Configuration

Configuration

How do I register new permissions?

```
// add a single permission
\Aero\Admin\Permissions::add('custom');

// add multiple permissions
\Aero\Admin\Permissions::add(['custom', 'custom.view', 'custom.manage']);
```

How do I check for permissions?

When checking for permission, you should always be as explicit as you can. For example, checking a user has the `custom.view` or `custom.manage` permissions.

Guarding content in a Blade file:

```
@can('custom.view')
    ...
@endcan

@canany(['orders', 'custom'])
    ...
@endcan
```

Checking the admin user has permission to access an endpoint:

```
Route::get('/my-module', [MyModuleController::class, 'index'])
    ->name('admin.modules.my-module')
    ->middleware('can:custom.view');

Route::get('/my-module/manage', [MyModuleController::class, 'manage'])
    ->name('admin.modules.my-module.manage')
    ->middleware('can:custom.manage');
```

Guarding the execution of code in a Controller:

```
if ($this->can('custom')) {
    // ...
}

if ($this->canAny(['orders', 'custom'])) {
    // ...
```

```
}
```

How do I change how order references are generated?

You can set an order reference generator to define how order references are generated. To do this you need to use the static **setReferenceGenerator**method on the **Aero\Cart\Models\Order** model.

The method accepts a closure that accepts the order as a parameter and expects you to return a string (the reference for the order passed into the closure).

If you do not set an order reference generator the default generator will be used that does the following:

```
<?php

namespace Acme\MyModule;

use Aero\Cart\Models\Order;
use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       Order::setReferenceGenerator(function (Order $order) {
           $prefix = setting('order_reference_prefix');
           $random = substr(str_shuffle(str_repeat('0123456789', 10)), 1, 3);

           return "{$prefix}{$order->getKey()}-{$random}";
       });
   }
}
```

What is the robots file and how does it differ from normal?

The robots.txt file provides instructions for web crawlers and bots that index your store. Aero automatically registers core parts of the store that should avoid being indexed and visited by bots.

In order to extend this file and add modules or store specific restrictions, the robots.txt file is processed as a pipeline, where the content (the text) is made available and can be manipulated.

For ease of access, the **$content**passed through the pipeline is an **Illuminate\Support\Collection**, grouped by the user agent.

```
\Aero\Store\Pipelines\RobotsTxt::extend(static function ($content) {
    if ($agent = $content->get('User-agent: *')) {
        $agent->push('Disallow: /my_module/');
    }
});
```

How do I store product images on S3

You need to configure and set up your Laravel project for the S3 storage driver. You can read more about this in the laravel documentation**AWS_ACCESS_KEY_ID**, **AWS_SECRET_ACCESS_KEY**, **AWS_DEFAULT_REGION**, and **AWS_BUCKET**) and installing a composer package (league/flysystem-aws-s3-v3 ~1.0) that lets Laravel support S3.

Once your project is configured you should set **STORE_FILE_DRIVER**and **STORE_LOCAL_FILE_DRIVER**to your S3 driver. If you have followed the Laravel documentation steps above and are using the default S3 disk already created you will simply add this to your env file:

```
STORE_FILE_DRIVER=s3
STORE_LOCAL_FILE_DRIVER=s3
```

STORE_FILE_DRIVER
This by default is set as public and is the disk used by Aero when uploading files that should be publicly accessible.

STORE_LOCAL_FILE_DRIVER
This by default is set as local and is the disk used by Aero when uploading files that should not be publicly accessible directly.