# Maven 实战

# 传递依赖冲突解决（了解）

传递依赖：A（项目）依赖B，B依赖C（1.1版本），B是A的直接依赖，C就是A的传递依赖

导入依赖D，D依赖C（1.2版本）

## Maven自己调解原则

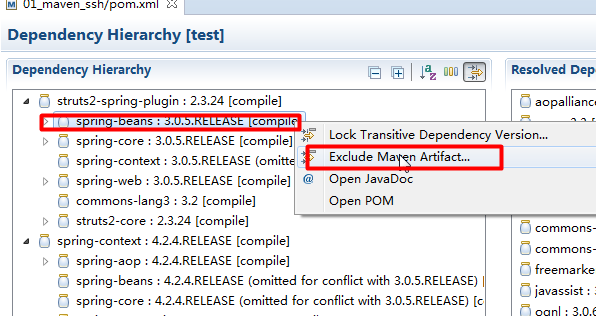
### 第一声明者优先原则

谁先定义的就用谁的传递依赖。

### 路径近者优先原则

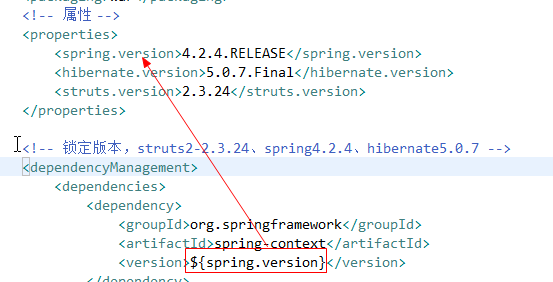
直接依赖级别高于传递依赖。

## 排除依赖





## 版本锁定（推荐使用）



# 通过maven整合SSH框架（重点）

## 搭建struts2环境

1. 创建strut2配置文件：struts.xml
2. 在web.xml中配置strut2的核心过滤器

<!-- 配置Struts核心过滤器 -->

<filter>

<filter-name>struts</filter-name>

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

</filter>

<filter-mapping>

<filter-name>struts</filter-name>

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

</filter-mapping>

## 搭建spring环境

1. 创建spring配置文件applicationContext.xml
2. 在web.xml中配置监听器：ContextLoaderListener

<!-- 配置监听器：默认加载WEB-INF/applicationContext.xml -->

<listener>

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

</listener>

<!-- 通过上下文参数指定spring配置文件路径 -->

<context-param>

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

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

</context-param>

## 搭建Hibernate环境

1. 创建Hibernate核心配置文件

<?xml version=*'1.0'* encoding=*'utf-8'*?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<!-- 会话工厂 -->

<session-factory>

<!-- 数据库方言，根据数据库选择 -->

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQL5Dialect</property>

<!--为了方便调试是否在运行hibernate时在日志中输出sql语句 -->

<property name=*"hibernate.show\_sql"*>true</property>

<!-- 是否对日志中输出的sql语句进行格式化 -->

<property name=*"hibernate.format\_sql"*>true</property>

<property name=*"hibernate.hbm2ddl.auto"*>update</property>

</session-factory>

</hibernate-configuration>

## Struts2跟spring整合

整合关键点：action对象创建，交给spring创建

1. 创建action类
2. 将action对象配置到spring配置文件中
3. 在struts.xml中在action节点中class属性配置为spring工厂中action对象bean的id

## Spring跟Hibernate框架整合

整合关键点：1、数据源dataSource交给spring 2、SessionFactory对象创建交给spring创建 3、事务管理

1. 配置dataSource

<!-- 加载属性文件 -->

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

<bean id=*"dataSource"* class=*"com.mchange.v2.c3p0.ComboPooledDataSource"*>

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

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

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

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

</bean>

1. 配置SessionFactory对象

<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

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

<property name="configLocations" value="classpath:applicationContext.xml"></property>

</bean>

1. 事务管理
   1. 配置事务管理器： PlatFormTransactionManager:接口
      1. Jdbc：DataSourceTransactionManager
      2. Hibernate:HibernateTransactionManager

<!-- 配置事务管理器 -->

<bean id=*"transactionManager"*

class=*"org.springframework.orm.hibernate5.HibernateTransactionManager"*>

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

</bean>

<!-- xml方式管理事务 -->

<!-- 配置通知：具体增强逻辑 -->

<tx:advice id=*"txAdvice"*>

<tx:attributes>

<!-- 匹配业务类中方法名称 -->

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

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

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

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

<tx:method name=*"\*"*/>

</tx:attributes>

</tx:advice>

<!-- 配置aop -->

<aop:config>

<!-- 配置切点：具体哪些方法要增强（真正被增强的方法）-->

<aop:pointcut expression=*"execution(\* cn.itcast.service.\*.\*(..))"* id=*"cut"*/>

<!-- 配置切面：将增强逻辑作用到切点 （通知+切入点） -->

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

</aop:config>

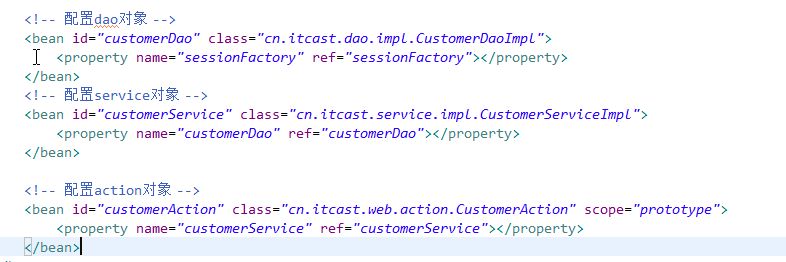
<!-- xml方式管理事务 -->

## 需求

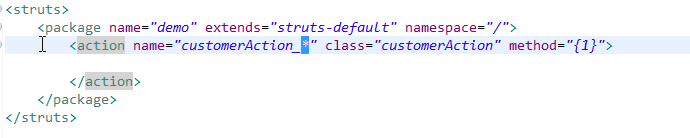
在地址栏输入action请求：http://locahost:8080/project/customerAction\_findOne.action?custId=1，action-service-dao。完成客户查询

## 具体实现

1. 创建客户实体类、映射文件、将映射文件引入Hibernate核心配置文件中
2. 创建action、service、dao。完成注入
   1. 在类中添加属性生成set方法
   2. 在spring配置文件中完成注入



1. 在struts.xml中配置action，配置结果视图



总结：

1. 页面提交参数，在服务端action接收参数 参
2. 调用业务层方法—》dao的方法—》DB 调
3. 将返回的数据存值栈 存
4. 配置结果视图，跳转页面 转

# 通过maven对项目进行拆分、聚合（重点）

对现在已有maven ssh项目进行拆分，拆分思路：将dao层的代码已经配置文件全体提取出来到一个表现上独立的工程中。同样service、action拆分。

ssh-parent: 父工程

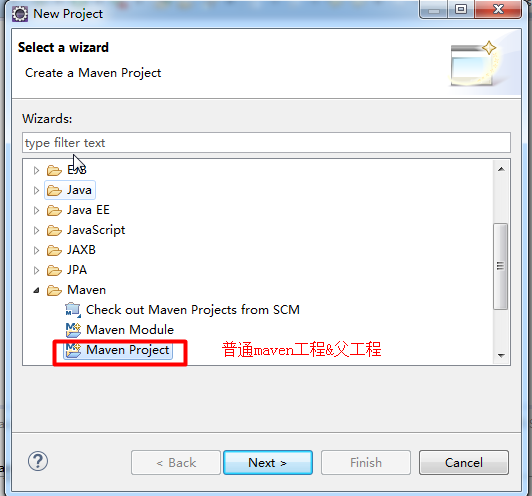
ssh-dao:（子模块）

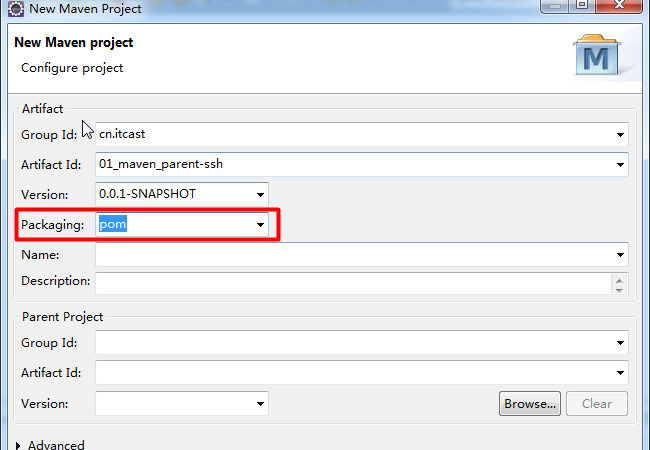
ssh-service

ssh-web

拆分完成对拆分后的项目进行聚合，提出概念父工程

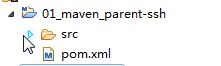
## 创建父工程





创建好父工程目录结构：只有pom.xml，可以推断父工程不进行编码。

1. 项目需要的依赖的信息，在父工程中定义，子模块继承过程
2. 将各个子模块聚合到一起



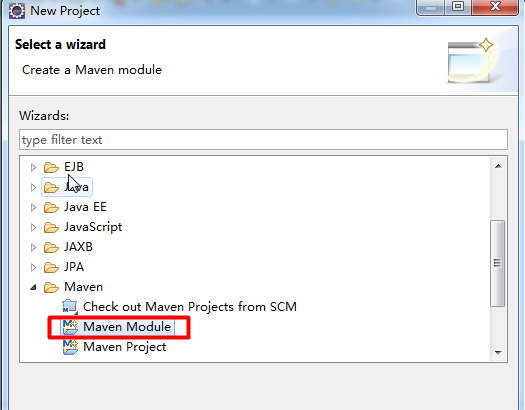
## 将创建父工程发布到本地仓库

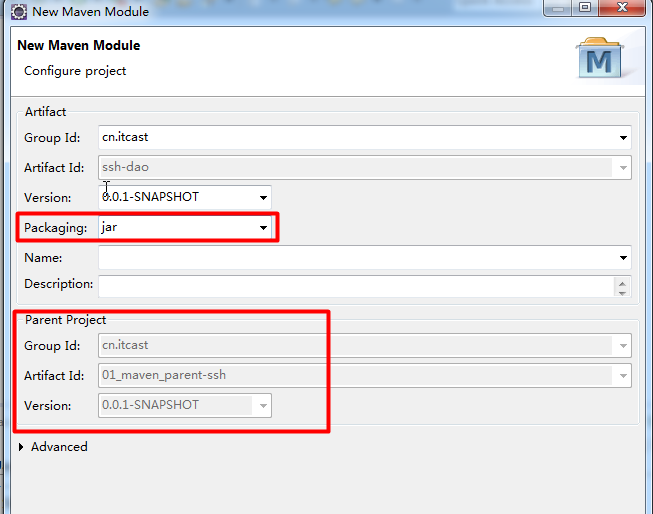
将来service、dao工程发布到本地仓库。发布的service工程会报错。

如果忘记此步骤，将父工程发布到本地仓库。

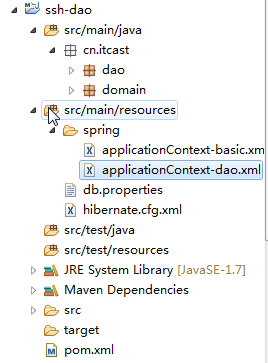
## 创建子模块ssh-dao

Ssh-dao负责数据访问层：包含dao相关代码&配置文件





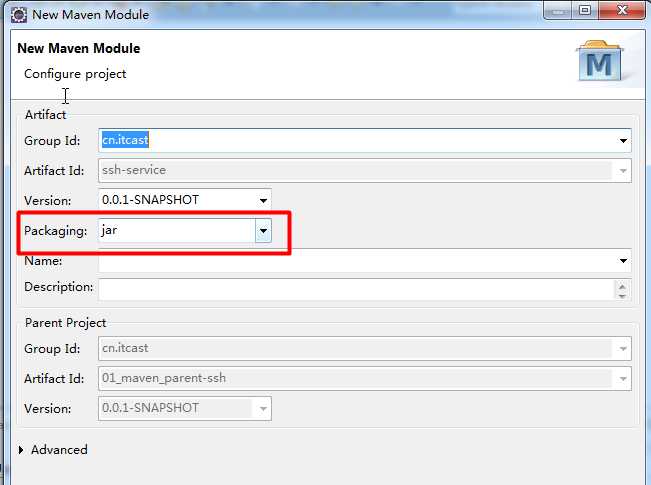
### 将spring的配置文件拆分

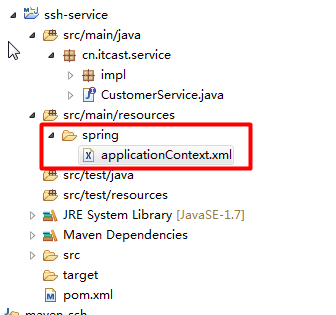


：项目基础的信息

：dao层bean对象

## 创建子模块ssh-service



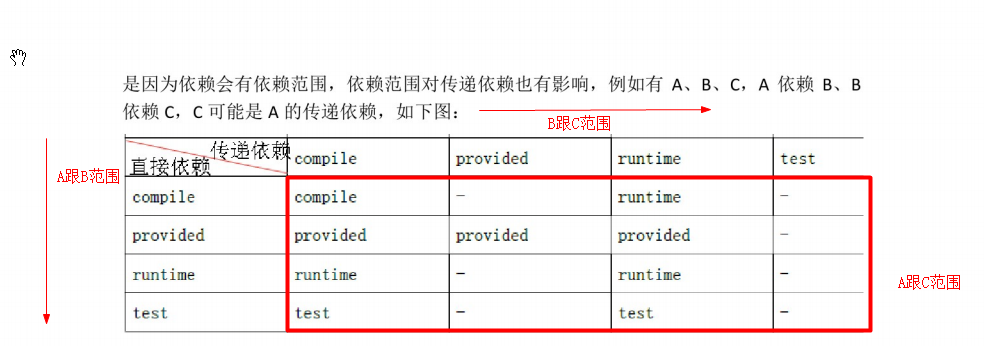


将ssh-dao工程打包到本地仓库；

在service工程pom.xml文件添加ssh-dao的依赖

### 单元测试

1. 批量加载spring配置文件
   1. classpath:spring/applicationContext-\*.xml
   2. classpath\*:spring/applicationContext-\*.xml：既要加载本项目中配置文件，还要加载jar包中配置文件
2. 传递依赖范围（了解）



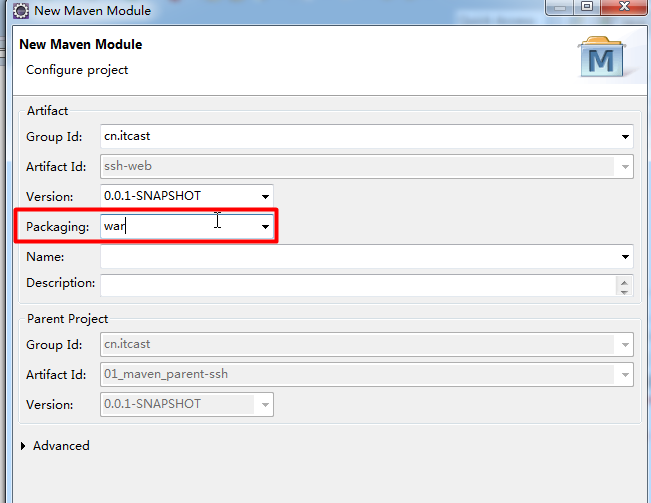
A:ssh-service 工程

B:ssh-dao工程

C:junit单元测试

总结：当项目中需要的某一个依赖没有传递过来。在自己工程中添加对应依赖就可以。

## 创建子模块ssh-web：war



在ssh-web项目中添加service工程依赖。

Web.xml



## 运行方式

Maven方式：

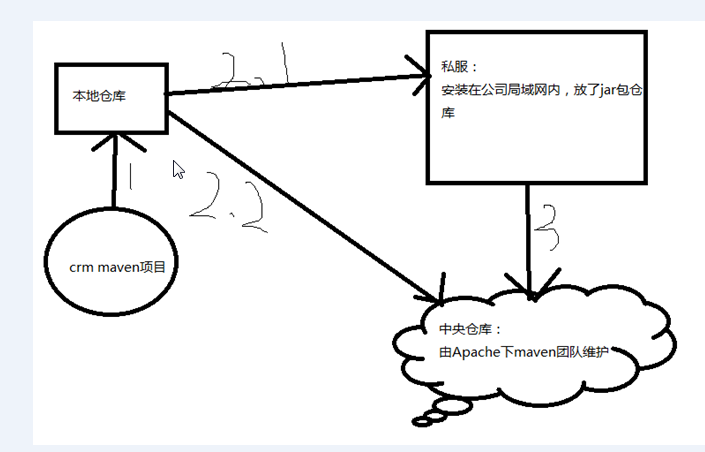
方式1：运行父工程。父工程将各个子模块聚合到一起。将ssh-web打war包发布到tomcat

方式2：直接运行web工程

其他方式：

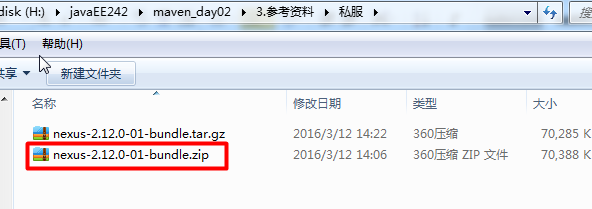
部署到tomcat

# 私服应用（了解）

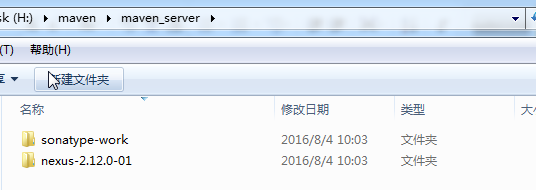


## 私服安装

1. 下载安装包



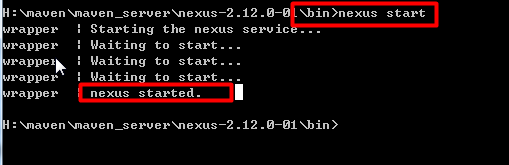
1. 解压到本地磁盘



1. 使用管理员权限打开dos,在dos下执行命令安装私服



1. 启动服务

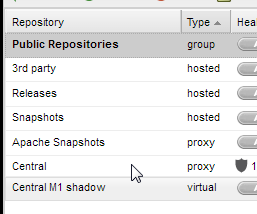


1. 找到私服的访问url：

<http://localhost:8081/nexus/#welcome>

登陆：admin/admin123

## 私服仓库类型



Hosted:宿主仓库

存放本公司开发的jar包（正式版本、测试版本、第三方：存在版权问题--Oracle）

Proxy:代理仓库

代理中央仓库、Apache下测试版本的jar包

Group：组仓库

将来连接组仓库。包含Hosted:宿主仓库，Proxy:代理仓库

## 上传jar包到私服上（应用）

1. 在maven目录下conf/settings.xml.认证：配置用户名密码

<server>

<id>releases</id>

<username>admin</username>

<password>admin123</password>

</server>

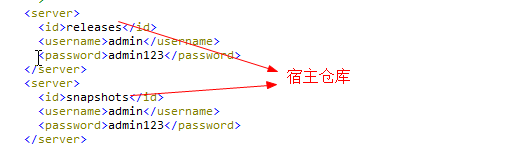
<server>

<id>snapshots</id>

<username>admin</username>

<password>admin123</password>

</server>



1. 在将要上传的项目的pom.xml中配置jar包上传路径url

<distributionManagement>

<repository>

<id>releases</id>

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

</repository>

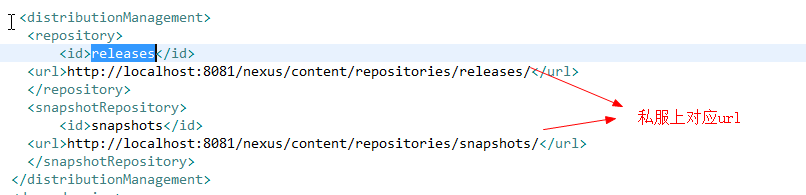
<snapshotRepository>

<id>snapshots</id>

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

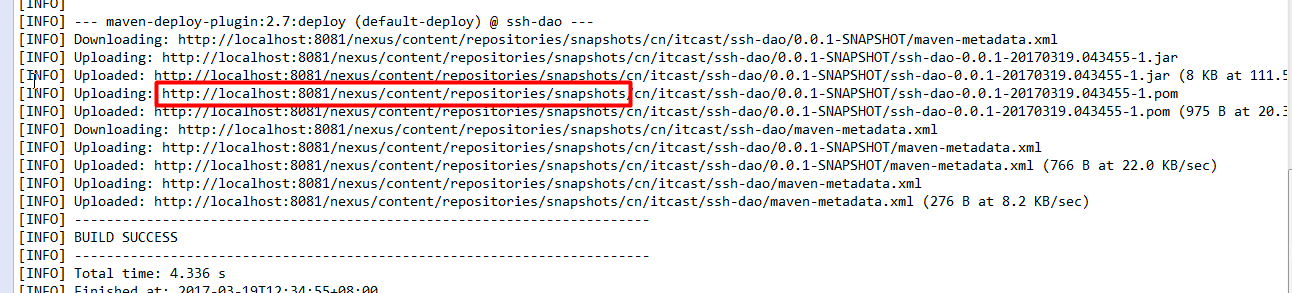
</snapshotRepository>

</distributionManagement>



1. 执行命令发布项目到私服（上传）

deploy



## 下载jar包到本地仓库（应用）

1. 在maven目录下conf/settings.xml。配置模板

<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>



1. 激活模板

<activeProfiles>

<activeProfile>dev</activeProfile>

</activeProfiles>

# Maven的好处

1. 不再拷贝jar包。项目不需要存放jar包。导致项目源代码小
2. 使用maven开发的工程,如果环境统一导入别的maven的项目不会报错
3. 代码耦合度进一步降低.
4. 方便项目进行升级
5. 节省人力成本

www.github.com

# 总结

1. 使用maven整合ssh框架（掌握）
2. 拆分maven工程（重点）
   1. 将每一层代码&配置文件全部提取一个表现独立工程
3. 私服（了解）