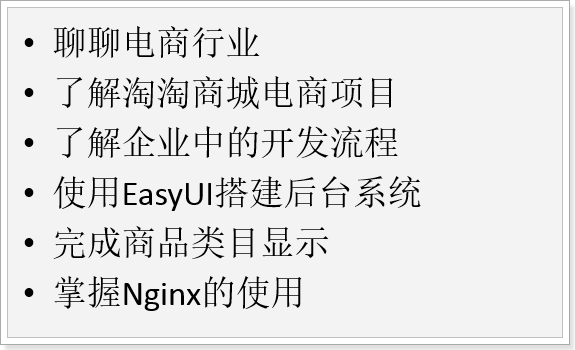
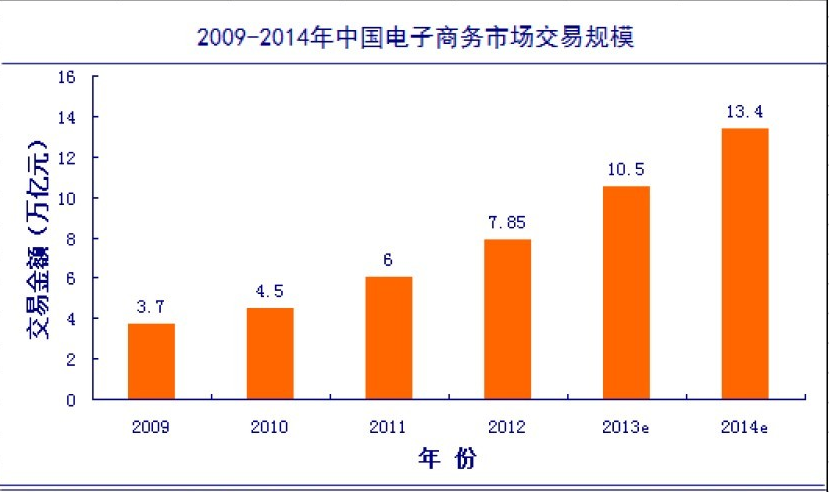
# 今日大纲



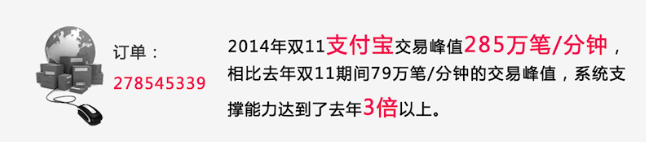
# 聊聊电商行业

## 电商行业发展



## 11.11

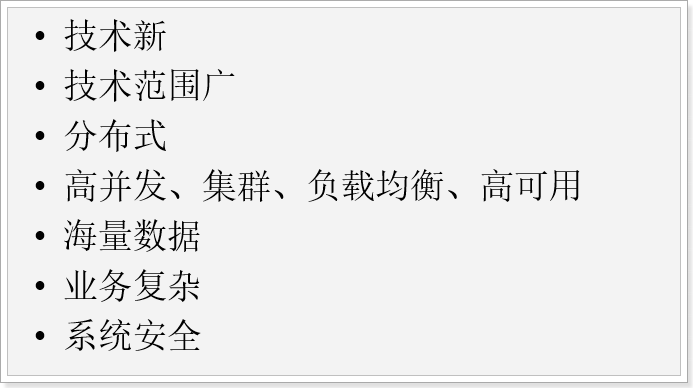




2015双11：



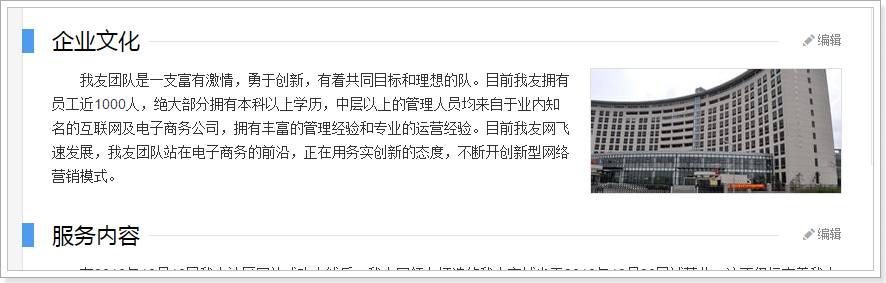
# 电商行业技术特点



# 淘淘商城简介

## 淘淘商城的前身





## 电商行业的概念

B2C：商家对个人，如：京东、亚马逊、当当等

C2C：个人对个人，如：淘宝集市、拍拍网

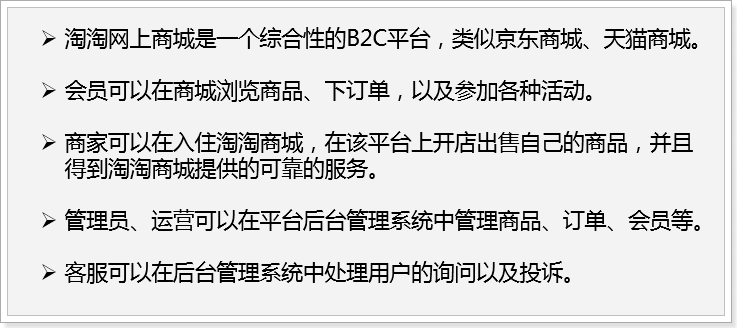
B2B：商家对商家，如：阿里巴巴、八方资源网等

O2O：线上和线下结合，如：饿了么、电影票、团购等

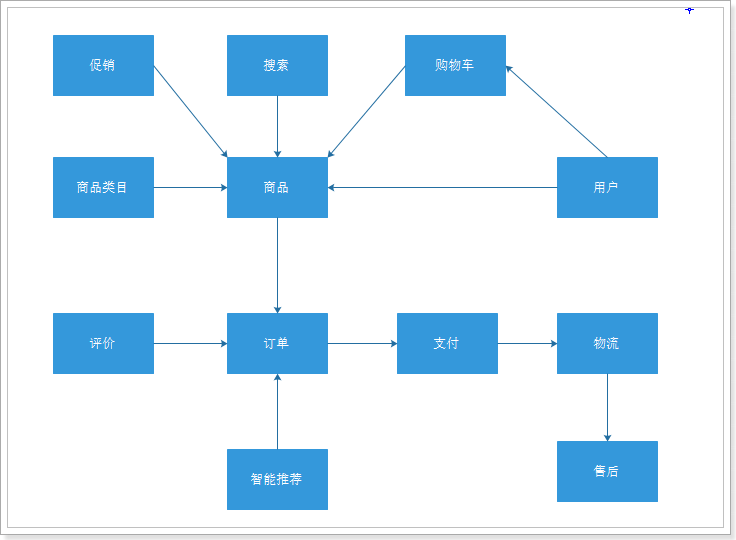
P2P：在线金融，贷款，如：网贷之家、人人聚财等

B2C平台：天猫、京东、一号店、当当等

## 简介



## 功能

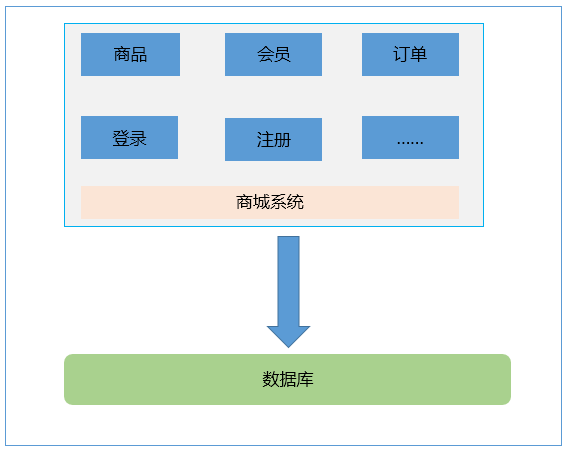


思维导图：



## 架构

### 传统架构



存在的问题：

1. 开发、维护代码比较困难
2. 系统内部的功能模块之间的耦合度太高了
3. 无法针对每个模块的特性做优化
4. 无法做服务器的水平扩展
5. 无法存储海量数据

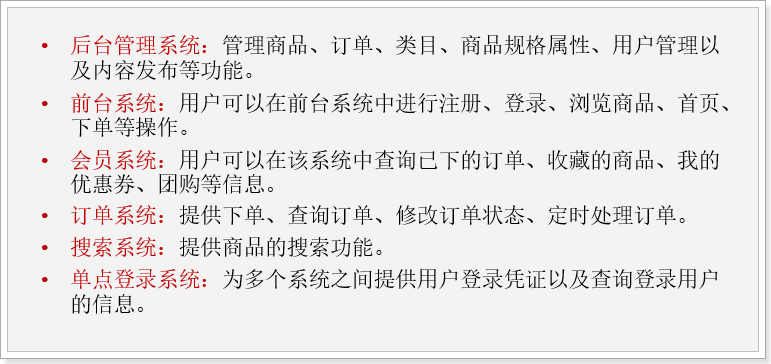
### 分布式系统架构



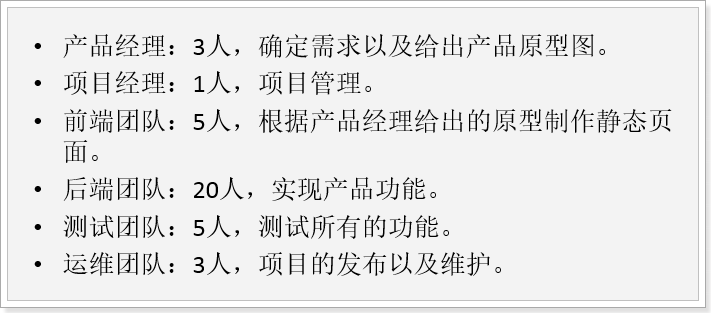
已经解决以上存在的问题，但是，带来了新问题：

1. 系统间的服务的调用变得复杂
2. 如何解决错综复杂的服务调用关系：服务注册中心。

### 各个系统说明



## 人员配置

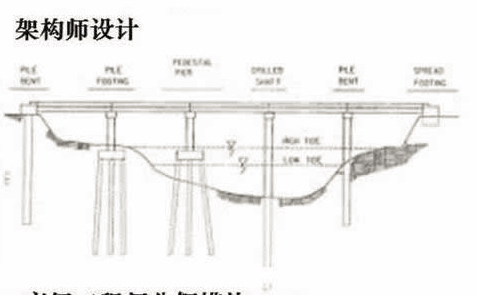


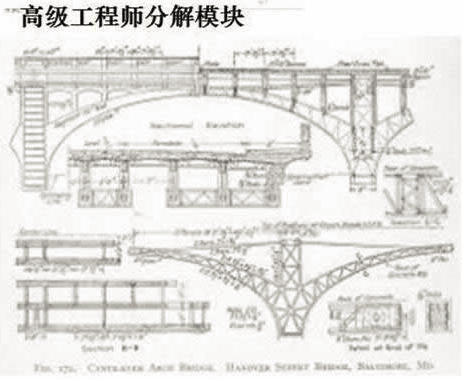
## 开发流程

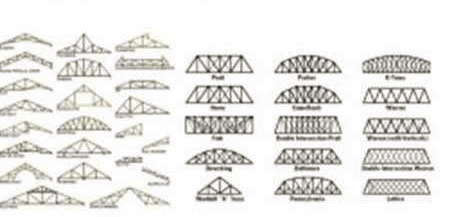


现状















## 开发技术



# 开发

## 开发工具和环境



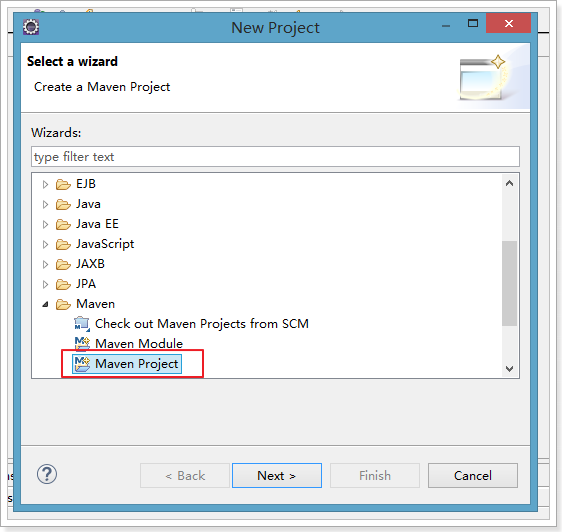
## 域名

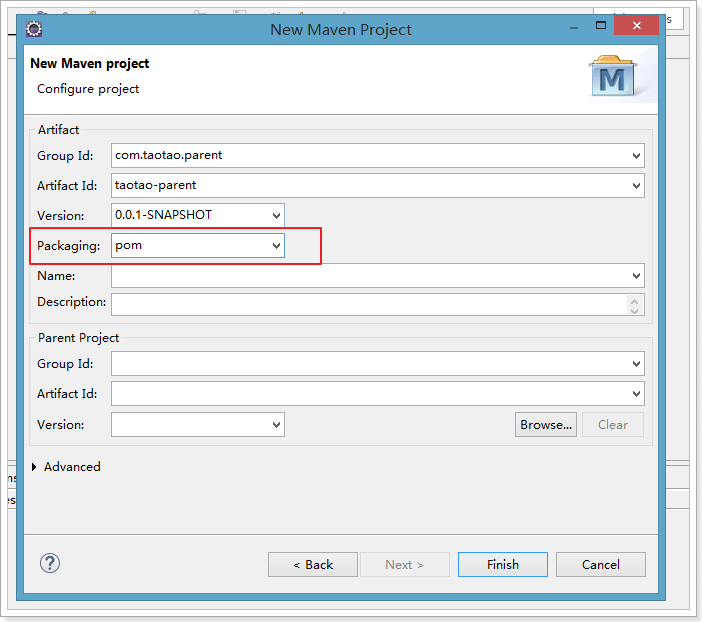
一级域名：[www.taotao.com](http://www.taotao.com) taotao.com

二级域名：maven.taotao.com / svn.taotao.com

三级域名：xxx.aa.taotao.com

## 创建taotao-parent





### Pom.xml

<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.taotao.parent</groupId>

<artifactId>taotao-parent</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

<!-- 集中定义依赖版本号 -->

<properties>

<junit.version>4.10</junit.version>

<spring.version>4.1.3.RELEASE</spring.version>

<mybatis.version>3.2.8</mybatis.version>

<mybatis.spring.version>1.2.2</mybatis.spring.version>

<mybatis.paginator.version>1.2.15</mybatis.paginator.version>

<mysql.version>5.1.32</mysql.version>

<slf4j.version>1.6.4</slf4j.version>

<jackson.version>2.4.2</jackson.version>

<druid.version>1.0.9</druid.version>

<httpclient.version>4.3.5</httpclient.version>

<jstl.version>1.2</jstl.version>

<servlet-api.version>2.5</servlet-api.version>

<jsp-api.version>2.0</jsp-api.version>

<joda-time.version>2.5</joda-time.version>

<commons-lang3.version>3.3.2</commons-lang3.version>

<commons-io.version>1.3.2</commons-io.version>

</properties>

<dependencyManagement>

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<!-- Spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

<!-- Mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>${mybatis.version}</version>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>${mybatis.spring.version}</version>

</dependency>

<!-- 分页助手 -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>3.7.5</version>

</dependency>

<dependency>

<groupId>com.github.jsqlparser</groupId>

<artifactId>jsqlparser</artifactId>

<version>0.9.1</version>

</dependency>

<!-- 通用Mapper -->

<dependency>

<groupId>com.github.abel533</groupId>

<artifactId>mapper</artifactId>

<version>2.3.4</version>

</dependency>

<!-- MySql -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>${mysql.version}</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>${slf4j.version}</version>

</dependency>

<!-- Jackson Json处理工具包 -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>${jackson.version}</version>

</dependency>

<!-- 连接池 -->

<dependency>

<groupId>com.jolbox</groupId>

<artifactId>bonecp-spring</artifactId>

<version>0.8.0.RELEASE</version>

</dependency>

<!-- httpclient -->

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>${httpclient.version}</version>

</dependency>

<!-- JSP相关 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet-api.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>${jsp-api.version}</version>

<scope>provided</scope>

</dependency>

<!-- 时间操作组件 -->

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

<version>${joda-time.version}</version>

</dependency>

<!-- Apache工具组件 -->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>${commons-lang3.version}</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-io</artifactId>

<version>${commons-io.version}</version>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<finalName>${project.artifactId}</finalName>

<plugins>

<!-- 资源文件拷贝插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-resources-plugin</artifactId>

<version>2.7</version>

<configuration>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<!-- java编译插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.2</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

<pluginManagement>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

</plugin>

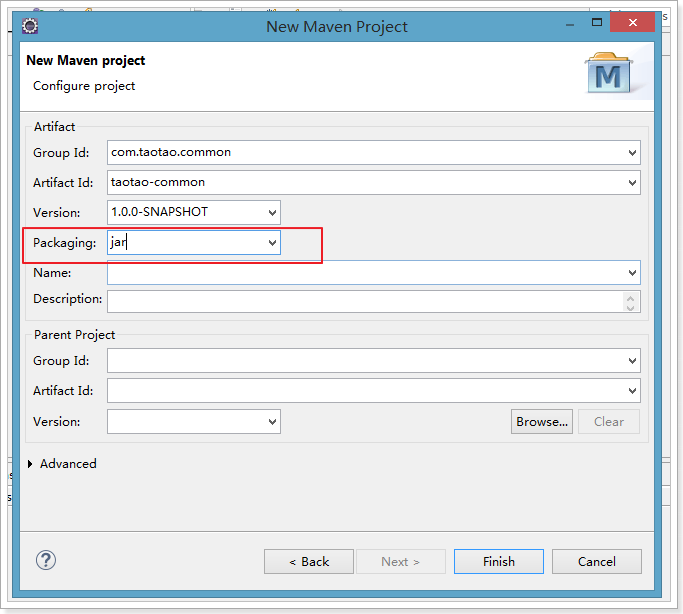
</plugins>

</pluginManagement>

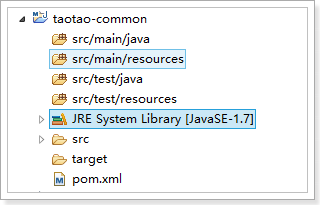
</build>

</project>

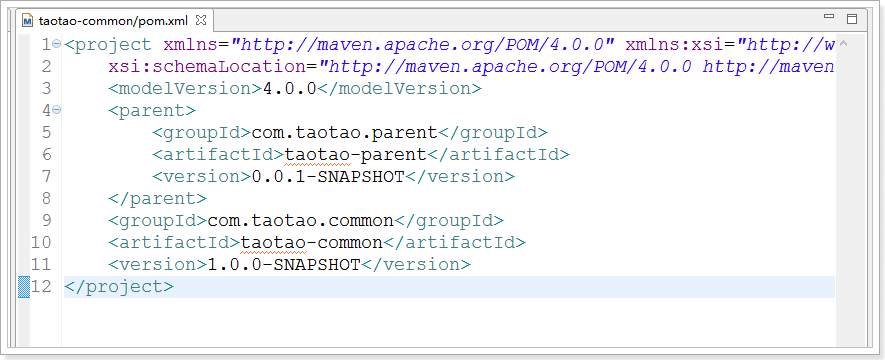
## 创建taotao-common



创建成功：



### Pom.xml



## 创建taotao-manage

### 分析

Com.taotao.manage.pojo

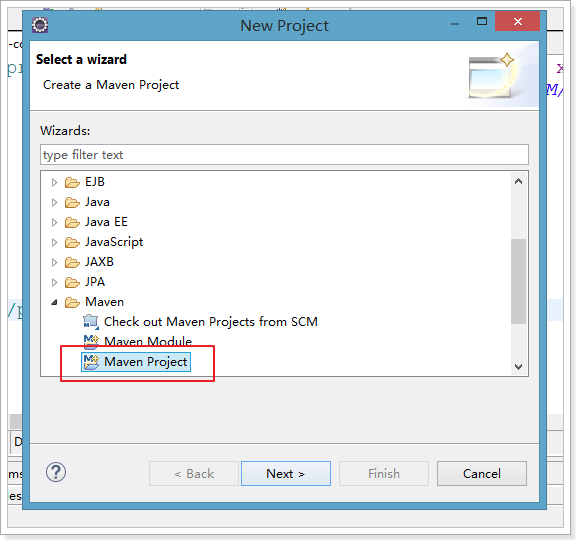
Com.taotao.manage.mapper

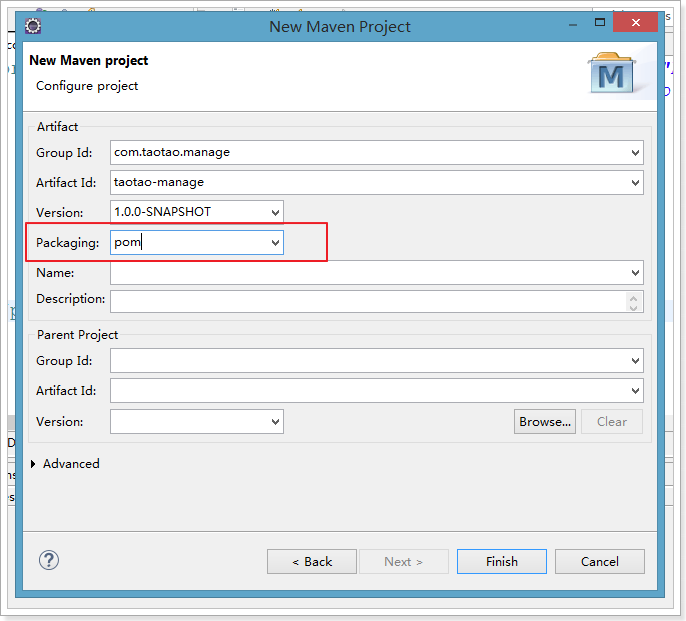
Com.taotao.manage.service

Com.taotao.manage.controller

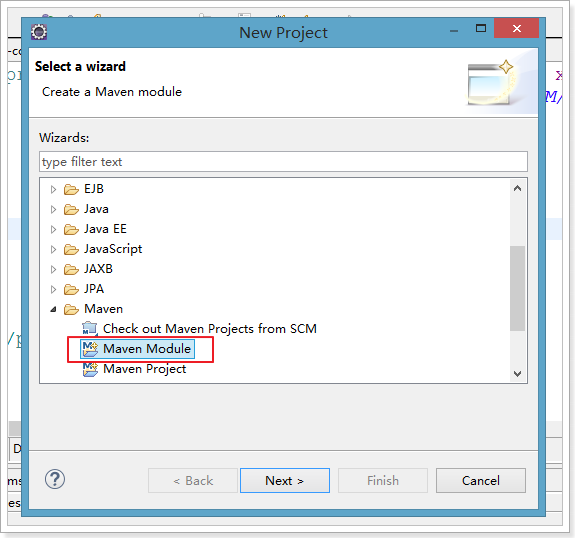
使用Maven的聚合将所有的子工程进行创建。

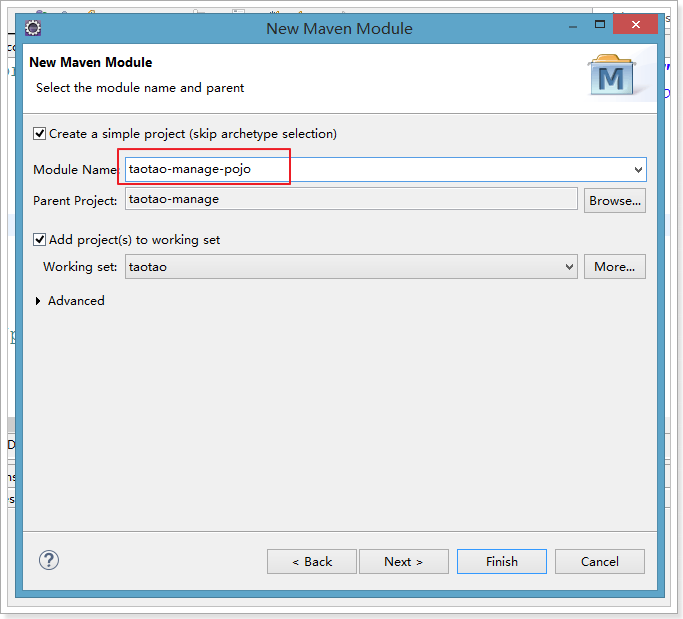
### 创建taotao-manage

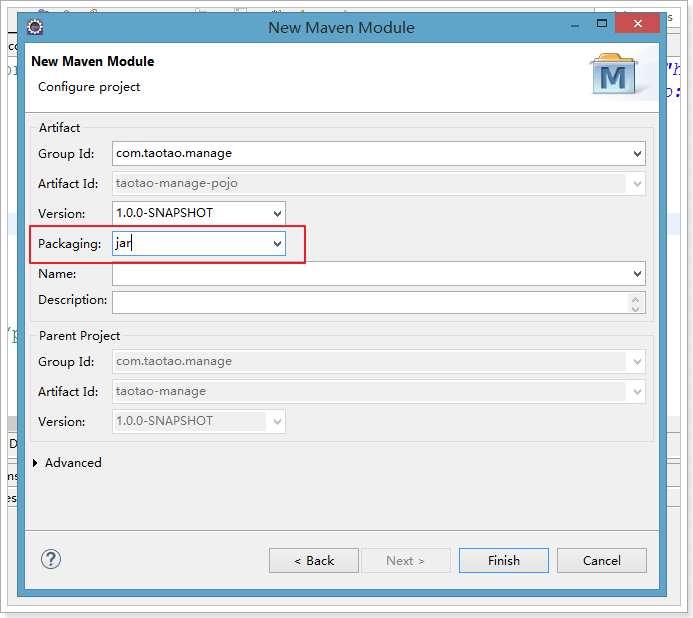


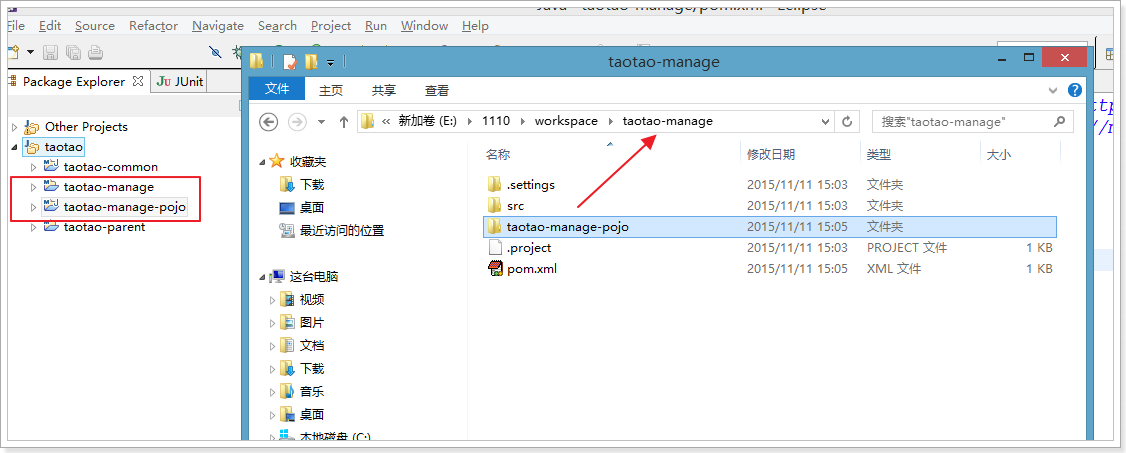


### Taotao-manage-pojo

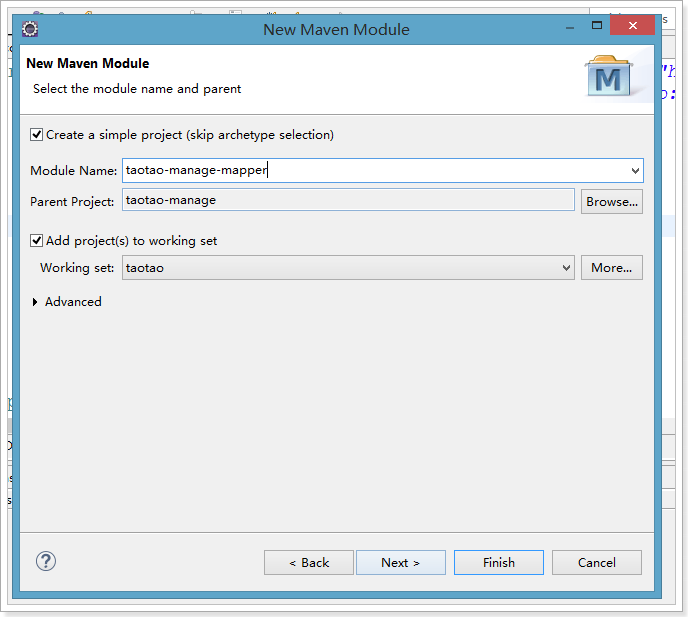


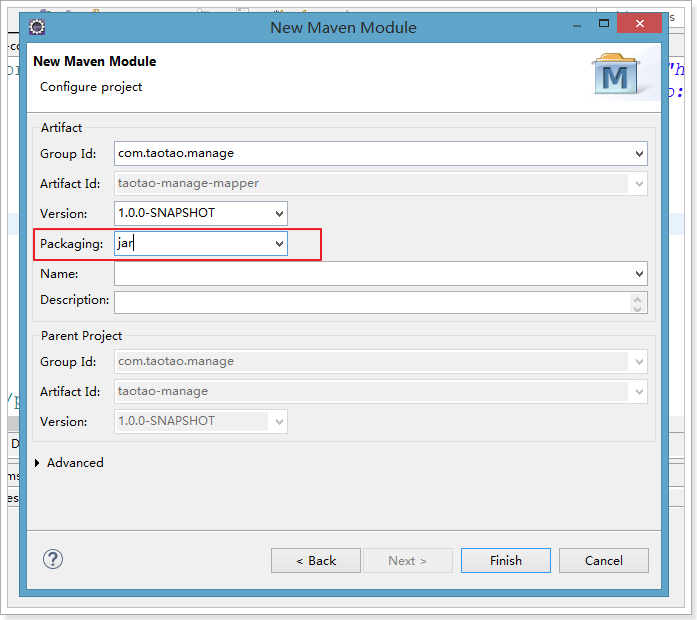




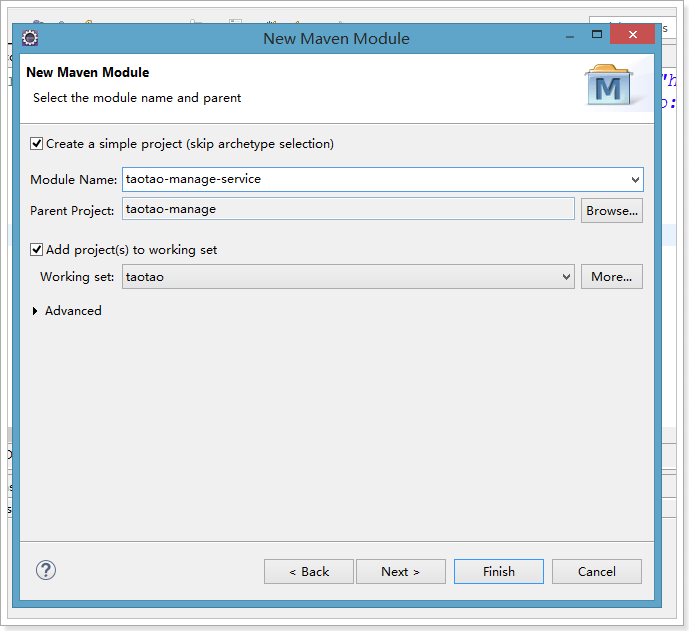


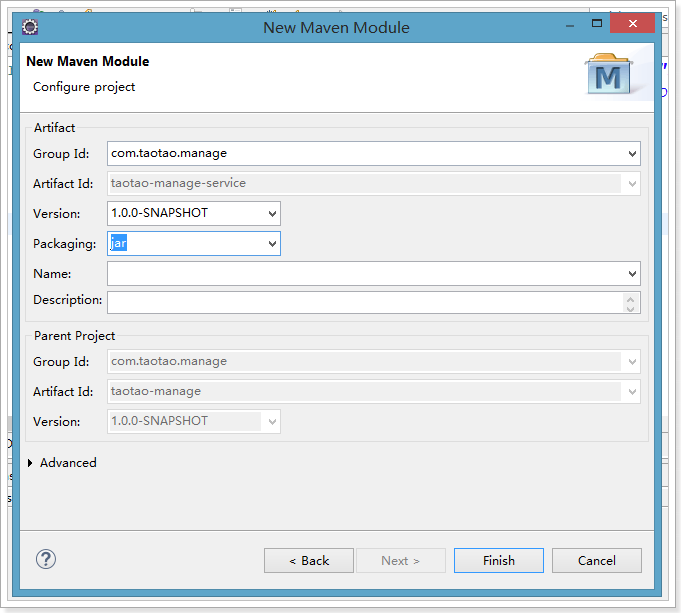
### Taotao-manage-mapper



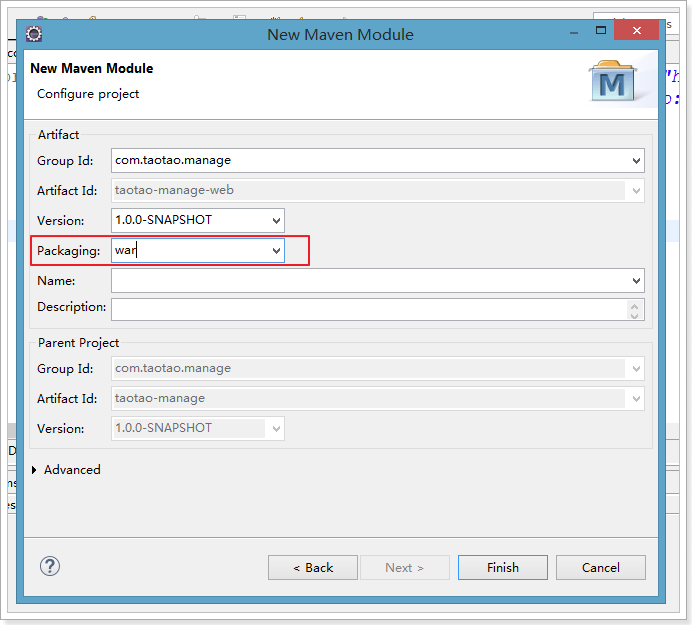


### Taotao-manage-service

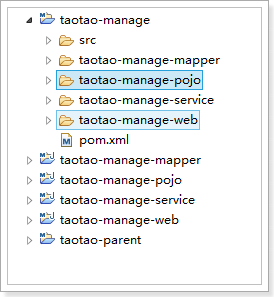




### Taotao-manage-web



### 创建完成



## 子工程之间的依赖关系

Taotao-manage-web 🡺 taotao-manage-service 🡺 taotao-manage-mapper 🡺 taotao-manage-pojo

## 导入依赖

导入依赖的原则：

1. 所有的工程都需要的依赖应该在聚合工程（taotao-manage）中导入。
2. 在使用依赖的最底层导入。

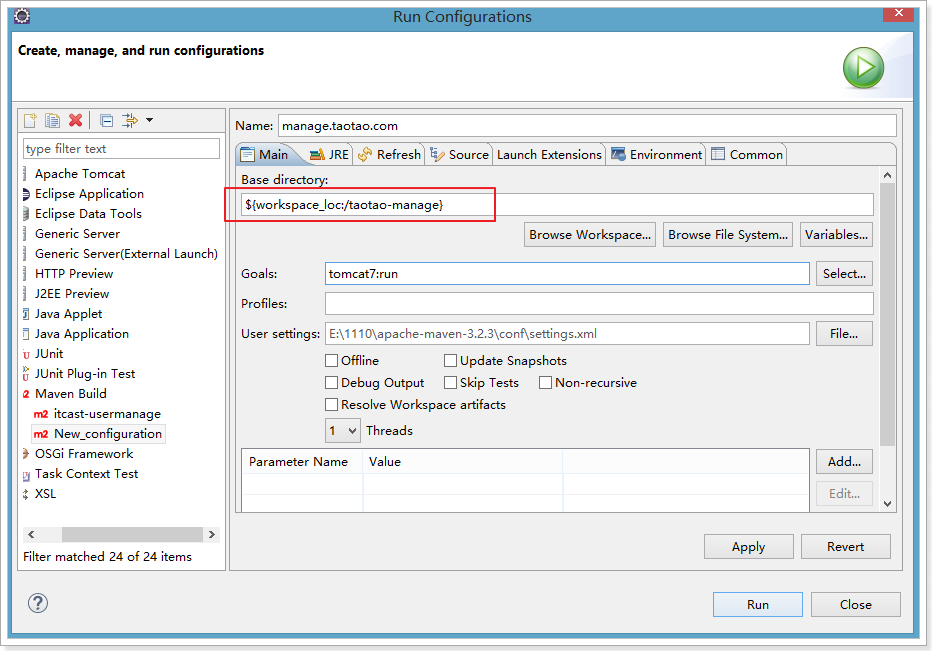
3、运行时所需要的依赖在web工程中加入。

## 导入tomcat插件

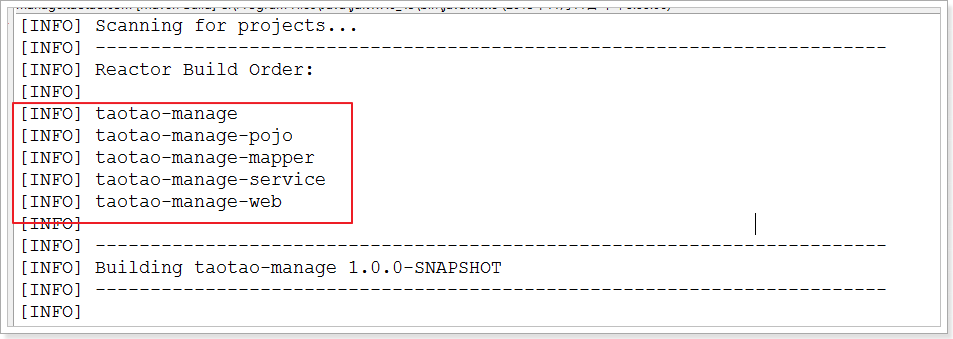
聚合工程的tomcat插件要在聚合工程中导入，taotao-manage中导入。



运行tomcat插件：



聚合后运行：



## 创建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"*

id=*"WebApp\_ID"* version=*"2.5"*>

<display-name>taotao-manage</display-name>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/applicationContext\*.xml</param-value>

</context-param>

<!--Spring的ApplicationContext 载入 -->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!-- 编码过滤器，以UTF8编码 -->

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 解决PUT请求无法提交表单数据的问题 -->

<filter>

<filter-name>HttpMethodFilter</filter-name>

<filter-class>org.springframework.web.filter.HttpPutFormContentFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>HttpMethodFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 将POST请求转化为DELETE或者是PUT 要用\_method指定真正的请求参数 -->

<filter>

<filter-name>HiddenHttpMethodFilter</filter-name>

<filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>HiddenHttpMethodFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 配置SpringMVC框架入口 -->

<servlet>

<servlet-name>taotao-manage</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/taotao-manage-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>taotao-manage</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

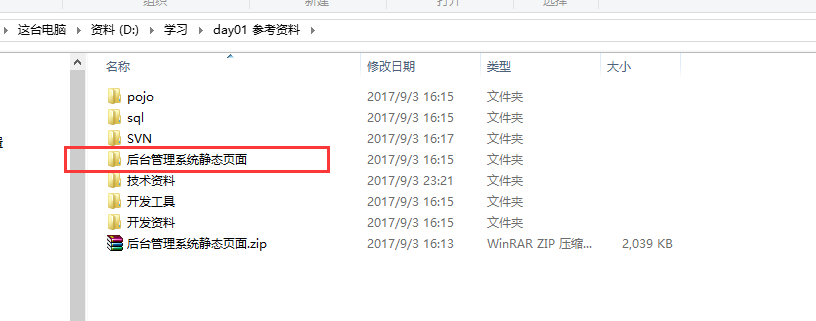
<welcome-file-list>

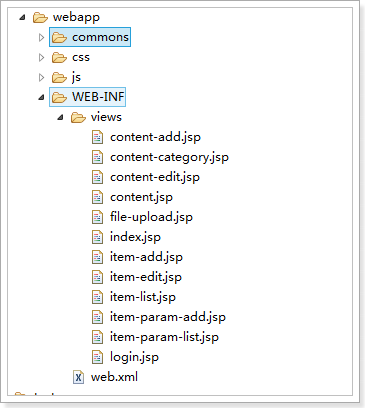
<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

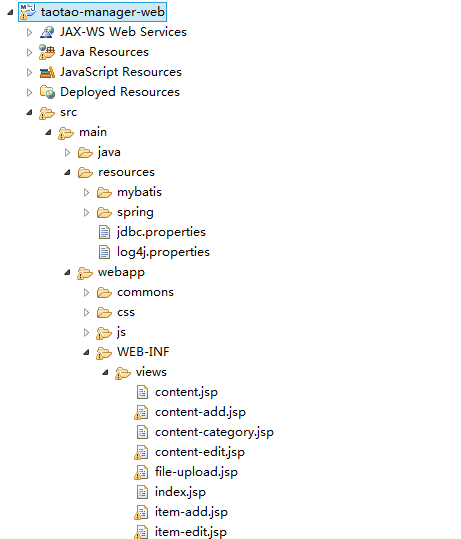
</web-app>

## 导入静态页面

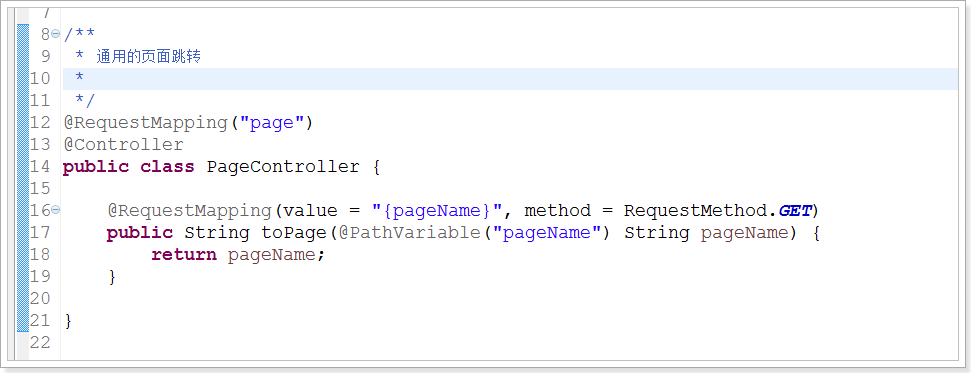




## 配置SSM

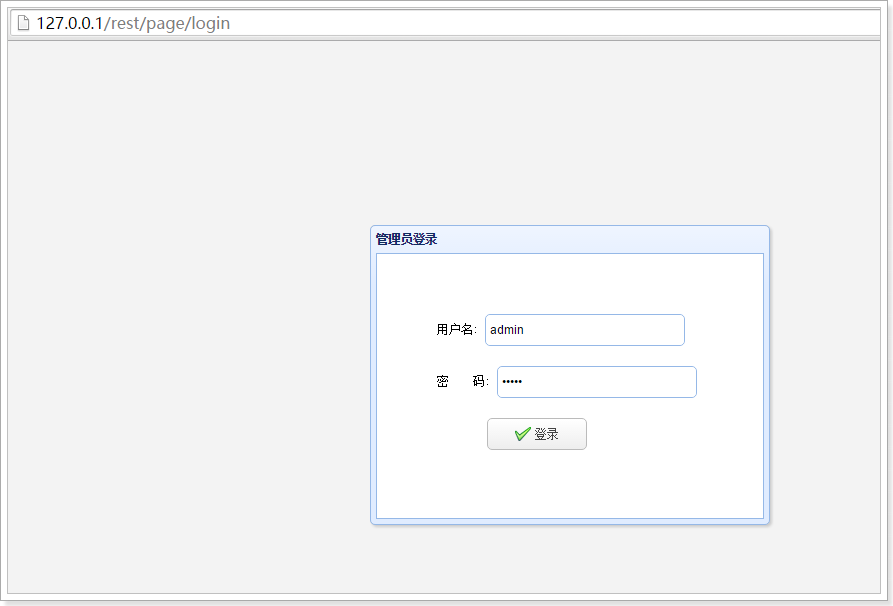


## 通用页面跳转



# 后台管理系统

## 登录页面



登录逻辑：

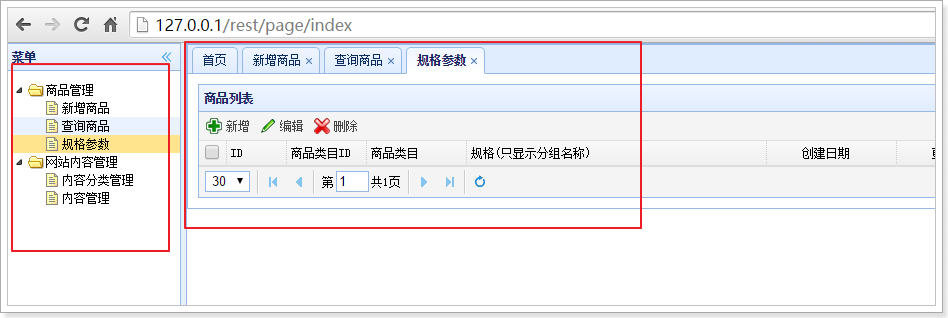


## 首页的js布局实现

### 布局

左中布局：

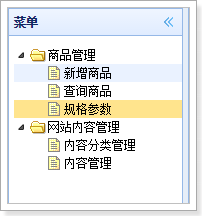




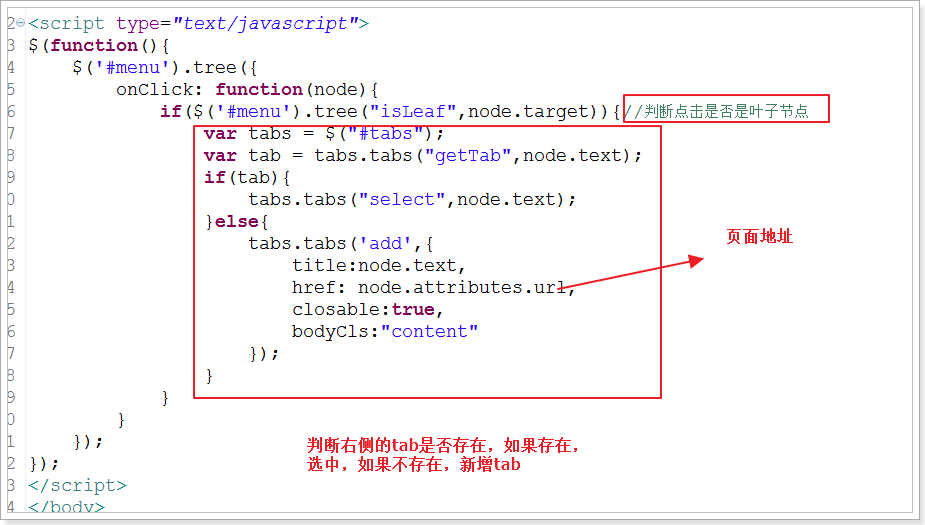
### 左侧菜单树



效果：



### 菜单的点击事件

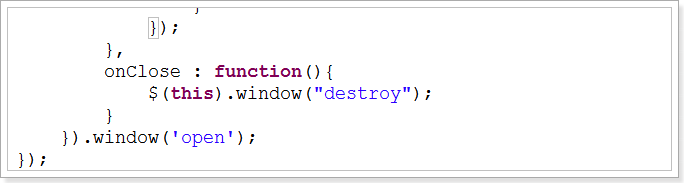


## 选择商品类目

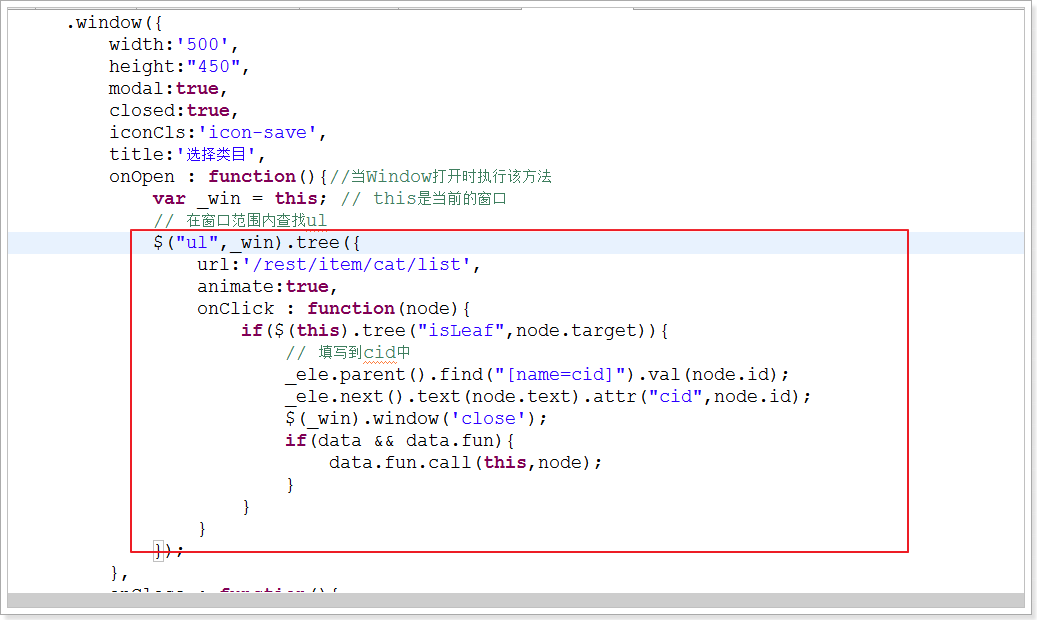
功能描述：在新增商品时点击选择类目，弹出窗口，在窗口中显示商品类目数据。

### 点击弹出框

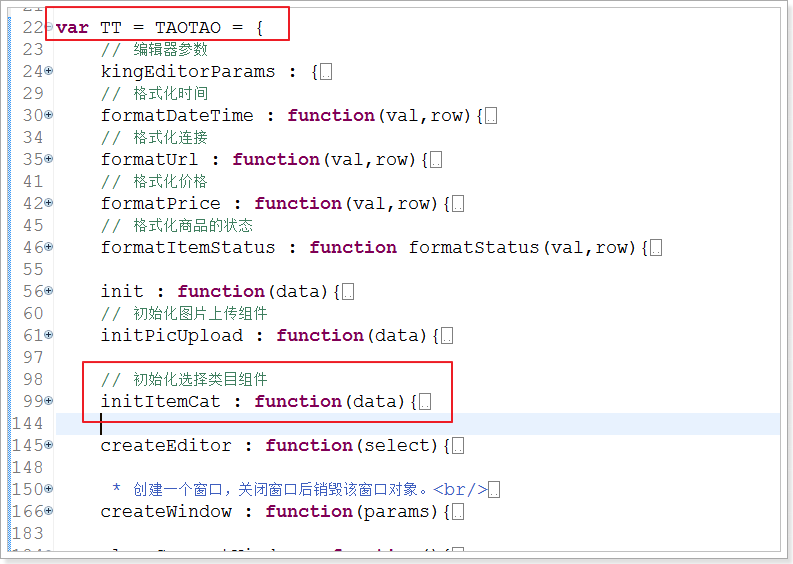




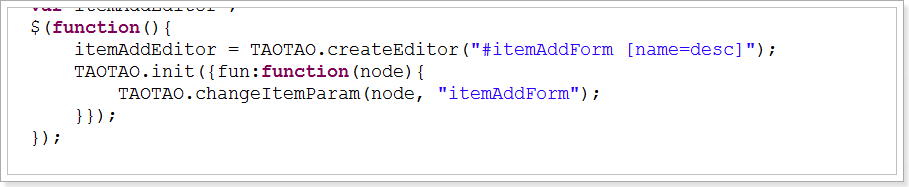
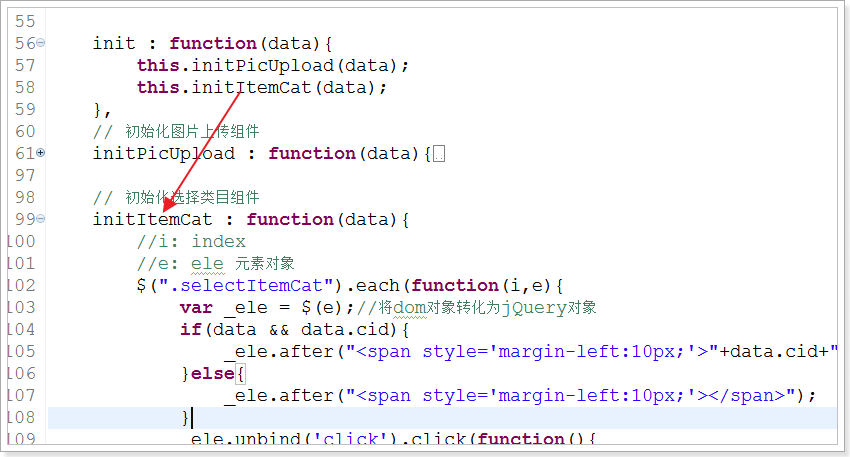
### 加载tree



### 定义TAOTAO对象

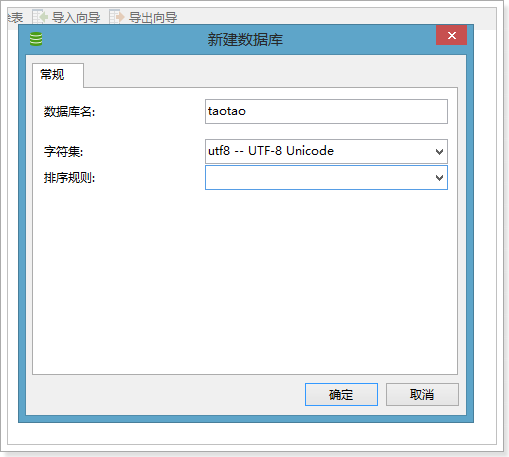


### 初始化过程

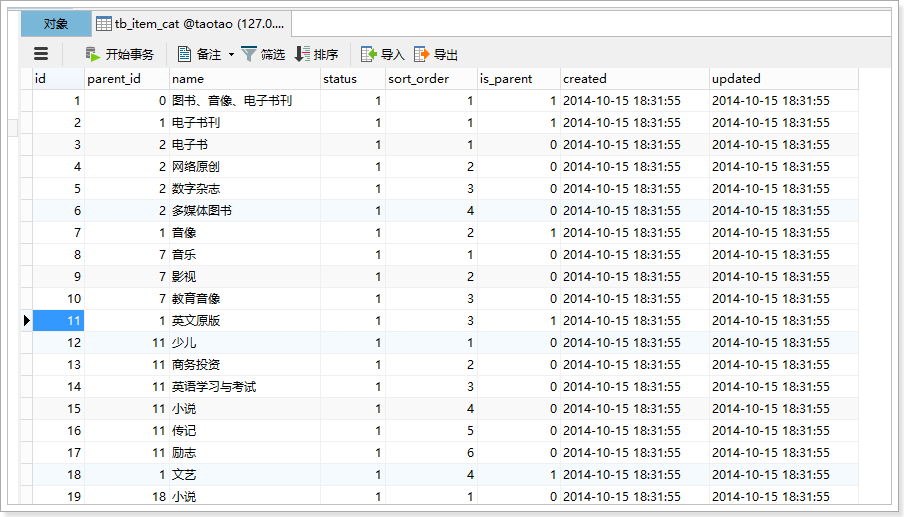
1. 通过TAOTAO.init方法完成初始化  
   
2. TT.init中调用initItemCat方法  
   
3. 给class=” selectItemCat”绑定click事件，即可实现效果

## 商品类目的数据结构

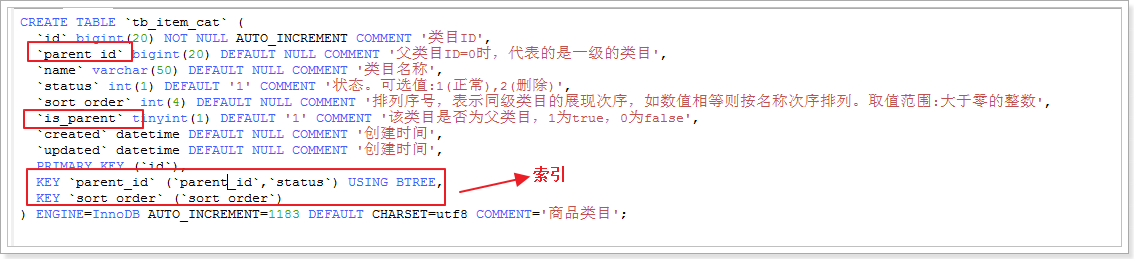
创建数据库：



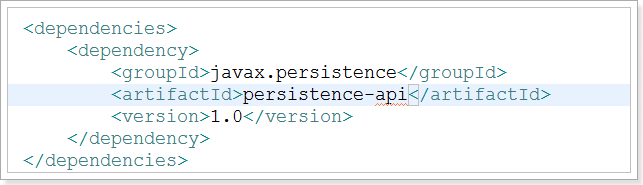
### 数据



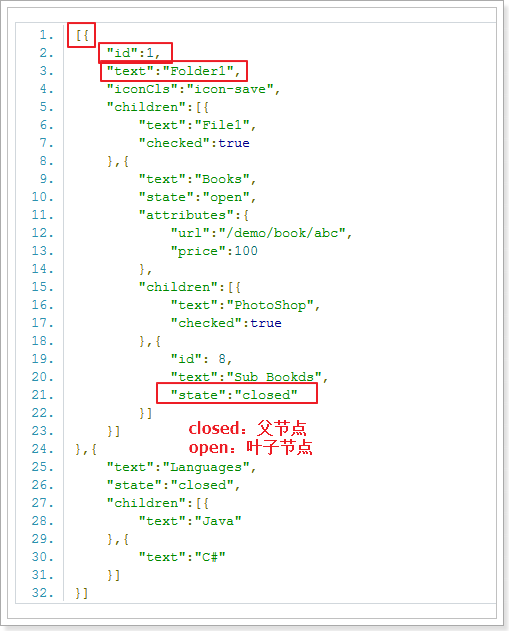
### 表结构



### 在taotao-manage-pojo中添加JPA注解依赖



### EasyUI的tree数据结构

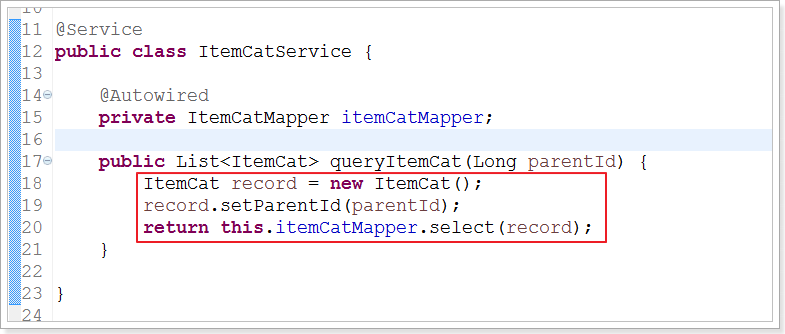


## 实现

### Controller



### Service



### ItemCat

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Table(name = "tb\_item\_cat")

**public** **class** ItemCat **extends** BasePojo {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** Long id;

**private** Long parentId;

**private** String name;

**private** Integer status;

**private** Integer sortOrder;

**private** Boolean isParent;

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Long getParentId() {

**return** parentId;

}

**public** **void** setParentId(Long parentId) {

**this**.parentId = parentId;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getStatus() {

**return** status;

}

**public** **void** setStatus(Integer status) {

**this**.status = status;

}

**public** Integer getSortOrder() {

**return** sortOrder;

}

**public** **void** setSortOrder(Integer sortOrder) {

**this**.sortOrder = sortOrder;

}

**public** Boolean getIsParent() {

**return** isParent;

}

**public** **void** setIsParent(Boolean isParent) {

**this**.isParent = isParent;

}

// 扩展字段，支持EasyUItree的显示

**public** String getText() {

**return** **this**.getName();

}

**public** String getState() {

**return** **this**.getIsParent() ? "closed" : "open";

}

}

### 效果

