Documentation for EXT:solr the Apache Solr integration for TYPO3

Release 8.0.1

Ingo Renner, Timo Hund, Markus Friedrich

2018-03-07 09:46

CONTENTS

CHAPTER

ONE

WHAT DOES IT DO?

Apache Solr for TYPO3 is the search engine you were looking for with special features such as **Facetted Search** or **Synonym Support** and an incredibly fast response times of results within milliseconds.

When development started, the primary goal was to create a replacement for Indexed Search. With the initial public release at T3CON09 in Frankfurt, Germany that goal was reached and even passed by adding features which Indexed Search does not support.

With EXT:solr it is simple to build a search for a TYPO3 website, that allows you to index any kind of TYPO3 records with TCA configuration and pages. The results can be rendered with flexible fluid templates, to render the results as you need them.

CHAPTER

TWO

FEATURE LIST

- Facetted Search
- Spellchecking / **Did you mean**
- **Multi Language Support**
- Search word highlighting
- Field Boosting for fine tuning the importance of certain index fields
- **Frontend User Group Access Restrictions Support**
- Stop word Support
- Synonym Support
- **Auto complete / Auto suggest**
- Language Analysis / Support for inflected word forms
- Content Elevation / **Paid Search Results** / Editorial Content
- Sorting of Results
- Content indexing through a near instant backend **Index Queue**
- Auto correction (search for the first suggestion)
- and more...

CHAPTER

THREE

DEVELOPMENT AND PARTNERSHIP

The extension is developed in an open source way and the source code is completely available on github. Releases to the TYPO3 TER are done in regular time frames (Usually every 3 months).

To make the development possible you can join a partner ship with dkd. By joining the partner program you have the following benefits:

- You support the further development on EXT:solr
- **You get access to addon's that are not public available**
    - Indexing of files from FAL (TYPO3 File Abstraction Layer)
- You can be mentioned as a sponsor on http://www.typo3-solr.com
- You can included support based on your subscription

By the financial invest of our partners it is possible to keep this extension uptodate and integrate great new features. If you like this extension please think about to become a partner as well!

If you are interested to become a partner visit http://www.typo3-solr.com or call dkd +49 (0)69 - 247 52 18-0.

CHAPTER

FOUR

GETTING STARTED

In this chapter we would like to give you a quick introduction into the very first steps with EXT:solr.

After reading this chapter you will know:

How to...

- Start a solr server
- Install the extension and connect your TYPO3 system to your solr server
- Index a few pages and see the search results on your website

4.1 Solr

First you need to install Solr itself. There are several ways to do so:

4.1.1 Using Hosted-solr.com

If you want to start simple and just create a solr core with a click. You can use hosted-solr.com. For a small fee you get your own solr core in seconds, configured to be used with EXT:solr.

4.1.2 Shipped install script

With the extension we ship and install script that can be used for a **development** context. It creates a solr server with a core for all languages. This script is located in "Resources/Private/Install" an it installs a configured solr server that is useable with EXT:solr.

By default this script is not executable and you need to add the execute permissions to your user to run it.

The example below shows how to install a solr server to /home/developer

```
chmod u+x ./Resources/Private/Install/install-solr.sh
./Resources/Private/Install/install-solr.sh -d /home/developer
```

After running the script you are able to open a solr server with over the loopback address. Which means, when you want to access it from outside, you need to create an ssh tunnel.

4.1.3 Docker

You can use docker to install your solr server with a small effort. With the extension we provide a Dockerfile, that creates a container with a core for all languages ready to run. This helps you to setup a container very quickly.

To build the images, simply type one of the following:

```
docker build -t typo3-solr .
```

Prepare the data folder (data is shared with the docker container by user and group with UID/GID 8983):

```
mkdir -p .solrdata
chmod g+w .solrdata
chown :8983 .solrdata
```

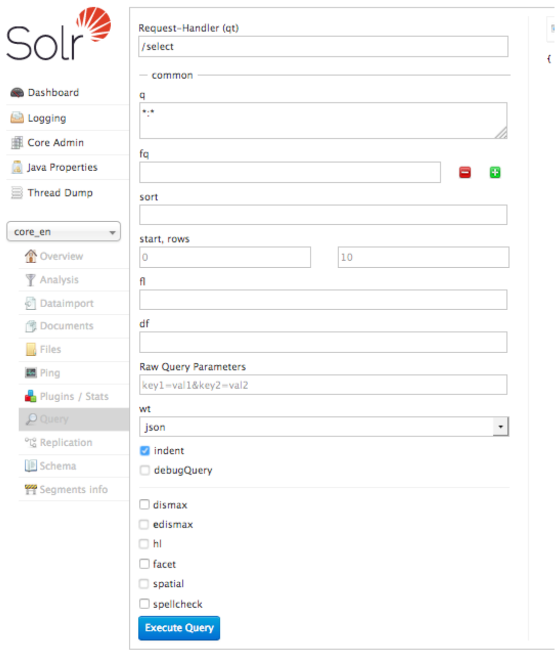
To run the container (only run one of the following):

```
docker run -d -p 127.0.0.1:8983:8983 -v "$PWD/.solrdata:/opt/solr/server/solr/data/
↪" typo3-solr
```

To check whether Solr is up and running head over to:

`http://<ip>:8983/solr/#/core_en/query`.

You should see the web interface of Solr to run queries:



Request-Handler (qt): /select
common
q: *:*
fq
sort
start, rows: 0 10
fl
df
Raw Query Parameters: key1=val1&key2=val2
wt: json
☑ indent
☐ debugQuery
☐ dismax
☐ edismax
☐ hl
☐ facet
☐ spatial
☐ spellcheck

Important: The image "typo3-solr" ships a default core for all languages. The data of the cores is stored on a data volume. When you want to update the container, you can just start a new container using the data volume of the old container. But at the same time this has the limitation, that you should only use this image with the default cores! If you want to create custom cores with a different configuration please read the section "Advanced Docker Usage"

Please note: The steps above show how to build the image from the Dockerfile. You can also download and use our compiled images from dockerhub:

https://hub.docker.com/r/typo3solr/ext-solr/

Advanced Docker Usage

Our image has the intension to create running cores out of the box. This implies, that the schema is inside the container. The intension in our integration was to stay as close as possible to the official Apache Solr docker images. Sometimes it might make sence that you use the official image directly instead of our image. An example could be when you want to have the solrconfig, schema and data outside of the container.

The following example shows how you can run our configuration with the official Apache Solr docker container by mounting the configuration and data from a volume (When using Docker on macOS make sure you've added the volume folder to "Preferences -> File Sharing").

```
mkdir -p ~/mysolr
cp -r Resources/Private/Solr/* ~/mysolr
sudo chown -R :8983 ~/mysolr
docker run -d -p 127.0.0.1:8983:8983 -v ~/mysolr:/opt/solr/server/solr/ solr:6.3.0
```

4.1.4 Other Setup

Beside the install script and Docker there are various possibilities to setup solr. All of these possibilities are not officially supported, but the simplify the setup i want to mention them shortly here and summarize the needed steps.

Known Installers

All of these installers can be used to setup a plain, reboot save solr server:

- Use the installer shipped with solr itself (bin/install_solr_service.sh):

Allows to install solr on many distributions including init scripts (At the time of development ubuntu 16.04 was not supported and therefore it was no option for us to use it).

- Use chef / ansible / whatever dev ops tool:

Allows you to setup a solr server with your DevOps tool.

e.g. https://galaxy.ansible.com/geerlingguy/solr/ (ansible) or https://supermarket.chef.io/cookbooks/solr (chef)

Deployment of EXT:solr configuration into Apache Solr

Since EXT:solr 6.0.0 the configuration and all jar files are shipped in one "configSet". The goal of this approach is to make the deployment much easier.

All you need to do is, you need to copy the configSet directory into your prepared solr installation and replace the solr.xml file. In the installer we do it like this:

```
cp -r ${EXTENSION_ROOTPATH}/Resources/Private/Solr/configsets ${SOLR_INSTALL_DIR}/
↪server/solr
cp ${EXTENSION_ROOTPATH}/Resources/Private/Solr/solr.xml ${SOLR_INSTALL_DIR}/server/
↪solr/solr.xml
```

After this, you can decide if you want to create the default cores by copying the default core.properties files or if you want to create a core with the solr rest api.

Copy the default cores:

```
cp -r ${EXTENSION_ROOTPATH}/Resources/Private/Solr/cores ${SOLR_INSTALL_DIR}/server/
↪solr
```

Create a core with the rest api:

```
curl "http://localhost:8983/solr/admin/cores?action=CREATE&name=core_de&
↪configSet=ext_solr_8_0_0&schema=german/schema.xml&dataDir=dataDir=../../data/
↪german"
```

After installing the solr server and deploying all schemata, the TYPO3 reports module helps you to verify if your setup fits to the requirements of EXT:solr

You now have a fully working, pre configured Solr running to start with

No you can continue with installing the extension *Install EXT:solr*.

4.2 Install EXT:solr

4.2.1 Install from TER using the TYPO3 Extension Manager

You can simply install stable versions of EXT:solr using the Extension Manager from the TYPO3 backend.

1. Go to the **Extension Manager**, select **Get Extensions** and search for “solr”.
2. Install the extension.
3. The Extension Manager will also install EXT:scheduler if not installed already for running the indexing tasks
4. While developing we recommend installing devlog for easier error detection, too.

4.2.2 Install from git

Alternatively, you can also get the latest development version from GitHub:

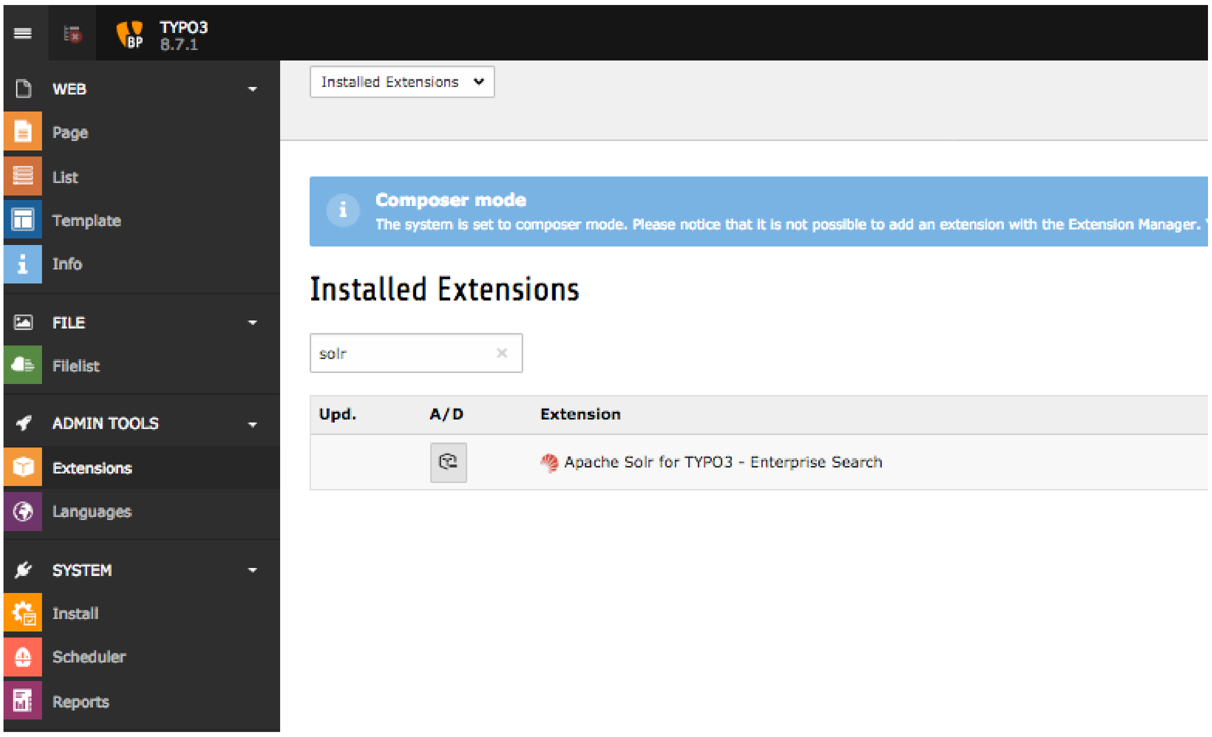
```
$ git clone git@github.com:TYPO3-Solr/ext-solr.git solr
```

4.2.3 Install with composer

Install this TYPO3 Extension `solr` via TYPO3 Extension Manager as usual, or via `composer` by running:

```
composer require apache-solr-for-typo3/solr
```

Head over to the Extension Manager module and activate the Extension.



Composer mode

The system is set to composer mode. Please notice that it is not possible to add an extension with the Extension Manager.

Installed Extensions

Upd.	A/D	Extension
		Apache Solr for TYPO3 - Enterprise Search

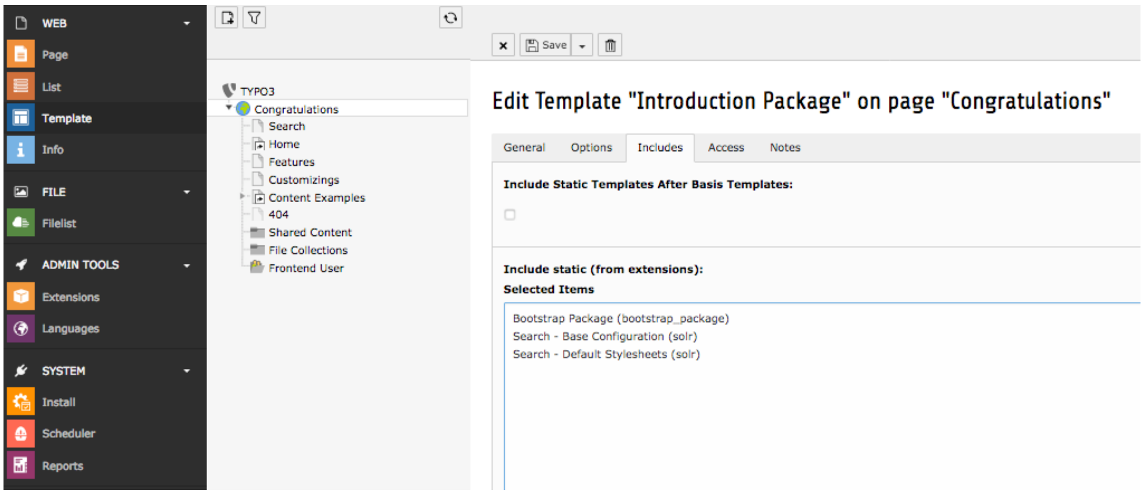
That’s all you have to do, now head over to *Configure Extension*.

4.3 Configure Extension

After *Install EXT:solr* you need to configure the extension. Only a few steps from below are necessary for *Index the first time*.

4.3.1 Static TypoScript

The extension already comes with basic configuration that will work for small pages out of the box. For now create, or edit an existing, TypoScript Template record in your page tree and add the provided static TypoScript:



Edit Template "Introduction Package" on page "Congratulations"

Include Static Templates After Basis Templates: ☐

Include static (from extensions):

Selected Items

Bootstrap Package (bootstrap_package)
Search - Base Configuration (solr)
Search - Default Stylesheets (solr)

Update the constants to match the current setup:

```
plugin {
    tx_solr {
        solr {
            host = 192.168.99.100
            port = 8983
        }
    }
}
```

Adjust the host according to where your Solr is reachable, see *Solr*.

Note:

The static template configures what you need to query the solr server and do the indexing. In most projects you want to add facets or custom styles. If you want to use the default style you need to add the template “Search - Default Stylesheets”. Beside that EXT:solr provides a few example typoscript templates that should help you to build your own configuration.

4.3.2 Search Markers

EXT:solr is indexing everything on a page between *<!– TYPO3SEARCH_begin –>* and *<!– TYPO3SEARCH_end –>* to ensure this is the case, check the output of you website and add the markers to your template.

If the markers are missing, you should add them to your template. To increase the quality of the search results the markes should only wrap the relevant content of a page and exclude e.g. menus, because they are same on each page.

The most simple configuration for my page was:

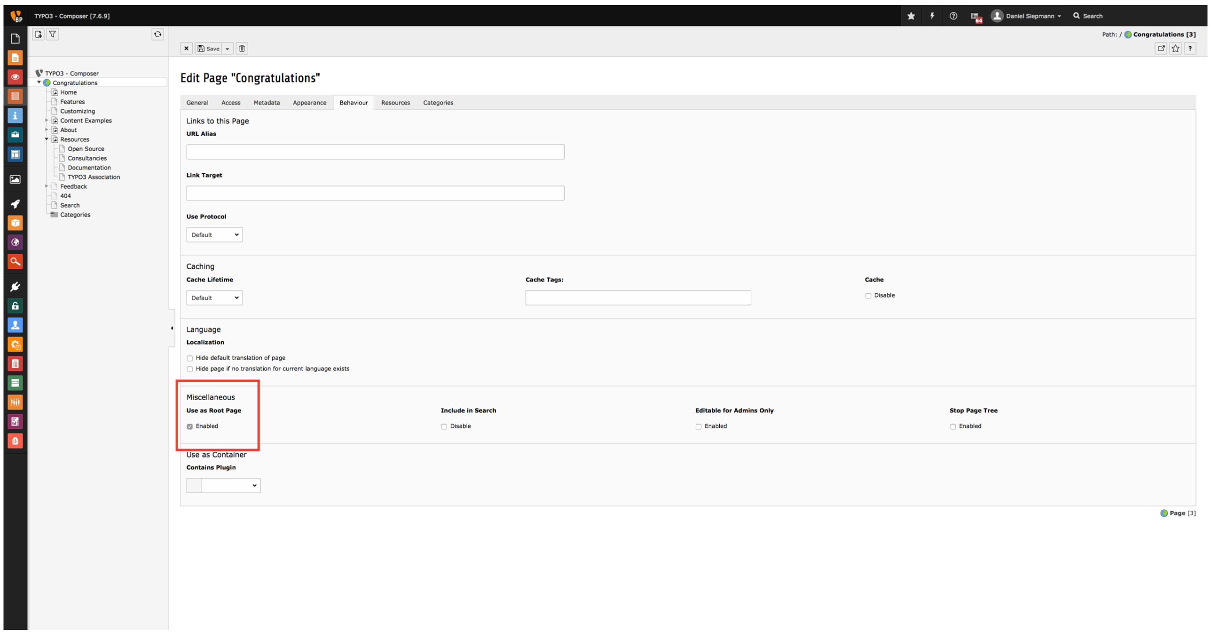
```
page.10 {
    stdWrap.dataWrap = <!--TYPO3SEARCH_begin-->|<!--TYPO3SEARCH_end-->
}
```

4.3.3 Domain Records and Indexing

To enable Solr connections, the extension needs a Domain Record and indexing has to be enabled. Therefore enable indexing by setting the following TypoScript:

```
config {
    index_enable = 1
}
```

Also define that your root page is actually a root page:



Edit Page "Congratulations"

General | Access | Metadata | Appearance | Behaviour | Resources | Categories

Links to this Page

URL Alias:

Link Target:

Use Protocol: Default

Caching

Cache Lifetime: Default

Cache Tags:

Cache: ☐ Disable

Language

Localization

☐ Hide default translation of page

☐ Hide page if no translation for current language exists

Miscellaneous

Use as Root Page: ☑ Enabled

Include in Search: ☐ Disable

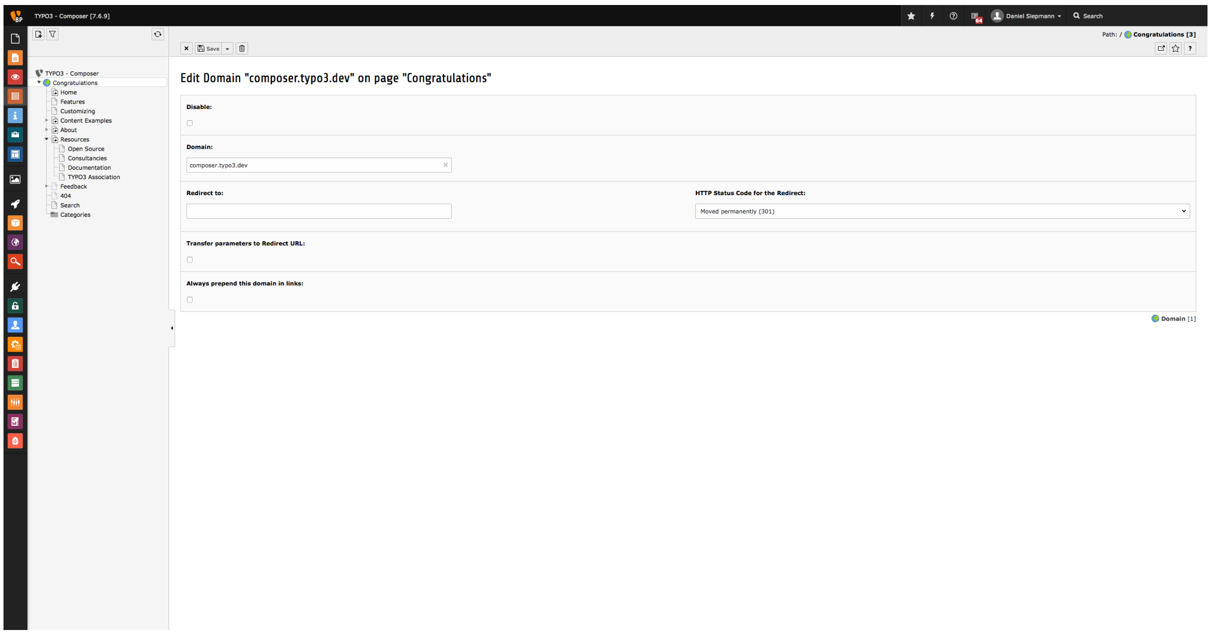
Editable for Admins Only: ☐ Enabled

Stop Page Tree: ☐ Enabled

Use as Container

Contains Plugin:

Last but not least, add the domain record to the root page:



Edit Domain "composer.typo3.dev" on page "Congratulations"

Disable: ☐

Domain: composer.typo3.dev

Redirect to:

HTTP Status Code for the Redirect: Moved permanently (301)

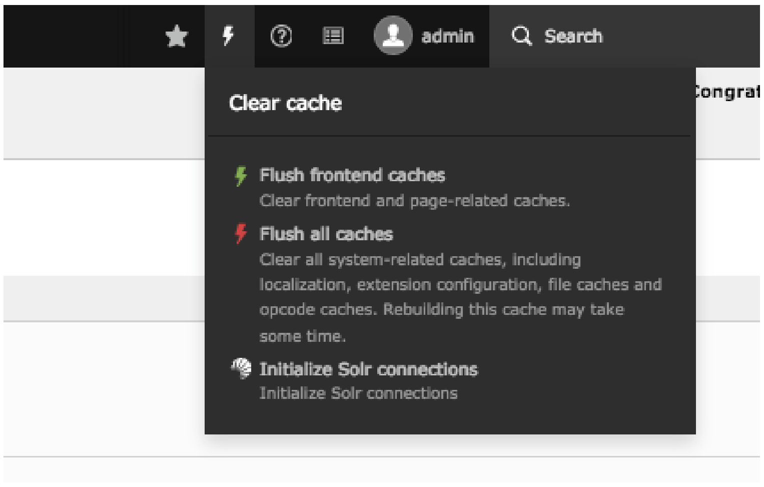
Transfer parameters to Redirect URL: ☐

Always prepend this domain in links: ☐

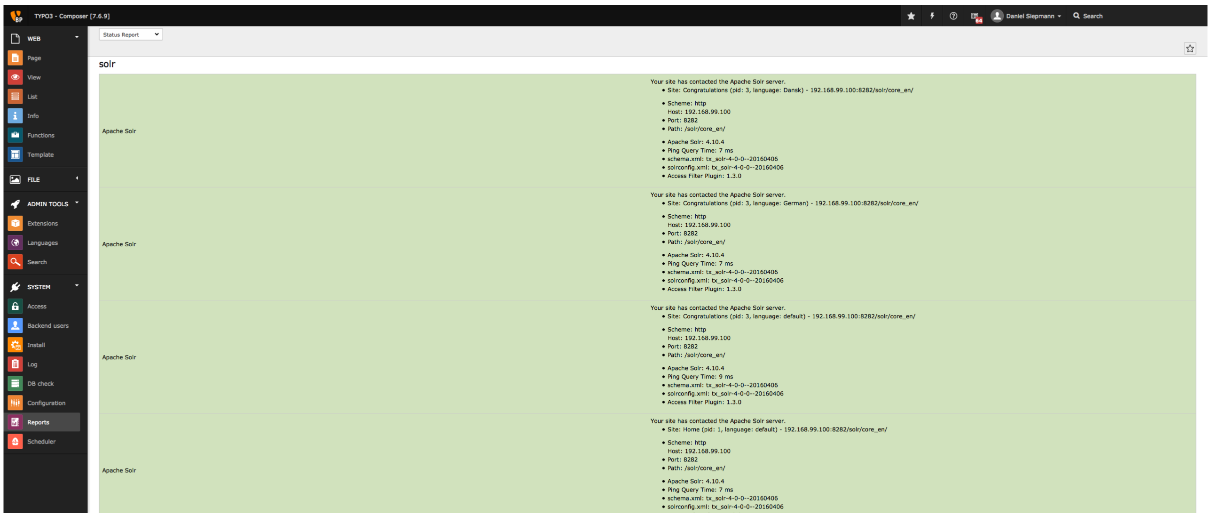
4.3.4 Initialize Solr Connection

Next, initialize the Solr Connection from TYPO3 and check whether everything works as expected.

To initialize the connection, open the Cache-Menu and start Initialization.


admin
Search
Clear cache
Flush frontend caches
Clear frontend and page-related caches.
Flush all caches
Clear all system-related caches, including localization, extension configuration, file caches and opcode caches. Rebuilding this cache may take some time.
Initialize Solr connections
Initialize Solr connections

Check whether connections to Solr could be established by opening the *Reports* module and go to *Status Report* view:

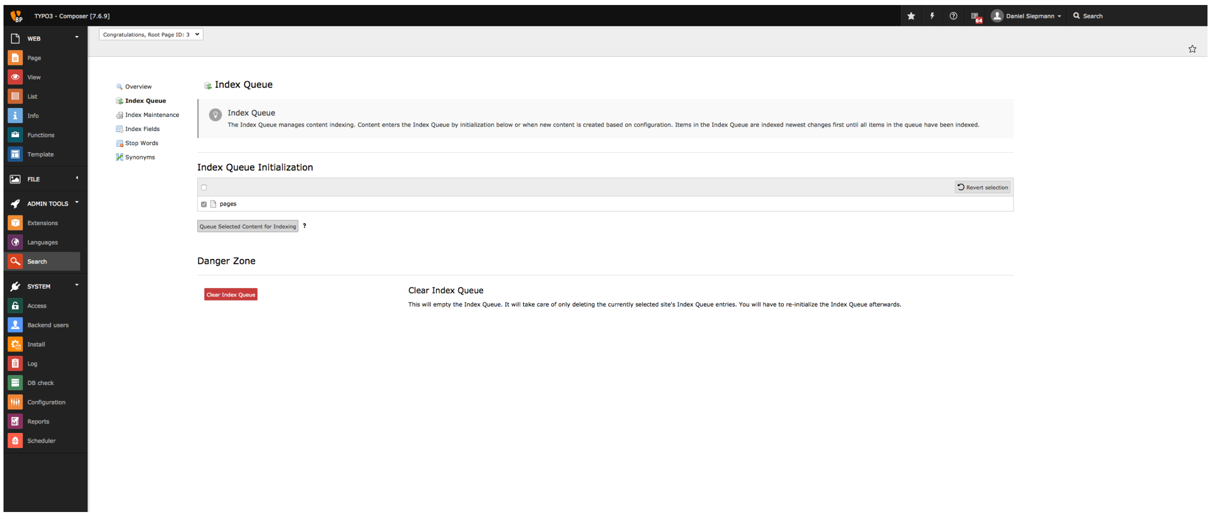


```
solr

Apache Solr    Your site has contacted the Apache Solr server.
                 • Site: Congratulations (pid: 3, language: Dansk) - 192.168.99.100:8282/solr/core_en/

                 • Scheme: http
                   Host: 192.168.99.100
                 • Port: 8282
                 • Path: /solr/core_en/

                 • Apache Solr: 4.10.4
                 • Ping Query Time: 7 ms
                 • schema.xml: tx_solr-4-0-0--20160406
                 • solrconfig.xml: tx_solr-4-0-0--20160406
                 • Access Filter Plugin: 1.3.0

Apache Solr    Your site has contacted the Apache Solr server.
                 • Site: Congratulations (pid: 3, language: German) - 192.168.99.100:8282/solr/core_en/

                 • Scheme: http
                   Host: 192.168.99.100
                 • Port: 8282
                 • Path: /solr/core_en/

                 • Apache Solr: 4.10.4
                 • Ping Query Time: 7 ms
                 • schema.xml: tx_solr-4-0-0--20160406
                 • solrconfig.xml: tx_solr-4-0-0--20160406
                 • Access Filter Plugin: 1.3.0

Apache Solr    Your site has contacted the Apache Solr server.
                 • Site: Congratulations (pid: 3, language: default) - 192.168.99.100:8282/solr/core_en/

                 • Scheme: http
                   Host: 192.168.99.100
                 • Port: 8282
                 • Path: /solr/core_en/

                 • Apache Solr: 4.10.4
                 • Ping Query Time: 9 ms
                 • schema.xml: tx_solr-4-0-0--20160406
                 • solrconfig.xml: tx_solr-4-0-0--20160406
                 • Access Filter Plugin: 1.3.0

Apache Solr    Your site has contacted the Apache Solr server.
                 • Site: Home (pid: 1, language: default) - 192.168.99.100:8282/solr/core_en/

                 • Scheme: http
                   Host: 192.168.99.100
                 • Port: 8282
                 • Path: /solr/core_en/

                 • Apache Solr: 4.10.4
                 • Ping Query Time: 7 ms
                 • schema.xml: tx_solr-4-0-0--20160406
                 • solrconfig.xml: tx_solr-4-0-0--20160406
```

That's it, head over to *Index the first time*.

4.4 Index the first time

After everything is setup, you need to index the contents of TYPO3 to enable searching in Solr. To do so open the *Apache Solr* module and navigate to the *Index Queue*. Select the contents to index and *Queue Selected Content for Indexing*.



Index Queue

Index Queue
The Index Queue manages content indexing. Content enters the Index Queue by initialization below or when new content is created based on configuration. Items in the Index Queue are indexed newest changes first until all items in the queue have been indexed.

Index Queue Initialization

Revert selection

☑ pages

Queue Selected Content for Indexing ?

Danger Zone

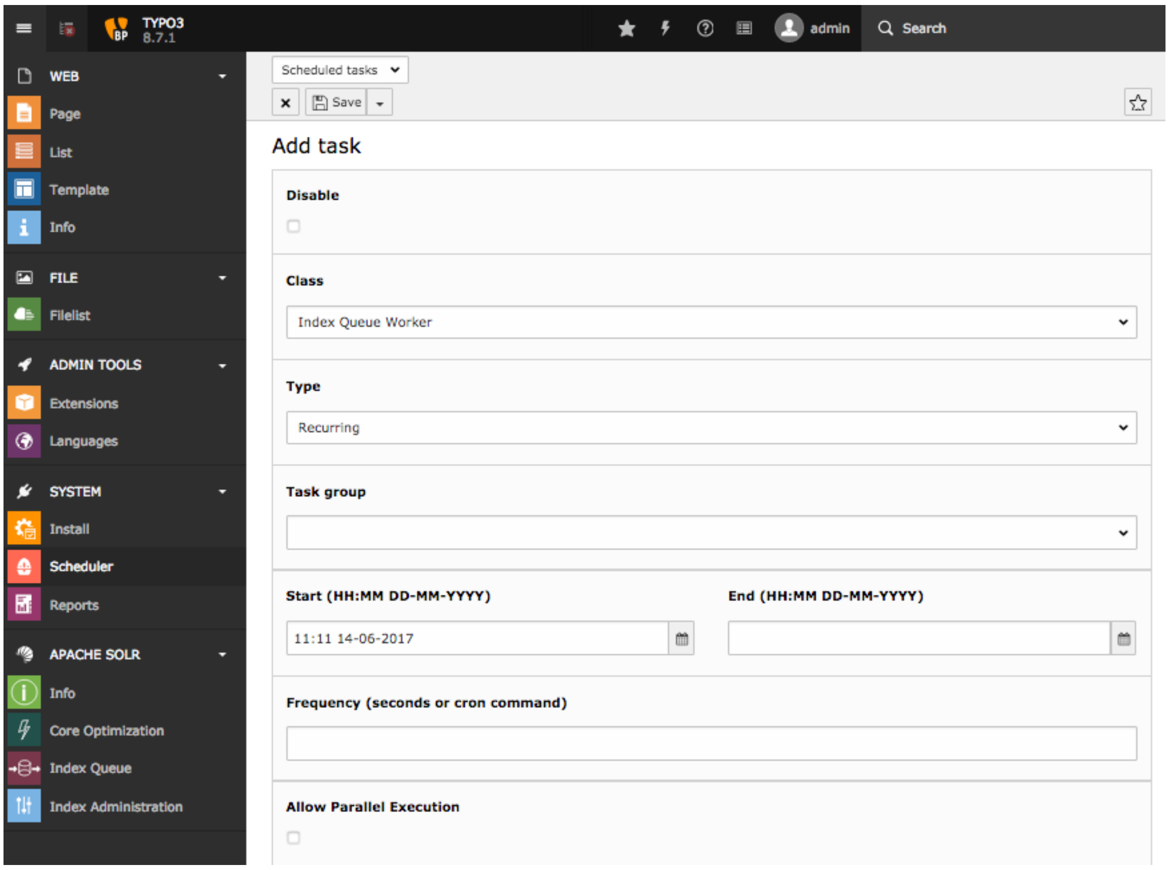
Clear Index Queue

Clear Index Queue
This will empty the Index Queue. It will take care of only deleting the currently selected site's Index Queue entries. You will have to re-initialize the Index Queue afterwards.

Now the records are added to the index queue and ready to index. The indexing can be triggered from the backend module, or can be triggered by the TYPO3 scheduler.

Switch to the *Scheduler* module. If the module is not available, make sure to enable the extension first. It comes bundled with TYPO3 CMS but is not enabled by default.

Create a new scheduler task to run the indexing:



Add task

Disable
☐

Class
Index Queue Worker

Type
Recurring

Task group

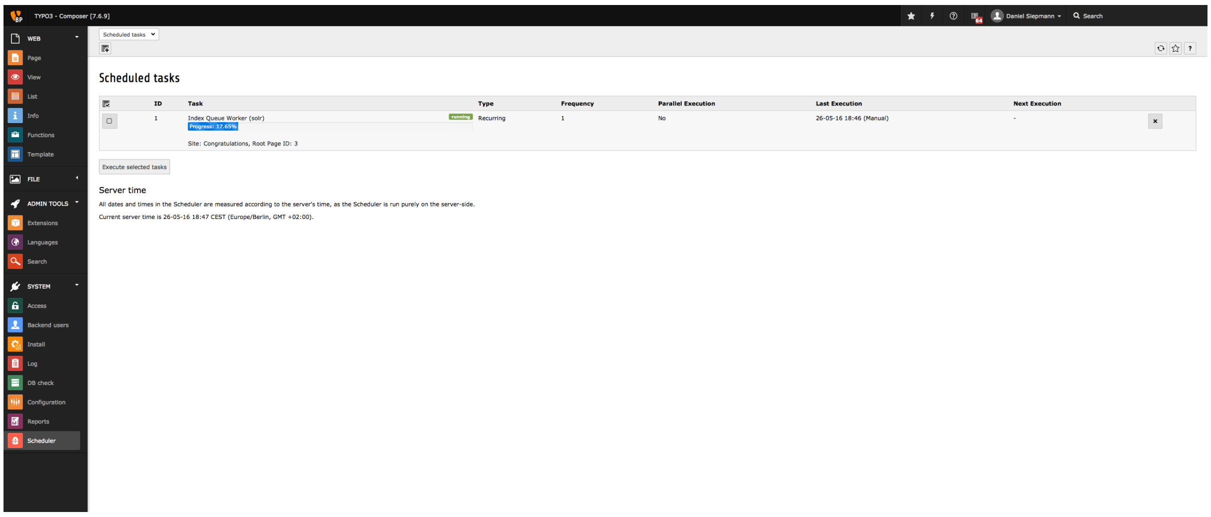
Start (HH:MM DD-MM-YYYY)
11:11 14-06-2017

End (HH:MM DD-MM-YYYY)

Frequency (seconds or cron command)

Allow Parallel Execution
☐

After the task was created, run it manually. The page will indicate a reload but won't reload after the task was run. Therefore you can reload the module to see the progress bar indicating the current progress of indexing:



Scheduled tasks

	ID	Task	Type	Frequency	Parallel Execution	Last Execution	Next Execution
▢	1	Index Queue Worker (solr) running Progress: 17.65% Site: Congratulations, Root Page ID: 3	Recurring	1	No	26-05-16 18:46 (Manual)	-

Server time

All dates and times in the Scheduler are measured according to the server's time, as the Scheduler is run purely on the server-side.

Current server time is 26-05-16 18:47 CEST (Europe/Berlin, GMT +02:00).

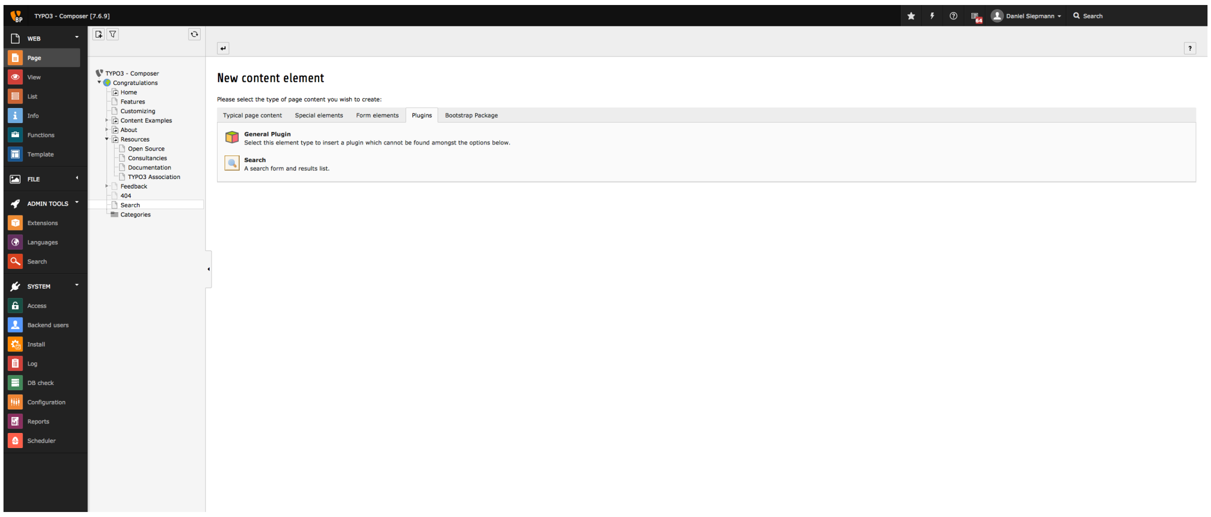
The duration depends on things like the number of records to index and the number of languages configured in your system. Also whether caching is enabled and warmed up.

The extension will now index all records in the queue and send them to Solrs index.

Once you have some records inside the index, you can *Display search and results*.

4.5 Display search and results

After Solr has some documents inside his index, you can insert the plugin to provide a search with results from Solr. To do so create a new content record of type *Search* on a page:



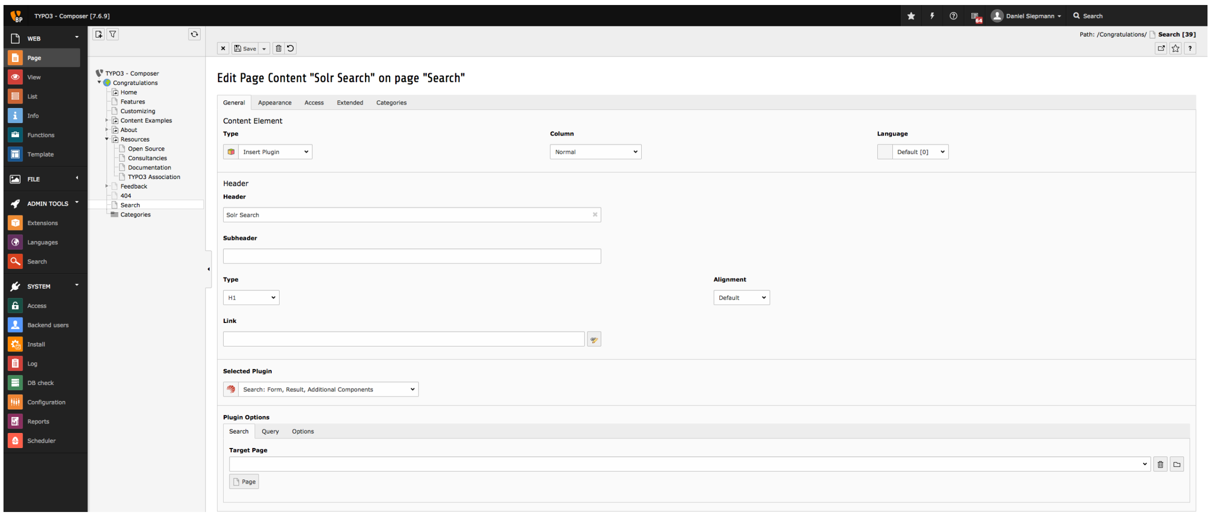
New content element

Please select the type of page content you wish to create:

General Plugin
Select this element type to insert a plugin which cannot be found amongst the options below.

Search
A search form and results list.

Select *Search: Form, Result, Additional Components* if not already selected inside the content element:



Edit Page Content "Solr Search" on page "Search"

Path: /Congratulations/ Search [39]

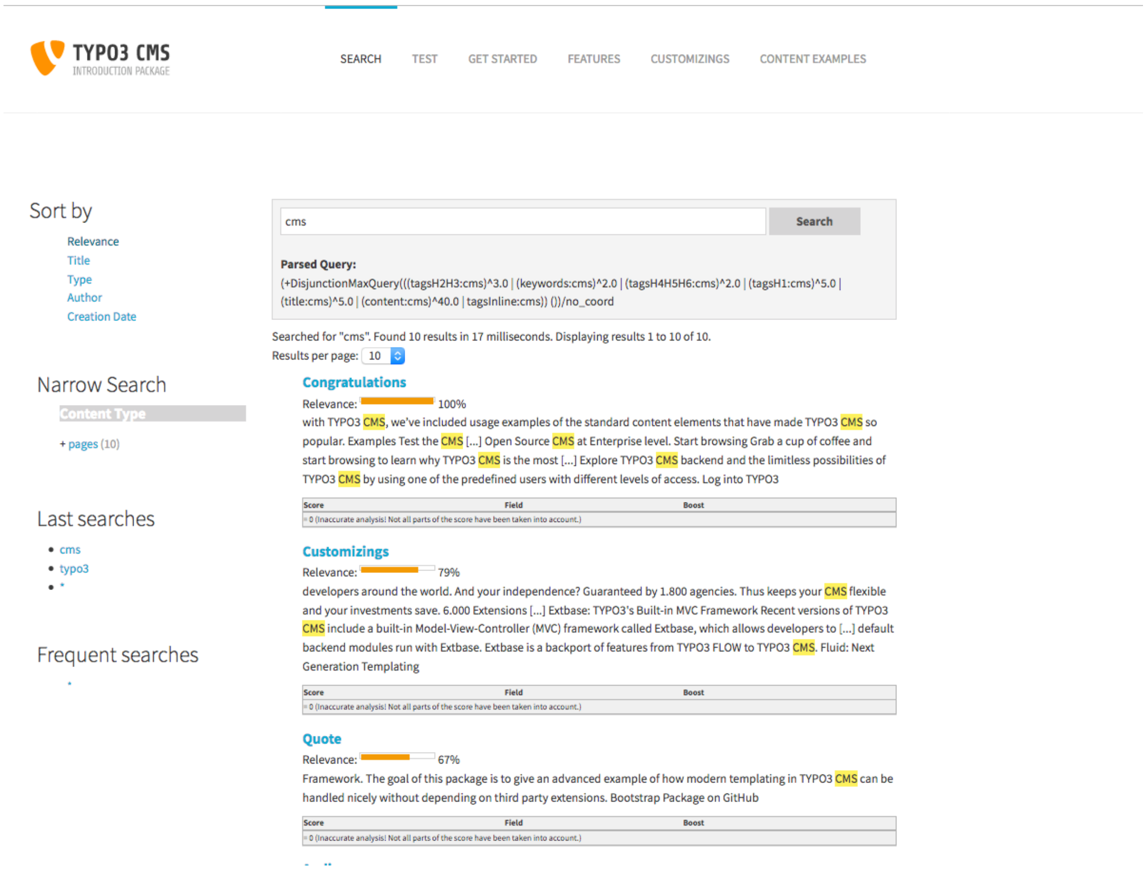
Content Element
Type: Insert Plugin
Column: Normal
Language: Default [0]

Header
Header: Solr Search
Subheader:
Type: H1
Alignment: Default
Link:

Selected Plugin: Search: Form, Result, Additional Components

Plugin Options
Target Page:

Open the page and search for $\star$, you should see all currently indexed records from Solr:



cms

Parsed Query:
(+DisjunctionMaxQuery(((tagsH2H3:cms)^3.0 | (keywords:cms)^2.0 | (tagsH4H5H6:cms)^2.0 | (tagsH1:cms)^5.0 | (title:cms)^5.0 | (content:cms)^40.0 | tagsInline:cms)) ())/no_coord

Searched for "cms". Found 10 results in 17 milliseconds. Displaying results 1 to 10 of 10.
Results per page: 10

Congratulations
Relevance: 100%
with TYPO3 CMS, we've included usage examples of the standard content elements that have made TYPO3 CMS so popular. Examples Test the CMS [...] Open Source CMS at Enterprise level. Start browsing Grab a cup of coffee and start browsing to learn why TYPO3 CMS is the most [...] Explore TYPO3 CMS backend and the limitless possibilities of TYPO3 CMS by using one of the predefined users with different levels of access. Log into TYPO3

Score	Field	Boost
0 (Inaccurate analysis! Not all parts of the score have been taken into account.)		

Customizings
Relevance: 79%
developers around the world. And your independence? Guaranteed by 1.800 agencies. Thus keeps your CMS flexible and your investments save. 6.000 Extensions [...] Extbase: TYPO3's Built-in MVC Framework Recent versions of TYPO3 CMS include a built-in Model-View-Controller (MVC) framework called Extbase, which allows developers to [...] default backend modules run with Extbase. Extbase is a backport of features from TYPO3 FLOW to TYPO3 CMS. Fluid: Next Generation Templating

Score	Field	Boost
0 (Inaccurate analysis! Not all parts of the score have been taken into account.)		

Quote
Relevance: 67%
Framework. The goal of this package is to give an advanced example of how modern templating in TYPO3 CMS can be handled nicely without depending on third party extensions. Bootstrap Package on GitHub

Score	Field	Boost
0 (Inaccurate analysis! Not all parts of the score have been taken into account.)		

That's it. You now have a working TYPO3 Installation with Solr integration. You are able to queue items for indexing, index them and provide an interface for visitors to search the indexed records.

You can now adjust the fluid templates to your needs.

CHAPTER

FIVE

BACKEND

The chapter before gives you a short introduction about how to setup a solr server, the extension and index a few documents into solr. In this chapter we want to go deeper and learn how to write more complex indexing configurations and see what other possibilities the backend of EXT:solr provides.

5.1 ConnectionManager

In EXT:solr all the configuration, including options affecting backend functions, are done in TypoScript. The clear cache menu provides an entry to initialize the Solr connections configured in TypoScript.

5.1.1 How it works

- Configure the Solr connection in TypoScript under plugin.tx_solr.solr, providing host, port, and path.
- On your site's root page set the flag Use as Root Page on the Behaviour tab.
- Initialize the Solr connections through the clear cache menu

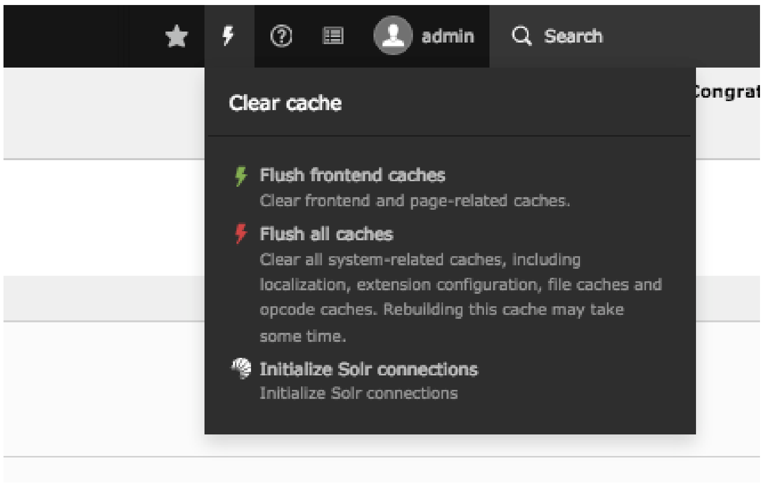

admin
Search
Congrat
Clear cache
Flush frontend caches
Clear frontend and page-related caches.
Flush all caches
Clear all system-related caches, including localization, extension configuration, file caches and opcode caches. Rebuilding this cache may take some time.
Initialize Solr connections
Initialize Solr connections

Fig. 5.1: Initialize all solr connections

When initializing the Solr connections the extensions looks for all the pages with the root flag set, generates the TypoScript configuration for that page like in the frontend and reads the Solr connection parameters.

The extension also repeats that process for each language configured on the installation's root (uid = 0). This way you can configure different Solr cores for each language by using regular conditions that change the path of the Solr connection depending on the currently selected language.

Once all the configured Solr connections in the installation are found, they're stored in TYPO3's registry so that they can easily be retrieved without needing to reevaluate the TypoScript configuration every time we connect to Solr.

All that magic happens in class source:Classes/ConnectionManager.php. The connection manager and it's public API actually must be used whenever a Solr connection is needed.

5.2 IndexQueue Configuration

As you already learned you can index pages very easy with EXT:solr and setup a search for pages in seconds. Beside pages, there might be other records in your TYPO3 CMS that you want to have available in your search results.

5.2.1 Indexing custom records

As a core feature EXT:solr allows you to write custom typoscript configuration to index records from any extension just with configuration. To see how this is working, we open the content of the TypoScript example **"Search - Index Queue Configuration for news"** that can be found in "Configuration/TypoScript/Examples/IndexQueueNews":

```
plugin.tx_solr.index.queue {

    news = 1
    news {
        table = tx_news_domain_model_news

        fields {
            abstract = teaser

            author = author
            authorEmail_stringS = author_email

            title = title

            content = SOLR_CONTENT
            content {
                cObject = COA
                cObject {
                    10 = TEXT
                    10 {
                        field = bodytext
                        noTrimWrap = || |
                    }
                }
            }

            category_stringM = SOLR_RELATION
            category_stringM {
                localField = categories
                multiValue = 1
            }

            keywords = SOLR_MULTIVALUE
            keywords {
                field = keywords
            }

            tags_stringM = SOLR_RELATION
            tags_stringM {
                localField = tags
                multiValue = 1
            }

            url = TEXT
            url {
                typolink.parameter =  {$plugin.tx_news.settings.detailPid}
                typolink.additionalParams = &tx_news_pi1[controller]=News&
                    tx_news_pi1[action]=detail&tx_news_pi1[news]={field:uid}
                typolink.additionalParams.insertData = 1
                typolink.useCacheHash = 1
                typolink.returnLast = url
            }
        }

        attachments {
            fields = related_files
        }
    }

}

plugin.tx_solr.logging.indexing.queue.news = 1
```

By reading the example above you might recognize the following facts:

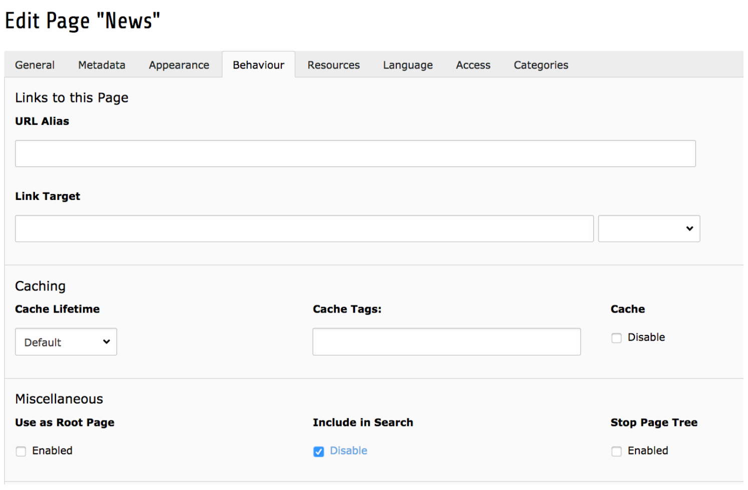
- The indexing configuration is done in the TypoScript path 'plugin. tx_solr. index. queue. [configName]' and there can be multiple queue configurations.
- The database table is configured in the property 'plugin.tx_solr.index.queue.[configName].table'. This allows you to have multiple index queue configurations for the same database table. This can be helpful when you have multiple queue configurations for news (e.g. if you have a press & corporate news section on your website).
- The solr fields are configured in 'plugin.tx_solr.index.queue.[configName].fields'. This allows you to flexibly fill any solr field. The combination of dynamic fields (*Appendix - Dynamic Fields*) and the queue configuration allows you to write any kind of data into solr without adapting the solr schema.
- **There are custom TypoScript objects from EXT:solr that are used in the index queue configuration**
  - *SOLR_CONTENT*
  - *SOLR_RELATION*
  - *SOLR_MULTIVALUE*

When the index queue configuration of your custom record is ready, you can check the index queue in the backend module and add the news items to the queue.

5.2.2 Custom records - links and detail page

In the example above *typolink* is used to build a link to the detail view of the news. This is required, because EXT:solr can not know the business logic of the news extension to build a detail link. The typoscript constant "plugin.tx_news.settings.detailPid" is used to configure the target pageId of the news single view. This has two important impacts:

- The constant (*plugin.tx_news.settings.detailPid*) need to point to a valid news single page.
- The page with the news single view, should be configured with *"Include in Search => Disable"* because indexing this page with the normal page indexing without a news id will produce an error page.



Edit Page "News"

General | Metadata | Appearance | Behaviour | Resources | Language | Access | Categories

Links to this Page

URL Alias

Link Target

Caching

Cache Lifetime: Default

Cache Tags:

Cache: Disable

Miscellaneous

Use as Root Page: Enabled

Include in Search: [✓ Disable]

Stop Page Tree: Enabled

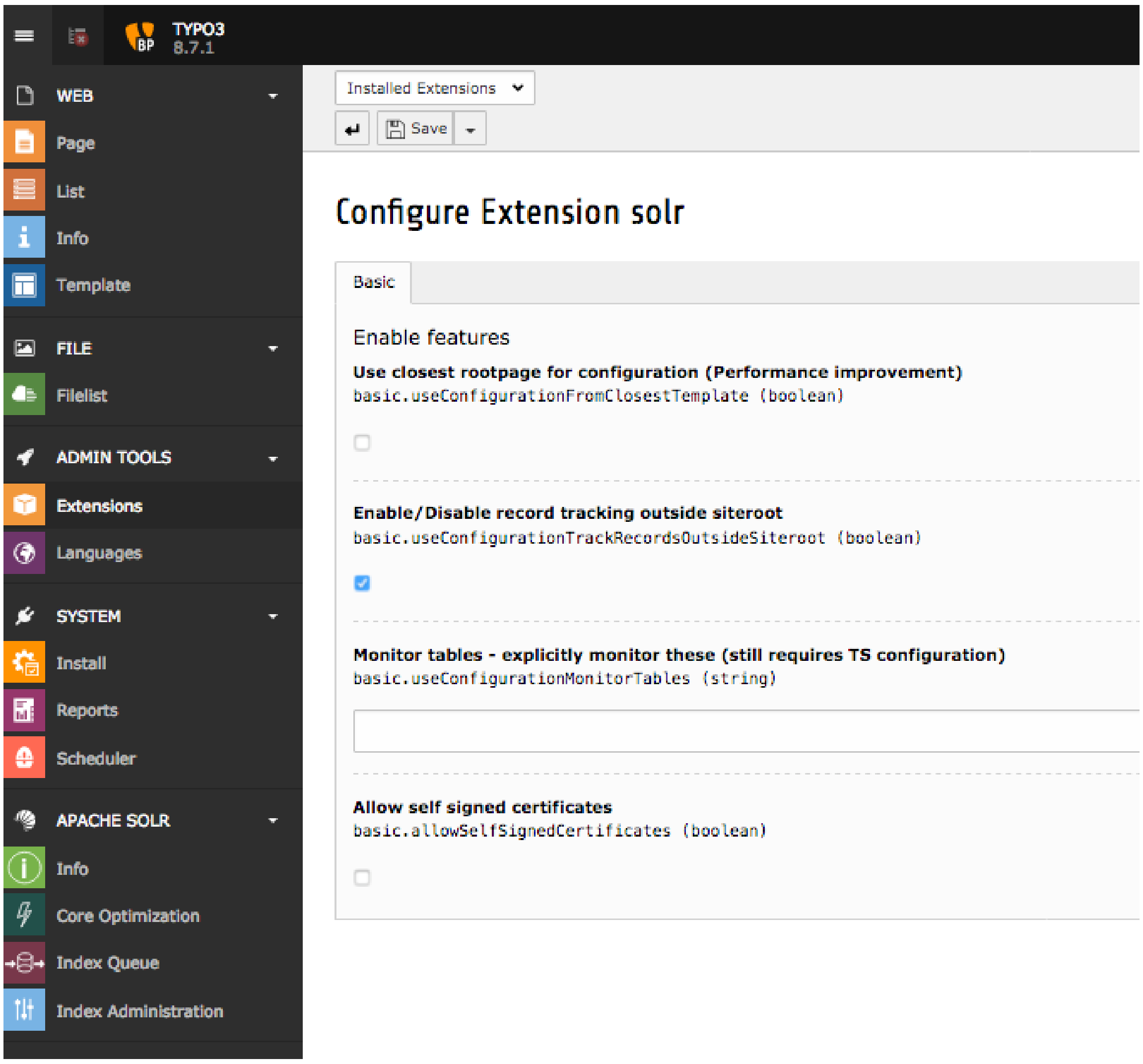
Fig. 5.2: Include in Search - Disable

5.2.3 Sysfolders outside the siteroot

The page with a domain record act as a siteroot in EXT:solr. It is a good practice not to nest the siteroots and do the configuration on the root page. Changes on records that are done in the TYPO3 backend are detected and the solr document will be readded to the index queue when something was changed.

By default only records are monitored for a site that are in the tree of the site. If you want to index records and detect changes on records in a different siteroot, the index queue configuration needs to contains "additionalPageIds" (e.g.: 'plugin.tx_solr.index.queue.<queueName>.additionalPageIds = 45,48').

Since the monitoring of changes in these records is expensive from performance perspective, you need to enable this feature in the extension configuration:



Configure Extension solr

Basic

Enable features

Use closest rootpage for configuration (Performance improvement)
`basic.useConfigurationFromClosestTemplate (boolean)`

☐

Enable/Disable record tracking outside siteroot
`basic.useConfigurationTrackRecordsOutsideSiteroot (boolean)`

☑

Monitor tables - explicitly monitor these (still requires TS configuration)
`basic.useConfigurationMonitorTables (string)`

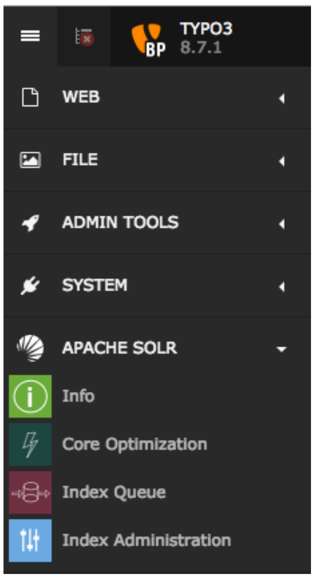
Allow self signed certificates
`basic.allowSelfSignedCertificates (boolean)`

☐

Fig. 5.3: Enable tracking of records outside siteroot

5.3 Backend Modules

The backend modules are in the "APACHE SOLR" section available and can be unlocked for BE-users or/and groups. The modules help you to do maintenance tasks and get an overview on the system status:



During the next paragraphs we will go over the modules and explain, what could be done with them.

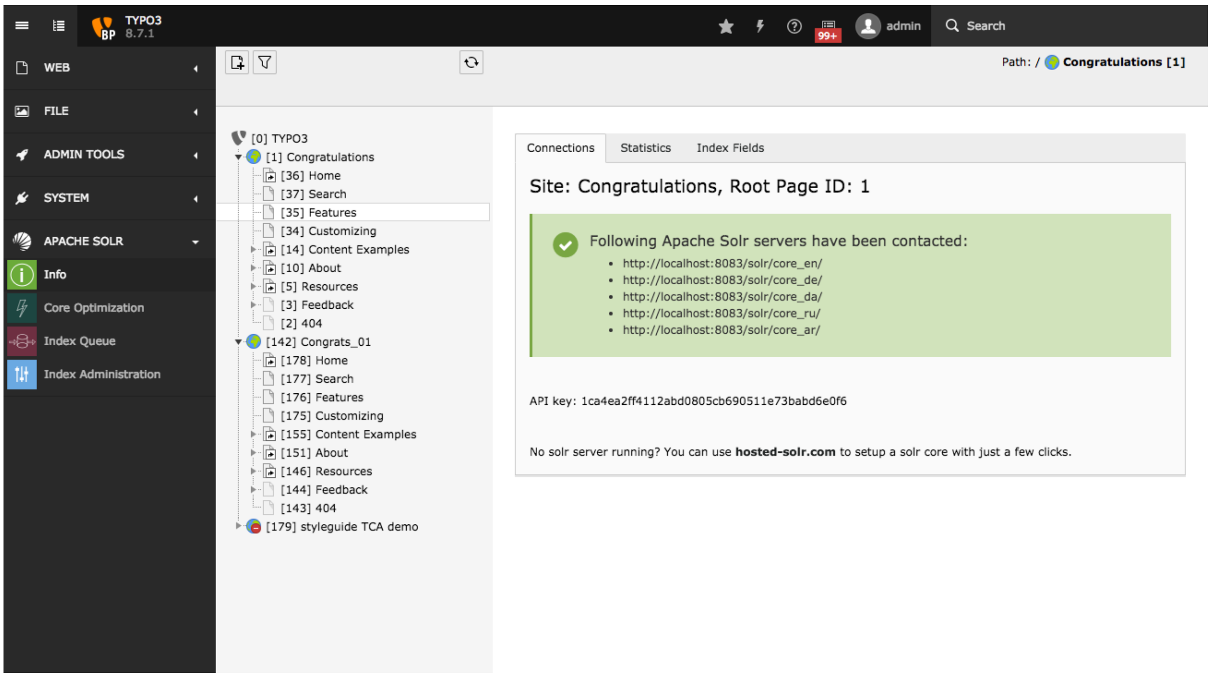
5.3.1 Info Module

Info Module shows you important infos about TYPO3 CMS and Solr state.

It contains different info tabs described below:

Connections

It lists all the for the site configured connections and their status.



Site: Congratulations, Root Page ID: 1

Following Apache Solr servers have been contacted:

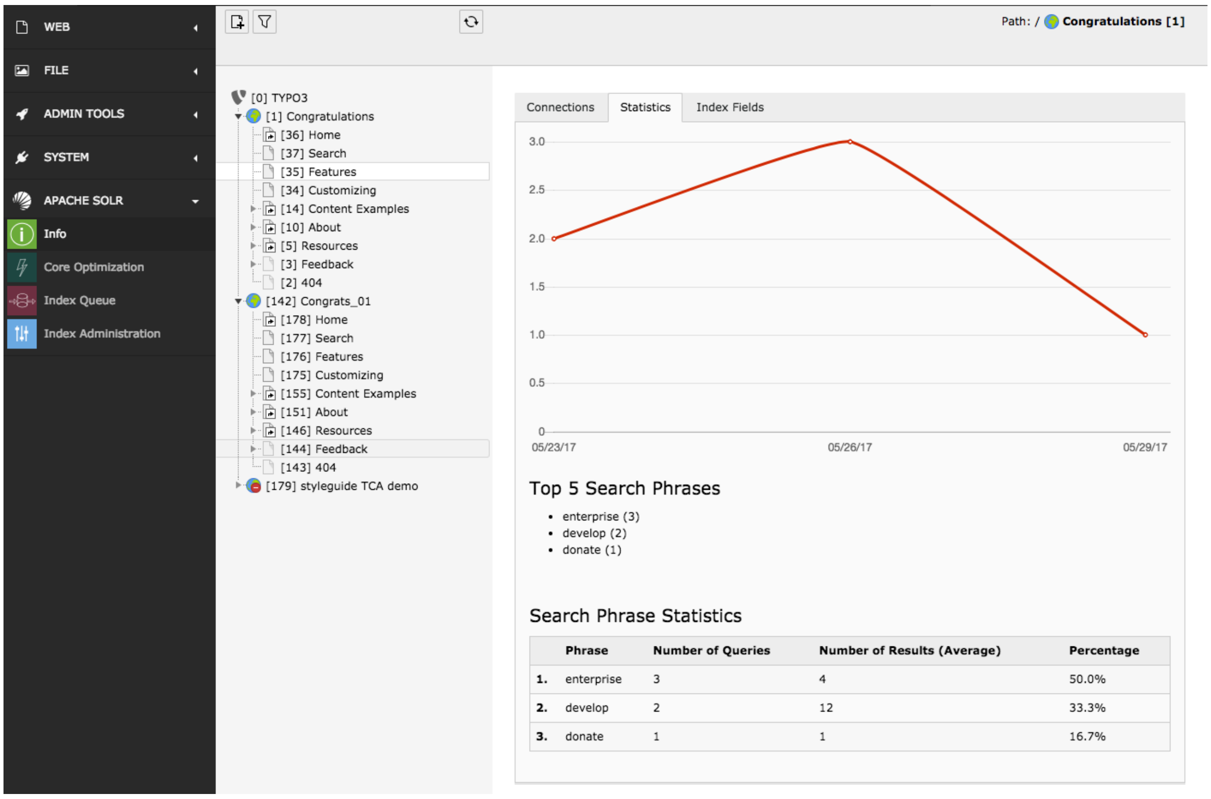
- http://localhost:8083/solr/core_en/
- http://localhost:8083/solr/core_de/
- http://localhost:8083/solr/core_da/
- http://localhost:8083/solr/core_ru/
- http://localhost:8083/solr/core_ar/

API key: 1ca4ea2ff4112abd0805cb690511e73babd6e0f6

No solr server running? You can use **hosted-solr.com** to setup a solr core with just a few clicks.

Statistics

The **Search Statistics** module allows you to see Top Search Phrases with and without results. In addition it possible to see a complete listing with hits etc. ranked by Top search keywords.



Top 5 Search Phrases

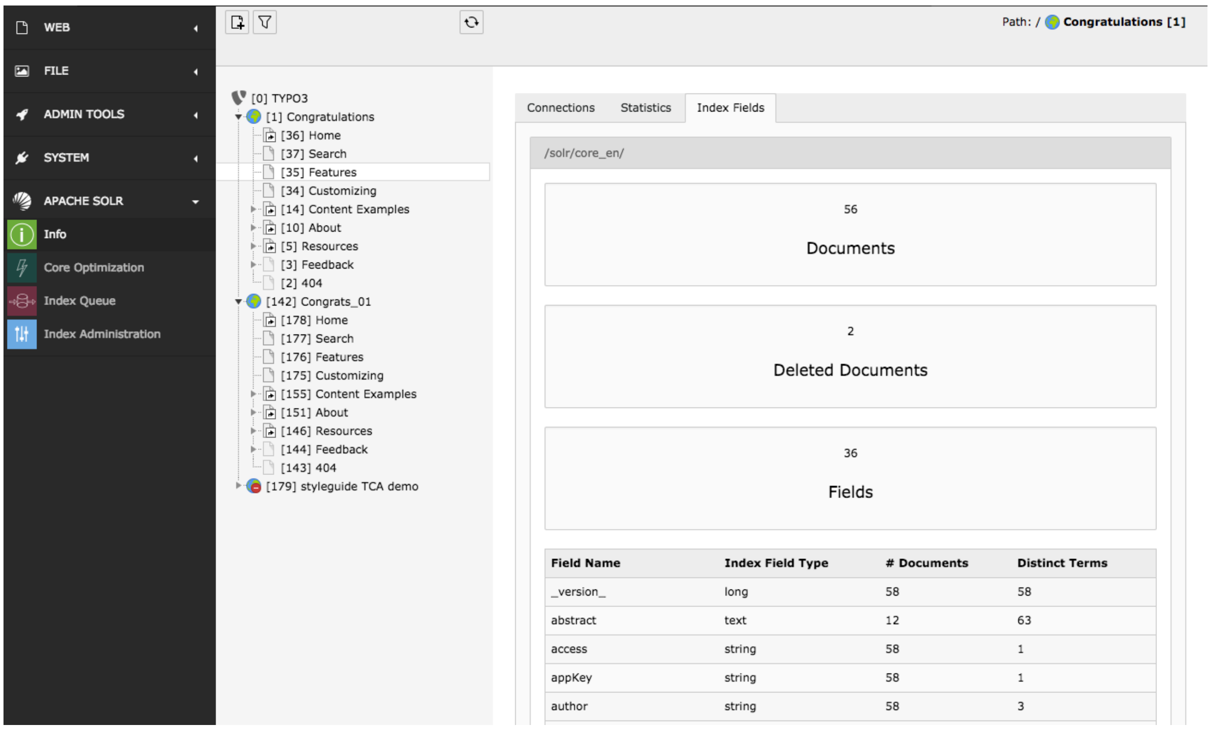
- enterprise (3)
- develop (2)
- donate (1)

Search Phrase Statistics

	Phrase	Number of Queries	Number of Results (Average)	Percentage
1.	enterprise	3	4	50.0%
2.	develop	2	12	33.3%
3.	donate	1	1	16.7%

Index Fields

The **Index Fields** module allows you to see, how many documents you have in which solr core and which fields those documents have.



/solr/core_en/

56
Documents

2
Deleted Documents

36
Fields

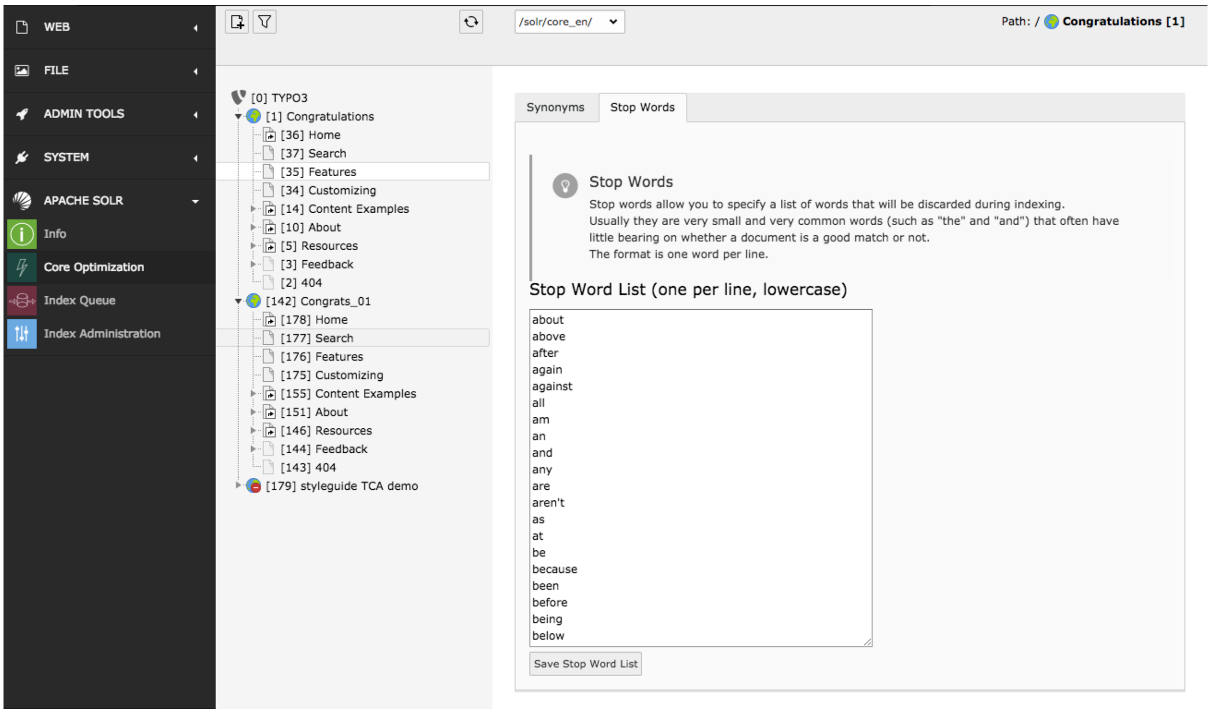
Field Name	Index Field Type	# Documents	Distinct Terms
version	long	58	58
abstract	text	12	63
access	string	58	1
appKey	string	58	1
author	string	58	3

5.3.2 Core Optimization

Core optimization Module is responsive for managing the behaviour of cores. By modifying of following things, you can also change the ranking and/or the results.

Stop Words

With the stopwords module you can define a list of words that should be excluded from the search.



Stop Words

Stop words allow you to specify a list of words that will be discarded during indexing.
Usually they are very small and very common words (such as "the" and "and") that often have little bearing on whether a document is a good match or not.
The format is one word per line.

Stop Word List (one per line, lowercase)

```
about
above
after
again
against
all
am
an
and
any
are
aren't
as
at
be
because
been
before
being
below
```

Common usecases are:

- Very often occurring words like “the”, “and” ... are excluded are filtered out because they are more or less “noize words”.
- You can add words that you want to avoid from indexing.

Synonyms

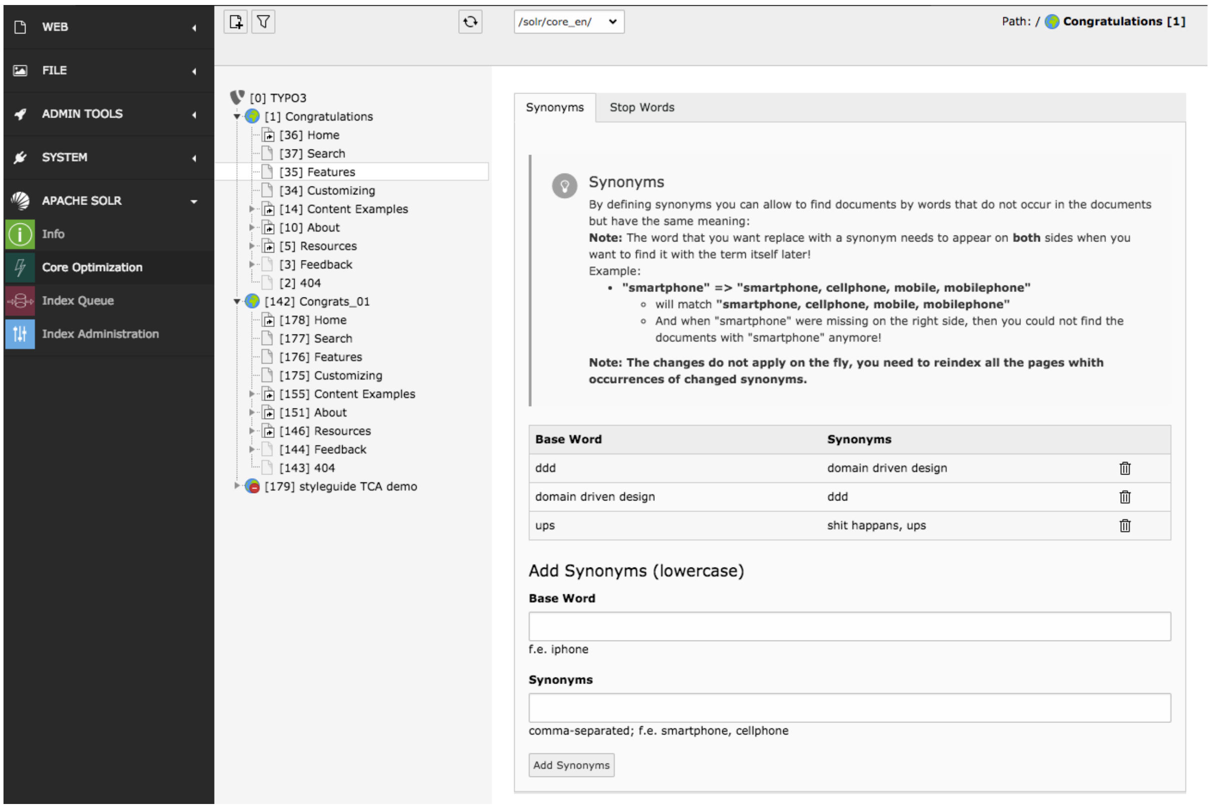
With the synonyms module you can allow to find documents by words that do not occur in the document but have the same meaning:

- E.g. smartphone, cellphone, mobile, mobilephone

Note: The word that you want replace with a synonym needs to appear on **both** sides when you want to find it with the term itself later

Example

smartphone => smartphone, cellphone, mobile, mobilephone will match “smartphone, cellphone, mobile, mobilephone”, when smartphone is missing on the right side, you will not find the document for smartphone anymore!



Synonyms

By defining synonyms you can allow to find documents by words that do not occur in the documents but have the same meaning:
Note: The word that you want replace with a synonym needs to appear on **both** sides when you want to find it with the term itself later!
Example:

- **"smartphone" => "smartphone, cellphone, mobile, mobilephone"**
  - will match **"smartphone, cellphone, mobile, mobilephone"**
  - And when "smartphone" were missing on the right side, then you could not find the documents with "smartphone" anymore!

Note: The changes do not apply on the fly, you need to reindex all the pages whith occurrences of changed synonyms.

Base Word	Synonyms	
ddd	domain driven design	🗑
domain driven design	ddd	🗑
ups	shit happans, ups	🗑

Add Synonyms (lowercase)

Base Word

f.e. iphone

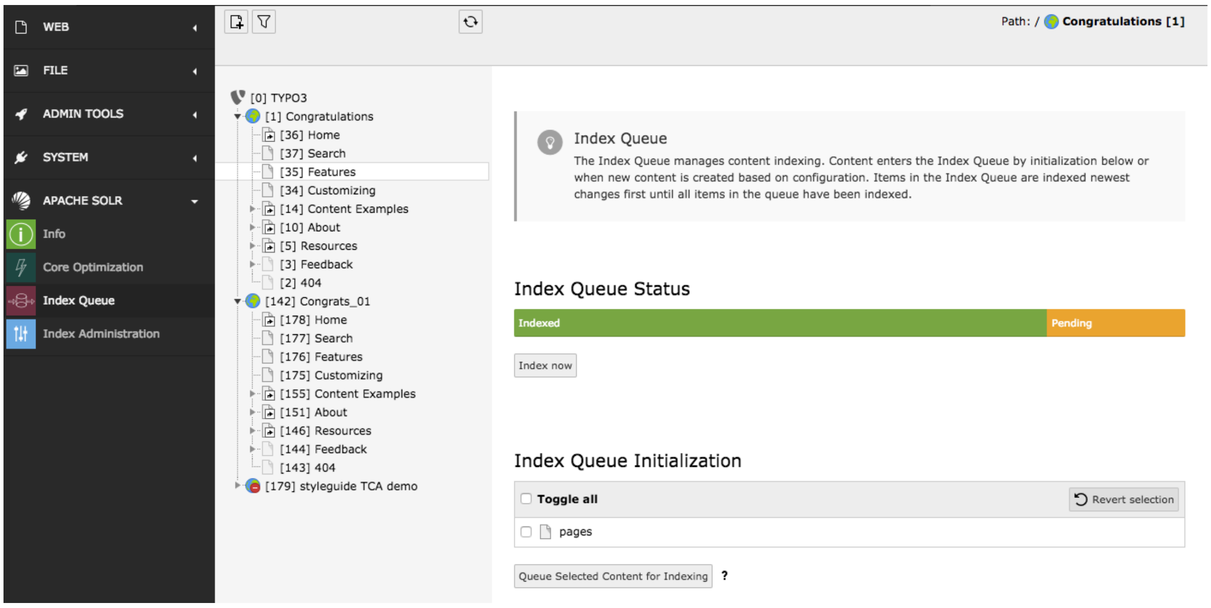
Synonyms

comma-separated; f.e. smartphone, cellphone

5.3.3 Index Queue

The **Index Queue** module is the most important module. It allows you to do the following things:

- Select item types and add them for indexing to the indexing queue.
- See the fill state of the indexing queue.
- Check the indexing queue for errors when the indexing of an items failed.
- Start an instant indexing run, directly from the module.
- Clear the indexing queue and re-queue items.



Index Queue

The Index Queue manages content indexing. Content enters the Index Queue by initialization below or when new content is created based on configuration. Items in the Index Queue are indexed newest changes first until all items in the queue have been indexed.

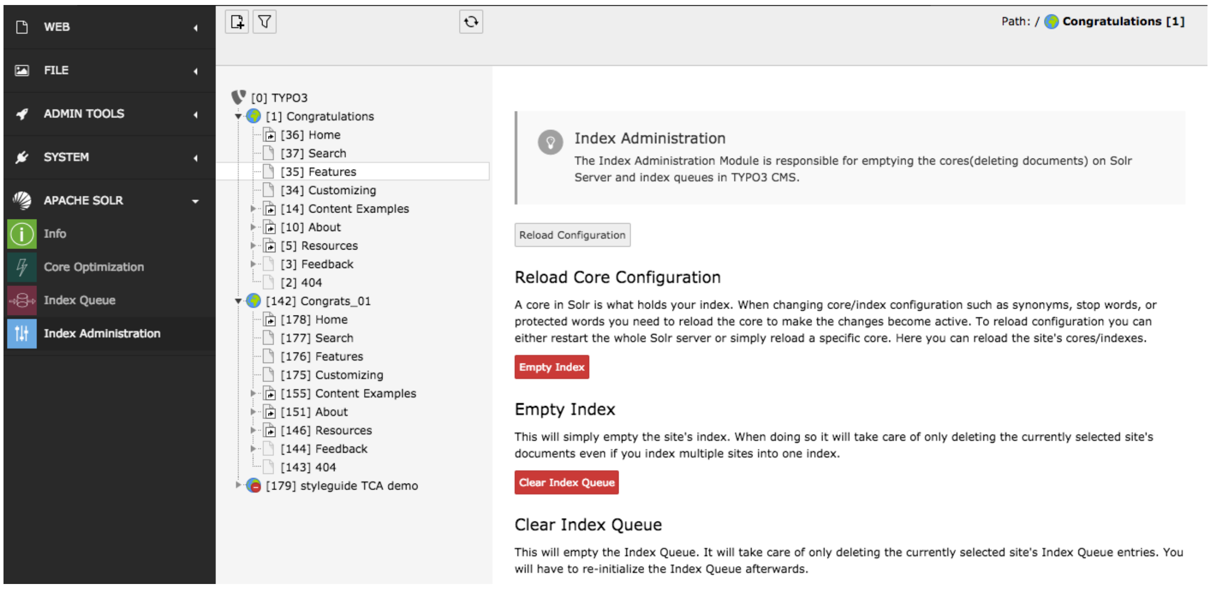
Index Queue Status

Index Queue Initialization

5.3.4 Index Administration (earlier Index Maintenance)

The **Index Administration** module allows you, to do the following administrative operations on your solr index:

- Reload the solr configuration.
- Empty your solr index. This removes all documents from the index of the current selected site.
- Clear the indexing queue.



Index Administration

The Index Administration Module is responsible for emptying the cores(deleting documents) on Solr Server and index queues in TYPO3 CMS.

Reload Core Configuration

A core in Solr is what holds your index. When changing core/index configuration such as synonyms, stop words, or protected words you need to reload the core to make the changes become active. To reload configuration you can either restart the whole Solr server or simply reload a specific core. Here you can reload the site's cores/indexes.

Empty Index

This will simply empty the site's index. When doing so it will take care of only deleting the currently selected site's documents even if you index multiple sites into one index.

Clear Index Queue

This will empty the Index Queue. It will take care of only deleting the currently selected site's Index Queue entries. You will have to re-initialize the Index Queue afterwards.

5.4 Index Inspector

Beside the own backend module, EXT:solr provides a feature called **"Search Index Inspector"**. This Tool allows you to select a page or sysfolder and check what data is stored for this entity in the solr index.

The **"Search Index Inspector"** can be opened with the TYPO3 Info Module (*"Web > Info > Search Index Inspector"*) it shows the stored data in solr from the page or sysfolder that is selected in the pagetree:

5.5 Scheduler

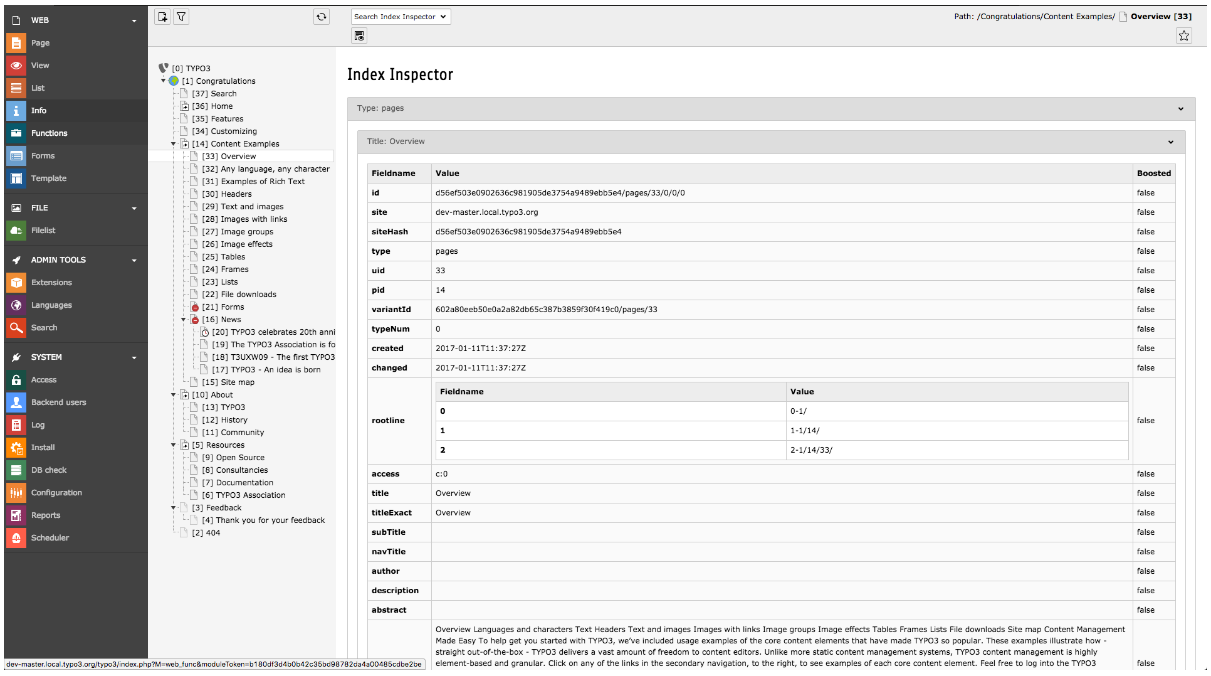
When you want to index content from TYPO3 into solr automatically EXT:solr ships scheduler tasks, that can be called at a configured time or directly from the backend.

5.5.1 Index Queue Worker

Changes that are done in the backend by an editor are written into a queue. This queue is processed asynchronously with a scheduler task and each item in the queue is indexed into solr.

The **"Index Queue Worker"** task has the following custom properties:

Site: Here you select the solr site, that you want to index with this task instance. **Number of documents to Index:** Here you can configure how many documents you want to index in one run. Depending on the performance of your system and the expected update time of the search you can choose a realistic number here. **Forced webroot:** The scheduler task can be executed in the cli context, because no webserver is used there, TYPO3 is unable to detect your webroot. As assumption we use PATH_site as default here. When you need to configure something else, you can do it with this option. You can use the marker ###PATH_site### and ###PATH_typo3### to define relative pathes here, to be independent from the concrete instance.



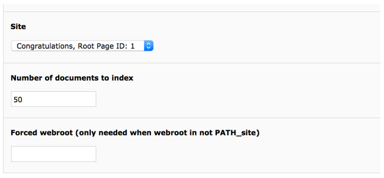
Index Inspector

Type: pages

Title: Overview

Fieldname	Value	Boosted
id	d56ef503e0902636c981905de3754a9489ebb5e4/pages/33/0/0/0	false
site	dev-master.local.typo3.org	false
siteHash	d56ef503e0902636c981905de3754a9489ebb5e4	false
type	pages	false
uid	33	false
pid	14	false
variantId	602a80eeb50e0a2a82db65c387b3859f30f419c0/pages/33	false
typeNum	0	false
created	2017-01-11T11:37:27Z	false
changed	2017-01-11T11:37:27Z	false
rootline	**Fieldname** / **Value**: **0** 0-1/; **1** 1-1/14/; **2** 2-1/14/33/	false
access	c:0	false
title	Overview	false
titleExact	Overview	false
subTitle		false
navTitle		false
author		false
description		false
abstract		false
	Overview Languages and characters Text Headers Text and images Images with links Image groups Image effects Tables Frames Lists File downloads Site map Content Management Made Easy To help get you started with TYPO3, we've included usage examples of the core content elements that have made TYPO3 so popular. These examples illustrate how - straight out-of-the-box - TYPO3 delivers a vast amount of freedom to content editors. Unlike more static content management systems, TYPO3 content management is highly element-based and granular. Click on any of the links in the secondary navigation, to the right, to see examples of each core content element. Feel free to log into the TYPO3	false

Fig. 5.4: The EXT:solr Search Index Inspector



Site

Congratulations, Root Page ID: 1

Number of documents to index

50

Forced webroot (only needed when webroot in not PATH_site)

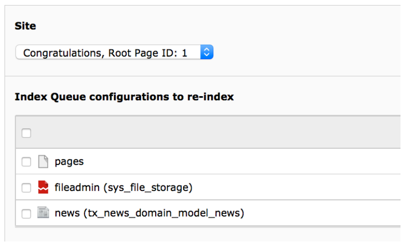
Fig. 5.5: The EXT:solr Index Queue Worker - Scheduler Task

5.5.2 Force Re-Indexing of a site

This task allows you to force the re-indexing of a site & indexing configuration at a planned time.

The **Force Re-Indexing of a site** task has the following custom properties:

Site: Here you select the solr site, that you want to index with this task instance. **Index Queue configurations to re-index**: Here you can limit the set of indexing configurations that should be Re-Indexed.



Site: Congratulations, Root Page ID: 1

Index Queue configurations to re-index

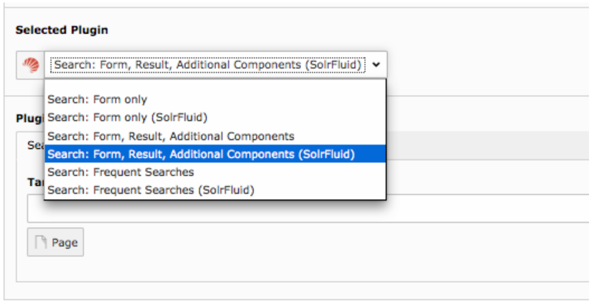
- pages
- fileadmin (sys_file_storage)
- news (tx_news_domain_model_news)

Fig. 5.6: The EXT:solr Force Re-Indexing of a site - Scheduler Task

5.6 Plugins

EXT:solr provides the following plugin instances that can be configured in the backend:

- Results plugin: **"Search: Form, Result, Additional Components"**
- Form plugin: **"Search: Form only"**
- Frequent Searches plugin: **"Search: Frequent Searches"**



Selected Plugin: Search: Form, Result, Additional Components (SolrFluid)

- Search: Form only
- Search: Form only (SolrFluid)
- Search: Form, Result, Additional Components
- Search: Form, Result, Additional Components (SolrFluid)
- Search: Frequent Searches
- Search: Frequent Searches (SolrFluid)

5.7 Results Plugin

The results plugin is the most important plugin of the extension. It is responsible to render a search form and the results.

5.7.1 Flexform Configuration

All configuration can be done with TypoScript and the settings from EXT:solr are used. For some settings it makes sence to overwrite them with the flexform in the plugin settings.

The following settings can be overwritten by instance with the flexform:

"Target Page":

Target page that should be used when a search is submitted. This can be usefull when you want to show the results on another page.

When nothing is configured the current page will be used.

Overwritten TypoScript Path	plugin.tx_solr.search.targetPage
Type:	Integer

“Initialize search with empty query”:

If enabled, the results plugin issues a “get everything” query during initialization. This is useful, if you want to create a page that shows all available facets although no search has been issued by the user yet.

Note: Enabling this option alone will not show results of the get everything query. To also show the results of the query, see option *Show results of initial empty query* below.

Overwritten TypoScript Path	plugin.tx_solr.search.initializeWithEmptyQuery
Type:	Boolean

“Show results of initial empty query”:

Requires **“Initialize search with empty query”** (above) to be enabled to have any effect. If enabled together with **“Initialize search with empty query”** the results of the initial “get everything” query are shown. This way, in combination with a filter you can easily list a predefined set of results.

Overwritten TypoScript Path	plugin.tx_solr.search.showResultsOfInitialEmptyQuery
Type:	Boolean

“Initialize with query”:

This configuration can be used to configure an initial query string that is triggered when the plugin is rendered.

Overwritten TypoScript Path	plugin.tx_solr.search.initializeWithQuery
Type:	String

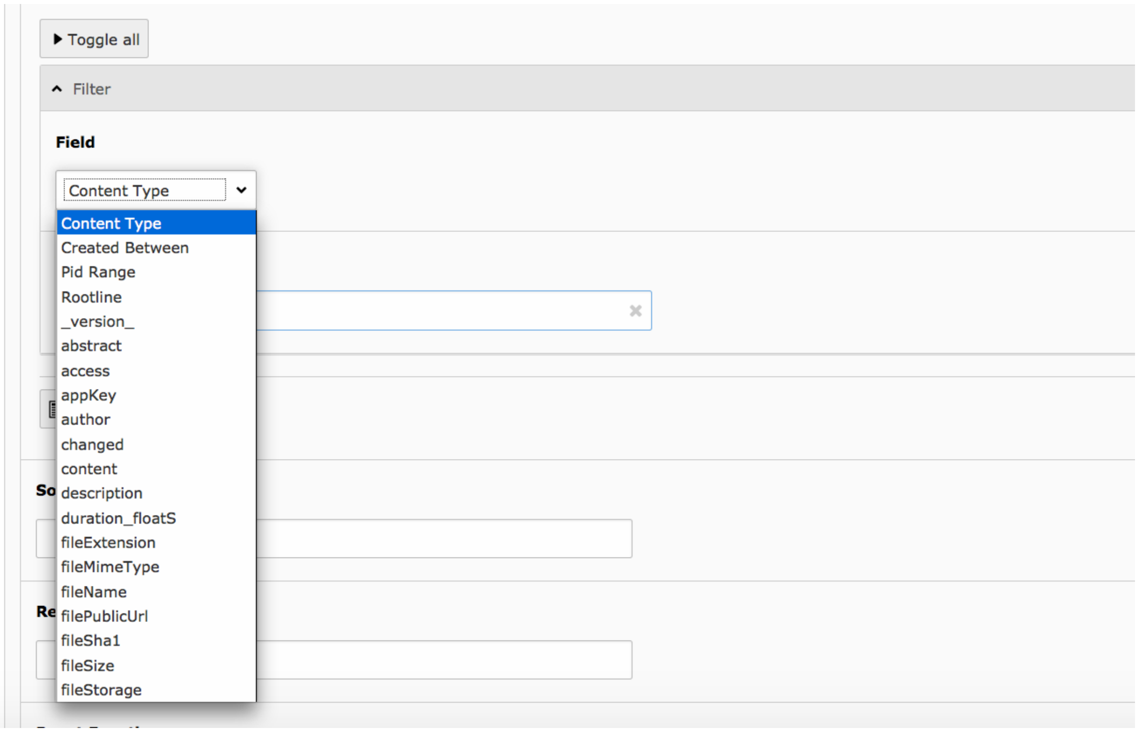
“Show results of initial query”:

This option is used to configure if the results of an initial query should be shown.

Overwritten TypoScript Path	plugin.tx_solr.search.showResultsOfInitialQuery
Type:	Boolean

“Filters”:

This flexform element allows you to define custom filters by selecting a solr field and a value:



Overwritten TypoScript Path	plugin.tx_solr.search.query.filter.
Type:	Array

"Sorting":

When you want to sort initially by a field value and not by relevance this can be configured here.

Overwritten TypoScript Path	plugin.tx_solr.search.query.sortBy
Type:	String
Example:	title desc

"Boost Function":

A boost function can be useful to influence the relevance calculation and boost some documents to appear more at the beginning of the result list. Technically the parameter will be mapped to the **"bf"** parameter in the solr query.

Use cases for example could be:

"Give never documents a higher priority":

This could be done with a recip function:

```
recip(ms(NOW,created),3.16e-11,1,1)
```

"Give documents with a certain field value a higher priority":

This could be done with:

```
termfreq(type,'tx_solr_file')
```

Overwritten TypoScript Path	plugin.tx_solr.search.query.boostFunction
Type:	String
Example:	recip(ms(NOW,created),3.16e-11,1,1)

See also:

https://cwiki.apache.org/confluence/display/solr/The+DisMax+Query+Parser#TheDisMaxQueryParser-Thebf%28BoostFunctions%29Parameter https://cwiki.apache.org/confluence/display/solr/Function+Queries

"Boost Query":

The boostQuery is a query that can be used for boosting. Technically it is mapped to the **"bq"** parameter of the solr query. Compared to boost a function a boost query provides less use cases.

An example could be to boost documents based on a certain field value:

type:tx_solr_file

Overwritten TypoScript Path	plugin.tx_solr.search.query.boostQuery
Type:	String
Example:	type:tx_solr_file

See also:

https://cwiki.apache.org/confluence/display/solr/The+DisMax+Query+Parser#TheDisMaxQueryParser-Thebq%28BoostQuery%29Parameter

CHAPTER

SIX

FRONTEND

This part describes the frontend part of EXT:solr. Since version 7.0.0 the templating is done with the fluid templating engine.

6.1 Concepts

Since EXT:solr 7.0.0 the old templating of EXT:solr was droppend and rendering with fluid was added.

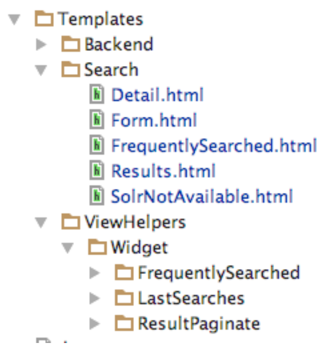
Along with this change some concepts have changed:

- Until EXT:solr 7.0.0 EXT:solr css and javascript was loaded by EXT:solr automatically. In most cases you want to use custom css or you have custom javascript and the integrator want to decide which css or javascript to use. Therefore EXT:solr does not load it by default anymore and the integrator can load it with typoscript. EXT:solr provides a lot of example typoscript templates that load the default css or load the javascript that is needed to use a specific feature. Maybe take the time to explore the typoscript templates that are shipped with the extension to see how they are implemented.
- **The were some old typosript settings that manipulate the data before it was passed to the view. With fluid this can also**
  - plugin.tx_solr.search.results.fieldRenderingInstructions
  - plugin.tx_solr.search.results.fieldProcessingInstructions (The fieldProcessingInstructions still exist at index time since there it is still needed)
- **Beside that there were some template related settings in the typoscript, that can be solved just with fluid:**
  - plugin.tx_solr.search.faceting.facetLinkATagParams or plugin.tx_solr.search.faceting.[facetName].facetLinkATagPara When you need something like this, you can just change the partials or render a facet with a custom partial (partialName = MyPartial) and add the properties there.
  - plugin.tx_solr.search.faceting.removeFacetLinkText This can be done just be rendering the text in the partial, that you need.
- The setting faceting.facets.[facetName].selectingSelectedFacetOptionRemovesFilter has been removed, since it is possible to build this functionality just with Fluid ViewHelpers. The file “EXT:solr/Resources/Private/Templates/Partials/Facets/OptionsToggle.html” shows

how f:if together with option.selected can be used to have this behaviour.

6.2 Fluid Template Structure

First we start with a short overview of the template structure. This is just to get an rought overview. The templates will be explained in detail in the template where they belong to:



- Layouts: Layouts that are used in the search and the faceting.
- **Partials:**
  - **Facets**: Partials that are use to render the specific facet types.
  - **Result**: Partials that are used during the result rendering (e.g. to render the result document, sorting or perPage selector)
  - **Search**: Partials that are used for the search also when no search was executed.
- **Templates:**
  - **Search**: All templates that are used to render the actions in the SearchController
  - **ViewHelper**: All templates that are use in the widgets (FrequentSearched, LastSearches, ResultPaginate)

All backend related files are in a "Backend" folder. Everything else is frontend related.

6.3 Result List

The most important part of a search are the results. The rendering of the results is done in the "Results.html" template (Located in Templates/Search/Results.html)

The following part of the default template iterates over the results and renders every document with the Document.html partial (Partials/Frontend/Result/Document.html)

```
<s:widget.resultPaginate resultSet="{resultSet}">
        <ol start="{pagination.displayRangeStart}" class="results-list">
                <f:for each="{documents}" as="document">
                        <f:render partial="Result/Document" section="Document"
                         arguments="{resultSet:resultSet, document:document}" />
                </f:for>
        </ol>
</s:widget.resultPaginate>
```

This structure allows you to use e.g. the fluid if ViewHelper to render a result with a different partial, based on a field value. But as you see in the template above, by default the partial "Result/Document" is used.

The "document" partial is getting the document object. In our case this is an instance of "ApacheSolrForTypo3SolrfluidDomainSearchResultSetSearchResult" the api of this object allows to get the solr field content with "Document->getFieldName()" that can be used as "document.fieldName" in fluid.

6.4 Facets

The goal of a good search is, that the user will find what he is looking for as fast as possible. To support this goal you can give information from the results to the user to "drill down" or "filter" the results up to a point where he exactly finds what he was looking for. This concept is called "faceting".

Imagine a user in an online shoe shop is searching for the term “shoe”, wouldn’t it be useful to allow the user to filter by “gender”, “color” and “brand” to find exactly the model where he is looking for?

In the following paragraphs we will get an overview about the different facet types that can be created on a solr field just by adding a few lines of configuration.

6.4.1 Facet Types

A solr field can contain different type of data, where different facets make sence. The simplest facet is an option “facet”. The “options facet” just contains a list of values and the user can choose one or many of them. A more complex type could be a “range facet” on a price field. A facet like this needs to allow to filter on a range of a minimum and a maximum value.

The “type” of a facet can be controlled with the “type” property. When nothing is configured there, the facet will be threated as option facet.

```
plugin.tx_solr.search.faceting.facets.[faceName].type = [typeName]
```

Valid types could be: options | queryGroup | hierarchy | dateRange | numericRange

In the following paragraphs we will introduce the available facet types in EXT:solr and show how to configure them.

Option

The simplest and most often used facet type is the options facet. It renders the items that could be filtered as a simple list.

To setup an simple options facet you can use the following TypoScript snipped:

```
plugin.tx_solr.search {
    faceting = 1
    faceting {
        facets {
            contentType {
                label = Content Type
                field = type
            }
        }
    }
}
```

By using this configuration you create an options facet on the solr field “type” with the name “contentType”. This field represents the record type, that was indexed into solr. Shown in the frontend it will look like this:

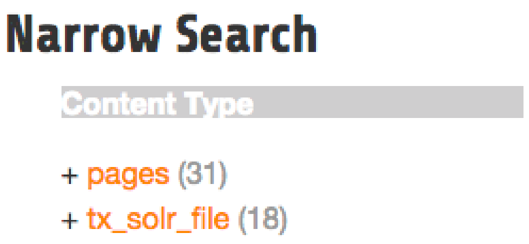


Fig. 6.1: Options Facet

Summary:

Type	options
DefaultPartial	Partials\Facets\Options.html
Domain Classes	Domain\Search\ResultSet\Facets\OptionBased\Options\*

Grouping by option prefix:

When you have an option facet with very much options you might want to group the options by an prefix of an option. This can be used e.g. to group the options alphabetically.

The following example shows how options can be grouped by prefix (from EXT:solr/Configuration/TypoScript/Examples/Facets/OptionsPrefixGrouped/setup.txt):

```
<s:facet.options.group.prefix.labelPrefixes options="{facet.options}" length="1"␣
↪sortBy="alpha">
    <f:for each="{prefixes}" as="prefix">
        <li>
            {prefix}
            <ul>
                <s:facet.options.group.prefix.labelFilter options="{facet.options}
↪" prefix="{prefix}">
                    <f:for each="{filteredOptions}" as="option">
                        <li class="facet-option" data-facet-item-value="{option.
↪value}">
                            + <a class="facet solr-ajaxified" href="{s:uri.facet.
↪addFacetItem(facet: facet, facetItem: option)}">{option.label}</a>
                            <span class="facet-result-count">({option.documentCount}
↪)</span>
                        </li>
                    </f:for>
                </s:facet.options.group.prefix.labelFilter>
            </ul>
        </li>
    </f:for>
</s:facet.options.group.prefix.labelPrefixes>
```

Query Group

The query group facet renders an option list, compareable to the options facet, but the single options are not created from plain solr field values. They are created from dynamic queries.

A typical usecase could be, when you want to offer the possiblity to filter on the creation date and want to offer options like “yesterday”, “last year” or “more then five years”.

With the following example you can configure a query facet:

```
plugin.tx_solr.search {
    faceting = 1
    faceting {
        facets {
             age {
                label = Age
                field = created
                type = queryGroup
                queryGroup {
                    week.query = [NOW/DAY-7DAYS TO *]
                    old.query = [* TO NOW/DAY-7DAYS]
                }
            }
        }
    }
}
```

The example above will generate an options facet with the output "week" (for items from the last week) and "old" (for items older then one week).

The output in the frontend will look like this:

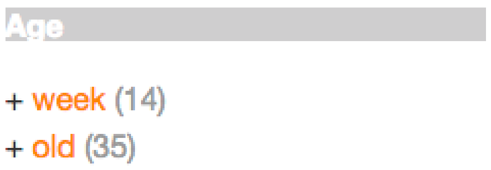


Fig. 6.2: Solr queryGroup facet

An more complex example is shipped with this extension and can be enabled by including the template **"Search - (Example) Fluid queryGroup facet on the field created"**, this example makes also use of renderingInstructions to render nice labels for the facet.

Summary:

Type	queryGroup
DefaultPartial	Partials\Facets\Options.html
Domain Classes	Domain\Search\ResultSet\Facets\OptionBased\QueryGroup\*

Hierarchical

With the hierarchical facets you can render a tree view in the frontend. A common usecase is to render a category tree where a document belongs to.

With the following example you render a very simple rootline tree in TYPO3:

```
plugin.tx_solr.search {
    faceting = 1
    faceting {
        facets {
            pageHierarchy {
                field = rootline
                label = Rootline
                type = hierarchy
            }
         }
    }
}
```

The example above just shows a simple example tree that is just rendering the uid's of the rootline as a tree:

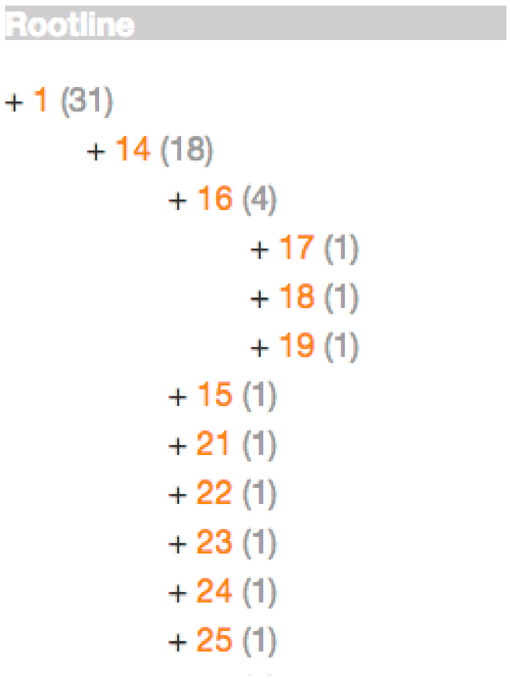
A more complex example, that is rendering the pagetree with titles is shipped in the extension. You can use it by including the example TypoScript **"Search - (Example) Fluid hierarchy facet on the rootline field"**:

Summary:

Type	hierarchy
DefaultPartial	Partials\Facets\Hierarchy.html
Domain Classes	Domain\Search\ResultSet\Facets\OptionBased\Hierarchy\*

Technical solr background:

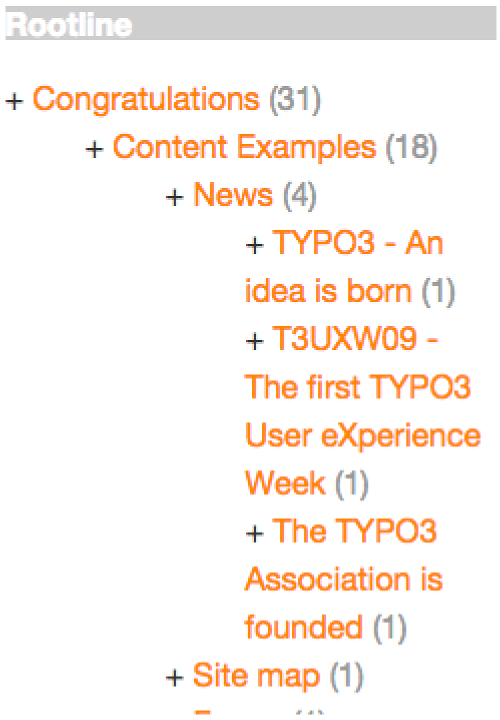
Technically the hierarchical facet for solr is the same as a flat options facet. The support of hierarchies is implemented, by writing and reading the facet options by a convention:



Rootline

- + 1 (31)
  - + 14 (18)
    - + 16 (4)
      - + 17 (1)
      - + 18 (1)
      - + 19 (1)
    - + 15 (1)
    - + 21 (1)
    - + 22 (1)
    - + 23 (1)
    - + 24 (1)
    - + 25 (1)

Fig. 6.3: Hierachy facet



Rootline

- + Congratulations (31)
  - + Content Examples (18)
    - + News (4)
      - + TYPO3 - An idea is born (1)
      - + T3UXW09 - The first TYPO3 User eXperience Week (1)
      - + The TYPO3 Association is founded (1)
    - + Site map (1)

Fig. 6.4: Hierachy rootline facet

```
[depth]-/Level1Label/Level2Label
```

When you follow this convention by writing date into a solr field you can render it as hierarchical facet. As example you can check indexing configuration in EXT:solr (EXT:solr/Configuration/ TypoScript/Solr/setup.txt)

```
plugin.tx_solr {
    index {
        fieldProcessingInstructions {
            rootline = pageUidToHierarchy
        }
    }
}
```

In this case the “fieldProcessingInstruction” “pageUidToHierarchy” is used to create the rootline for solr in the conventional way.

Date Range

When you want to provide a range filter on a date field in EXT:solr, you can use the type **“dateRange”**.

The default partial generates a markup with all needed values in data attributes. Together with the provided jQuery ui implementation you can create an out-of-the-box date range facet.

With the following typoscript you create a date range facet:

```
plugin.tx_solr.search {
    faceting = 1
    faceting.facets {
        creationDateRange {
            label = Created Between
            field = created
            type = dateRange
        }
    }
}
```

In the extension we ship the TypoScript example **“Search - (Example) dateRange facet with jquery ui datepicker on created field”** that shows how to configure a dateRange facet and load all required javascript files.

When you include this template a date range facet will be shown in the frontend that we look like this:

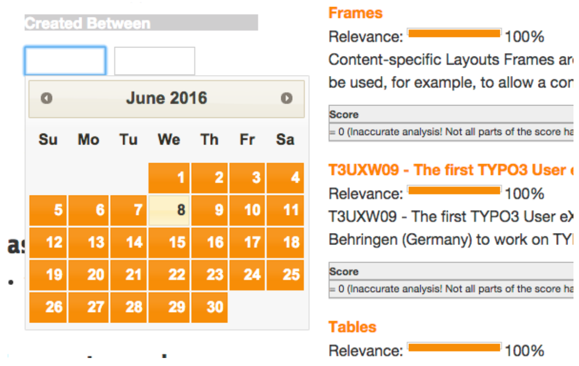

Created Between
June 2016
Su Mo Tu We Th Fr Sa
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
Frames
Relevance: 100%
Content-specific Layouts Frames ar
be used, for example, to allow a cor
Score
= 0 (Inaccurate analysis! Not all parts of the score ha
T3UXW09 - The first TYPO3 User
Relevance: 100%
T3UXW09 - The first TYPO3 User e
Behringen (Germany) to work on TY
Score
= 0 (Inaccurate analysis! Not all parts of the score ha
Tables
Relevance: 100%

Fig. 6.5: EXT:solr dateRange facet

As described before for the date range facet markup and javascript code is required, looking at the example template **“Search - (Example) dateRange facet with jquery ui datepicker on created field”** in “Configuration/TypoScript/Examples/DateRange” you see that for the jQueryUi implementation the following files are included:

```
page.includeJSFooterlibs {
    solr-jquery = EXT:solr/Resources/Public/JavaScript/JQuery/jquery.min.js
    solr-ui = EXT:solr/Resources/Public/JavaScript/JQuery/jquery-ui.min.js
    solr-daterange = EXT:solr/Resources/Public/JavaScript/facet_daterange.js
}

page.includeCSS {
    solr-ui = EXT:solr/Resources/Css/JQueryUi/jquery-ui.custom.css
}
```

Numeric Range

Beside dates ranges are also usefull for numeric values. A typical usecase could be a price slider for a products page. With the user interface you should be able to filter the documents for a certain price range.

In the default partial, we also ship a partial with data attributes here to support any custom implementation. By default we will use the current implementation from EXT:solr based on jQueryUi.

The following example configures a **numericRange** facet for the field **"pid"**:

```
plugin.tx_solr.search {
    faceting = 1
    faceting.facets {
        pidRangeRange {
            field = pid
            label = Pid Range
            type = numericRange
            numericRange {
                start = 0
                end = 100
                gap = 1
            }
        }
    }
}
```

The numeric range facet requires beside the template also a javascript library to render the slider. The example typoscript template **"Search - (Example) Fluid numericRange facet with jquery ui slider on pid field"** can be used to see the range slider with jQuery ui for the solr field pid by example.

When you configure a facet on the pid field like this, the frontend will output the following facet:

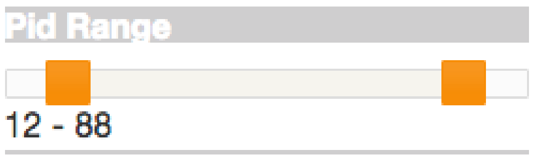


Fig. 6.6: Numeric range facet

Beside the implementation with jQueryUi you are free to implement a range slider with any other javascript framework.

6.4.2 Rendering with fluid

Rendering facets with fluid is very flexible, because you can use existing ViewHelpers and implement your own logic in ViewHelpers to support your custom rendering logic.

In the default template the main faceting area on the left side, is done in the following file:

```
Resources/Private/Partials/Frontend/Result/Facets.html
```

This template is used to render only the area for a few facets. The following part is the relevant part where we itterate over the facets:

```
<s:facet.area.group groupName="main" facets="{resultSet.facets.available}">
    <div class="facet-area-main">
        <div class="solr-facets-available secondaryContentSection">
            <div class="csc-header">
                <h3 class="csc-firstHeader">Narrow Search</h3>
            </div>
            <ul class="facets">
                <f:for each="{areaFacets}" as="facet">
                    <li class="facet facet-type facet-type-{facet.type}">
                        <f:render partial="Facets/{facet.partialName}"
                        arguments="{resultSet:resultSet, facet:facet}"/>
                    </li>
                </f:for>
            </ul>
        </div>
    </div>
</s:facet.area.group>
```

Looking at the code above we see to important details that are important for solr.

Facet Grouping (Areas)

The first important part if the **facet.area.group** ViewHelper. By default all facets in the group **main** will be rendered. This value is the default value.

When you now want to render the facet at another place you can change the group with the following TypoScript configuration:

```
plugin.tx_solr.search {
    faceting = 1
    faceting.facets {
        contentType {
            field = type
            label = Content Type
            groupName = bottom
        }
    }
}
```

Now the facet belongs to another group and will not be rendered in the “main” area anymore.

Default Partials

Another important fact is that *Facet->getPartianName()* is used to render the detail partial. The default implementation of a facet will return the default partial, that is able to render this facet.

If you need another rendering for one facet you can overwrite the used partial within the configuration:

```
plugin.tx_solr.search {
    faceting = 1
    faceting.facets {
        contentType {
            field = type
            label = Content Type
            partialName = mySpecialFacet
        }
    }
}
```

Combining all of these concepts together with the flexibility of fluid you are able to render facets in a very flexible way.

6.5 Autosuggest

A user of the search typically want to find the results a fast as possible. To support the user and avoid to much typing solr can create a drop down list of common suggested search terms right after the search input box.

This feature can be easily configured with the following typoscript setting:

```
plugin.tx_solr.search {
    suggest = 1
    suggest {
        numberOfSuggestions = 10
        suggestField = spell
    }
}
```

Beside the server related part solrfluid ships the jQueryUi autocomplete implementation to show the suggest results. If you want to configure an the autosuggest by example, you can include the typoscript example template **"(Example) Suggest/autocomplete with jquery"**.

When everything is configured the frontend will show you a drop down of suggestions when you are typing in the search field:

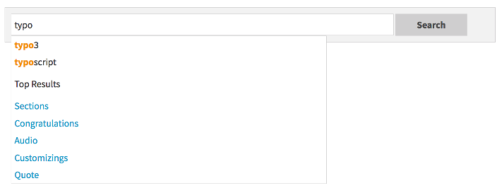


Fig. 6.7: Autocomplete with jQuery

Beside search term suggestions the autocomplete an also show you the top search results for the user input.

This feature can be enabled with:

```
plugin.tx_solr.search.suggest.showTopResults = 1
```

If you want to change the number of proposed top results you can also configure them:

```
plugin.tx_solr.search.suggest.numberOfTopResults = 5
```

6.6 Sorting

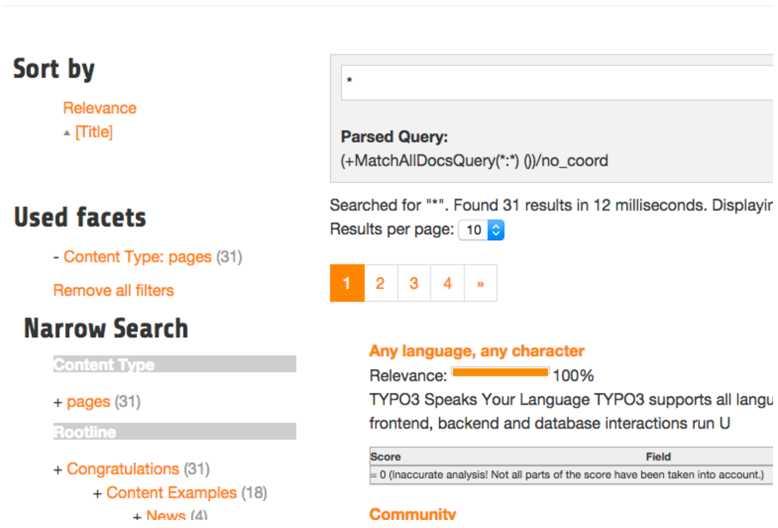
When no sorting is selected the search will order the results by **"relevance"**. This relevance is calculated by many factors and has the goal to deliver the best result for the query on the first position. That's what you expect from a search :)

For some usecases you want to change the sorting of the results by a certain field. In an onlineshop a user might want to order the results by the price to find the cheapest product that is matching his query.

A simple sorting can be configured with the following typoscript snipped:

```
plugin.tx_solr.search.sorting >
plugin.tx_solr.search {
        sorting = 1
        sorting {
                defaultOrder = asc

                options {
                        relevance {
                                field = relevance
                                label = Relevance
                        }

                        title {
                                field = sortTitle
                                label = Title
                        }
                }
        }
}
```

With the configuration above the possibility to sort by title is introduced. At the same time the sort by relevance link can be used to reset the sorting to sort by the natural solr relevance.


Sort by

Relevance
▲ [Title]

Used facets

- Content Type: pages (31)

Remove all filters

Narrow Search

Content Type

+ pages (31)

Rootline

+ Congratulations (31)
+ Content Examples (18)
+ News (4)

*

Parsed Query:
(+MatchAllDocsQuery(*:*) ())/no_coord

Searched for "*". Found 31 results in 12 milliseconds. Displayir
Results per page: 10

1 2 3 4 »

Any language, any character
Relevance: 100%
TYPO3 Speaks Your Language TYPO3 supports all langu
frontend, backend and database interactions run U

Score	Field
= 0 (Inaccurate analysis! Not all parts of the score have been taken into account.)	

Community

Fig. 6.8: Search with sorting

Templating

The rendering of the sorting is done on "Resources/Private/Partials/Results/Sorting.html" this partial is using the configuration and the view helpers to generate sorting links with the same behaviour as in ext:solr. For sure you can modifiy this template and use the ViewHelpers in the way how you want to implement your custom sorting.

6.7 Results per Page

EXT:solr allows you to configure how many result per page will be shown and at the same time the user can also change this value to an allowed value.

The following configuration can be used to configured the results per page:

```
plugin.tx_solr {
    search {
        results {
            resultsPerPage = 6
            resultsPerPageSwitchOptions = 12, 18, 24
        }
    }
}
```

When you apply the configuration above, the frontend will show 6 search results by default and show the options 12, 18 and 24 to the user to change the amount of visible results

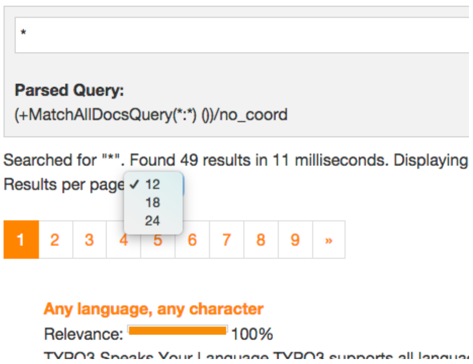

*
Parsed Query:
(+MatchAllDocsQuery(*:*) ())/no_coord
Searched for "*". Found 49 results in 11 milliseconds. Displaying
Results per page
✓ 12
18
24
1
2
3
4
5
6
7
8
9
»
Any language, any character
Relevance:
100%

Fig. 6.9: Results per page

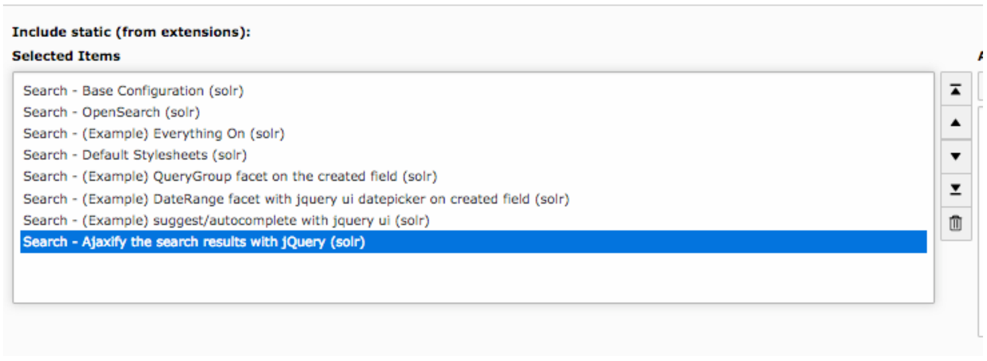
Templating

The rendering of the "perPage selector" is done on "Resources/Private/Partials/Frontend/Results/PerPage.html". This partial is build in a way that the behaviour of the perPage selector is the same as in EXT:solr. If you want to do your custom rendering for example with links instead of a for, you can customize the rendering there.

6.8 Ajaxified Results

To improve the use experience and the performance it is possible to load most of the sub requests with ajax.

To activate the **"ajaxification"** you need to include the typoscript template **"Search - Ajaxify the searchresults with jQuery"**



Include static (from extensions):
Selected Items

Search - Base Configuration (solr)
Search - OpenSearch (solr)
Search - (Example) Everything On (solr)
Search - Default Stylesheets (solr)
Search - (Example) QueryGroup facet on the created field (solr)
Search - (Example) DateRange facet with jquery ui datepicker on created field (solr)
Search - (Example) suggest/autocomplete with jquery ui (solr)
Search - Ajaxify the search results with jQuery (solr)

Fig. 6.10: Include EXT:solr ajaxify template

6.8.1 How it works?

For all links with the css class **"solr-ajaxified"** the javascript search controller triggers the request against the same search page but with the type *"7383"* which is just rendering the search request. The response is replacing everything in the container "div.tx_solr" with the content of the response.

6.8.2 Limitations

Since the example only renders the plugin on a specific page type and not the same plugin instance as on the page, settings from the flexform are not taken into account by now.

6.9 Customize

6.9.1 Use custom Fluid Templates

After these steps solrfluid is usable and using the default Templates, Layouts and Partials. If you want to overwrite them, you can change the TypoScript configuration:

```
plugin.tx_solr {
    view {
        layoutRootPaths.10 = EXT:yourpath/Layouts/
        partialRootPaths.10 = EXT:yourpath/Partials/
        templateRootPaths.10 = EXT:yourpath/Templates/
    }
}
```

Now you can copy the default partials from the extension to you project path and adapt them to your needs.

6.10 Languages

We recommend to create one solr core per language. The shipped solr example configuration provides a setup for the following languages:

- Arabic (core_ar)
- Armenian (core_hy)
- Basque (core_eu)
- Brazilian portuguese (core_ptbr)
- Bulgarian (core_bg)
- Burmese (core_my)
- Catalan (core_ca)
- Chinese (core_zh)
- Czech (core_cs)
- Danish (core_da)
- Dutch (core_nl)
- English (core_en)
- Finnish (core_fi)
- French (core_fr)
- Galician (core_gl)

- German (core_de)
- Greek (core_el)
- Hindi (core_hi)
- Hungarian (core_hu)
- Indonesian (core_id)
- Irish (core_ie)
- Italian (core_it)
- Japanese (core_ja)
- Khmer (core_km)
- Korean (core_ko)
- Lao (core_lo)
- Latvia (core_lv)
- Norwegian (core_no)
- Persian (core_fa)
- Polish (core_pl)
- Portuguese (core_pt)
- Romanian (core_ro)
- Russian (core_ru)
- Serbian (core_rs)
- Spanish (core_es)
- Swedish (core_sv)
- Thai (core_th)
- Turkish (core_tr)
- Ukrainian (core_uk)

The configuration of the connection between solr cores and sites is done in typoscript.

The following typoscript snipped shows an example how to configure multiple languages for the introduction package (EN, DE and DA):

```
plugin.tx_solr.solr {
   scheme = http
   port   = 8082
   path   = /solr/core_en/
   host   = localhost
}

[globalVar = GP:L = 1]
plugin.tx_solr.solr.path = /solr/core_de/
[end]

[globalVar = GP:L = 2]
plugin.tx_solr.solr.path = /solr/core_da/
[end]
```

After setting up the languages with typoscript you need to initialize the solr connections with the connection manager (*ConnectionManager*).

CHAPTER

SEVEN

LOGGING

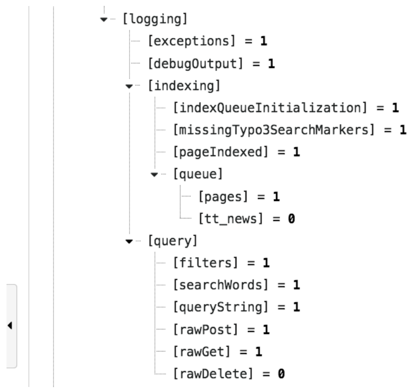
When you need to debug something in EXT:solr the extension allows you to write several information to the TYPO3 logging framework.

This can be helpful to analyze e.g.:

- What is happening during the indexing?
- What is going when a search in the frontend is triggered?

7.1 Configure Logging

There are several options that can be configured with TypoScript:



```
[logging]
    [exceptions] = 1
    [debugOutput] = 1
    [indexing]
        [indexQueueInitialization] = 1
        [missingTypo3SearchMarkers] = 1
        [pageIndexed] = 1
        [queue]
            [pages] = 1
            [tt_news] = 0
    [query]
        [filters] = 1
        [searchWords] = 1
        [queryString] = 1
        [rawPost] = 1
        [rawGet] = 1
        [rawDelete] = 0
```

Fig. 7.1: EXT:solr logging settings

You can find a detailed description of all options in the "*tx_solr.logging*" section in the reference.

Note: Make sure that you log on production only what you need, because log data can increase very quickly.

In the next steps we want to see as example how we can use the debug output to see which query was triggered in solr.

The first step is to debug the output for "rawGet":

```
plugin.tx_solr.logging.query.rawGet = 1
```

7.2 Show Logging Output

As written in the introduction, the logging framework functionally is used to write the log entries.

For a quick debugging without any extension there is the possibility to show the debug messages directly:

- For the frontend as formatted output.
- For the backend as TYPO3 console output.

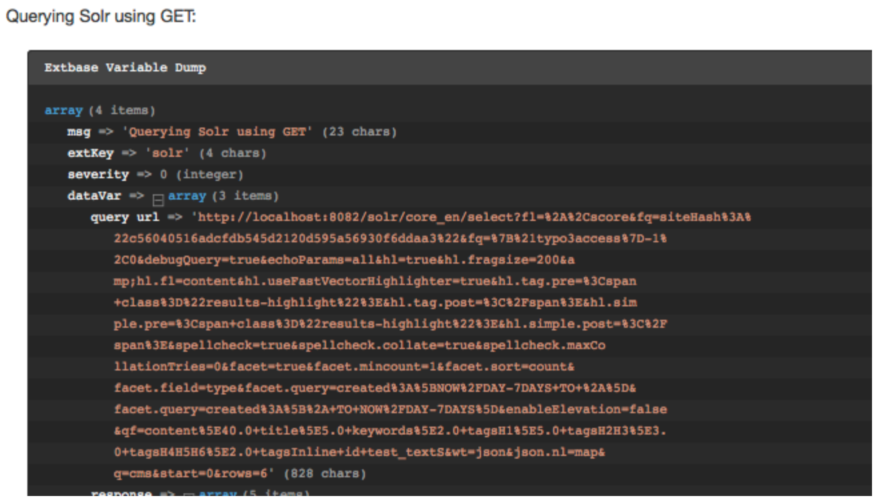
Note: You should use this on a development system only and make sure that the devIPmask does not allow other users to see the debug output.

To enable this features you need to:

- Make sure that your ip address is configured to match the devIPMask ($GLOBALS['TYPO3_CONF_VARS']['SYS']['devIPmask'])
- Enable the debug output in the EXT:solr typoscript:

```
plugin.tx_solr.logging.debugOutput = 1
```

You can now see the written log in the frontend:



Querying Solr using GET:

```
Extbase Variable Dump

array (4 items)
   msg => 'Querying Solr using GET' (23 chars)
   extKey => 'solr' (4 chars)
   severity => 0 (integer)
   dataVar => ⊟ array (3 items)
      query url => 'http://localhost:8082/solr/core_en/select?fl=%2A%2Cscore&fq=siteHash%3A%
         22c56040516adcfdb545d2120d595a56930f6ddaa3%22&fq=%7B%21typo3access%7D-1%
         2C0&debugQuery=true&echoParams=all&hl=true&hl.fragsize=200&a
         mp;hl.fl=content&hl.useFastVectorHighlighter=true&hl.tag.pre=%3Cspan
         +class%3D%22results-highlight%22%3E&hl.tag.post=%3C%2Fspan%3E&hl.sim
         ple.pre=%3Cspan+class%3D%22results-highlight%22%3E&hl.simple.post=%3C%2F
         span%3E&spellcheck=true&spellcheck.collate=true&spellcheck.maxCo
         llationTries=0&facet=true&facet.mincount=1&facet.sort=count&
         facet.field=type&facet.query=created%3A%5BNOW%2FDAY-7DAYS+TO+%2A%5D&
         facet.query=created%3A%5B%2A+TO+NOW%2FDAY-7DAYS%5D&enableElevation=false
         &qf=content%5E40.0+title%5E5.0+keywords%5E2.0+tagsH1%5E5.0+tagsH2H3%5E3.
         0+tagsH4H5H6%5E2.0+tagsInline+id+test_textS&wt=json&json.nl=map&
         q=cms&start=0&rows=6' (828 chars)
      response => ⊟ array (5 items)
```

Fig. 7.2: EXT:solr frontend debug output

CHAPTER

EIGHT

THE APACHE SOLR SERVER

The other chapters focused on the frontend or backend part of the extension. In this part we want to focus on the server part of apache solr.

8.1 Configuration Structure

The configuration can be found in the folder “Resources/Private/Solr”.

This folder contains:

- The folder “configsets”: This folder a set of configuration files that can be deployed into a solr server, as a template.

It contains the “solrconfig.xml” file, the “schema.xml” files for all languages and the accessfilter libary that belongs to this version as a jar file. This configSet needs to be in place on a solr server to create cores that are compatible to the EXT:solr extension.

- The folder “cores”: This folder ships an example “core.properties” file for all languages that are compatible with EXT:solr.

A “core.properties” file references a “configSet” that should be used. The path to the schema that is bound to a core is configured as “schema” relative to the root folder of the “configSet”.

By example a “core.properties” file looks like this:

```
configSet=ext_solr_6_0_0
schema=german/schema.xml
name=core_de
dataDir=../../data/german
```

- The solr.xml file: This file configures solr as required for the used Apache Solr version.

8.1.1 The EXT:solr configSet

As mentioned before the configSet is one single package, that contains all to customize a plain Apache Solr Server, to an Apache Solr Server, ready for EXT:solr & TYPO3.

The configset of the current release is located in “Resources/Private/Solr/configsets/ext_solr_<release>”.

Inside the configSet you find the following folders:

- conf: This folder contains the solr configuration and all schemata files. There is one directory for each language which ships the schema.xml file for this language.

The schema.xml file contains the language specific adaptions and includes all general schema fields and types with an XInclude statement.

```
<!-- xinclude fields -->
<xi:include href="../general_schema_types.xml" xmlns:xi="http://www.w3.org/2001/
↪XInclude"/>

<!--  xinclude fields-->
<xi:include href="../general_schema_fields.xml" xmlns:xi="http://www.w3.org/2001/
↪XInclude"/>
```

- typo3lib: This folder ships the compiled access filter jar file, that should be used with this EXT:solr version.

The solrconfig (conf/solrconfig.xml) is configured to load all jar files from typo3lib.

8.2 Setup steps

With the extension we ship an installer for development and a docker images that can be used to install solr.

When you want to install solr on your system in another way the following steps are required.

- Install the solr server
- Copy the configsets into the configset folder (by default $SOLR_HOME/server/solr/configsets)
- Make sure that the solr.xml file ($SOLR_HOME/server/solr/solr.xml) is in place an fits to your solr version
- Create an init script that start solr on boottime.
- Secure your solr port from outside.
- Make sure that solr is running with an own user.
- Backup your data folders

Hint: Apache Solr ships an install script in newer version that might cover your requirements for production ($SOLR_HOME/bin/install_solr_service.sh). We don't use it in EXT:solr because there are currently problems when using it with ubuntu xenial (16.04)

8.3 Index Maintenance

Solr offers a lot of request handlers to do maintenance tasks.

8.3.1 Committing pending documents

```
curl http://host:port/solr-path/update -H "Content-Type: text/xml"
    --data-binary '<commit />'
```

8.3.2 Clearing the index

```
curl http://host:port/solr-path/update -H "Content-Type: text/xml"
    --data-binary '<delete><query>*:*</query></delete>'

curl http://host:port/solr-path/update -H "Content-Type: text/xml"
    --data-binary '<commit />'
```

8.3.3 Optimizing the index

You should do this every once in a while, f.e. every day. For TYPO3, there is already a scheduler task available for this.

```
curl http://host:port/solr-path/update -H "Content-Type: text/xml"
    --data-binary '<optimize />'
```

8.3.4 Searching the index from the command line

Parameters:

- **q** what to search for. Format: fieldName:fieldValue
- **qt** defines the query type, for the command line we recommend "standard", the extension itself uses "dismax"
- **fl** comma separated list of fields to return
- **rows** number of rows to return
- **start** offset from where to return results

```
curl 'http://host:port/path-to-solr/select?q=hello&qt=standard&fl=title,content'
```

8.3.5 Getting information / statistics about the index

```
curl 'http://host:port/path-to-solr/admin/luke'
```

8.3.6 Create cores with the core admin api

The CoreAdmin API (https://cwiki.apache.org/confluence/display/solr/CoreAdmin+API) allows you, manipulate the cores in your solr server.

Since we support configSets a core could be generated with the following http call:

```
curl 'http://host:port/path-to-solr/admin/cores?action=CREATE&name=core_de&
↪configSet=ext_solr_8_0_0&schema=german/schema.xml&dataDir=../../data/german'
```

8.4 SolrConfig Parameters

There are several parameters in the solrconfig.xml that can be used to tune your solr server. Our solrconfig.xml is designed to ship a reasonable configuration for the most standard use cases.

For use cases with very large indexes or high performance requirements it makes sence to tune those parameters

8.4.1 indexConfig.useCompoundFile

This value is "true" by default in our configuration. By setting this value to true solr only writes one file for indexes instead of many. This is a little bit slower but more robust to prevent errors with "Too many open files".

CHAPTER

NINE

CONFIGURATION REFERENCE

9.1 tx_solr

This section defines general configuration options.

9.1.1 enabled

Type Boolean

TS Path plugin.tx_solr.enabled

Default 1

Options 0, 1

Since 1.2

A switch to completely turn on / off EXT:solr. Comes in handy with multi site installations where you want to enable EXT:solr only for certain sites, but still have the extension's configuration at a single place and include that for each site. Just set enabled = 0 for each site's root TS template or use conditions where you do not want EXT:solr.

Important: This also influences the connection manager; connections will be registered / detected only for enabled = 1.

9.1.2 enableDebugMode

Type Boolean

TS Path plugin.tx_solr.enableDebugMode

Default 0

Options 0, 1

Since 1.0

See http://wiki.apache.org/solr/CommonQueryParameters#debugQuery

If enabled, the debugQuery query parameter is added to the Solr queries. Solr will then return additional information explaining the the query, scoring, timing, and other information.

9.2 tx_solr.general

This section defines general settings.

- *dateFormat.date*

9.2.1 dateFormat.date

Type String

TS Path plugin.tx_solr.general.dateFormat.date

Default d.m.Y H:i

Since 1.0

See http://www.php.net/manual/de/function.strftime.php

Defines the format that is used for dates throughout the extension like in view helpers for example. The format uses the *strftime()* php function syntax, please consult the php documentation for available options.

9.3 tx_solr.view

All view related settings, these settings might also be relevant for Fluid.

9.3.1 pluginNamespace

Type String

TS Path plugin.tx_solr.view.pluginNamespace

Since 7.0

Default tx_solr

Plugin namespace. Can be used to change the plugin namespace and can be changed by instance in the flexform.

9.3.2 templateFiles

By convention the templates is loaded from EXT:solr/Resources/Private/Templates/Frontend/Search/(ActionName).html. If you want to define a different entry template, you can do this here to overwrite the conventional default template. If you want to use FLUID fallbacks you can just configure the template name, otherwise you could also use a full reference EXT:/.../.

The templates that you configure in availableTemplate can be used in the flexform by the editor to select a template for the concrete plugin instance.

9.3.3 templateFiles.results

Type String

TS Path plugin.tx_solr.view.templateFiles.results

Since 7.0 (Replaces previous setting plugin.tx_solr.templateFiles.result)

Default Results

By convention the "Results" template from you configured FLUID template path will be used As alternative you can configure a different template name here (e.g. MyResults or a full path to an entry template here).

9.3.4 templateFiles.results.availableTemplates

Type Array

TS Path plugin.tx_solr.view.templateFiles.results.availableTemplates

Since 7.0

Default none

Allows to configure templates that are available in the flexform to switch.

Example:

```
plugin.tx_solr.view.templateFiles.results.availableTemplates {
    default {
        label = Default Searchresults Template
        file = Results
    }
    finder {
        label = Productfinder Template
        file = ProductFinder
    }
}
```

9.3.5 templateFiles.form

Type String

TS Path plugin.tx_solr.view.templateFiles.form

Since 7.0 (Replaces previous setting plugin.tx_solr.templateFiles.form)

Default Form

By convention the "Form" template from you configured FLUID template path will be used . As alternative you can configure a different template name here (e.g. MyForm or a full path to an entry template here).

9.3.6 templateFiles.form.availableTemplates

Type Array

TS Path plugin.tx_solr.view.templateFiles.form.availableTemplates

Since 7.0

Default none

Allows to configure templates that are available in the flexform to switch.

Example:

```
plugin.tx_solr.view.templateFiles.form.availableTemplates {
    default {
        label = Default Searchform Template
        file = Form
    }
    specialform {
        label = Extended Search Form
        file = BetterForm
    }
}
```

9.3.7 templateFiles.frequentSearched

Type String

TS Path plugin.tx_solr.view.templateFiles.frequentSearched

Since 7.0 (Replaces previous setting plugin.tx_solr.templateFiles.frequentSearched)

Default FrequentlySearched

By convention the "FrequentlySearched" template from you configured FLUID template path will be used . As alternative you can configure a different template name here (e.g. FrequentlySearched or a full path to an entry template here).

9.4 tx_solr.solr

This section defines the address of the Solr server. As the communication with the Solr server happens over HTTP this is just a simple URL. Each of the URL's components can be defined separately.

9.4.1 scheme

Type String

TS Path plugin.tx_solr.solr.scheme

Default http

Options http, https

cObject supported yes

Since 1.2 2.0

Allows to set the connection scheme to "https" instead of the default "http".

9.4.2 host

Type String

TS Path plugin.tx_solr.solr.host

Default localhost

cObject supported yes

Since 1.0

Sets the host portion of the URL.

9.4.3 port

Type Integer

TS Path plugin.tx_solr.solr.port

Default 8983

cObject supported yes

Since 1.0

Sets the port portion of the URL.

9.4.4 path

Type String

TS Path plugin.tx_solr.solr.path

Default /

cObject supported yes

Since 1.0

Sets the path portion of the URL. Make sure to have the path end with a slash (/).

9.4.5 username

Type String

TS Path plugin.tx_solr.solr.username

Since 6.0

cObject supported yes

Sets the username required to access the solr server.

9.4.6 password

Type String

TS Path plugin.tx_solr.solr.password

Since 6.0

cObject supported yes

Sets the password required to access the solr server.

9.4.7 timeout

Type Float

TS Path plugin.tx_solr.solr.timeout

Default 0.0

Since 1.0

cObject supported no

Can be used to configure a connection timeout.

9.5 tx_solr.index

This part contains all configuration that is required to setup your indexing configuration. You can use EXT:solr to easily index pages or any kind of records of your TYPO3 CMS.

Allows to prevent frontend indexing of pages when a backend editor is logged in and browsing the website.

9.5.1 additionalFields (deprecated)

Type String, cObject (since 1.1)

TS Path plugin.tx_solr.index.additionalFields

Since 1.0

Deprecated 2.0

A mapping of Solr field names to additional string values to be indexed with page documents. Use dynamic fields to index additional data, this way you don't have to modify the schema.xml

Example:

```
plugin.tx_solr.index.additionalFields {
  myFirstAdditionalField_stringS = some string

  mySecondAdditionalField_stringS = TEXT
  mySecondAdditionalField_stringS {
    value = some other value that can be constructed using any TypoScript cObject
    case = upper
    // more processing here as needed
  }
}
```

Since version 1.1 you can use cObjects to generate the value for the field. The only thing to observe is that you generate strings. Other values may work, but haven't been tested yet.

Deprecated since 2.0, please use the Index Queue indexing configurations instead as it allows you to define more precisely for which types of documents you want which fields to be indexed.

9.5.2 fieldProcessingInstructions

Type cObject

TS Path plugin.tx_solr.index.fieldProcessingInstructions

Since 1.2 2.0

Options timestampToIsoDate, uppercase, pathToHierarchy (2.5-dkd), pageUidToHierarchy (2.5-dkd)

Assigns processing instructions to Solr fields during indexing (Syntax: Solr index field = processing instruction name). Currently it is not possible to extend / add own processing instructions. Before documents are sent to the Solr server they are processed by the field processor service. Currently you can make a filed's value all uppercase, convert a UNIX timestamp to an ISO date, or transform a path into a hierarchy for hierarchical facets (2.0 only). Currently you can use only one processing instruction at a time.

Example:

```
fieldProcessingInstructions {
    changed = timestampToIsoDate
    created = timestampToIsoDate
    endtime = timestampToIsoDate
}
```

9.5.3 queue

The Index Queue is a powerful feature introduced with version 2.0. It allows you to easily index any table in your TYPO3 installation by defining a mapping of SolrFieldName = DatabaseTableFieldNameOrContentObject. The table must be configured / described in TCA, though. To index other, external data sources you might want to check out Solr's Data Import Handler (DIH).

The Index Queue comes preconfigured to index pages (enabled by default) and an example configuration for tt_news (provided as a separate TypoScript template).

Type Array

TS Path plugin.tx_solr.index.queue

Since 2.0

Default pages

Defines a set of table indexing configurations. By convention the name of the indexing configuration also represents the table name. You can name the indexing configuration differently though by explicitly defining the table as a parameter within the indexing configuration. That's useful when indexing records from one table with different configuration - different single view pages / URLs for example.

Example:

```
// enables indexing of tt_news records
plugin.tx_solr.index.queue.news = 1
plugin.tx_solr.index.queue.news.fields {
    abstract = short
    author = author
    description = short
    title = title

    // the special SOLR_CONTENT content object cleans HTML and RTE fields
    content = SOLR_CONTENT
    content {
        field = bodytext
    }

    // the special SOLR_RELATION content object resolves relations
    category_stringM = SOLR_RELATION
    category_stringM {
        localField = category
        multiValue = 1
    }

    // the special SOLR_MULTIVALUE content object allows to index multivalue fields
    keywords = SOLR_MULTIVALUE
    keywords {
        field = keywords
    }

    // build the URL through typolink, make sure to use returnLast = url
    url = TEXT
    url {
        typolink.parameter = {$plugin.tt_news.singlePid}
        typolink.additionalParams = &tx_ttnews[tt_news]={field:uid}
        typolink.additionalParams.insertData = 1
        typolink.returnLast = url
        typolink.useCacheHash = 1
    }

    sortAuthor_stringS = author
    sortTitle_stringS  = title
}
```

9.5.4 queue.[indexConfig]

Type Boolean, Array

TS Path plugin.tx_solr.index.queue.[indexConfig]

Since 2.0

Default pages

An indexing configuration defines several parameters about how to index records of a table. By default the name of the indexing configuration is also the name of the table to index.

By setting *plugin.tx_solr.index.queue.[indexConfig] = 1 or 0* you can en- / disable an indexing configuration.

Note: you could add *L={field:__solr_index_language}* in the additionalParams of the typolink to link to the correct language version (this was removed from the example above to simplify the example)

9.5.5 queue.[indexConfig].additionalWhereClause

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].additionalWhereClause

Since 2.0

A WHERE clause that is used when initializing the Index Queue, limiting what goes into the Queue. Use this to limit records by page ID or the like.

```
// only index standard and mount pages, enabled for search
plugin.tx_solr.index.queue.pages.additionalWhereClause = doktype IN(1, 7)
```

9.5.6 queue.[indexConfig].initialPagesAdditionalWhereClause

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].initialPagesAdditionalWhereClause

Since 6.1

A WHERE clause that is used when initializing the Index Queue, limiting pages that goes into the Queue. This filter is applied **prior** to the plugin.tx_solr.index.queue.[indexConfig].additionalWhereClause filter and hence provides an even stronger filter mechanism - since it can be used to filter away page ID's that shouldn't be processed at all.

```
// Filter away pages that are "spacer" and have no_search, hidden and nav_hide set␣
↪to zero
plugin.tx_solr.index.queue.pages.initialPagesAdditionalWhereClause = doktype <>␣
↪199 AND no_search = 0 AND hidden = 0 AND nav_hide = 0
```

9.5.7 queue.[indexConfig].additionalPageIds

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].additionalPageIds

Since 2.0

Defines additional pages to take into account when indexing records for example. Especially useful for indexing DAM records or if you have your news outside your site root in a shared folder to use for multiple sites.

Additional page IDs can be provided as comma-separated list.

9.5.8 queue.[indexConfig].table

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].table

Since 2.0

Sometimes you may want to index records from a table with different configurations, f.e., to generate different single view URLs for tt_news records depending on their category or storage page ID. In these cases you can use a distinct name for the configuration and define the table explicitly.

```
plugin.tx_solr.index.queue.generalNews {
  table = tt_news
  fields.url = URL for the general news
  // more field configurations here ...
}

// extends the general news configuration
plugin.tx_solr.index.queue.pressNews < plugin.tx_solr.index.queue.generalNews
plugin.tx_solr.index.queue.pressNews {
  fields.url = overwriting URL for the press announcements
  // may overwrite or unset more settings from the general configuration
}

// completely different configuration
plugin.tx_solr.index.queue.productNews {
  table = tt_news
  fields.url = URL for the product news
}
```

9.5.9 queue.[indexConfig].initialization

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].initialization

Since 2.0

When initializing the Index Queue through the search backend module the queue tries to determine what records need to be indexed. Usually the default initializer will be enough for this purpose, but this option allows to define a class that will be used to initialize and add records to the Index Queue in special ways.

The extension uses this option for initializing the pages and more specifically to resolve Mount Page trees so they can be indexed too, although only being virtual pages.

9.5.10 queue.[indexConfig].indexer

Type String, Array

TS Path plugin.tx_solr.index.queue.[indexConfig].indexer

Since 2.0

When configuring tables to index a default indexer is used that comes with the extensions. The default indexer resolves the Solr field to database table field mapping as configured. However, in some cases you may reach the limits of TypoScript, when this happens you can configure a specialized indexer using this setting.

The indexer class is loaded using TYPO3's auto loading mechanism, so make sure your class is registered properly. The indexer must extend tx_solr_indexqueue_Indexer.

Example, pages use a specialized indexer:

```
plugin.tx_solr.index.queue.pages {
    indexer = tx_solr_indexqueue_PageIndexer
    indexer {
        // add options for the indexer here
    }
}
```

Within the indexer configuration you can also define options for the specialized indexer. These are then available within the indexer class in $this->options.

Example, the TypoScript settings are available in PHP:

TypoScript:

```
plugin.tx_solr.index.queue.myIndexingConfiguration {
    indexer = tx_myextension_indexqueue_MyIndexer
    indexer {
        someOption = x
        someOtherOption = y
    }
}
```

PHP:

```
namespace MyVendor\Namespace;

use ApacheSolrForTypo3\Solr\IndexQueue\Indexer;

class MyIndexer extends Indexer {
  public function index(tx_solr_indexqueue_Item $item) {
    if ($this->options['someOption']) {
      // ...
    }
  }
}
```

9.5.11 queue.[indexConfig].indexingPriority

Type Integer

TS Path plugin.tx_solr.index.queue.[indexConfig].indexingPriority

Since 2.2

Default 0

Allows to define the order in which Index Queue items of different kinds are indexed. Items with higher priority are indexed first.

9.5.12 queue.[indexConfig].fields

Type Array

TS Path plugin.tx_solr.index.queue.[indexConfig].fields

Since 2.0

Mapping of Solr field names on the left side to database table field names or content objects on the right side. You must at least provide the title, content, and url fields. TYPO3 system fields like uid, pid, crdate, tstamp and so on are added automatically by the indexer depending on the TCA information of a table.

Example:

```
plugin.tx_solr.index.queue.[indexConfig].fields {
  content = bodytext
  title = title
  url = TEXT
  url {
    typolink.parameter = {$plugin.tx_extensionkey.singlePid}
    typolink.additionalParams = &tx_extenionkey[record]={field:uid}
    typolink.additionalParams.insertData = 1
    typolink.returnLast = url
  }
}
```

9.5.13 queue.[indexConfig].attachments.fields

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].attachments.fields

Since 2.5-dkd

Comma-separated list of fields that hold files. Using this setting allows to tell the file indexer in which fields to look for files to index from records.

Example:

```
plugin.tx_solr.index.queue.tt_news.attachments.fields = news_files
```

9.5.14 queue.[indexConfig].recursiveUpdateFields

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].recursiveUpdateFields

Since 6.1

Default Empty

Allows to define a list of additional fields from the pages table that will trigger a recursive update.

```
plugin.tx_solr.index.queue.pages.recursiveUpdateFields = title
```

The example above will trigger a recursive update of pages if the title is changed.

Please note that the following columns should NOT be configured as recursive update fields: “hidden” and “extendToSubpages”. These fields are handled by EXT:solr already internally and thus they will have not effect.

9.5.15 queue.pages.excludeContentByClass

Type String

TS Path plugin.tx_solr.index.queue.pages.excludeContentByClass

Since 4.0

Can be used for page indexing to exclude a certain css class to be indexed.

Example:

```
plugin.tx_solr.index.queue.pages.excludeContentByClass = removeme
```

The example above will remove the content of items in the page that have the css class “removeme”.

9.5.16 queue.pages.allowedPageTypes

Type List of Integers

TS Path plugin.tx_solr.index.queue.pages.allowedPageTypes

Since 3.0

Default 1,7

Allows to set the pages types allowed to be indexed.

Even if you have multiple queues for pages, e.g. via different `additionalWhereClause`‘s, you have to set this value to allow further `doktype`‘s. Restrict the pages to be indexed by each queue via `additionalWhereClause`.

9.5.17 queue.pages.indexer.authorization.username

Type String

TS Path plugin.tx_solr.index.queue.pages.indexer.authorization.username

Since 2.0

Specifies the username to use when indexing pages protected by htaccess.

9.5.18 queue.pages.indexer.authorization.password

Type String

TS Path plugin.tx_solr.index.queue.pages.indexer.authorization.password

Since 2.0

Specifies the password to use when indexing pages protected by htaccess.

9.5.19 queue.pages.indexer.frontendDataHelper.scheme

Type String

TS Path plugin.tx_solr.index.queue.pages.indexer.frontendDataHelper.scheme

Since 2.0

Specifies the scheme to use when indexing pages.

9.5.20 queue.pages.indexer.frontendDataHelper.host

Type String

TS Path plugin.tx_solr.index.queue.pages.indexer.frontendDataHelper.host

Since 2.0

Specifies the host to use when indexing pages.

9.5.21 queue.pages.indexer.frontendDataHelper.path

Type String

TS Path plugin.tx_solr.index.queue.pages.indexer.frontendDataHelper.path

Since 2.0

Specifies the path to use when indexing pages.

9.5.22 Indexing Helpers

To make life even easier the Index Queue provides some indexing helpers. These helpers are content objects that perform cleanup tasks or content transformations.

SOLR_CONTENT

Since 2.0

Cleans a database field in a way so that it can be used to fill a Solr document's content field. It removes HTML markup, Javascript and invalid utf-8 chracters.

The helper supports stdWrap on its configuration root.

Example:

```
content = SOLR_CONTENT
content {
    field = bodytext
}
```

Parameters:

value

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].value

Since 2.0

Defines the content to clean up. In this case the value would be hard-coded.

SOLR_MULTIVALUE

Since 2.0

Turns comma separated strings into an array to be used in a multi value field of an Solr document.

The helper supports stdWrap on its configuration root.

Example:

```
keywords = SOLR_MULTIVALUE
keywords {
    field = tags
    separator = ,
    removeEmptyValues = 1
}
```

Parameters:

value

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].value

Since 2.0

Defines the content to clean up. In this case the value would be hard-coded.

separator

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].separator

Since 2.0

Default ,

The separator by which to split the content.

removeEmptyValues

Type Boolean

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].removeEmptyValues

Since 2.0

Options 0,1

Default 1

The helper will clean the resulting array from empty values by default. If, for some reason, you want to keep empty values just set this to 0.

removeDuplicateValues

Type Boolean

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].removeDuplicateValues

Since 2.9

Options 0,1

Default 0

Cleans the result from duplicate values.

SOLR_RELATION

Since 2.0

Resolves relations between tables.

Example:

```
category_stringM = SOLR_RELATION
category_stringM {
    localField = category
    multiValue = 1
}
```

Parameters:

localField

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].localField

Since 2.0

Required yes

The current record's field name to use to resolve the relation to the foreign table.

foreignLabelField

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].foreignLabelField

Since 2.0

Usually the label field to retrieve from the related records is determined automatically using TCA, using this option the desired field can be specified explicitly. To specify the label field for recursive relations, the field names can be separated by a dot, e.g. for a category hierarchy to get the name of the parent category one could use "parent.name" (since version:2.9).

multiValue

Type Boolean

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].multiValue

Since 2.0

Options 0,1

Default 0

Whether to return related records suitable for a multi value field. If this is disabled the related values will be concatenated using the following singleValueGlue.

singleValueGlue

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].singleValueGlue

Since 2.0

Default ”, “

When not using multiValue, the related records need to be concatenated using a glue string, by default this is ”, ” (comma followed by space). Using this option a custom glue can be specified. The custom value must be wrapped by pipe (|) characters to be able to have leading or trailing spaces.

relationTableSortingField

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].relationTableSortingField

Since 2.2

Field in an mm relation table to sort by, usually “sorting”.

enableRecursiveValueResolution

Type Boolean

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].enableRecursiveValueResolution

Since 2.9

Options 0,1

Default 1

If the specified remote table’s label field is a relation to another table, the value will be resolve by following the relation recursively.

removeEmptyValues

Type Boolean

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].removeEmptyValues

Since 2.9

Options 0,1

Default 1

Removes empty values when resolving relations.

removeDuplicateValues

Type Boolean

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].removeDuplicateValues

Since 2.9

Options 0,1

Default 0

Removes duplicate values

additionalWhereClause

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].additionalWhereClause

Since 5.0

Where clause that could be used to limit the related items to a subset that matches this where clause

Example:

```
category_stringM = SOLR_RELATION
category_stringM {
    localField = tags
    multiValue = 1
    additionalWhereClause = pid=2
}
```

SOLR_CLASSIFICATION

Since 8.0

Allows to classify documents based on a configured pattern

Example:

```
topic_stringM = SOLR_CLASSIFICATION
topic_stringM {
    field = __solr_content
    classes {
        programming {
            patterns = php, java, javascript, go
            class = programming
        }
        cms {
            patterns = TYPO3, joomla
            class = cms
        }
        database {
            patterns = mysql, MariaDB, postgreSQL
            class = database
        }
    }
}
```

field

Type String

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].field

Since 8.0

Name of the database field, that should be used to as content to classify. The special field __solr_content can be used during indexing to classify the content of the page or file or any other record that fills the content field before.

classes

Type Array

TS Path plugin.tx_solr.index.queue.[indexConfig].fields.[fieldName].field

Since 8.0

Array of classification configurations. Each configuration needs to have the property “patterns”, that is a list of patters that need to match and “class”, that is the mapped class that will be indexed then.

Note:

The output field needs to be a multivalue field since an indexed item can have multiple classes.

9.5.23 enableCommits

Type Boolean

TS Path plugin.tx_solr.index.enableCommits

Since 6.1

Default true

This setting controls whether ext-solr will implicitly cause solr commits as part of its operation.

If this settings is set to false, you need to ensure that something else will periodically call commits. The solr daemons AutoCommit feature would be a natural choice.

This feature is mainly useful, when you have many installations in the same solr core.

Note: Calling some APIs may still cause commits, but these can always be explicitly disabled.

9.6 tx_solr.search

The search section, you probably already guessed it, provides configuration options for the all things related to actually searching the index, setting query parameters, formatting and processing result documents and the result listing.

9.6.1 targetPage

Type Integer

TS Path plugin.tx_solr.search.targetPage

Default 0

Since 1.0

Sets the target page ID for links. If it is empty or 0, the current page ID will be used.

Note: This setting can be overwritten by the plugins flexform.

9.6.2 trustedFields

Type String

TS Path plugin.tx_solr.search.trustedFields

Default url

Since 3.1

The data in EXT:solr is escaped right after the retrieval from Solr. In rare cases when you need to store HTML in Solr documents you can use this configuration to mark these fields as trusted fields and skip the escaping. Typically this is needed when you want to retrieve html from solr.

The following example shows how to avoid html in the content field:

```
plugin.tx_solr.search.trustedFields = url, content
```

9.6.3 initializeWithEmptyQuery

Type Boolean

TS Path plugin.tx_solr.search.initializeWithEmptyQuery

Default 0

Options 0,1

Since 1.0

If enabled, the results plugin (pi_results) issues a “get everything” query during initialization. This is useful, if you want to create a page that shows all available facets although no search has been issued by the user yet. Note: Enabling this option alone will not show results of the get everything query. To also show the results of the query, see option showResultsOfInitialEmptyQuery below.

9.6.4 showResultsOfInitialEmptyQuery

Type Boolean

TS Path plugin.tx_solr.search.showResultsOfInitialEmptyQuery

Default 0

Options 0,1

Since 1.0

Requires initializeWithEmptyQuery (above) to be enabled to have any effect. If enabled together with initialize-WithEmptyQuery the results of the initial "get everything" query are shown. This way, in combination with a filter you can easily list a predefined set of results.

9.6.5 keepExistingParametersForNewSearches

Type Boolean

TS Path plugin.tx_solr.search.keepExistingParametersForNewSearches

Default 0

Options 0,1

Since 2.0

When doing a new search, existing parameters like filters will be carried over to the new search. This is useful for a scenario where you want to list all available documents first, then allow the user to filter the documents using facets and finally allow him to specify a search term to refine the search.

9.6.6 ignoreGlobalQParameter

Type Boolean

TS Path plugin.tx_solr.search.ignoreGlobalQParameter

Default 0

Options 0,1

Since 7.0

In some cases you want EXT:solr to react on the parameter "q" in the url. Normally plugins are bounded to a namespace to allow multiple instances of the search on the same page. In this case you might want to disable this and let EXT:solr only react on the namespaced query parameter (tx_solr[q] by default).

9.6.7 query

The query sub-section defines a few query parameters for the query that will be sent to the Solr server later on. Some query parameters are also generated and set by the extension itself, f.e. when using facets.

query.allowEmptyQuery

Type Boolean

TS Path plugin.tx_solr.search.query.allowEmptyQuery

Default 0

Options 0,1

Since 1.4

If enabled, empty queries are allowed.

query.allowedSites

Type String

TS Path plugin.tx_solr.search.query.allowedSites

Since 2.2

Default __solr_current_site

When indexing documents (pages, records, files, ...) into the Solr index, the solr extension adds a "siteHash". The siteHash is used to allow indexing multiple sites into one index and still have each site only find its own documents. This is achieved by adding a filter on the siteHash.

Sometimes though, you want to search across multiple domains, then the siteHash is a blocker. Using the allowedSites setting you can set a comma-separated list of domains who's documents are allowed to be included in the current domain's search results. The default value is **__solr_current_site** which is a magic string/variable that is replaced with the current site's domain when querying the Solr server.

Since 3.0

Version 3.0 introduced a couple more magic keywords that get replaced:

- **__current_site** same as **__solr_current_site**
- **__all** Adds all domains as allowed sites
- * (asterisk character) Everything is allowed as siteHash (same as no siteHash check). This option should only be used when you need a search across multiple system and you know the impact of turning of the siteHash check.

query.getParameter

Type String

TS Path plugin.tx_solr.search.query.getParameter

Since 2.2

Default tx_solr|q

The GET query parameter name used in URLs. Useful for cases f.e. when a website tracking tool does not support the default array GET parameters.

The option expects a string, you can also define an array in the form of arrayName|arrayKey.

Example:

```
plugin.tx_solr.search.query.getParameter = q
```

query.queryFields (query.fields)

Type String

TS Path plugin.tx_solr.search.query.queryFields

Since 1.0

Default content^40.0, title^5.0, keywords^2.0, tagsH1^5.0, tagsH2H3^3.0, tagsH4H5H6^2.0, tagsInline^1.0, description^4.0, abstract^1.0, subtitle^1.0, navtitle^1.0, author^1.0

Note query.fields has been renamed to query.queryFields in version 3.0

Defines what fields to search in the index. Fields are defined as a comma separated list. Each field can be given a boost by appending the boost value separated by the ^ character, that's Lucene query language. The boost value itself is a float value, pay attention to using a dot as the separator for the fractions. Use this option to add more fields to search.

The boost take influence on what score a document gets when searching and thus how documents are ranked and listed in the search results. A higher score will move documents up in the result listing. The boost is a multiplier for the original score value of a document for a search term.

By default if a search term is found in the content field the documents gets scored / ranked higher as if a term was found in the title or keywords field. Although the default should provide a good setting, you can play around with the boost values to find the best ranking for your content.

query.returnFields

Type String

TS Path plugin.tx_solr.search.query.returnFields

Since 3.0

Default *, score

Limits the fields returned in the result documents, by default returns all field plus the virtual score field.

query.minimumMatch

Type String

TS Path plugin.tx_solr.search.query.minimumMatch

Since 1.2, 2.0

Default (empty)

See Apache Solr Wiki mm / Minimum Should Match

Sets the minimum match mm query parameter. By default the mm query parameter is set in solrconfig.xml as 2<-35%. This means that for queries with less than three words they all must match the searched fields of a document. For queries with three or more words at least 65% of them must match rounded up.

Please consult the link to the Solr wiki for a more detailed description of the mm syntax.

query.boostFunction

Type String

TS Path plugin.tx_solr.search.query.boostFunction

Since 1.2, 2.0

Default (empty)

See Apache Solr Wiki / TheDisMaxQueryParser BoostFunction

See Apache Solr Wiki / Function Queries

Example recip(ms(NOW,created),3.16e-11,1,1)

A boost function can be useful to influence the relevance calculation and boost some documents to appear more at the beginning of the result list. Technically the parameter will be mapped to the **"bf"** parameter in the solr query.

Use cases for example could be:

"Give newer documents a higher priority":

This could be done with a recip function:

```
recip(ms(NOW,created),3.16e-11,1,1)
```

"Give documents with a certain field value a higher priority":

This could be done with:

```
termfreq(type,'tx_solr_file')
```

query.boostQuery

Type String

TS Path plugin.tx_solr.search.query.boostQuery

Since 2.0

Default (empty)

See Apache Solr Wiki / TheDisMaxQueryParser BoostQuery

Sets the boost function **bq** query parameter.

Allows to further manipulate the score of a document by using Lucene syntax queries. A common use case for boost queries is to rank documents of a specific type higher than others.

Please consult the link to the Solr wiki for a more detailed description of boost functions.

Example (boosts tt_news documents by factor 10):

```
plugin.tx_solr.search.query.boostQuery = (type:tt_news)^10
```

query.tieParameter

Type String

TS Path plugin.tx_solr.search.query.tieParameter

Since 8.0

See *Lucene Documentation / TheDisMaxQueryParser TieParameter <http://lucene.apache.org/solr/guide/7_0/the-dismax-query-parser.html#the-tie-tie-breaker-parameter>*

This parameter ties the scores together. Setting is to "0" (default) uses the maximum score of all computed scores. A value of "1.0" adds all scores. The value is a number between "0.0" and "1.0".

query.filter

Type Array

TS Path plugin.tx_solr.search.query.filter

Since 1.0

See Lucene Documentation / Query Parser Syntax

Allows to predefine filters to apply to a search query. You can add multiple filters through a name to Lucene filter mapping. The filters support stdWrap.

Example:

```
plugin.tx_solr.search.query.filter {
    pagesOnly = type:pages
    johnsPages = author:John
    badKeywords = {foo}
    badKeywords.wrap = -keywords:|
    badKeywords.data = GP:q
}
```

Note: When you want to filter for something with whitespaces you might need to quote the filter term.

```
plugin.tx_solr.search.query.filter {
    johnsDoesPages = author:"John Doe"
}
```

query.filter.__pageSections

Type comma-separated list of page IDs

TS Path plugin.tx_solr.search.query.filter.__pageSections

Since 3.0

This is a magic/reserved filter (thus the double underscore). It limits the query and the results to certain branches/-sections of the page tree. Multiple starting points can be provided as a comma-separated list of page IDs.

query.sortBy

Type String

TS Path plugin.tx_solr.search.query.sortBy

Since 1.0

Allows to set a custom sorting for the query. By default Solr will sort by relevance, using this setting you can sort by any sortable field.

Needs a Solr field name followed by asc for ascending order or desc for descending.

Example:

```
plugin.tx_solr.search.query.sortBy = title asc
```

query.phrase

Type Boolean

TS Path plugin.tx_solr.search.query.phrase

Since 8.0

Default 0

See "pf", "ps", "qs" https://lucene.apache.org/solr/guide/6_6/the-dismax-query-parser.html#TheDisMaxQueryParser-Thepf_PhraseFields_Parameter

This parameter enables the phrase search feature from Apache Solr. Setting is to "0" (default) does not change behaviour from Apache Solr if user searches for two and more words. Enabling phrase search feature influences the document set and/or the scores of documents.

query.phrase.fields

Type String

TS Path plugin.tx_solr.search.query.phrase.fields

Since 8.0

Default content^10.0, title^10.0, tagsH1^10.0, tagsH2H3^10.0, tagsH4H5H6^10.0, tagsInline^10.0, description^10.0, abstract^10.0, subtitle^10.0, navtitle^10.0

See "pf" parameter https://lucene.apache.org/solr/guide/6_6/the-dismax-query-parser.html#TheDisMaxQueryParser-Thepf_PhraseFields_Parameter

This parameter defines what fields should be used to search in the given phrase. Matched documents will be boosted according to fields boost value. Fields are defined as a comma separated list and same way as queryFields.

Note: The value of this setting has NO influence on explicit phrase search.

query.phrase.slop

Type Integer

TS Path plugin.tx_solr.search.query.phrase.slop

Since 8.0

Default 0

See "ps" parameter https://lucene.apache.org/solr/guide/6_6/the-dismax-query-parser.html#TheDisMaxQueryParser-Theps_PhraseSlop_Parameter

This parameter defines the "phrase slop" value, which represents the number of positions one word needs to be moved in relation to another word in order to match a phrase specified in a query.

Note: The value of this setting has NO influence on explicit phrase search.

query.phrase.querySlop

Type Integer

TS Path plugin.tx_solr.search.query.phrase.querySlop

Since 8.0

Default 0

See "qs" parameter https://lucene.apache.org/solr/guide/6_6/the-dismax-query-parser.html#TheDisMaxQueryParser-Theqs_QueryPhraseSlop_Parameter

This parameter defines the "phrase slop" value, which represents the number of positions one word needs to be moved in relation to another word in order to match a phrase specified in a explicit phrase search query. Note: On explicit("double quoted" phrase) phrase search Apache Solr searches in "qf" queryFields

Note: The value of this setting has no influence on implicit phrase search. On explicit phrase search the Solr searches in qf (plugin.tx_solr.search.query.queryFields) defined fields.

query.bigramPhrase

Type Boolean

TS Path plugin.tx_solr.search.query.bigramPhrase

Since 8.0

Default 0

See "pf2", "ps2" https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-Thepf2Parameter

This parameter enables the bigram phrase search feature from Apache Solr. Setting is to "0" (default) does not change behaviour from Apache Solr if user searches for three and more words. Enabling bigram phrase search feature influences the scores of documents with phrase occurrences.

query.bigramPhrase.fields

Type String

TS Path plugin.tx_solr.search.query.bigramPhrase.fields

Since 8.0

Default content^10.0, title^10.0, tagsH1^10.0, tagsH2H3^10.0, tagsH4H5H6^10.0, tagsInline^10.0, description^10.0, abstract^10.0, subtitle^10.0, navtitle^10.0

See "pf2" parameter https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-Thepf2Parameter

This parameter defines what fields should be used to search in the given sentence(three+ words). Matched documents will be boosted according to fields boost value. Fields are defined as a comma separated list and same way as queryFields.

Note: The value of this setting has NO influence on explicit phrase search.

query.bigramPhrase.slop

Type Integer

TS Path plugin.tx_solr.search.query.bigramPhrase.slop

Since 8.0

Default 0

See "ps2" parameter https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-Theps2Parameter

This parameter defines the "bigram phrase slop" value, which represents the number of positions one word needs to be moved in relation to another word in order to match a phrase specified in a query.

Note: The value of this setting has NO influence on explicit phrase search.

query.trigramPhrase

Type Boolean

TS Path plugin.tx_solr.search.query.trigramPhrase

Since 8.0

Default 0

See "pf3", "ps3" https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-Thepf3Parameter

This parameter enables the phrase search feature from Apache Solr. Setting is to "0" (default) does not change behaviour from Apache Solr if user searches for two and more words. Enabling phrase search feature influences the scores of documents with phrase occurrences.

query.trigramPhrase.fields

Type String

TS Path plugin.tx_solr.search.query.trigramPhrase.fields

Since 8.0

Default content^10.0, title^10.0, tagsH1^10.0, tagsH2H3^10.0, tagsH4H5H6^10.0, tagsInline^10.0, description^10.0, abstract^10.0, subtitle^10.0, navtitle^10.0

See "pf3" parameter https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-Thepf3Parameter

This parameter defines what fields should be used to search in the given phrase. Matched documents will be boosted according to fields boost value. Fields are defined as a comma separated list and same way as queryFields.

Note: The value of this setting has NO influence on explicit phrase search.

query.trigramPhrase.slop

Type Integer

TS Path plugin.tx_solr.search.query.trigramPhrase.slop

Since 8.0

Default 0

See "ps3" parameter https://lucene.apache.org/solr/guide/6_6/the-extended-dismax-query-parser.html#TheExtendedDisMaxQueryParser-Theps3Parameter

This parameter defines the "trigram phrase slop" value, which represents the number of positions one word needs to be moved in relation to another word in order to match a phrase specified in a query.

Note: The value of this setting has NO influence on explicit phrase search.

9.6.8 results

results.resultsHighlighting

Type Boolean

TS Path plugin.tx_solr.search.results.resultsHighlighting

Since 1.0

Default 0

See Apache Solr Wiki / FastVectorHighlighter

En-/disables search term highlighting on the results page.

Note: The FastVectorHighlighter is used by default (Since 4.0) if fragmentSize is set to at least 18 (this is required by the FastVectorHighlighter to work).

results.resultsHighlighting.highlightFields

Type String

TS Path plugin.tx_solr.search.results.resultsHighlighting.highlightFields

Since 1.0

Default content

A comma-separated list of fields to highlight.

Note: The highlighting in solr (based on FastVectorHighlighter requires a field datatype with **termVectors=on**, **termPositions=on** and **termOffsets=on** which is the case for the content field). If you add other fields here, make sure that you are using a datatype where this is configured.

results.resultsHighlighting.fragmentSize

Type Integer

TS Path plugin.tx_solr.search.results.resultsHighlighting.fragmentSize

Since 1.0

Default 200

The size, in characters, of fragments to consider for highlighting. “0” indicates that the whole field value should be used (no fragmenting).

results.resultsHighlighting.fragmentSeparator

Type String

TS Path plugin.tx_solr.search.results.resultsHighlighting.fragmentSeparator

Since 3.0

Default [...]

When highlighting is activated Solr highlights the fields configured in highlightFields and can return multiple fragments of fragmentSize around the highlighted search word. These fragments are used as teasers in the results list. fragmentSeparator allows to configure the glue string between those fragments.

results.resultsHighlighting.wrap

Type String

TS Path plugin.tx_solr.search.results.resultsHighlighting.wrap

Since 1.0

Default <span class=”results-highlight”>|</span>

The wrap for search terms to highlight.

results.siteHighlighting

Type Boolean

TS Path plugin.tx_solr.search.results.siteHighlighting

Since 2.0

Default 0

Activates TYPO3’s highlighting of search words on the actual pages. The words a user searched for will be wrapped with a span and CSS class csc-sword Highlighting can be styled using the CSS class csc-sword, you need to add the style definition yourself for the complete site.

results.resultsPerPage

Type Integer

TS Path plugin.tx_solr.search.results.resultsPerPage

Since 1.0

Default {$plugin.tx_solr.search.results.resultsPerPage}

Sets the number of shown results per page.

results.resultsPerPageSwitchOptions

Type String

TS Path plugin.tx_solr.search.results.resultsPerPageSwitchOptions

Since 1.0

Default 10, 25, 50

Defines the shown options of possible results per page.

results.pagebrowser

Since 2.0

TS Path plugin.tx_solr.search.results.pagebrowser

Allows to set configuration options for the pagebrowser provided by EXT:pagebrowse.

results.ignorePageBrowser

Type Boolean

TS Path plugin.tx_solr.search.results.ignorePageBrowser

Since 1.0

Default 0

Options 0, 1

If enabled, the selected page will be ignored. Useful if you place two search plugins on a page but want one of the to always show the first results only.

results.showDocumentScoreAnalysis

Type Boolean

TS Path plugin.tx_solr.search.results.showDocumentScoreAnalysis

Since 2.5-dkd

Default 0

Options 0,1

If enabled, the analysis and display of the score analysis for logged in backend users will be initialized.

9.6.9 spellchecking

spellchecking

Type Boolean

TS Path plugin.tx_solr.search.spellchecking

Since 1.0

Default 0

Set *plugin.tx_solr.search.spellchecking = 1* to enable spellchecking / did you mean.

spellchecking.wrap

Type String

TS Path plugin.tx_solr.search.spellchecking.wrap

Since 1.0

Default |<div class="spelling-suggestions">###LLL:didYouMean### |</div>|

This can be used to configure a custom wrap for your did you mean rendering.

spellchecking.searchUsingSpellCheckerSuggestion

Type Boolean

TS Path plugin.tx_solr.search.spellchecking.searchUsingSpellCheckerSuggestion

Since 4.0

Default 0

This setting can be used to trigger a new search automatically when the previous search had no results but suggestions from the spellchecking. In this case the user can directly see the results of the best correction option.

9.6.10 lastSearches

lastSearches

Type Boolean

TS Path plugin.tx_solr.search.lastSearches

Since 1.3-dkd

Default 0

Set *plugin.tx_solr.lastSearches = 1* to display a list of the last searches.

lastSearches.limit

Type Integer

TS Path plugin.tx_solr.search.lastSearches.limit

Since 1.3-dkd

Default 10

Defines the number of last searches, that should get minded.

lastSearches.mode

Type String

TS Path plugin.tx_solr.search.lastSearches.mode

Since 1.3-dkd

Default user

Options user, global

If mode is user, keywords will get stored into the session. If mode is global keywords will get stored into the database.

9.6.11 frequentSearches

frequentSearches

Type Boolean

TS Path plugin.tx_solr.search.frequentSearches

Since 1.3-dkd, 2.8

Default 0

Set *plugin.tx_solr.search.frequentSearches = 1* to display a list of the frequent / common searches.

frequentSearches.useLowercaseKeywords

Type Boolean

TS Path plugin.tx_solr.search.frequentSearches.useLowercaseKeywords

Since 2.9

Default 0

When enabled, keywords are written to the statistics table in lower case.

frequentSearches.minSize

Type Integer

TS Path plugin.tx_solr.search.frequentSearches.minSize

Since 1.3-dkd, 2.8

Default 14

The difference between frequentSearches.maxSize and frequentSearches.minSize is used for calculating the current step.

frequentSearches.maxSize

Type Integer

TS Path plugin.tx_solr.search.frequentSearches.maxSize

Since 1.3-dkd, 2.8

Default 32

The difference between frequentSearches.maxSize and frequentSearches.minSize is used for calculating the current step.

frequentSearches.limit

Type Integer

TS Path plugin.tx_solr.search.frequentSearches.limit

Since 1.3-dkd, 2.8

Default 20

Defines the maximum size of the list by frequentSearches.select.

frequentSearches.select

Type cObject

TS Path plugin.tx_solr.search.frequentSearches.select

Since 1.3-dkd, 2.8

Defines a database connection for retrieving statistics.

9.6.12 sorting

sorting

Type Boolean

TS Path plugin.tx_solr.search.sorting

Since 1.0

Default 0

Set *plugin.tx_solr.search.sorting = 1* to allow sorting of results.

sorting.defaultOrder

Type String

TS Path plugin.tx_solr.search.sorting.defaultOrder

Since 1.0

Default asc

Options asc, desc

Sets the default sort order for all sort options.

sorting.options

This is a list of sorting options. Each option has a field and label to be used. By default the options title, type, author, and created are configured, plus the virtual relevancy field which is used for sorting by default.

Example:

```
plugin.tx_solr.search {
    sorting {
        options {
            relevance {
                field = relevance
                label = Relevance
            }

            title {
                field = sortTitle
                label = Title
            }
        }
    }
}
```

Note: As mentioned before **relevance** is a virtual field that is used to **reset** the sorting. Sorting by relevance means to have the order provided by the scoring from solr. That the reason why sorting **descending** on relevance is not possible.

sorting.options.[optionName].label

Type String / stdWrap

TS Path plugin.tx_solr.search.sorting.options.[optionName].label

Since 1.0

Defines the name of the option's label. Supports stdWrap.

sorting.options.[optionName].field

Type String / stdWrap

TS Path plugin.tx_solr.search.sorting.options.[optionName].field

Since 1.0

Defines the option's field. Supports stdWrap.

sorting.options.[optionName].defaultOrder

Type String

TS Path plugin.tx_solr.search.sorting.options.[optionName].defaultOrder

Since 2.2

Default asc

Options asc, desc

Sets the default sort order for a particular sort option.

sorting.options.[optionName].fixedOrder

Type String

TS Path plugin.tx_solr.search.sorting.options.[optionName].fixedOrder

Since 2.2

Default asc

Options asc, desc

Sets a fixed sort order for a particular sort option that can not be changed.

9.6.13 faceting

faceting

Type Boolean

TS Path plugin.tx_solr.search.faceting

Since 1.0

Default 0

Set *plugin.tx_solr.search.faceting = 1* to enable faceting.

faceting.minimumCount

Type Integer

TS Path plugin.tx_solr.search.faceting.minimumCount

Since 1.0

Default 1

See Apache Solr Wiki / Faceting mincount Parameter

This indicates the minimum counts for facet fields should be included in the response.

faceting.sortBy

Type String

TS Path plugin.tx_solr.search.faceting.sortBy

Since 1.0

Default count

Options count, index, 1, 0, true, false, alpha (1.2, 2.0), lex (1.2, 2.0)

See Apache Solr Wiki / Faceting sortParameter Parameter

Defines how facet options are sorted, by default they are sorted by count of results, highest on top. count, 1, true are aliases for each other.

Facet options can also be sorted alphabetically (lexicographic by indexed term) by setting the option to index. index, 0, false, alpha (from version 1.2 and 2.0), and lex (from version 1.2 and 2.0) are aliases for index.

faceting.limit

Type Integer

TS Path plugin.tx_solr.search.faceting.limit

Since 1.0

Default 10

Number of options to display per facet. If more options are returned by Solr, they are hidden and can be expanded by clicking a "show more" link. This feature uses a small javascript function to collapse/expand the additional options.

faceting.facetLimit

Type Integer

TS Path plugin.tx_solr.search.faceting.facetLimit

Since 6.0

Default 100

Number of options of a facet returned from solr.

faceting.singleFacetMode

Type Boolean

TS Path plugin.tx_solr.search.faceting.singleFacetMode

Since 1.2, 2.0

Default 0

Options 0, 1

If enabled, the user can only select an option from one facet at a time.

Lets say you have two facets configured, type and author. If the user selects a facet option from type its filter is added to the query. Normally when selecting another option from the other facet - the author facet - this would lead to having two facet filters applied to the query. When this option is activated the option from the author facet will simply replace the first option from the type facet.

faceting.keepAllFacetsOnSelection

Type Boolean

TS Path plugin.tx_solr.search.faceting.keepAllFacetsOnSelection

Since 2.2

Default 0

Options 0, 1

When enabled selecting an option from a facet will not reduce the number of options available in other facets.

faceting.showAllLink.wrap

Type String

TS Path plugin.tx_solr.search.faceting.showAllLink.wrap

Since 1.0

Default <li>|</li>

Defines the output of the "Show more" link, that is rendered if there are more facets given than set by faceting.limit.

faceting.showEmptyFacets

Type Boolean

TS Path plugin.tx_solr.search.faceting.showEmptyFacets

Since 1.3

Default 0

Options 0, 1

By setting this option to 1, you will allow rendering of empty facets. Usually, if a facet does not offer any options to filter a resultset of documents, the facet header will not be shown. Using this option allows the header still to be rendered when no filter options are provided.

faceting.facetLinkUrlParameters

Type String

TS Path plugin.tx_solr.search.faceting.facetLinkUrlParameters

Since 2.8

Allows to add URL GET parameters to the links build in facets.

faceting.facetLinkUrlParameters.useForFacetResetLinkUrl

Type Boolean

TS Path plugin.tx_solr.search.faceting.facetLinkUrlParameters.useForFacetResetLinkUrl

Since 2.8

Allows to prevent adding the URL parameters to the facets reset link by setting the option to 0.

faceting.facets

Type Array

TS Path plugin.tx_solr.search.faceting.facets

Since 1.0

Default type

See Apache Solr Wiki / Faceting Overview

Defines which fields you want to use for faceting. It's a list of facet configurations.

```
plugin.tx_solr.search.faceting.facets {
  type {
    field = type
    label = Content Type
  }

  category {
    field = category_stringM
    label = Category
  }
}
```

faceting.facets.[facetName] - single facet configuration

You can add new facets by simply adding a new facet configuration in TypoScript. [facetName] represents the facet's name and acts as a configuration "container" for a single facet. All configuration options for a facet are defined within that "container".

A facet will use the values of a configured index field to offer these values as filter options to your site's visitors. You need to make sure that the facet field's type allows to sort the field's value; like string, int, and other primitive types.

To configure a facet you only need to provide the label and field configuration options, all other configuration options are optional.

faceting.facets.[facetName].field

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].field

Since 1.0

Required yes

Which field to use to create the facet.

faceting.facets.[facetName].label

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].label

Since 1.0

Required yes

Used as a headline or title to describe the options of a facet.

faceting.facets.[facetName].excludeValues

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].excludeValues

Since 7.0

Required no

Defines a comma separated list of options that are excluded (The value needs to match the value in solr)

Important: This setting only makes sence for option based facets (option, query, hierarchy)

faceting.facets.[facetName].facetLimit

Type Integer

TS Path plugin.tx_solr.search.faceting.facets.[facetName].facetLimit

Since 8.0

Default -1

Hard limit of options returned by solr.

Note: This is only available for options facets.

faceting.facets.[facetName].metrics

Type Array

TS Path plugin.tx_solr.search.faceting.facets.[facetName].metrics

Since 8.0

Default empty

Metrics can be use to collect and enhance facet options with statistical data of the facetted documents. They can be used to render useful information in the context of an facet option.

Example:

```
plugin.tx_solr.search.faceting.facets {
  category {
    field = field
    label = Category
    metrics {
        downloads = sum(downloads_intS)
    }
  }
}
```

The example above will make the metric “downloads” available for all category options. In this case it will be the sum of all downloads of this category item. In the frontend you can render this metric with “<facetoptions.>.metrics.downloads” and use it for example to show it instead of the normal option count.

faceting.facets.[facetName].partialName

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].partialName

Since 7.0

Required no

By convention a facet is rendered by it’s default partial that is located in “Resources/Private/Partials/-Facets/<Type>.html”.

If you want to render a single facet with another, none conventional partial, your can configure it with “partialName = MyFacetPartial”.

faceting.facets.[facetName].keepAllOptionsOnSelection

Type Boolean

TS Path plugin.tx_solr.search.faceting.facets.[facetName].keepAllOptionsOnSelection

Since 1.2, 2.0

Default 0

Options 0, 1

Normally, when clicking any option link of a facet this would result in only that one option being displayed afterwards. By setting this option to one, you can prevent this. All options will still be displayed.

This is useful if you want to allow the user to select more than one option from a single facet.

faceting.facets.[facetName].operator

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].operator

Since 1.2, 2.0

Default AND

Options OR, AND

When configuring a facet to allow selection of multiple options, you can use this option to decide whether multiple selected options should be combined using AND or OR.

faceting.facets.[facetName].sortBy

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].sortBy

Since 1.2

Default

-

Options alpha (aliases: index, lex)

Sets how a single facet's options are sorted, by default they are sorted by number of results, highest on top. Facet options can also be sorted alphabetically by setting the option to alpha.

Note: Since 8.0 sortBy for option facets can also be a function applied on the faceted documents (e.g. avg(price_floatS)).

faceting.facets.[facetName].manualSortOrder

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].manualSortOrder

Since 2.2

By default facet options are sorted by the amount of results they will return when applied. This option allows to manually adjust the order of the facet's options. The sorting is defined as a comma-separated list of options to re-order. Options listed will be moved to the top in the order defined, all other options will remain in their original order.

Example - We have a category facet like this:

```
News Category
+ Politics (256)
+ Sports (212)
+ Economy (185)
+ Culture (179)
+ Health (132)
+ Automobile (99)
+ Travel (51)
```

Using *faceting.facets.[facetName].manualSortOrder = Travel, Health* will result in the following order of options:

```
News Category
+ Travel (51)
+ Health (132)
+ Politics (256)
+ Sports (212)
+ Economy (185)
+ Culture (179)
+ Automobile (99)
```

faceting.facets.[facetName].minimumCount

Type Integer

TS Path plugin.tx_solr.search.faceting.facets.[facetName].minumumCount

Since 8.0

Default 1

Set's the minimumCount for a single facet. This can be usefull e.g. to set the minimumCount of a single facet to 0, to have the options available even when there is result available.

Note: This setting is only available for facets that are using the json faceting API of solr. By now this is only available for the options facets.

faceting.facets.[facetName].reverseOrder

Type Boolean

TS Path plugin.tx_solr.search.faceting.facets.[facetName].reverseOrder

Since 3.0

Default 0

Options 0, 1

Reverses the order of facet options.

faceting.facets.[facetName].showEvenWhenEmpty

Type Boolean

TS Path plugin.tx_solr.search.faceting.facets.[facetName].showEvenWhenEmpty

Since 2.0

Default 0

Options 0, 1

Allows you to display a facet even if it does not offer any options (is empty) and although you have set *plugin.tx_solr.search.faceting.showEmptyFacets = 0.*

faceting.facets.[facetName].includeInAvailableFacets

Type Boolean

TS Path plugin.tx_solr.search.faceting.facets.[facetName].includeInAvailableFacets

Since 1.3

Default 1

Options 0, 1

By setting this option to 0, you can prevent rendering of a given facet within the list of available facets.

This is useful if you render the facet somewhere eles on the page using the facet view helper and don't want the facet to be rendered twice.

faceting.facets.[facetName].includeInUsedFacets

Type Boolean

TS Path plugin.tx_solr.search.faceting.facets.[facetName].includeInUsedFacets

Since 2.0

Default 1

Options 0, 1

By setting this option to 0, you can prevent rendering of a given facet within the list of used facets.

faceting.facets.[facetName].type

Type String

TS Path plugin.tx_solr.search.faceting.facets.[facetName].type

Since 2.0

Defines the type of the facet. By default all facets will render their facet options as a list. PHP Classes can be registered to add new types. Using this setting will allow you to use such a type and then have the facet's options rendered and processed by the registered PHP class.

faceting.facets.[facetName].[type]

Type Array

TS Path plugin.tx_solr.search.faceting.facets.[facetName].[type]

Since 2.0

When setting a special type for a facet you can set further options for this type using this array.

Example (numericRange facet displayed as a slider):

```
plugin.tx_solr.search.faceting.facets.size {
     field = size_intS
     label = Size

     type = numericRange
     numericRange {
         start = 0
         end = 100
         gap = 1
    }
}
```

faceting.facets.[facetName].requirements.[requirementName]

Type Array

TS Path plugin.tx_solr.search.faceting.facets.[facetName].requirements.[requirementName]

Since 2.2

Allows to define requirements for a facet to be rendered. These requirements are dependencies on values of other facets being selected by the user. You can define multiple requirements for each facet. If multiple requirements are defined, all must be met before the facet is rendered.

Each requirement has a name so you can easily recognize what the requirement is about. The requirement is then defined by the name of another facet and a list of comma-separated values. At least one of the defined values must be selected by the user to meet the requirement.

There are two magic values for the requirement's values definition:

- __any: will mark the requirement as met if the user selects any of the required facet's options
- __none: marks the requirement as met if none of the required facet's options is selected. As soon as any of the required facet's options is selected the requirement will not be met and thus the facet will not be rendered

Example of a category facet showing only when the user selects the news type facet option:

```
plugin.tx_solr {
    search {
        faceting {
            facets {
                type {
                    label = Content Type
                    field = type
                }

                category {
                    label = Category
                    field = category_stringS
                    requirements {
                        typeIsNews {
                          # typeIsNews is the name of the requirement, c
                          # choose any so you can easily  recognize what it does
                          facet = type
                          # The name of the facet as defined above
                          values = news
                          # The value of the type facet option as
                          # it is stored in the Solr index
                        }
                    }
                }
            }
        }
    }
}
```

faceting.facets.[facetName].renderingInstruction

Type cObject

TS Path plugin.tx_solr.search.faceting.facets.[facetName].renderingInstruction

Since 1.0

Overwrites how single facet options are rendered using TypoScript cObjects.

Example: (taken from issue #5920)

```
plugin.tx_solr {
    search {
        faceting {
            facets {
                type {
                    renderingInstruction = CASE
                    renderingInstruction {
                        key.field = optionValue

                        pages = TEXT
                        pages.value = Pages
                        pages.lang.de = Seiten

                        tx_solr_file = TEXT
                        tx_solr_file.value = Files
                        tx_solr_file.lang.de = Dateien

                        tt_news = TEXT
                        tt_news.value = News
                        tt_news.lang.de = Nachrichten
                    }
                }

                language {
                    renderingInstruction = CASE
                    renderingInstruction {
                        key.field = optionValue

                        0 = TEXT
                        0.value = English
                        0.lang.de = Englisch

                        1 = TEXT
                        1.value = German
                        1.lang.de = Deutsch
                    }
                }
            }
        }
    }
}
```

EXT:solr provides the following renderingInstructions that you can use in your project:

FormatDate:

This rendering instruction can be used in combination with a date field or an integer field that hold a timestamp. You can use this rendering instruction to format the facet value on rendering. A common usecase for this is, when the datatype in solr needs to be sortable (date or int) but you need to render the date as readable date option in the frontend:

```
plugin.tx_solr.search.faceting.facets {
    created {
        field = created
        label = Created
        sortBy = alpha
        reverseOrder = 1
        renderingInstruction = TEXT
        renderingInstruction {
           field = optionValue
           postUserFunc =␣
↪ApacheSolrForTypo3\Solr\Domain\Search\ResultSet\Facets\RenderingInstructions\FormatDate-
↪>format
        }
    }
}
```

9.6.14 elevation

elevation

Type Boolean

TS Path plugin.tx_solr.search.elevation

Since 3.0

Default 0

Set plugin.tx_solr.search.elevation = 1 to enable content elevation in search results.

elevation.forceElevation

Type Boolean

TS Path plugin.tx_solr.search.elevation.forceElevation

Since 3.0

Default 1

Forces content elevation to be active.

elevation.markElevatedResults

Type Boolean

TS Path plugin.tx_solr.search.elevation.markElevatedResults

Since 3.0

Default 1

If enabled, elevated results are marked with CSS class “results-elevated”.

9.6.15 variants

By using variants you can shrink down multiple documents with the same value in one field into one document and make similar documents available in the variants property. By default the field variantId is used as Solr collapsing criteria. This can be used e.g. as one approach of deduplication to group similar documents into on “root” SearchResult.

To use the different variants of the documents you can access “document.variants” to access the expanded documents.

This can be used for example for de-duplication to list variants of the same document below a certain document.

Note: Internally this is implemented with Solr field collapsing

Type Boolean

TS Path plugin.tx_solr.search.variants

Since 6.0

Default 0

Set plugin.tx_solr.search.variants = 1 to enable the variants in search results.

variants.expand

Used to expand the document variants to the document.variants property.

Type Boolean

TS Path plugin.tx_solr.search.variants.expand

Since 6.0

Default 1

variants.variantField

Used to expand the document variants to the document.variants property.

Note:: The field must be a numeric field or a string field! Not a text field!

Type String

TS Path plugin.tx_solr.search.variants.variantField

Since 6.0

Default variantId

variants.limit

Limit of expanded documents.

Type Integer

TS Path plugin.tx_solr.search.variants.limit

Since 6.0

Default 10

9.7 tx_solr.suggest

This feature allows you to show a suggest layer that suggest terms that start with the letters that have been typed into the search field

9.7.1 numberOfSuggestions

Type Integer

TS Path plugin.tx_solr.suggest.numberOfSuggestions

Since 1.1

Default 10

Sets the number of suggestions returned and displayed in the layer attached to the search field.

9.7.2 suggestField

Type String

TS Path plugin.tx_solr.suggest.suggestField

Since 1.1

Default spell

Sets the Solr index field used to get suggestions from. A general advice is to use a field without stemming on it. For practical reasons this is currently the spell checker field.

9.7.3 forceHttps

Type Boolean

TS Path plugin.tx_solr.suggest.forceHttps

Since 1.1

Default 0

Options 0,1

If enabled, HTTPS will be used for querying suggestions. Otherwise HTTP will be used.

9.7.4 treatMultipleTermsAsSingleTerm

Type Boolean

TS Path plugin.tx_solr.suggest.treatMultipleTermsAsSingleTerm

Since 1.4 / 1.7-dkd

Default 0

Options 0,1

When a user types multiple words into your search field they usually are split up into full keywords used in the query's q parameter and the last part being the partial keyword in the facet.prefix parameter. Enabling this setting moves everything into the facet.prefix parameter. This is usually only useful when using a string field as suggest / auto-complete source field.

Example - Here "Hello Solr" are the full keywords and the user started typing "World" so that "Wo" is used as the partial keyword:

"Hello Solr Wo" -> q=Hello Solr, facet.prefix=Wo (default) "Hello Solr Wo" -> q=<empty>, facet.prefix=Hello Solr Wo (treatMultipleTermsAsSingleTerm)

9.7.5 showTopResults

Type Boolean

TS Path plugin.tx_solr.suggest.showTopResults

Since 8.0

Default 1

When this setting is enabled, the top results are shown in the suggest layer. The top results are build from the first search match, or when the first search delivers no hit, the results from the first suggestion are shown.

9.7.6 numberOfTopResults

Type Integer

TS Path plugin.tx_solr.suggest.numberOfTopResults

Since 8.0

Default 5

Defines the number of top results that will be shown.

9.8 tx_solr.statistics

This section allows you to configure the logging for statistics.

Note: The statistics are logged into a mysql table. This might not make sense for high frequently used searches. In this case you should think about to connect a dedicated tracking tool.

9.8.1 statistics

Type Boolean

TS Path plugin.tx_solr.statistics

Since 2.0

Default 0

Set *plugin.tx_solr.statistics = 1* to log statistics.

9.8.2 statistics.anonymizeIP

Type Integer

TS Path plugin.tx_solr.statistics.anonymizeIP

Since 2.0

Default 0

Defines the number of octets of the IP address to anonymize in the statistics log records.

9.8.3 statistics.addDebugData

Type Boolean

TS Path plugin.tx_solr.statistics.addDebugData

Since 6.1

Default 0

Adds debug data to the columns *time_total*, *time_preparation* and *time_processing* in the table *tx_solr_statistics* from the result of the search query.

Note: Enabling addDebugData can have performance impact since debugMode is appended to queries.

9.9 tx_solr.logging

This section defines logging options. All loggings will be available in the TYPO3 logging framework.

9.9.1 debugOutput

Type Boolean

TS Path plugin.tx_solr.logging.debugOutput

Default 0

Options 0,1

Since 6.1

If enabled the written log entries will be printed out as debug message in the frontend or to the TYPO3 debug console in the backend. This setting replaces the previous setting *plugin.tx_solr.logging.debugDevLogOutput* which was needed, when the devLog was used.

9.9.2 exceptions

Type Boolean

TS Path plugin.tx_solr.logging.exceptions

Default 1

Options 0,1

Since 1.0

If enabled, thrown exceptions are logged.

9.9.3 indexing

Type Boolean

TS Path plugin.tx_solr.logging.indexing

Default 1

Options 0,1

Since 1.0

If enabled, logs when pages / documents are indexed.

9.9.4 indexing.indexQueueInitialization

Type Boolean

TS Path plugin.tx_solr.logging.indexing.indexQueueInitialization

Default 1

Options 0,1

Since 2.0

If enabled, logs the query used to initialize the indexing queue.

9.9.5 indexing.indexQueuePageIndexerGetData

Type Boolean

TS Path plugin.tx_solr.logging.indexing.indexQueuePageIndexerGetData

Default 1

Options 0,1

Since 2.0

If enabled, the requested data will be logged. Request data includes item, url, parameters, headers, data, decodedData and report.

9.9.6 query.filters

Type Boolean

TS Path plugin.tx_solr.logging.query.filters

Default 1

Options 0,1

Since 1.0

If enabled, filters will be logged when they get added to the Solr query.

9.9.7 query.searchWords

Type Boolean

TS Path plugin.tx_solr.logging.query.searchWords

Default 1

Options 0,1

Since 1.0

If enabled, received search queries will be logged.

9.9.8 query.queryString

Type Boolean

TS Path plugin.tx_solr.logging.query.queryString

Default 1

Options 0,1

Since 1.0

If enabled, query string parameters and the respective response will be logged.

9.9.9 query.rawPost

Type Boolean

TS Path plugin.tx_solr.logging.query.rawPost

Default 1

Options 0,1

Since 1.0

If enabled, POST requests against the Solr server will be logged.

9.9.10 query.rawGet

Type Boolean

TS Path plugin.tx_solr.logging.query.rawGet

Default 1

Options 0,1

Since 1.0

If enabled, GET requests against the Solr server will be logged.

9.10 Extension Configuration

The following settings can be defined in the extension manager

9.10.1 useConfigurationFromClosestTemplate

Type Boolean

Since 6.1

Default 0

When this setting is active the closest page with a typoscript template will be used to fetch the configuration. This improves the performance but limits also the possibilities. E.g. conditions can not be used that are related to a certain page.

9.10.2 useConfigurationTrackRecordsOutsideSiteroot

Type Boolean

Since 6.1

Default 1

A common common scenario is to have a site and a storage folder for records parallel to it on the same level (f.e.) If you don't want this behaviour - it should be set to false.

9.10.3 allowSelfSignedCertificates

Type Boolean

Since 6.1

Default 0

Can be used to allow self signed certificates - when using the SSL protocol.

CHAPTER

TEN

DATABASE

10.1 Database indexes

Some of the SQL statements performed on the pages table in TYPO3 perform extensive operations while copying page-trees. These operations can be speeded by by adding 2 indexes to the standard table pages.

The indexes are: * content_from_pid_deleted (content_from_pid, deleted), * doktype_no_search_deleted (doktype, no_search, deleted)

It is not required that these indexes are created by in above scenarios considerable performance gains can be achieved.

CHAPTER

ELEVEN

DEVELOPMENT

There are many ways to extend and hook into EXT:solr to customize EXT:solr for your needs.

11.1 Indexing

In this section i describe the possibilities to extend page indexing in EXT:solr with custom code.

11.1.1 Page Indexing

There are several points to extend the Typo3PageIndexer class and register own classes that are used during the indexing.

indexPageAddDocuments

Registered classes can be used to add additional documents to solr when a page get's indexed.

Registration with: $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['Indexer']['indexPageAddDocuments']
Required Interface: AdditionalPageIndexer

indexPageSubstitutePageDocument

Registered classes can be used to replace/substitute a Solr document of a page.

Registration with: $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['Indexer']['indexPageSubstitutePageDocument']
Required Interface: SubstitutePageIndexer

indexPagePostProcessPageDocument

Registered classes can be used to post process a Solr document of a page.

Registration with: $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['Indexer']['indexPagePostProcessPageDocument']
Required Interface: PageDocumentPostProcessor

11.1.2 Independent indexer

If external data should be indexed or the RecordIndexer is not required, it is possible to fill the index with an extension as well. The class can then be called e.g. by a CLI call.

```
<?php

namespace Vendor\ExtensionName\Import;

use Apache_Solr_Document;
use ApacheSolrForTypo3\Solr\ConnectionManager;
use ApacheSolrForTypo3\Solr\Domain\Site\SiteRepository;
use TYPO3\CMS\Core\Utility\GeneralUtility;

class Indexer
{

    /** @var ConnectionManager */
    protected $connectionManager;

    public function __construct()
    {
        $this->connectionManager =␣
↪GeneralUtility::makeInstance(ConnectionManager::class);
    }

    /**
     * Send data to solr index
     *
     * @param array $rows Data to be indexed, e.g. multiple DB rows
     * @param int $pageId root page
     * @param int $language language id
     */
    public function index(array $rows, int $pageId = 1, int $language = 0)
    {
        $documents = [];

        foreach ($rows as $row) {
            $documents[] = $this->createDocument($row, $pageId);
        }

        $connection = $this->connectionManager->getConnectionByPageId($pageId,
↪$language);
        $connection->addDocuments($documents);
    }


    /**
     * Remove all from index
     *
     * @throws \ApacheSolrForTypo3\Solr\NoSolrConnectionFoundException
     */
    public function clearIndex() {
        $connections = $this->getSolrConnections();
        foreach ($connections as $connectionLanguage => $connection) {
            /** @var ConnectionManager */
            $connection->deleteByType('cutom_type');
        }
    }

    /**
     * Create a solr document which then is sent to solr
     *
     * @param array $row
     * @param int $pageId
     * @return Apache_Solr_Document
     */
    protected function createDocument(array $row, int $pageId): Apache_Solr_
↪Document
```

```
    {
        $document = $this->getBaseDocument($row, $pageId);

        $solrFieldMapping = [
            'title' => 'title',
            'summary' => 'abstract',
            'information' => 'content',
            'keywords' => 'keywords',
            'area' => 'area_stringS',
            'category' => 'category_stringS'
        ];

        foreach ($row as $key => $value) {
            if (isset($solrFieldMapping[$key])) {
                $document->setField($solrFieldMapping[$key], $value);
            }
        }

        // url generation
        $document->setField('url', 'todo'); // custom implementation required

        return $document;
    }

    /**
     * Creates a Solr document with the basic / core fields set already.
     *
     * @param array $itemRecord The record to use to build the base document
     * @param int $rootPageId root page id
     * @return Apache_Solr_Document A basic Solr document
     */
    protected function getBaseDocument(array $itemRecord, int $rootPageId): Apache_
↪Solr_Document
    {
        $siteRepository = GeneralUtility::makeInstance(SiteRepository::class);
        $site = $siteRepository->getSiteByRootPageId($rootPageId);

        $document = GeneralUtility::makeInstance(Apache_Solr_Document::class);

        // required fields
        $document->setField('id', 'cutom_type_' . $itemRecord['uid']);
        $document->setField('variantId', 'cutom_type' . $itemRecord['uid']);
        $document->setField('type', 'cutom_type');
        $document->setField('appKey', 'EXT:solr');
        $document->setField('access', ['r:0']);

        // site, siteHash
        $document->setField('site', $site->getDomain());
        $document->setField('siteHash', $site->getSiteHash());

        // uid, pid
        $document->setField('uid', $itemRecord['uid']);
        $document->setField('pid', 1);

        return $document;
    }
}
```

11.2 Developing Backend Modules

11.2.1 Backend Components

EXT:solr provides UI components for backend modules. Some components hold (GUI)state and some not, but all components calling actions(changing the extension and/or GUI state!), and then redirecting to the actions within the component was used(referrer also) or to the defined by callers action uri, if that is required by UX.

Below are all available components listed and their responsibility.

CoreSelector

Renders menu in backends doc header with available Solr cores for selected Site and changes the solr core by clicking on option in drop down menu.

- Provides following methods, which must be called inside the *initializeView(...)* method in your controller to render this component in Backend:
  - *generateCoreSelectorMenuUsingSiteSelector()*
    - Use this method together with SiteSelectorMenu component.
  - *generateCoreSelectorMenuUsingPageTree()*
    - Use this method if you are using original page tree from CMS.
- Provides following Actions for changing state, must be added to actions list of your controller:
  - *switchCore*
- Provides following fully initialized properties in utilizing action controller:
  - *$selectedSolrCoreConnection from type ApacheSolrForTypo3SolrSolrService*

If you need the possibility to switch the core, you can extends the AbstractModuleController (in ApacheSolr-ForTypo3SolrControllerBackendSearch).

11.2.2 FAQ

Why should I add some action name to my action controller?

To allow calling this action within your controller, component can use own controller for changing state only if that is hardcoded with some module and allowed by ACL for all(or almost all) be users/groups, but this is a bag approach. Therefore allow changing something, only if that is needed.

What do I need to do for using Backend Components?

By extending ApacheSolrForTypo3SolrControllerBackendSearchAbstractModuleController your module has the pagetree (to select the side) and the core selector, to select the needed solr core.

11.3 Development Environment

To simplify the development for TYPO3 and solr related components we provide a development environment based on vagrant and the Homestead box of the TYPO3 core.

You can find the development box in the following git repository:

https://github.com/TYPO3-Solr/solr-typo3-devbox

When you start the box, you will find a pre-configured environment for the support TYPO3 LTS version and the solr installation that is needed for the installed EXT:solr version.

11.4 Testing and Continues Integration

The goal during the development of EXT:solr is, to test most of the components with unit and integration tests.

11.4.1 Unit Tests

For the single classes we've added unit tests whenever we though it is usefull. The unit tests should run very quickly, to git a quick response.

11.4.2 Integration Tests

As in EXT:solr the integration tests are more complex and test the integration of the different components. Since a database server and a solr server is required this is needed to run the integration test suite. During the bootstrap of the test environment, we use the TYPO3 core functionality for database tests and we install a local solr server with out install script.

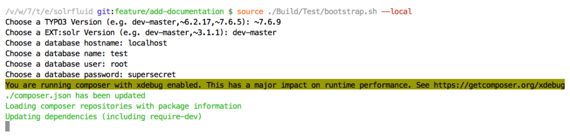
To simplify the local usage of the unit and integration tests, we ship a few bash script that support you to get everything started.

11.4.3 Bootstrapping the Test Environment

When you want to start the testrunner in your shell you need to bootstrap it once:

```
source ./Build/Test/bootstrap.sh --local
```

The bootstrapper will prompt for some values:



```
/v/w/7/t/e/solrfluid git:feature/add-documentation $ source ./Build/Test/bootstrap.sh --local
Choose a TYPO3 Version (e.g. dev-master,~6.2.17,~7.6.5): ~7.6.9
Choose a EXT:solr Version (e.g. dev-master,~3.1.1): dev-master
Choose a database hostname: localhost
Choose a database name: test
Choose a database user: root
Choose a database password: supersecret
You are running composer with xdebug enabled. This has a major impact on runtime performance. See https://getcomposer.org/xdebug
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
```

When the boostrapper was finished successful the following was done:

- Environment variables for the TYPO3 testing framework have been exported
- Test database was created
- Test solr instance was created

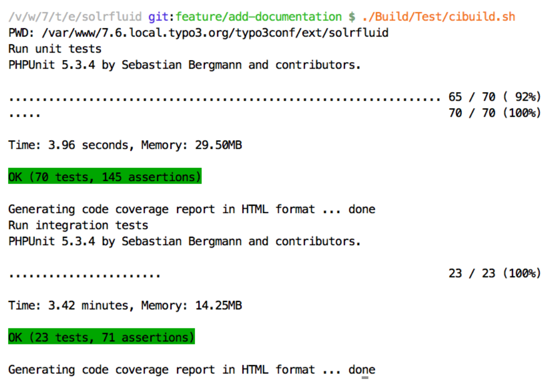
Afterwards you can run the ci suite in your shell

11.4.4 Running the ci Suite

When the test environment was boostrapped correctly you can start the test runner:

```
./Build/Test/cibuild.sh
```

When everything is configured correctly all tests should run through and you should get a green bar:



```
/v/w/7/t/e/solrfluid git:feature/add-documentation $ ./Build/Test/cibuild.sh
PWD: /var/www/7.6.local.typo3.org/typo3conf/ext/solrfluid
Run unit tests
PHPUnit 5.3.4 by Sebastian Bergmann and contributors.

.................................................................  65 / 70 ( 92%)
.....                                                              70 / 70 (100%)

Time: 3.96 seconds, Memory: 29.50MB

OK (70 tests, 145 assertions)

Generating code coverage report in HTML format ... done
Run integration tests
PHPUnit 5.3.4 by Sebastian Bergmann and contributors.

.......................                                            23 / 23 (100%)

Time: 3.42 minutes, Memory: 14.25MB

OK (23 tests, 71 assertions)

Generating code coverage report in HTML format ... done
```

11.5 Code Structure

The components of EXT:solrfluid have been developed with the domain driven design (DDD) approach (https://de.wikipedia.org/wiki/Domain-driven_Design) for our extension we tried to separate the code by the following layers:

- **Domain**: Everything that is related to the "search" domain should be implemented here.
- **System**: Everything that is related to the "system" (e.g. TYPO3 specific) should be implemented here.

11.5.1 Domain Layer & Domain Model

The classes of the domain layer are located in "Classes/Domain" and should contain everything that is related to the "search domain".

ResultSet

The "SearchResultSet" is the main entity that you get passed to the view. It can be used to access all search related objects on your result page.

The SearchResultSet can be used e.g. to get facets and spelling suggestions. A focus for the first release was a new domain model for facets, that can be rendered with fluid or any other template engine.

Facets

The following UML diagram shows the implemented facets in EXT:solrfluid. Every facet has one or more facet items attached. For the **OptionsFacet** the FacetItem is an **Option**, for the **NumericRangeFacet** a **NumericRange**.

Rendering of a facet:

Based on the **"type"** TypoScript configuration the **"FacetRegistry"** chooses the responsible facet package class that is used to create the object structure from the solr repsonse. Each facet type is shipped with a default fluid partial, that is able to render such a facet.

A facet package consists of the following parts:

- The facet parser: Responsible to parse the apache solr response into a domain object structure
- The url decoder: Responsible to decode the url fragment of the facet in EXT:solr
- The query builder: Responsible to build the needed query fragment for Apache Solr to apply the facet

The typoscript configuration **"partialName"** can be used to force the rendering with another fluid partial.

For advanced use cases you can use the **"FacetRegistry"** to register your own facet type or overwrite the facet behaviour for a certain facet type.

As you see in the diagram above solr ships a clean object structure of the facets, that you can render in your custom templates as you need them.

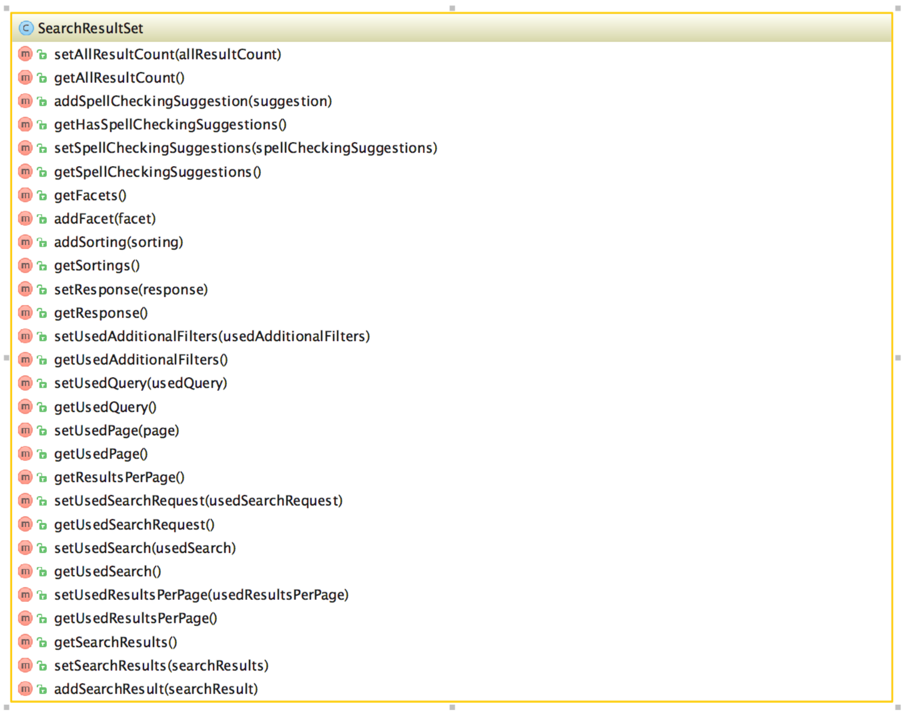

SearchResultSet
setAllResultCount(allResultCount)
getAllResultCount()
addSpellCheckingSuggestion(suggestion)
getHasSpellCheckingSuggestions()
setSpellCheckingSuggestions(spellCheckingSuggestions)
getSpellCheckingSuggestions()
getFacets()
addFacet(facet)
addSorting(sorting)
getSortings()
setResponse(response)
getResponse()
setUsedAdditionalFilters(usedAdditionalFilters)
getUsedAdditionalFilters()
setUsedQuery(usedQuery)
getUsedQuery()
setUsedPage(page)
getUsedPage()
getResultsPerPage()
setUsedSearchRequest(usedSearchRequest)
getUsedSearchRequest()
setUsedSearch(usedSearch)
getUsedSearch()
setUsedResultsPerPage(usedResultsPerPage)
getUsedResultsPerPage()
getSearchResults()
setSearchResults(searchResults)
addSearchResult(searchResult)

Fig. 11.1: UML diagram of the SearchResutSet

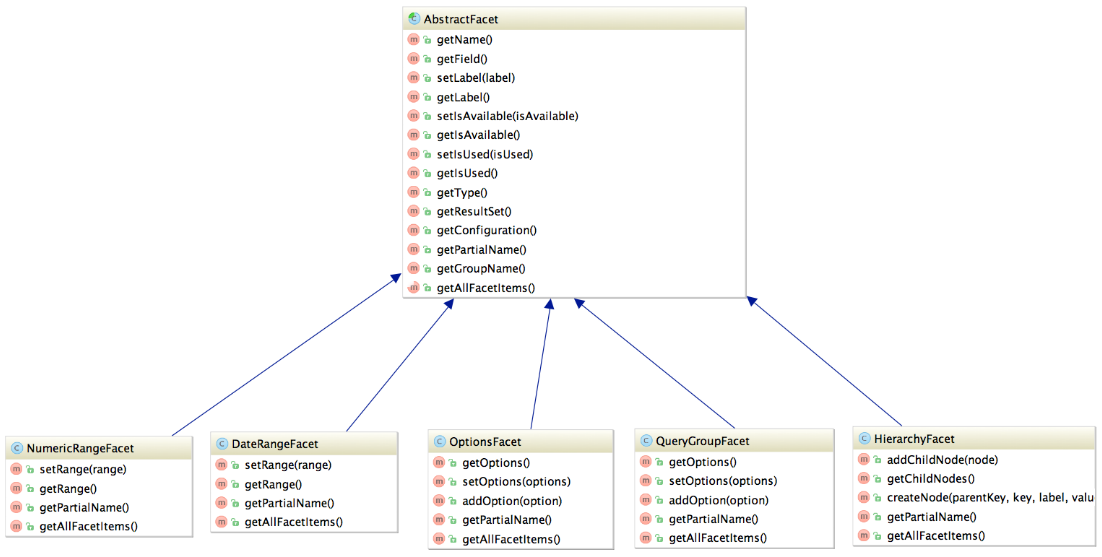


Fig. 11.2: UML diagram of the *Facet classes

11.6 ViewHelpers

Beside the controllers, the domain objects and the templates we ship a few useful view helpers. To avoid a strong coupling between the extension and fluid as template engine we tried to keep all ViewHelpers as "slim" as possible. Whenever it was possible we moved the logic into custom service classes and just use them in the ViewHelper.

Since everything belongs to the **"SearchResultSet"** and we wanted to avoid the need ob passing this object around from "template to template" and "partial to partial" we decided to provide an own "ControllerContext" that referenced the "SearchResultSet". With this approach, it is possible to access the **"SearchResultSet"** in every ViewHelper.

With the current release we ship the following concrete ViewHelpers:

Path	**Description**
s:debug.documentScoreAnalyzer	Used to render the score analysis.
s:debug.query	Shows the solr query debug information.
s:document.highlightResult	Performs the highlighting on a document.
s:document.relevance	Shows the relevance information for a document.
s:facet.area.group	Filters the facets in the rendering scope to one group.
s:facet.options.group.prefix.labelPrefixes	Provides an array of available label prefixes that can be used to filter with s:facet.options.group.prefix.labelFilter.
s:facet.options.group.prefix.labelFilter	Filters the options of a facet by a given prefix.
s:uri.facet.addFacetItem	Add's a facet item to the current url.
s:uri.facet.removeAllFacets	Removes all facet items from the current url.
s:uri.facet.removeFacet	Removes all options from one facet.
s:uri.facet.removeFacetItem	Removes a single facet item from the url.
s:uri.facet.setFacetItem	Sets one single item for a facet (and removes other setted)
s:uri.paginate.resultPage	Creates a link to a result page of the current search.
s:uri.search.currentSearch	Creates a link to the current search (with facets, sorting...)
s:uri.search.startNewSearch	Creates a link for a new search by a term.
s:uri.sorting.removeSorting	Creates a link to the current search and removes the sorting.
s:uri.sorting.setSorting	Creates a link to the current search and sets a new sorting.
s:pageBrowserRange	Provides the range data for the pagination.
s:searchForm	Renders the searchForm.
s:translate	Custom translate ViewHelper (uses translations from ext:solr)

CHAPTER

TWELVE

FAQ - FREQUENTLY ASKED QUESTIONS

What does the term '"Core"<https://cwiki.apache.org/confluence/display/solr/Solr+Cores+and+solr.xml>'_ mean?

This term relates to Apache Solr indexes and means a single distinct part of an index. It is possible to use multiple cores on one single Apache Solr instance. Good examples are using a different Apache Solr core for each language or of course a separate core for each website. For more informations please refer to the Apache Solr documentation.

Where can I report a bug?

Please make sure that this bug is not reported already, use also the search function of our issue tracker. Our issue tracker is on GitHub.

Where can I report a security issue?

If you have found a security issue in our extension, please do not post about it in a public channel. Please send an email to the TYPO3 security team with detailed description of found vulnerability. For more details about security issue handling see *https://typo3.org/teams/security/contact-us/*

Is there some chat/irc channel for EXT:solr available?

Join us on the official Slack for TYPO3 and get answers related to EXT:solr in the #ext-solr channel immediately!

Which plugins(TYPO3 Frontend) are avalable?

- Search: Form only
- Search: Form, Result, Additional Components
- Search: Frequent Searches

Just insert one of this plugins on corresponding page to fade in the search form and/or supply the front end with a search results.

When i open the search page i see the message 'Search is currently not available. ', whats wrong?

Did you configure your solr connection as required?

- Please read "*Configure Extension*" and check if you have configured everything
- Did you configure solr server and port and does the scheme and path match?
- Did you click "Initialize connection" after configuring the solr server?
- Can you access the solr server with wget or curl from the command line?
- Is the system report of EXT:solr green?

In which cases do I want to trigger indexing manually?

- after changing any configuration file.
- after modifying synonyms, stop words, protected words in TYPO3 Backend -> Search

Moreover by changing core/index configuration you need to reload the core to make the changes become active. To reload configuration you can either restart the whole Solr server or simply reload a specific core.

I want to index files with EXT:solr. How can i do that?

We provide an addon called EXT:solrfal, that allows you to index files from FAL into Solr. This addon is currently available for partner only.

How can i use Fluid templates with EXT:solr < v7.0.0?

For the Fluid rendering in EXT:Solr >= 5.0 <= 6.1 we provide the addon EXT:solrfluid, that allows you to render your search results with Fluid. Since EXT:Solr 7.0 Fluid is the default templating engine.

Which versions of EXT:solr / EXT:solrfal and EXT:solrfluid work together?

Please check the *Appendix - Version Matrix*, the you can find the proposed version combinations.

My indexed documents are empty, i can not find the content of a page?

Did you configure the search markers (“<!– TYPO3SEARCH_begin –>” and “<!– TYPO3SEARCH_end –>”) on your page? Check the paragraph *Search Markers* and make sure your page renders them.

I have languages in TYPO3 that are not relevant for the search. How can i exclude them?

You need to enable the search just for the relevant languages.

Example:

```
plugin.tx_solr.enabled = 0

[globalVar = GP:L = 0]
    plugin.tx_solr {
        enabled = 1
        solr.path = /solr/core_de/
    }
[globalVar = GP:L = 8|9]
    plugin.tx_solr {
        enabled = 1
        solr.path = /solr/core_en/
    }
[END]
```

The extension is indexing into the wrong core for multi-language sites. What’s wrong?

When indexing pages the page indexer retrieves the core from the TypoScript configuration. That configuration is determined by the language (GET L parameter). However, when the indexer tries to index a page that has not been translated TYPO3 will by default still render the page but falling back to the default language page. By that TYPO3 will also use the TypoScript configuration for the default language which usually points to a different Solr core.

Solution: Make sure you have configured config.sys_language_mode to use content_fallback. This way TYPO3 will fall back to the configured language’s content, but will use the TypoScript configuration for the requested language.

When i change a record, no update is detected. What’s wrong?

Are your records inside of your site root? EXT:solr record monitor processes records that belong to your site, which means they need to be below your site root. If you want to index records that are outside your sideroot, you need to configure the page id’s of the sysfolder as additionalPageIds:

```
plugin.tx_solr.index.queue.[yourQueueName].additionalPageIds = 4711,4712
```

There are two datatypes for text stringS and textS. When should i choose which datatype?

String data types like stringS store the *raw* string. No processing, like stemming, splitting etc. is applied. The processing is useful when you want to search in the field and support more then exact matches. When you just want to display the content you should choose a *stringS* type, when you want to search in the field you should choose *textS*.

I am adding content to a dynamic field but when i search for the content i can not find the document. What's wrong?

Beside the indexing part you need to configure the query part. Make sure that all relevant fields are configured as query fields:

```
plugin.tx_solr.search.query.queryFields := addToList(test_textS\^1.0)
```

I don't find the expected document on the first position. What can i do?

:) That's a good question. In the end, solr is a search and the sorting depends on the score, not as in a database on one or two simple criterion.

In the end solr provides a lot of settings that influence the score calculation and you need to tune the results to you needs. The following settings are helpful to tune your results.

Check your data

The quality of you data is important. Maybe a document is on the first position because, the search term is really relevant for it? Maybe it is an option to change the content?

Adjust the query field boost factors

For each query field there is a boost value after the ^ sign. To increase the factor of a single field for the whole query, you can increase the number in the query fields.

Example:

```
plugin.tx_solr.search.query.queryFields = title^20.0, title^15.0
```

Use boostFunctions or boostQueries

For use cases like "*news* are always more important then *pages*" or "Newer documents should be at the beginning" you can use boostFunctions (*query.boostFunction*) or boostQueries (*query.boostQuery*)

The search term only exists as a synonym

You can use the backend module synonyms (*Synonyms*) to maintain synonyms and configure solr to retrieve documents by a term that is not naturally inside the document.

Ask DKD support

Beside that, there are more options to tune. The DKD support can help you, to analyze and tune your search results. Call +49 (0)69 - 247 52 18-0.

Non ASCII characters like german umlauts do not work when i search, how do I fix that?

To allow search with umlauts Tomcat needs to be configured to use UTF-8 encoded urls. Go to apache-tomcat/conf/server.xml and change the URIEncoding parameter:

```
<Connector port="8080" protocol="HTTP/1.1"
    connectionTimeout="20000" redirectPort="8443"
    URIEncoding="UTF-8" />
```

How can I change Solr's schema and add custom fields?

Please do not change the shipped solr schema. There are a lot of dynamic fields (*Appendix - Dynamic Fields*) that can be used to index any kind of datatype.

I am using varnish before my site. How can i index pages properly?

SOLR Indexer might have some issues, when the page to index is behind a Varnish Proxy. We have collected two ways of solving this issue

Bypassing varnish:

Bypass when X-Tx-Solr-Iq is present

The SOLR indexer request send the header X-Tx-Solr-Iq.

To have bypass the Varnish caching, put this into your sub vcl_recv part of the configuration

```
if (req.http.X-Tx-Solr-Iq) {
    return(pipe);
}
```

Using Cache-Control:

Put this into your sub vcl_fetch part of the configuration

```
if (req.http.Cache-Control ~ "no-cache") {
    set beresp.ttl = 0s;
    # Make sure ESI includes are processed!
    esi;
    set beresp.http.X-Cacheable = "NO:force-reload";
    # Make sure that We remove all cache headers, so the Browser does not cache it␣
↪for us!
    remove beresp.http.Cache-Control;
    remove beresp.http.Expires;
    remove beresp.http.Last-Modified;
    remove beresp.http.ETag;
    remove beresp.http.Pragma;
```

```
      return (deliver);
}
```

I want to build the Dockerfile_full image on my mac with a local volume, how can i do that?

The following example shows how to build the Dockerfile image and start a container with a mapped local volume (only for the data). This was tested with “Docker for Mac” (not Docker Toolbox). Before executing the example, make sure, that you have added “~/solrdata” as allowed volume in the docker configuration.

```
# build the image
docker build -t typo3-solr -f Dockerfile .

# create volume directory locally
mkdir -p ~/solrdata

# add solr group to volume directory
sudo chown :8983 ~/solrdata

# run docker container from image with volume
docker run -d -p 127.0.0.1:8282:8983 -v ~/solrdata:/opt/solr/server/solr/data typo3-
↪solr
```

Can i index a https (SSL) site?

Yes. You need a ssl certificate (can be self signed) and change the following setting:

```
plugin.tx_solr.index.queue.pages.frontendDataHelper.scheme = https
```

I want to index a value into a multiValue field from a user function. How can i do that?

You can do that, by using SOLR_MULTIVALUE

```
plugin.tx_solr.index.queue.indexConfigName {
    fields {
      somevalue_stringM = SOLR_MULTIVALUE
      somevalue_stringM {
           stdWrap.cObject = USER
           stdWrap.cObject.userFunc = Vendor\Ext\Classname->getValues
           separator=,
      }
    }
}
```

How can i use a configuration from AdditionalConfiguration.php when i deploy my application on several instances?

The configuration of the connection is done with typoscript. When you want to use a configuration from TYPO3_CONF_VARS or from the system environment, you can apply an stdWrap on the configuration that reads from these configurations.

The following example shows how a host can be configured in the AdditionalConfiguration.php and used in your typoscript to connect to solr:

The following line is added to AdditionalConfiguration.php

```
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['host'] = 'mysolrserver.de';
```

To use this configuration for the host, you can use a TEXT element in the configuration and use override.data to use the value from the AdditionalConfiguration.php

```
plugin.tx_solr.solr {
   host = TEXT
   host {
     value = localhost
     override.data = global:TYPO3_CONF_VARS|EXTCONF|solr|host
   }
}
```

I want to index extension records, what do i need to do?

EXT:solr provides a flexible indexing for TYPO3 pages and records. You can add a custom indexing configuration for your own records with a valid TCA configuration.

You can read more about this in the section *IndexQueue Configuration*.

The following things are important:

- The extension ships several examples in the Folder "Configuration/TypoScript/Examples", read them and try to undestand them.
- EXT:solr can not know the business logic of an extension to generate a link to a detail view. You need to use typolink to build an url that points to a valid, existing detail page.
- When you index records, e.g. news it these records are indexed in solr and point to a news details page. That's the reason why it makes sence to exclude the news detail page from the normal page indexing. Otherwise the indexing of this page will produce an error message, because only a url with a valid news uid produces a valid output.

Are in EXT:solr some cli commands available?

Yes, currently(v. 6.1) only one for initializing solr connections. But check for new ones with `bin/typo3 list` command.

I want to overwrite the type field, why is this not possible?

The type field is a system field that EXT:solr uses to keep the system in sync. Overwritting this field might result in inconsistency. However, if you need something like a custom type you can also write the information to a dynamic solr field and use that one as a type.

The following example shows, how to fill the field "mytype_stringS" and build a facet on this field:

```
plugin.tx_solr {
    index{
        queue{
            news = 1
            news {
                table = tt_news

                fields {
                    mytype_stringS = TEXT
                    mytype_stringS.value = news

                }
            }
        }
    }
    search.faceting.facets.mytype_stringS {
         label = Type
         field = mytype_stringS
    }
}
```

I want to implement a toggle functionality for facet options as previously possible with selectingSelected-FacetOptionRemovesFilter. How can i do that?

This is completely possible with Fluid core ViewHelpers and the domain model. The following steps are required.

Register a custom partial to render the facet:

```
plugin.tx_solr.search.faceting.facets.<facetName>.partialName = OptionsToggle
```

This is the content of the OptionsToggle Partial (Feel free to adapt it to your needs):

```
<h5 class="facet-label">{facet.label}</h5>
<ul class="facet-option-list facet-type-options fluidfacet" data-facet-name="{facet.
↪name}" data-facet-label="{facet.label}">
    <f:for each="{facet.options}" as="option" iteration="iteration">
        <li class="facet-option{f:if(condition:'{iteration.index} > 9', then:' tx-
↪solr-facet-hidden')}" data-facet-item-value="{option.value}">
            <f:if condition="{option.selected}">
                <f:then><a class="facet solr-ajaxified" href="{s:uri.facet.
↪removeFacetItem(facet: facet, facetItem: option)}">{option.label}</a></f:then>
                <f:else><a class="facet solr-ajaxified" href="{s:uri.facet.
↪addFacetItem(facet: facet, facetItem: option)}">{option.label}</a></f:else>
            </f:if>
            <span class="facet-result-count">({option.documentCount})</span>
        </li>
    </f:for>
    <f:if condition="{facet.options -> f:count()} > 10">
        <li>
            <a href="#" class="tx-solr-facet-show-all" data-label-more="
↪{s:translate(key:'faceting_showMore', extensionName:'solr')}"
                data-label-less="{s:translate(key:'faceting_showFewer',␣
↪extensionName:'solr')}">
```

```
                <s:translate key="faceting_showMore" extensionName="solr">Show more
↪</s:translate>
            </a>
        </li>
    </f:if>
</ul>
```

I want to store HTML in solr, how can i retrieve that?

In general it is not recommend to allow html in the solr field. Especially when you index content that can be changed by the user.

However, if you want to allow html in a solr field, you need to add the field as trusted field and the content will not be escaped during the retrieval from solr.

The following example shows how to avoid html in the content field:

```
plugin.tx_solr.search.trustedFields = url, content
```

Note: When you allow html in the content please make sure that the usage of crop ViewHelpers or a limit of the field length does not break your markup.

I want to use two instances of the search plugin on the same page, how can i do that?

If you want to use two search plugins on the same page you can add two instances and assign a different "Plugin Namespace" in the flexform. If you want to avoid, that both plugins react on the global "q" parameter, you can disable this also in the flexform. Each instance is using the querystring from <pluginNamespace>[q] then.

How can i configure switchable templates for the results plugin?

The following example shows, how you can configure a custom switchable entry template for the Results plugin:

```
plugin.tx_solr {
    view {
        templateRootPaths.100 = EXT:your_config_extension/Resources/Private/
↪Templates/
        partialRootPaths.100 = EXT:your_config_extension/Resources/Private/Partials/
↪
        layoutRootPaths.100 = EXT:your_config_extension/Resources/Private/Layouts/
        templateFiles {
            results = Results
            results.availableTemplates {
                default {
                    label = Default Searchresults Template
                    file = Results
                }
                products {
                    label = Products Template
                    file = ProductResults
                }
            }
        }
    }
}
```

I want to use EXT:solr with a deployment and pass connection settings from outside e.g. by the environment, how can i do that?

When you deploy a system automatically and you use EXT:solr there are some things that might be complicated:

- You want to use a different solr endpoint for each environment
- EXT:solr depends on an existing domain record

To avoid that, you can set or generate these settings in the TYPO3 AdditionalConfigruation.php file and use them in your system.

To configure a used domain you cat set:

```
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['sites'][###rootPageId###]['domains
↪'] = ['mydomain.com'];
```

You can also define the data for your solr endpoints there and use them in the typoscript:

```
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['sites'][###rootPageId###]['solrhost
↪'] = 'solr1.local';
$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['sites'][###rootPageId###]['solrport
↪'] = 8083;
```

And use them in your TypoScript configuration:

```
plugin.tx_solr {
    solr {
        host = TEXT
        host {
            value = {$plugin.tx_solr.solr.host}
            override.data = global:TYPO3_CONF_VARS|EXTCONF|solr|sites|###rootPageId
↪###|solrhost
        }
        port = TEXT
        port {
            value = {$plugin.tx_solr.solr.port}
            override.data = global:TYPO3_CONF_VARS|EXTCONF|solr|sites|###rootPageId
↪###|solrport
        }
    }
}
```

I want to use faceting.facets.[facetName].singleOptionMode why was it removed?

This setting belongs to the rendering and not to the facet itself. You can implement the same behaviour just with the given ViewHelpers.

The behaviour is the same, when you just call the ViewHelper s:uri.facet.setFacetItem instead of s:uri.facet.addFacetItem, which semantically just overwrites the current value.

We've added an example partial "OptionsSinglemode" that shows this behaviour. The example TypoScript template "Search - (Example) Options with singlemode (only one option at a time)" shows how to use this partial in combination with the setting "keepAllOptionsOnSelection".

I want to build a tab facet where all options remain, even with an option count of 0. How can i do that?

This can be done with the combination of several settings:

```
plugin.tx_solr.search.faceting {
    minimumCount = 0
    keepAllFacetsOnSelection = 1
    facets {
        typeTab {
            field = type
            keepAllOptionsOnSelection = 1
        }
    }
}
```

The example above changes the minimumCount to 0, the default value i 1. Setting it to zero allows to have options without any results. The setting "keepAllFacetsOnSelection" let all facets remain and with keepAllOptionsOnSelection the options in the type facet remain.

How can i add a searchbox on every page?

In most projects you want to add a searchbox on every content page. To support this, the default EXT:solr typoscript template provides the typoscript template path “plugin.tx_solr_PiSearch_Search” that contains a configured typoscript code to render the searchbox. When you want to add that to your project in the most cases you would need to refer to a search result page. The following example shows how you can build a typoscript lib object that configures the target page for this plugin instance:

```
lib.searchbox < plugin.tx_solr_PiSearch_Search
lib.searchbox.search.targetPage = 4711
```

Afterwards you could render the typoscript path “lib.searchbox” with several ways in TYPO3, e.g. with a FLUID ViewHelper:

```
<f:cObject typoscriptObjectPath="lib.searchbox" />
```

By adding the snippet to a generic tempate you could render the searchbox on every page.

CHAPTER

THIRTEEN

APPENDIX - DYNAMIC FIELDS

Dynamic fields allow you to add custom fields to Solr documents. That said, you never need to modify Solr's schema (which could cause problems or at least unnecessary additional work when updating the Solr extension). The following sections describe how to use dynamic fields with your Solr for TYPO3 installation. Usage of dynamic fields

You can use dynamic fields by following a special naming convention for document fields. E.g. to create a dynamic field that is a string the field name should end with _stringS. So if you want to create a field for storing a title you would name it title_stringS. We suggest you use lower camel case for the field name followed by an underscore followed by the dynamic field type "extension".

We've predefined the following dynamic fields:

Extension	Type	Multivalue	Comment
*_stringS	String	No	
*_stringM	String	Yes	
*_stringCollatedS	string_collated	No	
*_stringCollatedM	string_collated	Yes	
*_binS	binary	No	Stored but not indexed
*_binM	binary	Yes	Stored but not indexed
*_boolS	Boolean	No	
*_boolM	Boolean	Yes	
*_intS	Integer	No	deprecated use _tIntS now
*_intM	Integer	Yes	deprecated use _tIntM now
*_sIntS	Sortable Integer	No	not available use _tIntS now
*_sIntM	Sortable Integer	Yes	not available use _tIntM now
*_tIntS	Trie Integer	No	
*_tIntM	Trie Integer	Yes	
*_longS	Long	No	deprecated use _tLongS now
*_longM	Long	Yes	deprecated use _tLongM now
*_sLongS	Sortable Long	No	not available use _tLongS now
*_sLongM	Sortable Long	Yes	not available use _tLongM now
*_tLongS	Trie Long	No	
*_tLongM	Trie Long	Yes	
*_floatS	Float	No	deprecated use _tFloatS now
*_floatM	Float	Yes	deprecated use _tFloatM now
*_sFloatS	Sortable Float	No	not available use _tFloatS now
*_sFloatM	Sortable Float	Yes	not available use _tFloatM now
*_tFloatS	Trie Float	No	
*_tFloatM	Trie Float	Yes	
*_doubleS	Double	No	deprecated use _tDoubleS now
*_doubleM	Double	Yes	deprecated use _tDoubleM now
*_sDoubleS	Sortable Double	No	not available use _tDoubleS now
*_sDoubleM	Sortable Double	Yes	not available use _tDoubleM now

Continued on next page

Table 13.1 – continued from previous page

Extension	Type	Multivalue	Comment
*_tDoubleS	Trie Double	No	
*_tDoubleM	Trie Double	Yes	
*_tDouble4S	Trie Double with Precision Step 4	No	
*_tDouble4M	Trie Double with Precision Step 4	Yes	
*_dateS	Date	No	deprecated use _tDateS now
*_dateM	Date	Yes	deprecated use _tDateM now
*_tDateS	Trie Date	No	
*_tDateM	Trie Date	Yes	
*_random	Random	No	
*_textS	Text	No	
*_textM	Text	Yes	
*_textTS	Text Tight	No	
*_textTM	Text Tight	Yes	
*_textSortS	Sortable Text	No	
*_textSortM	Sortable Text	Yes	
*_textWstS	Whitespace tokenized Text	No	
*_textWstM	Whitespace tokenized Text	Yes	
*_textEdgeNgramS	Edge Ngram (hello => hello, hell..)	No	
*_textEdgeNgramM	Edge Ngram (hello => hello, hell..)	Yes	
*_textNgramS	Ngram (hello => he,ll,lo,hel,llo)	No	
*_textNgramM	Ngram (hello => he,ll,lo,hel,llo)	Yes	
*_textPath	textPath	No	
*_textExactS	textExact	No	
*_textExactM	textExact	Yes	
*_textSpellS	textSpell	No	
*_textSpellM	textSpell	Yes	
*_phoneticS	Phonetic	No	
*_phoneticM	Phonetic	Yes	
*_point	point	No	
*_location	location	No	
*_coordinate	double		
*_locationRpt	locationRpt	No	
*_currency	currency	No	

CHAPTER

FOURTEEN

APPENDIX - VERSION MATRIX

This is a list of EXT:solr versions and what versions of Apache Solr and TYPO3 are supported for each release.

EXT: solr	Apache Solr	TYPO3	Schema	Solrconfig	EXT:tika	EXT:solrfal	EXT:solrfluid	EXT:solrgrouping	EXT:solrfluidgrouping	Access-plugin
3.1	4.10	6.2 - 7.6	tx_solr-3-1-0–20150614	tx_solr-3-1-0–20151012	2.0	2.1	N/A	1.1	N/A	1.3
4.0	4.10	7.6	tx_solr-4-0-0–20160406	tx_solr-4-0-0–20160406	2.1	3.0	N/A	1.2	N/A	1.3
5.0	4.10	7.6	tx_solr-4-0-0–20160406	tx_solr-4-0-0–20160406	2.1	3.1	1.0	1.3	N/A	1.3
5.1	4.10	7.6	tx_solr-5-1-0–20160725	tx_solr-4-0-0–20160406	2.1	3.2	1.2	1.3	N/A	1.3
6.0	6.3	7.6	tx_solr-6-0-0–20161209	tx_solr-6-0-0–20161122	2.2	4.0	1.2	1.3	N/A	1.7
6.1	6.3	7.6 - 8.7	tx_solr-6-1-0–20170206	tx_solr-6-1-0–20161220	2.3	4.1	2.0	1.3	N/A	2.0
6.5	6.6.2	7.6 - 8.x	tx_solr-6-5-0–20171023	tx_solr-6-5-0–20171023	2.3	4.1	2.0	1.3	N/A	2.0
7.0	6.3	8.7	tx_solr-7-0-0–20170530	tx_solr-7-0-0–20170530	2.4	4.2	N/A	N/A	N/A	2.0
7.5	6.6.2	8.7	tx_solr-7-5-0–20171023	tx_solr-7-5-0–20171023	2.4	4.2	N/A	N/A	N/A	2.0
8.0	6.6.2	8.7	tx_solr-8-0-0–20171020	tx_solr-8-0-0–20171020	3.0	5.0	N/A	N/A	1.0	2.0

CHAPTER

FIFTEEN

RELEASES

15.1 Apache Solr for TYPO3 7.0.0

We are happy to release EXT:solr 7.0.0. This release brings several smaller and some bigger changes

15.1.1 New in this release

FLUID Templating

One and a half years ago we started to implement FLUID templating for EXT:solr. This project was initially started as the addon solrfluid. Solrfuid was only available for our partners.

With EXT:solr 7.0.0 the new templating is the default templating in EXT:solr. A lot of code was added and several old stuff was removed. Since some things are conceptional different in FLUID and you also have a lot of possibilities we also dropped some parts, that can be build with fluid itself or do not make sence to do them before rendering the result in the view.

Most of the things just work like before but in the following parts we made conceptional changes (for good reasons):

- No css or javascript will be added to the page automatically with the page renderer! Because the integrator wants to have control on that and TYPO3 allows to add this with TypoScript we propose to add these things via typoscript. EXT:solr offers a lot of example typoscript templates e.g. to add the default css or to add the javascript for a range facet.

The following typoscript settings have been removed because they can be implemented with FLUID:

plugin.tx_solr.search.faceting.facetLinkATagParams

You can add them in your project partials. If you need it just for one facet, please overwrite the render partial with facet.partialName and render the attributes different there

plugin.tx_solr.search.faceting.[facetName].facetLinkATagParams

You can add them in your project partials. If you need it just for one facet, please overwrite the render partial with facet.partialName and render the attributes different there

plugin.tx_solr.search.faceting.facets.[facetName].selectingSelectedFacetOptionRemovesFilter

This can be implemented with FLUID logic. Please check the example "Search - (Example) Options with on/off toggle" that implements that (by using the partial Facets/OptionsToggle.html)

plugin.tx_solr.search.results.fieldRenderingInstructions

Please use custom ViewHelpers or the cObject ViewHelper for that.

plugin.tx_solr.search.results.fieldProcessingInstructions

Please use custom ViewHelpers or the cObject ViewHelper for that.

Important: The support of fluid templating would not have been possible without the financial support of all partners! If you want to support us with the implementation of features like this, please think about to join the EB 2017 or 2018. Special thanks also to Frans Saris and beech.it for working on solrfluid together!

- https://github.com/TYPO3-Solr/ext-solr/pull/1308

Backend Modules Restructured

In EXT:solr 7.0.0 the backend modules are structured into multiple backend modules. This makes the user experience in the TYPO3 backend more consistent and allows you, to give different permissions on each module.

When you login into the backend, you now have the following modules available:

- Info: Gives information of your Solr system, index fields and search usage.
- Core Optimization: This module can be used to maintain the synonyms and stopwords in the Apache Solr server.

- Index Queue: Gives an overview on indexed records and can be used to requeue records for indexing.
- Index Administration: This module can be used for administrator tasks on your solr system (clear index, index queue or reload a core)
- https://github.com/TYPO3-Solr/ext-solr/pull/1300

Add excludeValues for Facets

If you want to exclude an optionValue from the facets when they get retrieved, you can configure this now:

```
plugin.tx_solr.search.faceting.facets.colors_stringM.excludeValues = red
```

The example below will exclude the option "red" from the results when it is in the response.

- https://github.com/TYPO3-Solr/ext-solr/pull/1364

Allow to configure custom entry Template

In previous EXT:solr versions it was possible to set a custom entry templating using:

```
plugin.tx_solr.templateFiles.search = EXT:solr/Resources/Templates/PiSearch/search.
↪htm
```

This configuration could be overwritten with a text value in the flexform.

With the move to FLUID we improved this part and made it more editor friendly:

- Since the view related settings are located in the <view> section we've move the template configuration there as well.
- You can now set a Templatename only (e.g. MySearch) to benefit from FLUID fallbacks (while setting a full path is still supported.
- You can configure availableTemplates that can be selected by the editor in the flexform.

The following example shows, how you can load your own partials and provide different entry templates for the editor:

```
plugin.tx_solr {
    view {
        templateRootPaths.100 = EXT:your_config_extension/Resources/Private/
↪Templates/
        partialRootPaths.100 = EXT:your_config_extension/Resources/Private/Partials/
↪
        layoutRootPaths.100 = EXT:your_config_extension/Resources/Private/Layouts/
        templateFiles {
            results = Results
            results.availableTemplates {
                default {
                    label = Default Searchresults Template
                    file = Results
                }
                products {
                    label = Products Template
                    file = ProductResults
                }
            }
        }
    }
}
```

With the prevision configuration the editor can switch from "Default Searchresults Template" to "Products Template".

- https://github.com/TYPO3-Solr/ext-solr/pull/1325
- https://github.com/TYPO3-Solr/ext-solr/pull/1483

Refactoring of Query API

The Query class is one of the biggest classes in EXT:solr and grown over time. This class has now been splitted into several classes. Along with that a concept of "ParameterBuilder" has been introduced. A ParameterBuilder is responsible to build a parameter part of the query. E.g. the Grouping ParameterBuilder is responsible to build all parameters of the solr query for the grouping.

- https://github.com/TYPO3-Solr/ext-solr/pull/1385

Move FilterEncoder and FacetBuilder to Facet Package

In Solrfluid there was one folder for each facet, that contains the facet class and a parser that parsers the solr response into the facet object. The opposite part(parse the url, build the solr query) was previously done in EXT:solr, with a FilterEncoder that was registered in the FacetRendererFactory.

Now because solrfluid and solr have been merged, this logic can also be streamlined. Every facet is now structured in a FacetPackage.

A FacetPackage describes:

- Which parser should be used to parse the solr response
- Which url decoder should be used to parse the EXT:solr query data
- Which query builder should be used to build the faceting query part

You can also implement custom facet types by registering an on FacetPackage with the FacetRegistry.

Migration:

When you have implemented an own FacetParser for solrfluid, you should add a FacetPackage, that references a UrlDecoder and QueryBuilder. If you have used a custom FacetParser without registring a custom facet type in EXT:solr (ApacheSolrForTypo3SolrFacetFacetRendererFactory::registerFacetType) you can just reference DefaultUrlDecorder and DefaultFacerQueryBuilder in your FacetPackage.

- https://github.com/TYPO3-Solr/ext-solr/pull/1319

Custom plugin namespace - Multiple Instances

Before solrfluid was merged there were several parts in EXT:solr where the data was read using GeneralUtility::_GET. The drawback of this approach is that the structure of the urls is hard to change and it is not possible to have custom namespaces for each instance of a plugin.

With solrfuid a SearchRequest object was introduced. This object holds all data from the user request. Now this object is used, whenever data from the user action is read. This allows us to make the request namespace changeable. You can now add your custom plugin namespace to a search plugin instance.

- https://github.com/TYPO3-Solr/ext-solr/pull/1379

Doctrine Migration

As an ongoing task, we started with the migration of database queries to doctrine. Since the database is used in many parts of the extension there are still many parts open. If you want to work on that, your help is very welcome.

- https://github.com/TYPO3-Solr/ext-solr/pull/1259
- https://github.com/TYPO3-Solr/ext-solr/pull/1265
- https://github.com/TYPO3-Solr/ext-solr/pull/1270

- https://github.com/TYPO3-Solr/ext-solr/pull/1271

Add --rootpageid to CLI command

If you want or need to limit the initialization of solr connections to a special rootpage, you can now do this by adding the argument –rootpageid.

- https://github.com/TYPO3-Solr/ext-solr/pull/1305

Respect Setting includeInAvailableFacets and includeInUsedFacets

This setting was not evaluated in EXT:solrfluid before and is now available also with FLUID rendering.

- https://github.com/TYPO3-Solr/ext-solr/pull/1340

Respect requirements facet setting with fluid

This setting was not evaluated in EXT:solrfluid before and is now available also with the FLUID rendering.

- https://github.com/TYPO3-Solr/ext-solr/pull/1401

Respect setting searchUsingSpellCheckerSuggestion with fluid

This setting was not evaluated in EXT:solrfluid before and is now available also with the FLUID rendering.

- https://github.com/TYPO3-Solr/ext-solr/pull/1501

Get rid of dependency to sys_domain record

By now EXT:solr had the dependency on an existing domain record. This can be a problem, when you domain is dynamic or you need to be able to generate it.

Now you can configure a domain by the rootPageId in the TYPO3_CONF_VARS, the domain record is still used, when nothing is configured here.

$GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['sites'][###rootPageId###]['domains'] = ['mydomain.com'];

Note:

There might be an approach to support this in TYPO3 Version 9 by the core and we will adopt this then.

During the implementation of this the logic to retrieve the SiteHash and get the SolrConfiguration was moved to the SiteRepository, this requires an update of the scheduler instances because the scheduler saves a serialized task. Please run the shipped migration to update scheduler tasks created with 6.1.x.

- https://github.com/TYPO3-Solr/ext-solr/pull/1512

Preparations for TYPO3 9

Several things that will be removed with 9 have been changed:

- https://github.com/TYPO3-Solr/ext-solr/pull/1443
- https://github.com/TYPO3-Solr/ext-solr/pull/1452
- https://github.com/TYPO3-Solr/ext-solr/pull/1462

15.1.2 Bugfixes

- Enable zero-configuration use of Docker image: https://github.com/TYPO3-Solr/ext-solr/issues/1278
- Remove unused use statement: https://github.com/TYPO3-Solr/ext-solr/pull/1292
- Indexing record outside siteroot throws exception: https://github.com/TYPO3-Solr/ext-solr/pull/1299
- Mounted pages from outside of the page tree lead to index queue errores: https://github.com/TYPO3-Solr/ext-solr/pull/1294
- Ignore workspace in RootPageResolver: https://github.com/TYPO3-Solr/ext-solr/pull/1298
- Preserve sort order when fetching related records: https://github.com/TYPO3-Solr/ext-solr/pull/1326
- Fix logging of error when devlog is enabled: https://github.com/TYPO3-Solr/ext-solr/pull/1341
- Tracking changes in record from other siteroot is not working as expected https://github.com/TYPO3-Solr/ext-solr/pull/1348
- Relation handler should handle pages overlays correctly https://github.com/TYPO3-Solr/ext-solr/pull/1400

15.1.3 Removed Code

The following code has been removed since it is not used anymore:

Classes:

- ScriptViewHelper
- StyleViewHelper
- AbstractSolrBackendViewHelper
- StringUtility

Methods:

- Util::camelize
- Util::camelCaseToLowerCaseUnderscored
- Util::underscoredToUpperCamelCase
- Util::pageExists

15.1.4 Deprecated Code

Methods:

- Query::setQueryFieldsFromString use setQueryFields(QueryFields::fromString('foo')) with QueryFields instead, will be removed in 8.0
- Query::getQueryFieldsAsString use getQueryFields()->toString() now if needed, will be removed in 8.0
- Query::setQueryField use getQueryFields()->set() now, will be removed in 8.0
- Query::escape Use EscapeService::escape now, when needed
- Query::addReturnField use getReturnFields()->add() now, will be removed in 8.0
- Query::removeReturnField use getReturnFields()->remove() now, will be removed in 8.0
- Query::getFieldList use getReturnFields()->getValues() now, will be removed in 8.0
- Query::setFieldList use setReturnFields() now, will be removed in 8.0
- Query::escapeMarkers not needed anymore, use your own implementation when needed
- Query::setNumberOfGroups use getGrouping()->setNumberOfGroups() instead, will be removed in 8.0

- Query::getNumberOfGroups use getGrouping()->getNumberOfGroups() instead, will be removed in 8.0
- Query::addGroupField use getGrouping()->addField() instead, will be removed in 8.0
- Query::getGroupFields use getGrouping()->getFields() instead, will be removed in 8.0
- Query::addGroupSorting use getGrouping()->addSorting() instead, will be removed in 8.0
- Query::getGroupSortings use getGrouping()->getSortings() instead, will be removed in 8.0
- Query::addGroupQuery use getGrouping()->addQuery() instead, will be removed in 8.0
- Query::getGroupQueries use getGrouping()->getQueries() instead, will be removed in 8.0
- Query::setNumberOfResultsPerGroup use getGrouping()->setResultsPerGroup() instead, will be removed in 8.0
- Query::getNumberOfResultsPerGroup use getGrouping()->getResultsPerGroup() instead, will be removed in 8.0
- Query::setFacetFields use getFaceting()->setFields() instead, will be removed in 8.0
- Query::addFacetField use getFaceting()->addField() instead, will be removed in 8.0
- Query::removeFilter use getFilters()->removeByFieldName() instead, will be removed in 8.0
- Query::removeFilterByKey use getFilters()->removeByName() instead, will be removed in 8.0
- Query::removeFilterByValue use getFilters()->removeByValue() instead, will be removed in 8.0
- Query::addFilter use getFilters()->add() instead, will be removed in 8.0

Method Arguments:

- Query::setGrouping now expects the first argument to be a Grouping object, compatibility for the old argument (bool) will be dropped in 8.0
- Query::setHighlighting now expects the first argument to be a Highlighting object, compatibility for the old arguments (bool, int) will be dropped in 8.0
- Query::setFaceting now expects the first argument to be a Faceting object, compatibility for the old arguments (bool) will be dropped in 8.0

Hooks:

- $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['modifySearchResponse'] has been marked as deprecated and will be dropped in 8.0 please use a SearchResultSetProcessor registered in $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['afterSearch'] as replacement.
- $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['processSearchResponse'] has been marked as deprecated and will be dropped in 8.0 please use a SearchResultSetProcessor registered in $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['afterSearch'] as replacement.

15.1.5 Contributors

Like always this release would not have been possible without the help from our awesome community. Here are the contributors for this release.

(patches, comments, bug reports, reviews, ... in alphabetical order)

- Andreas Lappe
- Felix Eckhofer
- Frans Saris
- Georg Ringer
- Helmut Hummel
- Jonas Ulrich

- Marco Bresch
- Markus Friedrich
- Michael Skrynski
- Rafael Kähm
- Rémy DANIEL
- Sascha Egerer
- Sebastian Hofer
- Timo Hund

Also a big thanks to our partners that have joined the EB2017 program:

- .hausformat
- AGENTUR FRONTAG AG
- amarantus - media design & conding Mario Drengner & Enrico Nemack GbR
- Amedick & Sommer Neue Medien GmbH
- Andrea Pausch
- Animate Agentur für interaktive Medien GmbH
- artig GmbH & Co. KG
- b:dreizehn GmbH
- BIBUS AG Group
- Bitmotion GmbH
- cab services ag
- Causal Sarl
- CHIARI GmbH
- Citkomm services GmbH
- clickstorm GmbH
- Connecta AG
- Creative360
- cron IT GmbH
- CYBERhouse Agentur für interaktive Kommukation GmbH
- cyperfection GmbH
- data-graphis GmbH
- Deutsche Welthungerhilfe e.V.
- Deutscher Ärzteverlag
- Deutscher Volkshochschul-Verband
- Die Medialen GmbH
- die_schnittsteller gmbh
- Dörfer engineering services
- E-Magineurs
- EYE Communications AG
- Fachhochschule für öffentliche Verwaltung NRW Zentralverwaltung Gelsenkirchen

- familie redlich AG
- Fork Unstable Media GmbH
- hauptsache.net GmbH
- Havas Düsseldorf GmbH
- Hirsch & Wölfl GmbH
- Hochschule Furtwangen - IMZ Online Services
- Hochschule Konstanz
- Institut der deutschen Wirtschaft Köln Medien GmbH
- iresults gmbh
- ITK Rheinland
- itl Institut für technische Literatur AG
- jweiland.net
- Kassenärztliche Vereinigung Rheinland-Pfalz
- Kerstin Nägler Web & Social Media Beratung
- Landesinstitut für Schule und Medien Berlin-Brandenburg
- Leibniz Universität IT Services
- Libéo
- Lime Flavour GbR
- LINGNER CONSULTING NEW MEDIA GMBH
- LOUIS INTERNET
- Maximilian Walter
- MEDIA:ESSENZ
- mehrwert intermediäre kommunikation GmbH
- Mercedes-AMG GmbH
- mlm media process management GmbH
- n@work Internet Informationssystems GmbH
- Netcreators
- netz-haut GmbH
- neuwerk interactive
- Nintendo of Europe GmbH
- Onedrop Solutions GmbH
- Open New Media GmbH
- Paints Multimedia GmbG
- pixelcreation GmbH
- plan2net
- Pluswerk AG
- polargold GmbH
- punkt.de GmbH
- Raiffeisen OnLine GmbH

- ruhmesmeile GmbH
- Rundfunk und Telekom Regulierung GmbH
- Schweizer Alpen-Club SAC
- sitegeist media solutions GmbH
- Star Finanz-Software Entwicklung und Vertriebs GmbH
- Stefan Galinski Internetdienstleistungen
- Stratis - Toulon
- Studio Mitte Digital Media GmbH
- Studio 9 GmbH
- Systime A/S
- SYZYGY Deutschland GmbH
- takomat Agentur GbR
- THE BRETTINGHAMS GmbH
- TOUMORO
- Triplesense Reply GmbH
- Typoheads GmbH
- unternehmen online GmbH & Co. KG
- Universität Bremen
- VERDURE Medienteam GmbH
- WACON Internet GmbH
- webedit AG
- Webstore GmbH
- Webtech AG
- wegewerk GmbH
- Wohnungsbau- und Verwaltungsgesellschaft mbH Greifswald
- XIMA MEDIA GmbH
- zdreicom GmbH
- zimmer7 GmbH

Thanks to everyone who helped in creating this release!

15.1.6 Outlook

In the next release we want to focus on the user experience in the backend and in the frontend. As preparation we collected several tasks.

The goal of some of them (e.g. bootstrap templating, checkbox facets, filterable options partial) is to make more things possible out of the box and make the extension more user friendly:

https://github.com/TYPO3-Solr/ext-solr/issues?q=is%3Aissue+is%3Aopen+label%3AUX

If you have allready implemented one this (or something else), that you want to share or make available outofthebox feel free to contanct us!

15.1.7 How to Get Involved

There are many ways to get involved with Apache Solr for TYPO3:

- Submit bug reports and feature requests on [GitHub](https://github.com/TYPO3-Solr/ext-solr)
- Ask or help or answer questions in our [Slack channel](https://typo3.slack.com/messages/ext-solr/)
- Provide patches through Pull Request or review and comment on existing [Pull Requests](https://github.com/TYPO3-Solr/ext-solr/pulls)
- Go to www.typo3-solr.com or call [dkd](http://www.dkd.de) to sponsor the ongoing development of Apache Solr for TYPO3

Support us in 2017 by becoming an EB partner:

http://www.typo3-solr.com/en/contact/

or call:

+49 (0)69 - 2475218 0

15.2 Apache Solr for TYPO3 8.0.0

We are happy to release EXT:solr 8.0.0. The focus of EXT:solr 8.0.0 was, to improve the user experience in the frontend and backend.

15.2.1 New in this release

In the following paragraphs we want to summarize the new features that will be shipped with EXT:solr 8.0.0

New suggest

We've replaced the old jQuery UI based autosuggest with a new suggest (https://github.com/devbridge/jQuery-Autocomplete). The advanced suggest can not only show the suggestions, it can also show a configurable amount of top search results.

When the user clicks on the result, he can directly jump to the result page without opening the search results page.

Thanks:

- Frans Saris and http://www.beech.it for sharing the codebase of the initial patch!

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1638

JSON Faceting for options facets

Apache Solr offers a JSON API for faceting since several versions. Starting with the options facet we've added the support to use this JSON faceting API in EXT:solr.

The support of the JSON API, in general, allows us to build new features on top of that API, that was impossible before. With the first implementation we've added the following features:

By now an option was simply the value and the count, that reflects the number of documents that belong to that option. At EXT:solr 8.0.0 we've added a TypoScript option that is called "metrics", that allows us to collect and show several metrics from documents that belong to a facet option. Examples of metrics are e.g "sum of downloads", "average price",... These metrics will be available in Option model in the FLUID template and can also be used to sort the facet options.

The following example shows an configured options facet with a configured metric:

```
plugin.tx_solr.search.faceting.facets.type.metrics {
    newest = max(created)
    oldest = min(created)
}
```

In the FLUID template you could use the following code in the facet partial to render those metrics:

```
<span>
   newest: {option.metrics.newest -> f:format.date(format: 'Y-m-d H:i:s')}
</span>
<span>
   oldest: {option.metrics.oldest -> f:format.date(format: 'Y-m-d H:i:s')}
</span>
```

Thanks:

- Thanks to Jens Jacobsen and UEBERBIT for sponsoring Jens work on that feature at our code sprint.

Since we'replaced the whole internal communication from EXT:solr to Apache Solr when options facets are used we are very happy to get your feedback and bug reports when you use the options facets with EXT:solr

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1764

Group facet options by prefix

When you have option facets with a lot of options, it would be nice to group those options by a prefix. An example is that you group all options by the starting letter to organize them in tabs:

With EXT:solr 8 we ship the following components that allow grouping your facet options to arrange them as you need them in your template:

- LabelFilterViewHelper: Can be used to filter options based on a prefix of the label.
- LabelPrefixesViewHelper: Can be used to access all available prefixes of the facet options.
- TypoScript example template "(Example) Options grouped by prefix" that configures a grouped facet on the author field

Thanks: This feature was sponsored by https://www.linnearad.no/

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1717

Filterable options facet

In the previous section, the facets get grouped by prefix to organize a large number of options. Another way that you also often see on the web is to allow to filter the options with an additional input box above the facet.

The implementation of that feature is possible just with a partial and a few JavaScript components. To simplify the integration of that feature in a project we ship

- Example FLUID partial that uses the filter for options
- Example JavaScript that implements the filter functionality
- Example TypoScript "Search - (Example) Options filterable by option value" that uses the partials and javascript for a facet

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1741

Default partials with bootstrap.css

The old templating was created with custom CSS that was shipped with the extension. Since we want to decrease the effort that is required to create a mobile search and many integrators use bootstrap.css we decided to ship bootstrap templates by default. If you want to use another framework or your own custom CSS you are still able to do that with custom templates.

Nevertheless, the mobile search in a TYPO3 introduction installation with bootstrap is much better than before and your effort to adopt it should be reduced.

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1738

Performance improvements

In EXT:solr 7.x and below a ping request was done before each search. In EXT:solr 8.0.0 we just catch a failed search and handle the unavailability. This saves up to 30% time because we just need one HTTP request to Apache Solr instead of 2.

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1660

Improved index inspector

In the previous versions, we've introduced own backend modules that can also be used by regular TYPO3 users to perform several tasks. With EXT:solr 8.0.0 the index inspector will be moved from the common info module to our info module:

Besides the move, we also added the functionality to ReQueue a single document from the index inspector when you have permissions on the index queue module.

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1763

Use TYPO3 Guzzle for page index requests

The indexing of pages is now done with the shipped Guzzle client in TYPO3.

Thanks: Thanks to Benni Mack from b13 who has implemented that feature http://www.b13.de/

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1837

SOLR_CLASSIFICATION cObject

When you index a lot of documents you might want to create facets based on patterns that occur in the content.

The cObject SOLR_CLASSIFICATION allows you to do a lightweight classification based on regex patterns that you configure in the index configuration.

The following example shows how SOLR_CLASSIFICATION can be used to map patterns on classes that are indexed into a Solr field that could be used for faceting:

```
plugin.tx_solr.index.queue.pages.businessarea_stringM = SOLR_CLASSIFICATION
plugin.tx_solr.index.queue.pages.businessarea_stringM {
   field = __solr_content
   classes {
      automotive {
         patterns = car,jeep,SUV
         class = automotive
      }
      pharma {
         patterns = pharma,doc,medicine
         class = pharma
      }
   }
}
```

With the configuration above Solr documents get the value "automotive" assigned in the Solr field "businessarea_stringM" when the content contains the term "car", "jeep" or "SUV".

Thanks: Thanks to http://www.bibus.ch who sponsored the implementation of this feature.

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1723

Phrase support (phrase, bigram, trigram)

With plugin.tx_solr.search.query.(phrase/bigramPhrase/trigramPhrase).fields you can control what is passed to Solr with the ps,ps2 and ps3 value.

With these phrase fields, you can boost documents where phrases occur in close proximity. This can be very handy when you want to tune your search in terms of relevancy.

Related links:

- https://lucene.apache.org/solr/guide/66/the-dismax-query-parser.html#TheDisMaxQueryParser-ThepfPhraseFields_Parameter

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1735

Tie parameter support

With plugin.tx_solr.search.query.tieParameter you can now configure the tie value that is passed to Apache Solr.

This value allows you to configure the impact of low scoring fields to the overall score. 0.0 means, that only high score fields will matter, 0.99 means that all fields have the same impact

Related links:

- https://solr.pl/en/2012/02/06/what-can-we-use-dismax-tie-parameter-for/
- https://lucene.apache.org/solr/guide/66/the-dismax-query-parser.html#TheDisMaxQueryParser-ThetieTieBreaker_Parameter

Thanks: Thanks to Marcus Schwemer and in2code that sponsored and shared that feature.

Related pull request: https://github.com/TYPO3-Solr/ext-solr/pull/1690

Doctrine ready

TYPO3 8 introduced Doctrine DBAL for database queries and the old API will be removed in TYPO3 9. Since we've used a lot of repositories with custom SQL queries, we had to rewrite a lot of queries.

In EXT:solr we've used the chance to restructure the SQL related code and move them to repositories whenever this was possible.

With EXT:solr 8 every usage of the old database API is removed and we are prepared in that way to be ready for TYPO3 9.

Fluent API for Queries with the QueryBuilder

Many parts of the code of EXT:solr deal with queries for Apache Solr that's no surprise :). The corresponding parts in the code especially the Query class had grown over time and reached a huge complexity.

This has several drawbacks:

- It is hard to integrate new features (e.g the tiebreaker or bigram features)
- TYPO3 specific logic and common Apache Solr logic is mixed and makes it hard to switch to frameworks like e.g. Solarium
- The Query class does multiple things: Build the query, initialize the query from the configuration,... This could be split into multiple components.

To get better in that regards our goal is to split the Query into:

- Query: Aggregate that is responsible to build the Solr query string based on the options
- QueryBuilder: Builder class that is responsible to build an initialized Query object e.g. based on TypoScript configuration and user input.

With the current state the QueryBuilder does the following to build a Query from the user input:

```
$query = $queryBuilder->newSearchQuery($rawQuery)
 ->useResultsPerPage($resultsPerPage)
 ->useReturnFieldsFromTypoScript()
 ->useQueryFieldsFromTypoScript()
 ->useInitialQueryFromTypoScript()
 ->useFiltersFromTypoScript()
 ->useFacetingFromTypoScript()
 ->useVariantsFromTypoScript()
 ->useGroupingFromTypoScript()
 ->useHighlightingFromTypoScript()
 ->usePhraseFieldsFromTypoScript()
 ->useBigramPhraseFieldsFromTypoScript()
```

```
->useTrigramPhraseFieldsFromTypoScript()
->getQuery();
```

Finally, this allows us to:

- Integrate new features faster
- Allow devs to compose own queries that use or ignore several aspects of EXT:solr
- Simplify the switch or integration of a generic Solr API that is independent of TYPO3 (e.g. Solarium)

On the way to TYPO3 9

With EXT:solr 8.0.0 we will not officially support TYPO3 9 since it is not an LTS release! Nevertheless, we want to stay close to the TYPO3 core and allow the usage in 9 already.

By now we mainly fix Doctrine and Composer related issues and support the dropped "pageslanguageoverlay" table.

So to sum up... EXT:solr 8.0.0 will mainly support TYPO3 8 LTS and we will support TYPO3 9.x a good as we can without losing the backward compatibility to TYPO3 8 LTS.

15.2.2 Bugfixes

- Can not set the facet sorting to count when global sorting is set to index: https://github.com/TYPO3-Solr/ext-solr/pull/1667
- Filter with Flexform in backend does not work when value contains whitespaces: https://github.com/TYPO3-Solr/ext-solr/issues/1742
- SOLR_RELATION does not recognize sys_categories for translated pages: https://github.com/TYPO3-Solr/ext-solr/issues/1812
- Allow to use EXT:solr with sql strict mode: https://github.com/TYPO3-Solr/ext-solr/issues/1785
- Missing array keys in facet options after manual sorting: https://github.com/TYPO3-Solr/ext-solr/pull/1712
- partialName is missing in TypoScript reference: https://github.com/TYPO3-Solr/ext-solr/pull/1730

15.2.3 Removed Code

Query Refactoring

In the long run we want to be able to use other PHP frameworks for Apache Solr e.g. solarium(http://www.solarium-project.org/). To make this possible, we need to split the pure Solr query related logic from the TYPO3Solr specific query logic (e.g. accessFilter,...). To get a step closer into this direction, we've extracted the logic that is required to build a TYPO3 specific Solr query into the QueryBuilder. The pure Solr related query logic remains in the Query class.

Impact:

- Whenever you create or modify queries you should use the QueryBuilder class for that. In one of the next releases we will support to create solarium queries with this QueryBuilder.

Beside the query refactoring, that required to remove and change several methods, the following code has been removed:

Hooks:

- $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['modifySearchResponse'] has been marked as deprecated and will be dropped in 8.0 please use a SearchResultSetProcessor registered in $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['afterSearch'] as replacement.

- $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['processSearchResponse'] has been marked as deprecated and will be dropped in 8.0 please use a SearchResultSetProcessor registered in $GLOBALS['TYPO3_CONF_VARS']['EXTCONF']['solr']['afterSearch'] as replacement.

15.2.4 Deprecated Code

The following methods have been marked as deprecated and will be removed in EXT:solr 9.0.0

- ApacheSolrForTypo3SolrSearch::getResultDocumentsRaw - Use the SearchResultsSet::getSearchResults now
- ApacheSolrForTypo3SolrSearch::getResultDocumentsEscaped - Use the SearchResultsSet::getSearchResults now
- ApacheSolrForTypo3SolrSearch::getFacetCounts - Use the SearchResultSet::getFacets now
- ApacheSolrForTypo3SolrSearch::getFacetFieldOptions - Use the SearchResultSet::getFacets now
- ApacheSolrForTypo3SolrSearch::getFacetQueryOptions - Use the SearchResultSet::getFacets now
- ApacheSolrForTypo3SolrSearch::getFacetRangeOptions - Use the SearchResultSet::getFacets now
- ApacheSolrForTypo3SolrSearch::getSpellcheckingSuggestions - Use SearchResultSet::getSpellcheckingSuggestions
- ApacheSolrForTypo3SolrQuery is deprecated, use ApacheSolrForTypo3SolrDomainSearchQueryQuery now
- ApacheSolrForTypo3SolrSuggestQuery is deprecated, use ApacheSolrForTypo3SolrDomainSearchQuerySuggestQuery now

15.2.5 Contributors

Like always this release would not have been possible without the help from our awesome community. Here are the contributors for this release.

(patches, comments, bug reports, reviews, ... in alphabetical order)

- Andreas Lappe
- Andri Steiner
- Benni Mack
- Daniel Diesenreither
- Daniel Mann
- Daniel Ruf
- Georg Ringer
- Hannes Lau
- Jeffrey Nellissen
- Jens Jacobsen
- Marco Bresch
- Marcus Schwemer
- Markus Friedrich
- Markus Kobligk
- Markus Sommer
- Nicole Cordes
- Patrick Schriner

- 16. Golmann
- Rafael Kähm
- Sascha Egerer
- Simon Schmidt
- Thomas Löffler
- Timo Hund
- Tomas Norre Mikkelsen

Also a big thanks to our partners that have joined the EB2018 program:

- Albervanderveen
- Amedick & Sommer
- AUSY SA
- bgm Websolutions GmbH
- Citkomm services GmbH
- Consulting Piezunka und Schamoni - Information Technologies GmbH
- Cows Online GmbH
- food media Frank Wörner
- FTI Touristik GmbH
- Hirsch & Wölfl GmbH
- Hochschule Furtwangen
- JUNGMUT Communications GmbH
- Kreis Coesfeld
- LOUIS INTERNET GmbH
- L.N. Schaffrath DigitalMedien GmbH
- Mercedes AMG GmbH
- Petz & Co
- Pluswerk AG
- ressourcenmangel an der panke GmbH
- Site'nGo
- Studio B12 GmbH
- systime
- Talleux & Zöllner GbR
- TOUMORO
- TWT Interactive GmbH

Special thanks to our premium EB 2018 partners:

- b13 http://www.b13.de/
- dkd http://www.dkd.de/
- jweiland.net http://www.jweiland.net/

Thanks to everyone who helped in creating this release!

15.2.6 Outlook

In the next release we want to focus on the move to solarium and the support of the lastest Apache Solr version.

15.2.7 How to Get Involved

There are many ways to get involved with Apache Solr for TYPO3:

- Submit bug reports and feature requests on [GitHub](https://github.com/TYPO3-Solr/ext-solr)
- Ask or help or answer questions in our [Slack channel](https://typo3.slack.com/messages/ext-solr/)
- Provide patches through Pull Request or review and comment on existing [Pull Requests](https://github.com/TYPO3-Solr/ext-solr/pulls)
- Go to www.typo3-solr.com or call [dkd](http://www.dkd.de) to sponsor the ongoing development of Apache Solr for TYPO3

Support us in 2017 by becoming an EB partner:

http://www.typo3-solr.com/en/contact/

or call:

+49 (0)69 - 2475218 0