Anatomy of a custom module

```
└─ module
    └─ database
        └─ migrations
    └─ public
    └─ resources
        └─ css
        └─ js
       └─ views
           └─ admin
          └─ store
    └─ routes
    └─ src
        └─ Http
       └─ Controllers
       └─ Responses
           └─ Steps
       └─ Requests
       └─ Models
    ServiceProvider.php
.gitignore
composer.json
README.md
```

Database

The database folder is only required for purposes of migrating, seeding, or storing data in formats such as JSON.

Migrations

Migrations is a folder within the database directory structure used for storing migration files which allow for populating the database with missing tables, or updating them.

Running `php artisan migrate` in the **root project directory** after installing a module detects migrations from all modules installed on Aero, which have not yet been migrated, and migrates them into Aero.

Public

All files within the public directory of a module can be published, to then be accessed anywhere in Aero. Some modules might require the public folder published before it can be used.

This might include images or files used to instantiate a JavaScript file for the use of Vue or any other JS framework.

Resources

The resources are required by the module for rendering views and storing custom CSS and JS files. Separating views into sub-folders allows for more readability and clarity on where particular views are used.

For example, the **admin** and **store** sub-folders of `resources/views` can be used for separating the different routing the module has.

Routes

Modules can have custom routes which are then served to the rest of the app. These might serve as callback URLs for APIs in the module or rendering views.

Example:

```
<?php

use Aerocargo\Csv\Http\Controllers\CsvController;
use Illuminate\Support\Facades\Route;

Route::group(['prefix' => 'csv'], function ($route) {
      $route->get('/', [CsvController::class, 'index'])->name('admin.csv.import');
      $route->get('/mapping/{hash?}', [CsvController::class, 'mapping'])-
>name('admin.csv.mapping');
      $route->get('/search', [CsvController::class, 'aeroFieldsSearch'])-
>name('admin.csv.search.aero');
      $route->get('/search-csv', [CsvController::class, 'csvFieldsSearch'])-
```

```
>name('admin.csv.search.csv');
      $route->get('/processing', [CsvController::class, 'processing'])-
>name('admin.csv.processing');

      $route->post('/importing', [CsvController::class, 'importing'])-
>name('admin.csv.importing');
      $route->post('/process', [CsvController::class, 'process'])-
>name('admin.csv.process');
      $route->post('/import', [CsvController::class, 'import'])-
>name('admin.csv.process.import');
      $route->post('/ping-import', [CsvController::class, 'pingImport'])-
>name('admin.csv.process.ping');
      $route->post('/get-mapping/{id?}', [CsvController::class, 'getMappingData'])-
>name('admin.csv.get.mapping');
});
```

Source (src)

The source folder contains all of the backend serving the entire module, whether it is controllers, models, or traits.

Service Provider

The module **Service Provider**is used to instantiate the module, and usually contains configuration for the resources the module uses. This could range from routes, views to being able to display the module and its UI within the Modules section of Aero.

The scaffolded Service Provider has several functions that are hidden from view.

These include:

- **setup()**- **assetLinks()**- **boot()**- **seed()**- **getSeeds()**

setup()

Every scaffolded Service Provider will include the setup() function, which has to be configured to include all necessary resources.

Example:

```
public function setup()
{
     AdminSlot::inject('catalog.products.index.header.buttons', function () {
         return view('csv::buttons.import');
     });

     Router::addAdminRoutes(__DIR__.'/../routes/admin.php');

     $this->loadMigrationsFrom(__DIR__.'/../database/migrations');

     $this->loadViewsFrom(__DIR__.'/../resources/views', 'csv');

     BulkAction::create(ExportProducts::class, ProductsResourceList::class)
                 ->notRunnable()
                 ->permissions('products.export')
                 ->title('Export products');
}
```

assetLinks()

Asset links are used to publish the resources included in the public folder of the module so that they can be used anywhere on the Aero platform.

Example:

```
public function assetLinks()
{
     return [
         'vendor/module-name' => __DIR__ . '/../public',
     ];
}
```

boot()

The boot() method is run in the background if the setup() method is present within the Service Provider. This is also the case for the booted() method.

seed()

With the **seed()**method, it is possible to add a seed to a collection of seeds for the module.

getSeeds()

Returns a unique collection of seeds that have been added to the module.

If we require our module to listen to certain events, they can be added into an array property of the class:

Listen property

The listen property is an array of events for each given module. The syntax for adding a listener to an event looks like so:

```
protected $listen = [
        \Aero\Cart\Events\OrderSuccessful::class => [
            \Aerocargo\Testing\Listeners\ExportOrder::class,
        ],
];
```

Custom directories

It is possible to create custom directories for different types of classes, for example, **Traits**, **Jobs**, or **Helpers**.

Revision #1
Created 2026-09-02 13:28:31 UTC by Michael Price
Updated 2026-09-02 13:28:31 UTC by Michael Price