Angular 数据操作为主的SPA 组件化开发方式

工作方式

通过组件来构造模板内容

通过服务来封装常用的功能

通过输入器来注入需要用到的模块或服务

搭建环境

1. npm install –g @angular/cli

ng new my-app

cd my-app

ng serve –open

1. github.com

下载模板项目quickstart

npm install

npm start

每新建一个.component.ts文件，都会自动生成一个.component.map和一个.component.js

angular为模块开发，模块包含组件

angular中css样式要在html中自己引入

src 🡨-------该文件夹下创建组件

app

app.compoennet.ts

app.module.ts

main.ts

index.html

styles.css

组件的本质是可被反复使用的，带有特定功能的视图

import {Component} from ‘@angular/core’ 引入识别为组件类

@Component({//装饰器 装饰为模块

selector:’组件名’， 在模块中使用时的标签名

template:`模板`，

templateUrl: ‘html模板地址’,引入模板

styleUrls:[‘样式路径’，‘’]，引入样式

})

export class Angular1Component{

implements OnInit{

angular组件的生命周期的处理函数，在初始化时自动调用

constructor(){}

ngOnInit(){}

变量：类型=0/null;ts中声明变量的方式

}

} 定义导出的为类，所以引入要加{}

去相应的模块下引入

import {NgModule} from ‘@angular/core’ 引入识别为组件类

import {BrowserModule} from “@angular/platform-browser”引入ng的Browser功能

import {Angular1Component} from ‘引入相关组件相对路径

@NgModule({ 为装饰器，装饰为模块

imports：[BrowserModule], 引入模块，组件需要时引入相应的模块

declarations: [APPComponent,Angular1Component]，

bootstrap: [APPComponent] 响应核心组件

})

装饰器，原数据

@Component({ 装饰器

selector: 元数据：告诉装饰器类的配置

.

.

.

})

指令

结构型指令：

循环指令：ngFor <any \*ngFor=”let value of list;let myIndex=index”>{{value}}</any>

选择指令：<any \*ngIf=”布尔型”></any>

<any [ngSwitch]=”变量”></any>

<any \*ngSwitchCase=””></any>

<any \*ngSwitchDefault></any>

属性型指令：<any [attr]=”值”>

<any [ngstyle]=”{属性：变量}”>

<any [ngClass]=”{属性：变量}”>

事件指令： <any (eventHandler)=”函数（）”>

双向数据绑定：<input [(ngModel)]=”变量” >

数据监听：<input (ngModelChange)=”函数（）”> //存在先后顺序问题

自定义指令：

创建一个指令类：指定文件名：xxx.driective.ts

核心库中还需要import {ElementRef}

@Driective({selector:’[test]’})

export class TestDirective{

constructor(elRf:ElementRef){获取当前调用该指定的函数

this.elRf.nativeElement

}

}

完成后去app.model.ts中引入并 去ngModel中delcarations声明

ng中所有的html中变量都要声明

在类中的data为赋值原理

一个标签内不能用多个指令， 解决：

再此dom前加上一个<ng-container 其余指令 >包裹

ng中的过滤器（管道）

内置管道：{{变量 | uppercase}}大小写转换

{{变量 | date:“格式”}}日期格式化

{{变量 | number：‘x.x-x’|}}数字格式化

{{变量 | slice:x:x}}字符串切片

多重过滤

{{变量 | 管道函数 | 管道函数}}

自定义管道类： a-pipe

与自定义指令类似，也需要去app.module.ts中引入

import{Pipe,PipeTransform} from ‘@angular/core’

@Pipe({name:管道名称})

export class 导出的名称 implements

PipeTransform(value:any调用管道所传值,…args:any):any返回值{}

angular开发整体框架涉及8部分

模块 angular开发基本单位

组件 带有模板指令可被反复使用的带有特定功能的视图

模板 增强后的html标签 ngfor {{}}

数据绑定 [(ngModel)]

元数据 metaData告诉angular如何处理类

指令 是为了扩展html的功能

依赖注入 将当前所需要用到的东西拿到当前这个类中去使用的过程，为了降低耦合度

服务

service服务是为了封装 应用中逻辑，在组件中复用

1. service

import {Injectable} from ‘@angular/core’

@Injectable ()

export class nameService{

constructor(){

方法

}

}

在app.module.ts下

import {nameService} from .

@NgModule({

providers:[nameService]

});

在要使用的组件下

import {nameService} from…

constructor(private/publice myLs: LogService){

}

需要在app.module.ts中引入 HttPModule from @angular/http

@NgModule({imports: [HttpModule]})

1. service-http 创建http模板

constructor(){}

a-http-get 请求模板,参数为url地址、withCredentials:true允许携带凭证（且PHP文件的Access-Control-Allow-Origin:所提示的url”;

Access-Control-Allow-Origin:true;

）

另外模板中调用RxJS

先载入服务模板

定义事件函数

this.服务名.函数（参数）.subscrible((res:any) => {});