Maven项目管理工具

白面郎君

Svn eclipse maven量级

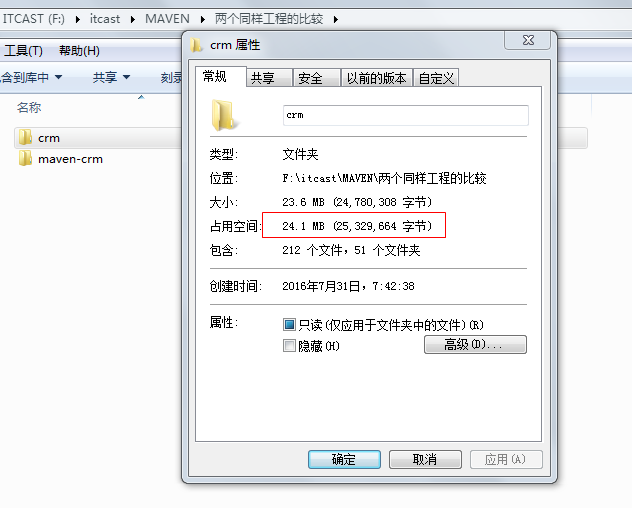
# Maven的简介

## 什么是maven

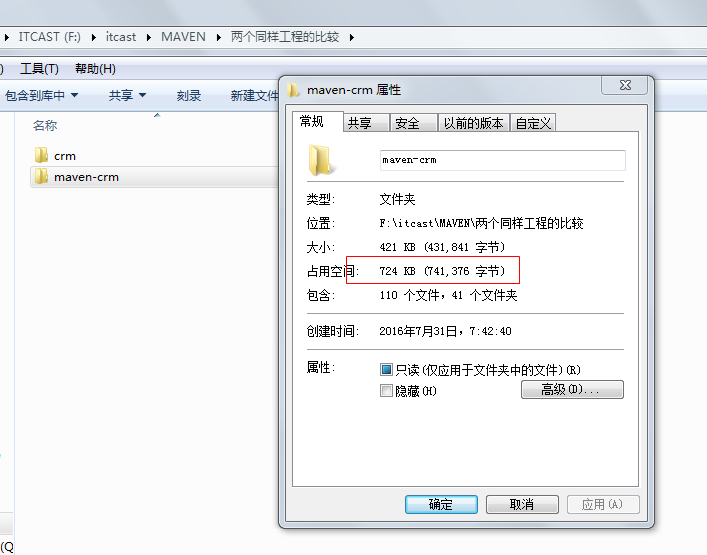
是apache下的一个开源项目，是纯java开发，并且只是用来管理java项目的

## Maven好处

普通的传统项目

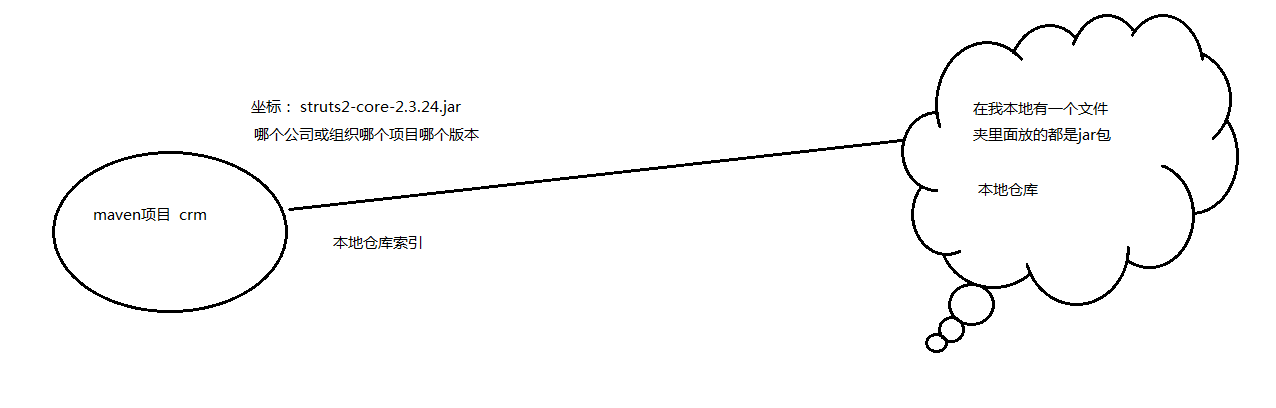


Maven项目



分析：maven项目为什么这么小？没有jar。 需要jar吗？肯定需要。没有存在于maven项目里面，jar存在于哪？

## 依赖管理



## 项目一键构建

编码 编译 测试(junit) 运行 打包 部署

一个 tomcat:run就能把项目运行起来

Maven能干的事：

编译 测试(junit) 运行 打包 部署

## 小结：

Maven的好处：

### 依赖管理

就是对jar包的统一管理 可以节省空间

### 一键构建

### 可以跨平台

### 应用于大型项目 可以提高开发效率

分析：

用户管理 订单管理 支付管理 。。。。。

Maven的分模块开发

互联网项目 按业务分

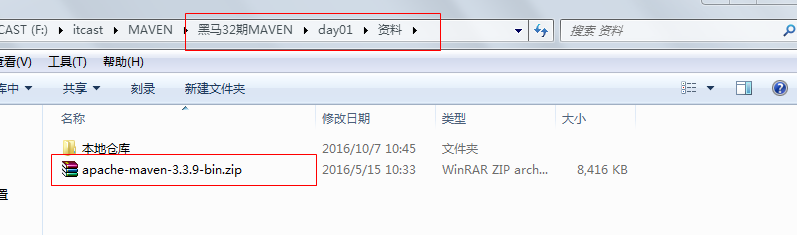
传统项目 按层分 entity dao service web

# Maven的安装配置

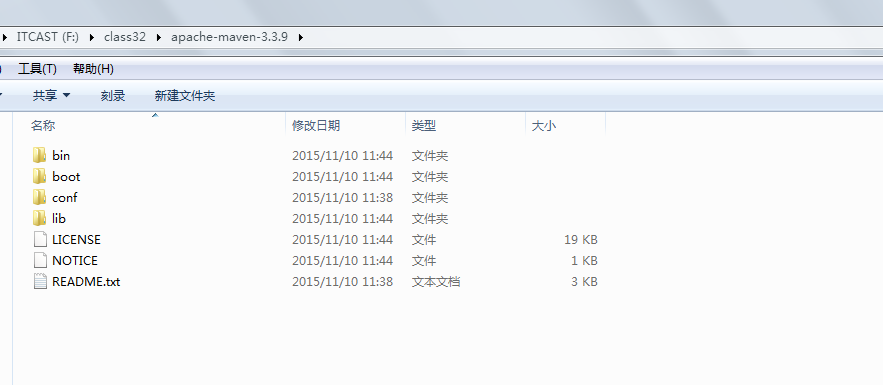
## 下载安装



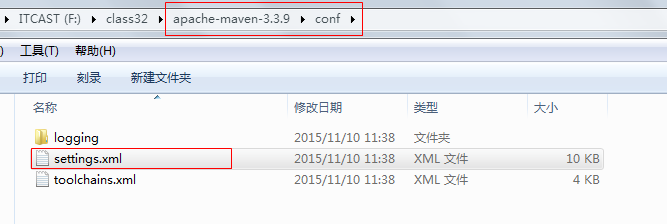
从此目录中拷贝文件：



直接解压到一个没有中文没有空格的文件夹里解压后的目录如下：



Maven软件的核心文件：



## Maven环境变量配置

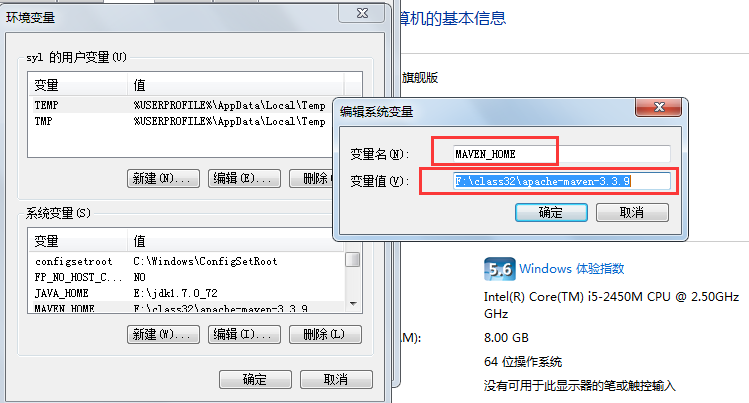
1. 要配置jdk， maven3.3.9这个版本所需的jdk版本必须要1.7以上
2. 最终要运行的是maven软件中bin目录的mvn命令

所以要配置maven的环境变量

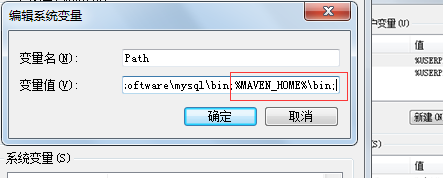
在系统变量添加

环境变量的名称：MAVEN\_HOME

变量值：就是maven软甲解压的目录F:\class32\apache-maven-3.3.9

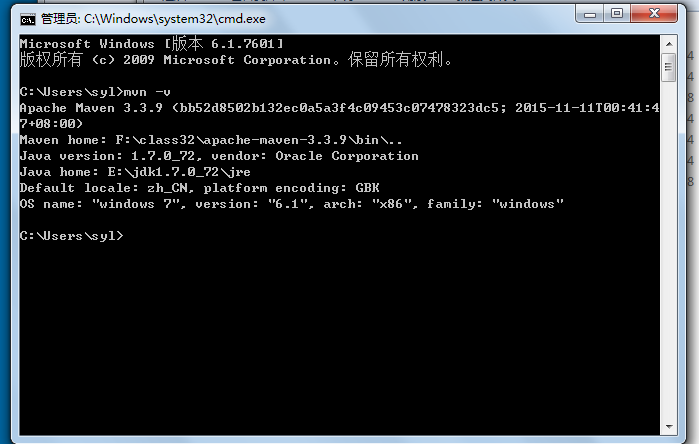


3、把MAVEN\_HOME添加到path里



4、验证maven是否配置成功：

打开dos窗口 输入： mvn –v

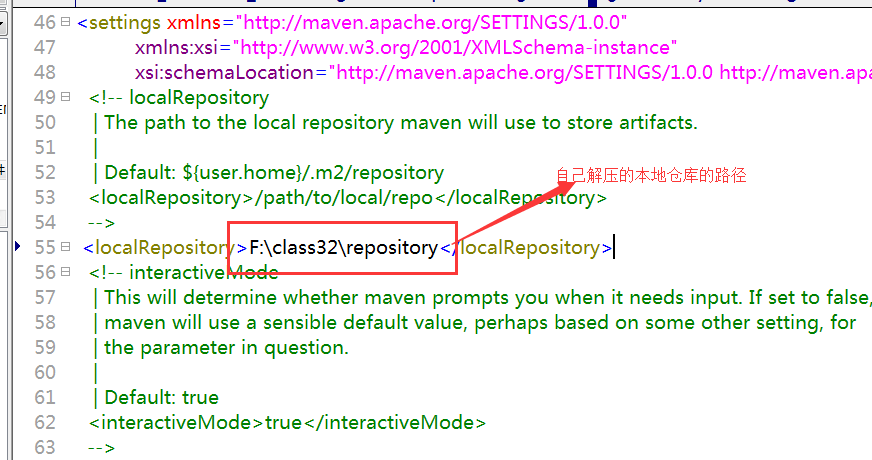


## Maven仓库

三种仓库

1、本地仓库 自己维护

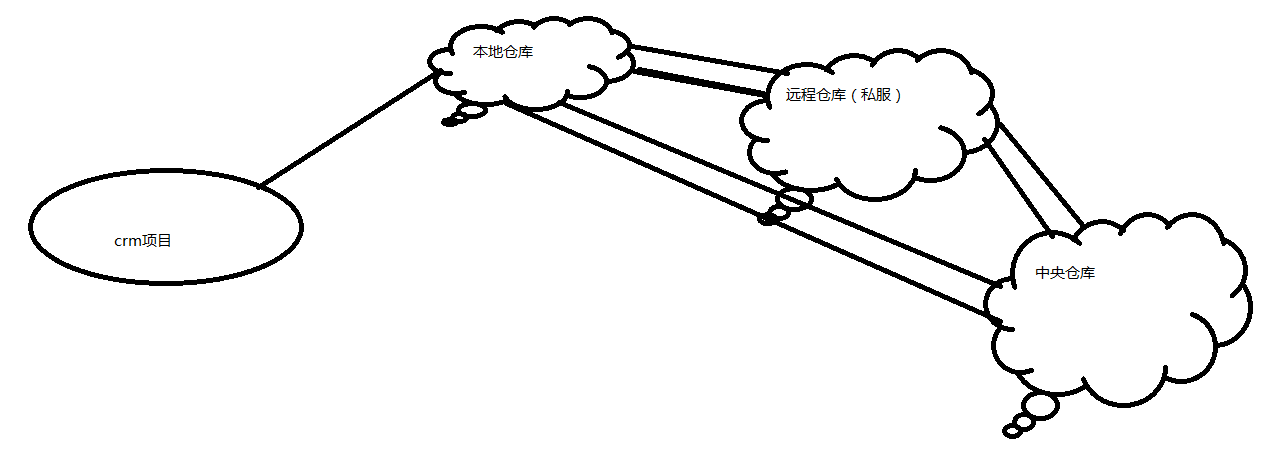
本地仓库的配置只需要修改settings.xml文件就可以



2、远程仓库（私服） 公司维护

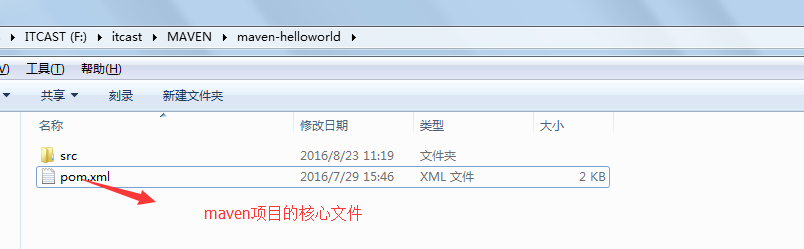
3、中央仓库 maven团队维护 两个亿

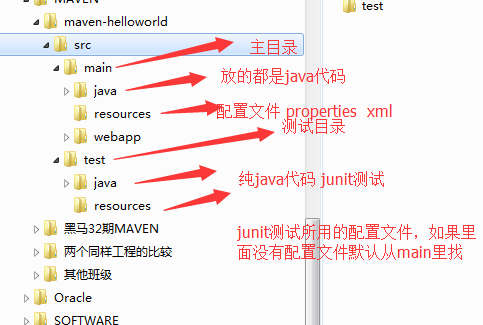
三种仓库的关系如下：



# 演示入门程序

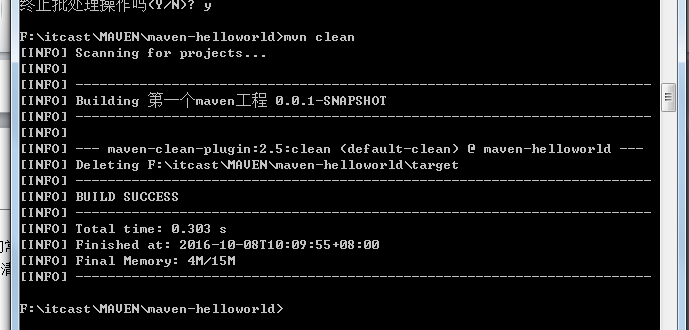
## Maven的目录结构



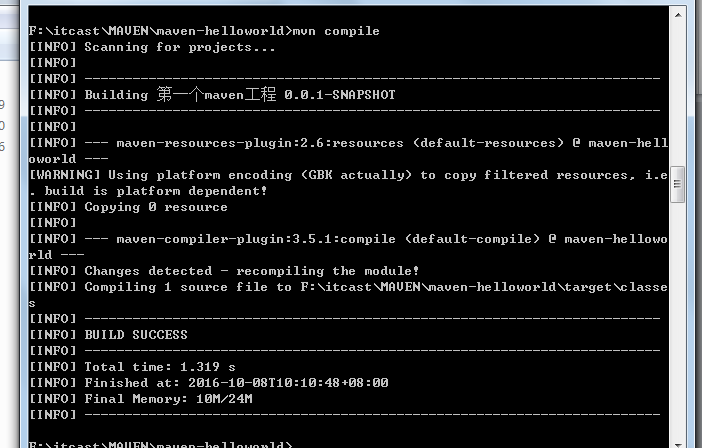


## Maven的常用命令

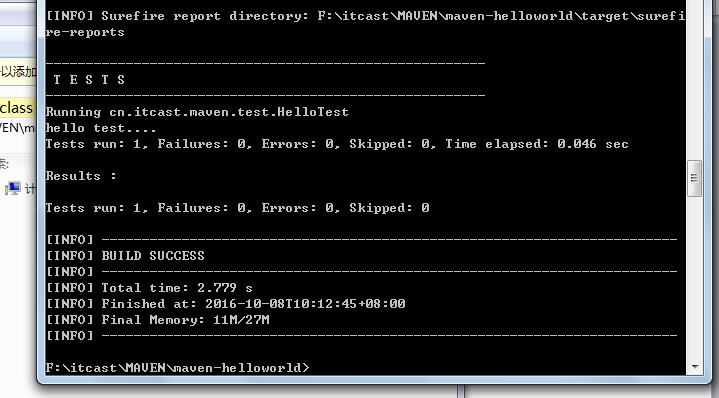
Clean 清理编译的文件



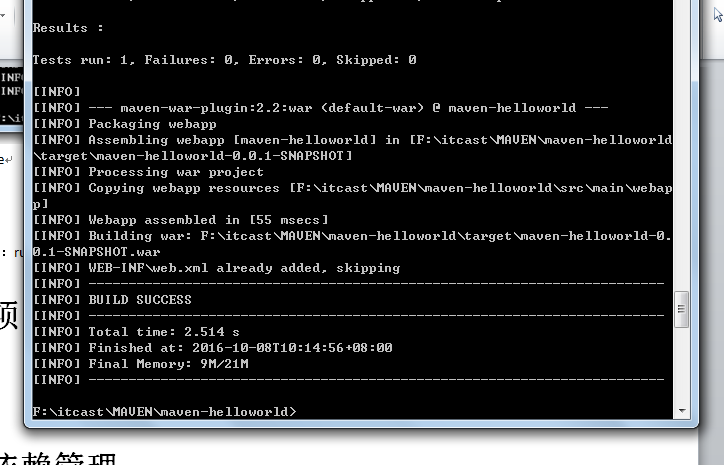
Compile 编译了主目录的文件



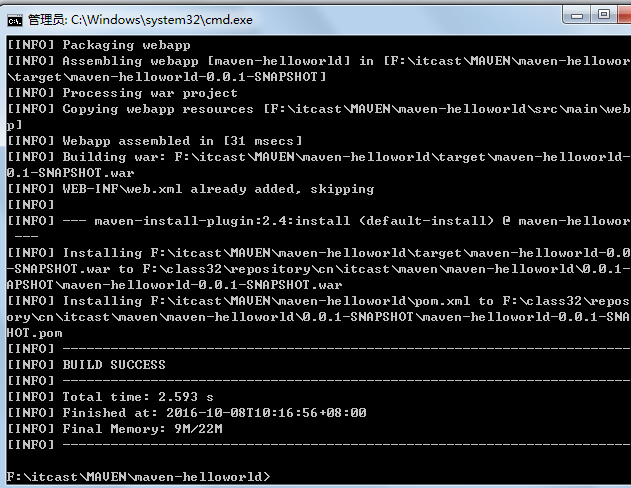
Test 编译并运行了test目录的代码



Package 打包



Install 就是把项目发布到本地仓库



Tomcat：run 一键启动

## Maven的生命周期（了解）

Compile test package install deploy（发布到私服）

三种生命周期

Clean生命周期

Clean

Default生命周期

Compile test package install deploy

Site生命周期

Site

## 命令和生命周期的阶段的关系

不同的生命周期的命令可以同时执行

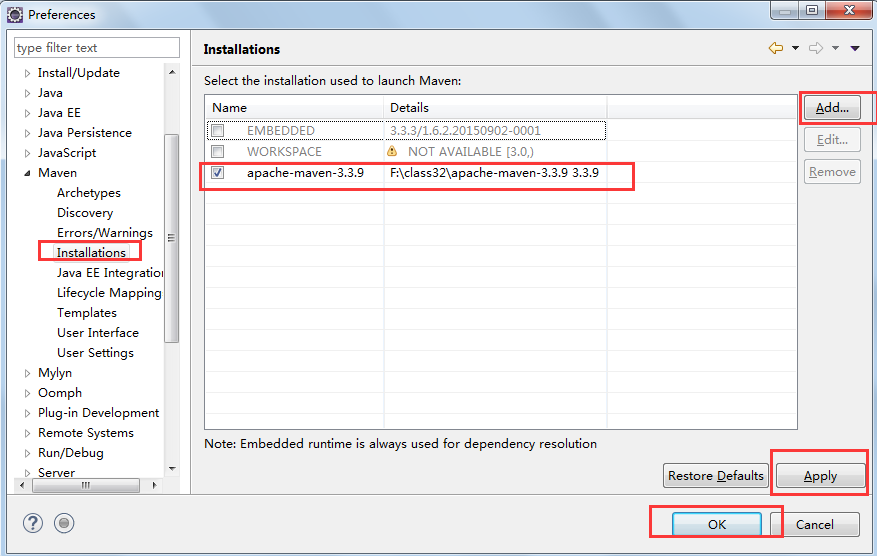
Mvn clean package

# 项目构建

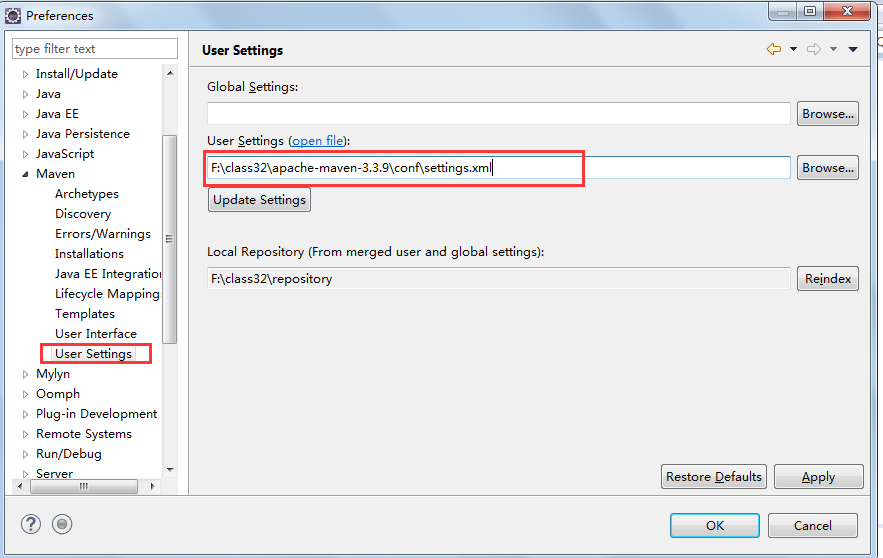
M2e插件的安装

Maven在eclipse的配置

1、选择3.3.9版本的maven软件

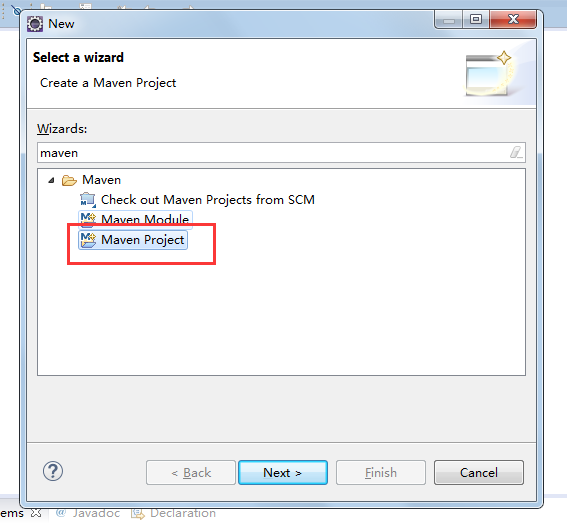


2、修改默认的本地仓库地址

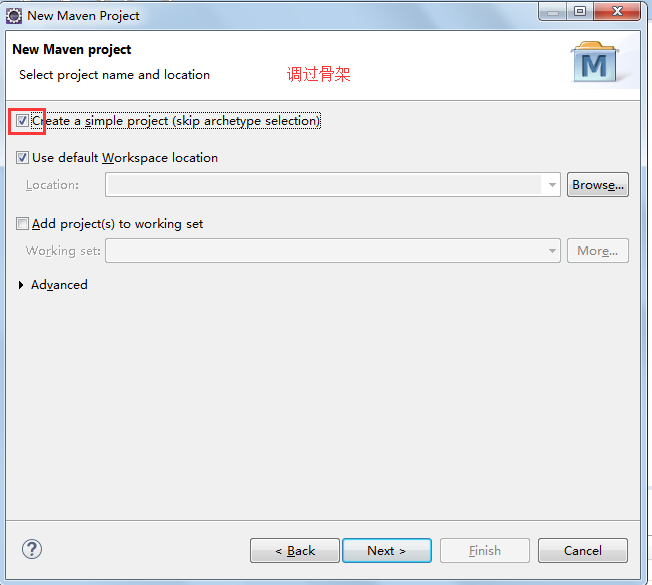


项目构建

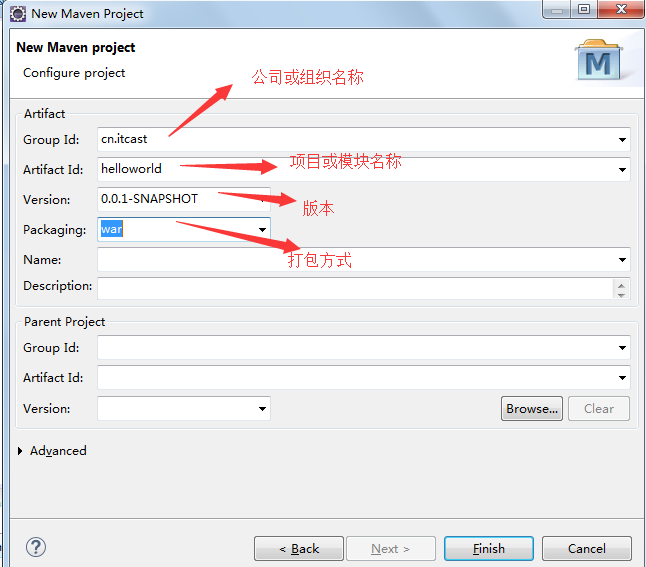
1. 新建maven项目



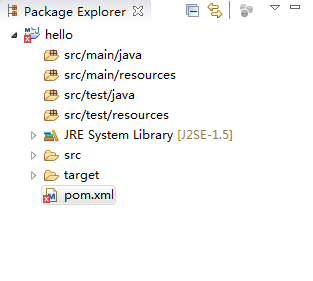
2、调过骨架，如果不跳过骨架选择创建出的项目目录是不全的



1. 填写坐标

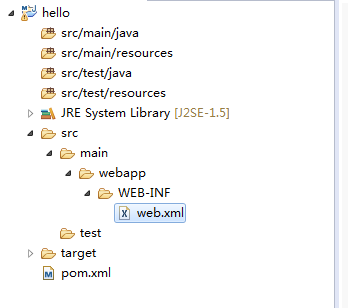


3、创建工程



4、处理红色叉号

**手动**在webapp文件夹下创建一个WEB-INF文件夹，在里面放一个web.xml文件



5、处理编译版本

在pom.xml中添加如下代码

<build>

<!-- 配置了很多插件 -->

<plugins>

<plugin>

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

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

<version>3.5.1</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

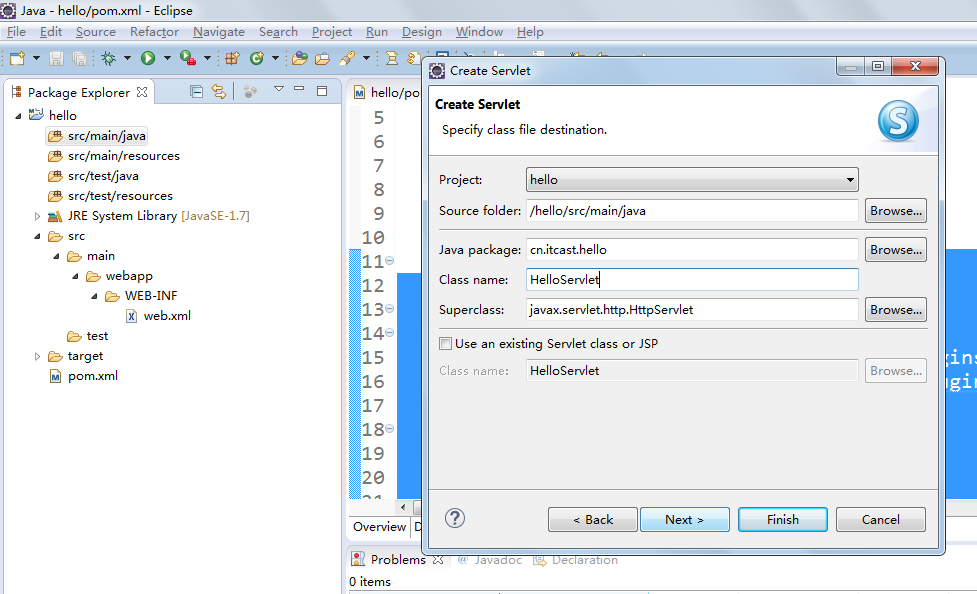
</configuration>

</plugin>

</plugins>

</build>

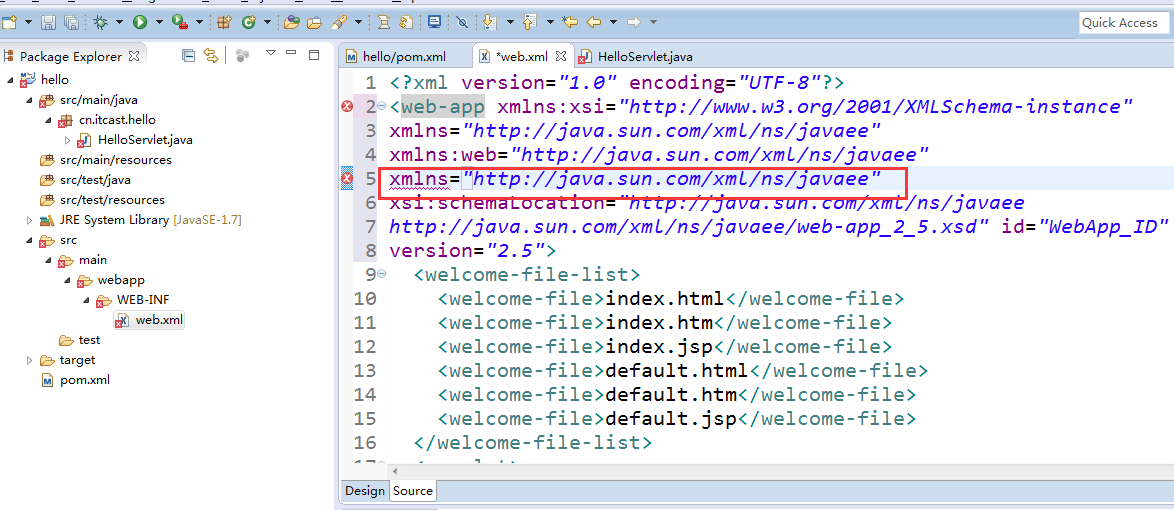
5、创建一个servlet



修改web.xml

删除重复的代码

xmlns=<http://java.sun.com/xml/ns/javaee>



添加jar包

在pom中添加如下代码：

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

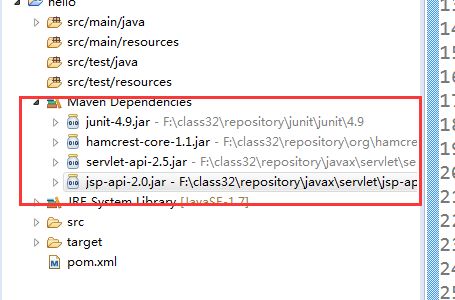
<version>2.0</version>

<scope>provided</scope>

</dependency>

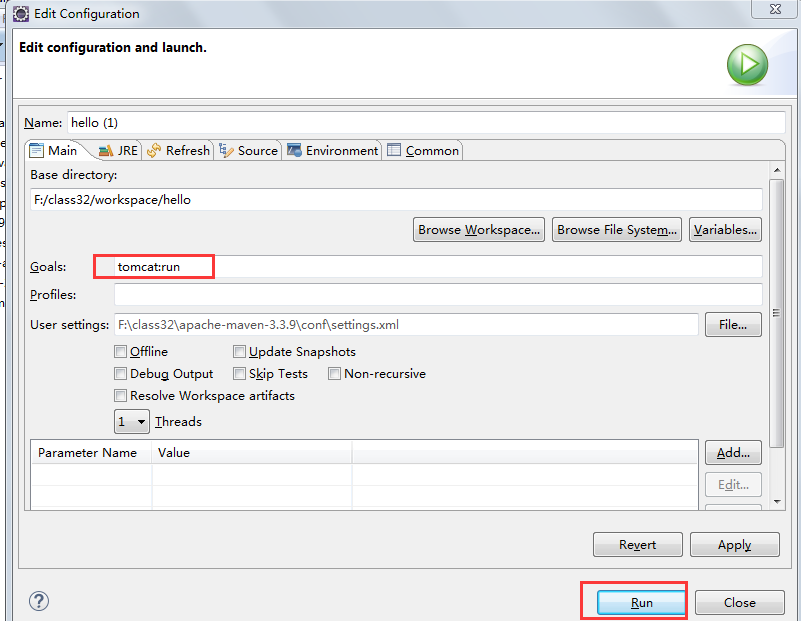
</dependencies>

会发现jar包已添加到项目中



启动项目

右击项目---run as –maven build ….



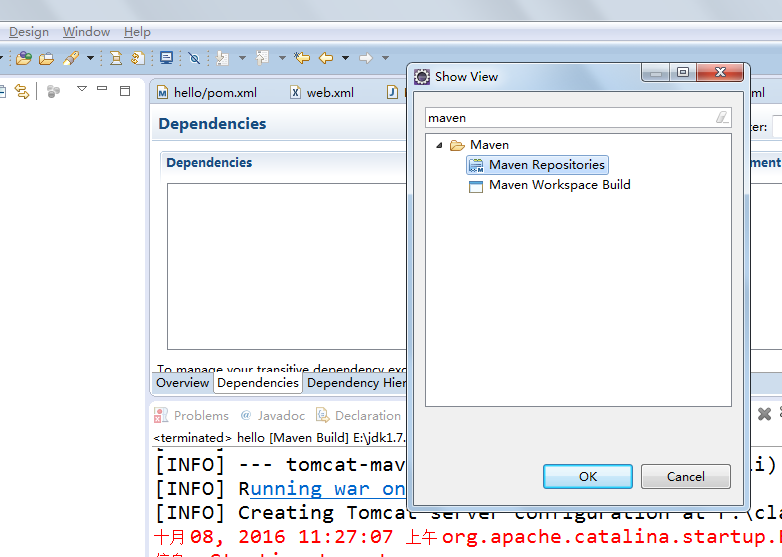
# 依赖管理

Jar包的管理

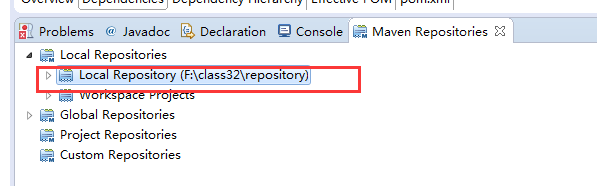
需求：整合struts2 页面上传一个客户id 跳转页面

## 添加依赖：

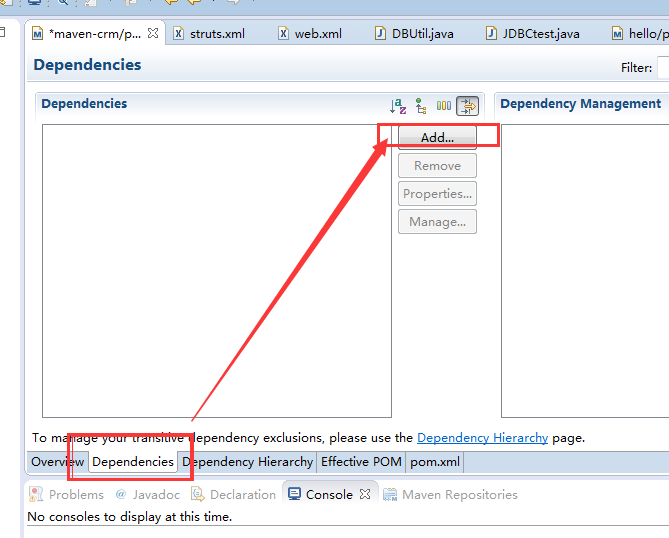
打开maven仓库的视图：



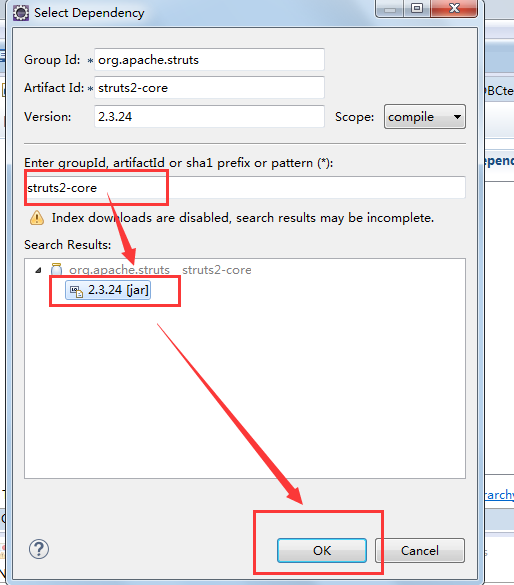
## 重建索引



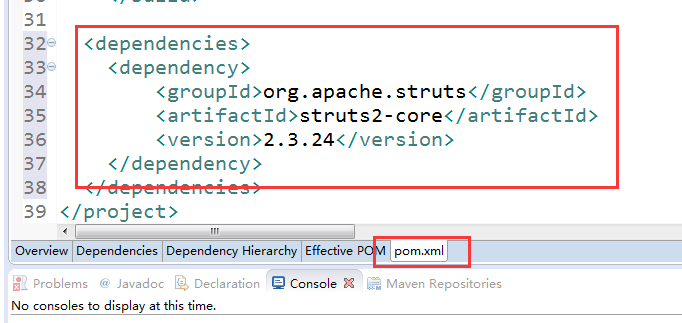
1. 创建maven项目（同上）
2. 跳过骨架（同上）
3. 填写坐标信息（同上）
4. 添加web.xml文件（同上）
5. 修改编译版本（同上）
6. 添加坐标 选择Dependencies标签 点击add



1. 手动输入要添加的坐标，选择版本



1. 可以看到 pom.xml文件中多出了如下代码



1. 同样的方式添加servlet-api.jar和jsp-api.jar **注意选择scope为provided**
2. 写action代码

**import** com.opensymphony.xwork2.ActionSupport;

**public** **class** CutomerAction **extends** ActionSupport {

**private** Long custId;

**public** Long getCustId() {

**return** custId;

}

**public** **void** setCustId(Long custId) {

**this**.custId = custId;

}

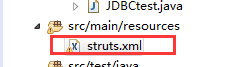
**public** String findById(){

**return** ***SUCCESS***;

}

}

1. 添加struts.xml文件放到resources目录中



内容：

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

<!DOCTYPE struts PUBLIC

"-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"

"http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>

<!-- 配置常量 -->

<!-- 字符集 -->

<constant name=*"struts.i18n.encoding"* value=*"UTF-8"*></constant>

<!-- 开发模式 -->

<constant name=*"struts.devMode"* value=*"true"*></constant>

<!-- 通用package -->

<package name=*"customer"* namespace=*"/"* extends=*"struts-default"*>

<action name=*"find"* class=*"cn.itcast.action.CutomerAction"*

method=*"findById"*>

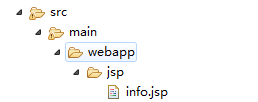
<result name=*"success"*>/jsp/info.jsp</result>

</action>

</package>

</struts>

1. 添加jsp页面



15、修改web.xml文件 添加过滤器

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

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

</filter-mapping>

## 依赖范围

### Compile struts2-core

编译（compile）时需要 测试时需要，，运行时需要，打包时需要

### Provided jsp-api.jar servlet-api.jar

编译（compile）时需要，测试（test）时也需要 ，运行时不需要，打包时不需要

### Runtime 数据库驱动包

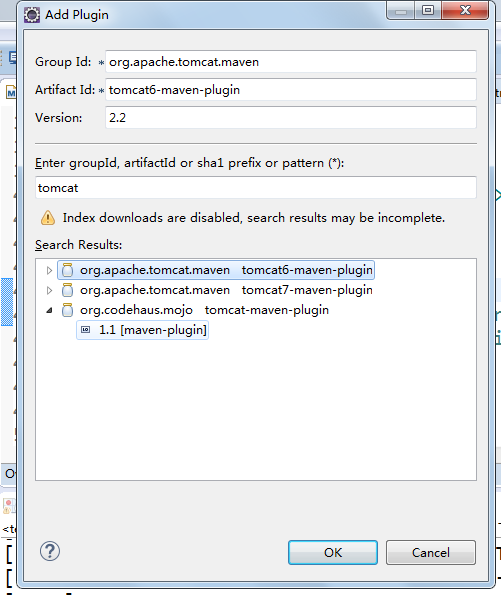
编译时不需要，测试时需要，，运行时需要，打包时需要

### Test junit.jar

编译时不需要，测试时需要，运行时不需要，打包也不需要

添加插件

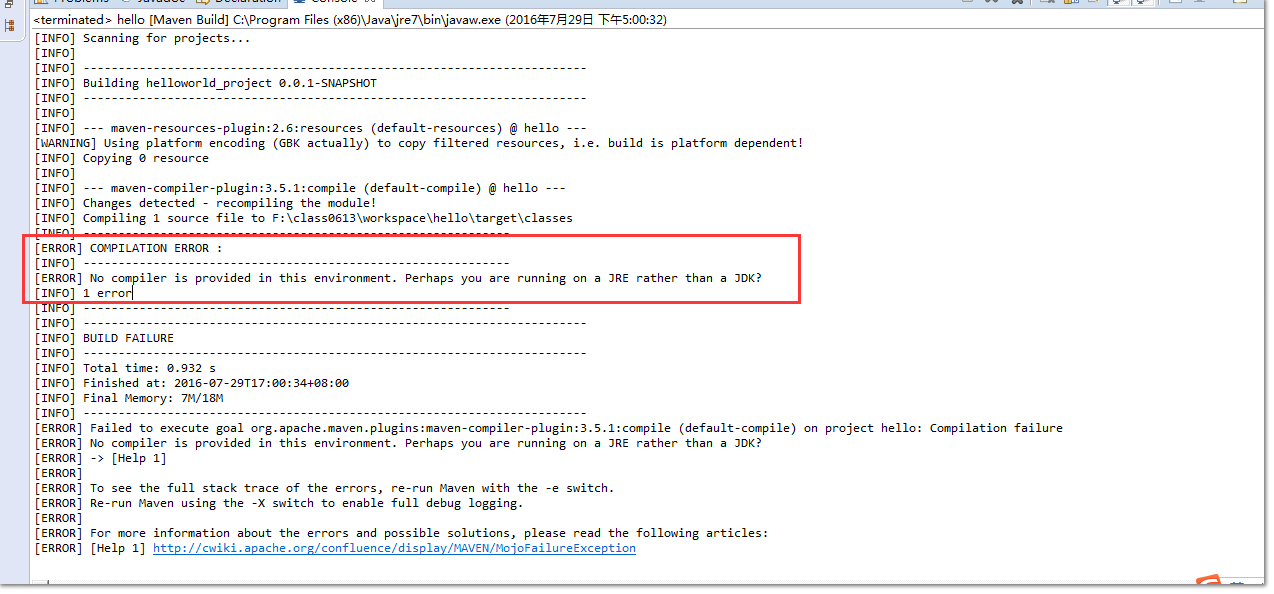
Maven add plugin



如果用tomcat7运行用命令：

Tomcat7:run

常见问题：

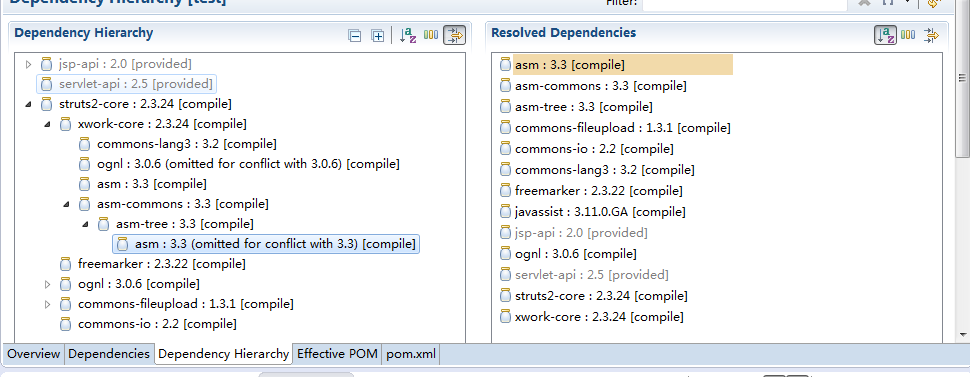


# 整合ssh框架

## 依赖传递

只添加了一个struts2-core依赖，发现项目中出现了很多jar，

这种情况 叫 依赖传递



## 依赖版本冲突的解决

1. 第一声明优先原则

<dependencies>

<!-- spring-beans-4.2.4 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.2.4.RELEASE</version>

</dependency>

<!-- spring-beans-3.0.5 -->

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

<version>2.3.24</version>

</dependency>

1. 路径近者优先原则

自己添加jar包

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

<version>4.2.4.RELEASE</version>

</dependency>

1. 排除原则

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

<version>2.3.24</version>

<exclusions>

<exclusion>

<groupId>org.springframework</groupId>

<artifactId>spring-beans</artifactId>

</exclusion>

</exclusions>

</dependency>

1. 版本锁定原则

<properties>

<spring.version>4.2.4.RELEASE</spring.version>

<hibernate.version>5.0.7.Final</hibernate.version>

<struts.version>2.3.24</struts.version>

</properties>

<!-- 锁定版本，struts2-2.3.24、spring4.2.4、hibernate5.0.7 -->

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

</dependencies>

</dependencyManagement>

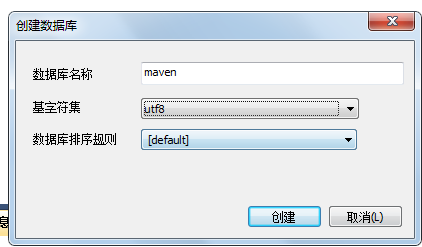
需求：

传客户ID 页面上显示客户信息

准备数据库

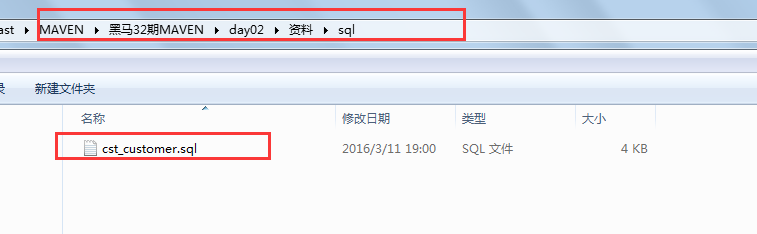
## 构建项目

1. 创建数据库，



1. 执行准备好的sql脚本

Sql脚本的位置：



1. 完善pom.xml文件，把ssh相关的依赖都添加上去

<!-- 属性 -->

<properties>

<spring.version>4.2.4.RELEASE</spring.version>

<hibernate.version>5.0.7.Final</hibernate.version>

<struts.version>2.3.24</struts.version>

</properties>

<!-- 锁定版本，struts2-2.3.24、spring4.2.4、hibernate5.0.7 -->

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</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-orm</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

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

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

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

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-core</artifactId>

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

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

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

</dependency>

</dependencies>

</dependencyManagement>

<!-- 依赖管理 -->

<dependencies>

<!-- spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

</dependency>

<!-- hibernate -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

</dependency>

<!-- 数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.6</version>

<scope>runtime</scope>

</dependency>

<!-- c3p0 -->

<dependency>

<groupId>c3p0</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.1.2</version>

</dependency>

<!-- 导入 struts2 -->

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-core</artifactId>

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

</dependency>

<!-- servlet jsp -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>2.0</version>

<scope>provided</scope>

</dependency>

<!-- 日志 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.2</version>

</dependency>

<!-- junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

<scope>test</scope>

</dependency>

<!-- jstl -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 设置编译版本为1.7 -->

<plugin>

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

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

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<!-- maven内置 的tomcat6插件 -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<version>1.1</version>

<configuration>

<!-- 可以灵活配置工程路径 -->

<path>/ssh</path>

<!-- 可以灵活配置端口号 -->

<port>8080</port>

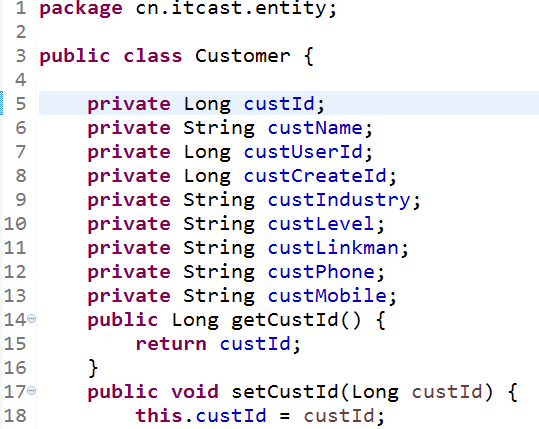
</configuration>

</plugin>

</plugins>

</build>

1. 完成实体类代码



1. 完成dao代码

接口

**package** cn.itcast.dao;

**import** cn.itcast.entity.Customer;

**public** **interface** CustomerDao {

**public** Customer getById(Long id);

}

实现类

**package** com.itcast.dao.impl;

**import** org.springframework.orm.hibernate5.support.HibernateDaoSupport;

**import** cn.itcast.dao.CustomerDao;

**import** cn.itcast.entity.Customer;

**public** **class** CustomerDaoImpl **extends** HibernateDaoSupport **implements** CustomerDao {

@Override

**public** Customer getById(Long id) {

**return** **this**.getHibernateTemplate().get(Customer.**class**, id);

}

}

1. 完成service代码

接口

**package** com.itcast.service;

**import** cn.itcast.entity.Customer;

**public** **interface** CustomerService {

**public** Customer getById(Long id);

}

实现类

package com.itcast.service.impl;

import com.itcast.service.CustomerService;

import cn.itcast.dao.CustomerDao;

import cn.itcast.entity.Customer;

public class CustomerServiceImpl implements CustomerService {

private CustomerDao customerDao;

public void setCustomerDao(CustomerDao customerDao) {

this.customerDao = customerDao;

}

@Override

public Customer getById(Long id) {

return customerDao.getById(id);

}

}

1. 完成action代码

package cn.itcast.action;

import com.itcast.service.CustomerService;

import com.opensymphony.xwork2.ActionSupport;

import cn.itcast.entity.Customer;

public class CutomerAction extends ActionSupport {

//两个成员变量

private Customer customer;

private Long custId;

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

private CustomerService customerService;

public void setCustomerService(CustomerService customerService) {

this.customerService = customerService;

}

public Long getCustId() {

return custId;

}

public void setCustId(Long custId) {

this.custId = custId;

}

public String findById(){

customer = customerService.getById(custId);

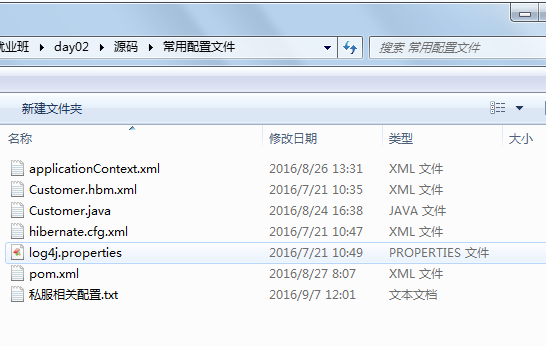
return SUCCESS;

}

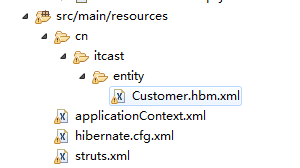
}

1. 拷贝配置文件并修改

从如下图位置拿到配置文件



放入到 src/main/resources目录中



修改内容 略

1. 修改web.xml 添加spring的监听

<listener>

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

</listener>

<context-param>

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

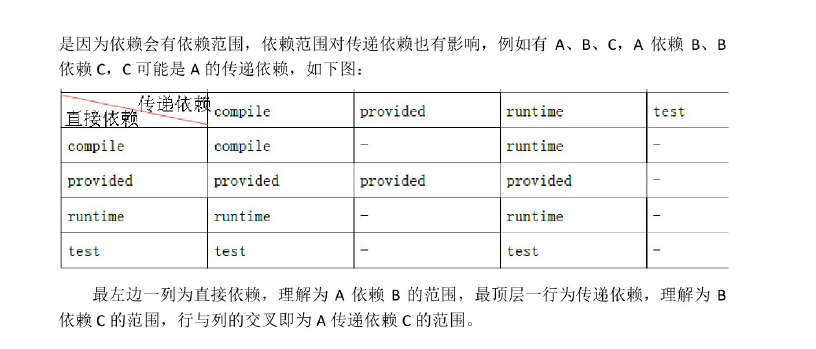
<param-value>classpath:applicationContext.xml</param-value>

</context-param>

1. 运行项目

# 分模块开发

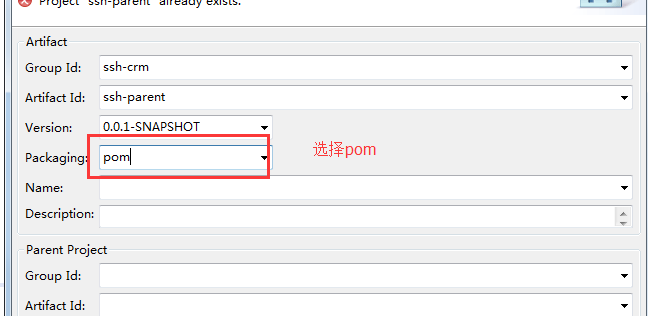
依赖范围对依赖传递造成的影响（了解）



父工程来管理 聚合

## 创建父工程：

1、



2、创建出的父工程如下



3、在pom.Xml中添加以下信息：

<!-- 属性 -->

<properties>

<spring.version>4.2.4.RELEASE</spring.version>

<hibernate.version>5.0.7.Final</hibernate.version>

<struts.version>2.3.24</struts.version>

</properties>

<!-- 锁定版本，struts2-2.3.24、spring4.2.4、hibernate5.0.7 -->

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</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-orm</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

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

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

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

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

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

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-core</artifactId>

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

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

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

</dependency>

</dependencies>

</dependencyManagement>

<!-- 依赖管理 -->

<dependencies>

<!-- spring -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

</dependency>

<!-- hibernate -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

</dependency>

<!-- 数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.6</version>

<scope>runtime</scope>

</dependency>

<!-- c3p0 -->

<dependency>

<groupId>c3p0</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.1.2</version>

</dependency>

<!-- 导入 struts2 -->

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-core</artifactId>

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

</dependency>

<!-- servlet jsp -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>2.0</version>

<scope>provided</scope>

</dependency>

<!-- 日志 -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.2</version>

</dependency>

<!-- junit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.9</version>

<scope>test</scope>

</dependency>

<!-- jstl -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 设置编译版本为1.7 -->

<plugin>

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

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

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<!-- maven内置 的tomcat6插件 -->

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>tomcat-maven-plugin</artifactId>

<version>1.1</version>

<configuration>

<!-- 可以灵活配置工程路径 -->

<path>/ssh</path>

<!-- 可以灵活配置端口号 -->

<port>8080</port>

</configuration>

</plugin>

</plugins>

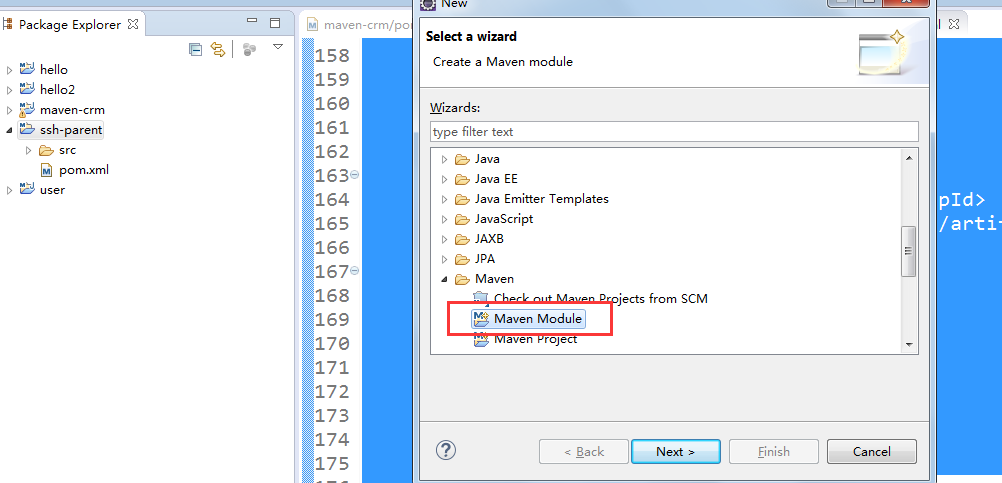
</build>

4、发布到本地仓库

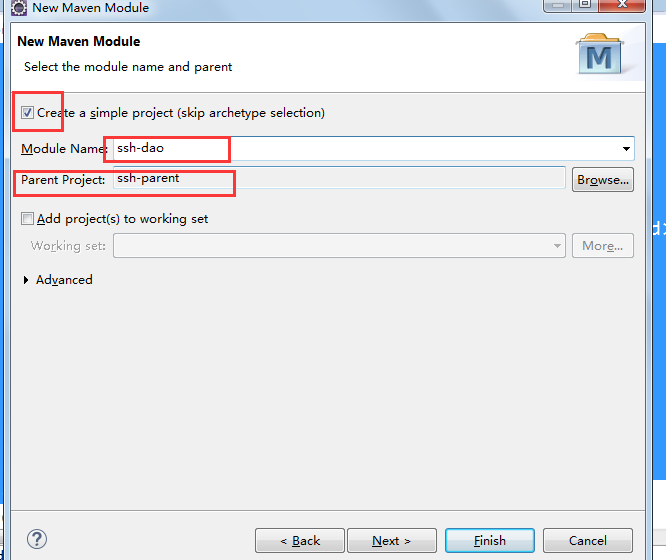
dao service web

## 创建dao子模块

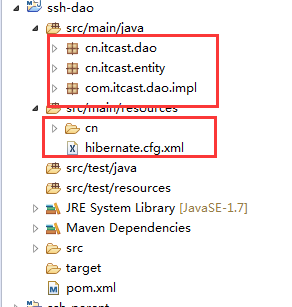
1、在ssh-parent项目上右击 ，创建时选择 Maven Module



2、填写子模块名称ssh-dao



3、把属于dao的代码拷贝到 该模块中：

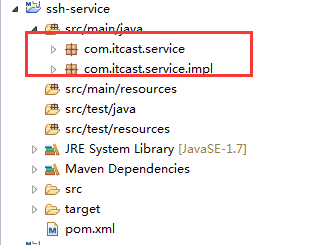


4、完成后发布到本地仓库中

## 创建service子模块

1、创建方式如上：

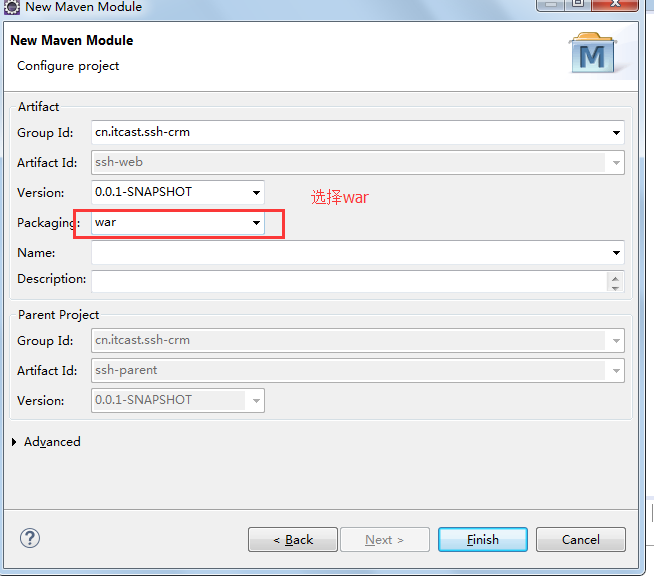
2、把属于service的代码拷贝到该工程中



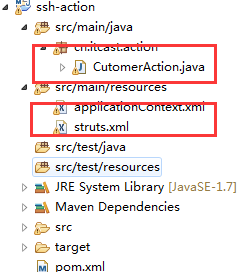
3、发布到本地仓库中

## 创建Action子模块

1、选择war的打包方式



1. 拷贝属于action的代码和配置文件



1. 修改web.xml 添加spring监听

<listener>

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

</listener>

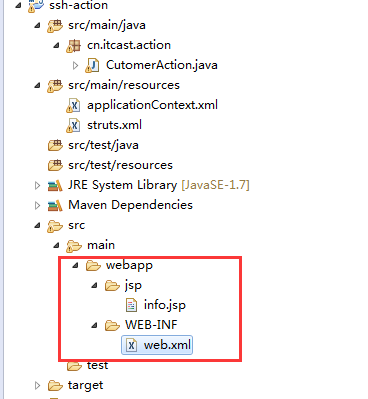
<context-param>

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

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

</context-param>

4、添加页面：

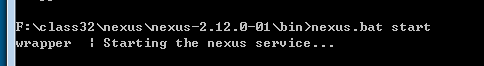
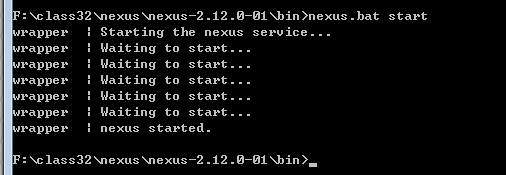


# 私服 nexus

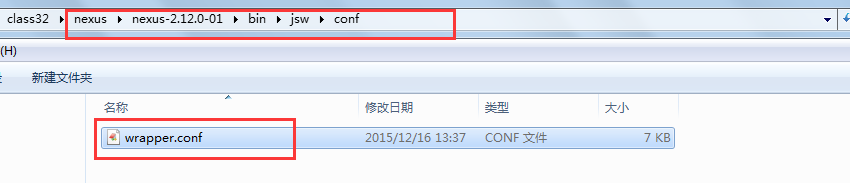
安装nexus

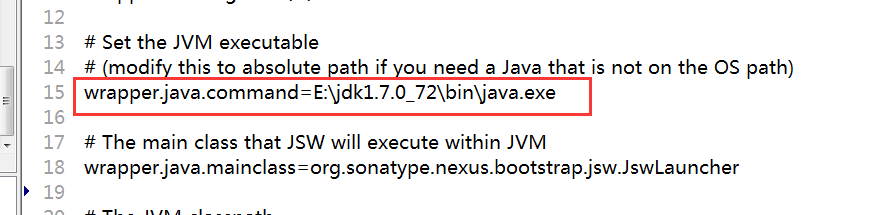


启动服务

启动失败的解决方法：

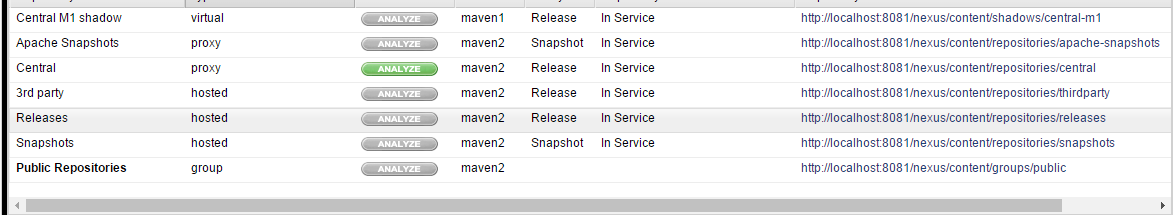




登录nexus

用户名/密码 admin/admin123

仓库类型



Virtual 虚拟仓库

Proxy 代理仓库

Hosted 宿主仓库 本地仓库

Group 组

需求：

把dao放到私服上，然后service从私服上下载

需求 ：将ssh\_dao的这个工程打成jar包，并放入到私服上去.

## 上传dao

第一步： 需要在客户端即部署dao工程的电脑上配置 maven环境，并修改 settings.xml 文件，配置连接私服的用户和密码 。

此用户名和密码用于私服校验，因为私服需要知道上传都 的账号和密码 是否和私服中的账号和密码 一致。

<server>

<id>releases</id>

<username>admin</username>

<password>admin123</password>

</server>

<server>

<id>snapshots</id>

<username>admin</username>

<password>admin123</password>

</server>

第二步： 配置项目pom.xml

配置私服仓库的地址，本公司的自己的jar包会上传到私服的宿主仓库，根据工程的版本号决定上传到哪个宿主仓库，如果版本为release则上传到私服的release仓库，如果版本为snapshot则上传到私服的snapshot仓库

<distributionManagement>

<repository>

<id>releases</id>

<url>http://localhost:8081/nexus/content/repositories/releases/</url>

</repository>

<snapshotRepository>

<id>snapshots</id>

<url>http://localhost:8081/nexus/content/repositories/snapshots/</url>

</snapshotRepository>

</distributionManagement>

注意：pom.xml这里<id> 和 settings.xml 配置 <id> 对应！

第三步：执行deploy命令发布到私服

## 下载dao

第一步 修改settings.xml

<profile>

<!--profile的id-->

<id>dev</id>

<repositories>

<repository>

<!--仓库id，repositories可以配置多个仓库，保证id不重复-->

<id>nexus</id>

<!--仓库地址，即nexus仓库组的地址-->

<url>http://localhost:8081/nexus/content/groups/public/</url>

<!--是否下载releases构件-->

<releases>

<enabled>true</enabled>

</releases>

<!--是否下载snapshots构件-->

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

<pluginRepositories>

<!-- 插件仓库，maven的运行依赖插件，也需要从私服下载插件 -->

<pluginRepository>

<!-- 插件仓库的id不允许重复，如果重复后边配置会覆盖前边 -->

<id>public</id>

<name>Public Repositories</name>

<url>http://localhost:8081/nexus/content/groups/public/</url>

</pluginRepository>

</pluginRepositories>

</profile>

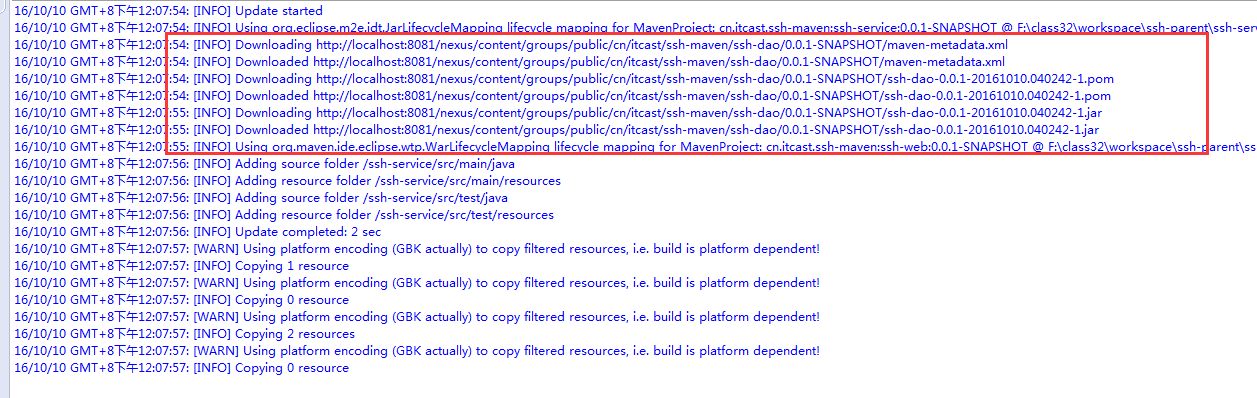
<activeProfiles>

<activeProfile>dev</activeProfile>

</activeProfiles>

第二步 删除本地仓库中的dao

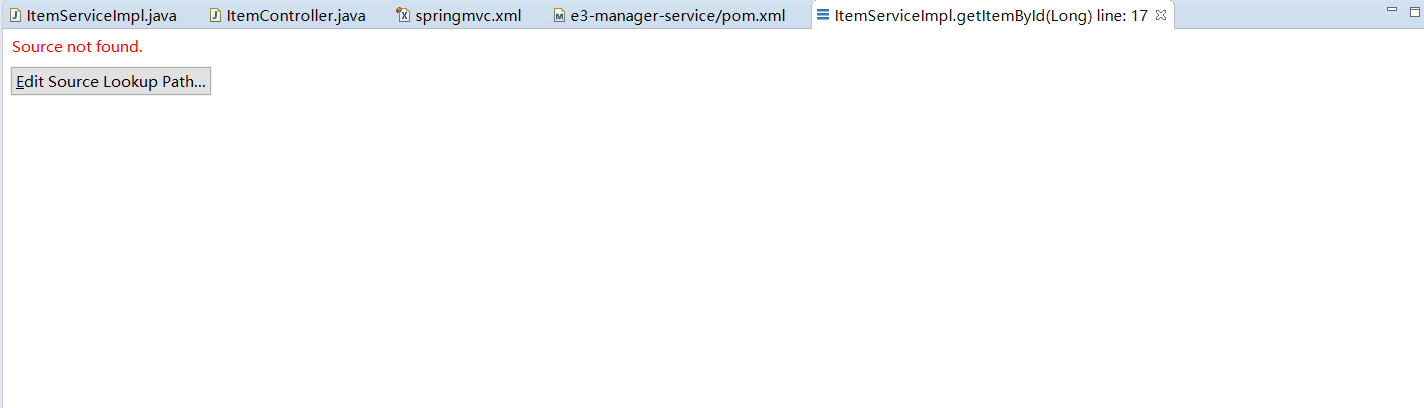
第三步 update service工程，出现以下信息说明已经成功



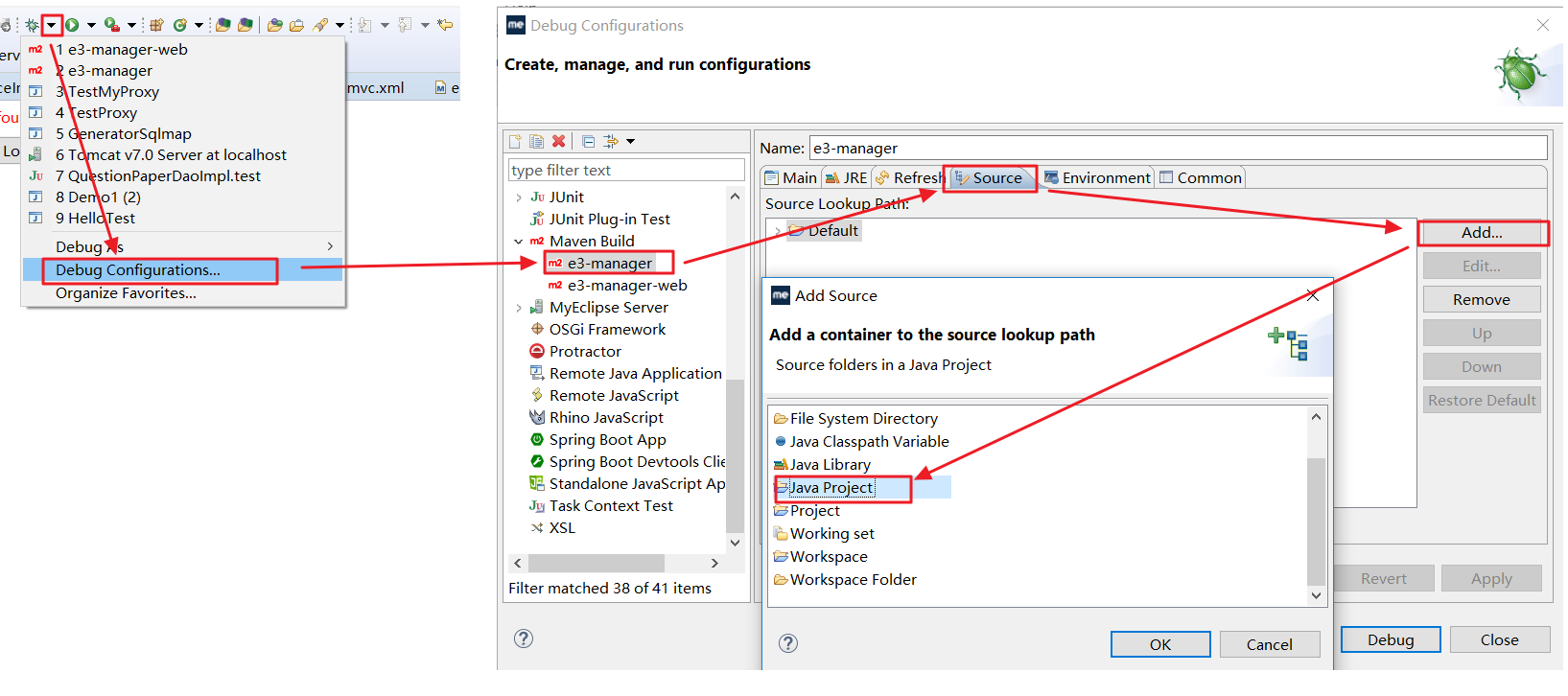
# Maven项目常见问题

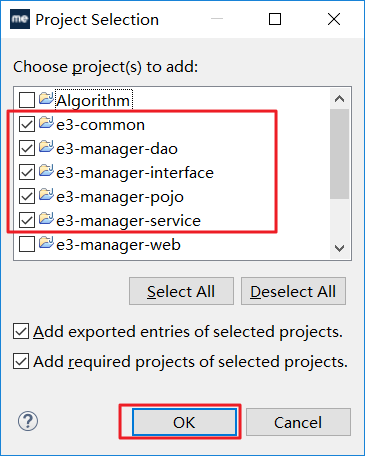
1. **在Maven项目使用Dubbo和Zookeeper搭建分布式架构时，debug显示找不到源码。**

问题详述：使用Dubbo将web项目和Service项目分离，通过Dubbo架构进行通信，那么如果我们在Service层打断点，然后通过浏览器访问Web层，Service层的断点有反应，但会显示找不到源码



**解决方案：**将与Service子项目相关的项目全部添加到源码当中。





1. **在使用Dubbo分布式架构时，Debug会抛超时异常。**

这是因为dubbo默认连接时间为1000ms，超过1000ms就会抛出超时异常，如果超时Dubbo会默认尝试三次，如果都失败则不再进行。

**解决方案：**

但在Debug模式下1S中显然不够，我们需要将时间延迟，我们只需要在**服务调用者**的配置文件中加入timaout属性，指定超时时间即可。

