# 基于JavaScript 全栈开发C2C订餐系统

姓名： 朱凌浩

学号： 16222010647

1 课题信息 2

3 系统架构 2

4 MSE-FOOD中所使用的前端流行技术 7

5 MSE-FOOD中所使用的自动化测试方案 8

6 MSE-FOOD中所使用的调试工具 9

7 MSE-FOOD下一版本畅想 9

8 MSE-FOOD系统设计 12

9 MSE-FOOD开发流程 14

10 MSE-FOOD配置运行 16

11 附录：相关链接 17

## 课题信息

## 项目名称：MSE-FOOD

## Github地址： <https://github.com/linghao518/mse-food> (源文件在src目录中)

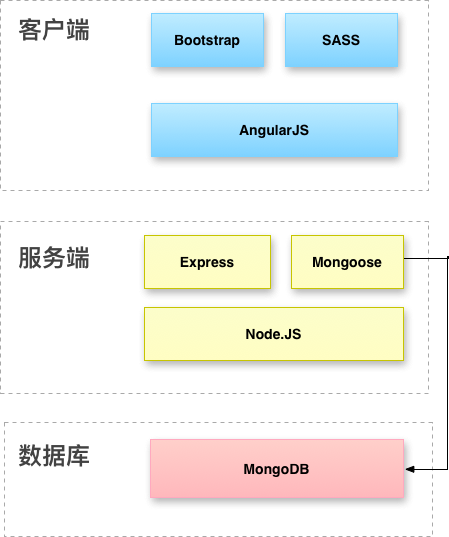
## 项目线上预览地址： http://linghao.me:5000 管理员账号：admin 管理员密码：admin

## 课题信息

## 本人是前端开发工程师，平时喜欢关注最近的JavaScript的全栈开发动向。因此，在本次课程设计中选择了使用JavaScript全栈技术开发Web应用。

项目选择开发一个简单的C2C在线订餐系统，用户可以发布自己所烹饪的菜肴供其他用户订餐。使用了MEAN全栈开发环境开发这个应用。

## 系统架构

* 1. **MEAN架构概述**  
     MEAN指的是以JavaScript为平台的现代WEB开发的技术栈，它是MongoDB + Express +AngularJS + NodeJS 四个框架的第一个字母组合。 在JavaScript界流行着一句话 “LAMP is so yesterday, MEAN is so tomorrow”，意为：以JavaScript为核心的全栈框架正在成为WEB开发界的主流。  
       
       
       
       
       
     系统架构图：  
     
  2. **AngularJS在MSE-FOOD中的应用**

最近几年前端技术飞速发展，从前端只是切图做页面，JavaScript做动效、做表单校验的过程到了现在前端成为一个完整的软件工程生态圈，JavaScript可以担任起各种复杂的业务逻辑的处理。JavaScript随着项目的复杂度日益增加，类似AngularJS这样的MV\* 框架应运而生，它借鉴了很多传统服务端面向对象的语言的一些特性，使得JavaScript开发也变得更为严肃，更为工程化。  
  
**在**MSE-FOOD项目中完整使用了AngularJS的大部分的特性：

* + - 1. Module (模块化)  
         项目分为四大模块：用户模块、菜品模块、订单模块、系统模块。AngularJS的模块化特性使各个模块之间解除耦合、更好的组织代码，在大型项目中模块化的特性可方便团队之间独立的高效开发。
      2. Router (路由)  
         在路由中配置了各个url所对应的template和controller。 项目中使用了angular-ui 开发的ui-router组件，可以使得view能够多级嵌套，能够实现复杂的页面布局。
      3. Service (服务层)  
         在服务层使用了$resource 内置provider，只需短短几行代码即可方便实现与服务端的RESTful 服务的交互。
      4. Contoller (控制器层)  
         在控制器层主要对RESTful获得的数据进行处理和对提交表单的数据进行处理。
      5. Directive (指令)  
         所有所需要操作DOM的代码均放入了指令中。一些可以重复的使用的组件也抽象出来放入了指令中。
      6. Template (模版)  
         AngularJS自带的模版引擎也非常强大，可以在模版中对数据进行遍历操作、逻辑操作等。
      7. Two-Way-Bind (双向绑定)  
         无需任何事件绑定即可方便对表单中的数据进行处理、验证等操作，在对页面的数据处理过程中几乎不需要任何DOM操作，大大提高开发效率。
      8. Inject (注入依赖)  
         借鉴自Java的设计模式。可以方便的使得Service注入到Controller层中使用Service中的方法，也可使得不同的模块之间进行交互，并且可以重用Service。

AngularJS为我们造好平时需要花费大量精力构建的模式、框架，使我们开发者只需要关注业务实现。在我平时工作的使用中也对团队协作的效率提升很大，并且团队中大家写的代码风格也更为规范、统一。

* 1. **Node.JS + Express在MSE-FOOD中的应用**Express是Node.JS的一个web应用开发框架，也是一个非常典型的MVC框架，它已经为我们封装好了http的request、response的操作。  
       
     在MSE-FOOD项目中使用到的Express特性：
     + 1. Router (路由)  
          Express的路由定义应用的端点（URIs）以及如何响应客户端的请求，将请求地址在回调中进行处理或分发到对应的中间件进行处理。
       2. Middleware (中间件)  
          Express的本质就是在应用中调用各种中间件。它可以访问请求对象（request）, 响应对象（response）, 和 web 应用中处于请求-响应循环流程中的中间件，一般被命名为 next 的变量。
       3. Template (模版引擎)  
          Express的模版引擎和AngularJS的模版引擎类似，只不过模版在服务端直接渲染后显示到客户端。MSE-FOOD项目使用的是单页面应用的设计方案，因此仅首页是通过Express在服务端渲染的。

* 1. **MongoDB在MSE-FOOD中的应用**MongoDB是目前比较流行的NoSQL，以键值对的形式存储为BSON格式(Binary JSON)，可以极为方便的通过Node.JS获取到MongoDB的数据，甚至在数据通过RESTful传到客户端的过程中无需对数据进行任何处理。并且可以通过MapReduce对数据进行分布式聚合，满足高并发项目的需求。  
     在MSE-FOOD项目中使用到的MongoDB特性：
     + 1. 面向集合  
          数据被分组存储在数据集中，被称为一个集合（Collenction)。每个集合在数据库中都有一个唯一的标识名，并且可以包含无限数目的文档。集合的概念类似关系型数据库（RDBMS）里的表（table），不同的是它不需要定义任何模式（schema)。
       2. 模式自由  
          存储在mongodb数据库中的文件，在插入新数据时不需要知道它的任何结构定义。并且可以把不同结构的文件存储在同一个集合里。
  2. **Mongoose在MSE-FOOD中的应用**Mongoose是node.js 中连接mongoDB的一个类库，可以将mongoDB的数据映射成为node.js中的一个对象，类似于Java中的ORM层，使得对象可以方便的进行save, update, remove等一系列操作。  
       
     Mongoose中非常常用的方法population可以将不同的document进行关联，解决NoSQL不能进行类似SQL表关联的操作。在MSE-FOOD中需要对Food和User，Order和Food、User的关联，即使用了population方法。

## MSE-FOOD中所使用的前端流行技术

* 1. **响应式布局**使用CSS3中的Media Query为不同的分辨率设置不同的显示样式，实现一份代码能够兼容多个不同分辨率的客户端，无需为移动端单独开发一个版本。
  2. **CSS3**MSE-FOOD项目中大量了使用了CSS3的动画效果，例如：首页菜肴缩略图的悬浮效果、点击后的大图、详情的展开效果。相较于传统的jQuery 动画操纵元素的坐标，CSS3的动画更为流畅，并且只需要在CSS样式中即可实现动画效果，无需任何JS代码。
  3. **Bootstrap**Bootstrap中包含了丰富的Web组件，使用这些组件，可以快速的搭建一个漂亮、功能完备的Web应用。使用Bootstrap提供的栅格布局系统能快速搭建响应式布局的页面。
  4. **SASS**Sass 扩展了 CSS，增加了变量、嵌套、混入、选择器、继承等等特性,易于组织和维护， 在实际开发过程中实现同样的样式效果代码量要比CSS少很多。
  5. **AngularJS**如上一章节所述，AngularJS提供了一个完整的MV\* 的JavaScript框架，摒弃了使用传统jQuery库花费在DOM操作上的大量时间，使用AngularJS的双向绑定的特性几乎不需要在处理数据时对DOM进行操作。
  6. **Yeoman**Yeoman 通过三个工具(Yo, Gulp, Bower)的协同工作，为开发者创建一个易用的工作流。它可以让前端开发者快速搭建一个前端应用脚手架。在MSE-FOOD中，我使用了Yeoman构建了MEAN的脚手架，脚手架中已包含了gulp的配置、数据库的连接配置、简单的CRUD示例代码，我只需要在这基础上实现自己的业务逻辑，繁琐的构建过程Yeoman已帮我搞定。
  7. **NPM**Node.JS 和 Gulp中的依赖包都有NPM统一管理，类似Java中的Maven。项目中所使用到Express、Gulp等等类库或工具都是通过NPM安装、管理的。
  8. **Bower**Bower解决JavaScript的依赖管理，MSE-FOOD中使用的Boostrap、AngularJS、jQuery都是通过Bower自动下载、管理的。
  9. **Gulp**使用Gulp 自动化构建工具, 可以使用它在项目开发过程中自动执行常见的前端构建任务。比如：Sass的编译、压缩，JavasScript校验、压缩、混淆，自动化测试，发布打包等操作，其中的livereload可以监听文件的修改，自动刷新浏览器的页面，不用每次改一行代码手动刷新一下页面，大大提高了开发效率。

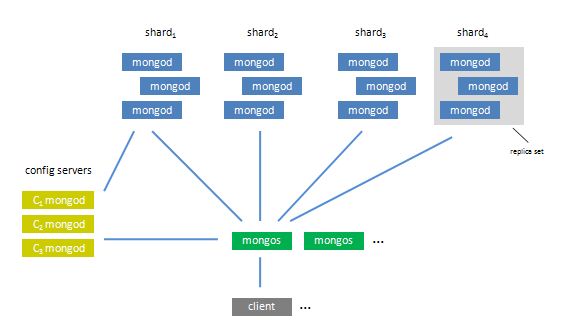
## MSE-FOOD中所使用的自动化测试方案

* 1. **前端自动化测试**使用karma + mocha对JavaScript代码进行自动化测试，加上AngularJS的依赖注入的特性，使得对Service层和Controller层的测试变的更为简便、灵活。并且Gulp中能调用测试脚本，通过Gulp对JS文件的监听，每次对文件进行修改后都可以自动触发测试脚本，保证代码的质量。
  2. **服务端自动化测试**Node.js同样也是使用mocha进行自动化测试，也是通过断言的方式对模块进行测试。同样的也可以加入Gulp中对JS文件进行监听自动执行测试脚本。

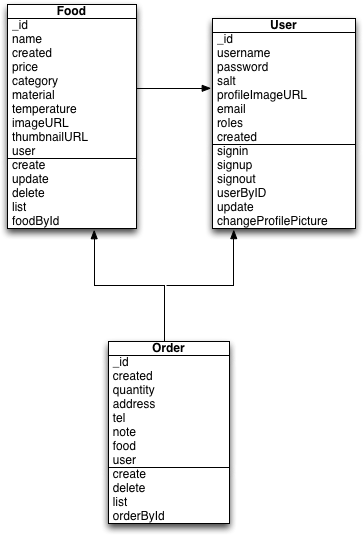
## MSE-FOOD中所使用的调试工具

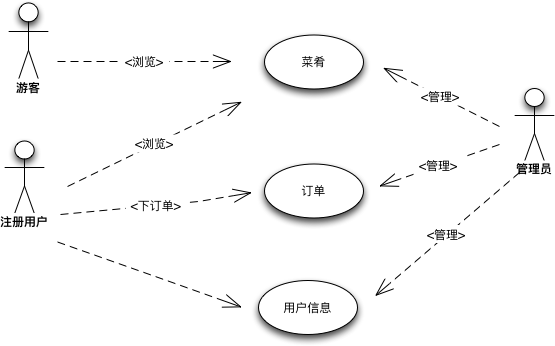
* 1. **前端调试工具**前端主要通过Chrome Dev Tool对JavaScript、CSS进行调试，Chrome Dev Tool可对JavaScript设置断点调试，查看http请求的请求、响应信息。并且可以模拟不同的客户端设备，对响应式布局进行调试。
  2. **Node.JS调试工具**使用Node Inspector工具，可通过Chrome Dev Tool对Node.JS的代码进行运行时的断点调试。可以快速定位错误，查看变量值的变化。

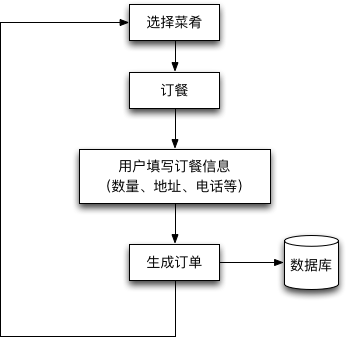
## MSE-FOOD下一版本畅想

* 1. **ES6 + AngularJS 2.0**ECMAScript 6离我们越来越近了，作为它最重要的实现语言，JavaScript也即将迎来语法上的重大变革， ES6将彻底改变编写JavaScript代码的方式。目前已经有许多浏览器已经开始支持部分ES6的特性了，这也意味着全面使用ES6的时代将不会太久。所以学习和使用ES6也是未来学习的重点。  
       
     AngularJS 2.0 也将是一个颠覆式的更新，运行效率将会更高，与ES6的相结合开发效率也会进一步的提高。  
       
     可以断言，ES6 + AngularJS 2.0一定是2016前端技术中讨论最热烈的技术之一。因此，在下一个版本中可以使用这样的组合，能够提高开发效率和应用的运行效率，同时也是一个学习的过程，以后也可以运用到实际工作开发中。
  2. **Node.JS 负载均衡及分布式集群方案**如果随着用户量、并发量的提高，使用目前的单机架构是肯定无法满足持续增长的需求。因此，需要在下阶段中构建一个分布式集群的架构方案。  
       
     Node.JS 作为异步脚本语言，使用事件驱动、 非阻塞I/O模型，非常适合在分布式设备上运行的数据密集型的实时应用。  
       
     Nginx配置负载均衡：  
     在Web服务器中使用Nginx配置反向代理实现负载均衡。  
     RabbitMQ：  
     RabbitMQ是一种应用程序对应用程序的通信方法，它可以支持集群高可用、简单易用、性能出色的特点。使用RabbitMQ中间件服务端实现解耦，实现后端的无缝扩容，并采用Publish/ Subscribe + RPC 实现调用解耦，并利用MQ 统一输入输出。  
       
     Zookeeper:  
     Zookeeper = 文件系统 + 通知机制。主要是用来解决分布式应用中经常遇到的一些数据管理问题，如：统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等。
  3. **使用Redis实现缓存层**在Express中使用Redis缓存和查询数据，可以在高并发的情况下减轻对MongoDB的请求。  
       
     使用Redis对Session进行持久化可解决分布式方案中对Session的共享问题。
  4. **MongoDB分布式集群方案**使用MongoDB分布式集群方案解决单机MongoDB的性能瓶颈，对数据库进行水平扩展，满足数据量很大的场景。 ****
  5. **MongoDB容灾备份方案**如果随着用户日益增加，数据的安全也成为首要考虑到的问题。因此，容灾备份也是下一步所需要考虑的问题。  
     使用MongoDB数据复制，目的是使数据得到最大的可用性,避免单点故障的发生,MongoDB支持在服务器之间进行数据的异步复制,以满足数据的最终一致性。当一台MongoDB服务器出现故障时进行主从切换， 防止数据库停用和数据丢失。

## MSE-FOOD系统设计

* 1. 类图  
     ****

* 1. **用例图   
     **
  2. **订餐流程**

****

* 1. **权限设计  
     游客：查看菜肴  
     注册用户：查看菜肴、添加菜肴、添加订单  
     管理员：查看菜肴、管理菜肴、管理订单、管理用户**

## MSE-FOOD开发流程

* 1. **使用Yomen构建MEAN脚手架**
     1. 安装全局mean.js依赖包：*$ npm install -g generator-meanjs*
     2. **在项目目录下执行：***$ yo meanjs*Yeomen安装向导会要求输入项目名称等信息，并且会自动安装npm和bower的依赖，无需手动执行npm和bower命令。
     3. **安装成后执行:***$ gulp*gulp会自动启动node服务器和mongoDB服务器。
     4. **使用MEAN.JS工具自动添加Food、User、Order模块**分别执行下面的命令，Yomen会自动创建这几个模块，并且已包含这些模块所需要的客户端和服务端的目录以及文件。  
        *$ yo meanjs:crud-module foods* **创建的目录结构如下：  
        **
     5. **自动构建的过程已完成，后面便是编写业务代码了，主要流程如下：**
  + 在Mongoose Schema中添加字段。
  + 在server/routes下添加express路由。
  + 在server/contollers中编写具体业务逻辑。
  + 在server/policies配置用户访问权限。
  + 在client/services中编写$resource服务获取RESTful接口。
  + 在client/config中编写AngularJS的路由。
  + 在client/controller中编写前端的业务逻辑：获取service中的数据或者数据加工。
  + 在client/directives中编写抽象组件或处理DOM操作。
  + 在client/views中编写页面模版。
  + 在client/scss 中编写SASS样式文件。

以上就是MSE-FOOD主要的开发流程，在命令行中运行*gulp*即可运行应用。

## MSE-FOOD配置运行

* 1. **运行环境要求**
  + Node.JS
  + MongoDB
  + Python (编译SASS时需要用到)
  1. **运行步骤：**1）安装npm依赖包：*$ sudo npm install*2）安装bower依赖包：*$ sudo bower install*3) 导入数据库：进入项目根目录，运行命令：*$ mongorestore -d mse-food ./dump/mse-food*  
     3）运行node server: *$ node server.js*或运行gulp: *$ gulp*4）在浏览器中打开 http://localhost:5000

*注：如果使用Gulp运行过程中遇到The `libsass` binding was not found错误，请运行 $ npm uninstall node-sass，然后重新安装: $ npm install node-sass*

## 附录：相关链接

* Node.JS: <https://nodejs.org>
* MongoDB: <https://www.mongodb.org>
* Mongoose: <http://mongoosejs.com>
* Express: <http://expressjs.com>
* MEAN.JS: <http://meanjs.org>
* AngularJS: <https://angularjs.org>
* YEOMAN: <http://yeoman.io>
* Gulp: <http://gulpjs.com>
* Bower: <http://bower.io>
* Bootstrap: <http://getbootstrap.com>
* RabbitMQ: <http://www.rabbitmq.com>
* Redis: <http://redis.io>