters

- PhoneticFilterFactory
- “sounds like” transformations and matching
- Algorithms:
  - Metaphone
  - Double Metaphone
  - Soundex
  - Refined Soundex

Reversing Filter

- Reverses the order of characters
- Use: allow “leading wildcards”
- *thing => gniht*
- A lot faster (prefixes)

Synonyms

- Inject synonyms for certain terms
- Language specific
- Best used for query time analysis
  - may inflate the search index too much
  - decreases relevancy

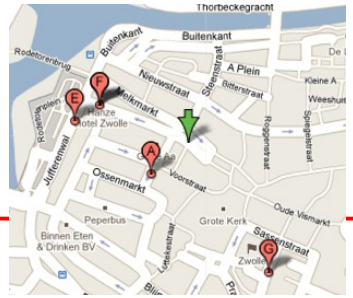
Stemming

- Reduce terms to their root form
  - Plural forms
  - Conjugations
- Language specific (or not relevant, CJK)
- Many specialised stemmers available
  - Most european languages
  - Some exotic ones through contributions outside ASF

Copy fields

- Analysis is done differently for
  - searching/filtering
  - faceting/sorting
- Stemming and not stemming in different fields can increase relevance of results

- Use copy fields in schema.xml or do it client side

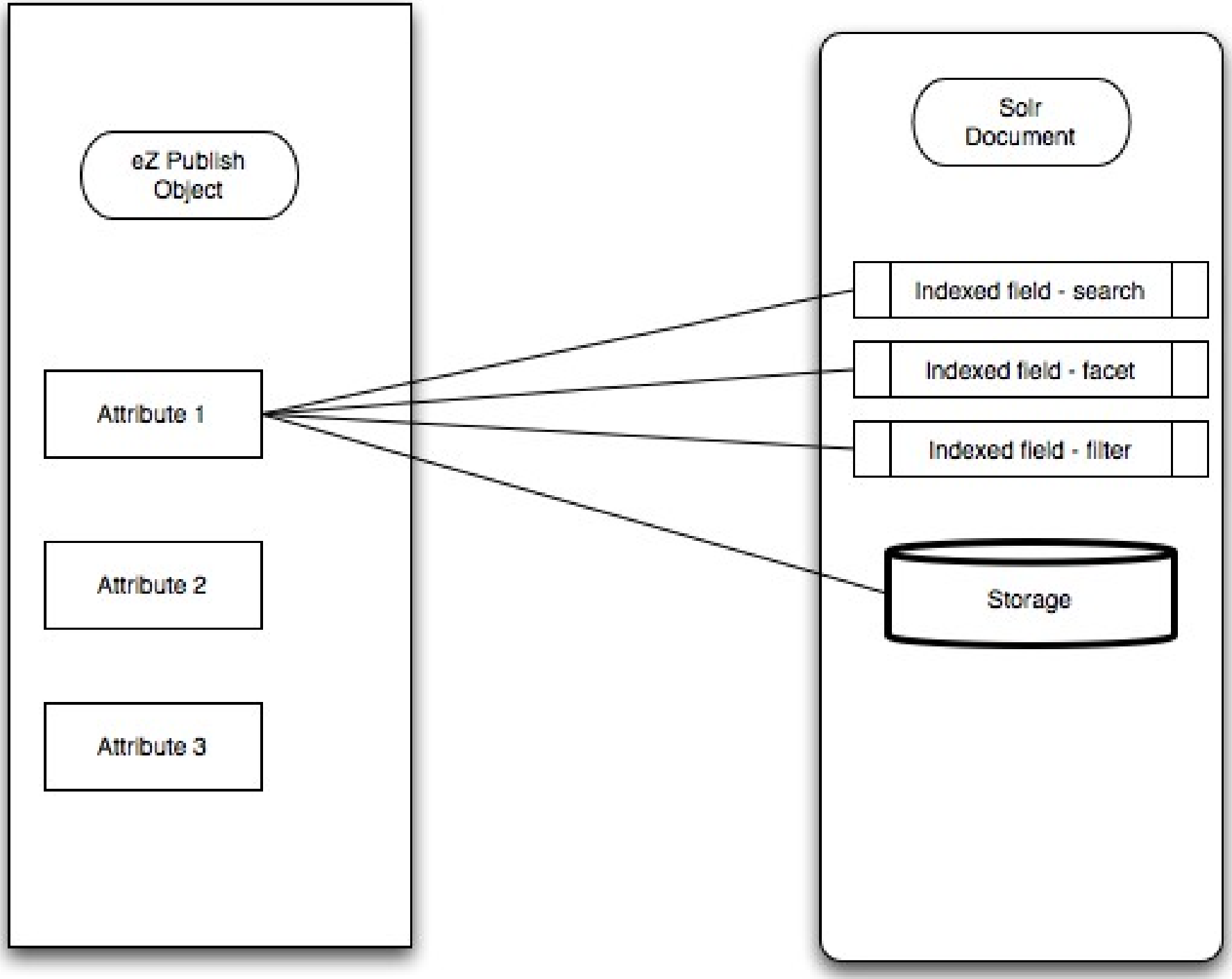


Geospatial fields

- Solr dedicated fields
  - Latitude Longitude type (trunk)
- Special geospatial functions in filtering & boosting
  - Haversine distance (geosphere)
  - Simple ranges (squares in 2-D)
  - Special query constructs (upcoming)

Dedicated fields for every context in eZ Find if configured

- Context
  - Search
  - Facets
  - Filtering (usually the same as search)
  - Sorting
- ezfind.ini
- Also for custom handlers if needed (see part 6)



Template fetch functions

- Searching
- Navigation elements
  - Focus on new range facets
- Lucene query syntax (mainly for filtering)
- Query time boosting parameters

Searching

- You can use the standard content/search templates and parameters
- But **much better**: dedicated fetch functions
  - fetch(ezfind, search, hash(query, 'eZ Systems'))

Dedicated fetch parameters

Name	Type	Description	Required
query	String	Search query string	No
offset	Integer	Result offset	No
limit	Integer	Result count limt	No
sort_by	Array	Sort definition	No
facet	Array	Facet query definition	No
filter	Mixed	Search filter, independent from ranking	No
class_id	Mixed	Class id limitation	No
subtree_array	Array	List of subtree limitations	no

See doc.ez.no for the full list of parameters

Syntax

- Query using “eZ Publish/eDismax” handler
  - Normal keywords
  - + or – prefix to denote required or excluded
  - Also wildcards, almost full Lucene syntax
- Filter
  - Full Lucene syntax (inluding boolean, ranges, ..)
- Sorting
  - If really needed, boosting usually a better option

Filtering

- AND logic connects array elements using Standard Lucene syntax.
- Attribute identifiers are mapped to Solr fields

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             filter, array( 'car/in_stock:1',
                            'car/make:Alfa Romeo',
                            'car/model:8C' ) ) )
```

eZ Find and field names

- The normal case for filtering: 3 ways
  - array('article/title:a*') //will generate 2 filters
  - array('title:a*') //cross class attribute filtering
  - array('attr_title_s:a*') //using raw field names

eZ Find raw field names

- Main principle: <source>_<identifier>_<type>
  - <source>: meta, attr, as
  - <identifier>: eZ Publish native identifier
  - <type>: Solr field type mapping (schema.xml)
- Extra
  - timestamp: time when the object was indexed
  - ezf_df_text: aggregator for all text
  - ezf_sp_words: spellcheck source
- Subattributes: another separator with 3x '_'
  <source>_<identifier>___<sub_attr_id>_<type>

Sorting

Key	Description
relevance	Relevance (default). Sorts the result by Solr internal relevancy estimates[1]
score	Alias to relevance
<class attribute>	Content class attribute. See description below for details.
modified	Modified time
published	Published time
author	Author name
class_name	Content class name
class_id	Content class identifier or content class id.
name	Content object name
path	Node location path

See doc.ez.no for the full list of parameters

Facets pre eZPublish 5

field	Specifies a specific field which should work as a facet. This is specified by <class_identifier>/<class_attribute>[/<sub_structure] The substructure is only available for complex data types. To enable <sub_structure> support, support for native indexing must implement this (See design for indexing native types). Other fields supported: • author - eZ Publish content object author. • class - eZ Publish content class. • translation – Translation.
query	Facet query1. The facet quries are used to specify facets for sub-selection of content objects attributes.
prefix	Limits the facet fields to only list facets where the field value starts with <facet.prefix>
sort	Sort by 'count' or 'alpha'. 'alpha' will sort the facet results by field value, alpha-numerical.
limit	Maximum number of facets. Default value is 20.
offset	Offset, default value 0.
mincount	Return only facets with more results than <facet.mincount>. Default value 0.
missing	If set to true, the facet count will include empty results. Default value false.
date.start	Start date for facet.
date.end	End date for facet
date.gap	Size of date range

New facet type: range

- For numerical and date ranges
  - (old date facets are deprecated)
- Example:

```
fetch( ezfind, search, hash( 'query',
'$queryString,
     facet',array(
       hash( 'range',
        hash('field', 'published',
         'start', 'NOW/YEAR-3YEARS',
         'end',   'NOW/YEAR+1YEAR',
         'gap',   '+1YEAR' ) ) ) ) )
```

Range facet parameters

- Mandatory
  - 'field' (can also be custom Solr fields)
  - 'start' (numeric/date)
  - 'end' (numeric/date)
- Optional
  - 'hardend'
  - 'include'
  - 'other'

See also http://ow.ly/dskqb (whats new in eZ Find 2.7)

Lucene/Solr query syntax

- For new eDismax/ezpublish, “standard” query handler and also filtering
- Very rich and complete
  - Maybe even a bit bloated
  - All boolean constructs (AND, OR, grouping, …)
  - Ranges
  - Fuzzy search
  - Wildcards (! without some tricks, only prefix like)
  - “sounds like”

Lucene/Solr query syntax

- Solr adds functions
  - Math
  - Geographical
  - …
- Resources
  - http://wiki.apache.org/solr/SolrQuerySyntax
  - http://wiki.apache.org/solr/FunctionQuery
  - http://lucene.apache.org/java/2_9_1/queryparsersyntax.html

Lucene/Solr Syntax examples

- Terms and phrases
  - Term: article_title_t:brown
  - Phrase: article_title_t:”the yellow note”
- Wildcards
  - Using '*': article_title_t:pro*
  - Using '?': article_title_t:ma?ch
- Allowing certain “edit distance”: fuzzy searches
  - article_title_t:march~0.7
- Proximity
  - article_body_t:”john doe”~10

Lucene/Solr Syntax examples (..)

- Ranges
  - Inclusive/exclusive
  - One part may be open ended using '*'
- Inclusive
  - attr_quantity_si:[1 TO 5]
- Exclusive
  - attr_quantity_si:{0 TO 6}
- Open ended
  - attr_quantity_si:[6 TO *]

Date handling

- No real limits like unix timestamps
- Date values in ISO 6801 format

  yyyy-mm-ddThh:mm:ssZ (in UTC)
- Macro like syntax
  - “NOW”
  - “NOW/DAY-1YEAR”
  - “NOW+3DAYS”
- Now in master: format datetime with 'solr' format (via Peter Keung)

Tuning options for relevancy

- Index time
  - Class, attribute, datatype “permanent boosting”
  - Best used after some real-life measurements (logs, user feedback, dedicated tests)
  - ezfind.ini
- Query time
  - For ezpublish/eDismax request handlers
  - Fields (also meta-data)
  - Function queries
  - Multiplicative and additive boosting

Index time boosting

- Available for:
  - Classes
  - Attributes
  - Datatypes
- Boost factor ranges
  - [0 … 1] suppression
  - [1 …] boosting
- ezfind.ini

Query time boosting

- Boosting types and corresponding sub-parameters
  - 'field'
  - 'mfunctions'
  - 'queries'
  - 'functions'
- Properly supported only since eZ Publish 5, eZ Find master

Query time boosting: 'fields'

- Example

```
.. 'boost_functions',
  hash('fields',array('article/tags:3'))..
```

or with a raw Solr field identifier

```
.. 'boost_functions',
  hash('fields',array('attr_tags_lk:3'))..
```

Query time boosting: 'mfunctions'

- Multiplicative
  - No need to know raw relevancy numbers
  - Multiplies the individual score with the specified function(s)
- Example

```
... 'boost_functions',
 hash('mfunctions', array('recip(
    ms(NOW/DAY,meta_published_dt),
    3.16e-11,0.5,0.5)' )) ...
```

Query time boosting: 'queries'

- These are added to the main query and need to follow the Solr/Lucene query format ans specify the boost factor explicitely for it
- Example

```
..'boost_functions', hash('queries',
  array(
   'meta_class_identifier_ms:article^10')).
```

- Also available in ini settings (applies always)

```
[QueryBoost]

#RawBoostQueries[]

RawBoostQueries[]=meta_class_identifier_ms:summary^4
```

Advanced configuration and tuning

- Tuning options for relevancy
  - Index time
  - Query time
- Elevation

Query time boosting: 'functions'

- These are like mfunctions, but add their value to the relevancy score
- Usually 'mfunctions' are the easier choice
- Example

```
..'boost_functions',
hash('functions',
array('sum(product(attr_importance_si,0.1),1)')
) ..
```

Solr has many functions to use

- Strings
- Numbers and mapping
- Date math
- Geospatial

http://wiki.apache.org/solr/FunctionQuery/

Absolute boosting: elevation

- If a query term matches, one or more objects are pushed to the top
- Dedicated admin interface :)
- Query term has to be part of the object

Extending eZ Find

- Custom handlers for (custom) datatypes
- Index time plugin mechanism
- Using the API for creating your own module/view functions and template operators

Custom datatype handlers

- Usually for “complex” datatypes
  - Subfields (!)
- Can optionally be context aware
  - Facets/Sort
  - Search
  - Filter

Create your own datatype handler

- Derive from a base class:
  - `ezfSolrDocumentFieldBase`
  - Naming convention
- Provide at least two methods
  - “schema” data: (sub)field names
  - Data to index
- Starting point
  - extension/ezfind/classes:
    ezfsolrdocumentfielddummyexample.php
- Add in ezfind.ini, [Indexoptions]

Overview of eZ Find / Solr lower level API

Base classes to know

- extension/ezfind/classes
  - ezsolrbase.php

    handles communication with Solr backends
  - ezsolrdoc.php

    creates proper XML structures for indexing
  - ezfsolrutils.php

    easy to use higher level functions
- Let's have a look ...

Index Time Plugin Mechanism

- Write your own functions to:
  - Expand the Solr fields per object
  - Modify existing fields
  - Change per object and per field boosting dynamically
- Use cases
  - Complex custom data, partially external
  - Boost documents based on page views, user score, ….

Index time plugins (...)

- Implement the following interface

```
interface ezfIndexPlugin
{
    /**
     * @var eZContentObject $contentObject
     * @var array $docList
     */
    public function modify(eZContentObject $contentObject, &$docList);
}
```

- `docList` is the array of eZSolrDocs to be sent to Solr, one per language for the given contentObject

Index time plugins (...)

- Activate your plugin in ezfind.ini
  - Global
  - Per content class

```
[IndexPlugins]

# Allow injection of custom fields and manipulation of fields/boost parameters
# at index time
# This can be defined at the class level or general
General[]
#General[]=ezfIndexParentName

#Classhooks will only be called for objects of the specified class
Class[]
Class[myspecialclass]=ezfIndexParentName
```

Thank You!

Exercises

Warm up exercise

- Make sure you are on the latest code base
- Play with the Lucene syntax supported by the new ezpubish/eDismax handler:
  - Proximity searches
  - Fuzzy searches
  - Wildcards
  - Ranges

And see what happens

Exercise: boosting

- Use the new 'mfunctions' parameter to boost more recent values
- Boost higher rated articles

Exercise: Facets & attribute filtering

- Try to facet and filters on classnames
  - As a field facet (enumerate all classes)
  - As a set of several query facets (enumerate only a selection)
- Range facets
  - Date ranges

Exercise: sub-attribute filtering on a related object

- Create an override template for a dummy node
- In the template add code for fetching with ez find, search with an empty query string, but use a filter with a subbatribute clause

```
{def $searchResults = fetch( 'ezfind', 'search',
                hash( 'query', '',
                'filter', array('article/testrelation/caption:specialvalue1')))
```

A last plug: You are invited!

conference.phpbenelux.eu/2013/