



传智播客.黑马程序员

第9章 项目实战——在线商城

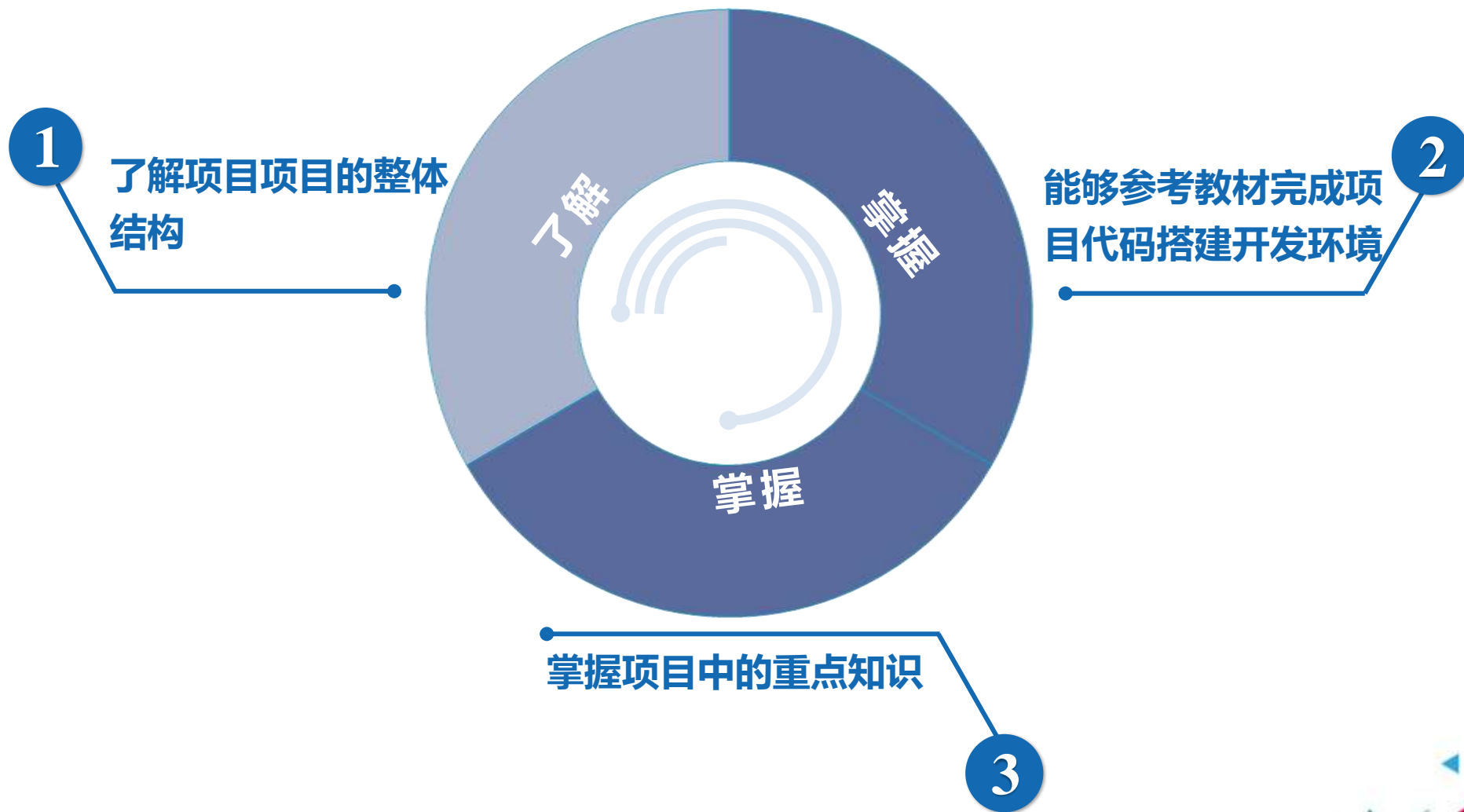
jQuery

- 技术方案
- 接口分析
- 开发环境搭建
- 功能实现




学习目标

传智播客·黑马程序员
改变中国IT教育 我们正在行动





目录

 传智播客.黑马程序员
改变中国IT教育 我们正在行动

9.1

项目简介

 [点击查看本小节知识架构](#)

9.2

搭建开发环境

 [点击查看本小节知识架构](#)


9.3

管理员登录

 [点击查看本小节知识架构](#)



目录

 传智播客. 黑马程序员
改变中国IT教育 我们正在行动

9.4

后台界面管理

 [点击查看本小节知识架构](#)

9.5

商品分类管理

 [点击查看本小节知识架构](#)


9.6

商品管理

 [点击查看本小节知识架构](#)



目录

 传智播客.黑马程序员
改变中国IT教育 我们正在行动

9.7

商品图片管理

 [点击查看本小节知识架构](#)

9.8

商城首页

 [点击查看本小节知识架构](#)


9.9

商品列表

 [点击查看本小节知识架构](#)



目录

 传智播客.黑马程序员
改变中国IT教育 我们正在行动

9.10

商品详情

 [点击查看本小节知识架构](#)

9.11

购物车

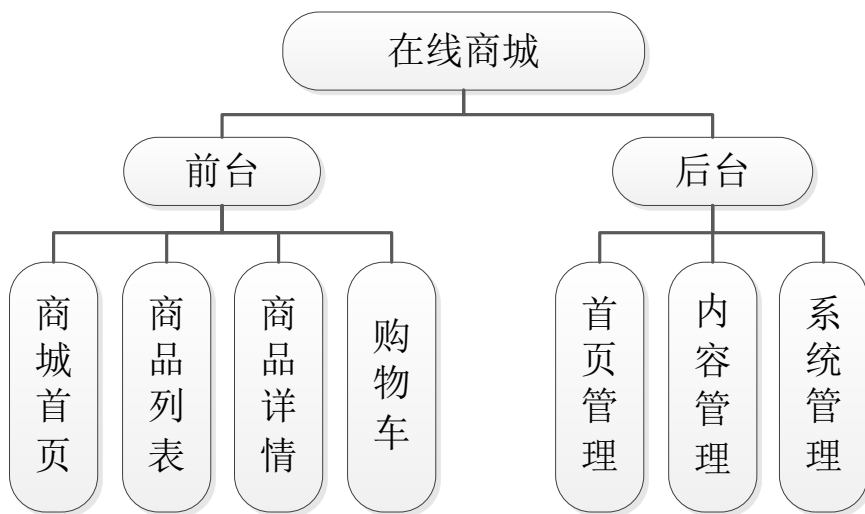
 [点击查看本小节知识架构](#)



9.1 项目简介

1. 项目展示

本项目的前台包括商城首页、商品列表、商品详情和购物车功能，后台包括首页管理、内容管理和系统管理功能。



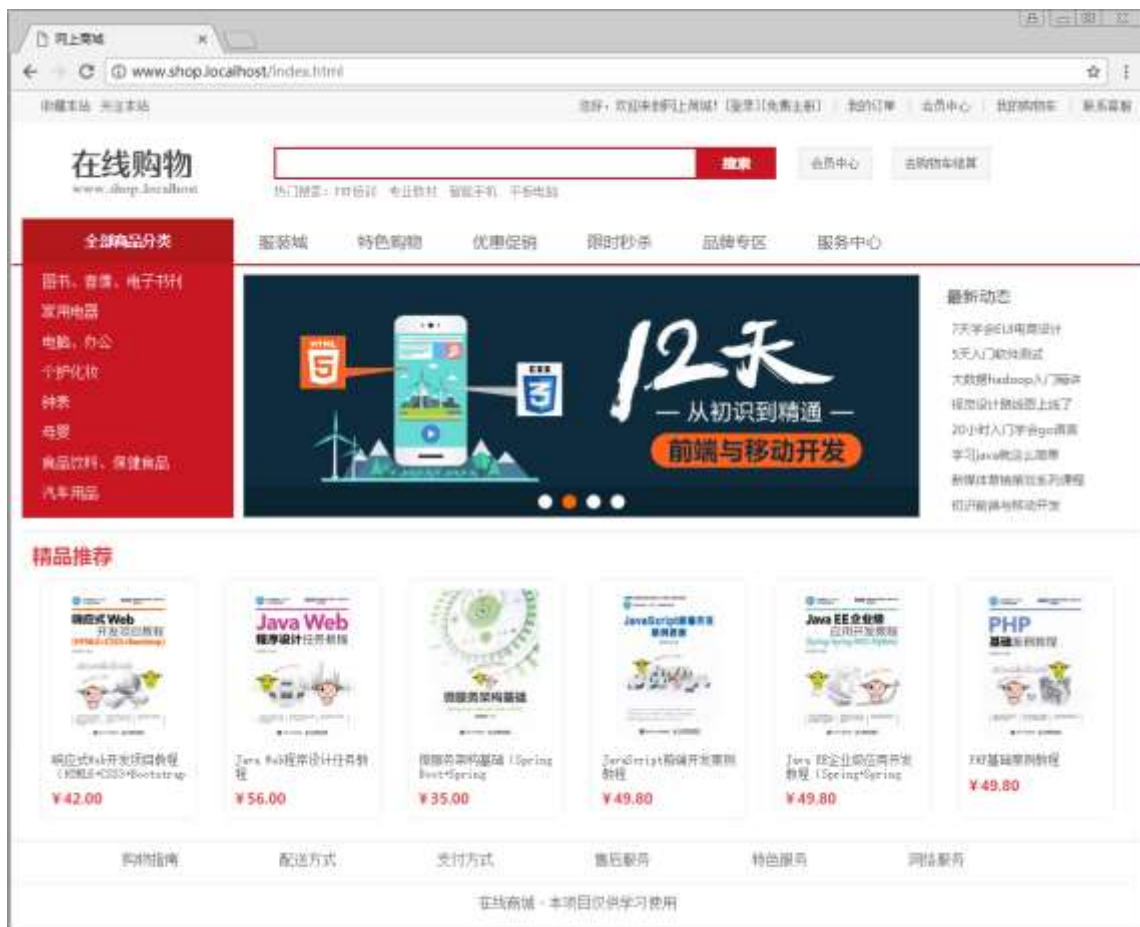


9.1 项目简介

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 项目展示

前台页面:



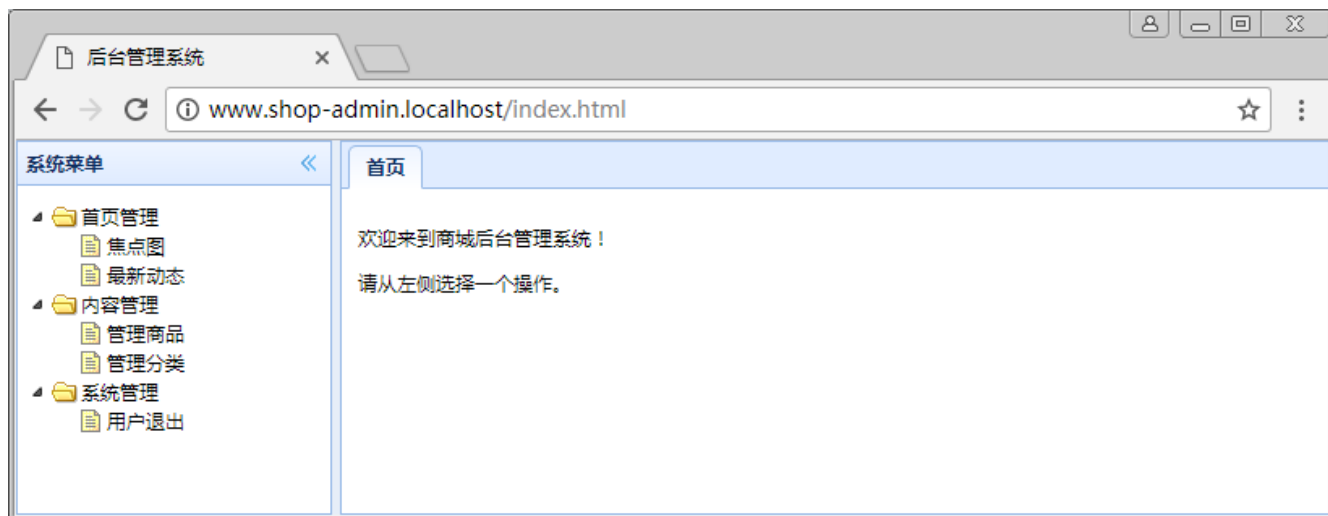


9.1 项目简介

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 项目展示

后台页面：





9.1 项目简介

2. 技术方案

前端方案:

jQuery: 简化JavaScript代码，提高开发效率。

jQuery EasyUI: 应用在后台管理系统中快速构建用户界面。

WebUploader: 用于商品图片的上传，支持图片预览、显示上传进度等功能。

UEditor: 应用在发布商品时编辑商品详情，提供了方便的文本格式处理。

art-template: 应用在前台进行数据展示时，将从服务器获取的数据填写到模板中渲染输出。



9.1 项目简介

2. 技术方案

后端方案:

Apache: 用于提供Web服务器的基础功能。

MySQL: 用于提供数据库服务器功能。

PHP: 负责处理Apache的动态请求，并与MySQL服务器进行交互。

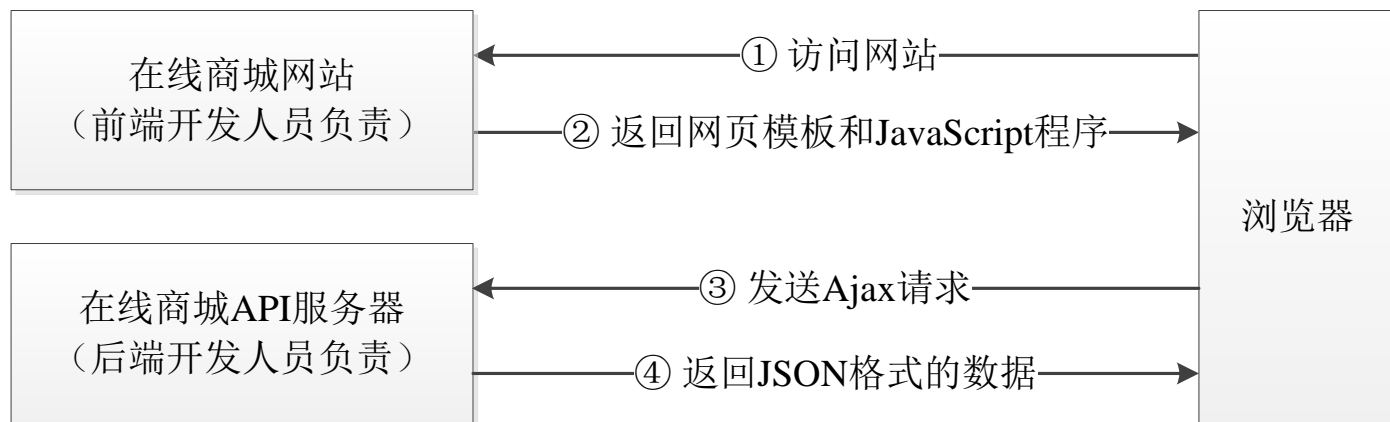


9.1 项目简介

2. 技术方案

前后端方案：采用API的方式进行前后端数据交互：

数据是通过Ajax请求API服务器获得。





9.2 搭建开发环境

1. 配置虚拟主机

站点的URL地址:

前台网站: <http://www.shop.localhost/>

前台API: <http://api.shop.localhost/>

后台网站: <http://www.shop-admin.localhost/>

后台API: <http://api.shop-admin.localhost/>



9.2 搭建开发环境

2. 项目部署

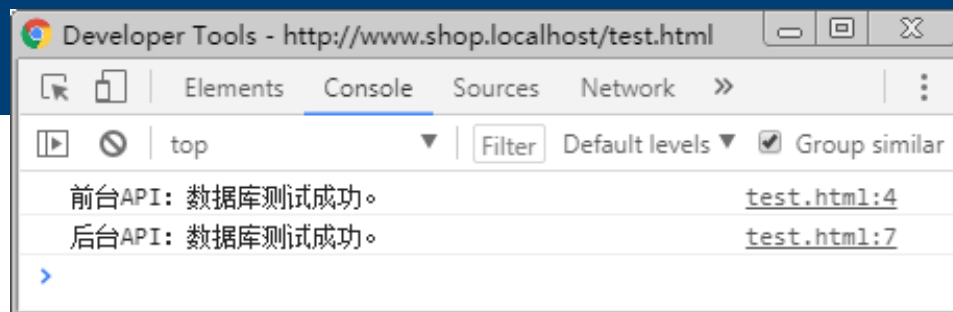
目录	作用
chapter09\data	数据库文件
chapter09\www.shop.localhost	前台网站相关素材
chapter09\api.shop.localhost	前台API服务器文件
chapter09\www.shop-admin.localhost	后台网站相关素材
chapter09\api.shop-admin.localhost	后台API服务器文件



9.2 搭建开发环境

3. 接口测试

```
$.get('http://api.shop.localhost', function(data) {  
    console.log(data);  
});  
$.get('http://api.shop-admin.localhost', function(data) {  
    console.log(data);  
});
```





9.2 搭建开发环境

4. 目录结构

[www.shop-admin.localhost](#)目录结构：

类型	文件名称	作用
文件	login.html	用户登录页面
	index.html	后台布局页面（不含内容区域）
目录	content	保存内容区域的页面文件
文件	content\category-list.html	商品分类列表页面
	content\category-add.html	商品分类添加页面
	content\item-list.html	商品列表页面
	content\item-edit.html	商品添加和修改页面
	content\home-slide.html	商城首页的焦点图管理页面
	content\home-news.html	商城首页的最新动态管理页面



9.2 搭建开发环境

4. 目录结构

[www.shop-admin.localhost](#)目录结构:

类型	文件名称	作用
目录	js	保存js文件
目录	js\jquery-easyui-1.5.4.2	jQuery EasyUI用户界面库
	js\webuploader-0.1.5	WebUploader文件上传组件
	js\ueditor1.4.3.3	UEditor在线编辑器
文件	js\jquery-1.12.4.min.js	jQuery库
	js\config.js	后台配置文件
	js\auth.js	后台判断用户是否登录的文件



9.2 搭建开发环境

4. 目录结构

[www.shop.localhost](#)目录结构:

类型	文件名称	作用
文件	.htaccess	Apache分布式配置文件（用于URL重写）
	index.html	前台布局页面（不含内容区域）
目录	content	保存内容区域的页面文件
文件	content\index.html	商城首页
	content\find.html	商品列表页
	content\item.html	商品查看页
	content\cart.html	购物车



9.2 搭建开发环境

4. 目录结构

www.shop.localhost目录结构:

类型	文件名称	作用
目录	upload	保存上传文件
	css	保存css文件
	img	保存图片文件
	js	保存js文件和前端库



9.2 搭建开发环境

4. 目录结构

[www.shop.localhost](#)目录结构:

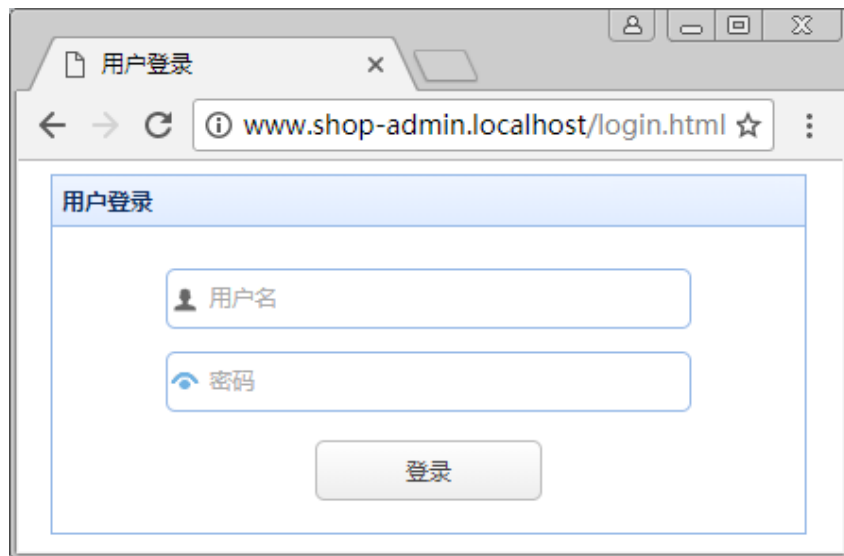
类型	文件名称	作用
目录	js\art-template-4.12.1	art-template模板引擎
文件	js\jquery-1.12.4.min.js	jQuery库
	js\config.js	前台配置文件
	js\common.js	前台公共js文件
	js\jquery.slide.js	jQuery焦点图插件
	js\jquery.page.js	jQuery分页导航插件
	js\jquery.album.js	jQuery商品相册插件



9.3 管理员登录

1. 任务描述

管理员登录功能：需要输入正确的用户名和密码，才能够登录网站后台进行相关操作。





9.3 管理员登录

2. 接口分析

开发管理员登录功能，后台API接口设计：

URL	http://api.shop-admin.localhost/v1/auth	
基本信息	请求方式	内容类型
	POST	application/x-www-form-urlencoded
主体	参数名	说明
	username	用户名
	password	密码



9.3 管理员登录

2. 接口分析

请求API:

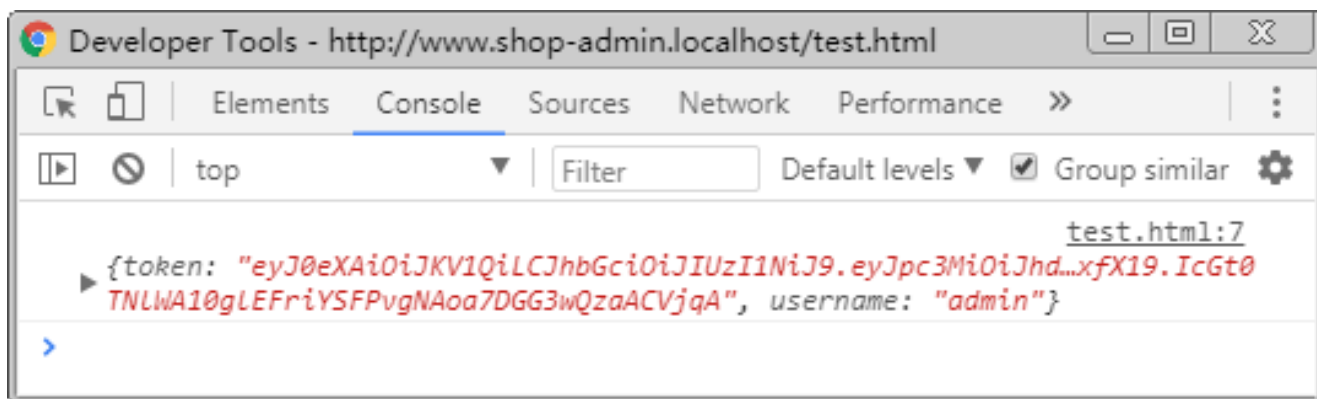
```
var url = 'http://api.shop-admin.localhost/v1/auth';  
var data = {username: 'admin', password: '123456'};  
$.post(url, data, function(data) {  
    console.log(data);  
}).fail(function(xhr) {  
    console.log(xhr.responseText);  
});
```



9.3 管理员登录

2. 接口分析

认证成功:

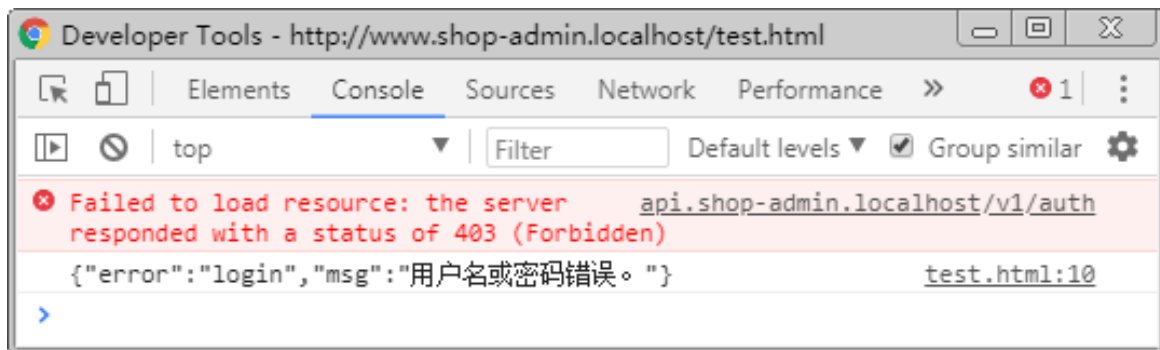




9.3 管理员登录

2. 接口分析

认证失败:





9.3 管理员登录

3. 代码实现

准备页面：

```
<form id="login_form">
  <p><input class="easyui-textbox" data-options="prompt:'用户名',
    iconCls:'icon-man',iconAlign:'left',width:'100%',height:'32px'"
    name="username"></p>
  <p><input class="easyui-passwordbox" data-options="prompt:'密码',
    iconAlign:'left',width:'100%',height:'32px'" name="password"></p>
</form>
<a id="login_btn" href="#" class="easyui-linkbutton"
  data-options="width:'45%',height:'32px'">登录</a>
```



9.3 管理员登录

3. 代码实现

提交表单:

```
$('#login_btn').click(function() {  
    $.post(Config.api + 'auth', $('#login_form').serialize())  
    .done(function(data) {  
        localStorage.setItem(Config.tokenName, 'Bearer ' + data.token);  
        location.href = 'index.html';  
    })  
    .fail(function(xhr) {  
        $.messager.alert('登录失败', JSON.parse(xhr.responseText).msg);  
    });  
});
```



9.3 管理员登录

3. 代码实现

[js/config.js](#)

```
var Config = {  
  api: 'http://api.shop-admin.localhost/v1/',  
  tokenName: 'shop_admin_auth_token',  
};
```



9.4 后台管理界面

1. 任务描述

后台管理界面：

- 首页管理
- 内容管理
- 系统管理



9.4 后台管理界面

2. 接口分析

URL	http://api.shop-admin.localhost/v1/test	
基本信息	请求方式	-
	GET	-



9.4 后台管理界面

2. 接口分析

API判断用户已经登录的依据为：

在当前的HTTP请求消息中收到了名称为Authorization的请求头字段，且该字段中保存了正确的令牌。



9.4 后台管理界面

2. 接口分析

如果API认证失败，则返回的错误信息有如下几种：

- ❑ 未收到Authorization请求头；
- ❑ Authorization请求头格式有误；
- ❑ token认证失败。



9.4 后台管理界面

2. 接口分析

如果API认证失败，则返回的错误信息有如下几种：

```
var token = localStorage.getItem('shop_admin_auth_token');  
$.ajax({  
  type: 'GET', headers: {Authorization: token},  
  url: 'http://api.shop-admin.localhost/v1/test',  
  error: function(xhr) {  
    console.log(xhr.responseText);  
  }  
});
```

将令牌放在请求头中的Authorization字段中发送，API收到后就会检测令牌是否正确



9.4 后台管理界面

3. 代码实现——判断用户是否已经登录

准备页面：

引入config.js文件

```
<script src="js/config.js"></script>
```



9.4 后台管理界面

3. 代码实现——判断用户是否已经登录

准备页面：

引入auth.js文件

```
<script src="js/auth.js"></script>
```



9.4 后台管理界面

3. 代码实现——判断用户是否已经登录

js/auth.js文件:

```
// 从本地存储中获取令牌  
var token = localStorage.getItem(Config.tokenName);  
Config.token = token;  
// 发送Ajax请求时，在请求头中携带令牌用于认证  
$.ajaxSetup({headers: {Authorization: token}});  
$.get(Config.api).fail(function(xhr) { // 测试令牌是否正确  
    var data = JSON.parse(xhr.responseText);  
    if (data.error === 'auth') {alert(data.msg + '。 \n\n请重新登录！ \n');  
        location.href = 'login.html';}  
});
```

获取的令牌保存到Config.token中

配置了Ajax的全局默认选项



9.4 后台管理界面

3. 代码实现——判断用户是否已经登录

用户退出功能:

```
if ($(this).tree('isLeaf', node.target)) {  
    if (node.text === '用户退出') {  
        return logout();  
    }  
}
```



9.4 后台管理界面

3. 代码实现——判断用户是否已经登录

用户退出功能:

logout()函数

```
function logout() {  
    $.messager.confirm('确认退出', '您确定要退出?', function(r) {  
        if (r) {  
            localStorage.removeItem(Config.tokenName);  
            location.href = 'login.html';  
        }  
    });  
}
```

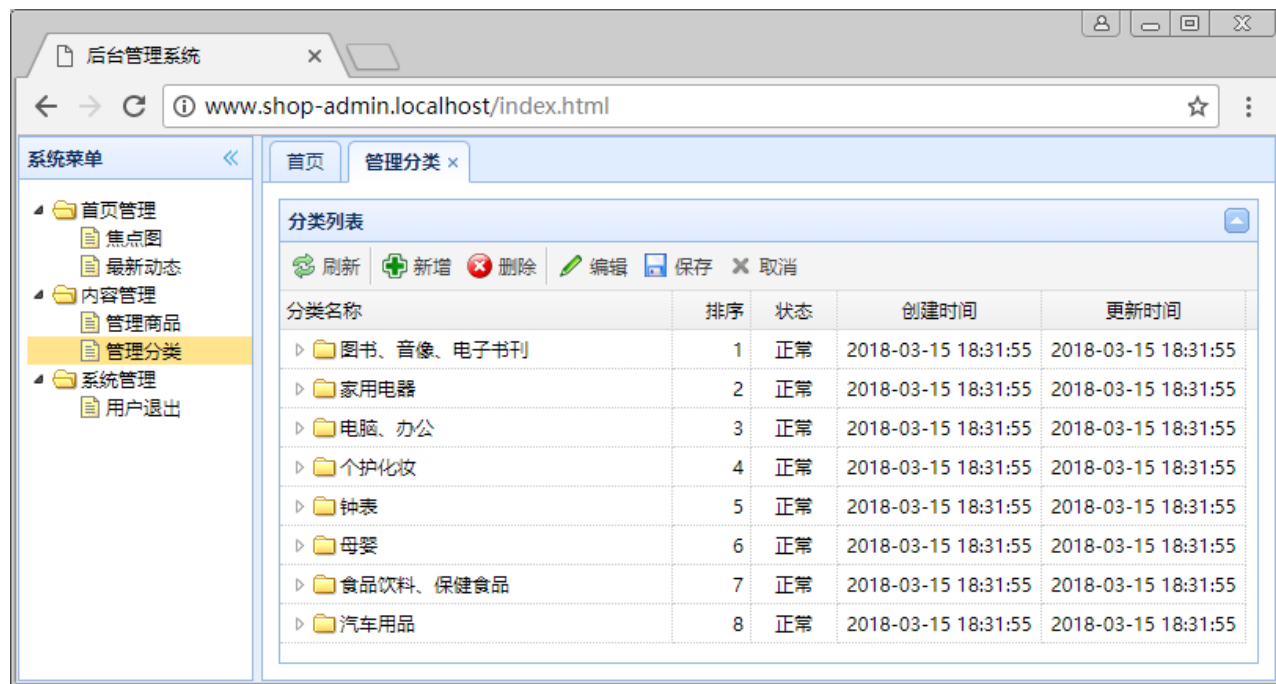


9.5 商品分类管理

1. 任务描述

商品分类:

- 分类列表
- 新增分类
- 修改分类
- 删除分类

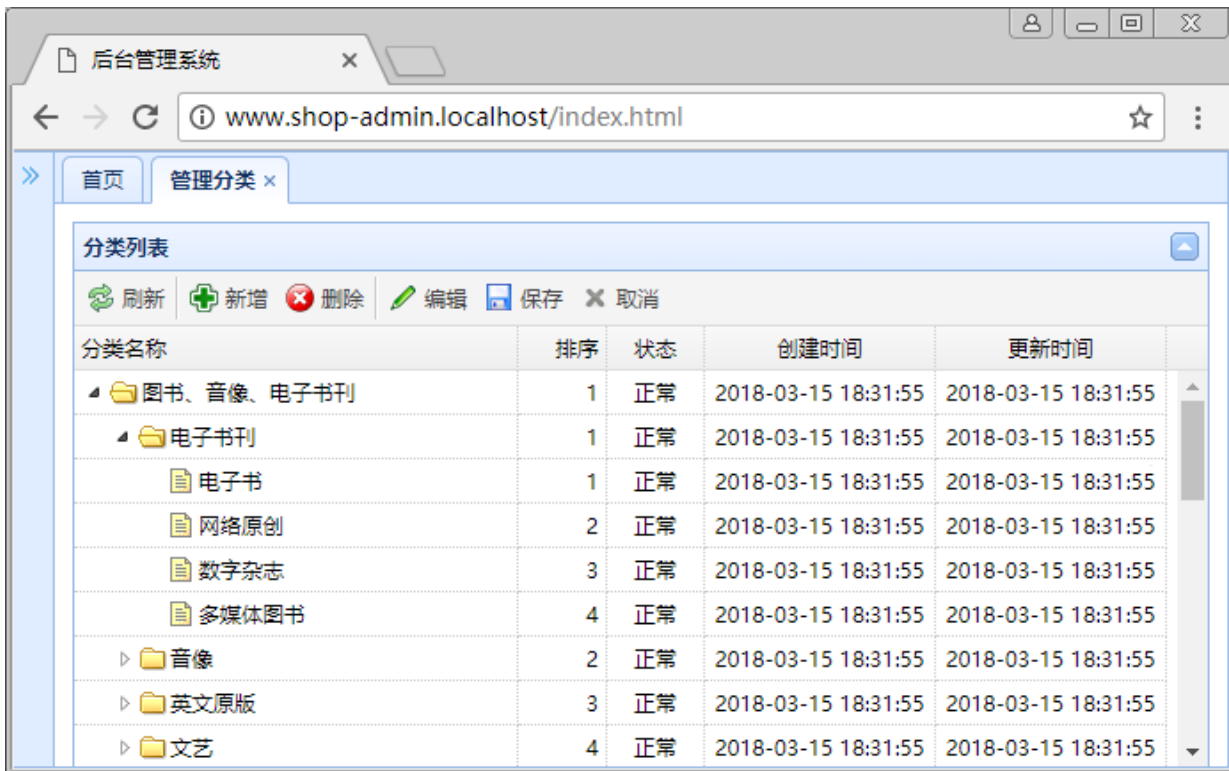




9.5 商品分类管理

1. 任务描述

分类列表：



分类列表				
刷新 新增 删除 编辑 保存 取消				
分类名称	排序	状态	创建时间	更新时间
▲ 图书、音像、电子书刊	1	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
▲ 电子书刊	1	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
电子书	1	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
网络原创	2	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
数字杂志	3	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
多媒体图书	4	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
▶ 音像	2	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
▶ 英文原版	3	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
▶ 文艺	4	正常	2018-03-15 18:31:55	2018-03-15 18:31:55



9.5 商品分类管理

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 任务描述

新增分类:

后台管理系统

www.shop-admin.localhost/index.html

分类列表

刷新 新增 删除

分类名称

- 图书、音像、电子书刊
 - 电子书刊
 - 电子书
 - 网络原创
 - 数字杂志
 - 多媒体图书
 - 音像
 - 英文原版
 - 文艺

新增分类

层次: 自定义 电子书刊

名称:

排序:

状态: 正常

提交 重置

4	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
---	----	---------------------	---------------------

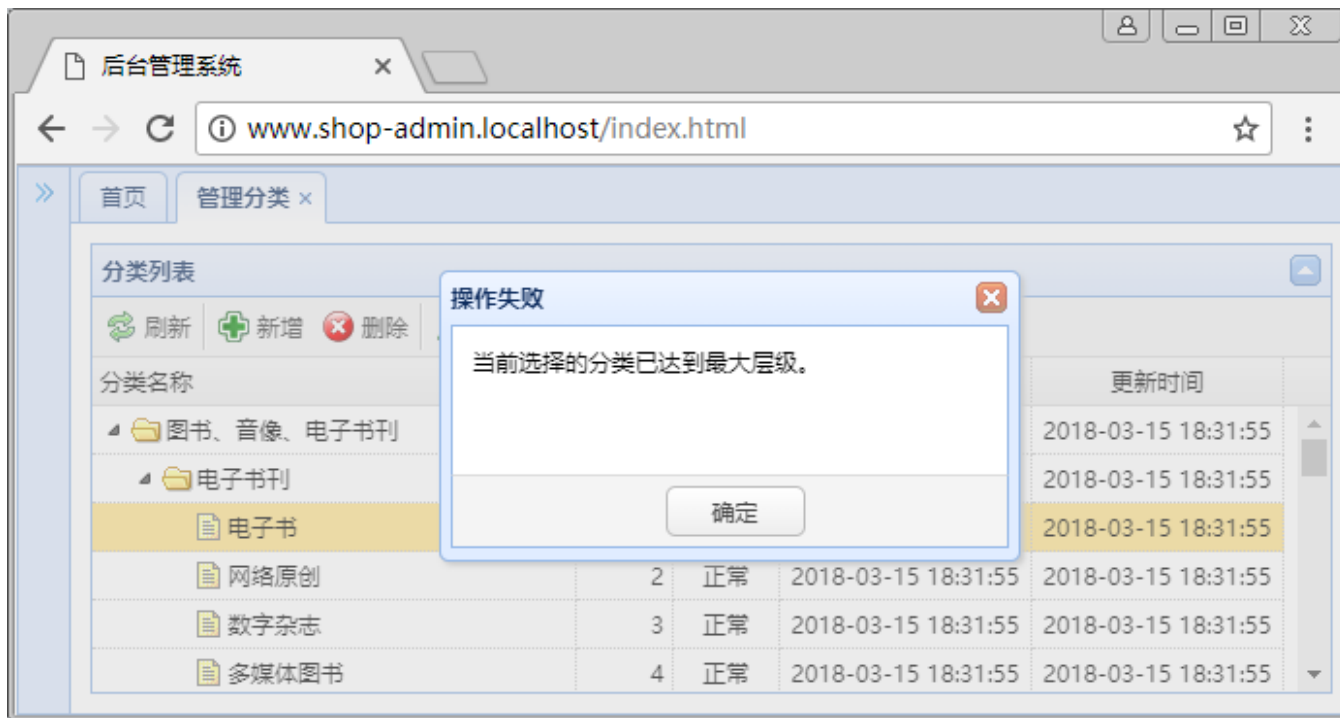


9.5 商品分类管理

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 任务描述

新增分类:





9.5 商品分类管理

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 任务描述

修改分类：

后台管理系统

www.shop-admin.localhost/index.html

首页 管理分类 x

分类列表

刷新 新增 删除 编辑 保存 取消

分类名称	排序	状态	创建时间	更新时间
图书、音像、电子书刊	1	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
电子书刊	1	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
电子书	1	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
网络原创	2	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
数字杂志	3	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
多媒体图书	4	正常	2018-03-15 18:31:55	2018-03-15 18:31:55



9.5 商品分类管理

2. 接口分析

查询分类列表:

向categories资源发送GET请求即可查询分类列表

URL	http://api.shop-admin.localhost/v1/categories	
基本信息	请求方式	-
	GET	-
URL参数	参数名	说明
	view	前端组件名（可选值：treegrid、tree）
	id	上级分类id（默认值为0，表示顶级分类）



9.5 商品分类管理

2. 接口分析

查询分类列表：当view的值为treegrid时，返回的示例结果如下所示

```
[{  
  "id": 1, // 分类id  
  "name": "图书、音像、电子书刊", // 分类名称  
  "status": 1, // 上架状态，1表上架，2表示下架  
  "sort": 1, // 排序数值（按数字大小升序排列）  
  "created": "2018-03-15 18:31:55", // 该条记录的创建时间  
  "updated": "2018-03-15 18:31:55", // 该条记录的更新时间  
  "state": "closed " // 节点状态，值为closed或open  
},.....]
```



9.5 商品分类管理

2. 接口分析

新增分类:

向categories资源发送POST请求表示新增分类.

URL	http://api.shop-admin.localhost/v1/categories	
基本信息	请求方式	内容类型
	POST	application/x-www-form-urlencoded
URL参数	参数名	说明
	parent_id	上级分类id（默认为0）
	name	分类名称
	sort	排序数值
	status	上架状态（1表示上架，2表示下架）



9.5 商品分类管理

2. 接口分析

修改分类：向categories资源发送PUT请求表示新增分类

URL	http://api.shop-admin.localhost/v1/categories/:id	
基本信息	请求方式	内容类型
	PUT	application/x-www-form-urlencoded
URL参数	参数名	说明
	name	分类名称
	sort	排序数值
	status	上架状态（1表示上架，2表示下架）



9.5 商品分类管理

2. 接口分析

删除分类:

向categories资源发送PUT请求表示新增分类

URL	http://api.shop-admin.localhost/v1/categories/:id	
基本信息	请求方式	-
	DELETE	-



9.5 商品分类管理

传智播客·黑马程序员
改变中国IT教育 我们正在行动

3. 代码实现

商品分类管理

1 分类列表

2 分类修改

3 列表刷新

4 分类删除

5 分类添加



9.6 商品管理

1. 任务描述

商品管理:

- 商品列表
- 新增商品
- 修改商品
- 删除商品
- 商品上下架

The screenshot shows a web browser window with the address bar displaying `www.shop-admin.localhost/index.html`. The page title is "后台管理系统" (Backend Management System). The main content area is titled "商品列表" (Product List) and includes a toolbar with icons for refresh, add, edit, delete,上架 (shelve), and下架 (unshelve). Below the toolbar is a table with the following columns: 商品名称 (Product Name), 卖点 (Selling Point), 价格 (Price), 库存数量 (Inventory Quantity), 状态 (Status), 创建日期 (Creation Date), and 更新日期 (Update Date). The table contains 8 rows of product data. At the bottom of the table, there is a pagination bar showing "20" items per page, "第 1 页" (Page 1), and "共 3 页" (Total 3 pages). A status bar at the bottom right indicates "显示 1 到 20, 共 41 记录" (Display 1 to 20, total 41 records).

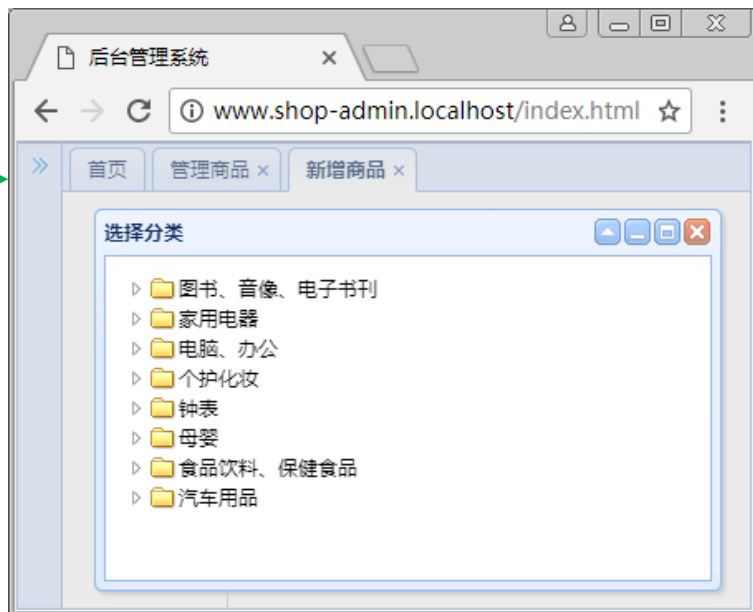
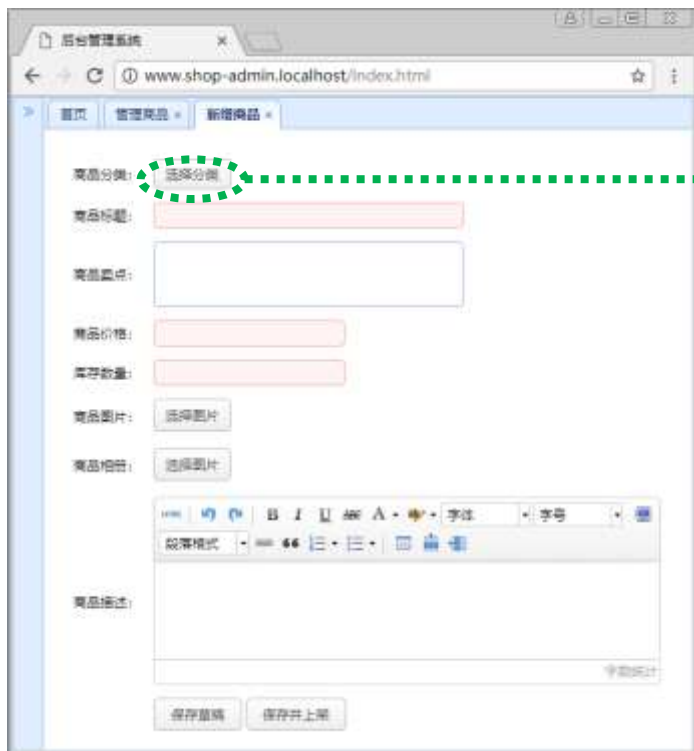
<input type="checkbox"/>	商品名称	卖点	价格	库存数量	状态	创建日期	更新日期
<input type="checkbox"/>	响应式Web开发项目教程 (HTML5+	工业和信息化 "十	42.00	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	Java Web程序设计任务教程	工业和信息化 "十	56.00	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	微服务架构基础 (Spring Boot+Spri	微服务已逐步进入	35.00	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	JavaScript前端开发案例教程	工业和信息化 "十	49.80	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	Java EE企业级应用开发教程 (Spring	工业和信息化 "十	49.80	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	PHP基础案例教程	工业和信息化 "十	49.80	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	新媒体营销教程	新媒体时代网络营	35.00	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55
<input type="checkbox"/>	Python快速编程入门	工业和信息化 "十	39.80	100	正常	2018-03-15 18:31:55	2018-03-15 18:31:55



9.6 商品管理

1. 任务描述

新增商品:





9.6 商品管理

2. 接口分析

查询商品列表:

向items资源发送GET请求即可查询商品列表

URL	http://api.shop-admin.localhost/v1/items	
基本信息	请求方式	-
	GET	-
URL参数	参数名	说明
	page	查询的页码（默认为1）
	rows	每页的记录数（默认为20，最多100）



9.6 商品管理

2. 接口分析

查询商品列表: 请求结果

```
{ "total": 26, // 总记录数
  "rows": [ // 当前查询的页码下的记录
    { "id": 1, "cid": 3, // 商品id、所属分类id
      "title": "", "sell_point": "", // 商品名称、卖点
      "price": "100.00", "num": 11, // 价格、库存数
      "pic": "", "status": 1, // 商品图片、上架状态（1上架，2下架）
      "content": "", "album": "", // 商品描述、商品相册
      "created": "", "updated": "" // 创建时间、更新时间
    }, ..... ]
}
```



9.6 商品管理

2. 接口分析

查询商品信息:

通过GET方式请求items资源

URL	http://api.shop-admin.localhost/v1/items/:id	
基本信息	请求方式	-
	GET	-



9.6 商品管理

2. 接口分析

查询商品信息:

请求结果

```
{  
  "id": 1, "cid": 3, "title": "", "sell_point": "", "price": "100.00",  
  "num": 11, "pic": "", "status": 1, "content": "", "album": "",  
  "created": "", "updated": "", "category_name": "电子书"  
}
```



9.6 商品管理

2. 接口分析

新增商品：向items资源发送POST请求表示新增商品

URL	http://api.shop-admin.localhost/v1/items	
基本信息	请求方式	内容类型
	POST	application/x-www-form-urlencoded
主体	参数名	说明
	cid	所属分类id
	title	商品名称
	sell_point	卖点
	price	价格
	num	库存数
	content	商品描述
	status	上架状态（1表示上架，2表示下架）



9.6 商品管理

2. 接口分析

新增商品:

返回数据库为新商品分配的id

```
{ "id " : "10 " }
```



9.6 商品管理

2. 接口分析

修改商品：向items资源发送PUT请求表示修改商品

URL	http://api.shop-admin.localhost/v1/items/:id	
基本信息	请求方式	内容类型
	PUT	application/x-www-form-urlencoded
主体	参数名	说明
	cid	所属分类id
	title	商品名称
	sell_point	卖点
	price	价格
	num	库存数
	content	商品描述
	status	上架状态（1表示上架，2表示下架）



9.6 商品管理

2. 接口分析

商品上下架切换:

向items资源发送PATCH请求表示修改某些字段

URL	http://api.shop-admin.localhost/v1/items	
基本信息	请求方式	内容类型
	PATCH	application/x-www-form-urlencoded
URL参数	参数名	说明
	ids	商品id字符串（多个用“,”分隔）
主体	参数名	说明
	status	上架状态（1表示上架，2表示下架）



9.6 商品管理

2. 接口分析

删除商品：

向items资源发送DELETE请求表示删除商品

URL	http://api.shop-admin.localhost/v1/items	
基本信息	请求方式	-
	DELETE	-
URL参数	参数名	说明
	ids	商品id字符串（多个用“,”分隔）



9.6 商品管理

3. 代码实现

商品列表页面：

content\item-list.html

```
list.datagrid({  
    collapsible: true, fit: true, pagination: true, pageSize: 20,  
    url: Config.api + 'items', method: 'GET',  
    columns: [], toolbar: []  
});
```

列表数据项

列表工具栏



9.6 商品管理

3. 代码实现

列表数据项：“columns: [[]],” 改为如下代码

```
columns: [[
  {field: 'checkbox', checkbox: true},
  {title: '商品名称', field: 'title', width: 200},
  {title: '卖点', field: 'sell_point', width: 100},
  {title: '价格', field: 'price', width: 70, align: 'right'},
  {title: '库存数量', field: 'num', width: 70, align: 'right'},
  {title: '状态', field: 'status', width: 60, align: 'center',
    formatter: formatStatus },
  {title: '创建日期', field: 'created', width: 130, align: 'center'},
  {title: '更新日期', field: 'updated', width: 130, align: 'center'}]]],
```

处理状态信息



9.6 商品管理

3. 代码实现

列表数据项:

formatStatus

```
function formatStatus(val) {  
    return {1: '正常', 2: '下架'}[val];  
}
```



9.6 商品管理

3. 代码实现

列表数据项:

“toolbar: []” 改为如下代码

```
toolbar: [ {text: '刷新', iconCls: 'icon-reload', handler: reload}, '-',  
  {text: '新增', iconCls: 'icon-add', handler: add},  
  {text: '编辑', iconCls: 'icon-edit', handler: edit},  
  {text: '删除', iconCls: 'icon-cancel', handler: del}, '-',  
  {text: '上架', iconCls: 'icon-ok', handler: function() {status(1);}},  
  {text: '下架', iconCls: 'icon-no', handler: function() {status(2);}}]
```




9.6 商品管理

3. 代码实现

列表刷新:

datagrid重新从后端API获取商品列表数据

```
function reload() {  
    list.datagrid( 'reload' );  
}
```



9.6 商品管理

3. 代码实现

商品删除:

请求API，删除指定商品

```
var ids = getSelectionsIds();  
var url = Config.api + 'items?ids=' + ids.join(',');  
$.ajax({type: 'DELETE', url: url, success: function() {  
    $.messager.alert('提示', '删除商品成功!', undefined, reload);  
}});
```

获取列表中当前选中的商品



9.6 商品管理

3. 代码实现

商品删除:

getSelectionsIds方法的封装

```
function getSelectionsIds() {  
    var sels = list.datagrid('getSelections');  
    var ids = [];  
    for (var i in sels) {  
        ids.push(sels[i].id);  
    }  
    return ids;  
}
```



9.6 商品管理图片管理

3. 代码实现

商品上下架：

1

判断是否选中商品

2

提示用户是否改变所选商品状态

3

请求API，改变所选商品的状态



9.6 商品管理

3. 代码实现

商品添加和修改页面：

edit(): 商品修改

1

判断是否选中商品

2

判断了目标标签页是否存在

3

弹出“编辑商品”标签页



9.6 商品管理

3. 代码实现

商品添加和修改页面：

add()：商品添加

1

判断标签页是否已经展示

2

弹出“新增商品”标签页



9.6 商品管理

3. 代码实现

商品添加和修改页面:

itemEdit(): 对标签页内的表单进行处理

```
function itemEdit(obj, id) {  
    createForm();  
    createCategory();  
    function createForm() {} // 处理表单控件  
    function createCategory() {} // 处理分类选择控件  
    function createEditor() {} // 处理编辑器  
    function createImage(image) {} // 处理图片组件  
}
```



9.6 商品管理

3. 代码实现

分类选择组件：

1

为“选择分类”按钮添加单击事件

2

打开子窗口

3

在子窗口内创建一个tree组件

4

为tree中的每一个节点添加单击事件



9.6 商品管理

3. 代码实现

将商品原有数据填写到表单：

1

获取商品数据

2

判断是否是新商品

3

非新商品时向服务器请求数据

4

将数据填入标签页中



9.6 商品管理

3. 代码实现——编辑器

将商品原有数据填写到表单：

1

引入UEditor

2

编写createEditor()函数创建编辑器

3

非新商品时向服务器请求数据



9.6 商品管理

3. 代码实现——编辑器

提交表单：

1

获取标签页

2

检测是否验证成功

3

判断当前是添加商品还是修改商品

4

提交数据到服务器

5

显示了处理结果



9.7 商品图片管理

1. 任务描述

分类选择组件：





9.7 商品图片管理

2. 接口分析

上传文件:

向files资源发送POST请求表示上传文件

URL	http://api.shop-admin.localhost/v1/files	
基本信息	请求方式	内容类型
	POST	multipart/form-data



9.7 商品图片管理

2. 接口分析

上传文件:

向files资源发送POST请求表示上传文件

URL参数	参数名	说明
	type	文件类型，目前只支持图片（image）
	relation	关联的表名（如item表示商品表）
	relation_id	关联的id（如item表中的商品id）
	name	关联的字段名（pic或album）
主体	参数名	说明
	name	与URL参数中的name一致
	（其他参数由程序自动发送）



9.7 商品图片管理

2. 接口分析

上传文件:

返回结果

```
{  
  "path": "image.....90b124.jpg", // 图片保存路径  
  "relation_id": 1 // 关联的id (若新增了记录, 则为新增的id)  
}
```



9.7 商品图片管理

2. 接口分析

上传文件:

上传文件时name属性的值

name	参数名	说明
	pic	服务器会自动生成220*220像素的缩略图，不会保留用户上传的原图
	album	服务器会自动生成3种尺寸的缩略图，分别为100*100、350*350、800*800



9.7 商品图片管理

2. 接口分析

删除文件：

向files资源发送POST请求表示删除文件

URL	http://api.shop-admin.localhost/v1/files	
基本信息	请求方式	-
	DELETE	-
URL参数	参数名	说明
	relation	关联的表名（如item表示商品表）
	relation_id	关联的id（如item表中的商品id）
	name	关联的字段名（pic或album）
	path	图片路径



9.7 商品图片管理

3. 代码实现

创建图片上传组件：

引入WebUploader组件所需文件

```
<link rel="stylesheet" href="js/webuploader-0.1.5/webuploader.css">  
<script src="js/webuploader-  
0.1.5/webuploader.html5only.min.js"> </script>
```



9.7 商品图片管理

3. 代码实现

创建图片上传组件：

编写createImage()函数

```
function createImage(image) {  
    var pic = imagesUploader(obj.find('.upload_pic'), 'pic', 1);  
    var album = imagesUploader(obj.find('.upload_album'), 'album', 10);  
    function imagesUploader(obj, name, max) { }  
}
```



9.7 商品图片管理

3. 代码实现

创建图片上传组件：

编写imagesUploader ()函数

1

为“选择图片”按钮添加id属性

2

根据后端API的要求将参数放入URL中

3

创建WebUploader组件

4

将创建后的组件对象返回



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能:

执行上传前，将认证令牌放入请求头

```
uploader.on('uploadBeforeSend', function (obj, data, headers) {  
    headers.Authorization = Config.token;  
});
```



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能:

当有文件添加时，判断文件数量

```
uploader.on('beforeFileQueued', function() {  
    if (obj.find('.webuploader-url').length >= max) {  
        $.messager.alert('操作失败', '最多只能上传' + max + '张图片!');  
        return false;  
    }  
});
```



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能：当有文件添加后，显示预览图

返回一个新创建的图片元素对象

```
uploader.on('fileQueued', function(file) {  
    var li = fileItem(file.id, file.name); var img = li.find('img');  
    obj.find('.webuploader-list').append(li);  
    uploader.makeThumb(file, function (error, src) {  
        if (error) {img.replaceWith('<span class="webuploader-tip">不能预览</span>'); return;}  
        img.attr('src', src);  
    }, 100, 100);  
});
```

预览图大小



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能: 上传过程中创建进度条显示进度

```
uploader.on('uploadProgress', function(file, percentage) {  
    var li = $('#' + file.id);  
    var percent = li.find('.webuploader-progress span');  
    if (!percent.length) {  
        percent = $('<p class="webuploader-  
        progress"><span></span></p>').appendTo(li).find('span');  
    }  
    percent.css('width', percentage * 100 + '%');  
});
```

上传进度



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能:

上传完成，删除进度条

```
uploader.on('uploadComplete', function ( file ) {  
    $('#'+ file.id ).find('.webuploader-progress'). remove();  
});
```



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能：

上传失败，显示错误提示

```
uploader.on('uploadError', function( file ) {  
    var li = $('#' + file.id);  
    var error = li.find('div.webuploader-error');  
    if (!error.length) {  
        error = $( '<div class= "webuploader-error" >上传失败  
                    </div> ').appendTo(li);}  
    });
```



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能：`fileItem()`函数：该函数用于返回一个新创建的图片元素对象

图片的URL地址

```
function fileItem( fid , name , src ) {  
    var item = $('<div class="webuploader-file-item webuploader-  
    thumbnail"><img></div>');  
    fid && item.attr('id', fid);  
    name && item.append('<div class="webuploader-info">'+  
    name + '</div>');  
    src && item.find('img').attr('src', src); return item;}  

```



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能:

处理上传成功事件

将后端API返回的商品id替换当前保存的商品id

```
uploader.on('uploadSuccess', function(file, response) {  
    var li = $('#'+ file.id);  
    if (!li.find('.webuploader-done').length) {  
        id === 0 && updateId(response.relation_id);  
        fileControl(li, response.path);  
    });
```

为图片增加“删除”按钮和保存文件路径的隐藏域



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能:

updateId 函数: 在商品添加时, 将后端API返回的商品id替换当前保存的商

```
function updateId(relation_id) {  
  function replace(obj) {  
    obj.server = obj.server.replace('&relation_id=0',  
    '&relation_id=' + id);}  
  id = relation_id;  
  replace(pic.options);  
  replace(album.options);  
}
```

更改pic和album对象中保存的后端API地址



9.7 商品图片管理

3. 代码实现

扩展上传组件的功能：

fileControl 函数：为图片增加“删除”按钮和保存文件路径的隐藏域

```
function fileControl(obj, path) {  
    obj.append('<div class="webuploader-opt">  
        <span class="webuploader-del">[删除]</span></div>');  
    obj.append('<input class="webuploader-url"  
        type="hidden" value="'+ path + '">');  
};
```



9.7 商品图片管理

3. 代码实现

删除图片：

```
$.ajax({  
    type: 'DELETE',  
    url: Config.api + 'file?relation=item&relation_id=' + id,  
    data: {name: name, path: obj.find('.webuploader-url').val()},  
    success: function() {  
        obj.find('.webuploader-file-item').remove();  
    }  
});
```

请求服务器删除数据库数据

请求成功后删除页面内容



9.7 商品图片管理

3. 代码实现

修改商品时显示已有图片：

读取图片路径，将图片显示在页面中

```
var url = Config.uploadURL;
```

图片上传后的URL

```
if (image.pic !== '') {  
    var picList = obj.find('.upload_pic .webuploader-list');  
    var item = fileItem(false, false, url + image.pic);  
    picList.append(item);  
    fileControl(item, image.pic);  
}
```




9.7 商品图片管理

3. 代码实现

修改商品时显示已有图片：

读取图片路径，将图片显示在页面中

```
if (image.album !== '') {  
    var albumList = obj.find('.upload_album .webuploader-list');  
    var albs = image.album;  
    for (var i in albs) {  
        var item = fileItem(false, false,  
            url + albs[i].replace('[prefix]', 'album_small_'));  
        albumList.append(item); fileControl(item, albs[i]);  
    }  
}
```

将路径中的 “[prefix]” 替换成图片尺寸前缀

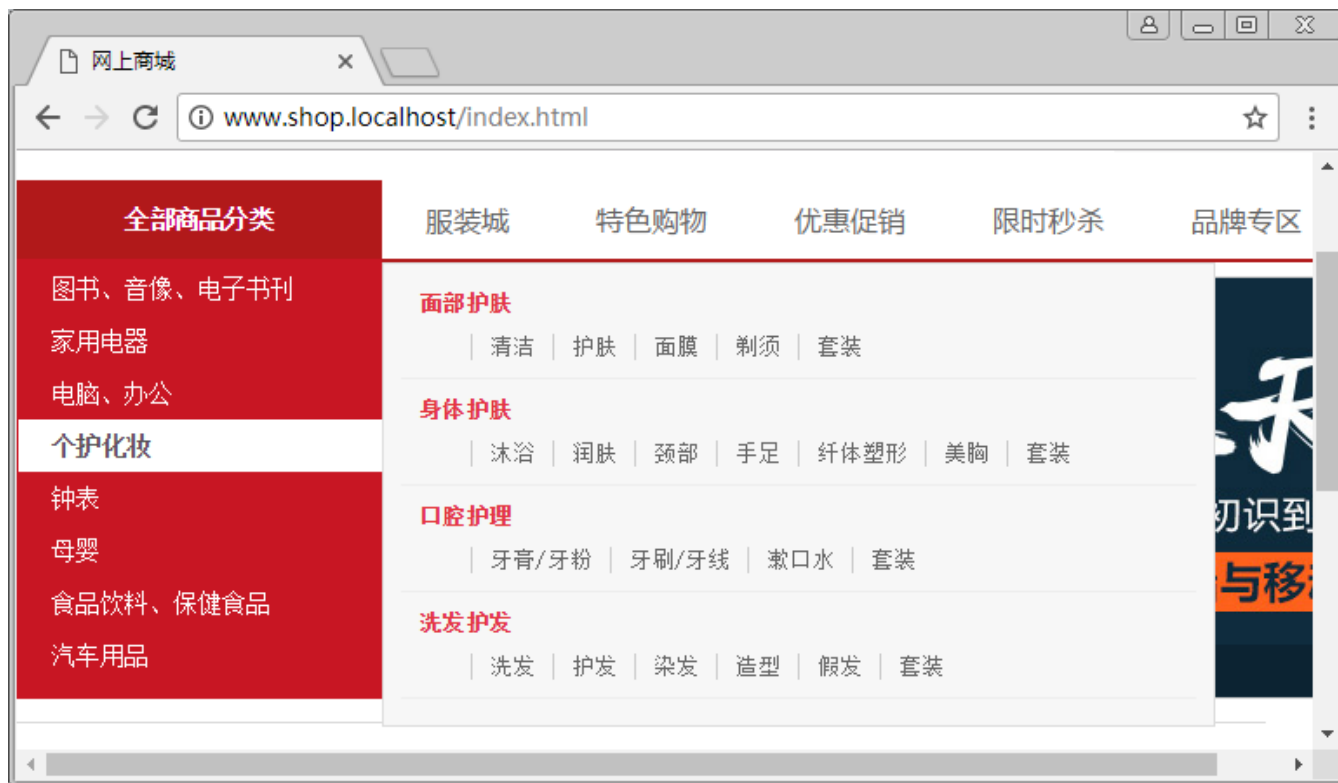


9.8 商城首页

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 任务描述

商城首页：





9.8 商城首页

2. 接口分析

查询分类:

向categories资源发送GET请求表示查询分类数据

URL	http://api.shop.localhost/v1/categories	
基本信息	请求方式	-
	GET	-
URL参数	参数名	说明
	view	前端组件名（可选值：nav、crumbs）
	id	分类id（view=crumbs时有效）



9.8 商城首页

2. 接口分析

查询分类:

当view为nav时，会返回所有status为1的分类数据

```
[  
  {"id": 1183, "parent_id": 1, "name": "test"}, .....  
]
```



9.8 商城首页

2. 接口分析

查询分类:

当view为nav时，会返回所有status为1的分类数据

```
[  
  {"id": 1, "parent_id": 0, "name": "图书、音像、电子书刊"},  
  {"id": 2, "parent_id": 1, "name": "电子书刊"},  
  {"id": 3, "parent_id": 2, "name": "电子书"}  
]
```



9.8 商城首页

2. 接口分析

查询焦点图:

向slides资源发送GET请求表示查询焦点图数据

URL	http://api.shop.localhost/v1/slides	
基本信息	请求方式	-
	GET	-



9.8 商城首页

2. 接口分析

查询焦点图:

请求成功后，返回的结果示例如下

```
[  
  {"id": 1, "title": "图片1", "pic": "image.....32.jpg", "url": "#"},  
]
```



9.8 商城首页

2. 接口分析

查询最新状态:

向news资源发送GET请求表示查询最新动态数据

URL	http://api.shop.localhost/v1/news	
基本信息	请求方式	-
	GET	-



9.8 商城首页

2. 接口分析

查询焦点图:

请求成功后，返回的结果示例如下

```
[  
  {"id": 1, "title": "标题1", "color": "", "url": "#"},  
]
```



9.8 商城首页

2. 接口分析

查询精品推荐:

向goods资源发送GET请求表示查询精品推荐数据

URL	http://api.shop.localhost/v1/goods	
基本信息	请求方式	-
	GET	-
URL参数	参数名	说明
	count	返回的商品数量
URL	http://api.shop.localhost/v1/goods	



9.8 商城首页

2. 接口分析

查询焦点图:

请求成功后，返回的结果示例如下

```
[  
  {"id": 26, "title": "", "price": "100.00", "pic": "image.....60f.jpg"},  
]
```



9.8 商城首页

3. 代码实现

书写“商城首页”文档结构

- ❑ 顶部导航栏
- ❑ 头部（从左到右依次是LOGO、搜索框、快捷按钮）
- ❑ 主导航栏
- ❑ 内容区域
- ❑ 服务链接
- ❑ 页脚
- ❑ 引入JavaScript公共代码文件



9.8 商城首页

3. 代码实现

创建前台的配置文件:

js\config.js

```
var Config = {  
  api: 'http://api.shop.localhost/v1/',  
  uploadURL: 'upload/'  
};
```



9.8 商城首页

3. 代码实现

载入内容区域：将内容区域分离出来，保存到单独的文件中，在页面打开后使用Ajax进行载入。其中Apache分布式配置文件“.htaccess”文件对前台进行URL重写，请求URL与实际请求网页、通过Ajax载入的内容区域网页的对应关系如下：

页面名称	请求URI	实际请求网页	内容区域网页
首页	/	index.html	content\index.html
	/index.html	index.html	content\index.html
商品列表页	/find-{分类id}.html	index.html	content\find.html
商品查看页	/item-{商品id}.html	index.html	content\item.html
购物车	/cart.html	index.html	content\cart.html



9.8 商城首页

3. 代码实现

处理URL:

```
var Common = {  
  id: 0, // 当前请求id  
  content: 'index', // 当前请求的内容区域 ( 对应content/index.html )  
  init: function() {this.getParams(location.href);},  
  getParams: function(url) { }, getContent: function() { } };
```

Common.init();

从参数url中获取请求的内容区域和id

Common.getContent();

根据当前请求的内容区域，通过Ajax
载入对应的页面



9.8 商城首页

3. 代码实现

处理URL:

getParams: 从参数url中获取请求的内容区域和id

```
getParams: function(url) {  
    var url = url.match(/\/(\w+)(?:-(\d+))?.html/);  
    if (url !== null) {  
        this.content = url[1];  
        this.id = url[2] === undefined ? 0 : url[2];  
    }  
}
```




9.8 商城首页

3. 代码实现

处理URL:

getContent: 根据当前请求的内容区域，通过Ajax载入对应的页面

```
getContent: function() {  
    var content = this.content;  
    $.ajax({  
        type: 'GET', url: 'content/' + content + '.html', cache: false,  
        success: function (data) {  
            $('#content').html(data);  
        });  
    }  
}
```



9.8 商城首页

3. 代码实现

主导航栏的商品分类:

从后端API获取数据,获取数据后与页面模板绑定展示数据。

```
var categoryList = $('#category_list');  
$.get(Config.api + 'categories?view=nav', function(data) {  
    var html = template('category_list_tpl', {item: data});  
    categoryList.html(html);  
});
```



9.8 商城首页

3. 代码实现

主导航栏的商品分类:

当用户使用鼠标滑到分类菜单中时，显示所选分类下的二级、三级分类

```
$('.category-item').hover(function() {  
    $(this).find('.category-sub').show();  
    $(this).children('.category-main').children('a').addClass('on');  
}, function() {  
    $(this).find('.category-sub').hide();  
    $(this).children('.category-main').children('a').removeClass('on');  
});
```



9.8 商城首页

3. 代码实现

主导航栏的商品分类：

分类菜单在首页默认展开，其他页面默认收起。

```
var categoryList = $('#category_list');  
var categoryShow = $('#category_show');  
if (content === 'index') {categoryList.show();  
    categoryShow.off('mouseenter').off('mouseleave');  
} else {categoryList.hide();  
    categoryShow.hover(function() {  
        categoryList.show();}, function() {categoryList.hide();});}
```



9.8 商城首页

3. 代码实现

焦点图:

为Common对象添加slidePath方法对图片路径进行处理

```
slidePath: function(data) {  
    for (var i in data) {  
        data[i].pic = Config.uploadURL + data[i].pic;  
    }  
}
```



9.8 商城首页

3. 代码实现

最新动态:

编写“最新动态”模块的相关代码，请求数据后展示相关内容

```
$.get(Config.api + 'news', function(data) {  
    $('.news-content').html(template('news_tpl', {item: data}));  
});
```



9.8 商城首页

3. 代码实现

精品推荐:

编写“最新动态”模块的相关代码，请求数据后展示相关内容

```
$.get(Config.api + 'goods?count=6', function(data) {  
    Common.itemPicPath(data);  
    $('.best-content').html(template('best_tpl', {item: data}));  
});
```

对图片路径进行处理



9.8 商城首页

3. 代码实现

精品推荐:

为Common对象添加itemPicPath方法对图片路径进行处理

```
itemPicPath: function(data) {  
    for (var i in data) {  
        if (data[i].pic === '') {  
            data[i].pic = 'img/preview.jpg';  
        } else {  
            data[i].pic = Config.uploadURL + data[i].pic;  
        }  
    }  
}
```

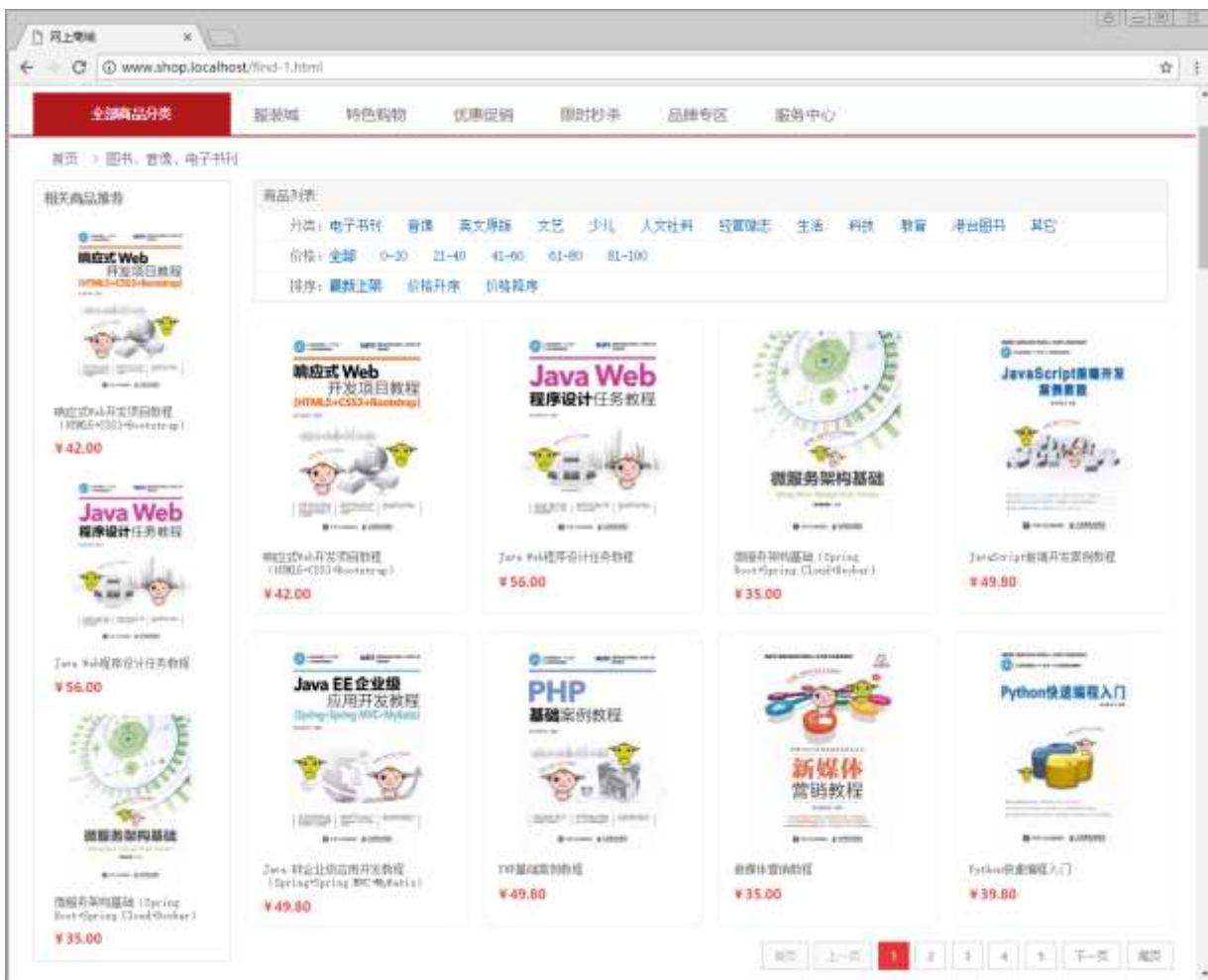



9.9 商品列表

传智播客·黑马程序员
改变中国IT教育 我们正在行动

1. 任务描述

商品列表页：





9.9 商品列表

2. 接口分析

接口提供了查询商品列表的功能：

URL	http://api.shop.localhost/v1/items	
基本信息	请求方式	-
	GET	-
URL参数	参数名	说明
	price_max	价格区间最大值
	price_min	价格区间最小值
	sort	排序方式
	cid	分类id
	page	页码值（最小值为1）
	pagesize	每页条数（最大值为50）



9.9 商品列表

2. 接口分析

请求成功后，返回的示例结果如下：

```
{
  "priceMax": 100,
  "item": [
    {"id": 6, "title": "", "price": "100.00", "pic": "image...0f.jpg"}, ...],
  "total": 1,
  "category": [
    {"id": 1183, "name": "test"}, ...],
  "recommend": [
    {"id": 8, "title": "", "price": "100.00", "pic": "image.....0f.jpg"}, ...]
} //
```

符合查询条件的商品中的最大价格值

当前页码值下的商品

符合查询条件的总记录数

分类id (cid) 的子分类

相关商品推荐



9.9 商品列表

3. 代码实现

分类导航:

```
var url = Config.api + 'categories?view=crumbs&id=' + params.cid;  
$.get(url, function(data) {  
    $('.crumbs').html(template('crumbs_tpl', {item: data}));  
});
```



9.9 商品列表

3. 代码实现

商品列表:

```
var params = {price_max: 0, price_min: 0, sort: 'id-desc',  
              cid: Common.id, page: 1, pagesize: 20};  
$.get(Config.api + 'items', params, function(data) {  
    Common.itemPicPath(data.item);  
    $('.find-item').html(template('find_item_tpl', {item: data.item}));  
    if (isFirst) {isFirst = false; firstLoadData(data);}});
```

首次加载时的操作



9.9 商品列表

3. 代码实现

相关商品推荐：

获取推荐商品信息

```
function firstLoadData(data) {  
    // 相关商品推荐  
    Common.itemPicPath(data.recommend);  
    $('.find-ad-content').html(template('find_ad_tpl', {item:  
    data.recommend}));  
}
```



9.9 商品列表

3. 代码实现

按分类筛选商品：

通过模板输出数据

```
$('.selector-category').html(template('selector_category_tpl',{id: data.cid ,item: data.category}));
```



9.9 商品列表

3. 代码实现

按价格筛选商品：

通过模板输出数据

输出价格筛选的链接

```
$('.selector-price').html(template('selector_price_tpl',  
    {item: priceDist(data.priceMax,5)}))  
.find('a:first').addClass('selector-curr');
```




9.9 商品列表

3. 代码实现

按价格筛选商品：

priceDist()函数用于生成从1~max价格范围内的count个价格链接

```
function priceDist(max, count) {  
  if (max <= 0) {return '';}  
  var size = Math.ceil(max / Math.max(count, 1));  
  var end = size, start = 0, rst = [];  
  for (var i = 0; i < count; ++i) {  
    rst.push(start + '-' + end); start = end + 1; end += size;  
  }  
  return rst;  
}
```



9.9 商品列表

3. 代码实现

按价格筛选商品：

为价格链接添加单击事件，发送新的价格区间参数，重新请求商品列表数据

```
$('.selector-price').on('click', 'a', function() {  
    var param = $(this).attr('data-param');  
    params.price_max = param[0];  
    params.price_min = param[1];  
    loadData();  
    $(this).addClass('selector-curr').siblings().removeClass('selector-curr');  
    return false;  
});
```



9.9 商品列表

3. 代码实现

商品列表排序：

在单击排序链接时，发送新的排序参数，重新请求商品列表数据

```
$('.selector-sort a').click(function() {  
    params.sort = $(this).attr('data-param');  
    loadData();  
    $(this).addClass('selector-curr').siblings().  
        removeClass('selector-curr');  
    return false;  
});
```



9.9 商品列表

3. 代码实现

商品列表排序：

在单击排序链接时，发送新的排序参数，重新请求商品列表数据

```
var pagelist = $('.pagelist').page({  
    total: data.total,  
    size: params.pagesize  
});  
pagelist.click(function(page) {  
    params.page = page;  
    loadData();  
});
```

将params.page更新为被单击的链接所对应的页码值



9.10 商品详情

1. 任务描述

案例演示：

- ① 单击一件商品后，
- ② 就会打开商品详情页，
- ③ 显示商品的详细信息。





9.10 商品详情

2. 接口分析

查询指定id的商品信息的功能:

URL	http://api.shop.localhost/v1/items/:id	
基本信息	请求方式	-
	GET	-



9.10 商品详情

3. 代码实现

查询商品数据：

```
var id = Common.id;
$.get(Config.api + 'items/' + id, function(data) {
    // 查询成功后，将数据填入到模板中
}).fail(function() {
    alert('商品数据不存在！');
    location.href = 'index.html';
});
```



9.10 商品详情

3. 代码实现

分类导航:

查询到商品所属分类后，即可通过data.cid查询分类导航。

```
$.get(Config.api + 'categories?view=crumbs&id=' + data.cid,  
    function(data) {  
        $('.crumbs').html(template('crumbs_tpl', {item: data}));  
    }  
);
```




9.10 商品详情

3. 代码实现

商品信息和商品详情:

```
$('.item-info').html(template('item_info_tpl', {item: data}));  
$('.item-desc-content').html(data.content);
```

```
loadSuccess();
```

通过该函数对商品的购买数量进行控制



9.10 商品详情

3. 代码实现

控制商品数量：减少购买数量

```
var num = $('#item_num');  
var stock = parseInt($('.item-num-stock').text());  
$('.item-num-sub').click(function() { // 减少购买数量  
    var n = parseInt(num.val());  
    if (n < 1) {return false;}  
    num.val(n - 1);});
```



9.10 商品详情

3. 代码实现

控制商品数量: 增加购买数量

```
$('.item-num-add').click(function() {  
    var n = parseInt(num.val());  
    if (n > stock) {return false;}  
    num.val(n + 1);  
});
```



9.10 商品详情

3. 代码实现

控制商品数量: 增加购买数量

```
num.keyup(function() {  
    var n = parseInt($(this).val());  
    if ( n < 1 ) {  
        $(this).val(1);  
    } else if ( n > stock ) {  
        $(this).val(stock);  
    }  
});
```

将数字限制在“1~库存数”这个范围内



9.10 商品详情

3. 代码实现

商品相册：对商品相册数据进行处理

```
Common.itemAlbumPath(data.album);  
$('.album').html(template('album_tpl', {item: data.album}));
```



9.10 商品详情

3. 代码实现

商品相册：

为Common对象添加itemAlbumPath方法

```
itemAlbumPath: function(data) {  
    for (var i in data) {  
        data[i] = Config.uploadURL +  
        data[i].replace('[prefix]', 'album_small_');  
    },  
}
```



9.10 商品详情

3. 代码实现

相关商品推荐:

```
Common.itemPicPath(data.recommend);  
$('.item-ads-content').html(template('item_ads_tpl',  
{item: data.recommend}));
```



9.11 购物车

1. 任务描述

当用户在商品详情页单击“加入购物车”按钮后，就会将当前商品加入到购物车，然后跳转到购物车页面，显示购物车中的商品。





9.11 购物车

2. 接口分析

通过商品接口查询购物车页面所需的信息：

URL	http://api.shop.localhost/v1/items	
基本信息	请求方式	-
	GET	-
URL参数	参数名	说明
	view	前端组件名（值为shopcart表示购物车）
	ids	商品id字符串（多个用“,”分隔）



9.11 购物车

3. 代码实现

购物车数据管理:

将商品添加到购物车: 获取商品数据

```
addCart: function(id, num) {  
    id = parseInt(id);  
    num = parseInt(num);  
    var data = this.getCart();  
    var found = false;  
    .....  
}
```



9.11 购物车

3. 代码实现

购物车数据管理：

将商品添加到购物车： 处理数据

```
addCart: function(id, num) {  
    .....  
    for (var i in data) {if (data[i].id === id) { found = i; break; }}  
    if (found === false) {data.unshift({id: id, num: num});}  
    } else {data[found].num += num;}  
    .....  
}
```



9.11 购物车

3. 代码实现

购物车数据管理:

将商品添加到购物车: 存储数据

```
addCart: function(id, num) {  
    .....  
    localStorage.setItem('shop_front_cart', JSON.stringify(data));  
}
```



9.11 购物车

3. 代码实现

购物车数据管理:

获取购物车中的商品

```
getCart: function() {  
    var data = JSON.parse(localStorage.getItem('shop_front_cart'));  
    return data === null ? [] : data;  
},
```



9.11 购物车

3. 代码实现

添加商品到购物车：

为“加入购物车”按钮添加单击事件

```
$('#cart_add').click(function() {  
    // 将 商品id 和 购买数量 添加到本地存储（如果存在，则增加数量）  
    Common.addCart(id, num.val());  
    alert('商品已成功加入购物车！');  
    location.href = 'cart.html';  
    return false;  
});
```



9.11 购物车

3. 代码实现

查看购物车中的商品：

将购物车中的商品id取出，然后向后端API请求商品名称、价格、库存等数据

```
var cart = Common.getCart();  
var ids = []; for (var i in cart) {ids.push(cart[i].id);}  
$.get(Config.api + 'carts', {ids: ids.join(',')}, function(data) {  
    $('.shopcart').html(template('shopcart_tpl',  
        {item: data, cart: cart})); loadSuccess();  
});
```



9.11 购物车

3. 代码实现

全选和计算总价：

将购物车中的商品id取出，然后向后端API请求商品名称、价格、库存等数据

```
function loadSuccess() {  
    var check = $('.shopcart-check'); var checked = false;  
    check.click(total); // 复选框被单击时更新统计  
    $('.shopcart-checkall').click(checkAll); // 全选 或 全不选  
    checkAll();      // 默认为全选状态  
    .....  
}
```




9.11 购物车

3. 代码实现

全选和计算总价：

根据checked的值全选或全不选

```
function checkAll() {  
    checked = !checked;  
    check.prop('checked', checked);  
    check.attr('checked', checked);  
    total();  
}
```



9.11 购物车

3. 代码实现

全选和计算总价：统计商品数量和总价格

```
function total() {  
    var sum = 0; // 总数量 var total = 0; // 总价格  
    $('.shopcart-check:checked').each(function() {  
        var item = $(this).parents('.shopcart-item');  
        var price = parseFloat(item.find('.shopcart-price').text());  
        var num = parseInt(item.find('.shopcart-num').val());  
        sum += num; total += price * num;  
    });  
    $('.shopcart-total span').text(total.toFixed(2));  
    $('.shopcart-sum').text(sum);  
}
```



9.11 购物车

3. 代码实现

调整购买数量：为控制购买数量的按钮和文本框添加事件：减少购买数量

```
$('.shopcart-num-sub').click(function() {  
    var id = $(this).parent().find('.shopcart-id');  
    var num = $(this).parent().find('.shopcart-num');  
    var n = parseInt(num.val());  
    if (n <= 1) {return false;}  
    num.val(n - 1);  
    Common.editCart(id.val(), n - 1); total();  
});
```



9.11 购物车

3. 代码实现

调整购买数量：为控制购买数量的按钮和文本框添加事件：增加购买数量

```
$('.shopcart-num-add').click(function() {  
    var id = $(this).parent().find('.shopcart-id');  
    var num = $(this).parent().find('.shopcart-num');  
    var stock = $(this).parent().find('.shopcart-stock');  
    var n = parseInt(num.val());  
    if (n >= parseInt(stock.val())) {return false;}  
    num.val(n + 1); Common.editCart(id.val(), n + 1); total();  
});
```



9.11 购物车

3. 代码实现

调整购买数量:

为控制购买数量的按钮和文本框添加事件: 自动纠正购买数量

```
$('#shopcart-num').keyup(function() {  
    var stock = $(this).parent().find('.shopcart-stock').val();  
    var n = parseInt($(this).val());  
    if (n < 1) {$(this).val(1);} else if (n > stock) {$(this).val(stock);}  
    total();});
```



9.11 购物车

3. 代码实现

调整购买数量：

为控制购买数量的按钮和文本框添加事件：保存数量变更结果

```
$('.shopcart-num').blur(function() {  
    var id = $(this).parent().find('.shopcart-id').val();  
    var n = $(this).val();  
    Common.editCart(id, n);  
});
```



9.11 购物车

3. 代码实现

调整购买数量:

Common.editCart(): 将购买数量的更改结果保存到localStorage中

```
editCart: function(id, num) {  
    id = parseInt(id); num = parseInt(num); var data = this.getCart();  
    for (var i in data) {  
        if (data[i].id === id) {  
            data[i].num = num; if (num <= 0) { data.splice(i, 1); } break;  
        }  
        localStorage.setItem('shop_front_cart', JSON.stringify(data));  
    }  
}
```



9.11 购物车

3. 代码实现

从购物车中删除商品：

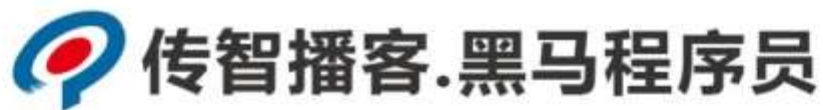
为“删除”链接添加单击事件

```
$('.shopcart-del').click(function() {  
    var id = $(this).parents().find('.shopcart-id').val();  
    $(this).parents('.shopcart-item').remove();  
    Common.editCart(id, 0);  
});
```




本章小结

本章通过一个在线商城的项目开发，讲解了jQuery、jQuery EasyUI、WebUploader、UEditor、art-template等前端库和插件在项目中的实际应用。为了提高课程的实战性，本项目采用了前后端分离的开发方式，前后端通过API接口进行数据交互，由前端完全负责页面呈现的逻辑代码。通过本章的学习，可以帮助读者将所学技术综合运用，积累开发经验。



Thank You!

yx.boxuegu.com

