



# Seminggu Belajar Laravel



Rahmat Awaludin

# Seminggu Belajar Laravel

Laravel itu framework PHP yang bikin hidup programmer lebih menyenangkan. Jadi, belajarnya juga mesti menyenangkan.

Rahmat Awaludin



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/)

## **Tweet This Book!**

Please help Rahmat Awaludin by spreading the word about this book on [Twitter](#)!

The suggested hashtag for this book is [#seminggubelajararavel](#).

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

<https://twitter.com/search?q=#seminggubelajararavel>

*Untuk istriku tercinta Irna Rahyu dan jagon kecilku Shidqi Abdullah El-Mubarak.*

# Contents

|                                  |          |
|----------------------------------|----------|
| <b>Routing dan MVC</b> . . . . . | <b>1</b> |
| Routing . . . . .                | 1        |
| MVC . . . . .                    | 3        |
| View . . . . .                   | 9        |
| Controller . . . . .             | 11       |
| Ringkasan . . . . .              | 12       |

# Routing dan MVC

Really, the web is pretty simple stuff. It's just request - response. I told myself this over and over again when building Laravel. I just want a simple way to catch requests and send out responses. That's it.

*Why Laravel? - Taylor Otwell<sup>1</sup>*

Pada hari 2 ini, kita akan belajar struktur dasar dari sebuah aplikasi yang dibangun dengan framework Laravel. Untuk memudahkan, kita akan menggunakan aplikasi webapp yang dibuat di hari 1.

## Routing

Jika diibaratkan sebuah Mall, routing itu ibarat Bagian Informasi. Jika Anda bertanya lokasi toko Sepatu Wiwi, maka Bagian Informasi akan mengarahkan Anda ke toko tersebut.

Routing untuk Laravel dapat diatur pada file `app/routes.php`. Pada aplikasi yang lebih kompleks, kita dapat meletakkan routing pada file lain agar `app/routes.php` tidak terlihat berantakan. Berikut isian dari `app/routes.php`:

`app/routes.php`

---

```
1 ....
2 Route::get('/', function()
3 {
4     return View::make('hello');
5 });
```

---

Misalnya, jika kita ingin membuat halaman statis yang bisa diakses di `http://webapp.site/about`, tambahkan isian seperti ini pada `app/routes.php`:

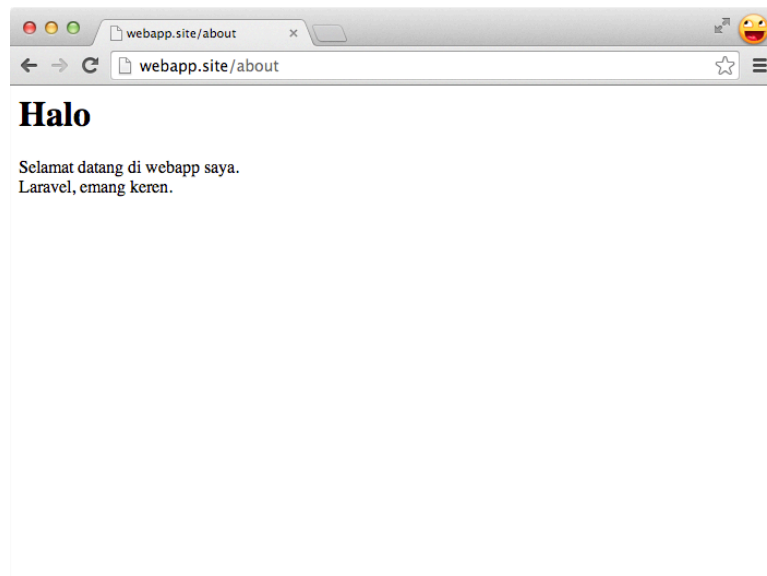
---

<sup>1</sup><http://taylorotwell.tumblr.com/post/21038245018/why-laravel>

**app/routes.php**

```
1 ....
2 Route::get('/about', function() {
3     return '<h1>Halo</h1>'
4         .'Selamat datang di webapp saya<br>'
5         .'Laravel, emang keren.';
6 });
```

Maka, kita dapat akses `http://webapp.site/about`.



**Berhasil membuat route about**

Mari kita perhatikan syntax `Route::get('/about', function() { ... })`.

- Parameter `get` menjelaskan jenis request yang diterima dalam contoh ini GET request. Kita dapat merubah post untuk mengakses POST request.
- `/about` merupakan URL yang kita inginkan untuk diakses.
- `function() { ... }` merupakan closure (anonymous function) yang akan memberikan jawaban atas request. Selain menggunakan closure, kita juga dapat mengarahkan request ke fungsi pada sebuah controller.

Untuk mengecek route apa saja yang telah kita buat, dapat menggunakan perintah berikut di terminal:

```
$ php artisan routes
```

```

2. rahmatawaludin@MorphaWorks: ~/Sites/webapp (zsh)
webapp [master] ⚡ php artisan routes
+-----+-----+-----+-----+-----+-----+
| Domain | URI      | Name | Action                               | Before Filters | After Filters |
+-----+-----+-----+-----+-----+-----+
|         | GET|HEAD / |      | Closure                             |                |                |
|         | GET|HEAD about |    | HomeController@showAbout           |                |                |
+-----+-----+-----+-----+-----+-----+
webapp [master] ⚡

```

Mengecek route yang telah dibuat

Fitur routing ini sangat banyak, dan saya tidak akan menjelaskan semua fiturnya. Fitur lainnya akan saya jelaskan selama kita membuat sample aplikasi. Jika Anda tertarik mempelajari lebih jauh tentang routing, silahkan akses di [dokumentasi routing](#)<sup>2</sup>.

## MVC

Jika kita membuka folder `app/` akan kita temui tiga subdirektori yaitu `models/`, `views/`, dan `controllers/`. Ini menandakan bahwa Laravel mendukung penuh design arsitektur software secara **Model View Controller (MVC)**<sup>3</sup> untuk memisahkan logic untuk *manipulasi data*, *antarmuka pengguna* dan *kontrol aplikasi*.

## Model

Model mewakili struktur data yang kita gunakan dalam aplikasi. Model ini dibuat berdasarkan objek dalam aplikasi kita. Misalnya, jika kita membuat aplikasi blog, maka *artikel* dan *komentar* dapat menjadi model. Selain sebagai struktur data, model juga menyimpan *business rules* dari aplikasi. Misalnya, dalam sebuah blog komentar minimal berisi 10 karakter, maka model komentar memastikan bahwa data yang isi sudah sesuai dengan aturan tersebut.

Ketika kita membahas model, pasti akan membahas tentang [database](#)<sup>4</sup>. Laravel memiliki fitur menarik untuk manajemen database, diantaranya [migrations](#) dan [seeding](#)<sup>5</sup>.

## Migrations

Laravel memudahkan kita untuk membuat struktur database yang dapat disimpan dalam VCS semisal GIT. Dengan menggunakan migrations, perubahan struktur database selama pengembangan aplikasi dapat tercatat dan terdistribusikan ke semua anggota tim.

Mari kita buat migrations untuk membuat table Post dengan struktur:

<sup>2</sup><http://laravel.com/docs/routing>

<sup>3</sup><http://id.wikipedia.org/wiki/MVC>

<sup>4</sup><http://laravel.com/docs/database>

<sup>5</sup><http://laravel.com/docs/migrations>



Struktur table Post

| Field      | Type             | Null | Key | Default             | Extra          |
|------------|------------------|------|-----|---------------------|----------------|
| id         | int(10) unsigned | NO   | PRI | NULL                | auto_increment |
| title      | varchar(255)     | NO   | UNI | NULL                |                |
| content    | varchar(255)     | NO   | UNI | NULL                |                |
| created_at | timestamp        | NO   |     | 0000-00-00 00:00:00 |                |
| updated_at | timestamp        | NO   |     | 0000-00-00 00:00:00 |                |

1. Buka terminal, masuk ke folder webapp, jalankan perintah berikut:

```
$ php artisan migrate:make create_posts_table
```

2. Perintah diatas akan menghasilkan sebuah file, misalnya dengan nama app/database/migrations/2014\_03\_26\_033903\_create\_posts\_table.php. Ubah isian file ini menjadi:

app/database/migrations/20140326\_033903\_create\_posts\_table.php

```

1  <?php
2  use Illuminate\Database\Schema\Blueprint;
3  use Illuminate\Database\Migrations\Migration;
4
5  class CreatePostsTable extends Migration {
6      /**
7       * Run the migrations.
8       *
9       * @return void
10      */
11     public function up()
12     {
13         Schema::create('posts', function(Blueprint $table)
14         {
15             $table->increments('id');
16             $table->string('title')->unique();
17             $table->string('content');
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      *
25      * @return void
26      */
27     public function down()
28     {
29         Schema::drop('posts');
```

```

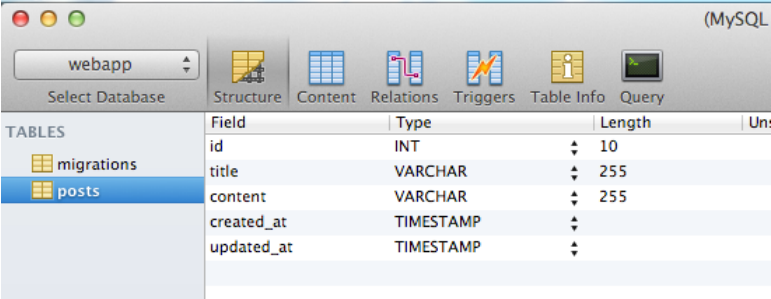
30     }
31 }

```

3. Pada fungsi up diatas laravel akan membuat table posts. Sedangkan, pada fungsi down, laravel akan menghapus table posts.
4. Jalankan perintah ini untuk melakukan migrasi :

```
$ php artisan migrate
```

5. Cek pada database Anda, akan terdapat table migrations dan posts. Table migrations berfungsi untuk mencatat migrasi database yang telah kita lakukan. Table posts adalah table yang didefinisikan di file migrasi yang telah kita buat.



The screenshot shows the MySQL Workbench interface with the 'webapp' database selected. The 'Structure' tab is active, displaying the table 'posts'. The table has five columns: 'id' (INT, 10), 'title' (VARCHAR, 255), 'content' (VARCHAR, 255), 'created\_at' (TIMESTAMP), and 'updated\_at' (TIMESTAMP).

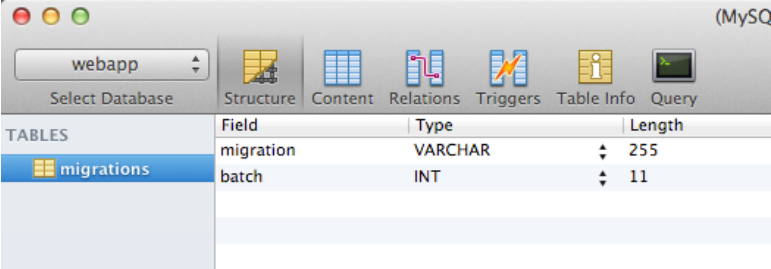
| Field      | Type      | Length | Un: |
|------------|-----------|--------|-----|
| id         | INT       | 10     |     |
| title      | VARCHAR   | 255    |     |
| content    | VARCHAR   | 255    |     |
| created_at | TIMESTAMP |        |     |
| updated_at | TIMESTAMP |        |     |

Migrasi berhasil

6. Untuk mendemonstrasikan kegunaan migration, mari kita lakukan rollback untuk meng-undo migrasi yang telah kita lakukan. Jalankan perintah ini :

```
$ php artisan migrate:rollback
```

7. Cek kembali database Anda, maka table posts akan terhapus.



The screenshot shows the MySQL Workbench interface with the 'webapp' database selected. The 'Structure' tab is active, displaying the table 'migrations'. The table has two columns: 'migration' (VARCHAR, 255) and 'batch' (INT, 11).

| Field     | Type    | Length |
|-----------|---------|--------|
| migration | VARCHAR | 255    |
| batch     | INT     | 11     |

Rollback berhasil

Sekarang berhenti sejenak, renungkan apa yang telah kita lakukan. Dengan menggunakan metode migrasi, struktur database dapat lebih dipahami dan di *maintenance*. Bandingkan jika menggunakan import/export file .sql. Tentunya cukup merepotkan jika struktur database sering berubah ketika develop aplikasi. Saran saya, gunakan migrasi untuk manajemen database selama pengembangan aplikasi dan export/import file sql ketika produksi.

Yang lebih powerfull, migrasi ini tidak hanya bisa menambah/menghapus table. Migrasi juga memungkinkan kita merubah struktur suatu table, misalnya menambah/hapus/ubah suatu kolom. Jika ingin belajar lebih lanjut tentang migrasi, kunjungi [dokumentasi migrasi](http://laravel.com/docs/migrations)<sup>6</sup>.

## Database Seeder

Terkadang ketika kita mengembangkan sebuah aplikasi, dibutuhkan contoh data. Bisa saja contoh data tersebut kita inject langsung ke database, namun cukup merepotkan jika kita ingin menginject banyak data. Database Seeder berfungsi untuk membuat contoh data bagi aplikasi.

Mari kita buat database seeder untuk table posts:

1. Buatlah file `app/database/seeds/PostsTableSeeder.php` isi seperti berikut untuk menentukan contoh data yang akan kita masukkan ke dalam database:

`app/database/seeds/PostsTableSeeder.php`

---

```
1  <?php
2
3  class PostsTableSeeder extends Seeder {
4
5      public function run()
6      {
7          // kosongkan table posts
8          DB::table('posts')->delete();
9
10         // buat data berupa array untuk diinput ke database
11         $posts = array(
12             array('id'=>1, 'title'=>'Tips Cepat Nikah', 'content'=>'lorem ipsum'),
13             array('id'=>2, 'title'=>'Haruskah Menunda Nikah?', 'content'=>'lorem ipsum'),
14             array('id'=>3, 'title'=>'Membangun Visi Misi Keluarga', 'content'=>'lorem ips\
15 um')
16         );
17
18         // masukkan data ke database
19         DB::table('posts')->insert($posts);
20     }
21
22 }
```

---

2. Tambahkan `PostsTableSeeder` ke dalam `app/database/seeds/DatabaseSeeder.php` :

---

<sup>6</sup><http://laravel.com/docs/migrations>

app/database/seeds/DatabaseSeeder.php

```

1  <?php
2
3  class DatabaseSeeder extends Seeder {
4
5      /**
6       * Run the database seeds.
7       *
8       * @return void
9       */
10     public function run()
11     {
12         Eloquent::unguard();
13
14         $this->call('PostsTableSeeder');
15     }
16
17 }

```

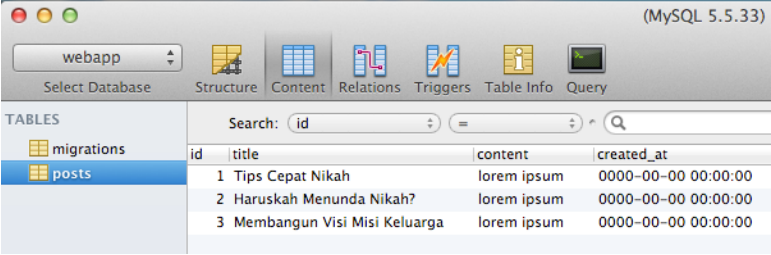
3. Untuk melakukan *seeding*, jalankan perintah ini:

```

$ php artisan migrate
$ php artisan db:seed

```

4. Cek kembali database, maka contoh data telah masuk ke dalam database.



The screenshot shows the MySQL 5.5.33 interface. The 'webapp' database is selected. The 'posts' table is highlighted in the 'TABLES' list. The table structure is displayed with columns: id, title, content, and created\_at. There are 3 rows of data.

| id | title                        | content     | created_at          |
|----|------------------------------|-------------|---------------------|
| 1  | Tips Cepat Nikah             | lorem ipsum | 0000-00-00 00:00:00 |
| 2  | Haruskah Menunda Nikah?      | lorem ipsum | 0000-00-00 00:00:00 |
| 3  | Membangun Visi Misi Keluarga | lorem ipsum | 0000-00-00 00:00:00 |

Database Seeding berhasil

Tips: Jika database telah terisi dengan data dari aplikasi dan Anda ingin mereset ke kondisi setelah seeding, gunakan perintah `php artisan migrate:refresh --seed`

## Membuat Model

Model dalam Laravel dibuat dengan cara melakukan *extends* class **Eloquent**. Mari kita buat model untuk mengakses table posts yang telah dibuat. Buat file `app/models/Post.php` dengan isi seperti berikut:

**app/models/Post.php**


---

```

1 <?php
2 class Post extends Eloquent {
3     /**
4      * white list column for mass assignment
5      * @link http://laravel.com/docs/eloquent#mass-assignment
6      * @var array
7      */
8     protected $fillable = ['title', 'post'];
9 }

```

---

**Mengakses model**

Model dapat diakses dengan langsung memanggil class model tersebut dimanapun kita butuhkan. Eloquent merupakan ORM (Object Relational Mapper) yang powerfull untuk manipulasi data. Berikut ini akan saya berikan beberapa contoh fitur Eloquent:

1. Buat route test pada file app/routes.php dengan isi sebagai berikut:

**app/routes**


---

```

1 Route::get('/testmodel', function() {
2     $query = /* isi sample query */ ;
3     return $query;
4 });

```

---

2. Pada beberapa contoh dibawah, silahkan ubah `/* isi sample query */` dengan contoh yang ingin dicoba. Untuk mengecek hasilnya, buka <http://webapp.site/test><sup>7</sup>.
  - Mencari semua model:

```
1 Post::all();
```

- Mencari model berdasarkan id:

```
1 Post::find(1);
```

- Mencari model berdasarkan title:

```
1 Post::where('title', 'like', '%cepat nikah%')->get();
```

- Mengubah record, (hapus semua isi function) :

---

<sup>7</sup><http://webapp.site/test>

```

1  $post = Post::find(1);
2  $post->title = "Ciri Keluarga Sakinah"
3  return $post;

```

- Menghapus record, (hapus semua isi function) :

```

1  $post = Post::find(1);
2  $post->delete();
3  // check data di database

```

- Menambah record, (hapus semua isi function) :

```

1  $post = new Post;
2  $post->title = "7 Amalan Pembuka Jodoh";
3  $post->content = "shalat malam, sedekah, puasa sunah, silaturahmi, senyum, doa, tobat";
4  $post->save();
5  return $post;
6  // check record baru di database

```

- Penjelasan lengkap untuk beberapa syntax query berikut dapat ditemukan di [dokumentasi query builder](http://laravel.com/docs/queries)<sup>8</sup> dan [eloquent](http://laravel.com/docs/eloquent)<sup>9</sup>.

## View

View atau istilah lainnya *presentation logic* berfungsi untuk menampilkan data yang telah kita olah di bussiness logic. Laravel memudahkan kita untuk membuat view. Mari kita ubah route about yang sudah dibuat menjadi view:

1. Ubah route about menjadi:

app/routes.php

---

```

1  Route::get('/about', function() {
2      return View::make('about');
3  });

```

---

2. Buat file app/views/about.php dengan isi:

---

<sup>8</sup><http://laravel.com/docs/queries>

<sup>9</sup><http://laravel.com/docs/eloquent>

**app/routes.php**

---

```
1  <html>
2      <body>
3          <h1>Halo</h1>
4          Selamat datang di webapp saya.<br>
5          Laravel, emang keren.
6      </body>
7  </html>
```

---

3. Cek kembali route `http://webapp.site/about` dan hasilnya, tetap sama. Yang berubah adalah *logic*-nya, sekarang kita memindahkan logic untuk menampilkan html ke file view terpisah.

## Template dengan Blade

Selain dengan memisahkan peletakan view pada file berbeda, Laravel juga lebih menekankan penggunaan view ini dengan templating. Dengan templating ini, developer akan ‘terpaksa’ hanya menggunakan syntax untuk tampilan dan logic sederhana pada *view* nya. Templating pada Laravel menggunakan [Blade](#)<sup>10</sup>.

Untuk menggunakan view dengan blade template, kita cukup merubah ekstensi file view menjadi `.blade.php`. Pada contoh file `about.php`, maka kita ubah menjadi `about.blade.php` untuk menggunakan blade template.

## Blade Syntax

Syntax yang paling sederhana dalam blade adalah `{{ }}` (double curly braces). Syntax ini dapat menggantikan fungsi `<?php echo ;?>` pada file view. Jadi, syntax `{{ $variabel }}` akan berubah menjadi syntax `<?php echo $variable; ?>`. Jika akan menampilkan variable hasil input user, gunakan `{{{ }}}` (triple curly braces) agar variable yang ditampilkan di escape (dibersihkan dari script) terlebih dahulu.

Selain mendukung control structures semisal `@if`, `@for`, `@foreach`, `@while`, `@unless`, dll untuk templating. Silahkan rujuk ke [dokumentasi resmi](#)<sup>11</sup> untuk penjelasan lebih lengkapnya.

## Form

Membuat form di Laravel sangat mudah, berikut ini syntax dasar untuk membuat form:

```
1  {{ Form::open(array('url' => 'post/save')) }}
2      //
3  {{ Form::close() }}
```

Membuat elemen-elemen form dalam Laravel juga sangat mudah, berikut ini contoh untuk menampilkan label dengan *placeholder* E-Mail Address dan *class awesome*:

---

<sup>10</sup><http://laravel.com/docs/templates>

<sup>11</sup><http://laravel.com/docs/templates#other-blade-control-structures>

```
1 {{ Form::label('email', 'E-Mail Address', array('class' => 'awesome')) }}
```

Untuk membuat input :

```
1 {{ Form::text('username') }}
```

Penjelasan lebih lengkapnya, dapat diakses di [dokumentasi form](#)<sup>12</sup>.

## Request

Data request yang dikirim ke aplikasi, dapat diambil menggunakan class Input. Contoh untuk mengambil \$\_GET / \$\_POST data dengan key 'username':

```
1 $username = Input::get('username');
```

Penjelasan lebih lengkapnya, dapat diakses di [dokumentasi request](#)<sup>13</sup>.

## Controller

Pada contoh-contoh sebelumnya, saya selalu meletakkan logic di app/routes.php. Hal ini bisa saja dilakukan, tapi tidak efektif jika aplikasinya sudah besar. Cara yang sering digunakan adalah router mengarahkan request ke fungsi di controller. Nah, fungsi itulah yang akan melakukan logic untuk request tersebut dan memberikan response.

Mari kita praktekan, dengan mengubah route about ke fungsi showAbout di HomeController.php :

1. Ubah route '/about' menjadi :

app/routes.php

```
1 ....  
2 Route::get('/about', 'HomeController@showAbout');  
3 ....
```

2. Tambah fungsi showAbout pada class HomeController :

---

<sup>12</sup><http://laravel.com/docs/html>

<sup>13</sup><http://laravel.com/docs/requests>



app/controllers/HomeController.php

---

```
1  ....
2  public function showAbout()
3  {
4      return View::make('about');
5  }
6  ....
```

---

3. Akses kembali <http://webapp.site/about>, maka hasilnya akan tetap sama. Yang berubah adalah logic untuk memberikan response sekarang berada pada controller, router hanya mengarahkan request saja.

Fitur dari controller ini sangat banyak, diantaranya:

- Filtering request yang datang
- RESTfull untuk memetakan setiap fungsi pada controller menjadi routes
- Resource Controller untuk membuat restfull controller pada sebuah model

Penjelasan lebih lengkapnya, dapat diakses di [dokumentasi controller](http://laravel.com/docs/controllers)<sup>14</sup>.

## Ringkasan

Pada hari 2 ini, saya harap Anda telah dapat memahami:

- Konsep routing
- Konsep MVC

Pada hari 3, kita akan memulai membuat perencanaan untuk project yang akan dibangun dengan framework Laravel. Semangat! :)

---

<sup>14</sup><http://laravel.com/docs/controllers>