

淘淘商城

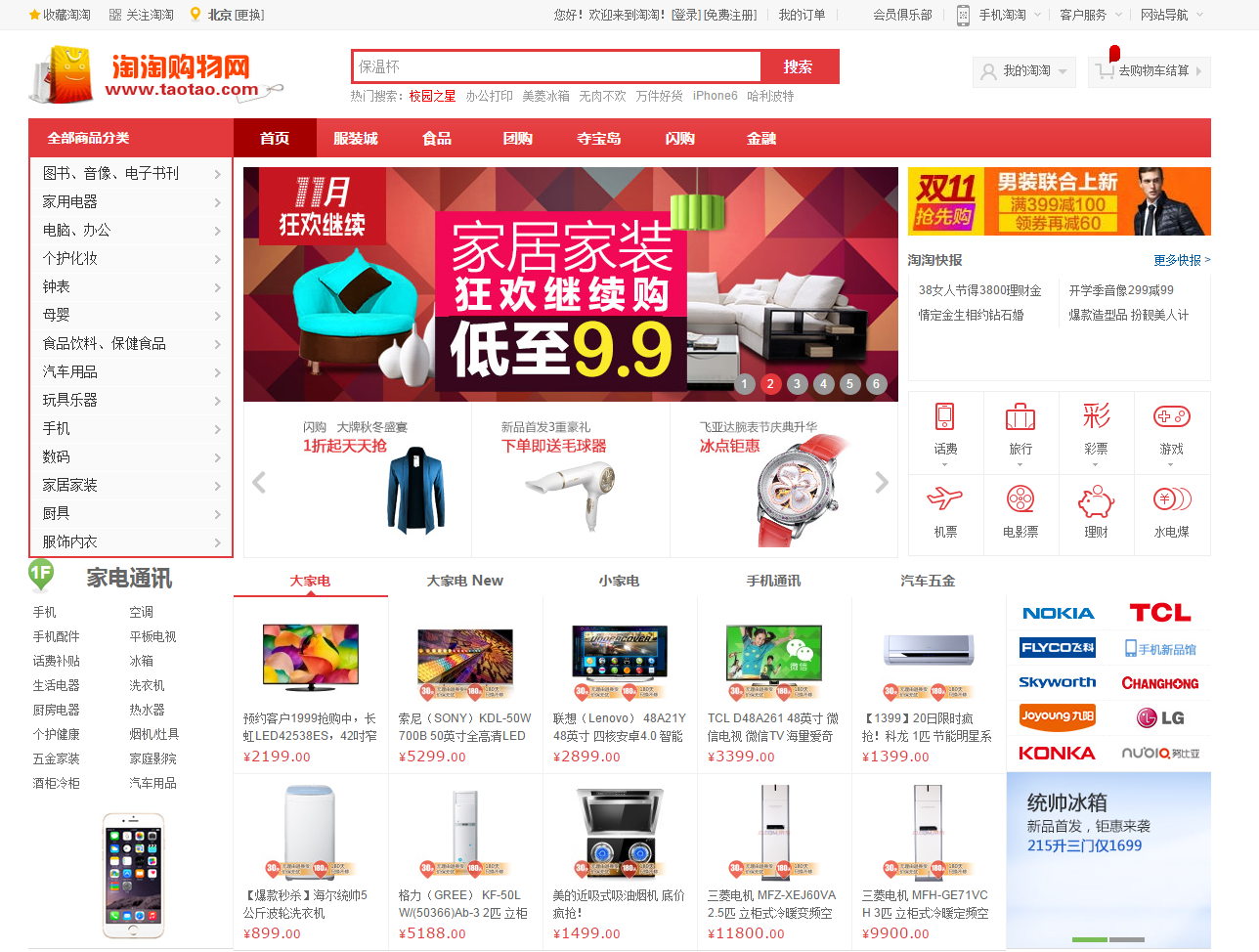
第五天

**讲师：传智.入云龙**

# 课程计划

1. 完成前台系统的搭建
2. 淘淘商城首页的实现
3. 商品分类展示

首页图片：



分类展示图片：



# 前台系统系统架构



在互联网系统开发当中，我们一般都是采用了分层的方式来架构系统，但是为什么我们需要分层进行架构呢？

采用分层架构有利于系统的维护，系统的扩展。这其实就是系统的可维护性和可扩展性。分层就是按照功能把系统切分细分，细分之后就能分布式部署，就能引入伸缩性，就能提高性能。

好处：

1. 基于soa理念将服务层抽出对外提供服务
2. 可以实现灵活的分布式部署

# 搭建服务系统

服务形式：对外提供rest形式的服务，供其他系统调用。使用http协议传递json数据。

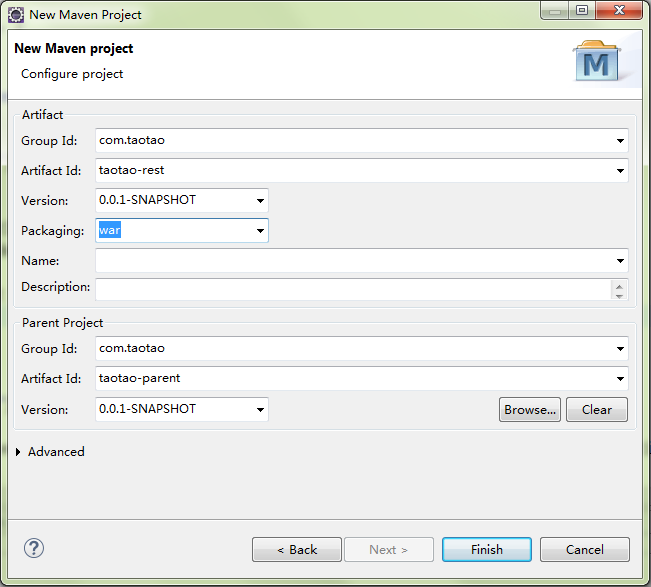
## 使用的技术

1、Mybatis

2、spring

3、springmvc

## 创建maven工程



## 添加一个web.xml文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  id=*"taotao"* version=*"2.5"*>  <display-name>taotao-rest</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  </welcome-file-list>    </web-app> |

## Pom文件

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-rest</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <!-- 依赖taotao-manager-mapper工程 -->  <dependency>  <groupId>com.taotao</groupId>  <artifactId>taotao-manager-mapper</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <!-- MySql -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  </dependency>  <!-- 连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  </dependency>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <scope>provided</scope>  </dependency>  </dependencies>  <build>  <plugins>  <!-- 配置Tomcat插件 -->  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8081</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

## 整合ssm

参考taotao-manager的整合过程。

# 搭建门户系统

## 什么是门户？

广义上的门户就是将各种应用系统、数据资源和互联网资源集成到一个信息管理平台之上，并以统一的用户界面提供给用户，并建立企业对客户、企业对内部员工和企业对企业的信息通道。

**简单来说就是网站的入口。**

## 所使用技术

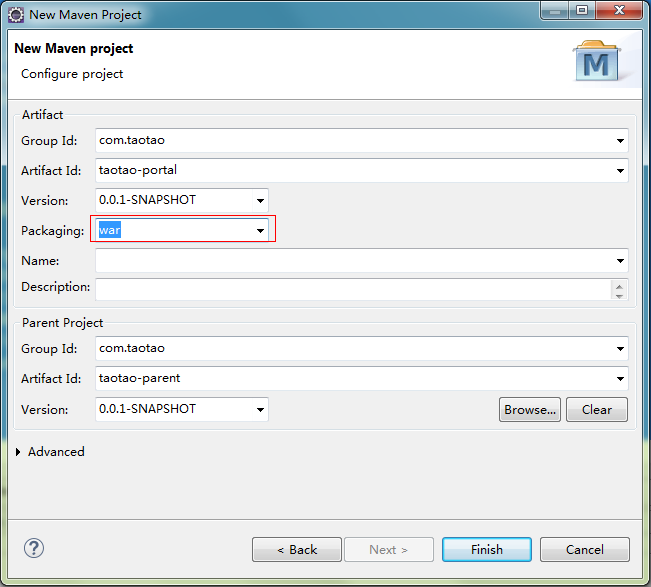
Srping + SpringMVC

JS + CSS

门户系统不直接调用数据库，而是通过服务系统提供的接口获取数据。

电商、互联网行业开发都是面向服务开发。

## 创建maven工程



## Pom文件

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-parent</artifactId>  <version>0.0.1-SNAPSHOT</version>  </parent>  <groupId>com.taotao</groupId>  <artifactId>taotao-portal</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>com.taotao</groupId>  <artifactId>taotao-common</artifactId>  <version>0.0.1-SNAPSHOT</version>  </dependency>  <!-- Spring -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  </dependency>  <!-- JSP相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jsp-api</artifactId>  <scope>provided</scope>  </dependency>  <!-- 文件上传组件 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  </dependency>  <!-- Redis客户端 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  </dependency>  <!-- solr客户端 -->  <dependency>  <groupId>org.apache.solr</groupId>  <artifactId>solr-solrj</artifactId>  </dependency>    </dependencies>  <build>  <plugins>  <!-- 配置Tomcat插件 -->  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <configuration>  <port>8082</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

## 配置文件

### Web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  id=*"taotao"* version=*"2.5"*>  <display-name>taotao-portal</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  </welcome-file-list>  <!-- 加载spring容器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/applicationContext\*.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  <!-- <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param> -->  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- springmvc的前端控制器 -->  <servlet>  <servlet-name>taotao-protal</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- contextConfigLocation不是必须的， 如果不配置contextConfigLocation， springmvc的配置文件默认在：WEB-INF/servlet的name+"-servlet.xml" -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/springmvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>taotao-protal</servlet-name>  <!-- 做伪静态，做搜索引擎优化（SEO） -->  <url-pattern>\*.html</url-pattern>  </servlet-mapping>  </web-app> |

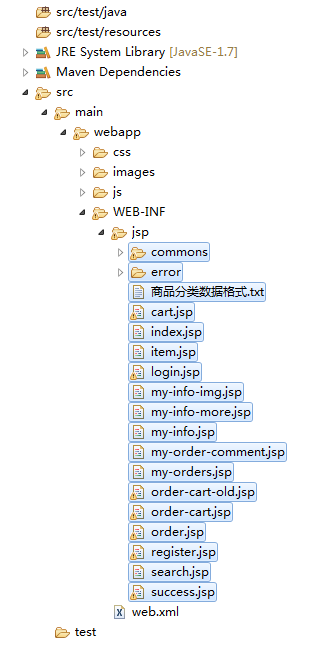
### Springmvc.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"*>    <context:component-scan base-package=*"com.taotao.portal.controller"*></context:component-scan>  <mvc:annotation-driven/>  <bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>    </beans> |

### applicationContext-service.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>    <!-- 加载配置文件 -->  <context:property-placeholder location=*"classpath:resource/\*.properties"*/>    <context:component-scan base-package=*"com.taotao.portal.service"*/>    </beans> |

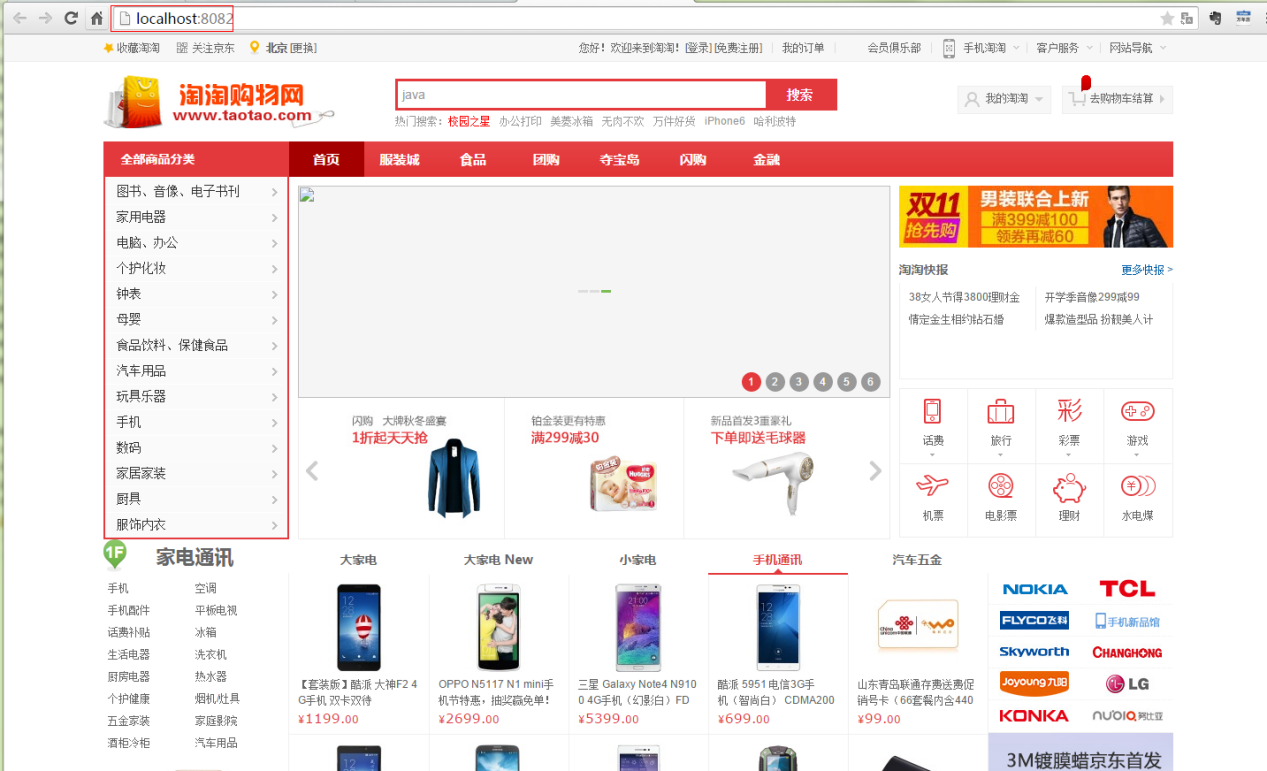
## 添加静态页面



## 显示欢迎页Controller

|  |
| --- |
| @Controller  **public** **class** PageController {  @RequestMapping("/index")  **public** String showIndex() **throws** Exception {  **return** "index";  }    } |

## 启动portal系统



# 首页商品类目展示流程

Taotao-portal

Taotao-rest服务层

1、请求首页

index.html

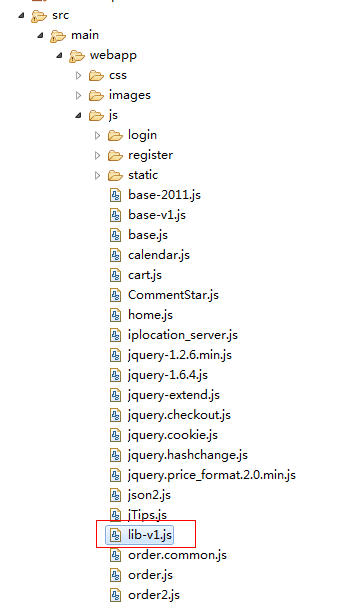
2、Ajax调用服务查询商品分类列表，返回json数据

浏览器

# Taotao-rest发布服务

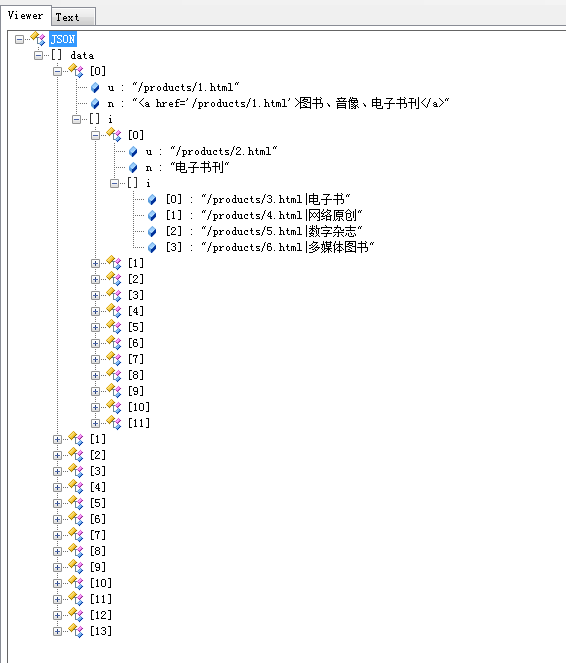
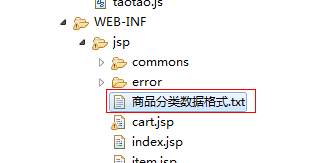
## 需求

### 请求的url



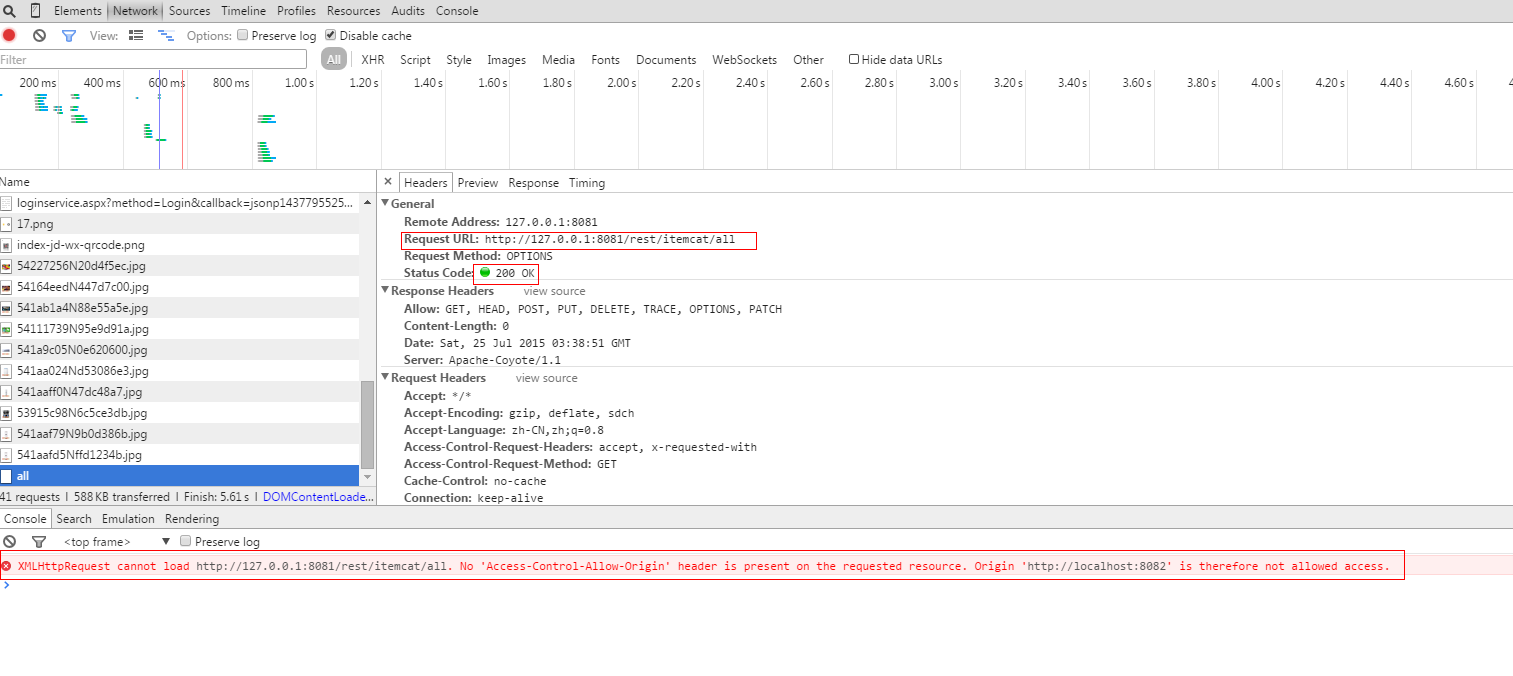
http://127.0.0.1:8081/rest/itemcat/all

### 响应的数据格式



## 跨域问题

使用json数据测试。如果ajax请求的是同一个工程中taotao-portal的json数据没有问题，可以直接显示出来。如果请求的是taotao-rest工程中json数据，会发生错误。



跨域问题：浏览器一个安全的限制，不允许js跨域请求资源，

[www.taotao.com](http://www.taotao.com/) 🡺 manage.taotao.com 跨域

[www.taotao.com](http://www.taotao.com/) 🡺 [www.taotao.com](http://www.taotao.com/) 非跨域

[www.taotao.com](http://www.taotao.com/) 🡺 [www.taotao.com](http://www.taotao.com/):8081 跨域

如何解决跨域问题：使用jsonp来解决跨域问题。

jsonp的原理：

浏览器在js请求中，是允许通过script标签的src跨域请求，可以在请求的结果中添加回调方法名，在请求页面中定义方法，既可获取到跨域请求的数据。

Taotao-portal

Taotao-rest

json

Ajax请求json数据

Ajax请求js

1、请求js数据

Js，其中包含json

json

js执行，调用js方法，要求此方法已经定义。

2、响应js

3、取出json

json

## 服务实现

### 对应数据格式创建pojo类：

|  |
| --- |
| **public** **class** ItemCat {  //转换成json数据时使用u作为key  @JsonProperty("u")  **private** String url;    @JsonProperty("n")  **private** String name;    @JsonProperty("i")  **private** List<?> item;  } |

### 返回值POJO：

|  |
| --- |
| /\*\*  \* 查询分类列表返回值  \* <p>Title: ItemCatResult</p>  \* <p>Description: </p>  \* <p>Company: www.itcast.com</p>  \* **@author** 入云龙  \* **@date** 2015年7月24日下午6:35:59  \* **@version** 1.0  \*/  **public** **class** ItemCatResult {    **private** List<?> data;  **public** List<?> getData() {  **return** data;  }  **public** **void** setData(List<?> data) {  **this**.data = data;  }    } |

### Service

|  |
| --- |
| @Service  **public** **class** ItemCatServiceImpl **implements** ItemCatService {  @Autowired  **private** TbItemCatMapper itemCatMapper;    @Override  **public** ItemCatResult queryAllCategory() **throws** Exception {    ItemCatResult result = **new** ItemCatResult();  result.setData(getItemCatList(0l));    **return** result;  }    /\*\*  \* 查询分类列表  \* <p>Title: getItemCatList</p>  \* <p>Description: </p>  \* **@param** parentid  \* **@return**  \*/  **private** List<?> getItemCatList(**long** parentid) {  TbItemCatExample example = **new** TbItemCatExample();  Criteria criteria = example.createCriteria();  //查询parentid为0的分类信息  criteria.andParentIdEqualTo(parentid);  List<TbItemCat> list = itemCatMapper.selectByExample(example);  List dataList = **new** ArrayList();  **for** (TbItemCat tbItemCat : list) {  //判断是否为父节点  **if** (tbItemCat.getIsParent()) {  ItemCat itemCat = **new** ItemCat();  itemCat.setUrl("/category/" + tbItemCat.getId() + ".html");  itemCat.setName(tbItemCat.getName());  //递归调用  itemCat.setItem(getItemCatList(tbItemCat.getId()));  //添加到列表  dataList.add(itemCat);  } **else** {  String catItem = "/item/" + tbItemCat.getId() + ".html|" + tbItemCat.getName();  dataList.add(catItem);  }  }  **return** dataList;  }  } |

### Controller

要返回json数据，还需要使用回调方法把json数据包装起来。所以需要controller添加回调支持，不能直接返回一个ItemCatResult对象。

#### 方法一：

使用MappingJacksonValue对象包装返回结果，并设置jsonp的回调方法。

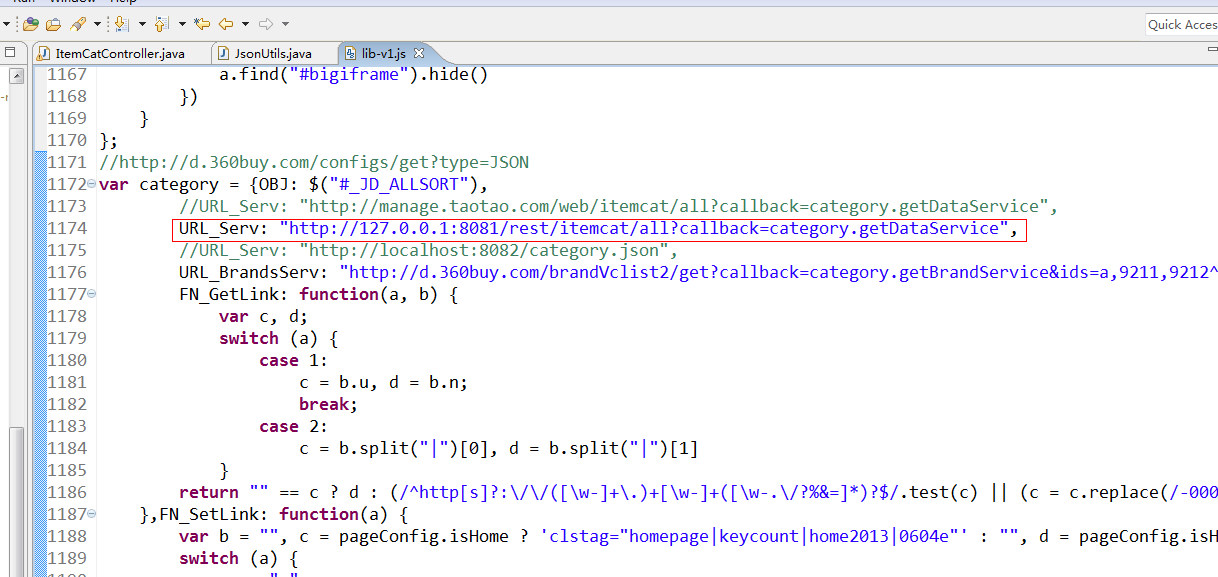
|  |
| --- |
| @RequestMapping("/all")  @ResponseBody  **public** MappingJacksonValue queryAll(String callback) **throws** Exception {  //查询分类列表  ItemCatResult result = itemCatService.queryAllCategory();  //包装jsonp  MappingJacksonValue jacksonValue = **new** MappingJacksonValue(result);  //设置包装的回调方法名  jacksonValue.setJsonpFunction(callback);    **return** jacksonValue;  } |

#### 方法二

先把ItemCatResult对象转换成json字符串，然后使用字符串拼接的方法拼装成jsonp格式的数据。需要设置相应结果的MediaType。

|  |
| --- |
| @RequestMapping(value="/all", produces=MediaType.***APPLICATION\_JSON\_VALUE*** + ";charset=utf-8")  @ResponseBody  **public** String queryAll(String callback) **throws** Exception {  //查询分类列表  ItemCatResult result = itemCatService.queryAllCategory();  //把对象转换成json数据  String jsonResult = JsonUtils.*objectToJson*(result);  //拼接字符串  String resultStr = callback + "(" + jsonResult + ");";    **return** resultStr;  } |

## 页面实现



页面效果：



# 附录:

Maven执行跳过测试：

mvn clean install -DskipTests