

How do I add settings to my model?

This tutorial will explain the steps required to add settings to your model and assumes you have your model setup with create and edit pages.

Adding a Trait to your Model

The first step is to add the `Aero\Common\Traits\CanHaveSettings` trait to your model.

```
<?php

namespace Acme\MyModule\Models;

use Aero\Common\Models\Model;
use Aero\Common\Traits\CanHaveSettings;

class MyModel extends Model
{
    use CanHaveSettings;
}
```

Defining the Settings for your Model

Now that your model has the `Aero\Common\Traits\CanHaveSettings` trait you can use the static `settings()` function on your models class to define your settings.

The settings are defined in the same way as when you create a normal setting group, the only difference being that you use your models class instead of the normal settings facade. [You can read more about settings here](#)

```
<?php

namespace Acme\MyModule;

use Acme\MyModule\Models\MyModel;
use Aero\Common\Providers\ModuleServiceProvider;
use Aero\Common\Settings\SettingGroup;

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

Adding the Settings Fields to your Create/Edit Pages

Now that you have defined the settings for your model you can update your models create/edit pages to let users update the settings.

To add a card that lets user edit the models settings to your create/edit pages you need to include the `admin::settings.model-settings` view and pass in your model, like this:

```
@include('admin::settings.model-settings', ['model' => $myModel])
```

If you don't have a model to pass in (because you're on a create page so the model hasn't been created yet), pass in a fresh instance of your model, like this:

```
@include('admin::settings.model-settings', ['model' => new \Acme\MyModule\Models\MyModel()])
```

It's important to ensure that the include is inside of your form. A more "complete" example may look something like this:

```

@extends('admin::layouts.main')

@section('content')
    <div class="max-w-2xl mx-auto">
        <div class="flex w-full justify-between">
            <h2><a href="{{ route('admin.modules', request()->all()) }}" class="btn mr-4">@include('admin::icons.back') Back</a> Managing My Model</h2>
        </div>
        @include('admin::partials.alerts')
        <form action="#" method="post" class="flex flex-wrap">
            @csrf
            @method('put')
            <fieldset class="w-full">
                {{--
                    Your other fields etc--}}

                @include('admin::settings.model-settings', ['model' => $myModel])
                <div class="form-buttons fieldset-disabled-hide">
                    <div class="card w-full">
                        <button class="btn btn-secondary" type="submit">Save</button>
                    </div>
                </div>
            </fieldset>
        </form>
    </div>
@endsection

```

Adding the Settings Validation to your Create/Edit Requests

You need to update your request validators so that the settings data will be correctly validated and formatted.

Adding the Rules

You need to update your validators `rules` method to merge in the settings rules with your current rules. To do this you need to `array_merge` your rules with the array of rules returned by the `Aero\Admin\Utils\SettingHelpers::getRulesForModel` method. The

`Aero\Admin\Utils\SettingHelpers::getRulesForModel` method expects you to pass the class string of your model.

```
<?php

namespace Acme\MyModule\Requests;

use Acme\MyModule\Models\MyModel;
use Aero\Admin\Utils\SettingHelpers;
use Aero\Common\Requests\AeroRequest;

class StoreMyModelRequest extends AeroRequest
{
    public function rules(): array
    {
        return array_merge([
            'name' => 'required|max:255', // You can add your models settings here
        ], SettingHelpers::getRulesForModel(MyModel::class));
    }
}
```

Adding the Attributes

You need to do the same thing for the attributes.

```
<?php

namespace Acme\MyModule\Requests;

use Acme\MyModule\Models\MyModel;
use Aero\Admin\Utils\SettingHelpers;
use Aero\Common\Requests\AeroRequest;

class StoreMyModelRequest extends AeroRequest
{
    public function attributes(): array
    {
        return array_merge([], SettingHelpers::getRuleAttributesForModel(MyModel::class));
    }
}
```

```

public function rules(): array
{
    return array_merge([
        'name' => 'required|max:255', // You can add your models settings here
    ], SettingHelpers::getRulesForModel(MyModel::class));
}
}

```

Formatting the Data for Validation

You need to add a `prepareForValidation` method to your validator and ensure it calls the `SettingHelpers::formatRequestDataForModel` method like shown below. This method will format the incoming settings request data so that it is ready to be validated.

```

<?php

namespace Acme\MyModule\Requests;

use Acme\MyModule\Models\MyModel;
use Aero\Admin\Utils\SettingHelpers;
use Aero\Common\Requests\AeroRequest;

class StoreMyModelRequest extends AeroRequest
{
    public function prepareForValidation()
    {
        $this->replace(
            SettingHelpers::formatRequestDataForModel(MyModel::class, $this->all())
        );
    }

    public function attributes(): array
    {
        return array_merge([], SettingHelpers::getRuleAttributesForModel(MyModel::class));
    }

    public function rules(): array
    {

```

```

        return array_merge([
            'name' => 'required|max:255', // You can add your models settings here
        ], SettingHelpers::getRulesForModel(MyModel::class));
    }
}

```

Saving the Validated Settings for your Model

To save the settings you need to pass your model and data into the `Aero\Admin\Utils\SettingHelpers::saveForModel` method, like this:

```

<?php

namespace Acme\MyModule\Http\Controllers;

use Acme\MyModule\Models\MyModel;
use Acme\MyModule\Requests\UpdateMyModelRequest;
use Aero\Admin\Http\Controllers\Controller;
use Aero\Admin\Utils\SettingHelpers;

class MyModelController extends Controller
{
    public function update(UpdateMyModelRequest $request, MyModel $model)
    {
        $model->update($data = $request->validated());

        SettingHelpers::saveForModel($model, $request->validated()['settings'] ?? []);

        return redirect(route('my-model.index'))->with([
            'message' => __('Your changes have been saved'),
        ]);
    }
}

```