# 第十一单元 豪歌商城项目概述与工程搭建

# 【授课重点】

1. 项目概述及电商模式
2. 系统架构及功能模块
3. 数据库表设计
4. 创建hgshop-parent聚合工程
5. 创建hgshop-goods-interface接口工程
6. 创建hgshop-goods-service商品服务工程
7. 创建hgshop-manage后台工程

# 【考核要求】

1. 掌握系统架构及功能模块
2. 掌握数据库表设计
3. 能够独立搭建maven聚合工程

# 【教学内容】

## 11.1 项目概述及电商模式

### 11.1.1 项目概述

豪歌商城是一个综合性的B2C平台，类似京东商城、天猫商城。会员可以在商城浏览商品、下订 单，以及参加各种活动。

管理员可以在平台后台管理系统中管理商品、会员、订单等。

### 11.1.2 电商模式

B2B：商家到商家。阿里巴巴，慧聪网、铭万网。

B2C：商家到用户。京东。

C2C：用户到用户。淘宝。

B2B2C：商家到商家到用户。天猫。

O2O：线上到线下。美团、饿了么。

## 11.2 系统功能及模块

### 11.2.1 系统架构

### 11.2.2 功能模块

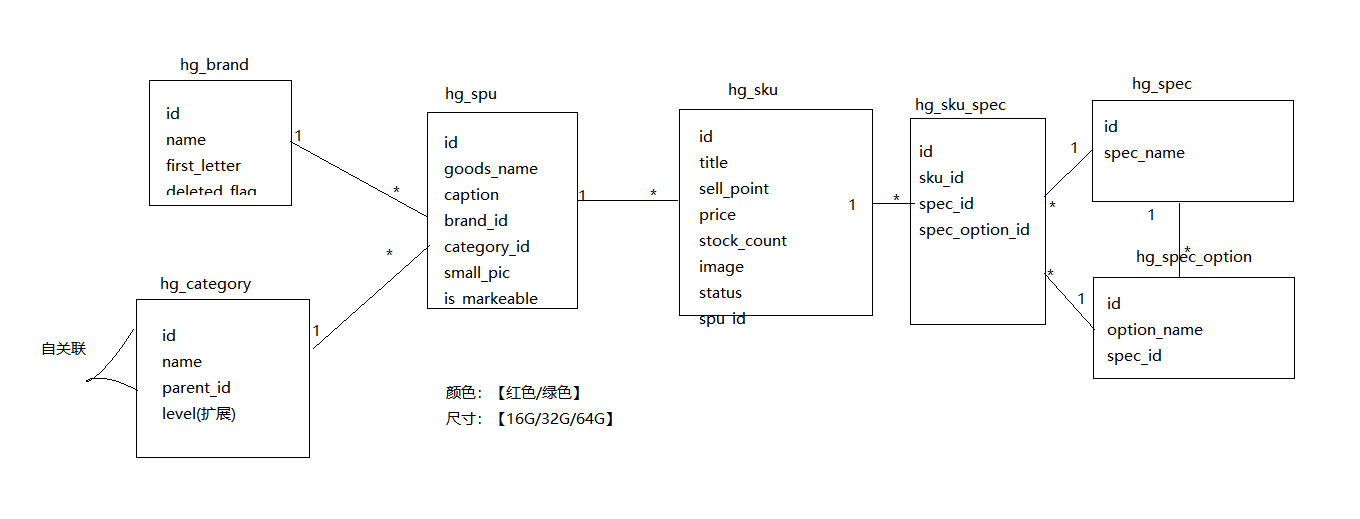
#### 11.2.2.1 后台系统

管理员登录模块、品牌管理模块、商品分类模块、商品模块(spu/sku)、规格参数、用户管理模块

#### 11.2.2.2 前台系统

首页、商品列表页面、商品详情页、用户注册、用户登录、购物车模块、订单模块

## 11.3 数据库表设计



### spu

stand product unit (标准商品单元)

华为P30 p40 小米：R11

## sku

标准库存单元。 用户购买的具体产品，一个sku 就是一个条形码。

## 11.4 创建hgshop-parent工程

首先创建maven聚合工程，打包方式为pom.

### 11.4.1 pom.xml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>  
 <groupId>com.bw.hgshop</groupId>  
 <artifactId>hgshop-parent</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>pom</packaging>  
   
 <!-- 定义主要版本号 -->  
 <properties>  
 <spring.version>5.1.5.RELEASE</spring.version>  
 <mybatis.version>3.4.2</mybatis.version>  
 <mybatis.spring.version>1.3.0</mybatis.spring.version>  
 <pagehelper-version>5.1.2</pagehelper-version>  
 <druid.version>1.0.9</druid.version>  
 <mysql.version>5.1.6</mysql.version>  
 <aspectj-version>1.8.0</aspectj-version>  
   
 <dubbo.version>2.7.3</dubbo.version>  
   
 <log4j.version>1.2.17</log4j.version>  
 <jackson.version>2.9.1</jackson.version>  
 <jstl.version>1.2</jstl.version>  
 <servlet-api.version>2.5</servlet-api.version>  
 <jsp-api.version>2.2</jsp-api.version>  
   
 <commons-lang3.version>3.3.2</commons-lang3.version>  
 <commons-io.version>1.3.1</commons-io.version>  
 <commons-net.version>3.3</commons-net.version>  
 <commons-fileupload.version>1.3.1</commons-fileupload.version>  
   
 <junit-version>4.12</junit-version>  
   
 <ik.version>2012\_u6</ik.version>  
 <lucene.version>7.2.1</lucene.version>  
 <spring-data-elasticsearch.version>3.1.5.RELEASE</spring-data-elasticsearch.version>  
 <!-- <kafka.version>0.8.2.1</kafka.version> -->  
 <kafka.version>2.1.0</kafka.version>  
 <spring-kafka.version>2.2.4.RELEASE</spring-kafka.version>  
 <jedis.version>2.9.0</jedis.version>  
 <spring-data-redis.version>2.1.5.RELEASE</spring-data-redis.version>  
 </properties>  
  
 <dependencyManagement>  
  
 <dependencies>  
 <!-- 引入dubbo的依赖配置 --  
 <dependency>  
 <groupId>org.apache.dubbo</groupId>  
 <artifactId>dubbo</artifactId>  
 <version>${dubbo.version}</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>spring</artifactId>  
 <groupId>org.springframework</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.dubbo</groupId>  
 <artifactId>dubbo-dependencies-zookeeper</artifactId>  
 <version>${dubbo.version}</version>  
 <type>pom</type>  
 </dependency>  
 <!-- spring-webmvc依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <!-- spring-jdbc依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <!-- spring-tx 事务依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
 <!-- 引入spring-test依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 <version>${spring.version}</version>  
 </dependency>  
  
 <!-- mybatis核心包 -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 <version>${mybatis.version}</version>  
 </dependency>  
 <!-- mybatis-spring 整合jar -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 <version>${mybatis.spring.version}</version>  
 </dependency>  
 <!-- druid数据源 -->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>${druid.version}</version>  
 </dependency>  
 <!-- Mysql数据库驱动包 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>${mysql.version}</version>  
 </dependency>  
 <!-- 日志文件管理包 -->  
 <!-- log start -->  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 <version>${log4j.version}</version>  
 </dependency>  
 <!-- 单元测试 -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>${junit-version}</version>  
 <scope>test</scope>  
 </dependency>  
 <!-- 上传组件包 -->  
 <dependency>  
 <groupId>commons-fileupload</groupId>  
 <artifactId>commons-fileupload</artifactId>  
 <version>${commons-fileupload.version}</version>  
 </dependency>  
 <!-- common-io依赖 -->  
 <dependency>  
 <groupId>commons-io</groupId>  
 <artifactId>commons-io</artifactId>  
 <version>${commons-io.version}</version>  
 </dependency>  
 <!-- 引入jstl依赖 -->  
 <dependency>  
 <groupId>jstl</groupId>  
 <artifactId>jstl</artifactId>  
 <version>${jstl.version}</version>  
 </dependency>  
 <!-- 引入jsp-api依赖 -->  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <version>${jsp-api.version}</version>  
 <scope>provided</scope>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>${servlet-api.version}</version>  
 <scope>provided</scope>  
 </dependency>  
  
 <!-- 引入jackson的依赖 -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 <version>${jackson.version}</version>  
 </dependency>  
  
 <!-- 依赖的公共包 -->  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 <version>${commons-lang3.version}</version>  
 </dependency>  
  
 <!-- 引入org.aspectj依赖 -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 <version>${aspectj-version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 <version>${aspectj-version}</version>  
 </dependency>  
 <!-- 引入mybaits pagehelper分页助手依赖 -->  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 <version>${pagehelper-version}</version>  
 </dependency>  
   
 <dependency>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-core</artifactId>  
 <version>${lucene.version}</version>  
 </dependency>  
 <!-- 加入ik分词器 -->  
 <dependency>  
 <groupId>com.janeluo</groupId>  
 <artifactId>ikanalyzer</artifactId>  
 <version>${ik.version}</version>  
 <!-- 排除自带的lucene操作 -->  
 <exclusions>  
 <exclusion>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-queries</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-queryparser</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-elasticsearch</artifactId>  
 <version>${spring-data-elasticsearch.version}</version>  
 </dependency>  
   
 <dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 <version>${jedis.version}</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 <version>${spring-data-redis.version}</version>  
 </dependency>  
   
 <dependency>  
 <groupId>org.apache.kafka</groupId>  
 <artifactId>kafka\_2.12</artifactId>  
 <version>${kafka.version}</version>  
 <exclusions>  
 <exclusion>  
 <artifactId>jmxri</artifactId>  
 <groupId>com.sun.jmx</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>jms</artifactId>  
 <groupId>javax.jms</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>jmxtools</artifactId>  
 <groupId>com.sun.jdmk</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.kafka</groupId>  
 <artifactId>spring-kafka</artifactId>  
 <version>${spring-kafka.version}</version>  
 </dependency>  
   
 </dependencies>  
 </dependencyManagement>  
  
 <build>  
 <pluginManagement>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.tomcat.maven</groupId>  
 <artifactId>tomcat7-maven-plugin</artifactId>  
 <version>2.2</version>  
 </plugin>  
 </plugins>  
 </pluginManagement>  
   
 </build>  
 <modules>  
 <module>hgshop-goods-interface</module>  
 <module>hgshop-goods-service</module>  
 <module>hgshop-manage</module>  
 <module>hgshop-web</module>  
 </modules>  
</project>

## 11.5 创建hgshop-goods-interface工程

首先创建商品接口模块，打包方式为jar.

### 11.5.1 pom.xml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.bw.hgshop</groupId>  
 <artifactId>hgshop-parent</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 </parent>  
 <artifactId>hgshop-goods-interface</artifactId>  
   
 <dependencies>  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 </dependency>  
   
 <dependency>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-core</artifactId>  
 </dependency>  
  
 <!-- 加入ik分词器 -->  
 <dependency>  
 <groupId>com.janeluo</groupId>  
 <artifactId>ikanalyzer</artifactId>  
 <!-- 排除自带的lucene操作 -->  
 <exclusions>  
 <exclusion>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-queries</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-queryparser</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-elasticsearch</artifactId>  
 </dependency>  
 </dependencies>  
   
</project>

### 11.5.2 实体类

#### 11.5.2.1 品牌

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
  
/\*\*  
 \* 品牌实体类  
 \* @author coolface  
 \*  
 \*/  
public class Brand implements Serializable {  
  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
 private Integer id; //id  
 private String name; //品牌名称  
 private String firstChar; //品牌首字母  
 private Integer deletedFlag; //伪删除标识  
 public Brand() {  
 super();  
 }  
 public Brand(Integer id, String name, String firstChar, Integer deletedFlag) {  
 super();  
 this.id = id;  
 this.name = name;  
 this.firstChar = firstChar;  
 this.deletedFlag = deletedFlag;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public String getFirstChar() {  
 return firstChar;  
 }  
 public void setFirstChar(String firstChar) {  
 this.firstChar = firstChar;  
 }  
 public Integer getDeletedFlag() {  
 return deletedFlag;  
 }  
 public void setDeletedFlag(Integer deletedFlag) {  
 this.deletedFlag = deletedFlag;  
 }  
 @Override  
 public String toString() {  
 return "Brand [id=" + id + ", name=" + name + ", firstChar=" + firstChar + ", deletedFlag=" + deletedFlag + "]";  
 }  
   
   
}

#### 11.5.2.2 分类

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
import java.util.List;  
  
import com.fasterxml.jackson.annotation.JsonProperty;  
  
/\*\*  
 \* 商品分类实体类  
 \* @author coolface  
 \*  
 \*/  
public class Category implements Serializable {  
  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
   
 private Integer id; //id  
 @JsonProperty("text")  
 private String name; //分类名称  
 private Integer parentId; //父分类id  
 @JsonProperty("nodes")  
 private List<Category> childs; //子分类列表  
 private String parentName; //父分类名称  
   
 //用于在使用treeview时,控制一级二级不能选中  
 private boolean selectable = true;  
   
   
 public Category() {  
 super();  
 // TODO Auto-generated constructor stub  
 }  
 public Category(Integer id, String name, Integer parentId) {  
 super();  
 this.id = id;  
 this.name = name;  
 this.parentId = parentId;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getName() {  
 return name;  
 }  
 public void setName(String name) {  
 this.name = name;  
 }  
 public Integer getParentId() {  
 return parentId;  
 }  
 public void setParentId(Integer parentId) {  
 this.parentId = parentId;  
 }  
   
 public List<Category> getChilds() {  
 return childs;  
 }  
 public void setChilds(List<Category> childs) {  
 this.childs = childs;  
 }  
   
 public String getParentName() {  
 return parentName;  
 }  
 public void setParentName(String parentName) {  
 this.parentName = parentName;  
 }  
   
   
 public boolean isSelectable() {  
 return selectable;  
 }  
 public void setSelectable(boolean selectable) {  
 this.selectable = selectable;  
 }  
 @Override  
 public String toString() {  
 return "Category [id=" + id + ", name=" + name + ", parentId=" + parentId + ", childs=" + childs  
 + ", parentName=" + parentName + ", selectable=" + selectable + "]";  
 }  
}

#### 11.5.2.3 规格参数

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
import java.util.List;  
  
/\*\*  
 \* 规格参数实体类  
 \* @author coolface  
 \*  
 \*/  
public class Spec implements Serializable {  
  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
  
 private Integer id; //id  
 private String specName; //规格参数名称  
 private List<SpecOption> options; //规格参数选项列表  
   
 private String optionNames;  
   
 public Spec() {  
 super();  
 // TODO Auto-generated constructor stub  
 }  
 public Spec(Integer id, String specName, List<SpecOption> options) {  
 super();  
 this.id = id;  
 this.specName = specName;  
 this.options = options;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getSpecName() {  
 return specName;  
 }  
 public void setSpecName(String specName) {  
 this.specName = specName;  
 }  
 public List<SpecOption> getOptions() {  
 return options;  
 }  
 public void setOptions(List<SpecOption> options) {  
 this.options = options;  
 }  
 public String getOptionNames() {  
 return optionNames;  
 }  
 public void setOptionNames(String optionNames) {  
 this.optionNames = optionNames;  
 }  
 @Override  
 public String toString() {  
 return "Spec [id=" + id + ", specName=" + specName + ", options=" + options + "]";  
 }   
}

#### 11.5.2.4 规格参数选项

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
  
/\*\*  
 \* 规格参数选项实体类  
 \* @author coolface  
 \*  
 \*/  
public class SpecOption implements Serializable {  
  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
  
   
 private Integer id; //id  
 private String optionName; //选项值  
 private Integer specId; //规格参数Id  
 private Integer orders; //排序字段  
 public SpecOption() {  
 super();  
 // TODO Auto-generated constructor stub  
 }  
 public SpecOption(Integer id, String optionName, Integer specId, Integer orders) {  
 super();  
 this.id = id;  
 this.optionName = optionName;  
 this.specId = specId;  
 this.orders = orders;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getOptionName() {  
 return optionName;  
 }  
 public void setOptionName(String optionName) {  
 this.optionName = optionName;  
 }  
 public Integer getSpecId() {  
 return specId;  
 }  
 public void setSpecId(Integer specId) {  
 this.specId = specId;  
 }  
 public Integer getOrders() {  
 return orders;  
 }  
 public void setOrders(Integer orders) {  
 this.orders = orders;  
 }  
 @Override  
 public String toString() {  
 return "SpecOption [id=" + id + ", optionName=" + optionName + ", specId=" + specId + ", orders=" + orders  
 + "]";  
 }   
}

#### 11.5.2.5 spu

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
import java.util.List;  
  
/\*\*  
 \* spu实体类  
 \* @author coolface  
 \*  
 \*/  
public class Spu implements Serializable {  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
 private Integer id; //id  
 private String goodsName; //商品名称  
 private String isMarketable; //是否上架 0:上架 1:下架  
 private Integer brandId; //品牌Id  
 private String caption; //副标题  
 private Integer categoryId; //分类Id  
 private String smallPic; //商品图片  
 //spu列表展示属性  
 private String bName;  
 private String cName;  
   
   
 private List<Sku> skuList;  
   
// private List<List<Integer>> optionIds;  
   
 private Integer optionId;  
   
 private Double startPrice;  
 private Double endPrice;  
   
   
 public Spu() {  
 super();  
 // TODO Auto-generated constructor stub  
 }  
 public Spu(Integer id, String goodsName, String isMarketable, Integer brandId, String caption, Integer categoryId,  
 String smallPic) {  
 super();  
 this.id = id;  
 this.goodsName = goodsName;  
 this.isMarketable = isMarketable;  
 this.brandId = brandId;  
 this.caption = caption;  
 this.categoryId = categoryId;  
 this.smallPic = smallPic;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getGoodsName() {  
 return goodsName;  
 }  
 public void setGoodsName(String goodsName) {  
 this.goodsName = goodsName;  
 }  
 public String getIsMarketable() {  
 return isMarketable;  
 }  
 public void setIsMarketable(String isMarketable) {  
 this.isMarketable = isMarketable;  
 }  
 public Integer getBrandId() {  
 return brandId;  
 }  
 public void setBrandId(Integer brandId) {  
 this.brandId = brandId;  
 }  
 public String getCaption() {  
 return caption;  
 }  
 public void setCaption(String caption) {  
 this.caption = caption;  
 }  
 public Integer getCategoryId() {  
 return categoryId;  
 }  
 public void setCategoryId(Integer categoryId) {  
 this.categoryId = categoryId;  
 }  
 public String getSmallPic() {  
 return smallPic;  
 }  
 public void setSmallPic(String smallPic) {  
 this.smallPic = smallPic;  
 }  
 public String getbName() {  
 return bName;  
 }  
 public void setbName(String bName) {  
 this.bName = bName;  
 }  
 public String getcName() {  
 return cName;  
 }  
 public void setcName(String cName) {  
 this.cName = cName;  
 }  
   
   
   
   
   
   
   
// public List<List<Integer>> getOptionIds() {  
// return optionIds;  
// }  
// public void setOptionIds(List<List<Integer>> optionIds) {  
// this.optionIds = optionIds;  
// }  
   
   
   
 public Double getStartPrice() {  
 return startPrice;  
 }  
 public Integer getOptionId() {  
 return optionId;  
 }  
 public void setOptionId(Integer optionId) {  
 this.optionId = optionId;  
 }  
 public void setStartPrice(Double startPrice) {  
 this.startPrice = startPrice;  
 }  
 public Double getEndPrice() {  
 return endPrice;  
 }  
 public void setEndPrice(Double endPrice) {  
 this.endPrice = endPrice;  
 }  
   
   
 public List<Sku> getSkuList() {  
 return skuList;  
 }  
 public void setSkuList(List<Sku> skuList) {  
 this.skuList = skuList;  
 }  
 @Override  
 public String toString() {  
 return "Spu [id=" + id + ", goodsName=" + goodsName + ", isMarketable=" + isMarketable + ", brandId=" + brandId  
 + ", caption=" + caption + ", categoryId=" + categoryId + ", smallPic=" + smallPic + "]";  
 }   
}

#### 11.5.2.6 sku

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
import java.util.Date;  
import java.util.List;  
  
/\*\*  
 \* sku实体类  
 \* @author coolface  
 \*  
 \*/  
public class Sku implements Serializable {  
  
 /\*\*  
 \*   
 \*/  
 private static final long serialVersionUID = 1L;  
  
 private Integer id; //id  
 private String title; //标题  
 private String sellPoint; //卖点  
 private Integer price; //价格  
 private Integer stockCount; //库存  
 private String barcode; //条形码  
 private String image; //图片  
 private String status; //状态 0:上架 1:下架  
 private Date createTime; //创建时间  
 private Date updateTime; //修改时间  
 private Integer costPrice; //成本价格  
 private Integer marketPrice; //市场价格  
 private Integer spuId; //spuId  
 private String cartThumbnail; //购物车页面展示的缩略图  
   
 private List<SkuSpec> skuSpec; //sku对应的规格参数列表  
   
   
   
 public Sku() {  
 super();  
 // TODO Auto-generated constructor stub  
 }  
 public Sku(Integer id, String title, String sellPoint, Integer price, Integer stockCount, String barcode,  
 String image, String status, Date createTime, Date updateTime, Integer costPrice, Integer marketPrice,  
 Integer spuId, String cartThumbnail) {  
 super();  
 this.id = id;  
 this.title = title;  
 this.sellPoint = sellPoint;  
 this.price = price;  
 this.stockCount = stockCount;  
 this.barcode = barcode;  
 this.image = image;  
 this.status = status;  
 this.createTime = createTime;  
 this.updateTime = updateTime;  
 this.costPrice = costPrice;  
 this.marketPrice = marketPrice;  
 this.spuId = spuId;  
 this.cartThumbnail = cartThumbnail;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public String getTitle() {  
 return title;  
 }  
 public void setTitle(String title) {  
 this.title = title;  
 }  
 public String getSellPoint() {  
 return sellPoint;  
 }  
 public void setSellPoint(String sellPoint) {  
 this.sellPoint = sellPoint;  
 }  
 public Integer getPrice() {  
 return price;  
 }  
 public void setPrice(Integer price) {  
 this.price = price;  
 }  
 public Integer getStockCount() {  
 return stockCount;  
 }  
 public void setStockCount(Integer stockCount) {  
 this.stockCount = stockCount;  
 }  
 public String getBarcode() {  
 return barcode;  
 }  
 public void setBarcode(String barcode) {  
 this.barcode = barcode;  
 }  
 public String getImage() {  
 return image;  
 }  
 public void setImage(String image) {  
 this.image = image;  
 }  
 public String getStatus() {  
 return status;  
 }  
 public void setStatus(String status) {  
 this.status = status;  
 }  
 public Date getCreateTime() {  
 return createTime;  
 }  
 public void setCreateTime(Date createTime) {  
 this.createTime = createTime;  
 }  
 public Date getUpdateTime() {  
 return updateTime;  
 }  
 public void setUpdateTime(Date updateTime) {  
 this.updateTime = updateTime;  
 }  
 public Integer getCostPrice() {  
 return costPrice;  
 }  
 public void setCostPrice(Integer costPrice) {  
 this.costPrice = costPrice;  
 }  
 public Integer getMarketPrice() {  
 return marketPrice;  
 }  
 public void setMarketPrice(Integer marketPrice) {  
 this.marketPrice = marketPrice;  
 }  
 public Integer getSpuId() {  
 return spuId;  
 }  
 public void setSpuId(Integer spuId) {  
 this.spuId = spuId;  
 }  
 public String getCartThumbnail() {  
 return cartThumbnail;  
 }  
 public void setCartThumbnail(String cartThumbnail) {  
 this.cartThumbnail = cartThumbnail;  
 }  
   
   
 public List<SkuSpec> getSkuSpec() {  
 return skuSpec;  
 }  
 public void setSkuSpec(List<SkuSpec> skuSpec) {  
 this.skuSpec = skuSpec;  
 }  
   
   
 @Override  
 public String toString() {  
 return "Sku [id=" + id + ", title=" + title + ", sellPoint=" + sellPoint + ", price=" + price + ", stockCount="  
 + stockCount + ", barcode=" + barcode + ", image=" + image + ", status=" + status + ", createTime="  
 + createTime + ", updateTime=" + updateTime + ", costPrice=" + costPrice + ", marketPrice="  
 + marketPrice + ", spuId=" + spuId + ", cartThumbnail=" + cartThumbnail + "]";  
 }  
   
   
}

#### 11.5.2.7 sku与规格参数的中间表

package com.bawei.hgshop.pojo;  
  
import java.io.Serializable;  
  
/\*\*  
 \* sku与规格参数关联的中间表  
 \* @author coolface  
 \*  
 \*/  
public class SkuSpec implements Serializable {  
  
 private static final long serialVersionUID = 1L;  
 private Integer id;  
 private Integer skuId; //sku的id  
 private Integer specId; //spec的id  
 private Integer specOptionId; //spec\_option的id  
   
 public SkuSpec() {  
 super();  
 }  
 public SkuSpec(Integer id, Integer skuId, Integer specId, Integer specOptionId) {  
 super();  
 this.id = id;  
 this.skuId = skuId;  
 this.specId = specId;  
 this.specOptionId = specOptionId;  
 }  
 public Integer getId() {  
 return id;  
 }  
 public void setId(Integer id) {  
 this.id = id;  
 }  
 public Integer getSkuId() {  
 return skuId;  
 }  
 public void setSkuId(Integer skuId) {  
 this.skuId = skuId;  
 }  
 public Integer getSpecId() {  
 return specId;  
 }  
 public void setSpecId(Integer specId) {  
 this.specId = specId;  
 }  
 public Integer getSpecOptionId() {  
 return specOptionId;  
 }  
 public void setSpecOptionId(Integer specOptionId) {  
 this.specOptionId = specOptionId;  
 }  
   
 @Override  
 public String toString() {  
 return "SkuSpec [id=" + id + ", skuId=" + skuId + ", specId=" + specId + ", specOptionId=" + specOptionId + "]";  
 }   
}

## 11.6 创建hgshop-goods-service工程

创建hgshop-goods-service服务模块，打成war包。

### 11.6.1 pom.xml文件加入依赖

注意：

1. **加入对应的依赖关系 hgshop-goods-interface**
2. **添加tomcat启动的端口号8081**

* 具体的pom.xml文件如下

<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.bw.hgshop</groupId>  
 <artifactId>hgshop-parent</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 </parent>  
 <artifactId>hgshop-goods-service</artifactId>  
 <packaging>war</packaging>  
   
 <dependencies>  
 <!-- https://mvnrepository.com/artifact/commons-beanutils/commons-beanutils -->  
 <dependency>  
 <groupId>commons-beanutils</groupId>  
 <artifactId>commons-beanutils</artifactId>  
 <version>1.9.4</version>  
 </dependency>  
   
 <dependency>  
 <groupId>com.bw.hgshop</groupId>  
 <artifactId>hgshop-goods-interface</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 </dependency>  
 <!-- 引入dubbo的依赖配置 -->  
 <dependency>  
 <groupId>org.apache.dubbo</groupId>  
 <artifactId>dubbo</artifactId>  
 <exclusions>  
 <exclusion>  
 <artifactId>spring-context</artifactId>  
 <groupId>org.springframework</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>netty-all</artifactId>  
 <groupId>io.netty</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.dubbo</groupId>  
 <artifactId>dubbo-dependencies-zookeeper</artifactId>  
 <type>pom</type>  
 </dependency>  
 <!-- spring-webmvc依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <exclusions>  
 <exclusion>  
 <artifactId>spring-context</artifactId>  
 <groupId>org.springframework</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 </dependency>  
 <!-- spring-jdbc依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jdbc</artifactId>  
 </dependency>  
 <!-- spring-tx 事务依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-tx</artifactId>  
 </dependency>  
 <!-- 引入spring-test依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-test</artifactId>  
 </dependency>  
  
 <!-- mybatis核心包 -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis</artifactId>  
 </dependency>  
 <!-- mybatis-spring 整合jar -->  
 <dependency>  
 <groupId>org.mybatis</groupId>  
 <artifactId>mybatis-spring</artifactId>  
 </dependency>  
 <!-- druid数据源 -->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 </dependency>  
 <!-- Mysql数据库驱动包 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 </dependency>  
 <!-- 日志文件管理包 -->  
 <!-- log start -->  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 </dependency>  
 <!-- 单元测试 -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <!-- 依赖的公共包 -->  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 </dependency>  
  
 <!-- 引入org.aspectj依赖 -->  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjweaver</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.aspectj</groupId>  
 <artifactId>aspectjrt</artifactId>  
 </dependency>  
 <!-- 引入mybaits pagehelper分页助手依赖 -->  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-core</artifactId>  
 </dependency>  
  
 <!-- 加入ik分词器 -->  
 <dependency>  
 <groupId>com.janeluo</groupId>  
 <artifactId>ikanalyzer</artifactId>  
 <!-- 排除自带的lucene操作 -->  
 <exclusions>  
 <exclusion>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-queries</artifactId>  
 </exclusion>  
 <exclusion>  
 <groupId>org.apache.lucene</groupId>  
 <artifactId>lucene-queryparser</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-elasticsearch</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>redis.clients</groupId>  
 <artifactId>jedis</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.data</groupId>  
 <artifactId>spring-data-redis</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.apache.kafka</groupId>  
 <artifactId>kafka\_2.12</artifactId>  
 <exclusions>  
 <exclusion>  
 <artifactId>jmxri</artifactId>  
 <groupId>com.sun.jmx</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>jms</artifactId>  
 <groupId>javax.jms</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>jmxtools</artifactId>  
 <groupId>com.sun.jdmk</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.kafka</groupId>  
 <artifactId>spring-kafka</artifactId>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.tomcat.maven</groupId>  
 <artifactId>tomcat7-maven-plugin</artifactId>  
 <configuration>  
 <port>8081</port>  
 <path>/</path>  
 <uriEncoding>UTF-8</uriEncoding>  
 <contextReloadable>true</contextReloadable>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

### 11.6.2 配置文件

#### 11.6.2.1 db.properties

jdbc.driver=com.mysql.jdbc.Driver  
jdbc.url=jdbc:mysql://localhost:3306/hg\_shop?useUnicode=true&characterEncoding=UTF8  
jdbc.username=root  
jdbc.password=root

#### 11.6.2.2 log4j.properties

# fatal error warning info debug trace   
log4j.rootLogger=warn, A1  
log4j.appender.A1=org.apache.log4j.ConsoleAppender  
log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
#log4j.appender.A1.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n  
  
log4j.logger.com.mmcro.woker=debug,MYTESt  
log4j.appender.MYTESt=org.apache.log4j.ConsoleAppender  
log4j.appender.MYTESt.layout=org.apache.log4j.PatternLayout

#### 11.6.2.2--2

创建文件log4j2.xml

<configuration monitorInterval="60">  
<Appenders>  
 <Console name="Console" target="SYSTEM\_OUT">  
 <PatternLayout pattern="%-4r [%t] %-5p %c{1.} - %msg%n"/>  
 </Console>  
</Appenders>  
<Loggers>  
 <Logger name="org.apache.zookeeper" level="ERROR"/>  
 <Root level="error">  
 <AppenderRef ref="Console"/>  
 </Root>  
</Loggers>

#### 11.6.2.3 mybatis.xml

<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration  
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
"http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
  
<settings>  
 <!-- 表字段下划线与实体类属性驼峰标识的映射 -->  
 <setting name="mapUnderscoreToCamelCase" value="true"/>  
</settings>  
  
<!-- 别名配置 -->  
<!-- <typeAliases>  
 <package name="com.bawei.hgshop.pojo"/>  
 </typeAliases>  
 -->  
   
 <!-- 配置分页助手拦截器 -->  
<plugins>  
 <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>  
   
 </plugins>  
</configuration>

#### 11.6.2.4 applicationContext-dao.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:aop="http://www.springframework.org/schema/aop"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd  
 http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd">  
  
  
 <!--指定外部属性文件的位置 -->  
 <context:property-placeholder location="classpath:db.properties" />  
  
 <!-- 使用druid数据源 连接池 -->  
 <bean id="dataSource"  
 class="com.alibaba.druid.pool.DruidDataSource">  
 <property name="driverClassName" value="${jdbc.driver}">  
 </property>  
 <property name="url" value="${jdbc.url}">  
 </property>  
 <property name="username" value="${jdbc.username}">  
 </property>  
 <property name="password" value="${jdbc.password}">  
 </property>  
 </bean>  
  
  
 <!--4、 配置mybatis SqlSessionFactory -->  
 <bean id="sqlSessionFactory"  
 class="org.mybatis.spring.SqlSessionFactoryBean" scope="singleton"  
 autowire="default">  
 <!-- 注入数据源 -->  
 <property name="dataSource" ref="dataSource" />  
 <!-- 关联mybatis配置文件 -->  
 <property name="configLocation" value="classpath:mybatis.xml" />  
 <!-- mapper加载的位置 -->  
 <property name="mapperLocations" value="classpath:mappers/\*" />  
 <property name="typeAliasesPackage" value="com.bawei.hgshop.pojo"/>  
 </bean>  
  
 <!--8 扫描mapper -->  
 <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">  
 <!-- basePackage :mapper接口所在的包 -->  
 <property name="basePackage" value="com.bawei.hgshop.dao" />  
 </bean>  
  
</beans>

#### 11.6.2.5 applicationContext-service.xml

<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  
 xmlns="http://www.springframework.org/schema/beans"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xmlns:p="http://www.springframework.org/schema/p"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans   
 http://www.springframework.org/schema/beans/spring-beans.xsd  
 http://dubbo.apache.org/schema/dubbo   
 http://dubbo.apache.org/schema/dubbo/dubbo.xsd  
 http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd  
 http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">  
  
 <!--5、 配置jdbc事务管理器 -->  
 <bean id="transactionManager"  
 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">  
 <property name="dataSource" ref="dataSource"></property>  
 </bean>  
  
 <!--配置事务 transaction-manager:表示关联的事务管理器是谁 -->  
 <tx:advice transaction-manager="transactionManager"  
 id="txAdvice">  
 <!-- 事务属性配置 -->  
 <tx:attributes>  
 <!-- 以find.select get开头的方法为只读事务，用来提高数据库的性能 -->  
 <tx:method name="find\*" read-only="true" />  
 <tx:method name="select\*" read-only="true" />  
 <tx:method name="get\*" read-only="true" />  
 <tx:method name="load\*" read-only="true" />  
 <tx:method name="list\*" read-only="true" />  
 <!-- 其他的方法为默认事务 -->  
 <tx:method name="\*" propagation="REQUIRED" />  
 </tx:attributes>  
 </tx:advice>  
  
 <!-- 7、配置aop -->  
 <aop:config>  
 <!-- 配置切点表达式 -->  
 <aop:pointcut  
 expression="execution(\* com.bawei.hgshop.service.\*.\*(..))"  
 id="pointcut" />  
 <!-- 关联事务 -->  
 <aop:advisor advice-ref="txAdvice" pointcut-ref="pointcut" />  
 </aop:config>  
  
  
  
  
 <!-- 应用程序 -->  
 <dubbo:application name="hgshop-goods-service" />  
 <!-- 注册中心 采用zookeeper -->  
 <!-- <dubbo:registry  
 address="zookeeper://192.168.73.129:2181" /> -->  
 <!-- 直连 -->  
 <dubbo:registry address="N/A" />  
 <!-- 通讯协议 -->  
 <dubbo:protocol name="dubbo" port="20890" />  
 <!-- 注解扫描配置 指定了扫描的包 -->  
 <dubbo:annotation  
 package="com.bawei.hgshop.service.impl" />  
</beans>

### 11.6.3 web.xm

<?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"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"  
 version="2.5">  
 <display-name>hgshop-goods-service</display-name>  
 <context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:applicationContext\*.xml</param-value>  
 </context-param>  
  
 <listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
 </listener>  
  
</web-app>

## 11.7 创建hgshop-manage工程

创建hgshop-manage模块，打成war包。

### 11.7.1 pom.xml文件加入依赖

注意：

1. **加入对应的依赖关系 hgshop-goods-interface**
2. **修改tomcat启动的端口号8080**

* 具体的pom.xml文件如下

<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.bw.hgshop</groupId>  
 <artifactId>hgshop-parent</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 </parent>  
 <artifactId>hgshop-manage</artifactId>  
 <packaging>war</packaging>  
   
 <dependencies>  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper</artifactId>  
 </dependency>  
 <--这一个部分根据自己的实际情况写接口的包名 -->  
   
 <dependency>  
 <groupId>com.bw.hgshop</groupId>  
 <artifactId>hgshop-goods-interface</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 </dependency>  
   
 <!-- 引入dubbo的依赖配置 -->  
 <dependency>  
 <groupId>org.apache.dubbo</groupId>  
 <artifactId>dubbo</artifactId>  
 <exclusions>  
 <exclusion>  
 <artifactId>spring-context</artifactId>  
 <groupId>org.springframework</groupId>  
 </exclusion>  
 <exclusion>  
 <artifactId>netty-all</artifactId>  
 <groupId>io.netty</groupId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.apache.dubbo</groupId>  
 <artifactId>dubbo-dependencies-zookeeper</artifactId>  
 <type>pom</type>  
 </dependency>  
 <!-- spring-webmvc依赖 -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 </dependency>  
 <!-- 日志文件管理包 -->  
 <!-- log start -->  
 <dependency>  
 <groupId>log4j</groupId>  
 <artifactId>log4j</artifactId>  
 </dependency>  
 <!-- 上传组件包 -->  
 <dependency>  
 <groupId>commons-fileupload</groupId>  
 <artifactId>commons-fileupload</artifactId>  
 </dependency>  
 <!-- common-io依赖 -->  
 <dependency>  
 <groupId>commons-io</groupId>  
 <artifactId>commons-io</artifactId>  
 </dependency>  
 <!-- 引入jstl依赖 -->  
 <dependency>  
 <groupId>jstl</groupId>  
 <artifactId>jstl</artifactId>  
 </dependency>  
 <!-- 引入jsp-api依赖 -->  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <scope>provided</scope>  
 </dependency>  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <scope>provided</scope>  
 </dependency>  
  
 <!-- 引入jackson的依赖 -->  
 <dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-databind</artifactId>  
 </dependency>  
  
 <!-- 依赖的公共包 -->  
 <dependency>  
 <groupId>org.apache.commons</groupId>  
 <artifactId>commons-lang3</artifactId>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.tomcat.maven</groupId>  
 <artifactId>tomcat7-maven-plugin</artifactId>  
 <configuration>  
 <port>8080</port>  
 <path>/</path>  
 <uriEncoding>UTF-8</uriEncoding>  
 <contextReloadable>true</contextReloadable>  
 <staticContextDocbase>G://pic</staticContextDocbase>  
 <staticContextPath>/pic</staticContextPath>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

### 11.7.2 配置文件

#### 11.7.2.1 admin.properties

adminUser=admin  
adminPass=123456

#### 11.7.2.2 spring-mvc.xml

<beans xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:dubbo="http://dubbo.apache.org/schema/dubbo"  
 xmlns="http://www.springframework.org/schema/beans"  
 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://dubbo.apache.org/schema/dubbo http://dubbo.apache.org/schema/dubbo/dubbo.xsd  
 http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd  
 http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">  
  
 <!-- 扫描器 -->  
 <context:component-scan  
 base-package="com.bawei.hgshop.controller" />  
 <!-- 视图解析图 -->  
 <bean  
 class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <!-- 配置前缀 -->  
 <property name="prefix" value="/WEB-INF/view/"></property>  
 <!-- 配置后缀 -->  
 <property name="suffix" value=".jsp"></property>  
 </bean>  
 <!-- 不拦截静态资源 -->  
 <mvc:default-servlet-handler />  
 <!-- mvc注解驱动 -->  
 <mvc:annotation-driven/>  
   
 <!-- 文件上传的处理类 -->  
 <bean id="multipartResolver"  
 class="org.springframework.web.multipart.commons.CommonsMultipartResolver"></bean>  
   
 <!-- 应用程序名称 -->  
 <dubbo:application name="hgshop-manage" />  
 <!-- 注册中心 -->  
 <!-- <dubbo:registry  
 address="zookeeper://192.168.73.129:2181" /> -->  
 <!-- 配置扫描包的路径 -->   
 <dubbo:annotation package="com.bawei.hgshop.controller"/>  
</beans>

#### 11.7.2.3 log4j.properties

log4j.rootLogger=debug, A1  
log4j.appender.A1=org.apache.log4j.ConsoleAppender  
log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
#log4j.appender.A1.layout.ConversionPattern=%-4r %-5p [%t] %37c %3x - %m%n

### 11.7.3 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"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"  
 version="2.5">  
   
 <welcome-file-list>  
 <welcome-file>login.jsp</welcome-file>  
 </welcome-file-list>  
 <servlet>  
 <servlet-name>springDispatcherServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring-mvc.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>springDispatcherServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>  
   
 <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>  
 </filter>  
 <filter-mapping>  
 <filter-name>CharacterEncodingFilter</filter-name>  
 <url-pattern>/\*</url-pattern>  
 </filter-mapping>  
   
</web-app>

### 11.7.4 拷贝静态资源文件