Angular框架的研究

梁文杰 17212010063

第一章 研究的背景及意义

1.1 研究工作的背景与意义

前端发展日新月异, 甚至有一句戏言: "每六星期重写一个前端框架", 行业是像火箭一样, 但是web前端的发展又该何去何从呢?

客户端是大厂商的必争之地，今后很多年内，都将受大厂商博弈的影响，处在多浏览器纷争的状态，这意味着开发者仍将在有限的前端支撑下，进行开发。目前看好的是各种ajax框架，这是在有限的前端能力下，小有成效的创新和努力，但是不能从根本上改变，也不能替代直接集成在browser的本地对象，这一点对于前端开发者来说，将处在反复变化的状态，造成web前端技术学习和开发的成本居高不下。

服务端是充满机遇的领地，目前以comet为首的http长连接技术，正试图突破web非实时的传统观念，并且这些应用也将解放客户对web实时能力的需求。随着更高的对web实时性，并发性和可伸缩性的需要，服务端将从传统的web server中分离出，更特殊更多元的服务支撑。可能不仅仅是web services，还应包括直接基于socket的支持。

这种来自服务端的突破口，也将反过来，带动前端的改进。例如ajax对xmlhttprequest、http push对iframe和htmlfile的重新利用，都是为前端更有效地获取来自服务端的强劲支撑而做的变化。今后我们不难想到，前端将衍生出更多的数据传输方式，带来分布式能力更强，实时性更强，和开放性更强的web平台。

但是，web前端的开发成本不但没有降低，反而提高了。有两个理由，一个是技术选择太纷繁，一个是客户对web界面效果的要求被刺激出来了，这都提高了对前端开发的要求。

web应用的技术服务市场，更大的空间来自服务端，原因是客户端早就被浏览器厂商挟持了，只能跟着他们走。服务端则可以自由发挥，来自实时性，并发能力，可伸缩性等方面，都可以有进一步的革新，并且这种革新相对客户端来说，更加不容易被盗版。同时，web的后劲来自对服务端技术的利用，占据了服务端集成和开发能力的厂商将在web市场上保持一定的技术领先。

1.2 目前WEB发展的概况

从web1.0到现在的web2.0以及之后的web3.0，web的内容已经有了非常大的改变了。

Web2.0 所强调的更多的是人与人之间的关系，而不是 人与内容的关系。它的作用在于个体对整个社会信息的接受 , 与个体信息放大到整个社会的转换。在一定程度上网络结构 的划分，是继承了传统信息所具有自己的特质的系统，也是 建立在内容之上的文化网络。现代人的生活离不开网络，聊 天、邮件增进人们之间的感情。恋爱、网络会议、订单提高 效率增加利润。网上直播、新闻、网络购物足不出户了解天下事。科学发展，卫星探测，学术研讨，军事演习这些我们 都可以通过互联网来了解。为什么 Web 这么重要?因为我们生活在一个实时的世界中，因此 Web 的最终最自然的状 态也应当是实时的。用户需要实时的沟通、数据和搜索。这 就是我们对互联网信息实时性的要求也越来越高了。也让 Web 无孔不入的渗透进我们的日常中，更多的为服务带来便 利。 Web2.0 到来给我们的生活带来了那些改变呢?科技改 变生活，对每个人都是息息相关的。对于传感器访问赋予页 面对用户环境的感知能力，很多年以来 Web 页面掌握的用 户情况十分有限，只有用户的屏幕尺寸以及浏览器类型等。 但现在各种 W3C 标准把环境光、麦克风、摄像头等各种传 感器数据都提供出来了。这使得网站可以跳出页面之外掌握 更为丰富的信息。有一些人也许会对浏览器具备这种功能感 到忧虑。其实不用担心，这些功能的使用都需要经过权限许 可，在使用的时候会有可见的指示。

同时，移动设备的发展使得web又有了新的内涵。

随着移动互联网的不断发展，移动设备的快速崛起， App市场的发展也日渐迅猛，目前iOS和Android平台的 App数量已经达到非常之多，包括移动操作系统在内的竞争也愈演愈烈。目前iOS、Android、Windows Phone、等多个平台的移动应用开发就让程序员疲惫不堪。但随着HTML5技术的逐渐成熟以及被主流厂商所接受之后，使得Web App发展势头强劲，前途一片光明，已经成为了主流的趋势。Web App因为是使用W3C标准的HTML语言开发，因此能够轻松实现跨平台，Web App开发者也不再需要考虑复杂的底层适配和跨平台开发语言的问题。同时，使用HTML来开发的Web App在投入上也会大大的低于传统的Native App。移动端已经成为了web发展的一大趋势。

同时，随着ES6的逐渐落实，web技术也发生着翻天覆地的变化。

ES6是一次重大的版本升级，与此同时，由于ES6秉承着最大化兼容已有代码的设计理念，你过去编写的JS代码将继续正常运行。事实上，许多浏览器已经支持部分ES6特性，并将继续努力实现其余特性。这意味着，在一些已经实现部分特性的浏览器中，你的JS代码已经可以正常运行。如果到目前为止你尚未遇到任何兼容性问题，那么你很有可能将不会遇到这些问题，浏览器正飞速实现各种新特性。

但是，ES6并非如此！经过持续几年的磨砺，它已成为JS有史以来最实质的升级，新的语言和库特性就像无主之宝，等待有识之士的发掘。新的语言特性涵盖范围甚广，小到受欢迎的语法糖，例如箭头函数（arrow functions）和简单的字符串插值（string interpolation），大到烧脑的新概念，例如代理（proxies）和生成器（generators）。

同时，随着各种前后端框架的发展，web技术相较往日已经变得更加健壮，更加复杂。

1.3 目前常见的WEB技术

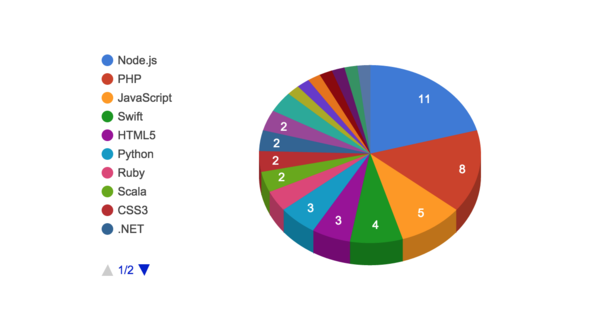
开发WEB可以使用很多种技术，比较流行的主要如下图1-1.

图1-1 目前几种主要的WEB技术和使用情况

Node.js第一，PHP第二，JavaScript第三。因为Node.js的社区很多，所以这个结果你也不用太意外。如果你知道JavaScript,你已经一只脚踏进了用Node.js来构建Web APP的可能。而webapp将逐渐主导移动端应用市场（成本低、开发效率高、几乎替代80%以上的原生调用、重点：免审核、更新太麻烦）

同时，web技术使用了非常多的框架。好的框架可以帮助开发者们快速的构建程序，节省的大量的时间，提高了开发效率。而且好的框架的开发思想，代码的使用效率都是非常好的，开发者们在这些框架的基础上可以拥有更好的开发体验，为开发出更好的web应用提供了必要条件。

目前主要流行的框架如下图1-2所示。

图1-2 目前流行的WEB框架和使用情况

AngularJS处于框架的领导地位。大量的公司和企业已经采用Angular，这就要求开发者更多的具备使用这个框架的技能。其中有个一重要的原因是Google的支持。

但在WEB这个高速发展的领域中，框架的更新非常之快，技术的更迭也极为快速，许多框架正在诞生，也有许多框架正在消亡。如Vue.js在统计图统计之时并没有很大的使用量，但现在其流行程度已经不亚于Angular和React了。

第二章 Angular框架的介绍

2.1 Angular的基本信息

Angular（通常称为“ Angular 2+ ”或“ Angular 2 ”）是由Google角色团队和个人和公司社区领导的基于TypeScript的开源前端Web应用平台，以解决所有在构建复杂的Web应用程序时，开发人员的工作流程。Angular是来自构建AngularJS的同一个团队的完整重写。

早在2009年，Angular的前一代AngularJS已经诞生，由Misko Hevery 等人创建，后为Google所收购。是一款优秀的前端JS框架，已经被用于Google的多款产品当中。AngularJS有着诸多特性，最为核心的是：MVC、模块化、自动化双向数据绑定、语义化标签、依赖注入等等。

AngularJS是为了克服HTML在构建应用上的不足而设计的。HTML是一门很好的为静态文本展示设计的声明式语言，但要构建WEB应用的话它就显得乏力了。

AngularJS是基于JavaScript的开源前端Web应用程序框架，主要由Google和个人和企业社区维护，以解决许多在发展中遇到的挑战单页面应用程序。JavaScript组件补充了Apache Cordova，用于开发跨平台移动应用程序的框架。它旨在通过为客户端提供一个模型-视图-控制器（MVC）框架，简化开发和测试这些应用程序的model-view-viewmodel（MVVM）架构，以及丰富的Internet应用程序中通常使用的组件。

AngularJS双向数据绑定是其最显着的功能，大大减轻了模板职责的服务器后端。相反，根据模型中定义的范围中包含的数据，模板将以纯HTML呈现。$scopeAngular中的服务检测到模型部分的更改，并通过控制器修改视图中的HTML表达式。同样，对该视图的任何改变都反映在模型中。这规避了主动操纵DOM的需要，并鼓励Web应用程序的引导和快速原型设计。AngularJS通过将当前值与脏检查过程中较早存储的值进行比较来检测模型中的变化，与Ember.js和Backbone.js不同当模型值更改时触发侦听器。

在2014年，原来的AngularJS团队开始了Angular的开发。

Angular和AngularJS之间的差异：

1、Angular没有“范围”或控制器的概念，而是使用组件层次结构作为其主要架构概念

2、移动开发 - 首先处理移动性能问题时，桌面开发更容易

3、模块化 - 许多核心功能已经转移到模块，产生更轻，更快的核心

4、只有现代浏览器能支持——减少了需要浏览器兼容的解决方法

5、Angular建议使用Microsoft的TypeScript语言，它引入以下功能：

基于类的面向对象编程

静态类型

泛型

（TypeScript是ECMAScript 6（ES6）的超集，并且与ECMAScript 5（即：JavaScript）向后兼容。）

2.2 Angular CLI

Angular CLI是一个命令行界面工具，它可以创建项目、添加文件以及执行一大堆开发任务，比如测试、打包和发布。Angular CLI需要node.js环境。

在Demo中会使用Angular-CLI来让Angular应用从中获益。

打开终端窗口，运行下列命令来生成一个新项目以及应用的骨架代码：

ng new anction

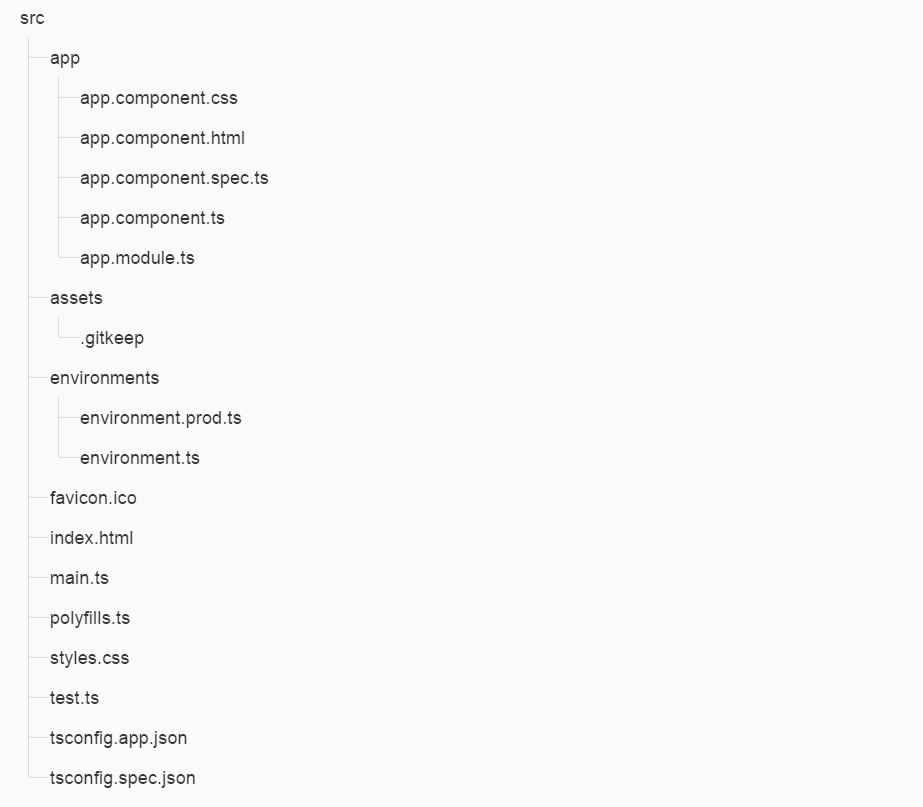
进入项目目录，并启动服务器：

cd auction

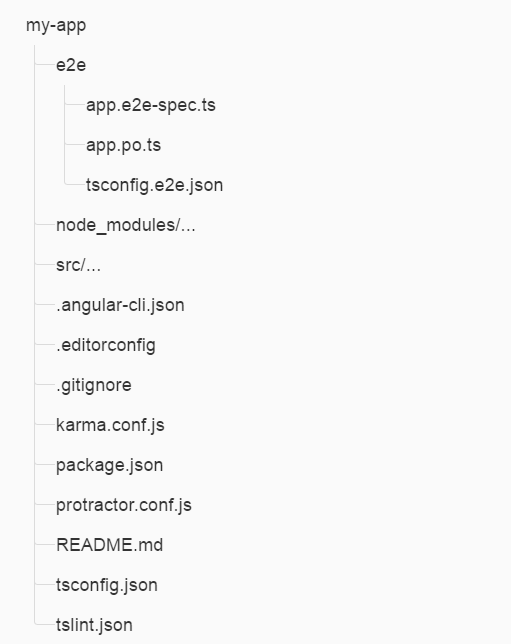
ng server

ng server命令会启动开发服务器，监听文件变化，并在修改这些文件时重新构建此应用。使用--open（或-o）参数可以自动打开浏览器并访问<http://localhost:4200/>。

你的应用代码位于src文件夹中。 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。 这个文件夹之外的文件都是为构建应用提供支持用的。构建的src目录树如下图：



src/文件夹是项目的根文件夹之一。 其它文件是用来帮助你构建、测试、维护、文档化和发布应用的。它们放在根目录下，和src/平级。根目录下文件如下图：



ng build 将应用程序编译到输出目录中

构建工件将被存储在dist/目录中。

所有构建或提供项目的命令ng build/serve/e2e都将删除输出目录（dist/默认情况下）。这可以通过--no-delete-output-path（或--delete-output-path=false）标志禁用。

2.3 Angular的架构

Angular 是一个用 HTML 和 JavaScript 或者一个可以编译成 JavaScript 的语言（例如 Dart 或者 TypeScript ），来构建客户端应用的框架。

该框架包括一系列库，有些是核心库，有些是可选库。

我们是这样写 Angular 应用的：用 Angular 扩展语法编写 HTML 模板， 用组件类管理这些模板，用服务添加应用逻辑， 用模块打包发布组件与服务。

然后，我们通过引导根模块来启动该应用。 Angular 在浏览器中接管、展现应用的内容，并根据我们提供的操作指令响应用户的交互。

总体架构图下所示。

**模块：**

Angular 应用是模块化的，并且 Angular 有自己的模块系统，它被称为 Angular 模块或 NgModules。

每个 Angular 应用至少有一个模块（根模块），习惯上命名为AppModule。

根模块在一些小型应用中可能是唯一的模块，大多数应用会有很多特性模块，每个模块都是一个内聚的代码块专注于某个应用领域、工作流或紧密相关的功能。

Angular 模块（无论是根模块还是特性模块）都是一个带有@NgModule装饰器的类。

**组件：**

组件负责控制屏幕上的一小块区域，我们称之为视图。

我们在类中定义组件的应用逻辑，为视图提供支持。 组件通过一些由属性和方法组成的 API 与视图交互。

当用户在应用中漫游时， Angular 会创建、更新和销毁组件。 应用可以通过生命周期钩子在组件生命周期的各个时间点上插入自己的操作。

**模板：**

我们通过组件的自带的模板来定义组件视图。模板以 HTML 形式存在，告诉 Angular 如何渲染组件。

多数情况下，模板看起来很像标准 HTML，当然也有一点不同的地方。

模板除了可以使用像<h2>和<p>这样的典型的 HTML 元素，还能使用其它元素。 例如，像\*ngFor、{{hero.name}}、(click)、[hero]和<hero-detail>这样的代码使用了 Angular 的模板语法。

**数据绑定：**

如果没有框架，我们就得自己把数据值推送到 HTML 控件中，并把用户的反馈转换成动作和值更新。 如果手工写代码来实现这些推/拉逻辑，肯定会枯燥乏味、容易出错，读起来简直是噩梦 —— 写过 jQuery 的程序员大概都对此深有体会。

Angular 支持数据绑定，一种让模板的各部分与组件的各部分相互合作的机制。 我们往模板 HTML 中添加绑定标记，来告诉 Angular 如何把二者联系起来。

图下所示，数据绑定的语法有四种形式。每种形式都有一个方向 —— 绑定到 DOM 、绑定自 DOM 以及双向绑定。

例如：

{{hero.name}}插值表达式在<li>标签中显示组件的hero.name属性的值。

[hero]属性绑定把父组件HeroListComponent的selectedHero的值传到子组件HeroDetailComponent的hero属性中。

(click) 事件绑定在用户点击名字时调用组件的selectHero方法。

[(ngModel)]=”hero.name” 双向数据绑定。

在双向绑定中，数据属性值通过属性绑定从组件流到输入框。用户的修改通过事件绑定流回组件，把属性值设置为最新的值。

Angular 在每个 JavaScript 事件循环中处理所有的数据绑定，它会从组件树的根部开始，递归处理全部子组件。

**服务：**

服务是一个广义范畴，包括：值、函数，或应用所需的特性。

几乎任何东西都可以是一个服务。 典型的服务是一个类，具有专注的、明确的用途。它应该做一件特定的事情，并把它做好。

例如：

日志服务

数据服务

消息总线

税款计算器

应用程序配置

服务没有什么特别属于 Angular 的特性。 Angular 对于服务也没有什么定义。 它甚至都没有定义服务的基类，也没有地方注册一个服务。

即便如此，服务仍然是任何 Angular 应用的基础。组件就是最大的服务消费者。

服务无处不在。

组件类应保持精简。组件本身不从服务器获得数据、不进行验证输入，也不直接往控制台写日志。 它们把这些任务委托给服务。

组件的任务就是提供用户体验，仅此而已。它介于视图（由模板渲染）和应用逻辑（通常包括模型的某些概念）之间。 设计良好的组件为数据绑定提供属性和方法，把其它琐事都委托给服务。

Angular 不会强制要求我们遵循这些原则。 即使我们花 3000 行代码写了一个组件，它也不会抱怨什么。

Angular 帮助我们遵循这些原则 —— 它让我们能轻易地把应用逻辑拆分到服务，并通过依赖注入来在组件中使用这些服务。

**依赖注入：**

“依赖注入”是提供类的新实例的一种方式，还负责处理好类所需的全部依赖。大多数依赖都是服务。 Angular 使用依赖注入来提供新组件以及组件所需的服务。

Angular 通过查看构造函数的参数类型得知组件需要哪些服务。

当 Angular 创建组件时，会首先为组件所需的服务请求一个注入器 (injector)。注入器维护了一个服务实例的容器，存放着以前创建的实例。 如果所请求的服务实例不在容器中，注入器就会创建一个服务实例，并且添加到容器中，然后把这个服务返回给 Angular。 当所有请求的服务都被解析完并返回时，Angular 会以这些服务为参数去调用组件的构造函数。 这就是依赖注入 。

简单点说，我们必须先用注入器（injector）为HeroService注册一个提供商（provider）。 提供商用来创建或返回服务，通常就是这个服务类本身

我们可以在模块中或组件中注册提供商。但通常会把提供商添加到根模块上，以便在任何地方都使用服务的同一个实例。或者，也可以在@Component元数据中的providers属性中把它注册在组件层：

把它注册在组件级表示该组件的每一个新实例都会有一个服务的新实例。

需要记住的关于依赖注入的要点是：

·依赖注入渗透在整个 Angular 框架中，被到处使用。

·注入器 (injector) 是本机制的核心。

注入器负责维护一个容器，用于存放它创建过的服务实例。

注入器能使用提供商创建一个新的服务实例。

·提供商是一个用于创建服务的配方。

·把提供商注册到注入器。

2.4 Angular的优缺点

Angular的优点：

1. 模板功能强大丰富，并且是声明式的，自带了丰富的Angular指令；

2. 是一个比较完善的前端MVC框架，包含模板，数据双向绑定，路由，模块化，服务，过滤器，依赖注入等所有功能；

3. 自定义Directive(指令)，比jQuery插件还灵活，但是需要深入了解Directive的一些特性，简单的封装容易，复杂一点官方没有提供详细的介绍文档，可以通过阅读源代码来找到某些我们需要的东西，如：在directive使用 $parse；

4. ng模块化比较大胆的引入了Java的一些东西（依赖注入），能够很容易的写出可复用的代码，对于敏捷开发的团队来说非常有帮助。

Angular的缺点：

1. 验证功能错误信息显示比较薄弱，需要写很多模板标签，没有jQuery Validate方便，所以可以自己封装了验证的错误信息提示；

2. ngView只能有一个，不能嵌套多个视图，虽然有

angular-ui/ui-router GitHub 解决，但是貌似ui-router 对于URL的控制不是很灵活，必须是嵌套式的；

3. 对于特别复杂的应用场景，貌似性能有点问题，特别是在Windows下使用chrome浏览器，不知道是内存泄漏了还是什么其他问题，没有找到好的解决方案，奇怪的是在IE10下反而很快；

4. 这次从1.0.X升级到1.2.X，貌似有比较大的调整，没有完美兼容低版本，升级之后可能会导致一个兼容性的BUG，具体详细信息参考官方文档

AngularJS ，对应的中文版本：Angular 1.0到1.2 迁移指南

5. ng提倡在控制器里面不要有操作DOM的代码，对于一些jQuery 插件的使用，如果想不破坏代码的整洁性，需要写一些directive去封装插件，但是现在有很多插件的版本已经支持Angular了，如：jQuery。

6. Angular 太笨重了，没有让用户选择一个轻量级的版本，当然1.2.X后，Angular也在做一些更改，比如把route，animate等模块独立出去，让用户自己去选择。

第三章 使用Angular搭建的DEMO

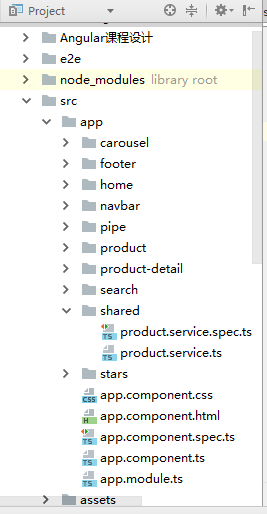
3.1架构及概览

DEMO提供了一个商品信息的展示的系统，用户可以浏览商品并且进行相应的评论和打分。

页面总体界面如下：



创建的组件如图所示：



具体为：

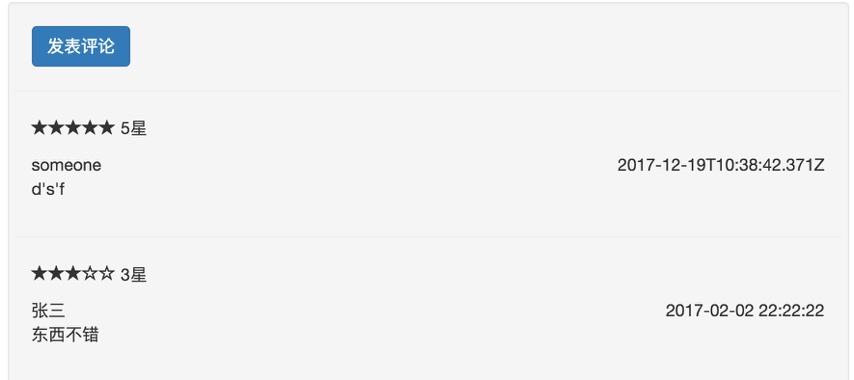
1. carousel 轮播器组件，用以展示最新信息的轮播图。
2. footer 页脚组件，用以显示页脚。
3. home 主页组件，是页面的整体框架，包含了一些子组件。
4. navbar 页头导航栏组件，显示页头的导航栏条。
5. pipe 管道 编写一些自定义的管道格式化页面数据。
6. product 产品组件，负责产品列表在主页按需展示。
7. product-detail 产品细节组件 负责点击产品进入的产品细节页面。

页面如下：



同时，发表评论下还可以给当前产添加自己的评论，同时评价星级。如下图：





1. search 搜索栏组件。
2. shared 一些共享的服务。
3. star 星级评分组件，嵌入在各个需要它的地方。

除了上述的组成之外，项目还引入了bootstrap和jquery来完成一些前端样式和一些其他的操作。

3.2主要组件的具体实现

**根组件appCcomponent：**

<app-navbar></app-navbar>  
  
<div class="container">  
 <div class="row">  
 <div class="col-md-3">  
 <app-search></app-search>  
 </div>  
 <div class="col-md-9">  
 <router-outlet></router-outlet>  
 </div>  
 </div>  
</div>  
  
<app-footer></app-footer>

页面带入如上，其中包含了页头页脚，然后主体按一三分分成两部分，左侧三分之一为搜索组件，右边定义了一个路由插座以实现路由功能。

路由是 Angular 应用程序的核心，它加载与所请求路由相关联的组件，以及获取特定路由的相关数据。这允许我们通过控制不同的路由，获取不同的数据，从而渲染不同的页面。

**产品组件Product：**

前端展示代码如下：

<div class="row">  
 <div class="col-sm-12">  
 <div class="form-group">  
 <input class="form-control" placeholder="请输入商品名称"  
 [formControl]="titleFilter">  
 </div>  
 </div>  
</div>  
  
<div \*ngFor="let product of products | filter:'title':keyword" class="col-md-4 col-sm-4 col-lg-4">  
 <div class="thumbnail">  
 <img [src]="imgUrl" alt="加载失败">  
 <div class="caption">  
 <h4 class="pull-right">{{product.price}}元</h4>  
 <h4><a [routerLink]="['/product', product.id]">{{product.title}}</a></h4>  
 <p>{{product.desc}}</p>  
 </div>  
 <div>  
 <app-stars [rating]="product.rating"></app-stars>  
 </div>  
 </div>  
</div>

主要包括了产品列表在主页下的显示方式。

TS脚本注入了ProduceService服务，并且对产品搜索框定义了响应式的方法，每隔0.5秒就会检查搜索框内绑定的数据然后根据产品名称过滤应该显示的产品列表，显示回馈的节奏非常的符合用户体验的要求。主要代码如下：

**constructor**(**private** productService: ProductService) {  
 **this**.titleFilter.valueChanges  
 .debounceTime(500)  
 .subscribe(  
 value => **this**.keyword = value  
 );  
}  
ngOnInit() {  
 **this**.products = **this**.productService.getProducts();  
}

**产品详情组件Product-detail：**

前端展示代码如下：

<div class="thumbnail">  
 <img src="http://via.placeholder.com/820x230" alt="">  
 <div>  
 <h4 class="pull-right">{{product.price}}元</h4>  
 <h4>{{product.title}}</h4>  
 <p>{{product.desc}}</p>  
 </div>  
 <div>  
 <p class="pull-right">{{comments.length}}</p>  
 <p>  
 <app-stars [rating]="product.rating"></app-stars>  
 </p>  
 </div>  
</div>  
  
<div class="well">  
 <div>  
 <button class="btn btn-primary" (click)="isCommentHidden = !isCommentHidden">发表评论</button>  
 </div>  
 <div [hidden]="isCommentHidden">  
 <div><app-stars [(rating)]="newRating" [readonly]="false"></app-stars></div>  
 <div>  
 <textarea [(ngModel)]="newComment"></textarea>  
 <div>  
 <button class="btn btn-primary " (click)="addComment()">提交</button>  
 </div>  
 </div>  
 </div>  
 <div class="row" \*ngFor="let comment of comments">  
 <hr>  
 <div class="col-md-12">  
 <app-stars [rating]="comment.rating"></app-stars>  
 <span>{{comment.user}}</span>  
 <span class="pull-right">{{comment.timestamp}}</span>  
 <p>{{comment.content}}</p>  
 </div>  
 </div>  
</div>

主要包含了产品的介绍和评论相关的内容。

其中发表评论中的星级采用了双向数据绑定，将点击绑定到后台处理，并把更改够的星级同时反映到前端的显示，实现了良好的交互。

**星级组件Star：**

前端的星级布局如下：

<p>  
 <span \*ngFor="let star of stars; let i=index" class="glyphicon glyphicon-star"  
 [class.glyphicon-star-empty]="star" (click)="clickStar(i)"></span>  
 <span>{{rating | number:'1.0-1'}}星</span>  
</p>

主要按星级变量控制了显示亮起来的星星个数。

主要TS脚本函数如下：

ngOnChanges(changes: SimpleChanges): **void** {  
 **this**.stars = [];  
 **for** (**let** i = 1; i <= 5; i++){  
 **this**.stars.push(i > **this**.rating);  
 }  
}

clickStar(index: **number**) {  
 **if** (!**this**.readonly) {  
 **this**.rating = index + 1;  
 // this.ngOnInit();  
 **this**.ratingChange.emit(**this**.rating);  
 }  
}

主要为如果当前的星星比评分小，就点亮这颗星，表示这一分得到了。

这其中还包含了一个emit发射函数把星级数据发送出去，实现了和产品细节组件中需要显示星级那部分的数据交互

**自定义管道Pipe：**

主要实现的功能代码如下：

@Pipe({  
 name: 'filter'  
})  
**export class** FilterPipe **implements** PipeTransform {  
 transform(list: **any**[], filterField: **string**, keyword: **string**): **any** {  
 **if** (!filterField || !keyword) {  
 **return** list;  
 }  
 **return** list.filter( item => {  
 **let** fieldValue = item[filterField];  
 **return** fieldValue.indexOf(keyword) >= 0;  
 });  
 }  
}

定义了一个名为filter的管道，可以直接在html页面中使用 | 来对数据进行操作。这里的意义是通过输入值去过滤内容，实现了简单的搜索功能。

**ProductService服务：**

服务主要是为了减少重复书写需要使用的功能代码，已提高编码效率，减少工作量。ProductService服务代码如下：

getProducts(): Product[] {  
 **return this**.products;  
 }  
 getProduct(id: **number**): Product {  
 **return this**.products.find((product) => product.id == id);  
 }  
 getCommentsForProductId(id: **number**): Comment[] {  
 **return this**.comments.filter((comment: Comment) => comment.productId == id);  
 }  
}  
**export class** Product {  
 **constructor**(  
 **public** id: **number**,  
 **public** title: **string**,  
 **public** price: **number**,  
 **public** rating: **number**,  
 **public** desc: **string**,  
 **public** categories: Array<**string**>  
 ) {  
 }  
}  
**export class** Comment {  
 **constructor**(**public** id: **number**,  
 **public** productId: **number**,  
 **public** timestamp: **string**,  
 **public** user: **string**,  
 **public** rating: **number**,  
 **public** content: **string** ){  
 }  
}

TS脚本中主要对Product产品和Comment评论所包含的数据模型进行了定义。同时，定义了一些对这些模型操作的基本函数接口，比如得到Product列表、按id得到产品、从产品id得到这个产品的评论。

第四章 总结

目前WEB技术发展迅速，各种新技术层出不穷，更新迭代速度可以说是非常之快。各种新方法、新思想之间的相互碰撞，也促进着这个行业的快速发展。

但是，最为开发者，我们不应在纷繁复杂的各种技术中迷失了方向，要时刻保持自己头脑的清醒，选择最适合自己的技术。各种框架、标准都是会随着时代的发展而发展的，不可能存在永恒不变的万金油技术。要是我们盲目追捧新技术而忽略了自己的实际需求，永远只停留在浅薄的层面上，那等待我们的只有无底洞。

同时，保持自己技术的新颖性也是必不可少的。如果只是守着一些过时的老东西，不但会停止自己的进步，也会被时代的潮流所淹没。一切都是在不断发展着的，web也是如此。

参考文献

1. https://en.wikipedia.org/wiki/AngularJS
2. https://en.wikipedia.org/wiki/Angular\_(application\_platform)
3. Ari Lerner, Felipe Coury, Nate Murray, Carlos Taborda. Angular权威教程. 人名邮电出版社. 2017,3
4. https://angularjs.org
5. https://reactjs.org
6. http://www.typescriptlang.org
7. 阮一峰. ECMAScript 6入门. 电子工业出版社, 2017,9
8. Kyriakos-Ioannis D.Kyriakou, Ioannis K.Chaniotis, Nikolaos D.Tselikas. The GPM meta-transcompiler: Harmonizing JavaScript-oriented Web Development with the upcoming ECMAScript 6 “Harmony” Specification. Consumer Communications & Networking Conference, 2015:176-181
9. 廖月, 刘娜. 浅谈Web 2.0时代的现状及发展. 中国通信, 2017, 19(8):62-62