CRUD with Laravel

Developed by: Wai Phyo San Tin

Date: 01 May 2016

Revision History

Date	Description
01 May 2016	First Version

ဒီ project မှာ Laravel သုံးပြီး CRUD (Create, Read, Update & Delete) Operation တွေကို sample လုပ်ပြပါမယ်။

Prerequisite

- Laravel version 5.2.31 ကို သုံးထားပါတယ်။
- ဒီ tutorial ကို လိုက်လုပ်ဖို့ အတွက် PHP နှင့် Laravel ကို အခြေခံသိထားပြီးသား ဖြစ်ရင် ပိုကောင်းပါတယ်။
- PHP command line tools နှင့် bootstrap css framwork ကို သိထားပြီးသား ဖြစ်ရပါမယ်။
- Software design pattern တွေ အကြောင်း နည်းနည်းသိပြီး MVC pattern ကိုတော့ သိထားရပါမယ်။

Database Setup

- MySQL Database တစ်ခု ဆောက်ပါမယ်။ နာမည်က laravel_crud ပါ။
- Tables : categories (with sample data), products
- categories က parent table ပါ။ products က child table ပါ။

```
DROP TABLE IF EXISTS `categories`;
CREATE TABLE `categories` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

INSERT INTO `categories` (`id`, `title`) VALUES
(1, 'Stationery'),
(2, 'Electronics'),
(3, 'Children Toys'),
(4, 'Food & Beverage'),
(5, 'Clothing');

DROP TABLE IF EXISTS `products`;
CREATE TABLE IF NOT EXISTS `products` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(255) NOT NULL,
  `category_id` int(11) unsigned NOT NULL,
  `quantity` int(11) NOT NULL,
  `price` decimal(10,2) NOT NULL,
  `created_at` timestamp NOT NULL,
  `updated_at` timestamp NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_category_id` (`category_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 ;
```

Installing Laravel

ပထမဆုံး ကိုယ်သုံးတဲ့ webserver ရဲ့ root directory အောက်မှာ command line ကနေ တဆင့် Laravel ကို composer သုံးပြီး install လုပ်ပါမယ်။ ကျွန်တော်က wamp server သုံးပါတယ်။ ဒါကြောင့် c:\wamp\www အောက်ကို ဝင်ပြီး composer command ကို run ပါမယ်။

```
composer create-project laravel/laravel laravel_crud --prefer-dist 5.2.*
```

Updating .env file

"laravel_crud" folder အောက်မှာ ရှိတဲ့ ".env" ထဲမှာ ရှိတဲ့ တန်ဖိုးတွေကို အောက်ပါအတိုင်း ပြင်ပါမယ်။

```
DB_HOST=localhost
DB_DATABASE=laravel_crud
DB_USERNAME=root
DB_PASSWORD=
```

Creating Routes

"laravel_crud\app\Http\" အောက်မှာ ရှိတဲ့ "routes.php" file ထဲမှာ route တွေ အောက်ပါအတိုင်း ထပ်ထည့်ပါမယ်။

routes.php

```
// Product Listing Display
Route::get('product/index', 'ProductController@index');
// Product Create/Update Form Display
Route::get('product/setup/{productID?}', 'ProductController@setup');
// Product Create/Update POST Operation
Route::post('product/save', 'ProductController@save');
// Product Delete
Route::delete('/product/delete/{productID}',
    'ProductController@delete');
```

Code Explanation

- Route (၄) ခု ဖန်တီးပါတယ်။
- Product Display အတွက် Route တစ်ခု
- Product Create/Update Form Display အတွက် Route တစ်ခု
- Product Form POST အတွက် Route တစ်ခု
- Product Delete အတွက် Route တစ်ခု
- {productID} ဆိုတာ route parameter ပါ။ {productID?} ဆိုတာ optional parameter ပါ။ ဆိုလိုတာက အဲဒီ route မှာ parameter ပါရင်လည်းပါမယ်။ မပါတာလည်း ဖြစ်နိုင်ပါတယ်။

Creating Models

“Product” နှင့် “Category” Model ကို artisan သုံးပြီး ဖန်တီးပါမယ်။ command line ကနေ “laravel_crud” folder ထဲကို ဝင်ပါ။ ဥပမာ c:\wamp\www\laravel_crud။ ပြီးရင် artisan command run ပါ။

“Category” Model ပါ။

```
php artisan make:model Category
```

“Product” Model ပါ။

```
php artisan make:model Product
```

Model files တွေကို “laravel_crud\app” အောက်မှာ တွေ့ရပါလိမ့်မယ်။ File နာမည်တွေက “Category.php” နှင့် “Product.php” ပါ။

Model ထဲမှာ relationship တွေကိုပါ တစ်ခါတည်း define လုပ်ပါမယ်။ Category တစ်ခုမှာ Product တွေ အများကြီး ပါဝင်ပါတယ်။

“Category.php” ရဲ့ code တွေပါ။

Category Model

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Category extends Model
{
    /*
     * Defining "Relationship"
     * "Category" has Many "Products"
     */
    public function products()
    {
        return $this->hasMany(Product::class);
    }
}
?>
```

Product တစ်ခုကတော့ Category တစ်ခုမှာ belong to ဖြစ်ပါတယ်။ “Product.php” ရဲ့ code တွေပါ။

Product Model

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    /*
     * Defining "Relationship"
     * "Product" belongs to "Category"
     */
    public function category()
    {
        return $this->belongsTo(Category::class);
    }
}
?>
```

Creating Repository

Database operation တွေ အတွက် repository class တွေ ဖန်တီးပါတယ်။ repository class ဆိုတာ model နဲ့ controller ထဲမှာ database နဲ့ ပတ်သက်တဲ့ code တွေကို မထားချင်လို့ အပြင်မှာ သပ်သပ်ထုတ်ရေးတဲ့ class တွေပါ။ တခြားထွေထွေ ထူးထူးမဟုတ်ပါဘူး။

“laravel_crud\app” အောက်မှာ “Repositories” ဆိုတဲ့ folder တစ်ခုကို ဖန်တီးပါမယ်။ “Repositories” folder အောက်မှာ “CategoryRepository.php” ဆိုတဲ့ php file တစ်ခုကို ဖန်တီးပါမယ်။ Namespace က “App\Repositories” ပါ။ “CategoryRepository.php” ထဲမှာ category နဲ့ ပတ်သက်တဲ့ database operation code တွေကို ထားပါမယ်။

Category Repository

```
<?php

namespace App\Repositories;

use App\Category;

class CategoryRepository
{
    /*
     * Getting all "Categories" ordered by "Title"
     */
    public function getCategory()
    {
        return Category::orderBy('title')
            ->get();
    }
}
?>
```

getCategory() function က Database ထဲမှာ ရှိတဲ့ category တွေကို title နဲ့ order စီပြီး ဆွဲထုတ်မယ့် function ပါ။

“Repositories” folder အောက်မှာ “ProductRepository.php” ဆိုတဲ့ php file တစ်ခုကို ထပ်ပြီး ဖန်တီးပါမယ်။ Namespace က “App\Repositories” ပါ။ “ProductRepository.php” ထဲမှာ product နဲ့ ပတ်သက်တဲ့ database operation code တွေကို ထားပါမယ်။

Product Repository

```
<?php

namespace App\Repositories;

use App\Product;

class ProductRepository
{
    /*
     * Getting all "Products" together with "Category"
     * order by Product "Title"
     */
    public function getAllProduct()
    {
        //eager loading of "product" with "category"
        return Product::with('category')
            ->orderBy('title')
            ->get();
    }

    /*
     * Getting "Product" by "ProductID (Primary Key)"
     */
    public function getProductByProductID($productID)
    {
        return Product::find($productID);
    }

    /*
     * Saving/Updating "Product"
     */
    public function saveProduct($title, $categoryID,
                            $quantity, $price, $productID = null)
    {
        if (isset($productID)) {
            $product = Product::find($productID);
        } else {
            $product = new Product();
        }

        $product->title = $title;
        $product->category_id = $categoryID;
        $product->quantity = $quantity;
        $product->price = $price;

        return $product->save();
    }
```

```
/*
     * Deleting "Product"
     */
    public function deleteProduct($productID)
    {
        $product = Product::find($productID);
        return $product->delete();
    }
}
?>
```

Code Explanation

- getAllProduct() function မှာ product ရော category ပါ တစ်ခါတည်း loading တင်ပါတယ်။ Laravel မှာ eager load လို့ခေါ်ပါတယ်။
- getProductByProductID() function က product တွေကို Primary Key ဖြစ်တဲ့ id နဲ့ select လုပ်မယ့် function ပါ။
- saveProduct() function မှာ product တစ်ခု create လုပ်တာရော update လုပ်တာပါ တစ်ခါတည်း ပေါင်းရေးထားပါတယ်။ parmater မှာ productID ပါရင် update။ မပါရင် create ပါ။ program ရေးတဲ့ အခါ တတ်နိုင်သရွေ့ create and update operation ကို function တစ်ခုတည်းနဲ့ အလုပ်ဖြစ်အောင် ရေးသင့်ပါတယ်။
- deleteProduct() function ကတော့ product တွေကို delete လုပ်မယ့် function ပါ။

Creating Controller

Product Controller ကို php artisan သုံးပြီး command line ကနေ ဖန်တီးပါမယ်။

```
php artisan make:controller ProductController
```

command run ပြီးရင် Product Conroller ကို “laravel_crud\app\Http\Controllers” folder အောက်မှာ ရောက်နေတာကို တွေ့ရပါမယ်။ “ProductController.php” ပါ။ ကျွန်တော်တို့ တတွေ controller ထဲမှာ “CategoryRepository” နဲ့ “ProductRepository” တွေကို သုံးပါမယ်။ ဒါကြောင့် use statement သုံးပြီး repository နှစ်ခုကို import လုပ်ပါမယ်။ line no (၈) နှင့် (၉) မှာ ကြည့်ပါ။

အခုဆို “ProductController” က အောက်မှာ ပြထားတဲ့ အတိုင်း ဖြစ်နေပါပြီ။

Product Controller

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Http\Requests;

use App\Repositories\CategoryRepository;
use App\Repositories\ProductRepository;

class ProductController extends Controller
{

}
?>
```

- “ProductController” ထဲမှာ code တွေ ထပ်ထည့်ပါမယ်။
- “$categoryRepository” နဲ့ “$productRepository” ဆိုတဲ့ protected variables နှစ်ခု ကြေညာပါမယ်။
- constructer ရေးပါမယ်။ constructor မှာ parmeter တွေ အနေနဲ့ “CategoryRepository” နဲ့ “ProductRepository” ကို ကြေညာပါမယ်။ ကျွန်တော် အခု ရေးတဲ့ ပုံစံကို Dependencey Injection လုပ်တယ် လို့ခေါ်ပါတယ်။ Laravel Framework ရဲ့ Service Container က “CategoryRepository” နဲ့ “ProductRepository” ကို “ProductController” မှာ Dependency Inject လုပ်ပေးပါတယ်။ “ProductController” က အခုဆို repository နှစ်ခုမှာ ရှိတဲ့ function တွေကို ခေါ်သုံးလို့ရပြီ ဖြစ်ပါတယ်။
- Larvel မှာ Service container ကို သုံးပြီး Dependency Inject လုပ်တာတွေကို အသုံးများပါတယ်။ အဲဒီ အကြောင်းကို မသိသေးရင် အသေးစိတ် လေ့လာဖို့ တိုက်တွန်းလိုက်ပါတယ်။

Product Controller (Continue)

```
protected $categoryRepository;
protected $productRepository;

/*
* "Dependency Injection" with "constructor"
*/
public function __construct(CategoryRepository $categoryRepository,
ProductRepository $productRepository)
{
    $this->categoryRepository = $categoryRepository;
    $this->productRepository = $productRepository;
}
```

“ProductController” ထဲမှာ product create/update form ကို display ပြမယ့် function ကို ထပ်ထည့်ပါမယ်။

Product Controller (Continue)

```
/*
 * Rendering "Product" Add New/Update Form
 */
public function setup($productID = null)
{
    if (!isset($productID)) {
        $product = null;
    } else {
        $product = $this->productRepository-
>getProductByProductID($productID);
    }

    return view(
        'products.setup',
        [
            'categories' => $this->categoryRepository->getCategory(),
            'selectedCategoryID' => isset($product->category_id) ? $product-
>category_id : null,
            'product' => $product
        ]
    );
}
```

Code Explanation

- Function Parameter မှာ ProductID ပါတာ မပါတာကို ကြည့်ပြီး create/update opeartion ကို ဆုံးဖြတ်ပါမယ်။
- ProductID မပါရင် Form က Create ပါ။
- ProductID ပါရင် Form က Update အတွက်ပါ။ Update ဆိုရင် Database ထဲက Product Information တွေကို form ပေါ်မှာ ပြပါမယ်။ line no (၉) မှာ ProductID ပါရင် Database ထဲက Product Information တွေကို select လုပ်မယ့် code ကို ရေးထားပါတယ်။
- category information ကိုတော့ အမြဲတန်း select လုပ်ပြီး view ဆီကို ပြန်ပို့ပါမယ်။ line no (၁၅) ကိုကြည့်ပါ။

Creating View

“app.blade.php” file ကို “laravel_crud\resources\views\layouts” အောက်မှာ ဖန်တီးပါမယ်။ “layouts” folder မရှိရင် create လုပ်ပါ။ “app.blade.php” ရဲ့ code တွေကို အောက်မှာ ပြထားပါတယ်။ bootstrap css framework ကို သုံးမှာပါ။ bootstrap ရဲ့ content delivery network (CDN) ကနေပဲ css file ကို တန်းယူပါမယ်။

app.blade.php

```
<html>
<head>
    <title>Laravel - CRUD</title>
    <!-- CSS And JavaScript -->
    <link rel="stylesheet"

href="{{asset('https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap
.min.css')}}">
</head>

<body>
    @yield('content')
</body>
</html>
```

“laravel_crud\resources\views” folder အောက်မှာ “products” folder ကို create လုပ်ပါမယ်။ အဲဒီ folder အောက်မှာ “setup.blade.php” file ကို ဖန်တီးပါမယ်။ သူ့အတွက် code တွေက အောက်မှာ ပြထားပါတယ်။

setup.blade.php

```
@extends('layouts.app')

@section('content')

    <div class="row">
        <div class="col-md-offset-3 col-md-6">
            <h3>Product Form</h3>
            <form class="form-horizontal" action="{{ url('/product/save')
}}" method="post">
                {{ csrf_field() }}
                <!-- We will add more code here later -->
            </form>
        </div>
    </div>

@endsection
```

အပေါ်က code မှာ နောက်ထပ် code တွေ ထပ်ထည့်ဖို့နေရာ ချန်ထားပါတယ်။ အဲဒီနေရာမှာ code တွေ ထပ်ထည့်ပါမယ်။

setup.blade.php (continue)

```
@if (isset($product->id))
    <input type="hidden" id="product_id" name="product_id" value={{ $product-
>id }} />
@endif

<div class="form-group">
    <label for="title" class="col-sm-2 control-label">Title</label>
    <div class="col-sm-10">
        <input type="text" class="form-control" id="title" name="title"
placeholder="title" value="{{ isset($product->title) ? $product->title :
''}}">
    </div>
</div>

<div class="form-group">
    <label for="category" class="col-sm-2 control-label">Category</label>
    <div class="col-sm-10">
        <select class="form-control" id="category_id" name="category_id">
            @foreach ($categories as $category)
```

```
                <option value={{ $category['id'] }} {{
$selectedCategoryID==$category['id'] ? "selected='selected'" : '' }}>
                        {{ $category['title'] }}
                </option>
            @endforeach
        </select>
    </div>
</div>

<div class="form-group">
    <label for="quantity" class="col-sm-2 control-label">Quantity</label>
    <div class="col-sm-10">
        <input type="text" class="form-control" id="quantity" name="quantity"
placeholder="quantity" value="{{ isset($product->quantity) ? $product-
>quantity : ''}}">
    </div>
</div>

<div class="form-group">
    <label for="price" class="col-sm-2 control-label">Price</label>
    <div class="col-sm-10">
        <input type="text" class="form-control" id="price" name="price"
placeholder="price" value="{{ isset($product->price) ? $product->price :
''}}">
    </div>
</div>

<div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
        <button type="submit" class="btn btn-primary">Save</button>
    </div>
</div>
```

Code Explanation

- form တစ်ခုတည်းတဲ့ create/update ကို ရအောင် ရေးထားတာပါ။
- product_id က hidden value ပါ။ သူနဲ့ form ရဲ့ create/update mode ကို ဆုံးဖြတ်မှာပါ။
- title, quantity, price တွေက text box တွေပါ။ update mode ဆိုရင် value ကို ဖြည့်ပါမယ်။
- category_id ကတော့ dropdown list ပါ။ update mode ဆိုရင် selected value ကို ဖြည့်ပါမယ်။
- အခုပြီး သလောက်ကို run ကြည့်လို့ရပါပြီ။ ကျွန်တော်က wamp သုံးတော့ virtual host ဆောက်ပြီး run ပါတယ်။ ကျွန်တော့် virtual host sample က ဒီလိုပါ။

```
<VirtualHost *:80>
    ServerAdmin wpst@laravel_crud
    DocumentRoot "C:\wamp\www\laravel_crud\public"
    ServerName laravel.crud
</VirtualHost>
```

http://laravel.crud/product/setup ဆိုပြီး browser မှာ run ကြည့်ပါမယ်။ အောက်မှာ ပြထားတဲ့ အတိုင်းတွေ့ရပါမယ်။

Product Index Page

ProductController ထဲမှာ index() function ထပ်ထည့်ပါမယ်။

ProductController.php

```
public function index()
{
    return view('products.index');
}
```

“laravel_crud\resources\views\products” folder အောက်မှာ “index.blade.php” file ကို ဖန်တီးပါမယ်။ သူ့အတွက် code တွေက အောက်မှာ ပြထားပါတယ်။

index.blade.php

```
@extends('layouts.app')

@section('content')

    <div class="row">
        <div class="col-md-offset-2 col-md-8">
            <h3>Product Listing</h3>

            <!-- Link for adding new "Product" -->
            <a href="/product/setup" class="btn btn-primary">Add New
Product</a>
        </div>
    </div>
@endsection
```

Inserting New Product

Product တွေကို Create/Update လုပ်တဲ့ function လေးတစ်ခုကို ProductController ထဲမှာ ထပ်ထည့်ပါမယ်။

ProductController.php

```
/*
 * Accepting the Form POST of adding/updating "Product"
 */
public function save(Request $request)
{
    $this->productRepository->saveProduct(
        $request->title, $request->category_id,
        $request->quantity, $request->price, $request->product_id);

    // Displaying Message
    \Session::flash('message', 'Product ' . $request->title . ' is
successfully saved.');
    // Redirection to Product Display Page
    return \Redirect::to('/product/index');
}
```

Code Explanation

- Product ကို save လုပ်ပါတယ်။
- Flash Session ကိုသုံးပြီး Success Message ကို ပြပါတယ်။
- index page ကို redirect ပြန်လုပ်ပါတယ်။

Displaying Message

User ကို Message display ပြဖို့ အတွက် “message.blade.php” ဆိုတဲ့ view file ကို “laravel_crud\resources\views\common” folder အောက်မှာ ဖန်တီးပါမယ်။ “common” folder ဆောက်ပေးရပါမယ်။

message.blade.php

```
<div class="row">
    <div class="col-md-offset-2 col-md-8">
        @if (Session::has('message'))
        <!-- Success Message -->
        <div class="alert alert-success">
            <strong>Success</strong>
            <br>
            {{ Session::get('message') }}
        </div>
        @endif
    </div>
</div>
```

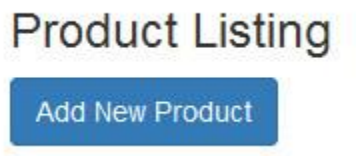
“message.blade.php” file က ဘာလုပ်သလဲဆိုတော့ flash session ထဲမှာ message ရှိရင် ပြပါတယ်။ flash session ကို သုံးထားတဲ့ အတွက် တစ်ခါ message ပြပြီးရင် ပျောက်သွားမှာပါ။ “message.blade.php” file ကို “app.blade.php” ထဲမှာ include လုပ်ပါမယ်။ အောက်မှာပြထားတဲ့ code ရဲ့ line no (၂) နှင့် (၃) ကို ကြည့်ပါ။

app.blade.php

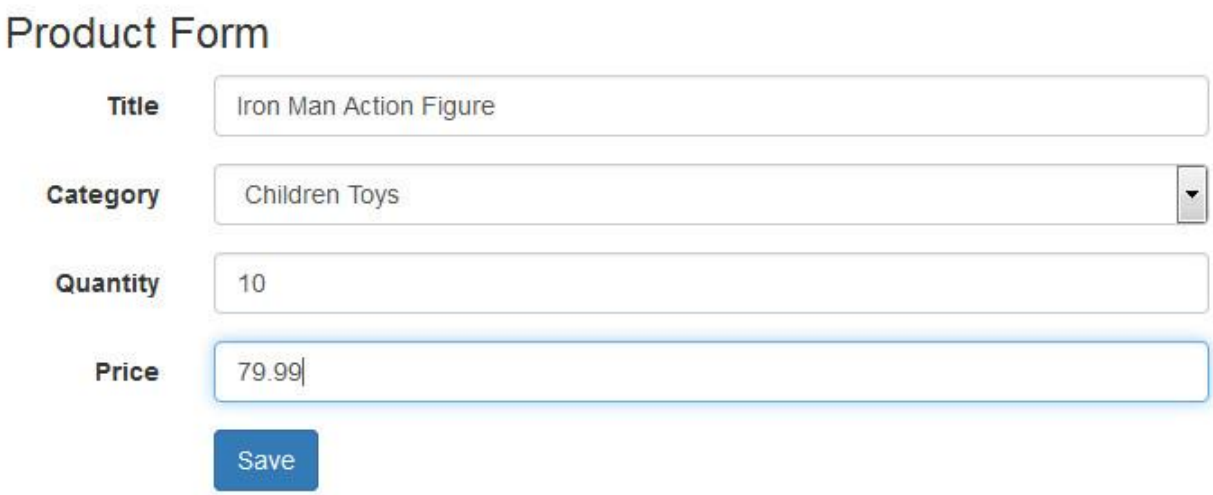
```
<body>
    <!-- Displaying Message -->
    @include('common.message')

    @yield('content')
</body>
```

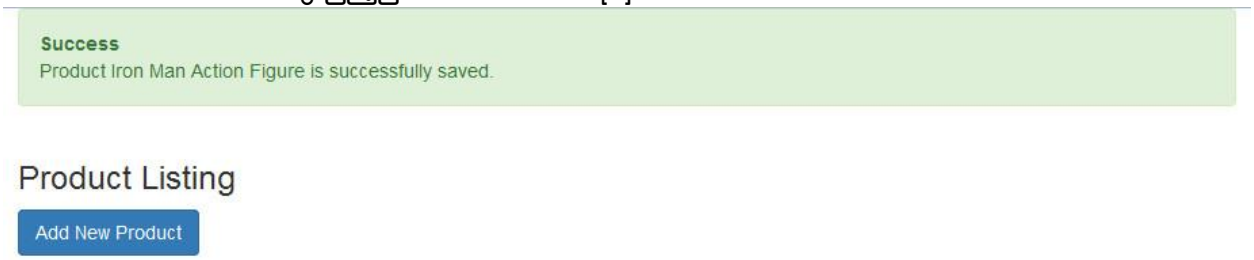
http://laravel.crud/product/index ဆိုပြီး browser မှာ run ကြည့်ပါမယ်။ အောက်မှာ ပြထားတဲ့ အတိုင်းတွေ့ရပါမယ်။



“Add New Product” Button ကို နှိပ်ပြီး product အသစ် ထည့်လို့ရပါပြီ။



Product Information တွေ ဖြည့်ပြီး Save button ကို နှိပ်ပါ။



Index page ကို ပြန်ရောက်လာတာကို တွေ့ရပါမယ်။ Save success message လည်းပြပါတယ်။

Displaying Products

Product တွေကို display ပြဖို့ အတွက် ProductController.php ထဲက index() function ကို အောက်ပါအတိုင်း ပြန်ပြင်ပါမယ်။

ProductController.php

```
/*
 * Displaying "Products" and their "Categories"
 */
public function index()
{
    $products = $this->productRepository->getAllProduct();
    return view('products.index', ['products' => $products]);
}
```

Product တွေကို select လုပ်ပြီး product index view ကို ပြန်ပေးလိုက်တာပါ။ Product တွေကို display ပြဖို့ အတွက် “index.blade.php” ကိုလည်း ပြန်ပြင်ပါမယ်။ product data တွေကို table သုံးပြီး ပြပါမယ်။ နောက်ဆုံးဖြစ်လာမယ့် index.blade.php file ကို အောက်မှာ ပြထားပါတယ်။

index.blade.php

```
@extends('layouts.app')

@section('content')

    <div class="row">
        <div class="col-md-offset-2 col-md-8">
            <h3>Product Listing</h3>

            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Title</th>
                        <th>Category</th>
                        <th>Quantity</th>
                        <th>Price</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
                @foreach ($products as $product)
                    <tr>
                        <td>{{ $product->title }}</td>
                        <td>{{ $product->category->title }}</td>
                        <td>{{ $product->quantity }}</td>
                        <td>{{ $product->price }}</td>
                        <td>
                            <!-- We will add Edit and Delete button here
later -->
                        </td>
                    </tr>
                @endforeach
                </tbody>
            </table>

            <!-- Link for adding new "Product" -->
            <a href="/product/setup" class="btn btn-primary">Add New
Product</a>
        </div>
    </div>

@endsection
```

Product နည်းနည်း လောက်ထပ်ထည့်ပါ။ ပြီးရင် http://laravel.crud/product/index ကို browser မှာ run ကြည့်ပါမယ်။ အောက်မှာ ပြထားတဲ့ အတိုင်းတွေ့ရပါမယ်။

Product Listing

Title	Category	Quantity	Price
Coca Cola	Food & Beverage	100	0.99
Iron Man Action Figure	Children Toys	10	79.99
Nike Pants	Clothing	20	29.99
Sony TV	Electronics	10	499.99
Uniball Ballpen	Stationery	20	1.99

Add New Product

Updating & Deleting Product

ProductController ထဲမှာ delete အတွက် function ထပ်ထည့်ပါမယ်။ update အတွက်က function ထပ်ထည့်စရာ မလိုတော့ပါဘူး။ အရင်တုန်းက ရေးထားတဲ့ save() function တစ်ခုတည်းနဲ့ အလုပ်ဖြစ်ပါတယ်။ အဲဒီလို create ရော update ပါ function တစ်ခုတည်းနဲ့ ရအောင် အစကတည်းက ရေးထားတာပါ။

ProductController.php

```
/*
 * Deleting the "Product"
 */
public function delete($productID)
{
    $this->productRepository->deleteProduct($productID);

    // Displaying Message
    \Session::flash('message', 'Product is successfully deleted.');
    // Redirection to Product Display Page
    return \Redirect::to('/product/index');
}
```

“index.blade.php” file ကိုလည်း ပြင်ပါမယ်။ “index.blade.php” ထဲမှာ “Edit” နဲ့ “Delete” button အတွက် နေရာ ချန်ထားပါတယ်။ အဲဒီနေရာမှာ အခု ပြထားတဲ့ code ကို ထည့်ပါမယ်။

index.blade.php

```
<!-- Edit Link -->
<a href="{{ url('product/setup/' . $product->id ) }}" class="btn btn-
info">Edit</a>
<!-- Delete Button -->
<form action="{{ url('product/delete/'.$product->id) }}" method="POST"
style="margin-bottom:0px;display:inline;">
    {!! csrf_field() !!}
    {!! method_field('DELETE') !!}
    <button type="submit" id="delete-product-{{ $product->id }}" class="btn
btn-danger">Delete</button>
</form>
```

http://laravel.crud/product/index ဆိုပြီး browser မှာ run ကြည့်ပါမယ်။ အောက်မှာ ပြထားတဲ့ အတိုင်းတွေ့ရပါမယ်။

Product Listing

Title	Category	Quantity	Price	
Coca Cola	Food & Beverage	100	0.99	Edit Delete
Iron Man Action Figure	Children Toys	10	79.99	Edit Delete
Nike Pants	Clothing	20	29.99	Edit Delete
Sony TV	Electronics	10	499.99	Edit Delete
Uniball Ballpen	Stationery	20	1.99	Edit Delete

Add New Product

“Edit” နဲ့ “Delete” အလုပ်လုပ်နေပါပြီ။ စမ်းကြည့်လို့ ရပါတယ်။

Complete Code Links

- GIT URL : https://github.com/wpst006/laravel_crud.git
- ZIP File On Mediafire : http://www.mediafire.com/download/45i8ce7i7x16x83/laravel_crud.zip

Read Online

- Part I : https://wpst006.wordpress.com/2016/05/02/crud-with-laravel-part-i/
- Part II : https://wpst006.wordpress.com/2016/05/02/crud-with-laravel-part-ii/