<菜宝生鲜>

软件构架文档

版本 <1.3>

小组成员：张逸飞 邓积懋 盛泽宇

郭家骅 张琮昊 徐 涣

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2019.3.30 | 0.0.0 | 创建项目github | shanghaixuhuan |
| 2019.3.30 | 0.0.1 | 删除.idea | shanghaixuhuan |
| 2019.3.30 | 0.0.2 | 张琮昊的需求分析文档 | ECUST-Zhang |
| 2019.3.31 | 0.0.3 | 盛泽宇的测试 | cs-Zeyu |
| 2019.4.1 | 0.1.0 | 架构日常更新4.1\_1 | shanghaixuhuan |
| 2019.4.1 | 0.1.1 | 架构日常更新4.1\_2 | shanghaixuhuan |
| 2019.4.2 | 0.1.2 | 架构日常更新4.2\_1 | shanghaixuhuan |
| 2019.4.2 | 0.1.3 | 架构日常更新4.2\_2 | shanghaixuhuan |
| 2019.4.2 | 0.1.4 | 架构日常更新4.2\_3 | shanghaixuhuan |
| 2019.4.2 | 0.1.5 | 架构日常更新4.2\_4 | shanghaixuhuan |
| 2019.4.2 | 0.1.6 | 架构日常更新4.2\_5 | shanghaixuhuan |
| 2019.4.2 | 0.1.7 | 架构日常更新4.2\_6 | shanghaixuhuan |
| 2019.4.2 | 0.1.8 | 架构日常更新4.2\_7 | shanghaixuhuan |
| 2019.4.3 | 0.2.0 | 邓积懋前端界面的初步上线 | DJM199708 |
| 2019.4.3 | 0.2.1 | 盛泽宇后端更新\_4.3\_1 | cs-Zeyu |
| 2019.4.3 | 0.2.2 | 盛泽宇后端更新\_4.3.2 | cs-Zeyu |
| 2019.4.4 | 0.2.3 | 前后端分离 | Shanghaixuhuan |
| 2019.4.4 | 0.3.0 | 邓积懋前端更新\_4.3\_1 | DJM199708 |
| 2019.4.6 | 0.3.1 | 创建Github.io开发版本 | shanghaixuhuan |
| 2019.4.6 | 0.3.2 | 架构日常更新4.6\_1 | shanghaixuhuan |
| 2019.5.8 | 0.4.0 | 邓积懋前端更新\_4.8\_1 | DJM199708 |
| 2019.5.8 | 0.4.1 | 盛泽宇后端更新\_4.8\_1 | cs-Zeyu |
| 2019.5.9 | 0.5.0 | 邓积懋前端更新\_5.9\_1 | DJM199708 |
| 2019.5.9 | 0.5.1 | 邓积懋前端更新\_5.9\_2 | DJM199708 |
| 2019.5.9 | 0.5.2 | 架构日常更新5.9\_1 | Shanghaixuhuan |
| 2019.5.9 | 0.5.3 | 架构日常更新5.9\_2 | shanghaixuhuan |
| 2019.5.16 | 0.6.0 | 邓积懋前端更新\_5.16\_1 | DJM199708 |
| 2019.5.16 | 0.6.1 | 盛泽宇后端更新\_5.16\_1 | cs-Zeyu |
| 2019.5.16 | 0.6.1 | 张琮昊添加后台管理界面 | ECUST-Zhang |
| 2019.5.16 | 0.6.2 | 郭家骅修改说明 | JamesGuoo |
| 2019.5.16 | 0.6.3 | 项目经理张逸飞验收阶段阶段性工作 | TimeShield |
| 2019.5.21 | 1.0.0 | 第一代产品上线 | Shanghaixuhuan |
| 2019.5.29 | 1.0.0 | 软件架构文档 | TimeShield |

**目录**

1 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2 构架表示方式 5

3 构架目标和约束 5

4 用例视图 5

4.1 用例实现 6

5逻辑视图 7

5.1 概述 7

5.2 在构架方面具有重要意义的设计包 8

6 部署视图 9

7 实施视图 9

7.1 概述 9

7.2 层 10

8 数据视图 11

9 接口 11

9.1 用户登录接口 11

9.2 用户注册接口 12

9.3 获取商品列表 13

9.4 用户下单 14

9.5 获取产品总数 15

9.6 商品列表搜索 15

9.7 获取指定类别商品接口 17

10 数据库 18

10.1 用户表 user 18

10.2 商品表 item 19

10.3 订单表 vorder 19

10.4 订单详情表 orderdetail 19

11 大小和性能 19

12 质量 20

软件构架文档

# 1 简介

菜宝生鲜开发团队将菜宝生鲜平台开发过程中的软件架构编档成为文档，通过简介、构架表示方式、构架目标和约束、用例视图、逻辑视图、进程视图、部署视图、实施视图、数据视图、质量等方面对菜宝生鲜平台的软件架构进行详细的说明。

## 1.1 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

菜宝生鲜开发团队希望将软件架构作为项目关系人之间交流的平台。软件系统的项目关系人分别关注系统的不同特性，而这些特性都由架构所决定，因此，架构提供了一个共同语言（公共的参考点），项目关系人以此作为彼此理解、协商、达成共识或相互沟通的基础。架构分析既依赖于又促进了这个层次上的交流。

在开发过程中，菜宝生鲜开发团队发现软件架构给予整个开发过程的意义极大。架构明确了对系统实现的约束条件，影响着系统的质量属性，可以用来预测系统的质量，有助于原型开发，并且为维护的决策提供了根据。

菜宝生鲜开发团队希望能够通过软件构架文档的设计，作为系统的总体设计，它指导后续的详细设计和编码。架构使基于模板的开发成为可能，有利于开发的规范化和一致性，减少开发与维护成本。

## 1.2 范围

架构文档的使用者是菜宝生鲜平台的项目关系人，技术文档从读者的角度来编写。

架构的主要用途是充当菜宝生鲜平台项目关系人之间进行交流的工具，文档则促进了这种交流—— 架构项目关系人希望从架构文档中获得自己所关心的架构信息，如：系统实现人员希望文档提供关于开发活动的不能违反的限制及可利用的自由；测试人员和集成人员希望能从文档中得到必须组合在一起的各部分，并以此得到一个正确的测试黑箱；项目经理希望根据所确定的工作任务组建开发小组，规划和分配项目资源。

## 1.3 定义、首字母缩写词和缩略语

|  |  |  |
| --- | --- | --- |
| 简称 | 全称 | 详情 |
| AJAX | Asynchronous Javascript And XML | 一种创建交互式网页应用的网页开发技术 |

## 1.4 参考资料

《Koa框架教程》阮一峰

《软件架构设计》温昱

《了不起的Node.js》Guillermo Spcript

## 1.5 概述

软件构架文档分为表示方式、构架目标和约束、用例视图、逻辑视图、进程视图、部署视图、实施视图、数据视图、质量其中用例视图中包含用例实现，逻辑视图中包含在构架方面具有重要意义的设计包，实施视图中包含层。

# 2 构架表示方式

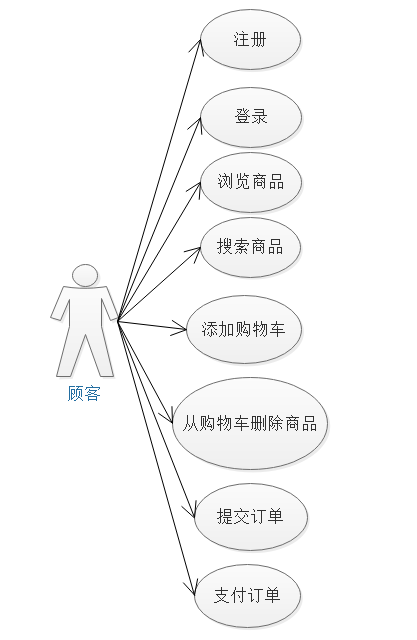
文档将通过文字介绍和图例展示对构架进行说明，之后具体会从用例视图、逻辑视图、进程视图、部署视图和实施视图中列出必需的那些视图，并分别说明这些视图包含哪些类型的模型元素。

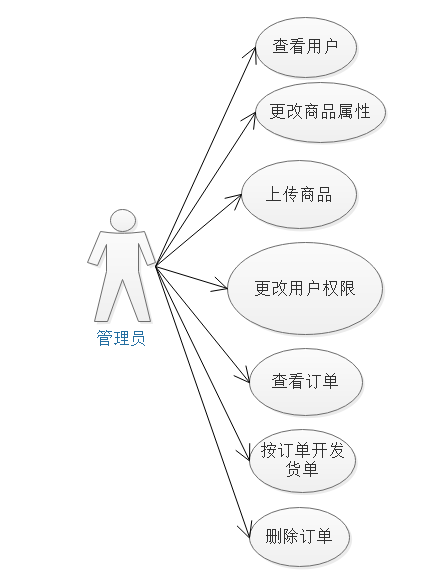
# 3 构架目标和约束

构架目标在于充当菜宝生鲜平台项目关系人之间进行交流的工具，文档则促进了这种交流—— 架构项目关系人希望从架构文档中获得自己所关心的架构信息。

构架约束方面，前端框架使用bootstrap，前端逻辑处理技术使用jquery.js，网页缓冲使用localstorage，前端使用ajax进行数据库的请求。后端方面，使用node.js作为后端框架，使用koa作为web框架。服务器部署方面，使用腾讯云服务器，并使用nginx作为web服务器。数据库选择mysql。使用git对版本进行管理。开发工具方面，前端使用Hbuilder，后端使用vscode，服务器操作系统为centos。编码规范方面，项目组参照《华为代码规范》。

# 4 用例视图





## 4.1 用例实现

登录功能：在mysql中创建用户表，用户登录时在前端表单中填写账号和密码，然后在提交表单之前，会先用javascript写的方法对表单元素进行验证，使其具有正确的格式；当格式正确时，才会把表单元素里的标签提交到node.js。node.js调用数据库中用户表，对登录进行响应，如果输入的密码与从数据库中返回的密码相同，那么转到登录成功后的界面，如果登录失败，那么则提示用户重新登录。

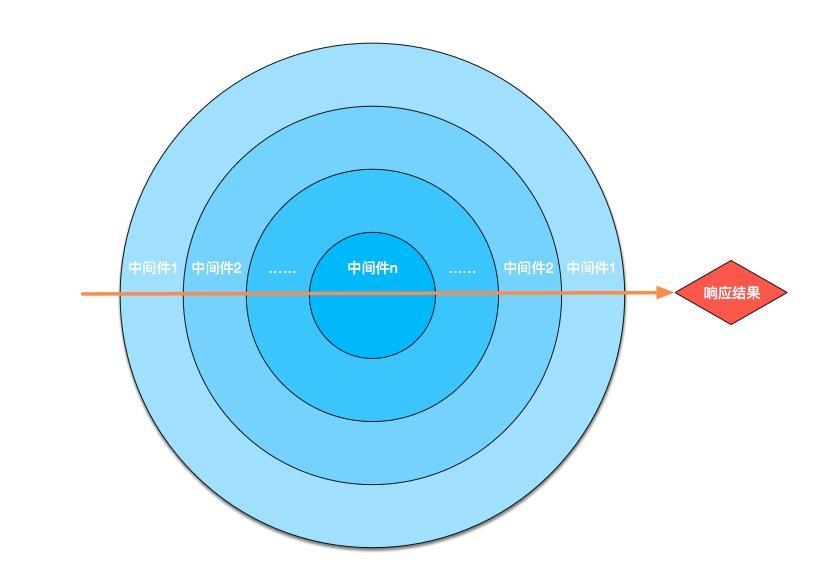
添加购物车功能：将需要加入购物车的商品信息保存到LocalStorage中。包括商品的名称、价格以及添加时间等各类信息，并且购物车上面的显示商品个数的图标内容也随之改变。当登录账号后，进入个人购物车，购物车内商品信息从LocalStorage中获取。在完成提交订单之后，将本地LocalStorage内保存的信息清除掉。

提交订单功能：用户点击提交订单后，使用ajax请求将前端localtorage中购物车的数据插入至数据库中订单表中。

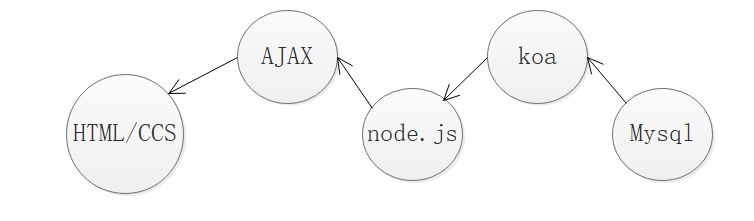
# 5逻辑视图

## 5.1 概述

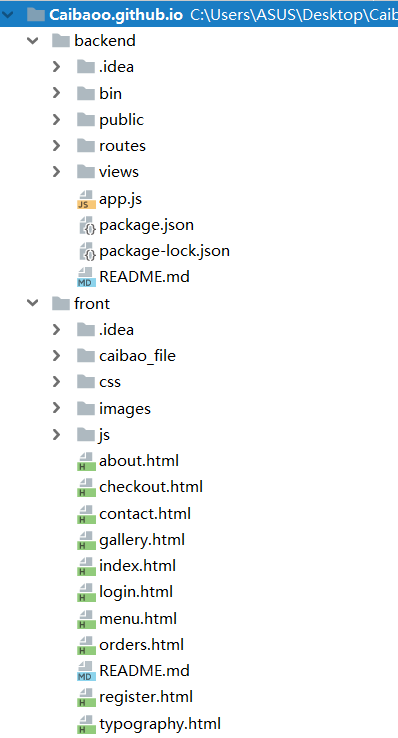
菜宝生鲜平台使用基于node.js平台的web开发框架Koa，Koa 是一个新的 web 框架， 致力于成为 web 应用和 API 开发领域中的一个更小、更富有表现力、更健壮的基石。 通过利用 async 函数，Koa 能够丢弃回调函数，并有力地增强错误处理。 Koa 并没有捆绑任何中间件， 而是提供了一套优雅的方法，因此能够快速而愉快地编写服务端应用程序。



各模块调用关系如下：



## 5.2 在构架方面具有重要意义的设计包



bin：存放项目启动的执行文件。  
 config：项目的配置文件，json格式，对应开发、测试到正式环境的配置文件各一份，内容包括环境名称、redis、es、mysql、上传的oss-path、域名等信息。  
 node\_modules：项目需要依赖的模块，在该目录下执行npm install安装项目需要的模块。  
 public：静态资源文件夹。  
 routes：路由文件，相当于mvc中的controller，默认创建的express项目包括index.js和user.js。  
 views：视图文件，相当于mvc中的view。  
 package.json：项目的依赖配置与版本信息。

# 6 部署视图

使用nginx作为Web服务器， Nginx是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。占有内存少，并发能力强，事实上nginx的并发能力确实在同类型的网页服务器中表现较好。

# 7 实施视图



## 7.1 概述

表现层是用户和系统之间交流的桥梁，它一方面为用户提供了交互的工具，另一方面也为显示和提交数据实现了一定的逻辑，以便协调用户和系统的操作。

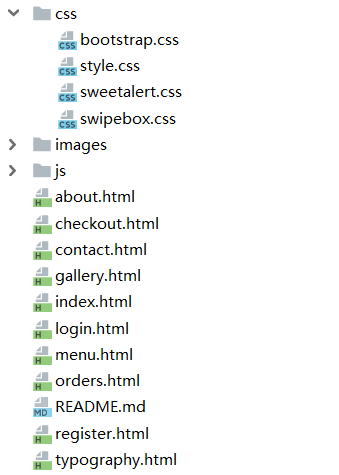
服务层就是相当于中间类的作用，中间的工厂类提供了另一个通用放任接口让调用者可以使用接口暴漏的方法，而无需关注架构或底层发生的怎样的变化，服务层的原理和这个非常类似，只不过它将工厂模式应用到更高层面的抽象之上。在服务层上项目使用AJAX，AJAX是指一种创建交互式网页应用的网页开发技术。通过在后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

业务层中的模块包含了系统所需要的所有功能上的算法和计算过程，并与数据访问层和表现层交互。抽象的说，业务逻辑层就是处理与业务相关的部分，业务层包含一系列的执行与数据的操作。在业务层上我们使用了Node.js，是一个让 JavaScript 运行在服务端的开发平台,用于从服务器的数据库中取出数据并对其进行操作。

数据层是一个代码类库，提供访问位于持久化容器中数据的功能，在分层设计中，所有从介质化读取数据或写入数据的工作都属于这一层的任务。我们在阿里云服务器中的MySQL中存放项目相关的数据。

## 7.2 层

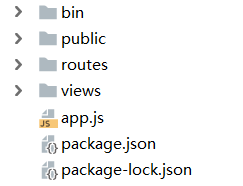
1.表示层



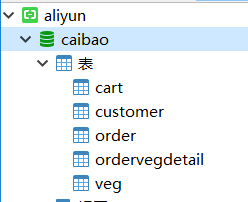
2.服务层



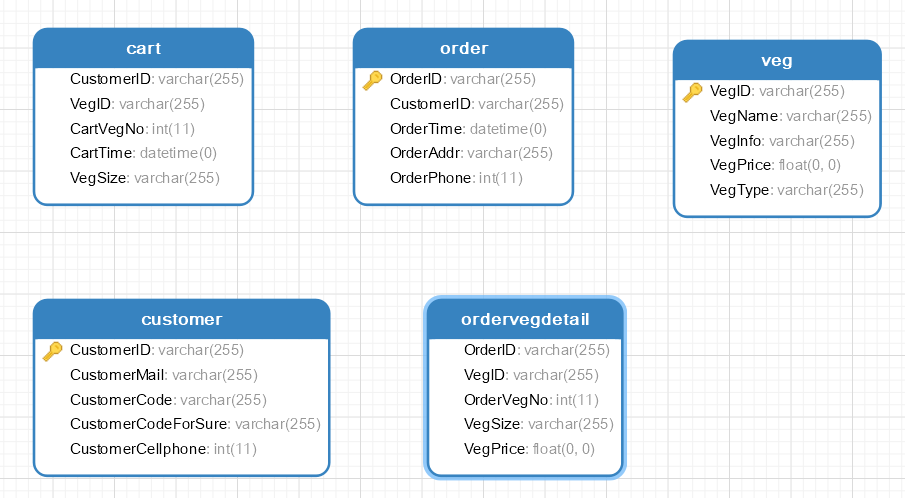
3.业务层



4.数据层



# 8 数据视图



# 9 接口

## 9.1 用户登录接口

**请求**

| **请求URL** | **请求方式** |
| --- | --- |
| http://101.132.184.238:81/v0/user/logincheck | POST |

**参数**

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| userid | 是 | string | 用户名 |
| userpass | 是 | string | 密码 |

**返回示例**

1. {
2. "message": "success"，
3. "userdata": {
4. "UserName": "邓积懋",
5. "UserPhone": "12121212"，
6. "UserEmail": "12345@qq.com",
7. }
8. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| message | string | 登陆成功与否 |
| userdata | 数组 | 用户个人信息 |

**备注**

“message”: “iderror” 用户id不存在  
“message”: “passerror” 用户密码错误

## 9.2 用户注册接口

**请求**

| **请求URL** | **请求方式** |
| --- | --- |
| http://101.132.184.238:81/v0/user/register | POST |

**参数**

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| userid | 是 | string | 用户名 |
| userpass | 是 | string | 密码 |
| username | 否 | string | 昵称 |
| userphone | 是 | string | 电话 |

**返回示例**

1. {
2. message:"error"
3. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| message | string | 注册成功与否 |

## 9.3 获取商品列表

**请求**

| **请求URL** | | | **请求方式** | |
| --- | --- | --- | --- | --- |
| http://101.132.184.238:81/v0/itemlist | | | GET | |
|  |  |  | |  | |

**返回示例**

1. {
2. "data": [
3. {
4. "VegID": "v190001",
5. "VegName": "小青菜",
6. "VegInfo": "菜菜菜",
7. "VegPrice": 2.88,
8. "VegType": "蔬菜"
9. },
10. {
11. "VegID": "v190002",
12. "VegName": "黄瓜",
13. "VegInfo": "黄瓜",
14. "VegPrice": 3.88,
15. "VegType": "蔬菜"
16. },
17. {
18. "VegID": "v190003",
19. "VegName": "西红柿",
20. "VegInfo": "西红柿",
21. "VegPrice": 4.99,
22. "VegType": "蔬菜"
23. }
24. ]
25. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| vegID | string | id |
| VegName | string | 商品名 |
| VegInfo | string | 图片地址 |
| VegPrice | string | 商品价格 |

## 9.4 用户下单

**请求**

| **请求URL** | **请求方式** |
| --- | --- |
| http://101.132.184.238:81/v0/order | POST |

**参数**

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| userid | 是 | string | 用户id |
| veglist | 是 | 数组 | 包含商品信息 |
| totalprice | 是 | float | 订单金额 |

**返回示例**

1. {
2. message:"error"
3. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| message | string | 成功与否 |

## 9.5 获取产品总数

**请求**

| **请求URL** | **请求方式** |
| --- | --- |
| http://101.132.184.238:81/v0/itemnum | GET |

**返回示例**

1. {
2. "sum":1352
3. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| sum | int | 产品总数 |

## 9.6 商品列表搜索

**请求**

| **请求URL** | **请求方式** |
| --- | --- |
| http://101.132.184.238:81/v0/func/search | POST |

**参数**

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| key | 是 | string | 关键字 |

**返回示例**

1. {
2. "message": "success",
3. "data": [
4. {
5. "ItemID": "\nv190005",
6. "ItemName": "土豆",
7. "ItemURL": "肯德基特供薯条",
8. "ItemPrice": 5.6,
9. "ItemType": "蔬菜",
10. "Stock": 34
11. },
12. {
13. "ItemID": "\nv190008",
14. "ItemName": "时令蔬菜",
15. "ItemURL": "当季的新鲜蔬菜",
16. "ItemPrice": 7.1,
17. "ItemType": "蔬菜",
18. "Stock": 454
19. },
20. {
21. "ItemID": "\nv190013",
22. "ItemName": "菠菜",
23. "ItemURL": "大力水手的最爱",
24. "ItemPrice": 6.5,
25. "ItemType": "蔬菜",
26. "Stock": 36
27. },
28. {
29. "ItemID": "\nv190014",
30. "ItemName": "什锦蔬菜",
31. "ItemURL": "蔬蔬搭配一样价 喜欢吃啥就选啥",
32. "ItemPrice": 7.9,
33. "ItemType": "蔬菜",
34. "Stock": 25
35. }
36. ]
37. }
38. 错误示例：
39. {
40. "message": "error",
41. "tips": "no data qualified!"
42. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| message | string | 是否成功返回搜索项 |
| tips | string | 标注信息 |
| data | 数组 | 返回的数组 |

## 9.7 获取指定类别商品接口

**请求**

| **请求URL** | **请求方式** |
| --- | --- |
| http://101.132.184.238:81/v0/func/getType | POST |

**参数**

| **参数名** | **必选** | **类型** | **说明** |
| --- | --- | --- | --- |
| type | 是 | string | 商品种类 |

**返回示例**

1. {
2. "message": "success",
3. "data": [
4. {
5. "ItemID": "\nv190005",
6. "ItemName": "土豆",
7. "ItemURL": "肯德基特供薯条",
8. "ItemPrice": 5.6,
9. "ItemType": "蔬菜",
10. "Stock": 34
11. },
12. {
13. "ItemID": "\nv190008",
14. "ItemName": "时令蔬菜",
15. "ItemURL": "当季的新鲜蔬菜",
16. "ItemPrice": 7.1,
17. "ItemType": "蔬菜",
18. "Stock": 454
19. },
20. {
21. "ItemID": "\nv190013",
22. "ItemName": "菠菜",
23. "ItemURL": "大力水手的最爱",
24. "ItemPrice": 6.5,
25. "ItemType": "蔬菜",
26. "Stock": 36
27. },
28. {
29. "ItemID": "\nv190014",
30. "ItemName": "什锦蔬菜",
31. "ItemURL": "蔬蔬搭配一样价 喜欢吃啥就选啥",
32. "ItemPrice": 7.9,
33. "ItemType": "蔬菜",
34. "Stock": 25
35. }
36. ]
37. }

**返回参数说明**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| message | string | 是否取到指定数据 |
| data | 数组 | 包含指定数据的数组 |

# 10 数据库

## 10.1 用户表 user

| **参数名** | **类型** | **长度** | **是否能为空** | **是否为主键** |
| --- | --- | --- | --- | --- |
| UserID | varchar | 255 | × | √ |
| UserName | varchar | 255 | × |  |
| UserPass | varchar | 255 | × |  |
| UserPhone | varchar | 12 | √ |  |

## 10.2 商品表 item

| **参数名** | **类型** | **长度** | **是否能为空** | **是否为主键** |
| --- | --- | --- | --- | --- |
| ItemID | varchar | 255 | × | √ |
| ItemName | varchar | 255 | × |  |
| ItemUrl | varchar | 255 | × |  |
| ItemPrice | float |  | × |  |
| ItemType | varchar | 255 | × |  |
| Stock | int | 255 | √ |  |

## 10.3 订单表 vorder

| **参数名** | **类型** | **长度** | **是否能为空** | **是否为主键** |
| --- | --- | --- | --- | --- |
| OrderID | varchar | 255 | × | √ |
| CustomerID | varchar | 255 | × |  |
| OrderMoney | decimal | 10 | × |  |
| OrderTime | datetime |  | × |  |
| OrderAddr | varchar | 255 | √ |  |
| OrderPhone | int | 11 | √ |  |

## 10.4 订单详情表 orderdetail

| **参数名** | **类型** | **长度** | **是否能为空** | **是否为主键** |
| --- | --- | --- | --- | --- |
| OrderID | varchar | 255 | × | √ |
| CustomerID | varchar | 255 | × |  |
| ItemID | varchar | 255 | × |  |
| CartTime | datetime |  | × |  |
| ItemSize | varchar | 255 | √ |  |

# 11 大小和性能

前端使用AJAX，仅向服务器发送并取回必需的数据，它使用SOAP或其它一些 基于XML的Web Service接口，并在客户端采用JavaScript处理来自服务器的响应。因为在服务器和浏览器之间交换的数据大量减少，结果我们就能看到响应更快的应用。同时很多的处理工作可以在发出请求的客户端机器上完成，所以Web服务器的处理时间也减少了。

后端处理使用node.js，以事件驱动为核心，事件驱动在于异步回调，他提供的大多数api都是基于事件的、异步的风格。而事件驱动的优势在于充分利用系统资源，执行代码无须阻塞等待某种操作完成，有限的资源用于其他任务。事件驱动机制是通过内部单线程高效率地维护事件循环队列来实现的，没有多线程的资源占用和上下文的切换。

# 12 质量

使用jquery.js用于前端逻辑处理，jquery.js具有较强的可扩展性，可以链式调用，通过对某个类的扩展写法来实现。Jquery.js拥有较多插件，用户能方便地处理HTMLdocuments、events、实现动画效果，并且方便地为网站提供AJAX交互。

使用nginx作为web服务器，占用更少的内存及资源 ，支持更多的并发连接，体现更高的效率，抗并发，nginx 处理请求是异步非阻塞的，负载均衡能力突出，Nginx 既可以在内部直接支持 Rails 和 PHP 程序对外进行服务, 也可以支持作为 HTTP代理 服务器对外进行服务. Nginx采用C进行编写, 不论是系统资源开销还是CPU使用效率都比 Perlbal 要好很多。