ng-template、ng-content、ng-container

<https://www.jianshu.com/p/0f5332f2bbf8>

ng-template、ng-content、ng-container三者应该是自定义组件需要经常用到的指令。今天咱们就来简单了解下ng-template、ng-content、ng-container的基本用法。

一、ng-content

       ng-content是内容映射指令(也叫内容嵌入)，内容映射指的是在组件中嵌入模板代码，方便定制可复用的组件，很好地扩充组件的功能，方便代码的复用。

我们再往简单一点想ng-content相当于一个占位符(留了个位置)，类似于路由出口router-outlet一样。之后会把相应的内容放到这个位置上来。

**1.2 ng-content select 组件选择**

       ng-content提供select 属性，方便我们选择投影的内容(组件或者html里面的标签)，如果我们没有设置select属性则所有的内容都可以投影上来。select的选择规则很简单，就三种规则：

select="xx"选择，xx对应html里面标签或者组件的名字。比如select="div"表示ng-content位置只会放div标签。

<ng-content select="div"></ng-content>

select=".xx"选择，xx对应html标签或者组件的class名字。比如select=".select-class"表示ng-content位置只会放设有class="select-class"的html标签或者组件。

<ng-content select=".select-class"></ng-content>

select="[key=value]"选择，key-value的形式。选择设置了属性key="value“的html标签或者组件。比如select="[name=test]"表示ng-content位置只会放设置了属性name=”test“的html标签或者组件。

select="[key]" 也是类型，ng-content会选择设置有key的属性的html标签或者组件。

<ng-content select="[name=test]"></ng-content>

<div name="test">我是第一号位置 div[name="test"]</div>

强调一点select的值不能设置为动态的

**1.2 ngProjectAs**

       通过ng-content的select属性可以指定html标签或者组件投射ng-content位置上来。但是呢有个限制条件。不管是select标签或者组件的名字、或者class、或者是属性他们都是作用在直接子节点上。还是用一个简单但额实例来说明

// 我们先自定义一个组件app-content-section 里面会使用ng-content

@Component({

selector: 'app-content-section',

template: `

<div>

<h1>ng content</h1>

<div style="background-color: #039be5">

**<ng-content select="app-content-child"></ng-content>**

</div>

</div>

`,

styleUrls: ['./content-section.component.less']

})

export class ContentSectionComponent {

}

// 下面中情况下 ng-content没有投射到对应的内容

<app-content-section>

<ng-container>

<app-content-child [title]="'测试下'"></app-content-child>

</ng-container>

</app-content-section>

// 通过使用 ngProjectAs 让ng-content的内容能正确的投射过来。

<app-content-section>

<**ng-container ngProjectAs="app-content-child"**>

<app-content-child [title]="'测试下'"></app-content-child>

</ng-container>

</app-content-section>

**1.3 ng-conent 包含组件的获取**

       ng-conent提供了@ContentChild和@ContentChildren来获取ng-conent里面包含的组件(类似@ViewChild和@ViewChildren)。获取到ng-conent里面的组件之后你就可以为所欲为了。

       我觉得如果有需要获取ng-content包含组件的情况，前提条件是咱得对放在ng-content位置的是啥类型的组件心里有数。有一点要注意@ContentChild和@ContentChildren所在的ts文件一定是有ng-content对应的哪个ts文件。可千万别搞错了。

接下来我们用一个特别简单的例子

// 定义一个app-content-section组件

/\*\*

\* 想获取ng-content里面的组件的使用@ContentChild或者@ContentChildren

\*/

@Component({

selector: 'app-content-section',

template: `

<div>

<h1>ng content</h1>

<!--这里我们确定我们这里会放ContentChildComponent组件，才好使用@ContentChild和@ContentChildren-->

<ng-content></ng-content>

</div>

`,

styleUrls: ['./content-section.component.less']

})

export class ContentSectionComponent implements AfterContentInit {

// 通过 #section\_child\_0 获取组件

@ContentChild('section\_child\_0')

childOne: ContentChildComponent;

// 通过 ContentChildComponent 组件名获取组件

@ContentChildren(ContentChildComponent)

childrenList: QueryList<ContentChildComponent>;

ngAfterContentInit(): void {

console.log(this.childOne);

this.childrenList.forEach((item) => {

console.log(item);

});

}

}

// 使用app-content-section

@Component({

selector: 'app-ng-content',

template: `

<app-content-section>

<app-content-child #section\_child\_0 [title]="title\_0"></app-content-child>

<app-content-child #section\_child\_1 [title]="title\_1"></app-content-child>

</app-content-section>

`,

styleUrls: ['./ng-content.component.less']

})

export class NgContentComponent {

title\_0 = 'child\_0';

title\_1 = 'child\_1';

}

**二、ng-template**

       ng-template是Angular 结构型指令中的一种，用于定义模板渲染HTML(模板加载)。定义的模板不会直接显示出来，需要通过其他结构型指令（如 ng-if）或 template-ref 将模块内容渲染到页面中。

       上面既然说了ng-template的内容默认是不会在页面上显示出来的。这肯定不行呀，咱们使用ng-template不管中间的原因是啥，反正最后肯定是要把这些内容显示在界面上的。那么咱们有哪些办法来显示ng-template的内容呢。

借助其他结构型指令如×ngIf，来显示ng-template的内容。

<!-- 通过ngIf结构型指令显示ng-template的内容 -->

<div class="lessons-list" \*ngIf="condition else elseTemplate">

判断条件为真

</div>

<ng-template #elseTemplate>

<div>判断条件为假</div>

</ng-template>

通过TemplateRef、ViewContainerRef把ng-template对应的元素显示出来。TemplateRef对应ng-template的引用，ViewContainerRef呢则是view容器的引用用来操作DOM元素。

import {AfterViewInit, Component, TemplateRef, ViewChild, ViewContainerRef} from '@angular/core';

@Component({

selector: 'app-template-section',

template: `

<ng-template #tpl>

Hello, ng-template!

</ng-template>

`,

styleUrls: ['./template-section.component.less']

})

export class TemplateSectionComponent implements AfterViewInit {

@ViewChild('tpl')

tplRef: TemplateRef<any>;

constructor(private vcRef: ViewContainerRef) {

}

ngAfterViewInit() {

// 这样tplRef对应的ng-template内容就显示出来了

this.vcRef.createEmbeddedView(this.tplRef);

}

}

通过NgTemplateOutlet指令来显示已有的ng-template对应的视图。NgTemplateOutlet指令用于基于已有的 TemplateRef 对象，插入对应的内嵌视图。同时我们可以通过 [ngTemplateOutletContext] 属性来设置 ng-template 的上下文对象。绑定的上下文应该是一个对象，ng-template中通过 let 语法来声明绑定上下文对象属性名。

import {AfterViewInit, Component, TemplateRef, ViewChild, ViewContainerRef} from '@angular/core';

@Component({

selector: 'app-ng-template',

template: `

<!-- let-param 取的是绑定对象myContext的$implicit字段的指，let-param相和let-param="$implicit"是等价的， -->

<!-- let-name="name" 取的是绑定对象myContext里面name字段对应的值-->

<ng-template #inputTemplateWithContent let-param let-name="name">

<div>{{param}} - {{name}}</div>

</ng-template>

<ng-container \*ngTemplateOutlet="inputTemplateWithContent; context: myContext"></ng-container>

`,

styleUrls: ['./ng-template.component.less']

})

export class NgTemplateComponent {

myContext = {$implicit: '默认值', name: 'tuacy'};

}

这里想稍微提下通过NgTemplateOutlet绑定上下文对象的问题，在ng-template中我们通过let-xx="yy" xx是在ng-template内部使用的变量名字。xx对应的值是上下文对象yy属性的值。而且let-xx等同于let-xx="implicit: '默认值', name: 'tuacy'};

ng-template里面let-param param对应的是myContext对象里面$implicit属性的值(let-param和let-param=”implicit“是一样的意思)，let-name="name" name对应的是myContext对象里面name属性对应的值。

       我们来一个稍微复杂一点的例子，我们把ng-template的内容，做为一个组件的参数。

import {Component, Input, TemplateRef} from '@angular/core';

@Component({

selector: 'app-template-input',

template: `

<!-- 没有传递参数的时候就使用defaultTemplate里面的布局 -->

<ng-template #defaultTemplate>

<div>咱们没有传递参数</div>

</ng-template>

<ng-container \*ngTemplateOutlet="inputTemplate ? inputTemplate: defaultTemplate"></ng-container>

`,

styleUrls: ['./template-input.component.less']

})

export class TemplateInputComponent {

/\*\*

\* 模板作为参数

\*/

@Input()

inputTemplate: TemplateRef<any>;

}

// 使用的时候的代码

<!-- 我们定义一个组件，把ng-template的内容作为参数传递进去 -->

<ng-template #paramTemplate>

<div>我是参数</div>

</ng-template>

<app-template-input [inputTemplate]="paramTemplate"></app-template-input>

**2.1 ngTemplateOutlet**

       ngTemplateOutlet指令用于基于已有的 TemplateRef对象，插入对应的内嵌视图。同时在使用 ngTemplateOutlet 指令的时候，我们可以通过 [ngTemplateOutletContext]属性来设置 EmbeddedViewRef 的上下文对象。绑定的上下文应该是一个对象，而且可以通过 let 语法来声明绑定上下文对象属性名。

在渲染页面之前，Angular 会把及其内容替换为注释

1. **ng-container**

       ng-container既不是一个Component，也不是一个Directive，只是单纯的一个特殊tag。ng-container可以直接包裹任何元素，包括文本，但本身不会生成元素标签，也不会影响页面样式和布局。包裹的内容，如果不通过其他指令控制，会直接渲染到页面中。

咱们可以把ng-container简单理解为一个逻辑容器。用来做一些逻辑处理的。

       咱们在讲ng-template的时候就出现了ng-container。ng-container一个重要的作用就是和ng-template一起使用。ng-container还有一个用处就是配合ngFor和×ngIf使用。我们知道ngFor和×ngIf不能同时处在一个元素上。所以咱们想要在不添加额外的html标签的情况下达到同样的效果就得请出ng-container。具体参见如下的代码。

import {Component} from '@angular/core';

@Component({

selector: 'app-ng-container',

template: `

<h1>ng-container</h1>

<ul>

<ng-container \*ngFor="let item of list;let index=index">

<li \*ngIf="index%2 === 0">

{{"index is " + index + " value is " + item}}

</li>

</ng-container>

</ul>

`,

styleUrls: ['./ng-container.component.less']

})

export class NgContainerComponent {

list = ['1号位置', '2号位置', '3号位置', '4号位置', '5号位置', '6号位置', '7号位置', '8号位置', '9号位置'];

}

       关于ng-template、ng-content、ng-container的内容咱们就讲这些。也是为了咱们解析来的自定义控件做一些准备。 本文中设计到的代码下载地址 https://github.com/tuacy/ng-dynamic