How do I add a custom field to the new and edit product page?

In this mini tutorial we will add a notes field to the product new and edit page for the product and each variant.

This mini tutorial assumes that you have a module setup or you're happy working from the app service provider (using the boot method). You can see how to set up a module here (link).

Adding the Database Migration

To get started we will add a migration file that will update the products and variants tables to have a notes column. To do this we'll need to create the migration file and load them from within the modules service provider.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class AddNotesFieldsToProductsAndVariants extends Migration
{
   /**
    * Run the migrations.
    *
    * @return void
    */
   public function up()
   {
       Schema::table('products', function (Blueprint $table) {
```

```
            $table->string('notes')->nullable()->after('description');
        });

        Schema::table('variants', function (Blueprint $table) {
            $table->string('notes')->nullable()->after('name');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::table('products', function (Blueprint $table) {
            $table->dropColumn('notes');
        });

        Schema::table('variants', function (Blueprint $table) {
            $table->dropColumn('notes');
        });
    }
}
```

Adding the Slot View

Now we'll create two views, one that will be injected for the product and one that will be injected for the variants. These views will have a text area in them for the notes input.

product-notes-field.blade.php

```
<div class="card mt-4">
    <label for="notes" class="block mb-2">Notes</label>
    <textarea name="notes" id="notes" v-model="product.notes" class="w-full"></textarea>
</div>
```

variant-notes-field.blade.php

```
<div class="p-4" v-if="!isSimpleProduct">
   <label :for="'variant-notes-' + key" class="block mb-2">Notes</label>
   <textarea :name="'variants[' + key + '][notes]'" :id="'variant-notes-' + key" v-
model="variants[key].notes" class="w-full"></textarea>
</div>
```

After creating the views we will load them in the module service provider using the `$this->loadViewsFrom` method and inject them into the relevant slots using the `Aero\Admin\AdminSlot::inject` method.

```
<?php

namespace Acme\MyModule;

use Aero\Admin\AdminSlot;
use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       if ($this->app->runningInConsole()) {
           $this->loadMigrationsFrom(__DIR__.'/../database/migrations');
       }

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

       AdminSlot::inject('catalog.product.new.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.edit.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.new.variant', 'my-module::variant-notes-field');
       AdminSlot::inject('catalog.product.edit.variant', 'my-module::variant-notes-field');
   }
}
```

Making the Notes Save

Adding Notes as a Fillable

We need to make the notes fillable as Laravel will only mass assign fillable attributes (`Aero\Catalog\Models\Product and Aero\Catalog\Models\Variant`).

```
<?php

namespace Acme\MyModule;

use Aero\Admin\AdminSlot;
use Aero\Catalog\Models\Product;
use Aero\Catalog\Models\Variant;
use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       if ($this->app->runningInConsole()) {
           $this->loadMigrationsFrom(__DIR__.'/../database/migrations');
       }

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

       AdminSlot::inject('catalog.product.new.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.edit.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.new.variant', 'my-module::variant-notes-field');
       AdminSlot::inject('catalog.product.edit.variant', 'my-module::variant-notes-field');

       Product::makeFillable('notes');
       Variant::makeFillable('notes');
   }
}
```

Adding Notes to the Validators

To let the notes input data get through validation it needs to be added to the `Aero\Admin\Http\Requests\Catalog\CreateProductRequest` and

`Aero\Admin\Http\Requests\Catalog\UpdateProductRequest` validators. To do this we need to use the `expects` method on the two validators and pass in the notes field and its rules.

```
<?php

namespace Acme\MyModule;

use Aero\Admin\AdminSlot;
use Aero\Admin\Http\Requests\Catalog\CreateProductRequest;
use Aero\Admin\Http\Requests\Catalog\UpdateProductRequest;
use Aero\Catalog\Models\Product;
use Aero\Catalog\Models\Variant;
use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       if ($this->app->runningInConsole()) {
           $this->loadMigrationsFrom(__DIR__.'/../database/migrations');
       }

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

       AdminSlot::inject('catalog.product.new.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.edit.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.new.variant', 'my-module::variant-notes-field');
       AdminSlot::inject('catalog.product.edit.variant', 'my-module::variant-notes-field');

       Product::makeFillable('notes');
       Variant::makeFillable('notes');

       CreateProductRequest::expects('notes', 'nullable|string');
       UpdateProductRequest::expects('notes', 'nullable|string');
   }
}
```

Adding Notes to the Transformers

Adding notes to the transformers will ensure that it's available in our views through Vue. We need to use the add method on the `Aero\Admin\Transformers\BaseVariantTransformer`, `Aero\Admin\Transformers\ProductTransformer`, and `Aero\Admin\Transformers\VariantTransformer` transformers.

```
<?php

namespace Acme\MyModule;

use Aero\Admin\AdminSlot;
use Aero\Admin\Http\Requests\Catalog\CreateProductRequest;
use Aero\Admin\Http\Requests\Catalog\UpdateProductRequest;
use Aero\Admin\Transformers\BaseVariantTransformer;
use Aero\Admin\Transformers\ProductTransformer;
use Aero\Admin\Transformers\VariantTransformer;
use Aero\Catalog\Models\Product;
use Aero\Catalog\Models\Variant;
use Aero\Common\Providers\ModuleServiceProvider;

class ServiceProvider extends ModuleServiceProvider
{
   public function setup()
   {
       if ($this->app->runningInConsole()) {
           $this->loadMigrationsFrom(__DIR__.'/../database/migrations');
       }

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

       AdminSlot::inject('catalog.product.new.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.edit.cards', 'my-module::product-notes-field');
       AdminSlot::inject('catalog.product.new.variant', 'my-module::variant-notes-field');
       AdminSlot::inject('catalog.product.edit.variant', 'my-module::variant-notes-field');

       Product::makeFillable('notes');
       Variant::makeFillable('notes');

       CreateProductRequest::expects('notes', 'nullable|string');
       UpdateProductRequest::expects('notes', 'nullable|string');

       ProductTransformer::add(function ($data) {
```

```
            return [
                'notes' => $data['product']->notes ?? '',
            ];
        });

        VariantTransformer::add(function ($data) {
            return [
                'notes' => $data['variant']->notes ?? '',
            ];
        });

        BaseVariantTransformer::add(function ($data) {
            return [
                'notes' => '',
            ];
        });
    }
}
```

Revision #1
Created 2026-09-02 13:27:55 UTC by Michael Price
Updated 2026-09-02 13:27:55 UTC by Michael Price