

试谈前后端分离及基于前端 MVC 框架的开发

林嘉婷

(广东工业大学自动化学院, 广州 510006)

摘要: 为了解决前后端的强耦合, 彻底解放前端, 提高工作效率, 提出了前后端分离的思想, 给出了基于前端 MVC 框架, 以 Angular2.0 为例的开发流程。

关键词: 前后端分离; Ajax; 前端 MVC 框架; Angular2.0

1 概述

从现在的开发环境来看, 项目大多是前后端强耦合的, 甚至不存在前端的概念。在不重视前端的公司或者部门, 前端开发人员的主要职责就是负责页面的制作和 JS 功能的实现, 有些时候前端的一些页面也由后台人员完成的。如果简单地认为前后端分离就是后台不再需要写 HTML 和 JS 是不对的, 这只是前后端分工。

2 场景及要求

首先后端开发者依赖于前端的静态页面, 只有看到 HTML 文件他们才能开始实现 View 层。而前端又依赖于后端开发者完成整体的开发, 才能通过网络访问来检查最终的效果, 否则他们无法获取真实的数据。

在前后端不分离的情况下, 服务端要单独针对 Web 端做处理, 返回完整 HTML, 这样势必增加服务端的复杂度, 可维护性差, 而 Web 端需要加载完整的 HTML, 一定程度上影响网页的性能。

更糟糕的是, 一旦需求发生变动, 上述的流程还需要重新走一遍, 前后端频繁的交流依旧无法避免。概括来说就是前后端对接成本太高。

现在大多数项目都能通过前后端分离来实现。个人认为前后端分离就是以后开发工作的趋势。大多数应用, 无论是企业级后台应用的前端应用, 还是展示类网站和移动 APP 页面中都可以做成单页面应用, 而单页面最主要的特点就是局部刷新, 这通过前端控制路由调用 Ajax, 后台提供接口便可以实现, 而且这样的方式用户体验更加友好, 网页加载更加快速, 开发和维护成本也降低了不少, 效率明显提升。

3 Ajax 与前端 MVC

Ajax 的全称是 Asynchronous JavaScript And XML, 即“异步 JavaScript 和 XML”。它利用 JavaScript 异步发起请求, 结果以 XML 格式返回, 随后 JavaScript 可以根据返回结果局部操作 DOM。

Ajax 最大的优点是不用重新加载全部数据, 而是只要获取改动的内容即可。Ajax 适合做单页面更新, 得益于 Ajax 的提出, HTML 在前端渲染变成了可能。下载一个空壳 HTML 文件和一个 JavaScript 脚本, 然后在 JavaScript 脚本中获取数据, 为 DOM 添加节点。

这时候出现了很多前端的 MVC 框架, 比如 AngularJS、VueJS 和 ReactJS (可以认为 MVVM 是 MVC 的变种) 等一堆名词, 利用它们可以轻松构建起一个无需服务器端渲染就可以展示的网站, 同时这类框架都提供了前端路由功能, 后台可以不再控制路由的跳转, 将原来属于前端的业务逻辑全部丢给前端, 这样的前后端分离可以说是最为彻底。

4 基于 MVC 框架 AngularJS2.0 的学生信息编辑器

4.1 前期准备

成功安装 node.js 和 npm。

4.2 angular2 项目整体架构

(1) 创建项目文件夹 angular2。

(2) 创建开发环境, 添加 package definition 和 configuration files (以下 4 个文件)。

1) package.json 列出 angular2 应用程序依赖的库和声明一些有用的 scripts。

2) tsconfig.json 是 TypeScript 的编译配置文件;

3) typings.json 标识 TypeScript 声明的文件;

4) systemjs.config.js 是 SystemJS 配置文件。

(3) 安装在 package.json 列出的库, 在终端命令行窗口键入以下命令: npm install。

(4) 在 angular2 项目文件夹下创建子文件夹 app, 用于存放编写的 component file (app/app.component.ts), 记住每一个 angular 应用程序至少有一个根组件, 为了方便人们可以命名为 AppComponent。

```
import {Component} from '@angular/core';
@Component({
  selector: 'my-app',
  template: '<h1>My First Angular 2 App</h1>'
})
export class AppComponent{}
```

Angular 应用是模块化的, app.component.ts 导出 AppComponent。

```
export class AppComponent{}
```

这个导出的行为将这个文件转化为了一个模块 module。

模块依赖其他的模块。在 TypeScript 版本的 Angular 应用

收稿日期: 2016-07-16

中，当其他的模块需要使用 AppComponent 的时候，可以使用下面的语法导入 AppComponent，如下所示：

```
app/boot.ts (import)
import { AppComponent } from './app.component';
```

Angular 也是模块化的，它本身是一系列库模块的集合，每个库就是一个提供服务的模块，被未来的模块所使用。

当需要 Angular 中提供服务的时候，就从 Angular 的库模块中导入。现在就需要 Angular 帮助，以便定义 Component 中的元数据。使用 Angular 提供的 Component 函数来定义 Component 的元数据，通过导入 Angular 的基础库 angular2/core，可以访问这个函数。

```
app/app.component.ts (import)
import { Component } from '@angular/core';
```

在 TypeScript 中，将这个函数应用到类的前缀来装饰这个类，使用 @ 符号，在组件类的上方调用这个函数。

```
app/app.component.ts (metadata)
@Component({
  selector: 'my-app',
  template: '<h1>My First Angular 2 App</h1>'
})
```

@Component 告诉 Angular 这个类是一个组件 Component，配置对象传递给了 @Component 函数，包含两个字段 selector 和 template。selector 指定了一个简单的 CSS 选择器，选择名为 my-app 的 HTML 元素。在 Angular 遇到页面中的 my-app 元素的时候，Angular 将会创建 AppComponent 实例并显示它。template 属性保存伴随的模板，模板来自 HTML，告诉 Angular 如何渲染视图，目前视图只有一行 " My First Angular App"。

(5) Angular 如何加载组件

在 app 文件夹中，添加一个新文件 main.ts。

```
app/main.ts
import { bootstrap } from '@angular/platform-browser-dynamic';
import { AppComponent } from './app.component';
bootstrap(AppComponent);
```

需要两件事情来启动应用：

- 1) Angular 的浏览器函数 bootstrap
- 2) 刚刚编写的根组件

把它们导入进来，然后调用 bootstrap，将 AppComponent 组件作为参数传递给 bootstrap。

已经请求 Angular 将组件作为根来启动应用。Angular 将会把它放在哪里呢？

(6) 添加 index.html 文件

在 index.html 中特定的位置显示应用。不会将 index.html 保存在 app/文件夹中，将它保存在项目的根目录中。

```
index.html
<html>
<head>
</head>
```

```
<base href="/">
<title>Angular 2 QuickStart</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
initial-scale=1">
<link rel="stylesheet" href="stylesheet.css">
<script src="node_modules/core-js/client/shim.min.js"></script>
<script src="node_modules/zone.js/dist/zone.js"></script>
<script src="node_modules/reflect-metadata/Reflect.js"></script>
<script src="node_modules/systemjs/dist/system.src.js"></script>
<script src="systemjs.config.js"></script>
<script>
  System.import('app').catch(function(err){ console.error
(err); });
</script>
</head>
<body>
  <my-app>Loading...</my-app>
</body>
</html>
```

这个文件由 3 个主要的部分组成：

- 1) 加载需要的脚本文件，如 angular2-polyfills.js 和 Rx.js。
- 2) 通过 SystemJS 来导入和加载刚刚编写的启动文件。
- 3) 在 <body> 标记中添加了 <my-app>，这是应用执行的

位置。

当 Angular 在 main.ts 中调用 bootstrap 函数的时候，它将会读取 AppComponent 的元数据，发现定位名为 my-app 的元素，然后在这个元素的标记之间加载应用。

(7) 编译和运行

到这里，一个简单的 Angular2 App 完成，打开一个终端窗口，输入下面的命令：

```
npm start
```

稍等一下，一个浏览器的 Tab 页会被打开，然后显示出：

My First Angular 2 App

4.3 丰富和完善 Angular2 App

(1) 在 APP 中显示信息

在 APP 文件夹中创建 app/student.component.ts。

```
import { Component } from '@angular/core';
@Component({
  selector: 'my-student',
  template: '<h1>{{title}}</h1><h2>{{student}} details! </h2>'
})
export class StudentsComponent {
  title='Tour of Students';
  student='leon';
}
```

在 StudentsComponent 中添加两个属性，并在 @Component

的 template 显示数据中绑定这两个属性。浏览器会刷新和显示这个 title 和 student 属性。如果想让 student 属性内容显示更加丰富，可以在 student.component.ts 顶部，import 语句下面声明一个 Student 类，拥有 id 和 name 属性：

```
export class Student{
  id: number;
  name: string;
}
```

现在，可以重写 component 的 student 属性为 Student 类型：

```
student: Student={
  id:1,
  name:'leon';
}
```

相应地，需要修改 template 里的内容：

```
template:
<h1>{{title}}</h1>
<h2>{{student.name}} details! </h2>
<div><label>id:</label>{{student.id}}</div>
<div>
  <label>name:</label>
  <input value="{{student.name}}" placeholder="name">
</div>
```

注意：这样书写，在<input>标签里面修改名字，并不会在<h2>标签里面反映出来，此时将单向数据绑定转为双向数据绑定。

(2) 实现双向数据绑定
在 template 中使用 ngModel 来实现双向数据绑定，将上述对应的<input>语句改写为

```
<input [(ngModel)]="{{student.name}}" placeholder="name">
```

此时在编辑 student.name 时，会及时反映在<h2>标签里。

(3) 循环显示数据
在 student.component.ts 的底部创建一个数组，如下：

```
var STUDENTS: Student[] = [
  {id: 11, name: 'Aaron'},
  {id: 12, name: 'Narco'},
  {id: 13, name: 'Bombasto'},
  {id: 14, name: 'Celeritas'},
  {id: 15, name: 'Magneta'},
  {id: 16, name: 'Beck'},
  {id: 17, name: 'Dynamia'},
  {id: 18, name: 'Bob'},
  {id: 19, name: 'Magma'},
  {id: 20, name: 'Tornado'}];
```

在 StudentComponent 类中添加这样的属性：
public students=STUDENTS;

没有定义 students 类型，但知道引用了 STUDENTS 数组。首先在标签中添加 *ngFor 属性，并给每个 li 添加绑

定事件，点击事件执行 onSelect() 方法，传入 template 的 student 变量作为参数。

```
<li *ngFor="let student of students" (click)="onSelect(student)">
  <span class="badge">{{student.id}}</span> {{student.name}}
</li>
```

接下来定义 onSelect 方法，在 StudentComponent 中添加 selectedStudent 属性：

```
selectStudent: Student;
```

并在 onSelect 方法中将用户点击的 student 设置给 selectedStudent 属性：

```
onSelect(student: Student) { this.selectedStudent = student; }
```

在 template 中显示点击中的信息，在 template 中绑定新的 selectedStudent 属性：

```
<div *ngIf="selectedStudent">
  <h2>{{selectedStudent.name}} details! </h2>
  <div><label>id:</label>{{selectedStudent.id}}</div>
  <div>
    <label>name:</label>
    <input [(ngModel)]="selectedStudent.name" placeholder="name"/>
  </div>
</div>
```

*ngIf 的作用：如果没有 selectedStudent，会从 DOM 中移除这段 HTML，当用户点击 student，SelectedStudent 存在，则在 DOM 里会显示这个 SelectedStudent 的细节。

4.4 创建提供数据的服务

目前在 StudentsComponent 中定义 STUDENTS 数组去显示，这里，由两个不合适的地方：

- (1) 定义显示的数据不是 component 的工作
- (2) 不方便与其他 component 共享数据

因此，写一个 service 去提供数据，也可以使所有的 component 共享这个 service。

在 APP 下新建文件 student.service.ts 和 mock-students.ts

```
import { Injectable } from '@angular/core';
@Injectable()
export class StudentService {
}
```

TypeScript 看见 @Injectable() 就从 service 发出元数据，在 export 中添加 getStudents() 方法用来获取数据，StudentService 可以是 Web 服务器或者本地文件或者虚拟的数据源。

这里把数据放在自己的文件 mock-students.ts 里面，同时把 Student 类型的定义放到 student.ts 文件里面，所有引用 Student 类型的文件都要如下引入：

```
import { Student } from './student';
```

从 student.component.ts 文件将 STUDENTS 数组剪切到

mock-students.ts 文件中。

在 StudentService 引入 STUDENTS 并将它通过 getStudents 返回：

```
import { Injectable } from '@angular/core';
import { STUDENTS } from './mock-students';
@Injectable()
export class StudentService {
  getStudents() {
    return STUDENTS;
  }
}
```

在其他 component 使用 StudentService：

```
import { StudentService } from './student.service';
```

两行语句创建和引入 StudentService；

(1) 添加 construct，声明了一个私有属性 studentService，用它将 StudentService 引入。

```
constructor(private studentService: StudentService) {
```

(2) 在 @Component 中底部添加 provider 属性

```
providers: [StudentService]
```

在 StudentComponent 中定义 getStudents 方法：

```
getStudents() {
  this.students = this.studentService.getStudents();
}
```

同时在 The ngOnInit Lifecycle Hook 中唤醒 getStudents 方法：

```
import { OnInit } from '@angular/core';
export class AppComponent implements OnInit {
  ngOnInit() {
    this.getStudents();
  }
}
```

4.5 添加路由

(1) 在 index.html 的 <head> 头部部分添加

```
<head>
  <base href="/">
```

(2) 组件 Router 是一个服务，必须 import 它并加入到 providers 数组中，这个组件 Router 是多重服务，要同时引入：

```
(ROUTER_PROVIDERS), multiple directives (ROUTER_DIRECTIVES), and a configuration decorator (RouteConfig).
import { RouteConfig, ROUTER_DIRECTIVES, ROUTER_PROVIDERS } from '@angular/router-deprecated';
```

(3) 在 app.component.ts 中设置 directives 和 providers 数组去包含 router 信息：

```
directives: [ROUTER_DIRECTIVES],
providers: [ROUTER_PROVIDERS, StudentService]
```

(4) 通过 @RouterConfig 给 component 添加配置路由，这个 routers 将告诉 Angular 当用户点击一个链接展示哪个视图界面并在浏览器地址栏复制 Url：

```
@RouteConfig([
  {
```

```
path: '/students',
name: 'Students',
component: StudentsComponent
}
])
```

(5) 重写 app.component.ts 的 template 部分，加入一个 <a> 标签，当点击的时候导航到 StudentComponent 组件：

```
template: `
  <h1>{{title}}</h1>
  <a [routerLink]="['/Students']">Students</a>
  <router-outlet></router-outlet>
`
```

[routerLink] 告诉路由当点击链接的时候导航到哪个组件。

5 结语

对于前后端分离的意义，前端不再需要向后台提供模板或是后台在前端 HTML 中嵌入后台代码。前后端分离的工作流程让前端和后台开发可以同时进行，在后台还没有时间提供接口的时候，前端先将数据写死或者调用本地的 json 文件即可，页面的增加和修改也不必麻烦后台。通过前端路由的配置，可以实现页面的按需加载，服务器不再需要解析前端页面，在页面交互及用户体验上有所提升。通过主流的前端 MVC 框架，可以非常迅速地定位及发现问题的所在，客户端的问题不再需要后台人员参与调试，代码重构及可维护性强。

参考文献

- [1] 郭丹丹. 基于 MVC 的前端开发研究与应用. 北京邮电大学, 2012.
- [2] 乔淑夷. 基于 MVC 模式的 Web 前端框架关键技术研究. 中国海洋大学, 2014.
- [3] 王海洋. 企业级 WEB 前端 MVC 框架设计. 电子科技大学, 2014.
- [4] 陈幼凌. 一种开发和执行均衡高效的 Web 前端框架的研究和实现. 北京邮电大学, 2014.
- [5] 伯乐在线. UED-常胤. 前后端分离的思考与实践, 2014.
- [6] Mario Anzures-García & Luz A. Sánchez-Gálvez & Miguel J. Hornos & Patricia Paderewski-Rodríguez.