Search

Introduction

eZ Platform exposes very powerful Search API, allowing both Full Text search and querying the content repository using several built-in Criteria and Sort Clauses. These are supported across several search engines, allowing you to plug in another search engine without changing your code.

Available Search Engines

Currently 3 search engines exists on their own eZ Platform Bundles:

1. Legacy, a database-powered search engines for basic needs.
2. Solr, an integration providing better overall performance, much better scalability and support for more advance search capabilities **RECOMMENDED**
3. ElasticSearch, similar to Solr integration, however currently not under active development **EXPERIMENTAL, NOT SUPPORTED**

Usage

Search Criteria and Sort Clauses

Search Criteria and Sort Clauses are `value` object classes used for building Search Query, to define filter criteria and ordering of the result set. eZ Platform provides a number of standard Criteria and Sort Clauses that you can use out of the box and that should cover the majority of use cases.

Example of standard ContentId criterion

```
<?php

namespace
eZ\Publish\API\Repository\Values\Content\Query\Criterion
;

use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
\Operator\Specifications;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
Interface;

/**
 * A criterion that matches content based on its id
 *
 * Supported operators:
 * - IN: will match from a list of ContentId
 * - EQ: will match against one ContentId
 */
class ContentId extends Criterion implements
```

In this topic:

- Introduction
  - Available Search Engines
- Usage
  - Search Criteria and Sort Clauses
    - Search Engine Handling of Search Criteria and Sort Clauses
    - Custom Criteria and Sort Clauses
    - Difference between Content and Location Search
    - How to configure your own Criterion and Sort Clause Handlers
    - Criteria Independence Example
  - Reindexing
- Reference
  - Search Criteria Reference
  - Sort Clauses Reference

Related topics:

Legacy Search Engine Bundle

Solr Bundle

ElasticSearch Bundle

```
CriterionInterface
{
    /**
     * Creates a new ContentId criterion
     *
     * @param int|int[] $value One or more content Id
that must be matched.
     *
     * @throws \InvalidArgumentException if a non
numeric id is given
     * @throws \InvalidArgumentException if the value
type doesn't match the operator
     */
    public function __construct( $value )
    {
        parent::__construct( null, null, $value );
    }

    public function getSpecifications()
    {
        $types = Specifications::TYPE_INTEGER |
Specifications::TYPE_STRING;
        return array(
            new Specifications( Operator::IN,
Specifications::FORMAT_ARRAY, $types ),
            new Specifications( Operator::EQ,
Specifications::FORMAT_SINGLE, $types ),
        );
    }

    public static function createFromQueryBuilder(
$target, $operator, $value )
    {
```

```
        return new self( $value );
    }
}
```

Search Engine Handling of Search Criteria and Sort Clauses

As Search Criteria and Sort Clauses are `value` objects which are used to define the Query from API perspective, they are are common for all storage engines. Each storage engine needs to implement its own `handler` for corresponding Criterion and Sort Clause `value` object, which will be used to translate the value object into storage specific search query.

Example of ContentId criterion handler in Legacy Storage Engine

```
<?php

namespace
eZ\Publish\Core\Search\Legacy\Content\Common\Gateway\Cri
terionHandler;

use
eZ\Publish\Core\Search\Legacy\Content\Common\Gateway\Cri
terionHandler;
use
eZ\Publish\Core\Search\Legacy\Content\Common\Gateway\Cri
teriaConverter;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
;
use eZ\Publish\Core\Persistence\Database\SelectQuery;

/**
 * Content ID criterion handler
 */
class ContentId extends CriterionHandler
{
    /**
     * Check if this criterion handler accepts to handle
the given criterion.
     *
     * @param
\eZ\Publish\API\Repository\Values\Content\Query\Criterio
n $criterion
     *
     * @return boolean
     */
    public function accept( Criterion $criterion )
    {
        return $criterion instanceof
Criterion\ContentId;
    }


    /**
     * Generate query expression for a Criterion this
```

```
handler accepts
     *
     * accept() must be called before calling this
method.
     *
     * @param
\eZ\Publish\Core\Search\Legacy\Content\Common\Gateway\Cr
iteriaConverter $converter
     * @param
\eZ\Publish\Core\Persistence\Database\SelectQuery $query
     * @param
\eZ\Publish\API\Repository\Values\Content\Query\Criterio
n $criterion
     *
     * @return
\eZ\Publish\Core\Persistence\Database\Expression
     */
    public function handle( CriteriaConverter
$converter, SelectQuery $query, Criterion $criterion )
    {
        return $query->expr->in(
            $this->dbHandler->quoteColumn( "id",
"ezcontentobject" ),
            $criterion->value
```

```
        );
    }
}
```

Example of ContentId criterion handler in Solr Storage engine

```
<?php

namespace
EzSystems\EzPlatformSolrSearchEngine\Query\Content\Crite
rionVisitor;

use
EzSystems\EzPlatformSolrSearchEngine\Query\CriterionVisi
tor;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
\Operator;

/**
 * Visits the ContentId criterion
 */
class ContentIdIn extends CriterionVisitor
{
    /**
     * CHeck if visitor is applicable to current
criterion
     *
     * @param Criterion $criterion
     *
     * @return boolean
     */
    public function canVisit( Criterion $criterion )
    {
        return
            $criterion instanceof Criterion\ContentId &&
            ( ( $criterion->operator ?: Operator::IN )
=== Operator::IN ||
              $criterion->operator === Operator::EQ );
    }


    /**
     * Map field value to a proper Solr representation
     *
     * @param Criterion $criterion
     * @param CriterionVisitor $subVisitor
     *
     * @return string
     */
    public function visit( Criterion $criterion,
```

```
CriterionVisitor $subVisitor = null )
    {
        return '(' .
            implode(
                ' OR ',
                array_map(
                    function ( $value )
                    {
                        return 'id:"' . $value . '"';
                    },
                    $criterion->value
                )
            ) .
```

```
            ')';
    }
}
```

Custom Criteria and Sort Clauses

Sometimes you will find that standard Search Criteria and Sort Clauses provided with eZ Publish are not sufficient for you needs. Most often this will be the case if you have developed a custom Fie ld Type using external storage, which therefore can not be searched using standard Field Criterion.

> **On use of Field Criterion/SortClause with large databases**
> Field Criterion/SortClause does not perform well by design when using SQL database, so if you have a large database and want to use them you either need to use Solr search engine, or develop your own Custom Criterion / Sort Clause to avoid use of attributes (Fields) database table, and instead uses a custom simplified table which can handle the amount of data you have.

In this case you can implement a custom Criterion or Sort Clause, together with the corresponding handlers for the storage engine you are using.

Difference between Content and Location Search

These are two basic types of searches, you can either search for Locations or for Content. Each has dedicated methods in Search Service:

Type of search	Method in Search Service
Content	`findContent()`
Content	`findSingle()`
Location	`findLocations()`

All Criteria and Sort Clauses will be accepted with Location Search, but not all of them can be used with Content Search. Reason for this is that while one Location always has exactly one Content item, one Content item can have multiple Locations. In this context some Criteria and Sort Clauses would produce ambiguous queries and such will therefore not be accepted by Content Search.

Content Search will explicitly refuse to accept Criteria and Sort Clauses implementing these abstract classes:

- `eZ\Publish\API\Repository\Values\Content\Query\Criterion\Location`
- `eZ\Publish\API\Repository\Values\Content\SortClause\Criterion\Locat ion`

How to configure your own Criterion and Sort Clause Handlers

After you have implemented your Criterion / Sort Clause and its handler, you will need to configure the handler for the service container using dedicated service tags for each type of search. Doing so will automatically register it and handle your Criterion / Search Clause when it is given as a parameter to one of the Search Service methods.

Available tags for Criterion handlers in Legacy Storage Engine are:

- `ezpublish.search.legacy.gateway.criterion_handler.content`
- `ezpublish.search.legacy.gateway.criterion_handler.location`

Available tags for Sort Clause handlers in Legacy Storage Engine are:

- `ezpublish.search.legacy.gateway.sort_clause_handler.content`
- `ezpublish.search.legacy.gateway.sort_clause_handler.location`

> You will find all the native handlers and the tags for the Legacy Storage Engine available

in the eZ/Publish/Core/settings/storage_engines/legacy/**search_query_handlers.yml** file.

Example of registering ContentId Criterion handler, common for both Content and Location Search

Registering Criterion handler

```
services:

ezpublish.search.legacy.gateway.criterion_handler.common
.content_id:
        class:
eZ\Publish\Core\Search\Legacy\Content\Common\Gateway\Cri
terionHandler\ContentId
        arguments:
[@ezpublish.api.storage_engine.legacy.dbhandler]
        tags:
          - {name:
ezpublish.search.legacy.gateway.criterion_handler.conten
t}
          - {name:
ezpublish.search.legacy.gateway.criterion_handler.locati
on}
```

Example of registering Depth Sort Clause handler for Location Search

Registering Sort Clause handler

```
ezpublish.search.legacy.gateway.sort_clause_handler.loca
tion.depth:
    class:
eZ\Publish\Core\Search\Legacy\Content\Location\Gateway\S
ortClauseHandler\Location\Depth
    arguments:
[@ezpublish.api.storage_engine.legacy.dbhandler]
    tags:
        - {name:
ezpublish.search.legacy.gateway.sort_clause_handler.loca
tion}
```

See also
See also Symfony documentation about Service Container for passing parameters

Criteria Independence Example

With eZ Publish you can have multiple location content. Criterions do not relate to others criterion you can use the Public API and Criterion to search this content, it can lead to a comportement you

are not expecting.
Example of Criterions not relating to each other:

There is a Content with two Locations: Location A and Location B

- Location A is visible
- Location B is hidden

Searching with LocationId criterion with Location B id and Visibility criterion with Visibility::VISIBLE will return the Content because conditions are satisfied:

- Content has Location B
- Content is visible (it has Location A that is visible)

```
<?php

use eZ\Publish\API\Repository\Values\Content\Query;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
\LogicalAnd;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
\LocationId;
use
eZ\Publish\API\Repository\Values\Content\Query\Criterion
\Visibility;

/** @var string|int $locationBId */
/** @var \eZ\Publish\API\Repository\Repository
$repository */

$searchService = $repository->getSearchService();

$query = new Query(
    array(
        'filter' => new LogicalAnd(
            array(
                new LocationId( $locationBId ),
                new Visibility( Visibility::VISIBLE ),
            )
        )
    )
);

$searchResult = $searchService->findContent( $query );

// Content is found
$content = $searchResult->searchHits[0];
```

Reindexing

To (re)create the search engine index for configured search engines (per siteaccess repository), use the `php app/console ezplatform:reindex` command.

Reference

Search Criteria Reference

See the dedicated page with Search Criteria reference.

Sort Clauses Reference

See the dedicated page with Sort Clauses reference.