# 天天商城\_Day01

**一.电商行业**

## 电商行业发展

近年来，中国的电子商务快速发展，交易额连创新高，电子商务在各领域的应用不断拓展和深化、相关服务业蓬勃发展、支撑体系不断健全完善、创新的动力和能力 不断增强。电子商务正在与实体经济深度融合，进入规模性发展阶段，对经济社会生活的影响不断增大，正成为我国经济发展的新引擎。

中国电子商务研究中心数据显示，截止到2012年底，中国电子商务市场交易规模达7.85万亿人民币，同比增长30.83%。其中，B2B电子商务交易额 达6.25万亿，同比增长27%。而2011年全年，中国电子商务市场交易额达6万亿人民币，同比增长33%，占GDP比重上升到13%；2012年，电 子商务占GDP的比重已经高达15%。预计2013年我国电子商务规模将突破十万亿大关。

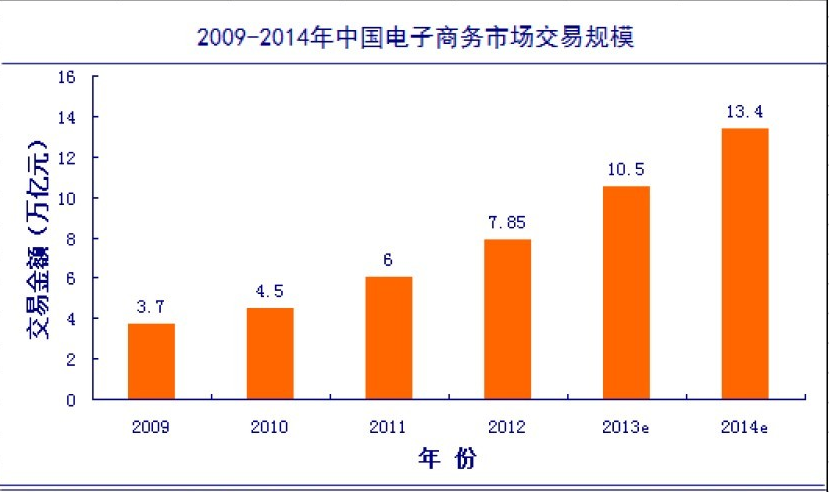
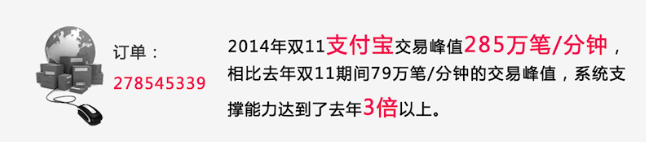


图1.2009-2014年中国电子商务市场交易规模（万亿元）

* 1. **双十一11.11**





* 1. **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

**二.天天商城**

**2.1.天天商城简介**

天天网上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、运营可以在平台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。

**2.2.功能架构**

**2.2.1系统功能图**



**2.2.2功能描述**

后台管理系统：管理商品、订单、类目、商品规格属性、用户管理以及内容发布等功能。

前台系统：用户可以在前台系统中进行注册、登录、浏览商品、首页、下单等操作。

会员系统：用户可以在该系统中查询已下的订单、收藏的商品、我的优惠券、团购等信息。

订单系统：提供下单、查询订单、修改订单状态、定时处理订单。

搜索系统：提供商品的搜索功能。

单点登录系统：为多个系统之间提供用户登录凭证以及查询登录用户的信息。

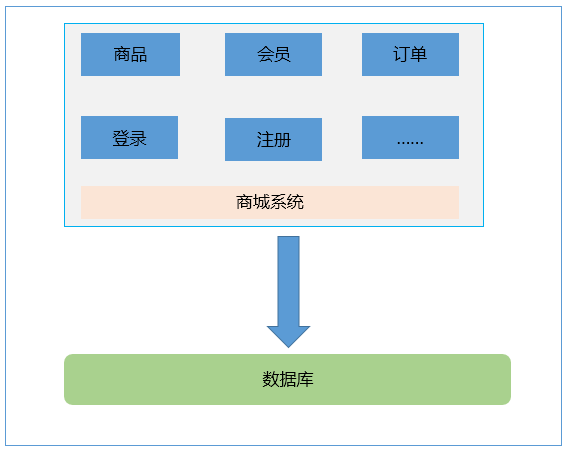
工程结构：

后台：用maven,ssm

创建一个父工程:对依赖的jar包的版本进行管理 ,打包方式必须是pom

**2.3技术架构**

**2.3.1传统架构**



思考：有什么问题？

1. 模块之间耦合度太高，其中一个升级其他都得升级
2. 开发困难，各个团队开发最后都要整合一起
3. 系统的扩展性差

4、不能灵活的进行分布式部署。

**2.3.2分布式系统架构**



分布式架构：

把系统按照模块拆分成多个子系统。

优点：

1. 把模块拆分，使用接口通信，降低模块之间的耦合度。
2. 把项目拆分成若干个子项目，不同的团队负责不同的子项目。
3. 增加功能时只需要再增加一个子项目，调用其他系统的接口就可以。
4. 可以灵活的进行分布式部署。

缺点：

系统之间交互需要使用远程通信，接口开发增加工作量。

**2.3.3技术选型（主要技术）**

* Spring、SpringMVC、Mybatis
* JSP、JSTL、jQuery、jQuery plugin、EasyUI、KindEditor（富文本编辑器）、CSS+DIV
* Redis（缓存服务器）mongdb
* Solr（搜索），freemarker(页面静态化)
* httpclient（调用系统服务）jsonp(跨域访问)
* Mysql
* Nginx（web服务器，负载均衡）
* ActiveMQ,QuartZ

**2.4开发工具和环境**

Eclipse 4.6.3(Neno)，自带maven插件，需要手工安装svn插件。

Maven 3.3.3（开发工具自带）

Tomcat 7.0.53（Maven Tomcat Plugin）

JDK 1.8

Mysql 5.6

Nginx 1.8.0

Redis 3.0.0

Win7 操作系统

SVN（版本管理）

**2.5人员配置**

产品经理：2人，确定需求以及给出产品原型图。

项目经理：1人，项目管理。

前端团队：3人，根据产品经理给出的原型制作静态页面。

后端团队：5人，实现产品功能。

测试团队：3人，测试所有的功能。

运维团队：3人，项目的发布以及维护。

**三.后台管理系统工程结构**

## 3.1 maven管理的好处

1、项目构建。Maven定义了软件开发的整套流程体系，并进行了封装，开发人员只需要指定项目的构建流程，无需针对每个流程编写自己的构建脚本。

2、依赖管理。除了项目构建，Maven最核心的功能是软件包的依赖管理，能够自动分析项目所需要的依赖软件包，并到Maven中心仓库去下载。

A)管理依赖的jar包

B)管理工程之间的依赖关系。

## 3.2Maven本地仓库

在当前系统用户的文件夹下。例如当前用户是Administrator那么本地仓库就是在

C:\Users\Administrator\.m2目录下。

只需要用老师提供的.m2覆盖本地的就可以。

Maven插件使用eclipse mars自带maven插件。只需要统一开发环境。

## 3.3依赖管理

传统工程结构：

商品管理

Controller

com.taotao.controller

Service

com.taotao.service

Mapper

com.taotao.mapper

Pojo

com.taotao.pojo

Maven管理的工程结构：

不使用maven：工程部署时需要手动复制jar包,完成工程构建,非常繁琐。

使用maven进行工程构建：

使用maven可以实现一步构建。

## 3.4后台管理系统的工程结构

。。。。。。

前台工程

Tt-manager(pom工程)聚合工程

Tt-parent(父工程pom类型)

Tt-common(工具类,jar类型)

Tt-pojo(jar包)

Tt-mapper(jar包)

Tt-service(jar包)

Tt-controller(war包)

继承：

依赖：

后台管理系统工程结构：

tt-parent -- 管理依赖jar包的版本，全局，公司级别

|--tt-common --- 通用组件、工具类

|--tt-manage --- 后台系统

|--com.tt.manage.web

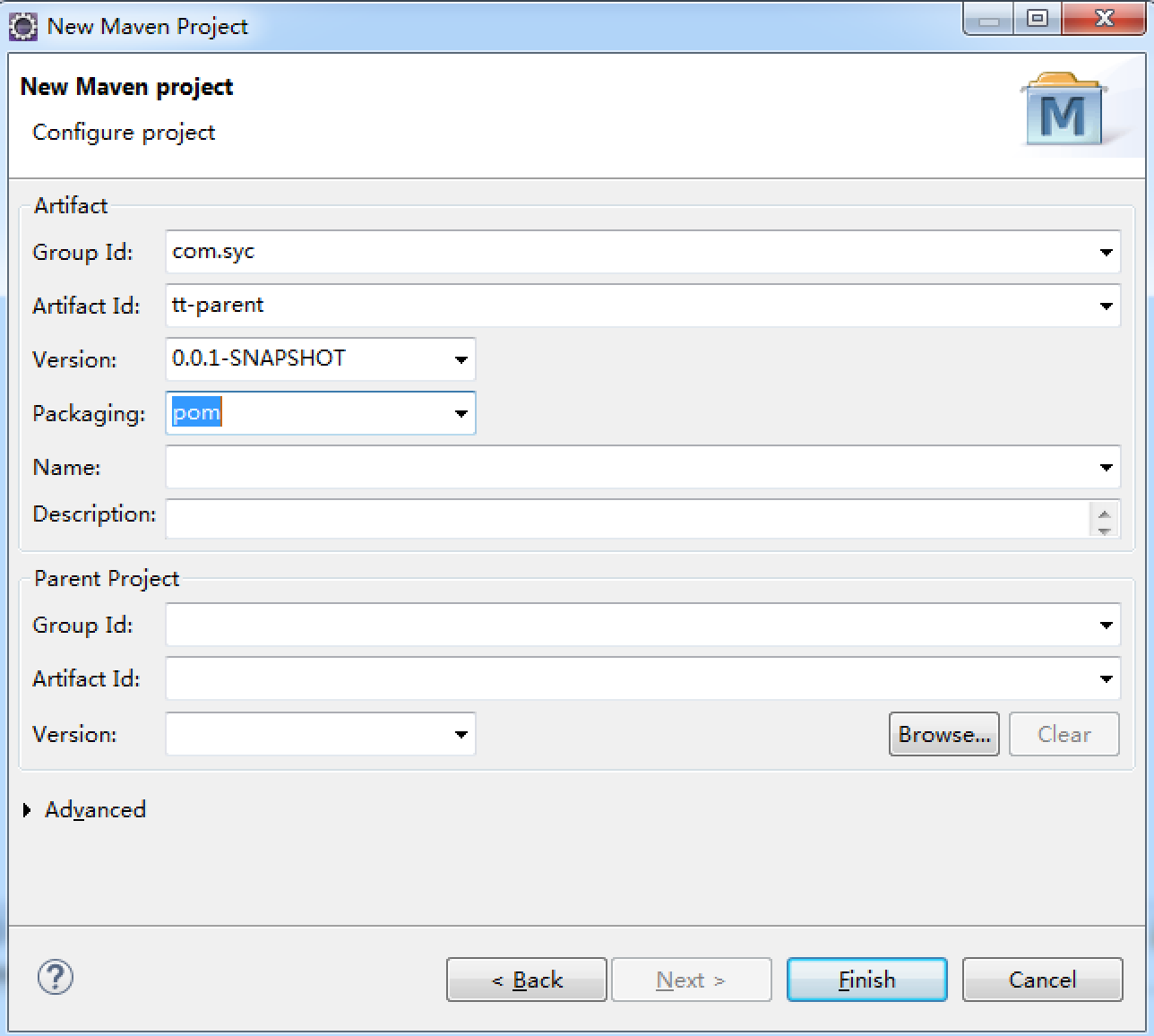
|--com.tt.manage.service

|--com.tt.manage.mapper

|--com.tt.manage.pojo

## 3.5创建tt-parent

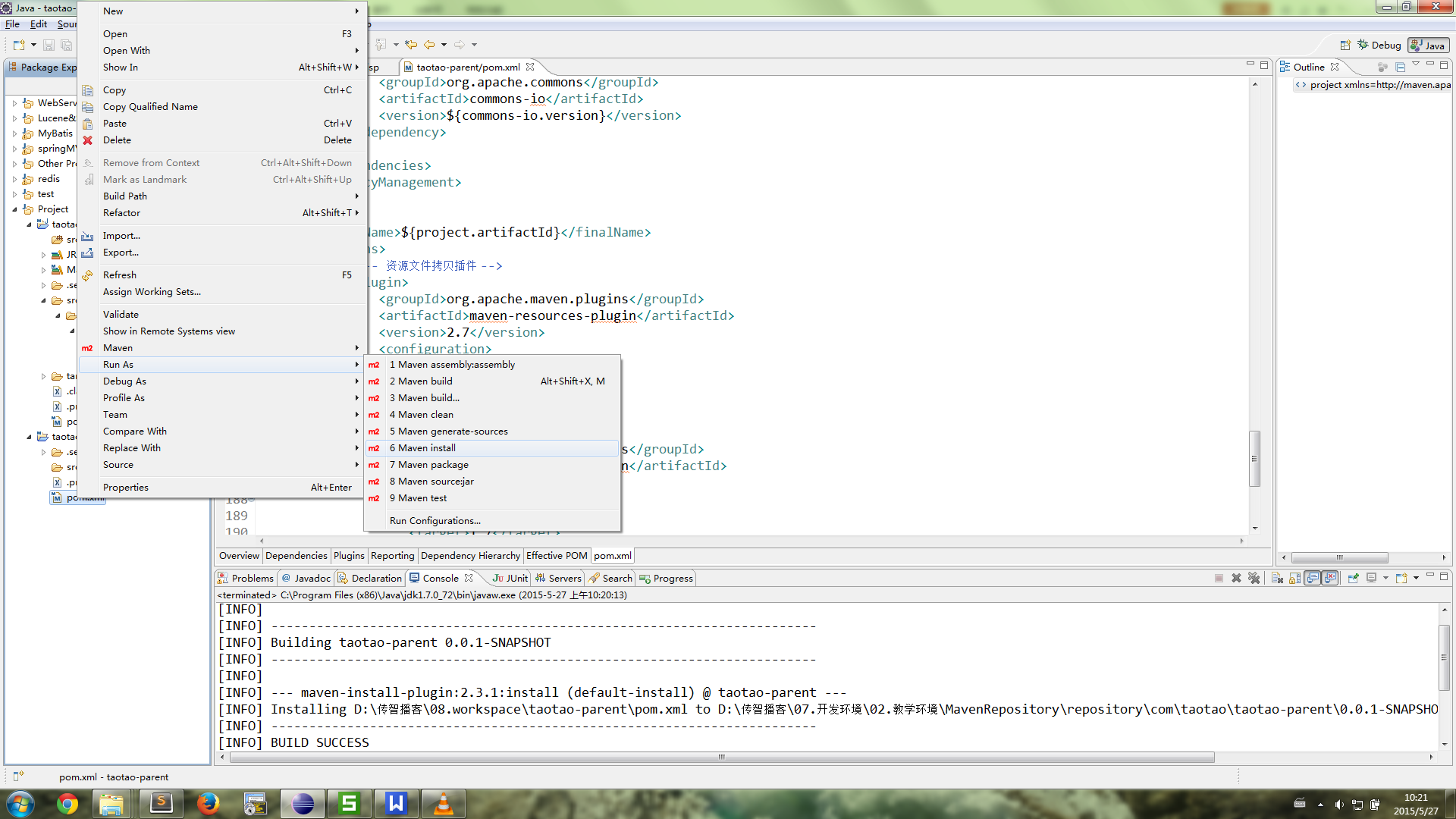
### 3.5.1创建maven工程



### 3.5.2修改pom文件

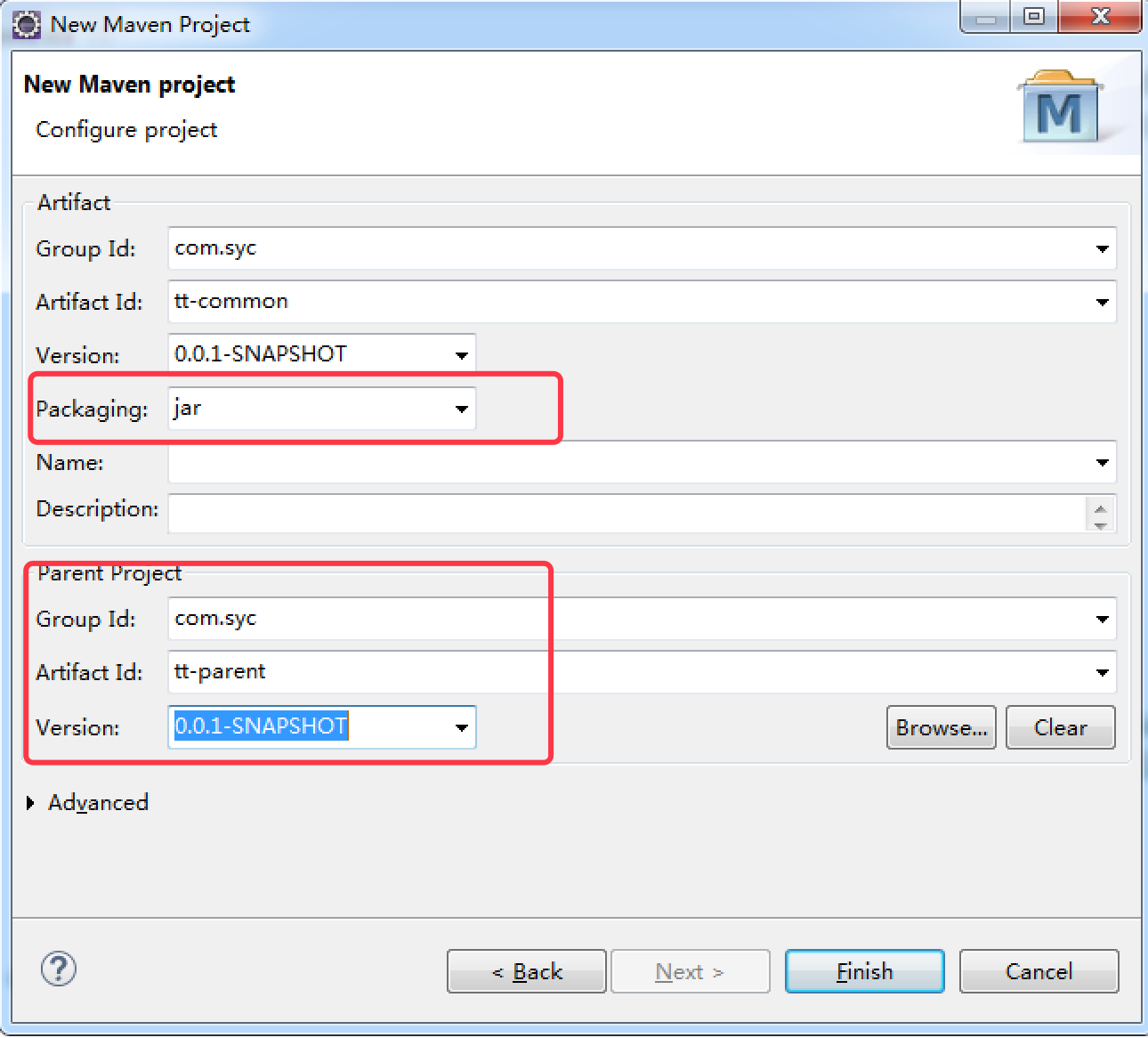
|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.syc</groupId>  <artifactId>tt-parent</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>4.2.1.RELEASE</spring.version>  <mybatis.version>3.2.7</mybatis.version>  <mybatis.spring.version>1.3.0</mybatis.spring.version>  <!-- 分页插件 -->  <mybatis.paginator.version>1.2.15</mybatis.paginator.version>  <!-- 用于分页的插件版本 -->  <pagehelper.version>3.4.2</pagehelper.version>  <mysql.version>5.0.7</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>  <commons-net.version>3.3</commons-net.version>  <jsqlparser.version>0.9.1</jsqlparser.version>  <commons-fileupload.version>1.3.1</commons-fileupload.version>  <jedis.version>2.7.2</jedis.version>  <solrj.version>4.10.3</solrj.version>  </properties>  <dependencyManagement>  <dependencies>  <!-- 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>${commons-io.version}</version>  </dependency>  <dependency>  <groupId>commons-net</groupId>  <artifactId>commons-net</artifactId>  <version>${commons-net.version}</version>  </dependency>  <!-- Jackson Json处理工具包 -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>${jackson.version}</version>  </dependency>  <!-- httpclient -->  <dependency>  <groupId>org.apache.httpcomponents</groupId>  <artifactId>httpclient</artifactId>  <version>${httpclient.version}</version>  </dependency>  <!-- 单元测试 -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>${junit.version}</version>  <scope>test</scope>  </dependency>  <!-- 日志处理 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>${slf4j.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>  <dependency>  <groupId>com.github.miemiedev</groupId>  <artifactId>mybatis-paginator</artifactId>  <version>${mybatis.paginator.version}</version>  </dependency>  <dependency>  <groupId>com.github.pagehelper</groupId>  <artifactId>pagehelper</artifactId>  <version>${pagehelper.version}</version>  </dependency>  <!-- MySql -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>${mysql.version}</version>  </dependency>  <!-- 连接池 -->  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>${druid.version}</version>  </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>${spring.version}</version>  </dependency>  <!-- JSP相关 -->  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>${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>  <!-- 文件上传组件 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  <version>${commons-fileupload.version}</version>  </dependency>  <!-- Redis客户端 -->  <dependency>  <groupId>redis.clients</groupId>  <artifactId>jedis</artifactId>  <version>${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>  </build>  </project> |

### 3.5.3将tt-parent安装到本地仓库。



## 3.6创建tt-common

### 3.6.1创建工程



### 3.6.2修改pom文件

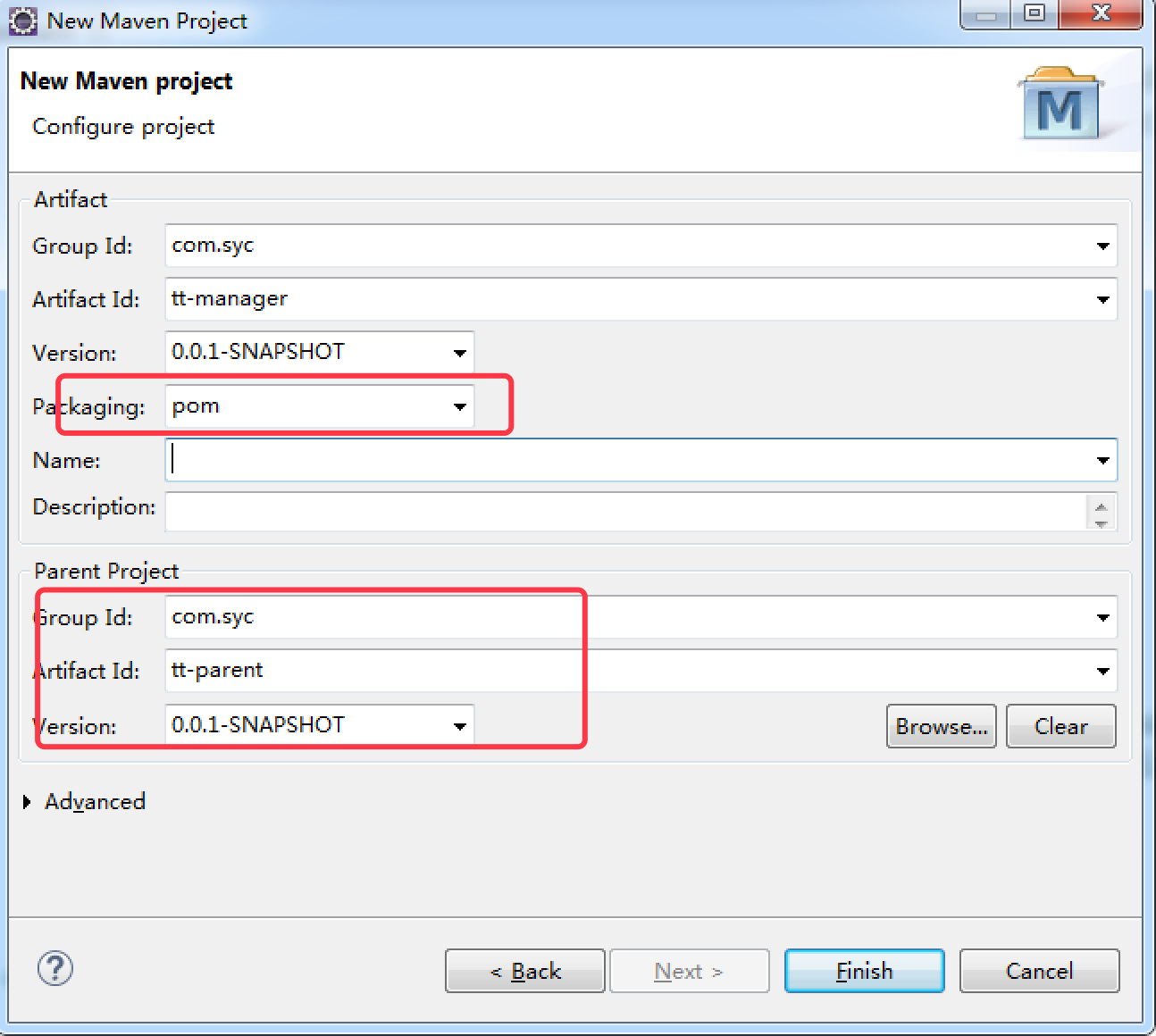
修改tt-common工程的pom文件，在文件中添加对tt-parent的继承。

|  |
| --- |
| <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.syc</groupId>**  **<artifactId>tt-parent</artifactId>**  **<version>0.0.1-SNAPSHOT</version>**  **</parent>**  <groupId>com.syc</groupId>  <artifactId>tt-common</artifactId>  <version>0.0.1-SNAPSHOT</version>  </project> |

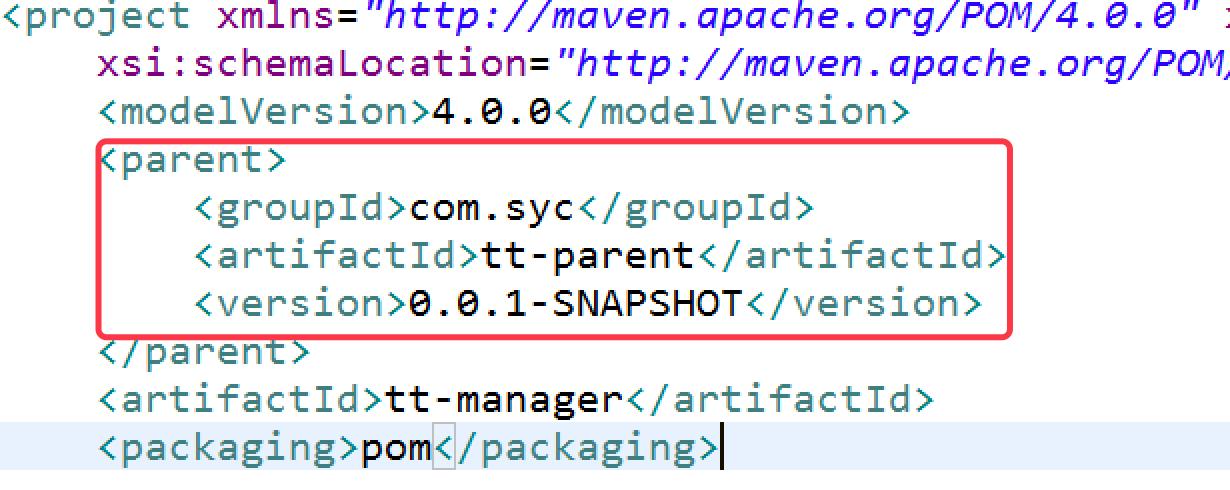
### 3.6.3更新工程

工程点击右键→maven→update Project Configuration

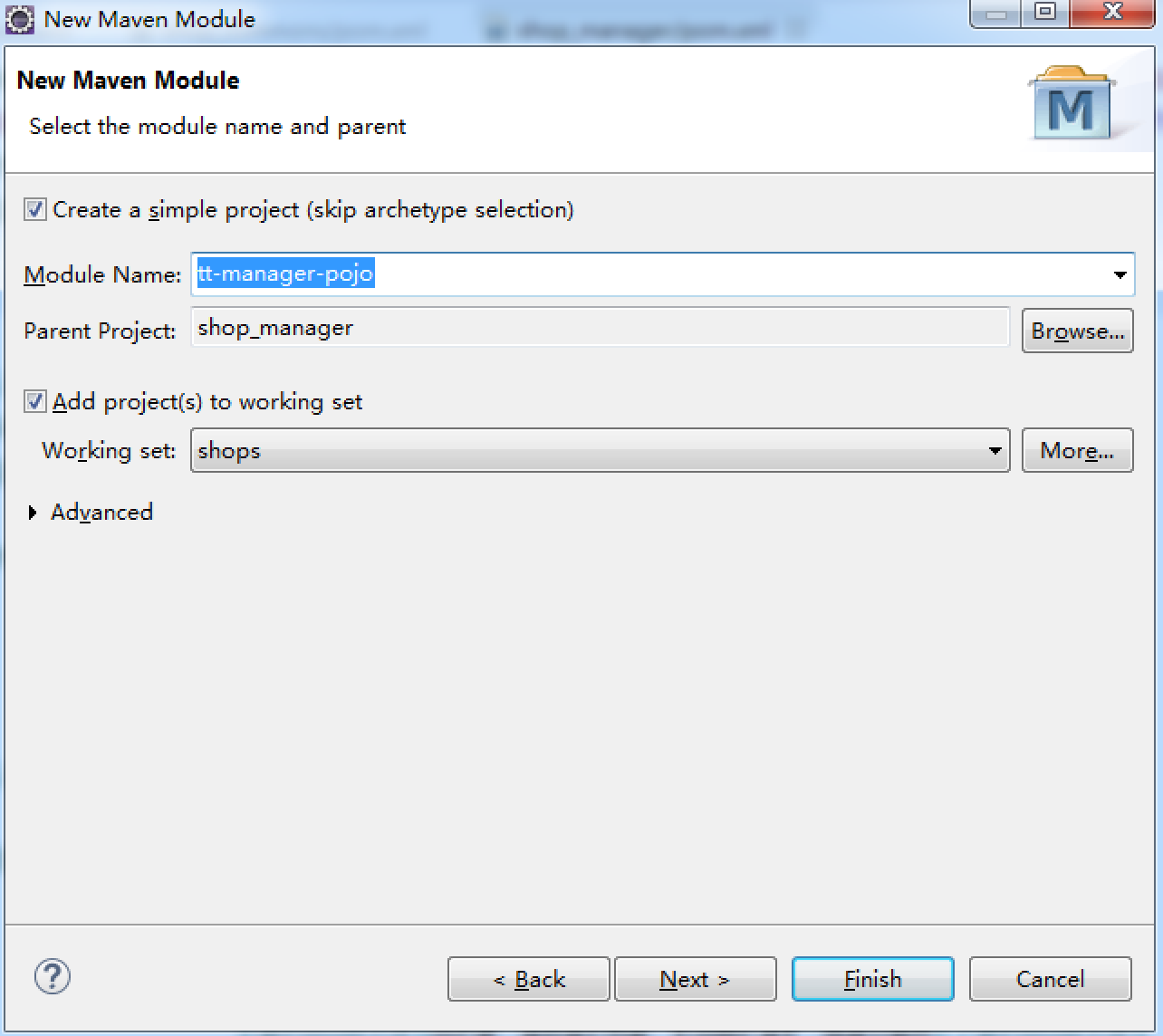
## 3.7创建tt-manage

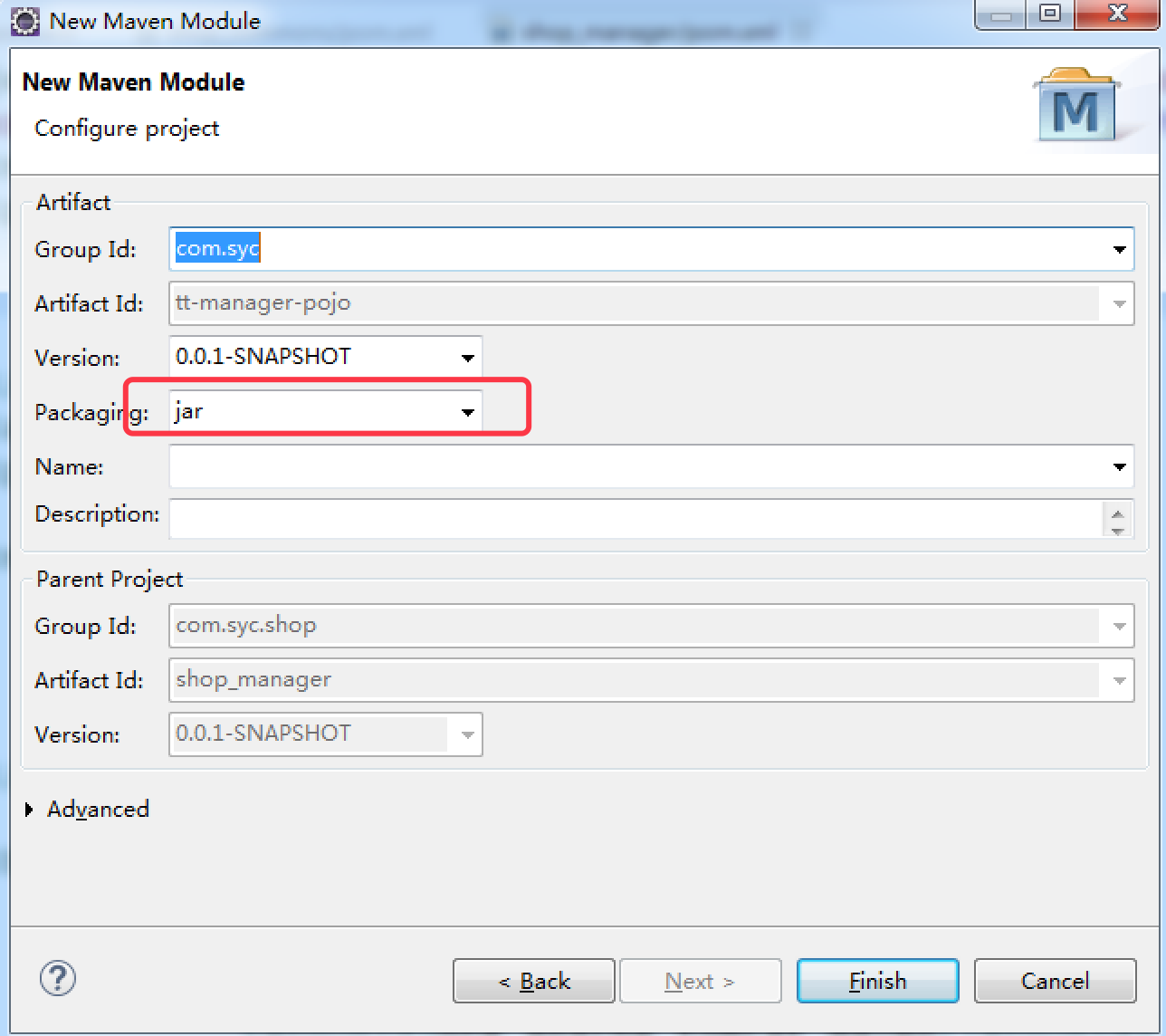


### 修改pom文件：

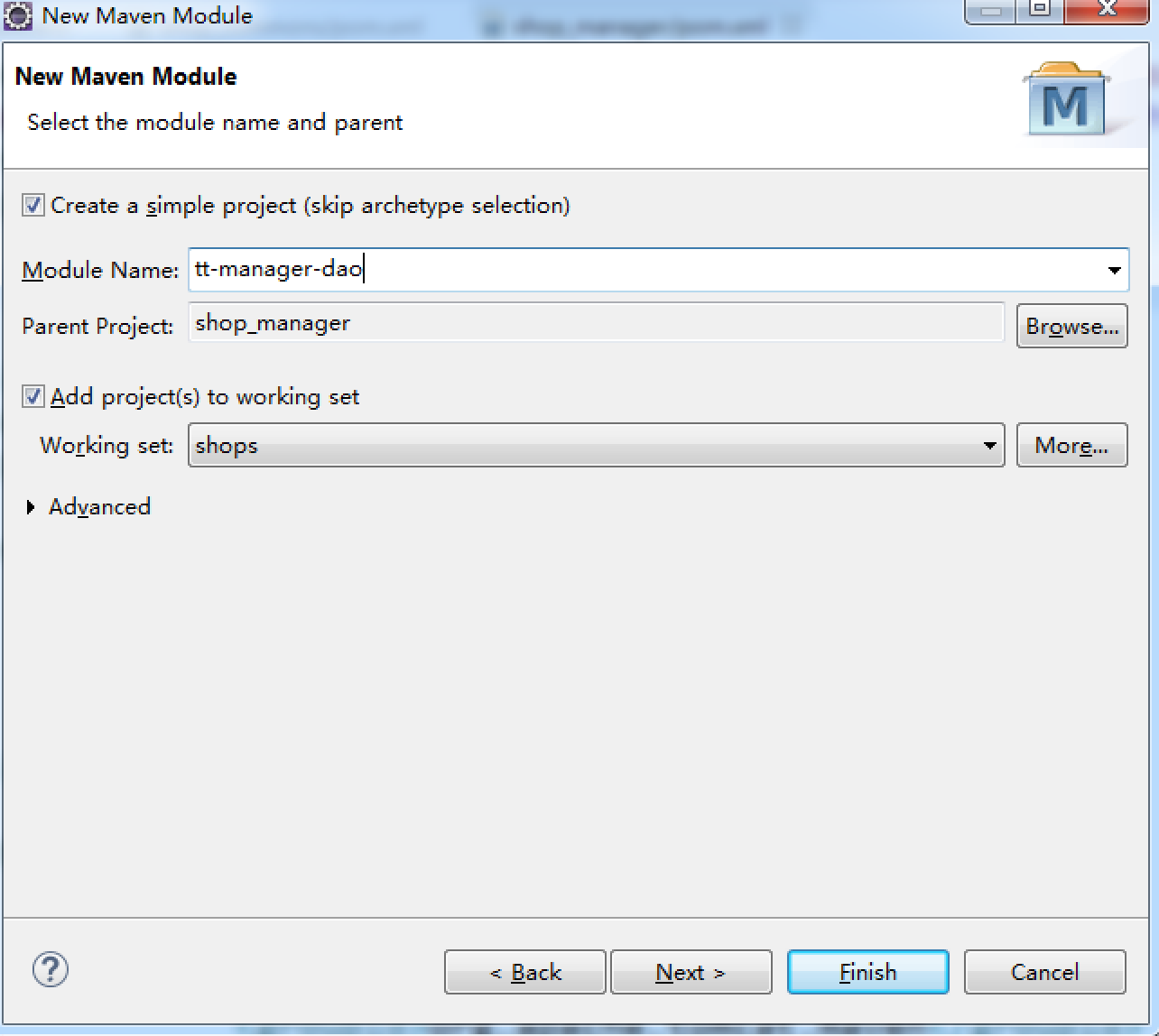


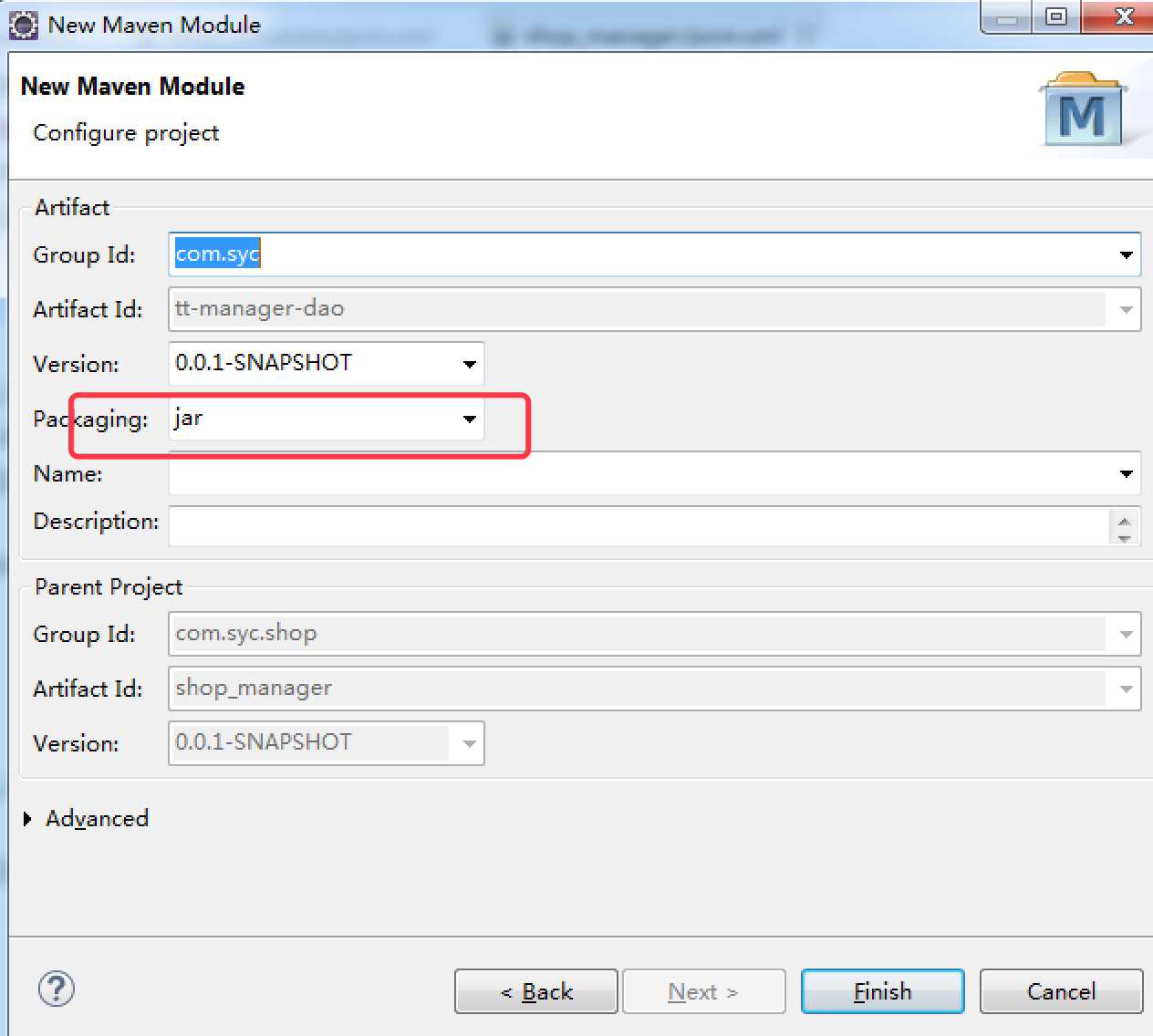
## 3.8 创建tt-manage-pojo模块module



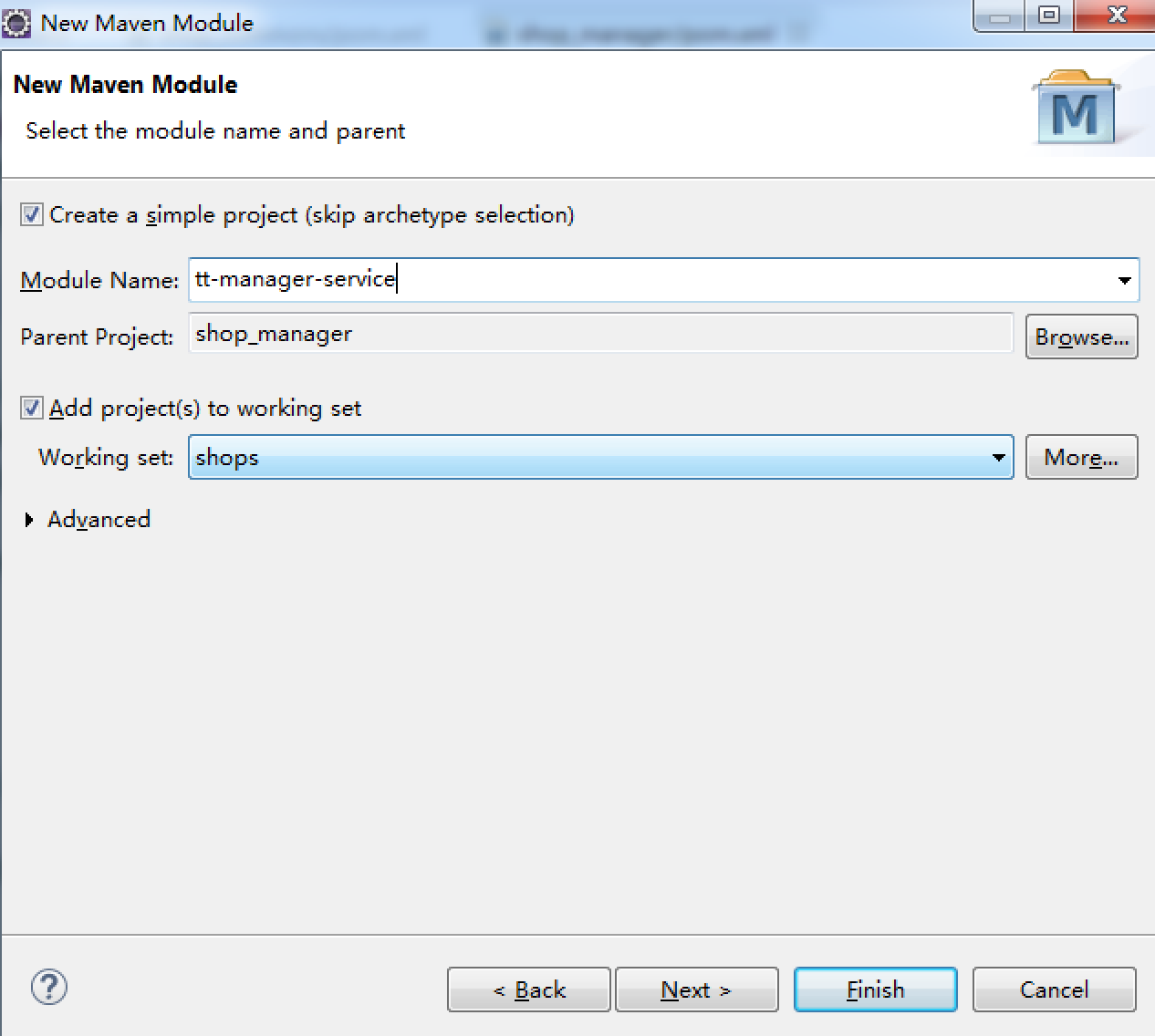


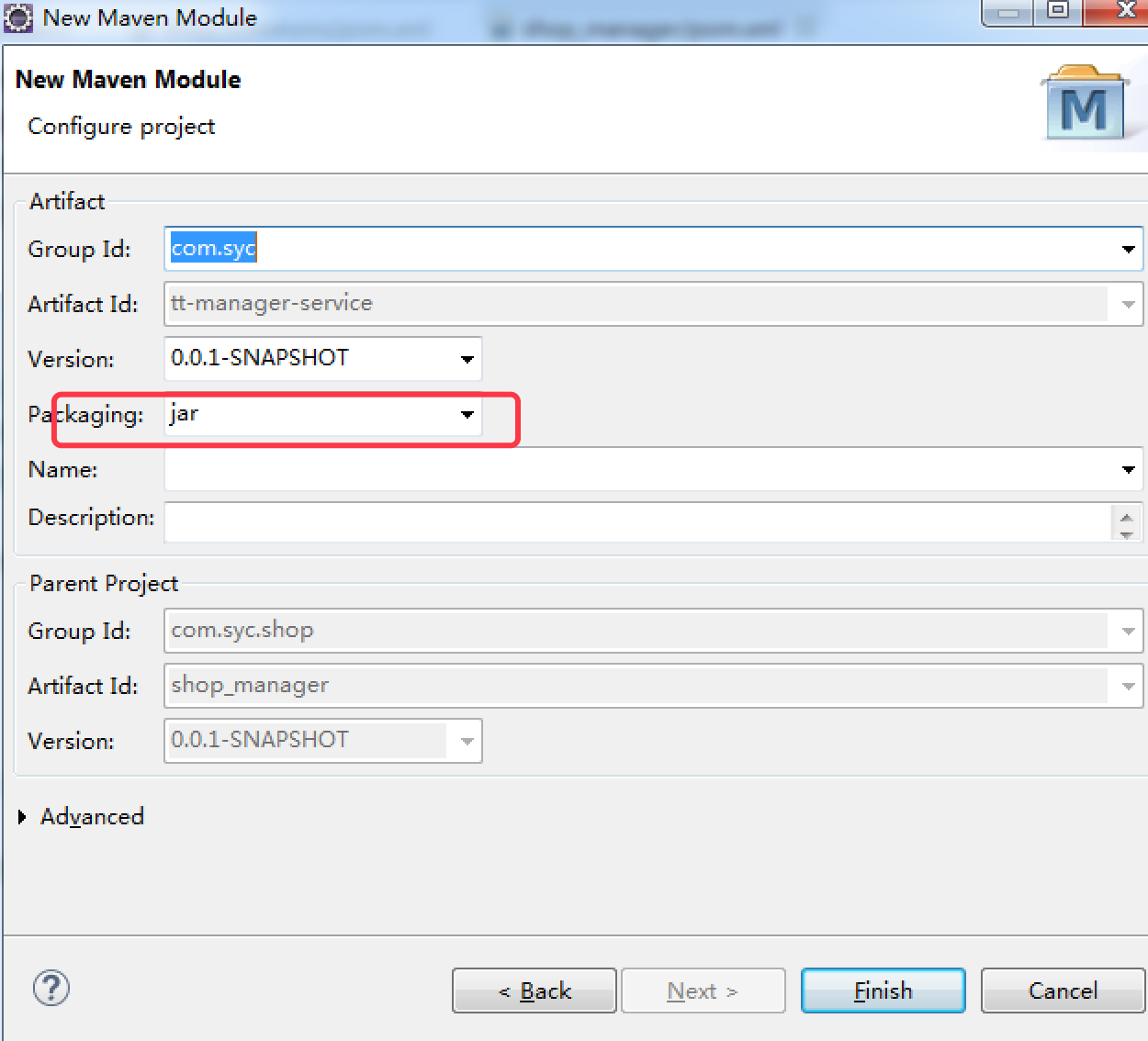
## 创建tt-manager-dao模块



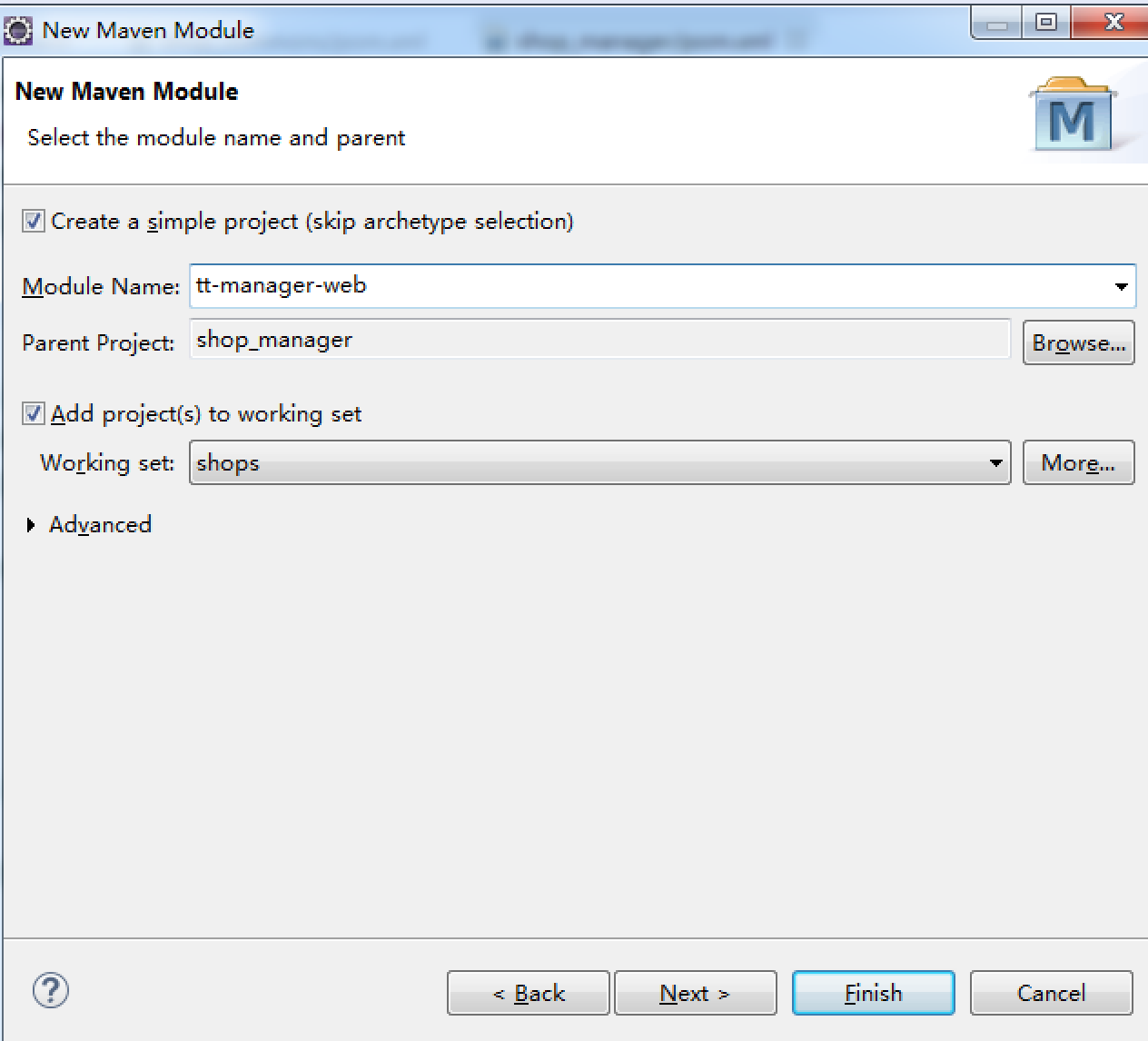


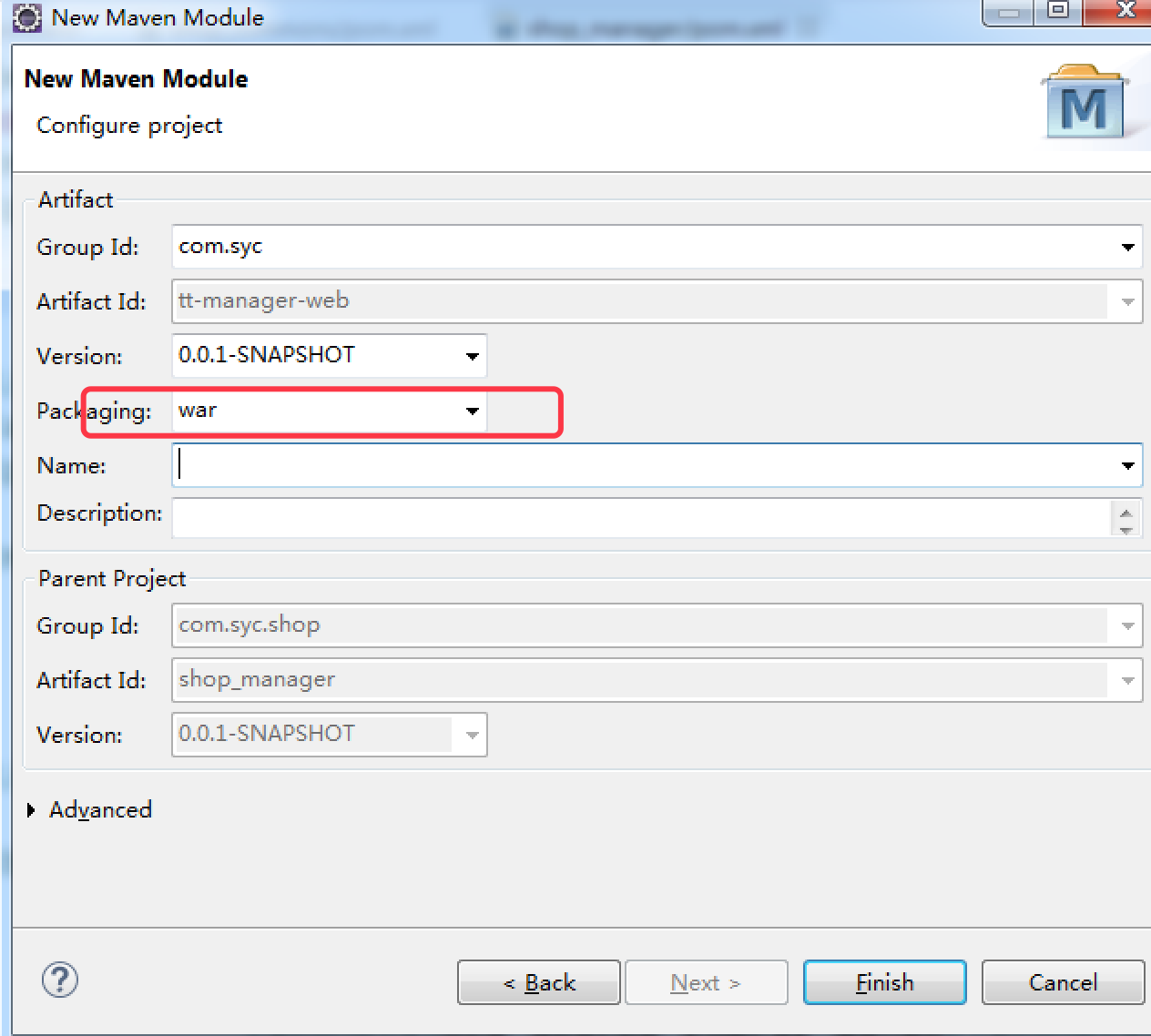
## 创建tt-manager-service模块



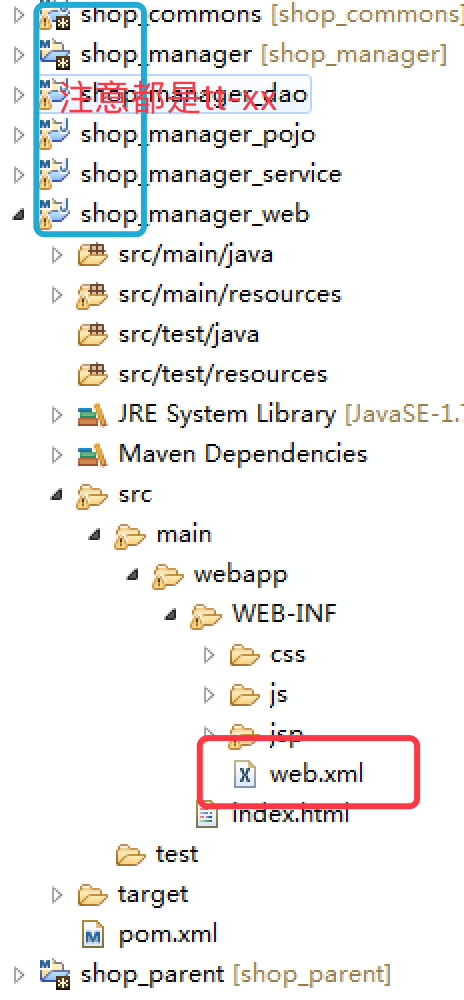


## 3.11 创建tt-manager-web模块





### 3.11.1配置工程



### 生成web.xml

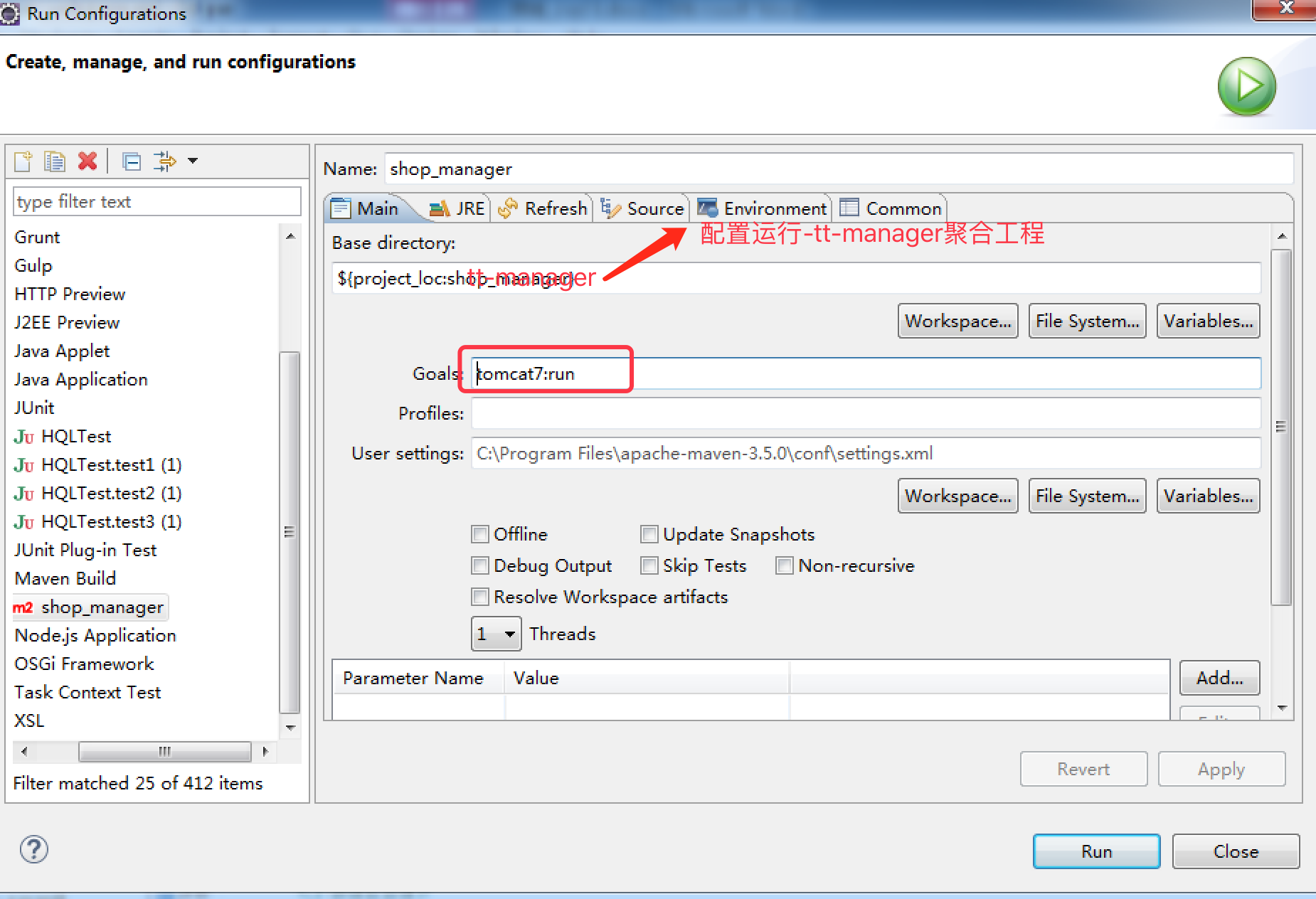
|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*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

## 3.12 在tt-manager工程中配置tomcat插件

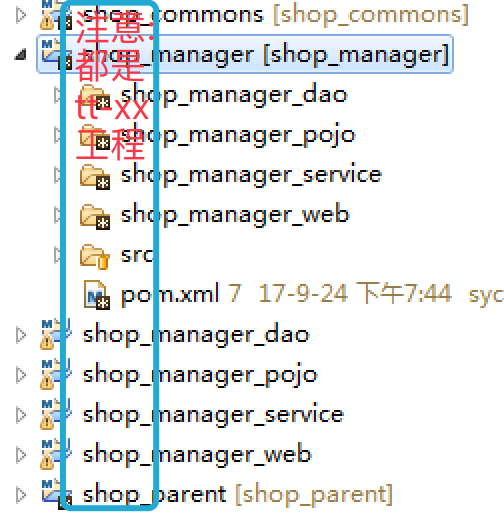
运行web工程需要添加一个tomcat插件。插件必须添加到tt-manager工程中,因为tt-manager是聚合工程,在运行时需要把子工程聚合到一起才能运行。

|  |
| --- |
| <build>  <plugins>  <!-- 配置Tomcat插件 -->  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <version>2.2</version>  <configuration>  <port>8080</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build> |

启动tomcat命令：tomcat7:run



## 3.13 tt-manager子模块依赖关系



### 依赖关系

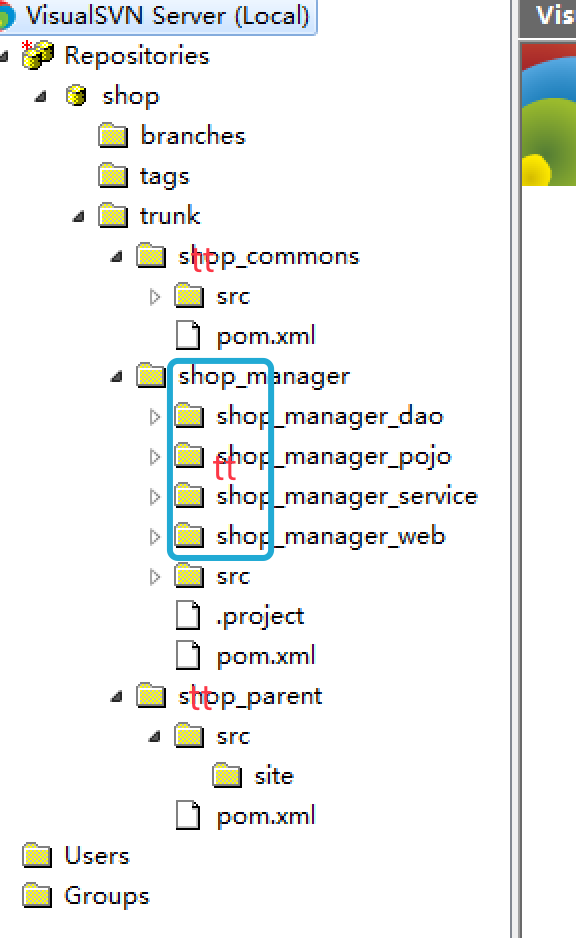
web 🡺 service

service 🡺 mapper

mapper 🡺 pojo

# 四.提交代码到SVN

## 4.1提交代码

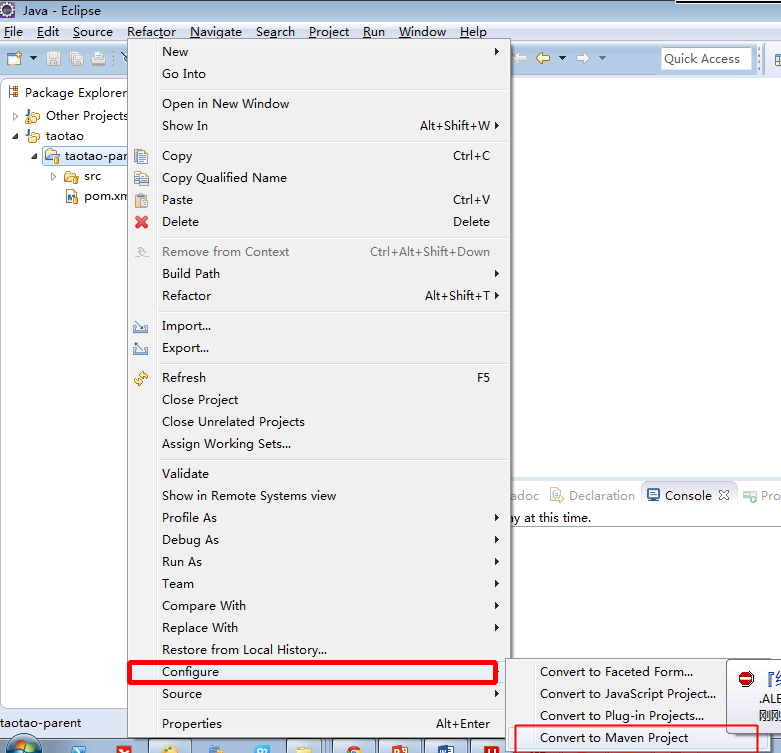


注意：提交到SVN的Maven项目，只提交src和pom.xml

## 4.2从SVN检出项目

### 1.从trunk检出项目，并且重命名项目名称

### 2.转化为maven项目



### 3.聚合项目中子项目需要从父工程中【导入】

选择 【已经存在的maven项目】，不能从SVN再次检出子项目

