Settings

Settings

What are the settings available to developers?

Settings allow you to add dynamic values to your modules that can be modified by admin users easily through the admin.

The values of your settings (unless the value is the default you set) are stored in json files inside of your storage directory.

```
└─ storage
    └─ app
        └─ settings
```

Do do I set the default value of my custom setting?

You can add a default value to any setting by using the **default**method and passing your default value.

```
<?php

namespace Acme\MyModule;

use Aero\Common\Facades\Settings;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       Settings::group('acme-my-module', function (SettingGroup $group) {
           $group->string('denied_message')->default('Access Denied');
           $group->array('allow_list')->default(['admin']);
           $group->boolean('allow_list_enabled')->default(true);
       });
   }
}
```

How do I register custom settings?

To register settings you need to use the **Aero\Common\Facades\Settings**facade from the **setup** method of your modules service provider. You should call the **group**method on the facade and pass in a unique name for your settings group and a closure that defines your settings group.

```
<?php

namespace Acme\MyModule;

use Aero\Common\Facades\Settings;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       Settings::group('acme-my-module', function (SettingGroup $group) {
           $group->string('denied_message');
           $group->array('allow_list');
           $group->boolean('allow_list_enabled');
       });
   }
}
```

How do I access a custom setting?

You can access a setting by using the **setting**function that is available in PHP, Blade, and Twig. You simply need to pass your setting group name, a full stop, and then the settings name.

```
setting(‘acme-my-module.allow_list’)
```

What are the available setting types?

Setting	Description
$group->array(‘allow_list’);	Returns an array.
$group->boolean(‘enabled’);	Returns a boolean, true or false.
$group->eloquent('order_status', OrderStatus::class);	Stores the key of the eloquent model you pass and returns an instance of the model.
$group->encrypted('api_key');	Stores a string encrypted and returns the string decrypted.
$group->float('min_value');	Returns a number as a float.
$group->integer('per_page');	Returns a number as an integer.
$group->string(‘message’);	Returns a string.
$group->date('go_live');	Returns a Carbon date instance.
$group->dateRange('sale_period');	Returns an array with start and end keys that hold a Carbon date instance.

How do I configure the admin edit page?

By default your setting group is editable through the admin by a generated edit page. You cannot remove this generated edit page but you can stop users from being able to edit the settings using it. It’s important to note that if you disable editing through this edit page that users can still edit the settings manually through the storage json file.

To make your setting group not editable you should call the **notEditable**method when defining your group.

```
<?php

namespace Acme\MyModule;

use Aero\Common\Facades\Settings;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       Settings::group('acme-my-module', function (SettingGroup $group) {
           $group->string('denied_message');
           $group->array('allow_list');
           $group->boolean('allow_list_enabled');
           $group->notEditable();
       });
   }
}
```

Validation Rules

These are helper methods that map to their equivalent Laravel Validation Rules

Rule	Works For
$group->string('key')->required();	All
$group->string('key')->between($min, $max);	String, Integer, Float, Array, Encrypted
$group->string('key')->min($min);	String, Integer, Float, Array, Encrypted
$group->string('key')->max($max);	String, Integer, Float, Array, Encrypted
$group->string('key')->size($size);	String, Integer, Float, Array, Encrypted
$group->string('key')->greaterThan($size);	String, Integer, Float, Array, Encrypted
$group->string('key')->lessThan($size);	String, Integer, Float, Array, Encrypted
$group->string('key')->in($needles);	String, Integer, Float
$group->string('key')->email();	String, Encrypted
$group->string('key')->uuid();	String, Encrypted
$group->string('key')->url();	String, Encrypted
$group->string('key')->ip();	String, Encrypted
$group->string('key')->startsWith($needle);	String, Encrypted
$group->string('key')->startsWithOneOf($needles);	String, Encrypted
$group->float('key')->step($step);	Float
$group->date('key')->beforeToday();	Date
$group->date('key')->before($date);	Date
$group->date('key')->afterToday();	Date
$group->date('key')->after($date);	Date

Methods

Method	For	Description
$group->title($title);	-	Set the title shown in the list of settings (by default this is generated from the group name).
$group->summary($summary);	-	Set the summary shown in the list of settings.
$group->notEditable();	-	Disable editing through the generated admin page.

Method	For	Description
$group->string('key')->label($label);	All	Set the label for the edit field (by default this is generated from the settings key).
$group->string('key')->hint($hint);	All	Add a helpful hint to the edit field.
$group->string('key')->textarea();	String	Make the string field appear as a textarea.
$group->string('key')->wysiwyg();	String	Make the string field appear as a wysiwyg.
$group->eloquent('order_status', OrderStatus::class)->searchField($field);	Eloquent	Define the field used for searching in the searchable select.
$group->eloquent('order_status', OrderStatus::class)->searchFields($field);	Eloquent	Define multiple fields used for searching in the searchable select (first field is the value).
$group->array('key')->associative();	Array	Define the array as an associative array - an array that has a key and value.
$group->array('key')->definition($definition);	Array	Set a custom definition for your array (link:section below this).

Complex Array Definitions

If you have a complex array (an array of arrays that have multiple keys), you need to define the contents of the array and any validation for each key.

Here is a code example that defines an additional_checkboxes array that is an array of arrays. The child arrays have a text key (that has validation ensuring it is a string and required) and a required key (that has validation ensuring it is a boolean).

Although this example uses the string, required, and boolean rules you can use any Laravel Validation rule

```
<?php

namespace Acme\MyModule;

use Aero\Common\Facades\Settings;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
```

```
{
   public function setup()
   {
       Settings::group('acme-my-module', function (SettingGroup $group) {
           $group->array('additional_checkboxes')->definition([
               'text' => ['string', 'required'],
               'required' => ['boolean']
           ]);
       });
   }
}
```

How do I handle translations?

When defining your setting groups title/summary, or a settings label/hint you can make use of Laravel Localization for translations.

```
<?php

namespace Acme\MyModule;

use Aero\Common\Facades\Settings;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       Settings::group('acme-my-module', function (SettingGroup $group) {
           $group->title(__('acme-my-module.title'));
           $group->summary(__('acme-my-module.summary'));
           $group->string('message')
               ->label(__('acme-my-module.labels.message'))
               ->hint(__('acme-my-module.hints.message'));
       });
   }
}
```

How do I add settings to the product model?

Using the same syntax as when you create settings

```
<?php

namespace Acme\MyModule;

use Aero\Catalog\Models\Product;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       Product::settings('my-module', function (SettingGroup $group) {
           $group->encrypted('password');
           $group->boolean('require_password_to_view')->default(false);
       });
   }
}
```

To access your settings value you need to use the settings method on the product model, passing in the key for the setting you want.

```
<?php

namespace Acme\MyModule;

use Aero\Catalog\Models\Product;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

class ServiceProvider extends ModuleServiceProvider
```

```
{
   public function setup()
   {
       Product::settings('my-module', function (SettingGroup $group) {
           $group->encrypted('password');
           $group->boolean('require_password_to_view')->default(false);
       });

       $product = Product::first();

       $password = $product->settings('my-module.password');

       dd($password);
   }
}
```