PENGENALAN FRAMEWORK ANGULAR BAGI PEMULA

Irfan Maulana

SOFTWARE DEVELOPMENT ENGINEER di BLIBLI.COM

1. Daftar Isi

2. Mengenal Framework Angular

Angular merupakan salah satu framework yang digunakan untuk membangun modern web application yang digawangi oleh raksasa perusahan teknologi Google. Angular yang akan kita bahas adalah Angular versi 2 keatas sesuai dengan perubahan branding dari team Angular dimana Angular versi 1 kebawah disebut AngularJS dan untuk versi 2 keatas akan disebut Angular. Pada saat ebook ini ditulis versi Angular yang akan kita gunakan adalah versi stable terakhir yakni versi **2.4.0**.

AngularJS sendiri telah dikenal sebagai salah satu framework paling popular dan paling banyak dipakai sejak versi 1.x di release ke public, namun seiring dengan pergeseran kebutuhan pembuatan web sekarang ini AngularJS menjadi sulit bersaing dan menjadi sulit untuk di kembangkan karena memiliki core behavior yang tidak lagi sejalan dengan kebutuhan para pengguna. Meskipun pada faktanya sampai ebook ini dibuat AngularJS masih menjadi framework yang paling banyak digunakan namun team Angular merasa bahwa Angular mesti melakukan perubahan mendasar pada konsep yang diusungnya, untuk alasan itu Angular versi 2 dibuat.

Angular adalah versi modern dari AngularJS dimana masih membawa kekuatan AngularJS dan mencoba memenuhi kebutuhan para pengguna yang membutuhkan performa yang jauh lebih baik disbanding pendahulunya.

Angular juga menggandeng Typescript dari Microsoft sebagai official language yang digunakan. Typescript ini sendiri merupakan bahasa pre-processor bagi Javascript dimana Typescript menambahkan kemampuan *Type Safe* atau pengenalan tipe data pada syntaxnya. Pada saat ebook ini dibuat kita menggunakan Typescript versi 2.0.0.

Selain Typescript, Angular juga menggandeng RxJS dari ReactiveX sebagai library pendukungnya. RxJS memberikan arsitektur Observer pattern pada AngularJS dimana hal ini sangat dibutuhkan saat kita berurusan dengan asynchronous / parallel programming. Sederhananya Observer pada RxJS adalah merupakan penyempurnaan dari Promise yang sangat popular digunakan para Javascript developer saat ini untuk menangani asynchronous function.

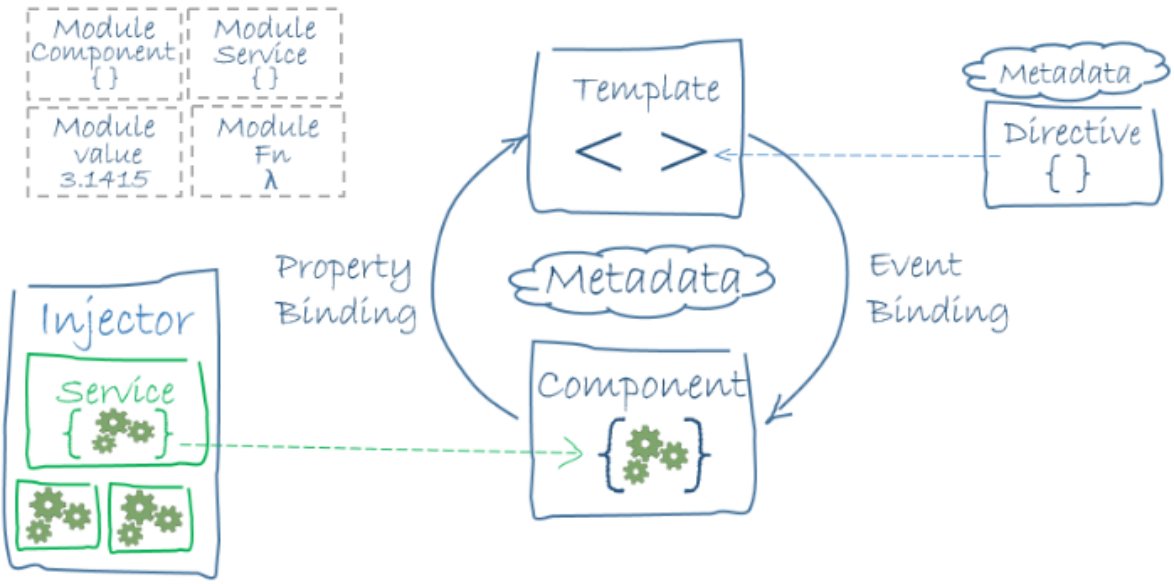
Dengan mengusung 2 vendor diatas jelas bahwa AngularJS akan sangat cocok bagi project dengan kompleksitas tinggi, karena Typescript sangat menawarkan kemampuan untuk menangani kompleksitas dengan kejelasan dan kemudahan memahami syntaxnya, sementara RxJS memiliki kemampuan mengolah dan memodelkan data dengan intensitas yang tinggi.

Menggunakan Typescript juga akan mengurangi unexpected error saat runtime dimana kesalahan syntax akan langsung dijumpai bahkan pada saat development dengan dukungan berbagai Tools di berbagai IDE.

Seperti dijelaskan diatas bahwa salah satu fokus masalah yang ingin coba dipecahkan di Angular adalah *dirty checking* yang menjadi momok bagi performance di AngularJS versi 1.x sehingga tentu saja semua hal yang dibangun di Angular saat ini selalu berorientasi pada masalah performance termasuk pada pemilihan library pendukung.

3. Arsitektur Framework Angular

Arsitektur yang dibawa oleh Angular sangat berbeda dengan AngularJS versi 1.x meskipun beberapa bagian dan istilah yang telah kita kenal di AngularJS akan kembali kita temui di Angular, jadi pada dasarnya bukan hal yang begitu sulit untuk memahami Angular bagi mereka yang sudah mempelajari AngularJS di versi 1.x.



Sumber Gambar : https://angular.io/docs/ts/latest/guide/architecture.html

Ketika kita membangun aplikasi web dengan Angular it berarti kita kan menulis kode menggunakan HTML dengan beberapa attribute yang telah di-*Angularize,* menulis per komponen dan *class* menggunakan Javascript ataupun Typescript untuk mengatur HTML tersebut, menambahkan logika ke dalam *services*, menyatukan service dan komponen ini kedalam sebuah module, kemudian mem-*bootstrapping* ke dalam module utama. Anda bisa baca mengenai arsitektur Angular disini https://angular.io/docs/ts/latest/guide/architecture.html.

4. Persiapan Pembuatan Project Angular

Ada beberapa hal yang harus disiapkan untuk memula development menggunakan framework Angular, kita coba jabarkan satu-satu berikut penjelasannya :

1. NodeJS

NodeJS menjadi salah satu hal yang wajib Anda punya ketika ingin memulai development Angular karena hampir semua tools developmentnya berjalan diatas environment NodeJS. Anda bisa langsung mulai mendownload installer NodeJS disini https://nodejs.org/en/download/ saya sarankan untuk mendownload versi LTS yang terakhir.
Anda bisa mulai menginstall NodeJS di komputer Anda dengan cara ikuti perintah di halaman website official nya. Setalah terinstall Anda bisa memastikan apakah NodeJS ini telah terinstall dengan baik atau belum dengan mengecek versi NodeJS dengan perintah `node -v`, Anda juga bisa mengecek versi NPM dengan perintah `npm -v`.

2. Angular-CLI

Beberapa dependency ada yang lebih disarankan untuk diinstall di global repository Anda salah satunya Angular-CLI ini. Angular-CLI merupakan command line interface resmi hasil buatan team Angular sendiri yang terinspirasi dari Ember-CLI. Angular-CLI sangat memudahkan kita dalam memulai project Angular karena beberapa setup sudah bisa langsung di generate dan disediakan oleh Angular-CLI sehingga kita tidak perlu terlalu pusing dengan sedemikian banyak setup dan bisa langsung fokus mempelajari frameworknya. Angular-CLI tersedia di NPM repository sehingga Anda bisa menginstallnya dengan perintah `npm i @angular/cli -g`.

Setelah terinstall anda bisa mengecek versi Angular-CLI dengan perintah `ng -v`.

5. Pembuatan Project dengan Angular

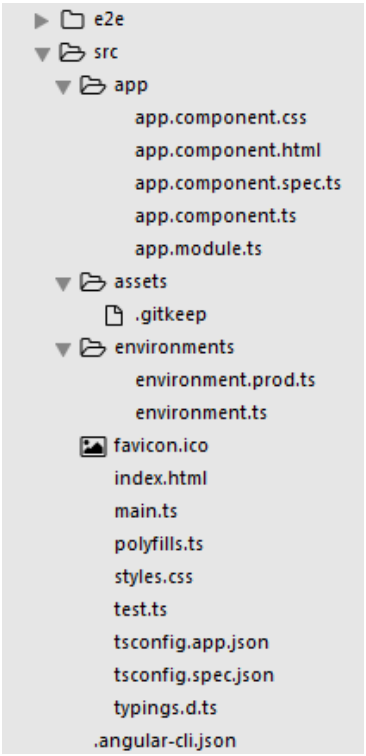
Setelah kita mempersiapkan tools diatas kita bisa memulai untuk membuat project menggunakan framework Angular.

5.1 Setup Project

Kita akan setup project Angular dengan Angular-CLI versi **1.0.0-rc.1**, katakanlah kita akan buat project dengan nama ng2-starwars maka kita bisa generate project menggunakan Angular-CLI dengan cara :

```
ng new ng2-starwars
```

Setelah menjalankan perintah ini, maka akan terbuat struktur folder project Angular yang kurang lebih seperti gambar berikut :



setelah itu kita bisa masuk ke folder project tersebut dengan menggunakan perintah `cd ng2-starwars`, untuk menjalankan project tersebut bisa menggunakan perintah `ng serve`, dan Anda bisa membuka nya di browser dengan alamat standar bawaan, yakni `http://localhost:4200`.

5.2 Memahami Beberapa File Penting

Ada beberapa file yang penting untuk dipahami untuk bisa memahami bagaimana Angular bekerja, dengan memahami beberapa fungsi file ini akan meudahkan kita juga untuk melakukan modifikasi baik penambahan, pengubahan maupun peng-hapusan file tertentu.

Yang pertama adalah file `angular-cli.json`, file ini merupakan file konfigurasi dari project yang kita buat ini. By default semua seharusnya sudah berjalan dan semua path sudah pada tempatnya seperti letak `index.html, main.ts, polyfills.ts` serta beberapa file untuk IDE configuration seperti `tsconfig.app.json`. Beberapa hal yang mungkin harus kita ganti adalah `project.name` disesuaikan dengan nama project kita, `styleExt` bila kita menginginkan menggunakan pre-processor dibandingkan menggunakan CSS biasa. Selain itu silahkan dipelajari sendiri, berikut contoh file `angular-cli.json` yang bisa ditemukan di sini https://github.com/mazipan/ng2-starwars/blob/master/.angular-cli.json.

Selanjutnya kita akan coba masuk ke folder `src` dan memahami sedikit tentang file-file yang harus kita tahu.

Ada file `styles.css` yang merupakan stylesheet global yang ingin kita terapkan ke semua component dan halaman website kita nantinya. Setelah itu kita lihat file `test.ts` yang merupakan root dari unit testing semua komponen kita, file ini akan memanggil

semua file yang memiliki nama dengan akhiran `spec.ts` di dalam folder `src` tersebut. Setelahnya kita lihat file `polyfills.ts` yang merupakan polyfills untuk beberapa browser yang belum mendukung seperti `es6, reflect` dan `zone.js`.

Selanjutnya kita akan masuk ke folder `app`, kita akan melihat kedalam file `app.module.ts` dimana file ini merupakan file yang menjadi pintu awal bagi semua kebutuhan aplikasi kita jadi didalamnya akan berisi module, komponen, directive dan hal lain yang dibutuhkan oleh aplikasi. Hasil dari generate oleh Angular-CLI akan membuat file dengan kebutuhan dasar seperti BrowserModule, NgModule, FormModule, HttpModule dan App.Component yang akan di bootstrap sebagai komponen root kita. Berikut kurang lebih kode hasil generate dari Angular-CLI :

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

5.3 Membuat Komponen Sederhana

Untuk belajar membuat komponen sederhana dalam Angular kita butuh melihat contoh bagaimana cara Angular-CLI membuat komponen nya, untuk hal ini kita akan melihat file `app.component.ts` di dalam folder `app.` Kita lihat file tersebut kurang lebih berisi seperti ini :

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app works!';
}
```

Kita bisa belajar bahwa membuat komponen sederhana bisa semudah itu, kita perlu menyiapkan file .html sebagai template dan file .css yang berisi style yang akan diaplikasikan ke komponen tersebut dimana kedua file tersebut bisa juga kita tiadakan bila kita memilih menggunakan internal template dan style tanpa menggunakan external file lagi. Yang perlu diperhatikan dalam membuat komponen pertama adalah pastikan ambil dependensi Component dari `@angular/core` dengan cara meng-importnya, setelahnya bagian komponen ditandai dengan diawali decorator `@Component` dimana di dalamnya ada beberapa built-in config yang bisa kita gunakan seperti `selector` yang merupakan HTML komponen yang akan kita gantikan posisinya dengan komponen terebut seperti dalam index.html ada `<app-root>Loading…</app-root>`, kemudian ada `tempateUrl` yang merupakan path dari posisi template HTML kita, bisa juga digantikan dengan `tempate` saja yang artinya menggunakan internal template menggunakan ES6 yakni tick symbol (``) untuk menambahkan html di dalamnya, ada juga `styleUrls` yang berisi

array dari style yang akan kita pakai pada komponen tersebut. Selanjutnya yang dilakukan adalah menambahkan `export` agar bisa digunakan atau dipanggil oleh file lain. Di dalam export ini juga kita bisa menambahkan berbagai variable data yang nantinya akan kita gunakan di dalam template kita seperti dalam gambar diatas ada `title` maupun berbagai fungsi dan kemampuan lain yang kita harapkan ada dalam komponen tersebut. Untuk menggunakan data ini dalam file template pun sederhana dan hampir sama dengan penggunaan di dalam AngularJS v.1.x sebelumnya yakni menggunakan `{{title}}`.

5.4 Membuat Komponen Nested

Membuat komponen didalam komponen, ya karena Angular menggunakan konsep komponen dimana kesemua DOM nya dipandang sebagai satu komponen, maka pada prakteknya kita akan memecah DOM kita kedalam beberapa komponen kecil yang akan sangat mungkin terjadi nesting atau bersarang seperti HTML pada umumnya. Pada dasarnya tidak ada yang berbeda dengan cara kita membuat komponen sederhana diatas hanya saja kita perlu menggantikan DOM yang ada didalam template menggunakan selector tag komponen yang akan kita gunakan. Misalkan komponen root kita `app.component.ts` di dalam template nya akan berisi komponen lain yakni `header.component.ts` dengan selector HeaderBlock maka kita bisa mengganti header yang ada di HTML template dari komponen root kita dengan `<HeaderBlock></ HeaderBlock>`.

5.5 Contoh Menarik Data API

Pada bagian ini kita akan belajar dan praktek bagaimana cara menarik data dari suatu API kemudian ditampilkan kedalam view template menggunakan Angular dimana kita akan mengambil contoh utnuk menarik data dari public API starwars api (https://swapi.co).

Kita akan membuat satu contoh yakni mengambil data film yang bisa diakses di url API https://swapi.co/api/films/, pertama saya akan menyiapkan file layaknya enum namun saya tidak ingin menggunakan enumeration maka saya buat file class dengan static final variable dibawah folder `Helpers` dengan nama file `UrlCollection.ts` seperti berikut :

```
export class UrlCollection {
  public static readonly FILM = "https://swapi.co/api/films/";
  public static readonly PEOPLE = "https://swapi.co/api/people/";
  public static readonly PLANET = "https://swapi.co/api/planets/";
  public static readonly SPECIES = "https://swapi.co/api/species/";
  public static readonly STARSHIP = "https://swapi.co/api/starships/";
  public static readonly VEHICLE = "https://swapi.co/api/vehicles/";
}
```

Setelahnya kita perlu siapkan class Film sebagai object penampung data yang akan digunakan sebagai model dari data Film ini sendiri dengan kode yang telah saya minimalisasi kurang lebih sebagai berikut :

```
export class Film {
  constructor(
    public title: string,
    public episode_id: number,
    public opening_crawl: string,
    public director: string,
    public producer: string,
    public release_date: string,
  ) { }
}
```

Setelah kita membuat class Film kita akan membuat file converter dari response yang memiliki tipe any ke dalam tipe Film, kita akan membuat file dibawah folder `Helpers` dengan nama file `ObjectConverter.ts` dengan kode seperti berikut :

```
import { Film }  from '../Film/Film';

export class ObjectConverter {

  public convertResponseToFilm(r:any): Film{

    let film = <Film>({
      title: r.title,
      episode_id: r.episode_id,
      opening_crawl: r.opening_crawl,
      director: r.director,
      producer: r.producer,
      release_date: r.release_date,
    });

    return film;
  };

}
```

Berikutnya kita akan membuat Service sebagai ujung tombak dari flow kita dalam menarik data API karena di dalam service ini kita akan melakukan request permintaan data. Service sendiri merupakan bagian dalam Angular yang Injectable artinya bisa disisipkan kedalam komponen untuk digunakan baik sekali maupun berulang oleh satu komponen atau lebih.
Kita akan membuat file dengan nama `film-list-service.ts`, pertama kita harus mengimport beberapa kebutuhan yang akan digunakan dalam Service kita ini seperti `Injectable, Http, Response, Observable, rxjs/add/operator/map, rxjs/add/operator/catch, UrlCollection dan ObjectConverter`, Kodenya kurang lebih sebagai berikut :

```
import { Injectable }     from '@angular/core';
import { Http, Response} from '@angular/http';
import { Film }    from './Film';
import {Observable} from 'rxjs/Rx';

// Import RxJs required methods
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';

import { UrlCollection } from '../Helpers/UrlCollection';
import { ObjectConverter } from '../Helpers/ObjectConverter';
```

Service sendiri akan berada dalam decorator `@Injectable`, di dalamnya kita akan menambahkan constructor http sebagai module untuk melakukan request data dan menambahkan satu public function yakni `getFilms()` yang akan mengembalikan nilai sebagai suatu `Observable<Film[]>`, kodenya kurang lebih sebagai berikut :

```
@Injectable()
export class FilmService {
  // Resolve HTTP using the constructor
  constructor (
    private http: Http,
  ) {}

  getFilms() : Observable<Film[]> {

    let thisService = this;
    let objectConverter = new ObjectConverter();

    function mapFilmResponse(response:Response): Film[]{
      // The response of the API has a results
      // property with the actual results
      return response.json().results.map(objectConverter.convertResponseToFilm)
    }

    // ...using get request
    return this.http.get(UrlCollection.FILM)
      .map(mapFilmResponse)
      //...errors if any
      .catch((error:any) => Observable.throw(error.json().error || 'Server error'));

  }

}
```

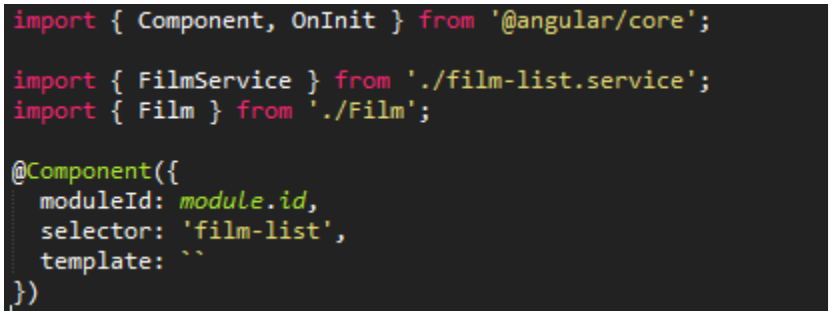
Setelah file Service kita jadi jangan lupa untuk menambahkan kedalah Array provider dalam `app.module.ts` tentunya setelah diimport terlebih dahulu Service tersebut di dalam `app.module.ts` dengan cara seperti berikut :

```
import { FilmService } from './Film/film-list.service';
```

Kemudian pada bagian provider ditambahkan kode berikut :

```
providers: [
    FilmService,
],
```

Terakhir kita buat file `film-list.component.ts` dimana kita akan menambahkan beberapa dependency sebagai berikut :



```
import { Component, OnInit } from '@angular/core';

import { FilmService } from './film-list.service';
import { Film } from './Film';

@Component({
  moduleId: module.id,
  selector: 'film-list',
  template: ``
})
```

Kode template sengaja kita kosongkan terlebih dahulu agar lebih mudah dan berurut dalam memahami alur pembuatan kodenya, berikutnya kita akan membuat kode untuk meng-inject Service kedalam komponen dan menggunakan fungsi mengambil data dar API yang telah kita buat di file Service kita pada tahap sebelumnya untuk kemudian memindahkan kedalam variable yang bisa digunakan oleh template kita nantinya, kita lihat kode berikut :

```
export class FilmListComponent implements OnInit {

  films: Film[];


  constructor(
    private filmService: FilmService
  ){}

  ngOnInit() {
    this.loadDataFilms()
  }

  loadDataFilms() {
    this.filmService.getFilms()
      .subscribe(
        films => this.films = films,
        err => {
          console.log(err);
        });
  }

  trackByEpisodId(index:number, film:Film) {
    return film.episode_id;
  }

}
```

Kita lihat kode diatas, dimana untuk menggunakan atau meng-inject service ke dalam komponen cukup dengan menambahakan constructor service tersebut kedalam komponen. Bisa kita lihat juga bahwa komponen tersebut implements `ngOnInit` yang artinya kita bisa melakukan sesuatu pada saat komponen tersebut di inisialisasi kedalam tampilan, dimana pada contoh diatas kita akan meload data film pada saat di inisialisasi. Kita juga membuat fungsi `trackByEpisodId` karena pada Angular tidak ada lagi fungsi `trackBy` seperti yang biasa digunakan pada AngularJS v.1.x.

Terakhir kita akan menampilkan data film kedalam template view yakni dengan menambahkan tag html kedalam template kita, seperti berikut :

```
@Component({
  moduleId: module.id,
  selector: 'film-list',
  template: `
      <h2>Films</h2>

      <table class="table">
        <thead class="table__head">
        <tr>
          <th>#</th>
          <th>Title</th>
          <th>Episode</th>
          <th>Release Date</th>
        </tr>
        </thead>
        <tbody class="table__body">
        <tr *ngFor="let film of films; let i = index; trackBy:trackByEpisodId;">
          <td>{{ i+1 }}</td>
          <td>{{ film.title }}</td>
          <td>{{ film.episode_id }}</td>
          <td>{{ film.release_date }}</td>
        </tr>
        </tbody>
      </table>
  `
})
```

Bisa diperhatikan juga bahwa ada sedikit perbedaan melakukan looping didalam template antara AngularJS v.1x dengan Angular.

5.6 Routing Menggunakan Angular

N/A (Content In Progress)

6. Setup Untuk Production

Seperti kita tahu bahwa untuk siap production ada beberapa tahap yang harus dipenuhi mulai dari compressing file sekecil mungkin, penghilangan dependency yang tidak dibutuhkan dan membaginya kedalam beberapa file berdasarkan fungsinya. Untungnya kita sudah sangat dimudahkan dengan adanya Angular-CLI sehingga kita tidak perlu banyak melakukan setup lagi untuk menyiapkan file yang layak ditampilkan di live production kita. Pada dasarnya Angular-CLI memanfaatkan kehebatan webpack dalam menangani ini semua namun dengan mengurangi segala kerumita konfigurasi yang ad ajika kita manual menggunakan webpack sendiri. Untuk mem-bundle semua file ini agar siap production kita bisa menggunakan perintah `ng-build` yang ketika dijalankan akan membuat satu folder baru yakni `dist` dan berisikan file `index.html, inline.js main.js, vendor.js dan style.css` kesemuanya telah di minify, dikompress, ditambahkan hash serta dibuatkan gzipped file nya. Mungkin hanya sedikit yang biasanya perlu disesuaikan yakni `<base href=”/”>` yang seharusnya diisikan dengan domain kita akan meletakan di production kita nantinya.

7. Bahan Belajar

Semua kode sumber yang ada didalam ebook ini bisa dilihat di github repository : http://mazipan.github.io/ng2-starwars/

Angular Quick Start : https://angular.io/docs/ts/latest/quickstart.html

Dokumentasi Angular : https://angular.io/docs/ts/latest/

Facebook Angular Indonesia : https://m.facebook.com/groups/462764390497214

7. Tentang Penulis

Irfan Maulana, saat menulis ebook ini masih bekerja sebagai Software Development Engineer di salah satu ecommerce di Indonesia Blibli.com. Selalu senang untuk berbagi pengetahuan kepada teman-teman di luar sana. Antusias dengan komunitas yang berkaitan dunia pemrograman di Indonesia khususnya di Jakarta.

Email : mazipanneh@gmail.com
Blog : mazipanneh.com
Facebook : /mazipanneh
Twitter : @Maz_Ipan
Linkedin : /in/irfanmaulanamazipan