



laravel

Oleh : Rahmat Fauzi, S.T.,M.T

Who Am I ?

❖ Graduated from:

- ITS Surabaya (Bachelor of Electrical Engineering, 2008-2012)
- ITS Surabaya (Master of Electrical Engineering, 2013-2015)

❖ Lecturer System Information Telkom University

❖ Member Profesional : IEEE and APTIKOM.

❖ Xsysware Academy Founder.

❖ Contact:

- Room: Karang Building C.202 FRI Telkom University
- Email: rahmatfauzi9013@gmail.com
- Phone: 0812 2025 6700

Apa itu Laravel ?

Laravel adalah Kerangka kerja (Framework) Bahasa pemrograman PHP (Hypertext Preprocessor) berbasis open source dengan konsep MVC (Model View Controller).

Framework Laravel **bertujuan** meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu.

Laravel Menggunakan **GITHUB** sebagai tempat untuk berbagi kode.

MVC Sendiri adalah sebuah pendekatan software yang memisahkan beberapa komponen aplikasi yakni komponen **manipulasi data**, komponen **controller dan** komponen **user interface**.

Bagaimana Arsitektur Laravel ?

Setelah kita mengetahui konsep Laravel Menggunakan MVC (model view controller).
Jika dirinci lebih detail fungsi dari masing – masing komponen tersebut adalah :

1. Model

Komponen yang berfungsi mengelola dengan sumber data dan logika data.

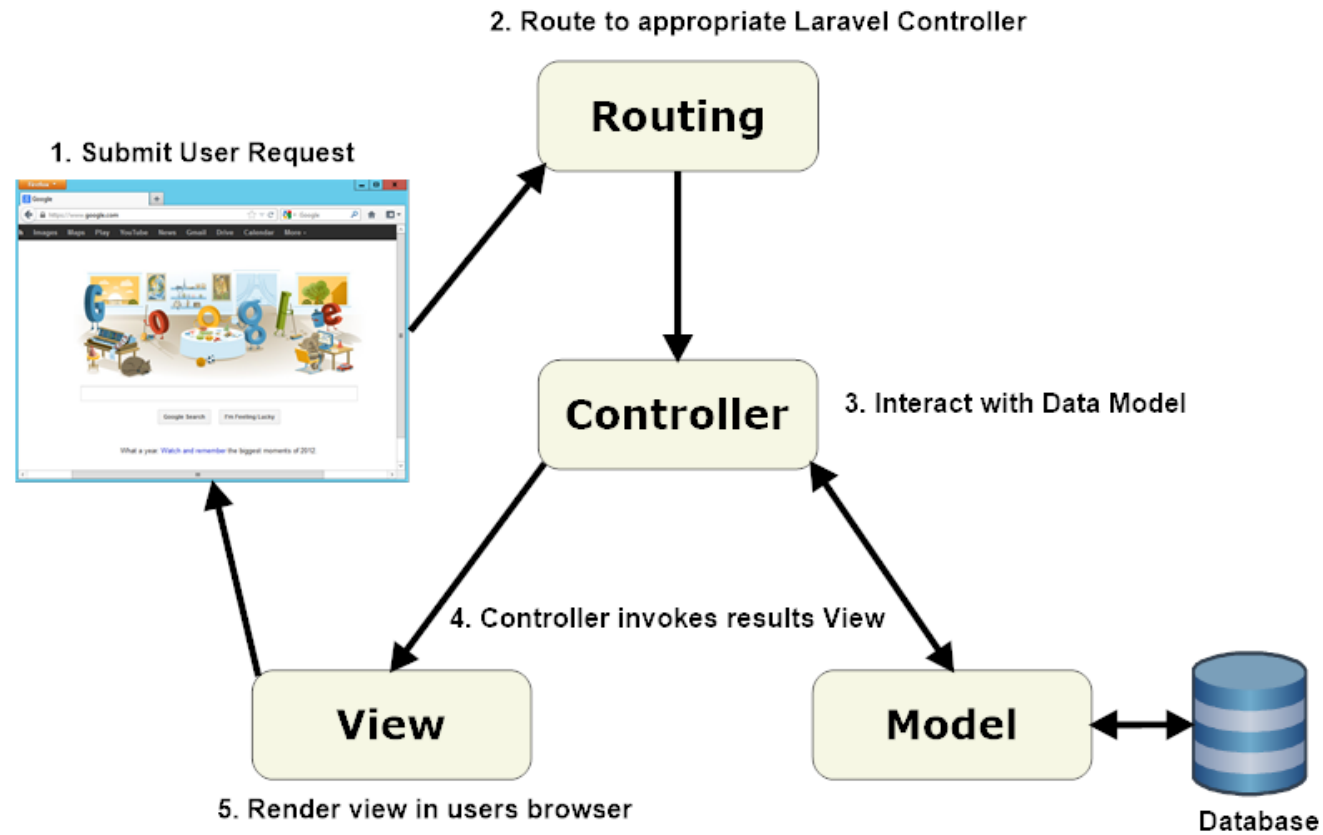
2. View

Komponen yang berfungsi membuat tampilan

3. Controller

Komponen yang berfungsi menerima input (request) dan memberikan output (response) data.

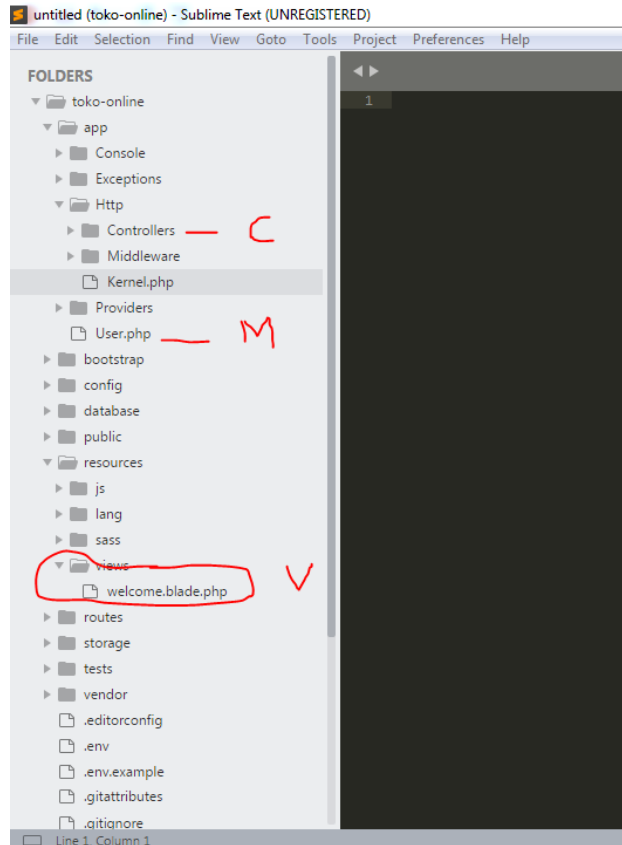
Visualisasi MVC Laravel



Penjelasan MVC

1. User mengakses aplikasi melalui route tertentu
2. Route tersebut oleh aplikasi telah dipetakan ke controller action
3. Controller action akan menggunakan model untuk mengakses data. Atau langsung mengembalikan view tanpa data (langsung ke step 5)
4. Model berinteraksi ke database untuk mendapatkan data atau menyimpan data
5. Setelah berhasil mendapatkan data melalui model, controller akan mengembalikan sebuah view
sekaligus data jika ada.
6. View tersebut pada akhirnya yang dilihat oleh user

MVC telah disediakan folder khusus di aplikasi laravel kecuali model.



1. Controller berada di folder app/Http/Controllers

2. View berada di folder resources/views

View berfungsi untuk menaruh kode tampilan ke pengguna aplikasi. Di file view ini lah kita letakkan kode html, css dan javascript bukan di controller, route atau model. File view bisa mengakses variable yang dilempar dari controller action seperti pada bahasan Controller.

Pada aplikasi Laravel baru, tersedia satu file view yaitu welcome.blade.php, silahkan dibuka pada resources/views/welcome.blade.php. File tersebut berisi kode html, css, javascript dan beberapa sintaks blade.

3. Khusus model tidak memiliki folder khusus, tetapi kita bisa meletakkannya di folder app, atau di folder lainnya sesuai kebutuhan.

Pengenalan Routing

Route

Route jika diterjemahkan bebas memiliki makna **Rute / Jalur**.

Sedangkan dalam Laravel, Route bisa diartikan **jalur URL yang bisa diakses oleh pengguna aplikasi dan ke mana jalur itu diproses**.

Misalkan routing /helo yang bisa diakses di http:localhost/helo yang menampilkan string "Hello World"

Sintak paling dasar dari routing di Laravel adalah sebagai berikut:

```
Route::verb("/path", callback);
```

Route::verb("/path", callback);

Kode di atas dapat kita jelaskan sebagai berikut :

1. Facade Route.

2. Verb

Verb merupakan HTTP verb, terdiri dari get, post, put, delete, options, patch.

3. Path

Merupakan path URL setelah domain aplikasi kita yang kita izinkan untuk diakses oleh pengguna aplikasi.

4. Callback

Callback bisa berupa *closure* function atau Controller action yang ingin kita eksekusi ketika path tertentu diakses oleh pengguna aplikasi.

Closure function merupakan istilah lain untuk function yang tidak memiliki nama dan seringkali digunakan sebagai callback. Jangan bingung ya.

```
Route::get("/helo", function(){  
    return "Hello World";  
});
```

Route di atas bisa kita baca seperti ini, ketika pengguna mengakses URL `http://toko-online.test/helo`, eksekusi **closure function** di parameter kedua, closure function tersebut menghasilkan string "Hello World". Sangat sederhana dan mudah dicerna, sehingga pengguna akan mendapatkan string Hello World di layar browser mereka.

Akan tetapi penggunaan closure function sebagai route callback jarang sekali kita pakai dalam pembuatan aplikasi sesungguhnya, karena kita memiliki controller.

Ingat, callback selain berupa closure function bisa juga kita isi dengan action dari controller. Sehingga misalnya kita memiliki `WelcomeController` dengan method `beriSalam`, kita bisa menulis ulang Route kita sebagai berikut

```
Route::get("/hello", "WelcomeController@beriSalam");
```

Dengan pemisalan method `beriSalam` akan menghasilkan "Hello World", maka route di atas akan menghasilkan string yang sama ke user seperti halnya ketika menggunakan definisi route yang kita tulis sebelumnya.

Route ke controller perlu diketahui terlebih dahulu, supaya kita paham saat membahas controller, untuk apa menulis controller dan action yang ada di dalamnya. Yaitu untuk kita isikan sebagai *callback* di router, sebagian besar penggunaan controller adalah untuk hal tersebut.

Di mana kita tuliskan definisi **routing**?

Definisi route : Untuk jalur Website Standar, kita bisa menulis di file **routes/web.php**

Sedangkan untuk penggunaan **web service / api service** bisa menuliskan **di routes/api.php**

Sedangkan, Jika kita ingin membuka jalur akses melalui command line kita bisa menggunakan **routes/console.php**

Adapun routes/channel.php dapat kita gunakan untuk jalur akses broadcast melalui websocket.

Umumnya aplikasi yang kita buat, cukup dengan **routes/web.php dan routes/api.php**. Bahkan jika aplikasi kita tidak perlu menyediakan web servie, kita hanya perlu menggunakan **routes/web.php** saja.

Pengenalan Model

Definisi Model ?

Model dalam laravel mempunyai makna **entitas yang berinteraksi dengan sumber data.**

Model **berfungsi** untuk query ke database, insert data baru, update, atau hapus record database.

Semua itu dilakukan dengan ORM (Object Relational Mapping) sehingga pada banyak kasus, kita tidak perlu menuliskan kode SQL secara langsung, akan tetapi langsung menggunakan method bawaan dari ORM seperti, **find, findOrFail, create, update**, dll.

Dan ORM bawaan yang dipakai oleh Laravel adalah **Eloquent.**

Membuat Model

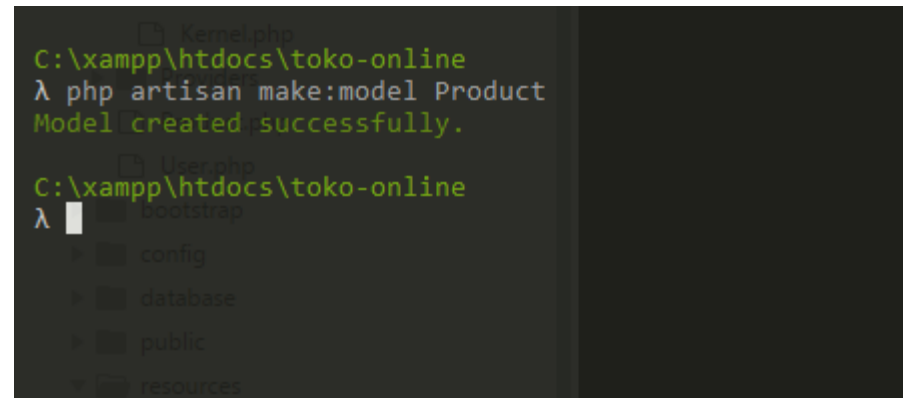
Untuk membuat model ketikkan perintah berikut ini pada **terminal**

```
php artisan make:model NamaModel
```

Misalkan

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

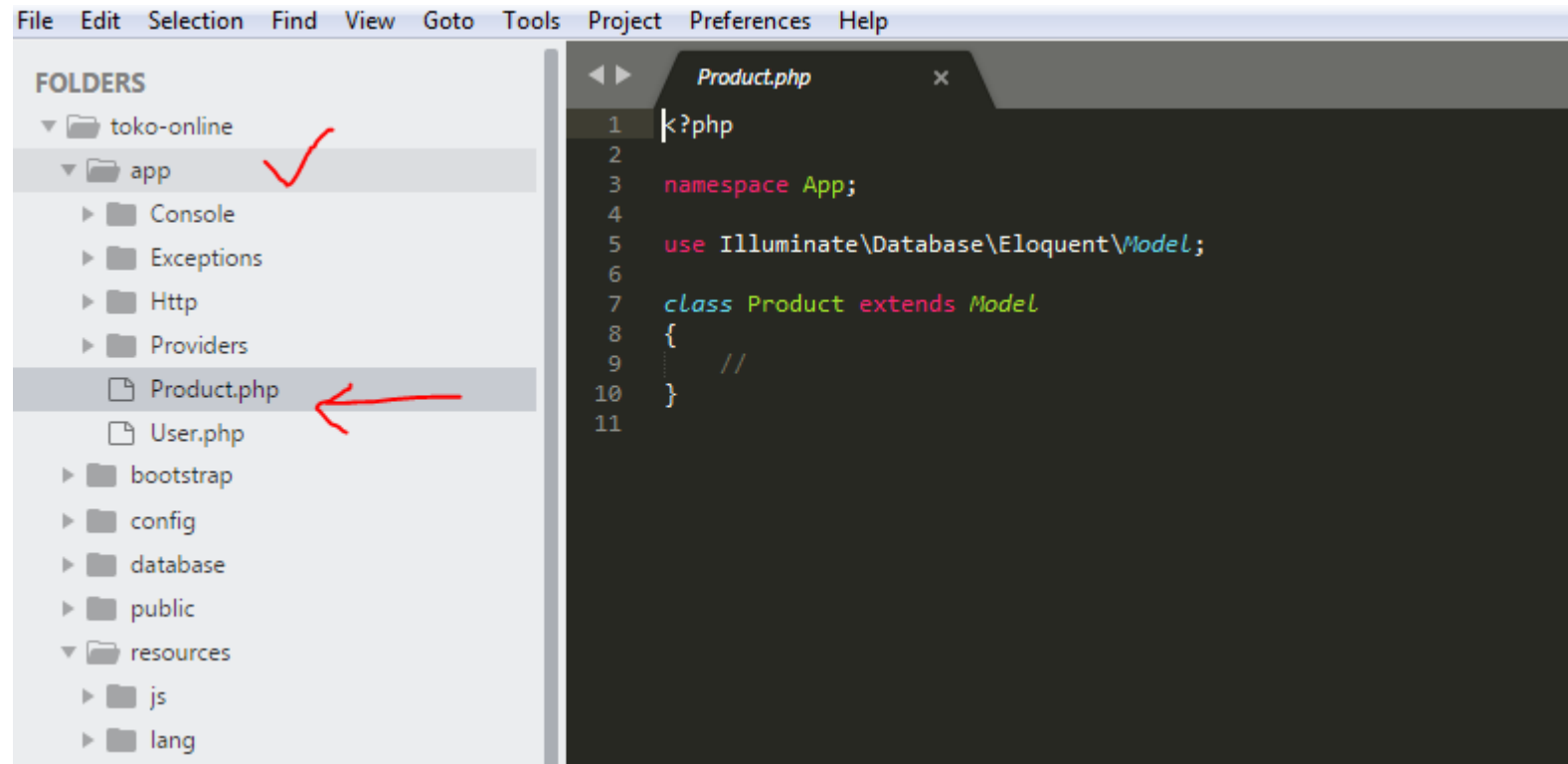

Bisa kita tulis seperti ini di terminal



```
C:\xampp\htdocs\toko-online  
λ php artisan make:model Product  
Model created successfully.  
C:\xampp\htdocs\toko-online  
λ
```

The screenshot shows a terminal window with a dark background. The prompt is a lambda symbol (λ). The command executed is 'php artisan make:model Product', and the output is 'Model created successfully.'. Below the command, there is a list of files and directories: kernel.php, User.php, bootstrap, config, database, public, and resources.

Akan muncul **product.php** di submenu **app**

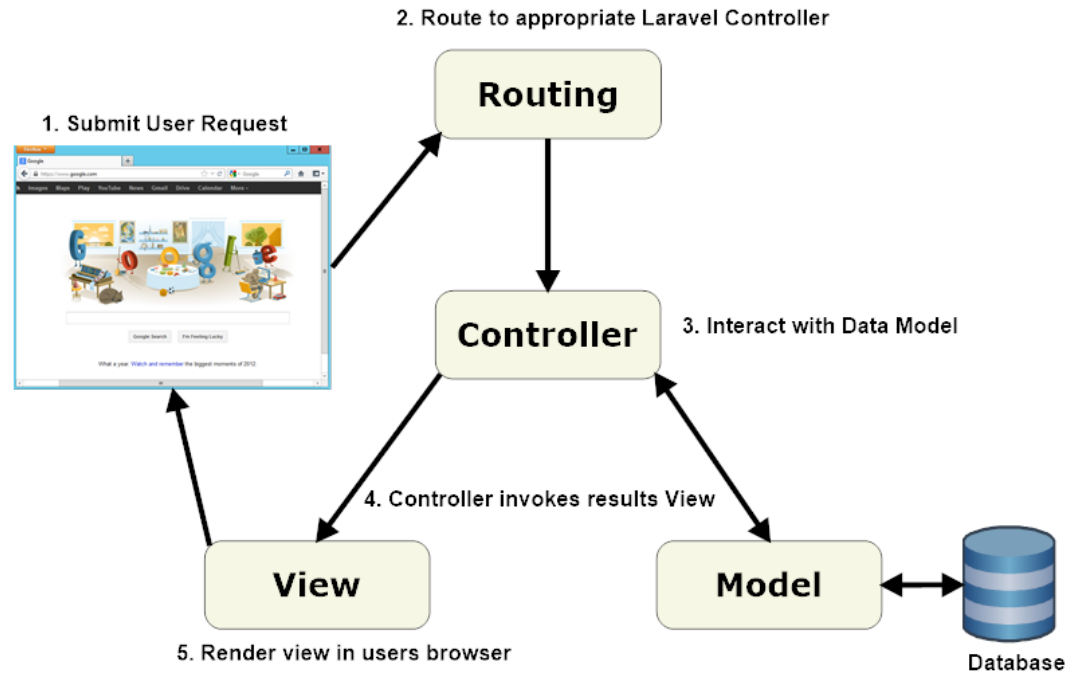


```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▼ toko-online
  ▼ app ✓
    ▶ Console
    ▶ Exceptions
    ▶ Http
    ▶ Providers
    Product.php ←
    User.php
    ▶ bootstrap
    ▶ config
    ▶ database
    ▶ public
  ▼ resources
    ▶ js
    ▶ lang
Product.php x
1 |<?php
2
3 namespace App;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 class Product extends Model
8 {
9     //
10 }
11
```

Pengenalan Controller

Controller

Di bagian **Pengenalan Model**, kita sudah membuat Model **Product**. Akan tetapi Model Product belum **bisa diakses** oleh User karena belum ada **Controller**.



Controller

Di bagian **Route**, kita ingat ada dua method pemanggilan ada **Closure route** dan Method **Callback route**.

```
Route::get("/helo", function(){  
    return "Hello World";  
});
```

```
Route::get("/products", "WelcomeController@index");
```

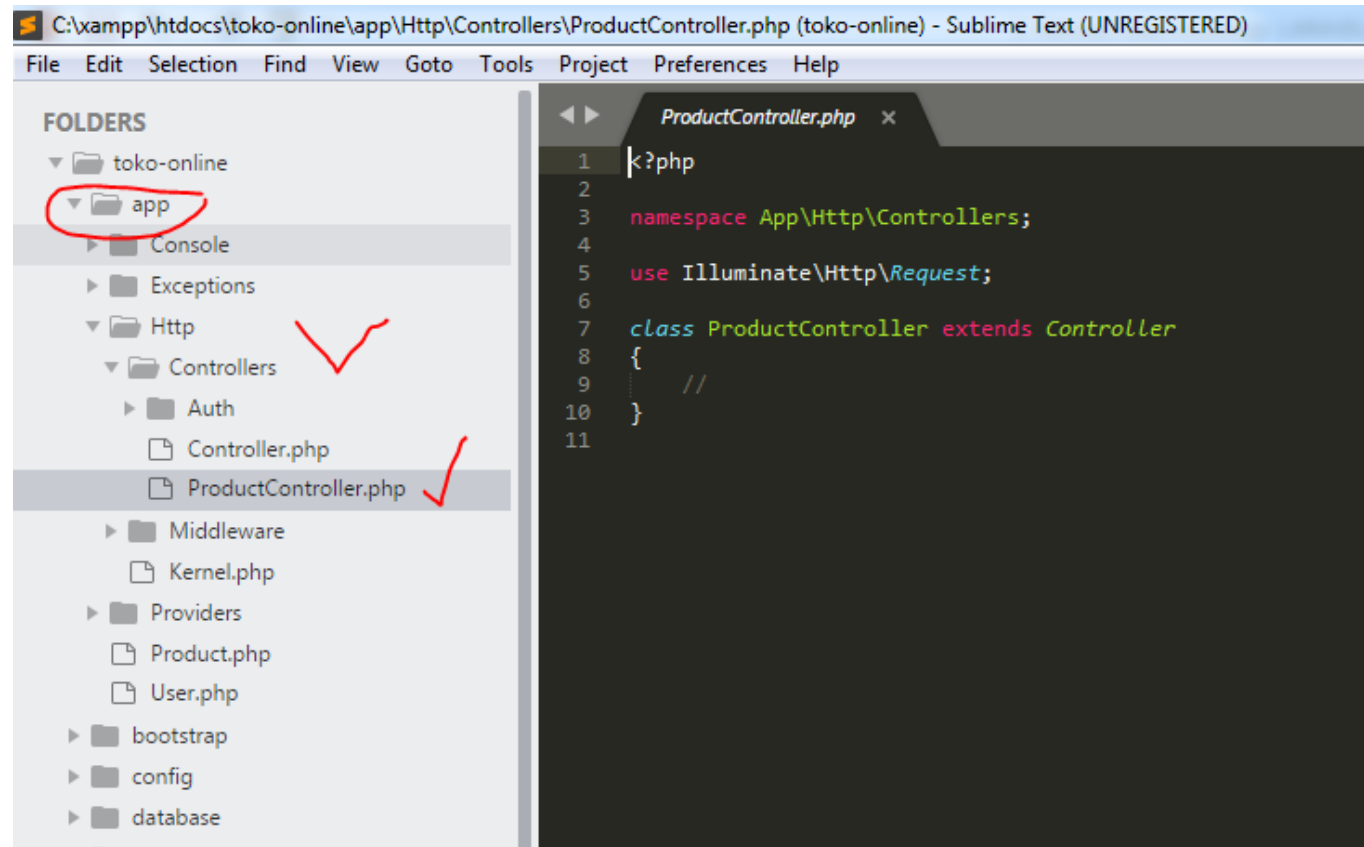
Membuat controller

Untuk membuat controller melalui artisan generator, masuk ke workspace, lalu masuk ke folder project kita, yaitu toko-online, lalu ketik perintah berikut

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

```
C:\xampp\htdocs\toko-online  
λ php artisan make:controller ProductController  
Controller created successfully.  
  
C:\xampp\htdocs\toko-online  
λ |
```

Sekarang kita memiliki file pada **app/Http/Controllers/ProductController.php**.



The screenshot shows the Sublime Text editor interface. The left sidebar displays the 'FOLDERS' panel with the following structure:

- toko-online
 - app
 - Console
 - Exceptions
 - Http
 - Controllers
 - Auth
 - Controller.php
 - ProductController.php
 - Middleware
 - Kernel.php
 - Providers
 - Product.php
 - User.php
 - bootstrap
 - config
 - database

Red annotations include a circle around the 'app' folder, a checkmark next to the 'Controllers' folder, and another checkmark next to the 'ProductController.php' file.

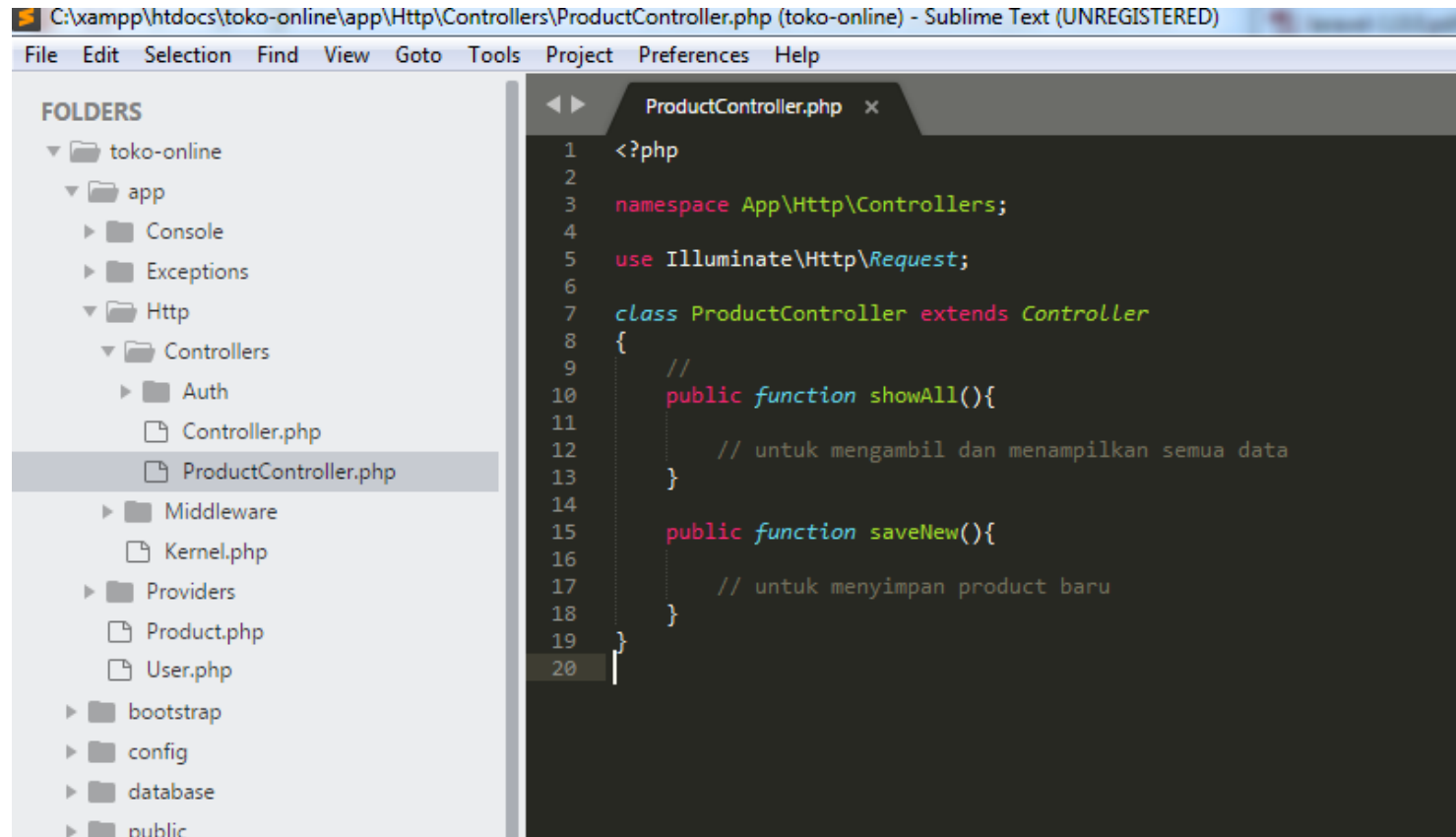
The main editor window shows the code for 'ProductController.php':

```
1 |<?php
2 |
3 | namespace App\Http\Controllers;
4 |
5 | use Illuminate\Http\Request;
6 |
7 | class ProductController extends Controller
8 | {
9 |     //
10 | }
11 |
```

Menambahkan Controller action

Controller action merupakan method yang berfungsi untuk melakukan operasi logika. Dalam operasi logika tersebut kita bisa mengambil, mengupdate, menghapus data melalui model lalu mengembalikan sebuah response kepada pengguna aplikasi, baik berupa JSON atau berupa view:

Menambahkan Controller action (Contoh)

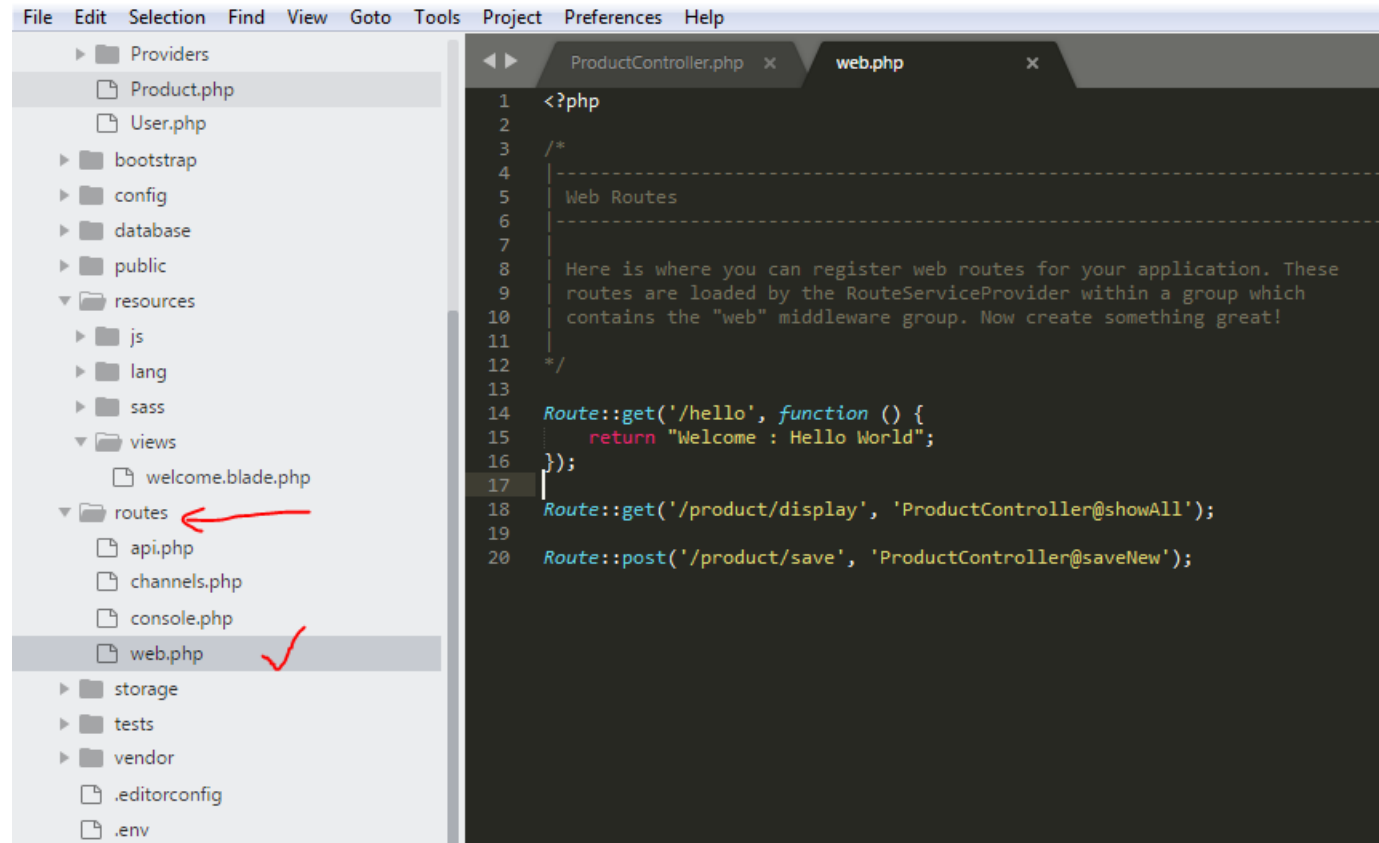


```
C:\xampp\htdocs\toko-online\app\Http\Controllers\ProductController.php (toko-online) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
  ▼ toko-online
    ▼ app
      ► Console
      ► Exceptions
      ▼ Http
        ▼ Controllers
          ► Auth
            Controller.php
            ProductController.php
          ► Middleware
            Kernel.php
          ► Providers
            Product.php
            User.php
        ► bootstrap
        ► config
        ► database
        ► public

ProductController.php x
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductController extends Controller
8 {
9     //
10    public function showAll(){
11
12        // untuk mengambil dan menampilkan semua data
13    }
14
15    public function saveNew(){
16
17        // untuk menyimpan product baru
18    }
19 }
20
```

Menambahkan Route

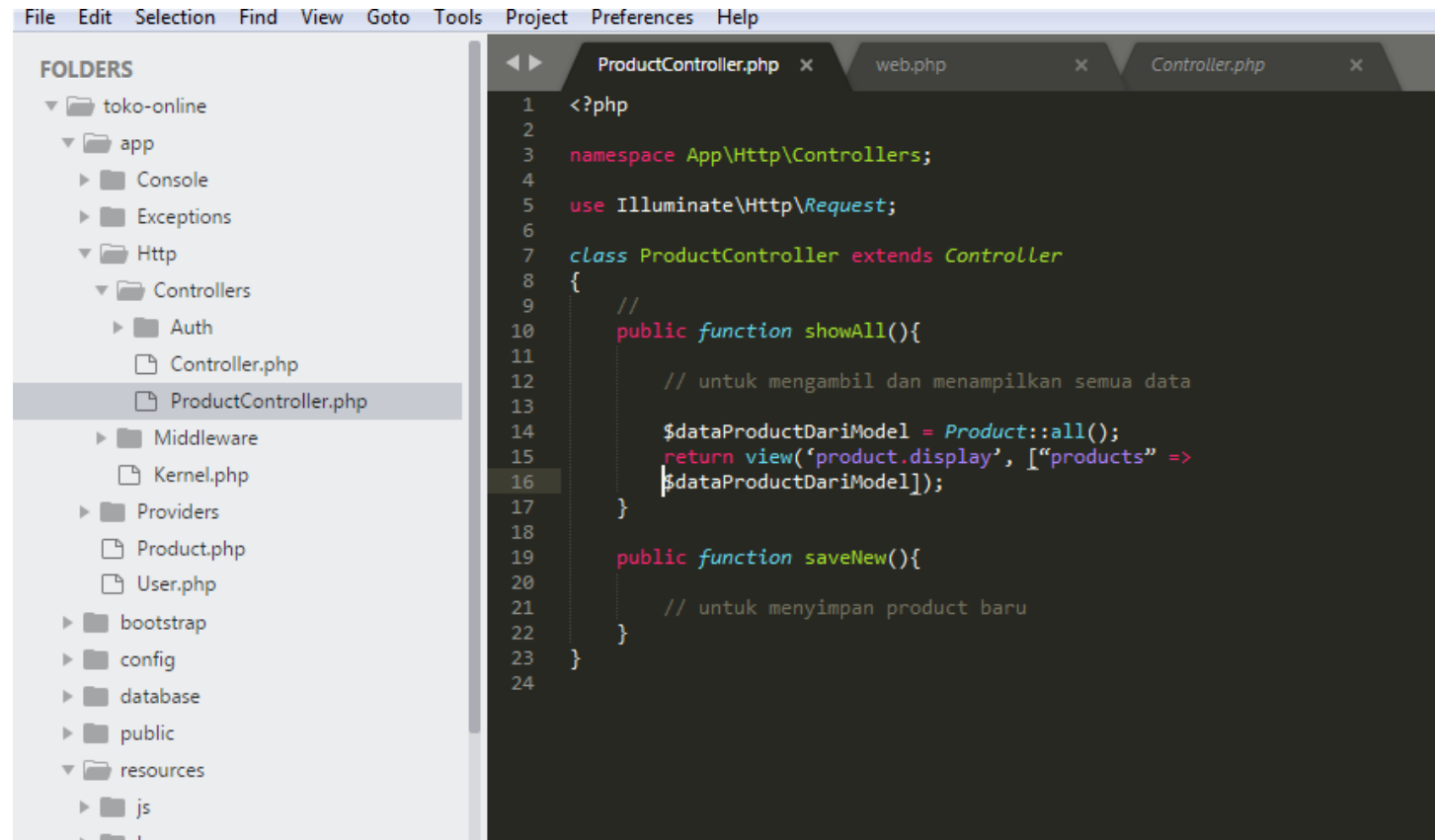


The screenshot shows an IDE interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like Providers, bootstrap, config, database, public, resources, storage, tests, and vendor. The 'resources' folder is expanded, showing subfolders like js, lang, sass, and views. The 'routes' folder is also expanded, showing files like api.php, channels.php, console.php, and web.php. The 'web.php' file is selected and highlighted with a red checkmark. The code editor shows the contents of 'web.php', which includes a PHP opening tag, a comment block for 'Web Routes', and three route definitions: a GET route for '/hello', a GET route for '/product/display', and a POST route for '/product/save'.

```
File Edit Selection Find View Goto Tools Project Preferences Help
Providers
  Product.php
  User.php
bootstrap
config
database
public
resources
  js
  lang
  sass
  views
    welcome.blade.php
  routes ←
    api.php
    channels.php
    console.php
    web.php ✓
storage
tests
vendor
.editorconfig
.env
```

```
1 <?php
2
3 /*
4  |-----
5  | Web Routes
6  |-----
7  |
8  | Here is where you can register web routes for your application. These
9  | routes are loaded by the RouteServiceProvider within a group which
10 | contains the "web" middleware group. Now create something great!
11 |
12 */
13
14 Route::get('/hello', function () {
15     return "Welcome : Hello World";
16 });
17
18 Route::get('/product/display', 'ProductController@showAll');
19
20 Route::post('/product/save', 'ProductController@saveNew');
```

Contoh Menambahkan code ProductController



The screenshot shows an IDE window with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for 'toko-online' with folders like 'app', 'bootstrap', 'config', 'database', 'public', and 'resources'. The 'app' folder is expanded to show 'Http' > 'Controllers', where 'ProductController.php' is selected. The code editor shows the following PHP code:

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 class ProductController extends Controller
8 {
9     //
10    public function showAll(){
11        // untuk mengambil dan menampilkan semua data
12
13
14        $dataProductDariModel = Product::all();
15        return view('product.display', ["products" =>
16        $dataProductDariModel]);
17    }
18
19    public function saveNew(){
20
21        // untuk menyimpan product baru
22    }
23 }
24
```

Akan muncul Pesan Error

← → ↻ ⓘ localhost/toko-online/product/display

Symfony \ Component \ Debug \ Exception \ FatalThrowableError (E_ERROR)
Class 'App\Http\Controllers\Product' not found

Application frames (1) All frames (1)

Symfony\Component\Debug\Exception\FatalThrowableError
...\app\Http\Controllers\ProductController.php:14

```
C:\xampp\htdocs\toko-online\app\Http\Controllers\ProductController.php
1. <?php
2.
3. namespace App\Http\Controllers;
4.
5. use Illuminate\Http\Request;
6.
7. class ProductController extends Controller
8. {
9.     //
10.    public function showAll(){
11.
12.        // untuk mengambil dan menampilkan semua data
13.
14.        $dataProductDariModel = Product::all();
15.        return view('product.display', ["products" =>
16.            $dataProductDariModel]);
17.    }
18.
19.    public function saveNew(){
20.
21.        // untuk menyimpan product baru
22.    }
23. }
24.
```

Arguments

- "Class 'App\Http\Controllers\Product' not found"

No comments for this stack frame.

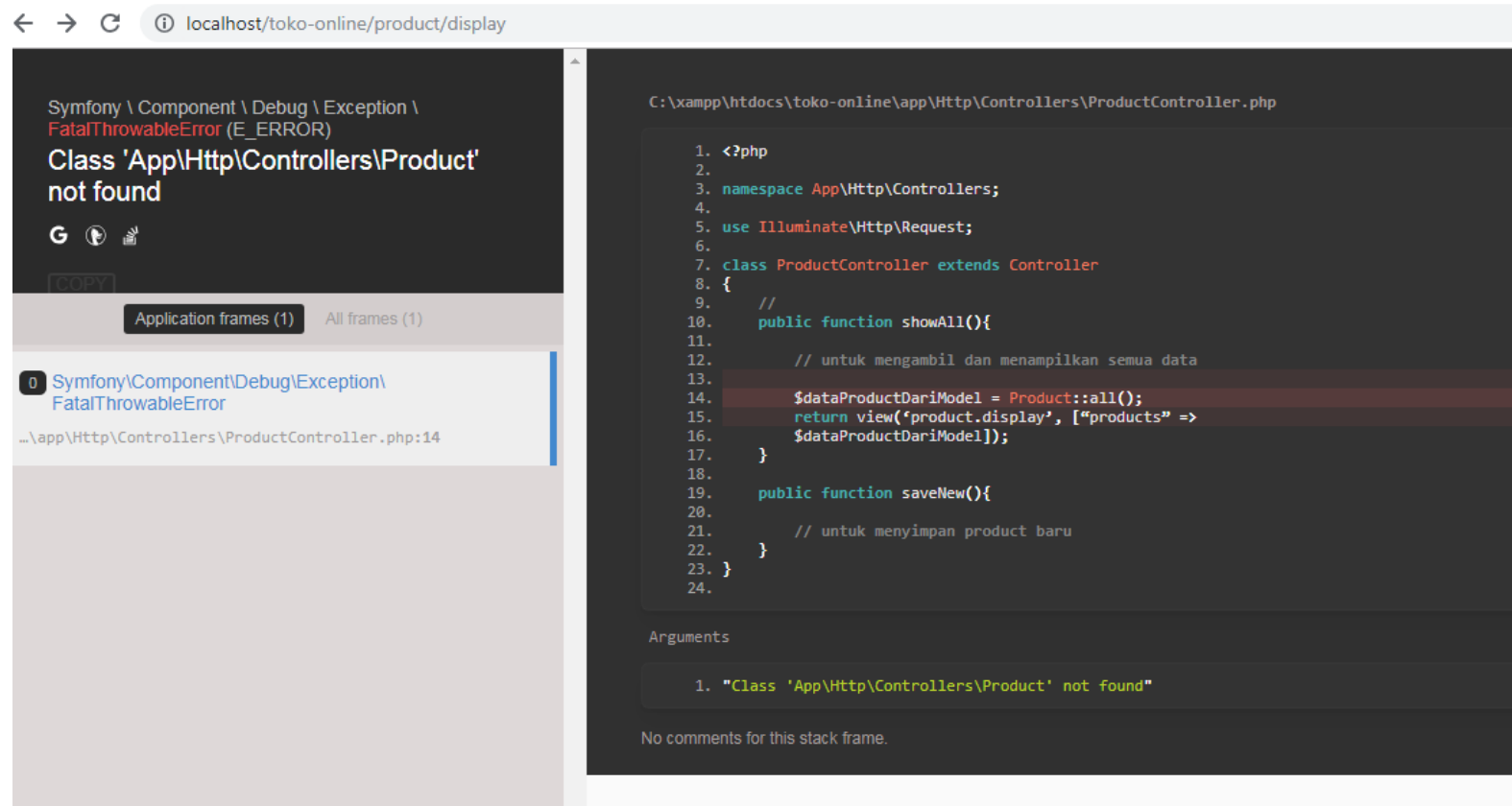
Pengenalan View

Pengenalan View

View bertanggungjawab untuk memberikan tampilan ke user. Jika kita ingin meletakkan kode html, css dan javascript di view lah tempatnya. Laravel membuat view lebih powerful dengan memanfaatkan templating engine.

Karena Laravel menggunakan templating engine bawaan Blade, maka file view diakhiri dengan `.blade.php`. Misal **product.blade.php, product-list.blade.php, product-detail.blade.php**.

Masih muncul Pesan Error-> Karena belum ada **View**



The screenshot displays a web browser window with the URL `localhost/toko-online/product/display`. The page shows a 500 Internal Server Error. The error message is:

```
Symfony \ Component \ Debug \ Exception \ FatalThrowableError (E_ERROR)  
Class 'App\Http\Controllers\Product'  
not found
```

The error details in the developer tools show the following information:

- Application frames (1) All frames (1)
- Symfony\Component\Debug\Exception\FatalThrowableError
- ...app\Http\Controllers\ProductController.php:14

The corresponding PHP code in `C:\xampp\htdocs\toko-online\app\Http\Controllers\ProductController.php` is shown below:

```
1. <?php  
2.  
3. namespace App\Http\Controllers;  
4.  
5. use Illuminate\Http\Request;  
6.  
7. class ProductController extends Controller  
8. {  
9.     //  
10.    public function showAll(){  
11.        // untuk mengambil dan menampilkan semua data  
12.  
13.        $dataProductDariModel = Product::all();  
14.        return view('product.display', ["products" =>  
15.            $dataProductDariModel]);  
16.    }  
17.  
18.    public function saveNew(){  
19.        // untuk menyimpan product baru  
20.  
21.    }  
22. }  
23.  
24.
```

The arguments for the error are:

- "Class 'App\Http\Controllers\Product' not found"

No comments for this stack frame.

Maka kita buat sebuah file baru di **resources/views/product/display.blade.php**. Lalu coba tambahkan text atau kode html, misalnya

** View display product siap .**

Jika sudah silahkan coba untuk mengakses route yang tadi error. Apa yang terjadi?

view berhasil menampilkan pesan.

Kesimpulan

Di bagian ini kita mempelajari alur kerja MVC Laravel, dengan memahami alur kerja MVC kita semakin mudah mencodding program selanjutnya.

Pertama kita definisikan jalur akses URL melalui route, route tersebut kita arahkan ke controller dan action tertentu, action tersebut akan menggunakan model dan query eloquent untuk mengambil data, lalu action tersebut mengembalikan views dengan data data dari model bila diperlukan.