**目录**

1. 搭建本地开发环境
2. 项目结构简述
3. 创建component
4. 数据绑定
5. 内置指令和组件生命周期钩子函数
6. 创建service
7. Http请求
8. 路由
9. 引用第三方插件

# 搭建本地开发环境

**第 1 步：安装 Angular CLI**

你可以使用 Angular CLI 来创建项目、生成应用和库代码，以及执行各种持续开发任务，比如测试、打包和部署。安装 Angular CLI。

要使用 npm 命令安装 CLI，请打开终端/控制台窗口，输入如下命令



**第 2 步：创建工作空间和初始应用**

你要在 Angular [工作区](https://angular.cn/guide/glossary#workspace)的上下文中开发应用。

要创建一个新的工作空间和初始入门应用：

1. 运行 CLI 命令 ng new 并提供 my-app 名称作为参数，如下所示：



1. ng new 命令会提示你提供要把哪些特性包含在初始应用中。按 Enter 或 Return 键可以接受默认值。

Angular CLI 会安装必要的 Angular npm 包和其他依赖包。这可能要花几分钟的时间。

CLI 会创建一个新的工作区和一个简单的欢迎应用，随时可以运行它。

**第 3 步：运行应用**

Angular CLI 中包含一个服务器，方便你在本地构建和提供应用。

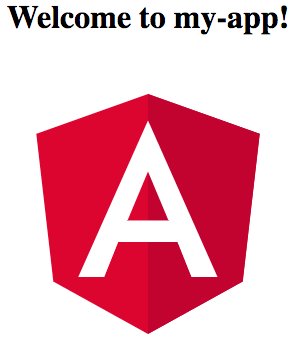
1. 转到 workspace 文件夹（my-app）。
2. 使用 CLI 命令 ng serve 和 --open 选项来启动服务器。



ng serve 命令会启动开发服务器、监视文件，并在这些文件发生更改时重建应用。

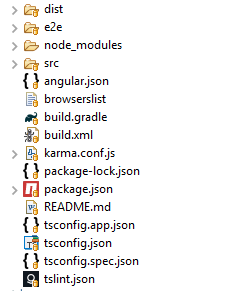
--open（或者只用 -o 缩写）选项会自动打开你的浏览器，并访问 [http](https://angular.cn/api/common/http)://localhost:4200/。

你的应用会展示出现：



# 项目结构简述

**根目录**



**dist** 打包后的压缩文件

**e2e** 测试使用 其他的一般不会用到

**node\_module**第三方依赖包存放目录

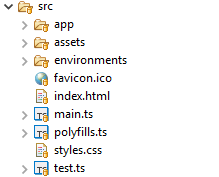
**angular.json** ng 8 的配置文件

**src** 应用源代码目录

**package.json**标准的npm工具的配置文件，这个文件里面列出了该应用程序所使用的第三方依赖包

**tslint.json** TS的配置文件，定义TypeScript代码质量检查的规则

**src目录**



**app**包含应用的组件和模块，我们要写的代码都在这个目录

**assets**资源目录，存储静态资源的 比如图片

**environments**环境配置。Angular是支持多环境开发的，我们可以在不同的环境下（开发环境，测试环境，生产环境）共用一套代码，主要用来配置环境的

**index.html** 整个应用的根html，程序启动就是访问这个页面

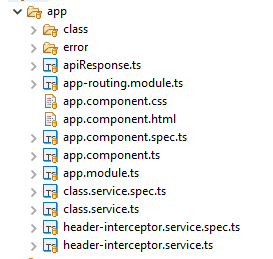
**main.ts** 整个项目的入口点，Angular通过这个文件来启动项目

**polyfills.ts** 主要是用来导入一些必要库，为了让Angular能正常运行在老版本浏览器下

**styles.css**主要是放一些全局的样式

**test.ts** 自动化测试用的

**app**目录



**app-routing.module.ts** 路由模块，定义页面跳转

**app.module.ts** 根模块，任何新建的组件，子模块，路由，第三方依赖模块想要在当前 module 中被使用，都必须在当前 module 中引入

**app.component.ts** 根组件，一个Angular程序至少需要一个模块和一个组件

**app.component.css**根组件样式

**app.component.html**根组件模板

**app.component.spec.ts** 测试文件

**class.service.ts** 为组件提供数据，主要用于发送http请求

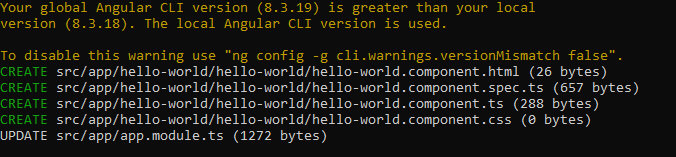
**header-interceptor.service.ts** 前端拦截器，请求转换，主要用于更改header，加token，响应错误拦截处理，加密解密等

# 创建component

创建一个组件的基本命令 ng g c 组件名

默认新建的组件、服务都会放在 src/app 目录下

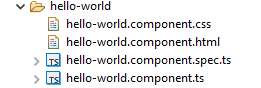
根目录下执行ng g c hellow-world：



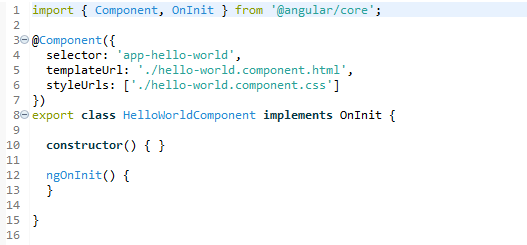
4 个创建分别是 **样式文件**、 **html文件**、**spec.ts测试文件**、**.ts文件**。

1 个更新是在当前 **module** 自动引入了我们新建的组件，可以打开 **app.module.ts** 查看。

创建的组件目录如下：



组件源码：



这里是从Angular核心模块里面引入了**@component**装饰器，将HelloWorld类定义为一个angular组件。

**@component**装饰器定义了一个组件以及组件的元数据，所有的组件都必须使用这个装饰器来注解组件元数据， Angular会通过这里面的属性来渲染组件并执行逻辑

**selector :** 就是css选择器，表示这个组件可以通过app-root的HTML页面标签来来调用，index.html中有个< app-hello-world ></ app-hello-world >标签，这个标签用来展示该组件的内容

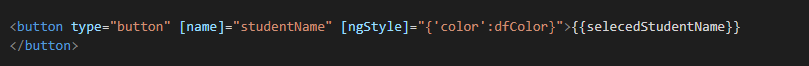
**templateUrl** : 指定了一个html文件作为组件的模板，定义了组件的布局和内容。在这里定义hello-world.component.html，最终在index.html中< app-hello-world >/< app-hello-world >这个标签的内容将展示hello-world.component.html里面的内容。也就是templateUrl所定义的页面定义了用户最终看见的页面的布局和内容。

**styleUrls** : 指定了一组css文件。可以在这个css中编写这个组件模板要用到的样式

# 数据绑定

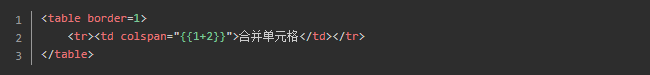
Angular里，有两种绑定，一种是数据绑定（Data Binding），另一种是事件绑定（Event Binding）

数据绑定根据数据流向又分**单向绑定**和**双向绑定**



**Dom元素属性绑定**：**[]**用于绑定dom元素一般属性比如id，name，class等

**插值绑定：**使用双大括号的语法**{{}}**，双大括号中可以是变量名、合法的模板表达式、或者是宿主组件的函数



**html标签特性绑定：**angular推荐使用dom元素属性绑定，但是当元素没有对应的属性的时候，可以使用html标签特性绑定来设置值。



**事件绑定：**数据从模板流向组件，事件绑定的语法是由“=”左侧小括号中的目标事件和“=”右侧引号中的模板语句组成。

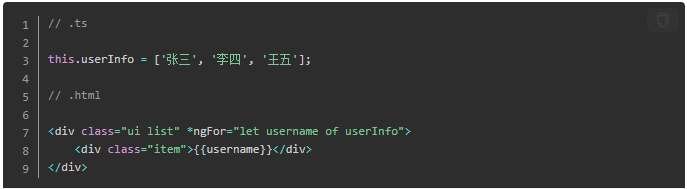


**数据双向绑定：**在angular中可以使用[(ngModel)]语法实现双向数据绑定。  
其中[]实现了数据流从组建到模板，()实现了数据流从模板到组件。

# 内置指令和组件生命周期钩子函数

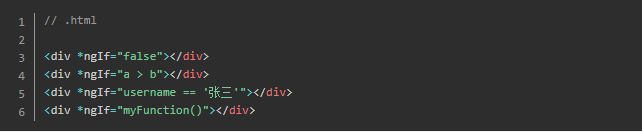
## **内置指令**

## **\*ngFor**



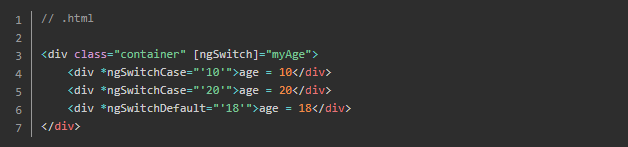
语法是 \*ngFor="let username of userInfo"，其中 userInfo 是从中取值的数组，username 是每次从中取出来的值。然后在这个标签里面的内容就会重复执行，并通过双向绑定，将 username 显示出来。

## **\*ngIf**



1. 永远不会显示
2. 当 a 大于 b 的时候显示
3. 当 username 等于 张三 的时候显示
4. 根据 myFunction() 这个函数的返回值，决定是否显示

## **ngSwitch**



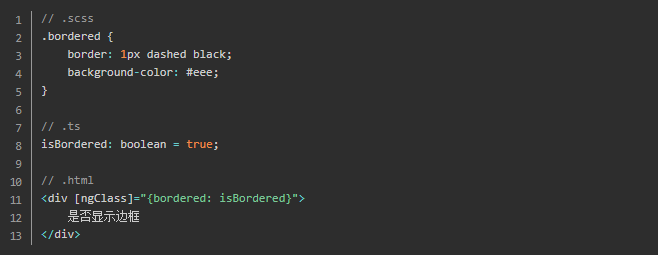
[ngSwitch] 先与目标进行绑定，\*ngSwitchCase 列出每个可能性，\*ngSwitchDefault 列出默认值。

## **ngStyle**



[ngStyle]可以使用动态值给特定的 DOM 元素设定 CSS 属性。

## **ngClass**



[ngClass]动态地设置和改变一个给定 DOM 元素的 CSS类。

## **ngNonBindable**



告诉 Angular 不要绑定页面的某个部分，代码中花括号内容不会被绑定，以字符串显示。

## **生命周期钩子函数**

Angular每个组件都存在一个生命周期，从创建，变更到销毁。Angular提供组件生命周期钩子，把这些关键时刻暴露出来，赋予在这些关键结点和组件进行交互的能力。（例如在ngOnInit方法里某个动态绑定的dom元素还无没渲染完成无法获取，则需要在ngAfterViewInit中才能获取）

## **ngOnChanges**

在有输入属性的情况下才会调用，该方法接受当前和上一属性值的SimpleChanges对象。如果有输入属性，会在ngOnInit之前调用。

## **ngOnInit**

在组件初始化的时候调用，只调用一次，在第一次调用ngOnChanges之后调用

## **ngDoCheck**

在组件定义的属性或方法变更时调用(用于脏值之检测，非常耗性能，因为会把所有的属性和方法都检测一遍），会在ngOnChanges()和ngOnInit()之后

## **ngAfterContentInit**

在组件内容初始化之后调用，在第一次ngDoCheck之后调用，只调用一次

## **ngAfterContentChecked**

在组件每次检查内容放生变更时调用。在ngAfterContentInit和每次ngDoCheck之后调用

## **ngAfterViewInit**

在组件相应的视图初始化之后调用，第一次ngAfterContentChecked之后调用，只调用一次

## **ngAfterViewChecked**

在组件每次检查视图发生变更时调用。ngAfterViewInit和每次ngAfterContentChecked之后调用。

## **ngOnDestroy**

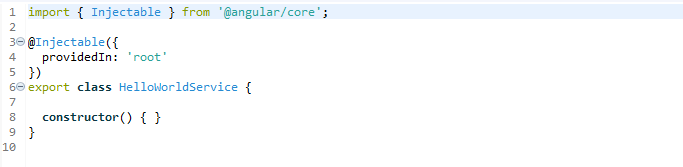
在组件销毁前调用，做一些清理工作，比如退订可观察对象和移除事件处理器，以免导致内存泄漏。

# 创建service

服务是 Angular 应用的重要组成部分。在 Angular 中，服务是一个类的实例，它可以借助 Angular 的[依赖注入系统](https://angular.cn/guide/glossary#dependency-injection-di)来让应用中的任何一个部件都能使用它。服务可以让你在应用的各个部件之间共享数据。

创建一个服务的基本命令 ng g s 服务名

service通常带有可注入装饰器（@Injectable()）并以注入的形式被使用。  
component 和 service 都是 class，只是通过不同的装饰器获得了不同的表现。



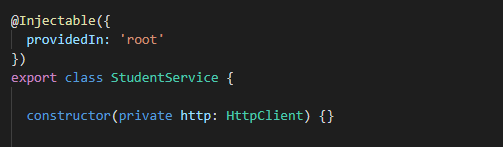
**providedIn: ‘root’**，表示服务加入到跟模块的providers中，这样该服务就可以在应用中的任何地方使用，否则需要手动引入到模块里

# Http请求

angular 中使用 http 请求的前提，需要在根模块引入 **httpClientModule** 模块

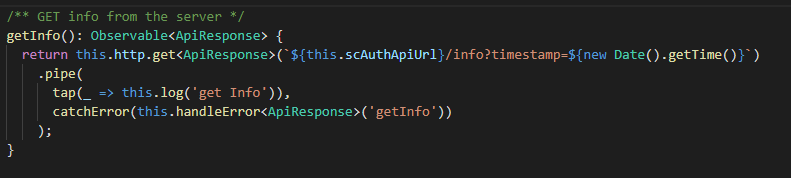


在服务中引用并在构造函数中实例化**HttpClient**

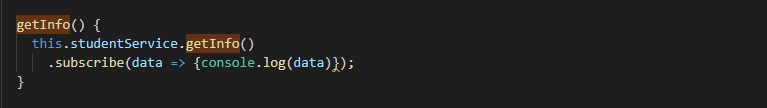


通过HttpClient中封装的方法发送http请求，组件通过调用服务中的方法获取数据

*StudentService:*



*StudentComponent:*



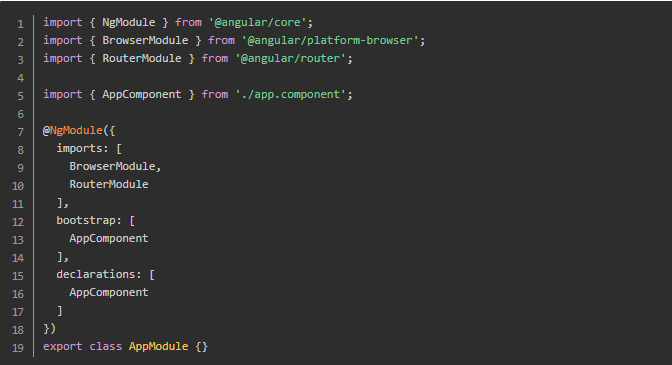
# 路由

Angular的页面跳转由路以插座为基础实现页面切换， 路由插座可以简单把它理解为： 页面的占位符，动态加载，会被替换掉的。  
当点击 路由中匹配的跳转页面时*<router-outlet></router-outlet>*标签会被url对应的 XX.component.html 替换掉。 这就是angular路由的原理

*app.componment.html:*



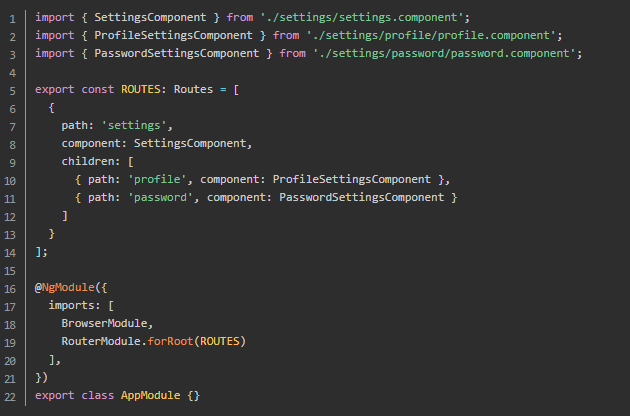
要使用路由，我们需要在 根模块中，导入 **RouterModule**



**RouterModule** 对象为我们提供了两个静态的方法：forRoot() 和 forChild() 来配置路由信息。

**RouterModule.***forRoot()* 方法用于在主模块中定义主要的路由信息，通过调用该方法使得我们的主模块可以访问路由模块中定义的所有指令。

通过使用 **const** 定义路由的配置信息，然后把它作为参数调用 **RouterModule**.*forRoot()* 方法，而不是直接使用 **RouterModule**.*forRoot([...])* 这种方式，这样做的好处是方便我们在需要的时候导出 ROUTES 到其它模块中。



**RouterModule**.*forChild()* 与 **RouterModule**.*forRoot()* 方法类似，但它只能应用在特性模块中。简单讲根模块中使用 forRoot()，子模块中使用 forChild()

路由的定义

典型的Angular路由（Route）有两个属性：

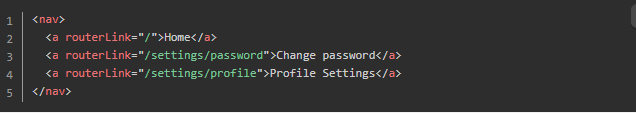
**path**：一个用于匹配浏览器地址栏中URL的字符串

**component**：当导航到此路由时，路由应该创建哪个组件

**children**：子路由

**outlet**：当同级路由插座有多个时需要指明对应插座的name

配置完路由后可以使用路由模块提供的routerLink指令实现跳转



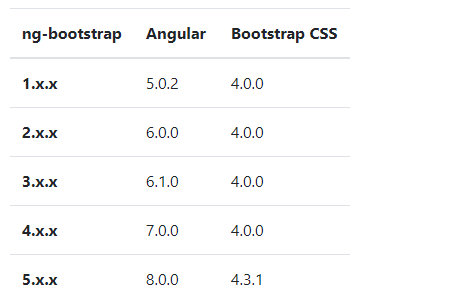
我们可以通过路由还提供的 API 实现与 routerLink 相同的功能。要使用 Router API，我们需要在组件类中注入 Router对象，

组件类中注入的 router 对象中有一个 navigate() 方法，该方法支持的参数类型与 routerLink 指令一样，当调用该方法后，页面将会自动跳转到对应的路由地址：



# 引用第三方插件

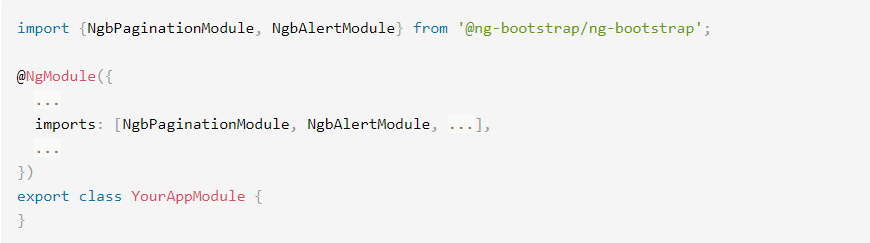
一般第三方插件官网都有导入angular项目的教程和注意事项，以ngBootstrap为例：



官网列出了插件版本对应的angular版本和bootstrap版本，根据列表在项目下安装对应版本

在根模块中引入插件模块

或者是只引入你需要的部分，例如项目只需要分页模块或弹出窗模块



然后就能在组件中使用这些插件

具体的插件使用规则详见<https://ng-bootstrap.github.io/#/home>

想要了解更多的angular技术请访问官网<https://angular.io/>