**目录**

1. 2018/2/28　……………………………………………………………　架构概览

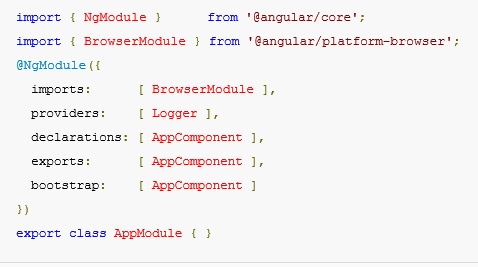
**架构概览**

要点：

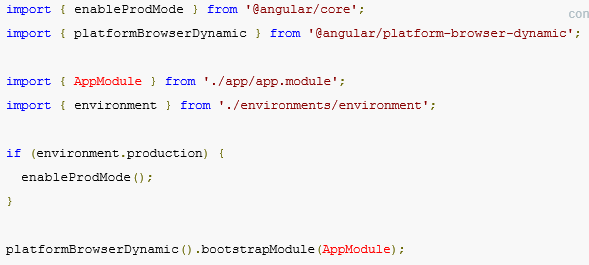
* Angular 是一个用 HTML 和 JavaScript 或者一个可以编译成 JavaScript 的语言来构建客户端应用的框架。
* 写法: 用 Angular 扩展语法编写 HTML *模板*， 用*组件*类管理这些模板，用*服务*添加应用逻辑， 用*模块*打包发布组件与服务。然后通过*引导根模块*来启动该应用。 Angular 在浏览器中接管、展现应用的内容，并根据我们提供的操作指令响应用户的交互。
* 全景图◆◆◆◆



* 模块: Angular 应用是模块化的，并且 Angular 有自己的模块系统，称为NgModules。每个 Angular 应用至少有一个模块（根模块），习惯上命名为AppModule。在一些小型应用中根模块可能是唯一的模块，大多数应用会有很多特性模块，每个模块都是一个内聚的代码块专注于某个应用领域、工作流或紧密相关的功能。Angular 模块（无论是根模块还是特性模块）都是一个带有@NgModule装饰器的类。
* 装饰器是用来修饰 JavaScript 类的函数。 Angular 有很多装饰器，它们负责把元数据附加到类上，以了解那些类的设计意图以及它们应如何工作。
* [NgModule](https://angular.cn/api/core/NgModule)是一个装饰器函数，它接收一个用来描述模块属性的元数据对象。重要属性:
* declarations - 声明本模块中拥有的视图类。Angular 有三种视图类：[组件](https://angular.cn/guide/architecture#components)、[指令](https://angular.cn/guide/architecture#directives)和[管道](https://angular.cn/guide/pipes)。
* exports - declarations 的子集，可用于其它模块的组件[模板](https://angular.cn/guide/architecture#templates)。
* imports - 本模块声明的组件模板需要的类所在的其它模块。
* providers - [服务](https://angular.cn/guide/architecture#services)的创建者，并加入到全局服务列表中，可用于应用任何部分。
* bootstrap - 指定应用的主视图（称为根组件），它是所有其它视图的宿主。只有根模块才能设置bootstrap属性。
* 简单的跟模块



* 我们通过引导根模块来启动应用。 在开发期间，通常在一个main.ts文件中引导AppModule，比如



* NgModules vs. JavaScript 模块
* NgModule（一个带@[NgModule](https://angular.cn/api/core/NgModule)装饰器的类）是 Angular 的基础特性之一。
* JavaScript 中，每个文件是一个模块，文件中定义的所有对象都从属于那个模块。 通过export关键字，模块可以把它的某些对象声明为公共的。 其它 JavaScript 模块可以使用import 语句来访问这些公共对象。



* Angular 模块库
* Angular提供了JavaScript模块(库模块)。每个 Angular 库的名字都带有@angular前缀。用 npm 包管理工具安装，用 JavaScript 的import语句导入部件。如图



* 从@angular/core库中导入[Component](https://angular.cn/api/core/Component)装饰器：



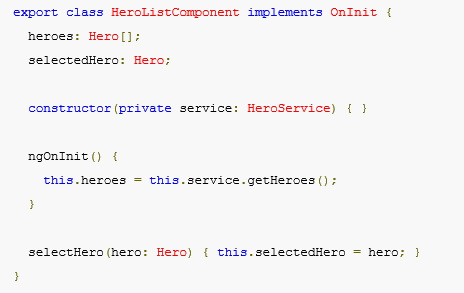
* 还可以使用 JavaScript 的导入语句从 Angular 库中导入 Angular 模块：



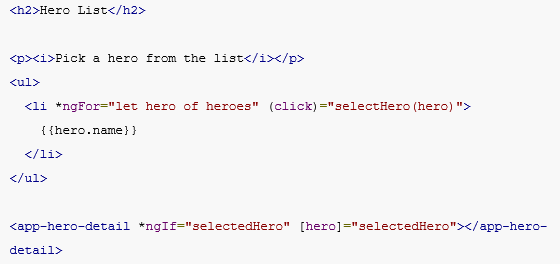
* 若应用模块需要BrowserModule的某些素材。要访问这些素材，就得把它加入@NgModule元数据的imports中：



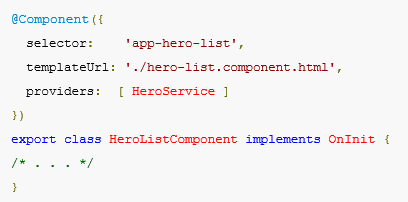
* 组件
* 组件负责控制屏幕上的一小块区域，我们称之为视图。
* 例如，HeroListComponent有一个heroes属性，它返回一个英雄数组，这个数组从一个服务获得。 HeroListComponent还有一个当用户从列表中点选一个英雄时设置selectedHero属性的selectHero()方法。当用户在这个应用中漫游时， Angular 会创建、更新和销毁组件。 应用可以通过生命周期钩子在组件生命周期的各个时间点上插入自己的操作，例如声明的ngOnInit()。



* 模板
* 通过组件的自带的模板来定义组件视图。模板以 HTML 形式存在，告诉 Angular 如何渲染组件。例如HeroListComponent组件的一个模板：



* 模板除了可以使用像<h2>和<p>这样的典型的 HTML 元素，还能使用其它元素。 例如，像\*ngFor、{{hero.name}}、(click)、[hero]和<hero-detail>这样的代码使用了 Angular 的模板语法。在模板的最后一行，<hero-detail>标签就是一个用来表示新组件HeroDetailComponent的自定义元素。HeroDetailComponent跟以前见到过的HeroListComponent是不同的组件。
* 元数据
* 元数据告诉 Angular 如何处理一个类
* 在TypeScript中，我们用**装饰器(decorator)** 来附加元数据。 下面就是HeroListComponent的一些元数据。



* 这里看到@Component装饰器，它把紧随其后的类标记成了组件类
* @Component装饰器能接受一个配置对象， Angular 会基于这些信息创建和展示组件及其视图。
* @Component的配置项包括：
* selector： CSS 选择器，它告诉 Angular 在父级 HTML 中查找<hero-list>标签，创建并插入该组件。 例如，如果应用的 HTML 包含<hero-list></hero-list>， Angular 就会把HeroListComponent的一个实例插入到这个标签中。
* templateUrl：组件 HTML 模板的模块相对地址，如前所示。
* providers - 组件所需服务的依赖注入提供商数组。 这是在告诉 Angular：该组件的构造函数需要一个HeroService服务，这样组件就可以从服务中获得英雄数据。
* @Component里面的元数据会告诉 Angular 从哪里获取你为组件指定的主要的构建块。
* 模板、元数据和组件共同描绘出这个视图。
* 其它元数据装饰器用类似的方式来指导 Angular 的行为。 例如@Injectable、@Input和@Output等是一些最常用的装饰器。
* 这种架构处理方式是：你向代码中添加元数据，以便 Angular 知道该怎么做。