Solr Search Engine Bundle

5.4.X >= 5.4.5

Support for Solr Search Engine Bundle when used in eZ Publish 5.4 with legacy is limited, as the legacy integration is done by Netgen's ezplatformsearch, a third party community-supported extension.

For instructions on the bundle for eZ Platform, see Solr Search Engine Bundle in the eZ Platform Developer documentation.

Version 1.0.x of Solr Bundle primarily aims to be drop in replacement for Legacy (SQL based) search engine for better scalability and performance *(mainly with field criteria and sort clauses).* And while it also provides better full text search thanks to Solr, more advance search features like Faceting, Spellchecking, plugin system, .. will come in later releases, most for for newer versions of eZ Platform only. See 5.4.5 Release Notes for further info.

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 Search API by default. By enabling Solr and re-indexing your content, all your exising Search queries using SearchService, will be powered by Solr automatically. This allows you to scale up your eZ Publish Platform installation and be able to continue development locally against SQL engine, and have test infrastructure, Staging and Prod powered by Solr. Thus remove considerable load from your database so it can focus on more important things, like publishing 😉.

Se Architecture page for further information on the architecture of eZ Publish Platform and eZ Platform.

How to setup Solr Search engine

Step 1: Enabling Bundle

In this step you'll enable the Solr Search Engine Bundle which handles indexing on all updates to Platform stack API, and to handle all search queries going to Search Service.

command line

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

2. Activate `EzPublishSolrSearchEngineBundle` by adding the following lines to your `ezpublish/EzPublishKernel.php` file:

```
new EzSystems\EzPlatformSolrSearchEngineBundle\EzSystemsEzPlatformS
olrSearchEngineBundle()
```

Step 2: Enabling Legacy extension

Being on 5.x, you'll need to also make sure searches in legacy *(like in admin interface)* is using the new solr search engine, this functionality is provided by ezplatformsearch extension made by NetGen.

1. Add/Update composer dependencies:

   command line

   ```
   composer require netgen/ezplatformsearch:~1.1
   ```

2. Activate extension in `site.ini`, typically `ezpublish_legacy/settings/override/site.ini.append.php`

   site.ini

   ```
   [ExtensionSettings]
   ActiveExtensions[]=ezplatformsearch

   # Also make sure to comment out or remove any
   occurrence of other search engines like ezfind
   #ActiveExtensions[]=ezfind
   ```

Step 3: Configuring & Starting Solr

Example here is for single core, look to Solr documentation for configuring Solr in other ways, also see the provided configuration for some examples.

First download and extract Solr, **we currently support 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, *here from root of your project to the place you extracted Solr*:

Command line example

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

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

Thirdly, Solr 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, example config:

solrconfig.xml

```
<autoCommit>
  <!-- autoCommit is here left as-is like it is out of
the box in Solr 4.10.4, 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 4: Configuring bundle

The Solr search engine bundle can be configured many ways, here are some examples

Before further configuration, make sure that the file `parameters.yml` contains line:

parameters.yml

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

Single Core example (default)

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

ezpublish.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 install contains several similar languages, and one very different language that should be recive proper language analysis for proper stemming and sorting behavior by Solr:

ezpublish.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 as it might perform worse then single core if your project uses a lot for language fallbacks per SiteAccess as queries will then be performed across several cores at once.

```
ezpublish.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 5: Configuring repository with the specific search engine

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

ezpublish.yml
```
doctrine:
   dbal:
       connections:
           my_repository_connection:
               #...
 ezpublish:
   #...
   repositories:
       my_repository:
           engine: legacy
           connection: my_repository_connection
           search:
               engine: solr # One of legacy (default) or solr
               connection: default # If legacy same as storage
applies, if solr use same as you defined in step 3
#...
ez_search_engine_solr:
   endpoints:
       endpoint0:
           dsn: %solr_dsn%
           core: collection1
   connections:
       default:
           entry_endpoints:
               - endpoint0
           mapping:
               default: endpoint0
```

Config in 5.4.5 and eZ Platform
In eZ Publish 5.4.5 when installed via composer create-project, the example above is in default config with one small difference *(same as on eZ Platform)*. There `search.engine` is set to `%search_engine%`, which is a parameter that needs to be set in parameters.yml, and for use with Solr it will needs to be set to `"solr"` as described in parameters.yml.dist, and as described above.

Step 6: 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 ezpublish/console --env=prod cache:clear
```

Step 7: Run CLI indexing command

Last step is to execute initial indexation of data:

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

Known issues
If you have issues using the indexing command there are some known issues and how you resolve them listed in the 5.4.5 Release Notes.

Providing feedback

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