UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Barcelona School of Informatics (FIB)
Master in Information Technology

Improving Acquia Search and the Apache Solr Search Integration Drupal module

by

Nick VEENHOF

Supervisor: B. Chris BROOKINS
Co-Supervisor: Dr. Ir. Peter WOLANIN
Tutor/Professor: Prof. Dr. Ir. Carles FARRÉ TOST

Academic Year 2011–2012

Preface

Preface, what should come here?

Nick Veenhof, February 2012

License Statement

Nick Veenhof, February 2012

Improving Acquia search and the ApacheSolr Module

by

Nick VEENHOF

Master Thesis in order to acquire a Master in Information Technology

Academic Year 2011–2012

Supervisor: B. Chris BROOKINS
Co-Supervisor: Dr. Ir. Peter WOLANIN
Tutor/Professor: Prof. Dr. Ir. Carles FARRÉ TOST
Barcelona School of Informatics (FIB)
Master in Information Technology
BarcelonaTech

Abstract

This work is intended to show the upgrade process of a module on drupal.org using community tools. This was performed as part of an internship at Acquia Inc. More specifically it was focussed on creating a stable release of the Apache Solr Search Integration Module for Drupal 7 and eventually also backport this to Drupal 6. Firstly there was an analysis state and a brief introduction to how the system worked and how to co-operate with a community. From this, existing problems were identified and thrown in a roadmap. Community projects have a very dynamic rythm and issues could rise up or get resolved because thousands of persons had access to the code base. These challenges are described and tips are given on how to cope with such a development process. Also it describes what challenges a backport has and how to resolve them. Finally, there is an explanation of the Acquia Search service and the process to upgrade the server park from Solr 1.4 to Solr 3.x (initially 3.4, finally 3.5) that includes the process of writing a java servlet for managing authentication over rest services using RFC2104 HMAC encryption.

Keywords

Drupal, Apache Solr, Lucene, Acquia, Acquia Search, Acquia Network, Search Technology

Contents

Chapter 1

Introduction

1.1 Web and Search

It wouldn't be wrong to start with a quote from the famous paper of Sergey Brin and Lawrence Page. "The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph" [61] In my personal opinion I also experienced that people "Google" more and more and this phenomenon intrigued me and many others. The web won't stop growing and content is added in amounts that we can't imagine. Even though Google does its very best to index every piece of content it still lacks in a more customizable way to find data. You, as a reader, will probably already have searched in depth in a search engine other than Google. For example, ebay.com has a very specific search engine that allows their customers to find products and goods that are are exactly what to user is searching for by narrowing down the results using Facets [1]

Another missing piece in the search part of the global web is the ability to search in restricted content. Say, for example, an intranet can't benefit from a global search, hence a search engine that indexes content in a customized way is necessary.

A numerous amount of projects [2]allow you to customize the indexing process while still sup-

[1] A faceted classification system allows the assignment of an object to multiple characteristics (attributes), enabling the classification to be ordered in multiple ways, rather than in a single, predetermined, taxonomic order. For example, a collection of books might be classified using an author facet, a subject facet, a date facet, etc.

[2] `http://en.wikipedia.org/wiki/List_of_enterprise_search_vendors` has a list with most of the current enterprise search solutions

porting hundreds of thousands documents which contains fields with customized data. Creating an application that includes integration with access permissions is not easy but it is do-able.

1.2 Open Source & Community

This work is the result of many hours hard work and not only from myself, as the author but also from a complete community. These communities have changed the way how we look at software. In programming classes in university a student is taught a different way of designing software, the control of this process is fully his. There are numerous courses going from basic Java to Advanced Web Technologies to IBM rational rose project management in the FIB department of the UPC. While you can learn a ton from these courses it is never enough and by being an active member of a community the obligation you have to follow and participate in life-long learning is fulfilled. Every day there might be an aha-erlebnis [3] or frustrations but in the end it is worthwhile for the personal evolution.

Also, since this topic is about Search Applications and Web Applications we only focus on Open Source tools that help us in achieving our goals. See section 3 on page 8 to find out more about the specifics of these tools and why these tools were chosen.

Working in a community is, similarly, another way of creating solutions for a set of existing problems but involves a different way of making decisions and looking at software. It is great if the code that is written can be shared and is being used by thousands of people and can be corrected by those same group of people. While code will never be perfect, different people have used the same codebase to solve existing problems and they have been saving time and resources. The company where this thesis was executed, Acquia [4], is doing an fantastic job in supporting these very necessary skills and promoting shared knowledge.

As Dries Buytaert, the man who initially built Drupal and founded Acquia, once said :

> First, Open Source adoption in the enterprise is trending at an incredible rate – Drupal adoption has grown a lot in 2009 but we saw by far the biggest relative growth in the enterprise. Fueling this movement is the notion that Open Source options present an innovative, economically friendly and more secure alternative to their

[3] An insight that manifests itself suddenly, such as understanding how to solve a difficult problem, is sometimes called by the German word Aha-Erlebnis. It is also known as an epiphany.

[4] `http://www.acquia.com`

> costly proprietary counterparts. Second, Cloud Computing is a transformational movement in that it enables continual innovation and updating - not to mention a highly expandable infrastructure that will reduce the burden on your IT team.
>
> It is no surprise that Acquia's strategy is closely aligned with those two trends: Drupal Gardens, Acquia Hosting and Acquia Search are all built on Open Source tools and delivered as Software as a Service in the cloud. Combining Open Source tools and Cloud Computing makes for the perfect storm for success. It provides real value to end-users and it enables companies to monetize Open Source. It creates a win-win situation. [5]

This quote mentions Acquia Search, the service that combines Apache Solr (The chosen search engine in this work) and Drupal to provide a superior search solution as a service especially focussed on integrating Drupal with Apache Solr in the Cloud. Everything that is done to improve this has also been open sourced, including this work.

Drupal and all contributed files hosted on Drupal.org are licensed under the GNU General Public License, version 2 or later. That means you are free to download, reuse, modify, and distribute any files hosted in Drupal.org's Git repositories under the terms of either the GPL version 2 or version 3, and to run Drupal in combination with any code with any license that is compatible with either versions 2 or 3, such as the Affero General Public License (AGPL) version 3. [62]

Apache Solr is licensed under the Apache License 2.0. Like any free software license, the Apache License allows the user of the software the freedom to use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software, under the terms of the license. The Apache License, like most other permissive licenses, does not require modified versions of the software to be distributed using the same license (in contrast to copyleft licenses such as the Drupal license). In every licensed file, any original copyright, patent, trademark, and attribution notices in redistributed code must be preserved (excluding notices that do not pertain to any part of the derivative works); and, in every licensed file changed, a notification must be added stating that changes have been made to that file. [63]

[5] `http://buytaert.net/open-source-in-the-enterprise-and-in-the-cloud`

1.3 Personal History

My story with Drupal starts in the beginning of 2007. I've done my Bachelor degree at the Katholic University of Ghent[6]. During the second year of my Bachelor I was asked, together with a 2 other people, to make a community site in Drupal to see what is was capable of. This was created in Drupal 5 and while it wasn't as powerful as it is now we were already able to integrate LDAP into the website and customize it to our needs. I do have to admit that we, as a group, made numerous mistakes against the ethics of customizing Drupal back in the days. [7]

This Drupal 5 project ended, my bachelor ended and I started looking for a job and ended up with a small company called Krimson [8]. This company taught me the correct way of programming Drupal and Immediately they said : "You can start with Drupal 6, very new and way better compared with the previous version". And so I did, I started creating websites full of interactivity and community, backends that connect directly to databases running on a mainframe and even planted the initial bean of interest in search (Solr) that later would appear to grow out as this thesis topic. That website is still active on the address of `http://www.kortingsreus.nl`. It is also there that I created my first Drupal module, namely apachesolr_ubercart [9]

Afterwards I moved to Spain to study at the UPC [10] and started to work half time at Ateneatech [11] and later for AT-Sistemas as one of the reference engineers for a huge Solr and Drupal powered website. [12]. Louis Toubes, one of the lead engineers, was able to give a small reference : "Nick tiene una capacidad innata de aprender por sí solo nuevas tecnologias y lo más importante es que el disfruta con ello. Sin duda, Nick es una de esas personas que desde el primer momento que la conoces sabes que aprenderás mucho de él."

During my studies at UPC I kept following the Drupal development and made numerous discussions with people and teachers on how software engineering should look at these projects. In the course of Advanced Web Technologies I even presented Drupal in classes : "Drupal as a

[6] `http://www.kaho.be`
[7] Insert drupal code standards here
[8] `http://www.krimson.be`
[9] `http://drupal.org/project/apachesolr_ubercart`
[10] `http://www.upc.edu/`
[11] `http://ateneatech.com/`
[12] `http://www.elsevier.es`

framework" [13]

There was only 1 logical step possible as my next step and that was doing an internship/thesis with Acquia. During my Erasmus period in Portugal I attended a Drupal Camp and I was also a presenter at the conference [14] and I've met Robert Douglas, one of the creators of the Apache Solr Integration Project for Drupal and approached him with the question if I would be able to do my internship with Acquia. After a long process with the UPC and with Acquia everything was set and the pieces of the puzzle fell in place.

Now, being 2012 and a couple of years later I still don't fully know what Drupal and all its derivatives are capable of since it keeps evolving and growing. And that's good because it keeps me as a person growing and it keeps me up to date with most of the latest web technologies. This work is a piece in the puzzle I tried to make during my short time involved with these concepts.

[13] http://prezi.com/10_1ssdjroao/

[14] http://lisboa2011.drupal-pt.org/sessoes/apachesolr-the-complete-search-solution

Chapter 2

Objectives

The student will be asked to be on-site at the headquarters in Boston for a couple of weeks in order to meet the team and to get to know the company in order to gather all the information necessary to reach the objectives set further in this document. He will follow and join meetings to obtain a good insight in the requirements of the project and learn how to work under a Agile/Scrum based development methodology.

Being responsible for improving the Drupal Apache Solr search integration [1] project and the Acquia Search service is the common theme of the whole internship. This means adding additional features, keeping high quality and create upgrades and updates. The objective will be to exploit as much as possible from the latest Apache Solr 3.x branch while merging and keeping the software compatible with Apache Solr 1.4.

Communication will be a crucial part in order to succeed. The project has a worldwide scope, reaching out to more companies then just Acquia. This means he will have to be able to consult and make decisions after talking with a lot of end-users and other stakeholders. English will be the language of choice. This can happen by means of chat (IRC), on the Drupal community website [**?**], giving presentations in conferences or taking interviews. Finally the ability to work remotely, over a large distance and in a team, is an important skill to acquire.

Roadmap

- Bring Apache Solr [**?**] for Drupal 7 to a stable Release Candidate.
- Bring Facet Api [**?**] for Drupal 7 to a stable Release Candidate.

- Update the Acquia Search service [**?**] to the latest stable Apache Solr version. Upgrade the custom java code that was written to be able to authenticate customers.
- Backport to a new Drupal 6 branch all the new features that have been programmed into the Drupal 7 version of the Apache Solr Search Integration Module. This includes the backporting of the multisite module.
- Achieve mastery of the agile/scrum process, the open source software engineering methods, and the team communication processes used by Acquia.
- Empower the community to use the Apache Solr Search Integration project by means of Presentations, Blog posts and other interactions with community members.
- Create a multisite module to search between 2 or more Drupal sites or integrate this into the existing modules.

Chapter 3

Description

This chapter gives a short overview of technical concepts used in this work. It is not intended to be exhaustive and references are given for further reading

3.1 Acquia

Acquia is a commercial open source software company providing products, services, and technical support for the open source Drupal social publishing system and was founded by Dries Buytaert, the original creator and project lead of the Drupal project. With over two million downloads since inception, Drupal is used by web developers worldwide to build sophisticated community websites. Diverse organizations use Drupal as their core social publishing system for external facing websites and internal collaboration applications.

Acquia Search[1] is a plug-and-play service within the Acquia Network [2], built on Apache Solr [3] and is available for any Drupal 6 or Drupal 7 site. Acquia Search offers site visitors faceted search navigation and content recommendations to help them find valuable information faster. It is a fully redundant, high performance cloud service, with no software to install or servers to manage.

[1] http://acquia.com/productsservices/acquia-search

[2] http://www.acquia.com/products-services/acquia-network

[3] http://drupal.org/project/apachesolr

3.2 Apache Solr

Apache Solr is an open source enterprise search platform created on top of the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document (e.g., Word, PDF) handling. Providing distributed search and index replication, Solr is highly scalable.

Apache Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Apache Tomcat. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language.

Solr's powerful external configuration allows it to be tailored to almost any type of application without Java coding, and it has an extensive plugin architecture when more advanced customization is required. Further in this work you can find an example of such a plugin written to provide extra functionality for Acquia.

When looking at a Lucene index, compared to a relational database, it seems that the index is one database table and has very fast lookups and different specific filters for text search. It takes time to create an index like this. Solr adds a front-end to the Lucene backend and many other additions.

Fundamentally, Solr is very simple. One feeds it with information (documents) and afterwards you query Solr and receive the documents that match the query. Solr allows applications to build indexes based on different fields[4]. These fields are defined in a schema which tells Solr how it should build the index.

Using analyzers and tokenizers a search query is processed. Field analyzers are used both during ingestion, when a document is indexed, and at query time. An analyzer examines the text of fields and generates a token stream. Analyzers may be a single class or they may be composed of a series of tokenizer and filter classes.

Tokenizers break field data into lexical units, or tokens. Filters examine a stream of tokens and keep them, transform or discard them, or create new ones. Tokenizers and filters may be combined to form pipelines, or chains, where the output of one is input to the next. Such a

[4]Fields are different kinds of entries

sequence of tokenizers and filters is called an analyzer and the resulting output of an analyzer is used to match query results or build indices.

Although the analysis process is used for both indexing and querying, the same analysis process need not be used for both operations. For indexing, you often want to simplify, or normalize, words. For example, setting all letters to lowercase, eliminating punctuation and accents, mapping words to their stems, and so on. Doing so can increase recall because, for example, "ram", "Ram" and "RAM" would all match a query for "ram". To increase query-time precision, a filter could be employed to narrow the matches by, for example, ignoring all-cap acronyms if you're interested in male sheep, but not Random Access Memory.

The tokens output by the analysis process define the values, or terms, of that field and are used either to build an index of those terms when a new document is added, or to identify which documents contain the terms your are querying for.

3.3 Drupal

History Drupal is originally written by Dries Buytaert as a message board. It became an open source project in 2001. Drupal is a free and open-source content management system (CMS) written in PHP and distributed under the GNU General Public License. It is used as a back-end system for at least 1.5% of all websites worldwide ranging from personal blogs to corporate, political, and government sites including whitehouse.gov and data.gov.uk. It is also used for knowledge management and business collaboration.

The standard release of Drupal, known as Drupal core, contains basic features common to content management systems. These include user account registration and maintenance, menu management, RSS-feeds, page layout customization, system administrationand even a basic search functionalty. The Drupal core installation can be used as a brochureware website, a single- or multi-user blog, an Internet forum, or a community website providing for user-generated content.

Basic understanding A single web site could contain many types of content, such as informational pages, news items, polls, blog posts, real estate listings, etc. In Drupal, each item of content is called a node (internally called an entity), and each node belongs to a single content type (internally called entity type), which defines various default settings for nodes of that type,

such as whether the node is published automatically and whether comments are permitted. (Note that in versions below 7 of Drupal, content types were known as node types.)

Contributed Modules There are more than 12,000 free community-contributed addons, known as contrib modules, available to alter and extend Drupal's core capabilities and add new features or customize Drupal's behavior and appearance. Because of this plug-in extensibility and modular design, Drupal is sometimes described as a content management framework. Drupal is also described as a web application framework, as it meets the generally accepted feature requirements for such frameworks. While Drupal core (7) comes with advanced search capabilities it is still restricted by regular databases. [5]. The module that was created during this work is also defined as a contributed module.

Apache Solr Search Integration The Drupal module integrates Drupal with the Apache Solr search platform. Faceted search is supported if the facet API module is used. Facets will be available for you ranging from content author to taxonomy to arbitrary fields. The module also includes functionalities such as :

- Search pages, eg.: multiple search pages with optionally customized search results.
- Multiple environments to support multiple Solr servers.
- Comes with support for the node content type including dynamic fields.
- Can override the taxonomy pages and use output from Solr to generate taxonomy overview pages.
- Can override the user content listing pages using output from Solr to generate these.
- Custom Content types (entities) indexing through hooks.
- Add biases and boosts to specific fields or content types
- Range Query type, that in combination with facet API and Facet Api Slider a very rich faceting experience delivers to the end user.
- Supports a lot of customizations without having to modify the source code

[5] Any database that is compliant with the SQL standard should be able to run Drupal 7

Chapter 4

Exploration

This chapter describes the process of the index concepts in Apache Solr, explains how the apache solr works when indexing points out some of the improvements that should be made. It also takes a deeper look in the Facet Api [1]module to see how it is structured and where it could use improvements. Finally Apache Solr was benchmarked with different configurations to find out the most optimal ones.

4.1 Apache Solr

Version conflicts Apache Solr exists out of a couple parts. The analysis part tries to explain you all it entails. When Drupal 6 came out and became popular, there was only one version of Apache Solr available. This was version 1.4 and was not yet merged with the Lucene branch. Solr's version number was synced with Lucene following the Lucene/Solr merge, so Solr 3.1 contains Lucene 3.1. Solr 3.1 is the first release after Solr 1.4.1. All the explanation that follows will be for Solr 3.x since this is the version that is used and was aimed at during the creation of this project.

Fields There are different field types defined in the original schema.xml that used to come with the module. A field type has four types of information.

- The name of the field type
- An implementation class name
- If the field type is TextField, a description of the field analysis for the field type

[1] `http://www.drupal.org/project/facetapi`

- Field attributes

To illustrate this there is listing 3 as an example of a field type definition as it is used in the schema provided with the Apache Solr Module and also a list of all possible field types in listing 1

Class	Description
BCDIntField	Binary-coded decimal (BCD) integer. BCD is a relatively inefficient encoding that offers the benefits of quick decimal calculations and quick conversion to a string.
BCDLongField	BCD long integer
BCDStrField	BCD string
*BinaryField	Binary data
*BoolField	Contains either true or false. Values of "1", "t", or "T" in the first character are interpreted as true. Any other values in the first character are interpreted as false.
ByteField	Contains an array of bytes.
*DateField	Represents a point in time with millisecond precision.
DoubleField	Double (64-bit IEEE floating point)
ExternalFileField	Pulls values from a file on disk.
FloatField	Floating point (32-bit IEEE floating point)
IntField	Integer (32-bit signed integer)
LongField	Long integer (64-bit signed integer)
*RandomSortField	Does not contain a value. Queries that sort on this field type will return results in random order. Use a dynamic field to use this feature.
ShortField	Short integer
SortableDoubleField	The Sortable* fields provide correct numeric sorting. If you use the plain types (DoubleField, IntField, and so on) sorting will be lexicographical instead of numeric.
SortableFloatField	Numerically sorted floating point
SortableIntField	Numerically sorted integer
SortableLongField	Numerically sorted long integer

*StrField	String (UTF-8 encoded string or Unicode)
*TextField	Text, usually multiple words or tokens
*TrieDateField	Date field accessible for Lucene TrieRange processing
*TrieDoubleField	Double field accessible Lucene TrieRange processing
TrieField	If this type is used, a "type" attribute must also be specified, with a value of either: integer, long, float, double, date. Using this field is the same as using any of the Trie*Fields.
*TrieFloatField	Floating point field accessible Lucene TrieRange processing
*TrieIntField	Int field accessible Lucene TrieRange processing
*TrieLongField	Long field accessible Lucene TrieRange processing
*PointType	For spatial search: An arbitrary n-dimensional point, useful for searching sources such as blueprints or CAD drawings.
*LatLonType	Latitude/Longitude as a 2 dimensional point. Latitude is always specified first.
*GeoHashField	Representing a Geohash[2] field. The field is provided as a lat/lon pair and is internally represented as a string.
UUIDField	Universally Unique Identifier (UUID). Pass in a value of "NEW" and Solr will create a new UUID.

Listing 1: All field type definitions. Marked with a star are the ones that are used in the Apache Solr Search Integration module for Drupal

Field properties Important to know is that each of these fields that is shown in listing 1 have configurable values. Drupal uses these properties to map different dynamic fields to specific types with specific configurations. These dynamic fields are what we call fields from the Field API (Drupal 7) or from the Content Construction Kit (CCK, Drupal 6). With these modules it is possible to add different fields to content types[3]

Field Property	Description	Values
indexed	If true, the value of the field can be used in queries to retrieve matching documents	true or false

[2]Geohash is a defined standard. More on wikipedia : `http://en.wikipedia.org/wiki/Geohash`

[3]Content types are a way of defining structured data that will be inputted by users

stored	If true, the actual value of the field can be retrieved by queries	true or false
sortMissingFirst / sortMissingLast	Control the placement of documents when a sort field is not present. As of Solr 3.5, these work for all numeric fields, including Trie and date fields.	true or false
multiValued	If true, indicates that a single document might contain multiple values for this field type	true or false
positionIncrementGap	For multivalued fields, specifies a distance between multiple values, which prevents spurious phrase matches	integer
omitNorms	If true, omits the norms associated with this field (this disables length normalization and index-time boosting for the field, and saves some memory). Only full-text fields or fields that need an index-time boost need norms.	true or false
omitTermFreqAndPositions	If true, omits term frequency, positions, and payloads from postings for this field. This can be a performance boost for fields that don't require that information. It also reduces the storage space required for the index. Queries that rely on position that are issued on a field with this option will silently fail to find documents. This property defaults to true for all fields that are not text fields.	true or false
autoGeneratePhraseQueries	For text fields. If true, Solr automatically generates phrase queries for adjacent terms. If false, terms must be enclosed in double-quotes to be treated as phrases.	true or false

Listing 2: Field type properties and their respective explanation

```
<!--  A text field that uses WordDelimiterFilter to enable splitting and matching of words
      on case-change, alpha numeric boundaries, and non-alphanumeric chars,
      so that a query of "wifi" or "wi fi" could match a document containing "Wi-Fi".
      Synonyms and stopwords are customized by external files, and stemming is enabled.
      Duplicate tokens at the same position (which may result from Stemmed Synonyms or
      WordDelim parts) are removed. -->
    <fieldType name="text" class="solr.TextField" positionIncrementGap="100">
      <analyzer type="index">
        <charFilter class="solr.MappingCharFilterFactory" mapping="mapping-ISOLatin1Accent.txt"/>
        <tokenizer class="solr.WhitespaceTokenizerFactory"/>
        <!-- Case insensitive stop word removal.
          add enablePositionIncrements=true in both the index and query
          analyzers to leave a "gap" for more accurate phrase queries. -->
        <filter class="solr.StopFilterFactory"
                ignoreCase="true"
                words="stopwords.txt"
                enablePositionIncrements="true" />
        <filter class="solr.WordDelimiterFilterFactory"
                protected="protwords.txt"
                generateWordParts="1"
                generateNumberParts="1"
                catenateWords="1"
                catenateNumbers="1"
                catenateAll="0"
                splitOnCaseChange="1"
                preserveOriginal="1" />
        <filter class="solr.LengthFilterFactory" min="2" max="100" />
        <filter class="solr.LowerCaseFilterFactory"/>
        <filter class="solr.SnowballPorterFilterFactory" language="English" protected="protwords.txt"/>
        <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
      </analyzer>
      <analyzer type="query">
        <!-- Similar configuration, but then at query time, see real schema.xml for full example -->
      </analyzer>
    </fieldType>
```

Listing 3: Example of a text field type definition

Analyzers, Filters and Tokenizers used by Apache Solr Search Integration In the snippet of the text field type definition there there are some unexplained entries. Filters, tokenizers and analyzers are used to process a value submitted by the application and to be saved properly into Solr so we optimize the content for faster search. In chapter 3 these concepts were shortly explained and what follows will be a list of analyzers, tokenizers and fil-

ters used in the Drupal module. Please note that these concepts can be used during query time and also at the index time. A complete list of the supported classes can be found at `http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters`.

WhitespaceTokenizerFactory Simple tokenizer that splits the text stream on whitespace and returns sequences of non-whitespace characters as tokens. Note that any punctuation will be included in the tokenization. Does not ship with any arguments.

```
1 <tokenizer class="solr.WhitespaceTokenizerFactory"/>
```

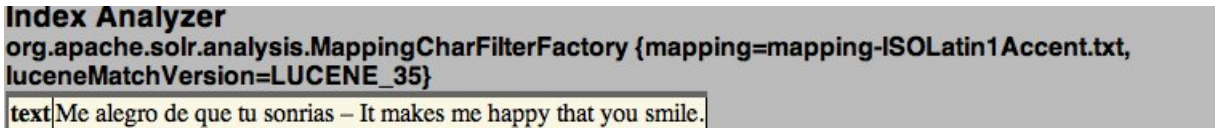
KeywordTokenizerFactory Treats the entire field as a single token, regardless of its content.

```
1 <tokenizer class="solr.KeywordTokenizerFactory"/>
```

MappingCharFilterFactory Maps Special characters to their plain equivalent

```
1 <charFilter class="solr.MappingCharFilterFactory" mapping="mapping-ISOLatin1Accent.txt"/>
```

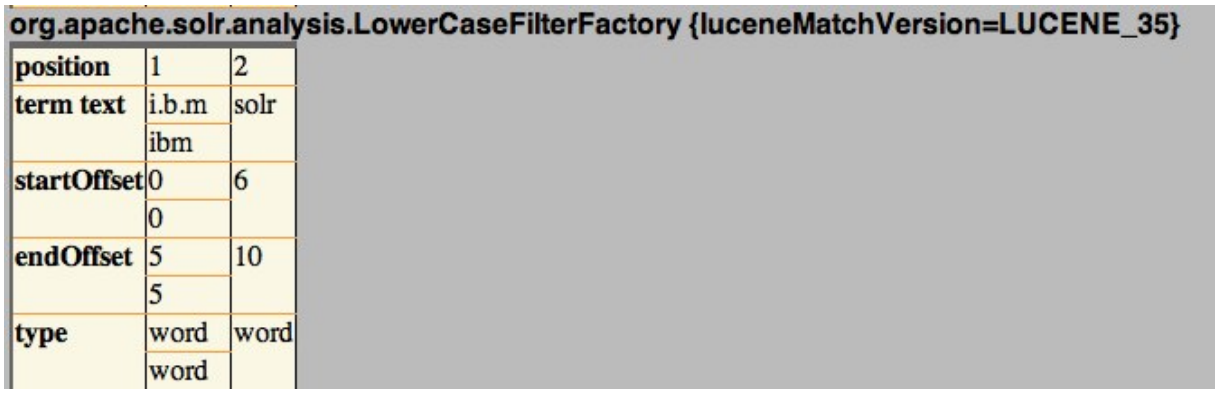
Example (index time): Me alegro de que tú sonrías – It makes me happy that you smile.



LowerCaseFilterFactory Lowercases the letters in each token. Leaves non-letter tokens alone.

```
1 <filter class="solr.LowerCaseFilterFactory"/>
```

Example (index time): ”I.B.M.”, ”Solr” ==¿ ”i.b.m.”, ”solr”.



StopFilterFactory Discards common words that are listed in the stopwords.txt file. This file is shipped in the module. Examples of these words are ”an, and, are, ...”. And as visible it comes with some configuration options such as ignoring the case of the text and the file from where to read the stopwords from. This should be a path starting from the conf folder. When enablePositionIncrements is true a token is stopped (discarded) and the position of the following token is incremented. This is useful if you want to know if certain words were discarded by looking at the token position.

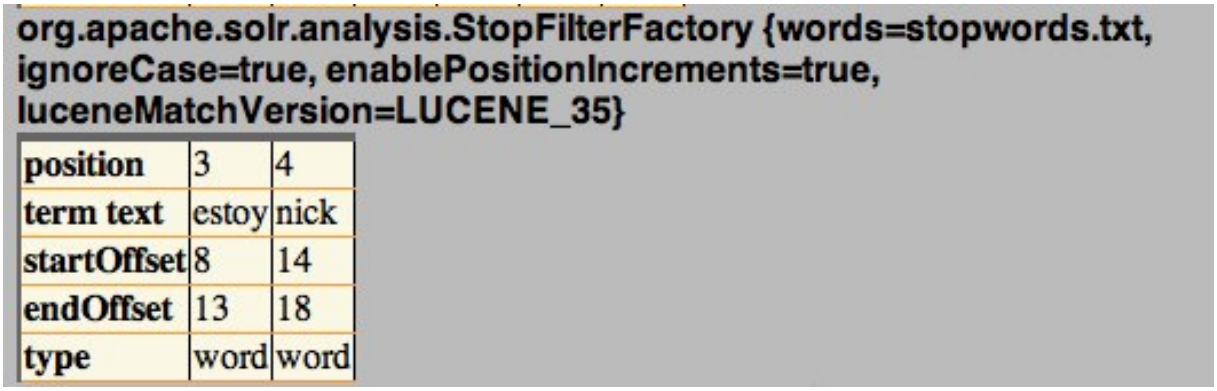
```
<filter class="solr.StopFilterFactory"
    ignoreCase="true"
    words="stopwords.txt"
    enablePositionIncrements="true"/>
```

```
# a couple of test stopwords to test that the words are really being
# configured from this file:
hola
si

# Standard english stop words taken from Lucene's StopAnalyzer
a
an
and
...
```

Listing 4: Example of the stopwords file

Example (index time): Si Hola estoy nick a on



WordDelimiterFilterFactory Delimits words based on parts of words. Was originally defined for the use in english based texts. It follows the following strict order but allows a number of configurations to happen. The original filter has more options but below are only the ones used in the Apache Solr schema.xml

- protected (optional) The pathname of a file that contains a list of protected words that should be passed though without splitting. In the case of Drupal these are predefined as some html entities.
- generateWordParts splits words at delimiters.
- generateNumberParts splits numeric strings at delimiters
- catenateWords maximal runs of word parts will be joined: "hot-spot-sensor's" -¿ "hotspot-sensor"
- catenateNumbers maximal runs of number parts will be joined: 1947-32" -¿ "194732"
- catenateAll Set at 0, runs of word and number parts will not be joined: "Zap-Master-9000" -¿ "Zap Master 9000"
- splitOnCaseChange words are not split on camel-case changes:"BugBlaster-XL" -¿ "Bug-Blaster", "XL"
- preserveOriginal the original token is preserved: "Zap-Master-9000" -¿ "Zap-Master-9000", "Zap", "Master", "9000"

```
1 <filter class="solr.WordDelimiterFilterFactory"
2     protected="protwords.txt"
3     generateWordParts="1"
4     generateNumberParts="1"
```

```
5       catenateWords="1"
6       catenateNumbers="1"
7       catenateAll="0"
8       splitOnCaseChange="1"
9       preserveOriginal="1"/>
```

Example text (index time): Zap-Master-9000 9000-12 BugBlaster-XL hot-spot-sensor's

org.apache.solr.analysis.WordDelimiterFactory {preserveOriginal=1, protected=protwords.txt, splitOnCaseChange=1, generateNumberParts=1, catenateWords=1, luceneMatchVersion=LUCENE_35, generateWordParts=1, catenateAll=0, catenateNumbers=1}

position	1	2	3	4	5	6	7	8	9	10	11
term text	Zap-Master-9000	Master	9000	9000-12	12	BugBlaster-XL	Blaster	XL	hot-spot-sensor's	spot	sensor
	Zap	ZapMaster		9000	900012	Bug		BugBlasterXL	hot		hotspotsensor
startOffset	0	4	11	16	21	24	27	35	38	42	47
	0	0		16	16	24		24	38		38
endOffset	15	10	15	23	23	37	34	37	55	46	53
	3	10		20	23	27		37	41		53
type	word	word	word	word	word	word	word	word	word	word	word
	word	word		word	word	word		word	word		word

LengthFilterFactory Words smaller than 2 chars and bigger than 100 will be discarded. This is useful to speed up the query process because a blog posting from large scale solr mentions that a query will be exponentially grow in query time when small words are used (Add reference!!!)

```
1   <filter class="solr.LengthFilterFactory" min="2" max="100" />
```

Example Text (index time): I am a dog a b c 123 iamawordoveronehundredcharactersiamawordoveronehundredcharactersiamawordoveronehundredcharactersiamawordoveronehundredcharactersiamawordoveronehundredcharactersiamawordoveronehundredcharacters

org.apache.solr.analysis.LengthFilterFactory {min=2, max=100, luceneMatchVersion=LUCENE_35}

position	1	2	3
term text	am	dog	123
startOffset	2	7	17
endOffset	4	10	20
type	word	word	word

Words smaller than 2 and bigger than 100 were discarded

SynonymFilterFactory This is quite a special one that is only executed during query time. Meaning that words will not be processed as synonyms in index time. If a user would type color it could also check the index for texts with the word "colour". Same is valid for the more concrete example "GB,gib,gigabyte,gigabytes"

```
1   <filter class="solr.SynonymFilterFactory" synonyms="synonyms.txt" ignoreCase="true" expand="true"/>
```

Example Text (query time): colour test

org.apache.solr.analysis.SynonymFilterFactory

position	1	2
term text	color	test
	colour	
type	SYNONYM	word
	SYNONYM	
startOffset	0	7
	0	
endOffset	6	11
	6	

TrimFilterFactory This filter trims leading and/or trailing whitespace from tokens. In Drupal usecase this is used for sortable text such as names or labels. The big difference with most other filters is that this filter does not break words on spaces.

```
1 <filter class="solr.TrimFilterFactory" />
```

Example Text (query time): Nick Veenhof

org.apache.solr.analysis.TrimFilterFactory {luceneMatchVersion=LUCENE_35}

position	1
term text	nick veenhof
startOffset	0
endOffset	12

EdgeNGramFilterFactory This filter generates edge n-gram tokens of sizes within the given range. In the module it was configured to return 2-gram tokens till 25-gram tokens. Especially useful for matching against queries with results. [4]

```
1 <filter class="solr.EdgeNGramFilterFactory" minGramSize="2" maxGramSize="25" />
```

Example Text (index time) : I am a dog with a longbigtext

org.apache.solr.analysis.EdgeNGramFilterFactory {maxGramSize=25, minGramSize=2, luceneMatchVersion=LUCENE_35}

position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
term text	i	i a	i am	i am	i am a	i am a	i am a d	i am a do	i am a dog	i am a dog	i am a dog w	i am a dog wi	i am a dog wit	i am a dog with	i am a dog with	i am a dog with a	i am a dog with a	i am a dog with a l	i am a dog with a lo	i am a dog with a lon	i am a dog with a long	i am a dog with a longb	i am a dog with a longbi	i am a dog with a longbig
startOffset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
endOffset	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

[4] http://www.lucidimagination.com/blog/2009/09/08/auto-suggest-from-popular-queries-using-edgengrams/

SnowballPorterFilterFactory Snowball is a software package that generates pattern-based word stemmers. It works efficiently and fast and one can configure the language that is preferred. Apache Solr comes with a whole range of languages. English is very well supported but also Catalan and Spanish. A list of all the languages can be found in the documentation of Apache Solr or in the Snowball website [5]. Also interesting to note is that there is a file called protwords.txt (Protected words) where you can define strings that won't be stemmed.

```
1 <filter class="solr.SnowballPorterFilterFactory" language="English" protected="protwords.txt"/>
```

Example Text (index time) : football footballing

org.apache.solr.analysis.SnowballPorterFilterFactory {protected=protwords.txt, language=English, luceneMatchVersion=LUCENE_35}

position	1	2
term text	footbal	footbal
keyword	false	false
startOffset	0	8
endOffset	7	19
type	word	word

RemoveDuplicatesTokenFilterFactory Removes duplicates from the query or the index value.

```
1 <filter class="solr.RemoveDuplicatesTokenFilterFactory"/>
```

Example : Nick Nick Nick

Dynamic Fields in Solr used by Drupal As explained, using a combination of Field types and field properties a schema can create lots of dynamic configurations for softwares that interact with Solr. In the case of Drupal the ApacheSolr Drupal module does not know in advance how the schema should look like because all Drupal sites are differently configured using different content types. The module should be able to cope with most of the use cases that site administrators come up with. If a field name is not found while submitting a new document, dynamicFields will be used if the name matches any of the patterns. Note that there are restrictions namely that the glob-like pattern in the name attribute must have a "*" only at the start of the end of the field definition. For example, name="_i" will match any field ending in _i (like myid_i, z_i). Longer patterns will be matched first and if equal size patterns both match, the first appearing in the schema will be used.

[5] http://snowball.tartarus.org/

Before starting this work not all of these dynamic fields were provided to the site administrators but with time a list was compiled to meet 99% of the use cases. See schema.xml in the project files for the complete list. A small snippet of some of these dynamic fields is included below. The 1st letter indicates the data type and the last letter is 's' for single valued, 'm' for multi-valued.

```
1  <fields>
2    <!-- We use long for integer since 64 bit ints are now common in PHP. -->
3    <dynamicField name="is_*"  type="long"    indexed="true"  stored="true" multiValued="false"/>
4    <dynamicField name="im_*"  type="long"    indexed="true"  stored="true" multiValued="true"/>
5    <!-- List of floats can be saved in a regular float field -->
6    <dynamicField name="fs_*"  type="float"   indexed="true"  stored="true" multiValued="false"/>
7    <dynamicField name="fm_*"  type="float"   indexed="true"  stored="true" multiValued="true"/>
8    <!-- List of doubles can be saved in a regular double field -->
9    <dynamicField name="ps_*"  type="double"   indexed="true"  stored="true" multiValued="false"/>
10   <dynamicField name="pm_*"  type="double"   indexed="true"  stored="true" multiValued="true"/>
11   <!-- List of booleans can be saved in a regular boolean field -->
12   <dynamicField name="bm_*"  type="boolean" indexed="true"  stored="true" multiValued="true"/>
13   <dynamicField name="bs_*"  type="boolean" indexed="true"  stored="true" multiValued="false"/>
14   <!-- Regular text (without processing) can be stored in a string field-->
15   <dynamicField name="ss_*"  type="string"  indexed="true"  stored="true" multiValued="false"/>
16   <dynamicField name="sm_*"  type="string"  indexed="true"  stored="true" multiValued="true"/>
17   <!-- Normal text fields are for full text - the relevance of a match depends on the length of the text -->
18   <dynamicField name="ts_*"  type="text"    indexed="true"  stored="true" multiValued="false" termVectors="true"/>
19   <dynamicField name="tm_*"  type="text"    indexed="true"  stored="true" multiValued="true" termVectors="true"/>
20
21 ...
22
23   <!-- The following causes solr to ignore any fields that don't already match an existing
24        field name or dynamic field, rather than reporting them as an error.
25        Alternately, change the type="ignored" to some other type e.g. "text" if you want
26        unknown fields indexed and/or stored by default -->
27   <dynamicField name="*" type="ignored" multiValued="true" />
28
29 </fields>
```

Listing 5: Example of some dynamic field type definitions

In the implementation chapter it will be explained how these dynamic fields are used to create new fields in solr using Drupal.

4.2 Standard Drupal Search

By default Drupal already ships with a search module that leverages Mysql to its far extent in order to create a search experience that works quite well in smaller scale websites.

In Drupal there is a concept called "cron". These are actions that are executed per a set amount of time, for example 30 minutes. Every 30 minutes the designated search actions will index a little set of the selected content, for example 100 pages. This will run until there is no more content to index. Naturally content will change and will need to be re-indexed. This concept is fairly basic and is also the one used for the Apache Solr module. However, I'd like to point out that the Search module that is shipped with Drupal differs greatly from the Apache Solr module.

Advantages The standard Drupal search module certainly has its advantages. There is, to start with, no extra server/service necessary and it does ship with Drupal core. The basic module also has support for basic text transformations, such as recognition of singular and plural words. It transforms special characters to basic text characters (Similar to the MappingCharFilterFactory in Apache Solr) and it scores items based on their tag where they are embedded in. Examples are H1, H2 and P tags.

Disadvantages However, it has a hard time handling a big data set. MySQL was not built to be a search engine. Mysql also has its limitations when building a full text search on top of its stack. Drupal also has to comply with the SQL standards so engine specific optimizations cannot be utilized. [6]. This leaves the SQL solution with a very restricted set of operators and inherently slow and not scalable in the long haul.

Conclusion Drupal SQL search An SQL backend does well in serving a full text search application as long as the number of indexed items stay stable and preferably ¡ 10000 items. [7]

4.3 Apachesolr Search Integration Drupal Module

The module found its origin around the end of 2007, at the time of Drupal 5. It's first author was Robert Douglass and lots of other people followed his lead in this initiative. Fast forward

[6] `http://dev.mysql.com/doc/refman/4.1/en/fulltext-restrictions.html`

[7] This number is an estimation, depending on the SQL database application and server configuration this can vary greatly

and at the moment of writing a Drupal 6 and 7 version exist. When this work started the Drupal 7 version was basically a port of the Drupal 6 version and needed lots of improvements. Acquia sponsors development of this module to ensure continuity and support.

State of the UI as of September 2011 What is shown below is a snapshot of how the module looked in the backend as of September 2011. There are markers that indicate problem area. Do take into account that this does not show you any comments made on the internals of the module.

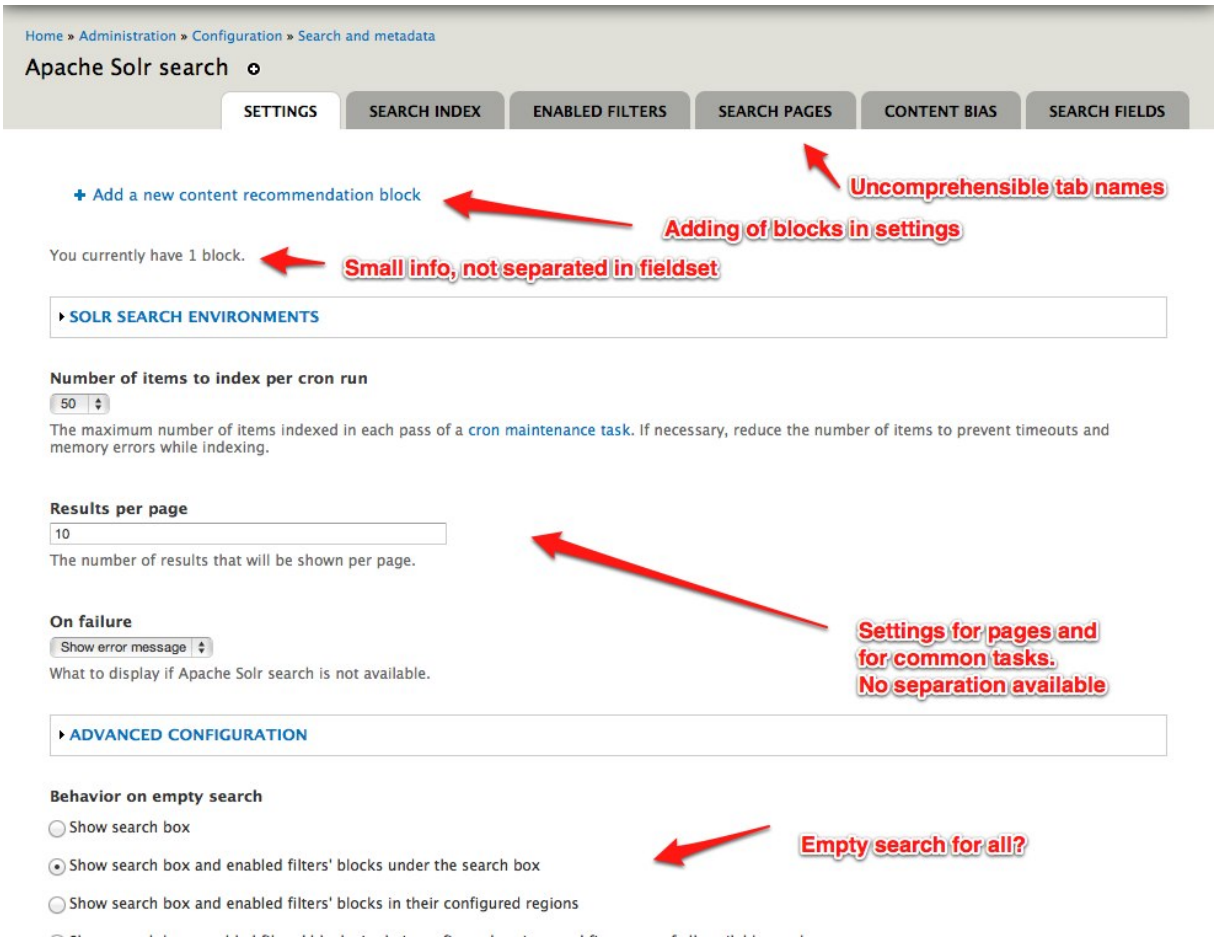


Figure 4.1: UI settings backend, September 2011

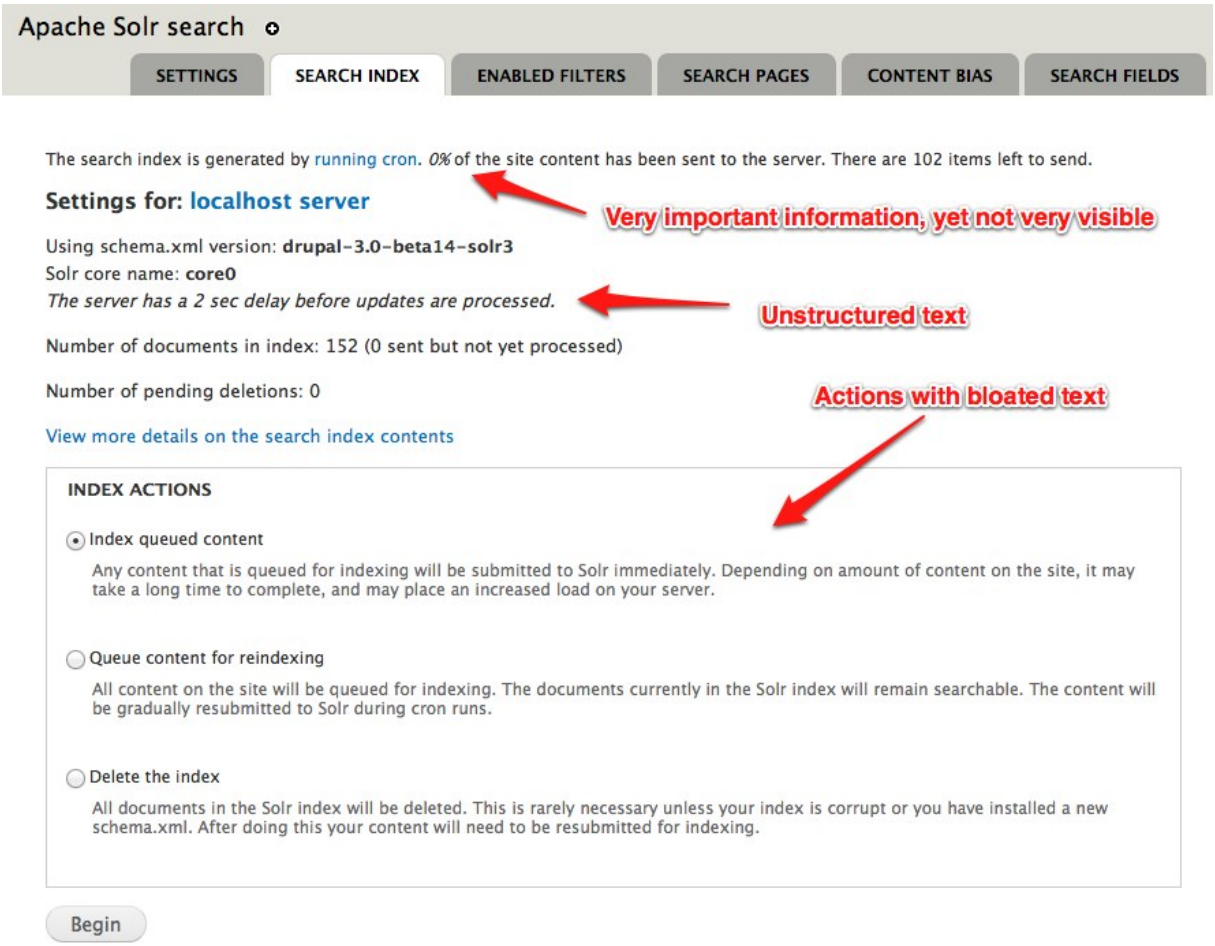


Figure 4.2: UI index report backend, September 2011

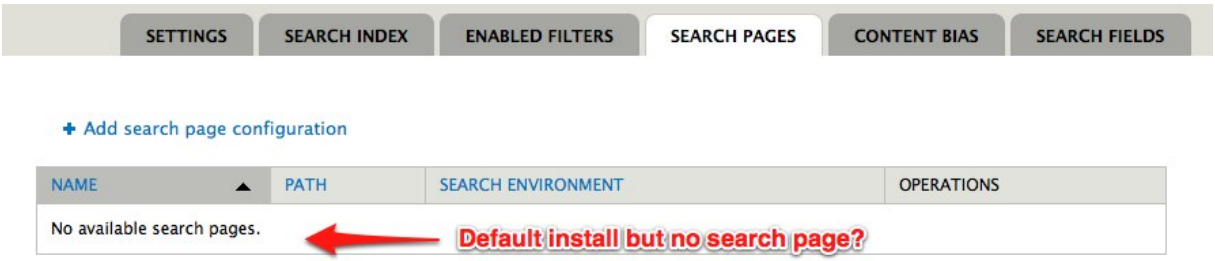


Figure 4.3: UI search pages backend, September 2011

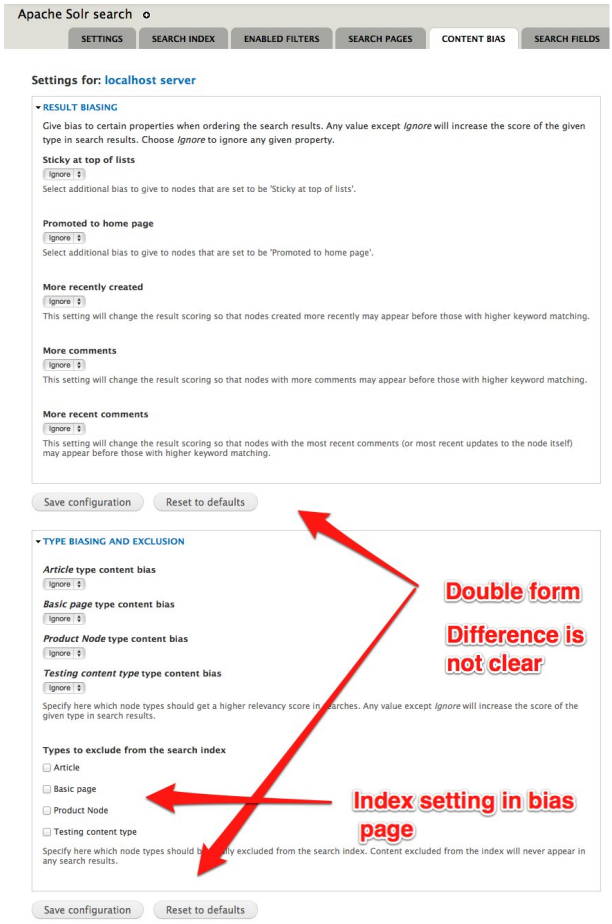


Figure 4.4: UI for result and index biasing backend, September 2011

Architectural challenges Drupal and the module were never intended to be built by architects but by people who solve problems, real world problems. Many people worked together to create a cohesive project that is very stable but might not be in agreement with that is taught in classes such as Object Oriented programming, other theories and best practices. A class diagram would be a faulty way to show you the beauty this module has to offer its users since there were hardly object oriented concepts applied to this module that were worthy enough to show a class diagram from. Together with Acquia we've set up a list of minimal achievements we should reach for the Apache Solr module by the end of February.

Improvements

- UI refactoring to make a better experience [8]

[8] `http://drupal.org/node/1292364`

- Support the indexing of multiple entities natively so the module would have an API to index users / terms / ... easily [9]
- Global functions should be context driven.[10]
- Get rid of dependencies in theme layer from core search [11]
- Hooks node_type, taxonomy and user knocks out our database server [12]
- Improve file listing and access control
- More like this blocks should get a delete button [13]
- De-duplicate core and custom search in order to obtain clarity in the code [14]
- Add 1 custom search block with generic render function for custom development
- The numeric field id should not be used for Solr index field names [15]
- Query type should be adjusted in order to allow different widgets in facetapi [16]
- Change php static to drupal_static [17]
- Non-current/valid Node Types are not excluded from index [18]
- add retain current filters checkbox to custom search page [19]
- add "retain-filters" param when in facet browsing mode [20]
- Handle 1 placeholder in a custom search page path with makes the taxonomy sub-module obsolete [21]

[9] `http://drupal.org/node/1292364`
[10] `http://drupal.org/node/1292364`
[11] Related to `http://drupal.org/node/1314406` (de-duplication)
[12] `http://drupal.org/node/592522`
[13] `http://drupal.org/node/1271964`
[14] `http://drupal.org/node/1314406`
[15] `http://drupal.org/node/1161538`
[16] `http://drupal.org/node/1161444`
[17] `http://drupal.org/node/1334216`
[18] `http://drupal.org/node/1000532`
[19] `http://drupal.org/node/1246422`
[20] `http://drupal.org/node/1116792`
[21] `http://drupal.org/node/1294846`

- Create tests for the module [22]
- bundle' is not a required field, but apachesolr treats it as such (Evaluate required fields in schema, make non-required if possible) [23]
- Improve Date faceting/date query type (combined with facetapi) [24]
- Facets are currently not linked to the appropriate search page
- Add clone operation for search environments [25]
- Backport Apache Solr Module to Drupal 6

4.4 Facetapi Drupal module

The Facet API module allows site builders to easily create and manage faceted search interfaces. In addition to the UI components that come out of the box, themers and module developers can build their own widgets that can optionally be contributed back to Drupal.org. Facet API works with the core Search, Search API, and Apache Solr Search Integration modules (including Acquia Search) meaning that code and configuration can be reused as-is with the most popular search solutions available to Drupal. It was created by Chris Pliakas and Peter Wolanin specifically for the Drupal 7 version of any search tool. Acquia sponsors development of this module to ensure continuity and support.

State of the UI as of September 2011 What is shown below is a snapshot of how the module looked in the backend and frontend as of September 2011. There are markers that indicate problem area. Do take into account that this does not show you any comments made on the internals of the module.

Screenshots of the implemented part of facetapi (Drupal 7)

[22] `http://drupal.org/node/989398`
[23] `http://drupal.org/node/1279164`
[24] `http://drupal.org/node/1201534`
[25] `http://drupal.org/node/1292328`

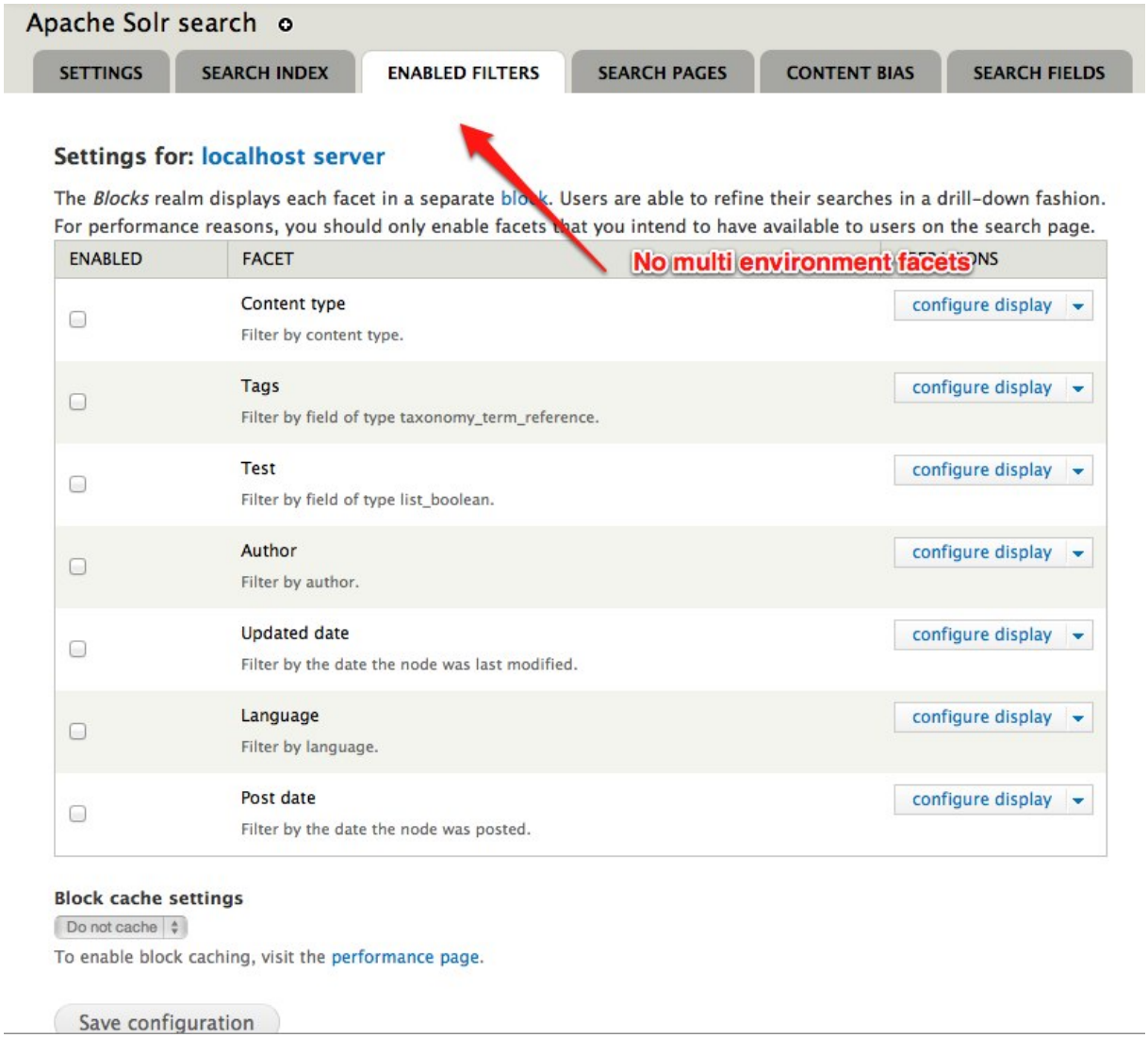


Figure 4.5: Apachesolr Facetapi Integration UI as of September 2011

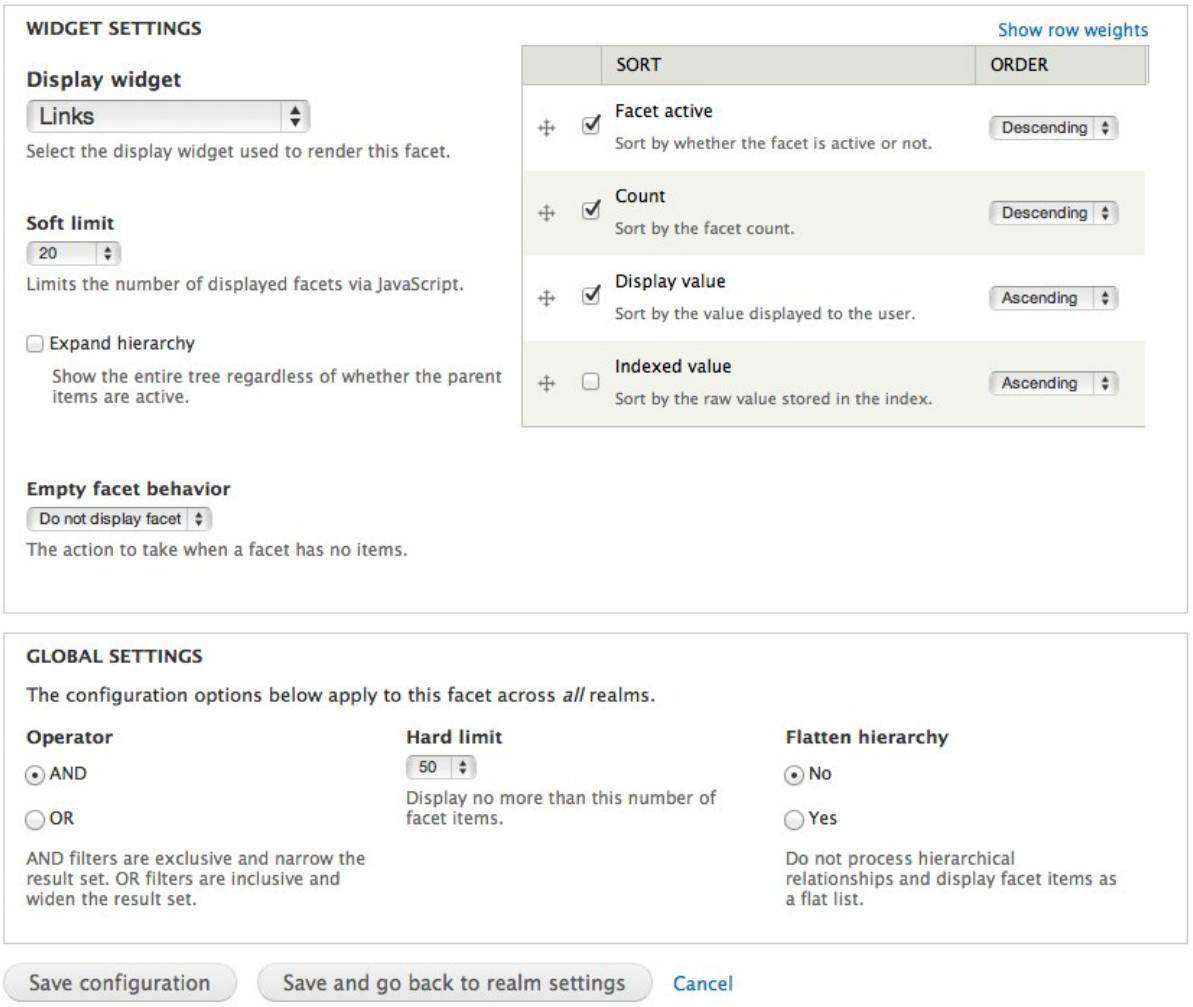


Figure 4.6: Apachesolr Facetapi Integration UI of 1 facet as of September 2011

Architecture As part of the analysis a class diagram was made from the Facet Api code to get a better understanding of the internals. An issue [26]was raised in the Facet Api issue queue on `drupal.org` for those that prefer to read up in detail.

[26] `http://drupal.org/node/1321136`

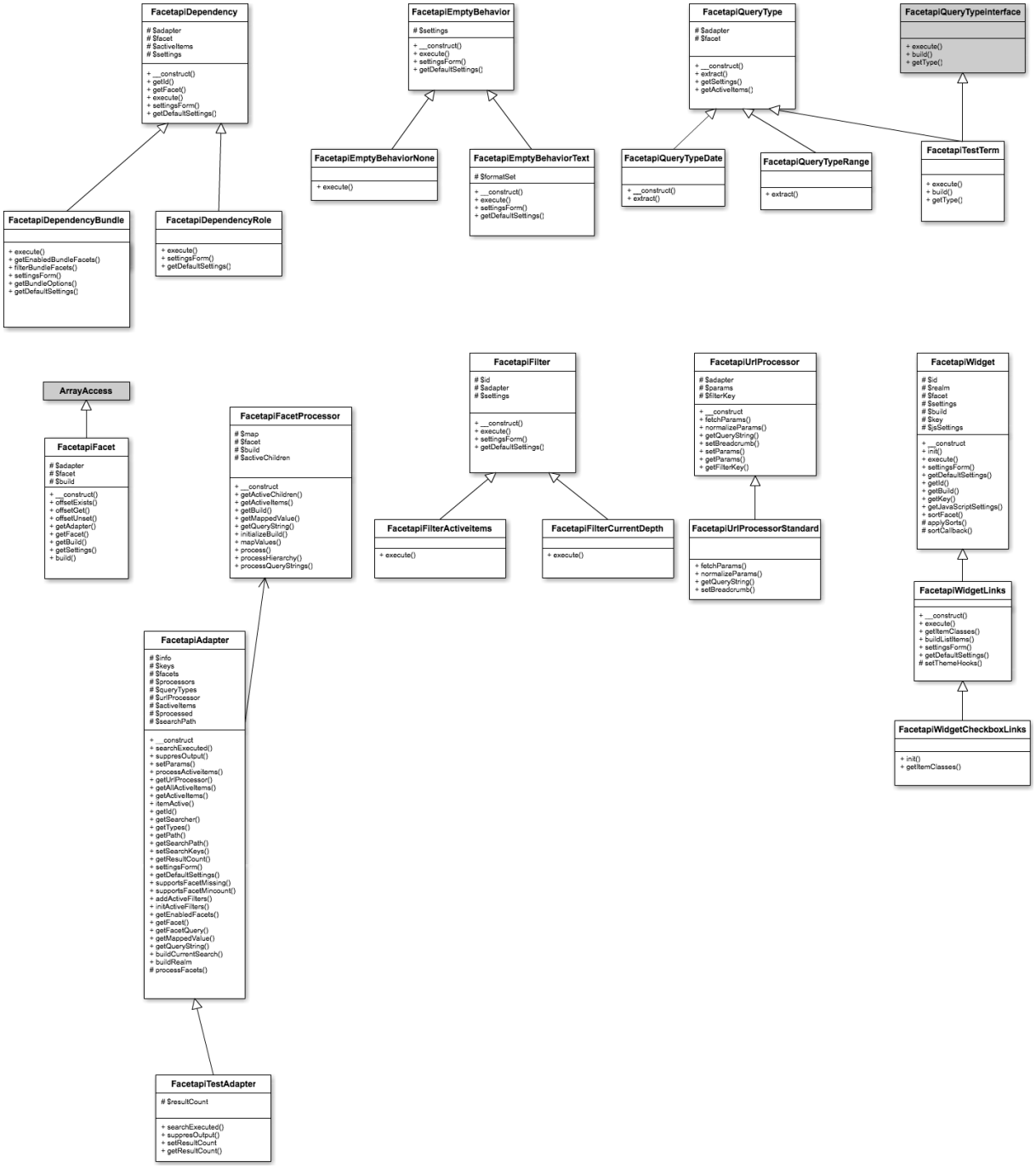


Figure 4.7: Extended information about the classes in FacetAPI, September 2011

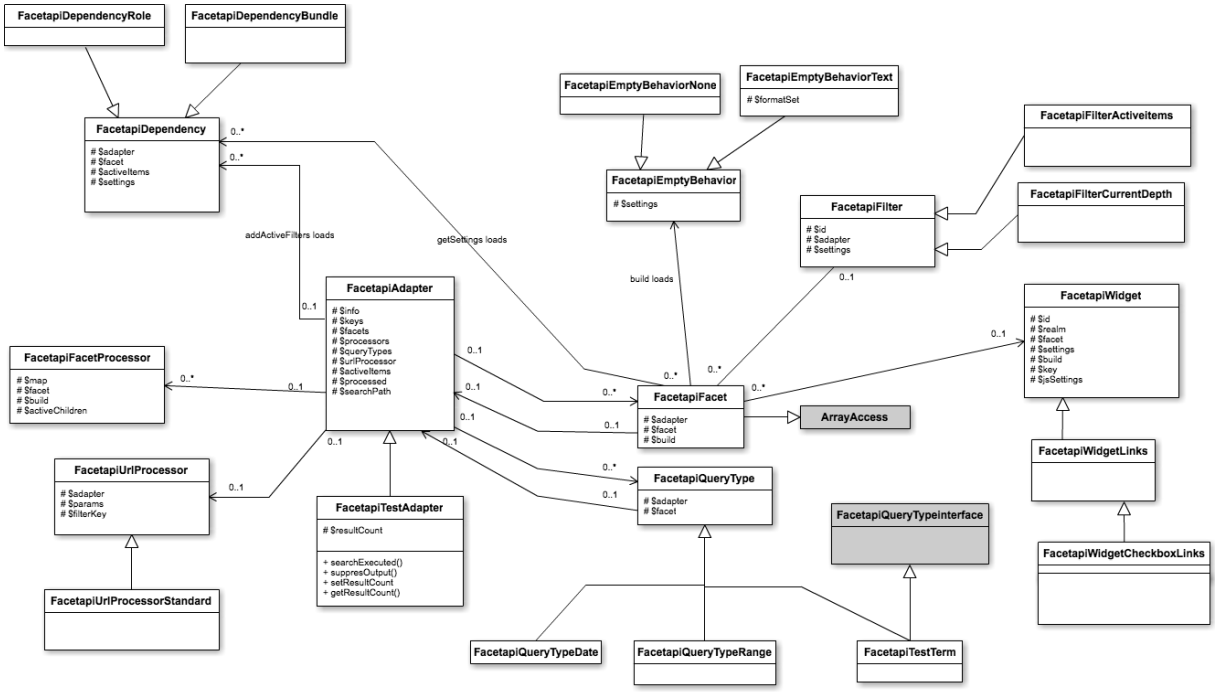


Figure 4.8: Class Diagram of FacetAPI, September 2011

- There are a number of loop-references between the adapter and it's relations. This can possibly be avoided by thinking the architecture through (see the references that have 2 lines to each other)
- The variable facet in FacetapiFacet might be a bit too un-descriptive and it looks like it could be renamed to "settings" or "facet_settings"
- Doxygen documentation with loads of diagrams and easy to read documentation was generated from this. In addition to the attached images it should make the facetapi module easier to understand. The documentation can be found on `http://facetapi.nickveenhof.be`

Improvements

- Modify "query type" key in facet definition to accept an array [27]
- Make the current search block more configurable [28]

[27] `http://drupal.org/node/1161434`
[28] `http://drupal.org/node/593658`

- Complete configuration import functionality [29]
- widget.inc change id/class to not reflect the field_id but a generic one for multisite (line 106) + apachesolr.module line 1860 to remove the id (integer) assumptions
- Backport Facet Api to Drupal 6

4.5 Acquia Search for Drupal 6 and 7

Quote from Dries' blog : "Acquia Search is a hosted search service based on the Software as a Service (SaaS) model. The way it works is that Drupal sites push their content to the search servers hosted by Acquia. We index the content, builds an index, and handle search queries. We provide the search results, facets, and content recommendations to your Drupal site over the network." [30]

As the reader of this paper would have guessed, Acquia Search is built using the Open Source Lucene and Solr distributions from the Apache project. Another quote from Dries' website : "Many organizations simply lack the Java expertise to deploy, manage and scale Java applications or their hosting environment may not accommodate it. Because Acquia Search is a hosted service, it takes away the burden of installation, configuration, and operational duties to keep the software fast, secure and up-to-date."

[29] `http://drupal.org/node/1147564`

[30] `http://buytaert.net/acquia-search-benefits-for-site-administrators`

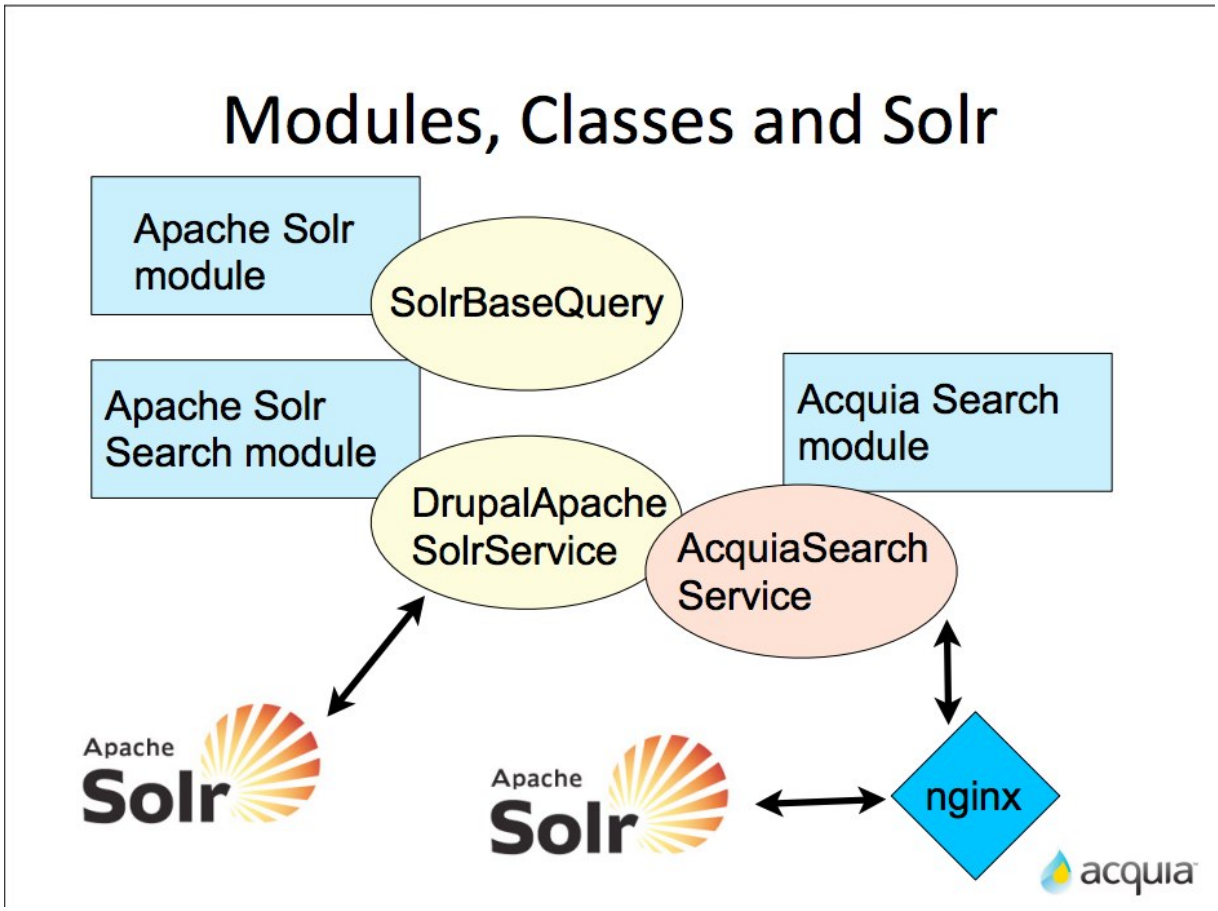


Figure 4.9: Overview of the classes and services used for Acquia Search at the website's end.

Architecture Even though the image was not build as a real class diagram it should be clear that there are 2 classes in the Apache Solr module that are pictured here (yellow). The only important one to cover here is the DrupalApacheSolrService. This class makes it possible to connect to an arbitrary Solr server. When the Acquia Search module is enabled on any website the AcquiaSearchService class extends the DrupalApacheSolrService class and adds the authentication information to all the requests.

The next figure will explain the server side handling of the requests that are being sent to Solr.

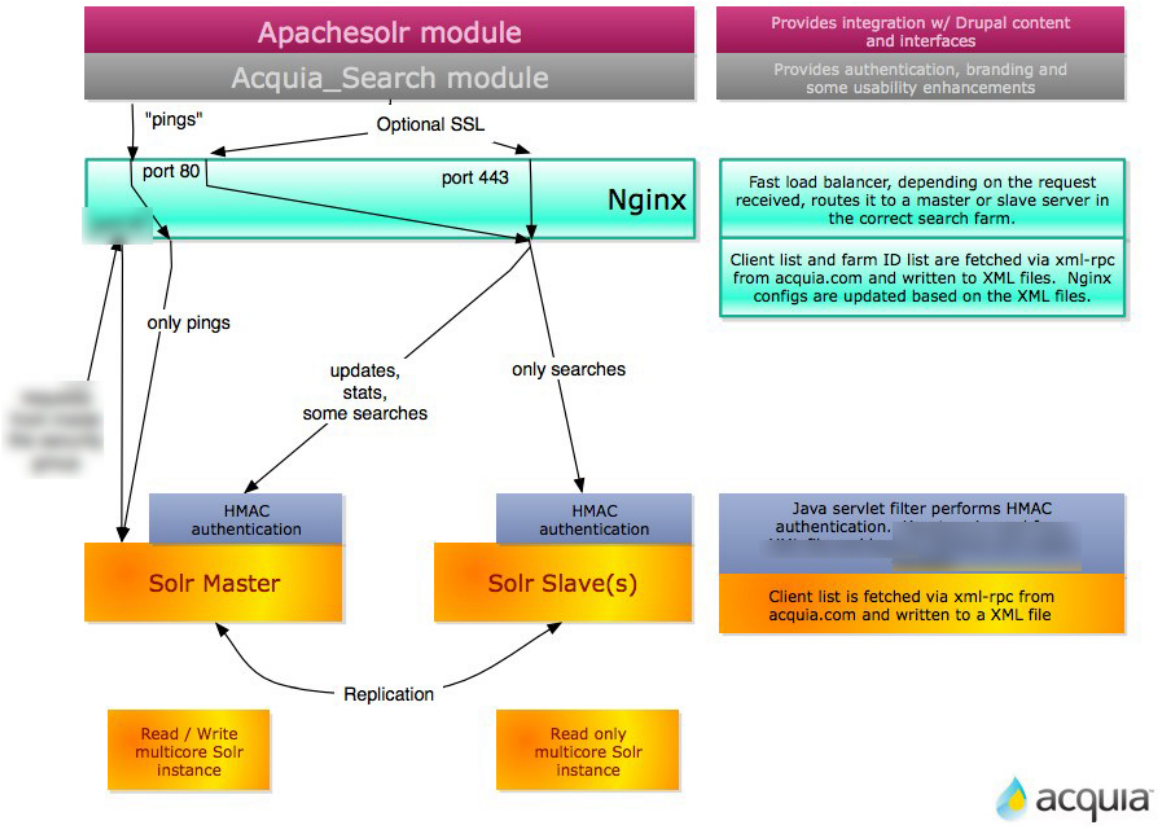


Figure 4.10: Server Side view of Acquia Search. Certain information has been blurred for confidentiality

State of Acquia Search When the internship started at Acquia there were some blanks left to be filled regarding Acquia Search. Since Solr 3.4 (now 3.5) came out it was only a logical step to convert the Acquia Search service to this newer version of Apache Solr. The version that is currently used is Apache Solr 1.4 and is still one of the defacto standards when deploying Solr.

Upgrading does not happen overnight without any effort. There are many clients running their active sites from Solr 1.4 and it is not guaranteed that all of these will work perfectly for Solr 3.5. Performance factors are also a key role during this migration.

As reviewed by the architecture it also needs a confirmed upgrade path for the HMAC authentication, which was written specifically for Solr 1.4 as a Java Servlet.

HMAC authentication filter The way it works is that the data is protected during the transport over the web by SSL and to authenticate to the search servers at Acquia an HMAC authentication layer is used. This means that the data is encrypted so no man in the middle

attack can be exploited. Acquia knows that you have sent the request and will verify, using this HMA authentication, if the data that was sent was not modified.

Get information about HMAC here : http://en.wikipedia.org/wiki/HMAC

http://tools.ietf.org/html/rfc2104

Chapter 5

Implementation

I started my internship September 22nd in Boston. While being on-site and I learned how Acquia manages processes and works together with the community in order to reach business goals. Also watching them work with a lot of remote employees was already a very valuable lesson. More about this experience can be read in the article [9] I wrote about this. In addition to this I have helped with the creation of some new modules such as Facet Slider [12] and Apachesolr sort UI [13].

As for addressing the public on this subject, I have recently given a presentation 'Drupal Search' [6] explaining to more than 60 attendees what was done with the module and where it was heading to in Belgium. Lots of open communication [10] has happened within the community in the Apache Solr Issue Queue. [11] In total I have given 4 presentations with a combined total of more then 200 attendants. Since my involvement with the project there are about 2500 websites using the Drupal 7 version of the module and about 10000 users in total using the module for Drupal 6 and Drupal 7 combined.

I've contributed several blog posts about this topic and this internship

- Presentation about Drupal Search for the DUG group in November 2011. [11]
- Simple guide to install Apache Solr 3.x for Drupal 7 on a unix machine [7]
- Adding a custom plugin to the Apache Solr Project [8]
- A Story of an intern at Acquia [9]

Objectives 1, 2 and 3 are in progress and are nearing its completion status. Objective 4, updating the Acquia Search service to the latest stable Apache Solr version, made great progress

and is currently being tested by a client of Acquia that required this change. Every day there is a daily call with Peter Wolanin to keep the daily objectives clear and to clear out any issues that could block progress.

http://drupal.org/node/1323650

5.1 Apachesolr module for Drupal 6 version 6.x-3.x

5.2 Apachesolr module for Drupal 7 version 7.x-1.0

5.3 Functional requirements

5.3.1 Search Environments

5.3.2 Search pages

5.3.3 Query Object

5.3.4 Apaches Solr Document

5.3.5 Entity layer

5.4 Non-Functional Requirements

5.4.1 User interface

5.4.2 Usability

5.4.3 Performance

5.4.4 Security

5.4.5 Legal

Chapter 6

Related Work

6.1 Search Appliances

6.1.1 Elastic Cloud

6.1.2 Sphinx

6.1.3 Some Other

6.2 Drupal Search Solutions

6.2.1 Search API

6.2.2 Drupal Lucene API

6.2.3 Google Search Appliance

Chapter 7

Conclusions

7.1 Overview of work

7.2 Reflection on Apache Solr

7.3 Reflection on Drupal 6 and Drupal 7 in regards to search integration

7.4 Future Work

7.4.1 Apache Solr Search Integration

7.4.2 Acquia Search

Chapter 8

Acknowledgements

Bibliography

[1] Apache Solr 3 Enterprise Search Server RAW Book & eBook — Packt Publishing Technical & IT Book and eBook Store
http://www.packtpub.com/apache-solr-3-enterprise-search-server/book

[2] Something referencable

[3] *http://people.apache.org/ hossman/apachecon2009us/apache-solr-out-of-the-box.pdf people.apache.org/ hossman/apachecon2009us/apache-solr-out-of-the-box.pdf*

[4] [solr-user] Limit number of docs that can be indexed (security) - Search by Lucid Imagination
http://www.lucidimagination.com/search/document/23d645855e8417a1/limit_number_of_docs_that_c

[5] *https://acquia.com/sites/default/files/blog/DCChicago2011SolrChopsv3.pdf*

[6] FieldCollapsing Solr Wiki
http://wiki.apache.org/solr/FieldCollapsing

[7] Issues for Apache Solr Search Integration — drupal.org
http://drupal.org/project/issues/apachesolr?text=&status=Open&priorities=All&categories=All&ve

[8] CoreAdmin Solr Wiki
http://wiki.apache.org/solr/CoreAdmin

[9] Double ellipses on search snippets. — drupal.org
http://drupal.org/node/1264786

[10] Dynamic queries — drupal.org
http://drupal.org/node/310075

[11] [#SOLR-232] let Solr set request headers (for logging) - ASF JIRA
https://issues.apache.org/jira/browse/SOLR-232

[12] [#SOLR-2452] rewrite solr build system - ASF JIRA
https://issues.apache.org/jira/browse/SOLR-2452

[13] Compiling with Ant – genoviz
http://sourceforge.net/apps/trac/genoviz/wiki/Compiling with Ant

[14] Update war file for report reader - Attias
http://attias.myftp.org/attias/index.php/Update_war_file_for_report_reader

[15] solr at master from distilledmedia/munin-plugins GitHub
https://github.com/distilledmedia/munin-plugins/tree/master/solr

[16] Date-boosting Solr / Drupal search results — Metal Toad Media
http://www.metaltoad.com/blog/date-boosting-solr-drupal-search-results

[17] Major Solr 4 Highlights — Javalobby
http://java.dzone.com/videos/major-solr-4-highlights

[18] Solr Black Belt Preconference
http://www.slideshare.net/erikhatcher/solrblackbeltpreconference

[19] Some tips for Solr tuning - Vizrt forum
http://forum.vizrt.com/showthread.php?t=5177

[20] Getting started faster with LucidWorks for Solr
http://www.slideshare.net/LucidImagination/improving-findability

[21] Spatial Indexes: Solr — Derick Rethans
http://derickrethans.nl/spatial-indexes-solr.html

[22] Using Apache Access Logs with JMeter
http://minaret.biz/tips/jmeter.html

[23] Jmeter used to playback Apache access logs to generate live-like server load — artur.ejsmont.org
http://artur.ejsmont.org/blog/content/jmeter-used-to-playback-apache-access-logs-to-generate-live-like-server-load

[24] Drupal Patching, Committing, and Squashing with Git — RandyFay.com
http://randyfay.com/node/97

[25] svn get last commit message - Alecś Web Log
http://www.alecjacobson.com/weblog/?p=2042

[26] GitX - See It
http://gitx.frim.nl/seeit.html

[27] Add SSh key to Server
http://oreilly.com/pub/h/66

[28] Maintaining patch series with Stacked GIT: a walk-through — drupal.org
http://drupal.org/node/337933

[29] Git Best Practices: Upgrading the Patch Process — Lullabot
http://www.lullabot.com/articles/git-best-practices-upgrading-patch-process

[30] Issues for Facet API — drupal.org
http://drupal.org/project/issues/facetapi?categories=All

[31] Issues for Apache Solr Search Integration — drupal.org
http://drupal.org/project/issues/apachesolr

[32] block_admin_display_form — Drupal API
http://api.drupal.org/api/drupal/modulesblockblock.admin.inc/function/block_admin_display_form/7

[33] Allow for vocab level facets — drupal.org
http://drupal.org/node/1163880

[34] help — dgo.to
http://dgo.to/

[35] All-day events missing or wrong in ical feed — drupal.org
http://drupal.org/node/1284170

[36] Issues for Drupal core — drupal.org
http://drupal.org/project/issues/drupal?status=1&categories=bug&version=7.x

[37] *http://bxl2011.drupaldays.org/sites/default/files/Search API Presentation.pdf*

[38] Interface text — drupal.org
http://drupal.org/node/604342

[39] Backpack: Debugging Drupal
https://ratatosk.backpackit.com/pub/1836982-debugging-drupal

[40] Using apachebench (ab) with Drupal 7 to load test site with authenticated users — Midwestern Mac, LLC
http://www.midwesternmac.com/blogs/jeff-geerling/using-apachebench-ab-drupal-7

[41] Apache Bench (ab) — drupal.org
http://drupal.org/node/659974

[42] Supercolliding a PHP array
http://nikic.github.com/2011/12/28/Supercolliding-a-PHP-array.html

[43] Awesome Testing Party Cheat Sheet
http://dmitrizone.com/Awesome Testing Party Cheat Sheet.html

[44] VAT — Dries Buytaert
http://buytaert.net/album/blog/vat

[45] Miscellaneous Simpletest Tips — drupal.org
http://drupal.org/node/30011

[46] Apache Solr search integration module — drupal.org
http://drupal.org/project/apachesolr

[47] Facet Api Module — drupal.org
http://drupal.org/project/facetapi

[48] Acquia Search product information
http://acquia.com/productsservices/acquia-search

[49] Apache Solr project page
http://lucene.apache.org/solr/

[50] Drupal User Group presentation about Search in Drupal 7
http://drupal.be/evenement/dug-over-search-in-drupal-7

[51] Drupal User Group presentation slides about Search in Drupal 7
http://nickveenhof.be/blog/drupal-search-and-solr-dug-november-2011

[52] Simple Guide to install Apache Solr 3.x Drupal 7
http://nickveenhof.be/blog/simple-guide-install-apache-solr-3x-drupal-7

[53] Adding custom plugins to Apache Solr
http://nickveenhof.be/blog/adding-custom-plugin-solr

[54] Story of the first few weeks as an intern at Acquia
http://nickveenhof.be/blog/story-intern-acquia

[55] Drupal.org user profile of Nick_vh (Nick Veenhof)
http://drupal.org/user/122682/track

[56] Issue queue of Apache Solr search integration module
http://drupal.org/project/issues/apachesolr

[57] Facet Api Slider module project page
http://drupal.org/project/facetapi_slider

[58] Apache Solr Sort UI project page
http://drupal.org/project/apachesolr_sort

[59] Drupal 7 search presentation in Toulouse
http://toulouse2011.drupalcamp.fr/en

[60] Acquia Network product information
http://www.acquia.com/products-services/acquia-network

[61] Brin, S. and L. Page (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30*, 107–117.
http://infolab.stanford.edu/~backrub/google.html

[62] Drupal License
http://drupal.org/licensing/faq#q1

[63] Apache License
http://en.wikipedia.org/wiki/Apache_License

List of Figures

List of Tables