**淘淘商城**

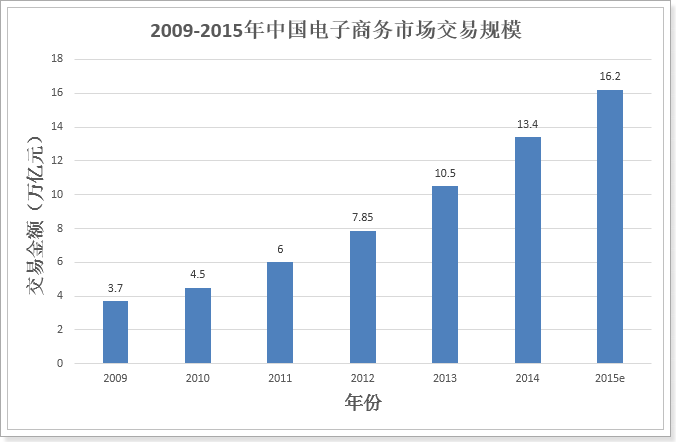
**第一天**

# 课程计划

1. 聊聊电商行业
   1. 电商行业
   2. 行业技术特点
2. 淘淘商城简介
   1. 常用名词
3. 系统功能
   1. 系统功能图
   2. 功能描述
4. 系统架构
   1. 传统架构
   2. 分布式架构
   3. 淘淘商城架构
5. 技术选型和开发环境
6. 创建公共工程
   1. taotao-parent
   2. taotao-common
7. dubbo介绍
8. 创建后台管理系统

# 聊聊电商

## 电商交易规模



## 电商行业技术特点

* 技术新
* 技术范围广
* 分布式
* 高并发，集群，负载均衡，高可用
* 海量数据
* 业务复杂
* 系统安全

分布式：一件事情，拆开来做

集群：同样的事情，大家一起做

# 淘淘商城简介

* 淘淘商城是一个综合性的B2C平台，类似京东商城、天猫。
* 会员可以在商城浏览商品、下单、以及参加各种活动。
* 商家可以入住淘淘商城，在该平台上开店出售自己的商品、并且得到淘淘商城的可靠服务。
* 管理员、运营可以在平台的后台管理系统中管理商品、订单、会员等
* 客服可以在后台管理系统中处理用户的询问以及投诉

（IT）传统行业：OA，CRM，ERP，医药采购管理系统，银行系统，财务软件

（IT）互联网行业：电商（淘宝、京东、淘淘），搜索（百度、谷歌），社交（人 人网，天涯论坛），新浪微博，门户网站（新浪，搜狐），12306.cn

电商：

B2C：B:商家，C：个人，商家对个人（苏宁易购，以前的京东，天猫超市）

C2C：C：个人，个人对个人（淘宝，闲鱼（个人拍卖））

O2O：Online To Offline，线上对线下(饿了么，网上买电影票）

B2B：B：商家，商家对商家的（阿里巴巴）

B2C平台：商家对个人的**平台**（天猫，现在的京东）

互联网金融

P2P:Person 2 Person ，个人对个人的信贷，互联网的个人对个人的信贷平台

# 系统功能

## 系统功能图



## 系统介绍

后台管理系统：提供商品管理、类目选择以及内容发布等功能。

前台系统：提供用户注册、用户登录、浏览商品、首页（门户）、下单等功能。

订单系统：提供下单、定时处理订单。

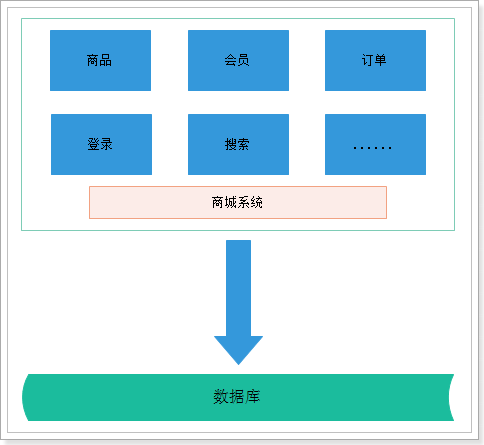
搜索系统：提供商品的搜索功能。

购物车系统：提供加入商品到购物车，修改购物车，删除购物车等功能。

单点登录系统：为多个系统提供用户登录以及查询登录用户信息等功能。

# 系统架构

## 传统架构

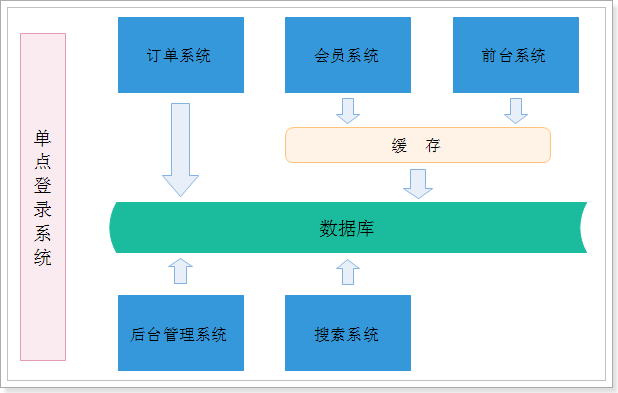


传统架构又称为集中式架构

存在的问题：

1. 功能耦合性高
   1. 学习成本高。
   2. 系统维护成本高。
   3. 各个功能不能灵活部署。
2. 无法进行水平扩展（集群）。

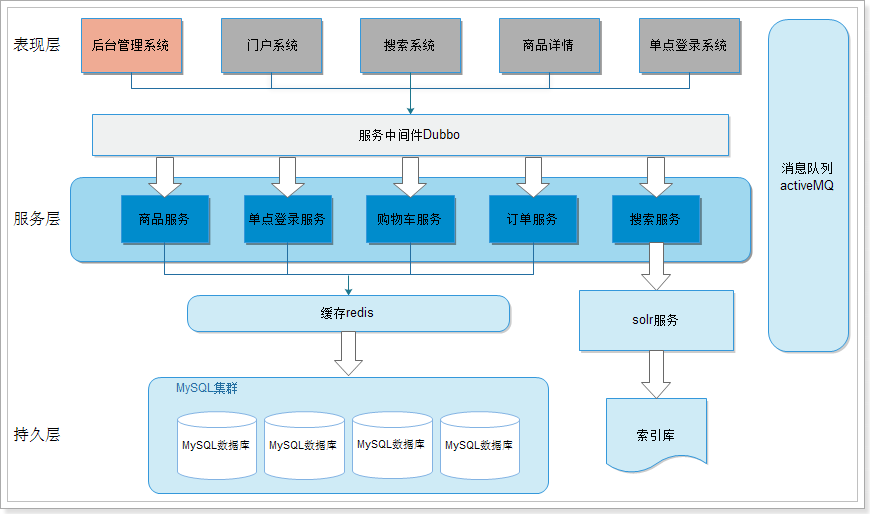
## 分布式架构



存在的问题：

1. 系统间的调用，效率比系统内的调用要低
2. 系统间的通知无法直接进行，需要使用消息机制（mq）

## 淘淘商城系统架构



# 技术选型和开发环境

## 技术选型

* Spring、SpringMVC、Mybatis
* JSP、JSTL、jQuery、EasyUI、KindEditor（富文本编辑器）
* Redis（缓存服务器，单点登录，购物车）
* Solr（搜索）
* dubbo（分布式服务框架）
* HttpClient（HTTP 协议访问客户端）
* ActiveMQ（消息队列）
* Quartz（定时任务）
* FastDFS（图片服务器）
* FreeMarker（静态化）
* Nginx（反向代理服务器）
* MyCat（数据库中间件）

## 开发工具和环境

* Eclipse Mars.2

自带maven插件，需要手工安装svn插件。properties

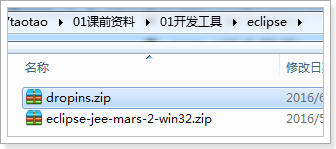
* Maven 3.3.3

使用开发工具（eclipse）自带的Maven

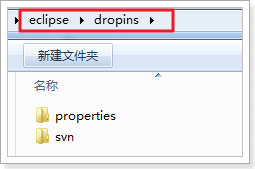
* Tomcat 7.0.47（Maven Tomcat Plugin）
* JDK 1.7
* Mysql 5.5
* Dubbo 2.5.3
* Nginx 1.5.0
* Redis 3.0.2
* ActiveMQ 5.12.0
* Win7 操作系统
* SVN（版本管理）

### svn插件安装：

解压课前资料中的dropins.zip到eclipse目录下（\01课前资料\01开发工具\eclipse\dropins.zip)

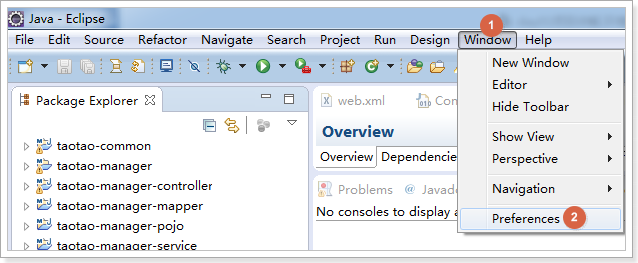


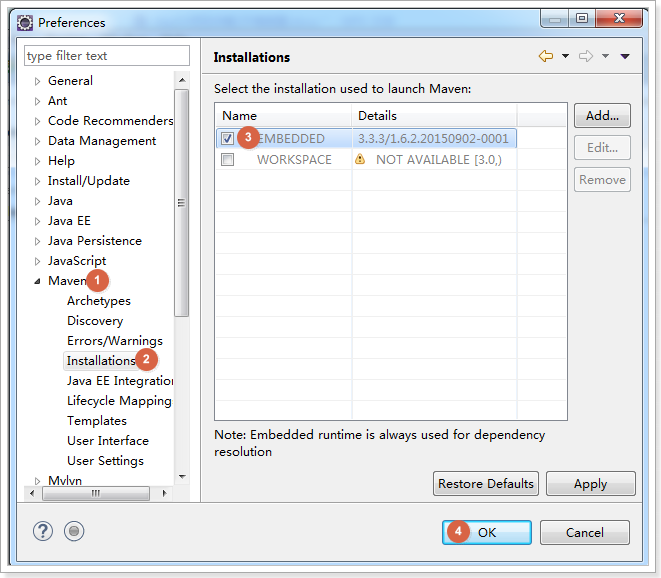
效果:

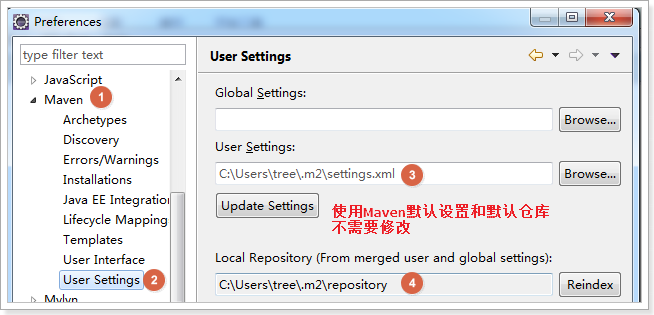


### Maven使用

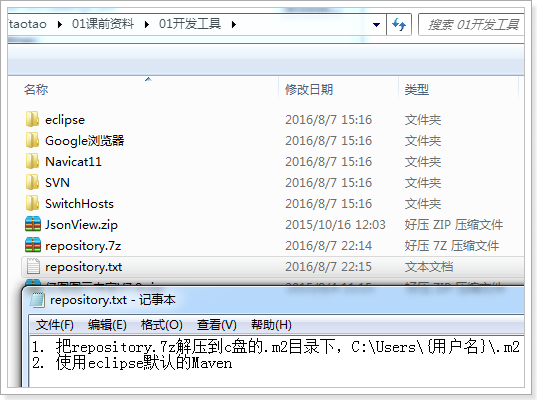
修改eclipse的Maven为默认设置



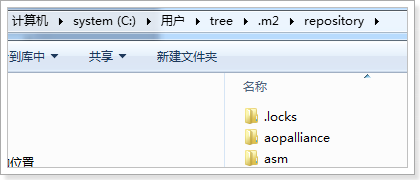




本地仓库使用



效果



# 创建公共工程

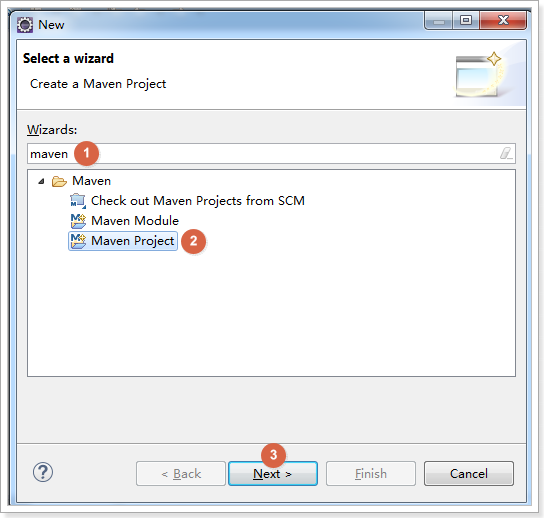
在Maven管理体系中需要将依赖的统一管理，所以需要taotao-parent工程

在开发过程中，需要将一些公用的类或者功能模块抽取出来已达到重用的目的，所以需要创建taotao-common工程

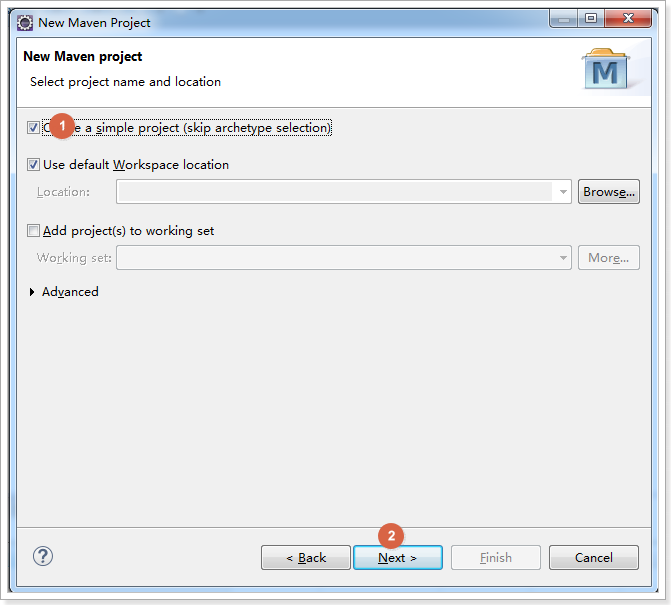
## 创建taotao-parent

*使用svn下载老师的代码，建议学员创建的工程名不要和老师一样。*

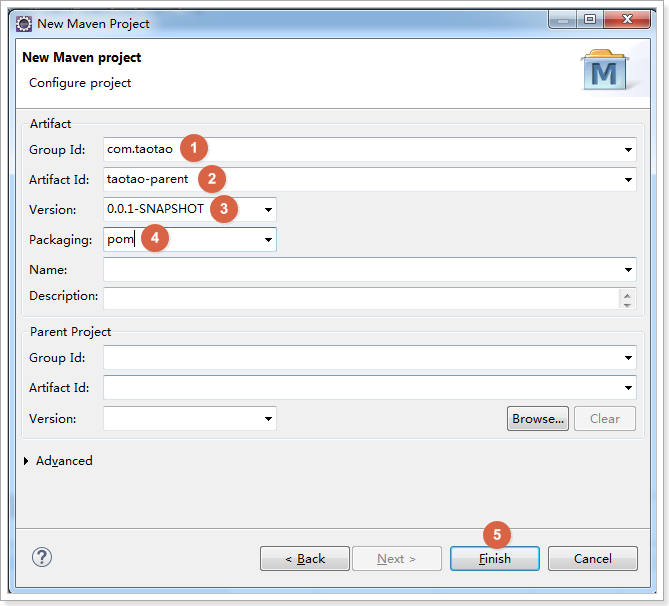
### 创建Maven工程



### 选择简单的Maven工程

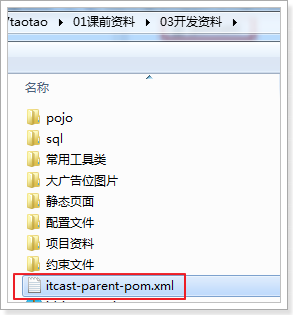


### 填写信息

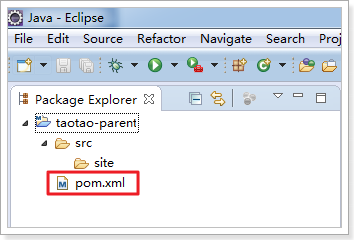


### 编辑pom.xml文件

把课前资料中的内容加入进来



编辑pom.xml文件



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.taotao</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>4.12</junit.version>

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

<mybatis.version>3.2.7</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>

<jolbox.version>0.8.0.RELEASE</jolbox.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>

<commons-net.version>3.3</commons-net.version>

<pagehelper.version>3.7.5</pagehelper.version>

<mapper.version>2.3.4</mapper.version>

<jsqlparser.version>0.9.1</jsqlparser.version>

<commons-fileupload.version>1.3.1</commons-fileupload.version>

<commons-codec.version>1.9</commons-codec.version>

<jedis.version>2.7.2</jedis.version>

<solrj.version>4.10.3</solrj.version>

<dubbo.version>2.5.3</dubbo.version>

<zookeeper.version>3.4.7</zookeeper.version>

<zkclient.version>0.1</zkclient.version>

<activemq.version>5.12.0</activemq.version>

<freemarker.version>2.3.23</freemarker.version>

<quartz.version>2.2.2</quartz.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-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>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</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>

<!-- 通用Mapper -->

<dependency>

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

<artifactId>mapper</artifactId>

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

</dependency>

<!-- 分页助手 -->

<dependency>

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

<artifactId>pagehelper</artifactId>

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

</dependency>

<dependency>

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

<artifactId>jsqlparser</artifactId>

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

</dependency>

<!-- MySql -->

<dependency>

<groupId>mysql</groupId>

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

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

</dependency>

<!-- 日志 -->

<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>${jolbox.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通用工具组件 -->

<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>

<!-- 文件上传组件 -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

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

</dependency>

<!-- dubbo相关 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

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

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

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

</dependency>

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

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

</dependency>

<!-- 加密解密 -->

<dependency>

<groupId>commons-codec</groupId>

<artifactId>commons-codec</artifactId>

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

</dependency>

<!-- 定时任务Quartz -->

<dependency>

<groupId>org.quartz-scheduler</groupId>

<artifactId>quartz</artifactId>

<version>2.2.1</version>

</dependency>

<!-- ActiveMQ依赖 -->

<dependency>

<groupId>org.apache.activemq</groupId>

<artifactId>activemq-all</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jms</artifactId>

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

</dependency>

<!-- 静态化freemarker -->

<dependency>

<groupId>org.freemarker</groupId>

<artifactId>freemarker</artifactId>

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

</dependency>

<!-- Redis客户端 -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

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

</dependency>

<!-- solr客户端 -->

<dependency>

<groupId>org.apache.solr</groupId>

<artifactId>solr-solrj</artifactId>

<version>${solrj.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>

<resources>

<!-- 使用Maven部署的时候，xml和properties配置文件也一起部署到Tomcat -->

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

<filtering>false</filtering>

</resource>

<!-- 默认是以下配置 -->

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

<filtering>false</filtering>

</resource>

</resources>

</build>

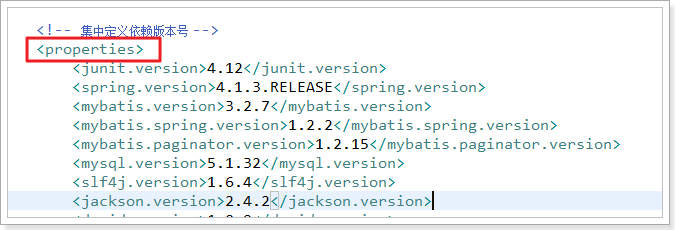
</project>

## 使用taotao-parent的好处

其他工程都需要继承taotao-parent，依赖的管理统一由taotao-parent来进行管理

### 依赖的版本号统一管理

使用properties标签统一管理依赖。



### 依赖统一管理

使用dependencyManagement标签对依赖统一管理



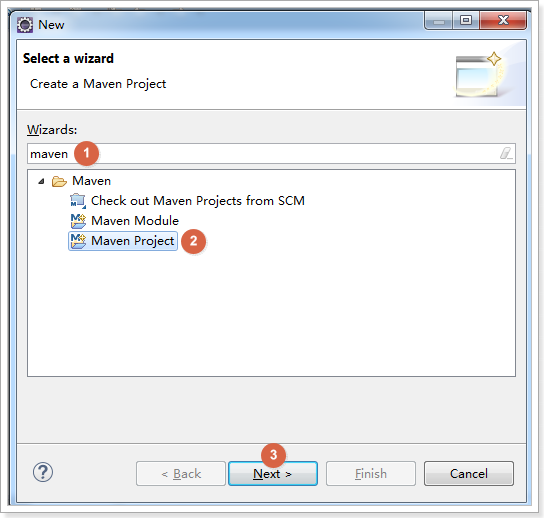
### Maven插件统一管理

对Maven的插件统一管理

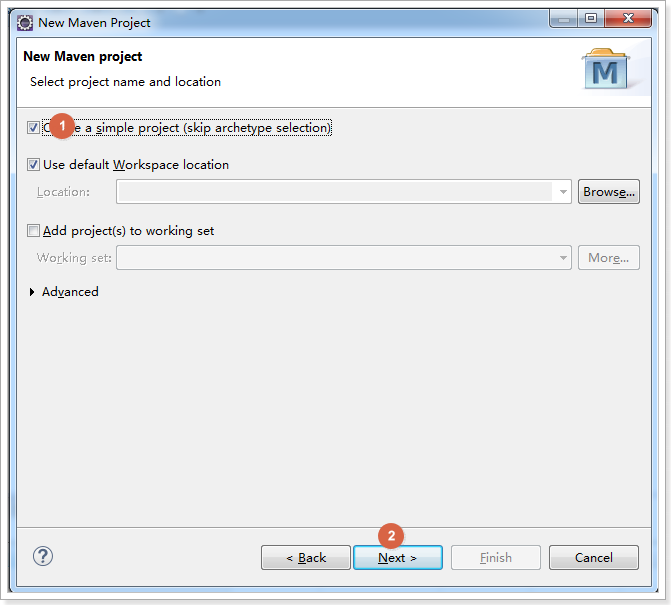


## 创建taotao-common

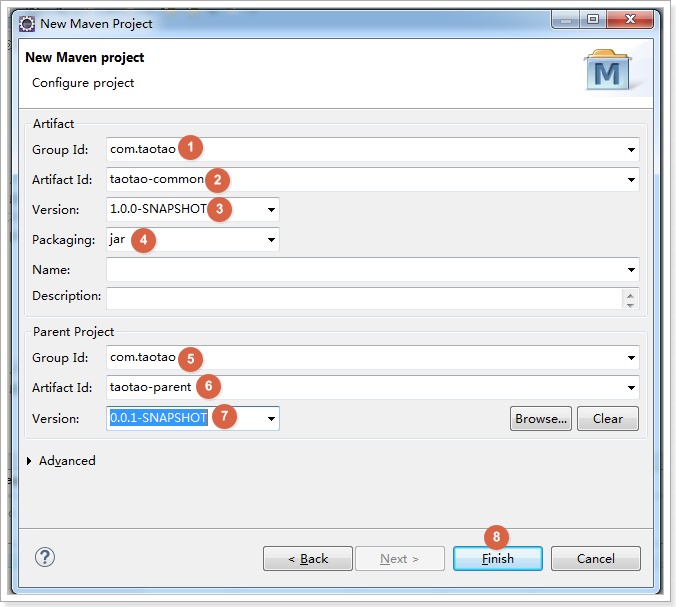
### 创建Maven工程



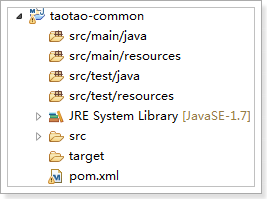
### 选择简单的Maven工程



### 填写信息



### 效果

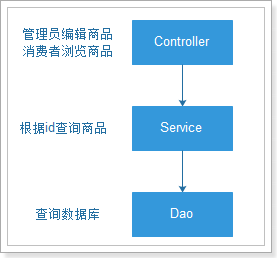


# DUBBO介绍

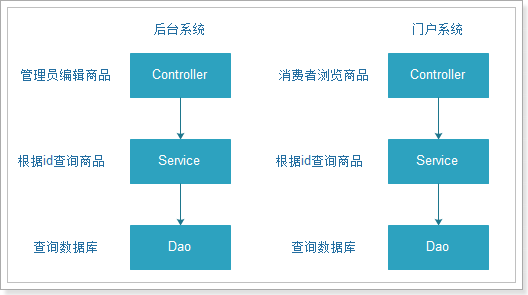
## 架构分析

原来使用集中式架构：

有耦合性高，无法水平扩展（集群）的问题



按功能进行垂直拆分：



分布式架构能解决集中式架构存在的问题，

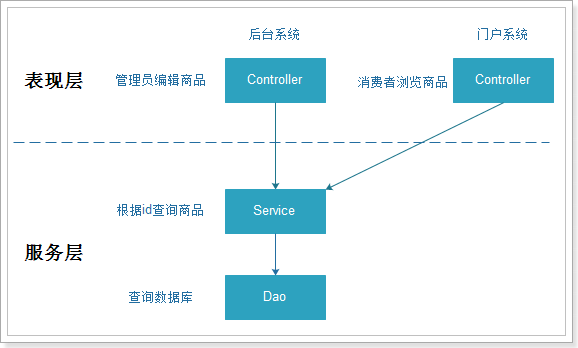
但是如果只是根据功能进行拆分，会有需要重复编写同样功能方法的问题。

如上图，两个系统都需要实现根据商品id查询商品的功能

可以再进行水平拆分：

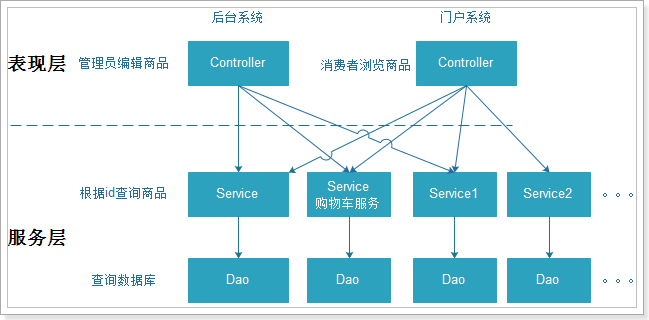
把页面控制层Controller单独拆出来，

业务层和持久层拆出来作为服务层



存在的问题：

1. 当服务越来越多，调用服务的URL就越来越难以管理
2. 当调用服务量越来越大，很难确定服务需要多少机器支撑，什么时候增加机器



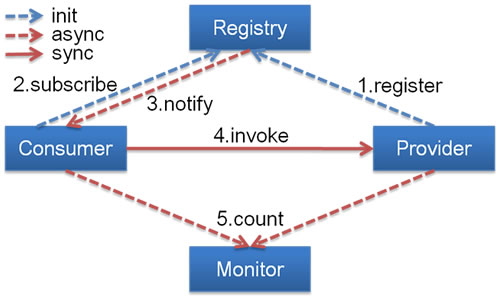
Dubbo就是资源调度和治理中心，用来解决这些问题的

## DUBBO架构

dubbo的官网：dubbo.io

DUBBO是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，是阿里巴巴SOA服务化治理方案的核心框架，每天为2,000+个服务提供3,000,000,000+次访问量支持，并被广泛应用于阿里巴巴集团的各成员站点。

下图是dubbo的架构图：



**节点角色说明：**

Provider: 暴露服务的服务提供方。

Consumer: 调用远程服务的服务消费方。

Registry: 服务注册与发现的注册中心。

Monitor: 统计服务的调用次调和调用时间的监控中心。

Container: 服务运行容器。

**调用关系说明：**

1. 服务提供者在启动时，向注册中心注册自己提供的服务。

2. 服务消费者在启动时，向注册中心订阅自己所需的服务。

3. 注册中心返回服务提供者地址列表给消费者，如果有变更，注册中心将基于长连接推送变更数据给消费者。

4. 服务消费者，从提供者地址列表中，基于软负载均衡算法，选一台提供者进行调用，如果调用失败，再选另一台调用。

5. 服务消费者和提供者，在内存中累计调用次数和调用时间，定时每分钟发送一次统计数据到监控中心。

# 搭建后台系统

后台管理系统前端UI是使用EasyUI框架

主要功能是实现商品管理，网站内容管理等

## 聚合工程分析

用原来的方式

com.taotao.manager.pojo javaBean

com.taotao.manager.mapper dao层，数据操作层

com.taotao.manager.service service层，业务处理层

com.taotao.manager.controller 页面控制层

现在是分布式架构，至少有两个系统，一个是前台系统商品详情展示，另一个是后台系统的商品管理功能

这两个功能都需要操作商品，那么都需要item这个javaBean

存在的问题：

1. 两个一模一样，代码重用性差
2. 维护成本高，一旦变更，需要修改所有的item

使用聚合工程

taotao-manager 聚合工程的父工程，负责把所有子工程聚合在一起

taotao-manager-pojo 聚合工程的子工程，javaBean

taotao-manager-mapper 聚合工程的子工程，DAO层

taotao-manager-service 聚合工程的子工程，业务处理层

taotao-manager-controller 聚合工程的子工程，页面控制层

可以在后台管理系统的taotao-manager-pojo的子工程中加入item这个JavaBean

其他系统需要使用这个JavaBean的时候，只需要依赖taotao-manager-pojo这个子工程就可以了

这样就解决了上面的问题

## Dubbo的Spring配置

Dubbo采用全Spring配置方式，透明化接入应用，对应用没有任何API侵入，只需用Spring加载Dubbo的配置即可。

在提供方增加暴露服务配置<dubbo:service>，

在消费方增加引用服务配置<dubbo:reference>。

发布服务：

<!-- 和本地服务一样实现远程服务 -->

<bean id="xxxService" class="com.xxx.XxxServiceImpl" />

<!-- 增加暴露远程服务配置 -->

<dubbo:service interface="com.xxx.XxxService" ref="xxxService" />

调用服务：

<!-- 增加引用远程服务配置 -->

<dubbo:reference id="xxxService" interface="com.xxx.XxxService" />

<!-- 和本地服务一样使用远程服务 -->

<bean id="xxxAction" class="com.xxx.XxxAction">

<property name="xxxService" ref="xxxService" />

</bean>

简单来说，和之前的开发项目，代码没有变化，只是比之前多了dubbo的配置而已。

## 创建taotao-manager服务

创建后台管理系统taotao-manager服务层

根据之前的分析，需要创建的工程：

taotao-manager 父工程

taotao-manager-pojo 子工程，pojo

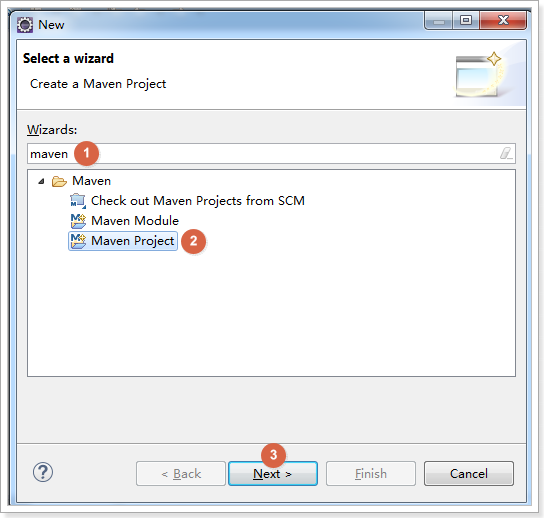
taotao-manager-mapper 子工程，Dao持久层

taotao-manager-interfacer 子工程，Service接口，服务层和表现层都需要 jar

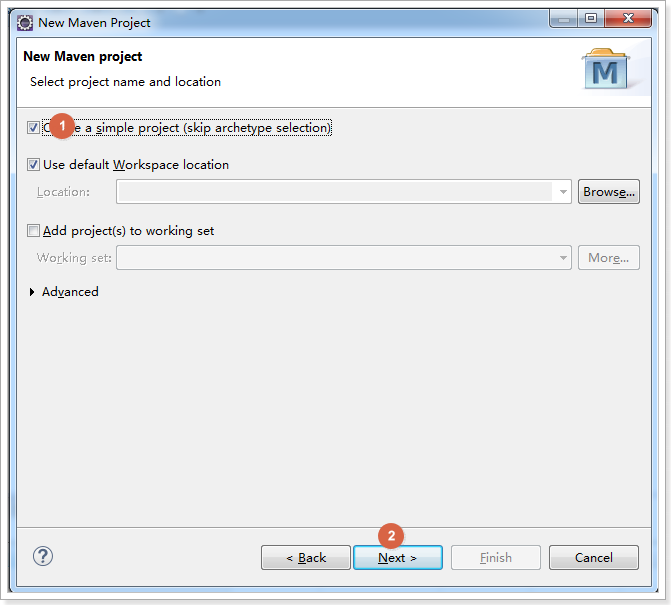
taotao-manager-service 子工程，Service业务层 war

### taotao-manager父工程

#### 创建Maven工程



#### 选择简单Maven工程



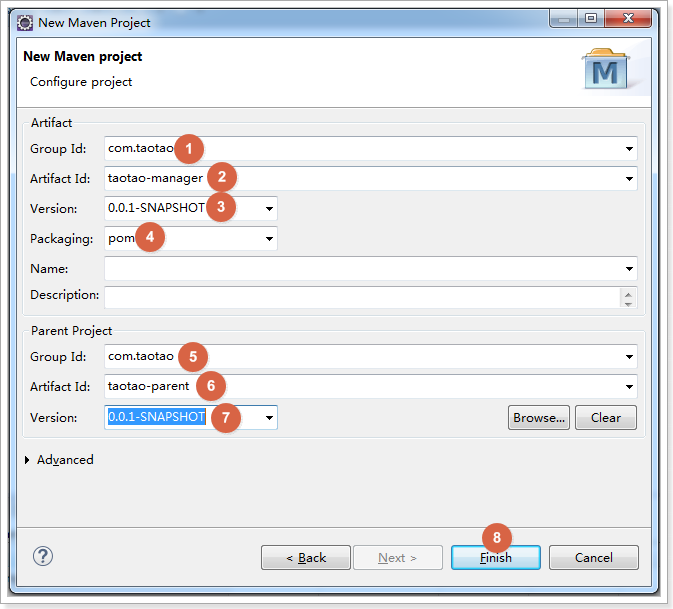
#### 填写信息

工程的Packaging打包（下图的第4步）：

创建taotao-parent和聚合工程的父工程时，打pom包

创建web工程，需要部署的时候打war包

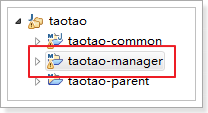
其他工程打jar包



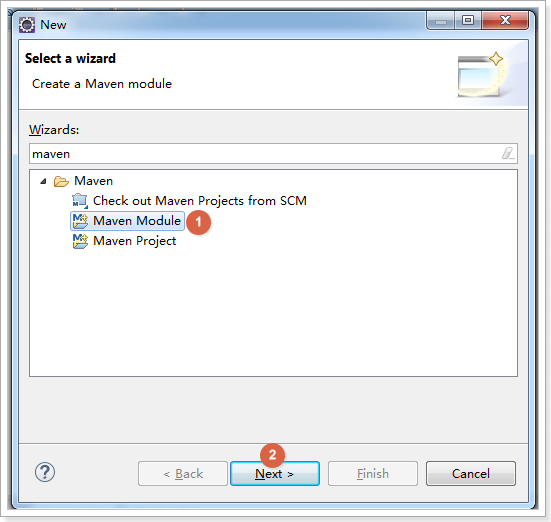
### taotao-manager-pojo子工程

#### 选择Maven组件

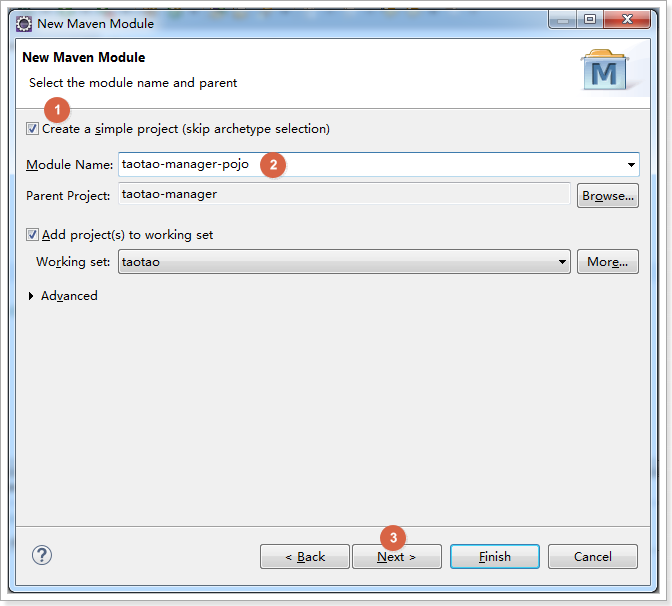
先点击选中聚合工程父工程



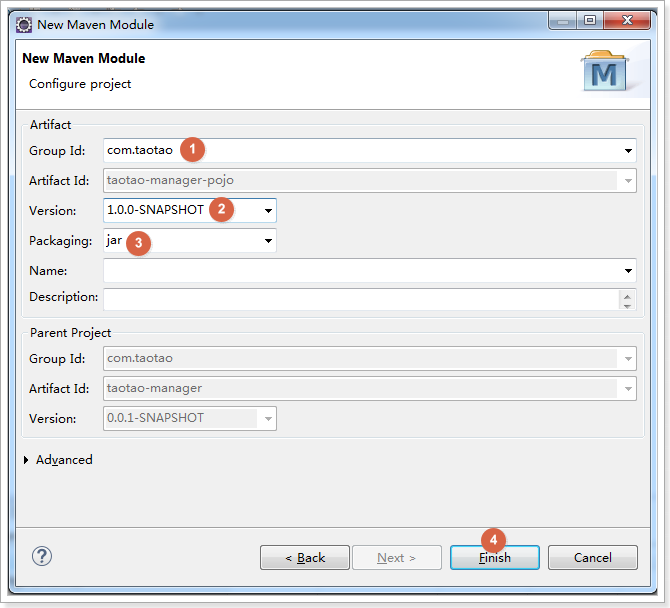
再按Ctrl+N准备新建，选择Maven Module



#### 选择简单工程并填写组件名称



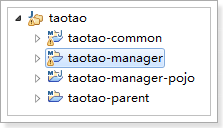
#### 填写子工程信息



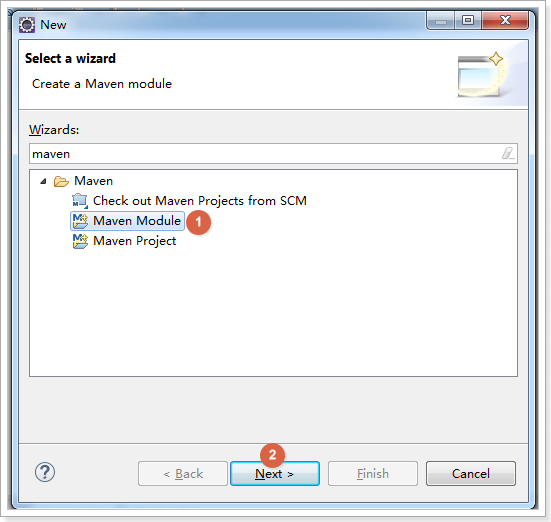
### taotao-manager-mapper子工程

#### 选择Maven组件

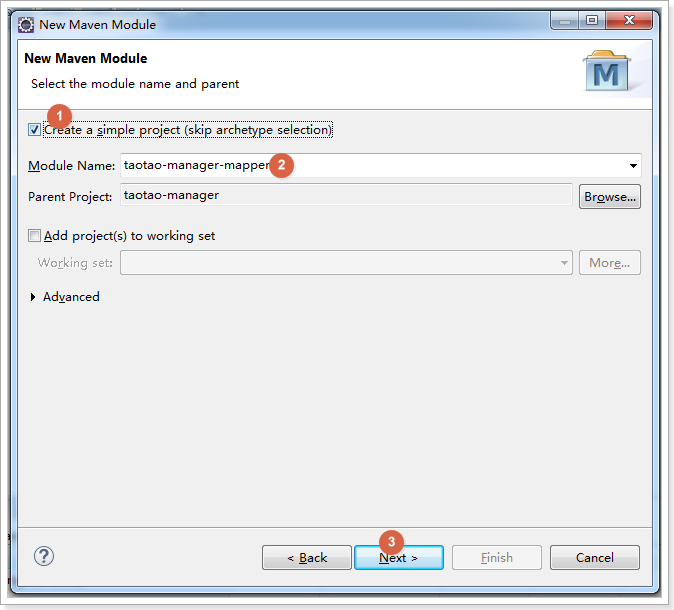
先点击选中聚合工程父工程



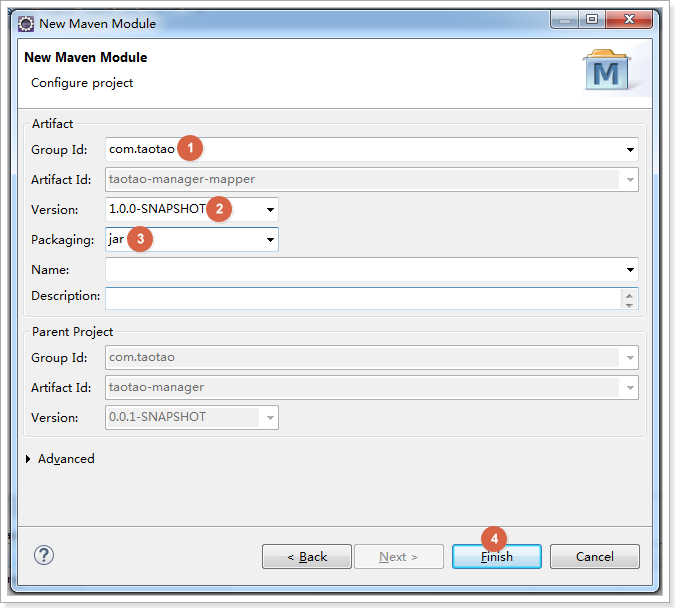
再按Ctrl+N准备新建，选择Maven Module



#### 选择简单工程并填写组件名称



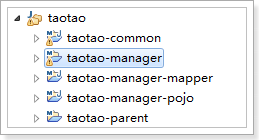
#### 填写子工程信息



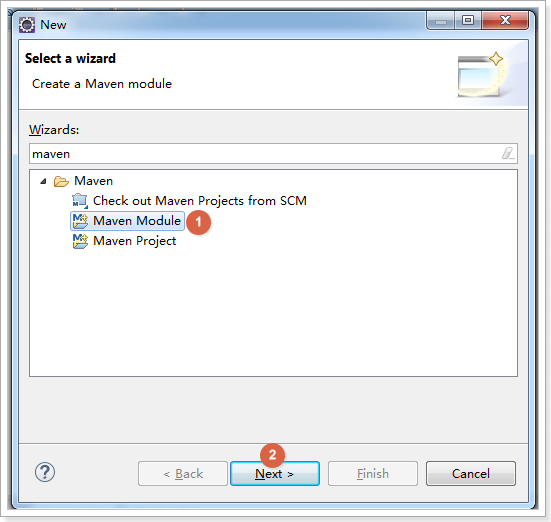
### taotao-manager-interface子工程

#### 选择Maven组件

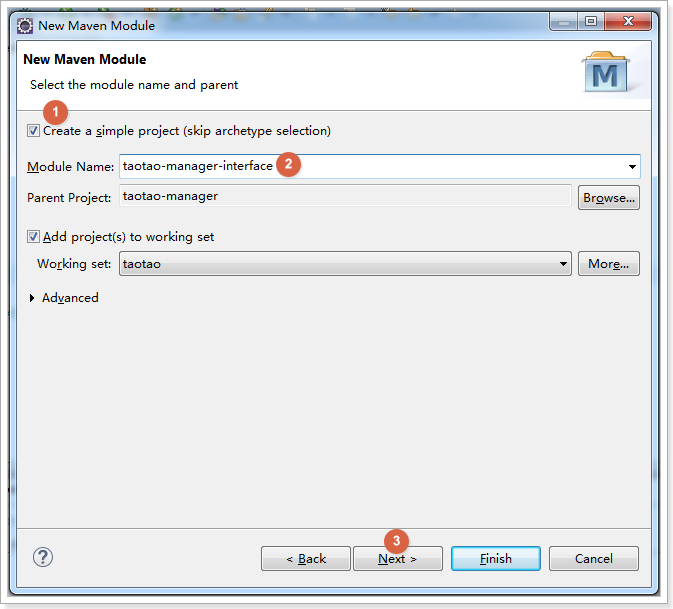
先点击选中聚合工程父工程



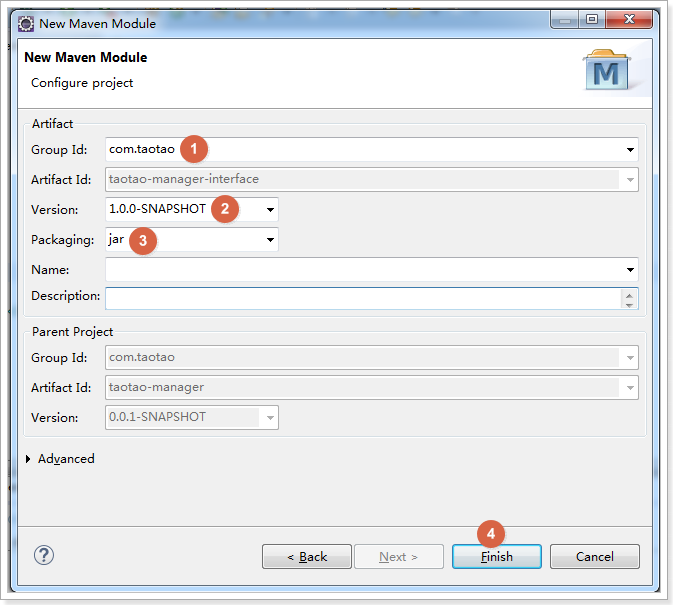
再按Ctrl+N准备新建，选择Maven Module



#### 选择简单工程并填写组件名称



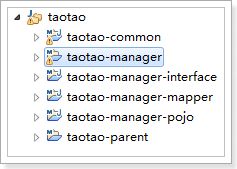
#### 填写子工程信息



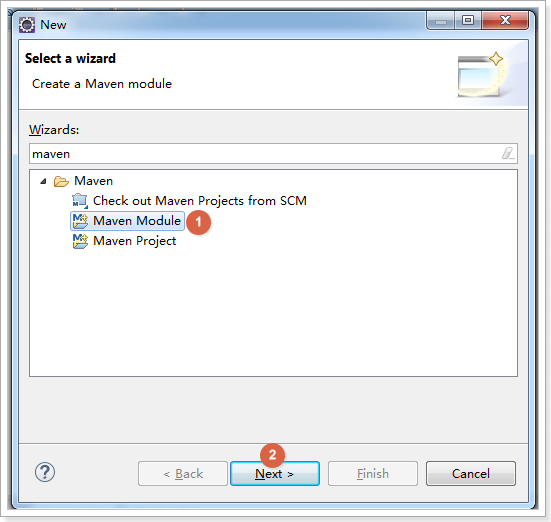
### taotao-manager-service子工程

#### 选择Maven组件

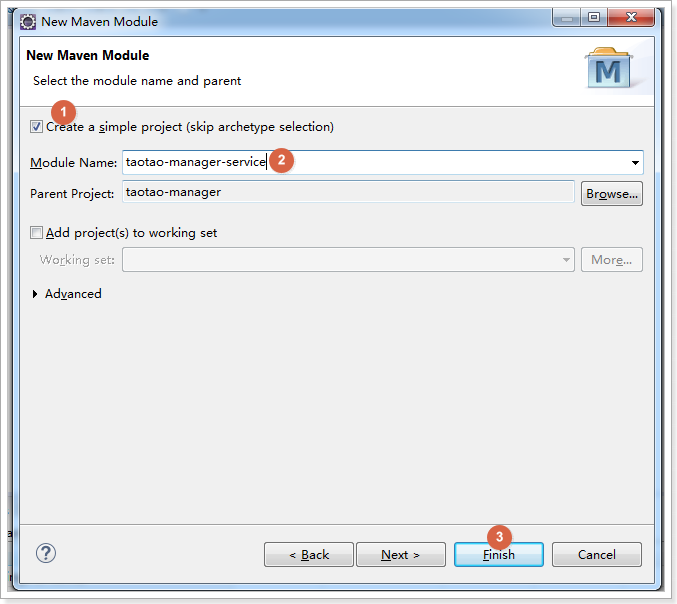
先点击选中聚合工程父工程



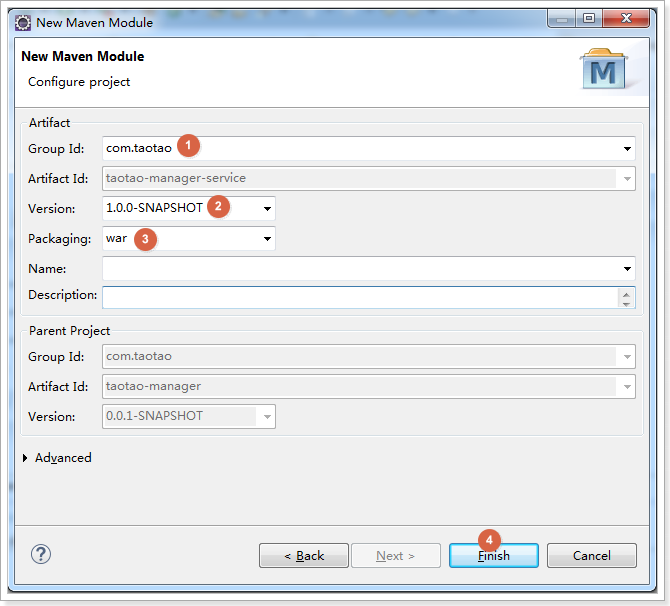
再按Ctrl+N准备新建，选择Maven Module



#### 选择简单工程并填写组件名称

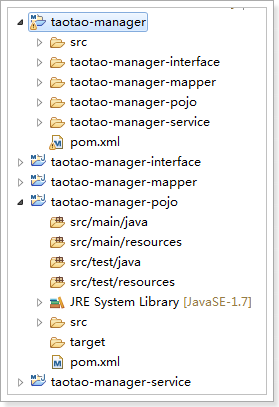


#### 填写子工程信息（需要打war包）



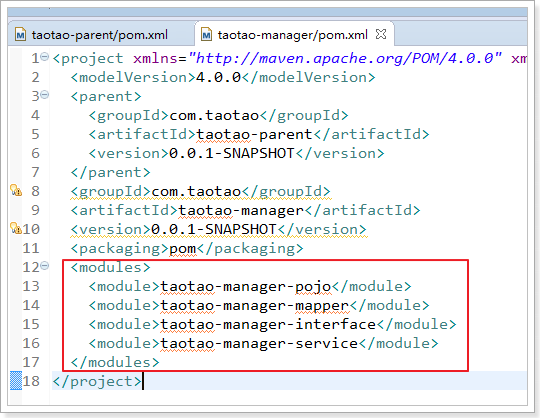
### 创建效果

#### 项目结构



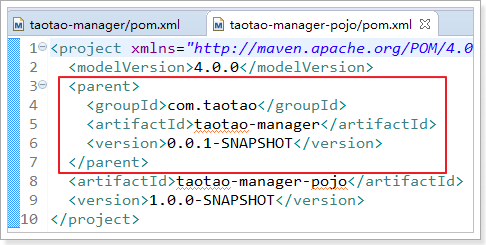
#### 父工程pom.xml

taotao-manager父工程的pom.xml自动增加了4个子工程的配置



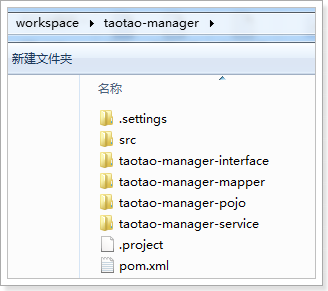
#### 子工程pom.xml

自动继承父工程



#### 物理存储结构

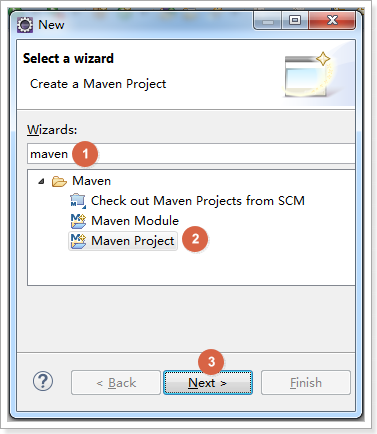
子工程放在父工程文件夹内

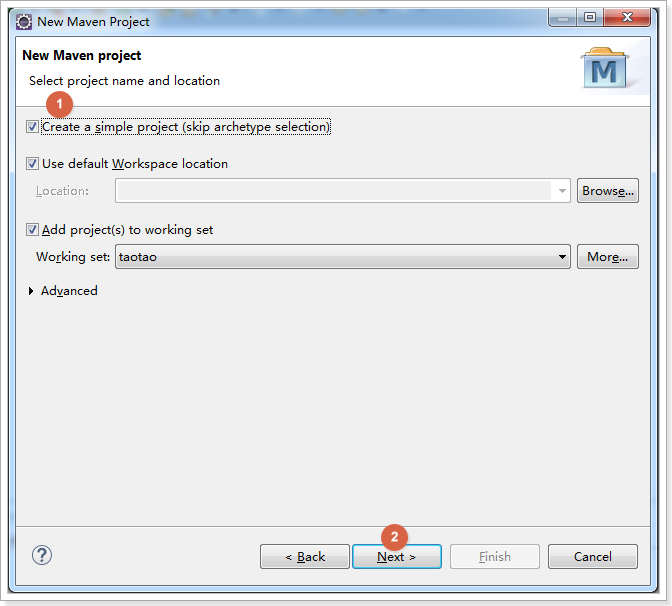


## 创建taotao-manager-web表现层

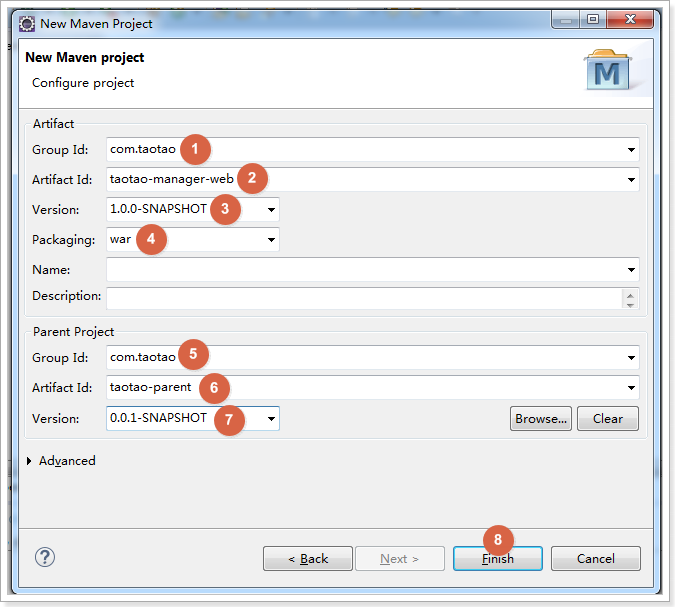
表现层是原来的Controller，不需要聚合工程，直接搭建工程即可

### 创建Maven工程





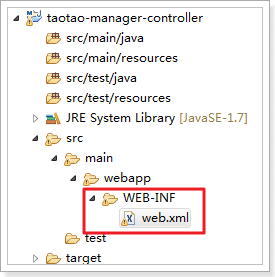
#### 填写子工程信息（需要打war包）



#### 没有web.xml文件

所有打war包的工程都需要web.xml，但是eclipse没有创建web.xml文件

需要创建WEB-INF文件夹，并在其内创建web.xml文件



## 加入依赖

添加依赖的原则：

使用的最底层添加依赖

服务层是聚合工程

父工程是所有工程父，父工程的依赖，所有子工程都能用

四个子工程的依赖关系

service>>mapper>>pojo

service>>interface>>pojo

表现层是单独的Maven工程。

不需要操作数据库，所以相关的MySQL、mybatis的依赖都不需要

需要配置dubbo，配置的时候需要使用接口，需要taotao-manager-interface依赖

### taotao-manager父工程

父工程加入所有工程都会使用的依赖和Tomcat插件

<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-manager</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

<modules>

<module>taotao-manager-pojo</module>

<module>taotao-manager-mapper</module>

<module>taotao-manager-interface</module>

<module>taotao-manager-service</module>

</modules>

<dependencies>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

<!-- 日志 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</dependency>

<!-- Apache工具组件 -->

<dependency>

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

<artifactId>commons-lang3</artifactId>

</dependency>

<dependency>

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

<artifactId>commons-io</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

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

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

<configuration>

<!-- http://127.0.0.1:${port}/${path} -->

<port>80</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

</project>

### taotao-manager-pojo子工程

<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-manager</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>taotao-manager-pojo</artifactId>

<version>1.0.0-SNAPSHOT</version>

</project>

### taotao-manager-mapper子工程

<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-manager</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>taotao-manager-mapper</artifactId>

<version>1.0.0-SNAPSHOT</version>

<dependencies>

<!-- taotao-manager-pojo依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-manager-pojo</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<!-- Mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

</dependency>

<!-- MySql -->

<dependency>

<groupId>mysql</groupId>

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

</dependency>

<!-- 连接池 -->

<dependency>

<groupId>com.jolbox</groupId>

<artifactId>bonecp-spring</artifactId>

</dependency>

</dependencies>

</project>

### taotao-manager-interface子工程

<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-manager</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>taotao-manager-interface</artifactId>

<version>1.0.0-SNAPSHOT</version>

<dependencies>

<!-- taotao-manager-pojo依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-manager-pojo</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

</dependencies>

</project>

### taotao-manager-service子工程

<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-manager</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>taotao-manager-service</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- taotao-manager-interface依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-manager-interface</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<!-- taotao-manager-mapper依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-manager-mapper</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<!-- spring -->

<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>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

</dependency>

<!-- Jackson Json处理工具包 -->

<dependency>

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

<artifactId>jackson-databind</artifactId>

</dependency>

<!-- dubbo相关 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

</dependency>

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

</dependency>

</dependencies>

</project>

### taotao-manager-web工程

不需要操作数据库，不需要MySQL、MyBatis等依赖

需要taotao-manager-interface

<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-manager-web</artifactId>

<version>1.0.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- taotao-manager-interface依赖 -->

<dependency>

<groupId>com.taotao</groupId>

<artifactId>taotao-manager-interface</artifactId>

<version>1.0.0-SNAPSHOT</version>

</dependency>

<!-- 单元测试 -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<scope>test</scope>

</dependency>

<!-- Spring -->

<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>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

</dependency>

<!-- 日志 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</dependency>

<!-- Jackson Json处理工具包 -->

<dependency>

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

<artifactId>jackson-databind</artifactId>

</dependency>

<!-- JSP相关 -->

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<scope>provided</scope>

</dependency>

<!-- Apache工具组件 -->

<dependency>

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

<artifactId>commons-lang3</artifactId>

</dependency>

<dependency>

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

<artifactId>commons-io</artifactId>

</dependency>

<!-- dubbo相关 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>dubbo</artifactId>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

</dependency>

<dependency>

<groupId>com.github.sgroschupf</groupId>

<artifactId>zkclient</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

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

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

<configuration>

<port>8080</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</build>

</project>

## 加入配置文件

**taotao-manager服务：**

Dao层：

1. SqlMapConfig.xml，空文件即可，但是需要文件头。
2. applicationContext-dao.xml
   1. 数据库连接池
   2. SqlSessionFactory，需要spring和mybatis整合包下的。
   3. 配置mapper文件扫描器。

Service层：

1. applicationContext-service.xml包扫描器，dubbo配置
2. applicationContext-trans.xml配置事务。

web.xml文件：

1、配置spring

**taotao-manager-web表现层：**

Controller层：

1. Springmvc.xml
   1. 包扫描器，扫描@Controller注解的类。
   2. 配置注解驱动
   3. 配置视图解析器
   4. dubbo配置

web.ml文件：

1、解决POST乱码过滤器

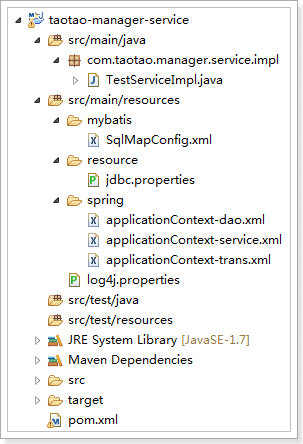
2、配置SpringMVC前端控制器。

### taotao-manager服务层配置

#### 配置文件结构

在taotao-manager-service子工程中进行配置，

在src/main/resources里创建mybatis和spring文件夹存放配置文件。resource存放资源文件



#### SqlMapConfig.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>

</configuration>

#### 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: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"*>

<!-- 配置 读取properties文件 jdbc.properties -->

<context:property-placeholder location=*"classpath:resource/jdbc.properties"* />

<!-- 配置数据源 -->

<bean id=*"dataSource"* class=*"com.jolbox.bonecp.BoneCPDataSource"*

destroy-method=*"close"*>

<!-- 数据库驱动 -->

<property name=*"driverClass"* value=*"${jdbc.driver}"* />

<!-- 相应驱动的jdbcUrl -->

<property name=*"jdbcUrl"* value=*"${jdbc.url}"* />

<!-- 数据库的用户名 -->

<property name=*"username"* value=*"${jdbc.username}"* />

<!-- 数据库的密码 -->

<property name=*"password"* value=*"${jdbc.password}"* />

<!-- 检查数据库连接池中空闲连接的间隔时间，单位是分，默认值：240，如果要取消则设置为0 -->

<property name=*"idleConnectionTestPeriod"* value=*"60"* />

<!-- 连接池中未使用的链接最大存活时间，单位是分，默认值：60，如果要永远存活设置为0 -->

<property name=*"idleMaxAge"* value=*"30"* />

<!-- 每个分区最大的连接数 -->

<property name=*"maxConnectionsPerPartition"* value=*"150"* />

<!-- 每个分区最小的连接数 -->

<property name=*"minConnectionsPerPartition"* value=*"5"* />

</bean>

<!-- 配置SqlSessionFactory -->

<bean class=*"org.mybatis.spring.SqlSessionFactoryBean"*>

<!-- 配置数据源 -->

<property name=*"dataSource"* ref=*"dataSource"* />

<!-- 加载MyBatis配置文件 -->

<property name=*"configLocation"* value=*"classpath:mybatis/SqlMapConfig.xml"* />

</bean>

<!-- 配置Mapper扫描 -->

<bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>

<!-- Mapper扫描包 -->

<property name=*"basePackage"* value=*"com.taotao.manager.mapper"* />

</bean>

</beans>

#### jdbc.properties

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://127.0.0.1:3306/taotao?useUnicode=true&characterEncoding=utf8&autoReconnect=true

jdbc.username=root

jdbc.password=root

#### log4j.properties

log4j.rootLogger=DEBUG,A1

log4j.logger.com.taotao = DEBUG

log4j.logger.org.mybatis = DEBUG

log4j.appender.A1=org.apache.log4j.ConsoleAppender

log4j.appender.A1.layout=org.apache.log4j.PatternLayout

log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c]-[%p] %m%n

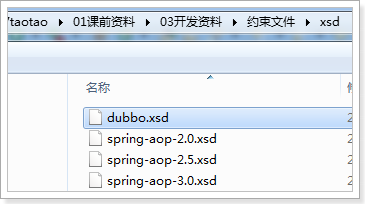
#### applicationContext-service.xml

需要配置dubbo

这里是配置dubbo的服务层声明。只需要配置<dubbo:service />即可。

加入dubbo需要两步

1. 在配置文件头部配置dubbo约束
2. 加入dubbo.xsd约束文件



配置文件内容：

<?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"* xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.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"*>

<!-- 配置service扫描 -->

<context:component-scan base-package=*"com.taotao.manager.service"* />

<!-- 配置dubbo -->

<!-- 提供方应用信息，用于计算依赖关系 -->

<dubbo:application name=*"taotao-manager-service"* />

<!-- 使用multicast广播注册中心暴露服务地址 -->

<dubbo:registry address=*"multicast://224.5.6.7:1234"* />

<!-- 用dubbo协议在20880端口暴露服务 -->

<dubbo:protocol name=*"dubbo"* port=*"20880"* />

<!-- 声明需要暴露的服务接口 -->

<!-- <dubbo:service interface="com.taotao.manager.service.ItemCatService" -->

<!-- ref="itemCatServiceImpl" /> -->

</beans>

#### applicationContext-trans.xml

<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"*>

<!-- 定义事务管理器 -->

<bean id=*"transactionManager"*

class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

<property name=*"dataSource"* ref=*"dataSource"* />

</bean>

<!-- 定义事务策略 -->

<tx:advice id=*"txAdvice"* transaction-manager=*"transactionManager"*>

<tx:attributes>

<!-- 传播行为 -->

<tx:method name=*"save\*"* propagation=*"REQUIRED"* />

<tx:method name=*"insert\*"* propagation=*"REQUIRED"* />

<tx:method name=*"add\*"* propagation=*"REQUIRED"* />

<tx:method name=*"create\*"* propagation=*"REQUIRED"* />

<tx:method name=*"delete\*"* propagation=*"REQUIRED"* />

<tx:method name=*"update\*"* propagation=*"REQUIRED"* />

<tx:method name=*"find\*"* propagation=*"SUPPORTS"* read-only=*"true"* />

<tx:method name=*"select\*"* propagation=*"SUPPORTS"* read-only=*"true"* />

<tx:method name=*"get\*"* propagation=*"SUPPORTS"* read-only=*"true"* />

<tx:method name=*"query\*"* propagation=*"SUPPORTS"* read-only=*"true"* />

</tx:attributes>

</tx:advice>

<aop:config>

<aop:pointcut id=*"myPointcut"*

expression=*"execution(\* com.taotao.manager.service.\*.\*(..))"* />

<!--将定义好的事务处理策略应用到上述的切入点 -->

<aop:advisor advice-ref=*"txAdvice"* pointcut-ref=*"myPointcut"* />

</aop:config>

</beans>

#### 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=*"MyWebApp"* version=*"2.5"*>

<display-name>taotao-manager</display-name>

<welcome-file-list>

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

</welcome-file-list>

<!-- 配置spring -->

<context-param>

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

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

</context-param>

<!-- 配置spring的监听器 -->

<listener>

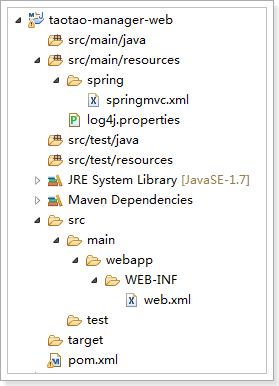
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

### taotao-manager-web表现层配置

#### 配置文件结构



#### 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"* xmlns:dubbo=*"http://code.alibabatech.com/schema/dubbo"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://code.alibabatech.com/schema/dubbo http://code.alibabatech.com/schema/dubbo/dubbo.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"*>

<!-- 配置controller扫描 -->

<context:component-scan base-package=*"com.taotao.manager.controller"* />

<!-- 配置注解驱动 -->

<mvc:annotation-driven />

<!-- 配置视图解析器 ,配置前缀和后缀 -->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*"/WEB-INF/views/"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

<!-- 配置dubbo服务 -->

<dubbo:application name=*"taotao-manager-web"* />

<!-- 使用广播 -->

<dubbo:registry address=*"multicast://224.5.6.7:1234"* />

<!-- 声明要调用的服务，timeout是设置连接超时最长时间,如果不设置，超时时间默认是3秒 -->

<!-- <dubbo:reference interface="com.taotao.manager.service.ItemCatService" -->

<!-- id="itemCatService" timeout="1000000" /> -->

</beans>

#### log4j.properties

log4j.rootLogger=DEBUG,A1

log4j.logger.com.taotao = DEBUG

log4j.logger.org.mybatis = DEBUG

log4j.appender.A1=org.apache.log4j.ConsoleAppender

log4j.appender.A1.layout=org.apache.log4j.PatternLayout

log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss,SSS} [%t] [%c]-[%p] %m%n

#### 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=*"MyWebApp"* version=*"2.5"*>

<display-name>taotao-manager-web</display-name>

<welcome-file-list>

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

</welcome-file-list>

<!-- 解决POST乱码问题 -->

<filter>

<filter-name>encoding</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>encoding</filter-name>

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

</filter-mapping>

<!-- 配置springMVC前端控制器 -->

<servlet>

<servlet-name>taotao-manager-web</servlet-name>

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

<init-param>

<!-- springMVC全局配置文件 -->

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

<param-value>classpath:spring/springmvc.xml</param-value>

</init-param>

<!-- springmvc随着容器的启动而启动 -->

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

</servlet>

<servlet-mapping>

<servlet-name>taotao-manager-web</servlet-name>

<!-- 所有以rest开头的请求进入SpringMVC -->

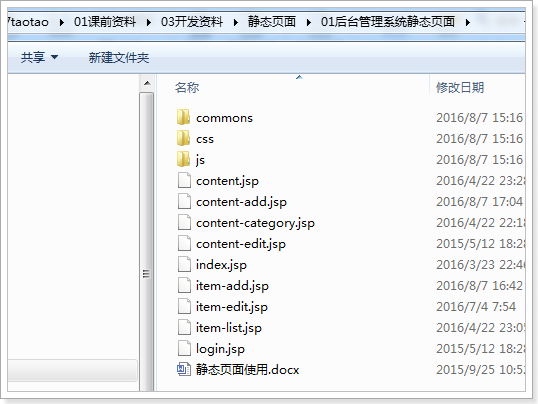
<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

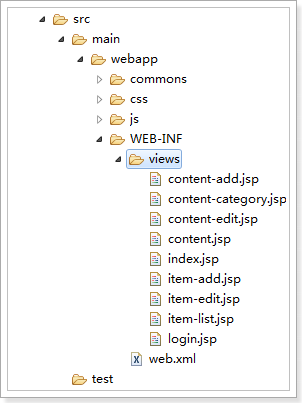
</web-app>

## 加入静态资源

加入课前资料中的静态资源



效果



## 测试dubbo

服务层创建Mapper、Service，表现层创建Controller，使用浏览器访问，确认是否能正常调用

PS：建议在断网，且只开启一个有线网卡的情况下进行测试（关闭虚拟网卡、无线网卡等）。项目第二天会讲解原因及解决办法。

### 服务层开发

#### 编写TestMapper接口

在taotao-manager-mapper子工程编写

**public** **interface** TestMapper {

/\*\*

\* 从数据库获取当前时间

\* **@return**

\*/

String queryDate();

}

#### 编写TestMapper.xml配置文件

在taotao-manager-mapper子工程编写

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"com.taotao.manager.mapper.TestMapper"*>

<!-- 获取数据库当前时间 -->

<select id=*"queryDate"* resultType=*"string"*>

SELECT NOW()

</select>

</mapper>

#### 编写TestService接口

在taotao-manager-interface子工程编写

**public** **interface** TestService {

/\*\*

\* 获取当前时间

\*

\* **@return**

\*/

String queryDate();

}

#### 编写TestServiceImpl实现类

在taotao-manager-service子工程编写

@Service

**public** **class** TestServiceImpl **implements** TestService {

@Autowired

**private** TestMapper testMapper;

@Override

**public** String queryDate() {

**return** **this**.testMapper.queryDate();

}

}

#### 发布服务

在服务层taotao-manager-service配置文件中发布服务



### 表现层开发

#### 编写TestController

表现层开发

在taotao-manager-web工程编写

@Controller

@RequestMapping("test")

**public** **class** TestController {

@Autowired

**private** TestService testService;

/\*\*

\* 查询当前时间

\*

\* **@return**

\*/

@RequestMapping("date")

@ResponseBody

**public** String queryDate() {

**return** **this**.testService.queryDate();

}

}

#### 声明服务调用

在表现层taotao-manager-web配置文件中声明服务调用



### 出现错误



### 测试效果

