Angular4.x 安装、创建项目、 目录结构介绍、创建组件

合作网站：www.itying.com （IT 营）
合作网站：www.ionic.wang （Ionic 中文网 ）

目录

一、安装最新版本的 nodejs

注意：请先在终端/控制台窗口中运行命令 `node -v` 和 `npm -v`，**来验证一下你正在运行 node 8.x 和 npm 5.x 以上的版本。** 更老的版本可能会出现错误，更新的版本则没问 题。

二、全局安装 Angular CLI 脚手架工具(只需要安装一次)。

1. **使用 npm 命令安装**

```
npm install -g @angular/cli
```

2. **安装 cnpm**

npm 可能安装失败建议先用 npm 安装一下 cnpm 用淘宝镜像安装
https://npm.taobao.org/

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

3. **使用 cnpm 命令安装**

```
cnpm install -g @angular/cli
```

三、创建项目

1. **打开 cmd 找到你要创建项目的目录**
2. **创建项目**

ng new 项目名称 **创建一个项目**

```
ng new my-app
```

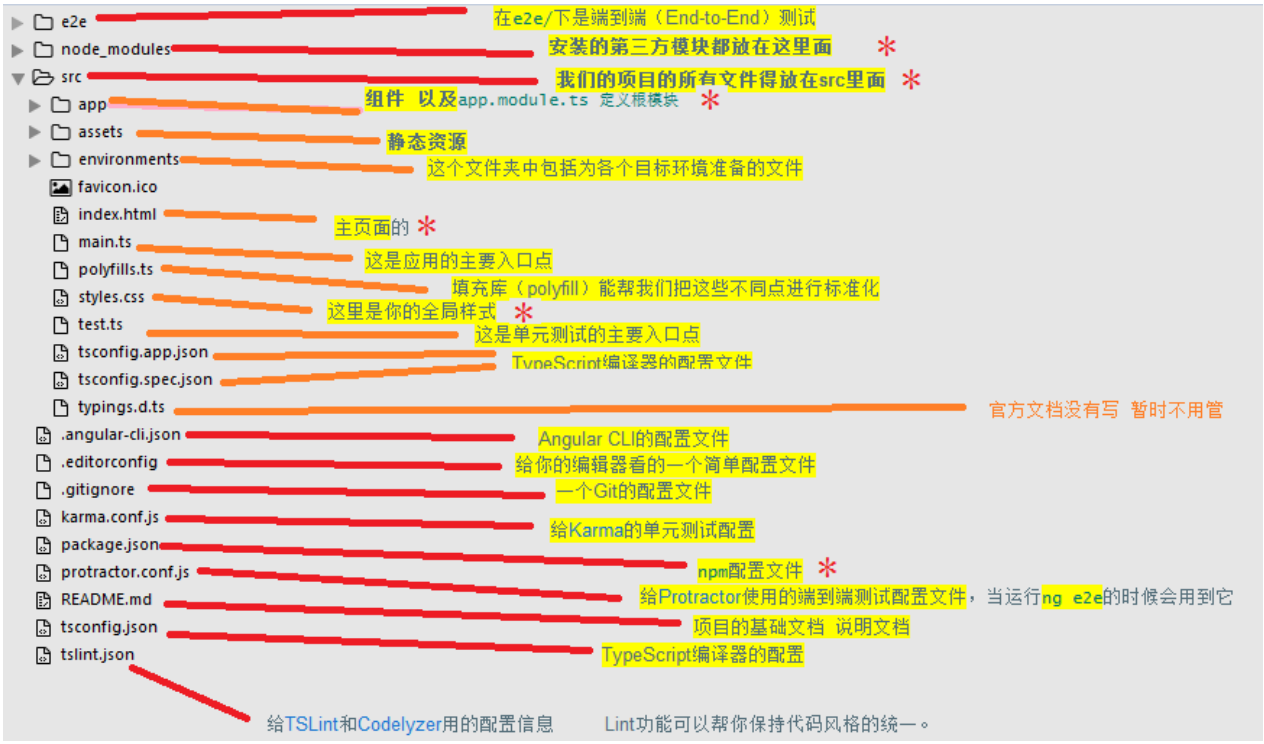
3. **进入刚才创建的项目里面启动服务**

```
cd my-app

cnpm install   //安装依赖

ng serve --open
```

四、目录结构分析



1.项目创建完成后的目录结构：

文件	用途
e2e/	在 e2e/下是端到端（End-to-End）测试。 它们不在 src/下，是因为端到端测试实际上和应用是相互独立的，它只适用于测试你的应用而已。 这也就是为什么它会拥有自己的 tsconfig.json。
Src	我们的项目的所有文件得放在 src 里面，所有的 Angular 组件、模板、样式、图片以及你的应用所需的任何东西都在那里。
node_modules/	Node.js 创建了这个文件夹，并且把 package.json 中列举的所有第三方模块都放在其中。
.angular-cli.json	Angular CLI 的配置文件。 在这个文件中，我们可以设置一系列默认值，还可以配置项目编译时要包含的那些文件。 要了解更多，请参阅它的官方文档。

.editorconfig	给你的编辑器看的一个简单配置文件，它用来确保参与你项目的每个人都具有基本的编辑器配置。 大多数的编辑器都支持.editorconfig 文件，详情参见 http://editorconfig.org 。
.gitignore	一个 Git 的配置文件，用来确保某些自动生成的文件不会被提交到源码控制系统中。
karma.conf.js	给 Karma 的单元测试配置，当运行 ng test 时会用到它。
package.json	npm 配置文件，其中列出了项目使用到的第三方依赖包。 你还可以在这里添加自己的自定义脚本。
protractor.conf.js	给 Protractor 使用的端到端测试配置文件，当运行 ng e2e 的时候会用到它。
README.md	项目的基础文档，预先写入了 CLI 命令的信息。 别忘了用项目文档改进它，以便每个查看此仓库的人都能据此构建出你的应用。
tsconfig.json	TypeScript 编译器的配置，你的 IDE 会借助它来给你提供更好的帮助。
tslint.json	给 TSLint 和 Codelyzer 用的配置信息，当运行 ng lint 时会用到。Lint 功能可以帮你保持代码风格的统一。

2.src 目录结构：

文件	用途
app/app.component.{ts,html,css,spec.ts}	组件 使用 HTML 模板、CSS 样式和单元测试定义 AppComponent 组件。 它是**根**组件，随着应用的成长它会成为一棵组件树的根节点。
app/app.module.ts	定义 AppModule，这个根模块会告诉

	Angular 如何组装该应用。 目前，它只声明了 AppComponent。 稍后它还会声明更多组件。
assets/*	**静态资源** 这个文件夹下你可以放图片等任何东西，在构建应用时，它们全都会拷贝到发布包中。
environments/*	这个文件夹中包括为各个目标环境准备的文件，它们导出了一些应用中要用到的配置变量。 这些文件会在构建应用时被替换。比如你可能在产品环境中使用不同的 API 端点地址，或使用不同的统计 Token 参数。甚至使用一些模拟服务。 所有这些，CLI 都替你考虑到了。
favicon.ico	每个网站都希望自己在书签栏中能好看一点。 请把它换成你自己的图标。
index.html	这是别人访问你的网站是看到的主页面的 HTML 文件。 大多数情况下你都不用编辑它。 在构建应用时，CLI 会自动把所有 js 和 css 文件添加进去，所以你不必在这里手动添加任何 <script> 或 <link> 标签。
main.ts	这是应用的主要入口点。使用 JIT compiler 编译器编译本应用，并启动应用的根模块 AppModule，使其运行在浏览器中。 你还可以使用 AOT compiler 编译器，而不用修改任何代码 —— 只要给 ng build 或 ng serve 传入 --aot 参数就可以了。
polyfills.ts	不同的浏览器对 Web 标准的支持程度也不同。 填充库（polyfill）能帮我们把这些不同点进行标准化。 你只要使用 core-js 和 zone.js 通常就够了，不过你

	也可以查看浏览器支持指南以了解更多信息。
styles.css	这里是你的全局样式。 大多数情况下，你会希望在组件中使用局部样式，以利于维护，不过那些会影响你整个应用的样式你还是需要集中存放在这里。
test.ts	这是单元测试的主要入口点。 它有一些你不熟悉的自定义配置，不过你并不需要编辑这里的任何东西。
tsconfig.{app|spec}.json	TypeScript 编译器的配置文件。 tsconfig.app.json 是为 Angular 应用准备的，而 tsconfig.spec.json 是为单元测试准备的。

五、Hello Angular 4.0 以及 app.module.ts、组件分析

1. app.module.ts

定义 AppModule，这个根模块会告诉 Angular 如何组装该应用。 目前，它只声明了 AppComponent。 稍后它还会声明更多组件。

```
//Angular 模块类描述应用的部件是如何组合在一起的。 每个应用都至少有一个 Angular 模块，也就是根模块，
// 用来引导并运行应用。 你可以为它取任何名字。常规名字是 AppModule。 也就是 app.module.ts 文件

/*引入组件*/

import { BrowserModule } from '@angular/platform-browser';  /*BrowserModule，浏览器解析的模块*/
import { NgModule } from '@angular/core';  /*angualrjs 核心模块*/
import { FormsModule } from '@angular/forms';  /*表单数据绑定 表单验证需要的模块*/
import { HttpModule } from '@angular/http';    /*数据请求模块*/
```

```
import { AppComponent } from './app.component'; /*根组件*/

/*@NgModule 装饰器将 AppModule 标记为 Angular 模块类（也叫 NgModule 类）。
 @NgModule 接受一个元数据对象，告诉 Angular 如何编译和启动应用。*/

@NgModule({
  declarations: [  /*引入当前项目运行的的组件*/
    AppComponent
  ],
  imports: [ /*引入当前模块运行依赖的其他模块*/
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  providers: [],  /*定义的服务  回头放在这个里面*/
  bootstrap: [AppComponent]  /* 指定应用的主视图（称为根组件） 通过引导根 AppModule 来启动
应用  ，这里一般写的是根组件*/
})

/*根模块不需要导出任何东西，    因为其它组件不需要导入根模块。 但是一定要写*/

export class AppModule { }
```

2.自定义组件

https://github.com/angular/angular-cli

创建组件：

```
ng g component components/header
```

组件内容详解：

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

@Component({
  selector: 'app-header',  /*使用这个组件的名称*/
  templateUrl: './header.component.html', /*html 模板*/
  styleUrls: ['./header.component.css'] /*css 样式*/
})
export class HeaderComponent implements OnInit {  /*实现接口*/

  constructor() {  /*构造函数*/

   }

  ngOnInit() {  /*初始化加载的生命周期函数*/

  }

}
```