## 仿JD电商系统

### 项目简介

系统模仿京东网站，功能需求来源于国内某培训班毕业项目，系统采用分布式开发，通过开启多个服务进程来模拟分布式环境。已实现功能：后台管理商品数据，文件上传至图片服务器，分布式系统单点登录注册功能，前台商品搜索功能，商品详情显示，购物车相关功能，创建订单功能，首页广告位展示等。

### 系统开发环境和工具：

|  |  |
| --- | --- |
| IDE | Eclipse |
| OS | Linux Ubuntu16.04 |
| 技术选型 | Spring3,SpringMVC,MyBatis,redis(cluster),solr,httpclient,nginx，PageHelper |
| 项目管理 | Maven |
| Http服务器 | Tomcat ,Nginx |
| 数据库 | MySQL |
| 连接池 | 阿里开源项目 DruidDataSource |
| 虚拟机 | 一台，安装Centos7系统 |

### 系统架构图

系统采用分布式开发，采用一台物理机（Ubuntu）+ 虚拟机（Centos），通过开启多个服务进程来模拟分布式环境。各个子系统说明如下

|  |  |
| --- | --- |
| Taotao-manager | 后台管理系统，目前实现商品相关数据，广告位，广告内容的维护 |
| Taotao-rest | Restful风格发布常规服务接口，支持缓存 |
| Taotao-sso | 单点登陆系统，Restful风格发布与登陆注册功能相关的服务，支持缓存 |
| Taotao-order | 订单系统，Restful风格发布与订单功能相关的服务，支持缓存，目前仅实现创建订单功能 |
| Taotao-search | 搜索服务，Restful风格，通过SolrJ操作索引库，目前实现商品数据的搜索 |
| Taotao-portal | 前台门户系统，调用相关服务接口实现业务逻辑，展示首页广告数据 |
| 图片服务器 | nginx |

### 技术重点难点

#### **上传图片**

在后台管理系统中，允许管理员新增商品并批量上传图片，图片应上传到静态服务器，为此在Centos虚拟机上搭建nginx，使用其作为图片服务器，虚拟机开启vsftpd服务，通过ftp上传静态资源到服务器，服务器上的图片资源按日期存储。上传成功的图片将返回图片的url。

#### **商品规格参数实现**

商品规格参数具有如下特征：

一个商品分类对应一个规格参数模板，

一件具体的商品对应一种具体的规格参数值，

一个商品分类对应多个商品，

一个规格参数模板对应多个规格参数值。

这种复杂的对应关系，如果为每个实体建立一个数据表，那么当查询商品规格参数时，需要多表联合查询而且这种连接关系将异常复杂，而且表中元组数据冗余大。 最终，选择以json格式的数据来存储这种对应关系，即商品分类实体维护一个‘规格参数模板’字段，商品实体维护一个‘规格参数值’字段。这种情况下，只需读出json格式数据作相应解析便能实现功能。

#### 跨域调用问题，JSONP

在Taotao-portal工程中，实现首页商品分类详情展示：当用户焦点移至页面左侧分类栏目时，要异步调用taotao-rest发布的服务，请求相应数据。这种异步请求是使用js实现的，由浏览器向Taotao-rest发起请求。由于浏览器同源策略，一般来说位于 server1.example.com 的网页无法与不是 server1.example.com的服务器沟通。利用浏览器同源策略的漏洞，使用jsonp实现，在请求参数中携带回调方法，取得远程服务器返回的javascript，用 JavaScript 直译器执行而不是用 JSON 解析器解析。

#### CMS系统实现

商城首页需要添加广告位，并发布广告数据。考虑到一个广告位对应多条广告数据。

用ContentCategory实体来抽象表示广告位，Content实体来表示广告数据，ContentCategory的叶子节点表示具体的广告位，而ContentCategory对应的Content条目则表示相应广告位中的广告数据。这样一来逻辑关系就处理清楚。

#### 分布式缓存

项目中数个工程都需要添加缓存逻辑，搭建了Redis-Cluster实现分布式缓存，将Redis集群中16384个slot分为3组，每组2个redis node，实现一主一备。Redis的可基于内存存储数据特性决定了虽然它是single thread, 却有很强的并发处理能力。然而，本项目由于硬件资源问题，仅在虚拟机中开启多个redis实例来模拟分布式环境。

#### 缓存同步问题

当缓存中数据在持久层更新时，触发缓存同步，即删除redis相应的缓存项

#### 搭建索引库

在虚拟机上搭建Solr服务（单机版），将数据库中3K条商品数据导入Solr 库中，当搜索商品时直接访问索引库，不再访问数据库，实现更高效的检索性能。 当数据库商品更新时，要同步索引库。

#### 单点登录

传统的Web工程登录，往往是把用户信息序列化后保存在Session中，将SessionID设置在Cookie中返回给浏览器。 而在本系统中，由于系统采用分布式部署，若采用传统的开发方式，则需要将各服务器的session共享，才能确保单点登录功能。

在本项目中，每次用户登录成功，即生成一个唯一的token，以token为键，序列化后的用户信息为值保存在redis中，并设置身份信息有效时间，将redis作为多服务器共享的存储介质。往后浏览器访问服务器，cookie中带上token，只要token在服务器中过期前，均可读取到用户登录身份。

（注，测试发现，若禁用ie浏览器cookie，则无法登录淘宝网站）

#### 拦截非法访问用户中心的请求

使用SpringMVC的HandlerInterceptor拦截用户模块，比如订单，购物车，若验证用户未登录，则将用户请求重定向到单点登录服务器，登录成功后再回显目标页面（登录页面维持一个redirect属性，在完成登录逻辑后实现页面跳转）

#### Nginx反向代理与负载均衡

使用nginx的反向代理功能，作为内容服务器的替身，并依据需要实现负载均衡

### 不足之处

1. 系统采用伪分布式开发，与真实的分布式系统存在不可预见的差距
2. 测试数据不够多，就商品数据仅有3K条，数据分片效果体现不明显，Solr服务暂未实现集群
3. **伪分布式部署，无法测试当前系统并发处理能力**
4. 业务逻辑较粗糙，不够完善，比如可以在用户注册时添加发激活邮件环节，增设验证码。
5. 没有实现在线支付功能
6. 购物车条目基于cookie存储，这就决定了购物车条目数量问题（以firefox为例，cookie大小在4KB左右），还会受到浏览器状态本身限制。 （当当网的购物车模块:若用户未登录，则写cookie，若已登录，则持久化到数据库，若用户首次登录，则把当前cookie中数据写数据库，这点还是看具体需求）
7. 异常捕获模块 写的比较粗糙，没有友好详尽的业务逻辑。
8. 仅有一个数据库，没有实现集群
9. 电商网站的倒计时活动，秒杀活动，没有实现

秒杀方案猜想：将商品数量放到redis（single thread），秒杀开始则对商品数量-1，一旦返回数值为负 说明商品已售完