


XI_RPL 1

Indahnya Mempelajari Laravel

Dicka Ananda A.N

Daftar Isi

Bab 1 Framework Laravel

1. Pengenalan Laravel
2. Kelebihan Dan Kekuragam
3. Framework Laravel
4. Mengapa Menggunakan Laravel?
- 5.7 Alasan Menggunakan Framework Laravel dibanding native PHP

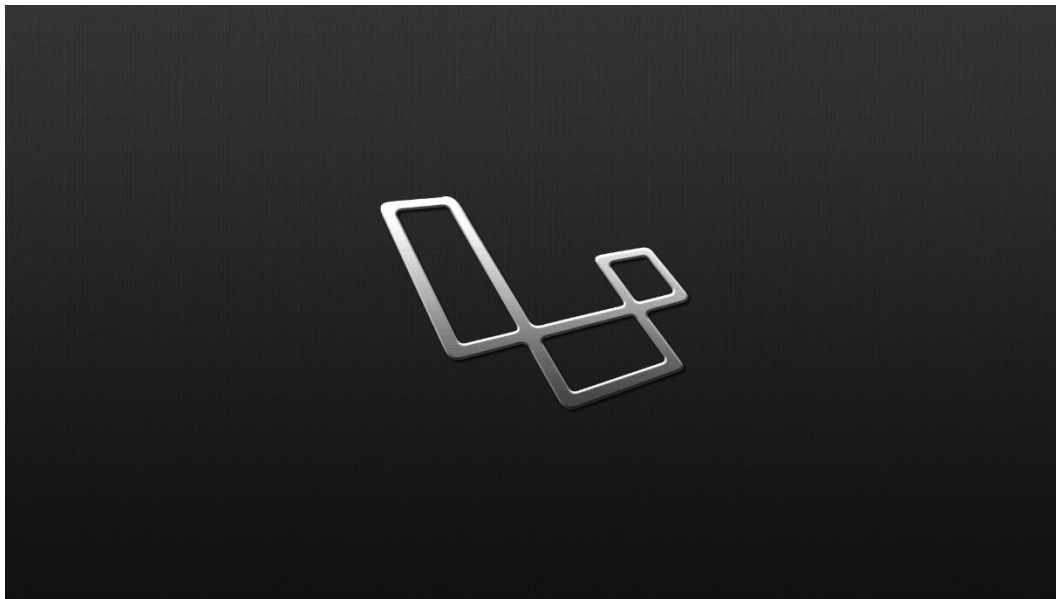
Bab 2 Fitur-Fitur Laravel

Bab 3 Laravel & PHP 5

JSON	
PHP5 Anonymous Function/Lambda & Closure	
PHP5 Autoloader	
PHP5 Abstract dan Interfaces	
PHP5 Namespace	
PHP5 Traits	
PHP5 Reflection	
Composer	

Bab 4 Instalasi & Konfigurasi Laravel

BAB 1. Pengenalan Laravel



1. Pengenalan Framework

Framework adalah kerangka kerja. Framework juga dapat diartikan sebagai kumpulan script (terutama class dan function) yang dapat membantu developer/programmer dalam menangani berbagai masalah-masalah dalam pemrograman seperti koneksi ke database, pemanggilan variabel, file, dll sehingga developer lebih fokus dan lebih cepat dalam membangun aplikasi. Bisa juga dikatakan Framework adalah komponen pemrograman yang siap re-use kapan saja, sehingga programmer tidak harus membuat skrip yang sama untuk tugas yang sama. Misalkan saat membuat aplikasi web berbasis ajax yang setiap kali harus melakukan XMLHttpRequest, maka Xajax telah mempermudahnya dengan menciptakan sebuah objek khusus yang siap digunakan untuk operasi Ajax berbasis PHP.

Secara sederhana bisa dijelaskan bahwa framework adalah kumpulan fungsi (libraries), maka seorang programmer tidak perlu lagi membuat fungsi-fungsi (biasanya disebut kumpulan library) dari awal, programmer tinggal memanggil kumpulan library atau fungsi yang sudah ada di dalam framework, dan cara menggunakan fungsi-fungsi itu sudah ditentukan sebelumnya oleh framework. Beberapa contoh fungsi-fungsi standar yang telah tersedia dalam suatu framework adalah fungsi paging, enkripsi, email, SEO, session, security, kalender, bahasa, manipulasi gambar, grafik, tabel bergaya zebra, validasi, upload, captcha, proteksi terhadap XSS(XSSfiltering), template, kompresi, XML dan lain-lain.

Beberapa fitur yang terdapat di Laravel :

- Bundles, yaitu sebuah fitur dengan sistem pengemasan modular dan tersedia beragam di aplikasi.
- Eloquent ORM, merupakan penerapan PHP lanjutan menyediakan metode internal dari pola “active record” yang mengatasi masalah pada hubungan objek database.
- Application Logic, merupakan bagian dari aplikasi, menggunakan controller atau bagian Route.
- Reverse Routing, mendefinisikan relasi atau hubungan antara Link dan Route.
- Restful controllers, memisahkan logika dalam melayani HTTP GET and POST.
- Class Auto Loading, menyediakan loading otomatis untuk class PHP.
- View Composer, adalah kode unit logikal yang dapat dieksekusi ketika view sedang loading.
- IoC Container, memungkinkan obyek baru dihasilkan dengan pembalikan controller.
- Migration, menyediakan sistem kontrol untuk skema database.
- Unit Testing, banyak tes untuk mendeteksi dan mencegah regresi.
- Automatic Pagination, menyederhanakan tugas dari penerapan halaman.

2. Kelebihan dan Kekurangan Framework

Kelebihan pertama dengan adanya framework akan lebih mempermudah memahami mekanisme kerja dari sebuah aplikasi. Ini akan sangat membantu proses pengembangan system yang dilakukan oleh team. Semua anggota diwajibkan untuk memahami dari pola kerja framework tersebut, selebihnya anggota team hanya mempelajari proses bisnis yang dikehendaki oleh system untuk kemudian di tuangkan kedalam framework tersebut. Dalam artian setiap orang harus mempunyai metode yang sama dalam menyelesaikan aplikasi tersebut.

Kedua dengan memakai framework akan menghemat waktu pengerjaan suatu aplikasi, karena setiap anggota sudah memiliki sebuah acuan dalam menyelesaikan modul. Dalam hal ini misalnya semakin banyak library yang ada semakin mempercepat anggota untuk

menemukan solusi karena tidak setiap anggota harus membuat Class atau fungsi untuk kasus yang relatif sama.

Berikutnya Team tidak akan di susahkan dengan adanya perputaran anggota, jadi jika ada anggota yang tidak bisa melanjutkan lagi pekerjaannya, anggota yang lain bisa meng-cover kekosongan tersebut. Bayangkan jika setiap modul yang dikembangkan mempunyai logika yang berbeda tentunya akan memakan waktu yang banyak untuk proses pemahan akan system tersebut.

Kesekian-kalinya dengan adanya framework akan menjaga integritas dari modul-modul yang dikembangkan. Hal ini juga tergantung dari metode yang dikembangkan sendiri. framework hanya membantu dan memungkinkan/mempermudah proses integrasi, tidak berarti dengan adanya framework system otomatis akan ter-integrasi. Dan masih ada kelebihan-kelebihan lain yang dimiliki oleh framework.

Kekurangan framework sejauh ini belum ada tapi pemilihan framework yang salah akan menjadi bencana

Beberapa kelebihan laravel :

1. ExpressifLaravel adalah sebuah framework PHP yang expressif artinya ketika melihat suatu syntax Laravel, seseorang programmer “diharapkan” akan langsung tahu kegunaan dari syntax tersebut meskipun belum pernah mempelajarinya apalagi menggunakannya.
2. SimpleSalah satu yang membuat laravel begitu simple adalah adanya Eloquent ORM. Misalkan, kita ingin mengambil semua data yang ada pada table users. Maka yang kita perlukan, hanya membuat sebuah class model bernama User: Kemudian kita tinggal masukkan semua data dari table users tersebut dengan cara sebagai berikut :

```
$all_user = User::all();
```

dengan begitu, semua data dari table users, akan dengan mudah diakses dengan melakukan looping terhadap variabel \$all_user.

3. Dikembangkan secara khusus untuk PHP 5.3Mungkin banyak yang sudah tahu bahwa php 5.3 miliki cukup banyak fitur baru dalam segi bahasa, yang membuat php terasa lebih modern dan powerfull. Laravel dikembangkan secara khusus untuk php 5.3, jadi framework ini bisa memanfaatkan berbagai macam kelebihan yang dimiliki php versi terbaru tersebut. Tidak ada backward compatibility dengan php versi sebelumnya.
4. Dokumentasi yang baikLaravel dibuat dengan dokumentasi yang sangat lengkap. Core Developer dari Laravel sendiri ber-komitmen, untuk selalu menyertakan dokumentasi yang lengkap setiap kali melakukan versi terbarunya.

Laravel Framework

Laravel adalah framework PHP yang dikembangkan pertama kali oleh Taylor Otwell. Walaupun termasuk pemain baru, namun komunitas pengguna laravel sudah berkembang pesat dan mampu menjadi alternatif utama dari sejumlah framework besar seperti CodeIgniter & Yii. Laravel oleh para developer disetarakan dengan CodeIgniter dan FuelPHP namun memiliki keunikan tersendiri dari sisi coding yang lebih ekspresif dan elegan.

Keunggulan Laravel daripada framework lain antara lain:

- Coding yang simple
- Tersedia generator yang canggih dan memudahkan, Artisan CLI
- Fitur Schema Builder untuk berbagai database,
- Fitur Migration & Seeding untuk berbagai database,
- Fitur Query Builder yang keren,
- Eloquent ORM yang luar biasa,
- Fitur pembuatan package dan bundle,

Laravel untuk pertama kali dikembangkan sendiri oleh Taylor Otwell. Namun, sampai versi ke-4 sekarang, framework opensource ini dikembangkan bersama oleh komunitas dengan tokoh-tokoh penting selain Otwell adalah Dayle Rees, Shawn McCool, Jeffrey Way, Jason Lewis, Ben Corlett, Franz Liedke, Dries Vints, Mior Muhammad Zaki dan Phil Sturgeon.

Alasan Memilih Laravel

1. Composer Support

Sebagai programmer terutama PHP kita banyak “reinvent the wheel” maksudnya mengulangi pekerjaan yang tidak perlu kita lakukan lagi, contoh nyatanya kita membuat Library untuk memvalidasi data, padahal kita tinggal download dan pakai. Dengan adanya composer, pekerjaan kita sangat dibantu.

Composer sendiri adalah Dependency Management PHP yang membantu kita untuk mencari library yang ingin kita pakai dan menginstallnya, semua library dari composer dihost di packagist.org dan laravel sendiri juga kita install melalui Composer. Sebagai contoh kita memerlukan library Guzzle untuk melakukan POST/GET request sebuah HTTP API. Kita cukup mengetikkan syntax ini di composer.json :

```
{
  "require": {
    "guzzlehttp/guzzle": "~6.0"
  }
}
```

Selanjutnya tinggal melakukan composer update, maka library Guzzle sudah dapat digunakan.

2. Command Line Tools – Artisan

Bagi kamu yang terbiasa menggunakan Linux pastinya tidak akan asing lagi dengan istilah “Command Line”. Di dalam framework laravel sendiri sudah dilengkapi dengan sebuah tools command line yang bernama “Artisan”. Artisan dapat membuat berbagai pekerjaan kita menjadi semakin cepat dan lebih mudah. Contoh yang paling simple adalah proses pembuatan controller.

Laravel sendiri telah dibekali dengan command line tools bernama “Artisan” apa saja yang bisa dilakukan oleh artisan?

1. Database Migration

2. Serve application (tanpa perlu menaruhnya di htdocs, keren kan?)
3. Merubah status aplikasi menjadi down dan up
4. Database Seeding (Memasukkan data awal ke database)
5. Tail (Melihat log server secara realtime)
6. Dan masih banyak lagi

Kalau biasanya kamu mengalami kesulitan membuat controller/model, kini kamu bisa melakukannya dengan lebih cepat hanya dengan mengetikkan perintah ini di terminal :

```
php artisan make:controller PostsController --resource
```

3.PowerfullRouting

Routing merupakan salah satu komponen yang sangat penting dalam pemrograman berbasis MVC. Setiap proses dalam sebuah sistem memang harus diatur bagaimana proses itu berjalan. Tentu saja, dalam hal ini Laravel memiliki teknik routing yang cukup powerfull sehingga sanggup merespon request dari berbagai method mulai dari GET, POST, STORE, PUT, PATCH, dan DELETE.

```
Route::get('/posts', 'PostsController@index');
```

Line diatas akan merespons method get ketika mengakses /post untuk diarahkan ke PostsController dengan function index.

4.CleansCode

Memiliki kodingan yang bersih dan tidak rumit untuk dibaca adalah keinginan setiap programmer. Dalam laravel, beberapa function telah disederhanakan dalam design pattern facade. Syntax-syntax yang biasanya agak panjang ditulis, kini bisa kita persingkat.

```
//register session
Session::set('name', 'mahirkoding');

//get name mahirkoding
Session::get('name');
```

5.EloquentORM

Dari sisi backend, Laravel juga memiliki kelebihan yang sangat powerfull. Permainan query-query database juga dapat dilakukan dengan mudah dan dengan syntax yang sangat singkat.

```
//set tabel
protected $table = 'posts';
//set id
protected $primaryKey = 'post_id';
//select semua posts
Post::all();
//insert data
User::create('id'=>'1', 'judul'=>'Pengenaln Laravel');
//delete data
$post = Post::find(1);
$post->destroy();
//select with where condition
Post::where('judul', '=', 'Pengenaln Laravel')->get();
```

```
//relationship
public function kategori()
{
    return $this->belongsTo('App\\Kategori');
}
```

6. Blade Engine Template

Belum bisa dikatakan pengguna laravel jika kamu belum menggunakan fitur blade engine. Blade Engine di laravel mampu mengkonversi syntax blade ke dalam html/php. Bayangkan saja, syntax yang panjang bisa menjadi pendek jika memanfaatkan fitur ini.

```
$nama = "laravel";

//echo $nama
{{ $nama }}

//set nama dalam array
$nama = ['ani', 'tono', 'budi'];

//mulai looping
@foreach ($nama as $data)
{{ $data }}
//tutup proses looping
@endforeach
```

7. Test Driven Development Ready

Test driven development adalah cara baru pengembangan software, dimana kita menulis test code terlebih dahulu sebelum kita menulis kode sebenarnya. Laravel sendiri memakai PHPUnit sebagai test frameworknya

8. Beautiful Templating Engine

Templating engine, adalah program yang memparse syntax template engine tersebut ke HTML. Laravel sendiri memiliki templating engine yang powerful bernama blade, sebenarnya dalam pembuatan program di laravel, kita bisa memilih untuk menggunakan pure PHP atau templating engine hanya dengan penamaan filenya. Untuk menggunakan blade penamaan file viewnya menjadi namafile.blade.php

Seperti apakah enaknya blade

```
1      Di PHP Biasa
2      Nama anda <?php echo $nama; ?>
3
4      Di Blade
5      Nama anda {{ $nama }}
6
7
8
9      Di PHP Biasa
10     if (isset($nama)) {
```



```

11  Nama anda <?php echo $nama; ?>
12  }
13
14  Di Blade
15  @if (isset($nama))
16  Nama anda {{ $nama }}
17  @endif
18
19
20
21  Di PHP Biasa
22  foreach ($dataKaryawan as $karyawan){
23  echo $karyawan->nik;
24  echo $karyawan->nama;
25  }
26
27  Di Blade
28  @foreach ($dataKaryawan as $karyawan)
    {{ $karyawan->nik }}
    {{ $karyawan->nama }}
  @endforeach

```

dan di blade ini kita bisa membuat layout, biasanya saat kita membuat website yang berubah hanya isinya kan tp header, menu, dan footernya tidak

9. Elegant Code

Laravel dibangun dengan design pattern facade, berarti kita tidak perlu berurusan dengan API dari library-library yang rumit, kita hanya perlu mengaksesnya dengan cara yang sangat mudah tanpa harus meload class tersebut dan menginstansiasi class tersebut. Contohnya:

```

1  Session::set('nama', 'David');
2  Route::get('/user', 'uses' => 'index@UserController');
3  Form::text('nama');

```

cepat dan mudah bukan? kita tidak perlu menginstansiasi class dan menginkludanya seperti dibawah ini

```

1  $sess = new Session();
2  Dll

```

5.7 Alasan menggunakan Laravel Framework Laravel dibanding native PHP

1. Menghindari repetitive work

Membangun berbagai fitur di website itu cukup memakan waktu. Apalagi jika kita membangunnya sendiri. Itulah sebabnya, dalam industri web itu ada yang namanya [NIH \(Not](#)

[Invented Here](#)). Jadi, untuk beberapa fitur web, kita serahkan ke library lain yang bukan hasil bikinan kita.

Contoh sederhana adalah fitur export excel. Fitur ini sangat sering saya temui dalam membuat website. Bisa saja saya membuat library sendiri yang merubah file teks biasa menjadi excel, tapi pasti memakan waktu. Oleh karena itu, fitur ini selalu saya percayakan ke library [PHPExcel](#).

Laravel, bekerja di bawah level library itu. Ia bekerja meng-abstraksi fitur-fitur native PHP. Banyak sekali contohnya, dari Session, Redirect, Autentikasi, Cache, Input, Lokalisasi, Routing, Database, Queue dan sebagainya.

Saya contohkan dengan Routing. Dalam mengembangkan aplikasi PHP, pasti kita sering berurusan dengan URL. Yang paling sederhana, adalah kita membuat subfolder dengan berbagai file php di dalamnya. Misalnya:

```
index.php
/blog/index.php
/blog/berita/index.php
/blog/kategori/index.php
/kontak.php
dst..
```

Tentunya, cara seperti ini tidak efektif untuk dilakukan untuk aplikasi yang besar. Karena, kita tidak akan paham struktur URL di website yang dibangun tanpa melihat isi tiap foldernya.

Teknik lain dalam routing ini adalah dengan menggunakan **pattern Front Controller**. Jika Anda belum paham, pattern ini memungkinkan kita membuat sebuah aplikasi PHP dengan url yang kompleks tapi hanya memiliki satu file index. Pattern ini sangat sering ditemui di berbagai CMS populer misalnya wordpress.

Untuk membangun pattern ini, ada cukup banyak hal yang harus Anda lakukan dari membuat logic front controller dan setup url writing di web server (misalnya dengan .htaccess di apache).

Routing di Laravel akan memudahkan developer membuat aplikasi dengan pattern Front Controller. **Routing Laravel telah mengabstraksi fitur URL di PHP untuk melakukan hal dasar ini.** Contoh sederhana, jika kita ingin membangun URL seperti diatas, kita cukup mengisi file **routes.php** dengan:

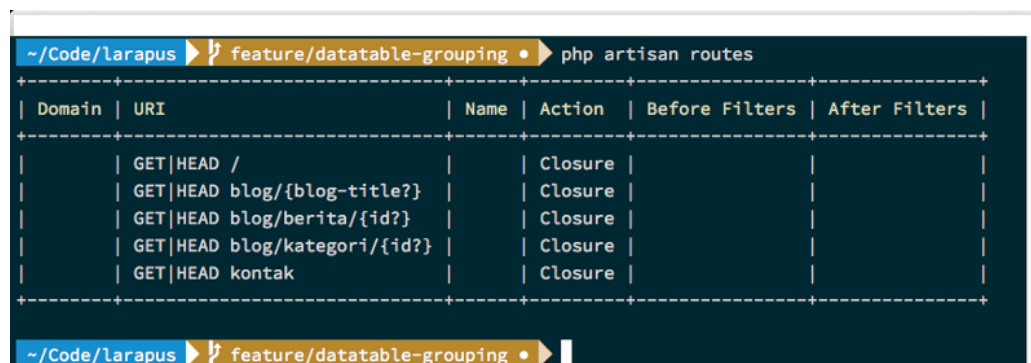
```
Route::get('/', ...);
Route::get('blog/{blog-title?}', ...);
```

```
Route::get('blog/berita/{id?}', ...);
Route::get('blog/kategori/{id?}', ...);
Route::get('kontak', ...);
dst...
```

Dengan cara ini, ada beberapa keuntungan:

- Untuk menambah/mengubah/menghapus URL, kita cukup memodifikasi file **routes.php**.
- Untuk mengetahui route apa saja yang tersedia di aplikasi, kita pun cukup membuka file **routes.php**.

Bahkan Laravel menyediakan fitur untuk mengetahui semua route dengan satu perintah, yaitu **php artisan routes**. Ini salah satu contoh outputnya:



```
~/Code/larapus > feature/datatables-grouping > php artisan routes
```

Domain	URI	Name	Action	Before Filters	After Filters
	GET HEAD /		Closure		
	GET HEAD blog/{blog-title?}		Closure		
	GET HEAD blog/berita/{id?}		Closure		
	GET HEAD blog/kategori/{id?}		Closure		
	GET HEAD kontak		Closure		

```
~/Code/larapus > feature/datatables-grouping > |
```

2. Security

Bagi saya, keamanan aplikasi merupakan nomor satu dalam mengembangkan website. Terlebih jika website tersebut menyimpan banyak data yang sangat penting.

Keamanan merupakan hal yang terus berkembang. Riset terus dilakukan. Tiap hari celah baru ditemukan. Begitupun dengan penutup celahnya.

Mengikuti perkembangan keamanan aplikasi PHP, misalnya di [phpsec](#), [exploit-db](#), [sensiolabs security](#), dsb bisa cukup melelahkan dan menyita waktu. Apalagi bagi seorang single fighter developer, udah capek bikin website, tambah capek lagi ketika websitenya di hack. *Sakitnya tuh, disini, disini, disini...* :D

Saya sendiri ketika masih menjadi asisten di Unpad, pernah menguji aplikasi yang dibuat oleh mahasiswa dengan melakukan SQL Injection. Dan hasilnya, cukup lumayan, saya dapat login ke aplikasi mereka tanpa mengetahui username dan passwordnya.. ☺

Laravel dengan komunitasnya, hadir memastikan fondasi dari aplikasi yang kita bangun benar-benar aman dari masalah security dasar seperti ini. Dari mulai pengamanan csrf, autentikasi, sanitasi data, validasi data dsb telah dilakukan oleh framework Laravel. **Semuanya itu kita dapat dengan gratis.**

Sekiranya ada lubang keamanan di framework ini, tentu akan banyak developer lain yang tahu. Dan, penutup celah keamanan ini akan segera dirilis. Misalnya seperti celah keamanan [ini](#). Untuk menutup celah ini, saya tinggal mengupdate versi laravel ke versi terbaru. Selesai.

3. *Quality*

Sebagaimana saya jelaskan di poin 2. **Laravel akan menjadi fondasi dari aplikasi yang kita bangun.** Jika kita hendak membangun rumah, tentunya fondasi yang kokoh dan kuat akan menopang rumah yang besar. Bahkan, gedung pencakar langit sekalipun bisa dibangun diatasnya.

Yap, begitulah Laravel. Dimulai dari standar code yang mengikuti [PSR0 dan PSR1](#). Begitupun dengan syntax dari framework ini, ada [raturan test](#) yang berjalan ketika sebuah syntax baru dimasukkan ke dalam framework ini. Jadi, kita bisa memastikan bahwa framework ini benar-benar solid dan aman. Siap menjadi fondasi dari aplikasi kita, sebesar dan sekompleks apapun.

4. Memudahkan Teamwork

Alasan ini saya tujukan bagi Anda yang membuat framework sendiri. Saya ingin menantang Anda, seberapa cepat developer yang baru mempelajari framework yang Anda buat?

Tentunya, pasti beragam. Tergantung seberapa kompleks arsitektur framework yang Anda buat dan seberapa lengkap dokumentasinya. Belum lagi Anda akan direpotkan ketika developer baru tersebut mendapatkan sebuah fitur yang tidak dijelaskan di dokumentasi. Mencari di google pun di ngga akan ketemu, sekiranya framework ini hanya Anda gunakan di internal kantor (bukan opensource).

Berbeda kasusnya dengan menggunakan Laravel. Jika ada developer baru yang masuk ke project, dia cukup mempelajari dokumentasi resmi Laravel. Ini akan sangat menghemat waktu. Bahkan, ada penulis buku yang berani menjamin Anda [paham Laravel dalam seminggu](#). (oke, itu saya, hahaha :D)

Suatu saat dia menemukan fitur yang tidak dipahami dari Laravel, dia cukup buka google dan lautan informasi akan datang menghampirinya. Ada pula [forum resmi](#) maupun [tidak resmi](#) yang siap menjawab pertanyaannya. Simple kan?

Yap, memang ada kepuasan tersendiri ketika kita membuat framework sendiri. Tapi ingat, framework yang solid itu tidak hanya dibangun dari codebase yang kompleks dan arsitektur yang berlapis-lapis. Tapi, ditambah pula dengan dokumentasi yang lengkap dan codebase yang selalu diaudit dari berbagai masalah keamanan dan performa.

Saran saya, untuk pekerjaan yang akan dikerjakan oleh banyak orang, mari kita turunkan ego kita dan gunakan framework yang banyak digunakan orang lain.

5. Interoperability

Jika Anda hendak membangun aplikasi open source, Anda harus memastikan codebase yang digunakan dipahami oleh banyak orang. Dengan begitu, akan lebih banyak orang yang akan berkontribusi di project tersebut.

Laravel dengan popularitas dan berbagai kelebihanannya menjawab masalah itu. Dapat kita temui banyak project opensource menggunakan Laravel, misalnya [Flarum](#), sebuah aplikasi forum yang dibangun diatas Laravel dan EmberJS.

6. Hemat Biaya

Perlu saya jelaskan lagi?? Oke, ini beberapa hal yang akan mengeluarkan biaya ketika Anda menggunakan PHP native atau membuat framework sendiri:

- Membuat dokumentasi penggunaan
- Membuat dokumentasi API framework
- Maintenance security
- Maintenance bug
- Maintenance performa
- dll.

7. Hemat Waktu

Dengan berbagai abstraksi yang dilakukan Laravel, Anda dapat lebih fokus memikirkan logic bisnis dari aplikasi yang Anda buat.

Waktu Anda tidak perlu lagi disia-siakan dengan mengurus hal-hal dasar di PHP. Titik.

Tentunya, dari semua alasan diatas, pilihan tetap saja jatuh ke tangan Anda. Jika aplikasi yang dikembangkan sangat simple, tentunya penggunaan Laravel bisa *overkill*.

BAB 2.Fitur-Fitur Laravel

Fitur Laravel

- *Bundles*, Bundel atau ikatan menyediakan sistem kemasan modular. Dengan fitur ini kita dapat dengan mudah untuk melakukan penambahan paket aplikasi ke dalam project kita. Laravel versi 4.x menggunakan *composer* sebagai manajer aplikasi.
- *Eloquent ORM (Object-Relational Mapping)*, merupakan implementasi PHP lanjutan dari *active record* yang menyediakan metode tersendiri dalam mengatur

relationship antar obyek di database. Laravel *query builder* adalah salah satu fitur yang *disupport Eloquent*.

- *Application logic*, fitur pengembangan aplikasi secara umum, baik dengan *controller* atau pendeklarasian *route*
- *Reverse routing*, fitur yang mampu mendefinisikan hubungan antara *link* dan *route*, sehingga memungkinkan perubahan *link* dari *route* atau tanpa melakukan perubahan di *view*.
- *Restfull controllers*, merupakan cara opsional untuk memisahkan logika antara HTTP GET dan POST
- *Class auto loading*, menyediakan fitur untuk *load* PHP *class* tanpa perlu melakukan *include*, *On-demand loading* hanya akan me-*load class* yang diperlukan.
- *View composers*, merupakan kode *logic* yang dieksekusi ketika *view* di-*load*
- *IoC container*, memungkinkan obyek baru yang akan dihasilkan sesuai prinsip kontrol, dengan instansiasi opsional dan referensi dari obyek baru.
- *Migrations*, menyediakan sistem kontrol untuk skema database, sehingga memungkinkan untuk menghubungkan antara perubahan kode aplikasi dengan layout database, memudahkan *deploy* dan *update* aplikasi.
- *Unit testing*, menyediakan fitur testing untuk mendeteksi atau mencegah kode ganda atau berulang (regresi), unit test ini dapat dijalankan melalui perintah *command line*.
- *Automatic pagination*, fitur yang memungkinkan pembuatan halaman/*paging* secara otomatis dengan metode yang sudah diintegrasikan ke laravel.

Basic Routing

Routing berfungsi untuk mengatur alur url dan mendefinisikan url tersebut akan memanggil controller mana atau mengeksekusi fungsi apa.

- Basic GET route

```

1 <?php
2
3 Route::get('/', function()
4 {
5     return "Halo dunia";
6 });
7
8

```

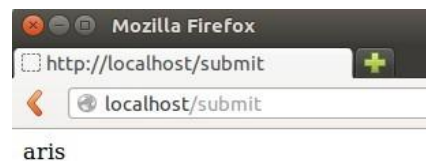


- Basic POST route

```

1 <?php
2
3 Route::get('form', function()
4 {
5     return "
6     <form action='submit' method='post'>
7         <input type='text' name='nama'>
8         <input type='submit' value='submit'>
9     </form>";
10 });
11
12 Route::post('submit', function()
13 {
14     return Input::get('nama');
15 });
16
17

```



Ketika button submit diklik maka halaman akan mengarah ke url /submit dan mengirimkan / post data yaitu nama.

- Route Parameters

```

1 <?php
2
3 Route::get('user/{id}', function($id)
4 {
5     return 'User '.$id;
6 });
7
8

```



Route '{id}' akan membaca karakter apapun yang terdapat di url setelah 'user/'

Request & Input

- Basic input

```
1 <?php
2
3 Route::get('form', function()
4 {
5     return "
6     <form action='submit' method='post'>
7         <input type='text' name='nama'>
8         <input type='submit' value='submit'>
9     </form>";
10 });
11
12 Route::post('submit', function()
13 {
14     return Input::get('nama');
15 });
16
17
```

untuk mengakses semua input dari user, tanpa memerlukan HTTP verb seperti \$_POST atau \$_GET.

- Jika value kosong, bisa juga diset default value:

```
$nama = Input::get('nama', 'Aris');
```

artinya jika value nama kosong, variabel \$nama akan diisi 'Aris'.

- Untuk melakukan pengecekan terhadap input value dapat dilakukan seperti berikut:

```
if (Input::has('name'))
```

```
{
    //
}
```

artinya jika value name bernilai true atau tidak kosong, maka akan masuk ke kondisi if

- Mengambil semua input value

```
$input = Input::all()
```

akan menghasilkan variable \$input yang berbentuk array.

- Mengambil beberapa inputan saja

Panduan Laravel PHP Framework

```
$input = Input::only('username', 'password');
```

\$input = Input::except('password_confirmation'); kode yang pertama akan mengambil hanya value username dan password saja, sementara kode yang kedua akan mengambil semua input value kecuali password_confirmation.

- Mengambil value dari form dengan input berupa array

```
$input = Input::get('produk.nama');
```

akan mengambil input value dari form dengan name='produk[nama]'

- **Cookies**

Semua cookies di laravel dienkripsi dan ditandai dengan kode autentikasi, yang berarti cookie akan invalid jika diubah dari sisi client sehingga menjadi keamanan tersendiri bagi aplikasi web kita.

- Mengambil cookie value

```
$value = Cookie::get('name');
```

- Menambahkan cookie baru ke response

```
$response = Response::make('Halo Dunia');
```

```
$response->withCookie(Cookie::make('name', 'value',  
$minutes))
```

- Mengantri ke queue untuk response berikutnya

```
Cookie::queue($name, $value, $minutes)
```

- Membuat cookie tanpa expired

```
$cookie = Cookie::forever('name', 'value');
```

Panduan Laravel PHP Framework

- **Old Input**

Digunakan menyimpan inputan dari halaman sebelumnya, misalnya untuk

mengembalikan input value ketika validasi error.

- Menyimpan sementara (flash) input ke dalam session

```
//flash semua inputan Input::flash();
```

```
//flash hanya username dan email
```

```
Input::flashOnly('username', 'email');
```

```
//flash semua inputan kecuali password
```

```
Input::flashExcept('password');
```

```
//membubuhkan langsung semua input ke Redirect return
```

```
Redirect::to('form')->withInput();
```

```
//membubuhkan langsung semua input kecuali password return
```

```
Redirect::to('form')
```

-

```
>withInput(Input::except('password '));
```

- Mengambil old data selain dari flash

```
Input::old('username')
```

- **Files**

Contoh kode untuk upload file

```
if(Input::hasFile('foto')) //mengecek value foto
```

```
{
```

Panduan Laravel PHP Framework

```
$file = Input::file('foto'); //semua berkas file
```

```
$filename = $file->getClientOriginalName(); //nama file
```

```
$file->move(public_path().'/uploads/', $filename);
```

```
}
```

- Mengambil path file

```
$path = Input::file('foto')->getRealPath();
```

- Mengambil nama file asli

```
$filename = Input::file('foto')->  
getClientOriginalName();
```

- Mengambil ekstensi atau format file

```
$extension = Input::file('foto')  
->getClientOriginalExtension();
```

- Mengambil size file

```
$size = Input::file('foto')->getSize();
```

- Mengambil MIME type file

```
$mime = Input::file('foto')->getMimeType();
```

- **Request Information**

- Mengambil URI request

```
$uri = Request::path()
```

- Mengambil dan mengecek request method

```
//mengambil method request
```

```
$method = Request::method();
```

//mengecek method request

```
if (Request::isMethod('post'))  
  
{  
  
    //  
  
}
```

- Menentukan apakah request sesuai pattern

```
if (Request::is('admin/*'))  
  
{  
  
    //  
  
}
```

- Mengambil request url dan URI Segment

```
//request url  
  
$url = Request::url();  
  
//uri segment  
  
$segment = Request::segment(1);
```

- Mengambil request header

```
$value = Request::header('Content-Type');
```

- Mengambil value dari \$_SERVER

```
$value = Request::server('PATH_INFO')
```

- Menentukan apakah request berasal dari protokol HTTPS

```
$if (Request::secure())
```

```
{  
    //  
}
```

- Menentukan apakah request menggunakan AJAX

```
$if (Request::ajax())  
  
{  
    //  
}
```

- Menentukan apakah request ber-type JSON

```
$if (Request::isJson())  
  
{  
    //  
}
```

- Menentukan apakah request menginginkan type JSON

```
$if (Request::wantsJson())  
  
{  
    //  
}
```

- Mengecek apakah request sesuai format yang diinginkan

```
$if (Request::format() == 'json')  
  
{  
    //  
}
```

Views & Responses

- Basic Responses

- Menghasilkan atau mengembalikan string dari routes

Route::get('/', function()

```
{  
  
    return 'Halo dunia';  
  
});
```

- Membuat custom responses

\$response = Response::make(\$contents, \$statusCode);

```
$response->header('Content-Type', $value); return  
  
$response;
```

- Redirects

Redirect digunakan untuk mengalihkan halaman. Redirect di laravel bermacam-macam jenisnya, bisa langsung ke route url, route name, dan juga langsung ke method dari sebuah controller

//redirect ke route url

```
return Redirect::to('user/login');
```

//redirect ke route url dengan flash data

```
return Redirect::to('user/login')->with('message', 'Login Failed');
```

//redirect ke route name

```
return Redirect::route('login');
```

//redirect ke route name dengan parameter

Panduan Laravel PHP Framework

```
return Redirect::route('profile', array(1));
```

//redirect ke route name dengan nama parameter dan value-nya

```
return Redirect::route('profile', array('user' => 1));
```

```
//redirect ke method controller
```

```
return Redirect::action('HomeController@index');
```

```
//redirect ke method controller dengan parameter return
```

```
Redirect::action('UserController@profile', array(1));
```

```
//redirect ke method controller dengan nama parameter dan
```

```
value-nya
```

```
return Redirect::action('UserController@profile', array('user' => 1));
```

- **Views**

Views pada umumnya berisi kode-kode HTML, dan menyediakan cara yang aman untuk memisahkan antara controller dan tampilan. Views disimpan pada direktori app/views.

- Simple view

Lokasi file di app/views/halo.php

```
<html>  
<body>  
        <h1>Halo, <?php echo $nama; ?></h1>  
    </body>  
</html>
```

Panduan Laravel PHP Framework

Kemudian membuat route sebagai berikut:

```
Route::get('/', function()  
    {  
        return View::make('halo', array('nama' => 'Aris'));  
    });
```

- Passing data ke views

// cara konvensional


```
$view = View::make('halo')->with('nama', 'Aris');
```

// cara lain

```
$view = View::make('halo')->withNama('Aris');
```

- Passing dengan parameter

```
$view = View::make('halo', $data);
```

- Passing data ke semua views

```
View::share('nama', 'Aris');
```

- Passing view dari sub view ke view

misalnya terdapat subview di app/views/template/footer.php, kita ingin memasukkan ke view halo, seperti berikut

```
$view = View::make('halo')
```

```
->nest('footer', 'template.footer');
```

//atau

```
$view = View::make('halo')
```

```
->nest('footer', 'template.footer',  
$data);
```

Panduan Laravel PHP Framework

kemudian di render di view halo seperti berikut

```
<html>
```

```
<body>
```

```
<h1>Halo, <?php echo $nama; ?></h1>
```

```
</body>
```

```
<?php echo $footer ?>
```

```
</html>
```

- Melihat keberadaan view

untuk mengecek keberadaan sebuah view bisa menggunakan method

View::exists

```
if (View::exists('template.content'))  
{  
    //  
}
```

- **View Composers**

View composers digunakan untuk mengorganisasikan view-view yang sudah ada menjadi satu lokasi, fungsi ini bisa disebut seperti “view models” atau “presenters”

- Menentukan view composer

```
View::composer('profile', function($view)  
{  
    $view->with('count', User::count());  
});
```

maka setiap kali view profile dirender, data count akan mengikuti view.

- Multi view composer

```
View::composer(array('profile', 'dashboard'),
```

Panduan Laravel PHP Framework

```
function($view)  
{  
    $view->with('count', User::count());  
});
```

- Class base composer

```
View::composer('profile', 'ProfileComposer');
```

dan class composer nya sebagai berikut:

```
class ProfileComposer {  
    public function compose($view)  
    {
```

```

        $view->with('count', User::count());
    }
}

```

- Mengatur multiple composer

```

View::composers(array(
    'AdminComposer' => array('admin.index',
        'admin.profile'),
    'UserComposer' => 'user',
    'ProductComposer@create' => 'product'
));

```

- **Special Responses**

- JSON Response

```

return Response::json(array('nama' => 'Aris', 'Kota' => 'Jakarta'));

```

- JSONP Response

```

return Response::jsonp(array('nama' => 'Aris',

```

Panduan Laravel PHP Framework

```

    'Kota' => 'Jakarta'))

```

```

-
    >setCallback(Input::get('callback'))
;

```

- File Download Response

```

return Response::download($pathToFile);

//atau

return Response::download($pathToFile, $name,
    $headers);

```

- **Response Macros**

```

Response::macro('caps', function($value)

```

```

{
    return Response::make(strtoupper($value));
});

```

Controllers

- **Basic Controllers**

Walaupun bisa juga mendefinisikan semua logika di route, controller juga penting digunakan untuk mendefinisikan logika berdasarkan kelas-kelas sehingga proses OOP dapat dilakukan. Controller biasanya disimpan di direktori app/controllers, direktori ini sudah terdaftar di dalam classMap file composer.json secara default. Jika ingin meletakkan file controller di direktori lain, maka harus dideklarasikan terlebih dahulu agar composer tahu letak file tersebut.

Contoh basic controller:

```

class UserController extends BaseController {

    /**

```

Panduan Laravel PHP Framework

```

        * Show the profile for the given user.
    */
    public function showProfile($id)
    {
        $user = User::find($id);

        return View::make('user.profile', array('user'
=> $user));
    }
}

```

Semua kelas controller seharusnya meng-extend kelas BaseController. Kelas BaseController juga disimpan di direktori app/controllers, dan mungkin dapat digunakan untuk menuliskan kode logika yang bersifat shared/untuk semua controller yang meng-extend-nya. Setelah contoh basic controller diatas dibuat,

sekarang kita bisa memanggil controller tersebut dari routes seperti berikut:

```
Route::get('user/{id}', 'UserController@showProfile');
```

- Memberi nama method controller di route

```
Route::get('user', array('uses'=>'UserController@index', 'as' => 'userindex'));
```

- **Controller Filter**

Filter pada controller ini memiliki fungsi seperti “regular” route, jika di route seperti:

```
Route::get('profile', array('before' => 'auth', 'uses' =>
'UserController@showProfile'));
```

maka bisa juga dilakukan di controller seperti berikut:

```
class UserController extends BaseController {

    /**
     * Instantiate a new UserController instance.
     */

    public function _____construct()
    {
        $this->beforeFilter('auth', array('except'
            => 'getLogin'));

        $this->beforeFilter('csrf', array('on' => 'post'));

        $this->afterFilter('log', array('only'
            => array('fooAction',
                'barAction')));
    }
}
```

- **Implicit Controllers**

Di laravel tidaklah harus membuat satu route untuk satu method controller, bisa juga satu route untuk satu controller dan semua method, seperti berikut:

```
Route::controller('users', 'UserController');
```

Dengan route di atas bisa dibuat controller dengan syarat, kata pertama pada method adalah HTTP verb (get, post, put, delete) dan any untuk mengizinkan semua jenis HTTP verb yang diikuti nama methodnya, contoh kelasnya sebagai berikut:

```
class UserController extends BaseController { public function getIndex()  
{
```

```

        //
    }

    public function postProfile()
    {
        //
    }

    public function anyLogin()
    {
        //
    }
}

```

- **RESTful Resource Controllers**

Resource controllers mempermudah dalam membuat RESTful controllers terhadap resource. Sebagai contoh ketika kita ingin memanajemen (create, read, update, delete) foto yang disimpan dalam aplikasi kita, bisa langsung menggunakan command line php artisan

php artisan controller:make PhotoController

kemudian daftarkan controller ke dalam route

Route::resource('foto', 'PhotoController');

satu route tersebut akan menghasilkan bermacam-macam route untuk menghandel berbagai macam RESTful action pada resource foto. Action yang di handel antara lain:

Verb	Path	Action	Route Name
------	------	--------	------------

GET	/foto	index	foto.index
GET	/foto/create	create	foto.create
POST	/foto	store	foto.store
GET	/foto/{resource}	show	foto.show
GET	/foto/{resource}/edit	edit	foto.edit
PUT/PATCH	/foto/{resource}	update	foto.update
DELETE	/foto/{resource}	destroy	foto.destroy

Bisa juga menentukan action apa yang dibutuhkan secara spesifik

```
Route::resource('photo', 'PhotoController', array('only' =>  
array('index', 'show')));
```

```
Route::resource('photo', 'PhotoController', array('except'  
=> array('create', 'store', 'update', 'destroy')));
```



```
class UserController extends BaseController {  
  
    /**  
     * Instantiate a new UserController instance.  
     */  
  
    public function _____construct()  
    {  
        $this->beforeFilter('auth', array('except'  
            => 'getLogin'));  
        $this->beforeFilter('csrf', array('on' => 'post'));  
        $this->afterFilter('log', array('only'  
            => array('fooAction',  
                'barAction')));  
    }  
}
```

- **Implicit Controllers**

Di laravel tidaklah harus membuat satu route untuk satu method controller, bisa juga satu route untuk satu controller dan semua method, seperti berikut:

```
Route::controller('users', 'UserController');
```

Dengan route di atas bisa dibuat controller dengan syarat, kata pertama pada method adalah HTTP verb (get, post, put, delete) dan any untuk mengijinkan semua jenis HTTP verb yang diikuti nama methodnya, contoh kelasnya sebagai berikut:

```
class UserController extends BaseController { public function getIndex()
```

```
        //
    }

    public function postProfile()
    {
        //
    }

    public function anyLogin()
    {
        //
    }
}
```

- **RESTful Resource Controllers**

Resource controllers mempermudah dalam membuat RESTful controllers terhadap resource. Sebagai contoh ketika kita ingin memanajemen (create, read, update, delete) foto yang disimpan dalam aplikasi kita, bisa langsung menggunakan command line php artisan

php artisan controller:make PhotoController

kemudian daftarkan controller ke dalam route

Route::resource('foto', 'PhotoController');

satu route tersebut akan menghasilkan bermacam-macam route untuk menghandel berbagai macam RESTful action pada resource foto. Action yang di handel antara lain:

Verb	Path	Action	Route Name
-------------	-------------	---------------	-------------------

GET	/foto	index	foto.index
GET	/foto/create	create	foto.create
POST	/foto	store	foto.store
GET	/foto/{resource}	show	foto.show
GET	/foto/{resource}/edit	edit	foto.edit
PUT/PATCH	/foto/{resource}	update	foto.update
DELETE	/foto/{resource}	destroy	foto.destroy

Bisa juga menentukan action apa yang dibutuhkan secara spesifik

```
Route::resource('photo', 'PhotoController', array('only' =>  
array('index', 'show')));
```

```
Route::resource('photo', 'PhotoController', array('except' =>  
array('create', 'store', 'update', 'destroy')));
```

BAB.3 Laravel&PHP

JSON

JSON (dilafalkan “Jason”), singkatan dari JavaScript Object Notation (bahasa Indonesia: notasi objek JavaScript), adalah suatu format ringkas pertukaran data komputer. Itu. Ngerti kan?

Oke, oke, becanda. :D

Saya jelaskan lebih detail ya, JSON itu format data yang digunakan untuk bertukar data antar bahasa pemrograman. Awalnya JSON ini hanya digunakan di Javascript. Tapi, seiring waktu, JSON juga digunakan oleh bahasa pemrograman yang lain.

Syntax dasar JSON kayak gini:

Contoh JSON

```
1 {  
2     key1 : value1,  
3     key2 : {
```

```
4         key3 : value3,
5         key4 : value4
6     }
7 }
```

¹www.sitepoint.com/best-php-frameworks-2014

JSON mendukung berbagai format data, yaitu :

- Number : berisi bilangan bulat atau desimal.
- String : teks yang diapit tanda petik.
- Boolean : Isian benar (true) atau salah (false).
- Array : Data terurut yang diapit [] dan dibatasi koma. Larik bisa berisi gabungan tipe data yang lain.
- Object : Data tidak terurut yang diapit { } dan dibatasi koma. Setiap elemen object berisi key dan value dibatasi :.
- null : nilai kosong, diisi dengan keyword null.

Jika digabungkan, semua tipe data diatas jadi seperti ini:

Contoh JSON

```
1 {
2     "nomor_antrian" : 20,
3     "nama" : "Firman",
4     "menikah" : false,
5     "hobi" : ["memancing", "berenang", "bersepeda"],
6     "alamat" : {
7         "jalan" : "Pakuan No. 12",
8         "kota" : "Bandung",
9         "provinsi" : "Jawa Barat",
10        "kode_pos" : "43655"
11    },
12    "telpon" : {
13        { "jenis" : "rumah", "nomor" : "022-345352" },
14        { "jenis" : "mobile", "nomor" : "0878-23422321" }
15    },
16    "istri" : null
17 }
```

JSON akan sangat sering kita temui dalam pengembangan web dengan Laravel. Oleh karena itu, pastikan Anda paham tentang JSON ini. Kalau belum paham, ngga apa-apa juga sih, ntar juga paham kalau sudah praktek. Sip.

PHP5 Anonymous Function/Lambda & Closure

Sebagaimana seorang hacker yang tidak ingin diketahui namanya, begitu pula dengan Anonymous Function/Lambda ini. Fitur PHP ini, memungkinkan kita membuat fungsi tanpa menghabiskan

ruang memori. Karena fungsi yang dibuat secara anonymous tidak akan menambah jumlah fungsi pada class yang ada. Jadi, kalau dari segi kesolehan, fungsi ini termasuk tipe tawadhu (rendah hati) karena ia banyak bekerja tanpa ingin diketahui namanya. Hadeuuhhh.. :v

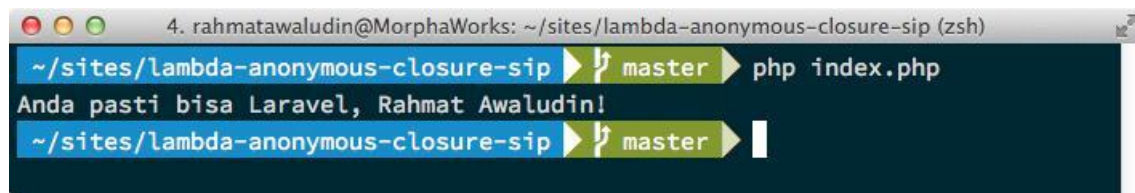
Perhatikan contoh syntax berikut:

/sites/lambda-anonymous-closure-sip/index.php

```
1 <?php
2 function namaAnda() {
3     return "Rahmat Awaludin";
4 }
5
6 function bisa($nama) {
7     echo "Anda pasti bisa Laravel, $nama!\n"; 8 } 9
10 bisa(namaAnda());
```

Yang syntax ini lakukan adalah membuat dua buah fungsi yaitu namaAnda() yang menampilkan nama yang Anda inginkan dan bisa() yang menampilkan tulisan motivasi mempelajari framework Laravel. Fungsi bisa() menerima parameter sebuah string yang kita beri nama \$nama.

Pada baris terakhir, kita memanggil fungsi bisa() dengan parameter output dari fungsi namaAnda(). Tentunya ini bisa dilakukan, karena output dari fungsi namaAnda() adalah string. Sehingga output nya menjadi:



```
4. rahmatawaludin@MorphaWorks: ~/sites/lambda-anonymous-closure-sip (zsh)
~/sites/lambda-anonymous-closure-sip > master > php index.php
Anda pasti bisa Laravel, Rahmat Awaludin!
~/sites/lambda-anonymous-closure-sip > master >
```

Menggunakan fungsi sebagai paramater

Jika membuat fungsi biasa harus pakai nama misal:

```
function namaAnda() {
    return "Rahmat Awaludin";
}
```

maka, Anonymous Function ditulis dengan menghilangkan nama fungsinya:

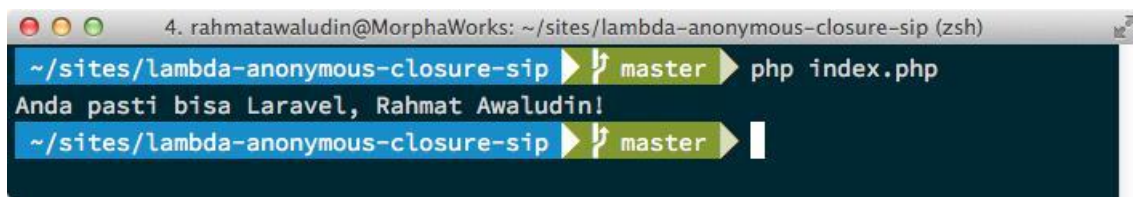
```
function() {  
    return "Rahmat Awaludin";  
}
```

Untuk menggunakan anonymous function ini, kita mempassing fungsi tersebut ke sebuah variable. Berikut ini contoh kalau saya mengubah fungsi namaAnda() menjadi anonymous function dan menyimpannya ke variable \$nama:

/sites/lambda-anonymous-closure-sip/index.php

```
1 <?php  
2 $nama = function() {  
3     return "Rahmat Awaludin";  
4 };  
5  
6 function bisa($nama) {  
7     echo "Anda pasti bisa Laravel, $nama!\n"; 8 } 9  
  
10 bisa($nama());
```

Coba jalankan lagi, hasilnya tetap sama kan?



Anonymous function sedang beraksi

Kalau udah paham, sekarang kita pelajari tentang *Closure*.

Closure mirip-mirip dengan lambda/anonymous function, bedanya closure ini *dapat menerima parameter dan mengakses variable dari luar fungsi itu*. Misal Seperti ini:

/sites/lambda-anonymous-closure-sip/closure.php

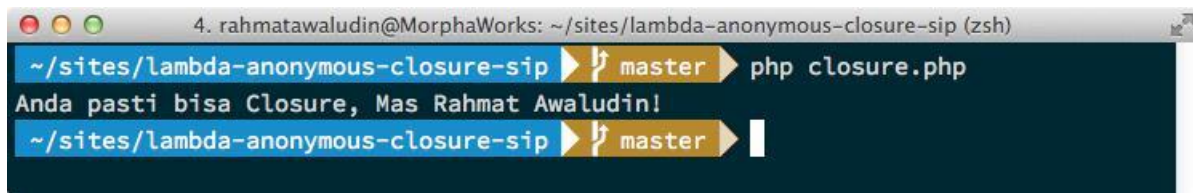
```
1 <?php  
2 $title = "Mas";  
3 $nama = "Rahmat Awaludin";  
4  
5 function namaAnda($title, $nama) {  
6     return $title . ' ' . $nama;  
7 }
```

```

8
9  function bisa($nama) {
10      echo "Anda pasti bisa Closure, $nama!\n";
11  }
12
13  bisa(namaAnda($title, $nama));

```

Pada file ini saya membuat fungsi namaAnda() yang menerima input title dan nama kemudian menampilkan output title dan nama. Saya juga membuat fungsi bisa() yang menerima input string dan menampilkan output kalimat motivasi. Di baris terakhir, saya memanggil fungsi bisa() dengan parameter output dari fungsi namaAnda(). Outputnya seperti ini:



```

4. rahmatawaludin@MorphaWorks: ~/sites/lambda-anonymous-closure-sip (zsh)
~/sites/lambda-anonymous-closure-sip > master > php closure.php
Anda pasti bisa Closure, Mas Rahmat Awaludin!
~/sites/lambda-anonymous-closure-sip > master >

```

Fungsi biasa dengan parameter

Jika kita merubah fungsi namaAnda() menjadi closure, maka syntaxnya akan berubah menjadi :

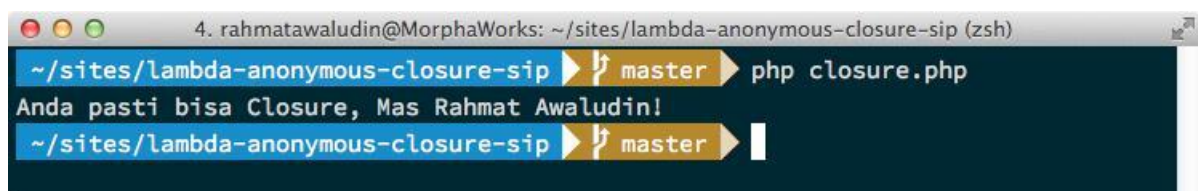
/sites/lambda-anonymous-closure-sip/closure.php

```

1  <?php
2  $title = "Mas";
3  $nama = "Rahmat Awaludin";
4
5  function bisa($nama) {
6      echo "Anda pasti bisa Closure, $nama!\n";
7  }
8
9  $nama = function() use ($title, $nama) {
10      return $title . ' ' . $nama;
11  };
12
13  bisa($nama());

```

Terlihat disini, *untuk menggunakan variable diluar closure* (dalam contoh ini variable \$title dan \$nama), kita menggunakan keyword *use*. Outputnya akan tetap sama:



```

4. rahmatawaludin@MorphaWorks: ~/sites/lambda-anonymous-closure-sip (zsh)
~/sites/lambda-anonymous-closure-sip > master > php closure.php
Anda pasti bisa Closure, Mas Rahmat Awaludin!
~/sites/lambda-anonymous-closure-sip > master >

```

Menggunakan closure

Closure ini biasanya digunakan pada fungsi yang menggunakan *callback* sebagai parameternya.

Apaan tuh callback?

callback adalah parameter berupa fungsi. Contohnya seperti fungsi [array_walk²](#) (bawaan PHP). Fungsi ini akan mengiterasi (mirip foreach) elemen dari suatu array. Parameter pertama dari fungsi ini adalah array dan parameter keduanya harus berupa callback. Nah, callback inilah closure. Perhatikan syntax berikut (boleh sambil dicoba):

/sites/lambda-anonymous-closure-sip/closure-as-callback.php

```
1 <?php
2 $minimal = 75;
3 $dataNilai = [
4     ["nama"=>"Agus", "nilai"=>90],
5     ["nama"=>"Deni", "nilai"=>80],
6     ["nama"=>"Budi", "nilai"=>40],
7     ["nama"=>"Bayu", "nilai"=>75],
8 ];
9
10 array_walk($dataNilai, function($siswa) use ($minimal) {
11     echo "Siswa : " . $siswa['nama'] . "\n";
12     echo "Nilai : " . $siswa['nilai'] . "\n";
13     echo "Status : ";
14     if ($siswa['nilai'] >= $minimal) {
15         echo "Lulus\n";
16     } else {
17         echo "Tidak Lulus\n";
18     }
19 });
```

²<http://php.net/manual/en/function.array-walk.php>

Syntax ini akan menggunakan fungsi `array_walk` untuk melooping semua element dari array `$dataNilai`. Pada parameter kedua di fungsi `array_walk` kita menggunakan closure untuk mem-proses setiap elemen array agar menampilkan detail nama, nilai dan status kelulusan siswa berdasarkan variabel `$minimal` yang kita buat sebelumnya. Berikut ini output dari syntax berikut:


```
4. rahmatawaludin@MorphaWorks: ~/sites/lambda-anonymous-closure-sip (zsh)
~/sites/lambda-anonymous-closure-sip master php closure-as-callback.php
Siswa : Agus
Nilai : 90
Status : Lulus
Closure sebagai callback

Siswa : Deni
Nilai : 80
Status : Lulus

Siswa : Budi
Nilai : 40
Status : Tidak Lulus

Siswa : Bayu
Nilai : 75
Status : Lulus

~/sites/lambda-anonymous-closure-sip master
```

Demikian penjelasan tentang Lambda dan Closure, saya harap Anda dapat memahaminya. Karena, fitur ini akan sering kita gunakan pada pembahasan di bab-bab selanjutnya.



Source code dari latihan ini bisa didapat di
<https://github.com/rahmatawaludin/lambda-anonymous-closure-sip>³

PHP5 Autoloader

Jika Anda seorang veteran di pemrograman PHP (*ehm*), tentu sudah sangat familiar dengan syntax include untuk memasukkan syntax dari file lain ke file yang sedang aktif. Biasanya, syntax ini digunakan jika Anda hendak menggunakan class yang berada pada file yang lain. Perhatikan contoh syntax berikut:

³<https://github.com/rahmatawaludin/lambda-anonymous-closure-sip/commits/master>

/sites/autoloader-oh-autoloader/Printer.php

```
1 <?php
2 class Printer {
3     public function cetakBuku($buku) {
4         echo "Class " . __CLASS__ . " : ";
5         echo "Mencetak buku $buku\n";
6         return "Buku $buku";
7     }
8 }
```

/sites/autoloader-oh-autoloader/Kurir.php

```
1 <?php
2 class Kurir {
3     public function kirim($file, $tujuan) {
4         echo "Class " . __CLASS__ . " : ";
5         echo "Mengirim $file ke $tujuan\n";
6     }
7 }
```

/sites/autoloader-oh-autoloader/index.php

```
1 <?php
2 include 'Printer.php';
3 $printer = new Printer();
4 $buku = $printer->cetakBuku('Menyelami Framework Laravel');
5
6 include 'Kurir.php';
7 $kurir = new Kurir();
8 $kurir->kirim($buku, 'Bandung');
```

Pada syntax ini, terlihat kita memiliki dua buah class yaitu Printer dan Kurir.

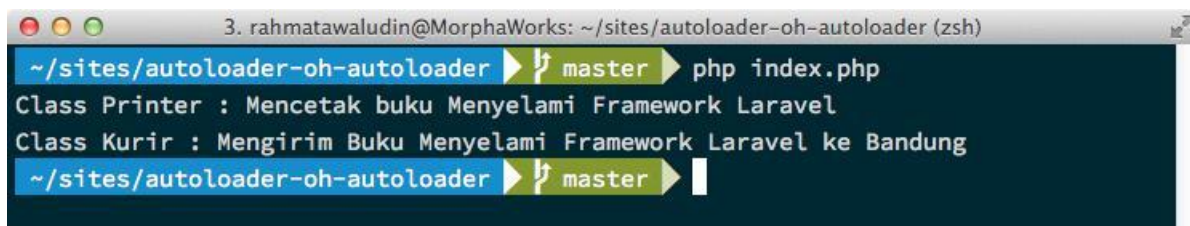
Pada Class Printer, terdapat method cetakBuku() yang berfungsi menampilkan tulisan “Mencetak Buku “ dengan nama buku yang menjadi parameter. Ketika kita mengambil method ini, kita juga memanggil nama Class yang aktif dengan syntax echo "Class " . __CLASS__ . " : ".

Pada Class Kurir, terdapat method kirim() yang berfungsi menampilkan tulisan “Mengirim \$file ke \$tujuan” dengan \$file dan \$tujuan merupakan parameter dari method ini. Kita juga menampilkan nama Class yang aktif ketika memanggil method ini.

Perhatikan file index.php.

Di baris ke-2 dan ke-6 saya menggunakan syntax include untuk memasukan file Printer.php dan Kurir.php.

Ini merupakan cara lama agar saya dapat membuat object dari class Printer dan Kurir yang berada di kedua file tersebut. Syntax selanjutnya adalah syntax biasa untuk membuat object Printer dan Kurir kemudian memanggil method pada masing-masing object tersebut. Output dari syntax ini seperti ini:



```
3. rahmatawaludin@MorphaWorks: ~/sites/autoloader-oh-autoloader (zsh)
~/sites/autoloader-oh-autoloader ▶ master ▶ php index.php
Class Printer : Mencetak buku Menyelami Framework Laravel
Class Kurir : Mengirim Buku Menyelami Framework Laravel ke Bandung
~/sites/autoloader-oh-autoloader ▶ master ▶
```

Sip. Seperti yang Anda lihat, syntax ini berjalan sebagaimana mestinya. Dan cara inilah yang sering digunakan dari dulu (bahkan sampai sekarang masih dipakai di Wordpress) untuk memanggil Class dari file yang berbeda.



Cara ini, meskipun dapat digunakan, tetapi kurang efektif. *Bayangkan, jika Anda memiliki 100 Class di 100 file yang berbeda, apakah Anda mau membuat 100 statement include?*

Tentu tidak. Dan para pendahulu kita yang telah lebih dulu memahami PHP pun memikirkan hal tersebut. Maka, lahirlah fitur autoloader di PHP. Dengan fitur ini, Anda tidak perlu menulis include untuk setiap file PHP yang akan di masukkan ke file.

Untuk menggunakan autoloader kita akan menggunakan fungsi `spl_autoload_register()`⁴. Fungsi ini menerima parameter closure/fungsi yang memiliki sebuah parameter `$class` yang berisi nama class yang akan dipanggil. Di dalam closure ini, kita melakukan include ke class yang diinginkan.

Mari kita praktekan, saya akan rubah file `index.php` menjadi :

```
<?php
// function  autoload($class) {
//   include 'classes/' . $class . '.class.php';
// }

function mv autoloader($class) {
    include 'classes/' . $class . '.class.php';
}

spl_autoload_register('my_autoloader');
—// Or. using an anonymous function as of PHP
5.3.0
spl_autoload_register(function ($class) {
    include 'classes/' . $class . '.class.php';
});
?>
```

⁴<http://php.net/manual/en/function.spl-autoload-register.php>

/sites/autoloader-oh-autoloader/index.php

```
1 <?php
2 spl_autoload_register(function ($class) {
3     include $class . '.php';
4 });
5
6 $printer = new Printer();
7 $buku = $printer-> cetakBuku('Menyelami Framework Laravel'); 8
9 $kurir = new Kurir();
```

10 \$kurir-> kirim(\$buku, 'Bandung');

Pada perubahan ini saya menghapus dua statement include dan menambahkan syntax:

/sites/autoloader-oh-autoloader/index.php

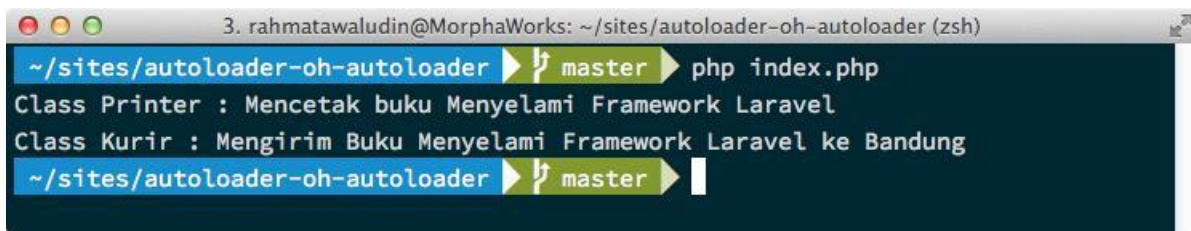
```
1 <?php
2 spl_autoload_register(function ($class) {
3     include $class . '.php';
4 });
```

Yang dilakukan syntax ini adalah melakukan include setiap kali saya melakukan new Class().

Terlihat di baris ke-3, saya menggunakan include \$class . '.php';.

Ketika code berjalan (runtime), misalnya saya memanggil new Printer(), maka syntax ini akan berubah menjadi include Printer.php; yang akan memasukkan konten file tersebut ke code yang sedang aktif. Itulah alasan saya membuat sebuah file untuk sebuah class. Dengan cara ini, Anda tidak perlu lagi melakukan include manual untuk tiap Class yang dibutuhkan. *Keren kan?*

Kalau dijalankan, outputnya akan tetap sama.



Setelah Autoloader

Tentunya, autoloader ini dapat disesuaikan dengan kebutuhan Anda. Misalnya, semua class berada di folder class dan berakhiran .inc.php, maka syntax autoloader berubah menjadi :

Merubah autoloader

```
1 <?php
2 spl_autoload_register(function ($class) {
3     include 'class/' . $class . '.inc.php';
4 });
```

Laravel sangat aktif menggunakan Autoloader ini. Dengan memahami dasar dari Autoloader ini, mudah-mudahan Anda tidak tenggelam dalam kebingungan ketika pembahasan tentang Laravel semakin dalam. Sip.



Source code dari latihan ini bisa didapat di
<https://github.com/rahmatawaludin/autoloader-oh-autoloader>⁵

PHP5 Abstract dan Interfaces

Memahami Abstract dan Interfaces sangat penting untuk dapat mendalami framework Laravel. Saya yakin, dalam mempelajari mempelajari OOP di PHP, Anda pasti telah memahami Class dan Inheritance. Jika Class dan Inheritance adalah nasi dan sayur asem, maka Abstract dan Interfaces adalah ikan asin dan sambelnya. Nah. :D

Abstract

Oke. Kita mulai dari Abstract. *Apa itu Abstract?*

Abstrak adalah tipe yang tidak dapat dipakai secara langsung. (Wikipedia)

Maksudnya, Abstract itu adalah semacam class di PHP tapi tidak bisa langsung dibuat objectnya. Misalnya, sebuah tombol. Kita semua pasti tahu, bahwa tombol apapun pasti bisa ditekan. Hanya saja, tiap tindakan yang terjadi ketika kita menekan tombol akan berbeda, tergantung jenis tombolnya. Perhatikan contoh ini:

⁵<https://github.com/rahmatawaludin/autoloader-oh-autoloader/commits/master>

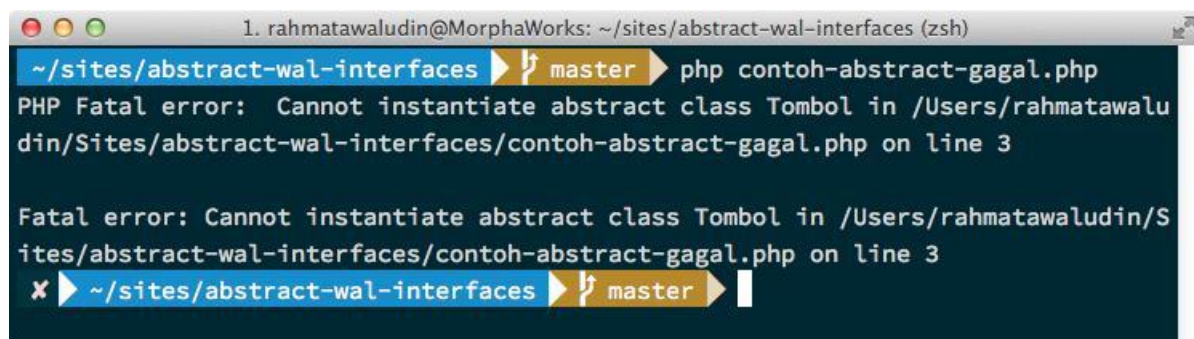
/sites/abstract-wal-interfaces/Tombol.php

```
1 <?php
2 abstract class Tombol {
3     abstract public function tekan();
4 }
```

/sites/abstract-wal-interfaces/contoh-abstract-gagal.php

```
1 <?php
2 include 'Tombol.php';
3 $tombol = new Tombol();
4 $tombol->tekan();
```

Terlihat disini, saya langsung mencoba membuat object (*instantiate*) dari abstract class Tombol. Maka, akan ada error seperti ini:



```
1. rahmatawaludin@MorphaWorks: ~/sites/abstract-wal-interfaces (zsh)
~/sites/abstract-wal-interfaces master php contoh-abstract-gagal.php
PHP Fatal error: Cannot instantiate abstract class Tombol in /Users/rahmatawaludin/Sites/abstract-wal-interfaces/contoh-abstract-gagal.php on line 3

Fatal error: Cannot instantiate abstract class Tombol in /Users/rahmatawaludin/Sites/abstract-wal-interfaces/contoh-abstract-gagal.php on line 3
x ~/sites/abstract-wal-interfaces master
```

Class Abstract tidak bisa langsung dibuat object

Ini menunjukkan bahwa abstract tidak bisa langsung di-*instantiate* menjadi object. Untuk membuat object, kita perlu membuat class baru yang merupakan turunan dari abstract class Tombol ini. Perhatikan syntax ini:

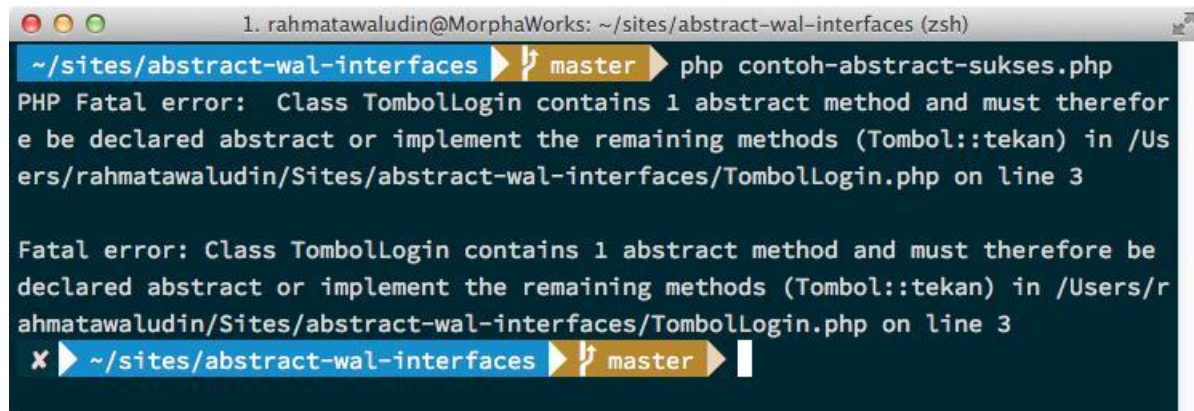
```
1 <?php
2 include "Tombol.php";
3 class TombolLogin extends Tombol {
4
5 }
```

/sites/abstract-wal-interfaces/TombolLogin.php

```
1 <?php
2 include "TombolLogin.php";
3 $tombol = new TombolLogin();
4 $tombol->tekan();
```

/sites/abstract-wal-interfaces/contoh-abstract-sukses.php

Disini kita membuat class baru bernama TombolLogin yang meng-*extend* abstract class Tombol. Kalau kita jalankan :



```
1. rahmatawaludin@MorphaWorks: ~/sites/abstract-wal-interfaces (zsh)
~/sites/abstract-wal-interfaces ▶ master ▶ php contoh-abstract-sukses.php
PHP Fatal error: Class TombolLogin contains 1 abstract method and must therefore
e be declared abstract or implement the remaining methods (Tombol::tekan) in /Us
ers/rahmatawaludin/Sites/abstract-wal-interfaces/TombolLogin.php on line 3

Fatal error: Class TombolLogin contains 1 abstract method and must therefore be
declared abstract or implement the remaining methods (Tombol::tekan) in /Users/r
ahmatawaludin/Sites/abstract-wal-interfaces/TombolLogin.php on line 3
X ▶ ~/sites/abstract-wal-interfaces ▶ master ▶
```

Gagal meng-extends interface, karena ada method yang belum diimplementasikan

Uuuppss... Dia error lagi.

Ini terjadi karena, kita belum mengimplementasikan method abstract yaitu method tekan() yang ada di abstract class Tombol. Mari kita perbaiki:

/sites/abstract-wal-interfaces/TombolLogin.php

```
1 <?php
```

```

2 include "Tombol.php";
3 class TombolNuklir extends Tombol {
4     public function tekan() {
5         echo "Bom nuklir telah diluncurkan!\n";
6         sleep(3);
7         echo "Boooooommmmm!!!\n";
8     }
9 }

```

Kalau kita jalankan lagi:



A terminal window titled '1. rahmatawaludin@MorphaWorks: ~/sites/abstract-wal-interfaces (zsh)'. The prompt is '~ /sites/abstract-wal-interfaces' with a 'master' branch indicator. The command 'php contoh-abstract-sukses.php' has been executed, resulting in the output 'Berhasil login!'.

Berhasil menggunakan abstract class Tombol

Tuhh.. dia berhasil kan? :)

Ini penting. Jadi, abstract itu sangat berguna kalau kita ingin memastikan bahwa suatu method selalu tersedia pada class, apapun implementasinya. Dalam contoh ini, kita selalu bisa memanggil method `tekan()` apapun jenis tombolnya. Biar lebih paham, saya tambah lagi contohnya. Kali ini, ceritanya saya mau bikin tombol untuk meluncurkan bom nuklir. Perlu dicatat, ini hanya contoh, tidak ada yang dilukai dalam pembuatan contoh ini. Perhatikan syntax ini:

```

/sites/abstract-wal-interfaces/TombolNuklir.php
1 <?php
2 include "Tombol.php";
3 class TombolNuklir extends Tombol {
4     public function tekan() {
5         echo "Bom nuklir telah diluncurkan!\n";
6         sleep(3);
7         echo "Boooooommmmm!!!\n";
8     }
9 }

```

Untuk menjalankan tombol ini, kita tetap menggunakan method yang sama, yaitu `tekan()`:

```

1 <?php
2 include "TombolNuklir.php";
3 $tombol = new TombolNuklir();
4 $tombol->tekan();

```

Kalau dijalankan..


```
1. rahmatawaludin@MorphaWorks: ~/sites/abstract-wal-interfaces (zsh)
~/sites/abstract-wal-interfaces ▶ master • ▶ php contoh-abstract-sukses-2.php

Bom nuklir telah diluncurkan!
Boooooooooommmmm!!!
~/sites/abstract-wal-interfaces ▶ master • ▶
```

Berhasil membuat Tombol Nuklir

Boooooooooommmmm!!!

Oke, saya rasa sudah cukup penjelasannya. Pertanyaanya, *kapan biasanya teknik abstract class ini digunakan?*

Contohnya banyak, salah satunya saya contohkan dengan dependency injection. Teknik ini memudahkan kita untuk memasukkan (*inject*) class yang kita butuhkan pada class yang sedang digunakan. Contoh dependency injection, misalnya seorang Pembeli *harus* punya kartu BNI untuk melakukan pembayaran. Bisa kita implementasikan dengan meng-*inject* class BNI ke class Pembeli. Seperti ini:

```
/sites/abstract-wal-interfaces/BNI.php
1 <?php
2
3 class BNI {
4
5     private $saldo;
6
7     public function __construct($pin) {
8         // ceritanya cek PIN ke database
9         if ($pin == '12345') {
10             echo "Berhasil mengaktifkan Kartu BNI!\n";
11         } else {
12             $pesan = "PIN yang Anda masukkan salah :(";
13             throw new Exception($pesan);
14         }
15     }
16
17     private function catatTransaksi($jenis, $jumlah) {
18         echo "Mencatat transaksi $jenis sejumlah $jumlah ke Buku Tabungan";
19     }
20
21     public function kredit($jumlah) {
22         $this->catatTransaksi('transfer keluar', $jumlah);
23         $this->saldo -= $jumlah;
24     }
25 }
```



```

25
26         public function deposit($jumlah) {
27             $this->catatTransaksi('deposit dana', $jumlah);
28             $this->saldo += $jumlah;
29         }
30
31         public function cekSaldo() {
32             return $this->saldo;
33         }
34     }

```

Class BNI ini mempunyai:

- attribute \$saldo yang berfungsi mencatat saldo terakhir.
- method __construct() yang berfungsi membangun object BNI, di method ini kita meng-haruskan input PIN. Dalam prakteknya, tentu saja PIN ini disimpan di database, tapi disini kita sederhanakan dengan menyimpannya langsung di method ini.
- method kredit() yang berfungsi untuk mengurangi jumlah saldo.
- method deposit() yang berfungsi untuk menambah jumlah saldo.
- method cekSaldo() yang berfungsi mengecek jumlah saldo terkini.

Mari kita buat class Pembeli, Class ini akan membutuhkan class BNI :

```

/sites/abstract-wal-interfaces/Pembeli-DI.php
1 <?php
2 include "BNI.php";
3 class Pembeli {
4     private $nama;
5     private $bni;
6
7     public function __construct($nama = "Seseorang", BNI $bni) {
8         $this->bni = $bni;
9         $this->nama = $nama;
10    }
11
12    public function beli($nama = "Barang", $harga = 0) {
13        $this->bni->kredit($harga);
14        echo "Berhasil melakukan pembelian $nama seharga Rp$harga.\n";
15        echo "Terima kasih $this->nama :)\n";
16    }
17 }

```

Class Pembeli ini mempunyai :

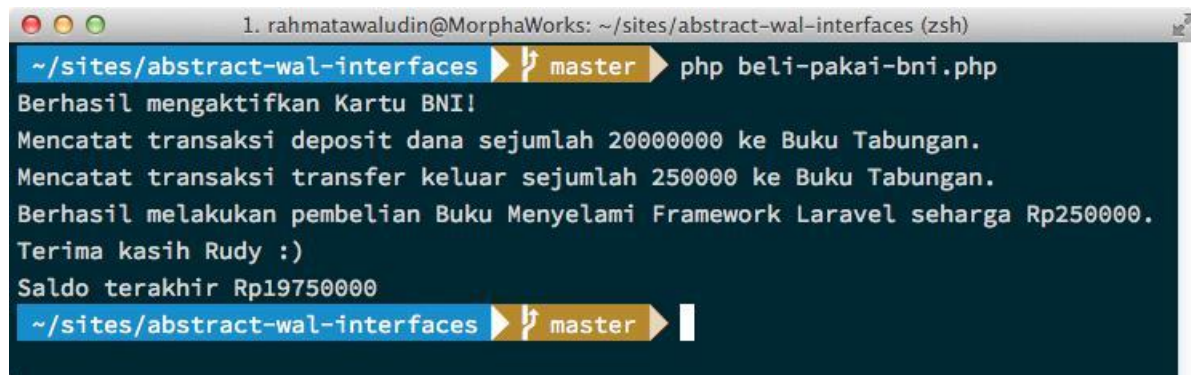
- atribut \$nama untuk menyimpan nama pembeli
- atribut \$bni untuk menyimpan object kartu BNI
- method beli() untuk melakukan pembelian barang
- method __construct() untuk membangun object Pembeli

Yang paling penting untuk diperhatikan disini adalah method `__construct()`. Di method ini kita menginject class BNI sebagai parameternya. Kalau didemokan syntaxnya seperti ini:

/sites/abstract-wal-interfaces/Pembeli-DI.php

```
1 <?php
2 require_once "Pembeli-DI.php";
3 // Melakukan pembelian dengan BNI
4 try {
5     $bniKu = new BNI('12345');
6     $bniKu->deposit(200000000);
7     $rudy = new Pembeli("Rudy", $bniKu);
8     $rudy->beli("Buku Menyelami Framework Laravel", 250000);
9     echo "Saldo terakhir Rp".$bniKu->cekSaldo()."\n";
10 } catch (Exception $e) {
11     echo $e->getMessage()."\n";
12 }
```

Terlihat disini, sebelum membuat object Pembeli, saya membuat object BNI dulu. Kemudian meng-*inject* object BNI itu ketika membuat object Pembeli. Jika dijalankan, hasilnya seperti berikut :



Berhasil meng-*inject* BNI ke Pembeli

Masalah dari dependency injection ini adalah bagaimana bila kita akan menggunakan metode pembayaran lain? Misalnya, Paypal. Tentunya, cara mengakses paypal ini pun akan berbeda dengan BNI, karena Paypal harus login dengan email dan password. Begitupun cara paypal melakukan kredit dan deposit, karena paypal perlu mengirim email setiap kali terjadi transaksi.

Kalau gitu langsung di extends dari class BNI saja gimana mas?

Memang, sekiranya implementasinya akan sama, kita cukup meng-extends class Paypal dari BNI. Namun, karena implementasi dari method `kredit()` dan `deposit()` ini bisa berbeda, maka fitur pembayaran ini cocok untuk di-abstraksi. Dengan abstraksi pula, akan lebih memudahkan jika akan ada implementasi jenis pembayaran yang baru, misalnya BitCoin.

Kita bisa membuat abstraksi dengan membuat class abstract yang berisi method apa saja yang harus ada di Class tersebut yang akan digunakan di class Pembeli. Tahapannya dimulai dari class Pembeli, ubah menjadi :

sites/abstract-wal-interfaces/Pembeli.php
sites/abstract-wal-interfaces/Pembeli.php

```

1  <?php
2  class Pembeli {
3      private $nama;
4      private $payment;
5
6      public function __construct($nama = "Seseorang", PaymentMethod $payment) {
7          $this->nama = $nama;
8          $this->payment = $payment;
9      }
10
11     public function beli($nama = "Barang", $harga = 0) {
12         if ($this->payment->cekSaldo() < $harga) {
13             echo "Uang tidak cukup\n";
14         } else {
15             $this->payment->kredit($harga);
16             echo "Terima kasih $this->nama :)\n";
17             echo "Berhasil melakukan pembelian $nama seharga
18         }
19     }
20 }

```

Perubahan terbesar dari class Pembeli adalah saya mengubah atribut \$bni menjadi \$payment dan mengubah mengabstraksi class BNI menjadi PaymentMethod.

Terlihat disini, class PaymentMethod perlu menambahkan beberapa method:

- cekSaldo() untuk mengecek saldo terakhir
- kredit() untuk mengambil sejumlah uang
- deposit() untuk mengisi sejumlah uang

Untuk memudahkan pengecekan, saya akan menambah method cekNamaPembayaran() yang berfungsi menampilkan nama Class yang digunakan untuk melakukan pembayaran. Mari kita buat abstract class PaymentMethod :

```

/sites/abstract-wal-interfaces/PaymentMethod.php
1  <?php
2  abstract class PaymentMethod {
3      public function cekNamaPembayaran() {
4          return "Anda melakukan pembayaran dengan ".get_class($this)."\n";
5      }
6      abstract public function kredit($jumlah);
7      abstract public function deposit($jumlah);
8      abstract public function cekSaldo();
9  }

```

Selanjutnya, ubah class BNI agar mengekstends PaymentMethod. Untuk memudahkan contoh, saya akan merubah nama classnya menjadi DebitBNI:

/sites/abstract-wal-interfaces/DebitBNI.php

```
1 <?php
2 require_once "PaymentMethod.php";
3 class DebitBNI extends PaymentMethod {
4     private $saldo;
5
6     public function __construct($pin) {
7         // ceritanya cek PIN ke database
8         if ($pin == '12345') {
9             echo "Berhasil mengaktifkan Kartu Debit!\n";
10        } else {
11            $pesan = "PIN yang Anda masukkan salah :(";
12            throw new Exception($pesan);
13        }
14    }
15
16    private function catatTransaksi($jenis, $jumlah) {
17        echo "Mencatat transaksi $jenis sejumlah $jumlah ke Buku Tabungan
18    }
19
20    public function kredit($jumlah) {
21        $this->catatTransaksi('transfer keluar', $jumlah);
22        $this->saldo -= $jumlah;
23    }
24
25    public function deposit($jumlah) {
26        $this->catatTransaksi('deposit dana', $jumlah);
27
28        $this->saldo += $jumlah;
29    }
30
31    public function cekSaldo() {
32        return $this->saldo;
33    }
34 }
```

Mari kita buat demo untuk metode pembayaran ini:

/sites/abstract-wal-interfaces/beli-pakai-debitbni.php

```
1 <?php
2 require_once "DebitBNI.php";
3 require_once "Pembeli.php";
4
5 // Melakukan pembelian dengan DebitBNI
```

```

6  try {
7      $bniKu = new DebitBNI("12345");
8      $bniKu->deposit(20000000);
9      $rahmat = new Pembeli("Rahmat Awaludin", $bniKu);
10     $rahmat->beli("Buku Menyelami Framework Laravel", 250000);
11     echo "Saldo terakhir Rp".$bniKu->cekSaldo()."\n";
12     echo $bniKu->cekNamaPembayaran();
13 } catch (Exception $e) {
14     echo $e->getMessage()."\n";
15 }

```

Jika dijalankan, hasilnya akan seperti ini :

```

1. rahmatawaludin@MorphaWorks: ~/sites/abstract-wal-interfaces (zsh)
~/sites/abstract-wal-interfaces master php beli-pakai-debitbni.php
Berhasil mengaktifkan Kartu Debit!
Mencatat transaksi deposit dana sejumlah 20000000 ke Buku Tabungan.
Mencatat transaksi transfer keluar sejumlah 250000 ke Buku Tabungan.
Terima kasih Rahmat Awaludin :)
Berhasil melakukan pembelian Buku Menyelami Framework Laravel seharga Rp250000.
Saldo terakhir Rp19750000
Anda melakukan pembayaran dengan DebitBNI
~/sites/abstract-wal-interfaces master

```

Membuat DebitBNI dengan abstract PaymentMethod

Di baris terakhir output terlihat kita menggunakan implementasi PaymentMethod dengan class DebitBNI.

Untuk implementasi Paypal, kita buat seperti ini:

/sites/abstract-wal-interfaces/Paypal.php

```

1  <?php
2  require_once 'PaymentMethod.php';
3  class Paypal extends PaymentMethod {
4      private $balance;
5
6      public function __construct($email, $password) {
7          // Ceritanya ini akses ke database
8          if ($email == "rahmat.awaludin@gmail.com" & $password == "12345")
9              $this->email = $email;
10             echo "Berhasil login ke Paypal!\n";
11         } else {
12             $pesan = "User ada user dengan username/password
13             throw new Exception($pesan);
14         }
15     }

```

```

16
17     private function kirimNotifikasi($pesan = "Informasi penting") {
18         echo "Mengirim email notifikasi $pesan ke $this->email \n";
19     }
20
21     public function kredit($jumlah) {
22         $this->kirimNotifikasi('pengeluaran dana');
23         $this->balance -= $jumlah;
24     }
25
26     public function deposit($jumlah) {
27         $this->kirimNotifikasi('penerimaan dana');
28         $this->balance += $jumlah;
29     }
30
31     public function cekSaldo() {
32         return $this->balance;
33     }
34 }

```

Terlihat disini, class Paypal ini mengimplementasikan semua method dari class abstract PaymentMethod, ini diharuskan. Karena, sebagaimana saya jelaskan di pembahasan sebelumnya, class yang meng *ekstends* abstract class harus mengimplementasikan semua abstract methodnya. Jika tidak, aplikasi akan error.

Perbedaan lain di class Paypal adalah :

- Untuk membuat object harus menggunakan email dan password yang kita hardcode di method `__construct()`. Tentunya, di kenyataannya Anda akan mengecek ini ke database.
- atribut `$balance` digunakan untuk menyimpan dana.
- Setiap kali ada transaksi uang masuk atau keluar, memanggil method `kirimNotifikasi()` yang disimulasikan akan mengirim email.

Demo dari metode pembayaran ini :

/sites/abstract-wal-interfaces/Paypal.php

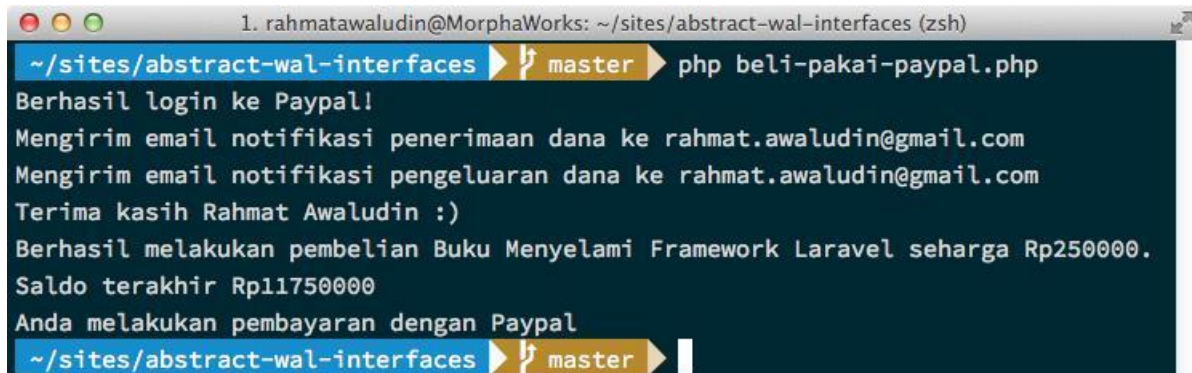
```

1 <?php
2 require_once "Paypal.php";
3 require_once "Pembeli.php";
4
5 // Melakukan pembelian dengan paypal
6 try {
7     $paypalSaya = new Paypal("rahmat.awaludin@gmail.com", "12345");
8     $paypalSaya->deposit(12000000);
9     $rahmat = new Pembeli("Rahmat Awaludin", $paypalSaya);
10    $rahmat->beli("Buku Menyelami Framework Laravel", 250000);
11    echo "Saldo terakhir Rp".$paypalSaya->cekSaldo()."\n";

```

```
12         echo $paypalSaya->cekNamaPembayaran();
13     } catch (Exception $e) {
14         echo $e->getMessage()."\n";
15     }
```

Outputnya akan menjadi :



Membuat Paypal dengan abstract PaymentMethod

Jika diperhatikan, semua syntax method yang saya gunakan disini, sama dengan method yang saya pakai di demo dengan pembayaran DebitBNI. Inilah kekuatan dari abstract class. Kita bisa melakukan standarisasi nama method, apapun bentuk implementasinya.

Itupun yang dilakukan oleh framework Laravel.

- Pernahkah Anda merubah konfigurasi server database dari MySQL ke SQL Server? Apakah syntax Eloquent nya berubah? *Tentu tidak.*
- Pernahkan Anda merubah konfigurasi email dari SMTP ke Mandrill/Mailgun? Apakah syntax untuk mengirim email di Laravel berubah? *Tentu tidak.*
- dll

Semua fleksibilitas itu muncul berkat bantuan abstraksi. Keren.

Huff.. Capek. Silahkan istirahat dulu, ambil snack dan minumannya. Oh iya, saya titip kopi capucino dan roti bakar keju ya. Saya tunggu 20 menit lagi...

....

....

Interfaces

Udah siap lagi?! Ckckck.. Anda ini memang bersemangat buat belajar Laravel.. Saya yakin dengan semangat seperti ini Anda akan menguasai framework ini kurang dari seminggu! :D

Baiklah, sekarang kita akan bahas Interfaces. Interfaces

PHP5 Namespace

PHP5 Traits

PHP5 Reflection

Composer

Jika Anda seorang developer PHP masa kini (*cieee..*), maka wajib kenal dengan Composer. Composer merupakan dependency management untuk PHP.

BAB 2

Instalasi dan Konfigurasi Laravel

Instalasi dan konfigurasi Laravel pada umumnya sama dengan framework

PHP lainnya, tapi ada sedikit perbedaan mendasar didalam cara instalasi-nya, maka didalam bab ini akan saya bahas instalasi dan konfigurasi Laravel beserta kebutuhan sistemnya.

Memulai belajar dengan menggunakan framework PHP bisa menjadi proses sulit dan melelahkan, apalagi dengan proses belajar yang sedikit agak lama. Untungnya ini tidak terjadi pada framework Laravel, karena framework Laravel seperti yang sudah dijelaskan pada Bab 1, source code yang friendly untuk para pemula pengguna framework, mudah untuk instalasi dan konfigurasi, bahkan **seseorang yang belum pernah menggunakan framework pun bisa dengan cepat mempelajarinya.**

Pada bab ini, di awal diberikan penjelasan tentang bagaimana pengembangan dengan framework Laravel secara umum. Kemudian Anda akan bertemu dengan alat/tool yang mendasari Laravel disebut sebagai PHP dependency manager yaitu

“Composer”, setelah itu dikenalkan juga struktur atau susunan file dan folder di

Laravel kemudian Anda juga akan belajar instalasi framework Laravel. Akhirnya, Anda juga akan belajar bagaimana membuat sebuah Artisan yang dibuat menggunakan Command Prompt.

2.1 Konfigurasi Sistem

Apabila Anda memiliki Laravel yang baru saja di install. Proses pembangunan untuk kebanyakan aplikasi web yang dibangun dengan Laravel umumnya terlihat seperti ini:

- 5 Konfigurasi database, cache, mail dan setting lain sesuai dengan kebutuhan.
- 5 Menciptakan titik akhir (routing) aplikasi Anda.
- 5 Membuat sebuah model dan struktur database untuk data.
- 5 Membuat controller dan mengintegrasikan dengan routing.
- 5 Membuat view template yang akan membuat sisi ke pengguna aplikasi.

- 9 Pengujian aplikasi.
- 9 Menyempurnakan kode aplikasi.

2.2 Kebutuhan Sistem

Kebutuhan sistem yang harus disediakan sebagai berikut:

- 1 Text Editor.

Pilih text editor yang sesuai dengan kebutuhan atau selera Anda. Penulis menggunakan **Notepad++** atau bisa juga menggunakan PHPStorm, Aptana, Netbeans, dan Dreamweaver.

- 1 Web Server.

Yang terpenting dalam instalasi Laravel yaitu bahwa persi PHP minimal versi 5.3 ke atas dan sudah di-install Mcrypt (salah satu ekstensi PHP). Penulis menggunakan **XAMPP versi 1.8.0** yang mendukung PHP 5.4 dan MCrypt.

- 1 **openSSL enabled di php.ini**

- 1 Composer

Seperti yang sudah dijelaskan sebelumnya bahwa Laravel berbeda dengan framework pada umumnya, jika framework pada umumnya itu download framework kemudian taruh didalam web server, setelah itu dijalankan, tetapi Laravel ini seperti pada Linux, jadi untuk update download dan sebagainya menggunakan command atau perintah didalam command prompt, Nah perintah tesebut yang dinamakan dengan composer, maka **langkah pertama untuk menjalankan Laravel adalah mendownload composer, lalu menginstallnya.**

Composer sendiri adalah fitur (dependency) tambahan untuk PHP yang memiliki basis layaknya Command Line dan berfungsi **sebagai penginstall third-party plugin untuk aplikasi web secara cepat.**

2.3 Instalasi Laravel

Sebenarnya ada 2 macam cara yang bisa dilakukan untuk melakukan instalasi Laravel, pertama melalui composer dan yang kedua langsung download di situs resminya di **<https://github.com/laravel/laravel>**.

Untuk pembahasan kali ini, kita akan menggunakan composer.

2.3.1 Instalasi Composer

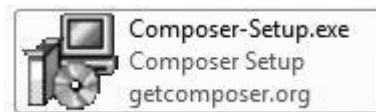
Sebelumnya, kita akan menginstall composer terlebih dahulu, langkah-langkahnya sebagai berikut:

- 2 Download dulu composer di **<http://getcomposer.org>**, lalu klik tombol **Download**. Lihat gambar 2.1.

Gambar 2.1 Website resmi composer

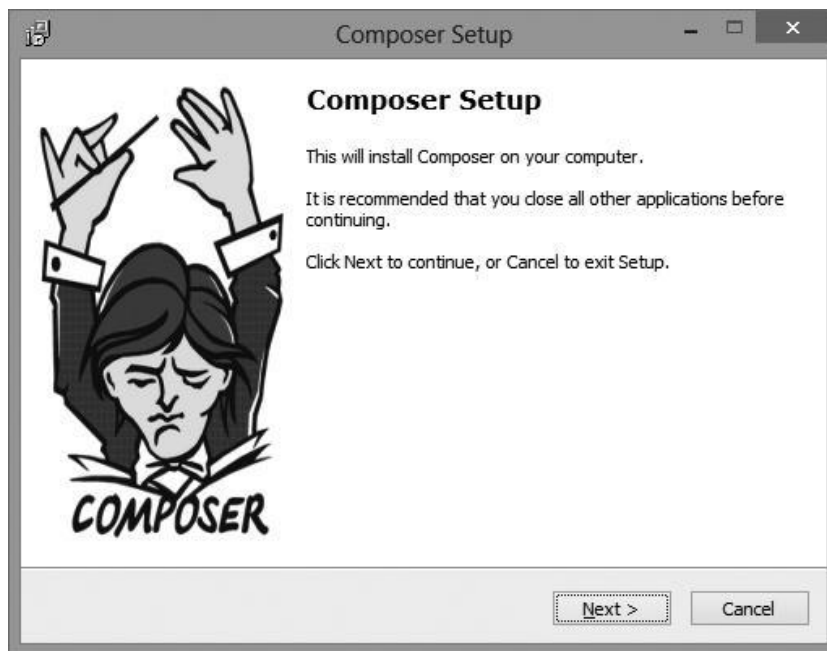


Pilih installer untuk Windows, nanti Anda akan mendapatkan file composer dengan nama **Composer-Setup.exe**. Lihat gambar 2.2.



Gambar 2.2 File installer composer untuk Windows (*Composer-Setup.exe*)

- 5 Klik 2x file **Composer-Setup.exe**, maka tampil kotak dialog Composer Setup, lalu klik tombol **Next**. Lihat gambar 2.3.



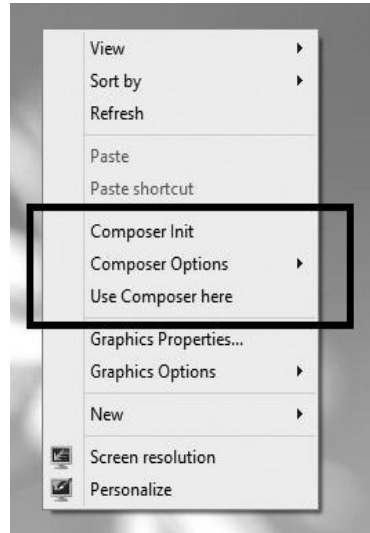
Gambar 2.3 Kotak dialog *Composer Setup*

- 9 Maka akan tampil kotak dialog Select Components, pilih **Install Shell Menus**, lalu klik tombol **Next**. Lihat gambar 2.4.



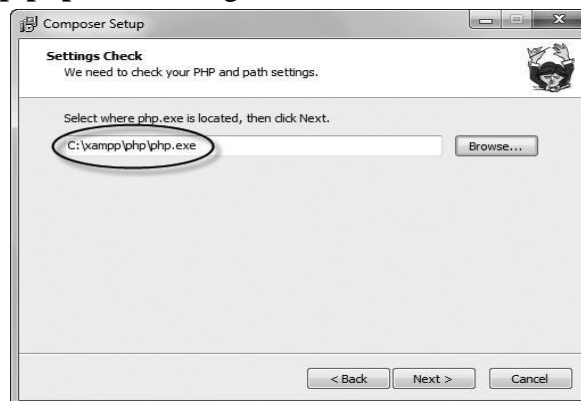
Gambar 2.4 Kotak dialog *Select Components*

Pada gambar 2.4 terdapat dua pilihan, yaitu **Install Shell Menus** dan **Do Not Install Shell Menus**, perbedaanya kalau memilih Install Shell Menus, maka ketika Anda mengklik kanan di desktop komputer Anda, maka disana terdapat beberapa menu Composer. Lihat gambar 2.5.



Gambar 2.5 Beberapa menu Composer di dalam menu klik kanan desktop

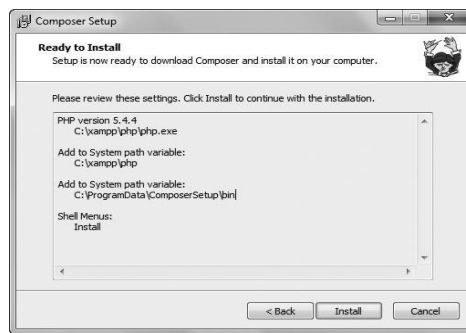
- 1 Selanjutnya akan tampil kotak dialog Settings Check, karena composer sifatnya embedded dengan PHP, maka installer composer akan langsung mencari php.exe yang ada didalam folder php. Untuk itu, cari php.exe di **C:/xampp/php/php.exe**. Lihat gambar 2.6.



Gambar 2.6 Kotak dialog Settings Check

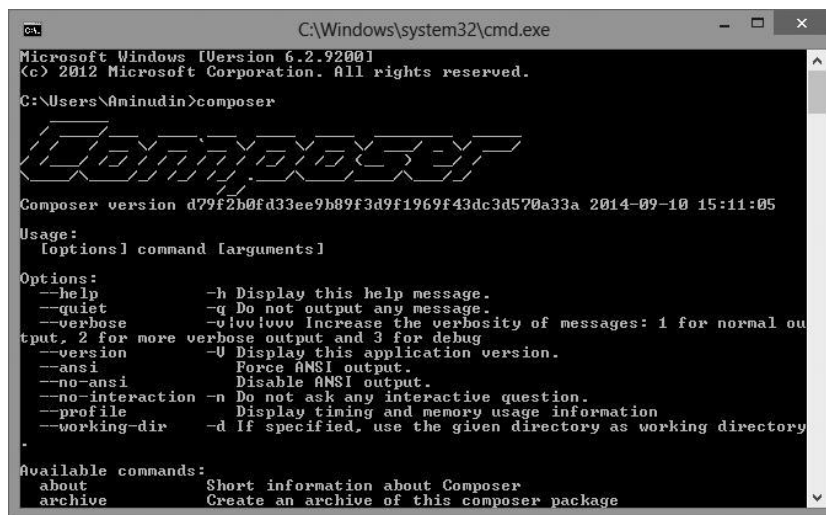
- 3 Maka akan tampil kotak dialog Ready to Install yang menampilkan informasi dan setting instalasi, klik tombol **Install**. Lihat gambar 2.7.





Gambar 2.7 Kotak dialog Ready to Install

- 1 Tunggu proses instalasi sampai selesai. Untuk menguji hasil instalasinya, masuk ke **Command Prompt**, kemudian ketikkan **composer**, lalu tekan tombol **Enter** di keyboard. Apabila instalasi berhasil, maka akan tampil seperti pada gambar 2.8.



Gambar 2.8 Instalasi composer sudah berhasil

1 INSTAL LARAVEL

INSTAL VERSI WINDOWS

LANGKAH PERTAMA

Instalasi Laravel di Windows terbilang cukup mudah dibandingkan dengan distro Linux. Sebelum melakukan instalasi Laravel ada baiknya sahabat mengecek apakah ekstensi openssl di php.ini sudah diaktifkan atau belum karena Laravel memerlukan ekstensi ini dalam mode CLI, jika belum silahkan buka php.ini untuk mengaktifkan ekstensi openssl dan cari tulisan ;extension=php_openssl.dll kemudian hilangkan tanda titik koma (;) didepannya dan simpan perubahan pada php.ini

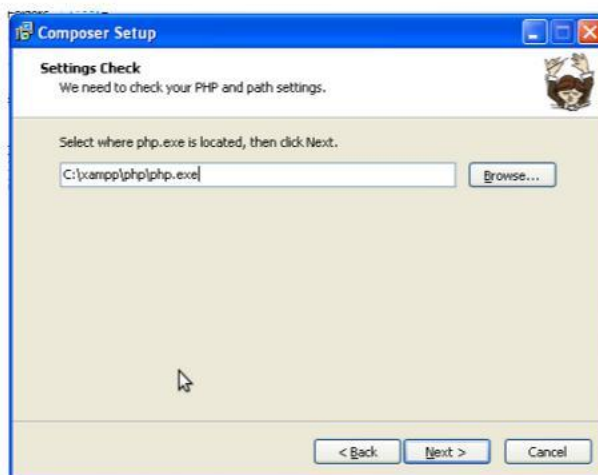
Jika langkah satu ini dilewati maka akan keluar pesan berikut ini:

Some settings on your machine make Composer unable to work properly. Make sure that you fix the issues listed below and run this script again:

The openssl extension is missing, which means that secure HTTPS transfers are impossible. If possible you should enable it or recompile php with `--with-openssl`

LANGKAH KEDUA

Langkah selanjutnya adalah kita memerlukan Composer untuk instalasi dan depensi manajemen framework Laravel, silahkan download Composer-Setup.exe Terlebih dahulu dan install.



Pada gambar di atas lokasi PHP saya adalah di `C:\xampp\php\php.exe`, nah silahkan sesuaikan dengan lokasi PHP sahabat.

LANGKAH KETIGA

Buka command prompt, kemudian pindah ke direktori htdocs dengan ketik `cd C:\xampp\htdocs` (silahkan sesuaikan dengan lokasi htdocs sahabat). Ketikan perintah di command prompt sebagai berikut untuk instalasi Laravel versi terbaru:

```
composer create-project laravel/laravel nama-proyek --prefer-dist
```

Silahkan ganti nama-proyek sesuai dengan keinginan sahabat, sekali lagi perlu diingat bahwa sahabat perlu terkoneksi ke internet untuk melakukan instalasi.

Setelah instalasi selesai silahkan buka Laravel di lokasi C:\xampp\htdocs\nama-proyek

Catatan: Jika sahabat sudah melakukan langkah 1-3, untuk berikutnya bila mau instalasi Laravel cukup lakukan langkah 3 saja.

2. INSTAL VERSI LINUX / UBUNTU

Untuk distro Linux agak terasa rumit memang, tetapi asik. Perlu sahabat ketahui bahwa saya menggunakan XAMPP for Linux dan sudah terinstal di direktori /opt/lampp, saya tidak tahu bagaimana dengan sahabat? Oleh karena itu saya akan menjelaskan konfigurasi sesuai dengan bundelan paket web server dari XAMPP tersebut pada langkah1.

Langkah 1

Saya akan menggunakan PHP dan Curl yang terdapat pada bundelan paket web server dari XAMPP tadi, nah PHP dan Curl tersebut belum terdeteksi di sistem Linux Ubuntu saya, namun apabila sahabat sudah punya instalasi web server dengan PHP dan Curl terdeteksi di sistem silahkan lewati langkah ini menuju langkah 2. Oke, supaya PHP dan Curl terdeteksi di sistem Linux maka ketikan perintah berikut di terminal:

```
$ sudo update-alternatives --install "/usr/bin/php" "php"  
"/opt/lampp/bin/php" 1 $ sudo update-alternatives --install  
"/usr/bin/curl" "curl" "/opt/lampp/bin/curl" 1
```

Langkah 2

Sekarang waktunya instal Composer, ketikan perintah berikut di terminal:

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

Langkah 3

Sekarang waktunya instal Laravel, terlebih dahulu pindah ke direktori htdocs, pada terminal dengan ketik cd htdocs (silahkan sesuaikan dengan lokasi htdocs atau lokasi virtual host sahabat). Ketikan perintah berikut untuk memulai instalasi Laravel versi terbaru:

```
composer create-project laravel/laravel nama-proyek --prefer-dist
```

Silahkan ganti nama-proyek sesuai dengan keinginan sahabat, sekali lagi saya ingatkan bahwa sahabat perlu terkoneksi ke internet untuk melakukan instalasi. Setelah instalasi maka Laravel sudah terinstal di htdocs dengan nama folder nama-proyek.

Langkah 4

Langkah terakhir adalah mengubah permisi direktori storage, ketik di terminal perintah berikut (silahkan disesuaikan dengan direktori sahabat):

```
chmod 777 -R nama-proyek/app/storage
```

Catatan: Jika sahabat sudah melakukan langkah 1-4, untuk berikutnya bila mau instalasi Laravel cukup lakukan langkah 3 dan 4 saja.

Catatan : Laravel 5.1 harus sudah instal PHP Terbaru