

Rumah Coding Course "Laravel"

Create Model with Migration

- So far, we've created model and migration separately, today we will create our models and let Laravel generate the migration file for us.
- If you would like to generate a database migration when you generate the model, you may use the --migration or -m option:

```
php artisan make:model Flight --migration

php artisan make:model Flight -m
```



Task #1 Create new model: Product (Also generate migration file)

```
Rumah Coding@DESKTOP-CGRG08T D:\laravel\sip
$ php artisan make:model Product --migration
```



Task #2 Modify migration file: create_table_products.php

```
public function up()
   Schema::create('products', function (Blueprint $table) {
       $table->increments('id');
       $table->unsignedInteger('category_id');
       $table->string('name');
       $table->text('description');
       $table->bigInteger('price');
       $table->string('sku');
       $table->text('image');
       $table->enum('status', ['active', 'inactive']);
       $table->timestamps();
       $table->foreign('category_id')->references('id')->on('categories');
   });
```



Task #3 Run migration

```
Rumah Coding@DESKTOP-CGRG08T D:\laravel\sip
$ php artisan migrate
```



Resource Controller

- So far, we've created all controllers and routes manually, today we will use **resource controller**.
- Laravel resource routing assigns the typical "CRUD" routes to a controller with a single line of code.
- For example, you may wish to create a controller that handles all HTTP requests for "photos" stored by your application. Using the make:controller Artisan command, we can quickly create such a controller:

php artisan make:controller PhotoController --resource



Task #4 Create new resource controller: ProductController

```
Rumah Coding@DESKTOP-CGRG08T D:\laravel\sip
$ php artisan make:controller ProductController --resource
```



Open ProductController.php, you'll see all methods already created: index, create, store, show, edit, update, destroy



Task #5 Create new view folder: **product**

Task #5.5 Create new resource route: /admin/product

Route::resource('/admin/product', 'ProductController')



Using single line Route::resource, Laravel will generate all routes for you on the fly. You can check all the routes using php artisan route:list



- Laravel comes with handy library to work with Form and HTML.
- But since version 5.0, the library not included in default installation anymore, you must install it manually.
- Begin by installing this package through Composer. Edit your project's composer.json file to require laravelcollective/html

composer require "laravelcollective/html": "^5.4.0"



 Next, add your new provider to the providers array of config/app.php

```
'providers' => [
   // ...
   Collective\Html\HtmlServiceProvider::class,
   // ...
],
```



 Finally, add two class aliases to the aliases array of config/app.php

```
'aliases' => [
   // ...
   'Form' => Collective\Html\FormFacade::class,
   'Html' => Collective\Html\HtmlFacade::class,
   // ...
],
```



 Now, the library has installed, you can use create a form using this code:

```
{{ Form::open(array('url' => 'foo/bar')) }}

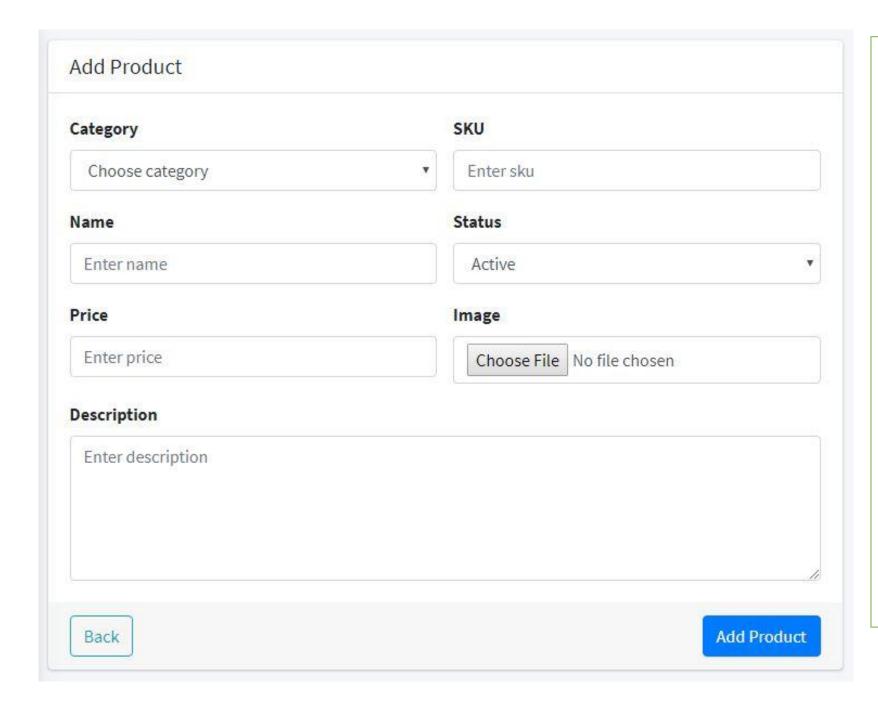
//
{{ Form::close() }}
```



HTML & Form Methods

- Form::open()
- Form::close
- Form::label
- Form::text
- Form::select
- Form::password
- Form::email
- Form::checkbox
- Form::radio
- Etc





Task #6
Create new view
file:
create.blade.php
(under product
folder & use Form
library)



Because our form contains upload file, don't forget to use files=>true

```
{{ Form::open(['route'=>'product.store', 'files'=>true]) }}
```



Task #6 Answer http://bit.ly/2uggeKG

Task #7 Modify create method on ProductController

```
public function create()
    // ambil dari database
   $categories = \App\Category::pluck('name', 'id');
    $data = [
        'categories' => $categories
    return view('admin/product/create')->with($data);
```



You can test by opening http://127.0.0.1:8000/admin/product/create on your browser

Public Disk

- The public disk is intended for files that are going to be publicly accessible.
- By default, the public disk uses the local driver and stores these files in storage/app/public.
- To make them accessible from the web, you should create a symbolic link from public/storage to storage/app/public.

php artisan storage:link



Public Disk

• To generate url for uploaded file, you can use this code:

```
asset('storage/'.$product['image'])
```



Task #8 Modify **store** method on **ProductController** (save the product data to database)

Task #8 Answer http://bit.ly/2m4AWcQ

Task #9 Create new view file: index.blade.php (under product folder)

ID	Category	Name	Price	SKU	Image	Status	Action
3	Fashion Wanita	Baju Kebaya	1000000	A001		Active	O P û
4	Fashion Wanita	Kerudung	123123	A002		Active	• / û
5	Fashion Pria	Celana Panjang	123123	B001		Active	O P O

Task #9 Answer http://bit.ly/2KZVHkx

Task #10 Modify index method on ProductController (load all products data)



You can test by opening http://127.0.0.1:8000/admin/product/ on your browser

Task #11 Modify view file: **index.blade.php**Add plus button and link to create product page

D	Category	Name	Price	SKU	Image	Status	Action
3	Fashion Wanita	Baju Kebaya	1000000	A001		Active	O
4	Fashion Wanita	Kerudung	123123	A002		Active	• / û
5	Fashion Pria	Celana Panjang	123123	B001		Active	



Database: Pagination

- In other frameworks, pagination can be very painful.
- Laravel's paginator is integrated with the query builder and Eloquent ORM and provides convenient, easy-to-use pagination of database results out of the box.
- The HTML generated by the paginator is compatible with the Bootstrap CSS framework.



Database: Pagination

- To paginate a database result, use paginate method on the query builder or an Eloquent query.
- The paginate method automatically takes care of setting the proper limit and offset based on the current page being viewed by the user

```
$users = DB::table('users')->paginate(15);
```



Task #12 Modify index method on ProductController paginate the result

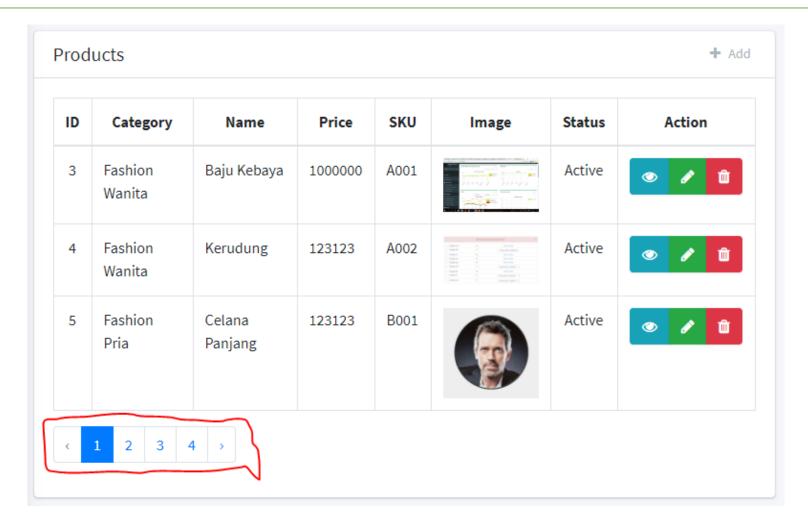
```
$paginate = 3;
$products = \App\Product::paginate($paginate);

$data = [
    'products' => $products,
];

return view('admin/product/index')->with($data);
```



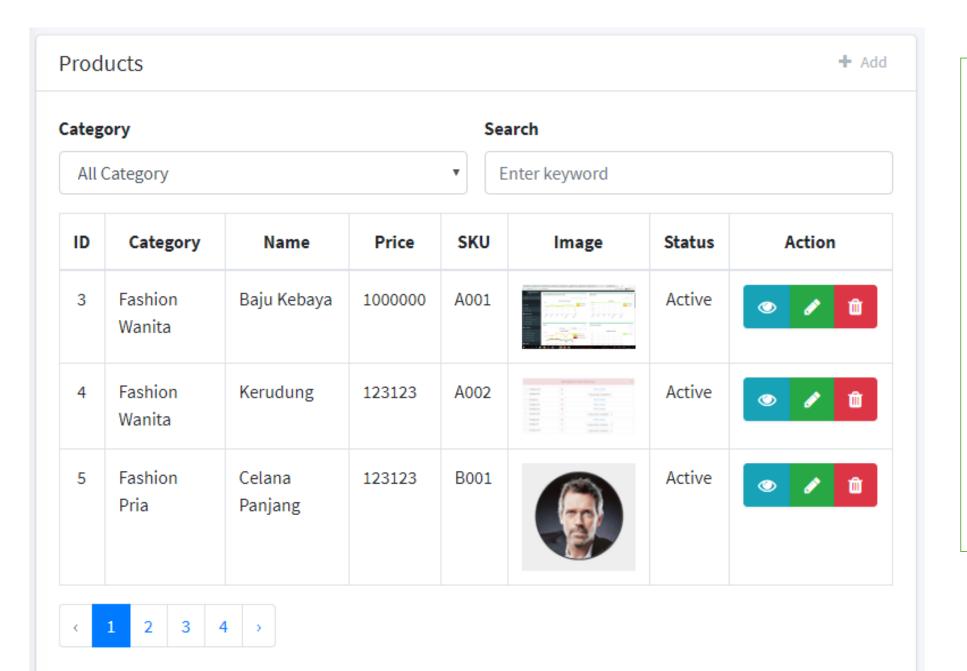
Task #13 Modify view file: **index.blade.php**Display pagination button





Task #13 Answer





Task #14
Modify view file:
index.blade.php
Add category
filter and search



Task #14 Answer

```
<div class="row">
   <div class="col-md-6">
       <div class="form-group">
           {{ Form::label('category', 'Category') }}
            {{ Form::select('category', $categories, $cat_id,
                    ['class'=>'form-control', 'placeholder' => 'All Category']) }}
       </div>
   </div>
   <div class="col-md-6">
       <div class="form-group">
            {{ Form::label('search', 'Search') }}
            {{ Form::text('search', $keyword, ['class'=>'form-control',
            'placeholder'=>'Enter keyword']) }}
       </div>
   </div>
</div>
```



Task #15 Modify index method on ProductController apply filter

Task #15 Answer http://bit.ly/2u3Mgdl

Is the filter work now? You need javascript

Task #16 Modify view file: **index.blade.php**Adding javascript to make filter works

```
$('#category').on('change', function() {
    filter();
});
$('#search').keypress(function(event) {
    if(event.keyCode == 13) {
        filter();
});
var filter = function() {
    var catId = $('#category').val();
    var keyword = $('#search').val();
    window.location.replace("{{ URL::to('admin/product') }}?cat_id=" +
    catId + "&keyword=" + keyword);
```

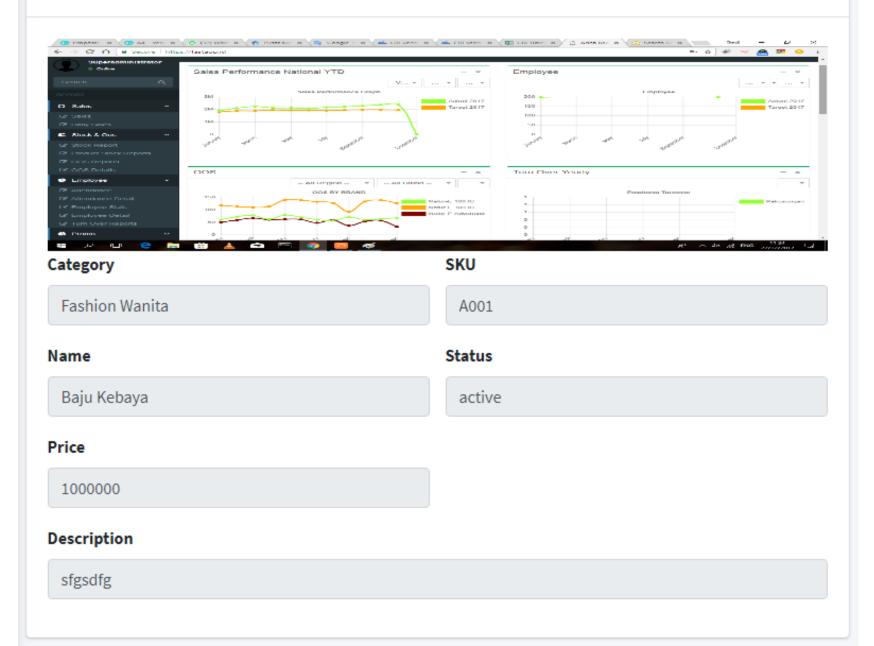


Try to simulate this! Filter to one category then navigate to next page. Is the filter state remain exists?

Task #17 Modify view file: **index.blade.php**Make filter and pagination works together



Detail Product



Task #18
Create new view file:
show.blade.php
(under product folder)



Task #18 Answer http://bit.ly/2MZP2Hz

Task #19 Modify **show** method on **ProductController**display the product

```
public function show($id)
    // ambil data product untuk id tertentu
    $product = \App\Product::find($id);
    $data = [
        'product' => $product
    ];
    return view('admin/product/show')->with($data);
```



Products

+ Add

Category

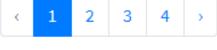
Search

All Category

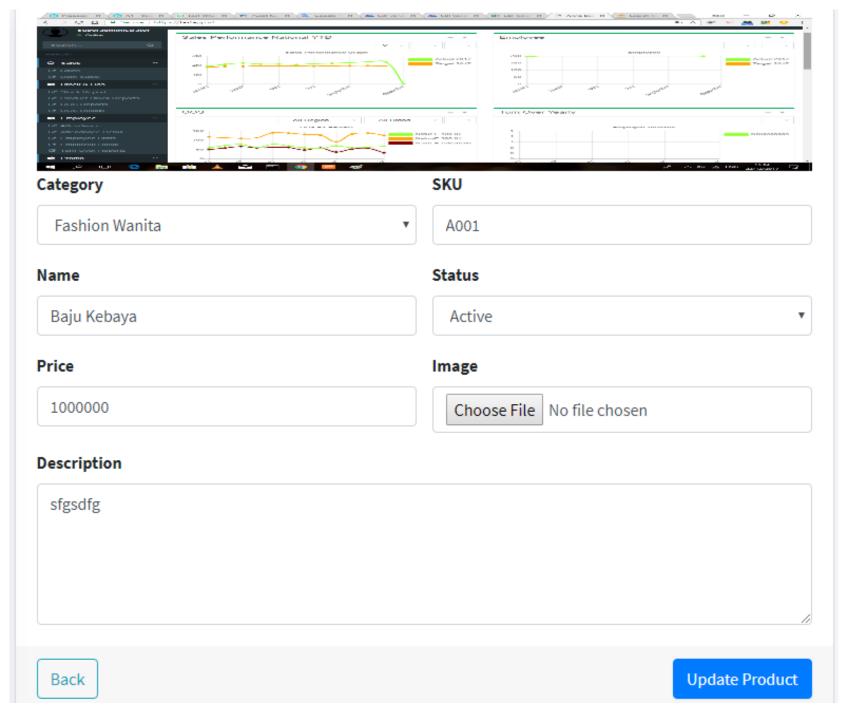
Enter keyword

ID	Category	Name	Price	SKU	Image	Status	Action
3	Fashion Wanita	Baju Kebaya	1000000	A001		Active	
4	Fashion Wanita	Kerudung	123123	A002	The	Active	
5	Fashion Pria	Celana Panjang	123123	B001		Active	

Task #20
Modify view file:
index.blade.php
Link eye button
to product show
page







Task #21
Create new view file:
edit.blade.php
(under product folder)



Task #22 Modify edit method on ProductController display the edit product page

```
public function edit($id)
    $categories = \App\Category::pluck('name', 'id');
    $product = \App\Product::find($id);
    $data = [
        'product' => $product,
        'categories' => $categories
    ];
    return view('admin/product/edit')->with($data);
```



Task #23 Modify **update** method on **ProductController**store product to database

Task #23 Answer http://bit.ly/2N1PIMX



Category Search

All Category

Enter keyword

ID	Category	Name	Price	SKU	Image	Status	Action
3	Fashion Wanita	Baju Kebaya	1000000	A001		Active	
4	Fashion Wanita	Kerudung	123123	A002		Active	
5	Fashion Pria	Celana Panjang	123123	B001		Active	

Task #24
Modify view file:
index.blade.php
Link pencil
button to product
edit page



Task #24 Modify **destroy** method on **ProductController**delete the product from database

```
public function destroy($id)
{
    // ambil dari database
    $product = \App\Product::find($id);
    $product->delete();

    return Redirect::to('admin/product');
}
```





Category Search

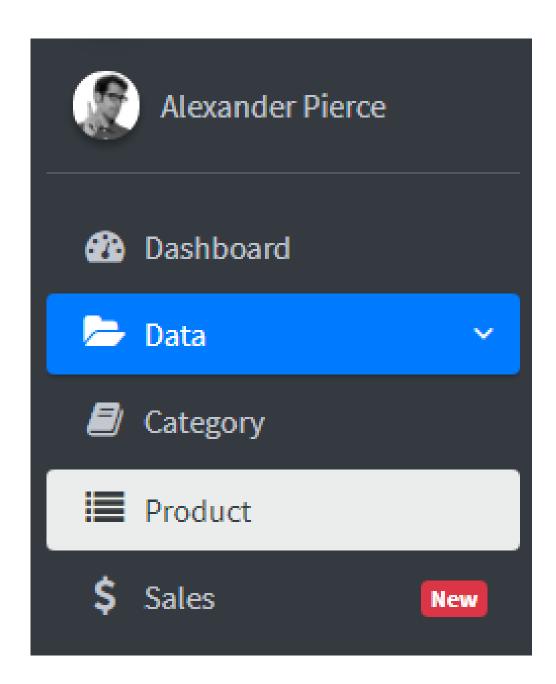
All Category

Enter keyword

ID	Category	Name	Price	SKU	Image	Status	Action
3	Fashion Wanita	Baju Kebaya	1000000	A001		Active	
4	Fashion Wanita	Kerudung	123123	A002	The control of the	Active	
5	Fashion Pria	Celana Panjang	123123	B001		Active	

Task #25
Modify view file:
index.blade.php
Make the delete
button work





Finally, modify your sidebar, link **Product** menu to **Product** Page