

Tutorials

- [How do I add a custom field to the new and edit product page?](#)

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](#).

1. 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:

```
class AddNotesFieldsToProductsAndVariants extends \Illuminate\Database\Migrations\Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        \Illuminate\Support\Facades\Schema::table('products', function
        (\Illuminate\Database\Schema\Blueprint $table) {
            $table->string('notes')->nullable()->after('description');
        });

        \Illuminate\Support\Facades\Schema::table('variants', function
        (\Illuminate\Database\Schema\Blueprint $table) {
```

```

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

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

    \Illuminate\Support\Facades\Schema::table('variants', function
(\Illuminate\Database\Schema\Blueprint $table) {
        $table->dropColumn('notes');
    });
}
}

```

2. 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 inject them into the relevant slots using the

`Aero\Admin\AdminSlot::inject()` method:

```
\Aero\Admin\AdminSlot::inject('catalog.product.new.cards', 'acme::product-notes-field');
\Aero\Admin\AdminSlot::inject('catalog.product.edit.cards', 'acme::product-notes-field');

\Aero\Admin\AdminSlot::inject('catalog.product.new.variant', 'acme::variant-notes-field');
\Aero\Admin\AdminSlot::inject('catalog.product.edit.variant', 'acme::variant-notes-field');
```

3. Making the notes save

3.1. Make the notes fillable

We need to make the notes fillable as Laravel will only mass assign [fillable attributes](#):

```
\Aero\Catalog\Models\Product::makeFillable('notes');
\Aero\Catalog\Models\Variant::makeFillable('notes');
```

3.2. Adding validation for notes field

To validate the notes, we need to extend the product create and update requests:

```
\Aero\Admin\Http\Requests\Catalog\CreateProductRequest::expects('notes', 'nullable|string');
\Aero\Admin\Http\Requests\Catalog\UpdateProductRequest::expects('notes', 'nullable|string');

\Aero\Admin\Http\Requests\Catalog\CreateProductRequest::expects('variants.*.notes',
'nullable|string');
\Aero\Admin\Http\Requests\Catalog\UpdateProductRequest::expects('variants.*.notes',
'nullable|string');
```

See [here](#) for more guidance on extending requests.

3.3. Adding notes to the transformers

Adding notes to the transformers will ensure that it's available in our views through Vue:

```
\Aero\Admin\Transformers\ProductTransformer::add(function ($data) {
    return [
        'notes' => $data['product']->notes ?? '',
    ];
});

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

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

See [here](#) for more information about transformers.

Complete Service Provider

```
<?php

namespace Aerocargo\Acme;

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', 'acme');

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

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

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

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

        CreateProductRequest::expects('variants.*.notes', 'nullable|string');
        UpdateProductRequest::expects('variants.*.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' => '',
```

```
    ];
```

```
});
```

```
}
```

```
}
```