How do I create a custom sort order for the listings pages?

When you set the sort get parameter in a listing pages url (e.g. ?sort=name-az), a method is called on the **\Aero\Search\Elastic\Query\Builder**class (for name-az the method **sortByNameAz**is called). You can find the default sort by methods in the **Aero\Search\Elastic\Concerns\CanBeSorted**trait.

To add a custom sort to the listings page you therefore just need to macro a method to the **\Aero\Search\Elastic\Query\Builder**class using the correct naming convention.

This example adds an ascending and descending sort by for the summary (to use it you need to add ?sort=summary-az or ?sort=summary-za to the listing page url):

```
<?php

namespace Acme\MyModule;

use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       \Aero\Search\Elastic\Query\Builder::macro('sortBySummaryAz', function () {
           $this->query->addSort([
               'string-sort.summary' => [
                   'order' => 'asc',
               ],
           ]);
       });

       \Aero\Search\Elastic\Query\Builder::macro('sortBySummaryZa', function () {
```

```
            $this->query->addSort([
                'string-sort.summary' => [
                    'order' => 'desc',
                ],
            ]);
        });
    }
}
```

If you try the above code you will notice that it throws an **Elastica\Exception\ResponseException**error. This is because the **string-sort.summary**field used in the Elasticsearch sort by query does not exist.

To add **summary**as a field to the **string-sort**key you need to use the **\Aero\Search\Elastic\Documents\ListingDocument::add()**method to define a closure that will return an array of data to merge into the document structure. It’s important to know that when you change the document structure you need to reindex. You can reindex using the **php artisan aero:search:reindex**command.

```
<?php

namespace Acme\MyModule;

use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
    public function setup()
    {
        \Aero\Search\Elastic\Documents\ListingDocument::add('string-sort', function ($document)
{
            return [
                'summary' => $document->getModel()->product->getTranslation('summary',
request()->store()->language),
            ];
        });

        \Aero\Search\Elastic\Query\Builder::macro('sortBySummaryAz', function () {
            $this->query->addSort([
                'string-sort.summary' => [
```

```
                'order' => 'asc',
            ],
        ]);
    });

    \Aero\Search\Elastic\Query\Builder::macro('sortBySummaryZa', function () {
        $this->query->addSort([
            'string-sort.summary' => [
                'order' => 'desc',
            ],
        ]);
    });
  }
}
```

Advanced Query

It’s possible to do a more advanced query for your Elasticsearch sort by. You can make use of [Elasticsearch’s Painless scripting language]

This example code shows how to add a rand sort that will execute a Java script to sort the listings.

```
<?php

namespace Acme\MyModule;

use Aero\Common\Providers\ModuleServiceProvider;
use Elastica\Script\Script;

class ServiceProvider extends ModuleServiceProvider
{
  public function setup()
  {
    \Aero\Search\Elastic\Query\Builder::macro('sortByRand', function () {
        $this->query->addSort([
            '_script' => [
                'type' => 'number',
```

```
                    'script' => [
                        'lang' => Script::LANG_PAINLESS,
                        'source' => $this->loadJavaFile(__DIR__.'/../scripts/random-
sort.java'),
                        'params' => $this->parameters,
                    ],
                    'order' => 'asc',
                ],
            ]);
        });
    }
}
```

This Java file is located in a scripts folder outside of the src folder:

```
def sorts = new ArrayList();

Random rand = new Random();
sorts.add(rand.nextInt(1000000));

if (sorts.size() === 0) {
  sorts.add(0);
}

Collections.min(sorts.stream().map(Double::doubleToRawLongBits).collect(Collectors.toList()));
```

Revision #1
Created 2026-09-02 13:28:24 UTC by Michael Price
Updated 2026-09-02 13:28:24 UTC by Michael Price