Solr Bundle

For use with eZ Publish 5.4, go to the corresponding documentation page which covers the v1.0 version of the bundle compatible with eZ Publish 5.4.

What is Solr Search Engine Bundle?

ezplatform-solr-search-engine,as the package is called, aims to be a transparent drop-in replacement for the SQL-based "Legacy" search engine powering eZ Platform Search API by default. When you enable Solr and re-index your content, all your existing Search queries using SearchService will be powered by Solr automatically. This allows you to scale up your eZ Platform installation and be able to continue development locally against SQL engine, and have a test infrastructure, Staging and Prod powered by Solr. This removes considerable load from your database. *See Architecture for further information on the architecture of eZ Platform.*

Status of features:

- ✓ Able to handle all eZ Platform queries
  - ✓ Much more suitable for handling field criteria *(performance)*
  - ✓ Scoring for content queries and sorting by them by default
- ✓ Indexing plugins *(Solr Bundle >= v1.2)*
- ✓ Solr 6 support *(Solr Bundle >= v1.3)*
  - ✓ Scoring for Location queries and sorting by them by default

- Work in progress:
  - ☐ Faceting *(possible to write your own since v1.0, ContentType/Section/User implemented since v1.4, suggested further changes to the API for Faceting can be found here)*
  - ✓ Index time Boosting *(Solr Bundle >= v1.4)*
- Future:
  - ☐ Solr cloud support
  - ☐ Highlighting
  - ☐ Spell checking
  - ☐ Query time Boosting

How to set up Solr Search engine

Step 0: Enable Solr Bundle

Not needed with eZ Platform
This step is not needed as of eZ Platform 15.09, however it is kept here for reference in case you have previously disabled the bundle.

1. Check in composer.json if you have the `ezsystems/ezplatform-solr-search-engine` package, if not add/update composer dependencies:

   command line

   ```
   composer require --no-update
   ezsystems/ezplatform-solr-search-engine:~1.0
   composer update
   ```

2. Make sure `EzPublishSolrSearchEngineBundle` is activated with the following line in

In this topic:

Related:

- Browsing, Finding/Searching, viewing content using the API the `app/AppKernel.php` file: new `EzSystems\EzPlatformSolrSearchEngineBundle\EzSystemsEzPlatformSolrSearchEngineBundle()`

Step 1: Configuring & Starting Solr

The example presents a configuration with single core, look to Solr documentation for configuring Solr in other ways, including examples.

Download and configure

Solr 4.10.4

First, download and extract Solr. Solr Bundle 1.x supports Solr 4.10.4:

- solr-4.10.4.tgz or solr-4.10.4.zip

Secondly, copy configuration files needed for eZ Solr Search Engine bundle, in the example below from the root of your project to the place you extracted Solr:

Command line example

```
# Make sure to replace the /opt/solr/ path with where you have placed Solr
cp -R vendor/ezsystems/ezplatform-solr-search-engine/lib/Resources/config/solr/* /opt/solr/example/solr/collection1/conf/

/opt/solr/bin/solr start -f
```

Solr 6

SOLR BUNDLE >= 1.3.0 First download and extract Solr, in Solr Bundle 1.3 and higher we also support Solr 6 *(currently tested with Solr 6.4.2)*:

- solr-6.4.2.tgz or solr-6.4.2.zip

Secondly, copy configuration files needed for eZ Solr Search Engine bundle, *here from the root of your project to the place you extracted Solr*:

Command line example

```
# Make sure to replace the /opt/solr/ path with where
you have placed Solr
mkdir -p /opt/solr/server/ez/template
cp -R
vendor/ezsystems/ezplatform-solr-search-engine/lib/Resou
rces/config/solr/* /opt/solr/server/ez/template
cp
/opt/solr/server/solr/configsets/basic_configs/conf/{cur
rency.xml,solrconfig.xml,stopwords.txt,synonyms.txt,elev
ate.xml} /opt/solr/server/ez/template
cp /opt/solr/server/solr/solr.xml /opt/solr/server/ez

# Modify solrconfig.xml to remove the section that
doesn't agree with your schema
sed -i.bak '/<updateRequestProcessorChain
name="add-unknown-fields-to-the-schema">/,/<\/updateRequ
estProcessorChain>/d'
/opt/solr/server/ez/template/solrconfig.xml

# Start Solr (but apply autocommit settings below first
if you need to)
/opt/solr/bin/solr -s ez
/opt/solr/bin/solr create_core -c collection1 -d
server/ez/template
```

Further configuration

Thirdly, on both Solr 4 and 6 Solr the bundle does not commit solr index changes directly on repository updates, leaving it up to you to tune this using `solrconfig.xml` as best practice suggests, for example:

solrconfig.xml

```
<autoCommit>
  <!-- autoCommit is here left as-is like it is out of
the box in Solr, this controls hard commits for
durability/replication -->
  <maxTime>${solr.autoCommit.maxTime:15000}</maxTime>
  <openSearcher>false</openSearcher>
</autoCommit>

<autoSoftCommit>
  <!-- Soft commits controls mainly when changes becomes
visible, by default we change value from -1 (disabled)
to 100ms, to try to strike a balance between Solr
performance and staleness of HttpCache generated by Solr
queries -->
  <maxTime>${solr.autoSoftCommit.maxTime:100}</maxTime>
</autoSoftCommit>
```

Step 2: Configuring bundle

The Solr search engine bundle can be configured in many ways. The config further below assumes you have parameters set up for solr dsn and search engine *(however both are optional)*, for example:

parameters.yml

```
search_engine: solr
    solr_dsn: 'http://localhost:8983/solr'
```

On to configuring the bundle.

Single Core example (default)

Out of the box in eZ Platform the following is enabled for a simple setup:

config.yml

```
ez_search_engine_solr:
    endpoints:
        endpoint0:
            dsn: %solr_dsn%
            core: collection1
    connections:
        default:
            entry_endpoints:
                - endpoint0
            mapping:
                default: endpoint0
```

Shared Core example

In the following example we have decided to separate one language as the installation contains several similar languages, and one very different language that should receive proper language analysis for proper stemming and sorting behavior by Solr:

config.yml

```
ez_search_engine_solr:
    endpoints:
        endpoint0:
            dsn: %solr_dsn%
            core: core0
        endpoint1:
            dsn: %solr_dsn%
            core: core1
    connections:
        default:
            entry_endpoints:
                - endpoint0
                - endpoint1
            mapping:
                translations:
                    jpn-JP: endpoint1
                # Other languages, for instance eng-US and other
western languages are sharing core
                default: endpoint0
```

Multi Core example

If full language analysis features are preferred, then each language can be configured to separate cores.

Note: Please make sure to test this setup against single core setup, as it might perform worse than single core if your project uses a lot of language fallbacks per SiteAccess, as queries will then be performed across several cores at once.

config.yml

```
ez_search_engine_solr:
    endpoints:
        endpoint0:
            dsn: %solr_dsn%
            core: core0
        endpoint1:
            dsn: %solr_dsn%
            core: core1
        endpoint2:
            dsn: %solr_dsn%
            core: core2
        endpoint3:
            dsn: %solr_dsn%
            core: core3
        endpoint4:
            dsn: %solr_dsn%
            core: core4
        endpoint5:
            dsn: %solr_dsn%
            core: core5
        endpoint6:
            dsn: %solr_dsn%
            core: core6
    connections:
        default:
            entry_endpoints:
                - endpoint0
                - endpoint1
                - endpoint2
                - endpoint3
                - endpoint4
                - endpoint5
                - endpoint6
            mapping:
                translations:
                    - jpn-JP: endpoint1
                    - eng-US: endpoint2
                    - fre-FR: endpoint3
                    - ger-DE: endpoint4
                    - esp-ES: endpoint5
                # Not really used, but specified here for
fallback if more languages are suddenly added by content admins
                default: endpoint0
                # Also use separate core for main languages
(differs from content object to content object)
                # This is useful to reduce number of cores
queried for always available language fallbacks
                main_translations: endpoint6
```

Step 3: Configuring repository with the specific search engine

The following is an example of configuring Solr Search Engine, where `connection` name is same as in the example above, and engine is set to `solr`:

ezplatform.yml

```
ezpublish:
    repositories:
        default:
            storage: ~
            search:
                engine: %search_engine%
                connection: default
```

`%search_engine%` is a parameter that is configured in `app/config/parameters.yml`, and should be changed from its default value "`legacy`" to "`solr`" to activate Solr as the Search engine.

Step 4: Clear prod cache

While Symfony `dev` environment keeps track of changes to yml files, `prod` does not, so to make sure Symfony reads the new config we clear cache:

```
php app/console --env=prod cache:clear
```

Step 5: Run CLI indexing command

Make sure to configure your setup for indexing
Some exceptions might happen on indexing if you have not configured your setup correctly, here are the most common issues you may encounter:

- Exception if Binary files in database have an invalid path prefix
    - Make sure `ezplatform.yml` configuration `var_dir` is configured properly.
    - If your database is inconsistent in regards to file paths, try to update entries to be correct *(but make sure to make a backup first)*.
- Exception on unsupported Field Types
    - Make sure to implement all Field Types in your installation, or to configure missing ones as NullType if implementation is not needed.
- Content not immediately available
    - Solr Bundle on purpose does not commit changes directly on Repository updates *(on indexing)*, but lets you control this using Solr configuration. Adjust Solr **autoSoftCommit** *visibility of change to search index)* and/or **autoCommit** *(hard commit, for durability and replication)* to balance performance and load on your Solr instance against needs you have for "NRT".
- Running out of memory during indexing
    - In general make sure to run indexing using prod environment to avoid debuggers and loggers from filing up memory.
    - Stash: Disable in_memory cache as recommended on Persistence cache for long running scripts.
    - Flysystem: You can find further info in: https://jira.ez.no/browse/EZP-25325.

The last step is to execute the initial indexation of data:

```
php app/console --env=prod --siteaccess=<name>
ezplatform:solr_create_index
```

SOLR BUNDLE >= 1.2 Since eZ Platform v1.7.0 the `ezplatform:solr_create_index` comm and is deprecated, use `php app/console ezplatform:reindex` instead:

```
php app/console --env=prod --siteaccess=<name>
ezplatform:reindex
```

Configuring the Solr Search engine Bundle

For configuration of Solr connection for your repository, see How to set up Solr Search engine above.

Boost configuration

SOLR BUNDLE >= 1.4, ETA JUNE 2017

Boosting currently happens when indexing, so if you change your configuration you'll need to re-index (this is expected behavior). This can possibly be solved by a contribution to change boosting to be performed on query time.

Boosting tells the search engine which parts of the content model have more importance when searching, and is an important part of tuning your search results relevance. Importance is defined using a numeric value, where `1.0` is default, values higher than that are more important, and values lower (down to `0.0`) are less important.

Boosting is configured per connection that you configure to use for a given repository, like in the example below:

config.yml snippet example

```
ez_search_engine_solr:
    connections:
        default:
            boost_factors:
                content_type:
                    # Boost a whole Content Type
                    article: 2.0
                field_definition:
     # Boost a content Field system wide, or for a given
Content Type
                    title: 3.0
                    blog_post:
                        # Don't boost title of blog
posts that high, but still higher than default
                        title: 1.5
                meta_field:
     # Boost a meta Field (name, text) system wide, or
for a given Content Type
                    name: 10.0
                    article:
                        # Boost the meta full text Field
for article more than 2.0 set above
                        text: 5.0
```

The configuration above will result in the following boosting (Content Type / Field):

- `article/title:` `2.0`
- `news/title:` `3.0`
- `blog_post/title: 1.5`
- `news/description: 1.0` (default)
- `article/text` *`(meta)`*`:` `5.0`
- `blog_post/name` *`(meta)`*`: 10.0`
- `article/name` *`(meta)`*`: 2.0`

Extending the Solr Search engine Bundle

Document Field Mappers

SOLR BUNDLE >= 1.2 Starting with eZ Platform 1.7: as a developer you will often find the need to index some additional data in the search engine. The use cases for this are varied, for example the data could come from an external source *(e.g. from recommendation system)*, or from an internal source. The common use case for the latter is indexing data through the Location hierarchy, for example from the parent Location to the child Location, or in the opposite direction, indexing child data on the parent Location. The reason might be you want to find the content with fulltext search, or you want to simplify search for a complicated data model. To do this effectively, you first need to understand how the data is indexed with Solr Search engine. Documents are indexed per translation, as Content blocks. In Solr, a block is a nested document structure. In our case, parent document represents Content, and Locations are indexed as child documents of the Content. To avoid duplication, full text data is indexed on the Content document only. Knowing this, you have the option to index additional data on:

- all block documents (meaning Content and its Locations, all translations)
- all block documents per translation
- Content documents
- Content documents per translation
- Location documents

Indexing additional data is done by implementing a document field mapper and registering it at one of the five extension points described above. You can create the field mapper class anywhere inside your bundle, as long as when you register it as a service, the "class" parameter" in your `services.yml` matches the correct path. We have three different field mappers. Each mapper implements two methods, by the same name, but accepting different arguments:

- `ContentFieldMapper`
  - `::accept(Content $content)`
  - `::mapFields(Content $content)`
- `ContentTranslationFieldMapper`
  - `::accept(Content $content, $languageCode)`
  - `::mapFields(Content $content, $languageCode)`
- `LocationFieldMapper`
  - `::accept(Location $content)`
  - `::mapFields(Location $content)`

These can be used on the extension points by registering them with the container using service tags, as follows:

- all block documents
  - `ContentFieldMapper`
  - `ezpublish.search.solr.document_field_mapper.block`
- all block documents per translation
  - `ContentTranslationFieldMapper`
  - `ezpublish.search.solr.field_mapper.block_translation`
- Content documents
  - `ContentFieldMapper`
  - `ezpublish.search.solr.document_field_mapper.content`
- Content documents per translation
  - `ContentTranslationFieldMapper`
  - `ezpublish.search.solr.field_mapper.content_translation`
- Location documents
  - `LocationFieldMapper`
  - `ezpublish.search.solr.field_mapper.location`

The following example shows how to index data from the parent Location content, in order to make it available for full text search on the children content. A concrete use case could be indexing webinar data on the webinar events, which are children of the webinar. Field mapper could then look something like this:

```
 <?php

namespace My\WebinarApp;

use
EzSystems\EzPlatformSolrSearchEngine\FieldMapper\Content
FieldMapper;
use eZ\Publish\SPI\Persistence\Content\Handler as
ContentHandler;
use eZ\Publish\SPI\Persistence\Content\Location\Handler
as LocationHandler;
use eZ\Publish\SPI\Persistence\Content;
use eZ\Publish\SPI\Search;

class WebinarEventTitleFulltextFieldMapper extends
ContentFieldMapper
{
    /**
     * @var
\eZ\Publish\SPI\Persistence\Content\Type\Handler
     */
    protected $contentHandler;

    /**
     * @var
\eZ\Publish\SPI\Persistence\Content\Location\Handler
     */
    protected $locationHandler;

    /**
     * @param
\eZ\Publish\SPI\Persistence\Content\Handler
$contentHandler
     * @param
\eZ\Publish\SPI\Persistence\Content\Location\Handler
$locationHandler
     */
    public function __construct(
        ContentHandler $contentHandler,
        LocationHandler $locationHandler
    ) {
        $this->contentHandler = $contentHandler;
        $this->locationHandler = $locationHandler;
    }

    public function accept(Content $content)
    {
        // ContentType with ID 42 is webinar event
        return
$content->versionInfo->contentInfo->contentTypeId == 42;
    }

    public function mapFields(Content $content)
    {
        $mainLocationId =
$content->versionInfo->contentInfo->mainLocationId;
        $location =
$this->locationHandler->load($mainLocationId);
        $parentLocation =
$this->locationHandler->load($location->parentId);
        $parentContentInfo =
$this->contentHandler->loadContentInfo($parentLocation->
contentId);

        return [
            new Search\Field(
                'fulltext',
                $parentContentInfo->name,
                new Search\FieldType\FullTextField()
            ),
        ];
    }
}
```

Since we index full text data only on the Content document, you would register the service like this:

```
my_webinar_app.webinar_event_title_fulltext_field_mapper
:
    class:
My\WebinarApp\WebinarEventTitleFulltextFieldMapper
    arguments:
        - '@ezpublish.spi.persistence.content_handler'
        - '@ezpublish.spi.persistence.location_handler'
    tags:
        - {name:
ezpublish.search.solr.field_mapper.content}
```

Providing feedback

After completing the installation you are now free to use your site as usual. If you get any exceptions for missing features, have feedback on performance, or want to discuss, join our community slack channel at https://ezcommunity.slack.com/messages/ezplatform-use/