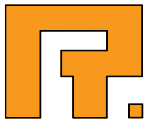


Roxen CMS 5.4

Search Engine Module Manual

2014-03-11
www.roxen.com

Roxen Internet Software
Box 449
SE-581 05 Linköping
Sweden
www.roxen.com

Contents

1 Search Engine Module

1.1 Find Anything on Your Site

The integrated search engine keeps an up-to-date search index of all web pages in your site. It is fully multilingual-aware and is tightly integrated meaning that it knows about metadata which Roxen CMS stores for each file and that edited documents are indexed as soon as they are published.

Another great benefit of integration is transparent security. This is a concept where confidential data are removed from pages that unauthorized visitors access, and in this case from search results.

1.2 Modular and Extensible

Using a modular architecture it possible to extend the search engine for various file formats. As shipped it handles text, XML, HTML, Microsoft Word and PDF files.

Building search forms is just a matter of incorporating an XML tag into a web page. The search form has a query language with boolean expressions, parenthesis, phrase search, word exclusion, wildcards and metadata field matching.

2 What is a Search Engine?

You are probably familiar with using a search engine such as Google in your daily usage of Internet, but how does a search engine really work? At first glance it may look like a database where you insert a query in one end and get a set of results in the other, but there are several significant differences.

To begin with, a database query is always “complete”. If you get an empty result set there was really no matching entry in the database, and if you get 105443 results there was really 105443 matching entries in the database, not 105442 or 105444. Completeness and accuracy is defining characteristics for a database, no matter how long it takes to complete the query.

On the Internet there is no time to let a query run for an undefined time to produce a 100% correct answer. Already after a few hundred milliseconds is the surfer inconvenienced. This is really no big problem, because it typically doesn’t matter for the user if there are 105443 or 800 documents in the database. Already before we reach a hundred documents it is too many hits for an average user to handle.

It doesn’t matter how many documents were found as long as all presented documents are as relevant as possible, sorted by relevance if possible. If taken one step further this means that a presented document might not even contain any of the words in the search query, what matters is whether the document is relevant to the query or not.

A search engine keeps all the words it knows in a specially designed “search database”. When a user enters a search query into the search engine, the search engine rephrases that query into several database queries. If the search engine gets enough relevant matching documents it presents the result to the user, otherwise it can continue to make additional database queries until it either is satisfied or gives up. In this way a strict database can be made behave more helpful.

Since the search engine keeps its words in a local database it will not know about new or changed pages on the Internet. To keep the information up-to-date it launches a “crawler” to fetch new pages. A crawler is like a browser without a page rendering unit. It will just get the raw HTML data from a web server, extract all interesting words and insert them into the search database. The process of automatically fetching documents from a web server is called crawling, the process of removing everything that isn’t words is called filtering, and the process of extracting the interesting words and inserting them into the database is called indexing. Indexing is however often also used as term for the whole process of fetching and inserting data from a document.

3 Query Syntax

The query is basically a list of words that you are searching for. Separate the words with spaces. There are two types of search modes, “implicit or” and “implicit and”. In the “implicit or” mode, it is sufficient that some of the words are present for a document to be considered a hit. However, the more of the words that are present in the page, the higher its score, and of course it is not a hit if none of the words are present. In the “implicit and” mode, all of the words in the query must be present in the document.

If the words in the document appear in the same order as in your query, and close to each other in the text, then the score of the document gets higher than if the words appear randomly scattered in the document.

The search engine doesn’t distinguish lowercase from uppercase so you don’t have to think about capitalization.

3.1 Phrase Search

If you write text inside quotation marks, the search engine is instructed to look for phrases and not just single words. So if you wish to search for the phrase “content management”, you write:

```
"content management"
```

Then, only documents with the two words “content” and “management” present, and in sequence, will be returned. Note that since the query disregards from interpunctuation and case, you may get a hit on a sentence border, e.g. “I am content. Management is needed.”.

3.2 Word Exclusion

Sometimes you want to search for documents that contain some specific words, but only if they don’t contain another word. For example you might be looking for documents about Java, not the programming language but rather the island. You would then write:

```
java -programming -computer
```

By preceding a word with a minus sign, you tell the search engine only to look for documents that match the rest of the query, and that don’t contain the word in question. The same goes for phrases, so if you wish to see pages about the Russian author Anton Chechov, but not those that deal with his most famous play, you should enter:

```
anton chechov -"the cherry orchard"
```

3.3 Prioritizing Words

Some words may be more important than others in your query. If the search is done in the “implicit or” mode, then it is not necessary for all words to be present. You can override this behaviour for specific words in your query. If you are looking for documents about Perl programming, but not documents about programming in general, you could enter:

```
+perl programming
```

By preceding a word with a plus sign, you tell the search engine that you are only looking for documents that contain that word. Similarly, you can precede a phrase by a plus sign, marking that is as necessary in your search.

3.4 Searching Fields

The words in a document belong to different fields, depending on where in a document they occur. You can specify a field to search in the search query. You do this by entering the name of the field, followed by a colon. Everything up to the next such field designator is then supposed to be found in that field. If you use the field name “any”, then all fields are searched. This is also the default if you don’t specify any fields. So let’s assume that you are looking for documents about programming and that they must have the words “easy” and “introduction” in their title:

```
programming title: +easy +introduction
```

Fields such as file category can also be used. The category reference is a bit complicated to specify by hand however, so it will help to build a form such as a drop-down menu for this:

```
category:cat.xml!1087894521-1
```

3.5 Date Search

Search results can be limited according to specified dates by using the date keyword with an operator and a date. Valid operators are =, <>, !=, <=, >=, < and >. Date expressions can be combined with logical expressions (see below). Valid dates are (YYYY is year, MM is month and DD is day):

```
YYYY
YYYY-MM
YYYY-MM-DD
YYYYMMDD
today
```

Examples:

```
date< 2010
date = 2000-02-29
date >= 2000-02 AND date <= 2000-05
date = 2000 OR date = 2001
date != 2004
date = today
```

3.6 Logical Expressions

Sometimes it is not sufficient to control just what words should be in the document and what words shouldn't. You can then combine several queries in one by using the logical operators. Say that you want to find documents that contain both the words "banana" "skid", or documents that contains the word "orange" in the title, then you would enter:

```
banana skid OR title: orange
```

A document containing both one of the words "banana" "skid", and the word "orange", will be given a higher score than a document that matches only one of the subqueries.

You can also use the operator AND to tell the search engine just to look for documents that match both subqueries. Both AND and OR can be used for more than two subqueries. If both operators are used, AND takes precedence over OR. You can use parentheses to group terms and change the precedence:

```
(sweden OR norway) AND "folk dance"
```

Note!

The operators can be written in lowercase as well. I have used uppercase just to make them more visible in the examples. If you want to search for the word "and", just write it inside quotes and it will not be recognized as an operator.

4 System Overview

The core component of Roxen Search is a search database that is implemented on top of a MySQL server. MySQL was selected because of its well known stability, wide acceptance and good performance for simple SQL queries. Since MySQL is a well known product there is already plenty of knowledge, books and support available for tasks such as administration, backup and replication.

Note!

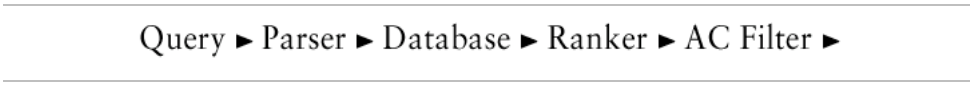
Search queries are not processed directly by MySQL, but on a higher level in Roxen Search, and the data stored in the MySQL tables is in a Roxen Search internal format that can not be processed by MySQL on its own.

The process of inserting new documents into the search database begins with the crawler being started. This can happen for several reasons, but the two major events are that a new or changed file is committed in SiteBuilder (Notification), or that the crawler is scheduled to run. Scheduled runs are needed on sites where all information is not stored in the SiteBuilder repository.

The crawler then fetches the pages slated for indexing and sends them to a filter. The appropriate filter is chosen based on the file type of the fetched file. The filter extracts the text from the file, and tries to maintain the context in which the text was found, e.g. if the text was in a heading or in the text body. All document formats does however not contain such context information, e.g. PDF documents.

After the document has been filtered the indexer takes the text, splits it into words and stores them into the search database. The database format holds, in addition to the word and in what document it can be found, the context it was found in and an order number so that phrase searches can be executed. The whole insertion process can be described as the following diagram.

When a user enters a search query, that search query is sent to the query parser for analysis, where it is rephrased into several subqueries which are then forwarded to the search database. The query results are then ranked according to their relevance to the search query and joined into one big list of search hits. The query response is filtered through an access control filter, removing pages to which the user has no access, and returns the rest to the user. The query process can be idealized into the following diagram.



5 Installation

It is possible to share a Roxen Search database with one or more read only query frontends. For such a setup, see Acceleration Server Setup chapter. If you are running a CMS Advanced site you should already have all the files in their proper locations. If you are running any other configuration, for instance a stand-alone Roxen WebServer site, you should refer to the separate installation guide that came with your Roxen Search package.

Once all your files are installed you can start the server and add the Search SiteBuilder Interface module to your site. The Roxen Search modules are included in the CMS Advanced site template, so if you create a new site from scratch you need not add any extra module to use Roxen Search.

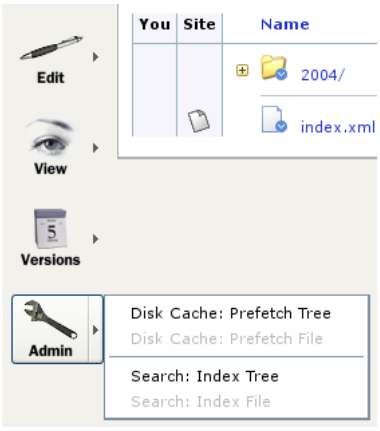
5.1 Initial Settings

The Search SiteBuilder Interface module needs a database in which it can save all settings and profiles so that all other systems can read them. It also needs a database to store its event log. These have to be set before the SiteBuilder interface can be used. Choose any of the available databases in the drop down menu, or define a new database in the database overview. If you have not defined a new database before adding the module, chose the “none” database. Go to the database tab and add a new database. Go back to the Search SiteBuilder Interface module settings and chose the new database in the drop down menu and save your changes by clicking the save button.

5.2 SiteBuilder Interface

The Roxen Search SiteBuilder Interface module will extend the SiteBuilder interface in two ways. First of all it will add a new top level tab, e.g. among the “Files” and “Access Control” tabs, named “Search Engine”. Under this tab you will be able to control the search engine and monitor its logs. In order to see this tab you must have at least read access to the protection point “Plugin: Search”. If you only have read access, and not write access, you will not be able to alter any settings, create or delete any profiles, control any indexer or clear any log.

In addition to the search engine tab two new actions will be defined in the SiteBuilder interface, “Search: Index Tree” and “Search: Index File”. Both of these are available among the admin actions, e.g. under the “Admin” button in a default SiteBuilder interface. These actions will only be available when the user has write access to the protection point “Plugin: Search”.



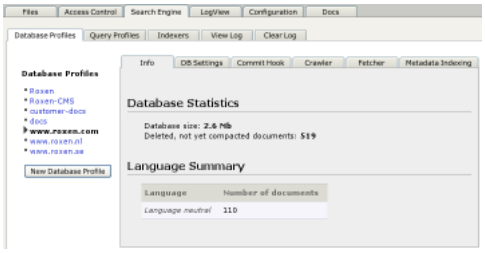
The index file action will add the selected file to the indexer queue in all database profiles that monitors the work area, thus making the file indexed or re-indexed the next time the indexer runs. See information about commit index latency in the “Commit Hook” section of the Database Profiles chapter for more information. Only files that pass through the path filter in the database profile will be indexed.

The index tree action is a recursive version of the index file action that will recurse from the selected directory into all subdirectories adding its files to the indexer queue. As with the index file action only files that passes though the path filter will be indexed.

6 Database Profiles

The database profile contains all information regarding a specific search database, including how information is added to it. In short that means the following information:

- Where to store the information, i.e. in which MySQL database.
- How the SiteBuilder commit hook should behave.
- How the Crawler should behave and how often it should be scheduled to run.
- Which files and file types to index.
- How often the database should be purged of unused information.



Adding a new database profile is as simple as pressing the “New Database Profile” button under the Database Profiles tab in the SiteBuilder interface. You will then be taken to a one step wizard where you will be asked to enter a name and decide if a new database should be created.

The database profile name is used as a label by which you can identify that database profile in the Search SiteBuilder interface, and also in attributes to Roxen Search RXML tags. A short but descriptive name without characters that you do not want in your XML/HTML files is best. Avoiding the characters <, > and & is also a good idea.

By default, a new database is created when you create a new database profile. If you want to use an existing local database or connect to an external database you should uncheck the checkbox. The database will then be set to “none” for that database profile, which must be changed before the database profile can be used.

6.1 Database Information

Under the first tab, labeled Info, can you find some overall statistics for the database of that specific database profile. Listed information includes the number of unique words, the size of the database, the number of deleted documents untouched by the compactor. In addition to this there are two tables, one showing the number of documents indexed in each language and the other shows the top ten most common words in the database.

6.2 Database Settings

Under the DB Settings tab you can configure things related to the whole database profile as well as settings regarding the actual search database.

The Database Maintenance widget controls when the database compactor is scheduled. When a word is no longer used, either because the document it was written in was revised, or the document was removed altogether, that word is marked as deleted. It will still remain in the search database though, until the database compactor has been run. The search database will, however, work just fine without the database compactor ever being run, so there is no need to run it often. On the other hand it is not a very resource consuming task, and it will not alter the database at all if there isn't any deleted words, so it can be scheduled often without any greater risk.

The Notes field is not used by Roxen Search and can be used as scratch board for the administrator where he can write notes or a profile description.

There are currently two Profile Actions defined. The first one, Delete profile, will delete the currently selected database profile, after confirmation. It will, however, not delete the actual MySQL database. The second one, Change database, will change the working database of the database profile.

Note!

A database may only be used by one database profile. If two database profiles are connected to the same database there is a risk of corrupt data.

6.3 Commit Hook

Under the Commit Hook tab you can configure things related to the SiteBuilder commit notification.

The Work Area is simply the work area of any running SiteBuilder site on the web server the commit hook should monitor. All commits done in that work area will notify Roxen Search and, if it passes the restrictions defined by the other settings under this tab, be indexed and added to the search database. Select "None" to disable commit triggered indexing.

The commit index latency settings enable the indexer to wait and queue file paths that it will index for at the most the number of seconds entered into this setting. By default the indexer waits for five minutes after a commit has been performed.

Only pages that are matched by at least one of the patterns under Allow Path, and not matched by any of the patterns under Disallow Path will be indexed. You can not perform the SiteBuilder action Index File on a file that does not match these patterns and all non matching files will be excluded when the SiteBuilder action Index Tree is performed. Global expressions are explained in appendix A and regular expressions in appendix B. You can test if a specific file will be indexed by entering its path in the Test Path widget at the bottom of the page.

6.4 Crawler Settings

Under the Crawler tab you can configure the Crawler behavior as well as Schedule when the crawler should be run. Note that if all your files are stored in the SiteBuilder repository you can use the Commit hook to index all new and altered files.

Under start pages you can list all the places where you want the crawler to start looking. The crawler will load these pages and then follow the links on them to other pages and continue to load pages until it can't find a link to a page it hasn't been on before, or isn't permitted to visit because of the Allow/Disallow URL settings.

Only URLs that are matched by at least one of the patterns under Allow URL, and not matched by any of the patterns under Disallow URL will be indexed.

Note!

If you allow all URLs, the crawler will follow links to external sites and index them as well, which can potentially result in a very large search database. Global expressions are explained in appendix A and regular expressions in appendix B. You can test if a specific file will be indexed by entering its URL in the Test URL widget at the bottom of the page.

Under Crawling behavior you can configure how often and when the crawler is scheduled to run. You can also enter the longest amount of time that the crawler should wait for a page before it gives up.

Under Browser settings you can add cookies that you want to have set in the crawler. These will be sent to all servers that the crawler fetches pages from. The crawler will however not accept any new cookies sent from any server. The user agent string what the crawler will identify itself as during crawling. Note that some servers might produce strange results if the user agent string deviates too much from clients (the browsers) agent strings.

6.5 Metadata Indexing

If the file came from the SiteBuilder commit hook it is possible to index its metadata, such as title and description, regardless of which kind of file type it has. However, the file has to match the Allow/Disallow Path patterns in order to be indexed. You can enable/disable this feature under SiteBuilder Exception.

7 Query Profiles

You can see database profiles as the information that is needed during indexing and query profiles as the information that is needed during querying. That is not completely true, since you need to know which search database to query, but otherwise it is a fairly accurate description. The query profile contains information about how to rank the search hits. There is always a default query profile available, named Default, which may be altered but not removed.

Adding a new query profile is easily done by pressing the button “New Query Profile” under the Query Profiles tab. You will then be taken to a one step wizard where you will be asked to enter a new name for your query profile.

The query profile name is used as a label by which you can identify that query profile, both in the Search SiteBuilder interface, but also in attributes to Roxen Search RXML tags. A short but descriptive name without characters that you don’t want in your XML/HTML files is best. Avoiding the characters <, > and & is also a good idea.

7.1 Info Tab

The Notes field is not used by Roxen Search and can be used as scratch board for the administrator where he can write notes or a profile description.

There is currently only one Profile Actions defined for query profiles, Delete profile, which will delete the currently selected database profile, after confirmation. This action is not available for the default search profile.

At the bottom of this page, as well as on the bottom of all the other settings tabs, is a test search widget where the settings in the current query profile can be applied and tested on any of the database profiles.

7.2 Field Ranking

By changing the field ranking values you can fine-tune the results of queries to the search engine. The number of occurrences in each field is multiplied with that field’s weight. Normally one occurrence of a search word in the documents title is “worth” more than an occurrence in the documents text body. Altering the field ranking is usually only useful when custom defined fields are indexed.

Before the number of hits in a field is multiplied with its ranking coefficient, the hits are limited by a cut-off value. This is done to remove extreme values that often produce an overly optimistic score for the document.

Field	Description
author	Taken from the document’s metadata field “Author”.
body	The text body.
category	Categories set for the document.
content_type	The documents content_type.
description	Taken from the document’s metadata field “Description”.
external_use	This field is used internally to determine whether a document is visible or not and thus if the document should turn up as a search hit or not. Therefore it is pointless to set

	the field ranking to anything.
headline	Content of `<h1>`, `<h2>` and `<h3>`.
intrawise.folderid	Deprecated.
intrawise.type	Deprecated.
keywords	Taken from the document's metadata field "Keywords".
modified	The content of the documents `<meta name="modified">`.
mtime	If the index has access to SiteBuilder then this will be the time of the last commit otherwise it's the time of indexing.
path1	The name of a directory directly under the root ("/").

path2	The name of a directory on the second level down from the root. I.e. limit a search with "`(path2: foo)`" would give results like:
	`/bar/foo/`
	`/gazonk/foo/`
	`/bar/foo/rakapparat/`

robots	The content of the documents `<meta name="robots">`.
size	The document's file size. Probably not relevant for field ranking.
summary	The 150 first characters of the document.
title	The document's title.
uri	The document's complete URI.

7.3 Proximity Ranking

When doing a multiple word query, e.g. "Roxen Platform", the search engine looks for documents that contain both (or as many as possible) of the words in the query. The search engine then calculates the distance between the locations of the two words in each matching document. With the default Proximity Ranking parameters, documents that contain the two words close to each other are ranked higher than those where they are far apart.

The query engine categorizes each word distance into one of eight possible different rankings, as seen in this table:

Distance	**Default ranking value**
0 (Perfect hit)	8
1-5 words	7
6-10 words	6
11-20 words	5
21-40 words	4
41-80 words	3
81-160 words	2
161- words	1

8 Indexers

The Roxen Search Indexer consists of five processes that communicates with each other in a chain through pipes. These processes are launched by the Interface module either by the scheduler or by a manual trigger from within the interface. The scheduler can launch the indexer processes for two reasons, either because the user has scheduled a crawler through the crawler settings (see the Crawler Settings sections on Database Profiles) or because there has been URLs in the crawler queue for too long. URLs are put into the crawler queue when the Index file or Index tree action in SiteBuilder is triggered or when a file is committed.

The five processes are a crawler, a crawler to filter buffer, a filter, a filter to indexer buffer and an indexer. In addition to talking with each other through pipes the crawler, filter and indexer also talks to the Roxen process through Remote Procedure Calls (RPC).

8.1 Control and Overview

Under the Indexers tab you will find a list of all database profiles together with information about its associated indexer processes. The table has the following columns:

Profile name

The first column of the table contains the name of each database profile, which is linked to the settings for that profile.

Waiting

The number of documents that are waiting in the queue to be indexed.

Fetching

The number of documents that are currently being downloaded by the crawler process.

Indexing

The number of documents that are currently being filtered, indexed and stored into the search database.

Completed

The number of successfully retrieved and stored documents.

Errors

The number of documents that failed indexing, typically because of a HTTP error while downloading.

Process status

The status of the associated indexer processes, here handled as a group. Can be either “Idle” or “Running”. If the processes are in the running state, the number of seconds it has been running is shown in a parenthesis after the status message.

In the last column of the table are four different buttons that control the state of the indexer and the search database. They have the following effects:

Continue

Starts up the indexer and lets it index the items currently in the URL queue. This action is available when the URL queue isn't empty and the indexer isn't running, which occurs either when the indexer is stopped during its work or when documents are committed/queued for indexing while the indexer is not active.

Re-index

Puts all indexed documents in the URL queue and starts the indexer. This action is always available when the indexer isn't running.

Stop

Stops the indexer, but lets already fetched documents be stored in the search database. This action is only available when the indexer is running.

Clear

Removes all documents from the search database. Generally not a recommended action. This action is always available when the indexer isn't running.

8.2 SiteBuilder/Crawler Interaction

When using the commit hook, indexing a multi language file will cause the file to be indexed once for every language it has. The document will then exist in several versions in the search database, all pointing to the same document.

Note!

When using language dependent XSL-templates on a contents file with limited or no languages, indexing will only be performed for the languages the contents file contains.

Indexing a work area with commit notification will run the crawler in privileged mode through a special "backdoor". This is done in order to be able to index all documents regardless of any access restrictions. Those are in stead applied when a user performs a query.

Some SiteBuilder metadata is also indexed. These includes title, keywords and description. They will take precedence over the corresponding fields in a HTML-file if they are nonempty. Also the metadata External Visibility is stored and taken into account at the same stage as the AC filtering in queries.

To get the metadata, a special header is sent to the web server when crawling; `request-metadata` with empty content. This will cause the SiteBuilder server to respond with the extra header `sb-metadata` containing the metadata in encoded form. The server also checks that the request is authorized to receive this metadata.

Some documents may be hidden by the External Visibility setting. The crawler sends another special header when crawling to be able to retrieve such documents; `request-invisible`. The same authorization check as for the metadata is performed on the server.

8.3 Logs

Under the tab View Log you will find a list of the actions performed by Roxen Search, most notably the Indexer. The list has three columns, "Date", "Profile" and "Log Entry". The date column contains the time and date when the event was logged, formatted in ISO style (YYYY-MM-DD HH:MM:SS). The profile column contains the

profile name and the log entry column contains the actual log message along with a icon signifying if the log entry was considered a notice, a warning or an error.

Above the list is a box with some filtering functions to reduce the size of the list and bring forward the information you want. In the first cell you can select which profile you are interested in, or select “All” if you want a combined report. In the next cell you can select which of the event types error, warning and notice that you want to have displayed. Finally there is a cell with the action button “Apply filter”, which performs the filtering and displays the result.

The action button cell also contains two buttons, “View log as HTML” and “View log as text”, with which you can toggle between the HTML table rendering and a pure preformatted text list. The HTML list is easier to quickly grasp the information while the text list is more compact and faster to render.

Under the tab Clear Log you can remove entries from the Search event log. You can select to remove entries either by one of the types error, warning or notice, or you can remove entries by which profile it is connected to.

9 Filters

The filter is called after a document has been fetched and before it is split into words. The function of the filter is to extract as much text as possible from document and try to preserve as much context information as possible, i.e. if the text was a heading or text body. The filter is also responsible for normalizing the text into Unicode, which is used internally by the search engine.

The programming API for filters are defined as a plugin API and it is easy to write new plugins for other file types.

9.1 Plain Text

The plain text filter, which takes care of documents with MIME type text/plain, is the simplest of all the filters. It simply takes all the text from the document and marks it as body text. Currently no attempt is made to try to identify the title of the document.

9.2 HTML

The HTML filter takes care of documents with the MIME type text/html. The HTML filter can decode almost 500 different character set, including iso-8859-1 to 15, iso-2022-jp, shiftjis, euc-jp, euc-kr, euc-cn, windows 1250 to 1258, utf-7, utf-7.5, utf-8, BIG5 and GB2312. It can also decode all iso-8859-1 entities as well as international and Greek entities.

Once decoded the HTML file is stripped from HTML tags. The following tags have special meaning to the filter. The filter will parse them case insensitive.

```
<noindex>…</noindex>
<no-index>…</no-index>
<script>…</script>
<style>…</style>
```

The contents of these tags will be discarded by the filter.

```
<meta name="keywords" content="" />
<meta name="description" content="" />
```

The information in the content attribute will be added to the appropriate keywords/description field of this document. To be forgiving against bad HTML `http-equiv` is accepted as alias for `name` and `contents` as well as `data` is accepted as alias for `content`.

```
<a href="">…</a>
<link href="">…</link>
```

Links found in anchor and link tags will be added to the database of known URLs, and will be added to the queue if it has not been indexed. Information found in the title attribute will be added to the body field.

```
<title>…</title>
```

The information in the title tag will be added to the title field of this document.

```
<base href=""/>
```

The filter module will replace the document root location with the path given in the base tag, should it encounter any.

Apart from these tags the HTML filter will also parse the following tags, looking for URLs and `alt`-attributes. All found URLs will be added to the database of known URLs, and will be added to the crawler queue if it has not been indexed already. Texts found in `alt` attributes will be added to the document body field.

```
<applet />
<area />
<bgsound />
<body>…</body>
<embed />
<frame />
<iframe />
<ilayer>…</ilayer>
<img />
<layer>…</layer>
<object />
<q>…</q>
<sound />
<table>…</table>
<td>…</td>
<xml>…</xml>
```

9.3 MS Word

The MS Word filter takes care of documents with the MIME type `application/msword` and `application/vnd.ms-word`. The MS Words filter relies on an external program called wvWare, which has been slightly modified to not output as much debug information into the debug log. The vwWare binary is located in the `bin` subdirectory of the Roxen Search directory.

9.4 PDF

The PDF filter takes care of documents with the MIME type `application/pdf`. The PDF filter relies on an external program called xpdf, which has been slightly modified to suit the plugin API. The xpdf binary is located in the `bin` subdirectory of the Roxen Search directory.

10 Search RXML interface

The RXML Interface consists of a set of tags that are used to input queries from the user and present the results of running the queries. The absolute minimum you have to write to get a search page with a search form and results listing is:

```
<html>
  <body>
    <search-form/>
    <search-results/>
  </body>
</html>
```

You can also have the search form and the result listing on different pages:

`form.html`:

```
<html>
  <body>
    <search-form action="results.html"/>
  </body>
</html>
```

`results.html`:

```
<html>
  <body>
    <search-results/>
  </body>
</html>
```

The search form and the results listing communicate via form variables. However you can override various aspects of their behaviour. E.g. you can write a results listing for a hard-coded query:

```
<search-results query="fruit"/>
```

10.1 Customized Forms

To create a customized form, use `<search-form>` as a container. Since it will generate a `<form>` container in the output, you can use all `<input>` and `<select>` elements that you would use in any `<form>`. You can add attributes to the resulting `<form>` by adding them to `<search-form>`, e.g. change the `method="GET"` (which is the default), to `method="POST"`.

Example (in French):

```
<search-form method="POST" action="results.html">
  Tapez quelques mots descriptifs et cliquez sur le bouton:<br />
  Requête: <input type="text" name="query" />
  <br clear="all" />
  <input type="submit" value=" Recherce " name="submit" /><br />
</search-form>
```

For convenience, there are two subtags that you can use inside `<search-form>`:

<search-form-db-select/>

Generates RXML to get the database profile to search in, and puts the name of it in the form variable `db`.

<search-form-type-select/>

Generates RXML to get the type of search to do, either “or” or “and”, and puts it in the form variable `type`.

Since the default RXML code for `<search-form>` and `<search-results/>` communicate via form variables, you have to make sure that if you alter either one of them, the other checks for the right variables. If you put this inside `<search-form>`:

Enter your query:

```
<input type="text" name="q"/>
```

Next you will have to see to it that `<search-results>` uses the variable `q`, and not `query`:

```
<search-results query='&form.q;'/>
```

10.2 Customized Results

The tag `<search-results>` is used to generate multipage search result listings. As seen above it can be used as a single tag, thereby using the default RXML content.

The search results listing can also be customized and given different layouts. There are two subtags to `<search-results>`, namely `<search-results-entries/>`, and `<search-results-tabs/>`.

<search-results-entries/>

Generates a listing of hits, but only the hits on the current page. When used as a single tag, a default piece of RXML code is inserted.

<search-results-tabs/>

Generates a tab list with links to other pages with hits. When used as a single tag, a default piece of RXML code is inserted.

Inside `<search-results>` there are entities available about the results of the search. For example, you can create a result page that does not show links to the matching documents, but rather just some info about the search:

```
<search-results>
  <if expr="&_.hits; == 0">
    Sorry, but your query: '&_.query;' did not match any documents!<br />
  </if>
  <else>
    You were looking documents matching '&_.query;'
    in the database '&_.db;'.<br />
    &_.hits; matching documents were found.<br />
    The search took about &_.time; seconds.<br />
  </else>
</search-results>
```

But then you would want to present the matching documents too. That is done with the `<search-results-entries>` container. It loops through the documents on the current listing page, in order. How many documents it will loop through can be

controlled by setting the `perpage` attribute of the surrounding `<search-results>` container, or if that attribute is missing, the default setting in the SiteBuilder Interface is used. A good value is probably somewhere between 10 and 20.

What document will start with depends on what page is focused. That is controlled by the `page=""` attribute to the surrounding `<search-results>` container, or if that is missing, by the form variable `page`.

For each document, the `<search-results-entries>` container runs through its content, setting the entities to different values before each loop to reflect the properties of the current document. There are more than ten different entities so refer to the documentation for the tag for all of them. An easy example using the most useful of them:

```
<search-results-entries>
  <h2>&_.title;</h2>
  <p>
    <i>&_.description;</i><br />
    <font size="-2">&_.body:none;</font><br />
    Size: &_.nice-size; Score: &_.score;<br />
    <a href="&_.uri;">&_.nice-uri;</a>
    <if variable="_.content-type == application/pdf"> [PDF file] </if>
  </p>
</search-results-entries>
```

The tag `<search-results-tabs/>`works similarly. But instead of looping through the documents on the current page, it loops through all pages in the listing. For each page it gives information about on which document it starts and ends, what page number it has and so on. This tag is used to make a list of links to the other pages. It can be used as a container if you want to make your own layout of the links.

In each link, make sure to pass, along with the page number, all variables that are needed to identify the search, namely:

```
&_.query;, &_.query-profile;, &_.db;, and &_.type;
```