#### Angular脱坑记

##### 之简化项目结构（目录树）

问题描述

Angular 6.1.5

TypeScript 2.7.2

Angular2采用MVC（模型、视图、控制器分离）设计模式。因此，理论上应为每个页面建立model、view、controller三个文件夹。当页面中需要嵌入子组件时，需要为每个子组件建立MVC目录结构。当子组件中又包含子组件时，会使项目结构显得复杂、凌乱。以下总结一些简化项目目录树结构的心得：

解决

1. 将Model合并进View

虽然具有强类型、面向对象的特点，执行TypeScript脚本归根结底还是需要依靠JavaScript。不论具有多复杂、整洁的目录结构，JavaScript脚本在编译时会被合并为单个文件。因此，【将Model合并进View】在简化项目结构的同时，并不会对Angular的设计模式造成副作用。示例如下：

chart.component.ts

@Component({  
 selector: 'chart',  
 templateUrl: 'chart.html',  
 styleUrls: ['chart.css']  
})  
export class ChartComponent implements OnInit {

mChartModel: ChartModel;

...

}

Export class ChartModel{

var: Type;

}

1. 采用ng-template实现模态框（Modal）

<ng-template>是Angular2项目的一个常用标签。Angular语法糖\*ngIf就会被模板引擎转换为这个标签：

<div \*ngIf="hero" class="name">{{hero.name}}</div>

<!-- \*ngIf翻译成ng-template元素之后 -->

<ng-template [ngIf]="hero">

<div class="name">{{hero.name}}</div>

</ng-template>

product-list-page.component.html

<ng-template #thresholdView let-c="close" let-d="dismiss">  
 <div class="modal-header">  
 <h4 class="modal-title"></h4>  
 <button type="button" class="close" (click)="d('Cross click')">  
 <span class="plx-ico-close-16"></span>  
 </button>  
 </div>  
 <div class="modal-body">  
 <div class="div-wrapper">  
   
 </div>  
 </div>  
 <div class="modal-footer" style="margin-top: -16px" *\*ngIf*="!pIsEditMode">  
 <div class="form-group w-100">  
 <div class="btnGroup modal-btn mx-auto float-none">  
 <button type="button" class="plx-btn" (click)="cancel()">关闭</button>  
 <button type="button" class="plx-btn plx-btn-primary" (click)="confirm()">修改</button>  
 </div>  
 </div>  
 </div>  
</ng-template>

product-list-page.component.ts

export class ThresholdPlpComponent implements OnInit {

modal: any;

@ViewChild('thresholdView') thresholdView: any;

constructor(private modalService: PlxModal,  
 ...  
 ) {  
 this.infoRepo = new BasicInfoPlpRepository();  
 this.onChange = new EventEmitter<ResourceTypeRepository>();  
 this.data = [];  
 }

openModal() {

const size: 'sm' | 'lg' = 'lg';  
const options = {  
 size: size,  
};  
this.modal = this.modalService.open(this.thresholdView, options);

}

}

1. 使用TypeScript高级语言特性

Intersection Types [#](http://www.typescriptlang.org/docs/handbook/advanced-types.html" \l "intersection-types" \o "Link to the header: Intersection Types)

An intersection type combines multiple types into one. This allows you to add together existing types to get a single type that has all the features you need. For example, Person & Serializable & Loggable is a Person and Serializable and Loggable. That means an object of this type will have all members of all three types.

product-list-page.component.ts

srcObj = {} as BasicInfo&DateAndTime&FormulaSet&SelectedPo&SelectedResInstance;

□