

Hybrid App

David

目錄

前言

What is Ionic?

環境安裝

建立App

What is Angular?

Angular架構

常用指令



前言

如果想要寫手機的 App，要學 Java 寫 Android App、學 Swift 寫 IOS App，假如需要同時開發 IOS及Android，就可以使用Hybrid App，跨越不同作業系統。

Hybrid App：混合語言程式的部份代碼會以 Web 技術編寫，如 HTML5、CSS 和 JavaScript 優點是程式碼能夠跨越不同的作業平台，不需要為每個操作系統編寫特定的編碼。



What is Ionic?

Hybrid Mobile開發技術使開發者能去開發一個運行在多個平台上的移動應用。不用學習相應的平台語言和使用已經存在的技術，其中一個Hybrid Mobile開發新平台就是Ionic。

Ionic 是一個HTML5(Hybrid Mobile)技術App框架。它是一個可以使用HTML5來創建一個移動應用的前端開源框架。

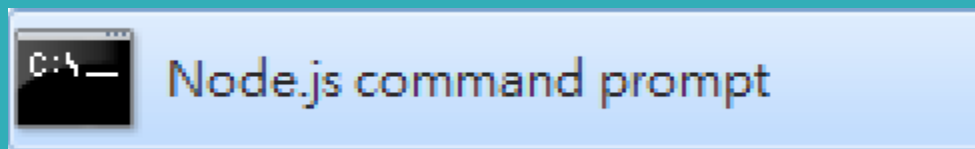


環境安裝

安裝的部份包括 NodeJS 、 Cordova 、 Ionic
Angular CLI 、 TypeScript

至NodeJS官網安裝node.js，如下列網址↓
<https://nodejs.org/en/>

再開啟NodeJS的命令列。



環境安裝

於NodeJS命令列安裝Cordova及Ionic，如下。

```
npm install -g cordova ionic
```

再繼續安裝Angular CLI及TypeScript

```
C:\Users\David>npm install -g typescript
```

```
C:\Users\David>npm install -g @angular/cli
```



建立App

於NodeJS命令列鍵入關鍵字↓

```
C:\Users\David>ionic start myApp tabs
```

安裝好後轉換資料夾並開啟專案，如下↓

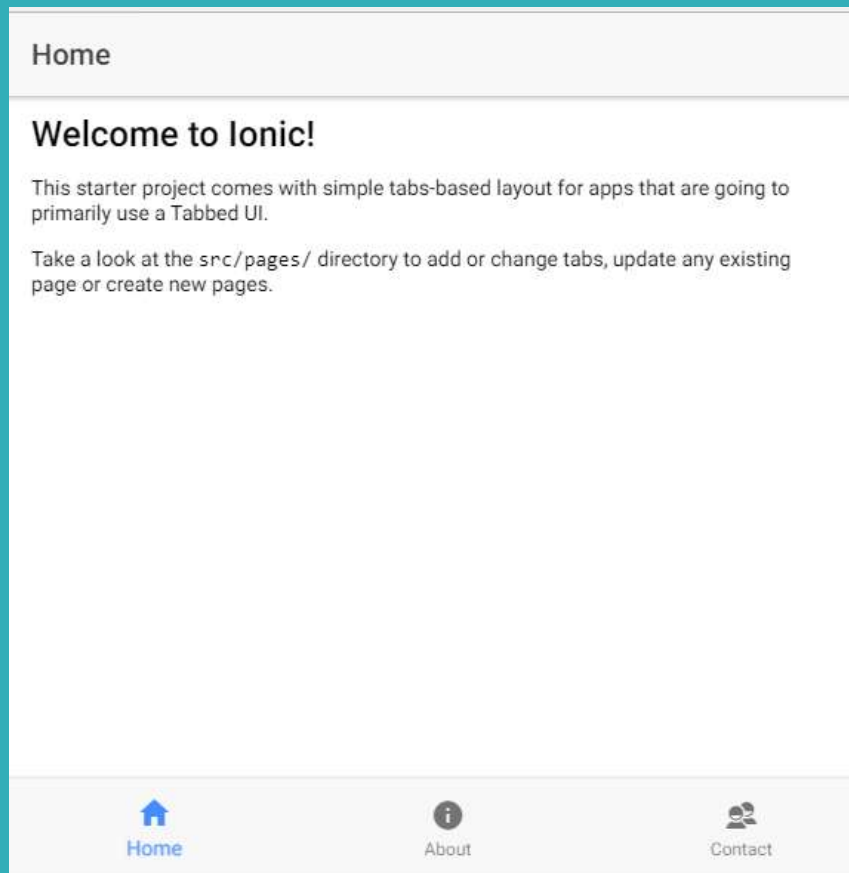
```
C:\Users\David>cd myApp
```

```
C:\Users\David\myApp>ionic serve
```



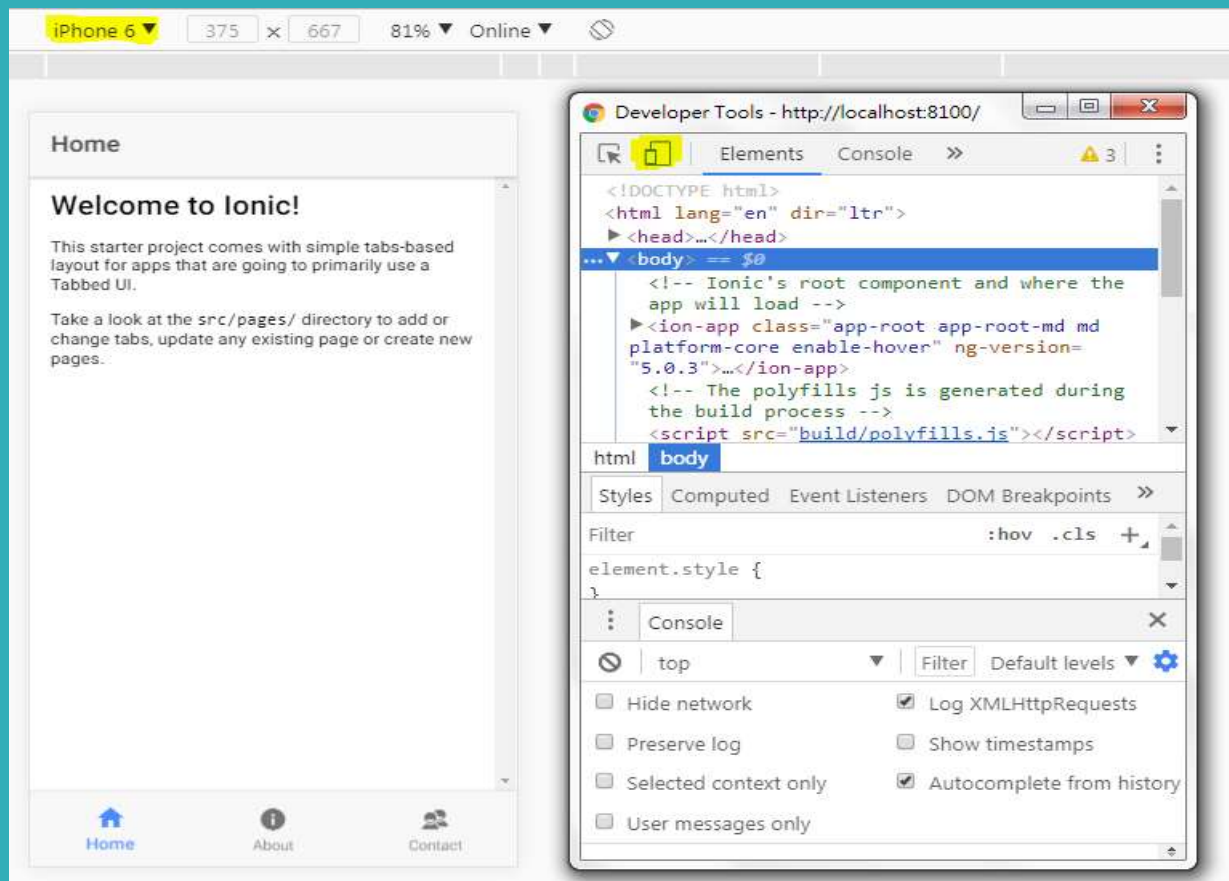
建立App

開啟後頁面如下，預設Port：8100



建立App

Chrome 點擊 F12 開啟 開發者工具，如下圖螢光塗料的按鈕，就可以選擇手機款式模擬。



建立App

VS Code 開啟App

於NodeJS命令列鍵入關鍵字↓，就可以將App
用VS Code 開啟

```
C:\Users\David\myApp>code .
```



What is Angular?

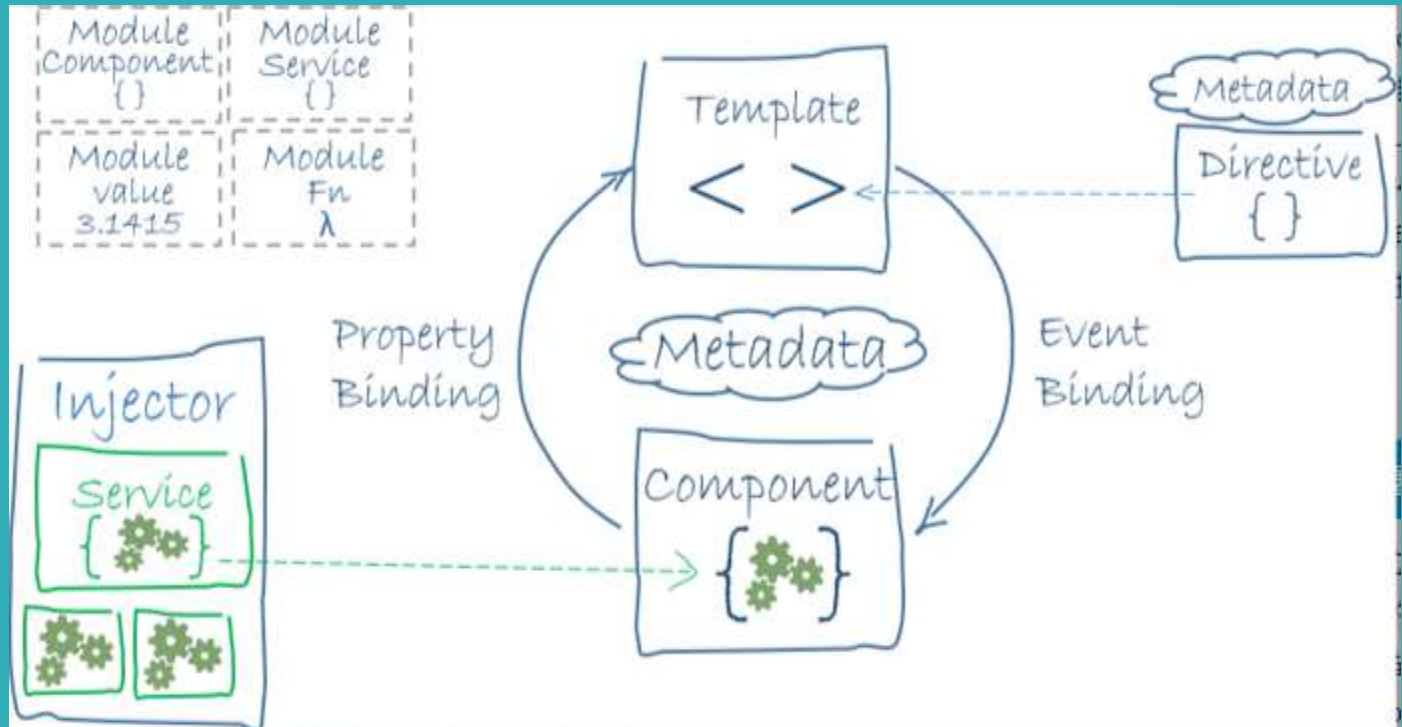
Google開發出來的一款開源 JavaScript框架，直接延伸現有的HTML架構，協助單一頁面應用程式運行。

Angular使開發人員能建立一個同時存在於Web、Mobile及Desktop的應用程式，也就是跨平台支援。



Angular 架構

Angular 的主要架構就是Components。



Angular 架構

其它架構還包含

Module

Template

Metadata

Data Binding

Service

Directive

Dependency Injection



Angular 架構

Component

Component 是用 TS 寫出來的 Class，這些 Component 可以被包裝變成獨立的 Module 讓其他 Module 使用，如下圖。

MetaData、Template

用來描述元件的相關設定。

Component 用稱作 decorator function的 @Component來表示 Metadata。

selector：告訴 Angular 這個 Component 要放在 html 裡的自訂標籤名稱 `<my-app>`。

template：置入HTML。其中`{{ }}`為Angular Data Binding的語法，裡面放著Component的屬性，如圖為name。

```
app/app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: `<h1>Hello {{name}}</h1>`
})
export class AppComponent { name = 'Angular'; }
```



Angular 架構

Module

一個Module可以包裝很多個Component。

imports：預期有一個陣列，包含要引入的模組。

declarations：預期有一個陣列，包含要引入的元件

bootstrap：定義根組件 (root component)，通常只給定一個，就是Angular cli所產生的app.module.ts



app/app.module.ts

```
import { NgModule }      from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
import { AppComponent }  from './app.component';
```

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



注入Component

Angular 架構

Data Binding

內嵌繫結(interpolation) :

{{property}}

屬性繫結(property binding) :

[property] = 'David'

事件繫結(event binding) :

(event) = 'getName(\$event)'

雙向繫結(Two-way binding) :

[[ngModel]] = 'property'



Angular 架構

Service：集中管理資料與運算邏輯

要成為可被注入的 Service 必須要加上 @Injectable() 的 Metadata。

app/hero.service.ts

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

@Injectable()
export class HeroService {

    // do something...

}
```

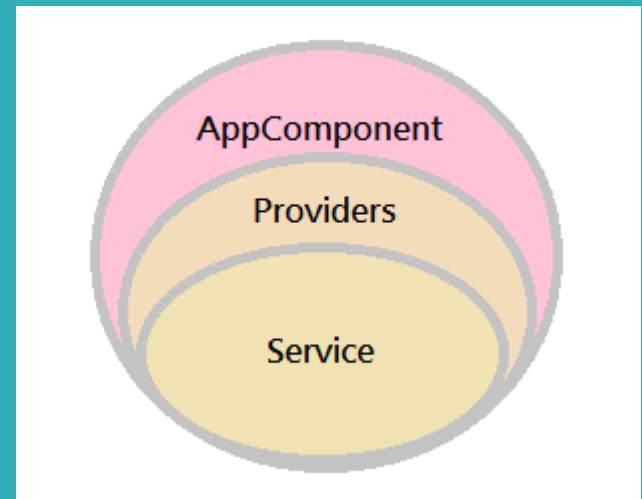


Angular 架構

註冊Service的方式有兩種

Component Providers :
將 Service 註冊在 Component 的 Providers 中
只有該Component(AppComponent)可以使用

```
@Component({  
  providers: [Service]  
})  
  
export class AppComponent{  
  // do something...  
}
```



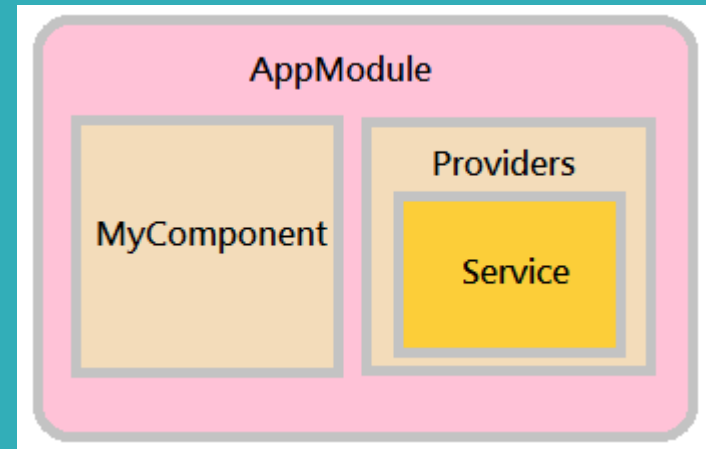
Angular 架構

Module Providers ↓

將 Service 註冊再 Module 裡，Module中所有的Component都會共用Service這個實體。

下圖範例：AppModule裡有一個MyComponent，而這個Component所使用的Service就是NgModule所提供的Service。

```
@NgModule({  
  providers:[Service]  
})  
export class AppModule { }  
  
@Component({  
  // ...  
})  
  
export class MyComponent {  
  constructor(private sampleService: Service) {  
    // ...  
  }  
}
```



Angular 架構

Dependency Injection :

由「相依注入」機制管理物件生命週期。

Angular DI 的運作是使用 Constructor Injection，也就是把實例化的物件從建構子傳入。

如下圖範例，Service的實體是透過Component的providers所建立。



```
app/app.component.ts
import { Component, OnInit } from '@angular/core';

import { Hero } from '../hero';
import { HeroService } from '../hero.service';

@Component({
  providers: [HeroService]
})
export class AppComponent implements OnInit {

  constructor(private heroService: HeroService) { }
}
```

Angular 架構

Directive : DOM 轉換為多功能的「宣告命令」，總共分為三種。

Components directives :

Component是最常用的directives，用來控制 HTML Template。



Angular 架構

結構性Directive (Structural directives) :
可藉由新增或刪除 DOM元素，來改變DOM(view)
結構。

例如：ngIf、ngFor、ngSwitch，這些語法都可以
控制 DOM 結構。

ngIf、ngFor、ngSwitchDefault、ngSwitchCa
se前面要加上 *。



Angular 架構

```
app/app.component.ts
import { Component } from '@angular/core';

import { Team } from './hero';

@Component({
  selector: 'app-root',
  template: `
    <h1>{{title}}</h1>
    <h2>My favorite team is: {{myTeam.name}}</h2>
    <p>Heroes:</p>
    <ul>
      <li *ngFor="let team of teams">
        {{ team.name }}
      </li>
    </ul>
    <p *ngIf="teams.length > 2">There are many teams!</p>
    <div [ngSwitch] = "teams[0].rank">
      <div *ngSwitchCase = "1">{{teams[0].name}} ranks 1st </div>
      <div *ngSwitchCase = "2">{{teams[0].rank}} ranks 2nd </div>
      <div *ngSwitchCase = "3">{{teams[0].rank}} ranks 3rd </div>
      <div *ngSwitchCase = "4">{{teams[0].rank}} ranks 4rd </div>
      <div *ngSwitchDefault>unknown</div>
    </div>
  `
})
export class AppComponent {
  title = 'CPBL';
  teams = [
    new Team(1, 'BrotherElephant'),
    new Team(2, 'Braves'),
    new Team(3, 'Lamigo'),
    new Team(4, 'Lions')
  ];
  myTeam = this.teams[0];
}
```

內嵌屬性，對應到Component

的title屬性

將元素動態插入

DOM Element中

ngIf裡面放條件式，條件為否，則將Element移除DOM

透過給定的條件，呈現不同的元素，可以針對同一個值做多次判斷

Preview

CPBL

My favorite team is: BrotherElephant

Heroes:

- BrotherElephant
- Braves
- Lamigo
- Lions

There are many teams!

BrotherElephant ranks 1st



Angular 架構

屬性Directive (Attribute directives) :

用來改變DOM元素的外觀或行為。

例如：[ngStyle]，是用來改變HTML 元素的Style。

下圖範例的方法：getbackgroundColor，就是再Angular 裡定義的Function，顯示顏色的邏輯可以再方法裡面做異動。

```
<p [ngStyle]="{backgroundColor: getbackgroundcolor()}"></p>
```



Angular 架構

範例：

下圖為一個AppComponent，它引進了兩個Component，此Component的template有個[default] Attribute，依照goodbye-component 這個tag，就會連結至GoodbyeComponent。

app/app.component.ts

```
import {Component, Input} from 'angular2/core';  
import HelloComponent from './hello.component';  
import GoodbyeComponent from './goodbye.component';
```



```
@Component({  
  selector: 'app', 引進Component  
  template: `<div>
```

```
    <hello-component [myName]="name"></hello-component>  
    <hello-component myName="Other World"></hello-component>  
    <goodbye-component [default]="name"></goodbye-component>  
    <goodbye-component default="Your World"></goodbye-component>  
  </div>`
```

```
  , directives: [HelloComponent, GoodbyeComponent] 記得加上此段，否則無法使用  
})
```

```
export class AppComponent {  
  name: string;  
  
  constructor() {  
    this.name = "World";  
  }  
}
```



Angular 架構

GoodByeComponent :

@Input('default')，這裡的default就是AppComponent(父)所給定的Attribute[default] 名稱，如下圖。

```
<goodbye-component [default]="name"></goodbye-component>
```

假如(父)所給定的Attribute與(子)Attribute名稱相同，只須填寫 `@Input() myName: string;` 即可。



```
app/goodbye.component.ts
import {Component, Input} from 'angular2/core';

@Component({
  selector: 'goodbye-component', Tag 名稱
  template: '<p>Good Bye, {{myName}}!</p>',
})
export default class GoodbyeComponent {
  @Input('default') myName: string;
  constructor() {}
}
```

myName互相對應

Angular 架構

HelloComponent :

和GoodByeComponent不一樣的地方為，此處是在Component裡加入Inputs，兩者都可以成功執行。



```
app/hello.component.ts
import {Component, Input} from 'angular2/core';

@Component({
  selector: 'hello-component',
  template: '<p>Hello, {{myName}}!</p>',
  inputs: ['myName']
})
export default class HelloComponent {
  myName: string;
  constructor() {}
}
```

Angular 架構

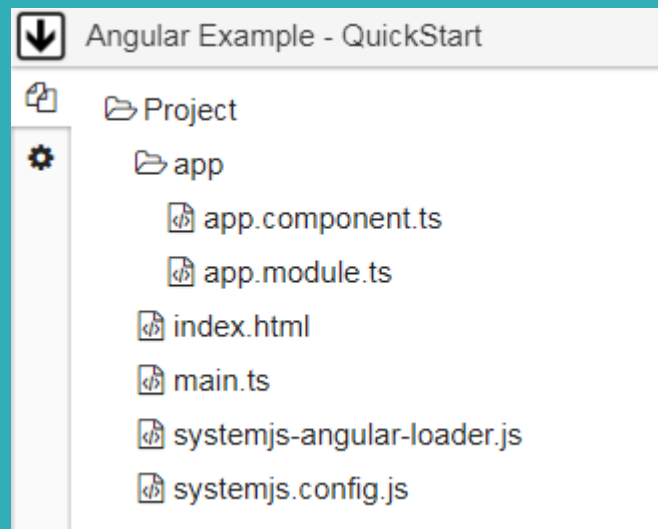
Angular 專案架構大致如下圖。

main.ts → 預設Angular的啟動器(主程式)

index.html → 網站預設的首頁

app.module.ts → 程式Module的定義檔

app.component.ts → 根元件主程式



index.html

Body 裡的 `<my-app>` Tag，會對應至根元件(AppComponent)裡的Component directive 宣告。

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Angular Quickstart</title>
    <script>document.write('<base href="' + document.location + '" />');</script>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
      body {color:#369;font-family: Arial,Helvetica,sans-serif;}
    </style>

    <!-- Polyfills -->
    <script src="https://unpkg.com/core-js/client/shim.min.js"></script>

    <script src="https://unpkg.com/zone.js@0.7.4?main=browser"></script>
    <script src="https://unpkg.com/systemjs@0.19.39/dist/system.src.js"></script>
    <script src="systemjs.config.js"></script>
    <script>
      System.import('main.js').catch(function(err){ console.error(err); });
    </script>
  </head>
  <body>
    <my-app>Loading AppComponent content here ...</my-app>
  </body>
</html>
```

根元件的Component directive宣告



main.ts

先引入所需的啟始模組，
再設定AppModule為起始模組。

main.ts

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';  
import { AppModule } from './app/app.module';  
platformBrowserDynamic().bootstrapModule(AppModule);
```

設定起始模組



app.module.ts

imports : 宣告所需模組(Module)

declarations : 宣告所需元件(Component)

bootstrap : 宣告根元件(root)

app/app.module.ts

```
import { NgModule }      from '@angular/core';
import { BrowserModule }  from '@angular/platform-browser';
import { AppComponent }   from './app.component';
```

```
@NgModule({
  imports:      [ BrowserModule ], ➡ 宣告引入所需的外部模組。
  declarations: [ AppComponent ], ➡ 宣告引入所需的元件。
  bootstrap:    [ AppComponent ] ➡ 宣告根元件，通常只有一個。
})
export class AppModule { }
```



app.component.ts

app/app.component.ts

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

@Component({
  selector: 'my-app', ⇒ Component directive 選擇器
  template: `<h1>Hello {{name}}</h1>` ⇒ 樣版
})
export class AppComponent { name = 'Angular'; }
```

類別 屬性



常用指令

cd : 轉換資料夾 , 語法為 cd + 資料夾名稱

```
C:\Users\David>cd angular-tour-of-heroes
```



常用指令

建立component：

語法為 `ng g component + component 名稱`

```
C:\Users\David\angular-tour-of-heroes>ng g component hero-detail
create src/app/hero-detail/hero-detail.component.html (30 bytes)
create src/app/hero-detail/hero-detail.component.spec.ts (657 bytes)
create src/app/hero-detail/hero-detail.component.ts (288 bytes)
create src/app/hero-detail/hero-detail.component.css (0 bytes)
update src/app/app.module.ts (559 bytes)
```



常用指令

src/app 資料夾底下會自動加入hero-detail(component名稱)，並且建立四個所需檔案，如紅色方框處。



```
src
├── app
│   └── hero-detail
│       ├── # hero-detail.component.css
│       ├── <> hero-detail.component.html
│       ├── TS hero-detail.component.spec.ts
│       └── TS hero-detail.component.ts
```

常用指令

app.module.ts這個檔案會根據剛剛所建立的Component做出更新，import {...} 引進component，並且再NgModule宣告剛剛所建立的component。

```
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { FormsModule } from '@angular/forms'
4
5  import { AppComponent } from './app.component';
6  import { HeroesComponent } from './heroes/heroes.component';
7  import { HeroDetailComponent } from './hero-detail/hero-detail.component';
8
9
10 @NgModule({
11   declarations: [
12     AppComponent,
13     HeroesComponent,
14     HeroDetailComponent
15   ],
16   imports: [
17     BrowserModule,
18     FormsModule
19   ],
20   providers: [],
21   bootstrap: [AppComponent]
22 })
23 export class AppModule { }
```



常用指令


建立service：

語法為 `ng g service+ service名稱`

```
C:\Users\David\angular-tour-of-heroes>ng g service hero
create src/app/hero.service.spec.ts (362 bytes)
create src/app/hero.service.ts (110 bytes)
```



常用指令



```
src
├─ app
│   ├── hero-detail
│   ├── heroes
│   ├── # app.component.css
│   ├── <> app.component.html
│   ├── TS app.component.spec.ts
│   ├── TS app.component.ts
│   ├── TS app.module.ts
│   └─ TS hero.service.spec.ts
│      TS hero.service.ts
│      TS hero.ts
│      TS mock-heroes.ts
```

src 資料夾底下會有剛剛所新增的service，如紅色方框處。

常用指令

建立module：

語法為 `ng g module+ module名稱`

註：`--flat` 是將檔案放至 `src/app` 底下，而不是在建立一個屬於自己的資料夾。

`--module = app` 是告訴cli，將該Module註冊於 `AppModule`的`imports`陣列裡。



```
C:\Users\David\angular-tour-of-heroes>ng g module app-routing --flat --module=app
create src/app/app-routing.module.ts (194 bytes)
update src/app/app.module.ts (853 bytes)
```

參考資料

Ionic 官方網站

<https://ionicframework.com/docs/pro/>

使用Ionic創建一個簡單的APP

<https://www.gitbook.com/book/afgnsu/ionic-app/details>

Angular 2 之 30 天邁向神乎其技之路

<https://ithelp.ithome.com.tw/users/20103745/ironman/1160>



參考資料

Angular 官方網站

<https://angular.io/>

快快樂樂學習Angular

<https://goo.gl/eMTWae>

Angular 教學

<https://blog.johnwu.cc/article/angular-4-services.html>



THE END

