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. 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" [48] 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. [49]

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. [50]

[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/products-services/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"/>
```

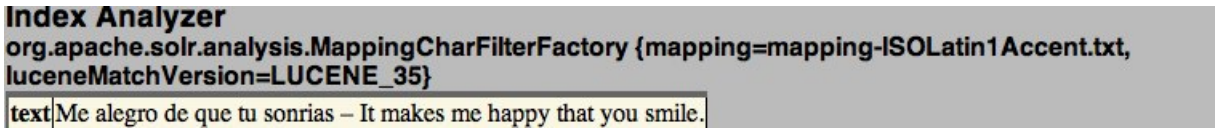
KeywordTokenizerFactory Treats the entire field as a single token, regardless of its content.

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

MappingCharFilterFactory Maps Special characters to their plain equivalent

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

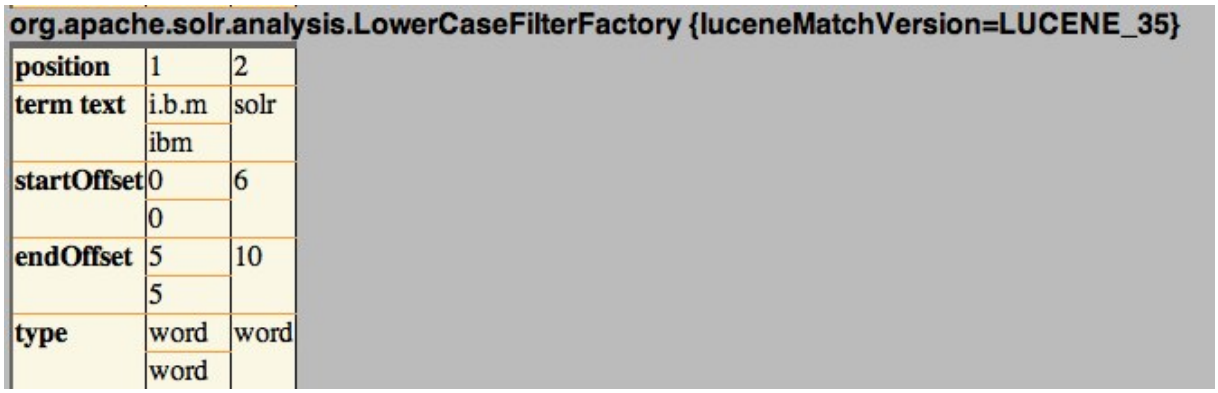
Example (index time): Me alegro de que tú sonrías – It makes me happy that you smile.



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.

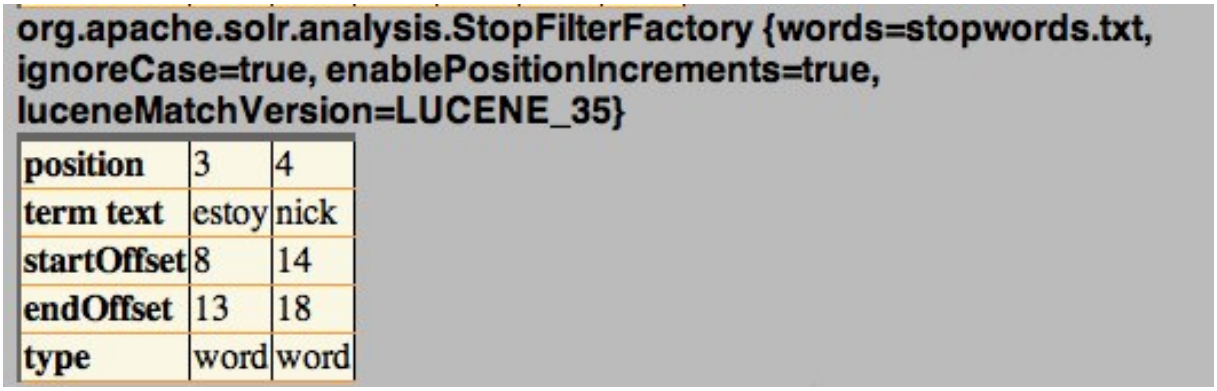
```
<filter class="solr.StopFilterFactory"
    ignoreCase="true"
    words="stopwords.txt"
    enablePositionIncrements="true"/>
```

```
# 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

Speed Speed is an important factor. During the research phase I found an interesting article [6] that showed a graph of the response time for their index. This graph shows that 97% of the requests were completed in less than a second. The average was found to be 673 milliseconds. Those 3% of the queries are slower because there is a longer disk seek time. This means that some queries contains commonly occurring words such as "a", "of", "the", "and", etc... Queries with common words take longer because the data structures containing the lists of documents containing those words in the index are larger. This same source mentions that the commongrams filter [7] for Apache Solr could resolve these queries but further investigation is due. As a conclusion it can be said that Apache Solr is a very fast add-on to Apache Lucene for full text searching, spelling corrections, faceted search and much more.

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

[6] http://www.hathitrust.org/blogs/large-scale-search/slow-queries-and-common-words-part-1

[7] http://wiki.apache.org/solr/AnalyzersTokenizersTokenFilters#solr.CommonGramsFilterFactory

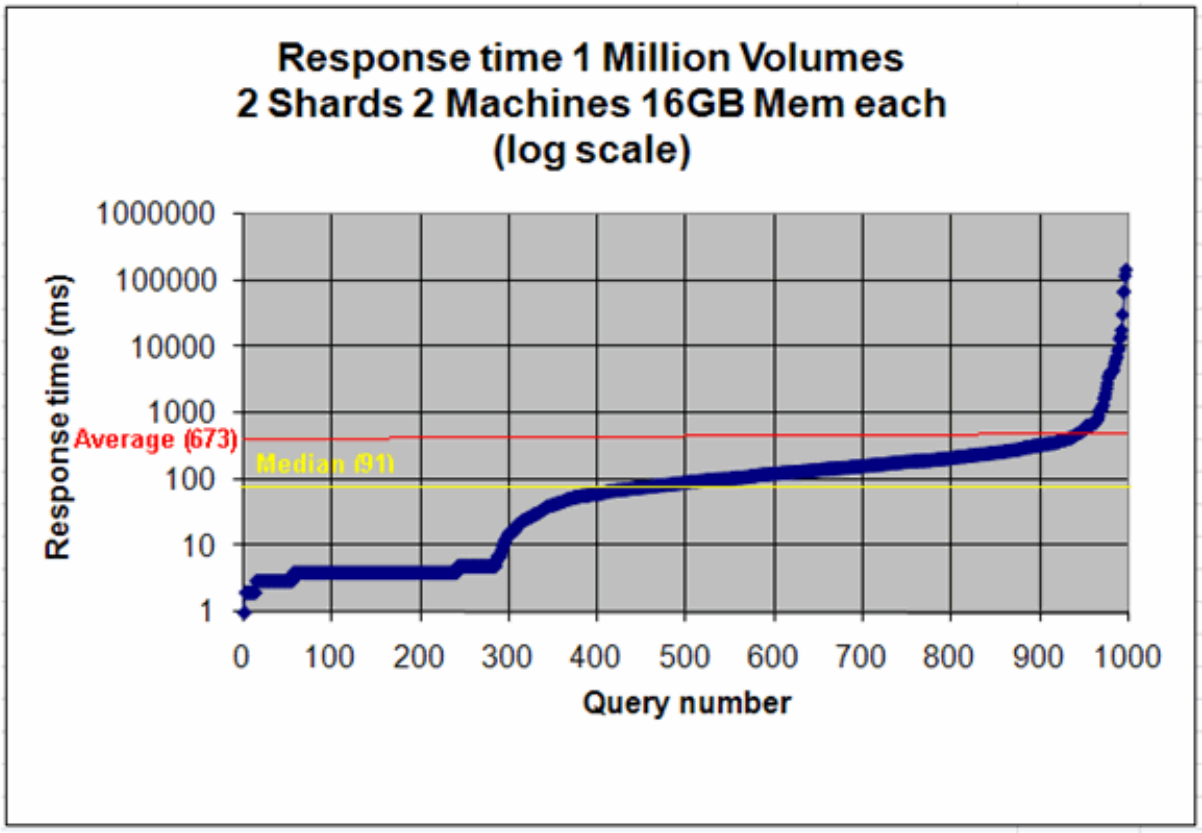


Figure 4.1: Response Time for a Solr Index with over 1 Million records (Logarithmic Scale)

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.

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.

```
 <fields>
   <!-- We use long for integer since 64 bit ints are now common in PHP. -->
   <dynamicField name="is_*"  type="long"    indexed="true"  stored="true" multiValued="false"/>
   <dynamicField name="im_*"  type="long"    indexed="true"  stored="true" multiValued="true"/>
   <!-- List of floats can be saved in a regular float field -->
   <dynamicField name="fs_*"  type="float"   indexed="true"  stored="true" multiValued="false"/>
   <dynamicField name="fm_*"  type="float"   indexed="true"  stored="true" multiValued="true"/>
   <!-- List of doubles can be saved in a regular double field -->
   <dynamicField name="ps_*"  type="double"   indexed="true"  stored="true" multiValued="false"/>
   <dynamicField name="pm_*"  type="double"   indexed="true"  stored="true" multiValued="true"/>
   <!-- List of booleans can be saved in a regular boolean field -->
   <dynamicField name="bm_*"  type="boolean" indexed="true"  stored="true" multiValued="true"/>
   <dynamicField name="bs_*"  type="boolean" indexed="true"  stored="true" multiValued="false"/>
   <!-- Regular text (without processing) can be stored in a string field-->
   <dynamicField name="ss_*"  type="string"  indexed="true"  stored="true" multiValued="false"/>
   <dynamicField name="sm_*"  type="string"  indexed="true"  stored="true" multiValued="true"/>
   <!-- Normal text fields are for full text - the relevance of a match depends on the length of the text -->
   <dynamicField name="ts_*"  type="text"    indexed="true"  stored="true" multiValued="false" termVectors="true"/>
   <dynamicField name="tm_*"  type="text"    indexed="true"  stored="true" multiValued="true" termVectors="true"/>

...

   <!-- The following causes solr to ignore any fields that don't already match an existing
        field name or dynamic field, rather than reporting them as an error.
        Alternately, change the type="ignored" to some other type e.g. "text" if you want
        unknown fields indexed and/or stored by default -->
   <dynamicField name="*" type="ignored" multiValued="true" />

 </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. [8]. 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. [9]

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 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.

Filtering Search facets, also known as filters allow users to refine or sort their search result set. Users can begin with a general search and narrow down the result set as they understand

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

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

better what content is available on a site.

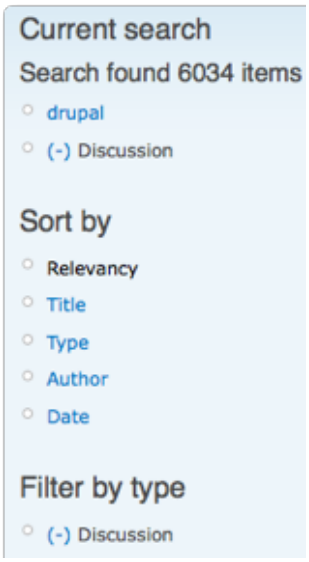


Figure 4.2: Filter by Type example. A user clicked on Discussion

When a user clicks on any term within a filter block it sends a new query to the Apache Solr server and it returns a resultset with the narrowed down results so it only includes content that matches the original query (text search) and the newly selected filter. The practice of this narrow-down method is that the user can keep selecting new filters until he found what he was looking for. The facets can be configured as OR or AND. When the user clicks the minus "(-)" the filter will be removed from the current set and show the results of the search minus that specific filter.

Configuration When the site creator wants to add more facets it is possible by going to a configuration page. as shown in the picture above. The site creator selects the facets he wants and then configures them in more detail in the block settings. The block configuration page allows you to configure, for example, the number of filters the block displays, how many it displays after clicking the show more link, the title of the block and many more. Some important facets to mention are Author, Content Type, Language, Vocabulary and Dynamic CCK/Field API filters

Content Recommendation Apache Solr can also show content suggestions to user that is viewing a specific piece of content. These suggestions are made based on the content of the viewed text. It can be used for suggestions similar to "customers who bought X also liked Y", or simply a list of relevant blog entries. The importance of certain parameters can be adjusted in the bias

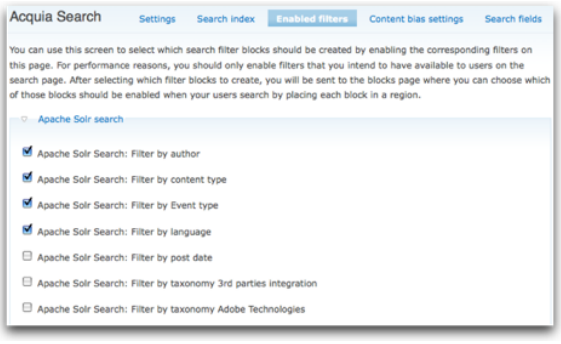


Figure 4.3: Configure the facets

settings.

Figure 4.4: Content Recommendations can be seen in the block "Related Posts"

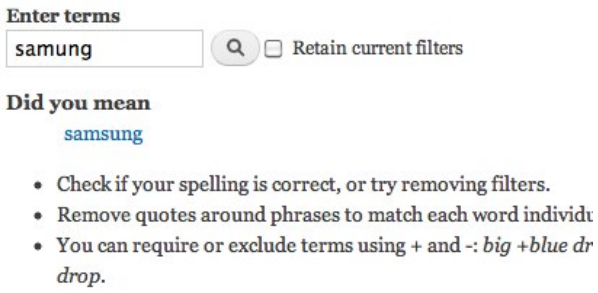


Figure 4.5: Spelling correction

Spelling Suggestions One of the other features of Apache Solr, and a feature that conquered a lot of hearts in the community was the spellchecker. Similarly to what Google does when you misspell a word it will search in the index for a word similar to your word, but with better/more relevant results.

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.

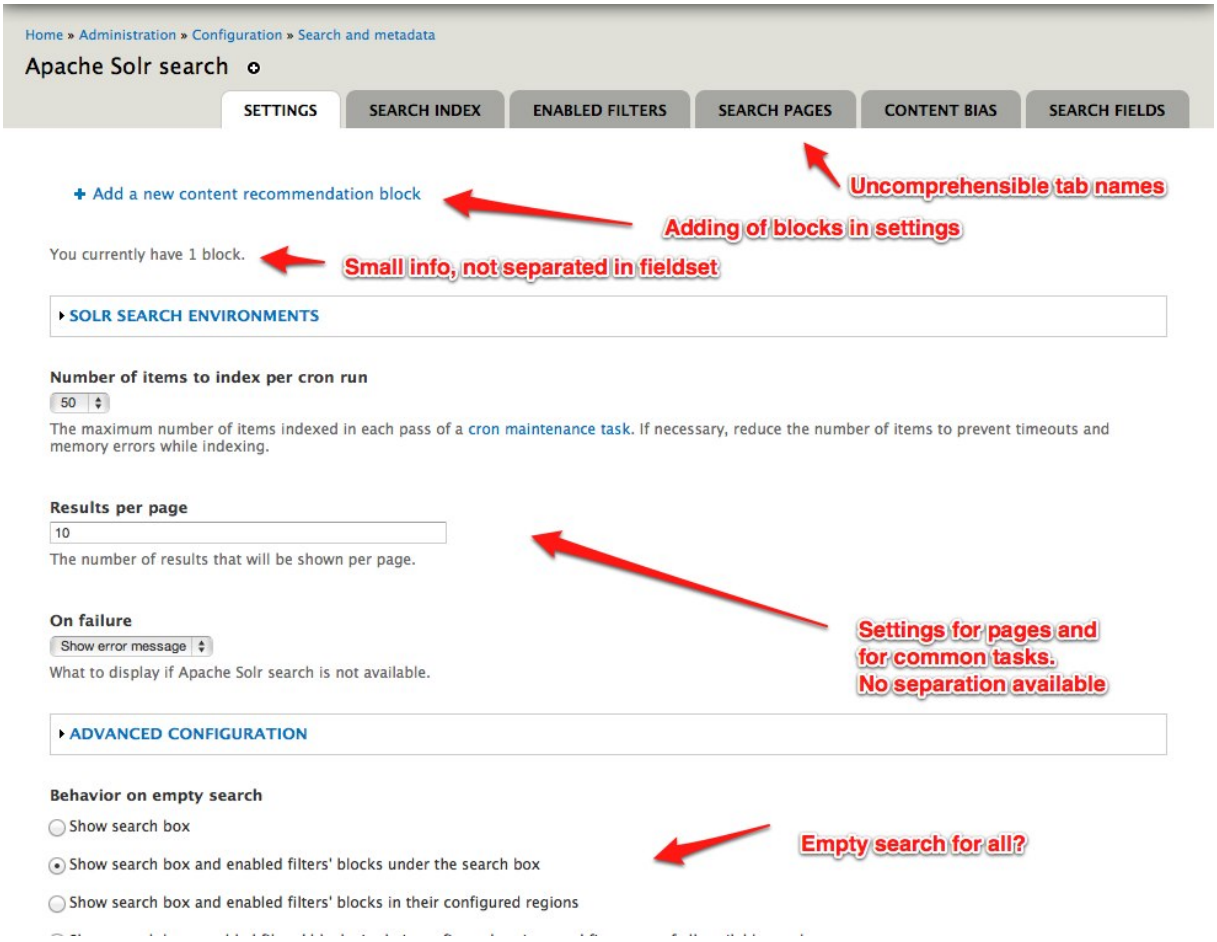


Figure 4.6: UI settings backend, September 2011

Summary for the UI settings backend

- The tab names are hard to understand. Search bias and/or Search fields are not comprehensible for a first time user.
- The first like is a link to add a content recommendation block. Surely there must be more important items to appear at the top.
- In the settings tab there are global search page settings and ideally those must be generalized so that each search page can use another preset.
- The use of the "Show search box" appears everywhere and is therefor obsolete.

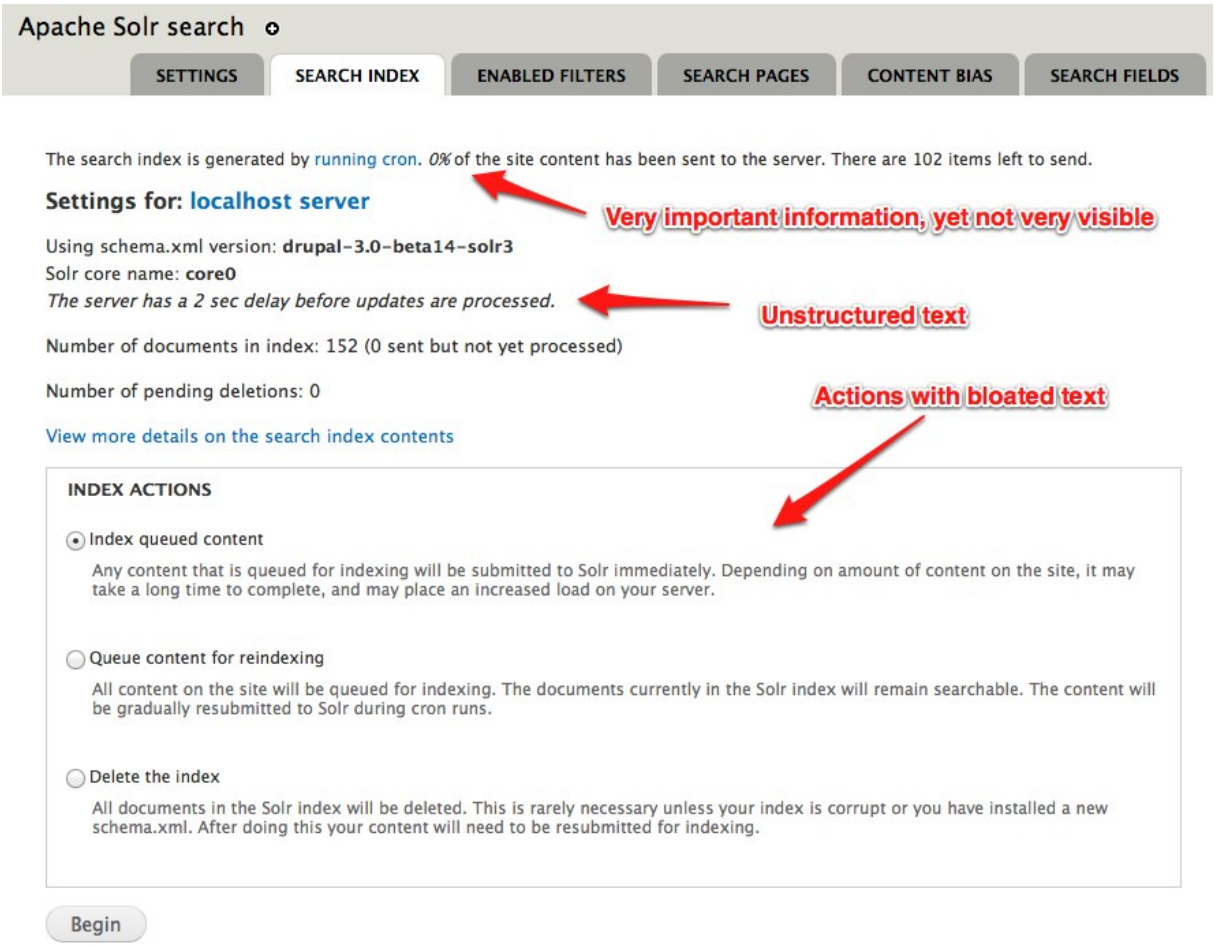


Figure 4.7: UI index report backend, September 2011

Summary for the UI report backend

- There is information spread out across the whole page. This should be structured and weight should be given to the more important parts.
- There are 3 types of actions but reading all of them makes you doubt even more about what they do. That should be clarified

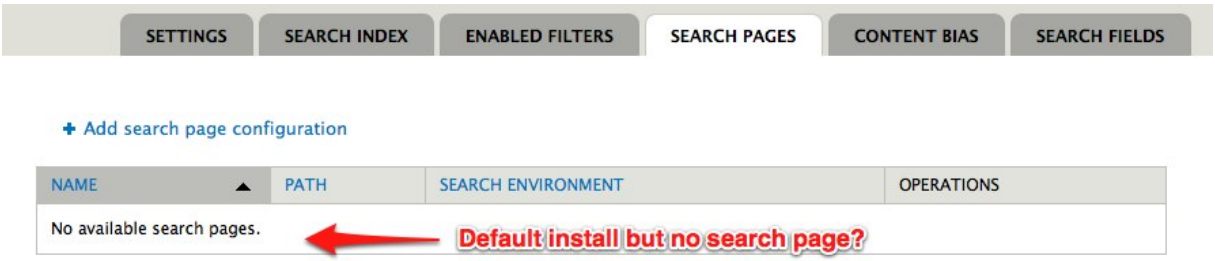


Figure 4.8: UI search pages backend, September 2011

Summary for the UI search pages backend

- When going to the search pages the first time, the user sees that the list is empty. However, when going to the search in Drupal a new search tab was added. This raises confusion and therefore the core search page (overtaken by Apache Solr) should appear in the search pages listing.

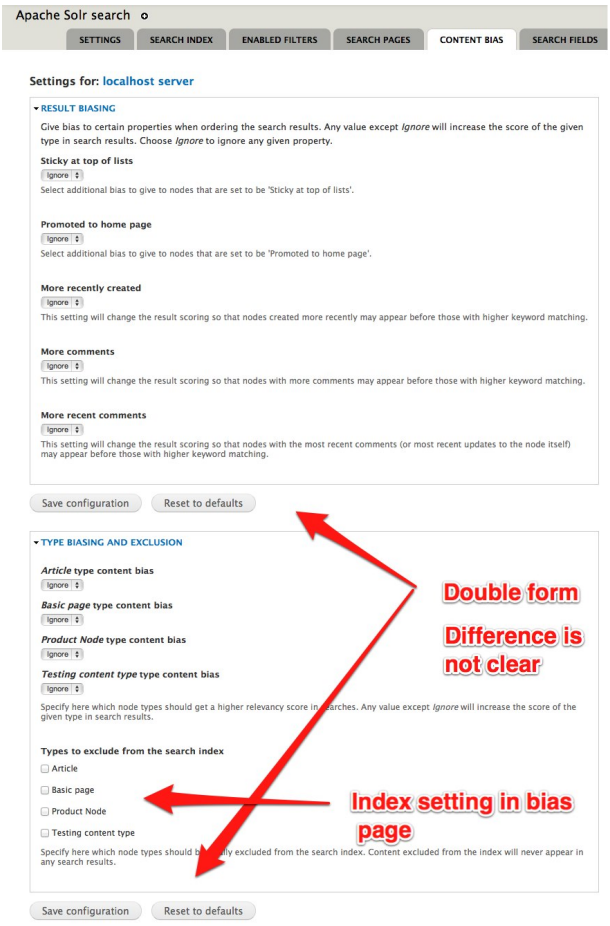


Figure 4.9: UI for result and index biasing backend, September 2011

Summary for the UI index biasing backend

- Multiple forms are visible in 1 page. According to the UI standards of Drupal this is a no-go.
- it also appears there is a settings available, a setting that excludes content types from the index, that should be moved to the index settings.

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 that should be reached by the end of February. Some of these items are also issues that were pointed out by other companies or users and they were being put into the issue queue waiting for an answer or a resolution.

Improvements

- UI refactoring to make a better experience [10]
- Support the indexing of multiple entities natively so the module would have an API to index users / terms / ... easily [11]
- Global functions should be context driven.[12]
- Get rid of dependencies in theme layer from core search [13]
- (Performance) Hooks node_type, taxonomy and user knocks out our database server [14]
- Improve file listing and access control
- More like this blocks should get a delete button [15]
- De-duplicate core and custom search in order to obtain clarity in the code [16]

[10] `http://drupal.org/node/1292364`
[11] `http://drupal.org/node/1292364`
[12] `http://drupal.org/node/1292364`
[13] Related to `http://drupal.org/node/1314406` (de-duplication)
[14] `http://drupal.org/node/592522`
[15] `http://drupal.org/node/1271964`
[16] `http://drupal.org/node/1314406`

- 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 [17]
- Query type should be adjusted in order to allow different widgets in facetapi [18]
- Change the PHP static to the Drupal function drupal_static() [19]
- Non-current/valid Node Types are not excluded from index [20]
- Add retain current filters checkbox to custom search page [21]
- Add "retain-filters" param when in facet browsing mode [22]
- Handle 1 placeholder in a custom search page path with makes the taxonomy sub-module obsolete [23]
- Create tests for the module [24]
- Bundle' is not a required field, but Apachesolr treats it as such (Evaluate required fields in schema, make non-required if possible) [25]
- Improve Date faceting/date query type (combined with facetapi) [26]
- Facets are currently not linked to the appropriate search page
- Add clone operation for search environments [27]
- 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

[17] http://drupal.org/node/1161538
[18] http://drupal.org/node/1161444
[19] http://drupal.org/node/1334216
[20] http://drupal.org/node/1000532
[21] http://drupal.org/node/1246422
[22] http://drupal.org/node/1116792
[23] http://drupal.org/node/1294846
[24] http://drupal.org/node/989398
[25] http://drupal.org/node/1279164
[26] http://drupal.org/node/1201534
[27] http://drupal.org/node/1292328

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)

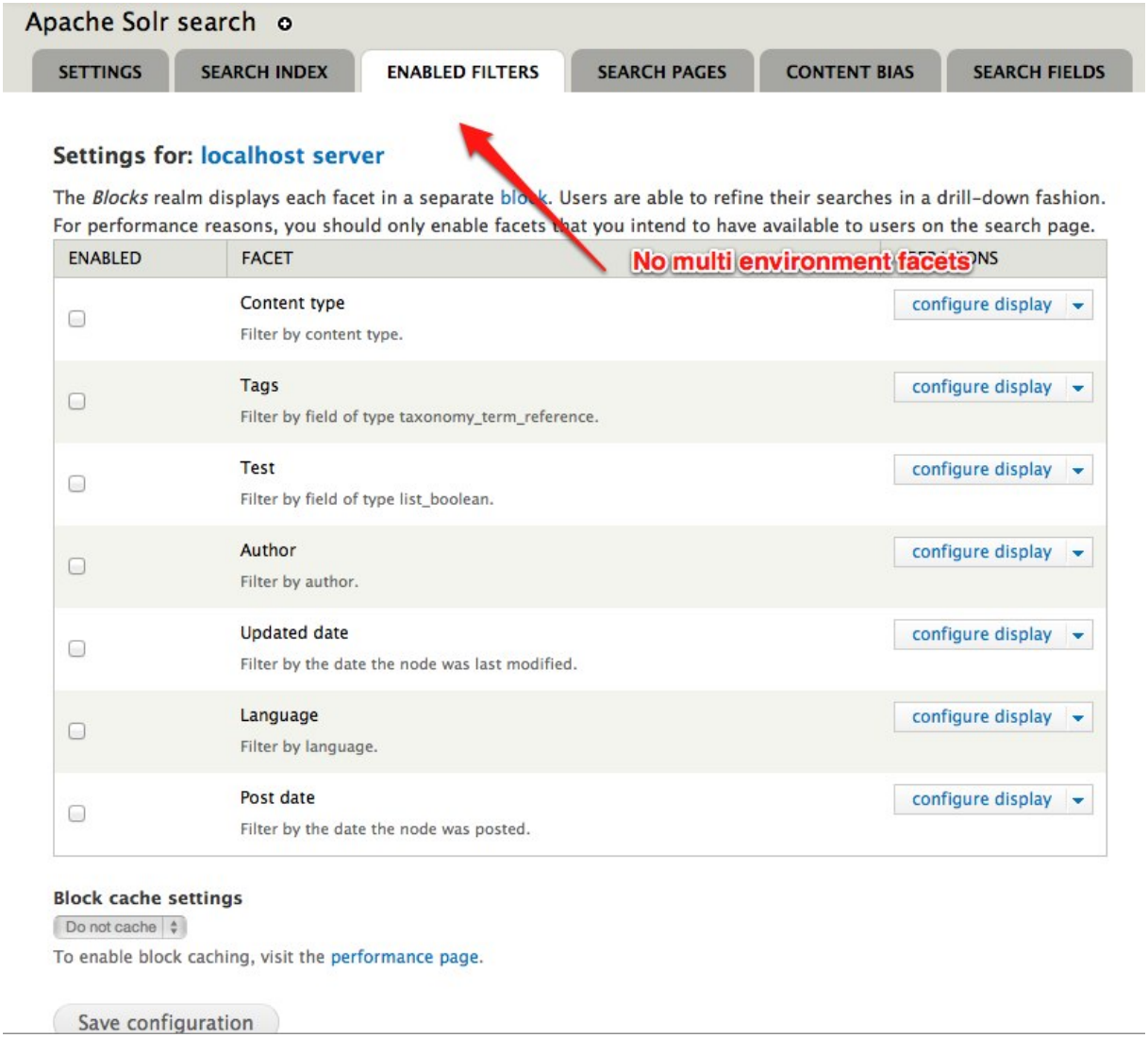


Figure 4.10: Apachesolr Facetapi Integration UI as of September 2011

Summary for the UI search pages backend

- There was no possibility to easily switch to facets from other environments because one

had to make the other environment the default one. This was a huge workaround and had to be fixed.

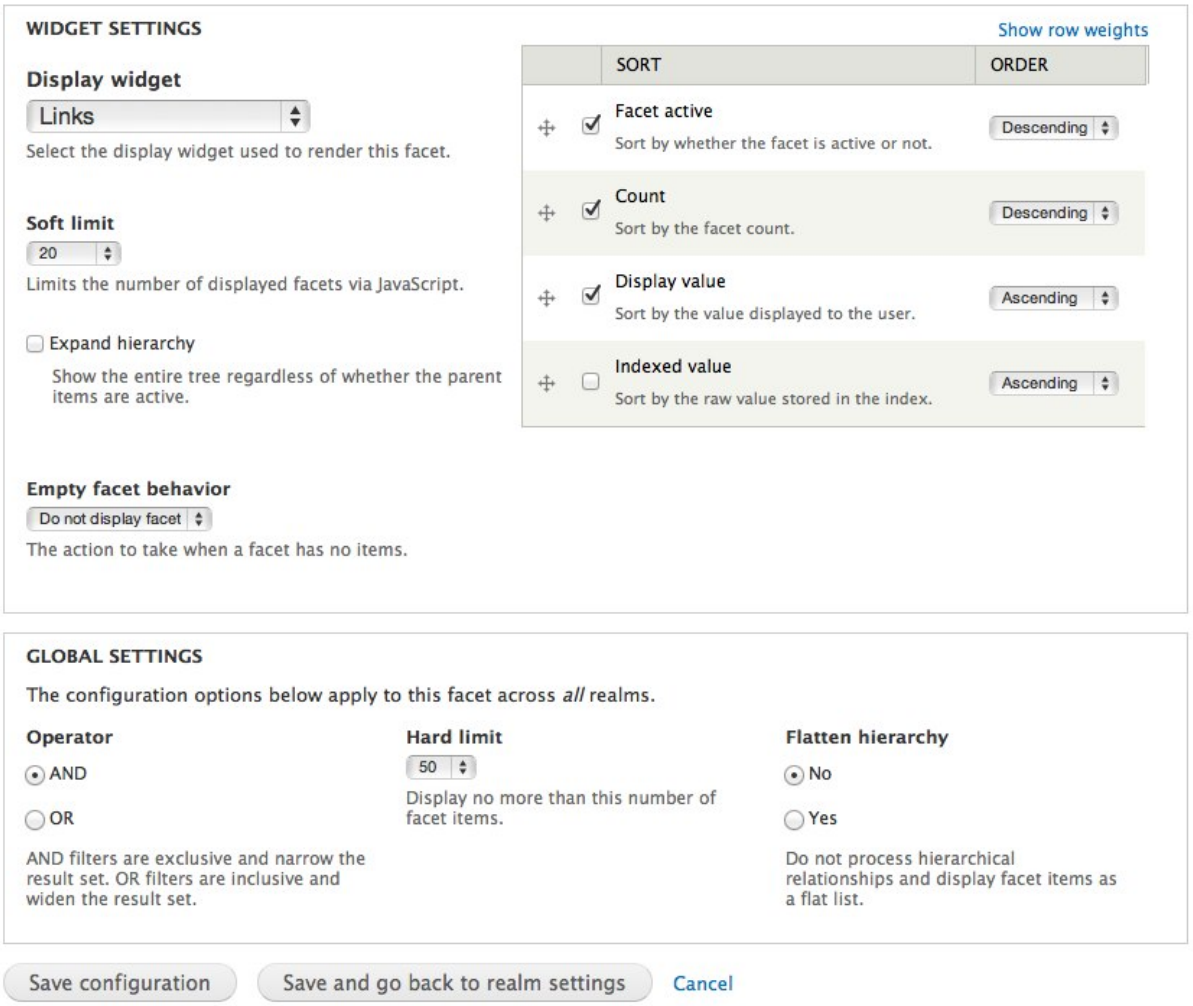


Figure 4.11: Apachesolr Facetapi Integration UI of 1 facet as of September 2011

- The Facet details page was ok in its use and therefor it did not need any further adjustments.

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 [28]was raised in the Facet Api issue queue on `drupal.org` for those that prefer to read up in detail.

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

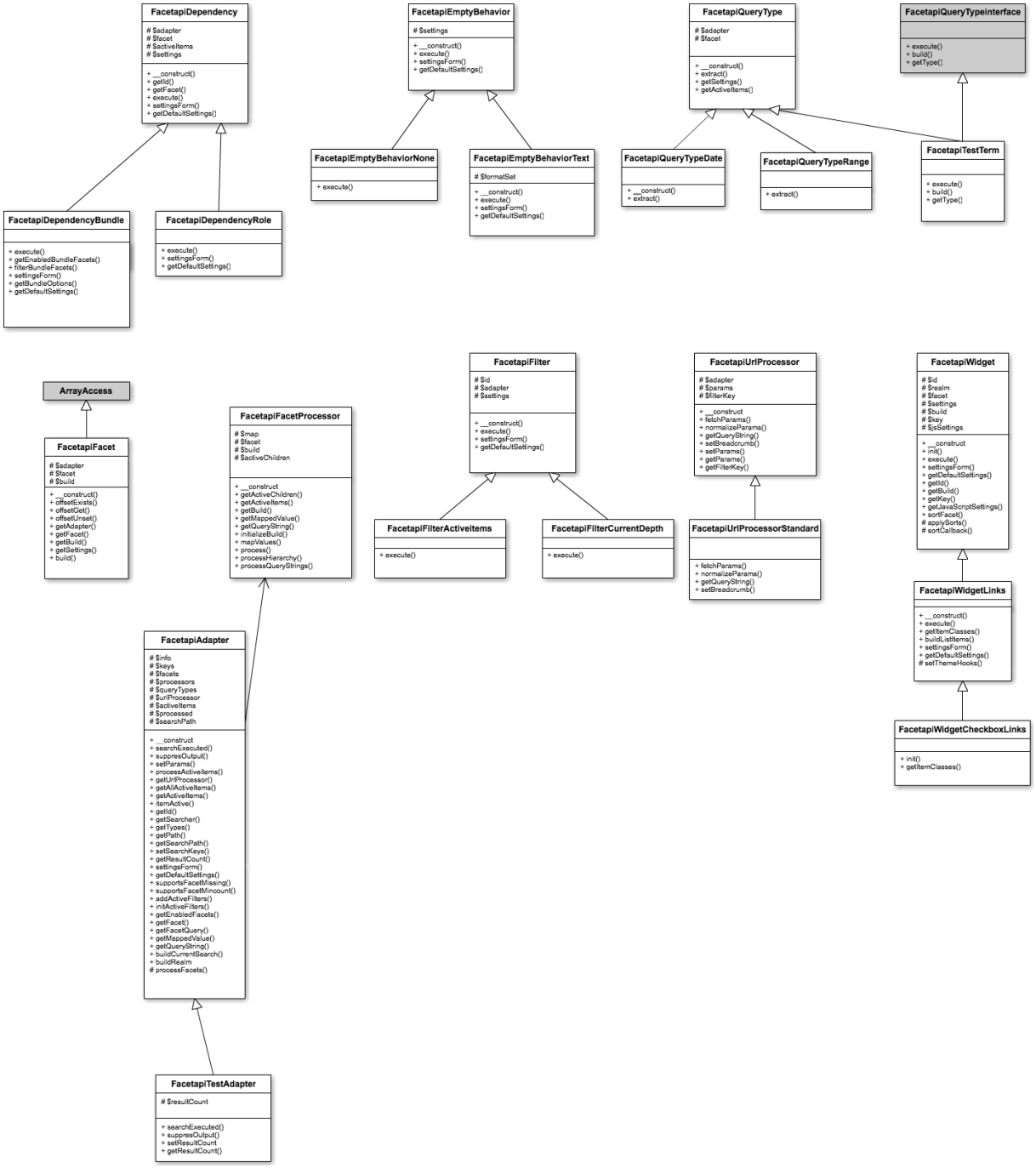


Figure 4.12: Extended information about the classes in FacetAPI, September 2011

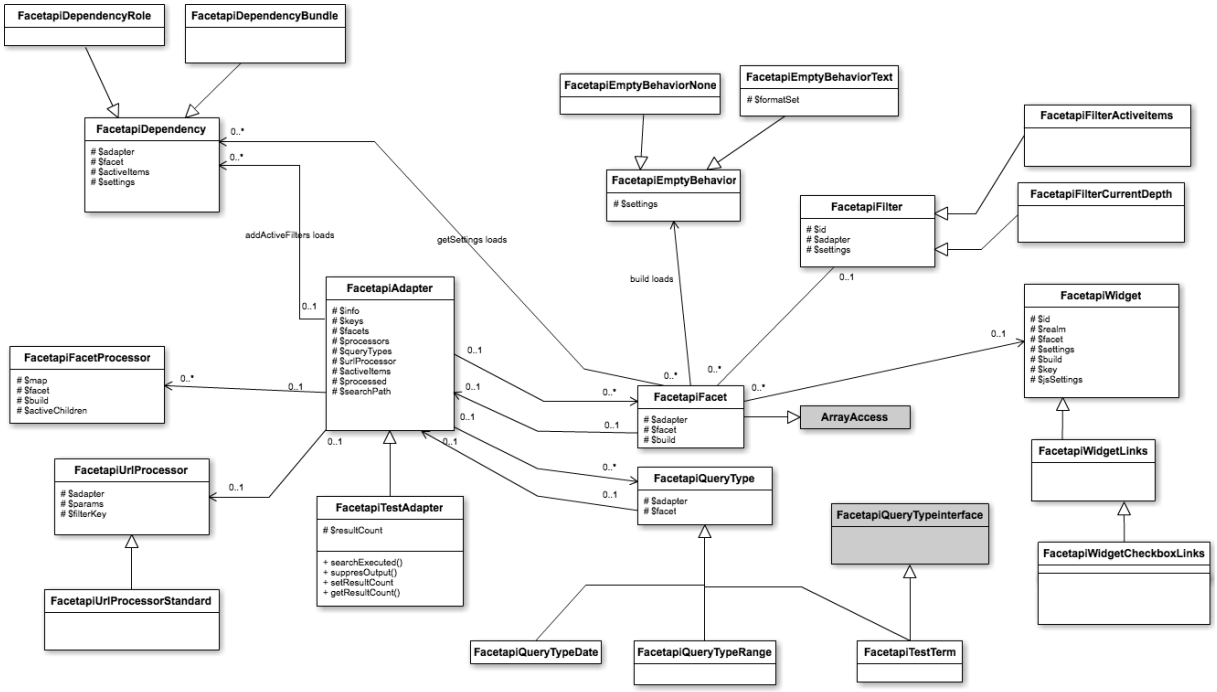


Figure 4.13: Class Diagram of FacetAPI, September 2011

- 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 [29]
- Make the current search block more configurable [30]
- Complete configuration import functionality [31]

[29] `http://drupal.org/node/1161434`
[30] `http://drupal.org/node/593658`
[31] `http://drupal.org/node/1147564`

- 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." [32]

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."

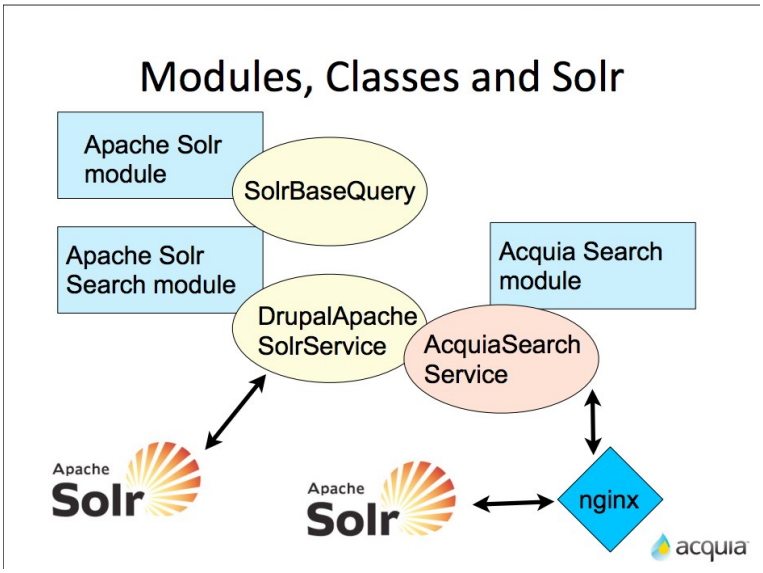


Figure 4.14: Overview of the classes and services used for Acquia Search at the website's end.

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

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.

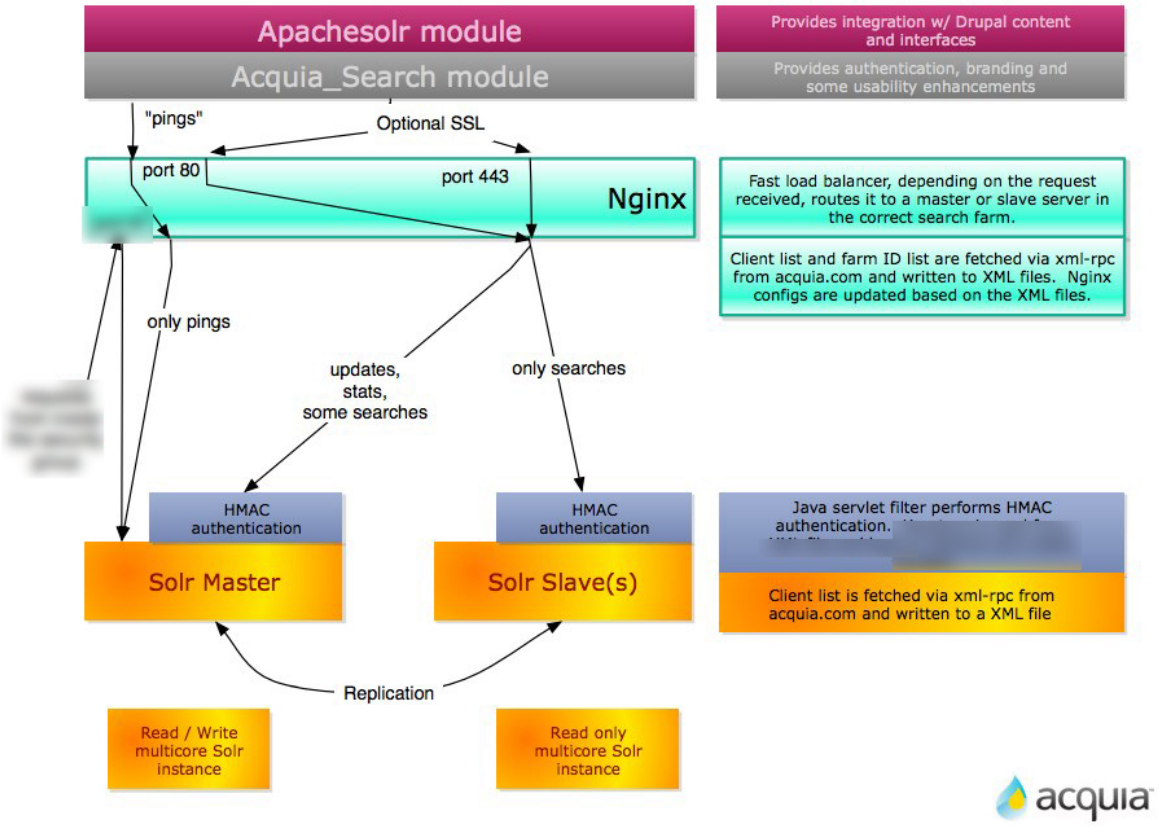


Figure 4.15: Server Side view of Acquia Search. Certain information has been blurred for confidentiality

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 SHA1-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 SHA1-HMAC authentication, if the data that was sent was not modified.

This kind of authentication is commonly called a symmetric signature. Using a shared secret the message is signed and also verified as can be seen in the image below.

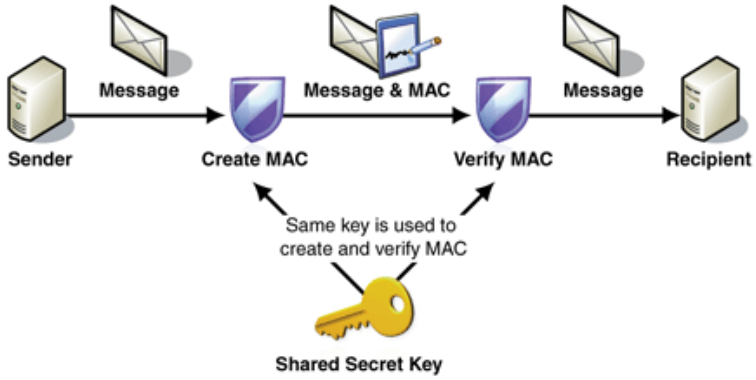


Figure 4.16: Signing a message using a symmetric signature

As illustrated in the figure above, signing a message using a symmetric signature involves the following steps:

- The sender creates a HMAC using a shared secret key and attaches it to the message.
- The sender sends the message and HMAC to the recipient.
- The recipient verifies that the HMAC that was sent with the message by using the same shared secret key that was used to create the HMAC.

By signing with a shared secret, both data integrity and data origin authenticity are provided for the signed message content. The only downside is that the receiver does not know who exactly wrote the message. It can only verify if it was encrypted with the right shared key. Acquia Search creates a hash from the network identifier (aka the subscription ID from the customer) and generates a secret shared key from this information. Using this derived key and a blob of data it can be easily encoded so the backend is able to verify the authenticity of the message

```
1  <?php
2  /**
3   * Derive a key for the solr hmac using the information shared with acquia.com.
4   */
5  function _acquia_search_derived_key() {
6    $key = ACQUIA_KEY;
```

```
7    $subscription = SUBSCRIPTION_INFO_ARRAY;
8    $identifier = ACQUIA_IDENTIFIER;
9    // We use a salt from acquia.com in key derivation since this is a shared
10   // value that we could change on the AN side if needed to force any
11   // or all clients to use a new derived key.  We also use a string
12   // ('solr') specific to the service, since we want each service using a
13   // derived key to have a separate one.
14   $salt = $subscription['derived_key_salt'];
15   $derivation_string = $identifier . 'solr' . $salt;
16   $derived_key = _acquia_search_hmac($key, str_pad($derivation_string, 80, $derivation_string));
17   return $derived_key;
18 }
19
20 /**
21  * Calculates a HMAC-SHA1 of a data string.
22  *
23  * See RFC2104 (http://www.ietf.org/rfc/rfc2104.txt). Note, the result of this
24  * must be identical to using hash_hmac('sha1', £string, £key);  We don't use
25  * that function since PHP can be missing it if it was compiled with the
26  * --disable-hash switch. However, the hash extension is enabled by default
27  * as of PHP 5.1.2, so we should consider requiring it and using the built-in
28  * function since it is a little faster (~1.5x).
29  */
30 function _acquia_search_hmac($key, $string) {
31   $output = str_pad($key, 64, chr(0x00)) ^ (str_repeat(chr(0x5c), 64));
32   $output .= pack("H*", sha1((str_pad($key, 64, chr(0x00)) ^ (str_repeat(chr(0x36), 64))) . $string));
33   return sha1($output);
34 }
```

Listing 6: Small excerpt to show how the Client side generates the HMAC message to send back to the Solr Service

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 so it could be used in the software as a service ideology that Acquia has. The version that is currently used is Apache Solr 1.4 and is still one of the defacto standards when deploying Solr.

As always, upgrading does not usually 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.

Improvements

- Upgrade the Java Servlet that was written to handle HMAC authentication Solr request to Solr 3.x
- Test if the Solr 3.x performs better or equally good using the upgraded code and by using existing indexes and a real infrastructure server setup to emulate a real-life situation.

Chapter 5

Implementation

5.1 Communication

Every thing has a start and an end 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 I wrote about this. Since my involvement with the project there are about 3000 websites using the Drupal 7 version of the module and about 11000 users in total using the module (Drupal 6 and Drupal 7). I've committed more than 220 patches to the Apache Solr project and all of them were made and created publicly and I've had help from a whole community when reviewing those on the fly. I will continue working on the module and I will continue motivating people to help out and make the project better as a whole.

Exposing the work to the public I have recently given a presentation 'Drupal Search' explaining to more than 60 attendees what was done with the module and where it was heading to in Belgium. Lots of open communication has happened within the community in the Apache Solr Issue Queue. In total I have given 5 presentations with a combined total of more then 400 attendants.

5.1.1 Daily communication

When the thesis/internship started I was a bit unclear how it would work. Working from your home every day, and having a time difference of 6 hours proved to be quite challenging.

At acquia they use an internal chatserver but also promote alternative ways of communication. During this period I've used Skype, Adobe Connect, Google Hangout, VOIP using my phone's 3G. Joining conference room calls while 50% of the people were attending the meeting physically and others remotely was also a daily event to look out for.

From October until the end of December I mainly communicated with Peter Wolanin to keep track of the daily work and to clear out any questions or issues that could block my progress. The methodology was similar to a scrum session but more personal. This sheet was divided in 4 main segments.

- What did I get done since last meeting.
- What will I get done before the next meeting
- What is slowing me down or blocking progress
- What did you discover that would be of interest, or needs to be discussed post scrum?

Below one can see how this looked like. The full document is available upon request.

	A	B	C	D	E
1					
2	**Date**	**Since last scrum**	**Before next Scrum**	**Impediments**	**Need to discuss**
3	*Nick answers 3-4 Qs*	*What did I get done since last meeting*	*What will I get done before the next meeting*	*What is slowing me down or blocking progress*	What did you discover that would be of interest, or needs to be discussed post scrum?
4		*Built the Ark. Figured out what a cubit is.*		*The tigers are eating the squirrels.*	*Competition is building another bigger boat*
48	11/3/2011	Loads of paperwork for legal receival of the payment from Acquia (Complicated in Belgium), Identified the problem (Filter has an output that is not in correspondence with the newer build of solr). Processed feedback on UI and worked on support questions in issue queue	Solve these authentication problems	get a better understanding on how custom filters in solr work	
49	11/2/2011	Plugin is committed to the SVN + war file also. Tests are working and simplified. Acquia Network Demo meeting. Improved the UI issue and asked for feedback. Numeric ID issue should be ready to go + blog is written	Debug And solve the authentication problems that are happening during a live connection		
50	11/1/2011	Acquiasearch plugin is working on solr 3.4 with a lot of simplifications. plugin has been mailed to pwolanin	Commit the war file + plugin. Make the tests work when doing ant example + Learn about server environment, finish UI improvements and finish removal of the numeric field id	SVN was working against me. Structure was not clear but it could be my own mistake	
51	10/31/2011	Understanding of the build.xml modes is getting there. A solution is near but the lack of documentation is a blocker.	Finish the Solr 3.4 Acquiasearch plugin		
52	10/28/2011	Ported a solr sort UI to D7 and tried to get some traction in the community for follow-ups, fixed an important issue with page titles in D7 dev	Sort out the Solr 1.4 Acquia modifications and try to understand the build.xml methods + get a version working for solr 3.4		
53	10/27/2011	Rebased the patches for solr (and tested to work, have images as proof) and started at the build patch but there is some info missing and my knowledge is not good enough. Slider min-max is working and a second query is used now to handle that. (search for optimization) + all issues from the slider are set to fixed right now + went to a business fair to make sure I have all legal papers to work for Acquia in Belgium + Read up on http://drupal.org/node/961604	Non-current/valid Node Types not excluded from index + Change static to drupal_static + issue queue gardening		Shards (unknown area for me)
54	10/26/2011	Class Diagram and Documentation has been updated (http://drupal.org/node/1321166 and http://drupal.org/node/1321136) + Facet_slider is working properly http://drupal.org/sandbox/cpliakas/1160920	rebase the patches for solr to be used by acquia search + continueing on query patch for the slider and slider itself (caching problem with the min and max)		
55		Documentation of facetapi and apachesolr			

Figure 5.1: Mini Daily Scrum worksheet

From the 1st of January I also joined the team of Michael cooper, who leads the network project in Acquia. He holds a daily standup where we do a similar exercise but ,instead of writing, the team members tell (preferable while standing up) what they have done, what they plan to do and what is blocking them. The network team uses a full scrum methodology. [1]

[1] More about scrum can be found here. This section does not elaborate further because since scrum itself could

5.1.2 Drupal Camps and seminars

Exposing the work that was done in a community such as Drupal is very important to get validation and also constructive comments to be able to built further. Those meetings (Drupalcamps, DrupalDays, anything similar) are important to meet people that are in the same technologic space. People with experience and knowledge, but also people with an opinion that makes one think, think hard. Also it is important to expose knowledge to people who are new to Drupal, it makes their lives easier and, if all goes well, the community can count on one more skilled person to help out in need.

After the redesign of the Apache Solr UI and a bunch of other changes under the hood there were a couple of events that accepted a talk from me and allowed me to explain what I did, why I was doing this and how they could help making all of this more successful.

Drupal User Group Belgium The 9th of November I gave a presentation in Ghent, Belgium about Search in Drupal 7. More than 60 attendees showed up and most of them were happy to learn about the details of Search and more about Apache Solr. Details can be found on `http://drupal.be/evenement/dug-drupal-user-group-meeting-over-search-in-drupal-7`

Drupal Camp Toulouse From the 26th of November until the 27th of November I've attended a Drupalcamp in Toulouse `http://toulouse2011.drupalcamp.fr/en`. Even though I was not allowed to present as an Acquian member I did go there and I've presented as myself. The topic of the presentation was similar but more focussed on questions and answers. The french audience was a bit smaller (around 30) compared to the Drupal User Group in Belgium but the talks I had afterwards with members of the community were great.

Acquia Internal Training On the 2nd of December I presented the same presentation as the one I did in Toulouse but then for the employees of Acquia, using an online platform. In the company this is a called a lunch and learn so a bunch of people gather during lunchtime and follow a presentation about a certain topic. Personally this was the most stressful presentation because there was less interaction and I could not see the audience. On the other hand, the audience could not see me neither. The only thing visible was my computer screen and the presentation. Around 15 people attended this presentation. The presentation

be a whole other thesis.`http://en.wikipedia.org/wiki/Scrum_(development)`

from Toulouse and the Internal Training can be found on http://prezi.com/j-wss0nznowb/acquia-lunch-and-learn-december-2011-drupal-search/

Drupal Science Camp A totally different experience was the Drupal Science Camp in Cambridge. This Camp was organized in 21st and the 22nd of January in a business complex and was crowded. everything was organized really quickly but that did not affect the quality of the conference. I gave a final overview of what I had done in those months and what could/should still happen when my time was over. More information about the session can be found on http://www.drupalsciencecamp.org.uk/sessions/drupal-search-and-solr-wizardry. Over 40 people were packed in a room to listen to the presentation and approached me with challenging questions.

5.1.3 Blog Posts

To continue on the dissemination of the results and get an even broader audience a series of blog posts were written. They have generated, when accumulated, over 20.000 page views and a series of comments and reactions.

- Using Github application in your patch workflow [2]
- Attending the Boston Drupal User Group [3]
- Changing a git commit message in the commit history [4]
- A story of an intern at Acquia [5]
- Adding a custom plugin to Solr [6]
- A simple guide to install Apache Solr 3.x for Drupal 7 [7]
- Slides of the Drupal Search and Solr sessions [8]
- Upgrading from Apache Solr 1.4 to Apache Solr 3.5 and its implications [9]

[2] http://nickveenhof.be/blog/using-github-application-patch-workflow-0
[3] http://nickveenhof.be/blog/boston-drupal-user-group
[4] http://nickveenhof.be/blog/changing-git-commit-message-commit-history
[5] http://nickveenhof.be/blog/story-intern-acquia
[6] http://nickveenhof.be/blog/adding-custom-plugin-solr
[7] http://nickveenhof.be/blog/simple-guide-install-apache-solr-3x-drupal-7
[8] http://nickveenhof.be/blog/drupal-search-and-solr-dug-november-2011
[9] http://nickveenhof.be/blog/upgrading-apache-solr-14-35-and-its-implications

Conferences + Blogging Result An estimated crowd of 150 people have seen the presentation and the work in front of me and a whole lot more have seen the blog posts. All this from start to end, early and a later stage. By presenting a work often, and exposing what one does quickly, it raises the chance on a better product in the end. All the constructive comments were written down and reported back to the daily scrum and were merged in to the roadmap. This feedback also gave an invaluable pointer to the importance of certain items, such as the multi entity support, in the roadmap. Working remotely removes a lot of physical contact but these social events surely compensate a missing piece in the puzzle for remotes.

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

The module got a major overhaul in these months. Over 219 commits from the author were made to the codebase of the Apache Solr project

5.2.1 Search Environments

UI Improvements As seen in chapter 4 it became clear that the old UI did not suffice for managing multiple environments. Also there was no way to remove environments and/or clone them. During the process the search environment got its own tab (under settings) and it is also directly visible if a search environment is online or not (green color). The advanced configuration got overhauled also and now includes options such as the amount of items to index per cron and also a select box of what to do when the module fails to connect/fails to execute a query. The benefit of having multiple environments is easy. retrieving data from two indexes or having indexes for the development/staging and production environment. Another addition that was made is that the search bias and boost information is now environment dependent instead of being applied to all queries.

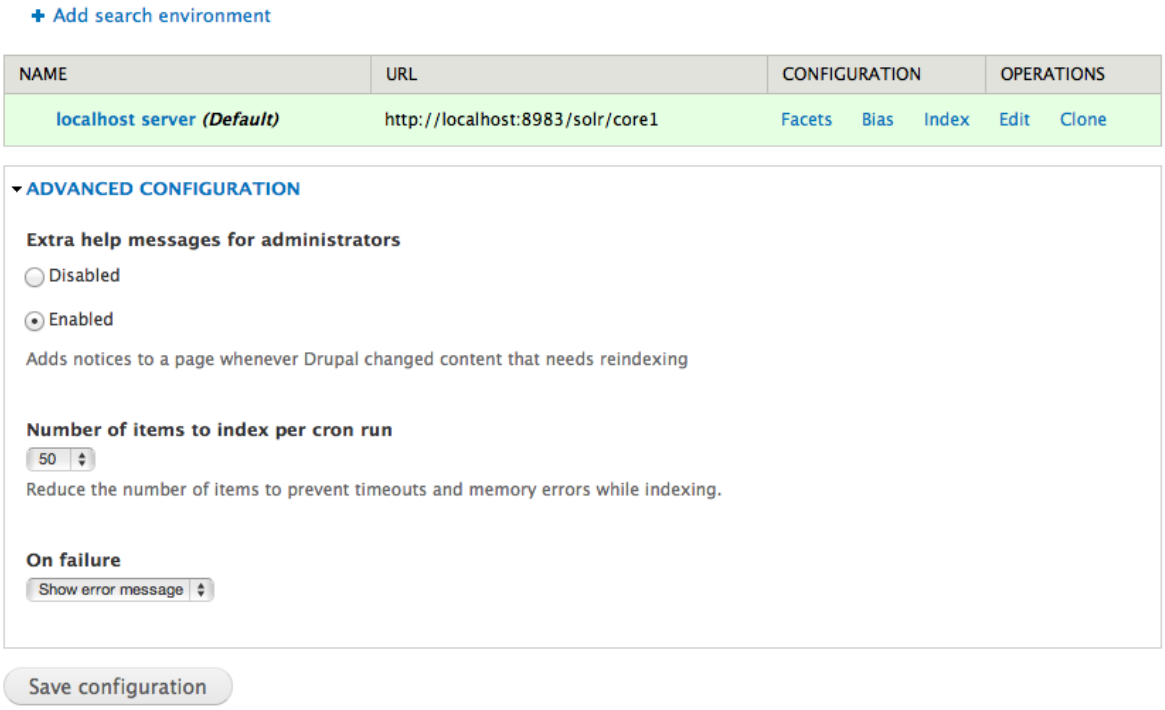


Figure 5.2: Search Environments

Functions Originally, the module did not have support for multiple search environments. Just moments before this thesis was executed some small support for multiple search environments was added. Most of the functions did not take a search environment argument and were dependent on the default search environment. During these months of dedication to the Apache Solr Module we improved the support for multiple search environments.

Listing 7 is a list of functions signatures with their parameters. All of them were adjusted to support multiple environments since none of these functions were already supporting this functionality. Changing all of them was a tedious work that involved a lot of testing, feedback and more testing. Which brings you, my dear reader to the next paragraph!

```
1  <?php
2  /**
3   * Batch Operation Callback
4   */
5  function apachesolr_index_batch_index_entities($env_id, &$context) {
6
7  /**
8   * Send up to $limit entities of each type into the index.
9   */
10 function apachesolr_index_entities($env_id, $limit) {
```

```
11
12  /**
13   * Returns an array of rows from a query based on an indexing environment.
14   * @todo Remove the read only because it is not environment specific
15   */
16  function apachesolr_index_get_entities_to_index($env_id, $entity_type, $limit) {
17
18  /**
19   * Delete an entity from the indexer.
20   */
21  function apachesolr_index_delete_entity_from_index($env_id, $entity_type, $entity) {
22
23  /**
24   *
25   * @param type £type
26   * @return type
27   * @todo Add Type support
28   */
29  function apachesolr_index_mark_for_reindex($env_id, $entity_type = NULL) {
30
31  /**
32   * Sets what bundles on the specified entity type should be indexed.
33   *
34   * @param string £env_id
35   *   The Solr core for which to index entities.
36   * @param string £entity_type
37   *   The entity type to index.
38   * @param array £bundles
39   *   The machine names of the bundles to index.
40   */
41  function apachesolr_index_set_bundles($env_id, $entity_type, array $bundles) {
42
43  /**
44   * Returns last changed and last ID for an environment and entity type.
45   */
46  function apachesolr_get_last_index_position($env_id, $entity_type) {
47
48  /**
49   * Sets last changed and last ID for an environment and entity type.
50   */
51  function apachesolr_set_last_index_position($env_id, $entity_type, $last_changed, $last_entity_id) {
52
53  /**
54   * Set the timestamp of the last index update
55   * @param £timestamp
56   *   A timestamp or zero. If zero, the variable is deleted.
```

```
57   */
58  function apachesolr_set_last_index_updated($env_id, $timestamp = 0) {
59
60  /**
61   * Semaphore that indicates whether a search has been done. Blocks use this
62   * later to decide whether they should load or not.
63   *
64   * @param £searched
65   *   A boolean indicating whether a search has been executed.
66   *
67   * @return
68   *   TRUE if a search has been executed.
69   *   FALSE otherwise.
70   */
71  function apachesolr_has_searched($env_id, $searched = NULL) {
72
73  /**
74   * Semaphore that indicates whether Blocks should be suppressed regardless
75   * of whether a search has run.
76   *
77   * @param £suppress
78   *   A boolean indicating whether to suppress.
79   *
80   * @return
81   *   TRUE if a search has been executed.
82   *   FALSE otherwise.
83   */
84  function apachesolr_suppress_blocks($env_id, $suppress = NULL) {
85
86  /**
87   * Get a named variable, or return the default.
88   *
89   * @see variable_get()
90   */
91  function apachesolr_environment_variable_get($env_id, $name, $default = NULL) {
92
93  /**
94   * Set a named variable, or return the default.
95   *
96   * @see variable_set()
97   */
98  function apachesolr_environment_variable_set($env_id, $name, $value) {
99
100 /**
101  * Get a named variable, or return the default.
102  *
```

```
103  * @see variable_del()
104  */
105 function apachesolr_environment_variable_del($env_id, $name) {
106
107 /**
108  * Static getter/setter for the current query. Only set once per page.
109  */
110 function apachesolr_current_query($env_id, DrupalSolrQueryInterface $query = NULL) {
111
112 /**
113  * Gets a list of the bundles on the specified entity type that should be indexed.
114  *
115  * @param string $core
116  *   The Solr environment for which to index entities.
117  * @param string $entity_type
118  *   The entity type to index.
119  * @return array
120  *   The bundles that should be indexed.
121  */
122 function apachesolr_get_index_bundles($env_id, $entity_type) {
```

Listing 7: List of functions that were modified to support multiple environments

Testing Drupal uses a specific framework for testing, called Simpletest [10] This testing framework is nested very deeply into the Drupal 7 development process.

The Functional tests are the most common. They create a fresh database installation and specifically create data for the test in the database and then make assertions based on expected results. Unit tests work without a database installation in the backend and are useful for isolated functions that don't make assumptions about the larger system. Upgrade tests use a database dump from an earlier version of Drupal and import that to run update.php and then check assertions.

For the search environments tests were written as functional tests. As explained above these test try to assess the functionality from an UI point of view.

```
1 <?php
2 /**
3  *        Asserts that we can edit a search environment
4  */
5 function testEditSearchEnvironment() {
6   $this->drupalLogin($this->admin_user);
7   $this->drupalGet('admin/config/search/apachesolr/settings');
```

[10] Simpletest info can be found on `http://drupal.org/simpletest`

```
8     $this->clickLink(t('Edit'));
9     $this->assertText(t('Example: http://localhost:8983/solr'), t('Edit page was succesfully loaded'));
10    $edit = array('name' => 'new description foo bar', 'url' => 'http://localhost:8983/solr/core_does_not_exists');
11    $this->drupalPost($this->getUrl(), $edit, t('Save'));
12    $this->assertResponse(200);
13    drupal_static_reset('apachesolr_load_all_environments');
14    drupal_static_reset('apachesolr_get_solr');
15    $this->drupalGet('admin/config/search/apachesolr/settings');
16    $this->assertText(t('new description foo bar'), t('Search environment description was succesfully edited'));
17    $this->assertText('http://localhost:8983/solr/core_does_not_exists', t('Search environment url was succesfully edit
18  }
```

Listing 8: Example of a search environment test

```
1   <?php
2
3   /**
4    *        Asserts that the module was installed and that a notice appears that the server is offline
5    */
6   function testServerOffline() {
7
8   /**
9    *        Asserts that the module was installed and that a notice appears that the server is offline
10   */
11  function testIndexFileIncluded() {
12
13  /**
14   *        Asserts that we can edit a search environment
15   */
16  function testEditSearchEnvironment() {
17
18  /**
19   *        Asserts that we can use various url forms for the search environment
20   */
21  function testEditSearchEnvironmentURLs() {
22
23  /**
24   *        Asserts that we can edit a search environment
25   */
26  function testCloneSearchEnvironment() {
27
28  /**
29   *        Asserts that we can edit a search environment
30   */
31  function testCreateNewSearchEnvironment() {
```

Listing 9: All the test signatures concerning the search environments

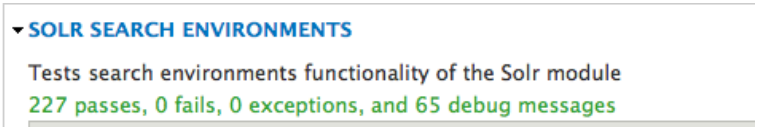


Figure 5.3: In total 227 tests assesments were written in 6 functional tests for the search environments, all of them pass

Exportable The environments are also exportable to code so that is becomes easier for deployment [11] to another server-environment. Ctools[12] is utilized to help enabling the export of this configuration.

5.2.2 Search pages

UI Improvements Also the Search Pages obtained a massive change UI wise. All the functionality is now merged in to one solid landing page for search pages. The core search page is now a part of the search pages list, whereas before it was a separate process and also separate code. Every search page has a title, a path, a designated environment where it should send its requests to and naturally an edit link, a clone link and a delete link where it is allowed. It is by default not allowed to remove the core search page since this would break the whole process. This breakage is due to the dependency on the Search module within Drupal Core. More about that in the next paragraph.

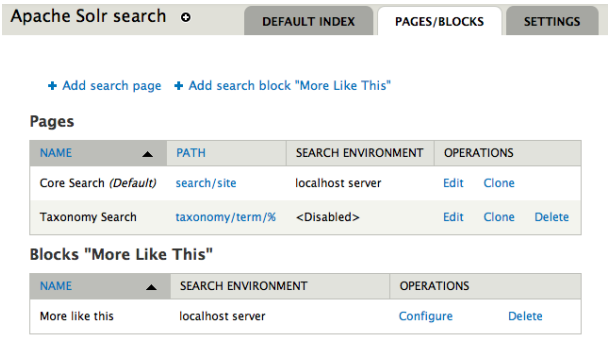


Figure 5.4: Search pages overview page

[11] View more about Deployment and Drupal at `http://drupalize.me/videos/introduction-drupal-features-module`

[12] Ctools is a module that is available on `http://drupal.org/project/ctools` and is primarily a set of APIs and tools to improve the developer experience

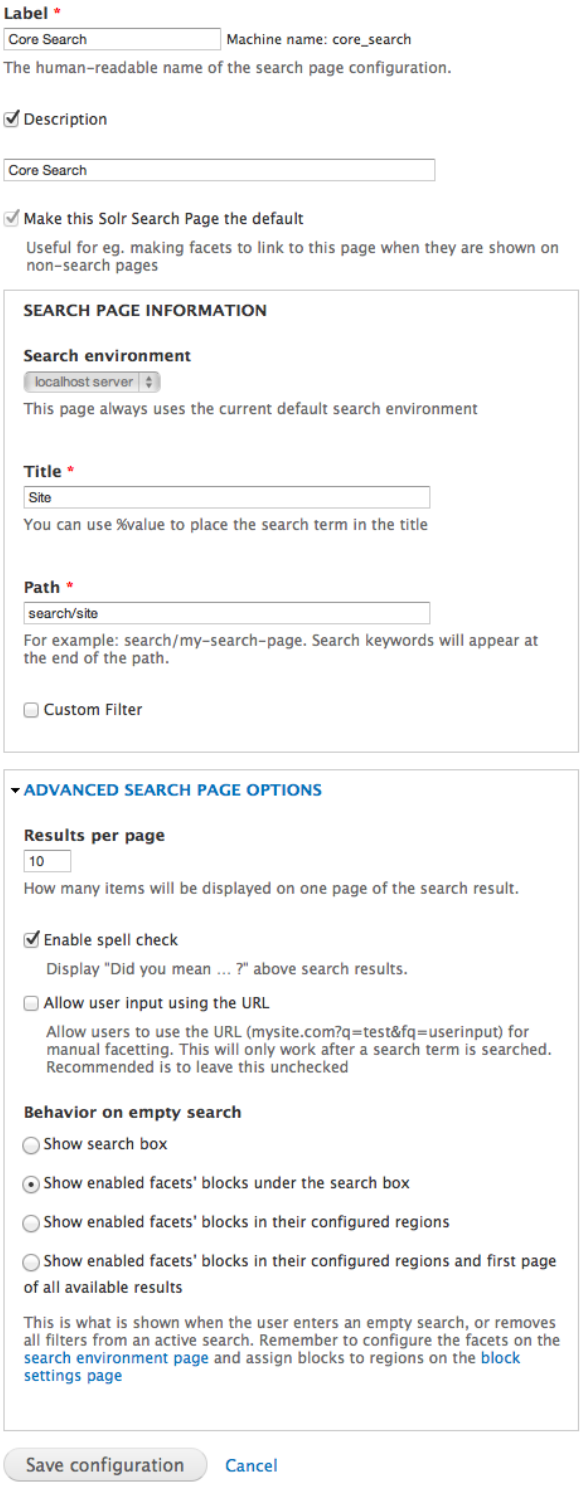


Figure 5.5: Search Page Configuration : Label and Description

Edit a Search Page First and foremost a search page needs a label and optionally a description. A compliant machine name will be automatically generated from the label. Another option that is critical is the question to the site creator if the search page, that is being shown, should be the default search page. This has implications such as to which search page the search block should forward its requests to.

Search Page Configuration : Information What follows is the selection of the search environment. As explained earlier a search environment is a place where the queries can be sent to. Each Drupal site can have multiple search environments if needed. By default it will select the default search environment in the search page configuration. The title field allows the site creator to have a dynamic search title whenever a search is being executed. For example, if a user searches for "Foo", the title of the search results will be : Search results for "Foo". To generate this the site creator should fill in the configuration with "Search results for %value". Next up is the path, a critical part of the search page since this will be the value that is shown in the URL. For the adventurous site creator there is an option to supply a custom filter. This means that when a search page is created to search within blogs only, the custom

filter should have the value "bundle:blog" [13]. This allows deep customizations to any search page and allows site creators to fine tune their site. Multiple filters can be entered if they are separated with a comma.

Search Page Configuration : Advanced options Most of the settings here as self-explanatory. For instance, one can set the amount of search results per page or enable the spellchecker. It also allows the site creator to allow user input from the URL (`mysite.com?q=test&fq=userinput`) for manual faceted search. This will only work after a search term was entered. The recommended course here is to leave this unchecked unless one really knows what he/she is doing. The behavior on empty search setting allows the site creator to choose what to do when an empty search is being executed. Figure 5.6 and Figure 5.7 show the impact of these settings for the end user.

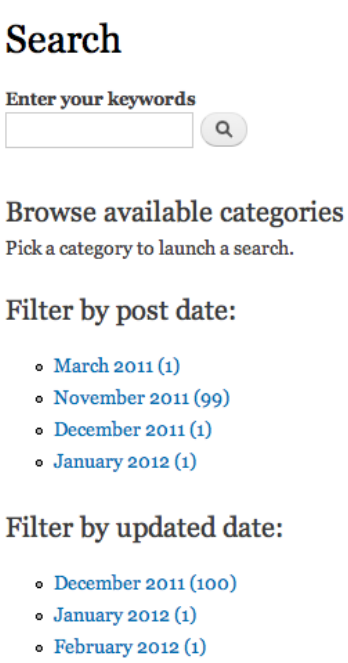


Figure 5.6: Setting : Show enabled facets' blocks under the search box

Functions One of the major issues that was resolved is #1314406 [14] which made it possible that the core search page is approached in the same was as any custom search page. Listed below are all the

```
1 <?php
2
3 /**
4  * Menu callback for the overview page showing custom search pages and blocks.
5  * @return array $build
6  */
```

[13] Granted that the content type has a machine name "blog"

[14] `http://drupal.org/node/1314406` by Nick_vh, scor: Fixed De-duplication of the apachesolr_search_execute() and apachesolr_search_user_defined_search_page().

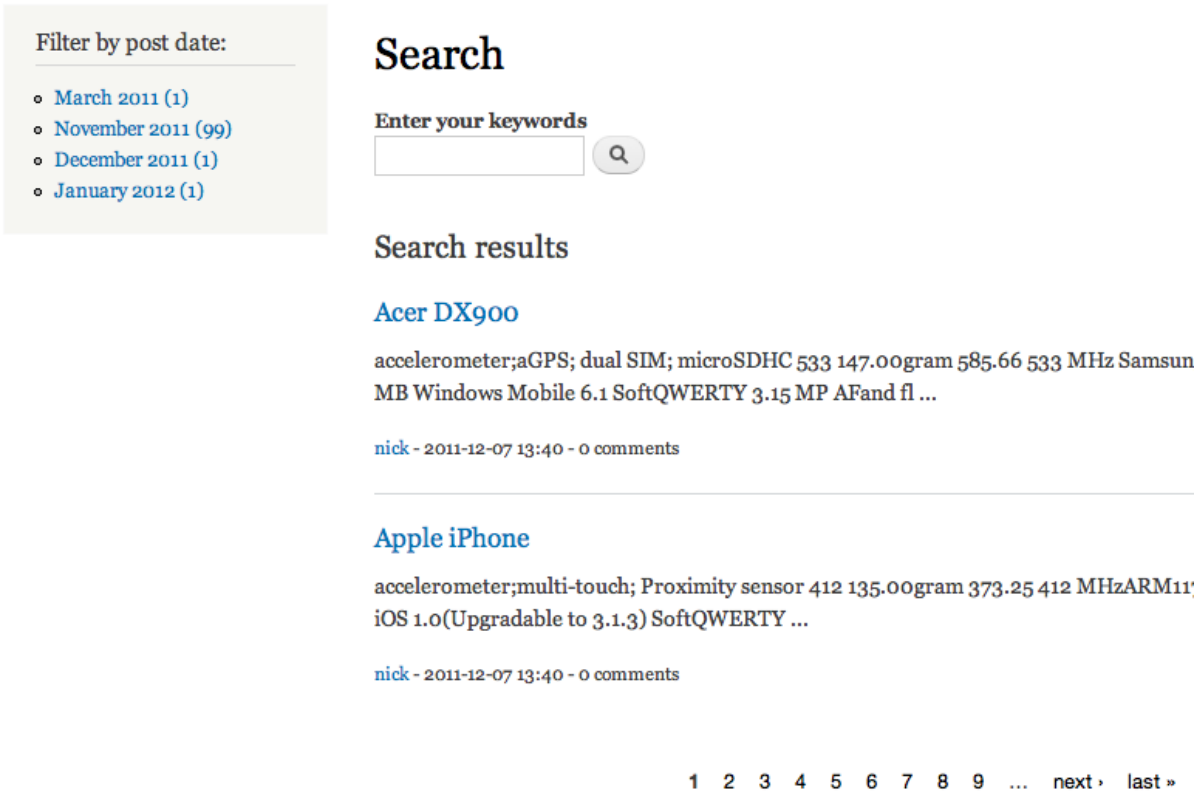


Figure 5.7: Setting : Show enabled facets' blocks in their configured regions and first page of all available results. As one can see it looks like a real search, but looking closer to the search box there is no search keyword entered.

```
7  function apachesolr_search_page_list_all() {
8
9  /**
10  * Listing of all the search pages
11  * @return array $build
12  */
13 function apachesolr_search_page_list_pages() {
14
15 /**
16  * Listing of all the search blocks
17  * @return array $build
18  */
19 function apachesolr_search_page_list_blocks() {
20
21 /**
22  * Menu callback/form-builder for the form to create or edit a search page.
23  * This function signature also involves a validate and submit functions, but
24  * are not shown in this document.
25  */
26 function apachesolr_search_page_settings_form($form, &$form_state, $search_page = NULL) {
27
28 /**
29  * Callback element needs only select the portion of the form to be updated.
30  * Since #ajax['callback'] return can be HTML or a renderable array (or an
31  * array of commands), we can just return a piece of the form.
32  */
33 function apachesolr_search_ajax_search_page_default($form, $form_state, $search_page = NULL) {
34
35 /**
36  * Used as a callback function to generate a title for the taxonomy term
37  * depending on the input in the configuration screen
38  * @param integer $search_page_id
39  * @param integer $value
40  * @return String
41  */
42 function apachesolr_search_get_taxonomy_term_title($search_page_id = NULL, $value = NULL) {
43
44 /**
45  * Used as a callback function to generate a title for a user name depending
46  * on the input in the configuration screen
47  * @param integer $search_page_id
48  * @param integer $value
49  * @return String
50  */
51 function apachesolr_search_get_user_title($search_page_id = NULL, $value = NULL) {
52
```

```
53  /**
54   * Used as a callback function to generate a title for a node/page depending
55   * on the input in the configuration screen
56   * @param integer £search_page_id
57   * @param integer £value
58   * @return String
59   */
60  function apachesolr_search_get_value_title($search_page_id = NULL, $value = NULL) {
61
62  /**
63   * Get or set the default search page id for the current page.
64   */
65  function apachesolr_search_default_search_page($page_id = NULL) {
66
67  /**
68   * Load a search page
69   * @param string £page_id
70   * @return array
71   */
72  function apachesolr_search_page_load($page_id) {
73
74  /**
75   * Save a search page
76   * @param stdObject £search_page
77   */
78  function apachesolr_search_page_save($search_page) {
79
80   /**
81   * Clone a search page
82   * @param £page_id
83   *   The page identifier it needs to clone.
84   */
85  function apachesolr_search_page_clone($page_id) {
86
87  /**
88   * Return all the saved search pages
89   * @return array £search_pages
90   *   Array of all search pages
91   */
92  function apachesolr_search_load_all_search_pages() {
93
94
95  /**
96   * Implements hook_search_page().
97   * @param £results
98   *   The results that came from apache solr
```

```
99   */
100  function apachesolr_search_search_page($results) {
101
102  /**
103   * Mimics apachesolr_search_search_page() but is used for custom search pages
104   * We prefer to keep them seperate so we are not dependent from core search
105   * when someone tries to disable the core search
106   * @param £results
107   *   The results that came from apache solr
108   * @param £build
109   *   the build array from where this function was called. Good to append output
110   *   to the build array
111   * @param £search_page
112   *   the search page that is requesting an output
113   */
114  function apachesolr_search_search_page_custom($results, $search_page, $build = array()) {
115
116  /**
117   * Executes search depending on the conditions given.
118   * See apachesolr_search.pages.inc for another use of this function
119   */
120  function apachesolr_search_search_results($keys = NULL, $conditions = NULL, $search_page = NULL) {
121
122  /**
123   * Handle browse results for empty searches.
124   */
125  function apachesolr_search_page_browse($empty_search_behavior, $env_id) {
126
127  /**
128   * Returns whether a search page exists.
129   */
130  function apachesolr_search_page_exists($search_page_id) {
```

Listing 10: All the function signatures that involve the search pages

Testing For the search environments tests were written as functional tests. As explained earlier these test try to assess the functionality from an UI point of view.

```
1  <?php
2  /**
3   *        Asserts that we create a new search page and remove it again
4   */
5  function testNewAndRemoveSearchPage() {
6    // Create a new search page
7    $this->drupalLogin($this->admin_user);
```

```
8     $this->drupalGet('admin/config/search/apachesolr/search-pages');
9     $this->assertText(t('Add search page'), t('Create new search page link is available'));
10    $this->clickLink(t('Add search page'));
11    $this->assertText(t('The human-readable name of the search page configuration.'), t('Search page creation page succ
12    $edit = array(
13      'page_id' => 'solr_testingsuite',
14      'env_id' => 'solr',
15      'label' => 'Test Search Page',
16      'description' => 'Test Description',
17      'page_title' => 'Test Title',
18      'search_path' => 'search/searchdifferentpath',
19    );
20    $this->drupalPost($this->getUrl(), $edit, t('Save configuration'));
21    $this->assertResponse(200);
22    // Make sure the menu is recognized
23    drupal_static_reset('apachesolr_search_page_load');
24    menu_cache_clear_all();
25    menu_rebuild();
26    $this->drupalGet('admin/config/search/apachesolr/search-pages');
27    $this->assertText(t('Test Search Page'), t('Search Page was succesfully created'));
28
29    // Remove the same environment
30    $this->clickLink(t('Delete'));
31    $this->assertText(t('search page configuration will be deleted.This action cannot be undone.'), t('Delete confirmat
32    $this->drupalPost($this->getUrl(), array(), t('Delete page'));
33    $this->assertResponse(200);
34    drupal_static_reset('apachesolr_search_page_load');
35    $this->drupalGet('admin/config/search/apachesolr/search-pages');
36    $this->assertNoText(t('Test Search Page'), t('Search Environment was succesfully deleted'));
37  }
```

Listing 11: Example of a search environment test

```
1   <?php
2
3   /**
4      *        Checks if the core search page is available
5      *    when the module is installed
6      */
7     function testCheckCoreSearchPage() {
8
9    /**
10     *        Asserts that we can edit a search page
11     */
12    function testEditSearchPage() {
13
14  /**
```

```
15     *        Asserts that we can clone a search environment
16     */
17    function testCloneSearchPage() {
18
19    /**
20     *        Asserts that we create a new search page and remove it again
21     */
22    function testNewAndRemoveSearchPage() {
```

Listing 12: All the test signatures concerning the search environments

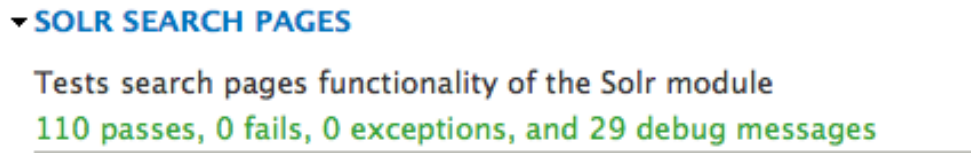


Figure 5.8: In total 110 tests assesments were written in 4 functional tests for the search environments, all of them pass

Exportable Similarly to the Search Environments all search pages are exportable. See Search Environments for a deeper understanding.

5.2.3 Query Object

This class allows you to make operations on a query that will be sent to Apache Solr. methods such as adding and removing sorts, remove and replace parameters, adding and removing filters, getters and setters for various parameters and more. During the timeframe of this work not a lot of changes were made to this class but some are worthy to mention.

User input validation The attached snippet takes care of a half-restrictive validation of user input. This snippet was written by thorough testing and reading up on the essence of regular expression. Gratitude goes out to all that try to explain regular expressions online and to people that helped testing it [15]http://drupal.org/node/1313698 by Nick_vh, denikin: Fixed Support for search of multiword content in facets/fields.. The problem was that a filter as complex as "`fq={!cache=falsecost=5}inStock:true&fq={!frangel=1u=4cache=falsecost=50}sqrt(popularity)`" but also a filter as simple as "`fq={!bundle:(articleORpage)}`" should be validated properly

[15]

The function was divided in 4 main parts and the function is also included for closer inspection to those that are interested.

1. Breaking up the string in to separate parts. The different parts are defined as ”name” and ”value”.
2. Validates the name and the value of the filter
3. Validates opening and closing brackets
4. Validate date value syntax if it is a date.

```
1  <?php
2  /**
3   * Make sure our query matches the pattern name:value or name:"value"
4   * Make sure that if we are ranges we use name:[ AND ]
5   * allowed inputs :
6   * a. bundle:article
7   * b. date:[1970-12-31T23:59:59Z TO NOW]
8   * Split the text in 4 different parts
9   * 1. name, eg.: bundle or date
10  * 2. The first opening bracket (or nothing), eg.: [
11  * 3. The value of the field, eg. article or 1970-12-31T23:59:59Z TO NOW
12  * 4. The last closing bracket, eg.: ]
13  * @param string $filter
14  *   The filter to validate
15  * @return boolean
16  */
17 public static function validFilterValue($filter) {
18   $opening = 0; $closing = 0; $name = NULL; $value = NULL;
19   if (preg_match('/(?P<name>[^:]+):(?P<value>.+)?$/', $filter, $matches)) {
20     foreach ($matches as $match_id => $match) {
21       switch($match_id) {
22         case 'name' :
23           $name = $match;
24           break;
25         case 'value' :
26           $value = $match;
27           break;
28       }
29     }
30     // For the name we allow any character that fits between the A-Z0-9 range and
31     // any alternative for this in other languages. No special characters allowed
32     if (!preg_match('/^[a-zA-Z0-9_\x7f-\xff]+$/', $name)) {
33       return FALSE;
34     }
```

```
35      // For the value we allow anything that is UTF8
36      if (!drupal_validate_utf8($value)) {
37        return FALSE;
38      }
39      // Check our bracket count. If it does not match it is also not valid
40      $valid_brackets = TRUE;
41      $brackets['opening']['{'] = substr_count($value, '{');
42      $brackets['closing']['}'] = substr_count($value, '}');
43      $valid_brackets = ($brackets['opening']['{'] != $brackets['closing']['}']) ? FALSE : TRUE;
44      $brackets['opening']['['] = substr_count($value, '[');
45      $brackets['closing'][']'] = substr_count($value, ']');
46      $valid_brackets = ($brackets['opening']['['] != $brackets['closing'][']']) ? FALSE : TRUE;
47      $brackets['opening']['('] = substr_count($value, '(');
48      $brackets['closing'][')'] = substr_count($value, ')');
49      $valid_brackets = ($brackets['opening']['('] != $brackets['closing'][')']) ? FALSE : TRUE;
50      if (!$valid_brackets) {
51        return FALSE;
52      }
53
54      // Check the date field inputs
55      if (preg_match('/\[(.+) TO (.+)\]$/', $value, $datefields)) {
56        // Only Allow a value in the form of
57        // http://lucene.apache.org/solr/api/org/apache/solr/schema/DateField.html
58        // http://lucene.apache.org/solr/api/org/apache/solr/util/DateMathParser.html
59        // http://wiki.apache.org/solr/SolrQuerySyntax
60        // 1976-03-06T23:59:59.999Z (valid)
61        // * (valid)
62        // 1995-12-31T23:59:59.999Z (valid)
63        // 2007-03-06T00:00:00Z (valid)
64        // NOW-1YEAR/DAY (valid)
65        // NOW/DAY+1DAY (valid)
66        // 1976-03-06T23:59:59.999Z (valid)
67        // 1976-03-06T23:59:59.999Z+1YEAR (valid)
68        // 1976-03-06T23:59:59.999Z/YEAR (valid)
69        // 1976-03-06T23:59:59.999Z (valid)
70        // 1976-03-06T23::59::59.999Z (invalid)
71        if (!empty($datefields[1]) && !empty($datefields[2])) {
72          // Do not check to full value, only the splitted ones
73          unset($datefields[0]);
74          // Check if both matches are valid datefields
75          foreach ($datefields as $datefield) {
76            if (!preg_match('/(\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:[\d\.]{2,6}Z(\S)*)|(^([A-Z\*]+)(\A-Z0-9\+\-\/)*)/', $datef
77              return FALSE;
78            }
79          }
80        }
```

```
81      }
82    }
83    return TRUE;
84  }
```

Listing 13: Validating user input. Custom filters in search pages or user input from URL

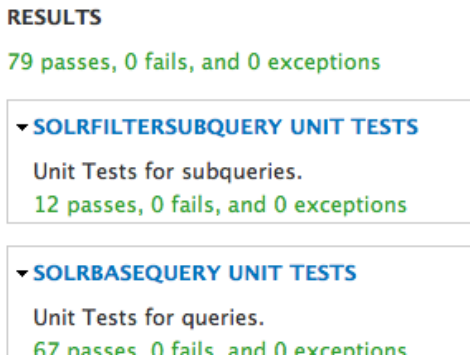


Figure 5.9: In total 79 test assesments were written in 3 unit tests for the search environments, all of them pass

Testing Naturally, functionality like this should be tested so mistakes like these do not occur again. To do this a list was set up with good and bad queries.

Good Combinations

- !cache=falseinStock:true']
- !frange l=1 u=4 cache=falsesqrt(popularity)']
- !cache=false cost=5inStock:true']
- !tag=impalamodel:Impala']
- !anything that appears to be local']
- bundle:(article OR page)']
- -bundle:(article OR page)']
- -!anything that appears to be local']
- title:"double words"']
- field_date:[1970-12-31T23:59:59Z TO NOW]']

Bad combinations

- wrong name:"double words"']
- field_date:[1970-12-31 TO NOW]']
- bundle:((article OR page)]']

These test now all report back as succeeded. The test suite for the base query also includes tests such as parsing the filters, adding and removing the filters, adding search keywords to the query and last but not least the support for subqueries [16].

[16] Subqueries basically allow two query objects to be merged into one

5.2.4 Entity layer

Entity support for Drupal 7 Drupal 7 introduced entities [17]. Originally, in Drupal 6 there was only 1 concept and that was content types. In Drupal 7 this changed, thanks to the Entity Api, and it can now add fields to any entity. A content type is an entity now, but also users and terms. An entity is a useful abstraction to group fields. On top of that we have bundles. Bundles are an implementation of an entity type, similar to a subtype of an entity.

Back when the Apache Solr module was first created, there was no such thing as the entity api and the architecture could also not take this change into account. After some time it became apparent that developers wanted to have this native support to fulfill the needs of their clients. During these months a particular issue [18] in the issue queue got a lot of attention. Over 145 comments were made and a bunch of patches were posted. Ultimately the patch got accepted and it opened up perspectives for new entity support in the Apache Solr module. A set of new API functions became available and some modules like apachesolr_user_indexing [19], apachesolr_commerce [20] and apachesolr_term [21] are already using the new API possibilities.

```
1  <?php
2  /**
3   * Add information to index other entities.
4   * There are some modules in http://drupal.org that can give a good example of
5   * custom entity indexing such as apachesolr_user_indexer, apachesolr_term
6   * @param array $entity_info
7   */
8  function hook_apachesolr_entity_info_alter(&$entity_info) {
9
10 /**
11  * Build the documents before sending them to Solr.
12  * The function is the follow-up for apachesolr_update_index
13  *
14  * @param integer $document_id
15  * @param array $entity
16  * @param string $entity_type
17  */
18 function hook_apachesolr_index_document_build(ApacheSolrDocument $document, $entity, $entity_type, $env_id) {
19
```

[17] Introduction to entities `http://drupal.org/node/1261744`

[18] `http://drupal.org/node/966796` by Nick_vh, scor, BarisW, wesnick, swentel, LSU_JBob — Crell: Added Separate indexer for multiple entity types.

[19] `http://drupal.org/project/apachesolr_user_indexer`

[20] `http://drupal.org/project/apachesolr_commerce`

[21] `http://drupal.org/project/apachesolr_term`

```
20 /**
21  * Build the documents before sending them to Solr.
22  *
23  * Supports all types of
24  * hook_apachesolr_index_document_build_' . $entity_type($documents[$id], $entity, $env_id);
25  *
26  * The function is the follow-up for apachesolr_update_index but then for
27  * specific entity types
28  *
29  * @param $document
30  * @param $entity
31  * @param $entity_type
32  */
33 function hook_apachesolr_index_document_build_ENTITY_TYPE(ApacheSolrDocument $document, $entity, $env_id) {
```

Listing 14: Api functions to support multiple entities

5.2.5 Performance optimizations

Performance during indexing There were a couple issues open in the queue that go back to 2009. One [22]of them is interesting enough to explain here. Before the problem is explained the reader should be aware that Drupal 7 incorporates a database layer, commonly referred to as DBTNG [23]. The Drupal database layer is built atop PHP's PDO library. PDO provides a unified, object-oriented API for accessing different databases but it does not provide an abstraction for the different dialects of SQL used by different databases.

Exactly this abstraction is strapping us here. MySQL has some flaws that become clear after reading the stackoverflow source [24] about in conditions in MySQL. In summary, whenever a query is done in the form of

```
SELECT FROM * WHERE TYPE IN(SELECT FROM * WHERE TYPE = "BLOG");
```

Listing 15: Example of a problematic query in MySQL

it could produce slow results. DBTNG follows the SQL 1999 standard [25] and this kind of query is absolutely allowed. So whenever a subquery is used to generate the parameters for the

[22] http://drupal.org/node/592522 by Nick_vh, pwolanin — quaoar: Hooks node_type(), taxonomy and user knocks out our database server.

[23] http://drupal.org/developing/api/database

[24] http://stackoverflow.com/questions/3417074/why-would-an-in-condition-be-slower-than-in-sql/3417190#3417190

[25] http://en.wikipedia.org/wiki/SQL:1999

IN set, it could lead to performance problems for MySQL. Most other database backends have resolved this problem already. Since it is known that most servers where Drupal is run are still using MySQL, the module had to be adjusted to solve this problem.

Different query per database type After some research was done we found a way to retrieve the same result set but by executing a different query against MySQL. Unfortunately there was no way to write this type of SQL with the DBTNG layer at the moment of the patch because the update query does not support adding extra JOIN's [26]. Finally it was decided that an extra query for MySQL would be added and that the program would switch depending on the database type the Drupal website was running on.

```
<?php
/**
 * £table = the table where the indexable nodes are located
 * £account = the account of the user that is indexing
 */
switch (db_driver()) {
  case 'mysql' :
    $table = db_escape_table($table);
    $query = "UPDATE {{$table}} asn
      INNER JOIN {node} n ON asn.entity_id = n.nid SET asn.changed = :changed
      WHERE n.uid = :uid";
    $result = db_query($query, array(':changed' => REQUEST_TIME,
      ':uid' => $account->uid,
     ));
     break;
   default :
     $nids = db_select('node')
       ->fields('node', array('nid'))
       ->where("uid = :uid", array(':uid' => $account->uid));
     $update = db_update($table)
       ->condition('nid', $nids, 'IN')
```

[26] http://api.drupal.org/api/drupal/includes!database!database.inc/function/db_update/7#comment-15459

```
        ->fields(array('changed' => REQUEST_TIME))
        ->execute();
}
```

Listing 16: Switching between database types

5.3 Facet Api module for Drupal 7 version 7.x-1.0

Facet Api As explained in Chapter 4 there was also some work that was assigned to the author to get a better integration with Facet API. Most of the work was already done by Cpliakas so unlimited gratefulness for him because the module works wonderfully. There were more problems solved and if you want to read all the suggestion is to read the changelog.txt of the module for full details.

5.3.1 Facet Query Types

A query type is in essence a way how Facet API connects with its backend module to provide him data about the facets. Apache Solr has 3 query types

- Query Type for Terms
- Query Type for Dates
- Query Type for Numeric Ranges

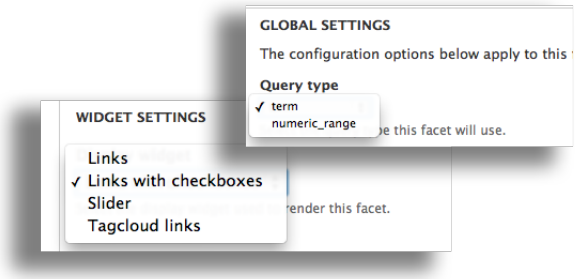


Figure 5.10: Setting : Query type

Each of those already existed in the Apache Solr Module. They might have received a small refinement but in essence they were there. As seen in our class diagram in chapter 4 Facet API has a concept of widgets. Widgets are a visual way in how data is presented to the user. The problem existed in the fact that 1 facet could obtain different widgets and as a consequence it should also be able to switch query types. For example, a list of prices (Terms) switches its widget to a slider (Numeric Range). This example was not possible because each field was mapped to only 1 query type.

This was solved by making a one-to-many relationship with query types and their facets. Progress was tracked in an issue [27] Notice that in the example the "query type" parameter is still present. This is due to the fact that backwards compatibility was important at that stage. Site creators that implement facet can choose either one of them.

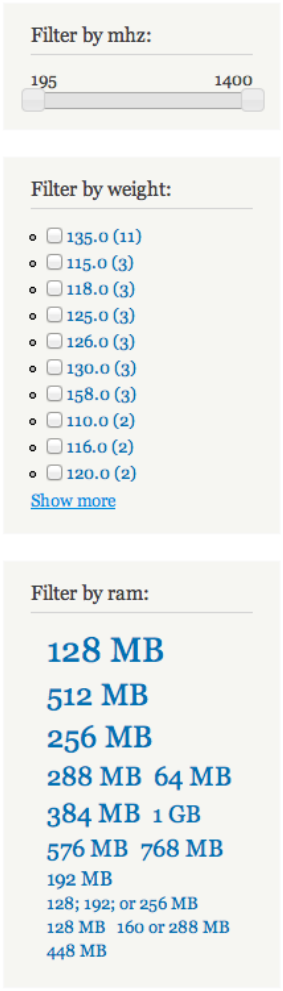


Figure 5.11: Different widgets in the user interface

```php
<?php
// Before
'list_integer' => array(
    'indexing_callback' => 'apachesolr_fields_default_indexing_ca
    'map callback' => 'apachesolr_fields_list_facet_map_callback'
    'index_type' => 'integer',
    'facets' => TRUE,
    'query type' => 'term',
    'facet missing allowed' => TRUE,
  ),

// After
'list_integer' => array(
    'indexing_callback' => 'apachesolr_fields_default_indexing_ca
    'map callback' => 'apachesolr_fields_list_facet_map_callback'
    'index_type' => 'integer',
    'facets' => TRUE,
    'query types' => array('term', 'numeric_range'),
    'query type' => 'term',
    'facet missing allowed' => TRUE,
  ),
```

Listing 17: A before and after view of the query types key

[27] `http://drupal.org/node/1161434` by Nick_vh, cpliakas: Modify 'query type" key in facet definition to accept an array.

5.4 Backporting Facet API And Apache Solr to Drupal 6

Backporting a module from Drupal 7 to Drupal 6 sounds easy but it is not. A programmer is used to all the new tools that are offered to him/her such as the new database layer, some of the Drupal 7 API changes (no more superfunctions [28]) and most importantly the better theming layer. All progress was also tracked publicly in issue [#1387164] [29] for Facet API and in [#1387628] [30] In this small chapter it tries to explain what was learned from this procedure.

DBTNG to MySQL Converting DBTNG was easy at first because there is a wonderful little helper module called DBTNG for Drupal 6. It is a backport of the Drupal 7 PDO database compatibility layer. [31] When everything runs you can start to backport the DBTNG queries to regular MySQL queries. An easy debug function exists in Drupal 7, namely dpq() [32] that prints out a SQL version of the query object.

Autoload Facet Api extensively uses classes and also the autoload functionality of Drupal 7. Drupal 6 does not have this functionality so again a helper module was installed for temporary purposes. This helper module is easily enough called "Autoload" [33] and takes care of the lazy loading of classes for you. near the end of february there was a small attempt to get rid of this dependency but there are just too many dynamic classes that it would be better if Autoload stayed. This might be revisited in the near future if Cpliakas decides it should not depend on Autoload.

Static Variables Drupal 7 has a concept of dynamic static variables using the drupal_static method [34]. This drupal_static are statics that are attached to the function instead of the whole project. In Drupal 6 we do not have this concept so in Facet Api the function ctools_static is used, that mimics this behavior. [35].

[28] Superfunctions are functions that try to do it all, in Drupal 7 most of those were replaced with separate functions, hence the big increase in amount of functions

[29] `http://drupal.org/node/1387164` by Nick_vh, gnucifer — cpliakas: Backport Facet API to Drupal 6.

[30] `http://drupal.org/node/1387628` by Nick_vh: Backport 7.x-1.x to 6.x-3.x.

[31] `http://drupal.org/project/dbtng`

[32] `http://api.drupal.org/api/devel/devel.module/function/dpq/7`

[33] `http://drupal.org/project/autoload`

[34] `http://api.drupal.org/api/drupal/includes!bootstrap.inc/function/drupal_static/7`

[35] ctools_static is part of the ctools module

Theming layer / Drupal Render Drupal 6 handles output very different. While in Drupal 7 everything renders at the end of a complete load, Drupal 6 renders at the level of a specific function. This change was not easy but finally it was quite easy. In the example can be seen that the Drupal 6 version really tries to mimic as much from Drupal 7 as possible. The reason is that when new patches come out for the Drupal 7 version they can be easily integrated/applied to the Drupal 6 version.

```php
<?php
  // Initializes output with information about which server's setting we are
  // editing, as it is otherwise not transparent to the end user.
  $output['apachesolr_index_action_status'] = array(
    '#prefix' => '<h3>' . t('@environment: Search Index Content', array('@environment'
    '#theme' => 'table',
    '#header' => array('type', 'value'),
    '#rows' => $messages,
  );
  $output[] = //more content in arrays
  return $output;
?>
```

Listing 18: Drupal 7 renderable arrays

```php
<?php
  // Initializes output with information about which server's setting we are
  // editing, as it is otherwise not transparent to the end user.
  // Initializes output with information about which server's setting we are
  // editing, as it is otherwise not transparent to the end user.
  $title = t('@environment: Search Index Content', array('@environment' => $environment
  $output['apachesolr_index_action_status_prefix'] = '<h3>' . $title . '</h3>';
  $output['apachesolr_index_action_status'] = theme('table', array('type', 'value'), $m
    // Print in a similar way as the Drupal 7 version
  $output_print = NULL;
  foreach ($output as $print) {
    $output_print .= $print;
```

```
  }
  return $output_print;
?>
```

Listing 19: Drupal 6 Direct output

Entity to Content Type Doing a backport of the whole entity support to support of only 1 entity type "node" with bundles was challenging. Certainly if most of the API functions don't exist in Drupal 6. This was solved by letting Apache Solr for Drupal 6 depend on the CCK project [36]. By doing this every content type in Drupal 6 got the possibility of adding extra information to it. This allowed the module to add callbacks to specific "bundles" in the same way as Drupal 7 did with entities and bundles. Learning by example is the best so a comparison is included between Drupal 7 and 6 for a function that allows a specific entity_type to reindex. In Drupal 6 there is only 1 entity_type available and that is "node".

```
<?php
function apachesolr_index_mark_for_reindex($env_id, $entity_type = NULL) {
  foreach (entity_get_info() as $type => $entity_info) {
    if (($type == $entity_type) || ($entity_type == NULL)) {
      if ($entity_info['apachesolr']['indexable']) {
        $bundles = apachesolr_get_index_bundles($env_id, $type);
        $reindex_callback = '';
        if (!empty($bundles)) {
          $reindex_callback = apachesolr_entity_get_callback($type, 'reindex callback')
        }
 ...
```

Listing 20: Drupal 7 version for indexing a specific entity type for reindexation.

```
<?php
function apachesolr_index_mark_for_reindex($env_id, $entity_type = 'node') {
  foreach (content_types() as $content_type => $entity_info) {
    if (!empty($entity_info['extra']['apachesolr']['index'])) {
```

[36] Content Construction Kit `http://drupal.org/project/cck`

```
      $reindex_callback = apachesolr_entity_get_callback($entity_type, 'reindex callbac
    }
...
```

Listing 21: Drupal 6 Port of the same function, using content_types. A function from CCK

5.5 Acquia Search Upgrade from 1.4 to 3.x

As clarified earlier, Acquia Search is the hosted search solution of Acquia. They provide first class support to customers that do not want to maintain their Apache Solr servers. They also take care of security and previously an employee of Acquia wrote an extension for Apache Solr 1.4 to add the HMAC authentication. This HMAC authentication was added by using a Java Filter Servlet.

The problem However, Acquia Search needed to keep up with the latest stable version of Apache Solr which is 3.5. When the Java Filter Servlet was applied to Apache Solr 3.5 it did not work. Acquia wanted to hire an external consultant at first to fix the problem but fortunately a solution was found.

5.5.1 Java Filter Servlet

in Solr 1.4, response.getWriter() was used by the SolrDispatchFilter for any character based responses – but in version of Solr 3.4+, because of the issues related to SOLR-2381 [37], SolrDispatchFilter was modified to use response.getOutputStream() for both binary and character based streams.

The filter that was written only had support for the getWriter method so support had to be added for the getOutputStream method. Without going much into detail a servlet was written in a couple of days that supported this. It took a lot of sweat because of unfamiliarity with Java code and more specifically with the Solr Source code but using the issue queue of the Solr project and the help from core Solr developers, who admittedly saw that the author was a beginner in the area [38], a servlet was written that supported this new getOutputStream. One of the biggest aha-moments was to change the ResponseWrapper to inherit the getOutputStream method.

[37] https://issues.apache.org/jira/browse/SOLR-2381
[38] https://issues.apache.org/jira/browse/SOLR-2878

```
response.setCharacterEncoding("UTF-8");
PrintWriter out = response.getWriter();
CharResponseWrapper wrapper = new CharResponseWrapper(
  (HttpServletResponse) response);

  chain.doFilter(request, wrapper);
  String responseBody = wrapper.toString();

  //write the outgoing header
  response.setHeader("Pragma", "hmac_digest=" +
    buildResponseHmac(authenticator, responseBody, s) + ";");

  out.write(responseBody);
  out.close();
```

Listing 22: A snippet of the original Acquia HMAC Filter for Solr 1.4

```
CharResponseWrapper newResponse = new CharResponseWrapper(
    (HttpServletResponse) response);

// CharResponseWrapper is responsible for the getWriter and
// getOutputStream support.

chain.doFilter(request, newResponse);
// The response works with byteArrays so we do to

byte[] responseBody = newResponse.getBytes();
// Write the outgoing header with the ByteArray (Performance)

response.addHeader("pragma", "auth=" +
    buildAuthentication(responseBody) + ";");
// Force the encoding to UTF-8 since we know Drupal works with UTF-8
response.setCharacterEncoding("UTF-8");
response.getOutputStream().write(responseBody);
```

```
response.getOutputStream().flush();
```

Listing 23: A snippet of the Acquia HMAC Filter for Solr 3.5. The buildAuthentication returns the value that is embedded in the HTTP headers so the client can verify the validity of the response

5.5.2 Performance testing

Merge Policies Drupal is an application that has very deep integration with the Apache Solr application and is updating Solr during cron runs (every 30 minutes for example). This does imply that the indexing speed should not be very high but the search speed should be. Apache Solr has a concept of segments (your index is spread over multiple segments) and if a search is executed it needs to gather all these segments and search them. Logically, more segments = slower results. Solr 1.4 already had some MergePolicy's such as LogByteSizeMergePolicy [39] and LogDocMergePolicy [40]. Solr 3.5 came with a new default MergePolicy and that required some testing to see if this new MergePolicy (TieredMergePolicy [41]) could be trusted and what effect it has on existing Drupal indexes. More information regarding these merge policies can be found online at `http://java.dzone.com/news/merge-policy-internals-solr`.

Summary of the testing procedure

1. Load existing index files in to a new core.
2. Extract Documents from this index
3. Use the extracted documents to insert them in a clean and new core with different configuration
4. Re-run the access log of that subscription for the searches, repeat this twice, use 10000 queries per access log and discard everything except the select queries and repeat this process 3 times to make sure we have a balanced result set

Charts and extra legend information

- S14 stands for Solr 1.4

[39] `http://lucene.apache.org/java/3_2_0/api/all/org/apache/lucene/index/LogByteSizeMergePolicy.html`

[40] `http://lucene.apache.org/java/3_2_0/api/all/org/apache/lucene/index/LogDocMergePolicy.html`

[41] `http://lucene.apache.org/java/3_2_0/api/all/org/apache/lucene/index/TieredMergePolicy.html`

- S35 stands for Solr 3.5
- LB stands for Load Balancer
- SL stands for Slave, this means that the attack happened from the LB to the SL (these results happened 3 times for to contain less variable delays)
- MA stands for Master, this means that the attack happened from the LB to the MA (these results happened 3 times for to contain less variable delays)
- MergeFactor for LogbyteMerge and LogDocMerge is set to 4
- Default means the Default merge policy, Solr 1.4 this is LogByteMergePolicy and for Solr 3.5 this depends on the LuceneMatchVersion
- L35 means that Lucene has been set to Lucene 3.5 instead of the default
- When Lucene 3.5 is set for Solr 3.5 and no merge policy was set, this defaults to TieredMergePolicy
- When Settings is defined, it applies to specific TieredMergePolicy settings
    1. maxMergeAtOnce says how many segments can be merged at a time for "normal" (not optimize) merging
    2. segmentsPerTier controls how many segments you can tolerate in the index (bigger number means more segments)

Specifications of the test machines

Specifications of the Master

- Large Instance
- 7.5 GB memory
- 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each)

Specifications of the Slave

- High-CPU Medium Instance
- 1.7 GB of memory
- 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each)

Results

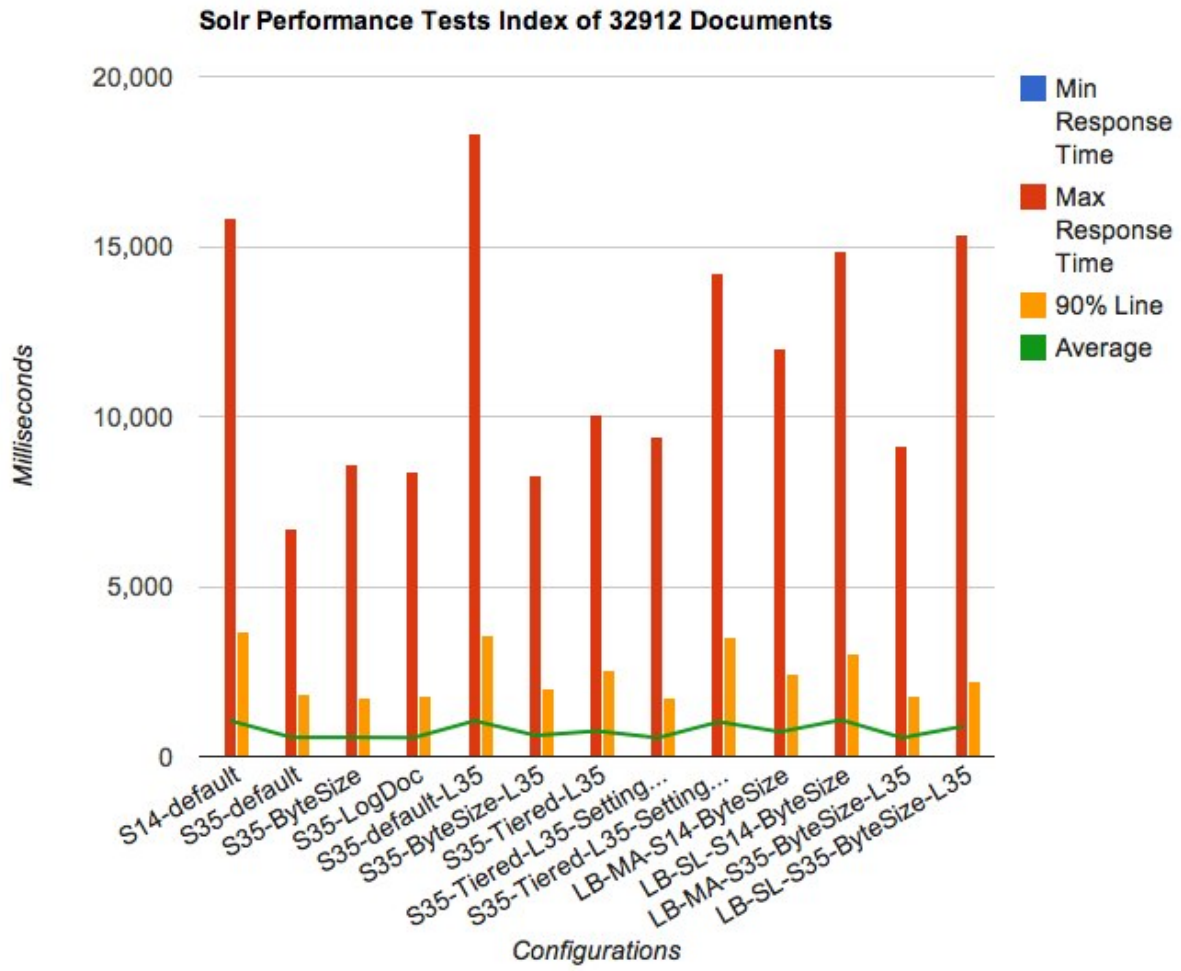


Figure 5.12: Speed of quering after a indexing a site with 32912 Documents

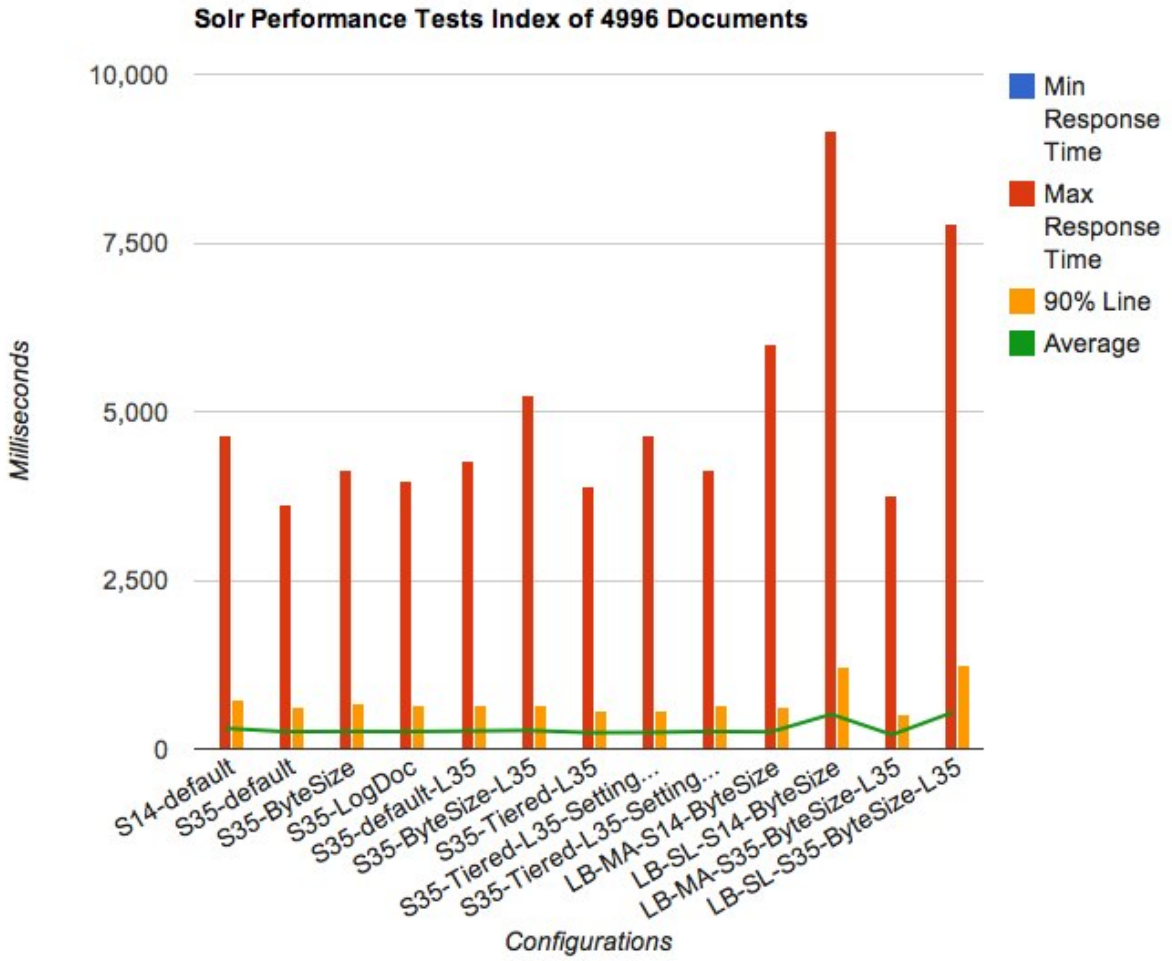


Figure 5.13: Speed of quering after a indexing a site with 4996 Documents

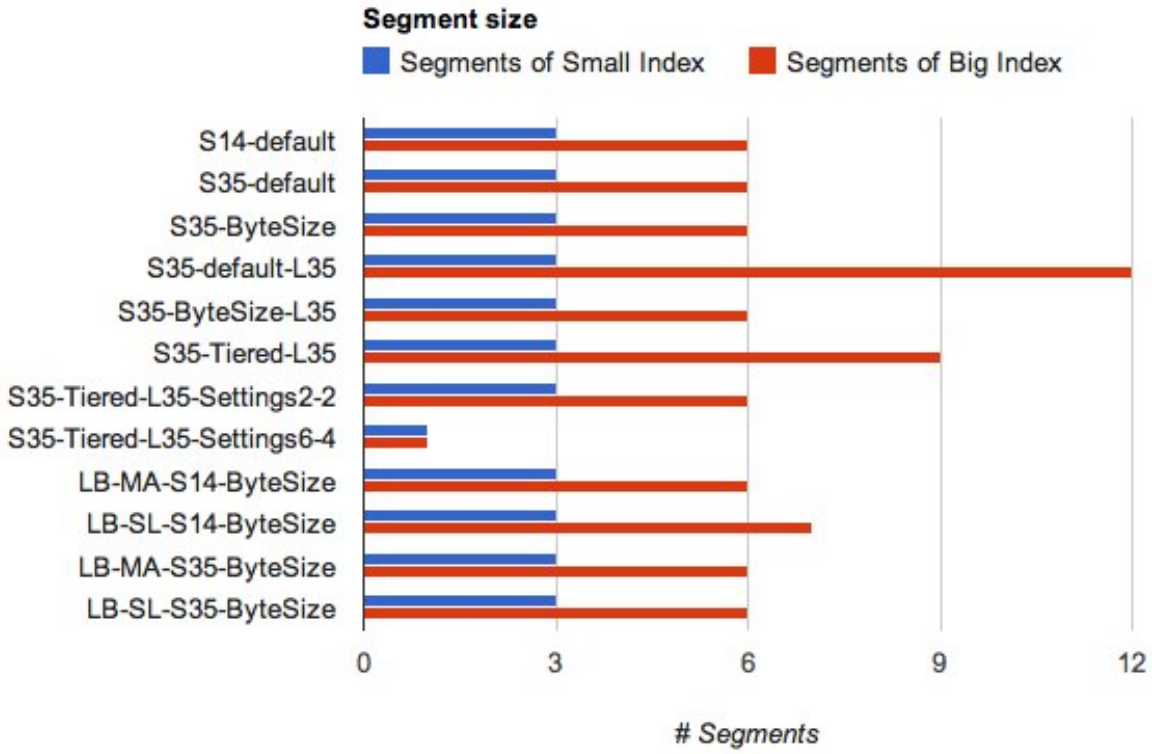


Figure 5.14: Size of the segments for all the different test results

Conclusions If one wants to migrate to Solr 3.5 coming from Solr 1.4 with low risk of changes you should keep using the LogByteMergePolicy with a merge-factor of 4 (Default in the Drupal configs). However, the TieredMergePolicy is interesting when understood correctly. For a better understanding more investigation should be done.

The big outcome of this test is that Solr 3.5 versus 1.4 is a big performance win. Also it is good to know that the MergePolicy should be set explicitly when using LuceneMatchVersion in the solrconfig.xml. This exact conclusion and result is also publicly available on the blog of the Author at `http://nickveenhof.be/blog/upgrading-apache-solr-14-35-and-its-implications`.

5.6 Additional Modules created to empower users to use the Apache Solr Module suite

5.6.1 Facet Api Slider

Use and reason The Facetapi Slider [42] was created to make maximum use of the range query type. It allows site creators to easily switch the UI from a list of numeric values to a slider where one can set the minimum and the maximum and the search will be filtered within this range.

Creating it was not very easy since we had to keep the minimum and the maximum for the global set and also the values that were set by the user. Creating a usable experience proved difficult and there are still discussions [43] ongoing how to improve this process.

Figure 5.15: A real life use case of the Facet Api Slider, implemented by Dataflow `http://dataflow.be/`

5.6.2 Apache Solr Term

The Apache Solr Term module [44] provides basic indexing of the taxonomy terms. It makes use of the new entity indexer that was pushed in to the Apache Solr module and allows site creators to index terms, with attached fields. But moreover, users can search for terms.

5.6.3 Apache Solr Commerce

The Apache Solr Term module [45] provides basic indexing of commerce entity types. It makes use of the new entity indexer that was pushed in to the Apache Solr module and allows site creators to index items that are for sale, including their price, with attached fields.

[42] `http://drupal.org/project/facetapi_slider`
[43] `http://drupal.org/project/issues/facetapi_slider`
[44] `http://drupal.org/project/apachesolr_term`
[45] `http://drupal.org/project/apachesolr_commerce`

When combined with the Facetapi Slider it makes up for a powerful experience in a webshop.

5.6.4 Apache Solr User

The Apache Solr User module [46] provides basic indexing of the user entities. It makes use of the new entity indexer that was pushed in to the Apache Solr module and allows site creators to index user entities, with attached fields. But moreover, users can search for other users and for example, in their profile.

5.6.5 Apache Solr Sort UI

The Apache Solr Sort UI module existed already in an earlier stage, primarily written by drupal_sensei [47] to serve the need of a configurable settings page that allows the site creator to choose which fields are available to sort on and what the weight should be when those are listed.

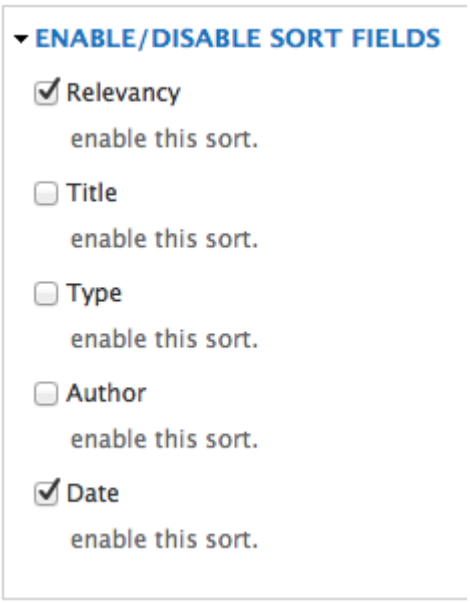


Figure 5.16: Apache Solr Sort UI sort selection

[46] http://drupal.org/project/apachesolr_user
[47] http://drupal.org/user/721548

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

[3] [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_can_be_indexed_security

[4] *https://acquia.com/sites/default/files/blog/DC-Chicago-2011-Solr-Chops-v3.pdf*

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

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

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

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

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

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

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

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

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

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

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

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

[17] Solr Black Belt Preconference
http://www.slideshare.net/erikhatcher/solr-black-belt-preconference

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

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

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

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

[22] 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

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

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

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

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

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

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

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

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

[31] *http://bxl2011.drupaldays.org/sites/default/files/SearchAPIPresentation.pdf*

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

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

[34] 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

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

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

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

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

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

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

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

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

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

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

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

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

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

[48] 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

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

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

http://msdn.microsoft.com/en-us/library/ff648434.aspx

List of Figures