



# 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 book is for sale at <http://leanpub.com/seminggubelajarlaravel>

This version was published on 2014-05-31



This is a [Leanpub](#) book. Leanpub empowers authors and publishers with the Lean Publishing process. [Lean Publishing](#) is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#)

## **Tweet This Book!**

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

The suggested tweet for this book is:

Hei, gue baru aja download buku Seminggu Belajar Laravel. Keren nih, buat belajar framework Laravel!

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

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=#seminggubelajarlaravel>

*Untuk istriku tercinta Irna Rahyu dan jagon kecilku Shidqi Abdulllah Mubarak.*

# Contents

Koreksi . . . . .	1
Saran dan Masukan . . . . .	2
Sekilas . . . . .	3
Metode Penulisan . . . . .	3
Apakah buku ini gratis? . . . . .	3
Tawaran Kerjaan . . . . .	4
Hari 1 : Instalasi dan Konfigurasi Laravel . . . . .	5
Text Editor . . . . .	5
Kebutuhan Sistem . . . . .	5
Composer . . . . .	5
Instalasi Laravel . . . . .	9
Konfigurasi . . . . .	12
Menjalankan Web Server . . . . .	12
Ringkasan . . . . .	15
Hari 2 : Routing dan MVC . . . . .	16
Routing . . . . .	16
MVC . . . . .	18
View . . . . .	24
Controller . . . . .	26
Ringkasan . . . . .	27
Hari 3 : Persiapan Project . . . . .	28
Design tampilan . . . . .	28
Database . . . . .	39
Package . . . . .	40
Server . . . . .	41
Code sample . . . . .	42
Layout . . . . .	42
Ringkasan . . . . .	50
Hari 4 : Develop Fitur Admin . . . . .	51
Authentikasi dan Filter . . . . .	51
Fitur Admin . . . . .	68

## CONTENTS

CRUD Penulis . . . . .	69
CRUD Buku . . . . .	95
Ringkasan . . . . .	115
<b>Hari 5 : Develop Fitur Non-Admin . . . . .</b>	<b>116</b>
Member . . . . .	116
Stok Buku . . . . .	154
Ringkasan . . . . .	167
<b>Hari 6 : User Management . . . . .</b>	<b>168</b>
Register . . . . .	168
Halaman Profil . . . . .	182
Ubah Password . . . . .	189
Lupa Password . . . . .	195
Listing Member . . . . .	206
Data Peminjaman . . . . .	214
Bonus : Penanganan Error . . . . .	231
Ringkasan . . . . .	234
<b>Hari 7 : Deploy Aplikasi . . . . .</b>	<b>236</b>
Konfigurasi Hosting . . . . .	236
Deploy Aplikasi . . . . .	246
Ringkasan . . . . .	249
<b>Penutup . . . . .</b>	<b>250</b>
Belajar lagi! . . . . .	250
Terima kasih . . . . .	251

# Koreksi

Setiap manusia pasti tidak pernah luput dari kesalahan, begitupun dengan penulis. Jika Anda menemukan kesalahan dalam penulisan buku ini, silahkan kirim di [Bitbucket](#)<sup>1</sup>.

Setiap kesalahan yang ditemukan akan segera diperbaiki pada buku, Anda cukup mendownload ulang buku ini dari Leanpub.

---

<sup>1</sup><https://bitbucket.org/rahmatawaludin/laravel-perpustakaan/issues>

# Saran dan Masukan

Saya sangat mengharapkan saran bagi perbaikan penulisan buku ini kedepan. Jika Anda punya saran yang ingin disampaikan dapat disampaikan ke email saya [rahmat.awaludin@gmail.com](mailto:rahmat.awaludin@gmail.com)<sup>2</sup> atau via twitter di [@rahmatawaludin](https://twitter.com/rahmatawaludin)<sup>3</sup>.

---

<sup>2</sup><mailto:rahmat.awaludin@gmail.com>

<sup>3</sup><https://twitter.com/rahmatawaludin>

# Sekilas

Perkenalkan, saya Rahmat Awaludin. Saya seorang fullstack web developer dari Bandung, Indonesia. Sudah sejak tahun 2011 saya berada di dunia web development, lebih tepatnya ketika saya masih berstatus sebagai mahasiswa di Universitas Padjadjaran.

Perkenalan saya dengan Laravel dimulai ketika saya mengerjakan sebuah project untuk LSM Save The Children. Dalam project ini saya belajar banyak tentang Laravel dan modern PHP development, termasuk didalamnya berbagai konsep dan tools seperti SOLID design, TDD, Composer, dll.

Buku ini saya tulis untuk programmer pemula framework Laravel, namun telah ada sedikit pengalaman menggunakan framework lain. Harapannya jika pembaca telah terbiasa menggunakan framework PHP lain, membaca buku ini sekaligus akan menjelaskan letak kemudahan Laravel untuk pengembangan aplikasi PHP.

## Metode Penulisan

Saya membuat aplikasi dalam buku ini dengan OS X. Sebagian syntax terminal, saya buat dua versi (\*nix dan windows). Namun, kebanyakan syntax yang saya tulis adalah syntax pada OS \*nix.

Untuk memudahkan pembaca, source code dari aplikasi di buku ini dapat di download dari [bitbucket](#)<sup>4</sup>. Pada setiap tahapan coding yang dijelaskan, saya usahakan untuk memberikan link lengkap tentang perubahan yang dilakukan di source code. Akses untuk repository dapat diminta melalui email.

## Apakah buku ini gratis?

Saya sangat menginginkan buku ini gratis, karena saya ingin ilmunya berkah. Saya khawatir, jika pembaca buku ini memperoleh buku ini dengan cara yang tidak diridhoi penulisnya (membajak), ilmu yang diperoleh dari buku ini tidak akan berkah. Karena, yang paling penting bukan banyaknya ilmu, tapi keberkahannya pada ilmu yang dimiliki.

Tapi, ternyata saya sudah punya istri dan anak yang harus dinafkahi tiap bulan. Untuk itu, saya mencoba mencari jalan tengah agar pembaca dapat memperoleh buku ini dengan terjangkau dan saya dapat tetap menafkahi keluarga saya walaupun disibukkan menulis buku.

## Oke mas, saya paham. Terus harganya berapa?

Buku ini kalau saya sendiri menghargainya Rp150.000. Tentunya harga ini bukan harga mutlak, setidaknya itu harga yang saya hitung cukup untuk mengganti waktu kebersamaan saya bersama keluarga selama saya menulis buku ini.

Saya ingin buku ini dapat dijangkau oleh semua kalangan. Oleh karena itu, bagi Anda yang belum memiliki uang senilai itu, tinggal menghubungi saya (kontak dibawah). Selebihnya, Anda sangat saya sarankan melebihkan harga buku ini, sesuai dengan besarnya rasa syukur yang dimiliki atas ilmu yang diperoleh.. :)

---

<sup>4</sup><https://bitbucket.org/rahmatawaludin/laravel-perpustakaan>

## Cara bayarnya gimana?

Pembayaran dapat dilakukan langsung di Leanpub atau via transfer ke Rekening BNI 0186355188 An. Rahmat Awaludin bin Asdi. Bagi pembaca yang membayar dengan transfer, dapat menghubungi saya via email atau sms untuk mendapatkan coupun code yang bisa digunakan untuk mendownload buku ini.

## Kalau udah beli buku ini dapat apa aja?

Bagi yang sudah membeli buku ini akan memperoleh 7 manfaat berikut ini:

1. Akses ke update buku.
2. Akses ke source code di bitbucket.
3. Membayai penulisan buku saya selanjutnya [Implementasi TDD dalam Laravel<sup>5</sup>](#).
4. Konsultasi privat melalui fb/email ke penulis selama 1 bulan (non-privat dapat dilakukan di grup fb Laravel Indonesia).
5. Membantu penulis menafkahi anak dan istri dengan harta yang halal.
6. Hati yang tenang karena telah memperoleh ilmu dengan cara yang diridhoi penulisnya.
7. Do'a dari penulis, agar ilmu yang didapatkan berbuah project bernilai jutaan, bahkan ratusan juta.  
Amiinn.. :)

## Apa buku ini sudah selesai?

Sudah.

## Kalau ada materi yang kurang gimana?

Boleh request ke saya, langsung kontak saja. Kalau memang penting, bisa saya tambahkan sebagai materi bonus.

## Tawaran Kerjaan

Sebagai freelance developer, saya terbuka untuk tawaran konsultasi, project, dan pembicara (seminar/training). Jika Anda tertarik mengundang saya kontak saya via:

- Email : [rahmat.awaludin@gmail.com<sup>6</sup>](mailto:rahmat.awaludin@gmail.com)
- FB : [www.facebook.com/rahmat.awaludin<sup>7</sup>](http://www.facebook.com/rahmat.awaludin)
- Twitter : [https://twitter.com/rahmatawaludin<sup>8</sup>](https://twitter.com/rahmatawaludin)
- linkedin : [www.linkedin.com/in/rahmatawaludin<sup>9</sup>](http://www.linkedin.com/in/rahmatawaludin)
- Phone : +62878 2225 0272<sup>10</sup>

<sup>5</sup><https://leanpub.com/tddlaravel>

<sup>6</sup><mailto:rahmat.awaludin@gmail.com>

<sup>7</sup>[www.facebook.com/rahmat.awaludin](http://www.facebook.com/rahmat.awaludin)

<sup>8</sup><https://twitter.com/rahmatawaludin>

<sup>9</sup>[www.linkedin.com/in/rahmatawaludin](http://www.linkedin.com/in/rahmatawaludin)

<sup>10</sup><tel:+6287822250272>

# Hari 1 : Instalasi dan Konfigurasi Laravel

Laravel sangat mudah dikonfigurasi untuk mengembangkan sebuah aplikasi. Pada bagian ini akan saya jelaskan apa saja yang harus dipersiapkan untuk memulai menggunakan framework Laravel. Untuk memudahkan pemahaman, saya tidak akan menjelaskan beberapa istilah secara rinci.

## Text Editor

Selama mengembangkan aplikasi di buku ini, saya menggunakan teks editor [Sublime Text 3](#)<sup>11</sup> dengan plugin [Laravel Blade Highlighter](#)<sup>12</sup> untuk menampilkan syntax highlighting blade dan [Emmet](#)<sup>13</sup> untuk mempercepat mengetik HTML. Jika Anda lebih menyukai sebuah IDE, saya sarankan menggunakan [PHPStorm](#)<sup>14</sup> atau [AksiIDE](#)<sup>15</sup> karya om [Luri Darmawan](#)<sup>16</sup>, seorang anak bangsa yang merupakan member yang disegani di grup PHP Indonesia<sup>17</sup>.

## Kebutuhan Sistem

Laravel mendukung penggunaan web server apache dan ngix. Pada buku ini, saya menggunakan web server Apache. Pastikan PHP yang Anda gunakan sudah versi 5.3.7 keatas. Saya sendiri menggunakan [MAMP](#)<sup>18</sup> untuk OSX, jika Anda pengguna windows bisa menggunakan [XAMPP](#)<sup>19</sup>. Jika Anda akan menggunakan Laravel 4.2, pastikan versi PHP minimal 5.4.

Untuk database Laravel dapat menggunakan database MySQL, PostgreSQL, SQLServer atau SQLite.

## Composer

Untuk menginstall laravel kita akan menggunakan composer. Composer adalah aplikasi yang digunakan untuk mengatur package-package dalam mengembangkan sebuah web dengan PHP. Jika dulu, mungkin Anda mengenal yang namanya [PEAR](#)<sup>20</sup>, composer tuh mirip-mirip PEAR lah.

Anggaplah kita belum kenal dengan PEAR/Composer. Jika kita akan mengembangkan sebuah aplikasi web dan membutuhkan library untuk user management misalnya ‘UserAuth’ maka kita akan download dari webnya, letakkan di folder tertentu (misalnya library), kemudian me-load dengan require atau include pada class yang kita butuhkan.

Setidaknya ada beberapa masalah dari solusi ini:

---

<sup>11</sup>[www.sublimetext.com](http://www.sublimetext.com)

<sup>12</sup><http://github.com/Medalink/laravel-blade>

<sup>13</sup><http://emmet.io>

<sup>14</sup><http://www.jetbrains.com/phpstorm>

<sup>15</sup><http://aksiide.com>

<sup>16</sup><https://www.facebook.com/luridarmawan>

<sup>17</sup><https://www.facebook.com/groups/35688476100>

<sup>18</sup>[www.mamp.info](http://www.mamp.info)

<sup>19</sup>[www.apachefriends.org](http://www.apachefriends.org)

<sup>20</sup>[pear.php.net](http://pear.php.net)

- Bagaimana jika web kita membutuhkan tidak hanya satu library, tapi 40 library? Mau download satu-persatu?
- Bagaimana jika library `UserAuth` bergantung dengan library lain? misalnya `SessionManager` dan `SessionManager` juga bergantung kepada library `Session`. Dan seterusnya, dan seterusnya..

Pada Laravel, kita akan menggunakan composer tidak hanya untuk menginstall library, tapi framework Laravel itu sendiri diinstall menggunakan composer.

Untuk memahami composer lebih lanjut, Anda dapat mengunjungi [dokumentasi resmi composer<sup>21</sup>](#).

## Install Composer

Instalasi composer agak berbeda untuk OS \*nix (Linux, OSX, dll) dan Windows, saya jelaskan masing-masing:

### Windows

Cukup download [composer-setup.exe<sup>22</sup>](#) dan jalankan file instalasi.

### \*nix

Jalankan terminal dan masukkan perintah berikut:

```
1 $ curl -sS https://getcomposer.org/installer | php  
2 $ sudo mv composer.phar /usr/local/bin/composer
```

## Penggunaan Composer

Secara default, composer akan menggunakan package yang teregister di [packagist.org<sup>23</sup>](#). Tentunya, kita juga dapat menyiapkan repositori package private menggunakan [satis<sup>24</sup>](#).

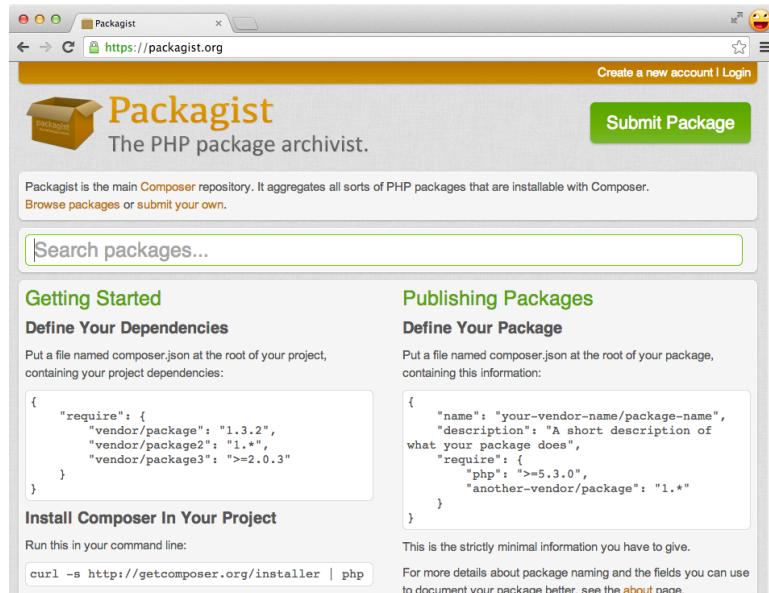
---

<sup>21</sup><https://getcomposer.org/doc/00-intro.md>

<sup>22</sup><https://getcomposer.org/Composer-Setup.exe>

<sup>23</sup><http://packagist.org>

<sup>24</sup><https://github.com/composer/satis>



### Packagist.org

Composer menggunakan file dengan [format JSON<sup>25</sup>](#). JSON merupakan format standar untuk menyimpan data name⇒value yang sudah sangat umum digunakan untuk transfer data. Contoh syntax JSON terlihat seperti ini:

#### contoh struktur JSON

```

1 {
2     name1 : {
3         subname1 : value,
4         subname2 : value
5     },
6     name2 : value
7 }
```

Composer menggunakan format json ini pada file bernama `composer.json`. Berikut contoh isi `composer.json` :

#### `composer.json`

```

1 {
2     "require": {
3         "monolog/monolog": "1.0.*"
4     }
5 }
```

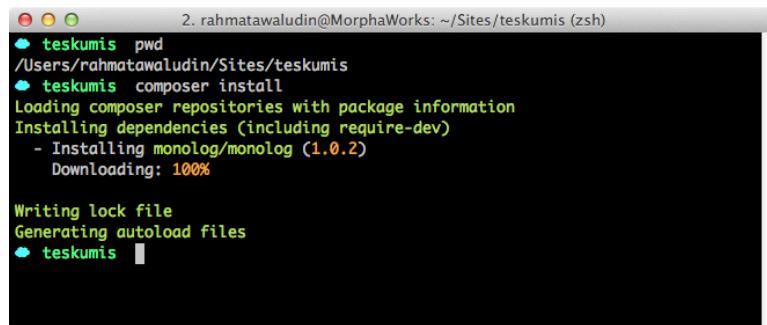
Sebagaimana terlihat pada syntax diatas, pada bagian `require` kita masukkan nama package yang kita butuhkan (`monolog/monolog`) dan versi yang diinginkan (`1.0.*`).

<sup>25</sup><http://json.org>

## Install Package

Untuk menginstall package dengan composer, pindahkan file composer.json diatas ke sebuah folder. Lalu jalankan perintah berikut di dalam folder tersebut:

```
1 $ composer install
```

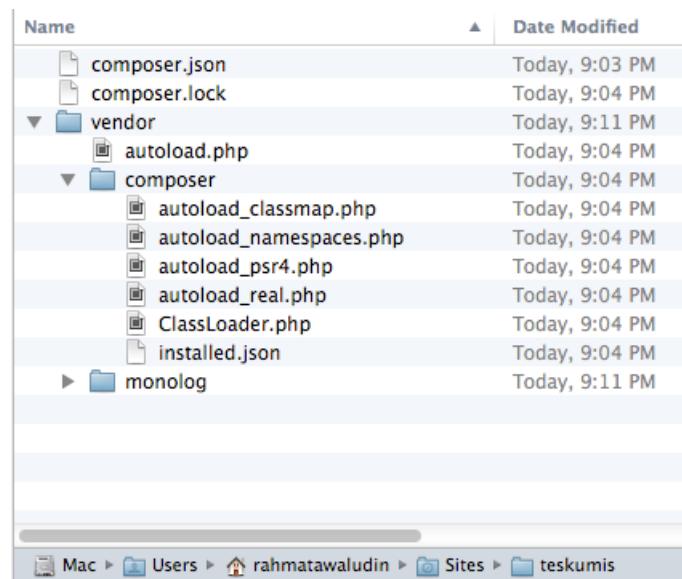


```
2. rahmatawaludin@MorphaWorks: ~/Sites/teskumis (zsh)
$ teskumis pwd
/Users/rahmatawaludin/Sites/teskumis
$ teskumis composer install
Loading composer repositories with package information
Installing dependencies (including require-dev)
- Installing monolog/monolog (1.0.2)
  Downloading: 100%

Writing lock file
Generating autoload files
$ teskumis
```

composer install

Perintah diatas akan melakukan instalasi package aplikasi yang kita tulis di bagian require. Setelah dieksekusi struktur folder kita akan berubah:



composer structure

- Folder vendor menyimpan package yang dibutuhkan, sebagaimana yang ditulis di bagian require
- File vendor/autoload.php dapat digunakan untuk mendapatkan fitur autoloading.
- File composer.lock berfungsi mencatat versi package yang saat ini sedang kita gunakan, jangan hapus/edit file ini, karena perintah composer install bergantung pada file ini.

## Update package

Jika package baru telah ditambah pada bagian require atau versi package yang digunakan dirubah, gunakan perintah ini untuk memperbarui package yang kita gunakan:

```
1 $ composer update
```

Perintah composer lainnya dapat dilihat dengan perintah :

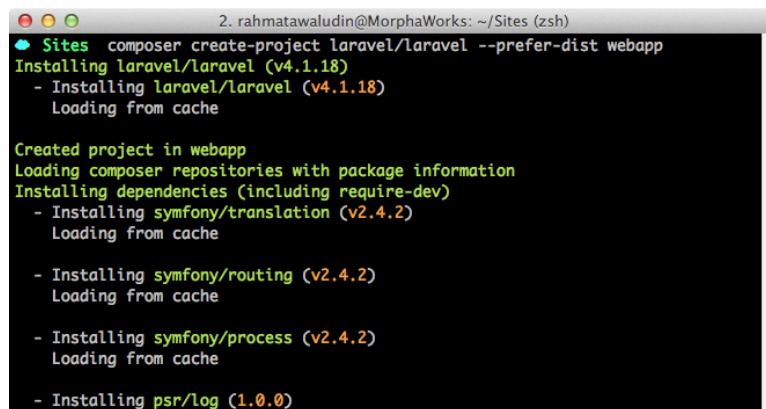
```
1 $ composer --help
```

Atau cek di [manual composer<sup>26</sup>](#).

## Instalasi Laravel

Sebagaimana disampaikan di bagian sebelumnya, Laravel diinstall menggunakan composer. Gunakan perintah ini untuk membuat project laravel di folder webapp:

```
1 $ composer create-project laravel/laravel --prefer-dist webapp
```



The screenshot shows a terminal window with the following output:

```
2. rahmatawaludin@MorphaWorks: ~/Sites (zsh)
$ composer create-project laravel/laravel --prefer-dist webapp
Installing laravel/laravel (v4.1.18)
- Installing laravel/laravel (v4.1.18)
  Loading from cache

Created project in webapp
Loading composer repositories with package information
Installing dependencies (including require-dev)
- Installing symfony/translation (v2.4.2)
  Loading from cache

- Installing symfony/routing (v2.4.2)
  Loading from cache

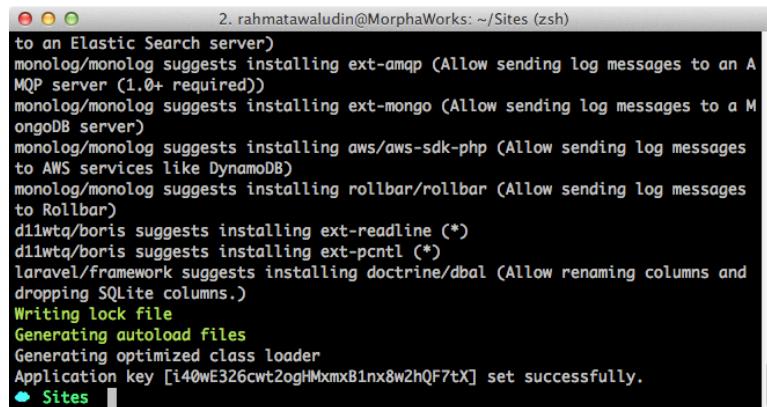
- Installing symfony/process (v2.4.2)
  Loading from cache

- Installing psr/log (1.0.0)
```

install laravel part 1

---

<sup>26</sup><https://getcomposer.org/doc>



```
2. rahmatawaludin@MorphaWorks: ~/Sites (zsh)
to an Elastic Search server)
monolog/monolog suggests installing ext-amqp (Allow sending log messages to an A
MQP server (1.0+ required))
monolog/monolog suggests installing ext-mongo (Allow sending log messages to a M
ongoDB server)
monolog/monolog suggests installing aws/aws-sdk-php (Allow sending log messages
to AWS services like DynamoDB)
monolog/monolog suggests installing rollbar/rollbar (Allow sending log messages
to Rollbar)
d1lwtq/boris suggests installing ext-readline (*)
d1lwtq/boris suggests installing ext-pcntl (*)
laravel/framework suggests installing doctrine/dbal (Allow renaming columns and
dropping SQLite columns.)
Writing lock file
Generating autoload files
Generating optimized class loader
Application key [i40WE326cwt2ogHMxmx81nx8w2hQF7tX] set successfully.
● Sites
```

install laravel part 2

Perintah ini akan menginstall framework laravel dan dependency packagenya.

Name	Date Modified
▼ app	Jan 19, 2014, 8:14
► commands	Jan 19, 2014, 8:14
► config	Jan 19, 2014, 8:14
► controllers	Jan 19, 2014, 8:14
► database	Jan 19, 2014, 8:14
filters.php	Jan 19, 2014, 8:14
► lang	Jan 19, 2014, 8:14
► models	Jan 19, 2014, 8:14
routes.php	Jan 19, 2014, 8:14
► start	Jan 19, 2014, 8:14
► storage	Jan 19, 2014, 8:14
► tests	Jan 19, 2014, 8:14
► views	Jan 19, 2014, 8:14
► artisan	Jan 19, 2014, 8:14
▼ bootstrap	Jan 19, 2014, 8:14
► autoload.php	Jan 19, 2014, 8:14
► paths.php	Jan 19, 2014, 8:14
► start.php	Jan 19, 2014, 8:14
composer.json	Jan 19, 2014, 8:14
composer.lock	Today, 9:45 PM
CONTRIBUTING.md	Jan 19, 2014, 8:14
phpunit.xml	Jan 19, 2014, 8:14
▼ public	Jan 19, 2014, 8:14
► favicon.ico	Jan 19, 2014, 8:14
► index.php	Jan 19, 2014, 8:14
► packages	Jan 19, 2014, 8:14
► robots.txt	Jan 19, 2014, 8:14
readme.md	Jan 19, 2014, 8:14
server.php	Jan 19, 2014, 8:14
► vendor	Today, 9:45 PM

laravel structure

Jika Anda menggunakan OS \*nix, instalasi laravel dapat pula dilakukan dengan menggunakan laravel.phar, caranya:

1. Download Laravel phar<sup>27</sup>

```
$ wget http://laravel.com/laravel.phar
```

2. Rename file yang telah didownload menjadi laravel

```
$ mv laravel.phar laravel
```

3. Pindahkan file tersebut ke /usr/local/bin/:

---

<sup>27</sup><http://laravel.com/laravel.phar>

```
$ sudo mv laravel /usr/local/bin/
```

4. Untuk membuat project laravel, jalan perintah :

```
$ laravel new webapp
```

## Konfigurasi

Setelah Laravel terinstall pastikan folder app/storage dapat diakses oleh web server. Cara sederhananya, jalankan perintah ini:

```
$ sudo chmod -R 777 app/storage
```

Jangan lupa isi juga konfigurasi database Anda (nama database, username, password) di app/config/database.php. Tentunya database harus Anda buat sendiri di aplikasi database yang Anda gunakan.

## Menjalankan Web Server

Web yang dikembangkan dengan Laravel dapat diakses menggunakan PHP builtin web server atau virtual host.

### PHP builtin web server

Jalankan perintah berikut di folder webapp:

```
$ php artisan serve
```

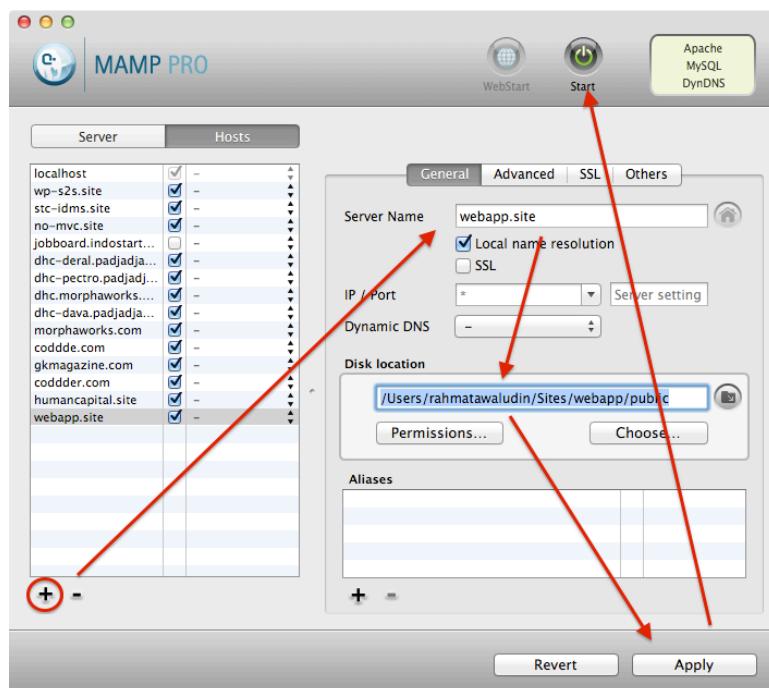
Setelah Anda menjalankan perintah diatas, aplikasi laravel dapat diakses di <http://localhost:8000>. Kekurangan dari PHP Builtin web server adalah ketika terjadi perubahan pada source code, terkadang server harus di restart.

### Virtual Host

Menggunakan virtual host, aplikasi dapat diakses dengan url seperti <http://webapp.site>, <http://www.webapp.com>, dan sebagainya walupun masih berada di lokal. Saya sendiri lebih sering menggunakan virtualhost ketika mengembangkan web dengan Laravel. Berikut cara membuat virtual host:

## MAMP

1. Menggunakan MAMP Pro, buka menu **Hosts**
2. Klik tombol [+]
3. Isi bagian **Server name** dengan url yang kita inginkan
4. Isi **Disk Location** dengan alamat folder public di webapp
5. Klik **Apply**
6. Klik **Start** untuk merestart server apache.



Setup VirtualHost di MAMP PRO

## XAMPP

1. Buka file hosts yang ada di alamat C:\WINDOWS\system32\drivers\etc\hosts
2. Di bagian paling bawah tambahkan alamat IP Address localhost 127.0.0.1 dan nama domain yang dibuat misalnya webapp.site

C:\WINDOWS\system32\drivers\etc\hosts

---

```

1   ....
2   127.0.0.1      webapp.site
3   ....

```

---

3. Buka file httpd.conf yang ada di alamat C:\xampp\apache\conf\httpd.conf
4. Cari bagian Directory, jika aplikasi kita berada di C:/xampp/htdocs/webapp isi seperti ini

C:\xampp\apache\conf\httpd.conf

```
1 <Directory "C:/xampp/htdocs/webapp/public">
2   Options Indexes FollowSymLinks Includes ExecCGI
3   AllowOverride All
4   Order allow,deny
5   Allow from all
6   Require all granted
7 </Directory>
```

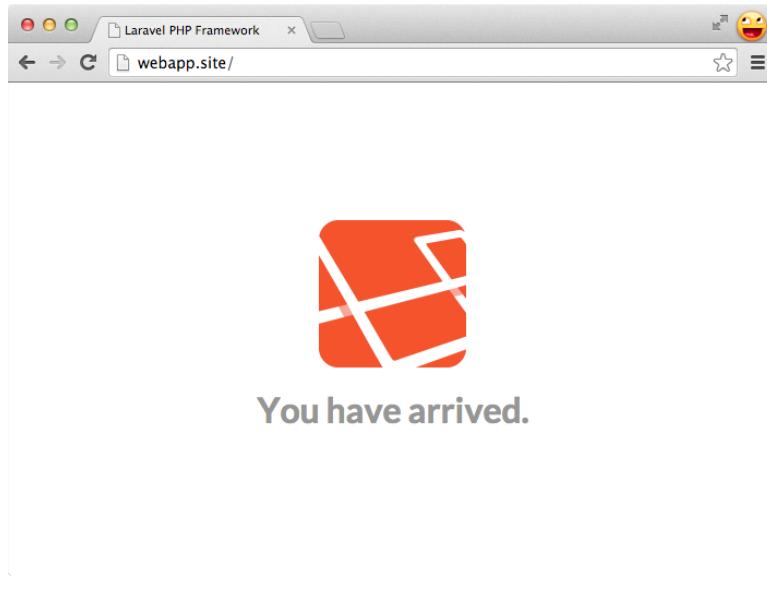
5. Buka file httpd-vhosts.conf yang ada di alamat C: \xampp\apache\conf\extra
6. Tambahkan setingan di bawah ini untuk membedakan website yang dipanggil dengan localhost dan website yang dipanggil dengan virtual host

C:\xampp\apache\conf\extra\httpd-vhosts.conf

```
1 NameVirtualHost *:80
2
3 #VirtualHost untuk webapp.site
4
5 <VirtualHost *:80>
6   DocumentRoot C:/xampp/htdocs/webapp/public
7   ServerName webapp.site
8 </VirtualHost>
9
10 #Untuk localhost yang biasa
11
12 <VirtualHost *:80>
13   DocumentRoot C:/xampp/htdocs
14   ServerName localhost
15 </VirtualHost>
```

7. Restart Apache pada XAMPP Control Panel dengan klik tombol **stop** kemudian klik tombol **start**.

Setelah berhasil, Anda dapat mengakses aplikasi di <http://webapp.site>.



Berhasil setup virtualhost

## Ringkasan

Di Hari 1 ini, saya harap Anda telah memahami bagaimana melakukan setup sebuah project Laravel, poin-poin yang telah kita bahas yaitu:

- Text Editor yang digunakan
- Penggunaan composer untuk development php modern
- Instalasi laravel
- Konfigurasi virtualhost

Pada hari 2 kita akan mempelajari konsep Routing dan MVC pada Laravel. Semangat! :)

# Hari 2 : 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>28</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 meletakan 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`:

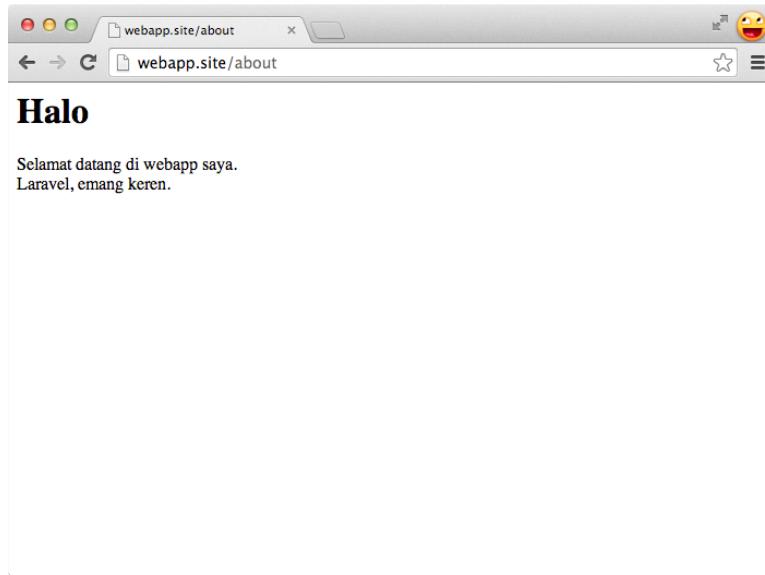
---

<sup>28</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>.

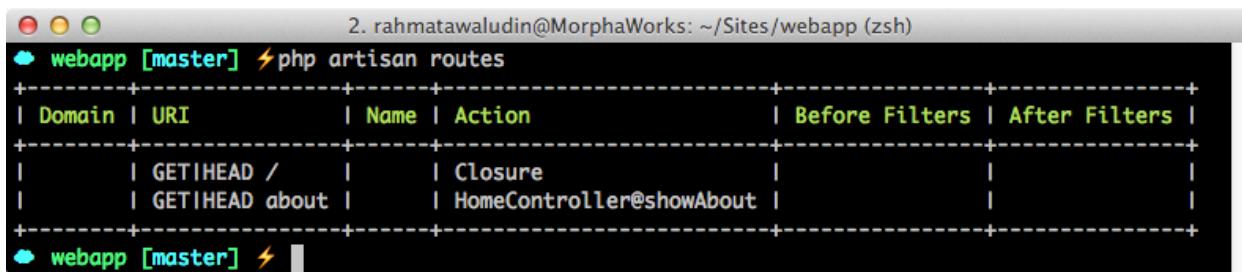
**Berhasil membuat route about**

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 | 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>29</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>30</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>31</sup>](#). Laravel memiliki fitur menarik untuk manajemen database, diantaranya [migrations dan seeding<sup>32</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>29</sup><http://laravel.com/docs/routing>

<sup>30</sup><http://id.wikipedia.org/wiki/MVC>

<sup>31</sup><http://laravel.com/docs/database>

<sup>32</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/2014\_03\_26\_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.

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.

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<sup>33</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. Terlebih jika kita ingin me-reset database ke kondisi sesuai sample data ini. 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>33</sup><http://laravel.com/docs/migrations>

---

app\database\seedsDatabaseSeeder.php

---

```

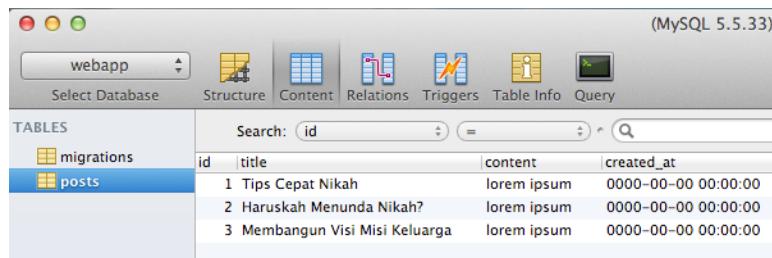
1 <?php
2
3 class DatabaseSeeder extends Seeder {
4
5     /**
6      * Run the database seeds.
7      *
8      * @return void
9      */
10     public function run()
11     {
12         Eloquent::unguard();
13
14         $this->call('PostsTableSeeder');
15     }
16 }
17 }
```

---

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 Workbench interface with the 'posts' table selected. The table has columns: id, title, content, and created\_at. The data is as follows:

	id	title	content	created_at
1	1	Tips Cepat Nikah	lorem ipsum	0000-00-00 00:00:00
2	2	Haruskah Menunda Nikah?	lorem ipsum	0000-00-00 00:00:00
3	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 asignment
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 testmodel 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/testmodel><sup>34</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>34</sup><http://webapp.site/testmodel>

```

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<sup>35</sup>](#) dan [eloquent<sup>36</sup>](#).

## View

View atau istilah lainnya *presentation logic* berfungsi untuk menampilkan data yang telah kita olah di business 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>35</sup><http://laravel.com/docs/queries>

<sup>36</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>37</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>38</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>37</sup><http://laravel.com/docs/templates>

<sup>38</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>39</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>40</sup>](#).

## Controller

Pada contoh-contoh sebelumnya, saya selalu meletakan 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 praktikan, 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>39</sup><http://laravel.com/docs/html>

<sup>40</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<sup>41</sup>](#).

## Ringkasan

Pada hari 2 ini, saya harap Anda telah memahami MVC dan model sebuah aplikasi dalam framework Laravel, poin-poin yang telah kita bahas yaitu:

- Konsep routing
- Konsep MVC
- Migrasi
- Database Seeder

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

---

<sup>41</sup><http://laravel.com/docs/controllers>

# Hari 3 : Persiapan Project

Kita akan membangun aplikasi perpustakaan dengan framework Laravel dengan nama Larapus (singkatan dari Laravel Perpus). Sengaja saya memilih ide aplikasi perpustakaan agar alur bisnisnya lebih mudah dipahami oleh pembaca. Setidaknya, setiap pembaca saya rasa sudah memahami cara kerja sebuah perpustakaan. Secara garis besar, fitur-fitur yang ada di aplikasi ini adalah:

- Fitur administrasi Buku untuk Admin
- Fitur administrasi User untuk Admin
- Fitur administrasi untuk Registered User
- Fitur peminjaman buku untuk Registered User
- Fitur browsing buku untuk non-member

## Design tampilan

Mockup dari tampilan yang akan dibuat seperti berikut:

## Tampilan Halaman Depan untuk Guest

The screenshot shows a web browser window with the title 'Perpustakaan Online dengan larapus.site'. The address bar displays 'larapus.site/'. The page itself is titled 'Daftar Buku' and contains a table of book entries. At the top right, there are 'Login | Daftar' links. Below the table, navigation links for 'First', 'Previous', 'Next', and 'Last' are visible.

Judul	Jumlah	Stok	Penulis	Aksi
Kupinang Engkau dengan Hamdalah	3	3	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	3	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">Pinjam</a>

Show 10 entries Search:

Showing 1 to 4 of 4 entries

First Previous **1** Next Last

Halaman Depan untuk Guest

## Halaman Admin

The screenshot shows a web browser window titled "Buku | Laravel Perpus". The URL in the address bar is "larapus.site/admin/books". The page header includes "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", and "Admin Larapus". Below the header, a breadcrumb navigation shows "Dashboard / Buku". The main content area is titled "Buku" with a "Tambah" button. It features a table with columns: Judul, Jumlah, Stok, and Penulis. The table contains four entries:

Judul	Jumlah	Stok	Penulis	edit	delete
Kupinang Engkau dengan Hamdalah	3	3	Mohammad Fauzil Adhim	<a href="#">edit</a>	<a href="#">delete</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">edit</a>	<a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	3	Aam Amiruddin	<a href="#">edit</a>	<a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">edit</a>	<a href="#">delete</a>

At the bottom, it says "Showing 1 to 4 of 4 entries" and has navigation buttons for First, Previous, Next, Last, and a page number "1".

Halaman Data Buku

The screenshot shows a web application interface for managing authors. At the top, there's a navigation bar with links for LaraPus, Dashboard, Buku, Penulis (which is the active tab), Member, Peminjaman, and Admin Larapus. Below the navigation is a breadcrumb trail: Dashboard / Penulis. A prominent blue button labeled "Tambah" is positioned above a table. The table has a header row with a single column labeled "Nama". Below the header, three rows of data are listed, each containing a name and two buttons: "edit" and "delete". At the bottom of the table, it says "Showing 1 to 3 of 3 entries". To the right of the table, there are navigation links for First, Previous, Next, and Last, with the number "1" highlighted.

Nama	
Mohammad Fauzil Adhim	<a href="#">edit</a> <a href="#">delete</a>
Salim A. Fillah	<a href="#">edit</a> <a href="#">delete</a>
Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>

Show 10 entries Search:

Showing 1 to 3 of 3 entries

First Previous **1** Next Last

Halaman Data Penulis

The screenshot shows a web browser window titled "Data Peminjaman | Laravel". The URL is "larapus.site/admin/borrow". The page header includes "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", and "Admin Larapus". Below the header, the breadcrumb navigation shows "Dashboard / Data Peminjaman". The main title is "Data Peminjaman". A search bar and a dropdown for "Status" (Semua) are present. The table has columns: "Buku", "Peminjam", "Tanggal Pinjam", and "Tanggal Kembali". The data in the table is as follows:

Buku	Peminjam	Tanggal Pinjam	Tanggal Kembali
Kupinang Engkau dengan Hamdalah	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Cinta & Seks Rumah Tangga Muslim	Shidqi Abdullah Mubarak	2014-05-29	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>
Membingkai Surga dalam Rumah Tangga	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>

At the bottom, it says "Showing 1 to 5 of 5 entries" and has navigation buttons: First, Previous, **1**, Next, Last.

Halaman Data Peminjaman Buku

The screenshot shows a web browser window titled "Member | Laravel Perpus". The address bar contains "larapus.site/admin/users". The page header includes "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", and "Admin Larapus". Below the header, the breadcrumb navigation shows "Dashboard / Member". The main content area is titled "Member" and displays a table of user data. The table has columns: Nama, Email, and Login Terakhir. A search bar and a "Show 10 entries" dropdown are at the top left. At the bottom, there are links for "First", "Previous", "Next", and "Last", along with a page number "1".

Nama	Email	Login Terakhir
Shidqi Abdullah Mubarak	shidqi.abdullah@gmail.com	2014-05-29 06:21:58

Show 10 entries Search:

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

Halaman Data User

## Halaman User

The screenshot shows a web browser window with the URL [larapus.site/signup](http://larapus.site/signup). The page has a light gray header with the text "LaraPus" and "Login | Daftar". Below the header, there are five input fields labeled "Nama Depan", "Nama Belakang", "Email", "Password", and "Konfirmasi Password". Each field has a placeholder text: "Nama depan Anda", "Nama belakang Anda", "emailmu@website.com", "\*\*\*\*\*", and "\*\*\*\*\*" respectively. Below these fields is a reCAPTCHA box containing the text "origaof their" and a text input field for "Ketikan teks diatas". To the right of the reCAPTCHA box are links for "Privacy & Terms" and "Daftar".

LaraPus

Login | Daftar

Nama Depan

Nama depan Anda

Nama Belakang

Nama belakang Anda

Email

emailmu@website.com

Password

Konfirmasi Password

Ketikan teks diatas

Privacy & Terms

Daftar

Halaman Registrasi User

The screenshot shows a web browser window with the URL [larapus.site/login](http://larapus.site/login). The page title is "LaraPus". On the right, there are links for "Login | Daftar" and a smiley face icon. The main content area contains two input fields labeled "Email" and "Password", a blue "Login" button, and a link "Anda lupa password?".

### Halaman Login

The screenshot shows a web browser window with the URL [larapus.site/forgot](http://larapus.site/forgot). The page title is "LaraPus". On the right, there are links for "Login | Daftar". The main content area has a heading "Masukkan email Anda", an "Email" input field containing "emailmu@website.com", a reCAPTCHA field with the text "Antonia *scratched*", a text input field for "Ketikan teks diatas", a "Privacy & Terms" link, and a large blue "Reset" button.

### Halaman Lupa Password

Selamat datang di Larapus.

Login Terakhir 1 second ago

Buku dipinjam

- Jalan Cinta Para Pejuang Kembalikan
- Membingkai Surga dalam Rumah Tangga Kembalikan

Halaman Dashboard dan data peminjaman buku

Pilih buku | Laravel Perpus

larapus.site/books

LaraPus Dashboard Buku Profil Shidqi Abdullah Mubarak

# Pilih buku

Show 10 entries Search:

Judul	Jumlah	Stok	Penulis	
Kupinang Engkau dengan Hamdalah	3	3	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	3	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">Pinjam</a>

Showing 1 to 4 of 4 entries

First Previous **1** Next Last

Halaman Peminjaman Buku

The screenshot shows a web browser window with the title 'Profil | Laravel Perpus'. The address bar contains 'larapus.site/profile'. The navigation menu includes 'LaraPus', 'Dashboard', 'Buku', and 'Profil'. The current page is 'Profil', with the user's name 'Shidqi Abdullah Mubarak' displayed. Below the menu, a breadcrumb trail shows 'Dashboard / Profil'. The main content area features a large title 'Profil' and a blue button labeled 'Perbaharui'. A table displays three user details: Name (Shidqi Abdullah Mubarak), Email (shidqi.abdullah@gmail.com), and Last Login (2014-05-31 01:58:29). The browser interface includes standard OS X window controls and a toolbar with icons for search, star, and more.

Nama	Shidqi Abdullah Mubarak
Email	shidqi.abdullah@gmail.com
Login Terakhir	2014-05-31 01:58:29

Halaman Profil User

The screenshot shows a web browser window with the title 'Ubah Password | Laravel'. The URL in the address bar is 'larapus.site/editpassword'. The page itself has a header with 'LaraPus', 'Dashboard', 'Buku', and 'Profil' links, and a user profile 'Shidqi Abdullah Mubarak'. Below the header, the breadcrumb navigation shows 'Dashboard / Ubah Password'. The main content area is titled 'Ubah Password' and contains three input fields: 'Password Lama' (with placeholder '\*\*\*\*\*'), 'Password Baru' (with placeholder '\*\*\*\*\*'), and 'Konfirmasi Password Baru' (with placeholder '\*\*\*\*\*'). At the bottom is a blue 'Simpan' button.

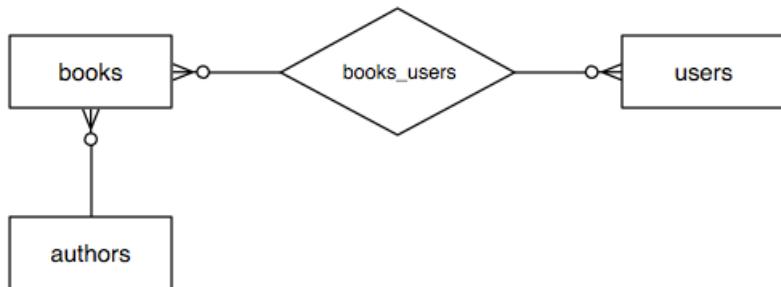
#### Halaman Ubah Password

Untuk memudahkan pembaca, html untuk design frontend dapat diakses di [bitbucket](#)<sup>42</sup>.

## Database

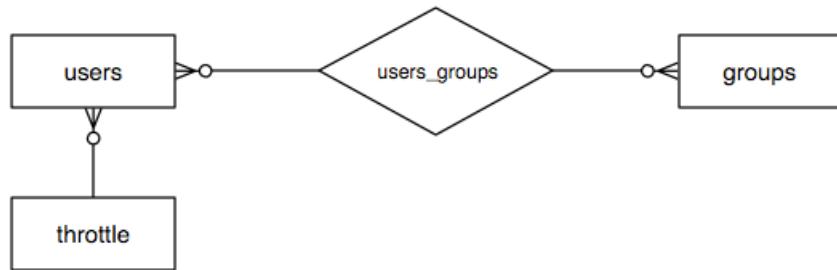
Struktur database yang akan saya gunakan untuk fungsi dasar aplikasi cukup sederhana:

<sup>42</sup><https://bitbucket.org/rahmatawaludin/larapus-frontend>



ERD core aplikasi

Sementara untuk manajemen user akan menggunakan package Sentry 2 dari Cartalyst, tentunya struktur databasenya ditentukan oleh package ini:



ERD User Management dari Sentry

Penjelasan untuk tiap entity:

- Authors : menyimpan data penulis buku
- Books : menyimpan data buku
- Users : menyimpan data users
- Groups : menyimpan data group
- Throttle : menyimpan data management user (banned, suspend, ip address, attempts)

Relasi antar entity : \* author : data buku memiliki relasi bertipe *many-to-one* antara books dengan authors  
 \* books : data buku memiliki relasi bertipe *one-to-many* antara authors dengan books \* books\_users : data peminjaman buku oleh user bertipe *many-to-many* dengan entity users dan books. \* users\_groups : data group dari user bertipe *many-to-many* dengan entity users dan groups.

Struktur database ini akan dikembangkan menggunakan migrations. Jadi, Anda tidak perlu membuat struktur database ini sekarang.

## Package

Dalam mengembangkan aplikasi ini, akan digunakan beberapa package untuk fitur-fitur dasar sebuah web:

- [Sentry](#)<sup>43</sup> : Untuk manajemen user.
- [UI Kit](#)<sup>44</sup> : Untuk design tampilan.
- [Jquery](#)<sup>45</sup> : Digunakan sebagai dependency dari UIKIT.
- [Laravel 4 Generators](#)<sup>46</sup> : untuk mengotomatisasi pembuatan Model, Controller, Migrasi, View dll.

## Server

Untuk memudahkan pengembangan lakukan hal berikut:

- Buat aplikasi laravel baru bernama larapus.

```
$ composer create-project laravel/laravel --prefer-dist larapus
```

- Setup koneksi ke database untuk aplikasi.
- Tambahkan package yang dibutuhkan ke aplikasi ke `composer.json` di bagian `require` :

### composer.json

---

```

1   ....
2   "require": {
3     "laravel/framework": "4.1.*",
4     "cartalyst/sentry": "2.0.*",
5     "uikit/uikit": "2.5.*",
6     "components/jquery": "2.1.0",
7     "way/generators": "2.6.*"
8   },
9   ....

```

---

- Untuk memberitahu lokasi kita akan meletakkan asset dari package `components/jquery` tambahkan baris berikut pada bagian `config` di `composer.json` :

### composer.json

---

```

1   ....
2   "config": {
3     "preferred-install": "dist",
4     "component-dir": "public/components"
5   },
6   ....

```

---

- Update composer untuk mendownload package :

---

<sup>43</sup><https://cartalyst.com/manual/sentry/>

<sup>44</sup><http://getuikit.com/>

<sup>45</sup><http://jquery.com/>

<sup>46</sup><https://github.com/JeffreyWay/Laravel-4-Generators>

```
$ composer update
```

- Setelah melakukan langkah ini, pastikan di folder public ada folder baru bernama components/jquery
- Buat *virtual host* untuk aplikasi tersebut di `http://larapus.site`.
- Atur koneksi ke database di `app/config/database.php`, dalam contoh ini saya menggunakan database mysql dengan database larapus, username root dan password toor.

#### app/config/database.php

---

```

1   ....
2   'default' => 'mysql',
3   'connections' => array(
4       ....
5       'mysql' => array(
6           'driver'      => 'mysql',
7           'host'        => 'localhost',
8           'database'    => 'larapus',
9           'username'    => 'root',
10          'password'    => 'toor',
11          'charset'     => 'utf8',
12          'collation'   => 'utf8_unicode_ci',
13          'prefix'      => '',
14          'unix_socket' => '/Applications/MAMP/tmp/mysql/mysql.sock',
15      ),
16      ....
17  ),

```

---

baris `unix_socket` diperlukan jika Anda menggunakan MAMP pada OSX. Jika Anda tidak menggunakan MAMP, tidak perlu diisi.

## Code sample

Untuk memudahkan pembaca, semua syntax yang saya tulis selama pengembangan aplikasi dapat diakses di [bitbucket](#)<sup>47</sup>.

## Layout

Dalam mengembangkan aplikasi ini kita akan menggunakan fasilitas layouting dari Laravel. Pada tahap ini, selain mendesain layout kita juga akan belajar mengenai penggunaan blade untuk coding view di Laravel.

Sebelum memulai mendesain layout, silahkan publish css, js, dan font dari uiikit ke folder public dengan perintah:

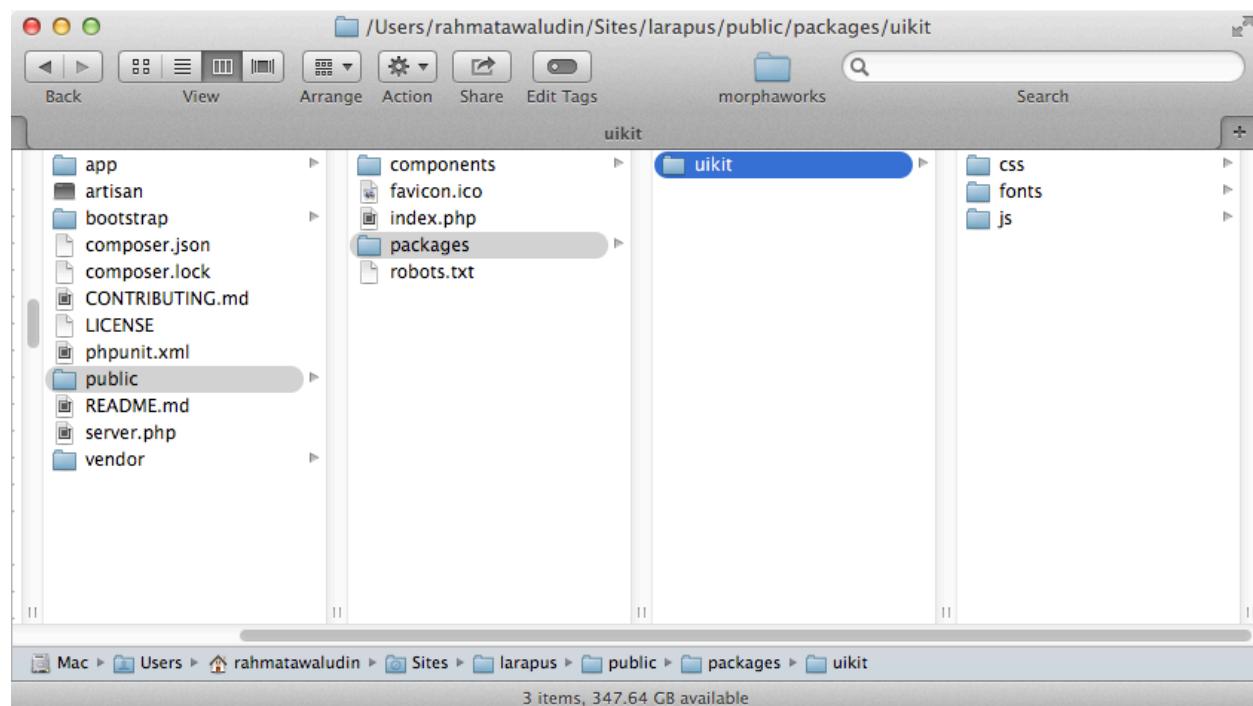
---

<sup>47</sup><https://bitbucket.org/rahmatawaludin/laravel-perpustakaan>

```
$ php artisan asset:publish --path="vendor/uikit/uikit/dist" uikit
```

Anda juga perlu mempublish jquery ke folder public

Setelah perintah diatas dijalankan, pastikan terdapat folder packages/uikit di dalam folder public.



Berhasil mempublish asset dari UIKIT

Laravel mendukung penggunaan templating dalam mengatur view dari aplikasi. Dalam aplikasi yang kita buat, ada dua buah layout utama yang kita gunakan, yaitu layout guest untuk non-authenticated user dan layout master untuk user yang sudah login.

## Membuat layout guest

Layout ini akan digunakan oleh dua halaman, yaitu halaman browsing buku dan login user. Untuk membuat layout ini, buatlah file baru di app/views/layouts/guest.blade.php isi dengan syntax sebagai berikut:

app/views/layouts/guest.blade.php

---

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Perpustakaan Online dengan Framework Laravel</title>
5     <link rel="stylesheet" href="{{ asset('packages/uikit/css/uikit.almost-flat.min.cs\
6 s') }}" />
7     <script src="{{ asset('components/jquery/jquery.min.js') }}"></script>
8     <script src="{{ asset('packages/uikit/js/uikit.min.js') }}"></script>
```

```

9   </head>
10  <body>
11    <div class="uk-container uk-container-center uk-margin-top">
12      <nav class="uk-navbar">
13          <a href="#" class="uk-navbar-brand uk-hidden-small">LaraPus</a>
14          <div class="uk-navbar-flip uk-navbar-content">
15              <a class="" href="#">Login</a> |
16              <a class="" href="#">Daftar</a>
17          </div>
18          <div class="uk-navbar-brand uk-navbar-center uk-visible-small">LaraPus</div>
19      </nav>
20      <div class="uk-container-center uk-margin-top">
21          @yield('content')
22      </div>
23  </div>
24
25  </body>
26 </html>

```

---

Penjelasan singkat dari file ini:

- Layout ini menggunakan [helpers<sup>48</sup>](#) dari Laravel untuk menghasilkan URL ke file css/js dari folder public pada baris 5-7.
- Layout ini menggunakan fitur [section dari blade<sup>49</sup>](#) untuk memudahkan mengisi konten pada baris 19. Konten disini akan dinamis tergantung isian dari child viewnya.

## Contoh Pembuatan Child View

Saya akan memberikan contoh bagaimana membuat childview dari layout sudah dibuat, kedepannya silahkan lakukan langkah yang sama untuk membuat child view. Pada contoh ini, saya akan membuat view untuk browse buku.

Untuk membuatnya, kita perlu meng-*extends* layout guest dan memberikan konten pada bagian content. Untuk saat ini, konten untuk table di halaman ini akan di-*hardcode*. Kedepannya akan dibuat berdasarkan data dari database. Buatlah file baru di `app/views/guest/index.blade.php` dengan isi sebagai berikut:

---

<sup>48</sup><http://laravel.com/docs/helpers#urls>

<sup>49</sup><http://laravel.com/docs/templates#blade-template-inheritance>

app/views/guest/index.blade.php

```
1 @extends('layouts.guest')
2
3 @section('content')
4     <h1 class="uk-heading-large">Daftar Buku</h1>
5     <table class="uk-table uk-table-striped uk-table-hover">
6         <thead>
7             <tr>
8                 <th>Judul</th>
9                 <th>Penulis</th>
10                <th>Stok</th>
11                <th></th>
12            </tr>
13        </thead>
14        <tbody>
15            <tr>
16                <td>Kupinang Engkau dengan Hamdalah</td>
17                <td>Mohammad Fauzil Adhim</td>
18                <td>2/3</td>
19                <td><a href="#">Pinjam buku</a></td>
20            </tr>
21            <tr>
22                <td>Jalan Cinta Para Pejuang</td>
23                <td>Salim A. Fillah</td>
24                <td>0/3</td>
25                <td><span class="uk-text-muted">Pinjam buku</span></td>
26            </tr>
27            <tr>
28                <td>Nikmatnya Pacaran Setelah Pernikahan</td>
29                <td>Salim A. Fillah</td>
30                <td>1/3</td>
31                <td><a href="#">Pinjam buku</a></td>
32            </tr>
33            <tr>
34                <td>Cinta & Seks Rumah Tangga Muslim</td>
35                <td>Aam Amiruddin</td>
36                <td>1/3</td>
37                <td><a href="#">Pinjam buku</a></td>
38            </tr>
39        </tbody>
40    </table>
41 @stop
```

Pada file ini terlihat bagaimana cara membuat child template yaitu dengan menggunakan syntax `@extends('layouts.guest')`

penulisan alamat view disesuaikan dengan letak file layout pada folder `view`. Pada contoh ini, alamat layout guest adalah `app\views\layouts\guest.blade.php` maka alamat ini diterjemahkan menjadi `layouts.guest`.

Sedangkan untuk mengisi section `content` kita gunakan syntax `@section('content')` yang diakhiri dengan `@stop`. Pastikan untuk mengakhiri dengan `@stop` agar tidak terjadi error.

Selanjutnya, untuk mengecek view ini kita akan menampilkan view melalui routes dengan cara membuat closure yang mengembalikan view dengan fungsi `View::make`. Silahkan ubah route untuk \ pada `routes.php` menjadi:

`app/routes.php`

---

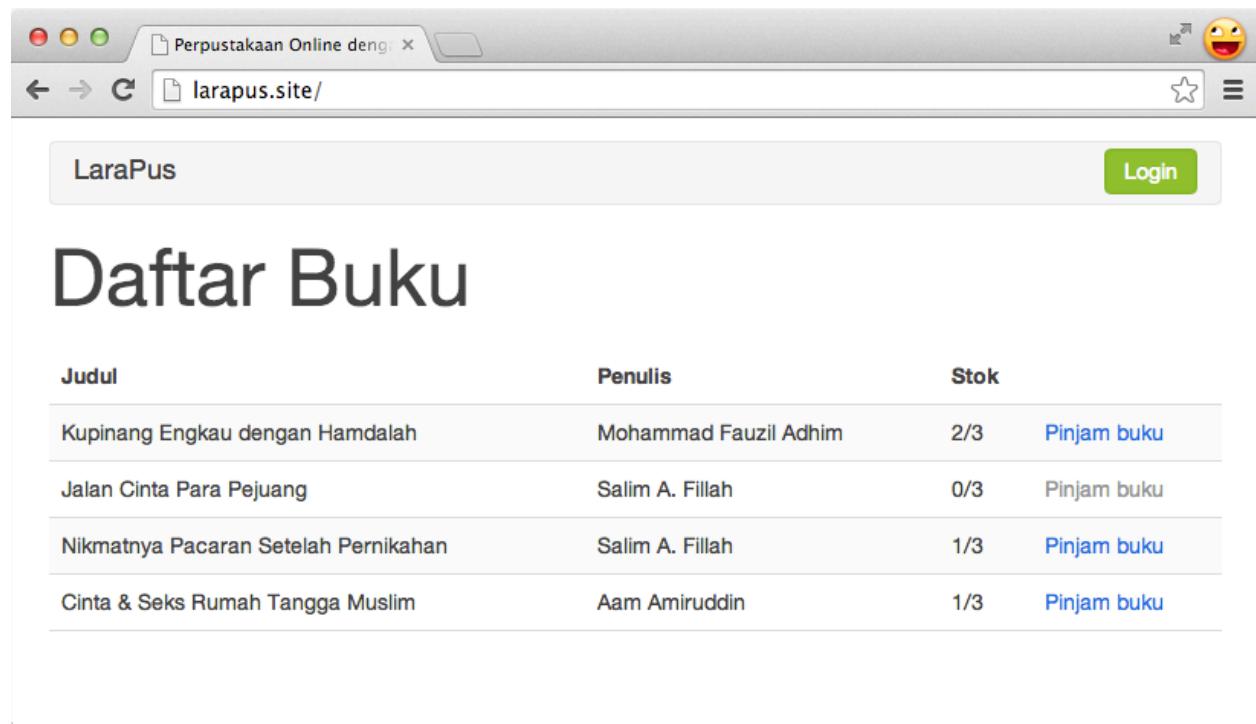
```

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

```

---

Coba akses kembali `http://larapus.site` maka halamannya akan berubah.



The screenshot shows a web browser window with the following details:

- Address Bar:** Shows the URL `larapus.site/`.
- Header:** Displays the title "LaraPus" and a "Login" button.
- Main Content:** A large heading "Daftar Buku" is centered. Below it is a table listing books:

Judul	Penulis	Stok	Aksi
Kupinang Engkau dengan Hamdalah	Mohammad Fauzil Adhim	2/3	<a href="#">Pinjam buku</a>
Jalan Cinta Para Pejuang	Salim A. Fillah	0/3	<a href="#">Pinjam buku</a>
Nikmatnya Pacaran Setelah Pernikahan	Salim A. Fillah	1/3	<a href="#">Pinjam buku</a>
Cinta & Seks Rumah Tangga Muslim	Aam Amiruddin	1/3	<a href="#">Pinjam buku</a>

Berhasil membuat layout guest

Inilah cara yang paling sederhana untuk mengecek view yang telah dibuat. Kedepannya kita akan menggunakan controller untuk mengatur view yang akan keluar, sehingga file `routes.php` menjadi lebih rapi.

## Layout Master

Ketika user sudah login baik dia admin maupun user biasa, akan digunakan layout terpisah dari layout guest. Sebenarnya, bisa saja kita menggunakan layout yang sama dengan guest, tapi saya lebih suka memisahkan layout untuk authenticated dan non-authenticated user untuk memudahkan pengembangan aplikasi. Jika diinginkan, bisa juga dibedakan layout untuk user yang login sebagai user biasa dan sebagai admin. Tapi, untuk menyederhanakan aplikasi, cukup dibuat satu layout dulu untuk authenticated user.

Buatlah file layout app/views/layouts/master.blade.php dengan konten :

app/views/layouts/master.blade.php

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>@yield('title') | Laravel Perpus</title>
5         <link rel="stylesheet" href="{{ asset('packages/uikit/css/uikit.almost-flat.min.cs\
6 s') }}" />
7         <script src="{{ asset('components/jquery/jquery.min.js') }}"></script>
8         <script src="{{ asset('packages/uikit/js/uikit.min.js') }}"></script>
9     </head>
10    <body>
11        <div class="uk-container uk-container-center uk-margin-top">
12            <nav class="uk-navbar">
13                <a href="#" class="uk-navbar-brand uk-hidden-small">LaraPus</a>
14                <ul class="uk-navbar-nav uk-hidden-small">
15                    @yield('nav')
16                </ul>
17                <div class="uk-navbar-flip uk-navbar-content">
18                    <a href="#">Admin</a> |
19                    <a href="#">Logout</a>
20                </div>
21                <div class="uk-navbar-brand uk-navbar-center uk-visible-small">LaraPus</div>
22            </nav>
23            <div class="uk-container-center uk-margin-top">
24                <ul class="uk-breadcrumb">
25                    @yield('breadcrumb')
26                </ul>
27                <h1 class="uk-heading-large">@yield('title')</h1>
28                @yield('content')
29            </div>
30        </div>
31    </body>
32 </html>
```

Pada layout master ini kita menggunakan fitur [navbar<sup>50</sup>](#) dan [breadcrumb<sup>51</sup>](#) untuk menunjukkan lokasi kita di aplikasi. Juga terdapat beberapa teks seperti Username dan beberapa link yang akan diisi pada bab selanjutnya. Selanjutnya, ubahlah isi `app/controllers/HomeController.php` menjadi :

`app/controllers/HomeController.php`

---

```

1 <?php
2 class HomeController extends BaseController {
3
4     /**
5      * Layout yang akan digunakan untuk controller ini
6      */
7     protected $layout = 'layouts.master';
8
9     public function dashboard()
10    {
11         $this->layout->content = View::make('dashboard.index')->withTitle('Dashboard');
12    }
13
14 }
```

---

Pada baris ke-7, kita menentukan layout yang akan digunakan dari controller menggunakan [Controller Layouts<sup>52</sup>](#). Dengan teknik ini, setiap view yang dibuat dari HomeController akan menggunakan layout master.

Pada baris ke-11, kita memanggil view `app/views/dashboard/index.blade.php` dan mengirimkan variable `$title` yang berisi Dashboard. Pengiriman variable dari controller dapat kita lakukan dengan banyak cara, yang kita lakukan disini adalah dengan menggunakan Magic Method. Untuk lebih lengkapnya silahkan baca dokumentasi lengkap [Variable Passing<sup>53</sup>](#).

Untuk menampilkan view ini buatlah file `app/views/dashboard/index.blade.php` dengan isi:

`app/views/dashboard/index.blade.php`

---

```

1 @section('title')
2     {{ $title }}
3 @stop
4
5 @section('nav')
6     <li><a href="#">Buku</a></li>
7     <li><a href="#">Member</a></li>
8     <li><a href="#">Peminjaman</a></li>
9 @stop
10
```

---

<sup>50</sup><http://getuikit.com/docs/navbar.html>

<sup>51</sup><http://getuikit.com/docs/breadcrumb.html>

<sup>52</sup><http://laravel.com/docs/templates#controller-layouts>

<sup>53</sup><http://laravel.com/docs/responses>

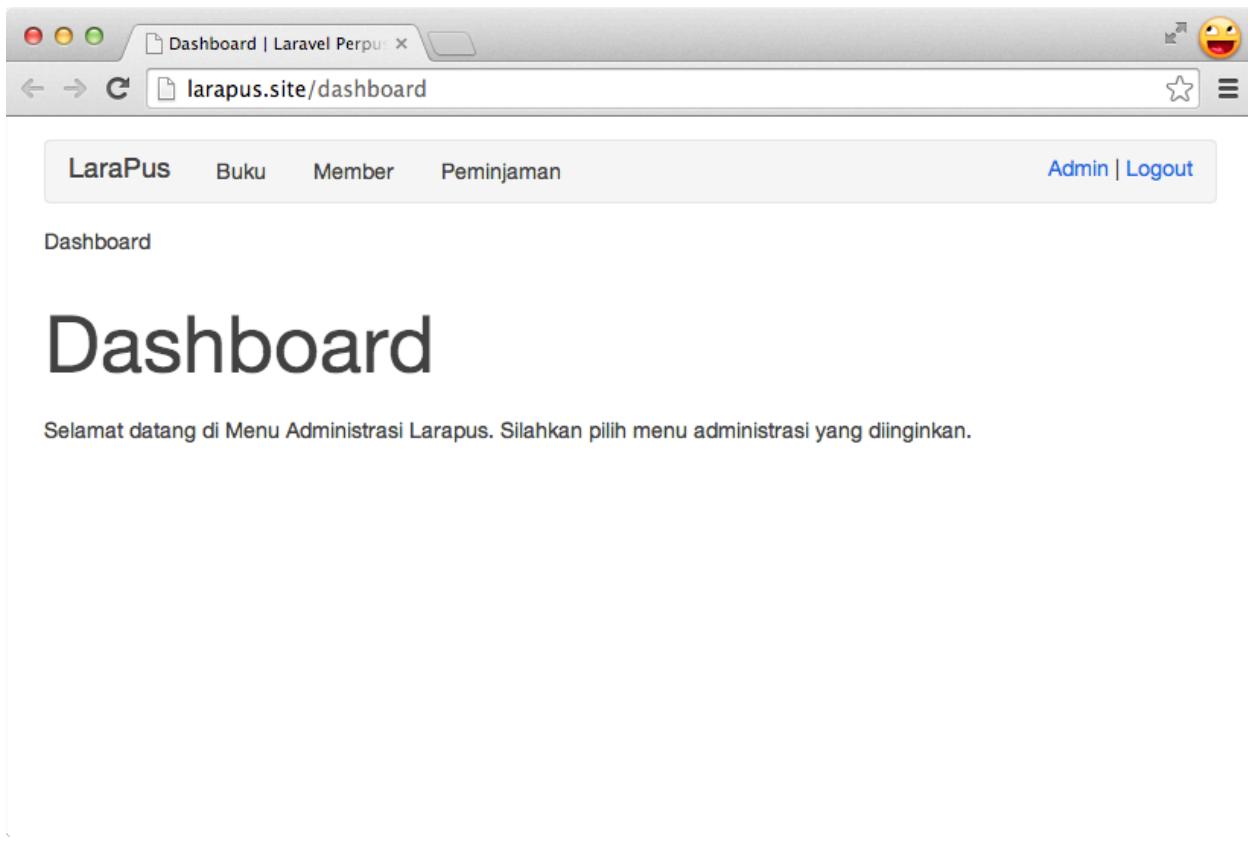
```
11 @section('breadcrumb')
12     <li>{{ $title }}</li>
13 @stop
14
15 @section('content')
16     Selamat datang di Menu Administrasi Larapus. Silahkan pilih menu administrasi yang diinginkan.
17
18 @stop
```

Untuk mengecek view ini, tambahkan baris berikut pada file app/routes.php :

app/routes.php

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

Sekarang buka alamat <http://larapus.site/dashboard> untuk mengecek view yang telah dibuat.



## Ringkasan

Di Hari 3 ini, saya harap Anda telah memahami bagaimana aplikasi yang telah dibuat, poin-poin yang telah kita bahas yaitu:

- Mempersiapkan mockup tampilan aplikasi
- Mempersiapkan struktur database
- Menginstall package yang dibutuhkan
- Mem-publish asset dari package
- Membuat virtual host untuk aplikasi
- Membuat layout untuk guest dan authenticated user

Pada hari 4, kita akan memulai implementasi dari fitur-fitur admin. Spirit! :)

# Hari 4 : Develop Fitur Admin

Keamanan adalah fitur yang sangat penting dalam membuat aplikasi. Dari sekian banyak usaha untuk mengamankan aplikasi, dua diantaranya yang saling berkaitan adalah authentikasi dan authorisasi.

*Authentikasi* dalam konteks sebuah aplikasi merupakan proses memvalidasi user pada saat memasuki sistem dengan cara memastikan username dan password yang diinputkan oleh user sesuai dengan yang tercatat di database. Dalam Laravel, fitur authentikasi merupakan fitur bawaan framework. Silahkan baca [dokumentasi lengkap authorisasi<sup>54</sup>](#) untuk memahami apa yang disediakan Laravel secara default.

*Authorisasi* merupakan proses verifikasi hak akses user terhadap resources tertentu di sistem. Ada banyak cara untuk mengatur authorisasi, salah satunya dengan menggunakan RBAC (Role Based Access Control). Dalam RBAC, seorang User diberi Role tertentu misalnya grup. Kemudian, hak akses user ini akan dicek berdasarkan group yang dia miliki.

Pada bab ini, kita akan membuat authentikasi dan autorisasi sebelum memulai mengimplementasikan fitur Admin.

## Authentikasi dan Filter

Dalam aplikasi yang kita buat, kita akan mengambil langkah yang lebih jauh untuk authentikasi dan authorisasi ini. kita akan menggunakan packages [Sentry 2 dari Catalyst<sup>55</sup>](#). Beberapa alasan kita menggunakan package ini:

- Sudah disediakan migrasi untuk model authentikasi dan authorisasi
- Mudah melakukan authentikasi
- Tersedia fitur aktivasi via email, forgot password, last login,
- Tersedia fitur group untuk user
- Tersedia fitur throttling (banned, suspended, login attempt)
- Hak akses dapat diatur di level user atau grup

Implementasi dari Sentry 2 pada sisi authentikasi dengan aktivasi user via email. Meskipun memungkinkan untuk menggunakan [access per resource<sup>56</sup>](#) pada Sentry, kita akan mengambil langkah yang lebih sederhana. Yaitu, meng-*group* user dan memberikan hak akses per grup user (Admin/Regular). Authorisasi ini dilakukan dengan melakukan [filter<sup>57</sup>](#) sebelum user mengakses sebuah fungsi controller.

## Instalasi

Untuk menginstall Sentry, ikuti langkah berikut.

1. Tambahkan baris berikut pada bagian require di composer.json :

---

<sup>54</sup><http://laravel.com/docs/security>

<sup>55</sup><https://cartalyst.com/manual/sentry>

<sup>56</sup><https://cartalyst.com/manual/sentry/permissions>

<sup>57</sup><http://laravel.com/docs/controllers#controller-filters>

**composer.json**

```
1  ....  
2  "require": {  
3      ....  
4      "cartalyst/sentry": "2.1.*",  
5  },  
6  ....
```

2. Jalankan `composer update` di terminal.
3. Tambahkan service provider `Cartalyst\Sentry\SentryServiceProvider` di `app/config/app.php` :

**app/config/app.php**

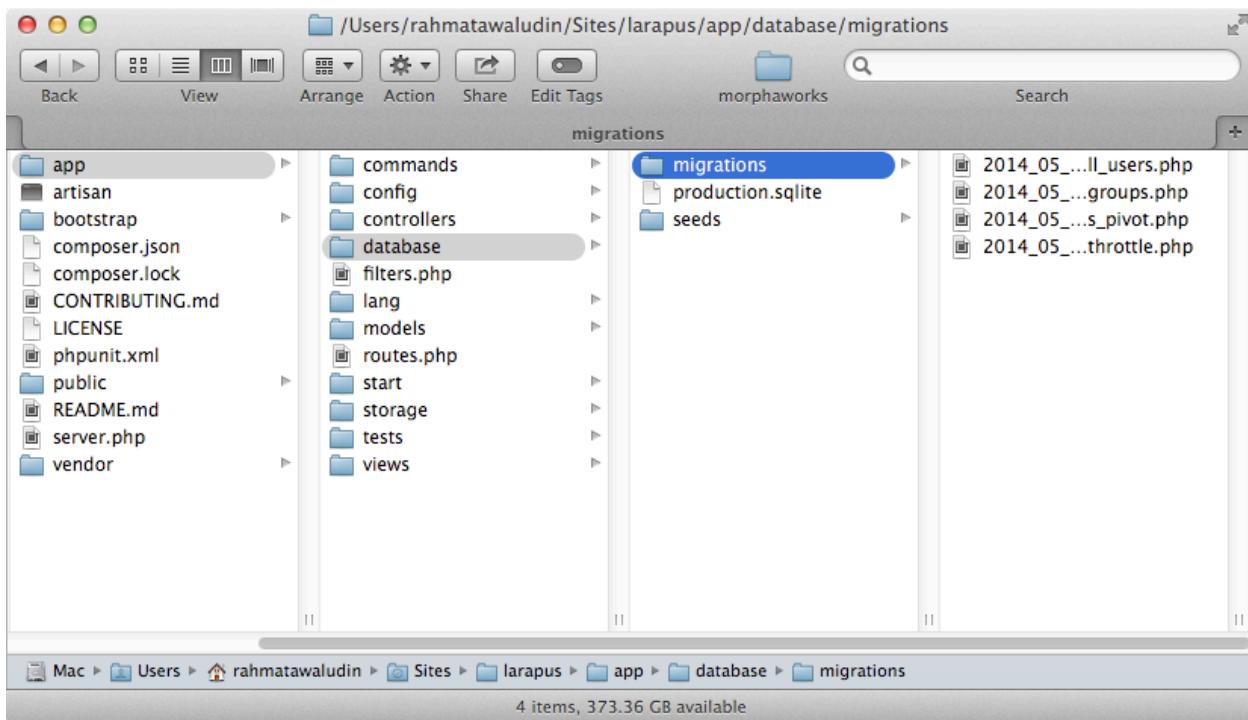
```
1  ....  
2  "providers" => array()  
3      ....  
4      'Cartalyst\Sentry\SentryServiceProvider',  
5  ),  
6  ....
```

4. Tambahkan alias untuk `Cartalyst\Sentry\Facades\Laravel\Sentry`, di `app/config/app.php` :

**app/config/app.php**

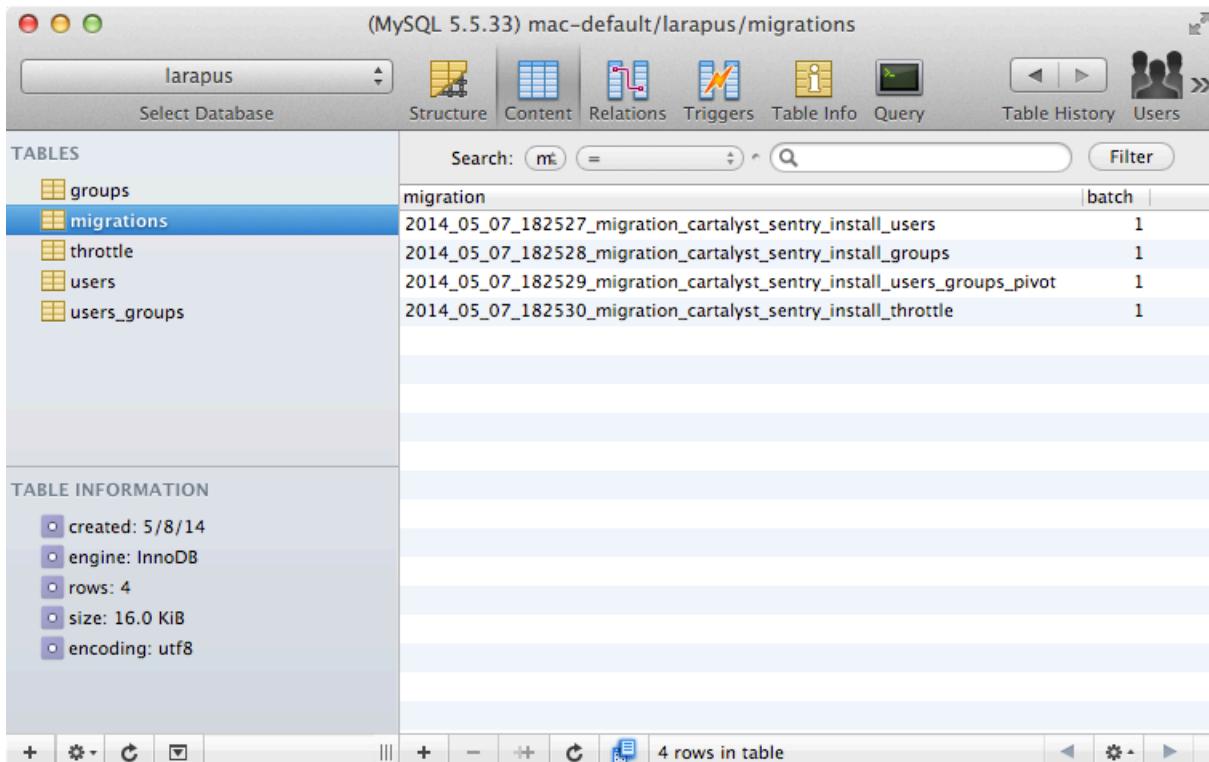
```
1  ....  
2  "aliases" => array()  
3      ....  
4      'Sentry' => 'Cartalyst\Sentry\Facades\Laravel\Sentry',  
5  ),  
6  ....
```

5. *Publish* migrasi database dari sentry dengan perintah `php artisan migrate:publish cartalyst/sentry`, pastikan folder `app/database/migrations` kini berisi 4 file migrasi baru dari Sentry.



Berhasil publish migrasi untuk Sentry

6. Jalankan migrasi database dengan perintah `php artisan migrate`, pastikan database kini berisi table migrations, userse, groups dan users\_groups.



Berhasil migrasi untuk Sentry

## Sample User

Untuk memudahkan development, mari buat database migration untuk membuat dua user masing-masing sebagai group admin dan group regular. Grup ini yang kelak akan digunakan untuk melakukan filtering user.

1. Buatlah file baru app/database/seeds/SentrySeeder.php dengan isi:

app/database/seeds/SentrySeeder.php

```

1 <?php
2
3 class SentrySeeder extends Seeder {
4
5     /**
6      * Jalankan database seeder
7      *
8      * @return void
9      */
10    public function run()
11    {
12        // Hapus isi table users, groups, users_groups dan throttle
13        DB::table('users_groups')->delete();
14        DB::table('groups')->delete();

```

```
15      DB::table('users')->delete();
16      DB::table('throttle')->delete();
17
18      try
19      {
20          // Membuat grup admin
21          $group = Sentry::createGroup(array(
22              'name'        => 'admin',
23              'permissions' => array(
24                  'admin' => 1,
25              ),
26          ));
27          // Membuat grup regular
28          $group = Sentry::createGroup(array(
29              'name'        => 'regular',
30              'permissions' => array(
31                  'regular' => 1,
32              ),
33          ));
34      }
35      catch (Cartalyst\Sentry\Groups\NameRequiredException $e)
36      {
37          echo 'Name field is required';
38      }
39      catch (Cartalyst\Sentry\Groups\GroupExistsException $e)
40      {
41          echo 'Group already exists';
42      }
43
44      try
45      {
46          // Membuat admin baru
47          $admin = Sentry::register(array(
48              // silahkan ganti sesuai keinginan
49              'email'      => 'laravel.perpustakaan@gmail.com',
50              'password'   => '#larapus2014',
51              'first_name' => 'Admin',
52              'last_name'  => 'Larapus'
53          ), true); // langsung diaktivasi
54
55          // Cari grup admin
56          $adminGroup = Sentry::findGroupByName('admin');
57
58          // Masukkan user ke grup admin
59          $admin->addGroup($adminGroup);
```

```

60
61         // Membuat user regular baru
62         $user = Sentry::register(array(
63             // silahkan ganti sesuai keinginan
64             'email'      => 'shidqi.abdullah@gmail.com',
65             'password'   => '#larapus2014',
66             'first_name' => 'Shidqi',
67             'last_name'  => 'Abdullah Mubarak'
68         ), true); // langsung diaktivasi
69
70         // Cari grup regular
71         $regularGroup = Sentry::findGroupByName('regular');
72
73         // Masukkan user ke grup regular
74         $user->addGroup($regularGroup);
75
76     }
77     catch (Cartalyst\Sentry\Users\LoginRequiredException $e)
78     {
79         echo 'Login field is required.';
80     }
81     catch (Cartalyst\Sentry\Users>PasswordRequiredException $e)
82     {
83         echo 'Password field is required.';
84     }
85     catch (Cartalyst\Sentry\Users\UserExistsException $e)
86     {
87         echo 'User with this login already exists.';
88     }
89     catch (Cartalyst\Sentry\Groups\GroupNotFoundException $e)
90     {
91         echo 'Group was not found.';
92     }
93 }
94 }
```

---

2. Daftarkan SentrySeeder yang baru dibuat dengan cara mengubah isi dari app/database/seeds/DatabaseSeeder.php menjadi :

---

app/database/seeds/DatabaseSeeder.php

---

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

---

3. Lakukan seeding database dengan perintah `php artisan db:seed` pastikan table `users`, `groups` dan `users_groups` sudah terisi dengan data yang telah disiapkan.

```
2. mysql -u root -ptoor (mysql)
mysql> select first_name, last_name, email from users;
+-----+-----+-----+
| first_name | last_name      | email           |
+-----+-----+-----+
| Admin      | Larapus        | laravel.perpustakaan@gmail.com |
| Shidqi     | Abdullah Mubarak | shidqi.abdullah@gmail.com   |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from groups;
+----+-----+-----+-----+-----+
| id | name      | permissions    | created_at      | updated_at      |
+----+-----+-----+-----+-----+
| 1  | admin     | {"admin":1}    | 2014-05-07 19:06:24 | 2014-05-07 19:06:24 |
| 2  | regular   | {"regular":1}  | 2014-05-07 19:06:24 | 2014-05-07 19:06:24 |
+----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from users_groups;
+-----+-----+
| user_id | group_id |
+-----+-----+
|       1 |       1 |
|       2 |       2 |
+-----+-----+
```

Berhasil melakukan seeding untuk Sentry

## Login & Logout

Agar User yang sudah dibuat dapat login, kita harus menyiapkan controller dan view yang menerima input username dan password. Untuk membuatnya kita akan membuat sebuah controller, view dan route yang dibutuhkan.

1. Buatlah controller untuk menampilkan halaman login di app/controllers/GuestController.php dengan isi:

**app/controllers/GuestController.php**


---

```

1 <?php
2
3 class GuestController extends BaseController {
4
5     /**
6      * Layout yang akan digunakan untuk controller ini
7      */
8     protected $layout = 'layouts.guest';
9
10    public function login()
11    {
12        $this->layout->content = View::make('guest.login');
13    }
14}
?>

```

---

2. Buatlah view untuk halaman login di app/views/guest/login.blade.php dengan isi:

**app/views/guest/login.blade.php**


---

```

1 @section('content')
2 <div class="uk-text-center">
3     <div class="uk-vertical-align-middle" style="width: 250px;">
4
5         {{ Form::open(array('url' => '/authenticate', 'class' => 'uk-panel uk-panel-box u\
6 k-form')) }}
7             <div class="uk-form-row">
8                 {{ Form::text('email', null, array('class'=>'uk-width-1-1 uk-form-large', \
9 'placeholder'=>'Email')) }}
10            </div>
11            <div class="uk-form-row">
12                {{ Form::password('password', array('class'=>'uk-width-1-1 uk-form-large'\ \
13 , 'placeholder'=>'Password')) }}
14            </div>
15            <div class="uk-form-row">
16                {{ Form::submit('Login', array('class'=>'uk-width-1-1 uk-button uk-button\ \
17 -primary uk-button-large')) }}
18            </div>
19        {{ Form::close() }}
20
21    </div>
22 </div>
23 @stop

```

---

Pada view ini, kita membuat form memiliki action ke url /authenticate. URL ini akan kita tentukan pada tahap selanjutnya.

3. Selanjutnya, kita buat fungsi authenticate di HomeController untuk melakukan authentikasi dari data username dan password yang diinputkan di halaman login. Tambahkan baris berikut setelah fungsi dashboard di app/controllers/HomeController.php :

app/controllers/HomeController.php

---

```

1   ....
2   public function authenticate()
3   {
4       // Ambil credentials dari $_POST variable
5       $credentials = array(
6           'email'      => Input::get('email'),
7           'password'  => Input::get('password'),
8       );
9
10      try {
11          // authentikasi user
12          $user = Sentry::authenticate($credentials, false);
13          // Redirect user ke dashboard
14          return Redirect::to('dashboard');
15      } catch (Exception $e) {
16          // kembalikan user ke halaman sebelumnya (login)
17          return Redirect::back();
18      }
19  }
20  ....

```

---

Pada fungsi ini, kita menerima input username dan password kemudian melakukan authentikasi menggunakan Sentry. Jika login berhasil, user akan diarahkan/redirect ke halaman dashboard. Sementara jika salah, user akan dikembalikan ke halaman sebelumnya atau halaman login. Untuk memahami lebih jauh mengenai redirect di Laravel, silahkan buka [dokumentasi resmi](#)<sup>58</sup>.

4. Selanjutnya, untuk menggabungkan semua view dan controller yang telah dibuat, ubahlah app/routes.php seperti berikut:

---

<sup>58</sup><http://laravel.com/docs/responses#redirects>

**app/routes.php**

```

1 <?php
2
3 Route::get('/', function()
4 {
5     return View::make('guest.index');
6 });
7 Route::get('dashboard', 'HomeController@dashboard');
8 Route::get('login', array('guest.login', 'uses'=>'GuestController@login'));
9 Route::post('authenticate', 'HomeController@authenticate');
10
11 ?>

```

- Pada baris ke-7, kita mengarahkan route dashboard ke fungsi dashboard di HomeController.
- Pada baris ke-8, kita menggunakan **named routes**<sup>59</sup> untuk memberikan nama routes login sebagai guest.login dan mengarahkannya ke fungsi login di GuestController. Penggunaan *named routes* akan memudahkan kita untuk memanggil routes ini dari aplikasi.
- Pada baris ke-9, kita membuat sebuah route dengan method POST bernama authenticate yang diarahkan ke fungsi authenticate di HomeController. Route ini akan berfungsi menerima data input username dan password kemudian melakukan authentikasi user.

5. Untuk memudahkan, update link untuk login di layout guest seperti berikut :

**app/views/layouts/guest.blade.php**

```

1 <a href="#" class="uk-navbar-brand uk-hidden-small">LaraPus</a>
2 <div class="uk-navbar-flip uk-navbar-content">
3     <a class="" href="#">Login</a> |
4     <a class="" href="#">Daftar</a>
5 </div>

```

Menjadi :

**app/views/layouts/guest.blade.php**

```

1 <a href="/" class="uk-navbar-brand uk-hidden-small">LaraPus</a>
2 <div class="uk-navbar-flip uk-navbar-content">
3     <a class="" href="{{ URL::to('login') }}>Login</a> |
4     <a class="" href="#">Daftar</a>
5 </div>

```

6. Setelah berhasil membuat halaman login, cobalah untuk login dengan akun yang telah dibuat pada tahapan sebelumnya. Pastikan jika login berhasil, Anda diarahkan ke halaman dashboard dan jika gagal kembali ke halaman login.
7. Selanjutnya, kita akan membuat fitur logout dari User yang telah login. Silahkan tambahkan fungsi berikut setelah fungsi authenticate pada HomeController :

<sup>59</sup><http://laravel.com/docs/routing#named-routes>

**app/controllers/HomeController.php**

```

1   ....
2   public function logout()
3   {
4       // Logout user
5       Sentry::logout();
6       // Redirect user ke halaman login
7       return Redirect::to('login');
8   }

```

Pada fungsi ini, kita me-*logout* user kemudian mengarahkannya kembali ke halaman login.

8. Selanjutnya, kita daftarkan fungsi ini ke app/routes.php :

**app/routes.php**

```

1   ....
2   Route::get('logout', 'HomeController@logout');
3   ....

```

9. Link untuk logout berada pada view app/views/layouts/master.blade.php oleh karena itu kita perlu mengupdate view ini, ubah baris:

**app/views/layouts/master.blade.php**

```

<div class="uk-navbar-flip uk-navbar-content">
    <a href="#">Admin</a> |
    <a href="#">Logout</a>
</div>

```

Menjadi :

**app/views/layouts/master.blade.php**

```

<div class="uk-navbar-flip uk-navbar-content">
    <a href="#">{{ Sentry::getUser()->first_name . ' ' . Sentry::getUser()->last_name }}< \
/a> |
    <a href="{{ URL::to('logout') }}">Logout</a>
</div>

```

Pada tahap ini, selain membuat link untuk logout, kita juga menampilkan nama user yang sedang login pada navigation bar.

10. Silahkan cek kembali fitur login dan logout ini, pastikan Anda telah di-*redirect* dengan tepat dan nama user muncul pada navigation bar.

## Flash Message

Umumnya, pada halaman login, jika user salah memasukkan password maka akan muncul pesan kesalahan. Begitupun jika user telah berhasil logout, biasanya akan muncul pesan sukses. Pada bagian ini, kita akan menggunakan fitur **flash message**<sup>60</sup> dari Laravel. Untuk lebih mudahnya, flash data akan kita gabungkan dengan **redirect**<sup>61</sup> yang dilakukan setelah login dan logout. Pada bagian ini kita juga akan belajar menggunakan fitur templating dari blade yaitu **subviews**<sup>62</sup>.

1. Buatlah view baru di `app/views/layouts/partials/alert.blade.php` dengan isi :

`app/views/layouts/partials/alert.blade.php`

---

```
@if (Session::has('successMessage'))
    <div class="uk-alert uk-alert-success" data-uk-alert>
        <a href="" class="uk-alert-close uk-close"></a>
        <p>{{ Session::get('successMessage') }}</p>
    </div>
@elseif (Session::has('errorMessage'))
    <div class="uk-alert uk-alert-danger" data-uk-alert>
        <a href="" class="uk-alert-close uk-close"></a>
        <p>{{ Session::get('errorMessage') }}</p>
    </div>
@endif
```

---

Pada file ini kita mengecek ketersediaan flash data dengan nama `successMessage` atau `errorMessage`. Jika ada, maka kita akan menampilkan **alert**<sup>63</sup> dengan konten message yang didapatkan.

2. Pada layout `app/views/layouts/guest.blade.php` tambahkan baris ini sebelum `@yield('content')` :

`app/views/layouts/guest.blade.php`

---

```
.....
@include('layouts.partials.alert')
@yield('content')
....
```

---

Syntax `@include('layouts.partials.alert')` menandakan bahwa kita akan menampilkan html view dari file `alert.blade.php` ke file `guest.blade.php`.

3. Pada layout `app/views/layouts/master.blade.php` tambahkan baris ini sebelum `<ul class="uk-breadcrumb">` :

<sup>60</sup><http://laravel.com/docs/session#flash-data>

<sup>61</sup><http://laravel.com/docs/responses#redirects>

<sup>62</sup><http://laravel.com/docs/templates#other-blade-control-structures>

<sup>63</sup><http://getuikit.com/docs/alert.html>

---

app/views/layouts/master.blade.php

---

```
.....
@include('layouts.partials.alert')
<ul class="uk-breadcrumb">
    @yield('breadcrumb')
</ul>
....
```

---

Di layout ini pun kita melakukan include ke alert.blade.php. Dengan menggunakan @include kita dapat menghemat coding dan menyeragamkan style dari alert.

4. Saatnya mengetes flash message yang kita buat. Ubah Redirect pada fungsi logout di HomeController menjadi :

---

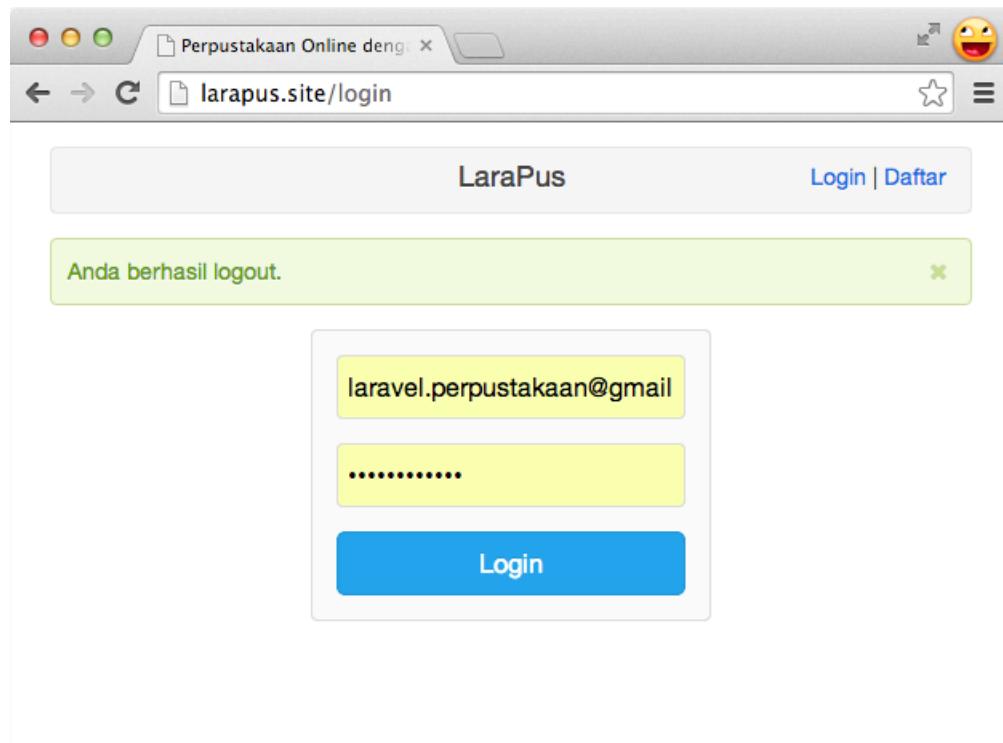
app/controllers/HomeController.php

---

```
return Redirect::to('login')->with('successMessage', 'Anda berhasil logout.');
```

---

Disini kita menggunakan redirect dengan flashdata. Cobalah login, kemudian logout kembali, pastikan flash message **Anda berhasil logout.** muncul di halaman login.



5. Selanjutnya, kita akan memberikan pesan kesalahan yang berbeda untuk login, yaitu ketika User tidak ditemukan atau ketika password salah. Untuk mendapatkan pesan yang berbeda ini, kita akan memanfaatkan *exception* yang dihasilkan oleh Sentry. Ubah bagian ini pada fungsi authenticate di HomeController :

**app/controllers/HomeController.php**

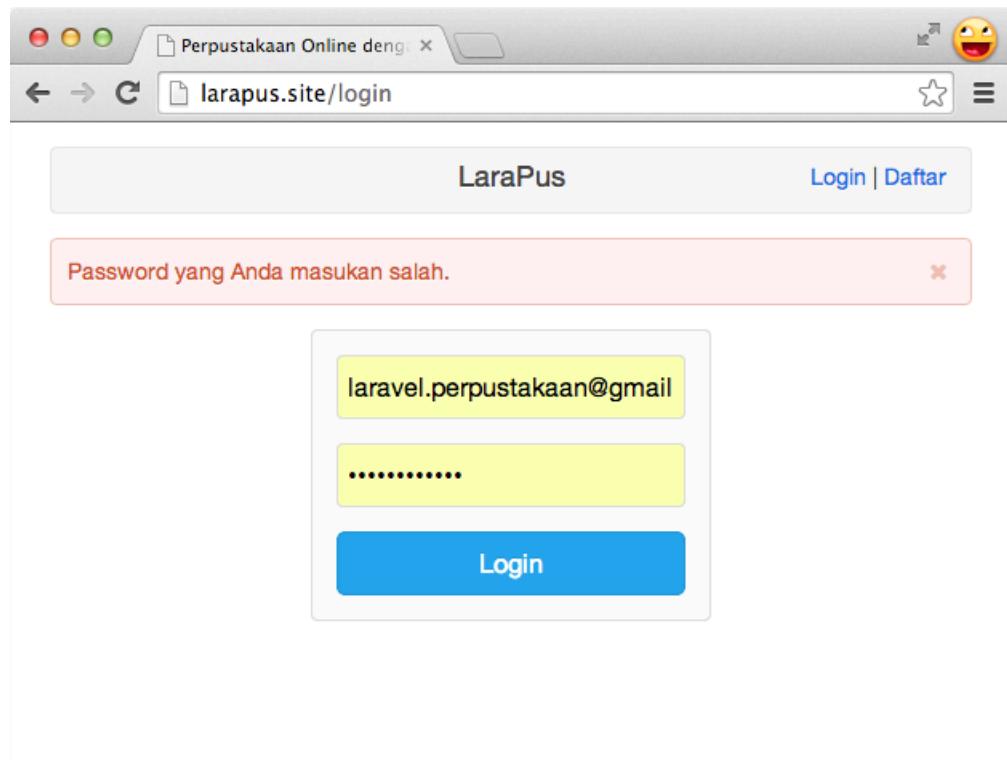
```
....  
} catch (Exception $e) {  
    // kembalikan user ke halaman sebelumnya (login)  
    return Redirect::back();  
}  
....
```

Menjadi :

**app/controllers/HomeController.php**

```
....  
} catch (Cartalyst\Sentry\Users\WrongPasswordException $e) {  
    return Redirect::back()->with('errorMessage', 'Password yang Anda masukan salah.');//  
} catch (Exception $e) {  
    return Redirect::back()->with('errorMessage', trans('Akun dengan email tersebut tidak\\  
ditemukan di sistem kami.'));  
}  
....
```

Cobalah login dengan username atau password yang salah, dan cek flash message yang keluar.



Berhasil membuat flash message error

## Filter

Filter<sup>64</sup> dalam Laravel berguna untuk membatasi akses terhadap route yang telah dibuat. Penggunaan filter memungkinkan kita untuk memaksa seorang user untuk login terlebih dahulu sebelum mengakses halaman. Filter juga memungkinkan kita untuk [membatasi akses ke controller](#)<sup>65</sup> berdasarkan hak akses user. Filter yang dibuat dapat disimpan di `app/filters.php`. Anda dapat mengecek filters yang telah disediakan oleh Laravel pada file tersebut.

Pada bagian ini, kita akan mengarahkan user untuk login terlebih dahulu sebelum mengakses dashboard.

1. Ubahlah filter auth bawaaaan Laravel, menjadi berikut :

`app/filters.php`

---

```
.....
Route::filter('auth', function()
{
    if (Auth::guest()) return Redirect::guest('login');
});
....
```

---

Menjadi :

`app/filters.php`

---

```
.....
Route::filter('auth', function()
{
    if ( !Sentry::check() )
    {
        return Redirect::guest('login')->with('errorMessage', 'Silahkan login terlebih da\
hulu.');
    }
});
....
```

---

Pada filter ini kita menggunakan `Sentry::check()` untuk memastikan user telah login.

Kita juga menggunakan `Redirect::guest()` untuk melakukan *redirect* sambil menyimpan url yang menjadi tujuan user. Sebagai contoh, bisa saja user mencoba mengakses halaman `help` yang dibatasi untuk *authenticated* user. Ketika user mencoba mengakses route `help` maka ia akan dihadapkan dengan halaman login, sedangkan URL tujuan sebenarnya akan disimpan pada session dengan nama `url.intended`.

2. Selanjutnya, kita akan melakukan filter untuk route dashboard. Secara umum penulisan *filter* dapat dibedakan menjadi dua, yaitu *before filter* untuk menjalankan filter sebelum mengakses route dan *after filter* untuk menjalankan filter setelah mengakses route. Pada contoh ini saya akan mencontohkan

<sup>64</sup><http://laravel.com/docs/routing#route-filters>

<sup>65</sup><http://laravel.com/docs/controllers#controller-filters>

dengan melakukan filter auth sebelum mengakses route. Untuk cara penulisan filter yang lain, silahkan merujuk ke [dokumentasi resmi](#)<sup>66</sup>. Ubah route dashboard pada app/routes.php menjadi:

**app/routes.php**

---

```
....  
Route::get('dashboard', array('before' => 'auth', 'uses' => 'HomeController@dashboard'));  
....
```

---

3. Sebagaimana dijelaskan di tahap 1, pada filter auth kita menggunakan Redirect::guest() untuk menyimpan variable tujuan ke variable session url.intended. Oleh karena itu, kita perlu merubah fungsi login agar mengarahkan ke isi url.intended, jika ada isinya. Silahkan ubah Redirect pada fungsi authenticate pada app/controllers/HomeController.php menjadi:

**app/controllers/HomeController.php**

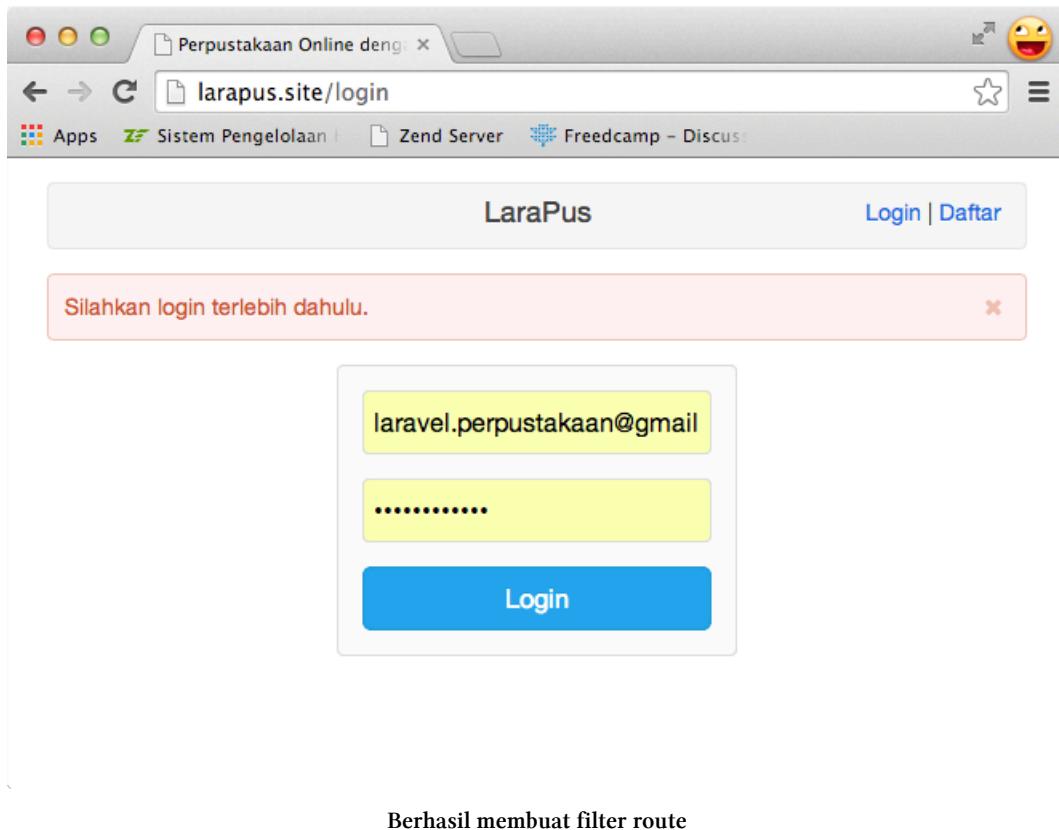
---

```
....  
// Kembalikan user ke dashboard  
return Redirect::intended('dashboard');  
....
```

---

4. Silahkan coba mengakses route dashboard pastikan Anda diarahkan ke halaman login dengan pesan error *\*Silahkan login terlebih dahulu.*

<sup>66</sup><http://laravel.com/docs/routing#route-filters>



Berhasil membuat filter route

## Fitur Admin

Untuk memudahkan dalam membuat fitur Admin, kita akan menggunakan package [Laravel-4-Generators](#)<sup>67</sup> dari Jeffrey Way. Untuk menginstallnya, ikuti langkah berikut:

1. Karena kita hanya akan menggunakan package ini selama proses pembuatan aplikasi, maka tambahkan package ini pada bagian require-dev setelah require pada app/composer.json :

app/composer.json

```

1  ....
2  'require' : {
3      ....
4  },
5  'require-dev' : {
6      "way/generators": "2.*"
7  },
8  ....

```

2. Jalankan perintah composer update --dev untuk menginstall package.

<sup>67</sup><https://github.com/JeffreyWay/Laravel-4-Generators>

3. Tambahkan 'Way\Generators\GeneratorsServiceProvider' pada array providers di app/config/app.php.
4. Untuk mengeceknya jalankan php artisan pastikan ada pilihan generate.

## CRUD Penulis

Sebuah buku dalam aplikasi ini ber-relasi ke table penulis. Pada tahap ini, kita akan membuat CRUD untuk penulis. Tentunya, kita juga harus mengatur filter agar fitur ini hanya bisa diakses oleh admin. Pada proses pembuatan CRUD ini saya akan menggunakan istilah *penulis* maupun *author* secara bergantian, namun makna keduanya sama.

### Persiapan

1. Pastikan generator telah terinstall, jalankan perintah php artisan generate:scaffold author --fields="name:string" dan pilih yes untuk semua pilihan.

```
2. rahmatawaludin@MorphaWorks: ~/sites/larapus (zsh)
~/sites/larapus ➜ master • ➜ php artisan generate:scaffold author --fields="name:string"
Do you want me to create a Author model? [yes|no]yes
Created: /Users/rahmatawaludin/Sites/larapus/app/models/Author.php
Do you want me to create views for this Author resource? [yes|no]yes
Created: /Users/rahmatawaludin/Sites/larapus/app/views/authors/index.blade.php
Created: /Users/rahmatawaludin/Sites/larapus/app/views/authors/show.blade.php
Created: /Users/rahmatawaludin/Sites/larapus/app/views/authors/create.blade.php
Created: /Users/rahmatawaludin/Sites/larapus/app/views/authors/edit.blade.php
Do you want me to create a AuthorsController controller? [yes|no]yes
Created: /Users/rahmatawaludin/Sites/larapus/app/controllers/AuthorsController.php
Do you want me to create a 'create_authors_table' migration and schema for this
resource? [yes|no]yes
Created: /Users/rahmatawaludin/Sites/larapus/app/database/migrations/2014_05_10_
031158_create_authors_table.php
Generating optimized class loader
Would you like a 'Authors' table seeder?yes
Created: /Users/rahmatawaludin/Sites/larapus/app/database/seeds/AuthorsTableSeeder.php
Do you want to go ahead and migrate the database? [yes|no]yes
Migrated: 2014_05_10_031158_create_authors_table
Done!
All done! Don't forget to add 'Route::resource('authors', 'AuthorsController');'
to app/routes.php.

~/sites/larapus ➜ master • ➜
```

Meng-generate authors

Perintah ini akan memberikan kita *model*, *views*, *controller*, *migration*, *seeder* dan menjalankan *migration* untuk author. Silahkan cek di database, pasti sudah ada table authors. Untuk mengecek fitur generator lainnya, silahkan cek di [dokumentasi generator](#)<sup>68</sup>.

2. Selanjutnya, tambahkan Route::resource('authors', 'AuthorsController'); di app/routes.php. Syntax ini akan membuat *resource controllers* yang akan memudahkan kita untuk [RESTful routing](#)<sup>69</sup>. Untuk memastikan routing untuk author ini telah dibuat, jalankan perintah php artisan routes.

Domain	URI	Name	Action	Before Filters	After Filters
	GET HEAD /		Closure		
	GET HEAD dashboard		HomeController@dashboard	auth	
	GET HEAD authors	authors.index	AuthorsController@index		
	GET HEAD authors/create	authors.create	AuthorsController@create		
	POST authors	authors.store	AuthorsController@store		
	GET HEAD authors/{authors}	authors.show	AuthorsController@show		
	GET HEAD authors/{authors}/edit	authors.edit	AuthorsController@edit		
	PUT authors/{authors}	authors.update	AuthorsController@update		
	PATCH authors/{authors}		AuthorsController@update		
	DELETE authors/{authors}	authors.destroy	AuthorsController@destroy		
	GET HEAD login		GuestController@login		
	POST authenticate		HomeController@authenticate		
	GET HEAD logout		HomeController@logout		

#### Mengecek resource route untuk Authors

Perintah artisan routes akan menampilkan semua routes yang terdaftar di aplikasi. Berikut ini, penjelasan singkat mengenai URI yang dibuat berikut tipe *request*-nya:

- /authors untuk menampilkan daftar (GET) atau menyimpan (POST) penulis.
- /authors/create untuk menampilkan form tambah penulis.
- /authors/{authors} untuk menampilkan detail penulis (GET), update data penulis (PUT/-PATCH), atau menghapus penulis (DELETE).
- /authors/{authors}/edit untuk menampilkan form edit data penulis.

Untuk memahami lebih jelas tentang URI yang dihasilkan saya sangat menyarankan untuk mengunjungi dokumentasi [resource controller](#)<sup>70</sup>.

3. Supaya ada sample data, ubah database seeder untuk authors dengan data penulis :

<sup>68</sup><https://github.com/JeffreyWay/Laravel-4-Generators>

<sup>69</sup><http://id.wikipedia.org/wiki/REST>

<sup>70</sup><http://laravel.com/docs/controllers#resource-controllers>

---

app/database/seeds/AuthorsTableSeeder.php

---

```

1 <?php
2
3 class AuthorsTableSeeder extends Seeder {
4
5     public function run()
6     {
7         // kosongkan database
8         DB::table('authors')->delete();
9
10        // buat array untuk diinput
11        $authors = [
12            ['id'=>1, 'name'=>'Mohammad Fauzil Adhim', 'created_at'=>'2014-05-01 11:15:22\',
13             'updated_at'=>'2014-05-01 11:15:22'],
14            ['id'=>2, 'name'=>'Salim A. Fillah', 'created_at'=>'2014-05-01 11:15:22', 'up\
15             dated_at'=>'2014-05-01 11:15:22'],
16            ['id'=>3, 'name'=>'Aam Amiruddin', 'created_at'=>'2014-05-01 11:15:22', 'upda\
17             ted_at'=>'2014-05-01 11:15:22'],
18        ];
19
20        // insert data ke database
21        DB::table('authors')->insert($authors);
22    }
23
24 }
```

---

4. Jangan lupa tambahkan `$this->call('AuthorsTableSeeder');` pada fungsi `run` di `app/database/seeds/DatabaseSeeder.php` untuk memudahkan kita dalam mereset data.
5. Selanjunya kita akan melakukan seeding untuk table `authors` ada dua cara:
  - Lakukan seeder hanya untuk table `authors` dengan perintah `php artisan db:seed --class="AuthorsTableSeeder"`
  - Melakukan refresh migration kemudian memberikan seeding sesuai `DatabaseSeeder.php` dengan perintah `php artisan migrate:refresh --seed`. Perintah ini juga berguna jika Anda ingin mereset kondisi database.
6. Routing yang telah kita buat dapat diakses oleh semua orang di `http://larapus.site/authors`. Untuk membuat route ini hanya dapat diakses oleh admin, dengan route `http://larapus.site/admin/authors` kita akan melakukan `route grouping`<sup>71</sup> (untuk filter auth) dan `route prefixing`<sup>72</sup>. Kita juga akan menambahkan filter `admin` sebelum mengakses resource `authors` untuk mengecek apakah User yang login adalah Admin. Sehingga isi `app/routes.php` menjadi seperti ini:

---

<sup>71</sup><http://laravel.com/docs/routing#route-groups>

<sup>72</sup><http://laravel.com/docs/routing#route-prefixing>

**app/routes.php**


---

```

1  <?php
2  Route::get('/', function()
3  {
4      return View::make('guest.index');
5  });
6  Route::group(array('before' => 'auth'), function () {
7      Route::get('dashboard', 'HomeController@dashboard');
8      Route::group(array('prefix' => 'admin', 'before' => 'admin'), function()
9      {
10         Route::resource('authors', 'AuthorsController');
11     });
12 });
13 Route::get('login', array('guest.login', 'uses'=>'GuestController@login'));
14 Route::post('authenticate', 'HomeController@authenticate');
15 Route::get('logout', 'HomeController@logout');
16 ?>

```

---

Tambahkan filter admin di app/filters.php:

**app/filters.php**


---

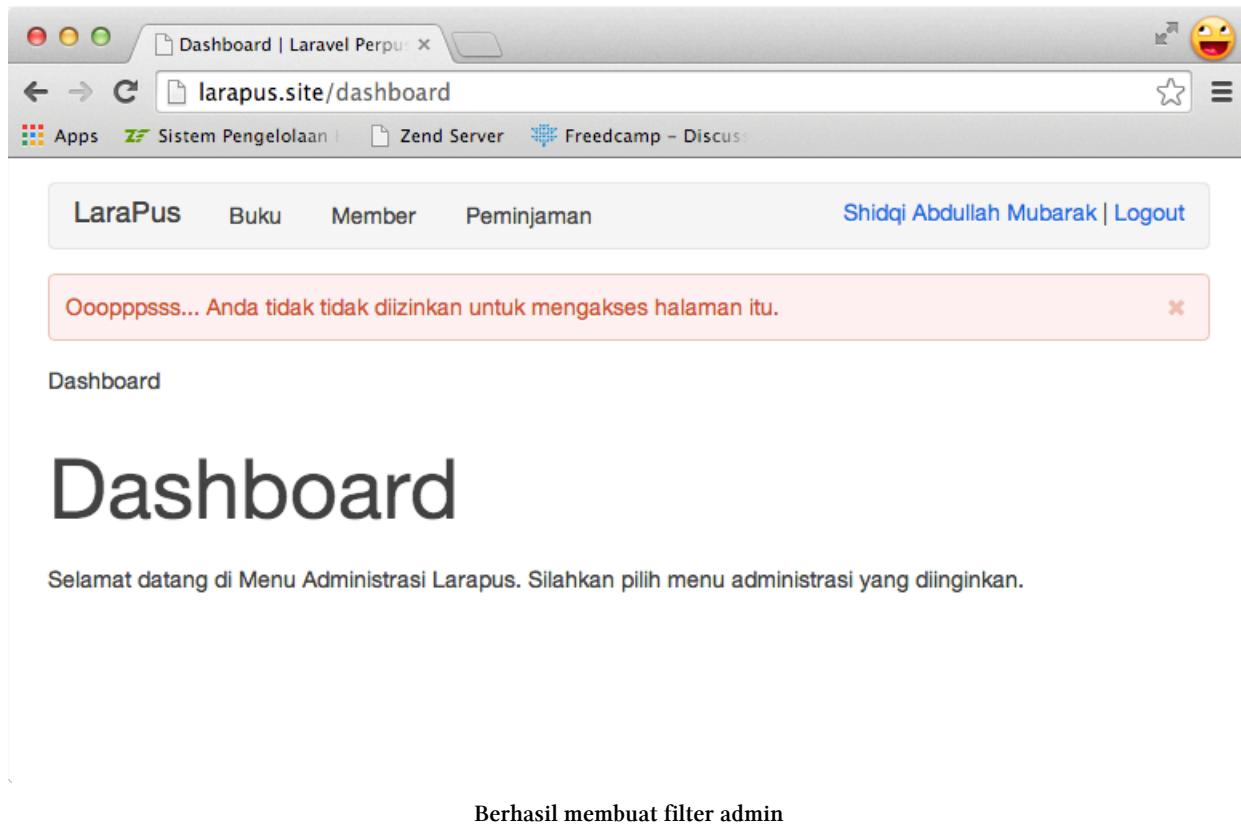
```

1  ....
2  Route::filter('admin', function()
3  {
4      $user = Sentry::getUser();
5      // Cari grup admin
6      $admin = Sentry::findGroupByName('admin');
7      if (!$user->inGroup($admin)) {
8          return Redirect::to('dashboard')->with("errorMessage", "Oooppsss... Anda tidak t\
9  idak diizinkan untuk mengakses halaman itu.");
10     }
11 });
12 ....

```

---

Dengan cara ini, setiap url untuk authors akan diawali dengan admin. Untuk mengeceknya, silahkan akses <http://larapus.site/admin/authors> dengan login sebagai bukan Admin. Anda seharusnya diarahkan ke halaman dashboard dengan pesan **Oooppsss... Anda tidak tidak diizinkan untuk mengakses halaman itu..**



## Listing Penulis

Untuk menampilkan daftar penulis yang sudah tersimpan, kita akan menggunakan [datatables<sup>73</sup>](#). Dengan menggunakan datatables, kita akan mendapatkan fitur filtering dan sorting data pada table yang ditampilkan.

1. Install dulu package yang dibutuhkan, tambahkan dua package ini pada app/composer.json :

app/routes.php

```

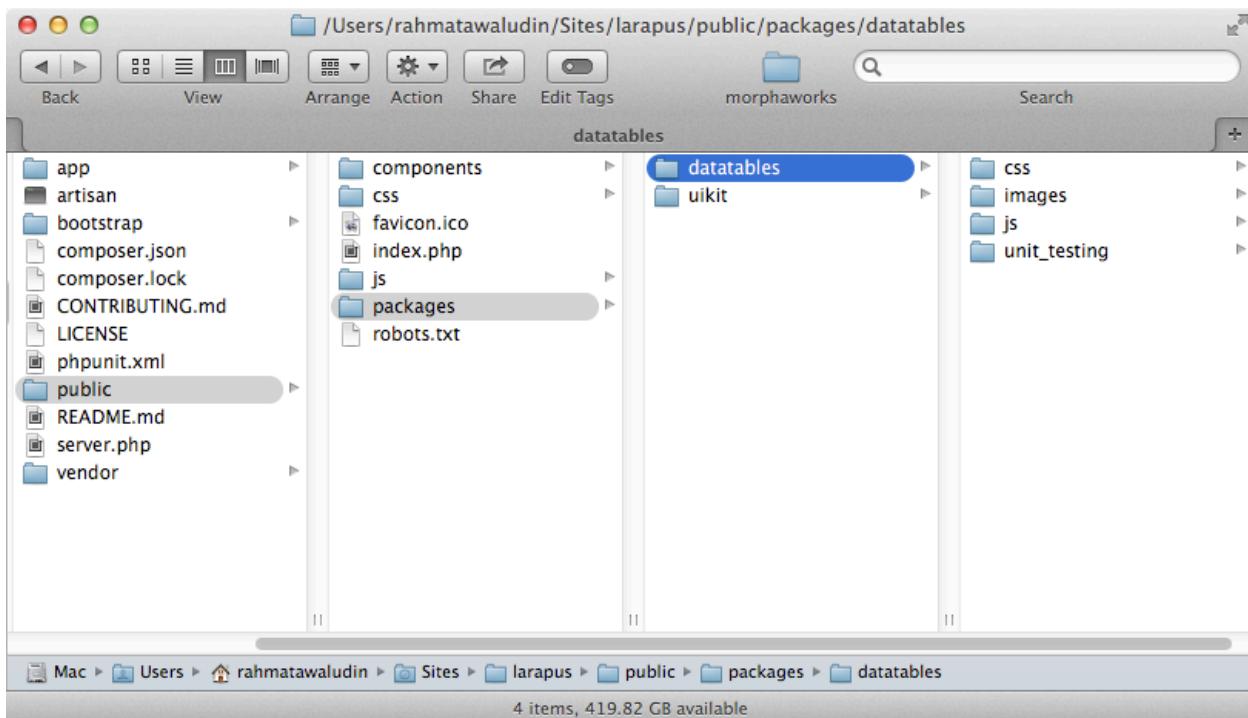
1   ....
2   "require" : {
3     ....
4     "datatables/datatables": "1.10.0",
5     "chumper/datatable": "2.2.2"
6   },
7   ....

```

2. Download package dengan menjalankan composer update.
3. Pada app/config/app.php tambahkan 'Chumper\Datatable\DatatableServiceProvider', di array providers dan 'Datatable' => 'Chumper\Datatable\Facades\DatatableFacade', pada array aliases.

<sup>73</sup><http://www.datatables.net/>

4. Selanjutnya publish asset dari datatables ke folder public dengan perintah `php artisan asset:publish datatables --path="vendor/datatables/datatables/media"`. Pastikan folder `app/public/packages/datatables` sudah dibuat.



#### Publish asset datatables

5. Siapkan Controller. Ubah fungsi index di AuthorsController menjadi :

`app/controllers/AuthorsController.php`

```

1   ....
2   public function index()
3   {
4       if(Datatable::shouldHandle())
5       {
6           return Datatable::collection(Author::all(array('id', 'name')))
7               ->showColumns('id', 'name')
8               ->addColumn('', function ($model) {
9                   return 'edit | hapus';
10                })
11               ->searchColumns('name')
12               ->orderColumns('name')
13               ->make();
14       }
15       return View::make('authors.index')->withTitle('Penulis');
16   }
17   ....

```

Pada controller ini kita mengatur agar menerima request untuk datatable dan request biasa. Untuk lebih jelas mengenai konfigurasi datatable, saya sangat menyarankan mengunjungi [dokumentasi resmi](#)<sup>74</sup>.

6. Siapkan View. Ubah app/views/authors/index.blade.php menjadi :

---

app/views/authors/index.blade.php

---

```

1  @extends('layouts.master')
2
3  @section('asset')
4      <link rel="stylesheet" href="{{ asset('packages/datatables/css/jquery.dataTables.css') \}}
5  }}" />
6      <script src="{{ asset('packages/datatables/js/jquery.dataTables.js') }}></script>
7  @stop
8
9  @section('title')
10     {{ $title }}
11 @stop
12
13 @section('breadcrumb')
14     <li>Dashboard</li>
15     <li>{{ $title }}</li>
16 @stop
17
18 @section('content')
19     Tambah Author <br>
20     {{ Datatable::table()
21         ->addColumn('id', 'Nama', '')
22         ->setOptions('aoColumnDefs', array(
23             array(
24                 'bVisible' => false,
25                 'aTargets' => [0]),
26             array(
27                 'sTitle' => 'Nama',
28                 'aTargets' => [1]),
29             array(
30                 'bSortable' => false,
31                 'aTargets' => [2])
32         )))
33         ->setOptions('bProcessing', true)
34         ->setUrl(route('admin.authors.index'))
35         ->render() }}
36
37 @stop

```

---

Terlihat diatas, kita me-*render* datatable menggunakan fungsi bawaan dari package datatable. Penje-

---

<sup>74</sup><https://github.com/Chumper/Datatable>

lasan dari opsi yang dibuat:

- `addColumn` memilih column yang akan di-*render*.
- `aoColumnDefs` mendefinisikan bagaimana tiap column akan di-*render*.
- `bVisible` menentukan apakah suatu column akan di-*render*.
- `bSortable` menentukan apakah suatu column dapat di-*sorting*.
- `bProcessing` memproses setiap filter/sorting ke backend via ajax. Hal ini berguna jika data Anda sangat banyak. Jika datanya sedikit < 100, hilangkan saja opsi ini untuk memproses filter/sorting melalui client side javascript.
- `setUrl` alamat route untuk me-*load* ajax. URL ini mengarahkan ke fungsi `index` di `AuthorsController`. Jika Anda lupa alamat route yang akan digunakan, gunakan perintah `php artisan routes`.
- `render` meng-generate html untuk datatable pada fungsi ini dapat ditambahkan template untuk datatable (jika ada).

Kita juga memanggil asset js dan css dari section `asset`. Section ini perlu ditambahkan di `app/views/layouts/master.blade.php`. Silahkan tambahkan `@yield('asset')` pada bagian `<head>`.

`app/views/authors/index.blade.php`

---

```
1  ....
2  <head>
3  ....
4  @yield('asset')
5  </head>
6  ....
```

---

7. Karena tampilan bawaan datatable belum sesuai dengan design uikit, kita perlu melakukan sedikit perubahan. Berikut ini tampilan awalnya :

Nama	
Mohammad Fauzil Adhim	<a href="#">edit</a>   <a href="#">hapus</a>
Salim A. Fillah	<a href="#">edit</a>   <a href="#">hapus</a>
Aam Amiruddin	<a href="#">edit</a>   <a href="#">hapus</a>

### Datatable before

Siapkan layout untuk datatable ini di app/views/datatable/uikit.blade.php dengan isi :

---

#### app/views/datatable/uikit.blade.php

---

```

1  <table class="uk-table uk-table-hover {{ $class }}">
2      <colgroup>
3          @for ($i = 0; $i < count($columns); $i++)
4              <col class="con{{ $i }}"/>
5      @endfor
6  </colgroup>
7  <thead>
8  <tr>
9      @foreach($columns as $i => $c)
10         <th align="center" valign="middle" class="head{{ $i }}">{{ $c }}

```

```

16     <tr>
17         @foreach($d as $dd)
18             <td>{{ $dd }}</td>
19         @endforeach
20     </tr>
21     @endforeach
22     </tbody>
23 </table>
24
25 @if (!$noScript)
26     @include('datatable::javascript', array('class' => $class, 'options' => $options, 'ca\
27 llbacks' => $callbacks))
28 @endif

```

---

Pada bagian render() di app/views/authors/index.blade.php ubah menjadi :

app/views/authors/index.blade.php

---

```

1     ....
2     ->render('datatable.uikit') }}
3     ....

```

---

Buat pula file app/public/css/app.css dengan isi :

app/public/css/app.css

---

```

1     /* -- datatable -- */
2     div.dataTables_wrapper {
3         margin-top: 15px;
4     }
5     div.dataTables_filter label {
6         font-weight: normal;
7         float: right;
8     }
9     div.dataTables_filter input,
10    div.dataTables_length select {
11        height: 30px;
12        padding: 4px 6px;
13        font-size: 14px;
14        max-width: 100%;
15        border: 1px solid #dddddd;
16        background: #ffffff;
17        color: #444444;
18        -webkit-transition: all linear 0.2s;
19        transition: all linear 0.2s;
20        border-radius: 4px;
21    }

```

```
22 div.dataTables_filter input:focus,  
23 div.dataTables_length select:focus {  
24     border-color: #99baca;  
25     outline: 0;  
26     background: #f5fbfe;  
27     color: #444444;  
28 }  
29 /* -- end datatables --
```

---

Jangan lupa untuk me-*load* file ini pada app/views/layouts/master.blade.php pada bagian <head>:

app/views/layouts/master.blade.php

---

```
1 ....  
2 <head>  
3     <title>@yield('title') | Laravel Perpus</title>  
4     <link rel="stylesheet" href="{{ asset('packages/uikit/css/uikit.almost-flat.min.css') }}\br/>5 }>  
6     <link rel="stylesheet" href="{{ asset('css/app.css') }}" />  
7     <script src="{{ asset('components/jquery/jquery.min.js') }}"></script>  
8     <script src="{{ asset('packages/uikit/js/uikit.min.js') }}"></script>  
9     @yield('asset')  
10    </head>  
11 ....
```

---

Reload kembali halaman admin/authors, kini tampilannya berubah menjadi :

Datatable after

## Tambah Author

Pada tahap ini kita akan membuat halaman untuk menambah Penulis baru. Berbeda dengan pembahasan di Hari 2, untuk memudahkan pembuatan form yang sesuai dengan tampilan UIKIT, kita akan menggunakan [custom macro<sup>75</sup>](#). Dengan menggunakan custom macro, kita dapat membuat HTML dengan lebih cepat.

Macro yang dibuat akan kita simpan di `app/helpers/frontend.php`. Silahkan isi dengan :

<sup>75</sup><http://laravel.com/docs/html#custom-macros>

**app/helpers/frontend.php**

```
1 <?php
2 /*
3 /-----
4 | Register custom macros
5 |-----
6 |
7 | Register custom macros for blade view.
8 */
9
10 /**
11 * Macro untuk membuat divider
12 * @return string html
13 */
14 HTML::macro('divider', function() {
15     return "<hr class=\"uk-article-divider\">";
16 });
17
18 /**
19 * Macro for UIKIT label
20 * @return string html
21 */
22 Form::macro('labelUk', function($name, $placeholder) {
23     return Form::label($name, $placeholder, array('class' => 'uk-form-label'));
24 });
25
26 /**
27 * Macro for UIKIT text
28 * @return string html
29 */
30 Form::macro('textUk', function($name, $placeholder = null, $icon = null) {
31     $html = '';
32     if ($icon) {
33         $html .= "<div class=\"uk-form-icon\">
34             <i class=\"$icon\"></i>";
35     } else {
36         $html .= "<div class=\"uk-form-controls\">";
37     }
38
39     $html .= Form::text($name, null, array('placeholder'=>$placeholder));
40     $html .= '</div>';
41
42     return $html;
43 });
44
```

```

45 /**
46 * Macro untuk menampilkan tombol submit
47 * @return string html
48 */
49 Form::macro('submitUk', function($title) {
50     return "<input type=\"submit\" value=\"$title\" class=\"uk-button uk-button-primary\">" ;
51 });

```

---

Selanjutnya, load macro ini dengan cara menambahkan `require app_path() . '/helpers/frontend.php';` ke bagian akhir `app/start/global.php`.

Sip, sekarang kita siap membuat fungsi Tambah Penulis.

1. Siapkan Controller untuk menampilkan form. Ubah `create` pada `AuthorsController` menjadi :

app/controllers/AuthorsController.php

---

```

1 ....
2 public function create()
3 {
4     return View::make('authors.create')->withTitle('Tambah Penulis');
5 }
6 ....

```

---

2. Persiapkan View. Ubah `app/views/authors/create.blade.php` menjadi :

app/views/authors/create.blade.php

---

```

1 @extends('layouts.master')
2
3 @section('title')
4     {{ $title }}
5 @stop
6
7 @section('breadcrumb')
8     <li><a href="/">Dashboard</a></li>
9     <li><a href="{{ route('admin.authors.index') }}">Penulis</a></li>
10    <li class="uk-active">{{ $title }}</li>
11 @stop
12
13 @section('content')
14     {{ Form::open(array('url' => route('admin.authors.store'), 'method' => 'post', 'class' \ 
15     => 'uk-form uk-form-horizontal')) }}
16     @include('authors._form')
17     {{ Form::close() }}
18 @stop

```

---

Karena form akan digunakan untuk menambah/mengedit penulis, maka kita menggunakan subviews. Silahkan buat di `app/views/authors/_form.blade.php` dengan isi :

app/views/authors/\_form.blade.php

```

1  <div class="uk-form-row">
2      {{ Form::labelUk('name', 'Nama Penulis') }}
3      {{ Form::textUk('name', 'Nama Penulis', 'uk-icon-user') }}
4  </div>
5  {{ HTML::divider() }}
6  <div class="uk-form-row">
7      {{ Form::submitUk('Simpan') }}
8  </div>
```

Kita juga perlu menambahkan tombol Tambah di halaman admin.authors.index. Tombol ini akan membuat link ke route create dari index yang sedang kita akses. Mari kita buat Macro untuk tombol ini di app/helpers/frontend.php :

app/helpers/frontend.php

```

1  ....
2  /**
3  * Macro untuk membuat tombol tambah
4  * @param $path string url to route (if any)
5  * @return string generate anchor
6  */
7  HTML::macro('buttonAdd', function($path = null) {
8      if ($path) {
9          $url = $path;
10     } else {
11         $url = explode('.', Route::currentRouteName());
12         array_pop($url);
13         array_push($url, 'create');
14         $url = implode('.', $url);
15         $url = route($url);
16     }
17     return '<a class="uk-button uk-button-primary" href="'. $url .'">Tambah</a>';
18 });
19 ....
```

Logika dari Macro ini jika tidak ada url yang diinputkan adalah merubah admin.authors.index menjadi admin.author.create.

Karena, tombol seperti ini akan digunakan di banyak view, mari tambahkan section di layouts master. Ubah isian @section('title') menjadi :

**app/views/layouts/master.blade.php**

```

1  ....
2  <h1 class="uk-heading-large">
3      @yield('title')
4      @yield('title-button')
5  </h1>
6  ....

```

Selanjutnya, tambahkan tombol Tambah di app/views/authors/index.blade.php. Isikan syntax ini setelah section title :

**app/views/authors/index.blade.php**

```

1  ....
2  @section('title-button')
3      {{ HTML::buttonAdd() }}
4  @stop
5  ....

```

3. Persiapkan model untuk [validasi](#)<sup>76</sup>. Laravel memudahkan kita dalam melakukan validasi, pada model Author, kita akan membuat validasi agar field name diisi dan bernilai unique (tidak ada yang sama). Ubah app/models/Authors.php menjadi

**app/models/Author.php**

```

1  <?php
2
3  class Author extends \Eloquent {
4
5      // Add your validation rules here
6      public static $rules = [
7          'name' => 'required|unique:authors'
8      ];
9
10     // Don't forget to fill this array
11     protected $fillable = ['name'];
12
13 }

```

Bagian `protected $fillable` menandakan field mana saja yang diizinkan untuk melakukan [mass assignment](#)<sup>77</sup>. Mass assignment memungkinkan kita mengisi atribut sebuah model dengan array. Hal ini memudahkan kita dalam membuat form, tapi bisa bahaya jika semua field boleh diisi. Dengan menggunakan `fillable` kita memberitahu Laravel, bahwa hanya `name` yang boleh diisi melalui array. Bisa juga menggunakan `guarded` untuk kebalikan dari `fillable`.

<sup>76</sup><http://laravel.com/docs/validation>

<sup>77</sup><http://laravel.com/docs/eloquent#mass-assignment>

4. Selanjutnya, persiapkan Controller untuk menerima input. Form ini memiliki method ke route `admin.authors.store` jika dicek melalui `php artisan routes` maka akan diketahui bahwa route tersebut mengarah ke fungsi `store` di `AuthorsController`. Ubah fungsi ini menjadi :

app/controllers/AuthorsController.php

---

```

1   ....
2   public function store()
3   {
4       $validator = Validator::make($data = Input::all(), Author::$rules);
5
6       if ($validator->fails())
7       {
8           return Redirect::back()->withErrors($validator)->withInput();
9       }
10
11       $author = Author::create($data);
12
13       return Redirect::route('admin.authors.index')->with("successMessage", "Berhasil menyimpan " . $author->name);
14   }
15
16   ....

```

---

Berikut penjelasan dari fungsi ini:

- Pada baris ke-4, kita membuat validasi berdasarkan data dari `$_POST` variable dan rules dari model `Author`.
- Pada baris ke-8, baris ini dieksekusi jika validatornya gagal. Kita me-redirect user ke halaman sebelumnya dan menyimpan variable `$errors` berisi object `MessageBag`. Kita akan membuat view terpisah untuk menampilkan pesan error ini.
- Pada baris ke-11, kita menyimpan data author ke database.
- Pada baris ke-13, kita me-*redirect* user ke halaman index author dan memberikan pesan sukses.

Untuk menampilkan pesan validasi kita perlu membuat view nya. Buatlah file `app/views/layouts/partials/validation.blade.php` dengan isi :

app/views/layouts/partials/validation.blade.php

---

```

1   @if ($errors->count() > 0)
2       <div class="uk-alert uk-alert-danger" data-uk-alert>
3           <a href="" class="uk-alert-close uk-close"></a>
4           @foreach ($errors->all('<p>:message</p>') as $error)
5               {{ $error }}
6           @endforeach
7       </div>
8   @endif

```

---

Tampilan error ini hanya akan muncul terdapat error. Selanjutnya, tambahkan `@include('layouts.partials.validation')` di `app/views/layouts/master.blade.php` sebelum `@yield('content')`.

5. Sip. Selanjutnya tinggal dicek, coba buat penulis baru dengan field penulis dikosongkan, penulis baru tapi namanya sudah dipakai, dan penulis yang namanya belum digunakan.

The screenshot shows a web browser window titled "Tambah Penulis | Laravel". The URL in the address bar is "larapus.site/admin/authors/create". The page header includes "LaraPus" on the left and "Admin Larapus | Logout" on the right. The main content area has a breadcrumb navigation: "Dashboard / Penulis / Tambah Penulis". Below the breadcrumb is the title "Tambah Penulis". A red error message box contains the text "The name field is required." To the right of the message box is a small red "X" icon. Below the message box is a form field labeled "Nama Penulis" with a placeholder "Nama Penulis". To the left of the form field is the label "Nama Penulis". At the bottom left of the form is a blue button labeled "Simpan". At the bottom center of the page, there is a success message: "Gagal karena field name kosong".

The screenshot shows a web browser window titled "Tambah Penulis | Laravel P..." with the URL "larapus.site/admin/authors/create". The page is part of a system named "LaraPus" and is being viewed by an administrator ("Admin Larapus"). The breadcrumb navigation indicates the user is on the "Tambah Penulis" (Add Author) page under the "Penulis" (Authors) section of the dashboard.

A red error message box contains the text "The name has already been taken." An "X" icon is present in the top right corner of the message box.

The form field for "Nama Penulis" (Author Name) is labeled "Nama Penulis" and contains the placeholder text "Nama Penulis".

A blue "Simpan" (Save) button is located below the form.

At the bottom of the page, a message in bold text states "Gagal karena field name harus unique".

Berhasil menyimpan Rahmat Awaludin

Dashboard / Penulis

# Penulis

[Tambah](#)

Nama	
Mohammad Fauzil Adhim	<a href="#">edit</a>   <a href="#">hapus</a>
Salim A. Fillah	<a href="#">edit</a>   <a href="#">hapus</a>
Aam Amiruddin	<a href="#">edit</a>   <a href="#">hapus</a>
Surya	<a href="#">edit</a>   <a href="#">hapus</a>
Firman	<a href="#">edit</a>   <a href="#">hapus</a>
Rahmat Awaludin	<a href="#">edit</a>   <a href="#">hapus</a>

Showing 1 to 6 of 6 entries

First Previous **1** Next Last

Akhirnya berhasil juga.. :)

## Ubah Author

Pada saat meng-*update* Author kita tidak bisa menggunakan \$rules yang sama seperti pada saat membuat author. Karena, bisa saja terjadi user tidak mengubah nama dari Author kemudian menekan tombol Simpan. Laravel menyediakan fitur agar kita melewati validasi unique jika bertemu dengan id tertentu<sup>78</sup>. Namun, kita harus merubah \$rules pada model. Cara yang akan saya jelaskan merupakan metode yang saya gunakan agar tetap menyimpan validasi di model.

1. Buat model baru di app/models/BaseModel.php isi dengan :

<sup>78</sup><http://laravel.com/docs/validation#rule-unique>

app/models/BaseModel.php

```

1 <?php
2
3 class BaseModel extends \Eloquent {
4
5     /**
6      * Model rules
7      * @var array
8      */
9     public static $rules = [];
10
11    /**
12     * Get rules for update by replacing :id with $model->id
13     * @return array
14     */
15    public function updateRules()
16    {
17        $rules = static::$rules;
18        foreach ($rules as $field => $rule) {
19            // replace :id with $model->id
20            $rules[$field] = str_replace(':id', $this->getKey(), $rule);
21        }
22        return $rules;
23    }
24 }
```

Pada Kelas ini, kita membuat fungsi `updateRules()` untuk mendapatkan `$rules` tapi dengan placeholder `:id` yang sudah diganti dengan id dari model yang sedang di update. Kelas ini yang akan di-extends oleh kelas model yang lain.

2. Extends kelas `BaseModel` pada model `Author` dan ubah `$rules` sehingga menjadi :

app/models/BaseModel.php

```

1 <?php
2
3 class Author extends BaseModel {
4
5     // Add your validation rules here
6     public static $rules = [
7         'name' => 'required|unique:authors,name,:id'
8     ];
9
10    // Don't forget to fill this array
11    protected $fillable = ['name'];
12
13 }
```

3. Kini siapkan controller. Ubah fungsi edit pada app/controllers/AuthorsController menjadi :

app/controllers/AuthorsController.php

---

```

1   ....
2   public function edit($id)
3   {
4       $author = Author::find($id);
5       return View::make('authors.edit', ['author'=>$author])->withTitle("Ubah $author->name\
6   ");
7   }
8   ....

```

---

4. Siapkan view. Ubah app/views/authors/edit.blade.php menjadi :

app/views/authors/edit.blade.php

---

```

1   @extends('layouts.master')
2
3   @section('title')
4       {{ $title }}
5   @stop
6
7   @section('breadcrumb')
8       <li><a href="/">Dashboard</a></li>
9       <li><a href="{{ route('admin.authors.index') }}">Penulis</a></li>
10      <li class="uk-active">{{ $title }}</li>
11  @stop
12
13 @section('content')
14     {{ Form::model($author, array('url' => route('admin.authors.update', ['authors'=>$aut\
15 hor->id]), 'method' => 'put', 'class'=>'uk-form uk-form-horizontal')) }}
16     @include('authors._form')
17     {{ Form::close() }}
18 @stop

```

---

Pada form edit ini, kita menggunakan fasilitas [Form Model Binding<sup>79</sup>](#) dari Laravel. Dengan menggunakan fitur ini, kita tidak perlu mempopulasikan form dengan data author yang sedang diedit, karena Laravel yang akan melakukannya secara otomatis. Mantap.

5. Selanjutnya, persiapkan fungsi update pada AuthorsController untuk menerima data penulis yang telah diupdate. Ubah menjadi :

---

<sup>79</sup><http://laravel.com/docs/html#form-model-binding>

---

app/controllers/AuthorsController.php

---

```

1   ....
2   public function update($id)
3   {
4       $author = Author::findOrFail($id);
5
6       $validator = Validator::make($data = Input::all(), $author->updateRules());
7
8       if ($validator->fails())
9       {
10           return Redirect::back()->withErrors($validator)->withInput();
11       }
12
13       $author->update($data);
14
15       return Redirect::route('admin.authors.index')->with("successMessage", "Berhasil menyimpan " . $author->name);
16   }
17
18   ....

```

---

6. Kini tinggal menampilkan link edit dari halaman index. Caranya, ubah fungsi index di AuthorsController menjadi :

---

app/controllers/AuthorsController.php

---

```

1   ....
2   public function index()
3   {
4       if(Datatable::shouldHandle())
5       {
6           return Datatable::collection(Author::all(array('id', 'name')))
7               ->showColumns('id', 'name')
8               ->addColumn('', function ($model) {
9                   return '<a href="'.route('admin.authors.edit', ['authors'=>$model->id]).'>edit</a> | hapus';
10                })
11            ->searchColumns('name')
12            ->orderColumns('name')
13            ->make();
14        }
15
16        return View::make('authors.index')->withTitle('Penulis');
17    }
18
19

```

---

Pada baris ke-9, kita menggunakan `url helper80` untuk menampilkan url ke halaman edit Author. Fungsi ini akan merubah `{authors}` pada `admin/authors/{authors}/edit` (cek hasil perintah `php artisan routes`) menjadi `id` dari hasil `$model->id`. Selanjutnya, `id` inilah yang akan digunakan sebagai parameter pada fungsi `edit` di `AuthorsController`.

7. Sip. Selanjutnya tinggal di test deh. Coba sekarang edit salah satu penulis, ubah namanya, harusnya berhasil. Kemudian, coba edit lagi, kosongkan namanya, harusnya gagal. Terakhir, coba edit lagi, namanya jangan dirubah, harusnya berhasil. Pastikan semua fiturnya jalan.

## Hapus Author

Penghapusan Author dilakukan dengan cara User menekan tombol `delete` di halaman index Author. Pada langkah ini kita akan membuat tombol ini dan memberikan user notifikasi ketika Author telah dihapus.

Resource controller yang dihasilkan oleh Laravel, meminta kita untuk mengirimkan `DELETE` request. Karena tidak mungkin kita membuat link/button dengan `DELETE` request, maka kita akan mengakalinya dengan membuat form dengan method `DELETE` dan menggunakan tombol `submit` sebagai tombol `delete`.

1. Tambahkan tombol delete dengan mengubah handle untuk datatable pada fungsi `index` di `AuthorsController` menjadi :

---

<sup>80</sup><http://laravel.com/docs/helpers#urls>

**app/controllers/AuthorsController.php**

```

1   ....
2   public function index()
3   {
4       if(Datatable::shouldHandle())
5       {
6           return Datatable::collection(Author::all(array('id', 'name')))
7               ->showColumns('id', 'name')
8               ->addColumn('', function ($model) {
9                   $html = '<a href="'.route('admin.authors.edit', ['authors'=>$model->id]).'";
10                  '' class="uk-button uk-button-small uk-button-link">edit</a> ';
11                  $html .= Form::open(array('url' => route('admin.authors.destroy', ['autho\
12 rs'=>$model->id]), 'method'=>'delete', 'class'=>'uk-display-inline'));
13                  $html .= Form::submit('delete', array('class' => 'uk-button uk-button-sma\
14 ll'));
15                  $html .= Form::close();
16                  return $html;
17             })
18             ->searchColumns('name')
19             ->orderColumns('name')
20             ->make();
21         }
22         return View::make('authors.index')->withTitle('Penulis');
23     }
24     ....

```

- Pada baris ke-9, kita merubah style dari link edit agar sesuai dengan tampilan tombol delete.
- Pada baris ke-10 sampai 12, kita membuat form dengan style inline dan method ‘delete’. Jika sudah jadi form, syntax di baris ini akan berubah menjadi :

**hasil**

```

1   <form method="POST" action="http://larapus.site/admin/authors/2" accept-charset="UTF-8" \
2   class="uk-display-inline">
3       <input name="_method" type="hidden" value="DELETE">
4       <input name="_token" type="hidden" value="aGN2nqlrI7Dp0nKGAJHpCGnDe1F1gaxMuXI90uor">
5       <input class="uk-button uk-button-small" type="submit" value="delete">
6   </form>

```

2. Untuk menerima request ini, persiapkan fungsi destroy di AuthorsController menjadi :

## app/controllers/AuthorsController.php

```

1  ....
2  public function destroy($id)
3  {
4      Author::destroy($id);
5
6      return Redirect::route('admin.authors.index')->with('successMessage', 'Penulis berhasil dihapus.');
7  }
8
9  ....

```

Pada baris ini, kita menggunakan fungsi `destroy` pada model `Author` untuk menghapus record dengan `id` yang dikirimkan. Kemudian mengarahkan user ke halaman `admin.authors.index` dengan flash message *Penulis berhasil dihapus.*

3. Silahkan dicek fitur penghapusan Author yang baru dibuat.

The screenshot shows a web browser window titled "Penulis | Laravel Perpus". The address bar displays "larapus.site/admin/authors". The page header includes "LaraPus" on the left and "Admin Larapus | Logout" on the right. A green success message box contains the text "Penulis berhasil dihapus." with a close button. Below the message, the breadcrumb navigation shows "Dashboard / Penulis". The main content area has a title "Penulis" with a "Tambah" button. It features a data table with two entries: "Salim A. Filla" and "Aam Amiruddin", each with "edit" and "delete" buttons. At the bottom, it shows "Showing 1 to 2 of 2 entries" and navigation links "First", "Previous", "1", "Next", and "Last". A footer message at the bottom center says "Berhasil menghapus author".

## CRUD Buku

Table buku dalam aplikasi ini memiliki field `title`, `author_id`, `amount` (untuk jumlah total), `created_at` dan `updated_at`. Pada proses pembuatan CRUD buku ini, kita juga akan belajar bagaimana menggunakan relasi<sup>81</sup> many-to-one dari buku ke penulis. Dalam konteks aplikasi ini sebuah buku hanya ditulis oleh satu penulis, untuk pengembangannya pembaca dapat membuat sebuah buku ditulis oleh beberapa penulis. Karena pada tahap ini seorang penulis terikat dengan buku, maka sebelum menghapus penulis kita juga akan mengecek relasi ke buku.

Untuk membuat dropdown lebih menarik, kita juga akan menggunakan select2<sup>82</sup>. Ikuti langkah berikut untuk menambahkan select2 :

1. Tambahkan "ivaynberg/select2": "3.4.8" di `composer.json`
2. Download select2 dengan perintah `composer update`
3. Publish asset dengan perintah `php artisan asset:publish select2 --path="vendor/ivaynberg/select2"`

## Persiapan

1. Generate resource untuk buku dengan perintah `php artisan generate:scaffold book --fields="title:string, author_id:integer:unsigned, amount:integer:unsigned"`.
2. Tambahkan resources route untuk buku di `app/routes.php` :

`app/routes.php`

---

```

1  ....
2  Route::group(array('prefix' => 'admin', 'before' => 'admin'), function()
3  {
4      Route::resource('authors', 'AuthorsController');
5      Route::resource('books', 'BooksController');
6  });
7  ....

```

---

3. Lakukan database seeder untuk sample data. Ubah file `app/database/seeds/BooksTableSeeder.php` menjadi :

---

<sup>81</sup><http://laravel.com/docs/eloquent#relationships>

<sup>82</sup><http://ivaynberg.github.io/select2/>

---

app/database/seeds/BooksTableSeeder.php

---

```

1 <?php
2
3 class BooksTableSeeder extends Seeder {
4
5     public function run()
6     {
7         // kosongkan database
8         DB::table('books')->delete();
9
10        // buat array untuk diinput
11        $books = [
12            ['id'=>1, 'title'=>'Kupinang Engkau dengan Hamdalah', 'author_id'=>1, 'amount'=>3, 'created_at'=>'2014-05-16 11:15:22', 'updated_at'=>'2014-05-16 11:15:22'],
13            ['id'=>2, 'title'=>'Jalan Cinta Para Pejuang', 'author_id'=>2, 'amount'=>2, 'created_at'=>'2014-05-16 11:15:22', 'updated_at'=>'2014-05-16 11:15:22'],
14            ['id'=>3, 'title'=>'Membingkai Surga dalam Rumah Tangga', 'author_id'=>3, 'amount'=>4, 'created_at'=>'2014-05-16 11:15:22', 'updated_at'=>'2014-05-16 11:15:22'],
15            ['id'=>4, 'title'=>'Cinta & Seks Rumah Tangga Muslim', 'author_id'=>3, 'amount'=>3, 'created_at'=>'2014-05-16 11:15:22', 'updated_at'=>'2014-05-16 11:15:22']
16        ];
17
18        // insert data ke database
19        DB::table('books')->insert($books);
20    }
21
22 }
23
24 }
```

---

Tambahkan BooksTableSeeder ke DatabaseSeeder :

---

app/database/seeds/DatabaseSeeder.php

---

```

1 ....
2 public function run()
3 {
4     Eloquent::unguard();
5     $this->call('SentrySeeder');
6     $this->call('AuthorsTableSeeder');
7     $this->call('BooksTableSeeder');
8 }
9 ....
```

---

Lakukan migration dengan perintah `php artisan db:seed`.

```
2. rahmatawaludin@MorphaWorks: ~/sites/larapus (zsh)
~/sites/larapus ➜ master • ➜ php artisan db:seed
Seeded: SentrySeeder
Seeded: AuthorsTableSeeder
Seeded: BooksTableSeeder
~/sites/larapus ➜ master •
```

Melakukan migrations untuk table books

Pada fitur CRUD buku ini juga akan menggunakan datatable. Tentunya, kita juga akan meng-include kembali javascript dan css untuk datatable. Supaya tetap mengikuti kaidah **DRY (Don't Repeat Yourself)**<sup>83</sup>, mari kita pindahkan include untuk datatable ini ke file berbeda.

1. Buat file app/views/layouts/partials/datatable.blade.php dengan isi :

app/views/layouts/partials/datatable.blade.php

---

```
1 <link rel="stylesheet" href="{{ asset('packages/datatables/css/jquery.dataTables.css') }}"\>
2 />
3 <script src="{{ asset('packages/datatables/js/jquery.dataTables.js') }}"></script>
```

---

2. Kini ubah isian @asset pada app/views/authors/index.blade.php menjadi :

app/views/authors/index.blade.php

---

```
1 ....
2 @section('asset')
3     @include('layouts.partials.datatable')
4 @stop
5 ....
```

---

3. Cek kembali listing author, pastikan masih berjalan dengan sempurna.

## Listing buku

Pada tampilan listing buku, user dapat melihat judul, nama penulis dan jumlah buku. Sementara, fitur stok yang belum dipinjam akan kita kerjakan di bagian peminjaman buku.

Untuk membuat fitur listing buku, silahkan ikuti langkah berikut ini:

1. **Relasi antar model.** Relasi author ke books adalah one-to-many, kita harus memberitahu eloquent dengan menambah relasi ke books dari model Author :

---

<sup>83</sup>[http://en.wikipedia.org/wiki/Don't\\_repeat\\_yourself](http://en.wikipedia.org/wiki/Don't_repeat_yourself)

---

app/models/Author.php

---

```

1   ....
2   public function books()
3   {
4       return $this->hasMany('Book');
5   }
6   ....

```

---

Kita juga perlu menambahkan kebalikan relasi one-to-many di model Book :

---

app/models/Book.php

---

```

1   ....
2   public function author()
3   {
4       return $this->belongsTo('Author');
5   }
6   ....

```

---

Dengan menggunakan relasi seperti ini, kita dapat dengan mudah mengambil nilai dari relasi (dibanding dengan query sql manual). Contohnya :

- Mengambil atribut nama dari penulis sebuah buku : \$book->author->name
- Mengambil semua buku yang berhubungan dengan penulis : \$author->books

Untuk lebih lengkapnya, silahkan baca di dokumentasi [cara mengakses relasi<sup>84</sup>](#).

2. **Controller.** Persiapkan fungsi index pada BooksController menjadi :

---

app/controllers/BooksController.php

---

```

1   ....
2   public function index()
3   {
4       if(Datatable::shouldHandle())
5       {
6           // eager load author untuk menghemat query sql
7           return Datatable::collection(Book::with('author')->get())
8               ->showColumns('id', 'title', 'amount')
9               // menggunakan closure untuk menampilkan nama penulis dari relasi
10              ->addColumn('author', function ($model) {
11                  return $model->author->name;
12              })
13              ->addColumn('', function ($model) {
14                  return 'edit | delete';
15              })
16              ->searchColumns('title', 'amount', 'author')
17              ->orderColumns('title', 'amount', 'author')

```

---

<sup>84</sup><http://laravel.com/docs/eloquent#querying-relations>

```

18         ->make();
19     }
20     return View::make('books.index')->withTitle('Buku');
21 }
22 ....

```

---

Isi fungsi index ini hampir sama dengan yang terdapat di fungsi di AuthorsController. Perbedaannya terletak pada **eager load**<sup>85</sup> relasi ke author untuk menghemat query SQL (baris ke-7) dan penggunaan closure untuk memanggil nama penulis dari relasi (baris ke-10 sampai 12).

3. **View.** Terakhir siapkan view di app/views/books/index.blade.php dengan isi :

app/views/books/index.blade.php

---

```

1  @extends('layouts.master')
2
3  @section('asset')
4      @include('layouts.partials.datatable')
5  @stop
6
7  @section('title')
8      {{ $title }}
9  @stop
10
11 @section('title-button')
12     {{ HTML::buttonAdd() }}
13 @stop
14
15 @section('breadcrumb')
16     <li><a href="/">Dashboard</a></li>
17     <li class="uk-active">{{ $title }}</li>
18 @stop
19
20 @section('content')
21
22     {{ Datatable::table()
23         ->addColumn('id', 'title', 'author', 'amount', '')
24         ->setOptions('aoColumnDefs', array(
25             array(
26                 'bVisible' => false,
27                 'aTargets' => [0]),
28             array(
29                 'sTitle' => 'Judul',
30                 'aTargets' => [1]),
31             array(
32                 'sTitle' => 'Jumlah',

```

<sup>85</sup><http://laravel.com/docs/eloquent#eager-loading>

```
33         'aTargets' => [2]),
34     array(
35         'sTitle' => 'Penulis',
36         'aTargets' => [3]),
37     array(
38         'bSortable' => false,
39         'aTargets' => [4])
40     ))
41     ->setOptions('bProcessing', true)
42     ->setUrl(route('admin.books.index')) // this is the route where data will be retrieved
43     ved
44     ->render('datatable.uikit') }}
```

---

Pada file ini kita me-*render* datatable, isinya hampir sama dengan cara kita me-*render* datatable di halaman author. Jika sudah dibuat coba dicek fitur listing bukunya. Tombol untuk menambah buku baru juga sudah kita tulis disini.

LaraPus

Admin Larapups | Logout

Dashboard / Buku

# Buku

[Tambah](#)

Judul	Jumlah	Penulis	
Kupinang Engkau dengan Hamdalah	3	Mohammad Fauzil Adhim	<a href="#">edit</a>   <a href="#">delete</a>
Jalan Cinta Para Pejuang	2	Salim A. Fillah	<a href="#">edit</a>   <a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	Aam Amiruddin	<a href="#">edit</a>   <a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	Aam Amiruddin	<a href="#">edit</a>   <a href="#">delete</a>

Show 10 entries Search:

Showing 1 to 4 of 4 entries First Previous **1** Next Last

Berhasil membuat listing buku

## Tambah Buku

1. **Controller.** Ubah fungsi create di BooksController menjadi :

app/controllers/BooksController.php

```

1  ....
2  public function create()
3  {
4      return View::make('books.create')->withTitle('Tambah Buku');
5  }
6  ....

```

2. **View.** View yang kita buat akan memiliki dropdown dari select2 dan memisahkan komponen form pada view yang berbeda. Karena komponen select2 akan kita gunakan pada form yang kita akan membuatnya menjadi view terpisah di app/views/layouts/partials/select2.blade.php :

---

app/views/layouts/partials/select2.blade.php

---

```

1   <link rel="stylesheet" href="{{ asset('packages/select2/select2.css')}}" />
2   <script src="{{ asset('packages/select2/select2.min.js')}}"></script>
3   <script src="{{ asset('packages/select2/select2_locale_id.js')}}"></script>
```

---

Agar tampilan select2 rapi, tambahkan baris ini di app/public/css/app.css :

---

app/public/css/app.css

---

```

1   ....
2   .select2-container .select2-choice {
3     width: 183px;
4   }
5   ....
```

---

Form untuk membuat buku juga akan dibuat secara partial. Silahkan buat dulu app/views/books/\_form.blade.php :

---

app/views/books/\_form.blade.php

---

```

1   <div class="uk-form-row">
2     {{ Form::labelUk('title', 'Judul') }}
3     {{ Form::textUk('title', 'Judul Buku', 'uk-icon-book') }}
4   </div>
5   <div class="uk-form-row">
6     {{ Form::labelUk('author_id', 'Penulis') }}
7     {{ Form::select('author_id', array('=>'')+Author::lists('name','id'), null, array(
8       'id'=>'author_id',
9       'placeholder' => "Pilih Penulis")) }}
10  </div>
11  <div class="uk-form-row">
12    {{ Form::labelUk('amount', 'Jumlah') }}
13    {{ Form::textUk('amount', 'Jumlah Buku', 'uk-icon-unsorted') }}
14  </div>
15  {{ HTML::divider() }}
16  <div class="uk-form-row">
17    {{ Form::submitUk('Simpan') }}
18  </div>
19  <script>
20    $(document).ready(function() { $("#author_id").select2(); });
21  </script>
```

---

Pada form ini kita membuat field teks untuk judul buku (title), dropdown list untuk author\_id dan teks untuk amount. Pembuatan dropdown list untuk author\_id menggunakan data dari model Author yang kita ambil dalam bentuk list dengan perintah Author::lists('name', 'id'), kita juga menggunakan array kosong array('=>') agar placeholder dropdown list select2 berjalan.

Terakhir, buat view di app/views/books/create.blade.php :

---

app/views/books/create.blade.php

---

```

1  @extends('layouts.master')
2
3  @section('title')
4      {{ $title }}
5  @stop
6
7  @section('asset')
8      @include('layouts.partials.select2')
9  @stop
10
11 @section('breadcrumb')
12     <li><a href="/">Dashboard</a></li>
13     <li><a href="{{ route('admin.books.index') }}">Buku</a></li>
14     <li class="uk-active">{{ $title }}</li>
15 @stop
16
17 @section('content')
18     {{ Form::open(array('url' => route('admin.books.store'), 'method' => 'post', 'class'=<br>'uk-form uk-form-horizontal')) }}
19     @include('books._form')
20     {{ Form::close() }}
21 @stop

```

---

Form ini akan menggunakan method POST dan mengirim data ke route admin.books.store.

3. **Model.** Kita perlu melakukan validasi data, diantaranya title harus diisi dan tidak boleh sama, author\_id harus diisi dan harus ada di table authors, dan amount jika diisi harus berupa angka. Wah, nampaknya banyak ya. Tapi, di Laravel membuat validasi seperti itu gampang banget. Mari kita edit model Book :

---

app/models/Book.php

---

```

1  <?php
2  class Book extends BaseModel {
3
4      // Add your validation rules here
5      public static $rules = [
6          'title' => 'required|unique:books,title,:id',
7          'author_id' => 'required|exists:authors,id',
8          'amount' => 'numeric'
9      ];
10
11     // Don't forget to fill this array
12     protected $fillable = ['title', 'author_id', 'amount'];
13
14     ....

```

```
15  
16 }
```

---

Pada model Book, kita meng-*extends* `BaseModel` yang telah kita buat pada langkah sebelumnya agar kita bisa melakukan validasi untuk update model. Validasi yang dibutuhkan diletakkan pada atribut `$rules`. Kita juga menambahkan field yang bisa diisi dengan *mass assignment* pada atribut `$fillable`.

4. **Controller.** Tahap terakhir, adalah merubah fungsi `store` pada `BooksController` menjadi :

app/controllers/BooksController.php

---

```
1 ....  
2 public function store()  
3 {  
4     // validasi  
5     $validator = Validator::make($data = Input::all(), Book::$rules);  
6  
7     if ($validator->fails())  
8     {  
9         // kembali ke halaman sebelumnya  
10        return Redirect::back()->withErrors($validator)->withInput();  
11    }  
12  
13    // insert ke database  
14    $book = Book::create($data);  
15  
16    // kembali ke index  
17    return Redirect::route('admin.books.index')->with("successMessage", "Berhasil menyimpan  
18    $book->title");  
19 }  
20 ....
```

---

Syntax ini hampir sama dengan syntax `store` di `AuthorsController`. Jika sudah selesai, silahkan coba membuat buku baru. Jangan lupa dicoba juga validasi yang sudah dibuat.

The screenshot shows a web browser window titled "Tambah Buku | Laravel Per". The URL in the address bar is "larapus.site/admin/books/create". The page header includes "LaraPus" on the left and "Admin Larapus | Logout" on the right. Below the header, the breadcrumb navigation shows "Dashboard / Buku / Tambah Buku". The main title is "Tambah Buku". The form has three fields: "Judul" (Title) with a placeholder "Judul Buku", "Penulis" (Author) with a dropdown menu placeholder "Pilih Penulis", and "Jumlah" (Quantity) with a dropdown menu placeholder "Jumlah Buku". A blue "Simpan" (Save) button is located at the bottom left of the form area.

Membuat buku

Judul	Jumlah	Penulis	
Kupinang Engkau dengan Hamdalah	3	Mohammad Fauzil Adhim	<a href="#">edit</a>   <a href="#">delete</a>
Jalan Cinta Para Pejuang	2	Salim A. Fillah	<a href="#">edit</a>   <a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	Aam Amiruddin	<a href="#">edit</a>   <a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	Aam Amiruddin	<a href="#">edit</a>   <a href="#">delete</a>
Indahnya Pacaran Setelah Nikah	4	Salim A. Fillah	<a href="#">edit</a>   <a href="#">delete</a>

Showing 1 to 5 of 5 entries

First Previous **1** Next Last

Berhasil membuat buku

## Ubah Buku

Ikuti langkah berikut:

1. **Controller.** Tambahkan link pada datatable dengan mengubah isian fungsi `index` pada bagian handle Datatable di `BooksController` menjadi :

app/controllers/BooksController.php

```

1  ....
2  ->addColumn('', function ($model) {
3      return '<a href="'.$route('admin.books.edit', [ 'books'=>$model->id]).'">edit</a> | del\
4      ete';
5  })
6  ....

```

Kita juga perlu mengubah fungsi `edit` pada `BooksController` menjadi :

**app/controllers/BooksController.php**

```

1   ....
2   public function edit($id)
3   {
4       $author = Author::find($id);
5
6       return View::make('authors.edit', ['author'=>$author])->withTitle("Ubah $author->name\
7   ");
8   }
9   ....

```

---

2. **View.** Siapkan view di app/views/books/edit.blade.php dengan isi :

**app/views/books/edit.blade.php**

```

1   @extends('layouts.master')
2
3   @section('title')
4       {{ $title }}
5   @stop
6
7   @section('asset')
8       @include('layouts.partials.select2')
9   @stop
10
11  @section('breadcrumb')
12      <li><a href="/">Dashboard</a></li>
13      <li><a href="{{ route('admin.books.index') }}">Buku</a></li>
14      <li class="uk-active">{{ $title }}</li>
15  @stop
16
17  @section('content')
18      {{ Form::model($book, array('url' => route('admin.books.update', ['books'=>$book->id]) \
19 ), 'method' => 'put', 'class'=>'uk-form uk-form-horizontal')) }}
20      @include('books._form')
21      {{ Form::close() }}
22  @stop

```

---

Sama seperti form edit di Author, disini kita juga melakukan model binding untuk data form.

3. **Controller.** Terakhir, siapkan fungsi update di BooksController untuk menerima data dari form, validasi dan melakukan redirect :

---

**app/controllers/BooksController.php**

---

```
1   ....
2   public function update($id)
3   {
4       // cari buku
5       $book = Book::findOrFail($id);
6
7       // mengganti dengan nilai yang baru
8       $validator = Validator::make($data = Input::all(), $book->updateRules());
9
10      // validasi data
11      if ($validator->fails())
12      {
13          return Redirect::back()->withErrors($validator)->withInput();
14      }
15
16      // insert ke database
17      $book->update($data);
18
19      // redirect ke index
20      return Redirect::route('admin.books.index')->with("successMessage", "Berhasil menyimpan " .
21      $book->title. " ");
22  }
23  ....
```

---

Setelah form dibuat, silahkan cek ubah buku. Pastikan semua validasi data berjalan sesuai yang dikehendaki.

LaraPus Admin Larapush | Logout

Dashboard / Buku / Ubah Jalan Cinta Para Pejuang

# Ubah Jalan Cinta Para Pejuang

Judul	<input type="text" value="Jalan Cinta Para Pejuang"/>
Penulis	<input type="text" value="Salim A. Filiah"/>
Jumlah	<input type="text" value="2"/>

**Simpan**

### Merubah data buku

## Hapus Buku

Untuk menambah fitur hapus buku, ikuti langkah berikut.

1. Ubah fungsi index untuk menampilkan tombol delete pada handle datatable di BooksController :

app/controllers/BooksController.php

```

1   ....
2   ->addColumn('', function ($model) {
3       $html = '<a href="'.$route('admin.books.edit', ['books'=>$model->id]).'" class="uk-button\'
4       ton uk-button-small uk-button-link">edit</a> ';
5       $html .= Form::open(array('url' => route('admin.books.destroy', ['books'=>$model->id]\
6 ), 'method'=>'delete', 'class'=>'uk-display-inline'));
7       $html .= Form::submit('delete', array('class' => 'uk-button uk-button-small'));
8       $html .= Form::close();
9       return $html;
10  })
11  ....

```

Untuk membuat tombol delete kita menggunakan logika yang sama dengan tombol delete di AuthorsController, yaitu dengan membuat form dengan method DELETE.

2. Ubah fungsi destroy di BooksController menjadi :

**app/controllers/BooksController.php**

```

1   ....
2   public function destroy($id)
3   {
4       Book::destroy($id);
5
6       return Redirect::route('admin.books.index')->with('successMessage', 'Buku berhasil di\
7   hapus.');
8   }
9   ....

```

Fungsi ini akan menghapus buku dan melakukan redirect user ke halaman index. Jika sudah, silahkan dicoba menghapus buku.

The screenshot shows a web browser window titled "Buku | Laravel Perpus". The address bar contains "larapus.site/admin/books". The page header includes "LaraPus" and "Admin Larapus | Logout". A green success message box displays "Buku berhasil dihapus." Below the message, the breadcrumb navigation shows "Dashboard / Buku". The main content area is titled "Buku" with a "Tambah" button. It features a table with columns "Judul", "Jumlah", and "Penulis". Three books are listed:

Judul	Jumlah	Penulis	Actions
Kupinang Engkau dengan Hamdalah	3	Mohammad Fauzil Adhim	<a href="#">edit</a> <a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>

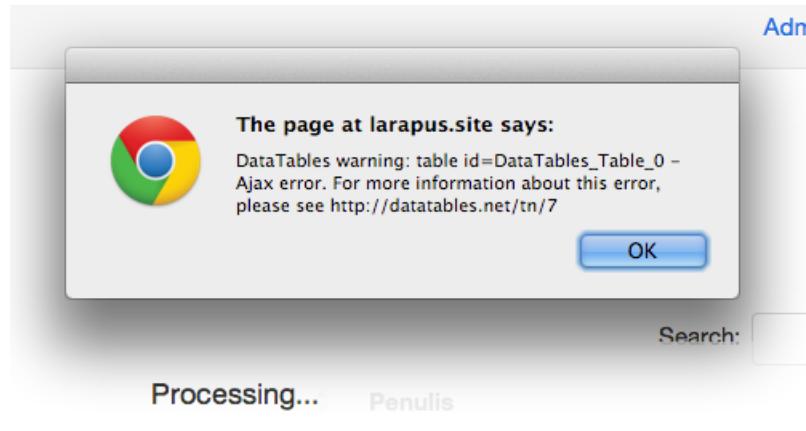
At the bottom, it says "Showing 1 to 3 of 3 entries" and has navigation buttons for First, Previous, Next, and Last. A modal dialog titled "Menghapus buku" is partially visible at the bottom.

Sebagai tugas, coba buat konfirmasi menggunakan javascript sebelum menghapus buku. Biar lebih menantang, gunakan [modal dari UIKIT<sup>86</sup>](#) untuk menampilkan pesan konfirmasinya. Saya serahkan tugas ini kepada Anda. Semangat! :)

<sup>86</sup><http://getuikit.com/docs/modal.html>

## Validasi Hapus Author

Sejauh ini semua fitur nampak sudah berjalan dengan sesuai yang diharapkan. Tapi, coba sekarang Anda hapus seorang penulis yang memiliki buku. Kemudian, cek kembali halaman buku, maka Anda akan mendapat error seperti berikut ini:



### Error karena penulis dihapus

Error ini terjadi karena record yang menjadi relasi dari buku dihapus. Ada dua langkah yang akan kita lakukan. Pertama, melakukan integrity constraint di database ketika delete menjadi restrict. Kedua, mengecek relasi ke buku sebelum menghapus author dengan menggunakan [model event](#)<sup>87</sup>.

- Karena pada migration sebelumnya kita belum menambahkan relasi di level database, maka kita harus membuat migration baru untuk menambah relasi. Jalankan perintah `php artisan generate:migration add_foreign_key_to_books`.

Setelah berhasil, ubahlah fungsi up dan down pada file migrasi yang dihasilkan :

`app/database/migrations/xxxx_xx_xx_xxxxxx_add_foreign_keys_to_books.php`

```

1  ....
2  public function up()
3  {
4      Schema::table('books', function(Blueprint $table) {
5          $table->foreign('author_id')->references('id')->on('authors')
6              ->onDelete('restrict')
7              ->onUpdate('cascade');
8      });
9  }
10 ....
11 ....
12 public function down()
13 {

```

<sup>87</sup><http://laravel.com/docs/eloquent#model-events>

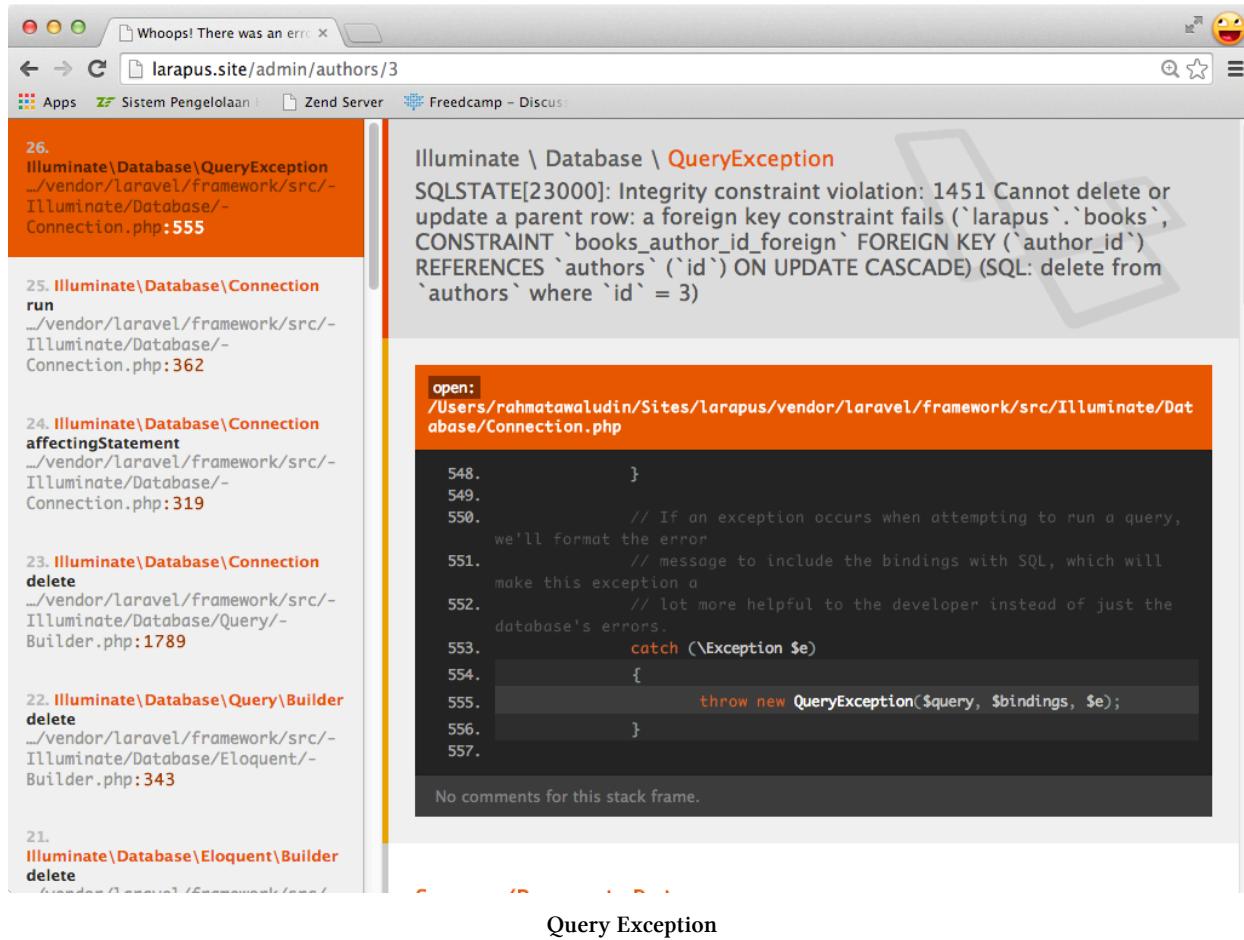
```

14     Schema::table('books', function(Blueprint $table) {
15         $table->dropForeign('books_author_id_foreign');
16     });
17 }
18 ...

```

Pada file migrasi kita menggunakan Schema Builder untuk membangun relasi antar table. Terlihat di baris ke-6 kita melarang records dihapus jika masih memiliki relasi (*restrict*).

Lakukan migrasi dengan `php artisan migrate`. Coba sekarang hapus lagi penulis yang memiliki buku, pasti tidak bisa. Hanya saja, sekarang muncul error :



Tips: untuk me-reset records di database ke data sample gunakan perintah `php artisan db:seed`

Exception ini terjadi karena kita mencoba menghapus record dengan constraint on delete restrict. Untuk menanganinya, kita perlu menggunakan menghentikan proses penghapusan records dan memberitahu user.

2. Model Event akan memberikan kita akses ke berbagai event CRUD sebuah model misalnya: creating, created, updating, updated, saving, saved, deleting, deleted, restoring, dan restored. Contohnya, jika kita ingin mengakses event sebelum object di update, kita bisa menggunakan event updating.

Contoh cara membatalkan pembuatan model dengan event creating :

---

contoh event creating

---

```

1 Book::creating(function($user)
2 {
3     return false;
4 });

```

---

Pada kasus yang kita miliki, kita ingin membatalkan proses delete author jika author masih memiliki buku dan memberitahu buku yang dimilikinya ke user. Kita akan meletakkannya pada fungsi boot di model Author. Fungsi ini merupakan fungsi bawaan Eloquent yang dipanggil ketika model dibuat.

---

app/models/Author.php

---

```

1 ....
2 public static function boot()
3 {
4     parent::boot();
5
6     self::deleting(function($author)
7     {
8         // mengecek apakah penulis masih punya buku
9         if ($author->books->count() > 0) {
10             $html = 'Penulis tidak bisa dihapus karena masih memiliki buku : ';
11             $html .= '<ul>';
12             foreach ($author->books as $book) {
13                 $html .= "<li>$book->title</li>";
14             }
15             $html .= '</ul>';
16             Session::flash('errorMessage', $html);
17             return false;
18         }
19     });
20 }
21 ....

```

---

Kita menggunakan flash message untuk mengirimkan pesan kesalahan berikut penjelasan mengenai buku yang masih terikat dengan author yang bersangkutan.

Terakhir, kita harus merubah fungsi destroy pada AuthorsController menjadi :

---

**app/controllers/AuthorsController.php**

---

```
1  ....
2  public function destroy($id)
3  {
4      // mengecek apakah author bisa dihapus
5      if (!Author::destroy($id))
6      {
7          return Redirect::back();
8      }
9
10     return Redirect::route('admin.authors.index')->with('successMessage', 'Penulis berhasil
11     dihapus.');
12 }
13 ....
```

---

Pada fungsi ini kita mengecek apakah author berhasil dihapus. Jika, tidak maka user akan di-redirect ke halaman sebelumnya. Pesan kesalahan akan diterima user hasil dari fungsi boot pada model Author.

Sekarang coba hapus kembali author yang memiliki buku. Pastikan pesan kesalahan yang dikehendaki muncul.

The screenshot shows a Laravel application interface. At the top, there's a navigation bar with links like 'Penulis | Laravel Perpus', 'larapus.site/admin/authors', 'Apps', 'Sistem Pengelolaan', 'Zend Server', and 'Freedcamp - DISCUS'. Below the navigation is a header with 'LaraPus' on the left and 'Admin Larapus | Logout' on the right. A red error message box contains the text: 'Penulis tidak bisa dihapus karena masih memiliki buku :'. Inside this list are two book titles: 'Membingkai Surga dalam Rumah Tangga' and 'Cinta & Seks Rumah Tangga Muslim'. Below the error message is a breadcrumb navigation: 'Dashboard / Penulis'. The main content area has a title 'Penulis' with a 'Tambah' button. It includes a search bar and a table with three entries:

Nama	edit	delete
Mohammad Fauzil Adhim	<a href="#">edit</a>	<a href="#">delete</a>
Salim A. Fillah	<a href="#">edit</a>	<a href="#">delete</a>
Aam Amiruddin	<a href="#">edit</a>	<a href="#">delete</a>

At the bottom of the table, it says 'Showing 1 to 3 of 3 entries' and has navigation buttons for 'First', 'Previous', '1', 'Next', and 'Last'.

Tidak bisa hapus penulis karena masih memiliki buku.

@todo guide for smartlink

## Ringkasan

Di Hari 4 ini, saya harap Anda telah memahami proses authentikasi dan CRUD untuk penulis dan buku dari halaman Admin, poin-poin yang telah kita bahas yaitu:

- Konfigurasi Sentry untuk authentikasi dan authorisasi aplikasi
- Konfigurasi Datatable untuk menampilkan data
- Penggunaan Flash Message
- Konfigurasi Select2 dari dengan data dari model
- Mass assignment pada model
- Validasi data pada model
- Penggunaan subview dalam blade
- Penggunaan macro dalam blade
- Penggunaan relasi pada Eloquent
- Penggunaan model event pada Eloquent

Pada hari 5, kita akan memulai implementasi beberapa fitur non-admin dan fitur stok buku. Spirit! :)

# Hari 5 : Develop Fitur Non-Admin

Dalam membangun aplikasi yang memiliki berbagai hak akses, sebenarnya cukup sulit untuk memisahkan pengembangan untuk fitur seorang admin maupun non-admin, karena keduanya pasti saling berkaitan. Contohnya, dalam aplikasi ini, jika kita sedang mengembangkan fitur peminjaman akan berkaitan dengan penghapusan buku oleh Admin. Jika buku sedang dipinjam, tentunya Admin tidak bisa menghapus buku tersebut.

Fitur lain yang saling berkaitan adalah fitur peminjaman dan fitur update jumlah total buku (`amount`). Fitur stok ini akan mempengaruhi alur ketika admin merubah jumlah total buku (`amount`), jika buku yang sedang dipinjam ada 3, maka admin tidak boleh mengubah jumlah buku (`amount`) kurang dari 3.

Untuk beberapa fitur yang saling berkaitan seperti itu, saya rasa harap Anda tidak akan protes kalau saya bahas disini.. :)

## Member

Member dalam Larapus adalah user yang telah login dan berada di dalam grup `regular`. Kita telah membuat sample user dengan grup `regular` pada bab 4. Pada tahapan ini kita akan membangun 2 fitur member yaitu peminjaman dan pengembalian buku.

### Peminjaman Buku

Dalam Larapus, alur untuk melakukan peminjaman buku diawali dengan user login, pilih menu buku, kemudian memilih buku yang akan dipinjam, dan User langsung tercatat meminjam buku tersebut. Dalam melakukan peminjaman ini, ada dua model yang akan saling berhubungan dengan relasi `many-to-many` yaitu model `Book` dan `User` yang akan kita simpan pada table `book_user`.

Peminjaman ini hanya diizinkan untuk user regular (non-admin) dan seorang user hanya diizinkan untuk meminjam sebanyak 1 buku per judul buku. Oleh karena itu, kita juga akan bahas filter untuk user regular dan validasi untuk mengecek buku yang sedang dipinjam oleh user ketika ia hendak meminjam buku.

1. Publish konfigurasi dari Sentry, dengan perintah `php artisan config:publish cartalyst/sentry`. Perintah ini akan menghasilkan file `app/config/packages/cartalyst/sentry/config.php`. Ubahlah isian file ini ditaris 123 :

```
app/config/packages/cartalyst/sentry/config.php
123   'model' => 'Cartalyst\Sentry\Users\Eloquent\User',
```

menjadi :

---

app/config/packages/cartalyst/sentry/config.php

---

```
123  'model' => 'User',
```

---

Persiapkan model User yang meng-*extends* Cartalyst\Sentry\Users\Eloquent\User :

app/models/User.php

---

```
1 <?php
2 use Cartalyst\Sentry\Users\Eloquent\User as SentryUserModel;
3
4 class User extends SentryUserModel
5 {
6
7 }
```

---

Setelah Anda membuat perubah ini, cek kembali fitur login, pastikan masih berjalan dengan baik.

2. Persiapkan table pivot. Dalam membuat fitur buku kita akan menggunakan relasi many-to-many, dimana seorang user dapat meminjam banyak buku dan sebuah buku dapat dipinjam banyak user. Untuk membuat relasi many-to-many<sup>88</sup> kita harus membuat table penengah.

Tentunya kita harus melakukan migration untuk menambah table tersebut. Laravel memudahkan kita untuk membuat table pivot ini dengan menggunakan generator. Jalankan perintah `php artisan generate:pivot users books`. Perintah ini akan menghasilkan file migrasi `xxxx_xx_xx_xxxxxx_create_books_users_table.php`. Dengan isian awal seperti ini :

app/database/migrations/xxxx\_xx\_xx\_xxxxxx\_create\_books\_users\_table.php

---

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5
6 class CreateBookUserTable extends Migration {
7
8     /**
9      * Run the migrations.
10     *
11     * @return void
12     */
13    public function up()
14    {
15        Schema::create('book_user', function(Blueprint $table) {
16            $table->increments('id');
17            $table->integer('book_id')->unsigned()->index();
18            $table->foreign('book_id')->references('id')->on('books')->onDelete('cascade');
19        });

```

---

<sup>88</sup><http://laravel.com/docs/eloquent#many-to-many>

```

20         $table->integer('user_id')->unsigned()->index();
21         $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade')\
22     );
23         $table->timestamps();
24     });
25 }
26
27
28 /**
29 * Reverse the migrations.
30 *
31 * @return void
32 */
33 public function down()
34 {
35     Schema::drop('book_user');
36 }
37
38 }
```

---

Kita perlu menambahkan kolom returned yang bertipe boolean dan constraint on update. Ubah fungsi up menjadi :

app/database/migrations/xxxx\_xx\_xx\_xxxxxx\_create\_books\_users\_table.php

---

```

1 ....
2 public function up()
3 {
4     Schema::create('book_user', function(Blueprint $table) {
5         $table->increments('id');
6         $table->integer('book_id')->unsigned()->index();
7         $table->foreign('book_id')->references('id')->on('books')
8             ->onDelete('cascade');
9             ->onUpdate('cascade');
10        $table->integer('user_id')->unsigned()->index();
11        $table->foreign('user_id')->references('id')->on('users')
12            ->onDelete('cascade')
13            ->onUpdate('cascade');
14        $table->boolean('returned')->default(false);
15        $table->timestamps();
16    });
17 }
18 ....
```

---

Sekarang jalankan migration dengan perintah `php artisan migrate`. Pastikan table book\_user muncul di database.

3. **Persiapkan relasi pivot.** Kita perlu mengkonfigurasi model Book dan User agar menggunakan relasi many-to-many yang dibuat. Syntax dasar untuk mengkonfigurasi relasi many-to-many seperti ini :

contoh relasi many-to-many dari user ke buku

---

```

1  public function books()
2  {
3      return $this->belongsToMany( 'Book' );
4 }
```

---

Dengan syntax ini, Laravel mengasumsikan terdapat table book\_user dengan kolom user\_id dan book\_id. Jika Anda akan menggunakan skema table yang berbeda, silahkan cek [dokumentasi resmi](#)<sup>89</sup>.

Karena kita menambahkan field returned maka kita perlu menggunakan withPivot('returned').

Untuk mendapatkan tanggal pinjam dan tanggal pengembalian kita akan menggunakan timestamps untuk membuat kolom created\_at dan updated\_at. Kolom created\_at akan kita gunakan sebagai tanggal pinjam, sedangkan kolom updated\_at akan kita gunakan sebagai tanggal kembali. Untuk menambahkan timestamps kita cukup menggunakan withTimestamps().

Hasil akhir dari relasi many-to-many ini sebagai berikut :

app/models/User.php

---

```

1  ....
2  public function books()
3  {
4      return $this->belongsToMany( 'Book' )->withPivot( 'returned' )->withTimestamps();
5  }
6  ....
```

---

app/models/Book.php

---

```

1  ....
2  public function users()
3  {
4      return $this->belongsToMany( 'User' )->withPivot( 'returned' )->withTimestamps();
5  }
6  ....
```

---

Setelah kita menambahkan relasi ini, kita dapat menggunakan method attach untuk menghubungkan model dan detach untuk melepas hubungan model. Contohnya :

```

1  $book = Book::find(1);
2  $user = User::find(1);
3  // menambah record
4  $user->books->attach($book);
```

<sup>89</sup><http://laravel.com/docs/eloquent#working-with-pivot-tables>

4. Persiapkan method di model Book. Untuk menambahkan peminjaman buku, kita akan membuat method borrow pada model Book yang akan mengecek User yang sedang login kemudian meng-attach user tersebut ke model Book.

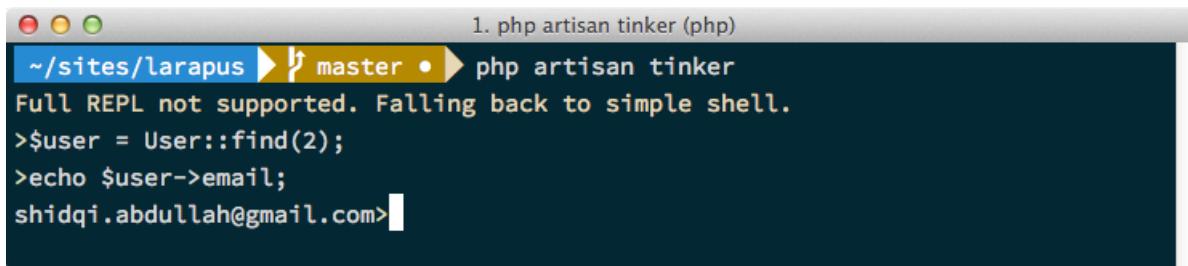
app/models/Book.php

```

1  ....
2  public function borrow()
3  {
4      // ambil user yang sedang login
5      $user = Sentry::getUser();
6
7      // attach user ke buku
8      return $this->users()->attach($user);
9  }
10 ....

```

Untuk mengetes method ini, kita akan menggunakan tinker. Tinker merupakan REPL (Read Eval Print Loop) untuk Laravel yang memudahkan kita untuk berinteraksi dengan aplikasi yang sedang dibangun menggunakan terminal. Untuk mengaktifkan tinker gunakan perintah `php artisan tinker`. Berikut ini contoh penggunaan tinker untuk mengakses model User.



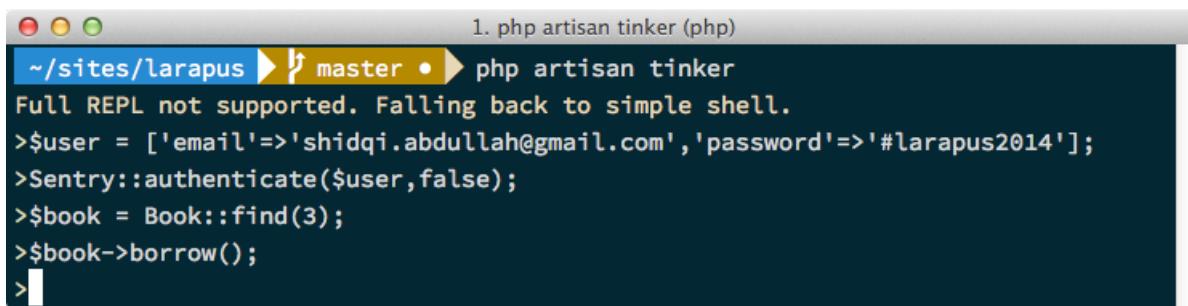
```

1. php artisan tinker (php)
~/sites/larapus ➔ master • ➔ php artisan tinker
Full REPL not supported. Falling back to simple shell.
>$user = User::find(2);
>echo $user->email;
shidqi.abdullah@gmail.com>

```

Contoh penggunaan tinker

Pada method yang akan kita cek, kita harus mengauthentikasikan dulu seorang user. Setelah user berhasil login, kemudian kita panggil method borrow di model Book.



```

1. php artisan tinker (php)
~/sites/larapus ➔ master • ➔ php artisan tinker
Full REPL not supported. Falling back to simple shell.
>$user = ['email'=>'shidqi.abdullah@gmail.com', 'password'=>'#larapus2014'];
>Sentry::authenticate($user, false);
>$book = Book::find(3);
>$book->borrow();
>

```

Mengecek method borrow dengan tinker

Sekarang cek database, pastikan telah muncul record baru di table book\_user.

The screenshot shows the MySQL Workbench interface with the database 'larapus' selected. The 'book\_user' table is open, displaying one record. The table structure includes columns for id, book\_id, user\_id, returned, created\_at, and updated\_at. The record shows a borrow entry for book\_id 3 by user\_id 2. The table information panel provides details about the table's creation date, engine, rows, size, encoding, and auto-increment settings.

Method `borrow` berhasil menambah record

Terlihat diatas, kita telah berhasil membuat record baru dengan isian `user_id` dan `book_id` yang sesuai dengan perintah yang kita jalankan.

5. **Persiapkan controller.** Tambahkan fungsi `borrow` pada `BooksController` :

app/controllers/BooksController.php

```

1  ....
2  public function borrow($id)
3  {
4      $book = Book::findOrFail($id);
5      $book->borrow();
6      return Redirect::back()->with("successMessage", "Anda telah meminjam $book->title");
7  }
8  ....

```

Pada fungsi ini kita mencari buku sesuai `$id` yang dikirim dan memanggil method `borrow` untuk meminjamkan buku itu pada user yang sedang login. Terakhir, kita mengarahkan user ke halaman sebelumnya dengan pesan sukses.

Untuk memudahkan mengatur semua fitur untuk user Member, kita akan membuat controller baru dengan nama `MemberController`. Fungsi pertama di file ini yaitu fungsi `books` untuk menampilkan halaman peminjaman buku :

---

app/controllers/MemberController.php

---

```
1   ....
2 <?php
3
4 class MemberController extends BaseController {
5
6 /**
7 * Tampilkan halaman peminjaman buku
8 * @return response
9 */
10 public function books()
11 {
12     if(Datatable::shouldHandle())
13     {
14         // eager load author untuk menghemat query sql
15         return Datatable::collection(Book::with('author')->get())
16             ->showColumns('id', 'title', 'amount')
17             // menggunakan closure untuk menampilkan nama penulis dari relasi
18             ->addColumn('author', function ($model) {
19                 return $model->author->name;
20             })
21             // menggunakan helper untuk membuat link
22             ->addColumn('', function ($model) {
23                 return link_to_route('borrow', 'Pinjam', ['book'=>$model->id]);
24             })
25             ->searchColumns('title', 'amount', 'author')
26             ->orderColumns('title', 'amount', 'author')
27             ->make();
28     }
29     return View::make('books.borrow')->withTitle('Pilih buku');
30 }
31 }
32 ....
```

---

Jangan lupa untuk menambahkan route untuk kedua fungsi ini :

**app/routes.php**

```

1   ....
2   Route::group(array('before' => 'auth'), function () {
3     ....
4     Route::get('books', array('as'=>'member.books', 'uses'=>'MemberController@books'));
5     Route::get('books/{book}/borrow', array('as'=>'books.borrow', 'uses'=>'BooksController\
6 r@borrow'));
7     ....
8   }
9   ....

```

Kita menggunakan [Named Routes<sup>90</sup>](#) untuk membuat kedua route ini. Dengan menggunakan named routes, akan memudahkan kita ketika memanggil sebuah route. Pada contoh diatas, untuk memanggil route books (MemberController@books) kita cukup menggunakan member.books.

Untuk membuat route borrow kita menggunakan [Route Parameters<sup>91</sup>](#). Dengan menggunakan route parameters, variabel {book} akan berubah menjadi variabel \$id pada fungsi borrow di BooksController.

6. **Persiapkan view**, untuk menampilkan halaman peminjaman buku. Halaman ini akan menggunakan datatable untuk menampilkan daftar buku dan sebuah link untuk meminjam buku. Buatlah file ini di app/views/books/borrow.blade.php.

**app/views/books/borrow.blade.php**

```

1 @extends('layouts.master')
2
3 @section('asset')
4   @include('layouts.partials.datatable')
5 @stop
6
7 @section('title')
8   {{ $title }}
9 @stop
10
11 @section('content')
12
13   {{ Datatable::table()
14     ->addColumn('id', 'title', 'author', 'amount', '')
15     ->setOptions('aoColumnDefs', array(
16       array(
17         'bVisible' => false,
18         'aTargets' => [0]),
19       array(
20         'sTitle' => 'Judul',
21         'aTargets' => [1]),


```

<sup>90</sup><http://laravel.com/docs/routing#named-routes>

<sup>91</sup><http://laravel.com/docs/routing#route-parameters>

```

22     array(
23         'sTitle' => 'Jumlah',
24         'aTargets' => [2]),
25     array(
26         'sTitle' => 'Penulis',
27         'aTargets' => [3]),
28     array(
29         'bSortable' => false,
30         'aTargets' => [4])
31   ))
32   ->setOptions('bProcessing', true)
33   ->setUrl(route('member.books'))
34   ->render('datatable.uikit'))}}
35 @stop

```

Pada view ini, kita telah membuat datatable yang berisi daftar buku dan link untuk meminjam buku. Cobalah fitur pinjam buku ini dan cek pastikan muncul record baru di database.

Judul	Jumlah	Penulis	Pinjam
Kupinang Engkau dengan Hamdalah	3	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	3	Aam Amiruddin	<a href="#">Pinjam</a>

### Fitur pinjam buku

Fitur pinjam buku ini selain dapat diakses oleh user yang sudah login, dapat pula diakses oleh user yang belum login. Tapi, ketika user tersebut klik link *pinjam*, maka dia akan diarahkan ke halaman login, kemudian ke dashboard. Mari saya tunjukan bagaimana saya membuatnya.

- Untuk menampilkan daftar buku yang bisa dipinjam, kita akan menggunakan tampilan yang sama seperti tampilan halaman peminjaman buku untuk user yang sudah login. Karena itu, mari kita *refactor* handle untuk datatable di MemberController dan dipindahkan ke BooksController.

Ubah method books pada MemberController menjadi :

app/controllers/MemberController.php

```

1 <?php
2
3 class MemberController extends BaseController {
4     public function books()
5     {
6         return View::make('books.borrow')->withTitle('Pilih buku');
7     }
8 }
```

Tambahkan method borrowDatatable di BooksController untuk menghandle datatable :

app/controllers/BooksController.php

```

1 ....
2 public function borrowDatatable()
3 {
4     // eager load author untuk menghemat query sql
5     return Datatable::collection(Book::with('author')->get())
6         ->showColumns('id', 'title', 'amount')
7         // menggunakan closure untuk menampilkan nama penulis dari relasi
8         ->addColumn('author', function ($model) {
9             return $model->author->name;
10        })
11        // menggunakan helper untuk membuat link
12        ->addColumn('', function ($model) {
13            return link_to_route('books.borrow', 'Pinjam', [ 'book'=>$model->id]);
14        })
15        ->searchColumns('title', 'amount', 'author')
16        ->orderColumns('title', 'amount', 'author')
17        ->make();
18    }
19 ....
```

Tambahkan route untuk handle datatable tersebut sebelum grup filter auth, tujuannya agar route ini bisa diakses walaupun user belum login :

**app/routes.php**


---

```

1   ....
2   Route::get('datatable/books/borrow', array('as'=>'datatable.books.borrow', 'uses'=>'Books\
3 Controller@borrowDatatable'));
4   Route::group(array('before' => 'auth'), function () { ...
5   ....

```

---

Karena tampilan datatable ini akan digunakan juga pada tampilan user yang belum login, mari kita buat datatable menjadi subview di app/views/books/\_borrowdatatable.blade.php :

**app/views/books/\_borrowdatatable.blade.php**


---

```

1  {{ Datatable::table()
2      ->addColumn('id', 'title', 'author', 'amount', '')
3      ->setOptions('aoColumnDefs', array(
4          array(
5              'bVisible' => false,
6              'aTargets' => [0]),
7          array(
8              'sTitle' => 'Judul',
9              'aTargets' => [1]),
10         array(
11             'sTitle' => 'Jumlah',
12             'aTargets' => [2]),
13         array(
14             'sTitle' => 'Penulis',
15             'aTargets' => [3]),
16         array(
17             'bSortable' => false,
18             'aTargets' => [4])
19     ))
20     ->setOptions('bProcessing', true)
21     ->setUrl(route('datatable.books.borrow'))
22     ->render('datatable.uikit') }}

```

---

Ubahlah isian app/views/books/borrow.blade.php menjadi :

---

app/views/books/\_borrowdatatable.blade.php

---

```

1  @extends('layouts.master')
2
3  @section('asset')
4      @include('layouts.partials.datatable')
5  @stop
6
7  @section('title')
8      {{ $title }}
9  @stop
10
11 @section('content')
12     @include('books._borrowdatatable')
13 @stop

```

---

2. Kini kita persiapkan method index di GuestController sebagai root dari aplikasi kita. Selain itu, method ini juga akan me-redirect user ke halaman dashboard jika dia sudah login.

---

app/controllers/GuestController.php

---

```

1  ....
2  public function index()
3  {
4      // Redirect ke dashboard jika user sudah login
5      if (Sentry::check()) {
6          return Redirect::to('dashboard');
7      }
8      // Tampilkan halaman index
9      $this->layout->content = View::make('guest.index')->withTitle("Daftar Buku");
10 }
11 ....

```

---

Buat view untuk method ini di app/views/guest/index.blade.php dengan isi :

---

app/views/guest/index.blade.php

---

```

1  @section('asset')
2      @include('layouts.partials.datatable')
3  @stop
4
5  @section('content')
6      <h1 class="uk-heading-large">{{ $title }}</h1>
7      @include('books._borrowdatatable')
8  @stop

```

---

Agar method ini dipanggil ketika mengakses root dari aplikasi, ubah route untuk / menjadi :

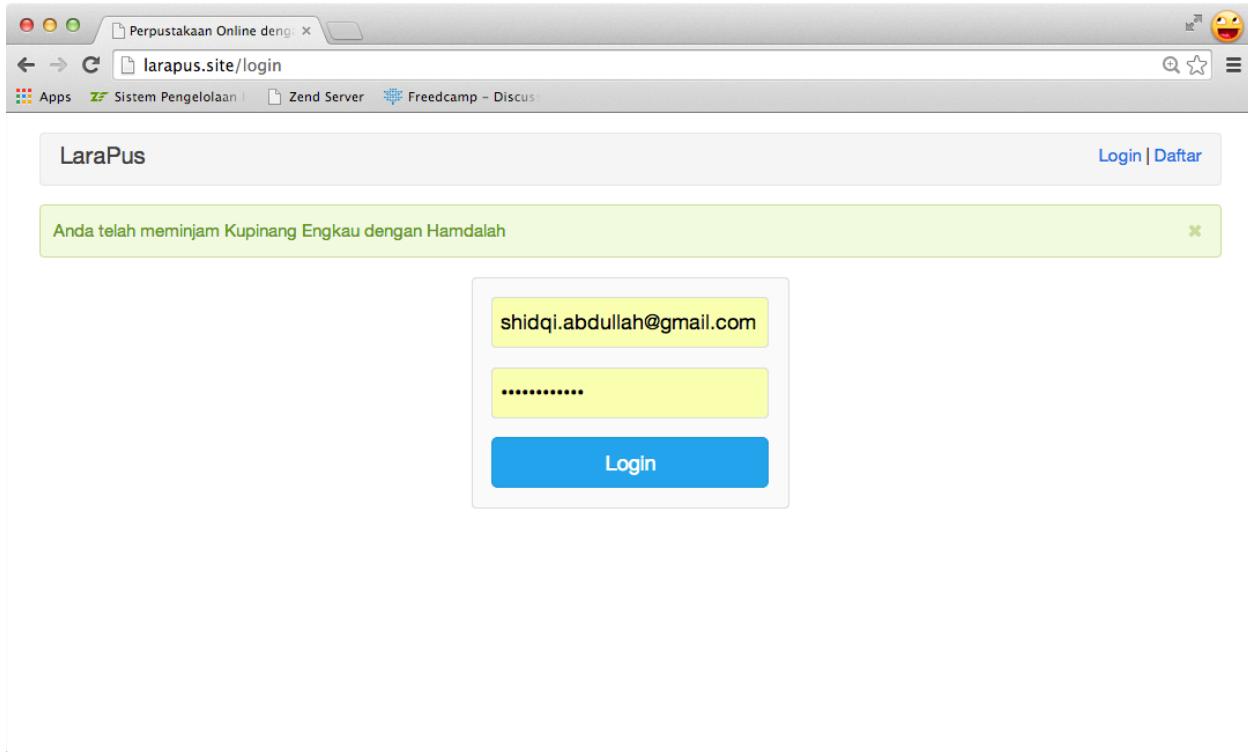
**app/routes.php**

```

1  ....
2  Route::get('/', 'GuestController@index');
3  ....

```

Kini cobalah meminjam buku ketika Anda belum login, dan ternyata...

**Error karena tidak redirect**

Ups, terlihat ada yang ganjil disini. User yang sudah login malah dikembalikan ke halaman login dan ditampilkan pesan berhasil pinjam buku. Sementara, *behaviour* yang kita inginkan adalah user diarahkan ke halaman dashboard. Untuk itu kita perlu merubah method `login` di `GuestController` yang berfungsi menampilkan halaman login menjadi :

**app/controllers/.php**

```

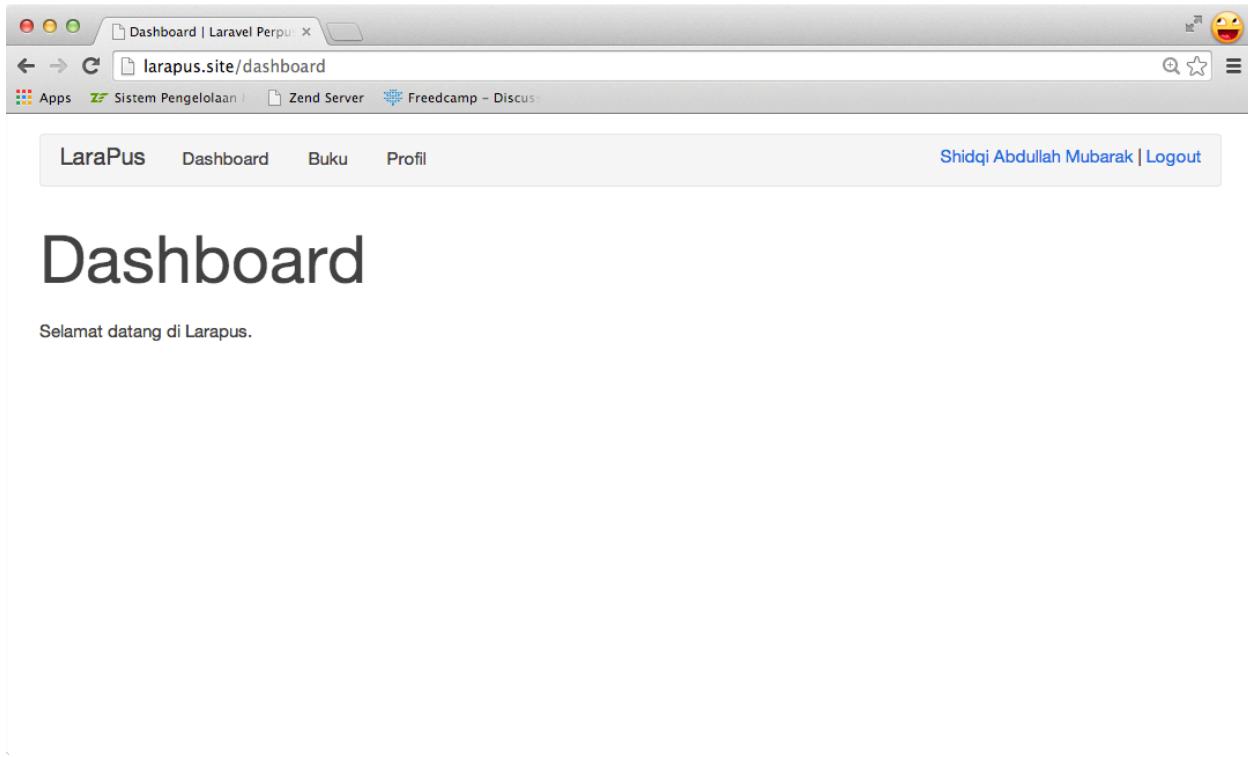
1  ....
2  public function login()
3  {
4      // redirect ke dashboard jika user sudah login
5      if (Sentry::check()) {
6          return Redirect::to('dashboard');
7      }
8
9      $this->layout->content = View::make('guest.login');

```

```
10     }
11     ....
```

---

Mari coba cek lagi pinjam buku sebelum login, dan kita berhasil di redirect ke halaman dashboard.



Redirect ke dashboard, tapi flash message tidak muncul

Tapi, masih ada yang kurang disini (#duh), tampilan notifikasinya bahwa kita telah meminjam buku tidak muncul di dashboard. Hal ini terjadi, karena kita menggunakan flash message. Flash message hanya akan muncul pada 1 kali pada request berikutnya. Agar flash message muncul kembali di halaman dashboard, kita perlu menggunakan `Session::reflash()` di method login :

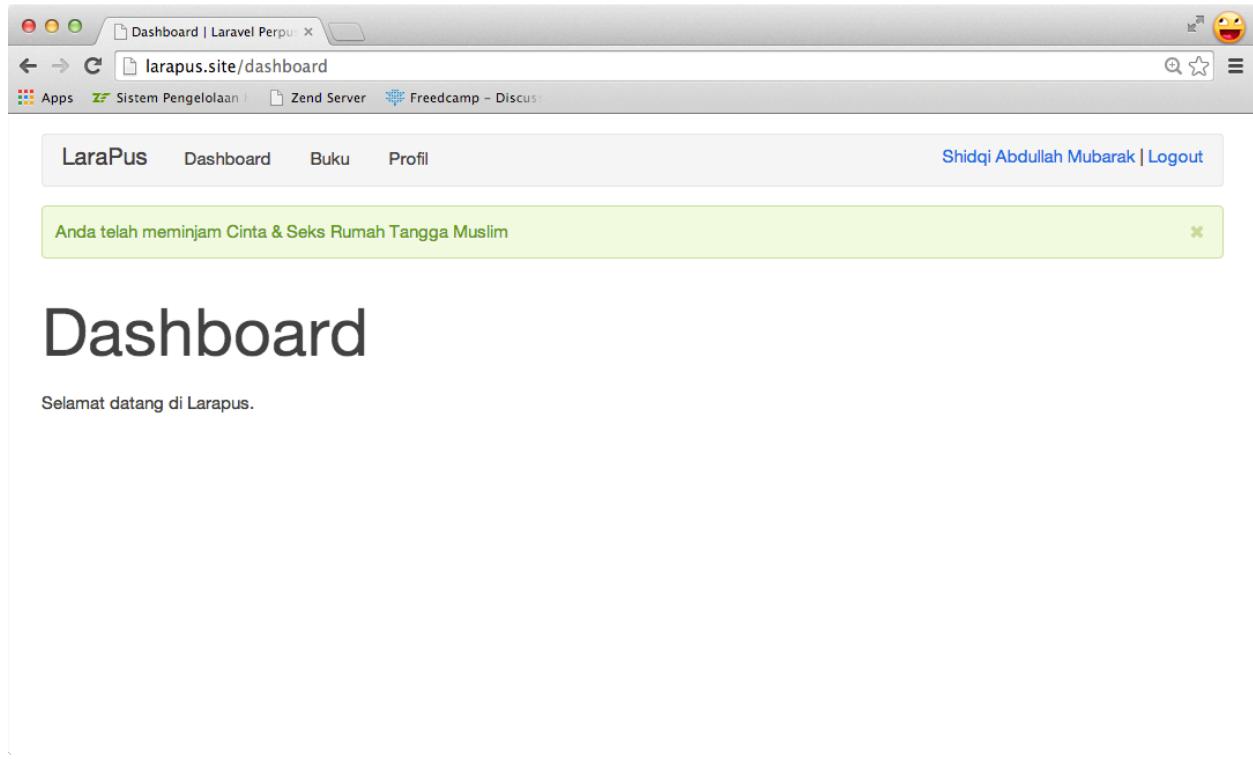
app/controllers/GuestController.php

---

```
1     ....
2     public function login()
3     {
4         // redirect ke dashboard jika user sudah login
5         if (Sentry::check()) {
6             Session::reflash();
7             return Redirect::to('dashboard');
8         }
9
10        $this->layout->content = View::make('guest.login');
11    }
12    ....
```

---

Coba cek lagi pinjam buku sebelum login, kali ini pastikan flash message muncul di halaman dashboard.



Flash message muncul di dashboard setelah di `reflash`

Terakhir, kita juga perlu memfilter peminjaman buku agar hanya dapat diakses oleh user regular. Tambahkan filter `member` di `app/filters.php` dengan isi :

#### app/filters.php

```

1  ....
2  Route::filter('regularUser', function() {
3      $user = Sentry::getUser();
4      // Cari grup regular
5      $regular = Sentry::findGroupByName('regular');
6      if (!$user->inGroup($regular)) {
7          return Redirect::to('dashboard')->with("errorMessage", "Hanya user regular yang d\
8  iizinkan untuk mengakses fitur tersebut.");
9      }
10 });
11 ....

```

Filter ini hanya akan dilakukan pada method `borrow` di `BooksController`. Oleh karena itu, kita akan menggunakan [Controller Filters<sup>92</sup>](#) untuk memfilter per method di controller. Filter ini harus ditambahkan di method `__construct` di controller. Berikut isian untuk `BooksController` :

<sup>92</sup><http://laravel.com/docs/controllers#controller-filters>

---

app/controllers/BooksController.php

---

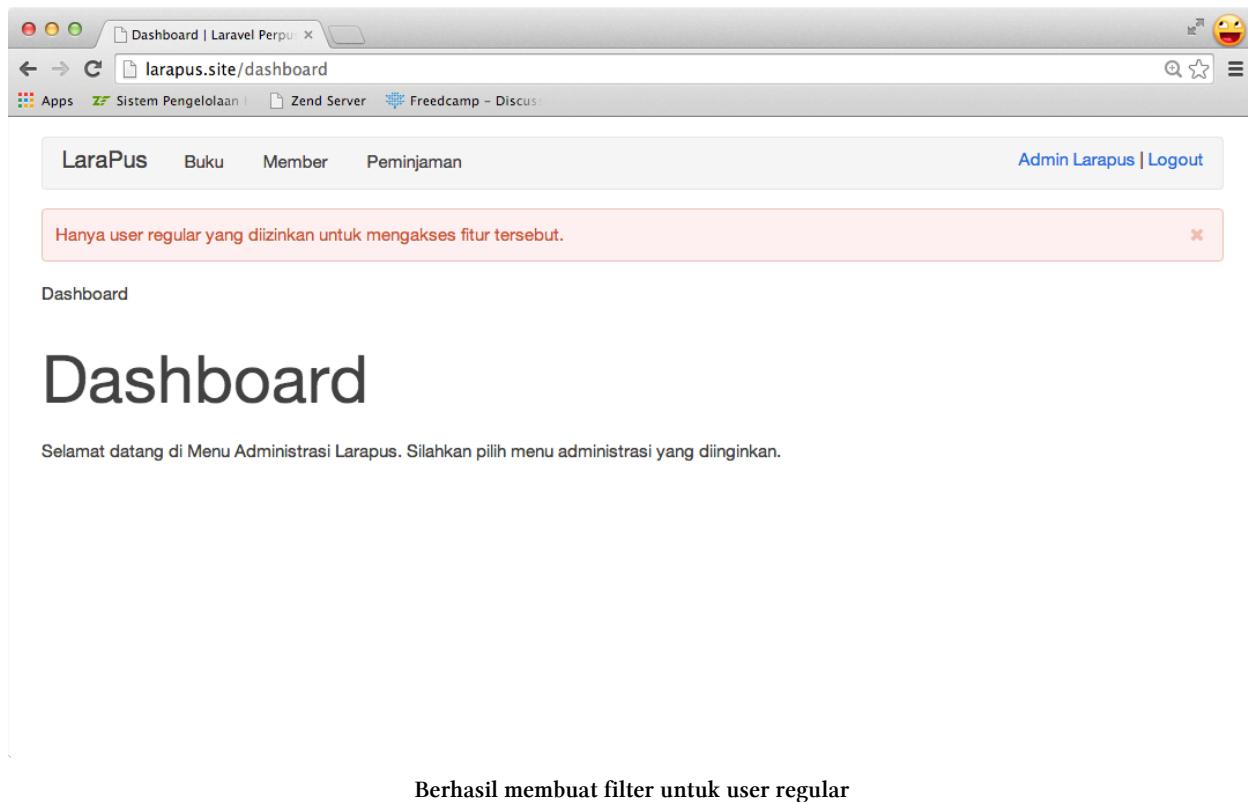
```

1   ....
2   public function __construct()
3   {
4       // Letakan filter regularUser sebelum memanggil fungsi borrow
5       $this->beforeFilter('regularUser', array('only' => array('borrow')));
6   }
7   ....

```

---

Syntax diatas akan memberikan filter `regularUser` hanya (only) pada method `borrow`. Sekarang test pinjam buku sebagai user yang belum login, kemudian ketika muncul halaman login, loginlah sebagai admin. Pastikan pesan kesalahan muncul.



Berhasil membuat filter untuk user regular

## Validasi

Dalam Larapus, jumlah buku yang boleh dipinjam oleh seorang user dalam satu waktu maksimal sebanyak satu buah. Dalam mengembangkan bagian ini saya akan menggunakan `Exception`<sup>93</sup> yang merupakan salah satu fitur PHP 5. Secara sederhana, `exception` berfungsi untuk melemparkan (istilahnya `throw`) pemberitahuan ke sistem bahwa ada sesuatu yang salah. Nah, exception ini bisa ditangkap (istilahnya `catch`) kemudian kita melakukan sesuatu dengan exception yang ditangkap itu.

Cara paling sederhana untuk membuat exception adalah dengan syntax :

---

<sup>93</sup><http://www.php.net/manual/en/language.exceptions.php>

```
1 throw new Exception("Ada yang salah disini.");
```

Kemudian kita catch exception ini dengan syntax :

```
1 try {
2     // syntax yang akan melemparkan
3 } catch (Exception $e) {
4     // lakukan sesuatu
5 }
```

Di Larapus, saya akan membuat exception baru dengan nama BookAlreadyBorrowedException. Exception ini akan di-throw ketika method borrow dipanggil dari model Book jika buku tersebut sudah pinjam. Tanda bahwa buku sedang dipinjam adalah field returned di book\_user berisi 0. Exception ini akan saya catch dari method borrow di BookController untuk selanjutnya me-*redirect* user ke halaman sebelumnya berikut flash message yang menjelaskan bahwa dia sedang meminjam buku tersebut.

Bingung ya? *Sama*. Tapi tenang, saya yakin Anda akan lebih paham kalau sudah mulai coding.. :)

1. Sebelum memulai, kita ingin database dikosongkan dan diberi sample data untuk table book\_user. Pada table ini kita akan membuat dua record yang menandakan seorang user sedang meminjam dua buku. Buatlah file ini di app/database/seeds/BookUserTableSeeder.php :

app/database/seeds/BookUserTableSeeder.php

---

```
1 <?php
2
3 class BookUserTableSeeder extends Seeder {
4
5     public function run()
6     {
7         // kosongkan database
8         DB::table('book_user')->delete();
9
10        // Karena id dari user dari Sentry selalu berubah ketika melakukan db:seed, kita \
11        ambil id user dengan query.
12        $userId = User::where('email', 'shidqi.abdullah@gmail.com')->first()->id;
13
14        // buat array untuk diinput
15        $bookusers = [
16            ['id'=>1, 'book_id'=>1, 'user_id'=>$userId, 'returned'=>0, 'created_at'=>'201\
17            4-05-20 08:03:57', 'updated_at'=>'2014-05-20 08:03:57'],
18            ['id'=>2, 'book_id'=>2, 'user_id'=>$userId, 'returned'=>0, 'created_at'=>'201\
19            4-05-20 08:03:57', 'updated_at'=>'2014-05-20 08:03:57'],
20        ];
21
22        // insert data ke database
```

```

23         DB::table('book_user')->insert($bookusers);
24     }
25
26 }
```

---

Tambahkan seeder ini ke DatabaseSeeder :

app/database/seeds/DatabaseSeeder.php

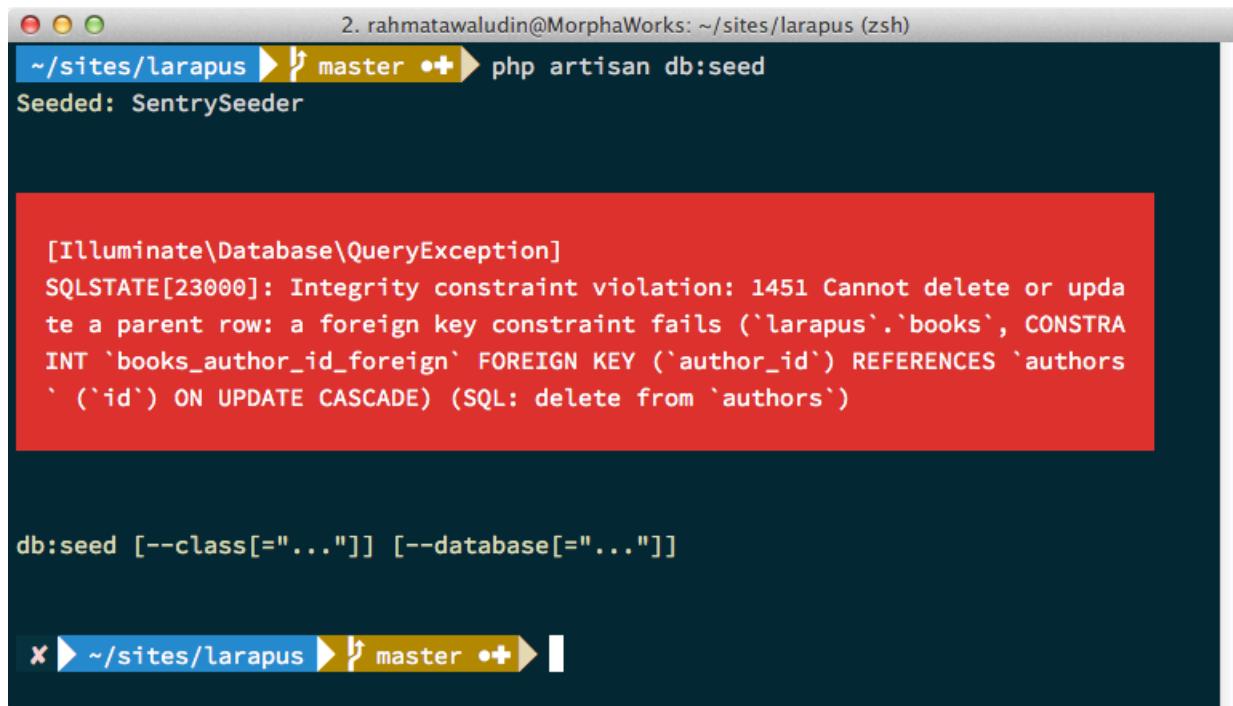
---

```

1 ....
2 public function run()
3 {
4     Eloquent::unguard();
5     $this->call('SentrySeeder');
6     $this->call('AuthorsTableSeeder');
7     $this->call('BooksTableSeeder');
8     $this->call('BookUserTableSeeder');
9 }
10 ....
```

---

Jalankan php artisan db:seed :



The screenshot shows a terminal window with the following output:

```

2. rahmatawaludin@MorphaWorks: ~/sites/larapus (zsh)
~/sites/larapus ▶ ⌂ master •+▶ php artisan db:seed
Seeded: SentrySeeder

[Illuminate\Database\QueryException]
SQLSTATE[23000]: Integrity constraint violation: 1451 Cannot delete or update a parent row: a foreign key constraint fails (`larapus`.`books`, CONSTRAINT `books_author_id_foreign` FOREIGN KEY (`author_id`) REFERENCES `authors` (`id`) ON UPDATE CASCADE) (SQL: delete from `authors`)

db:seed [--class[="..."]] [--database[="..."]]

x ▶ ~/sites/larapus ▶ ⌂ master •+▶
```

Seeding gagal karena relasi

Ops, ternyata ada error. Error ini terjadi ketika kita hendak menghapus record di table users tapi masih ada relasi ke record di table book\_user. Untuk itu kita perlu menonaktifkan pengecekan *foreign*

*key* untuk sementara. Ketika buku ini ditulis, Laravel belum menyediakan fitur untuk menonaktifkan pengecekan foreign key. Untuk menikapinya kita akan menggunakan `DB::statement()` untuk menonaktifkan pengecekan foreign key dengan perintah `SET FOREIGN_KEY_CHECKS=0;` untuk menonaktifkan dan `SET FOREIGN_KEY_CHECKS=1;` untuk mengaktifkannya kembali di `DatabaseSeeder` :

---

app/database/seeds/DatabaseSeeder.php

---

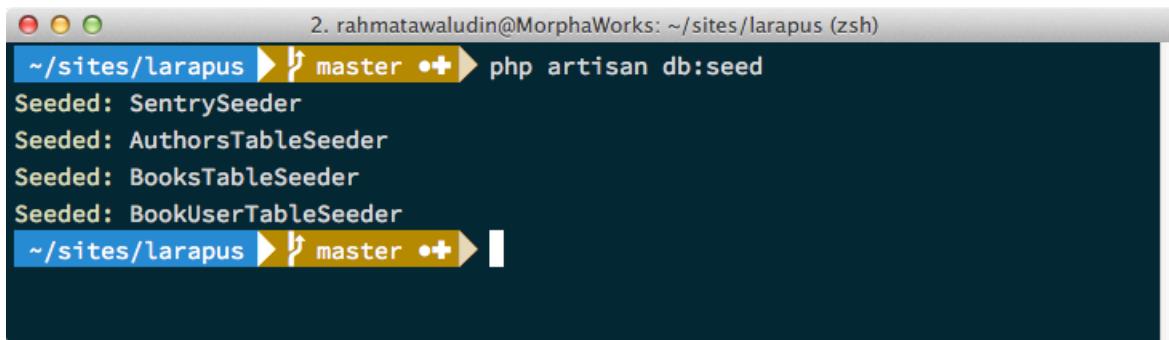
```

1   ....
2   public function run()
3   {
4       // disable foreign key checks
5       DB::statement('SET FOREIGN_KEY_CHECKS=0;');
6
7       Eloquent::unguard();
8       $this->call('SentrySeeder');
9       $this->call('AuthorsTableSeeder');
10      $this->call('BooksTableSeeder');
11      $this->call('BookUserTableSeeder');
12
13      // enable foreign key checks
14      DB::statement('SET FOREIGN_KEY_CHECKS=1;');
15  }
16  ....

```

---

Jalankan kembali `php artisan db:seed` :



```

2. rahmatawaludin@MorphaWorks: ~/sites/larapus (zsh)
~/sites/larapus ▶ ⌂ master •+▶ php artisan db:seed
Seeded: SentrySeeder
Seeded: AuthorsTableSeeder
Seeded: BooksTableSeeder
Seeded: BookUserTableSeeder
~/sites/larapus ▶ ⌂ master •+▶

```

Seeding berhasil setelah menonaktifkan pengecekan *foreign key*

2. *Exception* yang akan kita buat disimpan di file `app/exceptions/BookAlreadyBorrowedException.php` :

---

app/exceptions/BookAlreadyBorrowedException.php

---

```

1  <?php
2  class BookAlreadyBorrowedException extends Exception { }

```

---

Kita perlu me-*load* exception ini dengan class loader dari Laravel (bukan composer) dengan cara menambahkannya di `app/start/global.php` :

**app/start/global.php**

```

1   ....
2   ClassLoader::addDirectories(array(
3
4       app_path().'/commands',
5       app_path().'/controllers',
6       app_path().'/models',
7       app_path().'/database/seeds',
8       app_path().'/exceptions',
9
10    ));
11   ....

```

3. Menggunakan pivot, kita akan mengecek apakah user telah meminjam buku dengan mengecek field `returned`. Jika field ini berisi 0 berisi buku tersebut sedang dipinjam dan kita akan melemparkan (`throw`) exception `BookAlreadyBorrowedException` :

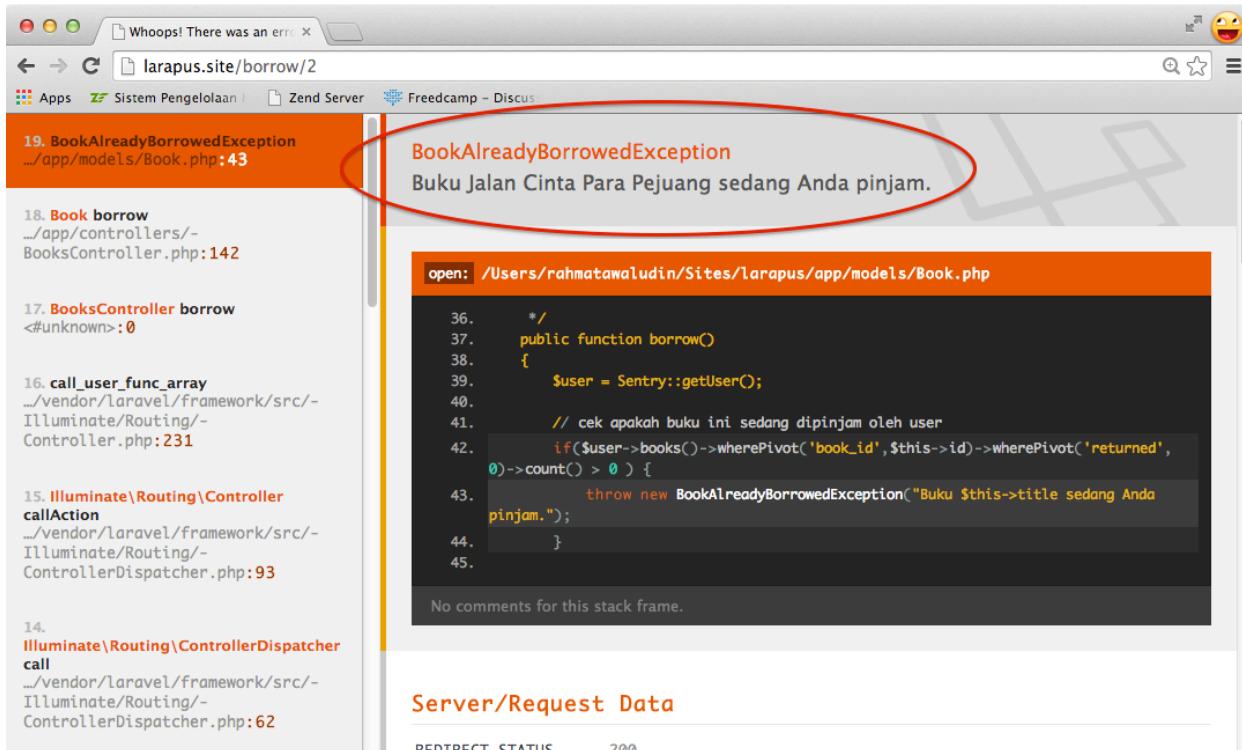
**app/start/global.php**

```

1   ....
2   public function borrow()
3   {
4       $user = Sentry::getUser();
5
6       // cek apakah buku ini sedang dipinjam oleh user
7       if( $user->books()->wherePivot('book_id',$this->id)->wherePivot('returned', 0)->count\(
8           () > 0 ) {
9               throw new BookAlreadyBorrowedException("Buku $this->title sedang Anda pinjam.");
10          }
11
12       return $this->users()->attach($user);
13   }
14   ....

```

Untuk mengecek apakah exception ini berhasil mengecek buku yang sedang dipinjam, loginlah kemudian pinjam buku yang saat ini sedang dipinjam.



Exception berhasil di throw

- Setelah berhasil melemparkan exception, kita akan menangkap exception tersebut di BooksController kemudian mengarahkan user ke halaman sebelumnya berikut pesan error.

#### app/controllers/BooksController.php

```

1   ....
2   public function borrow($id)
3   {
4       $book = Book::findOrFail($id);
5
6       try {
7           $book->borrow();
8       } catch (BookAlreadyBorrowedException $e) {
9           // masukkan message dari exception ke variable errorMessage
10          return Redirect::back()->with('errorMessage', $e->getMessage());
11      }
12
13      return Redirect::back()->with("successMessage", "Anda telah meminjam $book->title");
14  }
15

```

Cek kembali meminjam buku yang sedang dipinjam, pastikan muncul notifikasi seperti berikut ini.

The screenshot shows a web browser window titled "Pilih buku | Laravel Perpus". The URL in the address bar is "larapus.site/books". The page header includes "LaraPus" and the user "Shidqi Abdullah Mubarak | Logout". A message box at the top states "Buku Jalan Cinta Para Pejuang sedang Anda pinjam." Below this, the main content is titled "Pilih buku". It features a table listing four books:

Judul	Jumlah	Penulis	Aksi
Kupinang Engkau dengan Hamdalah	3	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	3	Aam Amiruddin	<a href="#">Pinjam</a>

At the bottom of the table, it says "Showing 1 to 4 of 4 entries". To the right, there are navigation links: First, Previous, **1**, Next, Last.

Berhasil menangkap exception

Dengan menggunakan teknik exception ini, kita akan lebih mudah dalam mengatur *bussiness logic* dari aplikasi kita. Sebagaimana terlihat disini, saya cenderung menaruh bussiness logic seperti validasi di Model. Tentunya, Anda dapat meletakkan validasi ini ditempat lain, tergantung kompleksitas aplikasi. Teknik exception ini, akan kita gunakan juga untuk mem-validasi buku yang akan dipinjam user berdasarkan stok yang masih tersedia pada pembahasan tentang Stok buku.

## Pengembalian Buku

Fitur ini dibangun dengan logika yang sangat sederhana, yaitu merubah isian kolom `returned` menjadi `1` pada table `book_user`. Dalam proses membangun fitur ini kita akan mempelajari bagaimana mengupdate pivot table dan mengguankan query builder dari Laravel.

1. Kita akan membuat method `returnBack` di model `Book` untuk mengembalikan buku.

## app/models/Book.php

```

1  ....
2  public function returnBack()
3  {
4      $user = Sentry::getUser();
5      return $user->books()->updateExistingPivot($this->id, ['returned'=>1], true);
6  }
7  ....

```

Pada method ini kita menggunakan fungsi `updateExistingPivot` untuk mengupdate table relasi `books` ke `users` dengan `book_id` berupa `$this->id` (id buku) dan kolom tambahan `returned` kita update menjadi 1. Parameter `true` menandakan kita juga mengupdate timestamp (`updated_at`) di table `books`. Untuk lebih jelasnya silahkan baca [dokumentasi resmi](#)<sup>94</sup>.

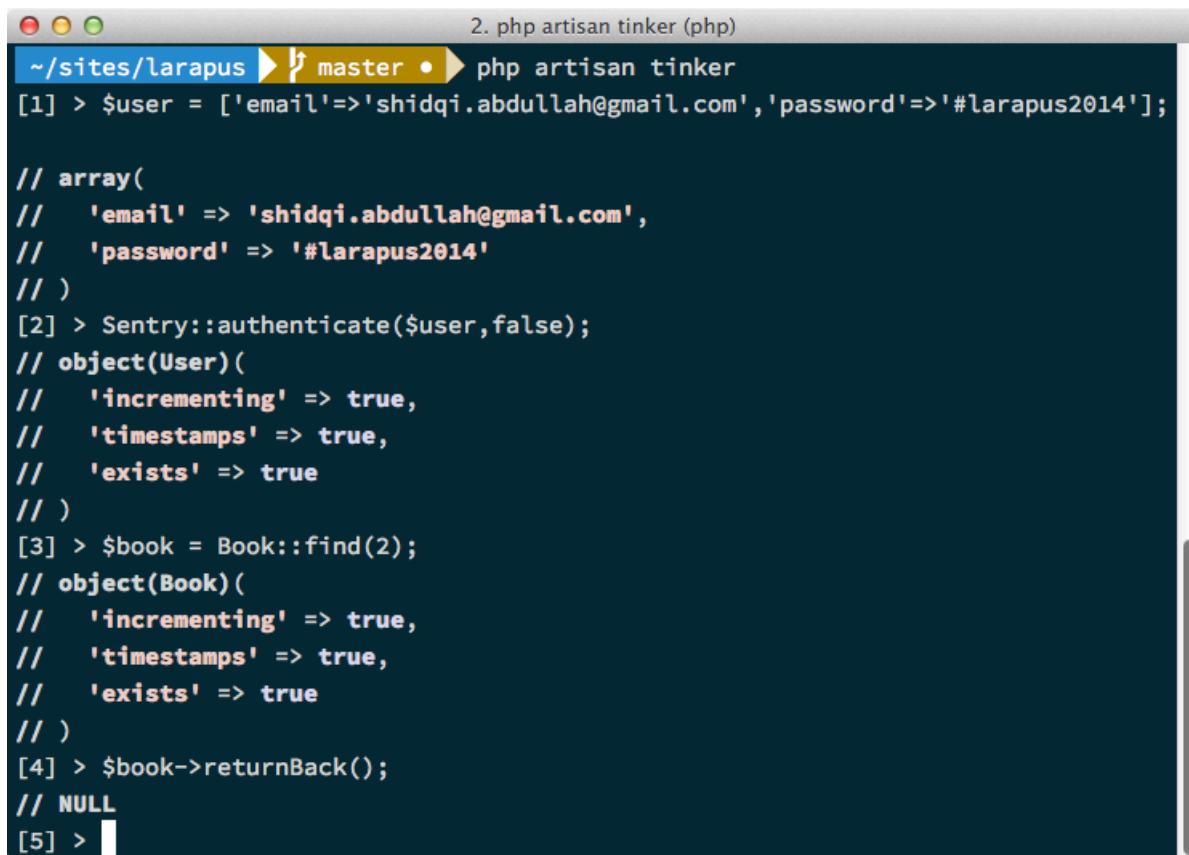
Untuk mengecek fitur ini berjalan dengan benar, kita akan menggunakan tinker. Namun sebelumnya, reset kondisi database dengan perintah `php artisan db:seed`. Pastikan kondisi database seperti berikut :

	Search: id = Filter					
	id	book_id	user_id	returned	created_at	updated_at
	1	1	16	1	2014-05-20 08:03:57	2014-05-22 06:26:12
	2	2	16	0	2014-05-20 08:03:57	2014-05-22 07:40:10

Sebelum peminjaman

Terlihat disini, user dengan id 16 meminjam buku dengan id 2. Menggunakan tinker kita akan mengembalikan buku ini:

<sup>94</sup><http://laravel.com/docs/eloquent#working-with-pivot-tables>

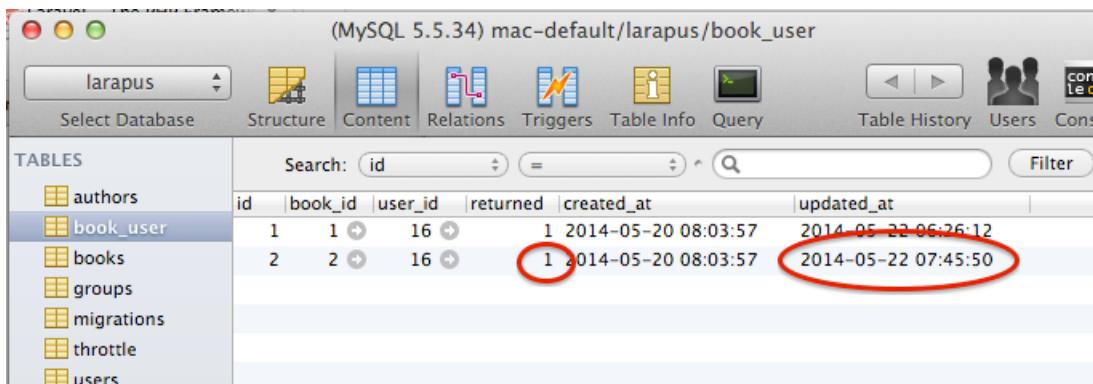


```
2. php artisan tinker (php)
~/sites/larapus ➜ master • ➜ php artisan tinker
[1] > $user = ['email'=>'shidqi.abdullah@gmail.com', 'password'=> '#larapus2014'];

// array(
//   'email' => 'shidqi.abdullah@gmail.com',
//   'password' => '#larapus2014'
// )
[2] > Sentry::authenticate($user, false);
// object(User)(
//   'incrementing' => true,
//   'timestamps' => true,
//   'exists' => true
// )
[3] > $book = Book::find(2);
// object(Book)(
//   'incrementing' => true,
//   'timestamps' => true,
//   'exists' => true
// )
[4] > $book->returnBack();
// NULL
[5] > 
```

Menggunakan tinker untuk mengembalikan buku

Cek kembali database, pastikan kolom returned berisi 1 dan kolom updated\_at sudah berubah nilainya. Nilai ini yang akan kita gunakan sebagai penanda waktu buku dikembalikan.



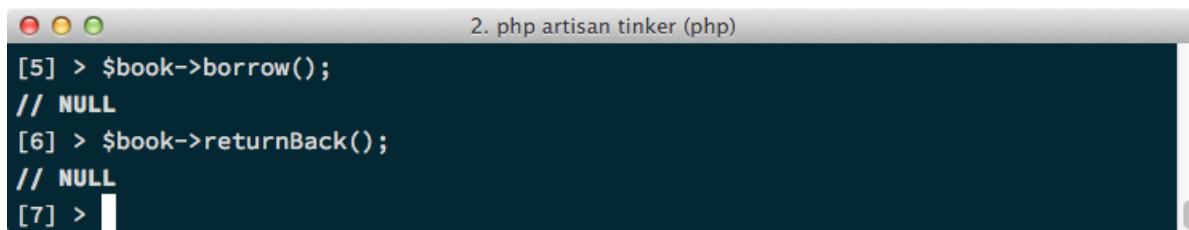
The screenshot shows the MySQL Workbench interface with the database 'larapus' selected. The 'book\_user' table is open, displaying the following data:

	id	book_id	user_id	returned	created_at	updated_at
1	1	16	+	1	2014-05-20 08:03:57	2014-05-22 06:26:12
2	2	16	+	1	2014-05-20 08:03:57	2014-05-22 07:45:50

Two rows are shown. Both rows have 'returned' set to 1 and both have an updated 'updated\_at' timestamp. The second row's timestamps are circled in red.

Berhasil mengembalikan buku

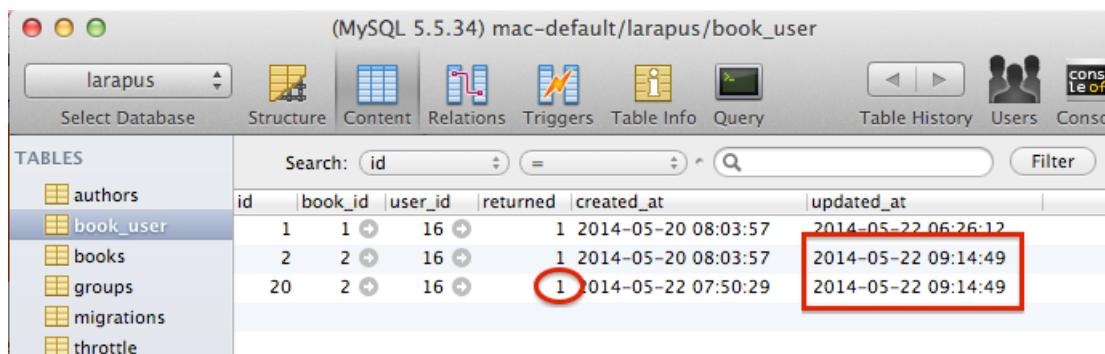
Untuk memastikan, mari kita pinjam kembali buku ini menggunakan tinker dan langsung mengembalikannya:



```
2. php artisan tinker (php)
[5] > $book->borrow();
// NULL
[6] > $book->returnBack();
// NULL
[7] >
```

Menggunakan tinker untuk meminjam dan mengembalikan buku

Cek kembali database, Opps... nampaknya ada masalah :



(MySQL 5.5.34) mac-default/larapus/book\_user

id	book_id	user_id	returned	created_at	updated_at
1	1	16	1	2014-05-20 08:03:57	2014-05-22 06:26:12
2	2	16	1	2014-05-20 08:03:57	2014-05-22 09:14:49
20	2	16	1	2014-05-22 07:50:29	2014-05-22 09:14:49

kolom updated\_at terupdate semuanya

Terlihat disini, kolom updated\_at untuk book\_id 2 berubah pada semua record. Sementara yang kita inginkan hanya untuk record dengan book\_id 2 dan returned 0. Hal ini terjadi, karena updateExistingPivot bawaan Laravel hanya akan mem-filter berdasarkan id relasi dan belum berdasarkan custom column (returned) yang kita buat.

Jika Anda mengalami hal seperti ini, yaitu keterbatasan fitur Eloquent untuk query yang ingin Anda jalankan, jangan menyerah. Seperti pepatah bilang, *banyak jalan menuju romo*, begitupun dengan fitur ini. Untuk menyelesaikan fitur ini, kita akan menggunakan *query builder* agar dapat memfilter berdasarkan book\_id dan returned.

Query Builder merupakan fitur Laravel agar kita dapat membuat query langsung pada table tanpa harus membuat model. Fitur ini sangat bermanfaat ketika kita butuh menjalankan query yang cukup rumit dan tidak terakomodasi dengan method pada Eloquent. Untuk mempelajari lebih lanjut, Anda dapat mengunjungi [dokumentasi Query Builder<sup>95</sup>](#).

Ubah method returnBack menjadi :

---

<sup>95</sup><http://laravel.com/docs/queries>

**app/models/Book.php**

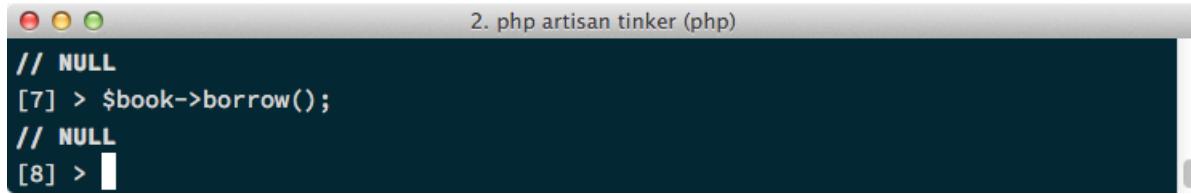
```

1   ....
2   public function returnBack()
3   {
4       $user = Sentry::getUser();
5       DB::table('book_user')
6           ->where('book_id', $this->id)
7           ->where('user_id', $user->id)
8           ->where('returned', 0)
9           ->update(array(
10               'returned' => 1,
11               'updated_at' => $this->freshTimestamp()
12           ));
13   }
14   ....

```

Pada query builder diatas, kita memilih table book\_user kemudian menggunakan where untuk memfilter berdasarkan book\_id, user\_id dan returned. Setelah records terfilter, kita mengupdate field returned menjadi 1 dan updated\_at menjadi waktu terkini. Sekedar catatan, fungsi freshTimestamp tidak ada di manual Laravel, tapi saya temukan ketika menjelajahi [API Laravel tentang class Eloquent](#)<sup>96</sup>.

Mari kita pinjam kembali buku dengan id 2 :



Meminjam buku

Pastikan database telah terisi dengan data peminjaman yang baru :

<sup>96</sup><http://laravel.com/api/source-class-Illuminate.Database\Eloquent.Model.html#1285-1293>

id	book_id	user_id	returned	created_at	updated_at
1	1	16	0	2014-05-20 08:03:57	2014-05-22 06:26:12
2	2	16	0	2014-05-20 08:03:57	2014-05-22 09:14:49
20	2	16	0	2014-05-22 07:50:29	2014-05-22 09:14:49
22	2	16	0	2014-05-22 09:16:07	2014-05-22 09:16:07

Peminjaman baru telah dibuat

Kembalikan kembali buku tersebut :

```
// NULL
[8] > $book->returnBack();
// NULL
[9] > 
```

Mengembalikan buku

Pastikan field returned berubah menjadi 1 dan hanya record tersebut yang berubah field updated\_at nya :

id	book_id	user_id	returned	created_at	updated_at
1	1	16	0	2014-05-20 08:03:57	2014-05-22 06:26:12
2	2	16	0	2014-05-20 08:03:57	2014-05-22 09:14:49
20	2	16	0	2014-05-22 07:50:29	2014-05-22 09:14:49
22	2	16	1	2014-05-22 09:16:07	2014-05-22 09:23:39

Berhasil membuat fitur pengembalian

2. Selanjutnya kita akan membuat method returnBack di BooksController dan membuat route untuk method tersebut.

**app/controllers/BooksController.php**

```

1   ....
2   public function returnBack($id)
3   {
4       $book = Book::findOrFail($id);
5       $book->returnBack();
6       return Redirect::back()->with("successMessage", "Anda telah mengembalikan $book->titl\
7 e.");
8   }
9   ....

```

Tambahkan route books.return untuk mengembalikan buku :

**app/routes.php**

```

1   ....
2   Route::get('books', array('as'=>'member.books', 'uses'=>'MemberController@books'));
3   Route::get('books/{book}/borrow', array('as'=>'books.borrow', 'uses'=>'BooksController@bo\
4 rrow'));
5   Route::get('books/{book}/return', array('as'=>'books.return', 'uses'=>'BooksController@re\
6 turnBack'));
7   ....

```

3. Pada Larapus, di halaman dashboard User akan ada daftar buku yang telah dipinjam dan disampingnya terdapat tombol kembalikan. Untuk mengembalikan buku, user cukup menekan tombol kembalikan tersebut.

Sebelum kita membuat fitur pengembalian, kita perlu membedakan view dashboar yang akan muncul untuk admin dan global admin. Caranya, kita perlu menghapus file app/views/dashboard/index.blade.php dan menggantinya dengan dua file admin.blade.php untuk tampilan admin dan regular.blade.php untuk tampilan user regular.

**app/views/dashboard/admin.blade.php**

```

1   @section('title')
2       {{ $title }}
3   @stop
4
5   @section('nav')
6       <li><a href="#">Buku</a></li>
7       <li><a href="#">Member</a></li>
8       <li><a href="#">Peminjaman</a></li>
9   @stop
10
11  @section('breadcrumb')
12      <li>{{ $title }}</li>
13  @stop
14

```

```

15 @section('content')
16     Selamat datang di Menu Administrasi Larapus. Silahkan pilih menu administrasi yang di\
17     inginkan.
18 @stop

```

---

**app/views/dashboard/regular.blade.php**

```

1  @section('title')
2      {{ $title }}
3  @stop
4
5  @section('nav')
6      <li><a href="#">Dashboard</a></li>
7      <li><a href="#">Buku</a></li>
8      <li><a href="#">Profil</a></li>
9  @stop
10
11 @section('content')
12     Selamat datang di Larapus.
13 @stop

```

---

Kita juga perlu merubah method dashboard di HomeController agar mengarahkan ke view yang sesuai dengan jenis user.

**app/controllers/HomeController.php**

```

1 ....
2 public function dashboard()
3 {
4     $user = Sentry::getUser();
5     $admin = Sentry::findGroupByName('admin');
6     $regular = Sentry::findGroupByName('regular');
7
8     // is admin
9     if ($user->inGroup($admin)) {
10         $this->layout->content = View::make('dashboard.admin')->withTitle('Dashboard');
11     }
12
13     // is regular user
14     if ($user->inGroup($regular)) {
15         $this->layout->content = View::make('dashboard.regular')->withTitle('Dashboard');
16     }
17 }
18 ....

```

---

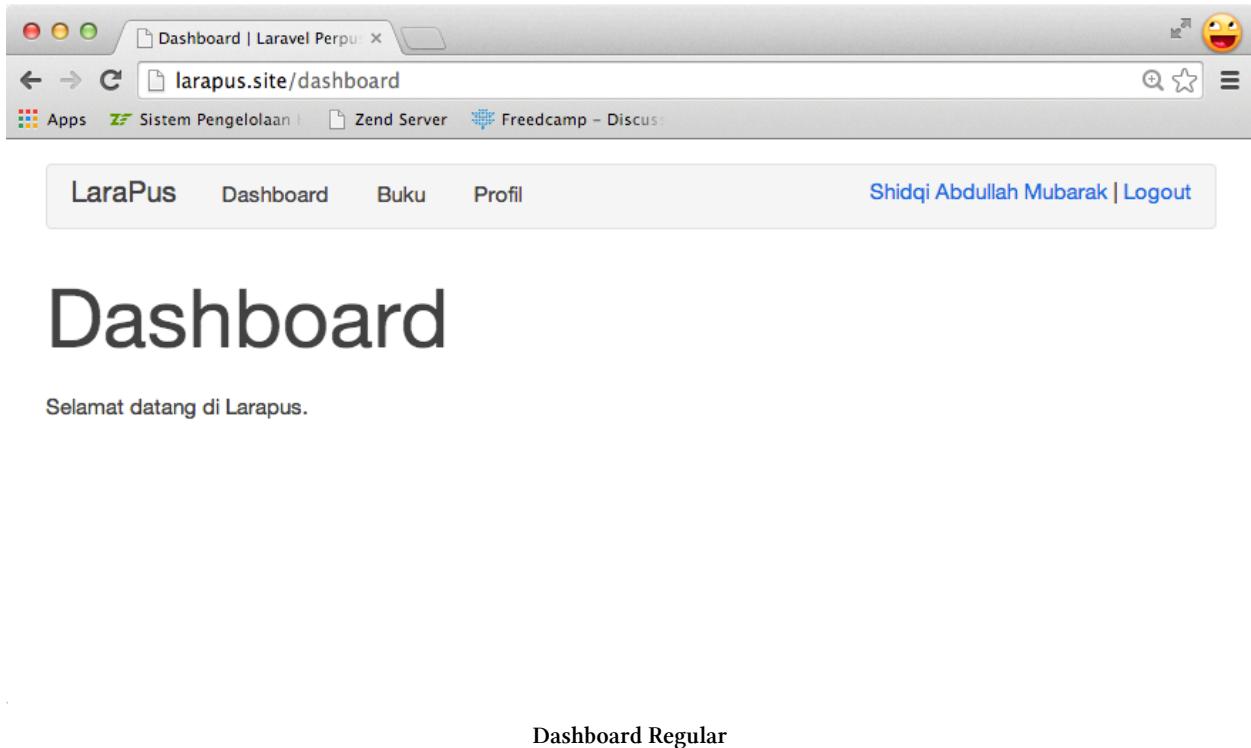
Dalam method ini, saya menggunakan grup untuk melakukan seleksi jenis User yang sedang login. Perlu dipahami, saya sengaja melakukan ini untuk menunjukkan cara mengecek grup dari User.

Sebenarnya ada teknik yang lebih sederhana, yaitu menggunakan permissions. Cara ini akan kita bahas di pembahasan tentang Dashboard.

Coba cek kembali login dengan user admin dan regular, pastikan Anda diarahkan ke dashboard yang sesuai dengan jenis User.

The screenshot shows a web browser window with the following details:

- Address Bar:** larapus.site/dashboard
- Header:** Dashboard | Laravel Perpu
- Toolbar:** Back, Forward, Stop, Refresh, Search, Favorites, Help, and a menu icon.
- Page Content:**
  - Top Navigation:** LaraPus, Buku, Member, Peminjaman, Admin Larapus | Logout
  - Main Content:** Dashboard Admin
  - Message:** Selamat datang di Menu Administrasi Larapus. Silahkan pilih menu administrasi yang diinginkan.



Untuk menampilkan buku yang sedang dipinjam, kita perlu membuat query untuk mengambil semua buku yang sedang dipinjam user yang sedang login di method dashboard :

app/controllers/HomeController.php

```
1  ....
2  public function dashboard() {
3      ....
4      // is regular user
5      if ($user->inGroup($regular)) {
6          $this->layout->content = View::make('dashboard.regular')
7              ->withTitle('Dashboard')
8              ->withBooks($user->books()->wherePivot('returned', 0)->get());
9      }
10 }
11 ....
```

Kita juga perlu mengubah `regular.index.php` menjadi :

---

app/views/dashboard/regular.blade.php

---

```
1   ....
2   @section('content')
3       Selamat datang di Larapus.
4       <table class="uk-table">
5           <tbody>
6               <tr>
7                   <td class="uk-text-muted">Buku dipinjam</td>
8                   <td>
9                       @if ($books->count() == 0)
10                           Tidak ada buku dipinjam
11                       @endif
12                       <ul>
13                           @foreach ($books as $book)
14                               <li>{{ $book->title }} <a href="{{ route('books.return', ['books' \>$book->id]) }}>Kembalikan</a></li>
15                           @endforeach
16                       </ul>
17                   </td>
18               </tr>
19           </tbody>
20       </table>
21   @stop
```

---

Pada view ini, jika User tidak sedang meminjam buku maka akan tampil tulisan **Tidak ada buku dipinjam**. Jika User sedang meminjam buku, maka akan tampil buku yang sedang dipinjam dan tombol untuk mengembalikan buku.

The screenshot shows a web browser window for 'larapus.site/dashboard'. The page title is 'Dashboard | Laravel Perpu'. The top navigation bar includes links for 'Dashboard', 'Buku', and 'Profil', along with a user profile for 'Shidqi Abdullah Mubarak' and a 'Logout' button. Below the navigation, the word 'Dashboard' is prominently displayed in large font. A message 'Selamat datang di Larapus.' is shown. Under the message, there's a section for 'Buku dipinjam' which lists a single item: 'Kupinang Engkau dengan Hamdalah' with a 'Kembalikan' button. A horizontal line separates this from a section titled 'Buku yang sedang dipinjam'.

This screenshot shows the same browser window after a book has been returned. A green notification bar at the top states 'Anda telah mengembalikan Kupinang Engkau dengan Hamdalah.' with a close button. The rest of the page structure is identical to the first screenshot, including the navigation bar, the 'Dashboard' heading, and the 'Buku dipinjam' section which now shows 'Tidak ada buku dipinjam'.

## Dashboard

Pada tahap ini User telah mampu meminjam buku dan mengembalikannya, namun ada yang masih belum berjalan yaitu link navigasi. Seperti sudah diketahui sebelumnya, link navigasi pada views yang telah ada sebelumnya dibuat secara hardcoded di tiap views. Ada dua masalah yang harus kita hadapi, pertama membuat link navigasi yang selalu aktif di tiap view. Kedua, membuat tanda *aktif* jika link URL menunjukkan link tersebut, hal ini bisa kita lakukan dengan menambahkan class `uk-active` pada element list navigasi.

Banyak cara yang bisa kita lakukan untuk menyelesaikan masalah ini. Tapi, untuk membuat pembahasan tidak terlalu meluas, untuk menyelesaikan masalah ini, kita akan memecahnya menjadi 3 tahap. Pertama, membuat subview untuk menampung URL. Kedua, membuat kondisi untuk menampilkan navigasi yang sesuai pada user yang sedang login. Ketiga, membuat Macro untuk menampilkan class `uk-active` pada link dengan URL yang sedang aktif.

`app/views/dashboard/navigations/regular.blade.php`

---

```

1 <li><a href="#">Dashboard</a></li>
2 <li><a href="#">Buku</a></li>
3 <li><a href="#">Profil</a></li>
```

---

`app/views/dashboard/navigations/admin.blade.php`

---

```

1 <li><a href="#">Buku</a></li>
2 <li><a href="#">Member</a></li>
3 <li><a href="#">Peminjaman</a></li>
```

---

Selanjutnya, ubah `@yield('nav')` di `layout master.blade.php` menjadi :

```
{title="app/views/layouts/master.blade.php", lang="html", line-numbers=on} ~~~~~ @if(Sentry::getUser()->hasPermission('regular')) @include('dashboard.navigations.regular') @endif
```

```
@if(Sentry::getUser()->hasPermission('admin')) @include('dashboard.navigations.admin') @endif ~~~~~
```

Pada bagian ini, saya menggunakan fitur `permissions` dari Sentry untuk mengecek permissions yang dimiliki oleh User. Secara singkat, field `permissions` ini terdapat pada table `users` dan `groups`. Dan seorang user yang terdapat di sebuah grup, dengan sendirinya ia memiliki permissions yang dimiliki grup tersebut. Contoh, jika Anda cek table `groups` untuk grup `regular` akan terdapat permissions dengan isian `{"regular":1}` ini menandakan setiap member grup `regular` memiliki permissions `regular`. Untuk lebih memahami penggunaan permissions di Sentry, silahkan merujuk ke [dokumentasi resmi](#)<sup>97</sup>.

Sekarang cobalah login dengan user regular dan admin, pastikan navigasi yang muncul sudah sesuai.

---

<sup>97</sup><https://cartalyst.com/manual/sentry>

**Dashboard**

Selamat datang di Menu Administrasi Larapus. Silahkan pilih menu administrasi yang diinginkan.

**Navigasi Admin**

**Dashboard**

Selamat datang di Larapus.

Buku dipinjam      Tidak ada buku dipinjam

**Navigasi Regular**

Jangan lupa untuk menghapus `isian@section('nav')` pada views yang memilikinya (`app/views/dashboard/regular.blade.php` dan `app/views/dashboard/admin.blade.php`).

Agar navigasi ini aktif untuk URL yang sedang kita kunjungi, mari kita buat Macro dengan nama `smartNav` :

**app/helpers/frontend.php**


---

```

1 ....
2 HTML::macro('smartNav', function($url, $title) {
3     $class = '';
4     if ($url == Request::url()) {
5         $class = 'uk-active';
6     }
7     return "<li class=\"$class\"><a href=\"$url\">$title</a></li>";
8 });

```

---

Macro ini akan menghasilkan sebuah list dengan link sesuai dengan url yang diberikan. Selain itu, macro ini juga akan memberikan class `uk-active` pada list jika url yang menjadi parameter sesuai dengan URL yang sedang aktif. Untuk menghasilkan url kita akan menggunakan named routes. Oleh karena itu, ubahlah route ke dashboard menjadi :

**app/route.php**


---

```

1 ....
2 Route::get('dashboard', array('as'=>'dashboard', 'uses'=>'HomeController@dashboard'));
3 ....

```

---

Selanjutnya, ubah link navigasi pada masing-masing file menjadi :

**app/views/dashboard/navigations/regular.blade.php**


---

```

1 {{ HTML::smartNav(route('dashboard'), 'Dashboard') }}
2 {{ HTML::smartNav(route('member.books'), 'Buku') }}
3 <li><a href="#">Profil</a></li>

```

---

**app/views/dashboard/navigations/admin.blade.php**


---

```

1 {{ HTML::smartNav(route('dashboard'), 'Dashboard') }}
2 {{ HTML::smartNav(route('admin.books.index'), 'Buku') }}
3 {{ HTML::smartNav(route('admin.authors.index'), 'Penulis') }}
4 {{ HTML::smartNav('#', 'Member') }}
5 {{ HTML::smartNav('#', 'Peminjaman') }}

```

---

Sekarang coba login, dan cek semua navigasi telah berjalan sebagaimana mestinya.

Judul	Jumlah	Penulis	
Kupinang Engkau dengan Hamdalah	3	Mohammad Fauzil Adhim	<a href="#">edit</a> <a href="#">delete</a>
Jalan Cinta Para Pejuang	2	Salim A. Fillah	<a href="#">edit</a> <a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>

Showing 1 to 4 of 4 entries

First Previous **1** Next Last

### Navigasi telah aktif

Agar tampilan dashboard User lebih berguna, mari kita tambahkan login terakhir User tambahkan baris ini di method dashboard :

app/controllers/HomeController.php

```

1 ....
2 // is regular user
3 if ($user->inGroup($regular)) {
4     $this->layout->content = View::make('dashboard.regular')
5         ->withTitle('Dashboard')
6         ->withBooks($user->books()->wherePivot('returned', 0)->get())
7         ->withLastlogin($user->last_login->diffForHumans());
8 }
9 ....

```

Pada fitur ini kita menggunakan field `last_login` di model User. Setiap field yang berupa timestamp, oleh

Laravel akan berubah menjadi object dari Carbon<sup>98</sup>, dan kita bisa menggunakan berbagai fitur menarik dari Carbon. Di contoh ini saya menggunakan `diffForHumans` supaya waktu yang tampil berupa *5 minutes ago*, *a week ago*, dll.

Tampilkan `last_login` ini di dashboard regular user :

app/views/dashboard/regular.blade.php

```
1 ....
2 @section('content')
3     Selamat datang di Larapus.
4     <table class="uk-table">
5         <tbody>
6             <tr>
7                 <td class="uk-text-muted">Login Terakhir</td>
8                 <td>{{ $lastlogin }}</td>
9             </tr>
10            <tr>
11                <td class="uk-text-muted">Buku dipinjam</td>
12                <td>
13                    @if ($books->count() == 0)
14                        Tidak ada buku dipinjam
15                    @endif
16                    <ul>
17                        @foreach ($books as $book)
18                            <li>{{ $book->title }} <a href="{{ route('books.return', ['books'=<
19 >$book->id]) }}>Kembalikan</a></li>
20                    @endforeach
21                </ul>
22            </td>
23        </tr>
24    </tbody>
25 </table>
26 @stop
27 ....
```

Cek kembali halaman login, pastikan detail login terakhir muncul.

<sup>98</sup><https://github.com/briannesbitt/Carbon>

The screenshot shows a web browser window with the title 'Dashboard | Laravel Perpu'. The URL in the address bar is 'larapus.site/dashboard'. The page header includes 'LaraPus' (active), 'Dashboard', 'Buku', and 'Profil'. On the right, it shows 'Shidqi Abdullah Mubarak | Logout'. Below the header, the word 'Dashboard' is prominently displayed in large font. A message 'Selamat datang di Larapus.' follows. Two data rows are listed: 'Login Terakhir' (Last login) at '7 minutes ago' and 'Buku dipinjam' (Books borrowed) with the status 'Tidak ada buku dipinjam' (No books borrowed). At the bottom of the page, a success message 'Berhasil membuat fitur login terakhir' (Successfully created the last login feature) is displayed.

Catatan : Masih ada fitur yang perlu dikembangkan yaitu validasi ketika User mengembalikan buku. Validasi ini harus mengecek apakah buku yang dikembalikan oleh User memang sedang dipinjam atau tidak. Saya serahkan pengembangan fitur ini kepada pembaca.

## Stok Buku

Pada table buku yang telah kita buat, kita hanya menyimpan jumlah buku yang kita miliki dalam field `amount`. Sementara, jumlah buku yang belum dipinjam belum tercatat. Untuk develop fitur ini, kita akan menggunakan [custom attributes](#)<sup>99</sup> agar dapat mengakses stok dari buku dengan syntax `$book->stock`.

Stok ini diambil dari query ke table `book_user` dengan filter `book_id` dan `returned` berisi nilai `0`.

Ubah model Book, tambahkan baris ini :

<sup>99</sup><http://laravel.com/docs/eloquent#accessors-and-mutators>

app/models/Book.php

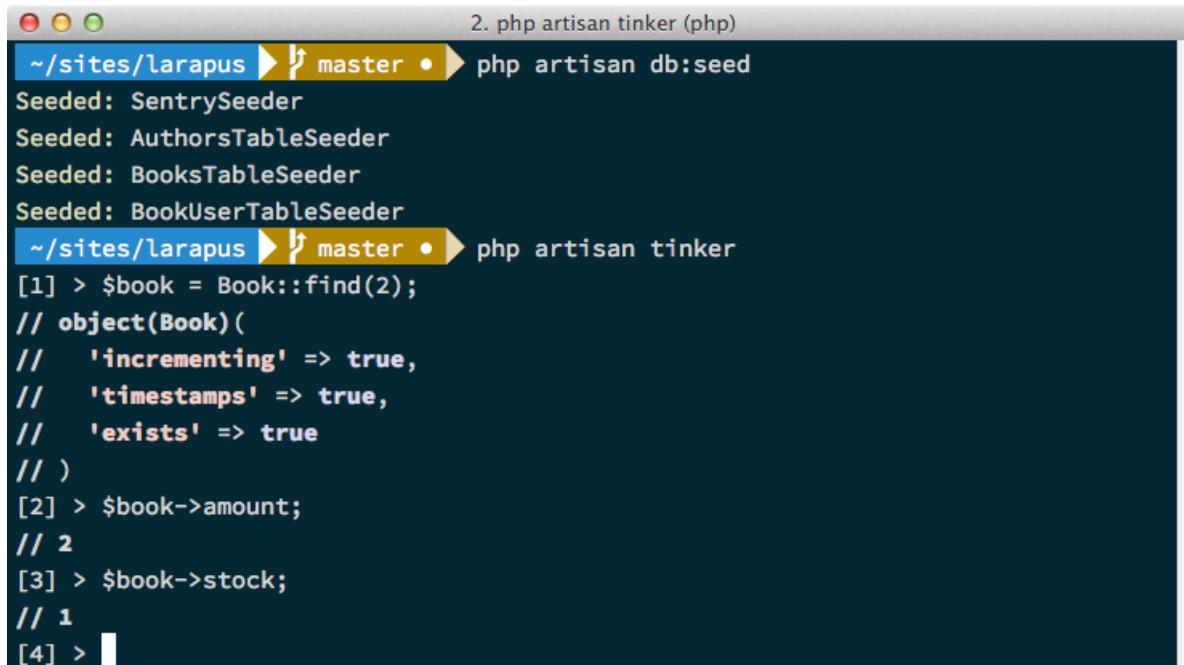
```

1 ...
2 protected $appends = ['stock'];
3
4 public function getStockAttribute()
5 {
6     $borrowed = DB::table('book_user')
7         ->where('book_id', $this->id)
8         ->where('returned', 0)
9         ->count();
10    $stock = $this->amount - $borrowed;
11    return $stock;
12 }
13 ...

```

Pada syntax ini kita menambahkan atribut stock pada model Book dengan isian berupa nilai amount dari buku dikurangi hasil count record dari table book\_user dengan filter book\_id berisi id buku tersebut dan returned yang berisi 0.

Mari kita gunakan tinker untuk mengecek fitur ini :



The screenshot shows a terminal window with the title "2. php artisan tinker (php)". It displays the following output:

```

~/sites/larapus ➔ master • ➔ php artisan db:seed
Seeded: SentrySeeder
Seeded: AuthorsTableSeeder
Seeded: BooksTableSeeder
Seeded: BookUserTableSeeder
~/sites/larapus ➔ master • ➔ php artisan tinker
[1] > $book = Book::find(2);
// object(Book)
//   'incrementing' => true,
//   'timestamps' => true,
//   'exists' => true
// )
[2] > $book->amount;
// 2
[3] > $book->stock;
// 1
[4] >

```

Berhasil membuat attribute stock

Terlihat disini kita me-reset kondisi database sehingga kita tahu buku dengan id 2 sedang dipinjam, dimana buku ini memiliki amount sebanyak 2 sehingga stock nya pasti 1.

## Menampilkan stok buku

Untuk menampilkan stock buku yang telah kita buat pada datatable, ikuti langkah berikut.

1. Ubahlah method borrowDatatable dengan menambahkan kolom stock pada output datatable yang dihasilkan :

app/controllers/BooksController.php

```

1   ....
2   public function borrowDatatable()
3   {
4       // eager load author untuk menghemat query sql
5       return Datatable::collection(Book::with('author')->get())
6           ->showColumns('id', 'title', 'amount', 'stock')
7           // menggunakan closure untuk menampilkan nama penulis dari relasi
8           ->addColumn('author', function ($model) {
9               return $model->author->name;
10          })
11         // menggunakan helper untuk membuat link
12         ->addColumn('', function ($model) {
13             return link_to_route('books.borrow', 'Pinjam', ['book'=>$model->id]);
14          })
15         ->searchColumns('title', 'amount', 'author')
16         ->orderColumns('title', 'amount', 'author')
17         ->make();
18     }
19   ....

```

2. Ubahlah \_borrowdatatable.blade.php agar menampilkan field stock yang dihasilkan :

app/views/books/\_borrowdatatable.blade.php

```

1 {{ Datatable::table()
2     ->addColumn('id', 'title', 'author', 'amount', 'stock', '')
3     ->setOptions('aoColumnDefs', array(
4         array(
5             'bVisible' => false,
6             'aTargets' => [0]),
7         array(
8             'sTitle' => 'Judul',
9             'aTargets' => [1]),
10        array(
11            'sTitle' => 'Jumlah',
12            'aTargets' => [2]),
13        array(
14            'sTitle' => 'Stok',
15            'aTargets' => [3]),

```

```

16     array(
17         'sTitle' => 'Penulis',
18         'aTargets' => [4]),
19     array(
20         'bSortable' => false,
21         'aTargets' => [5])
22     ))
23 ->setOptions('bProcessing', true)
24 ->setUrl(route('datatable.books.borrow'))
25 ->render('datatable.ukikit') }}
```

3. Lakukan langkah yang sama untuk menampilkan kolom stock di halaman admin.
4. Cek kembali halaman buku, pastikan sudah menampilkan kolom stok.

Judul	Jumlah	Stok	Penulis	
Kupinang Engkau dengan Hamdalah	3	2	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	4	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">Pinjam</a>

Show 10 entries Search:

Showing 1 to 4 of 4 entries

First Previous 1 Next Last

Berhasil menambahkan kolom stock di halaman member

Judul	Jumlah	Stok	Penulis	
Kupinang Engkau dengan Hamdalah	3	2	Mohammad Fauzil Adhim	<a href="#">edit</a> <a href="#">delete</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">edit</a> <a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	4	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>

Showing 1 to 4 of 4 entries

First Previous **1** Next Last

Berhasil menambahkan kolom stock di halaman admin

Setelah kita berhasil membuat fitur stok, ada validasi tambahan yang harus dilakukan. Mari kita bahas satu-persatu.

### Validasi sebelum meminjam buku berdasarkan ketersediaan stok

Ketika user meminjam buku, kita harus mengecek apakah buku tersebut masih tersedia dengan mengecek nilai stock apakah tidak 0.

1. Ubah fungsi borrow menjadi :

**app/models/Book.php**


---

```

1   ....
2   public function borrow()
3   {
4       $user = Sentry::getUser();
5
6       // cek apakah buku ini sedang dipinjam oleh user
7       if($user->books()->wherePivot('book_id',$this->id)->wherePivot('returned', 0)->count(\
8 ) > 0 ) {
9           throw new BookAlreadyBorrowedException("Buku $this->title sedang Anda pinjam.");
10      }
11
12      if($this->stock == 0) {
13          throw new BookOutOfStockException("Buku $this->title sudah tidak tersedia.");
14      }
15
16      return $this->users()->attach($user);
17  }
18  ....

```

---

Pada fungsi ini kita membuat validasi baru untuk mengecek apakah stok buku 0, jika iya, maka kita akan melempar exception BookOutOfStockException.

Tentunya, kita harus membuat exception ini. Untuk memudahkan handle try catch di BooksController, kita akan membuat exception yang baru yaitu BookException. Exception tentang buku yang lain akan meng-*extends* BookException, sehingga kita hanya perlu menangkap (catch) exception ini di BooksController.

Buatlah exception BookException :

**app/exceptions/BookException.php**


---

```

1  <?php
2
3  class BookException extends Exception {}

```

---

Ubahlah BookAlreadyBorrowedException menjadi :

**app/exceptions/BookException.php**


---

```

1  <?php
2
3  class BookAlreadyBorrowedException extends BookException { }

```

---

Buat juga BookOutOfStockException :

app/exceptions/BookOutOfStockException.php

---

```
1 <?php
2
3 class BookOutOfStockException extends BookException { }
```

---

2. Kita perlu menangkap exception BookException di BooksController :

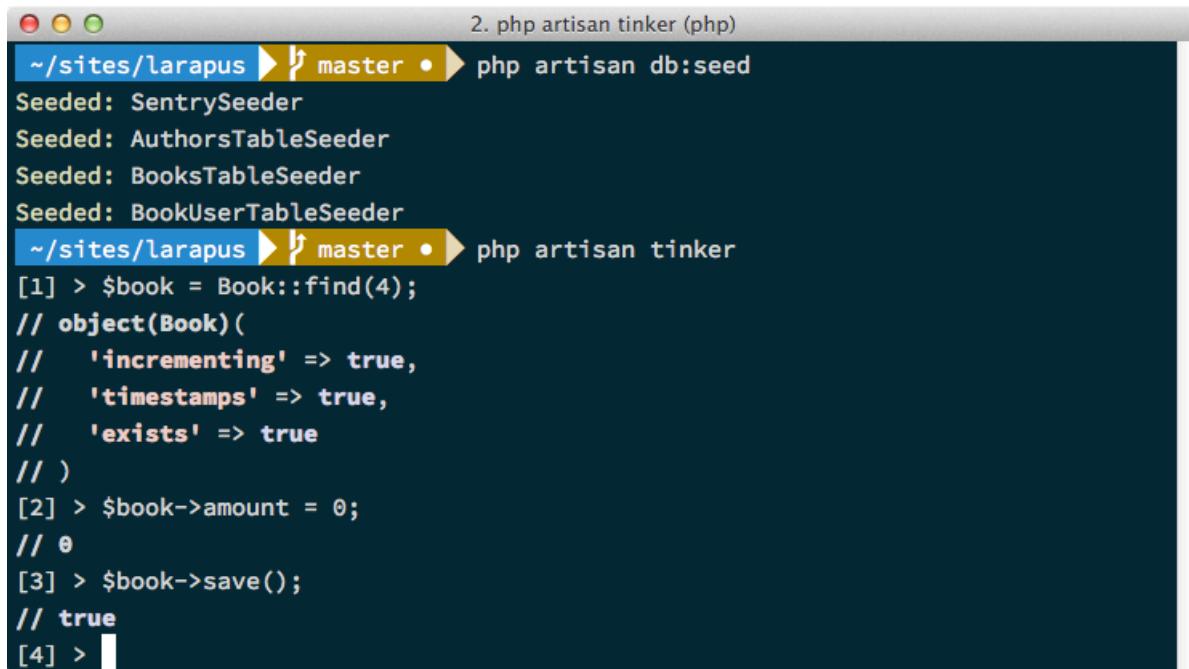
app/controllers/BooksController.php

---

```
1 ....
2 public function borrow($id)
3 {
4     $book = Book::findOrFail($id);
5
6     try {
7         $book->borrow();
8     } catch (BookException $e) {
9         return Redirect::back()->with('errorMessage', $e->getMessage());
10    }
11
12    return Redirect::back()->with("successMessage", "Anda telah meminjam $book->title");
13 }
14 ....
```

---

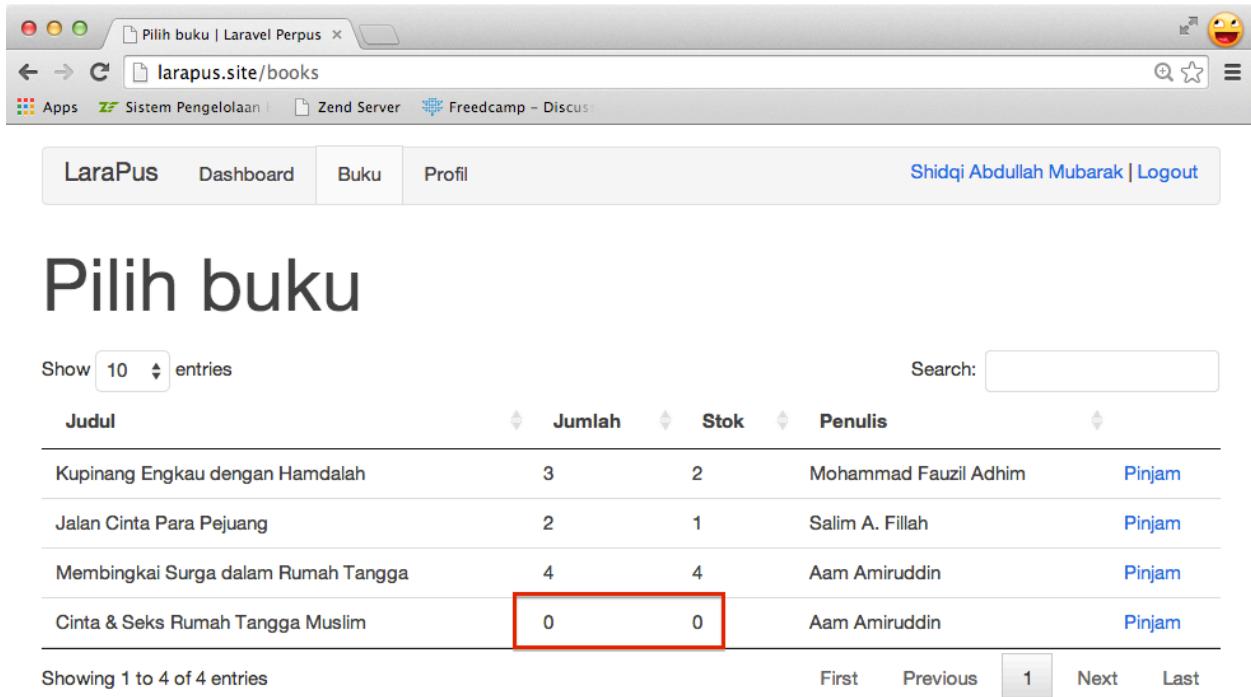
3. Sekarang kita perlu mengecek validasi ini. Mari kita buat sebuah buku memiliki stok 0 dengan mengisi amount nya menjadi 0 menggunakan tinker.



The screenshot shows a terminal window titled "2. php artisan tinker (php)". It displays the output of running `php artisan db:seed`, which seeds several database tables: SentrySeeder, AuthorsTableSeeder, BooksTableSeeder, and BookUserTableSeeder. Below this, the command `php artisan tinker` is run, leading to the following session:

```
[1] > $book = Book::find(4);
// object(Book){
//   'incrementing' => true,
//   'timestamps' => true,
//   'exists' => true
// }
[2] > $book->amount = 0;
// 0
[3] > $book->save();
// true
[4] >
```

Mengeset stok menjadi 0



Pilih buku

Show 10 entries Search:

Judul	Jumlah	Stok	Penulis	
Kupinang Engkau dengan Hamdalah	3	2	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	4	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	0	0	Aam Amiruddin	<a href="#">Pinjam</a>

Showing 1 to 4 of 4 entries First Previous  Next Last

Stok buku sudah 0

Sekarang pinjamlah buku yang memiliki stok 0, pastikan pesan validasi muncul.

Buku Cinta & Seks Rumah Tangga Muslim sudah tidak tersedia.

Judul	Jumlah	Stok	Penulis	Pinjam
Kupinang Engkau dengan Hamdalah	3	2	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	4	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	0	0	Aam Amiruddin	<a href="#">Pinjam</a>

Showing 1 to 4 of 4 entries

Berhasil membuat validasi ketika meminjam buku

## Validasi ketika mengubah jumlah buku

Ketika seorang Admin mengubah jumlah buku, kita perlu mengecek apakah nilai itu tidak boleh kurang dari jumlah buku yang sedang dipinjam. Untuk itu kita akan menggunakan Model Observer untuk mengecek ketika Admin mengubah (*update*) jumlah buku.

1. Tambahkan fungsi boot di model Book dengan isi :

app/models/Book.php

```

1  ....
2  public static function boot()
3  {
4      parent::boot();
5
6      self::updating(function($book)
7      {
8          $borrowed = DB::table('book_user')

```

```

9          ->where('book_id', $book->id)
10         ->where('returned', 0)
11         ->count();
12     if ($book->amount < $borrowed) {
13         Session::flash('errorMessage', "Jumlah buku $book->title harus >= $borrowed");
14         return false;
15     }
16 });
17 }
18 ....

```

---

Terlihat disini kita mengecek apakah nilai `amount` yang baru tidak boleh kurang dari jumlah buku yang sedang dipinjam. Jika kurang, maka proses update buku akan dibatalkan.

2. Tambahkan pengecekan ketika meng-*update* buku di BooksController :

app/models/Book.php

---

```

1 ....
2 public function update($id)
3 {
4     $book = Book::findOrFail($id);
5
6     $validator = Validator::make($data = Input::all(), $book->updateRules());
7
8     if ($validator->fails())
9     {
10         return Redirect::back()->withErrors($validator)->withInput();
11     }
12
13     if (!$book->update($data)) {
14         return Redirect::back();
15     }
16
17     return Redirect::route('admin.books.index')->with("successMessage", "Berhasil menyimpan $book->title.");
18 }
19 ....

```

---

3. Untuk mengeceknya, coba ubah `amount` sebuah buku yang sedang dipinjam menjadi 0. Pastikan pesan validasinya muncul.

The screenshot shows a web browser window with the URL [larapus.site/admin/books/1/edit](http://larapus.site/admin/books/1/edit). The page title is "Ubah Kupinang Engkau dengan Hamdalah". The navigation bar includes "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", "Admin Larapus | Logout". A red validation message box contains the text "Jumlah buku Kupinang Engkau dengan Hamdalah harus >= 1". Below the message is the breadcrumb "Dashboard / Buku / Ubah Kupinang Engkau dengan Hamdalah". The main content area has fields for "Judul" (with a placeholder "Kupinang Engkau dengar"), "Penulis" (selected "Mohammad Fauzil Adhim"), and "Jumlah" (set to "3"). At the bottom is a blue "Simpan" button.

Berhasil membuat validasi

## Validasi ketika menghapus buku

Validasi terakhir yang harus kita buat adalah ketika Admin menghapus buku. Pada saat itu, kita harus mengecek apakah buku tersebut masih dipinjam, jika iya, batalkan penghapusan buku dan berikan pesan error ke User.

- Untuk fitur ini kita juga akan menggunakan Model Observer pada event `deleting`. Tambahkan baris berikut di method `boot` :

`app/models/Book.php`

```

1  ....
2  public static function boot()
3  {
4      ....
5      self::deleting(function($book)
6      {
7          $borrowed = DB::table('book_user')
8              ->where('book_id', $book->id)
9              ->where('returned', 0)
10             ->count();
11         if ($borrowed > 0) {
12             Session::flash('errorMessage', "Buku $book->title masih dipinjam.");

```

```

13         return false;
14     }
15 });
16 }
17 ....

```

2. Ubah method destroy di BooksController :

app/models/Book.php

```

1 public function destroy($id)
2 {
3     if (!Book::destroy($id)) {
4         return Redirect::back();
5     }
6
7     return Redirect::route('admin.books.index')->with('successMessage', 'Buku berhasil di\
8 hapus.');
9 }

```

3. Terakhir, coba hapus sebuah buku yang sedang dipinjam. Pastikan pesan validasi muncul.

Judul	Jumlah	Stok	Penulis	
Kupinang Engkau dengan Hamdalih	3	2	Mohammad Fauzil Adhim	<a href="#">edit</a> <a href="#">delete</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">edit</a> <a href="#">delete</a>
Membingkai Surga dalam Rumah Tangga	4	4	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">edit</a> <a href="#">delete</a>

Berhasil membuat validasi ketika menghapus buku

Catatan : Jika Anda perhatikan, nampaknya cara validasi seperti ini kurang elegan. Saya pun merasakan demikian, seharusnya validasi disimpan di atribut `$rules` pada Model yang bersangkutan. Jika Anda tertarik mempelajari lebih jauh, saya sarankan untuk mempelajari pembuatan [Custom Validator<sup>100</sup>](#).

## Ringkasan

Di Hari 5 ini, banyak sekali yang kita pelajari diantaranya :

- Model User custom untuk Sentry
- Pivot (many-to-many) pada Eloquent
- Artisan Tinker, REPL nya Laravel
- Named Routes
- Menggunakan Model Observer
- Penggunaan DB Statement
- Penggunaan Query Builder
- Penggunaan Custom Attributes pada Eloquent

Pada hari 6, kita akan membuat fitur-fitur untuk manajemen user. Semangat! :)

---

<sup>100</sup><http://laravel.com/docs/validation#custom-validation-rules>

# Hari 6 : User Management

Aplikasi Larapus yang sedang kita bangun kini telah hampir lengkap. Pada hari ini kita akan membuat beberapa fitur yang berhubungan dengan manajemen user yang akan mengizinkan guest untuk mendaftar menjadi User, halaman profil, dan CRUD Member. Dalam proses pembuatan beberapa fitur ini, kita akan mempelajari beberapa hal diantaranya menggunakan validasi Eloquent pada Sentry dan penggunaan reCaptcha dari Google untuk mengurangi spam di form sign up.

## Register

Fitur Register merupakan fitur standar sebuah web. Fitur ini mengizinkan user untuk mendaftar menjadi member. Alur dari fitur ini adalah :

1. User mengisi Form dengan reCaptcha.
2. Setelah form dilengkapi, email aktivasi dikirim ke User.
3. User mengunjungi link aktivasi di email.
4. Setelah diaktifasi, user bisa login.

## Konfigurasi reCaptcha

Jika Anda belum mengerti Captcha, teknik ini adalah teknik untuk mengurangi spam di form yang muncul di web. Cara kerjanya adalah dengan menampilkan tulisan acak kemudian user diminta memasukan tulisan acak tersebut. Captcha salain digunakan untuk halaman register, biasanya digunakan juga di halaman komentar untuk mengurangi komentar spam. Sebenarnya ada berbagai implementasi untuk Captcha, tapi dalam buku ini saya akan menggunakan teknik tulisan acak. Untuk memudahkan kita akan menggunakan reCaptcha dari Google.

Untuk itu, silahkan buat dulu akun reCaptcha di <http://www.google.com/recaptcha><sup>101</sup>. Setelah, masuk ke halaman Admin pilih “Sign Up Now!”. Kemudian masukkan nama domain (virtualhost) yang telah Anda buat. Jika Anda akan menggunakan reCaptcha yang sama di semua domain, centang **Enable this key on all domains (global key)**. Tekan tombol **Create**.

---

<sup>101</sup><http://www.google.com/recaptcha>

reCAPTCHA: Stop Spam, Re

https://www.google.com/recaptcha/admin#createsite

Apps Sistem Pengelolaan Zend Server Freedcamp - Discuss

Google

HOME

Get reCAPTCHA

My account

Protect your email

Resources: docs & plugins

**Domain**

http://

e.g. recaptcha.net, example.com

Enable this key on all domains (global key)

Enter one domain address per line and we'll create sites for all of them.

You may want to use the export as CSV functionality to get a list of all your keys once they are created.

**Tips**

By default, your reCAPTCHA key is restricted to the specified domain, and any subdomains for additional security. A key for foo.com works on test.foo.com.

If you wish to use your key across a large number of domains (e.g., if you are a hosting provider, OEM, etc.), select the global key option. You may want to use a descriptive domain name such as "global-key.mycompany.com".

If you own multiple domain names (foocars.com and footrucks.com), you can sign up for multiple keys, or use a global key.

**CREATE**

Google | About Google | Privacy & Terms  
© 2014 Google, all rights reserved.

### Membuat reCaptcha untuk VirtualHost

Pada halaman selanjutnya klik nama domain yang telah kita daftarkan.

The screenshot shows a web browser window with the following details:

- Title Bar:** reCAPTCHA: Stop Spam, Re
- Address Bar:** https://www.google.com/recaptcha/admin#list
- Toolbar:** Apps, Sistem Pengelolaan, Zend Server, Freedcamp - Discuss
- Header:** Google, laravel.perpustakaan@gmail.com
- Main Content:**
  - Sites you Administer:** HOME, Get reCAPTCHA, My account (highlighted with a red vertical bar), Protect your email, Resources: docs & plugins.
  - Add Site Button:** + Add a New Site
  - Text:** If you have many sites, you may want to export the keys as a CSV file.
  - List of Sites:** larapus.site (highlighted with a red box)
- Footer:** Google | About Google | Privacy & Terms  
© 2014 Google, all rights reserved.

### Memilih domain yang telah didaftarkan

Pada halaman terakhir, catat **Private Keys** dan **Public Keys** yang muncul. Dua keys ini akan kita gunakan pada tahap selanjutnya

The screenshot shows a Google Chrome window with the URL <https://www.google.com/recaptcha/admin#site/317993955>. The page is titled 'reCAPTCHA: Stop Spam, Re...' and shows settings for 'larapus.site'. It includes sections for 'HOME', 'Get reCAPTCHA', 'My account', 'Protect your email', and 'Resources: docs & plugins'. The 'Public Key' and 'Private Key' fields are highlighted with red boxes. Below these fields, there are instructions: 'Use this in the JavaScript code that is served to your users' for the Public Key and 'Use this when communicating between your server and our server. Be sure to keep it a secret.' for the Private Key.

### Mendapatkan private dan public keys reCaptcha

Saya telah mencari implementasi reCaptcha untuk Laravel di [Packagist<sup>102</sup>](#), dan menemukan [greggilbert/recaptcha<sup>103</sup>](#). Package ini yang akan kita pakai, oleh karena itu silahkan tambahkan di `composer.json`.

#### app/composer.json

```

1 ....
2 "require": {
3     ...
4     "greggilbert/recaptcha": "1.0.7"
5 }
6 ....

```

Install dengan perintah `composer update`.

Selanjutnya, kita perlu mengkonfigurasi package ini. Tambahkan '`Greggilbert\Recaptcha\RecaptchaServiceProvider`' di array provider di `app/config/app.php` :

<sup>102</sup><https://packagist.org/search/?q=laravel%20recaptcha>

<sup>103</sup><https://github.com/greggilbert/recaptcha>

**app/composer.json**


---

```

1 ...
2 'providers' => array(
3     ...
4     'Giggilbert\Recaptcha\RecaptchaServiceProvider'
5 ),
6 ...

```

---

Tambahkan *private* dan *public keys* reCaptcha yang telah Anda dapatkan di langkah sebelumnya di app/config/packages/giggilbert/recaptcha/config.php. Selain itu, tambahkan juga array berikut untuk mengeset theme dari reCaptcha dan placeholder untuk input :

**app/config/packages/giggilbert/recaptcha/config.php**


---

```

1 'options' => array(
2     'theme'=>'clean',
3     'custom_translations'=>array(
4         'instructions_visual'=>'Ketikan teks diatas'
5     )
6 ),

```

---

Tambahkan baris berikut di app/lang/en/validation.php :

**app/lang/en/validation.php**


---

```

1 ...
2 "recaptcha" => 'The :attribute field is not correct.',
3 ...

```

---

Setelah semua konfigurasi Anda lakukan, untuk menampilkan reCaptcha di form gunakan syntax Form::captcha(). Untuk memvalidasi captcha ini kita perlu menambahkan rule 'recaptcha\_response\_field' => 'required|recaptcha' di model yang akan kita bahas di tahap selanjutnya.

## Konfigurasi Email

Ketika User melakukan registrasi, ia tidak akan langsung aktif. Agar user bisa aktif, kita akan menggunakan semacam *activation code* yang dikirim via email. Setelah Users klik link aktivasi, barulah dia bisa login dengan data user yang dimasukkan pada saat mendaftar. Sengaja saya menggunakan fitur ini untuk menunjukan kepada pembaca bagaimana saya menggunakan fitur email dari Laravel dan fitur aktivasi User dari Sentry. Tentunya, di dunia nyata, keputusan menggunakan fitur aktivasi user ini bergantung dengan kebutuhan aplikasi yang dibangun.

Karena, fitur aktivasi ini akan menggunakan email, saya akan menggunakan server SMTP gmail untuk mengirimkan email. Tentunya, Anda dapat menggunakan server SMTP sendiri untuk melakukannya. Ubahlah file app/config/mail.php (sesuaikan isian dengan alamat email Anda) :

---

app/config/mail.php

---

```

1 <?php
2
3 return array(
4     'driver' => 'smtp',
5     'host' => 'smtp.gmail.com',
6     'port' => 587,
7     'from' => array('address' => 'laravel.perpustakaan@gmail.com', 'name' => 'Larapus Admin'),
8 ),
9     'encryption' => 'tls',
10    'username' => 'laravel.perpustakaan@gmail.com',
11    'password' => '#larapus2014',
12    'pretend' => false,
13 );

```

---

Pada opsi terakhir, yaitu pretend jika Anda mengubah menjadi true maka email tidak akan dikirim, tetapi hanya dicatat telah dikirim di app/storage/logs/laravel.log. Ini berguna jika aplikasi masih dalam tahap development.

Untuk memahami bagaimana cara mengirim email, Anda dapat membaca di [dokumentasi resmi](#)<sup>104</sup>. Laravel versi 4.2 telah mendukung penggunaan API dari Mandrill dan Mailgun, fitur ini akan membuat pengiriman email cepat *banget*. Di halaman ini Anda juga akan menemukan cara penggunaan *queue* untuk pengiriman email yang *super cepat*. Saya sarankan Anda untuk mempelajari dua hal itu.

Untuk mengirimkan email, kita membutuhkan view sebagai template email, nantinya kita tinggal melakukan *passing variable* ke template email tersebut. Buatlah template ini di app/views/emails/auth/register.blade.php :

---

app/views/emails/auth/register.blade.php

---

```

1 <!DOCTYPE html>
2 <html lang="en-US">
3     <head>
4         <meta charset="utf-8">
5     </head>
6     <body>
7         <h2>Selamat datang di Larapus!</h2>
8
9         <div>
10            Untuk mengaktifkan akun Anda, silahkan klik link berikut: <br>
11            {{ link_to_route('user.activate', null, ['email'=>$email, 'activationCode'=>$activationCode]) }}
12        </div>
13
14    </body>
15 </html>

```

---

<sup>104</sup><http://laravel.com/docs/mail>

Di view ini, kita menggenerate link ke route user.activate dengan paramater email dan activation code dari User. Format diatas akan membuat routes kira-kira seperti ini `http://larapus.site/activate?email=blabla@gmail.com&act`. Route yang digunakan akan kita buat di tahapan selanjutnya.

Syntax untuk mengirim email di Laravel sangat mudah. Kita dapat menggunakan closure untuk mengirimnya, berikut ini contohnya :

contoh syntax pengiriman email

---

```

1 Mail::send('emails.welcome', $data, function($message)
2 {
3     $message->to('foo@example.com', 'John Smith')->subject('Welcome!');
4 });

```

---

Pada syntax ini, `emails.welcome` adalah view yang digunakan untuk template email. `$data` adalah array  $key \Rightarrow value$  berisi variable yang akan di passing ke view. Misalnya, jika kita di view membutuhkan variable `$title` dan `$code`, maka isi dari `$data` adalah `array('title'=>'Judul keren', 'code'=>'s3cr3t')`.

Sip, sekarang kita siap mengembangkan fitur register user.

1. Buat method `signup` dan routes untuk menampilkan halaman register User di `GuestController` :

app/controllers/GuestController.php

---

```

1 ....
2 /**
3 * Tampilkan halaman signup
4 * @return response
5 */
6 public function signup()
7 {
8     $this->layout->content = View::make('guest.signup');
9 }
10 ....

```

---

app/routes.php

---

```

1 ...
2 Route::get('signup', array('as'=>'home.signup', 'uses'=>'GuestController@signup'));
3 ?>

```

---

2. Buat view untuk form register di `app/views/guest/signup.blade.php` :

**app/views/guest/signup.blade.php**

```
1  @extends('layouts.guest')
2
3  @section('content')
4      <div class="uk-text-center">
5          <div class="uk-vertical-align-middle" style="width: 500px;">
6              @include('layouts.partials.validation')
7              {{ Form::open(array('url' => route('guest.register'), 'class' => 'uk-form uk-form\-
8 -horizontal')) }}
9                  <div class="uk-form-row">
10                     {{ Form::label('first_name', 'Nama Depan', array('class' => 'uk-form-labe\
11 l uk-text-left')) }}
12                     {{ Form::textUk('first_name', 'Nama depan Anda') }}
13                 </div>
14
15                 <div class="uk-form-row">
16                     {{ Form::label('last_name', 'Nama Belakang', array('class' => 'uk-form-la\
17 bel uk-text-left')) }}
18                     {{ Form::textUk('last_name', 'Nama belakang Anda') }}
19                 </div>
20
21                 <div class="uk-form-row">
22                     {{ Form::label('email', 'Email', array('class' => 'uk-form-label uk-text-\
23 left')) }}
24                     {{ Form::textUk('email', 'emailmu@website.com') }}
25                 </div>
26
27                 <div class="uk-form-row">
28                     {{ Form::label('password', 'Password', array('class' => 'uk-form-label uk\-
29 -text-left')) }}
30                     <div class="uk-form-controls">
31                         {{ Form::password('password', array('placeholder'=>'*****')) }}
32                     </div>
33                 </div>
34
35                 <div class="uk-form-row">
36                     {{ Form::label('password_confirmation', 'Konfirmasi Password', array('cla\
37 ss' => 'uk-form-label uk-text-left')) }}
38                     <div class="uk-form-controls">
39                         {{ Form::password('password_confirmation', array('placeholder'=>'***\
40 *****')) }}
41                     </div>
42                 </div>
43
44                 <div class="uk-form-row">
```

```

45         {{ Form::captcha() }}
46     </div>
47
48     <div class="uk-form-row">
49         {{ Form::submit('Daftar', array('class'=>'uk-width-1-1 uk-button uk-button\
50 n-primary uk-button-large')) }}
51     </div>
52     {{ Form::close() }}
53
54     </div>
55 </div>
56 @stop

```

---

Pada form ini kita menampilkan isian untuk nama depan, nama belakang, email, password, konfirmasi password dan captcha. Konfirmasi password merupakan fitur standar ketika kita melakukan registrasi user. Di Laravel, validasi field untuk konfirmasi ini sangat mudah. Anda cukup menambah *rule confirmed* dan membuat input baru dengan id yang sama dengan tambahan *\_confirmation*. Contoh, untuk konfirmasi field password, Anda cukup membuat input *password\_confirmation*.

Form ini mengirim data ke route `guest.register` yang akan kita buat di tahapan ke-4.

3. Dalam pembuatan User ini kita tidak akan menggunakan validasi dari Sentry. Untuk validasi, kita akan menggunakan model validation gaya Laravel. Mari kita setup modelnya :

#### app/models/User.php

---

```

1 ....
2 // Validation rules
3 public static $rules = [
4     'first_name' => 'required',
5     'email' => 'required|email|unique:users,email,:id',
6     'password' => 'confirmed|required|min:5',
7     'recaptcha_response_field' => 'required|recaptcha',
8 ];
9
10 // field yang bisa diisi dengan mass assignment
11 protected $fillable = ['first_name', 'last_name', 'email', 'password'];
12 ....

```

---

4. Untuk menerima form yang telah diisi kita perlu membuat method `register` di `GuestController` dan membuat route `guest.register`.

**app/controllers/GuestController.php**

```
1     ....
2     /**
3      * Register User dan kirim email untuk aktivasi
4      * @return response
5     */
6     public function register()
7     {
8         $validator = Validator::make($data = Input::all(), User::$rules);
9
10        if ($validator->fails())
11        {
12            return Redirect::back()->withErrors($validator)->withInput();
13        }
14
15        // Register User tanpa diaktivasi
16        $user = Sentry::register(array(
17            'email'    => Input::get('email'),
18            'password' => Input::get('password'),
19            'first_name' => Input::get('first_name'),
20            'last_name' => Input::get('last_name')
21        ), false);
22
23        // Cari grup regular
24        $regularGroup = Sentry::findGroupByName('regular');
25
26        // Masukkan user ke grup regular
27        $user->addGroup($regularGroup);
28
29        // Persiapkan activation code untuk dikirim ke email
30        $data = [
31            'email'=>$user->email,
32            // Generate kode aktivasi
33            'activationCode'=>$user->getActivationCode()
34        ];
35
36        // Kirim email aktivasi
37        Mail::send('emails.auth.register', $data, function ($message) use ($user) {
38            $message->to($user->email, $user->first_name . ' ' . $user->last_name)->subject('\
39 Aktivasi Akun Larapus');
40        });
41
42        // Redirect ke halaman login
43        return Redirect::route('guest.login')->with("successMessage", "Silahkan cek email Anda\
44 a ($user->email) untuk melakukan aktivasi akun.");

```

```
45     }
46     ....
```

---

Pada method ini kita memvalidasi data dengan *rules* dari Model. Setelah data divalidasi, barulah user dibuat. Dengan cara ini, Sentry tidak akan melempar *exception* karena data pasti sudah valid. Karena user yang daftar adalah member, kita memasukkannya ke dalam grup regular. Untuk mendapatkan kode aktivasi kita menggunakan method `getActivationCode` dari Sentry. Pada saat mengirimkan email, kita menggunakan `view emails.auth.register` dan melakukan passing variable `$email` dan `$activationCode`. Kita juga menggunakan [scoping<sup>105</sup>](#) pada closure untuk passing variable `$user` dari parent scope ke dalam closure menggunakan `use` supaya dapat mengakses atribut `email`, `first_name` dan `last_name`.

Setelah user email dikirim, kita *redirect* user ke halaman login. Kita perlu merubah `app/routes.php` untuk halaman login menjadi named routes. Kita juga harus menambahkan route untuk method register ini dengan tipe POST :

#### app/routes

---

```
1   ...
2   Route::get('login', array('as'=>'guest.login', 'uses'=>'GuestController@login'));
3   Route::post('register', array('as'=>'guest.register', 'uses'=>'GuestController@register'))\
4 );
5   ...
```

---

Ubah juga link route ke halaman register layout guest :

#### app/views/layouts/guest.blade.php

---

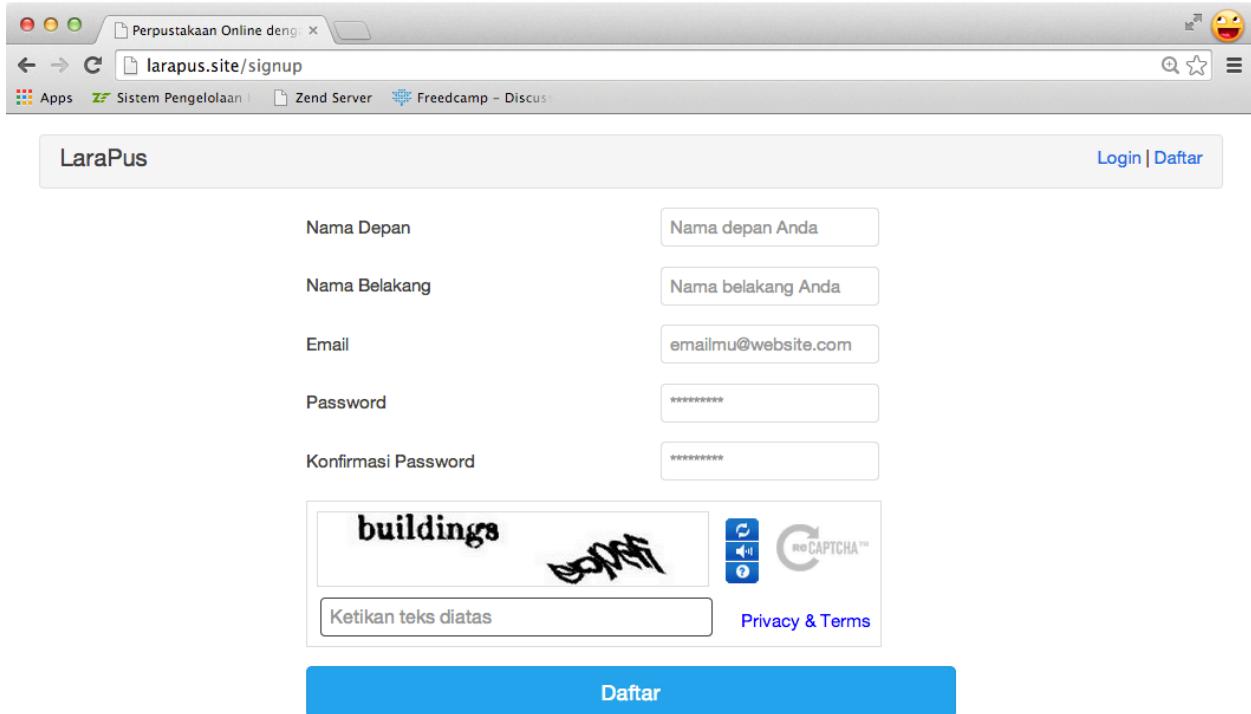
```
1   ....
2   <div class="uk-navbar-flip uk-navbar-content">
3       <a class="" href="{{ URL::to('login') }}">Login</a> |
4       <a class="" href="{{ URL::to('signup') }}">Daftar</a>
5   </div>
6   ....
```

---

Sekarang coba dulu isi form ini dengan data yang salah, pastikan captcha dan validasi muncul.

---

<sup>105</sup><http://www.php.net/manual/en/functions.anonymous.php>



### Membuat form signup

- Untuk aktivasi user, kita perlu membuat method `activate` di `GuestController` :

`app/controllers/GuestController.php`

```

1  /**
2   * Aktivasi seorang user
3   * @param string $activationCode
4   * @return response
5   */
6  public function activate()
7  {
8      $email = Input::get('email');
9      $activationCode = Input::get('activationCode');
10
11     try {
12         $user = Sentry::findUserByLogin($email);
13         $user->attemptActivation($activationCode);
14     } catch (UserAlreadyActivatedException $e) {
15         return Redirect::route('guest.login')->with('errorMessage', $e->getMessage());
16     } catch (UserNotFoundException $e) {
17         return Redirect::route('guest.login')->with('errorMessage', $e->getMessage());
18     }
19 }
```

```
20     return Redirect::route('guest.login')
21         ->with('successMessage', 'Akun Anda berhasil diaktivasi, silahkan login.');
22 }
```

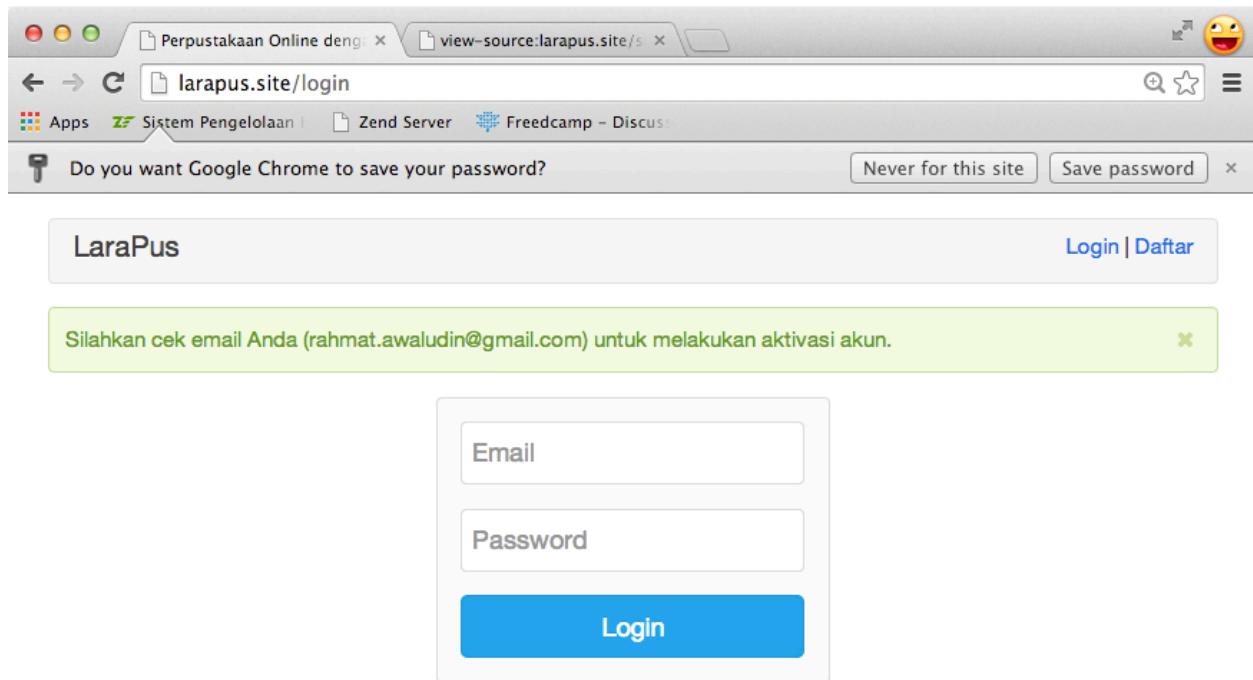
Pada method ini kita mencari user dengan email yang dikirim dan melakukan aktivasi dengan method attemptActivation. Kita pun menangkap UserAlreadyActivatedException jika user sudah diaktivasi dan UserNotFoundException jika user tidak ditemukan.

Tambahkan route untuk method ini :

#### app/routes.php

```
1 ...
2 Route::get('activate', array('as'=>'user.activate', 'uses'=>'GuestController@activate'));
3 ?>
```

Sekarang cobalah membuat register baru dengan email yang valid, pastikan email dikirim dan Anda bisa melakukan aktivasi user tersebut.



Email aktivasi dikirim

The screenshot shows a Gmail inbox with the following details:

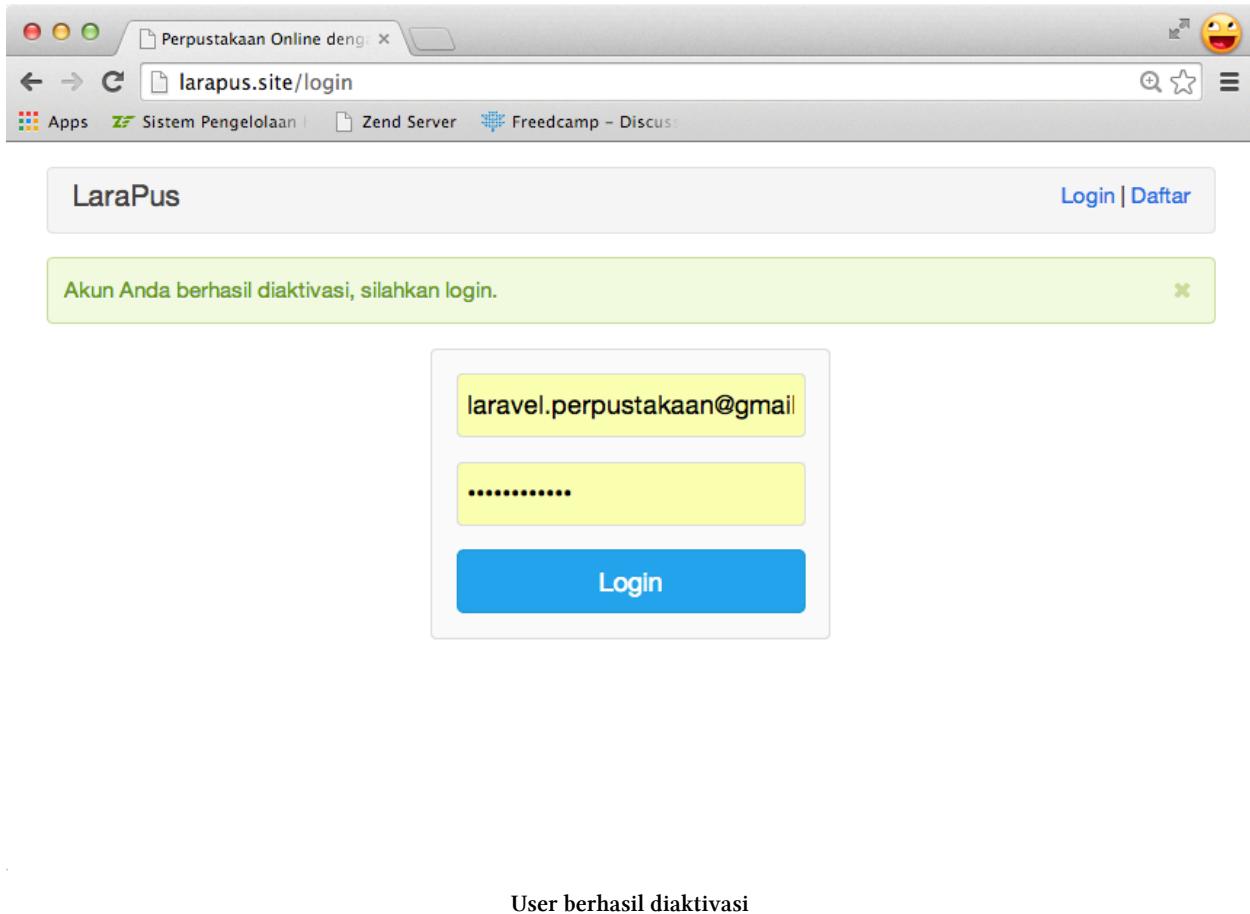
- Title bar:** Aktivasi Akun Larapus - ra
- URL:** https://mail.google.com/mail/u/0/#inbox/1463c65773647f6c
- Toolbar:** Apps, Sistem Pengelolaan, Zend Server, Freedcamp - Discuss
- Header:** Google, Search bar, +Rahmat, Share, Notifications (1)
- Left sidebar:** COMPOSE button, Inbox (1), Sent Mail, Drafts (1), Spam (20), [Airmail] (Done, Memo, To Do), AR EE, Sign in button, Signing in will sign you into Hangouts across Google, Learn more link.
- Message list:** Aktivasi Akun Larapus (Inbox) from Larapus Admin <laravel.perpustakaan@gmail.com> at 4:44 PM (2 minutes ago). The message content is:

**Selamat datang di Larapus!**

Untuk mengaktifkan akun Anda, silahkan klik link berikut:  
<http://larapus.site/activate?email=rahmat.awaludin%40gmail.com&activationCode=YwPDz7YYxXuBCL4kX0i2XyRHYcMN2PPWwZQse4K1K3>

Click here to [Reply](#) or [Forward](#)

Email aktivasi diterima



User berhasil diaktifasi

## Halaman Profil

Halaman profil digunakan untuk menampilkan data User yang sedang login. Penggunaan halaman ini hanya oleh User dengan level member, karenanya kita akan menggunakan filter fitur ini.

1. Buat method method profile di MemberController :

app/controller/MemberController.php

```
1  .....
2  /**
3  * Tampilkan halaman profile
4  * @return response
5  */
6  public function profile()
7  {
8      $user = Sentry::getUser();
9      return View::make('profile.show')->withTitle('Profil')
10         ->withUser($user);
```

```
11     }
12     ....
```

---

Tambahkan route untuk method ini di dalam grup dengan filter before auth :

app/routes.php

---

```
1     ....
2     Route::group(array('before' => 'auth'), function () {
3         Route::get('profile', array('as'=>'member.profile', 'uses'=>'MemberController@profile\
4     '));
5     ....
```

---

Agar method semua method di MemberController hanya dapat diakses user dengan grup regular, mari kita gunakan filter pada method \_\_construct :

app/controller/MemberController.php

---

```
1     ....
2     public function __construct()
3     {
4         // Letakan filter regularUser sebelum memanggil semua method
5         $this->beforeFilter('regularUser');
6     }
7     ....
```

---

2. Buat view untuk tampilan profil di app/views/profile/show.blade.php :

app/views/profile/show.blade.php

---

```
1     @extends('layouts.master')
2
3     @section('title')
4         {{ $title }}
5     @stop
6
7     @section('breadcrumb')
8         <li><a href="/">Dashboard</a></li>
9         <li class="uk-active">{{ $title }}</li>
10    @stop
11
12   @section('content')
13       <table class="uk-table">
14           <tbody>
15               <tr>
16                   <td class="uk-text-muted">Nama</td>
17                   <td>{{ $user->first_name . ' ' . $user->last_name }}</td>
18               </tr>
19               <tr>
```

```

20          <td class="uk-text-muted">Email</td>
21          <td>{{ $user->email }}</td>
22      </tr>
23      <tr>
24          <td class="uk-text-muted">Login Terakhir</td>
25          <td>{{ $user->last_login }}</td>
26      </tr>
27  </tbody>
28 </table>
29 @stop

```

---

3. Tambahkan link untuk profil di app/dashboard/navigations/regular.blade.php :

app/dashboard/navigations/regular.blade.php

---

```

1  {{ HTML::smartNav(route('dashboard'), 'Dashboard')}}
2  {{ HTML::smartNav(route('member.books'), 'Buku')}}
3  {{ HTML::smartNav(route('member.profile'), 'Profil')}}

```

---

4. Terakhir, coba cek navigasi ke halaman profil, pastikan halaman profil muncul dan hanya regular user yang bisa mengaksesnya.

Nama	Shidqi Abdullah Mubarak
Email	shidqi.abdullah@gmail.com
Login Terakhir	2014-05-27 15:57:48

Berhasil membuat halaman profil

## Ubah Profil

Mari kita buat halaman untuk mengedit data user.

1. Buat method editProfile di MemberController :

**app/controllers/MemberController.php**


---

```

1  /**
2   * Tampilkan halaman edit profile
3   * @return response
4   */
5  public function editProfile()
6  {
7      return View::make('profile.edit')->withTitle('Perbaharui Profil')
8          ->withUser(Sentry::getUser());
9  }

```

---

Tambahan route untuk method ini :

**app/routes.php**


---

```

1  ....
2  Route::group(array('before' => 'auth'), function () {
3      Route::get('profile/edit', array('as'=>'member.profile.edit', 'uses'=>'MemberControll\
4 er@editProfile'));
5  ....

```

---

2. Buat view untuk halaman edit di app/views/profiles/edit.blade.php :

**app/views/profiles/edit.blade.php**


---

```

1  @extends('layouts.master')
2
3  @section('title')
4      {{ $title }}
5  @stop
6
7  @section('breadcrumb')
8      <li><a href="/">Dashboard</a></li>
9      <li><a href="{{ route('member.profile') }}">Profil</a></li>
10     <li class="uk-active">{{ $title }}</li>
11  @stop
12
13  @section('content')
14      {{ Form::open(array('url' => route('member.profile.update'), 'method' => 'post', 'cla\
15 ss'=>'uk-form uk-form-horizontal')) }}
16      <div class="uk-form-row">
17          {{ Form::label('first_name', 'Nama Depan', array('class' => 'uk-form-label uk\
18 -text-left')) }}
19          <div class="uk-form-controls">
20              {{ Form::text('first_name', $user->first_name, array('placeholder'=>'Nama\
21 depan Anda')) }}
22          </div>

```

---

```

23         </div>
24
25         <div class="uk-form-row">
26             {{ Form::label('last_name', 'Nama Belakang', array('class' => 'uk-form-label \
27 uk-text-left')) }}
28             <div class="uk-form-controls">
29                 {{ Form::text('last_name', $user->last_name, array('placeholder'=>'Nama b\
30 elakang Anda')) }}
31             </div>
32         </div>
33
34         <div class="uk-form-row">
35             {{ Form::label('email', 'Email', array('class' => 'uk-form-label uk-text-left\
36 ')) }}
37             <div class="uk-form-controls">
38                 {{ Form::text('email', $user->email, array('placeholder'=>'emailmu@websit\
39 e.com')) }}
40             </div>
41         </div>
42         {{ HTML::divider() }}
43         <div class="uk-form-row">
44             {{ Form::submitUK('Simpan') }}
45         </div>
46     {{ Form::close() }}
47     @stop

```

---

Pada view ini saya mengarahkan action ke route `member.profile.update`. Route ini akan kita buat di tahapan selanjutnya. Untuk setiap field saya menggunakan default value dari attribute pada variable `$user` yang dikirim.

3. Tambahkan method `updateProfile` untuk menerima data dari form :

app/controllers/MemberController.php

---

```

1  /**
2   * Ubah profile user
3   * @return response
4   */
5  public function updateProfile()
6  {
7      $user = Sentry::getUser();
8      try
9      {
10          // Perbaharui data user
11          $user->email = Input::get('email');
12          $user->first_name = Input::get('first_name');
13          $user->last_name = Input::get('last_name');

```

```

14     $user->save();
15
16     // Simpan data user
17     if ($user->save())
18     {
19         return Redirect::route('member.profile')->with('successMessage', 'Profil berhasil diperbaharui.');
20     }
21 }
22
23 catch (Cartalyst\Sentry\Users\UserExistsException $e)
24 {
25     return Redirect::back()->with('errorMessage', $e->getMessage())->withInput();
26 }
27 }
```

---

Method ini akan menerima data dari form dan memperbaharui data user sesuai dengan data yang dikirim. Jika email baru dari User sudah ada yang pakai, maka ia akan melemparkan `UserExistsException` yang kemudian kita tampilkan pesan kesalahannya.

Tambahkan route dengan tipe POST untuk method ini :

#### app/routes.php

---

```

1 ....
2 Route::group(array('before' => 'auth'), function () {
3     Route::post('profile', array('as'=>'member.profile.update', 'uses'=>'MemberController@updateProfile'));
4 }
5 ....
```

---

4. Tambahkan tombol **Perbaharui Profil** dari halaman Profil :

#### app/views/profiles/show.blade.php

---

```

1 @extends('layouts.master')
2
3 @section('title')
4     {{ $title }}
5 @stop
6
7 @section('title-button')
8     {{ link_to_route('member.profile.edit', 'Perbaharui', null, array('class'=>'uk-button\uk-button-primary')) }}
9 @stop
10
11 ....
```

---

Tombol ini kita buat menggunakan URL helper menuju route `member.profile.edit` kita juga menambahkan class `uk-button uk-button-primary` agar link tampil sebagai tombol.

LaraPus Dashboard Profil Shidqi Abdullah Mubarak | Logout

**Dashboard / Profil**

# Profil

**Perbaharui**

Nama	Shidqi Abdullah Mubarak
Email	shidqi.abdullah@gmail.com
Login Terakhir	2014-05-27 15:57:48

#### Menambah tombol perbaharui profil

- Terakhir, silahkan test fitur perbaharui profil ini, pastikan data user dapat diubah dan validasi email muncul.

LaraPus Dashboard Profil Shidqi Abdullah Mubarak | Logout

**Dashboard / Profil / Perbaharui Profil**

## Perbaharui Profil

Nama Depan	Rahmat
Nama Belakang	Awaludin
Email	shidqi.abdullah@gmail.co

**Simpan**

Halaman perbaharui profil

The screenshot shows a web browser window titled 'Profil | Laravel Perpus'. The address bar contains 'larapus.site/profile'. The page header includes 'LaraPus', 'Dashboard', 'Buku', 'Profil' (which is active), and a user dropdown 'Rahmat Awaludin | Logout'. A green success message box displays 'Profil berhasil diperbarui.' (Profile successfully updated). Below the message, the breadcrumb navigation shows 'Dashboard / Profil'. The main content area is titled 'Profil' with a 'Perbarui' button. It lists three items: 'Nama' (Rahmat Awaludin), 'Email' (shidqi.abdullah@gmail.com), and 'Login Terakhir' (2014-05-27 15:57:48).

Berhasil perbarui profil

## Ubah Password

Fitur ubah password ini digunakan oleh User yang sudah login untuk mengganti passwordnya. Tentunya, sebelum mengubah password, user harus memasukkan passwordnya yang lama dan password baru harus dikonfirmasi. Fitur ini bisa diakses oleh semua jenis User, baik dia admin maupun regular user.

1. Tambahkan method `editPassword` di `HomeController` untuk menampilkan halaman ubah password :

`app/controllers/HomeController.php`

```

1  ....
2  /**
3   * Tampilkan halaman ubah password
4   * @return response
5   */
6  public function editPassword()
7  {
8      $this->layout->content = View::make('dashboard.editpassword')
9          ->withTitle('Ubah Password');
10 }
11 ....

```

Tambahkan route untuk method ini :

**app/routes.php**

```

1   ....
2   Route::group(array('before' => 'auth'), function () {
3       Route::get('editpassword', array('as'=>'home.editpassword', 'uses'=>'HomeController@e\
4       ditPassword'));
5   ....

```

---

2. Buat view untuk menampilkan form ubah password di app/views/dashboard/editpassword.blade.php  
:

**app/views/dashboard/editpassword.blade.php**

```

1  @extends('layouts.master')
2
3  @section('title')
4      {{ $title }}
5  @stop
6
7  @section('breadcrumb')
8      <li><a href="/">Dashboard</a></li>
9      <li class="uk-active">{{ $title }}</li>
10 @stop
11
12 @section('content')
13     {{ Form::open(array('url' => route('home.updatepassword'), 'method' => 'post', 'class' \
14     => 'uk-form uk-form-horizontal')) }}
15     <div class="uk-form-row">
16         {{ Form::label('oldpassword', 'Password Lama', array('class' => 'uk-form-labe\
17 l uk-text-left')) }}
18         <div class="uk-form-controls">
19             {{ Form::password('oldpassword', array('placeholder'=>'*****')) }}
20         </div>
21     </div>
22     <div class="uk-form-row">
23         {{ Form::label('newpassword', 'Password Baru', array('class' => 'uk-form-labe\
24 l uk-text-left')) }}
25         <div class="uk-form-controls">
26             {{ Form::password('newpassword', array('placeholder'=>'*****')) }}
27         </div>
28     </div>
29
30     <div class="uk-form-row">
31         {{ Form::label('newpassword_confirmation', 'Konfirmasi Password Baru', array(\
32         'class' => 'uk-form-label uk-text-left')) }}
33         <div class="uk-form-controls">
34             {{ Form::password('newpassword_confirmation', array('placeholder'=>'****\


```

```

35     ****')) }}  

36         </div>  

37     </div>  

38     {{ HTML::divider() }}  

39     <div class="uk-form-row">  

40         {{ Form::submitUk('Simpan') }}  

41     </div>  

42     {{ Form::close() }}  

43 @stop

```

---

Di view ini password lama saya masukkan field `oldpassword` sedangkan password baru pada field `newpassword` dan dikonfirmasi pada field `newpassword_confirmation`. Form ini memiliki action ke route `home.updatepassword` yang akan saya buat pada langkah selanjutnya.

Agar form ini mudah diakses, kita akan merubah link logout di pojok kanan atas menjadi dropdown. Ubah navbar di layout master menjadi :

app/views/layouts/master.blade.php

---

```

1   ....  

2   <nav class="uk-navbar">  

3       <a href="#" class="uk-navbar-brand uk-hidden-small">LaraPus</a>  

4       <ul class="uk-navbar-nav uk-hidden-small">  

5  

6           @if (Sentry::getUser()->hasPermission('regular'))  

7               @include('dashboard.navigations.regular')  

8           @endif  

9  

10          @if (Sentry::getUser()->hasPermission('admin'))  

11              @include('dashboard.navigations.admin')  

12          @endif  

13      </ul>  

14      <div class="uk-navbar-nav uk-navbar-flip">  

15          <li class="uk-parent" data-uk-dropdown>  

16              <a href="">{{ Sentry::getUser()->first_name . ' ' . Sentry::getUser()->last_n\ame }}</a>  

17                  <div class="uk-dropdown uk-dropdown-navbar">  

18                      <ul class="uk-nav uk-nav-navbar">  

19                          <li>{{ link_to_route('home.editpassword', 'Ubah Password') }}</li>  

20                          <li class="uk-nav-divider"></li>  

21                          <li><a href="{{ URL::to('logout') }}">Logout</a></li>  

22                      </ul>  

23                  </div>  

24          </li>  

25      </div>  

26      <div class="uk-navbar-brand uk-navbar-center uk-visible-small">LaraPus</div>

```

```
28    </nav>
29    ....
```

Pastikan link Ubah Password sudah muncul.

larapus.site/editpassword

Berhasil membuat link ubah password

- Untuk mengubah password, kita perlu melakukan validasi dengan password yang sudah di hash di database. Sentry menyediakan fitur untuk pengecekan itu yaitu method `checkPassword`.

app/controllers/HomeController.php

```
1  ....
2  /**
3   * Ubah password user
4   * @return response
5   */
6  public function updatePassword()
7  {
8      $user = Sentry::getUser();
9
10     // validasi password lama
11     if(!$user->checkPassword(Input::get('oldpassword'))) {
12         return Redirect::back()->with('errorMessage', 'Password Lama Anda salah.');
13     }
14 }
```

```
15     // validasi konfirmasi password baru
16     $rules = array('newpassword' => 'confirmed|required|min:5');
17     $validator = Validator::make($data = Input::all(), $rules);
18
19     if ($validator->fails()) {
20         return Redirect::back()->withErrors($validator)->withInput();
21     }
22
23     // Simpan password baru
24     $user->password = Input::get('newpassword');
25     $user->save();
26
27     // Redirect ke halaman sebelumnya
28     return Redirect::back()->with('successMessage', 'Password berhasil diubah.');
29 }
30 . . .
```

---

Pada method ini kita memvalidasi password yang diinputkan user apakah sudah sesuai dengan password yang tersimpan di database dengan fungsi `checkPassword`. Kita juga melakukan validasi untuk `newpassword` untuk memastikan field ini diisi dan dikonfirmasi. Ketika semua validasi telah berhasil, kita menyimpan data password user yang baru.

4. Terakhir, silahkan cek ubah password user baik admin maupun regular. Pastikan semua validasi berjalan dan Anda bisa login dengan password yang baru.

The screenshot shows a web browser window titled "Ubah Password | Laravel Project". The URL in the address bar is "larapus.site/editpassword". The page header includes "LaraPus", "Dashboard", "Buku", and "Profil", along with a user profile for "Shidqi Abdullah Mubarak". Below the header, the breadcrumb navigation shows "Dashboard / Ubah Password". The main content is a form titled "Ubah Password" with three input fields: "Password Lama" (old password), "Password Baru" (new password), and "Konfirmasi Password Baru" (confirm new password). A blue "Simpan" (Save) button is located at the bottom left of the form.

Ubah Password

Dashboard / Ubah Password

Ubah Password

Password Lama

\*\*\*\*\*

Password Baru

\*\*\*\*\*

Konfirmasi Password Baru

\*\*\*\*\*

Simpan

Form ubah password

Ubah Password

Dashboard / Ubah Password

>Password berhasil diubah.

Simpan

Berhasil mengubah password

## Lupa Password

Fitur lupa password merupakan fitur yang penting dalam membangun aplikasi web. Dalam Larapus alur untuk fitur ini terbagi menjadi 6 tahap :

1. User mengunjungi halaman lupa password, halaman ini juga diproteksi dengan captcha.
2. User memasukkan email yang biasa digunakan untuk login.
3. Sistem meng-generate reset code dan mengirimkan ke email.
4. User mengunjungi link dengan reset code dari email.
5. Pada halaman ubah password, sistem mengecek kecocokan email dan reset code dengan data di database.
6. User mengisi password baru.

Mari kita buat fitur ini.

1. Buat method `forgotPassword` di `GuestController` untuk menampilkan halaman lupa password :

**app/controllers/HomeController.php**

```

1   ....
2   /**
3    * Tampilkan halaman lupa password
4    * @return response
5   */
6   public function forgotPassword()
7   {
8       $this->layout->content = View::make('guest.forgot');
9   }
10  ....

```

Tambahkan route untuk method ini :

**app/routes.php**

```

1   ....
2   Route::get('forgot', array('as'=>'guest.forgot', 'uses'=>'GuestController@forgotPassword')\
3   ));
4   ?>

```

2. Buat view untuk menampilkan halaman lupa password di app/views/guest/forgot.blade.php :

**app/views/guest/forgot.blade.php**

```

1   @extends('layouts.guest')
2
3   @section('content')
4   <div class="uk-text-center">
5       <div class="uk-vertical-align-middle" style="width: 500px;">
6           <h2 class="uk-text-left">Masukkan email Anda</h2>
7           @include('layouts.partials.validation')
8           {{ Form::open(array('url' => route('guest.sendresetcode'), 'method'=>'post', 'cla\
9           ss' => 'uk-form uk-form-horizontal')) }}
10
11           <div class="uk-form-row">
12               {{ Form::label('email', 'Email', array('class' => 'uk-form-label uk-text-\\
13           left')) }}
14               {{ Form::text('email', 'emailmu@website.com') }}
15           </div>
16
17           <div class="uk-form-row">
18               {{ Form::captcha() }}
19           </div>
20
21           <div class="uk-form-row">
22               {{ Form::submit('Reset', array('class'=>'uk-width-1-1 uk-button uk-button\

```

```

23     -primary uk-button-large')) }}"
24             </div>
25         {{ Form::close() }}
26
27     </div>
28 </div>
29 @stop

```

---

Pada view ini, kita membuat input untuk email user dan captcha. Route guest.sendresetcode akan kita buat pada langkah selanjutnya.

3. Kita perlu membuat halaman ini dapat diakses oleh user. Oleh karena itu, mari kita tambahkan link ke halaman lupa password di halaman login. Tambahkan baris ini sebelum menutup form :

app/views/guest/login.blade.php

---

```

1     ....
2         <div class="uk-form-row">
3             {{ link_to_route('guest.forgot', 'Anda lupa password?') }}
4         </div>
5
6     {{ Form::close() }}
7     ....

```

---

4. Untuk menerima data yang dikirim dari form, kita perlu membuat method sendResetCode di GuestController yang akan memvalidasi email dan captcha yang dikirim, kemudian mengirimkan link reset password ke email user.

app/controllers/GuestController.php

---

```

1     ....
2     /**
3      * Kirim email reset password
4      * @return response
5      */
6     public function sendResetCode()
7     {
8         // Validasi email dan captcha
9         $rules = array(
10             'email' => 'required|exists:users',
11             'recaptcha_response_field' => 'required|recaptcha'
12         );
13
14         $validator = Validator::make($data = Input::all(), $rules);
15
16         // Redirect jika validasi gagal
17         if ($validator->fails()) {
18             return Redirect::back()->withErrors($validator)->withInput();

```

```

19     }
20
21     $user = Sentry::findUserByLogin(Input::get('email'));
22     $data = array(
23         'email' => $user->email,
24         // Generate code untuk password reset
25         'resetCode' => $user->getResetPasswordCode(),
26     );
27
28     // Kirim email untuk reset password
29     Mail::send('emails.auth.reminder', $data, function ($message) use ($user) {
30         $message->to($user->email, $user->first_name . ' ' . $user->last_name)->subject('\
31     Reset Password Larapus');
32     });
33
34     // Redirect ke halaman login
35     return Redirect::route('guest.login')->with("successMessage", "Silahkan cek email Anda\
36     a ($user->email) untuk me-reset password.");
37 }
38 ....

```

---

Pada method ini, kita menggunakan validasi dari Eloquent untuk melakukan validasi email dan captcha. Menggunakan Sentry, kita mencari User berdasarkan email. Untuk generate code reset password kita menggunakan method getResetPasswordCode. Data \$email dan \$resetCode ini yang akan dikirim ke email User. Terakhir, user akan di-*redirect* ke halaman login dengan pesan **Silahkan cek email Anda (alamatemail) untuk reset password..**

Tambahkan route untuk method ini dengan tipe POST :

#### app/routes.php

---

```

1     ....
2     Route::post('sendresetcode', array('as'=>'guest.sendresetcode', 'uses'=>'GuestController@\
3     sendResetCode'));
4     ?>

```

---

Pada method ini, kita juga mengirim email dengan template emails.auth.reminder mari kita buat view untuk template ini :

---

app/views/emails/auth/reminder.blade.php

---

```

1  <!DOCTYPE html>
2  <html lang="en-US">
3      <head>
4          <meta charset="utf-8">
5      </head>
6      <body>
7          <h2>Reset Password Larapus</h2>
8
9          <div>
10             Untuk me-reset password akun Anda, silahkan klik link berikut: <br>
11             {{ link_to_route('guest.create newPassword', null, ['email'=>$email, 'resetCode'=>$resetCode]) }}
12         </div>
13     </body>
14 </html>

```

---

Terlihat disini, pada email yang kita kirim kita membuat link ke route `guest.create newPassword`. Route ini akan kita buat di tahap selanjutnya.

5. Ketika user telah mengikuti link reset password, dia akan dihadapkan ke form untuk reset password. Dalam menampilkan form ini, kita akan memvalidasi email dan reset code yang dikirim dengan data di database. Buatlah method `createNewPassword` di `GuestController` untuk menampilkan halaman ini :

---

app/controllers/GuestController.php

---

```

1  ....
2  /**
3   * Tampilkan halaman untuk membuat password baru
4   * @param string $token
5   * @return response
6   */
7  public function createNewPassword()
8  {
9      try {
10         $user = Sentry::findUserByLogin(Input::get('email'));
11         if($user->checkResetPasswordCode(Input::get('resetCode'))) {
12             $this->layout->content = View::make('guest.resetpassword')
13                 ->with('email', $user->email)
14                 ->with('resetCode', Input::get('resetCode'));
15         } else {
16             return Redirect::back()->with('errorMessage', 'Reset code tidak valid.');
17         }
18     } catch (UserNotFoundException $e) {
19         return Redirect::route('guest.login')->with('errorMessage', $e->getMessage());
20     }

```

```
21     }
22     ....
```

---

Pada method ini, kita menggunakan method `findUserByLogin` untuk mencari user berdasarkan email yang dikirim. Jika user tidak ditemukan, kita menangkap exception `UserNotFoundException` dan menampilkan pesan kesalahan. Setelah user ditemukan, kita mengecek reset code yang dikirim dengan data di databatase. Jika reset code sesuai, maka kita tampilkan view `guest.resetpassword` sambil mengirimkan data `$email` dan `$resetCode`. Data ini akan kita gunakan sebagai `hidden` field pada form untuk merubah password.

Tambahkan route untuk method ini :

app/routes.php

---

```
1     ....
2     Route::get('reset', array('as'=>'guest.create newPassword', 'uses'=>'GuestController@creat\
3     eNewPassword'));
4     ?>
```

---

Buat juga view untuk method ini :

app/views/dashboard/resetpassword.blade.php

---

```
1     @extends('layouts.guest')
2
3     @section('content')
4     <div class="uk-text-center">
5         <div class="uk-vertical-align-middle" style="width: 500px;">
6             @include('layouts.partials.validation')
7
8             <h3>Masukkan password baru Anda :</h3>
9             {{ Form::open(array('url' => route('guest.storenewpassword'), 'method'=>'post', '\
10    class' => 'uk-form uk-form-horizontal')) }}
11
12             <div class="uk-form-row">
13                 {{ Form::label('password', 'Password', array('class' => 'uk-form-label uk\ \
14 -text-left')) }}
15                 <div class="uk-form-controls">
16                     {{ Form::password('password', array('placeholder'=>'*****')) }}
17                 </div>
18             </div>
19
20             <div class="uk-form-row">
21                 {{ Form::label('password_confirmation', 'Konfirmasi Password', array('cla\
22 ss' => 'uk-form-label uk-text-left')) }}
23                 <div class="uk-form-controls">
24                     {{ Form::password('password_confirmation', array('placeholder'=>'****\ \
25 *****')) }}}
```

```

26         </div>
27     </div>
28
29     <div class="uk-form-row">
30         {{ Form::hidden('email', $email) }}
31         {{ Form::hidden('resetCode', $resetCode) }}
32     </div>
33
34
35     <div class="uk-form-row">
36         {{ Form::submit('Reset', array('class'=>'uk-width-1-1 uk-button uk-button\
37 -primary uk-button-large')) }}
38     </div>
39     {{ Form::close() }}
40
41     </div>
42 </div>
43 @stop

```

---

Form di view ini akan menampilkan field password dan konfirmasi password. Kita juga menggunakan *hidden field* untuk menyimpan email dan reset code user.

6. Data yang dikirim oleh User pada form perubahan password akan kita terima di method `storeNewPassword` di `GuestController` :

`app/controllers/GuestController.php`

---

```

1   ....
2 /**
3 * Simpan password user yang baru
4 * @param string $token
5 * @return response
6 */
7 public function storeNewPassword()
8 {
9     try
10    {
11        // Cari user berdasarkan email
12        $user = Sentry::findUserByLogin(Input::get('email'));
13
14        // Check apakah resetCode valid
15        if ($user->checkResetPasswordCode(Input::get('resetCode')))
16        {
17            // Validasi password baru
18            $rules = array('password' => 'confirmed|required|min:5');
19            $validator = Validator::make($data = Input::all(), $rules);
20

```

```

21         // Redirect jika validasi gagal
22         if ($validator->fails()) {
23             return Redirect::back()->withErrors($validator)->withInput();
24         }
25
26         // Reset password user
27         $user->attemptResetPassword('8f1Z7wA4uVt7VemBpGSfaoI9mcjdEwtK8e1CnQ0b', Input\
28 ::get('password'));
29
30         return Redirect::route('guest.login')->with('successMessage', 'Berhasil me-re\
31 set password. Silahkan login dengan password baru.');
32     }
33     else
34     {
35         return Redirect::back()->with('errorMessage', 'Reset code tidak valid.');
36     }
37 }
38 catch (Cartalyst\Sentry\Users\UserNotFoundException $e)
39 {
40     return Redirect::back()->with('errorMessage', $e->getMessage());
41 }
42 }
43 ....

```

---

Pada method ini, kita mencari user dengan method `findUserByLogin`. Reset code yang diberikan, kita cek dengan data di database dengan method `checkResetPasswordCode`. Kita juga memvalidasi password baru yang dikirim oleh user dengan 3 aturan, *harus dikonfirmasi*, *harus diisi*, dan *minimal 5 karakter*. Setelah semua validasi berhasil, password user kita rubah dengan method `attemptResetPassword`. Terakhir, kita mengarahkan user ke halaman login dengan pesan **Berhasil me-reset password. Silahkan login dengan password baru.**.

Tambahkan route untuk method ini dengan tipe POST :

#### app/routes.php

---

```

1 ....
2 Route::post('reset', array('as'=>'guest.storenewpassword', 'uses'=>'GuestController@store\
3 NewPassword'));
4 ?>

```

---

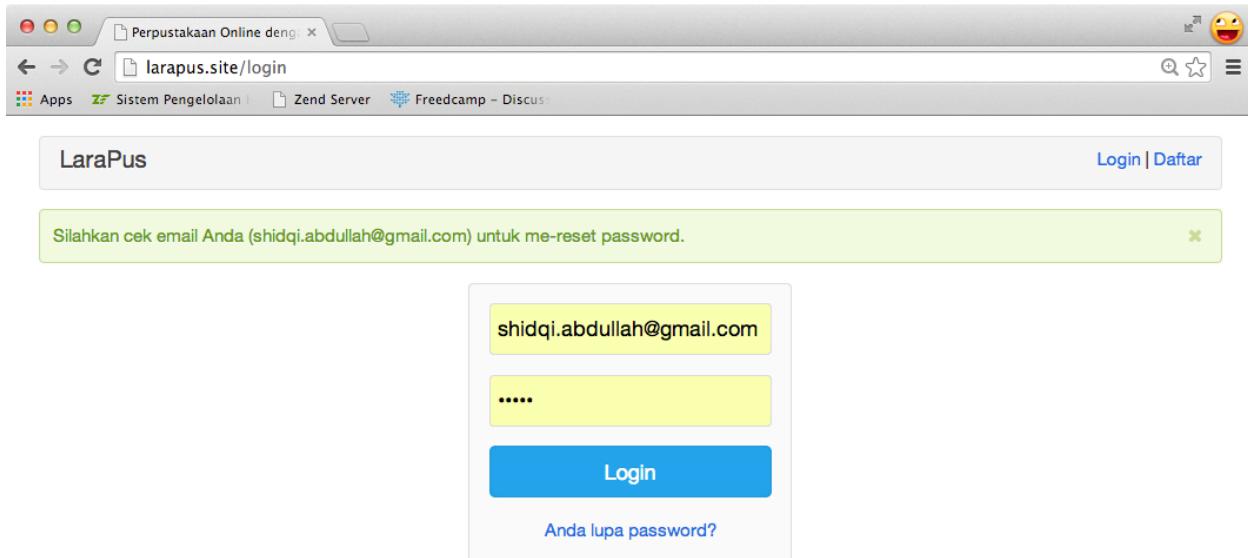
7. Terakhir, silahkan dites fitur lupa password ini. Pastikan link muncul, form lupa password berjalan, email diterima, validasi berjalan dan Anda bisa login dengan password yang baru.

The screenshot shows a web browser window with the URL [larapus.site/login](http://larapus.site/login). The page title is "LaraPus". It features a login form with two input fields: "Email" and "Password", and a blue "Login" button. Below the form is a link "Anda lupa password?". The browser's top bar includes various icons and tabs.

Link lupa password di halaman login

The screenshot shows a web browser window with the URL [larapus.site/forgot](http://larapus.site/forgot). The page title is "LaraPus". It displays a form asking "Masukkan email Anda" with an input field containing "emailmu@website.com". Below the input field is a CAPTCHA challenge showing the text "haventlos functions" and a reCAPTCHA interface. A "Reset" button is at the bottom of the form. The browser's top bar includes various icons and tabs.

Form lupa password



### Email lupa password dikirim

**Reset Password Larapus**

Larapus Admin <laravel.perpustakaan@gmail.com> 9:14 AM (4 minutes ago)

Untuk me-reset password akun Anda, silahkan klik link berikut:  
<http://larpus.site/reset?email=shidqi.abdullah%40gmail.com&resetCode=GqnijilanDbznLFdaeSRPMNmGHj9od0Zv0xCrDkRih>

### Email lupa password diterima

The screenshot shows a web browser window with the title bar "Reset Password Larapus". The URL in the address bar is "larapus.site/reset?email=shidqi.abdullah%40gmail.com&resetCode=GqnLjiianDbznLFdaeSRPMNmGHj9od0Zv0xCrDkRlh". The page content is titled "LaraPus" and contains a form for entering a new password. The form has two input fields: "Password" and "Konfirmasi Password", both represented by redacted text boxes. A blue "Reset" button is centered below the inputs.

### Form reset password

The screenshot shows a web browser window with the title bar "Reset Password Larapus". The URL in the address bar is "larapus.site/login". The page content is titled "LaraPus" and displays a green success message box containing the text "Berhasil me-reset password. Silahkan login dengan password baru." Below the message is a login form with two input fields: one for email ("shidqi.abdullah@gmail.com") and one for password ("....."). A blue "Login" button is at the bottom of the form. A link "Anda lupa password?" is also present.

Berhasil me-reset password

## Listing Member

Fitur member di halaman Admin akan kita gunakan untuk membuat listing user regular yang telah terdaftar di aplikasi. Pada pengembangan fitur ini kita akan menggunakan fitur dari Sentry untuk mendapatkan semua user berdasarkan permissions. Kendala dari fitur ini adalah output yang dihasilkan berupa Array, sedangkan package datatable yang kita gunakan hanya mengizinkan query dan collection sebagai parameternya. Karenanya, pada tahap ini saya juga akan menunjukkan bagaimana cara mengubah Array menjadi Collection di Laravel.

1. Persiapkan UsersController dengan method index untuk menampilkan listing user :

app/controllers/UsersController.php

```

1 <?php
2
3 class UsersController extends BaseController {
4
5     /**
6      * Tampilkan listing User regular
7      * @return response
8      */
9     public function index()
10    {
11        if(Datatable::shouldHandle())
12        {
13            // array dari Sentry
14            $usersArray = Sentry::findAllUsersWithAccess('regular');
15
16            // mengubah array menjadi collection
17            $usersCollection = new Illuminate\Database\Eloquent\Collection($usersArray);
18
19            // buat datatable
20            return Datatable::collection($usersCollection)
21                ->addColumn('full_name', function ($model) {
22                    return $model->first_name . ' ' . $model->last_name;
23                })
24                ->showColumns('email', 'last_login')
25                ->addColumn('', function ($model) {
26                    $html = '<a href="'.route('admin.users.show', ['users'=>$model->id]).'\' .
27 "' class="uk-button uk-button-small uk-button-link">Lihat data peminjaman</a> ';
28                    $html .= Form::open(array('url' => route('admin.users.destroy', ['use\
29 rs'=>$model->id]), 'method'=>'delete', 'class'=>'uk-display-inline'));
30                    $html .= Form::submit('delete', array('class' => 'uk-button uk-button\
31 -small'));
32                    $html .= Form::close();
33            return $html;

```

```

34         })
35         ->searchColumns('full_name', 'email', 'last_login')
36         ->orderColumns('full_name', 'email', 'last_login')
37         ->make();
38     }
39     return View::make('users.index')->withTitle('Member');
40 }
41
42 }
```

---

Di method `index` saya mengambil data user dengan method `findAllUsersWithAccess` untuk memfilter user regular. Output dari method ini adalah array. Untuk mengubahnya menjadi Collection, saya membuat instance/object baru `Illuminate\Database\Eloquent\Collection` dengan parameter array yang kita dapatkan.

Ketika membuat datatable saya membuat custom column `full_name` yang merupakan gabungan dari `first_name` dan `last_name`. Saya juga menambah link untuk melihat data peminjaman dan tombol untuk menghapus user.

Controller ini akan kita jadikan *resource controller*, silahkan tambah di `app/routes.php` :

#### app/routes.php

---

```

1 ....
2 Route::group(array('prefix' => 'admin', 'before' => 'admin'), function()
3 {
4     Route::resource('authors', 'AuthorsController');
5     Route::resource('books', 'BooksController');
6     Route::resource('users', 'UsersController');
7 });
8 ....
```

---

2. Buat view untuk menampilkan listing member di `app/views/users/index.blade.php` :

#### app/views/users/index.blade.php

---

```

1 @extends('layouts.master')
2
3 @section('asset')
4     @include('layouts.partials.datatable')
5 @stop
6
7 @section('title')
8     {{ $title }}
9 @stop
10
11 @section('breadcrumb')
12     <li><a href="/">Dashboard</a></li>
13     <li class="uk-active">{{ $title }}</li>
```

```

14     @stop
15
16     @section('content')
17
18     {{ Datatable::table()
19         ->addColumn('email', 'last_login', 'full_name', '')
20         ->setOptions('aoColumnDefs', array(
21             array(
22                 'sTitle' => 'Nama',
23                 'aTargets' => [0]),
24             array(
25                 'sTitle' => 'Email',
26                 'aTargets' => [1]),
27             array(
28                 'sTitle' => 'Login Terakhir',
29                 'aTargets' => [2]),
30             array(
31                 'bSortable' => false,
32                 'aTargets' => [3])
33         )))
34         ->setOptions('bProcessing', true)
35         ->setUrl(route('admin.users.index'))
36         ->render('datatable.uikit')  --}}
37
38     @stop

```

---

3. Tambahkan link untuk halaman member ini di `app/views/dashboard/navigations/admin.blade.php` :

`app/views/dashboard/navigations/admin.blade.php`

---

```

1  {{ HTML::smartNav(route('dashboard'), 'Dashboard')}}
2  {{ HTML::smartNav(route('admin.books.index'), 'Buku')}}
3  {{ HTML::smartNav(route('admin.authors.index'), 'Penulis')}}
4  {{ HTML::smartNav(route('admin.users.index'), 'Member')}}
5  {{ HTML::smartNav('#', 'Peminjaman')}}
```

---

4. Coba cek tampilan datatable yang baru dibuat..

#### Ada bug di kolom terakhir

Ops, ternyata ada bug. Terlihat disini, kolom login terakhir berisi [object Object]. Hal ini terjadi karena Laravel secara otomatis merubah output untuk kolom bertipe data timestamp menjadi object dari Carbon. Penjelasan lengkapnya dapat Anda baca [dokumentasi resmi<sup>106</sup>](#). Solusi untuk masalah ini adalah menggunakan method `toDateTimeString` untuk merubahnya menjadi string. Silahkan ubah di handle datatable di `UsersController` yang berisi `->showColumns('email', 'last_login')` menjadi :

`app/controllers/UsersController.php`

```

1   ....
2   ->showColumns('email')
3   ->addColumn('last_login', function($model) {
4       return ($model->last_login ? $model->last_login->toDateTimeString() : '');
5   })
6   ....

```

Pada perubahan ini kita juga mengecek apakah user sudah pernah login, jika belum maka sistem akan menampilkan string kosong.

Cek kembali output datatable yang dihasilkan :

<sup>106</sup><http://laravel.com/docs/eloquent#date-mutators>

The screenshot shows a web application interface for managing members. The top navigation bar includes links for LaraPus, Dashboard, Buku, Penulis, Member (which is active), Peminjaman, and Admin Larapus. The main content area is titled 'Member' and shows a table with one entry:

Nama	Email	Login Terakhir
Shidqi Abdullah Mubarak	shidqi.abdullah@gmail.com	2014-05-28 02:20:30

Below the table, there are buttons for 'Lihat data peminjaman' and 'delete'. At the bottom of the page, a message says 'Showing 1 to 1 of 1 entries' and provides navigation links for First, Previous, Next, and Last.

Berhasil membuat listing user

## Menampilkan Data Member

Fitur ini berguna agar Admin bisa melihat buku apa saja yang sedang dipinjam oleh seorang user.

1. Buat method show di UsersController untuk menampilkan detail user :

app/controllers/UsersController.php

```

1  /**
2   * Tampilkan detail user
3   * @param int $id
4   * @return response
5   */
6  public function show($id)
7  {
8      $user = User::find($id);
9      return View::make('users.show')->withUser($user)
10         ->withTitle("Detail $user->first_name $user->last_name");
11 }

```

2. Buat untuk menampilkan halaman ini di app/views/users/show.blade.php :

app/views/users/show.blade.php

---

```
1  @extends('layouts.master')
2
3  @section('asset')
4      @include('layouts.partials.datatable')
5  @stop
6
7  @section('title')
8      {{ $title }}
9  @stop
10
11 @section('breadcrumb')
12     <li><a href="/">Dashboard</a></li>
13     <li><a href="{{ route('admin.users.index') }}">Member</a></li>
14     <li class="uk-active">{{ $title }}</li>
15 @stop
16
17 @section('content')
18     <h3>Buku yang sedang dipinjam :</h3>
19     <ul>
20         @foreach ($user->books as $book)
21             <li>{{ $book->title }}</li>
22         @endforeach
23     </ul>
24
25 @stop
```

---

3. Cek apakah fitur ini berjalan.

The screenshot shows a web browser window with the following details:

- Title Bar:** Detail Shidqi Abdullah Mubar
- Address Bar:** larapus.site/admin/users/39
- Toolbar:** Apps, Sistem Pengelolaan, Zend Server, Freedcamp – Discuss
- Header:** LaraPus, Dashboard, Buku, Penulis, Member, Peminjaman, Admin Larapus
- Breadcrumbs:** Dashboard / Member / Detail Shidqi Abdullah Mubarak
- Main Content:**

# Detail Shidqi Abdullah Mubarak

**Buku yang sedang dipinjam :**

  - Kupinang Engkau dengan Hamdalah
  - Jalan Cinta Para Pejuang
- Message:** Berhasil membuat detail peminjaman

Tugas : Oke, saya paham yang Anda pikirkan. Fitur ini kurang keren kan? Nah, itu dia, biar tambah keren silahkan buat datatable di halaman ini yang menampilkan daftar buku yang telah dipinjam, sedang dipinjam, tanggal pinjam dan tanggal pengembalian. Tentunya, isian tanggal pengembalian harus kosong jika buku belum dikembalikan.

## Hapus Member

Untuk melengkapi fitur Member ini, mari kita buat fitur untuk menghapus member. Fitur ini cukup sederhana, Anda cukup menambah method `destroy` di `UsersController` :

**app/controllers/GuestController.php**

```
1 ...
2 /**
3  * Hapus user
4  * @param int $id
5  * @return response
6 */
7 public function destroy($id)
8 {
9     $user = User::find($id);
10    $user->delete();
11    return Redirect::route('admin.users.index')
12        ->with('successMessage', 'User berhasil dihapus');
13 }
14 ...
```

Pastikan User bisa dihapus.

The screenshot shows a web browser window with the title "Member | Laravel Perpus". The address bar displays "larapus.site/admin/users". The page content is for managing members, with a navigation bar including "LaraPus", "Dashboard", "Buku", "Penulis", "Member" (which is active), and "Peminjaman". On the right, it says "Admin Larapus". A green success message box contains the text "User berhasil dihapus". Below the message, the breadcrumb navigation shows "Dashboard / Member". The main area is titled "Member" and contains a table with three columns: "Nama", "Email", and "Login Terakhir". The table has a header row and a single data row that says "No data available in table". At the bottom, there are search and filter options ("Show 10 entries", "Search:"), and navigation links for "First", "Previous", "Next", and "Last".

Berhasil hapus user

Tugas : Penghapusan member disini langsung dilakukan walaupun member tersebut sedang meminjam buku. Buatlah observer ketika model User di hapus, pastikan sistem menghentikan penghapusan user jika user tersebut masih meminjam buku dan berikan pesan pemberitahuan buku apa saja yang masih dipinjam. Sebagai petunjuk, silahkan cek kembali validasi yang kita buat ketika menghapus buku di Hari 5.

## Data Peminjaman

Fitur terakhir di halaman Admin adalah fitur data peminjaman. Pada fitur ini Admin dapat melihat buku, member, dan tanggal peminjaman dari buku. Pada halaman ini Admin juga akan diizinkan untuk mengembalikan buku yang sedang dipinjam oleh member.

Data yang diperoleh pada tampilan ini adalah setiap baris pada table book\_user. Query seperti ini yang akan kita jalankan :

app/controllers/GuestController.php

---

```
1 select
2     books.id,
3     books.title as book,
4     concat(users.first_name, ' ', users.last_name) as user,
5     date(book_user.created_at) as borrow_at
6 from book_user
7 left join books on book_user.book_id = books.id
8 left join users on book_user.user_id = users.id;
```

---

Mari kita coba di *console mysql* :

```
2. mysql -u root -ptoor larapus (mysql)
mysql> select * from book_user;
+---+-----+-----+-----+-----+-----+
| id | book_id | user_id | returned | created_at | updated_at |
+---+-----+-----+-----+-----+-----+
| 1 | 1 | 45 | 1 | 2014-05-20 08:03:57 | 2014-05-29 06:22:02 |
| 2 | 2 | 45 | 1 | 2014-05-20 08:03:57 | 2014-05-29 06:22:03 |
| 3 | 4 | 45 | 0 | 2014-05-29 06:22:08 | 2014-05-29 07:18:38 |
| 4 | 2 | 45 | 0 | 2014-05-29 06:22:10 | 2014-05-29 07:17:57 |
| 5 | 3 | 45 | 0 | 2014-05-29 06:22:12 | 2014-05-29 06:22:12 |
+---+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql> select
    ->     books.id,
    ->     books.title as book,
    ->     concat(users.first_name, ' ', users.last_name) as user,
    ->     date(book_user.created_at) as borrow_at
    ->   from book_user
    -> left join books on book_user.book_id = books.id
    -> left join users on book_user.user_id = users.id;
+---+-----+-----+-----+
| id | book | user | borrow_at |
+---+-----+-----+-----+
| 1 | Kupinang Engkau dengan Hamdalah | Shidqi Abdullah Mubarak | 2014-05-20 |
| 2 | Jalan Cinta Para Pejuang | Shidqi Abdullah Mubarak | 2014-05-20 |
| 4 | Cinta & Seks Rumah Tangga Muslim | Shidqi Abdullah Mubarak | 2014-05-29 |
| 2 | Jalan Cinta Para Pejuang | Shidqi Abdullah Mubarak | 2014-05-29 |
| 3 | Membingkai Surga dalam Rumah Tangga | Shidqi Abdullah Mubarak | 2014-05-29 |
+---+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Coba query di mysql

Untuk memudahkan query, saya akan membuat model untuk table book\_user di app/models/Borrow.php :

app/models/Borrow.php

---

```
1 <?php
2
3 class Borrow extends BaseModel {
4     protected $table = 'book_user';
5 }
```

---

Disini, saya menggunakan atribut \$table untuk memberitahu Laravel table yang akan digunakan untuk

model Borrow ini.

Kita juga perlu menambah relasi ke model Book agar bisa menampilkan judul buku dan User agar bisa menampilkan nama user yang sedang meinjam buku.

app/models/Borrow.php

---

```

1 <?php
2
3 class Borrow extends BaseModel {
4     protected $table = 'book_user';
5
6     /**
7      * Relasi One-to-One dengan Book
8      * @return Book
9      */
10    public function book()
11    {
12        return $this->belongsTo('Book');
13    }
14
15    /**
16     * Relasi One-to-One dengan User
17     * @return User
18     */
19    public function user()
20    {
21        return $this->belongsTo('User');
22    }
23
24 }
```

---

Saya terbiasa mengetes terlebih dahulu apakah model yang saya buat berjalan. Ini yang saya lakukan :

1. Tambahkan route test yang melakukan query ke model Borrow :

app/routes.php

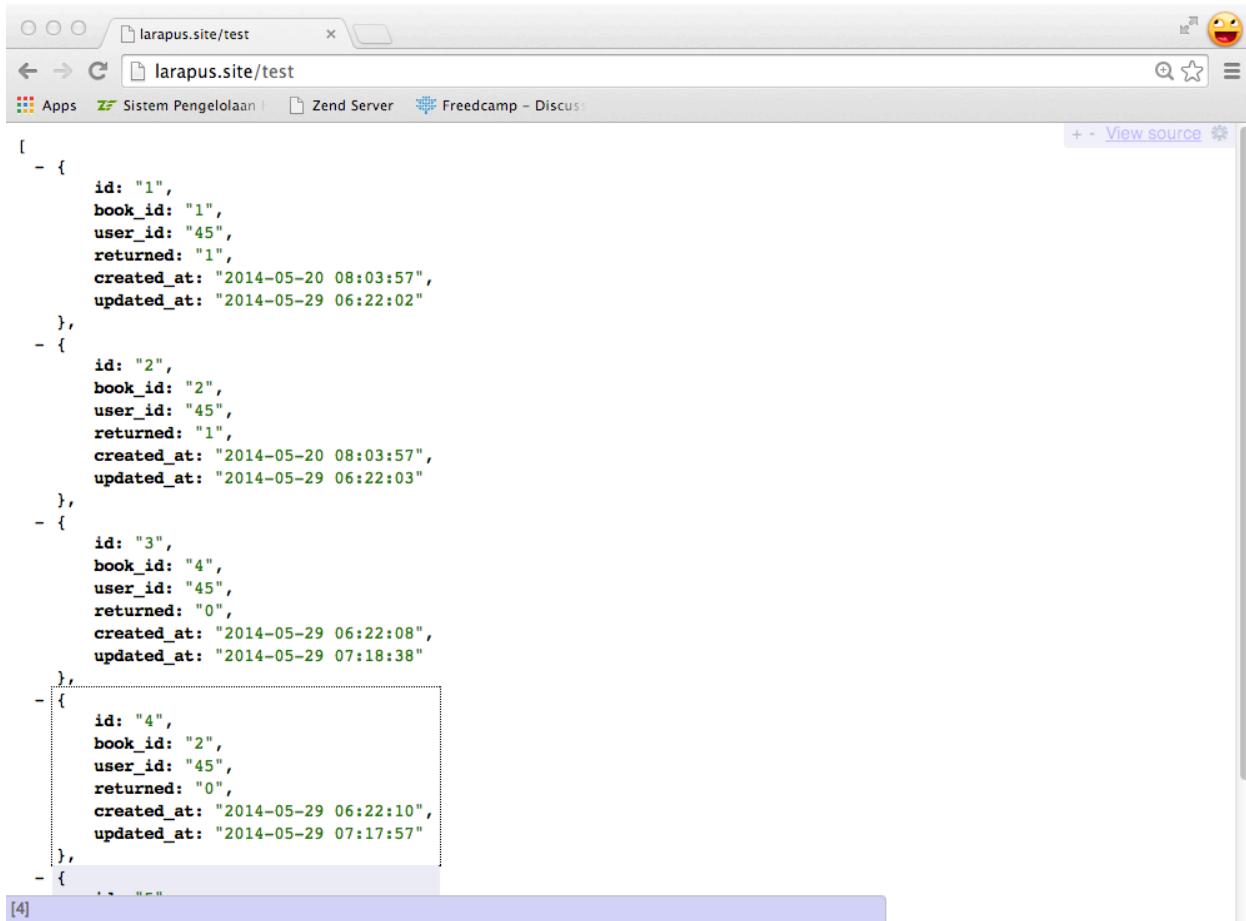
---

```

1 Route::get('test', function() {
2     $borrows = Borrow::all();
3     return $borrows;
4 });
```

---

2. Cek hasilnya di <http://larapus.site/test> :



The screenshot shows a browser window with the URL `larapus.site/test`. The page content is a JSON array of `Borrow` objects. Each object has the following fields: `id`, `book_id`, `user_id`, `returned`, `created_at`, and `updated_at`. The `id` field is highlighted for the fourth object in the array, which has a value of "4". The `created_at` and `updated_at` fields also show specific dates and times.

```
[{"id": "1", "book_id": "1", "user_id": "45", "returned": "1", "created_at": "2014-05-20 08:03:57", "updated_at": "2014-05-29 06:22:02"}, {"id": "2", "book_id": "2", "user_id": "45", "returned": "1", "created_at": "2014-05-20 08:03:57", "updated_at": "2014-05-29 06:22:03"}, {"id": "3", "book_id": "4", "user_id": "45", "returned": "0", "created_at": "2014-05-29 06:22:08", "updated_at": "2014-05-29 07:18:38"}, {"id": "4", "book_id": "2", "user_id": "45", "returned": "0", "created_at": "2014-05-29 06:22:10", "updated_at": "2014-05-29 07:17:57"}, {"id": "5", "book_id": "3", "user_id": "45", "returned": "0", "created_at": "2014-05-29 06:22:10", "updated_at": "2014-05-29 07:17:57"}]
```

Mengecek model Borrow

Nah, dengan gini saya yakin modelnya sudah berfungsi. Sekedar info, agar tampilan browsernya seperti saya, saya pakai Google Chrome dengan extention [JSONView](#)<sup>107</sup>.

## Eager Loading

Ada fitur Laravel lain yang akan kita gunakan disini, yaitu [Eager Loading](#)<sup>108</sup>. Dengan fitur ini, kita kita meload relasi dari model maka akan menggunakan query in. Misalnya ketika meload relasi dari `Borrow` ke `Book`, biasanya yang terjadi adalah `select * from books where id = ?`. Dengan query ini, jika records di table `book_user` ada 10, maka akan ada 10 query `select` ke table `books`. Menggunakan *eager loading*, query `select` ke table `books` akan berubah menjadi 1 query `select * from books where id in (?, ?, ?, ?)`. Teknik ini tentunya akan membuat aplikasi kita lebih cepat.

Cara mengaktifkan *eager loading* di Laravel adalah dengan menggunakan method `with` ketika melakukan query ke model. Mari saya tunjukan pada route test yang baru dibuat.

<sup>107</sup><https://chrome.google.com/webstore/detail/chklaanhfefbnpoihckbnefhakgolnmc>

<sup>108</sup><http://laravel.com/docs/eloquent#eager-loading>

- Untuk memudahkan pemahaman, mari kita log setiap query yang dilakukan langsung di browser. Tambahkan baris ini di baris setelah <?php di app/routes.php :

**app/routes.php**

```

1   ....
2   Event::listen('illuminate.query', function($query) {
3       echo $query . '<br>';
4   });
5   ....

```

- Untuk mengetes sebelum penggunaan *eager loading*, ubah route test menjadi :

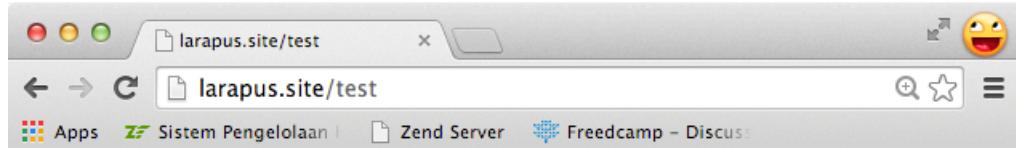
**app/routes.php**

```

1   ....
2   Route::get('test', function() {
3       $borrows = Borrow::all();
4       foreach ($borrows as $borrow) {
5           $borrow->book;
6           $borrow->user;
7       }
8       return '';
9   });
10  ....

```

Query ini akan me-load relasi ke model Book dan User. Akses hasilnya di <http://larapus.site/test>, berikut hasil yang akan didapatkan :



```

select * from `book_user`
select * from `books` where `books`.`id` = ? limit 1
select * from `users` where `users`.`id` = ? limit 1
select * from `books` where `books`.`id` = ? limit 1
select * from `users` where `users`.`id` = ? limit 1
select * from `books` where `books`.`id` = ? limit 1
select * from `users` where `users`.`id` = ? limit 1
select * from `books` where `books`.`id` = ? limit 1
select * from `users` where `users`.`id` = ? limit 1
select * from `books` where `books`.`id` = ? limit 1
select * from `users` where `users`.`id` = ? limit 1

```

Sebelum penggunaan eager loading

Terlihat disini, kita melakukan 11 query select. 1 query ke table book\_user (menghasilkan 5 record), 5 ke table users dan 5 ke table books.

3. Sekarang, kita aktifkan *eager loading* dengan mengubah `all()` menjadi `with() -> get()` parameter pada method `with` adalah nama relasi yang kita tentukan di model Book yaitu `book` dan `user`. Sehingga route test menjadi :

#### app/routes.php

---

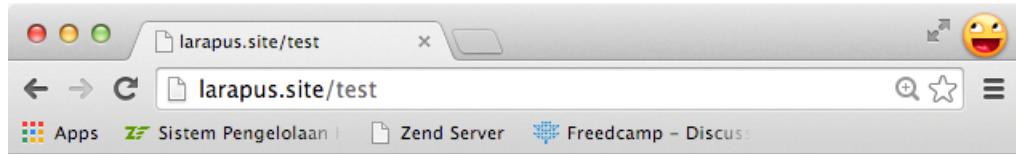
```

1   ....
2   Route::get('test', function() {
3       $borrows = Borrow::with('book', 'user')->get();
4       foreach ($borrows as $borrow) {
5           $borrow->book;
6           $borrow->user;
7       }
8       return '';
9   });
10  ....

```

---

Akses hasilnya di <http://larapus.site/test>, mari kita cek hasil yang kita dapatkan :



```

select * from `book_user`
select * from `books` where `books`.`id` in (?, ?, ?, ?, ?)
select * from `users` where `users`.`id` in (?)

```

#### Sesudah penggunaan eager loading

Terlihat jelas disini, kita hanya melakukan 3 query. 1 ke table book\_user, 1 ke table books dan 1 ke table users. Terlihat juga, Laravel menggunakan `in` untuk query nya.

Dengan teknik ini, cukup besar penghematan query yang bisa dilakukan. Bayangkan jika ada 10000 record di table book\_user, tanpa *eager loading* Laravel akan melakukan  $1 + 10000 + 10000 = 20001$  query! Dengan eager loading, Laravel hanya akan melakukan 3 query. Jika digunakan dengan tepat, eager loading dapat mempercepat performa aplikasi yang sedang dibuat.

Setelah memahami eager loading, mari kita develop fitur data peminjaman ini.

1. Buat controller BorrowController dengan method `index` untuk menampilkan halaman data peminjaman

## app/controllers/BorrowController.php

```

1  <?php
2
3  class BorrowController extends \BaseController {
4      /**
5       * Tampilkan halaman index
6       * @return response
7       */
8      public function index()
9      {
10         if(Datatable::shouldHandle())
11         {
12             // Membuat datatable
13             return Datatable::collection(Borrow::with('user', 'book')->get())
14                 ->showColumns('id')
15                 ->addColumn('book', function($model) {
16                     return $model->book->title;
17                 })
18                 ->addColumn('user', function($model) {
19                     return $model->user->first_name . ' ' . $model->user->last_name;
20                 })
21                 ->addColumn('borrowed_at', function($model) {
22                     return $model->created_at->toDateString();
23                 })
24                 ->addColumn('returned_at', function($model) {
25                     if ($model->returned == 0) {
26                         return '<a href="#" class="uk-button uk-button-small">kembalikan\
27 </a>';
28                     }
29                     return $model->updated_at->toDateString();
30                 })
31                 ->searchColumns('book', 'user', 'borrowed_at', 'returned_at')
32                 ->orderColumns('book', 'user', 'borrowed_at', 'returned_at')
33                 ->make();
34         }
35         return View::make('borrow.index')->withTitle('Data Peminjaman');
36     }
37 }
```

Pada method ini kita mengambil data dari model Borrow dengan *eager loading* ke model User dan Book. Kolom yang kita tampilkan yaitu

- id,
- book merupakan judul buku yang diambil dari relasi book,
- user merupakan nama user yang diambil dari relasi user,

- borrowed\_at merupakan field created\_at yang kita tampilkan hanya tanggallnya saja dengan method `toDateString`. Method ini kita dapatkan dari Carbon.
- returned\_at yang akan berisi field updated\_at jika buku telah dikembalikan atau berisi tombol **kembalikan** jika buku belum dikembalikan. Fitur untuk mengembalikan akan kita develop pada tahapan selanjutnya.

Tambahkan route untuk method ini pada group dengan before filter `admin` :

app/routes.php

```

1  ....
2  Route::group(array('prefix' => 'admin', 'before' => 'admin'), function()
3  {
4      Route::get('borrow', array('as'=>'admin.borrow', 'uses'=>'BorrowController@index'));
5  ....

```

2. Buat view untuk menampilkan datatable di app/views/borrow/index.blade.php :

app/views/borrow/index.blade.php

```

1  ....
2  @extends('layouts.master')
3
4  @section('asset')
5      @include('layouts.partials.datatable')
6  @stop
7
8  @section('title')
9      {{ $title }}
10 @stop
11
12 @section('breadcrumb')
13     <li><a href="/">Dashboard</a></li>
14     <li class="uk-active">{{ $title }}</li>
15 @stop
16
17 @section('content')
18
19     {{ Datatable::table()
20         ->addColumn('id', 'book', 'user', 'borrowed_at', 'returned_at')
21         ->setOptions('aoColumnDefs',array(
22             array(
23                 'bVisible' => false,
24                 'aTargets' => [0]),
25             array(
26                 'sTitle' => 'Buku',
27                 'aTargets' => [1]),
28             array(
29                 'sTitle' => 'Peminjam',

```

```
30         'aTargets' => [2]),
31     array(
32         'sTitle' => 'Tanggal Pinjam',
33         'aTargets' => [3]),
34     array(
35         'sTitle' => 'Tanggal Kembali',
36         'aTargets' => [4]),
37     ))
38 ->setOptions('bProcessing', true)
39 ->setUrl(route('admin.borrow'))
40 ->render('datatable.uikit') } }

41
42 @stop
43 ....
```

---

Pada view ini kita menggunakan layout master dan menampilkan datatable yang kita buat pada method sebelumnya. Jangan lupa untuk mengubah link navigasi :

app/views/dashboard/navigation/admin.blade.php

---

```
1 {{ HTML::smartNav(route('dashboard'), 'Dashboard')}}
2 {{ HTML::smartNav(route('admin.books.index'), 'Buku')}}
3 {{ HTML::smartNav(route('admin.authors.index'), 'Penulis')}}
4 {{ HTML::smartNav(route('admin.users.index'), 'Member')}}
5 {{ HTML::smartNav(route('admin.borrow'), 'Peminjaman')}}
```

---

3. Terakhir silahkan dicek tampilan datatablenya di menu **Peminjaman**.

Buku	Peminjam	Tanggal Pinjam	Tanggal Kembali
Kupinang Engkau dengan Hamdalah	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Cinta & Seks Rumah Tangga Muslim	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>
Membingkai Surga dalam Rumah Tangga	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>

Showing 1 to 5 of 5 entries

First   Previous   **1**   Next   Last

Berhasil membuat data peminjaman

## Pengembalian Buku

Bayangkan user tidak ada koneksi internet, dia datang ke perpustakaan dan mengembalikan buku begitu saja kepada Admin. Sedangkan Admin, tidak bisa login ke akun user tersebut, tentunya karena dia tidak tahu passwordnya. Nah, fitur pengembalian dari Admin ini berguna untuk itu. Fitur ini akan mengizinkan Admin mengembalikan buku yang sedang dipinjam oleh user langsung dari halaman Admin.

1. Buat method `returnBack()` di `BorrowController` untuk mengembalikan buku.

---

app/controllers/BorrowController.php

---

```

1   ....
2 /**
3  * Mengembalikan buku yang sedang dipinjam
4  * @param int $id
5  * @return response
6 */
7 public function returnBack($id)
8 {
9     $borrow = Borrow::find($id);
10    $borrow->returned = 1;
11    $borrow->save();
12    return Redirect::route('admin.borrow')
13        ->with("successMessage", "Buku " . $borrow->book->title . " berhasil dikembalikan\
14 ");
15 }
16 ....

```

---

Pada method ini, kita mencari data di model Borrow berdasarkan id yang dikirim. Setelah ditemukan, field returned diubah menjadi 1 dan user di-*redirect* ke halaman admin.borrow (data peminjaman) dengan pesan **Buku ##### berhasil dikembalikan**.

Tambahkan route untuk method ini :

---

app/routes.php

---

```

1 ....
2 Route::group(array('prefix' => 'admin', 'before' => 'admin'), function()
3 {
4     Route::get('borrow/return/{id}', array('as'=>'admin.borrow.return', 'uses'=>'BorrowCo\
5 ntroller@returnBack'));
6 ....

```

---

2. Agar route ini bisa diakses ubah handle datatable returned\_at menjadi :

---

app/controllers/BorrowController.php

---

```

1 ....
2 ->addColumn('returned_at', function($model) {
3     if ($model->returned == 0) {
4         return link_to_route('admin.borrow.return', 'kembalikan', ['id'=>$model->id], ['c\
5 lass'=>'uk-button uk-button-small']);
6     }
7     return $model->updated_at->toDateString();
8 })
9 ....

```

---

3. Sekarang coba cek kembalikan buku dari halaman Admin, pastikan fitur ini berjalan.

The screenshot shows a web browser window titled "Data Peminjaman | Laravel". The URL is "larapus.site/admin/borrow". The page header includes "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", and "Admin Larapus". A green success message at the top states "Buku Cinta & Seks Rumah Tangga Muslim berhasil dikembalikan". Below the message, the breadcrumb navigation shows "Dashboard / Data Peminjaman". The main title is "Data Peminjaman". There is a search bar and a dropdown for "Show 10 entries". The data table has columns: "Buku", "Peminjam", "Tanggal Pinjam", and "Tanggal Kembali". The table lists five entries. The last two entries have a "kembalikan" button next to them. At the bottom, it says "Showing 1 to 5 of 5 entries" and has navigation buttons for First, Previous, Next, and Last. The current page is "1".

Buku	Peminjam	Tanggal Pinjam	Tanggal Kembali
Kupinang Engkau dengan Hamdalah	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Cinta & Seks Rumah Tangga Muslim	Shidqi Abdullah Mubarak	2014-05-29	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>
Membingkai Surga dalam Rumah Tangga	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>

Berhasil mengembalikan buku

## Filter Buku berdasarkan status Peminjaman

Sejauh ini, datatable yang kita tampilkan hanya menggunakan sedikit dari sekian banyak fitur datatable. Ada banyak lagi fitur datatable yang belum kita gunakan diantaranya custom filter, print pdf, print excel, dll. Pada bagian ini, saya akan menunjukkan fitur custom filter dengan melakukan filtering data berdasarkan buku yang sudah dikembalikan.

Untuk memfilter datatable, kita akan menggunakan method `fnFilter109`. Method ini akan memfilter kolom berdasarkan isinya. Persis seperti ketika Anda menggunakan kotak pencarian di datatable.

1. Kita perlu menambah dropdown filter untuk memilih status pengembalian buku. Untuk menambahkan-

<sup>109</sup><http://legacy.datatables.net/ref#fnFilter>

nya kita akan menggunakan jQuery. Jika belum paham cara pakai jQuery, sebaiknya pelajari dulu. Tambahkan syntax berikut pada akhir @section('content') di app/views/borrow/index.blade.php :

---

**app/view/borrow/index.blade.php**


---

```

1   ....
2   <script>
3       $(function() {
4           $('\
5               <div id="filter_status" class="dataTables_length uk-margin-left" style="display\
6               : inline-block;">\
7                   <label>Status \
8                       <select size="1" name="filter_status" aria-controls="filter_status">\
9                           <option value="all" selected="selected">Semua</option> \
10                          <option value="returned">Sudah Dikembalikan</option> \
11                          <option value="notReturned">Belum Dikembalikan</option> \
12                      </select> \
13                  </label> \
14              </div> \
15          ).insertAfter('#DataTables_Table_0_length');
16      });
17  </script>
18  ....

```

---

Menggunakan method `insertAfter` dari `jQuery110` saya menambahkan dropdown `filter_status` dengan 3 pilihan Semua, Sudah Dikembalikan dan Belum Dikembalikan setelah dropdown untuk memilih jumlah record yang ditampilkan. Silahkan cek kembali datatable data peminjaman, pastikan dropdown baru ini muncul :

2. Kita perlu menentukan data yang akan difilter. Berdasarkan dokumentasi tentang `fnFilter`, method ini dapat memfilter berdasarkan nilai tertentu. Untuk itu, pada kolom **Tanggal Pengembalian** saya akan menggunakan `data attributes111` di HTML dengan nama `data-returned` yang akan berisi `true` atau `false` tergantung status pengembalian. Data ini yang akan saya cari pada method `fnFilter`.

Mari kita tambahkan `data-returned` pada handle datatable untuk kolom `returned_at` :

---

<sup>110</sup><http://api.jquery.com/insertafter/>

<sup>111</sup>[http://docs.webplatform.org/wiki/html/attributes/data-\\*](http://docs.webplatform.org/wiki/html/attributes/data-*)

---

app/controllers/BorrowController.php

---

```

1   ....
2   ->addColumn('returned_at', function($model) {
3       if ($model->returned == 0) {
4           return link_to_route('admin.borrow.return', 'kembalikan', ['id'=>$model->id], ['c\
5           lass'=>'uk-button uk-button-small', 'data-returned'=>"false"]);
6       }
7       return '<span data-returned="true">' . $model->updated_at->toDateString() . '</span>';
8   })
9   ....

```

---

Karena data-returned harus berada dalam sebuah elemen HTML, maka pada buku yang sudah dikembalikan saya mengapitnya dengan elemen span. Sedangkan pada buku yang belum dikembalikan saya cukup menambah pada array yang sudah ada.

- Pada view, kita perlu menambahkan ‘callback’ ketika dropdown status berubah. Untuk memfilter data peminjaman, kita juga perlu mendapatkan class yang di-generate oleh package Datatable. Oleh karena itu, kita akan memisah syntax inisialisasi, render dan mendapatkan class. Untuk mendapatkan class kita akan menggunakan method getClass. Method ini tidak ada di dokumentasi, tetapi saya temukan di [source code package](#)<sup>112</sup>.

Ubah syntax untuk menghasilkan datatable di view menjadi :

---

app/views/borrow.index.php

---

```

1   ....
2   <?php
3   $datatable = Datatable::table()
4       ->addColumn('id', 'book', 'user', 'borrowed_at', 'returned_at')
5       ->setOptions('aoColumnDefs',array(
6           array(
7               'bVisible' => false,
8               'aTargets' => [0]),
9           array(
10              'sTitle' => 'Buku',
11              'aTargets' => [1]),
12           array(
13              'sTitle' => 'Peminjam',
14              'aTargets' => [2]),
15           array(
16              'sTitle' => 'Tanggal Pinjam',
17              'aTargets' => [3]),
18           array(
19              'sTitle' => 'Tanggal Kembali',
20              'aTargets' => [4]),
21       ))

```

---

<sup>112</sup><https://github.com/Chumper/Datatable/blob/master/src/Chumper/Datatable/Table.php>

```

22      ->setOptions('bProcessing', true)
23      ->setUrl(route('admin.borrow'));
24  ?>
25
26  {{ $datatable->render() }}
27  ....

```

---

Terlihat disini, kita membuat variable \$datatable untuk menginisialisasi datatable. Proses pembuatan datatable kita pisah baris tersendiri dengan memanggil metho render.

Untuk membuat callback ketika dropdown status berubah, tambahkan baris ini di dalam tag script yang telah dibuat pada tahap sebelumnya:

#### app/views/borrow.index.php

---

```

1  <script>
2      $(function() {
3          ...
4          $('select[name="filter_status"]').change(function() {
5              var $oTable = $('.{{ $datatable->getClass() }}').dataTable();
6              switch (this.value) {
7                  case 'all' :
8                      $oTable.fnFilter('');
9                      break;
10                 case 'returned' :
11                     $oTable.fnFilter('data-returned="true"');
12                     break;
13                 case 'notReturned' :
14                     $oTable.fnFilter('data-returned="false"');
15                     break;
16             }
17         });
18     });
19 </script>

```

---

Menggunakan method [change dari jQuery<sup>113</sup>](#) kita mengecek ketika isian dari filter\_status berubah. Kemudian mengecek nilainya satu persatu :

- all akan menghilangkan semua filter
- returned akan memfilter baris yang memiliki isian data-returned="true"
- not-returned akan memfilter baris yang memiliki isian data-returned="false"

Ketika isian berubah, kita melakukan seleksi class yang dihasilkan dari syntax \$datatable->getClass() pada class inilah kita melakukan fnFilter.

4. Sekarang, silahkan cek filter yang telah Anda buat, pastikan semua filternya berjalan.

---

<sup>113</sup><http://api.jquery.com/change/>

The screenshot shows a web browser window titled "Data Peminjaman | Laravel". The URL is "larapus.site/admin/borrow". The page header includes "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", and "Admin Larapus". Below the header, the breadcrumb navigation shows "Dashboard / Data Peminjaman".

# Data Peminjaman

Show 10 entries Status Semua Search:

Buku	Peminjam	Tanggal Pinjam	Tanggal Kembali
Kupinang Engkau dengan Hamdalah	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Cinta & Seks Rumah Tangga Muslim	Shidqi Abdullah Mubarak	2014-05-29	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>
Membingkai Surga dalam Rumah Tangga	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>

Showing 1 to 5 of 5 entries

First Previous **1** Next Last

Tanpa filter

The screenshot shows a web application titled "Data Peminjaman | Laravel" running on "larapus.site/admin/borrow". The interface includes a navigation bar with links to "LaraPus", "Dashboard", "Buku", "Penulis", "Member", "Peminjaman", and "Admin Larapus". Below the navigation is a breadcrumb trail: "Dashboard / Data Peminjaman".

The main content area is titled "Data Peminjaman". It features a table with the following data:

Buku	Peminjam	Tanggal Pinjam	Tanggal Kembali
Kupinang Engkau dengan Hamdalah	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-20	2014-05-29
Cinta & Seks Rumah Tangga Muslim	Shidqi Abdullah Mubarak	2014-05-29	2014-05-29

Below the table, a message indicates: "Showing 1 to 3 of 3 entries (filtered from 5 total entries)". Navigation buttons for "First", "Previous", "Next", and "Last" are present, along with a page number "1".

Filter buku sudah dikembalikan

Buku	Peminjam	Tanggal Pinjam	Tanggal Kembali
Jalan Cinta Para Pejuang	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>
Membingkai Surga dalam Rumah Tangga	Shidqi Abdullah Mubarak	2014-05-29	<button>kembalikan</button>

Showing 1 to 2 of 2 entries (filtered from 5 total entries)

First    Previous    **1**    Next    Last

Filter buku belum dikembalikan

Jika pembaca membaca lebih lanjut, method `fnFilter` mengizinkan untuk filter berdasarkan kolom tertentu. Tapi saya masih belum berhasil mengimplementasikan filter dengan kolom tertentu ini. Nampaknya fitur ini belum di support di package yang kita gunakan. Jika Anda berhasil menggunakan filter per kolom ini, saya sangat senang jika bisa dijelaskan caranya. :)

## Bonus : Penanganan Error

Laravel memudahkan kita untuk menampilkan pesan ke user ketika aplikasi mengalami error. Sebagai contoh, jika kita mencoba mengunjungi route yang tidak ada misalnya `http://larapus.site/guengaada` maka aplikasi akan menampilkan `ErrorException` :

The screenshot shows a browser window with the URL `larapus.site/guenggaada`. The title bar says "Whoops! There was an error". The main content is a Symfony exception page for a `NotFoundHttpException`. The stack trace on the left shows the following calls:

12. `Symfony\Component\HttpKernel\.../vendor/laravel/-framework/src/Illuminate/-Routing/-RouteCollection.php:146`
11. `Illuminate\Routing\RouteCollection->match` `.../vendor/laravel/-framework/src/Illuminate/-Routing/Router.php:1021`
10. `Illuminate\Routing\Router->findRoute` `.../vendor/laravel/-framework/src/Illuminate/-Routing/Router.php:989`
9. `Illuminate\Routing\Router->dispatchToRoute` `.../vendor/laravel/-framework/src/Illuminate/-Routing/Router.php:968`
8. `Illuminate\Routing\Router->dispatch` `.../vendor/laravel/-framework/src/Illuminate/-Foundation/-Application.php:738`

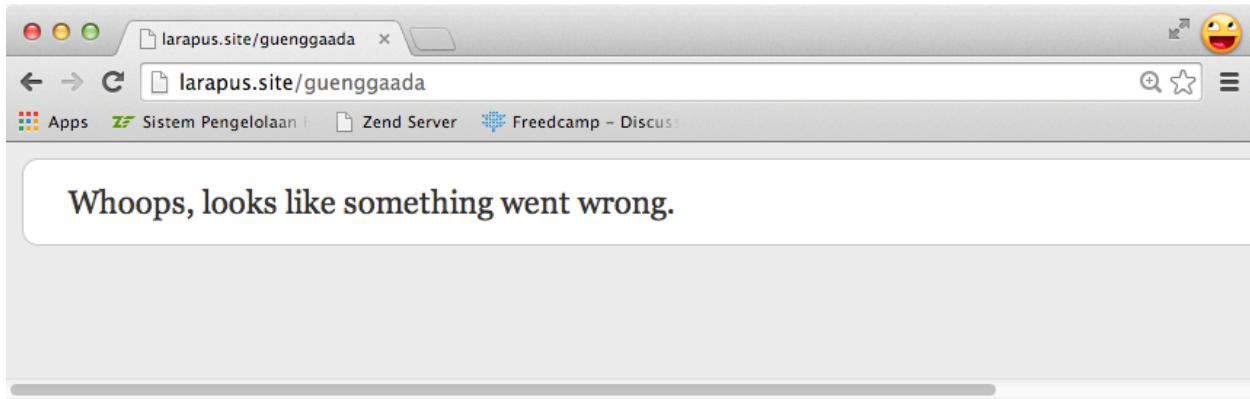
The right panel shows the source code for the `RouteCollection.php` file at line 146, where a `NotFoundHttpException` is thrown. Below the code, it says "No comments for this stack frame."

### Server/Request Data

REDIRECT_STATUS	200
HTTP_HOST	larapus.site
HTTP_CONNECTION	keep-alive

Error Exception, terlalu mainstream

Tampilan seperti ini biasanya bikin kaget user dan tidak aman untuk aplikasi (karena syntax jadi terlihat). Untuk mengatasinya, pada saat aplikasi sudah di upload ke server produksi pastikan mengubah isian debug menjadi `false` di `app/config/app.php`. Sehingga error yang muncul menjadi :



Tampilan error default

Tampilan ini bisa kita rubah dengan cara :

1. Membuat view khusus, misalnya di `app/views/errors/default.blade.php` :

`app/views/errors/default.blade.php`

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Perpustakaan Online dengan Framework Laravel</title>
5          <link rel="stylesheet" href="{{ asset('packages/uikit/css/uikit.almost-flat.min.c\
6 ss') }}" />
7          <link rel="stylesheet" href="{{ asset('css/app.css') }}" />
8          <script src="{{ asset('packages/uikit/js/uikit.min.js') }}"></script>
9      </head>
10     <body>
11         <div class="uk-container uk-margin-top">
12             <div class="uk-alert uk-alert-danger uk-text-center">
13                 <h1 class="uk-heading-large">:(</h1>
14                 <p>Kami tidak menemukan data yang Anda cari.</p>
15                 <a href="/" class="uk-button uk-button-large">Kembali ke Larapus</a>
16             </div>
17         </div>
18     </body>
19 </html>
```

2. Tampilkan view ini sebagai default tampilan error dengan menambahkan di `app/start/global.php` di bagian Application Error Handler sehingga syntax `App::error` berubah menjadi :

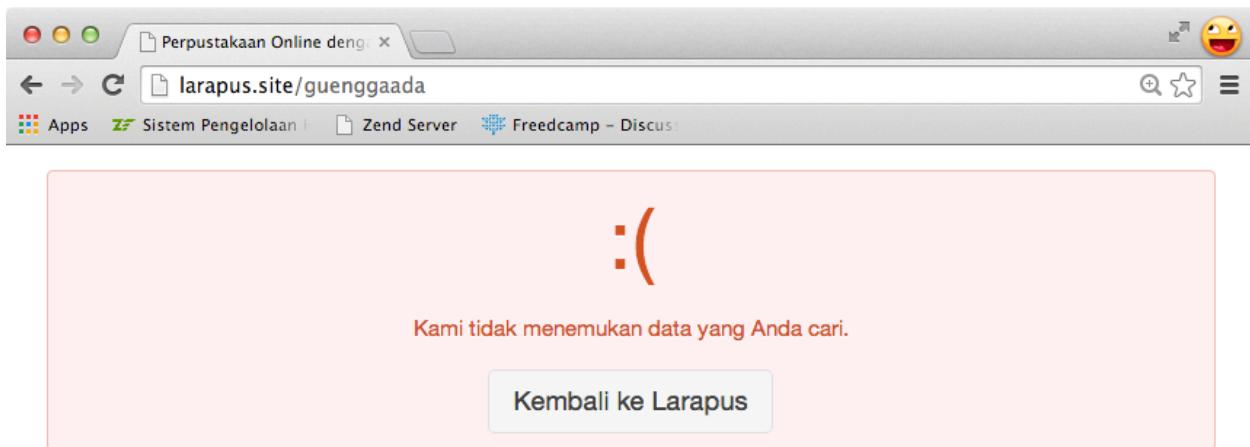
**app/start/global.php**

```

1  ....
2  App::error(function(Exception $exception, $code)
3  {
4      Log::error($exception);
5      if(!Config::get('app.debug')) return Response::view('errors.default', array(), $code);
6  });
7  ....

```

Syntax ini akan membuat halaman custom kita tampil ketika mode debug di set false. Sekarang refresh kembali halaman error yang tadi. Tampilan seperti ini yang akan muncul :

**Tampilan error custom**

Jangan lupa, jika Anda telah mematikan mode `debug`, detail dari error masih bisa diakses di `app/storage/logs/laravel.log`.

Banyak cara lain agar Anda bisa menangani error ini dengan lebih baik, misalnya dengan menangkap `Exception` tertentu dan menampilkan error sesuai exception, mengirim email ke Admin jika telah terjadi error, dll. Untuk lebih lengkapnya, kunjungi [dokumentasi resmi<sup>114</sup>](#).

## Ringkasan

Pada hari 6 ini kita telah melengkapi fitur Admin dari Larapus. Adapun hal yang telah kita pelajari yaitu :

- Penggunaan Register pada Sentry
- Penggunaan Activation pada Sentry
- Penggunaan Password Reminder pada Sentry

<sup>114</sup><http://laravel.com/docs/errors>

- Penggunaan reCaptcha
- Konfigurasi dan pengiriman Email
- Custom table pada Eloquent
- Log query sql
- Eager Loading
- Custom Filter pada Datatable
- Error Handling

Tugas : Cobalah Anda kunjungi halaman forgot password dan signup, kedua halaman ini masih bisa dikunjungi jika Anda telah login. Buatlah filter agar hanya user yang belum login yang bisa mengakses route tersebut, misalnya dengan nama guest. Anda bebas meletakkan filternya, tapi saran saya gunakan route grouping untuk mengelompokan filter pada route yang hanya bisa diakses oleh guest. Sebagai petunjuk, kita pernah melakukan filtering ini untuk halaman root ('/') dan mengarahkan user ke dashboard jika ia telah login.

# Hari 7 : Deploy Aplikasi

Di hari terakhir ini, kita akan mengupload Larapus yang telah kita buat ke web. Ada banyak cara untuk mengupload aplikasi yang dibangun dengan Laravel ke internet. Diantaranya ke shared hosting, fortrabbit, VPS, dll. Bahkan untuk memudahkan deployment Laravel menyediakan fitur berbayar bernama [forge<sup>115</sup>](#) yang akan memudahkan deployment aplikasi di Linode, DigitalOcean, AWS, & Rackspace.

Pada pembahasan di buku ini, saya akan menunjukkan bagaimana mengupload Laravel ke shared hosting. Agar pembaca bisa mengikuti, saya akan menggunakan shared hosting gratisan yaitu [idhostinger<sup>116</sup>](#). Sebelum memulai, silahkan Anda daftar terlebih dahulu.



idhostinger

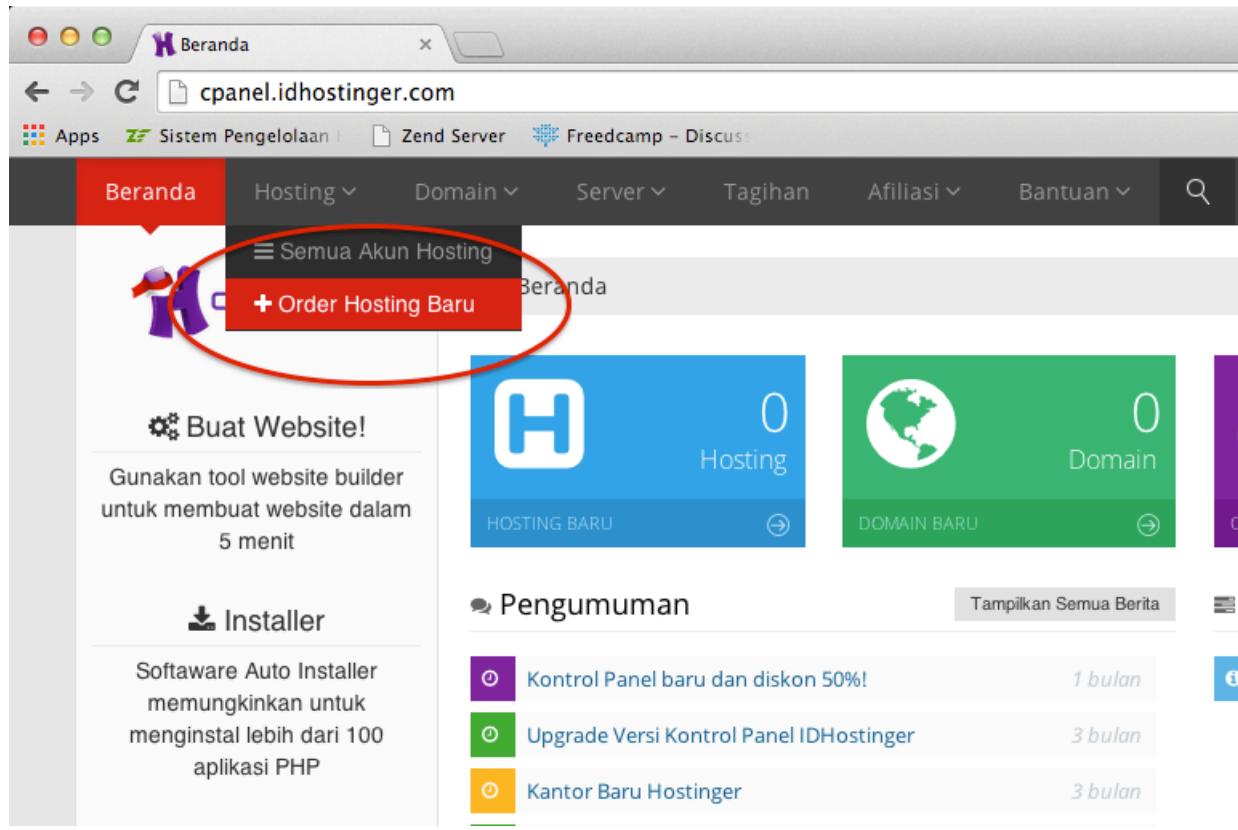
## Konfigurasi Hosting

Setelah Anda berhasil daftar di idhostinger, mari kita konfigurasi untuk deployment aplikasi Larapus.

1. Buatlah hosting baru dengan pilih **Hosting > Order Hosting** :

<sup>115</sup><https://forge.laravel.com/>

<sup>116</sup><https://www.idhostinger.com>



### Order Hosting

2. Pada halaman pemilihan paket hosting, pilih yang gratis :

## Order Hosting Baru

The screenshot shows a web interface for selecting a hosting package. At the top, there is a breadcrumb navigation: Beranda > Hosting > Pilih Paket. Below this, the title "Pilih Paket Hosting" is displayed. Three hosting plans are listed side-by-side:

- Gratis**: Price Rp. 0,00. An "Order" button is highlighted with a red circle. The plan includes:
  - Disk Space 2000 MB
  - Bandwidth 100 GB
  - 2 Database MySQL
  - 2 Akun Email
  - Website Builder
  - Script Autoinstaller
- Premium**: Price Rp. 28.000,00. The plan includes:
  - Disk Space Unlimited!
  - Bandwidth Unlimited!
  - Database MySQL Unlimited
  - Akun E-mail Unlimited
  - Website Builder
  - Script Autoinstaller
  - Gratis Pendaftaran Domain!
  - Akses Penuh SSH
- Bisnis**: Price Rp. 66.000,00. The plan includes:
  - Disk Space Unlimited!
  - Bandwidth Unlimited!
  - Database MySQL Unlimited
  - Akun E-mail Unlimited
  - Website Builder
  - Script Autoinstaller
  - Gratis Pendaftaran Domain!
  - Akses Penuh SSH
  - IP Dedicated

### Paket Gratis

3. Pada halaman pembuatan nama domain, pilih tipe subdomain. Masukkan juga nama domain (saya menggunakan larapus.hol.es) dan password yang akan digunakan. Password ini akan kita gunakan untuk akses FTP. Klik **Lanjutkan**.

Beranda > Hosting > Pilih Paket > Setup Hosting

### ≡ Order Hosting Baru "Gratis" - Langkah 2 dari 3

1 ✓ Ubah Paket      2 Setup Hosting      3 Ringkasan Order

Masukkan domain dan password

Ubah Tipe Domain: Subdomain

Subdomain: larapus .holes ▾

Password: ..... Hasilkan

Konfirmasi Password: ..... Masukkan password kembali.

Lanjutkan ⊞

Bikin subdomain

4. Pada form selanjutnya, masukkan captcha yang muncul dan centang **Saya setuju bla bla bla**. Klik **Order**.

≡ Order Hosting Baru "Gratis" - Langkah 3 dari 3

1 ✓ Ubah Paket      2 ✓ Setup Hosting      3 Ringkasan Order

Konfirmasi order Anda

Paket Hosting:	Gratis
Domain:	larapus.hol.es
Harga:	Rp. 0,00
Captcha:	 n9ho6
<input checked="" type="checkbox"/> Saya setuju dengan <a href="#">ketentuan penggunaan layanan</a>	
<a href="#">Kembali</a> <a href="#" style="background-color: green; color: white; border-radius: 5px;">Order</a>	

Bikin subdomain

5. Pastikan Anda berhasil membuat subdomain.

## Akun Hosting

Beranda > Hosting

Akun larapus.hol.es telah berhasil dengan baik.

**Daftar Akun Hosting**

Domain	Paket Hosting	Kadaluarsa Pada	Status
larapus.hol.es	Gratis	-	Aktif

Order Hosting Baru →

Berhasil membuat subdomain

- Selanjutnya kita akan menambahkan database untuk subdomain ini, klik pada tanda + di kiri subdomain, kemudian pilih **Kelola**.

## Akun Hosting

Beranda > Hosting

**Daftar Akun Hosting**

Domain	Paket Hosting	Kadaluarsa Pada	Status
larapus.hol.es	Gratis	-	Aktif

Kelola

Website builder

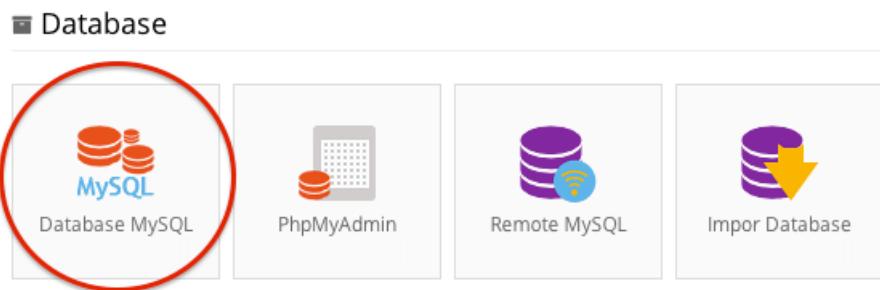
Auto Installer

Akun Email

← Sebelumnya 1 Selanjutnya →

### Kelola Subdomain

- Pada halaman selanjutnya, cari menu database dan pilih \*MySQL Database untuk menambah database.



#### Mengelola Database

8. Lengkapi data database yang akan dibuat pada form yang muncul. Catat username, nama database, dan password yang digunakan.

### Database MySQL

Buat database MySQL Baru dan lihat daftar user dan Database MySQL

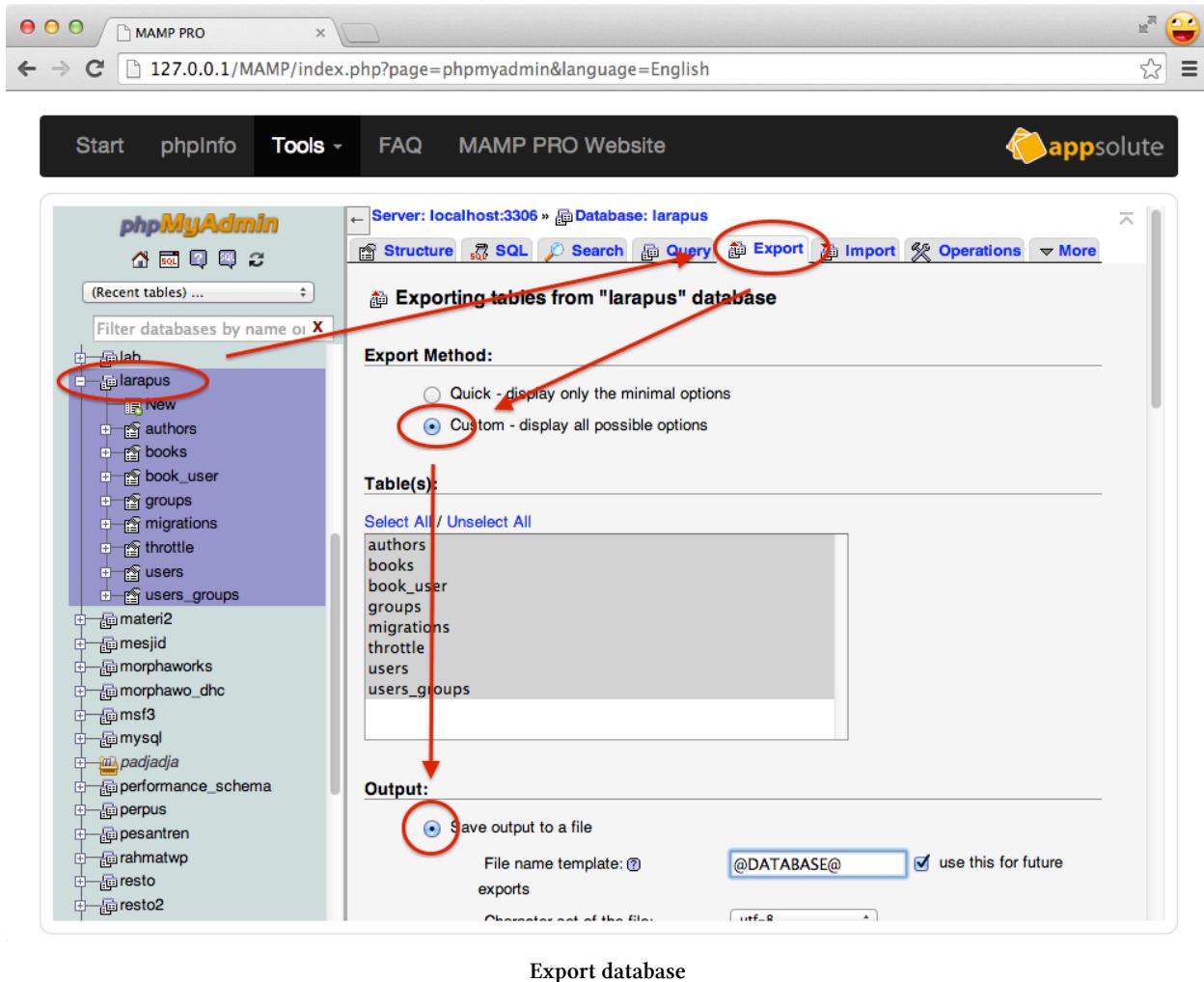
A screenshot of a web form titled 'Buat Database dan User Database MySQL'. The form has several input fields:

- Nama Database MySQL: u107618895\_lara
- Username MySQL: u107618895\_lara
- Password: ..... (with a 'Hasilkan' button next to it)
- Masukkan Password Kembali: ..... (repeated password field)

At the bottom, there is a blue 'Buat' button with a checkmark icon, which is circled in red.

#### Setup Database

9. Kita akan mengimpor database Larapus. Untuk itu, silahkan export terlebih dahulu database Larapus dari server mysql yang kita pakai. Disini saya menggunakan phpmyadmin :



The screenshot shows the MAMP PRO interface with the URL `127.0.0.1/MAMP/index.php?page=phpmyadmin&language=English`. The main content is the phpMyAdmin configuration page for the 'larapus' database. On the left, there's a sidebar with a tree view of databases: lab, larapus, materi2, mesjid, morphaworks, morphawo\_dhc, msf3, mysql, padjadja, performance\_schema, perpus, pesantren, rahmatwp, resto, and resto2. The 'larapus' database is selected. The right panel contains configuration options:

- Server: localhost:3306 > Database: larapus
- Structure, SQL, Search, Query, Export, Import, Operations, More tabs
- Instead of INSERT statements, use:
  - INSERT DELAYED statements
  - INSERT IGNORE statements
- Function to use when dumping data: **INSERT**
- Syntax to use when inserting data:
  - include column names in every INSERT statement  
Example: `INSERT INTO tbl_name (col_A,col_B,col_C) VALUES (1,2,3)`
  - insert multiple rows in every INSERT statement  
Example: `INSERT INTO tbl_name VALUES (1,2,3), (4,5,6), (7,8,9)`
  - both of the above  
Example: `INSERT INTO tbl_name (col_A,col_B) VALUES (1,2,3), (4,5,6), (7,8,9)`
  - neither of the above  
Example: `INSERT INTO tbl_name VALUES (1,2,3)`
- Maximal length of created query: **50000**
- Dump binary columns in hexadecimal notation (for example, "abc" becomes `0x616263`)
- Dump TIMESTAMP columns in UTC (enables TIMESTAMP columns to be dumped and reloaded between servers in different time zones)

A red circle highlights the 'Go' button at the bottom of the configuration panel.

#### Export database

10. Kembali ke halaman database idhostinger, pilih phpmyadmin pada database yang telah kita buat.

Daftar Database dan User Database MySQL

	Database MySQL	User MySQL	Host MySQL	Penggunaan Disk, MB
u107618895_lara	u107618895_lara	mysql.idhostinger.com	0.02	

**Action Buttons:**

- Hapus (Red)
- Perbaiki (Blue)
- Penggunaan (Green)
- Backup (Brown)
- Ubah password (Grey)
- Ubah Hak Akses (Orange)
- PhpMyAdmin (Yellow, circled in red)

← Sebelumnya | 1 | Selanjutnya →

### Mengakses phpmyadmin

11. Pada halaman phpmyadmin, pilih import > choose file > format [sql] > Go.

Importing into the database "u107618895\_lara"

**File to Import:**  
File may be compressed (gzip, bzip2, zip) or uncompressed.  
A compressed file's name must end in **[format].[compression]**. Example: **:sql.zip**

Browse your computer:  (Max: 8,192KiB)

Character set of the file:

**Partial Import:**  
 Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be good way to import large files, however it can break transactions.)*

Number of rows to skip, starting from the first row:

**Format:**  (circled in red)

**Format-Specific Options:**

SQL compatibility mode:   Do not use AUTO\_INCREMENT for zero values

**Go** (button circled in red)

Import database

12. Tunggu hingga import database berhasil.

The screenshot shows the phpMyAdmin interface for a database named "u107618895\_lara". The main navigation bar includes tabs for Structure, SQL, Search, Query, Export, Import, Operations, and Routines. Below the navigation bar, a message box displays "Import has been successfully finished, 27 queries executed. (larapus.sql)". On the left sidebar, there is a list of tables: authors, books, book\_user, groups, migrations, throttle, users, and users\_groups. A "Create table" button is also present. The central area contains sections for "File to Import", "Partial Import", and "Format". Under "File to Import", there is a note about compressed files and a "Choose File" button. Under "Partial Import", there is a checkbox for allowing interruptions and a field for skipping rows. Under "Format", there is a dropdown menu set to "SQL". At the bottom of the page, a success message "Berhasil import database" is displayed.

## Deploy Aplikasi

Untuk mengupload aplikasi ke server idhostinger, kita akan menggunakan ftp. Client FTP yang saya gunakan adalah Transmit. Di Windows, Anda bisa menggunakan FileZilla. Username ftp yang akan digunakan bisa dicek di halaman **File > Akses FTP**. Password yang digunakan adalah password yang sama ketika Anda membuat domain ini.

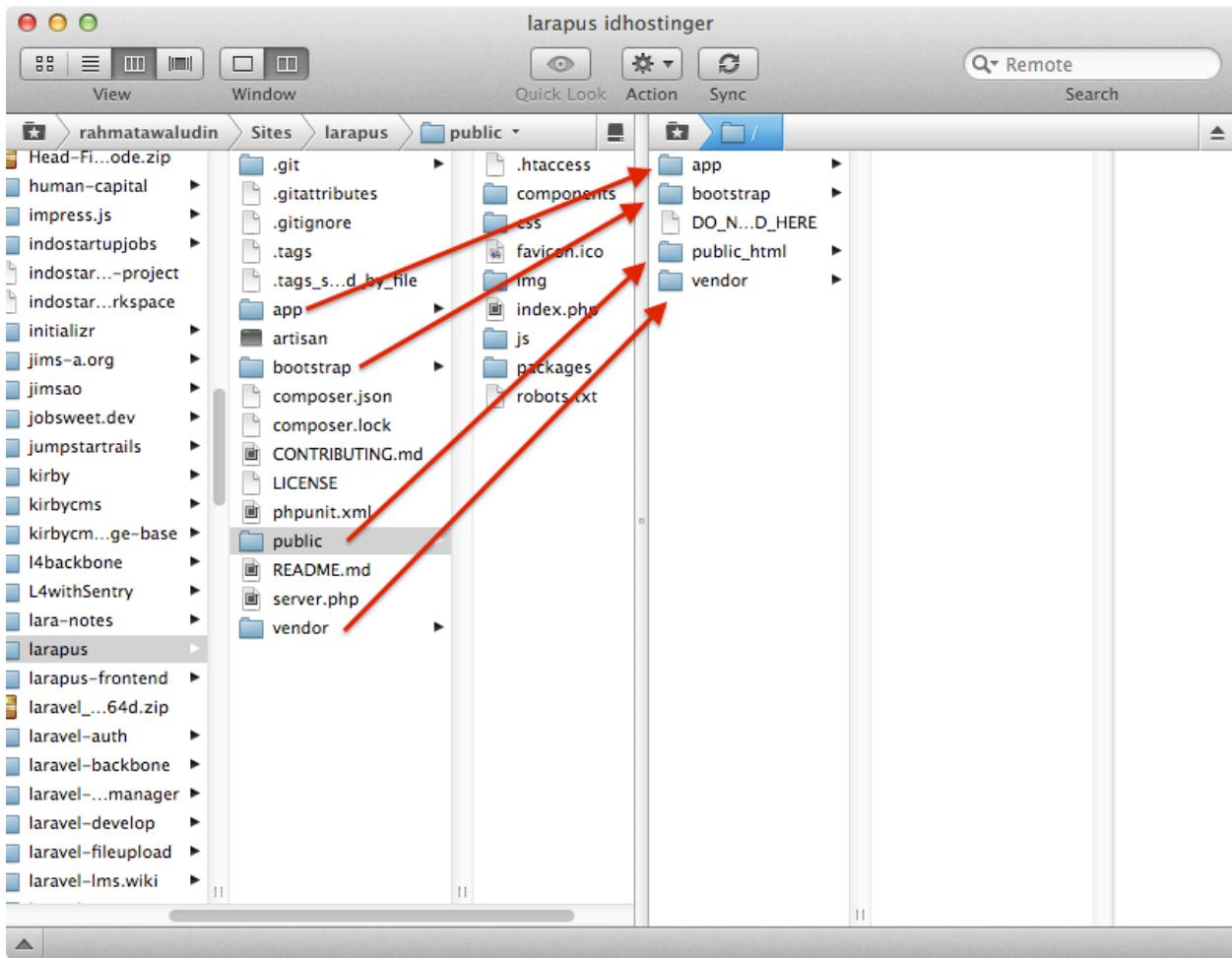
The screenshot shows the idhostinger cPanel interface. The top navigation bar includes links for Beranda, Hosting (highlighted in red), Domain, Server, Tagihan, Afiliasi, Bantuan, a search bar, and user profile icons. The main content area is titled "Akses FTP" with the subtitle "Informasi tentang layanan FTP". The breadcrumb navigation shows: Beranda > Hosting > larapus.hol.es > File > Akses FTP. The "Akses FTP" page displays the following information:

Akses FTP	
FTP host	ftp.larapus.hol.es
FTP IP	31.220.16.104
Port FTP	21
Username FTP	u107618895
Password FTP	*****
File akan diupload pada folder	public_html
Lupa Password FTP?	<a href="#">Ubah password akun</a>
Client FTP yang Disarankan	SmartFTP atau FileZilla

#### Credential FTP

Setelah Anda berhasil menghubungkan client FTP. Ikuti langkah berikut ini.

1. Ubah konfigurasi database di app/config/database.php agar sesuai dengan credential dari idhostinger.
2. Untuk keamanan, matikan mode debug dengan mengubah isian debug menjadi false di app/config/app.php
3. Upload folder app, bootstrap, dan vendor ke root direktori. Upload juga semua isi folder public ke public\_html.



Upload folder

4. Setelah semua folder di-upload, cek domain Larapus yang baru saja Anda buat.

LaraPus

Login | Daftar

# Daftar Buku

Show 10 entries

Judul	Jumlah	Stok	Penulis	
Kupinang Engkau dengan Hamdalah	3	3	Mohammad Fauzil Adhim	<a href="#">Pinjam</a>
Jalan Cinta Para Pejuang	2	1	Salim A. Fillah	<a href="#">Pinjam</a>
Membingkai Surga dalam Rumah Tangga	4	3	Aam Amiruddin	<a href="#">Pinjam</a>
Cinta & Seks Rumah Tangga Muslim	3	3	Aam Amiruddin	<a href="#">Pinjam</a>

Showing 1 to 4 of 4 entries

First Previous 1 Next Last

Berhasil mengupload Larapus ke idhostinger

## Ringkasan

Teknik deployment yang kita lakukan ini adalah yang paling sederhana dan murah. Jika Anda hendak mengembangkan aplikasi dengan skala besar, saya sarankan Anda untuk mempelajari penggunaan VPS.

Jika Anda malas melakukan setup server VPS, saya sarankan Anda untuk mempelajari beberapa penyedia layanan SaaS diantaranya fortrabbit, box, dll. Tentunya, untuk menggunakan beberapa penyedia layanan SaaS, pemahaman tentang GIT sangat diperlukan. Untuk memudahkan testing deployment aplikasi di Internet, Anda juga dapat menggunakan layanan [bowery](#)<sup>117</sup>.

<sup>117</sup><http://bowery.io>

# Penutup

Akhirnya selesai juga Anda mempelajari buku ini. Tidak terasa 7 hari telah dilalui. Memang masih banyak fitur Laravel yang belum kita bahas. Namun, saya harap pengalaman membangun aplikasi Larapus ini dapat menjadi bekal bagi Anda untuk mengembangkan aplikasi yang lebih besar.

Beberapa hal yang belum kita pelajari :

- HMVC/Modular Development
- Pembuatan Package
- Service Provider
- Facade
- Localization/fitur multi bahasa
- Cache
- Queue
- Session
- Testing
- Pagination
- dan masih banyak lagi

## Belajar lagi!

Saya harap Anda tidak puas dan terus belajar tentang framework ini. Beberapa sumber belajar Laravel yang bisa Anda manfaatkan :

- [http://laravel.com/docs<sup>118</sup>](http://laravel.com/docs) : Dokumentasi resmi framework Laravel
- [http://laravel.com/api<sup>119</sup>](http://laravel.com/api) : Dokumentasi API resmi framework Laravel
- [http://laravel.io/forum<sup>120</sup>](http://laravel.io/forum) : Forum resmi framework Laravel
- [http://laravel.io/chat<sup>121</sup>](http://laravel.io/chat) : Livechat resmi framework Laravel
- [http://laracasts.com<sup>122</sup>](http://laracasts.com) : Koleksi video tutorial Laravel dari Jeffrey Way
- [http://www.laravel-tricks.com<sup>123</sup>](http://www.laravel-tricks.com) : Koleksi tips seputar penggunaan Laravel
- [https://www.facebook.com/groups/laravel<sup>124</sup>](https://www.facebook.com/groups/laravel) : Grup unofficial Laravel Indonesia
- [https://medium.com/laravel-indonesia<sup>125</sup>](https://medium.com/laravel-indonesia) : Koleksi artikel Laravel berbahasa Indonesia. Anda dapat submit artikel tentang Laravel di web ini. Saya juga suka menulis disini.

---

<sup>118</sup><http://laravel.com/docs>

<sup>119</sup><http://laravel.com/api>

<sup>120</sup><http://laravel.io/forum>

<sup>121</sup><http://laravel.io/chat>

<sup>122</sup><http://laracasts.com>

<sup>123</sup><http://www.laravel-tricks.com>

<sup>124</sup><https://www.facebook.com/groups/laravel/>

<sup>125</sup><https://medium.com/laravel-indonesia>

Tips lain belajar, baca source code Laravel. Serius. Saya sendiri, kalau kurang paham beberapa fitur di Laravel, sering buka source codenya langsung.. :D

## Terima kasih

Teriring do'a, saya ucapan terima kasih bagi Anda yang telah membeli buku ini. Dengan membeli buku ini, Anda telah membantu saya untuk tetap konsisten berbagi ilmu di bidang web development. Anda juga telah membantu memberi nafkah untuk keluarga saya dengan nafkah yang halal. Saya yakin, rezeki dari Tuhan tidak akan pernah salah sasaran. Setiap orang pasti telah dikaruniai rezeki sesuai dengan kadar kebijaksanaan dalam dirinya. Semoga ilmu yang diperoleh berkah.. :)

*NB : Bagi yang masih belum beli buku ini atau yang masih copy pdf dari teman, saya do'akan agar segera dicukupkan rezekinya supaya bisa membeli (lisensi) buku ini. Supaya ilmunya berkah.. :)*

Sekian,

Salam hangat dari saya, istri dan putra saya untuk Anda.