第2天: Maven

一、学习目标

1. 理解依赖传递
2. 解决版本冲突
3. 理解继承特性
4. 理解聚合特性
5. 搭建和使用私服
6. 模块化构建工程
7. 重整SSM工程

二、Maven概念

- Apache Maven是一种创新的软件项目管理工具，提供了一个项目对象模型（POM）文件的新概念来管理项目的构建，相关性和文档。最强大的功能就是能够自动下载项目依赖。

2.1 官方网站

- 网址: http://maven.apache.org
- 下载: http://archive.apache.org/dist/maven

2.2 功能

- 主要服务于基于 Java 平台的项目构建，依赖管理和项目信息管理。

- 跨平台: 基于 Java 平台的项目构建
- 自动化: 自动下载依赖
- 标准化: 项目结构统一标准
- 可重用: POM可重用设计

2.3 核心概念

2.3.1 仓库

- 在Maven中，任何一个jar包、插件或者项目构建的输出，都可以称之为构件。
- Maven在某个统一的位置存储所有项目的共享的构件，这个统一的位置，我们就称之为仓库。
- **仓库就是存放构件的位置。**

中央仓库

- 包含了绝大多数流行的开源 Java 构件，以及源码、作者信息、SCM(Software config Managment)、 信息、许可证信息等。开
- 源的 Java 项目依赖的构件都可以在这里下载到。
- 地址: http://repo.maven.apache.org/maven2

本地仓库

- 默认位置：在操作系统用户的目录下.m2/repository/。仓库位置可以修改。

远程仓库

- 远程仓库是中央仓库的替代品(镜像)。
- 主要作用是加快周边地区的访问速度。

2.3.1 坐标

- 在Maven中, 任何一个构件都有一个唯一标识。
- 这个标识至少包含组织编码(**groupId**)、项目编码(**artifactId**)、版本信息(**version**)。
- **坐标就是构件在仓库中的定位。**

2.4 依赖管理

- 在 pom.xml 中的节点 dependency 为1个依赖。
- 坐标是依赖的基本信息。

```
<dependencies>
    <!-- Spring依赖包 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.1.3.RELEASE</version>
    </dependency>
</dependencies>
```

2.4.1 依赖范围

- 依赖范围 scope 用来控制依赖和编译，测试，运行的 classpath 的关系。

compile：编译域

- 这个是 maven 中 scope 的默认值，可以缺省。
- compile 表示被依赖项目需要同当前项目编译时一起进行编译;
- 项目测试期以及本项目运行时期同样生效;
- 打包的时候需要包含进去。

test：测试域

- 表示被依赖的项目仅在项目进行测试的时候生效;
- 一般单元测试（junit）等依赖包配置为 test
- 项目运行时不会生效。

provided：提供域

- provided 意味着打包的时候可以不用打包进去，别的容器会提供。
- 比如 servlet-api，jsp-api, tomcat 这些容器会提供;
- 所以打包，运行时无需提供。

runtime：运行域

- 表示被依赖项目不会参与项目的编译;
- 但项目的测试期和运行时期会参与, 比如jdbc;

- 自定义构件，指定 systemPath；
- 跟 provided 相似，在系统中要以外部 JAR 包的形式提供;
- maven 不会在 repository 查找它。

import：导入

- 只使用在<dependencyManagement标签使用;
- 表示从其它的 pom 中导入dependency 的配置。

依赖范围	说明	编译是否有效	测试是否有效	运行是否有效	是否打包	实际应用
compile	编译范围	√	√	√	√	ssm
test	测试范围	√	√	×	×	junit
provided	容器范围	√	√	×	×	servlet-api
runtime	运行范围	×	√	√	√	jdbc驱动

2.4.2 依赖传递

依赖传递案例

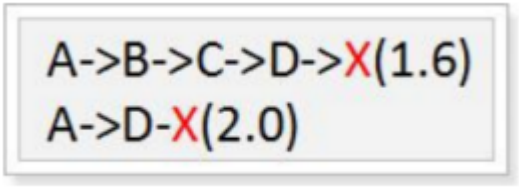
- 使用Spring框架时只需要依赖spring-context，即可完成spring的IOC容器的使用;
- 其它的jar包（spring-beans、spring-core、spring-expression），都是通过依赖传递导入。

依赖传递选项

- 在依赖节点 dependency 中, 可以使用optional标签控制当前的依赖是否向下传递;
- 默认值为 false，表示向下传递。

依赖冲突

- 短路优先



最终 A 依赖的 X 的版本为 2.0

```
    <dependency>
        <groupId>test</groupId>
        <artifactId>B</artifactId>
        <version>1.0</version>
    </dependency>
    <dependency>
        <groupId>test</groupId>
        <artifactId>D</artifactId>
        <version>1.0</version>
    </dependency>
</dependencies>
```

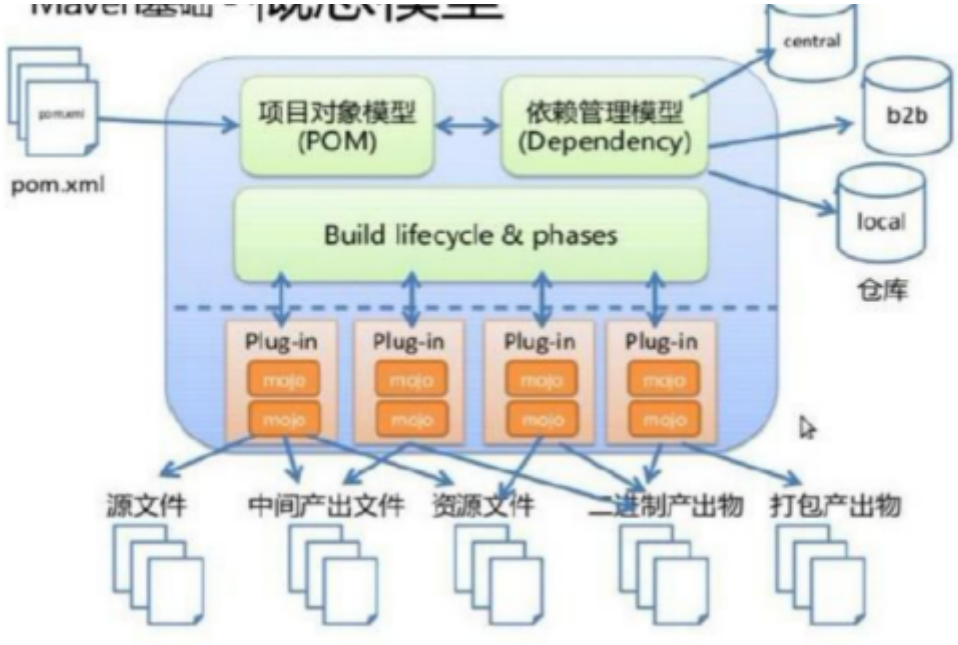
说明: A项目中依赖的B项目和D项目均有X项目的依赖传递, 根据短路优先原则, A项目最终传递的X项目 **来自于D项目中的2.0版本。**

- 解决冲突(**排除依赖**)

```
<dependency>
    <groupId>test</groupId>
    <artifactId>D</artifactId>
    <version>1.0</version>
    <!-- 排除不需要的依赖 -->
    <exclusions>
        <exclusion>
            <groupId>test</groupId>
            <artifactId>X</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

说明: 排除D项目中的X项目将继续根据短路优先原则 **选用其他X项目。**

2.5 概念模型



2.6 idea配置

2.5.1 为什么使用idea

- 手工操作较多，编译、测试、部署等工作都是独立的，很难一步完成。每个人的 IDE 配置都不同，很容易出现本地代码换个地方编译就报错。

2.5.2 使用ant

- 没有一个约定的目录结构，必须明确让 ant 做什么，什么时候做，然后编译，打包；没有生命周期，必须定义目标及其实现的任务序列；没有集成依赖管理。

2.5.3 使用maven

- 项目结构有标准化的约定, 只需要按约定存放源码, maven即可帮助完成一些事情。

2.5.4 idea配置

- File > Settings > Build, Execution, Deployment > Build Tools > Maven
- Maven home directory: 安装目录
- com.itheima.maven.day02.domain.User settings file: C:\Users\Administrator.m2\settings.xml
- Local repository: 本地仓库路径

三、Maven高级特性

3.1 继承

3.1.1 作用

- 继承为了消除重复，可以把 pom 中很多相同的配置提取出来。
- 在使用的时候子工程直接继承父工程的依赖版本号，子工程中可缺省版本号。
- 方便统一管控项目的依赖版本问题。

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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.itheima</groupId>
    <artifactId>maven-day02-parent</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!-- 父项目packaging必须是pom -->
    <packaging>pom</packaging>

    <modules>
        <module>child1</module>
    </modules>

    <!-- 定义全局变量(可被继承) -->
    <properties>
        <mybatis.version>3.5.0</mybatis.version>
    </properties>

    <!-- 依赖管理(可被继承有版本控制效果) -->
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.mybatis</groupId>
                <artifactId>mybatis</artifactId>
                <version>${mybatis.version}</version>
            </dependency>
        </dependencies>
    </dependencyManagement>

</project>
```

3.1.3 子工程: child1

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">
    <!-- 继承父工程 -->
    <parent>
        <artifactId>maven-day02-parent</artifactId>
        <groupId>com.itheima</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>

    <modelVersion>4.0.0</modelVersion>

    <artifactId>child1</artifactId>

```

```
            <artifactId>mybatis</artifactId>
            <!-- 可以引用父工程的变量(继承) -->
            <!-- 可以缺省此项(使用继承的版本) -->
            <version>${mybatis.version}</version>
        </dependency>
    </dependencies>
</project>
```

3.2 聚合

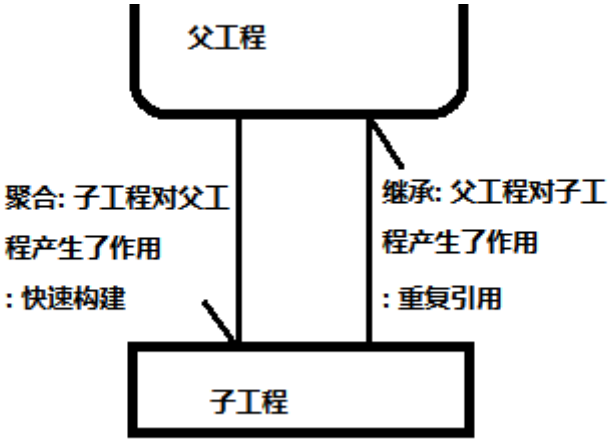
3.2.1 作用

- 快速构建,编译,测试,打包项目

3.2.2 聚合工程: agg

```
<?xml version="1.0" encoding="UTF-8"?>
<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.itheima</groupId>
    <artifactId>maven-day02-agg</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!-- 父项目packaging必须是pom -->
    <packaging>pom</packaging>

    <!-- 聚合现有工程 -->
    <!-- 聚合项目编译时将子项目亦编译并打包 -->
    <modules>
        <module>../maven-day02-parent/child1</module>
    </modules>

</project>
```

- 声明: 聚合与继承的区别在于其**作用方向不同** 。



四、Maven私服

4.1 中央仓库的缺点

- 地址
  - 目前来说: http://repo1.maven.org/maven2/是真正的 maven 中央仓库的地址;
  - 该地址内置在 maven 的源码中,其他的都是镜像。
- 索引
  - 中央仓库带有索引文件以方便用户对其进行搜索，完整的索引文件大小约为 60M，索引每周更新一次。
- 黑名单
  - 如果某个 IP 地址恶意的下载中央仓库内容，例如全公司 100 台机器使用同一个 IP 反复下载，这个 IP（甚至是 IP 段）会进入黑名单。
  - 因此稍有规模的使用maven 时，应该用 nexus 架设私服。

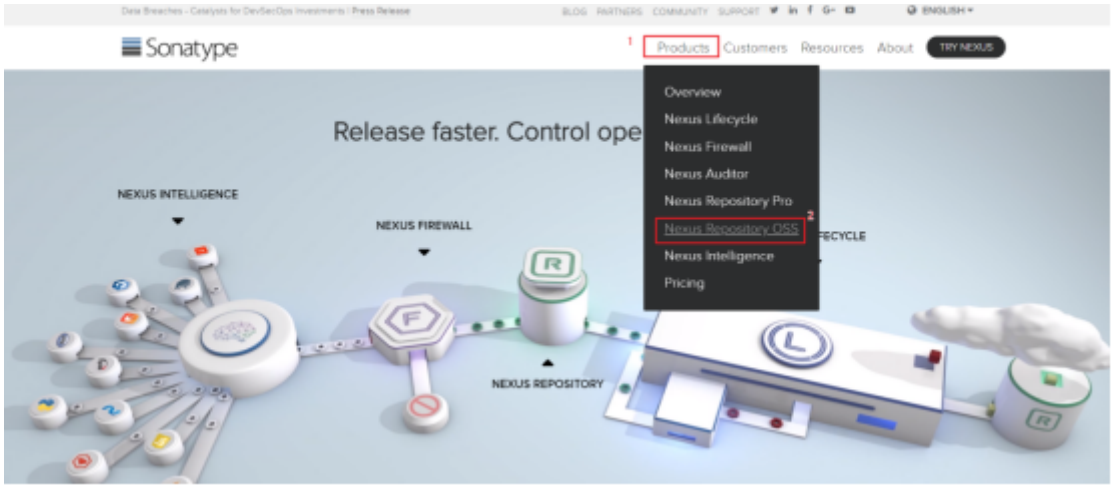
4.2 私服的好处

- 缓存 maven 中央仓库的 jar 包, 就不需要每次本地仓库没有 jar 包都到中央仓库下载, 而是到私服下载。
- 有些公司无法上网，那如何连接中央仓库呢？只需连接私服，而私服能连网 到中央仓库。
- 方便公司内部不同团队或者项目共享 jar 包，需要共享 jar 包，可以上传到私服，通过私服共享。

4.3 搭建私服

4.3.1 下载

- nexus 的官网 http://www.sonatype.com/ 。下载 Nexus Repository Manager OSS 2.xx 。
- 课前资料中已经下载 “ nexus-2.12.0-01-bundle.zip ” 。 也可以使用 “nexus-2.1.2.war”直接放置在 tomcat 的 webapps 目录下。



4.3.2 安装

1. 解压“nexus-2.12.0-01-bundle.zip”
2. 修改默认端口, 路径为：nexus-2.12.0-01\conf\nexus.properties。
3. “ nexus-2.12.0-01\bin\jsw\windows-x86-64\ ” 目录下, 双击 console-nexus.bat 为后台启动 nexus；
4. 双击“install-nexus.bat”则会注册为一个 windows 服务。

4.3.3 启动

- 在路径 “ nexus-2.12.0-01\bin\jsw\windows-x86-64\ ” 目录下 。
- 双击 console-nexus.bat 为后台启动 nexus；
- 双击“install-nexus.bat”则会注册为一个 windows 服务。

4.3.4 访问

- 启动 nexus 后，可以在浏览器中输入：http://localhost:8081/nexus
- 默认管理员账户: admin|admin123

4.4 私服配置

4.4.1 仓库类型

- nexus 里可以配置 **3 种类型** 的仓库和组，分别是 **proxy**、**hosted**、**virtual**、group。
- proxy：是远程仓库的代理。比如说在 nexus 中配置了一个 central repository 的 proxy，当用户向这个 proxy 请求一个 artifact，这个 proxy 就会先在本地查找，如果找不到的话，就会从远程仓库下载，然后返回给用户，相当于起到一个中转的作用。
- hosted：是宿主仓库，用户可以把自己的一些构件，deploy 到 hosted 中，也可以手工上传构件到 hosted 里。比如说 oracle 的驱动程序，在central repository 是获取不到的，就需要手工上传到 hosted 里。
- virtual：是中央仓库镜像,支持 M1 老版本。
- group：是仓库组，在 maven 里没有这个概念，是 nexus 特有的。目的是将上述多个仓库聚合，对用户暴露统一的地址，这样用户就不需要在 pom 中配置多个地址，只要统一配置 group 的地址就可以。右边那个 Repository Path 可以点击进去，看到仓库中artifact列表。

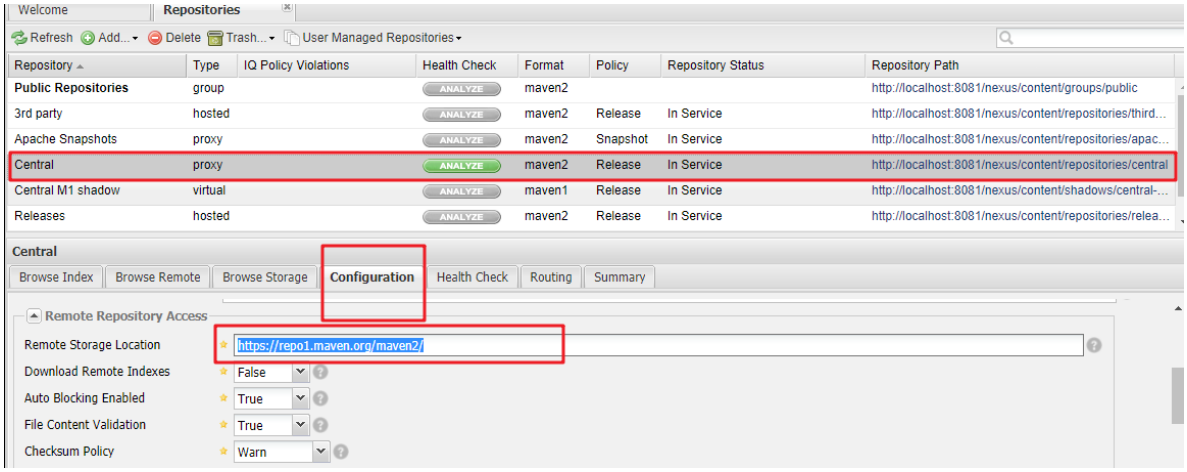
4.4.2 仓库示例

- 3rd party: 无法从公共仓库获得的第三方发布版本的构件仓库
- apache snapshots: 用来代理 apache maven 仓库快照版本的构件仓库

- snapshots: 用来部署管理内部的快照版本构件的宿主类型仓库

4.4.3 代理国内镜像

- Nexus的中央仓库，默认配置的是maven的中央仓库：http://repo1.maven.org/maven2
- 建议配置为阿里云：http://maven.aliyun.com/nexus/content/groups/public



4.4.4 下载私服资源

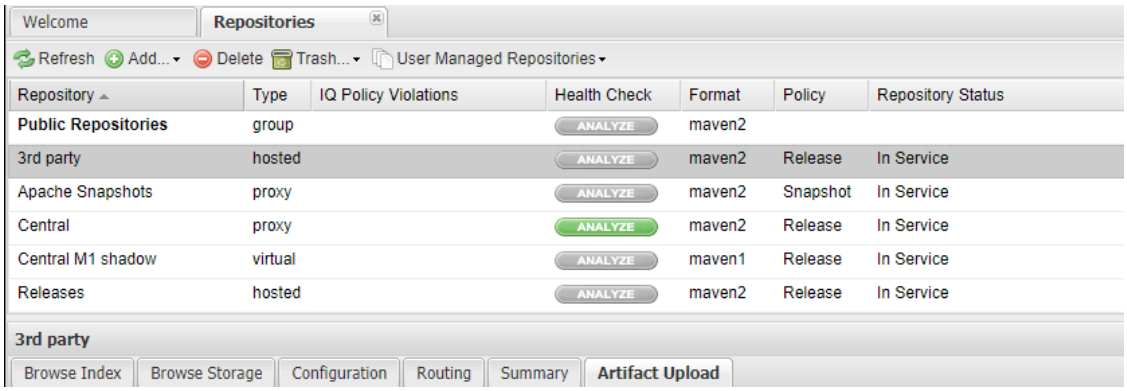
```
<!-- 远程仓库配置 -->
<mirror>
    <!-- ID -->
    <id>nexus</id>
    <!-- 名称: 私服 -->
    <name>nexus</name>
    <!-- 下载地址 -->
    <url>http://127.0.0.1:8081/nexus/content/groups/public/</url>
    <!-- 拦截请求 -->
    <mirrorOf>central</mirrorOf>
</mirror>
```

4.4.5 上传资源

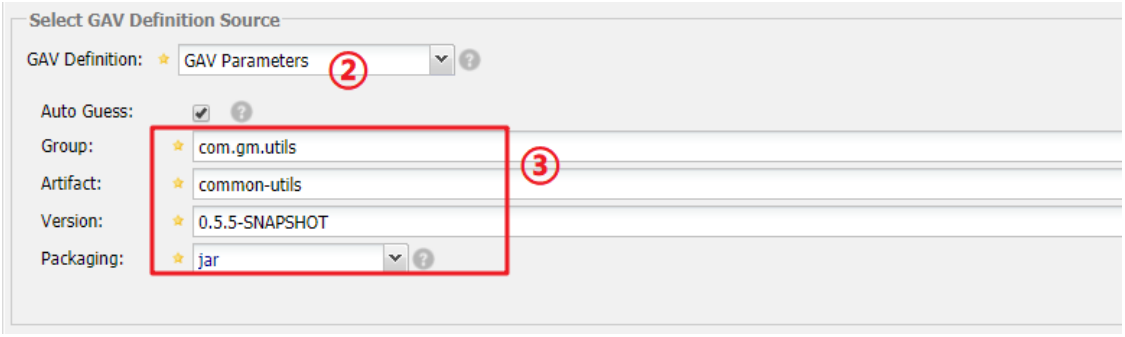
- 只能上传到**3rd party**这样的hosted类型仓库

后台上传

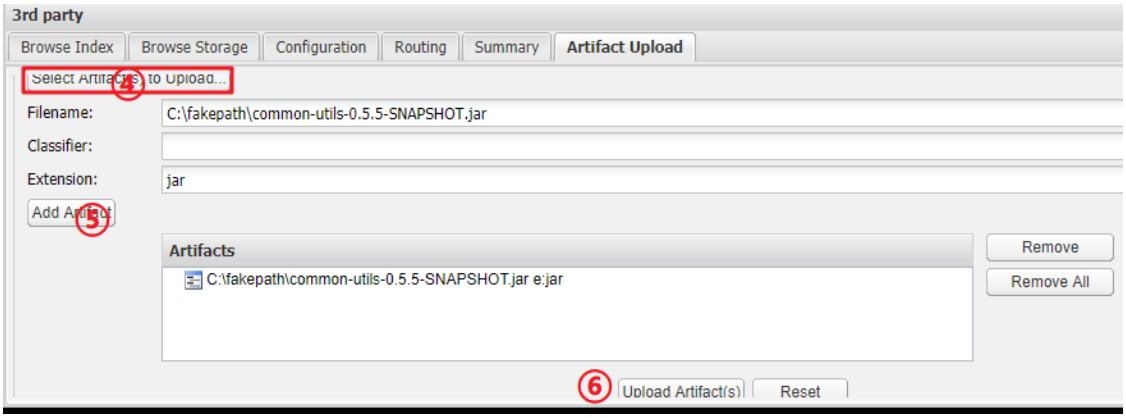
1. 进入后台选中3rd party仓库



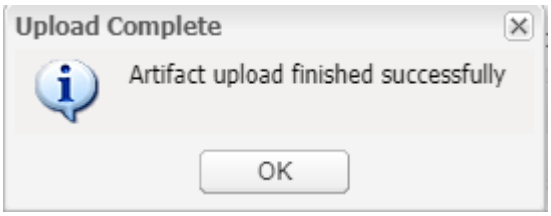
2. 进入-Select GAV Definition Source-

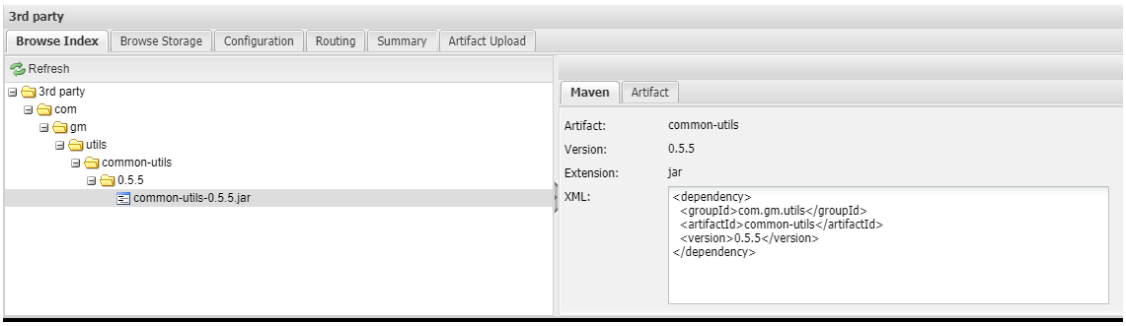


3. 下拉滚动条进入-Select Artifact(s) for Upload-



4. 上传成功后查看资源





自动上传

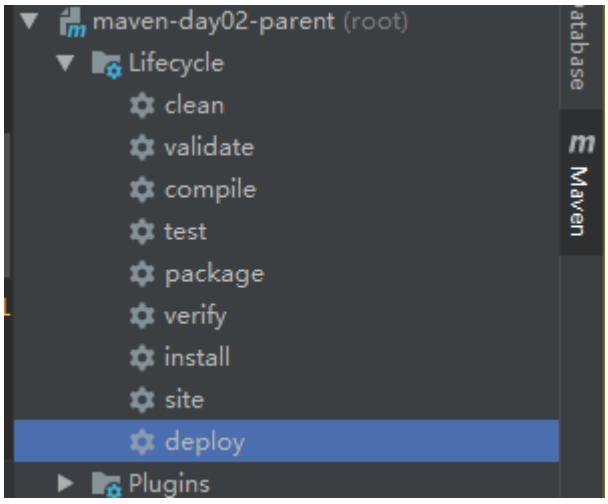
1. 在POM中配置仓库地址

```
        <id>releases</id>
        <name>Internal Releases</name>

 <url>http://localhost:8081/nexus/content/repositories/releases/</url>
    </repository>
    <snapshotRepository>
        <id>snapshots</id>
        <name>Internal Snapshots</name>

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

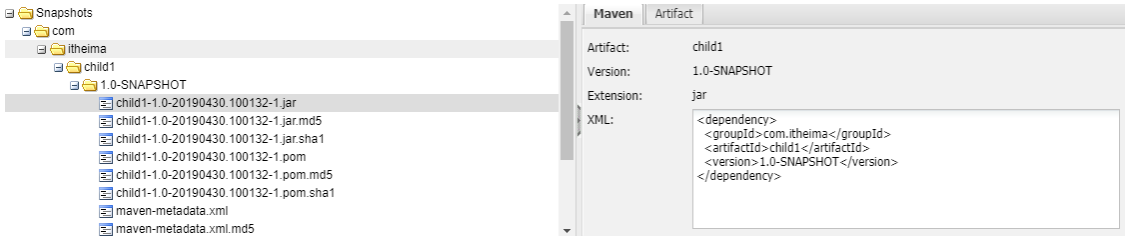
2. 在settings中配置认证信息

```
<servers>
    <!-- 服务器认证信息 -->
    <server>
        <!-- ID应与POM中对应 -->
        <id>releases</id>
        <username>admin</username>
        <password>admin123</password>
    </server>
    <server>
        <id>snapshots</id>
        <username>deployment</username>
        <password>deployment123</password>
    </server>
</servers>
```

3. 执行deploy操作或者`mvn deploy`命令



4. 查看上传资源



五、Maven常见插件

5.1 编译插件

- Maven 的核心仅仅定义了项目的抽象生命周期，具体的任务都是交由插件完成。
- 每个插件都能实现多个功能，每个功能就是一个插件目标。
- Maven 的生命周期与插件目标相互绑定，以完成某个具体的构建任务。
- 例如；compile 就是插件 maven-compiler-plugin 的一个插件目标。

```
<!-- java 编译插件 -->
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.0</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <encoding>UTF-8</encoding>
    </configuration>
</plugin>
```

5.2 Tomcat插件

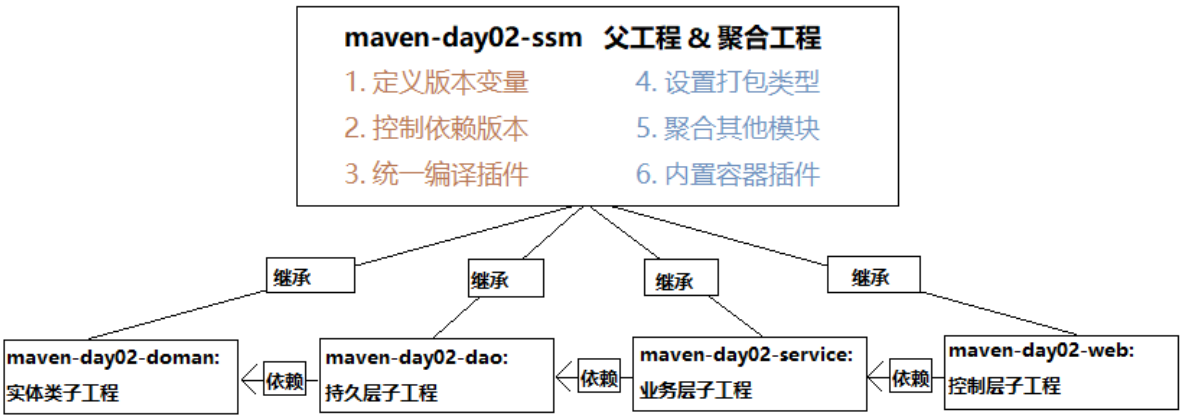
- 对于maven的web项目，我们可以发布到本地的tomca运行。
- 也可以配置maven提供的tomcat插件运行。

```
<!--配置tomcat插件-->
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <!-- tomcat 的端口号 -->
        <port>8080</port>
        <!-- 访问应用的路径 -->
        <path>/</path>
        <!-- URL按UTF-8进行编码，解决中文参数乱码 -->
        <uriEncoding>UTF-8</uriEncoding>
        <!-- tomcat名称 -->
        <server>tomcat7</server>
    </configuration>
</plugin>
```

六、构建SSM项目

- 使用maven的继承和聚合，搭建ssm整合项目。

6.2 项目架构



6.3 构建

6.3.1 创建ssm父工程

打包配置

```
<!--父工程packaging，必须设置为pom-->
<packaging>pom</packaging>
```

聚合模块

```
<!--聚合子工程-->
<modules>
    <module>maven-day02-domain</module>
    <module>maven-day02-dao</module>
    <module>maven-day02-service</module>
    <module>maven-day02-web</module>
</modules>
```

版本变量

```
<!--统一管理依赖包版本-->
<properties>
    <!--spring 版本-->
    <spring.version>5.0.2.RELEASE</spring.version>
    <!-- log4j日志包版本 -->
    <slf4j.version>1.7.7</slf4j.version>
    <log4j.version>1.2.17</log4j.version>
    <!-- jstl标签版本 -->
    <jstl.version>1.2</jstl.version>
    <!--servlet版本-->
    <servlet.version>2.5</servlet.version>
    <!--jsp版本-->
    <jsp.version>2.0</jsp.version>
```

```
    <mysql.version>5.1.30</mysql.version>
    <!-- mybatis-spring整合包版本 -->
    <mybatis.spring.version>1.3.1</mybatis.spring.version>
    <!--druid版本-->
    <druid.version>1.0.29</druid.version>
    <!-- aspectj版本号 -->
    <aspectj.version>1.6.12</aspectj.version>
</properties>
```

版本控制

```
<!--统一管理依赖包-->
<dependencyManagement>
    <dependencies>
        <!--spring依赖包-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!-- spring web包 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!-- spring mvc包 -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!--spring jdbc包-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!--spring tx包-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <!--aspectj包-->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>${aspectj.version}</version>
        </dependency>

        <!-- jstl标签类 -->
        <dependency>
            <groupId>jstl</groupId>
```

```
        </dependency>
        <!--servlet依赖-->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>${servlet.version}</version>
            <scope>provided</scope>
        </dependency>
        <!--jsp依赖-->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <version>${jsp.version}</version>
            <scope>provided</scope>
        </dependency>
        <!-- mybatis核心包 -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>${mybatis.version}</version>
        </dependency>
        <!-- mybatis-spring整合包 -->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>${mybatis.spring.version}</version>
        </dependency>
        <!-- mysql数据库依赖 -->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>${mysql.version}</version>
            <scope>runtime</scope>
        </dependency>
        <!--数据库连接池druid-->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>${druid.version}</version>
        </dependency>
        <!-- log4j包 -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>${log4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>${slf4j.version}</version>
```

插件配置

```
<!--统一配置java编译插件-->
<build>
    <plugins>
        <!-- java 编译插件 -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.2</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
                <encoding>UTF-8</encoding>
            </configuration>
        </plugin>
        <!--配置tomcat插件-->
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.1</version>
            <configuration>
                <!-- tomcat 的端口号 -->
                <port>8080</port>
                <!-- 访问应用的路径 -->
                <path>/ssm</path>
                <!-- URL按UTF-8进行编码，解决中文参数乱码 -->
                <uriEncoding>UTF-8</uriEncoding>
                <!-- tomcat名称 -->
                <server>tomcat7</server>
            </configuration>
        </plugin>
    </plugins>
</build>
```

6.3.2 创建domain子项目

pom.xml

```
<parent>
    <artifactId>maven-day02-ssm</artifactId>
    <groupId>com.itheima</groupId>
    <version>1.0-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<groupId>com.itheima</groupId>
<artifactId>maven-day02-domain</artifactId>
<version>1.0-SNAPSHOT</version>
```

```java
package com.itheima.maven.day02.domain;

import java.util.Date;

/**
 * 用户类.
 *
 * @author : Jason.lee
 * @version : 1.0
 * @date : 2019-5-3 14:04
 * @description : com.itheima.maven.day02.domain.User
 * @modified : -
 */
public class com.itheima.maven.day02.domain.User {
    private Integer id;
    private String username;
    private Date birthday;
    private String sex;
    private String address;

    @Override
    public String toString() {
        return "com.itheima.maven.day02.domain.User{" +
                "id=" + id +
                ", username='" + username + '\'' +
                ", birthday=" + birthday +
                ", sex='" + sex + '\'' +
                ", address='" + address + '\'' +
                '}';
    }
}
```

6.3.3 创建dao子项目

pom.xml

```xml
<parent>
    <artifactId>maven-day02-ssm</artifactId>
    <groupId>com.itheima</groupId>
    <version>1.0-SNAPSHOT</version>
</parent>
<modelVersion>4.0.0</modelVersion>

<groupId>com.itheima</groupId>
<artifactId>maven-day02-dao</artifactId>
<version>1.0-SNAPSHOT</version>

<!--配置依赖-->
<dependencies>
    <!--依赖maven-day02-domain-->
    <dependency>
        <groupId>com.itheima</groupId>
        <artifactId>maven-day02-domain</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
```

```
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
    <!--依赖数据库连接池包-->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid</artifactId>
    </dependency>
    <!--依赖mybatis框架包-->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
    </dependency>
</dependencies>
```

com.itheima.maven.day02.dao

```
package com.itheima.maven.day02.dao;

import com.itheima.maven.day02.domain.com.itheima.maven.day02.domain.User;

import java.util.List;

/**
 * 用户持久层.
 *
 * @author : Jason.lee
 * @version : 1.0
 * @date : 2019-5-3 14:15
 * @description : com.itheima.maven.day02.dao.UserDao
 * @modified : -
 */
public interface com.itheima.maven.day02.dao.UserDao {
    /**
     * 查询所有用户.
     *
     * @return 查询结果
     */
    List<com.itheima.maven.day02.domain.User> findAll();
}
```

6.3.4 创建service子项目

pom.xml

```
<!--配置依赖-->
<dependencies>
    <!--依赖maven-day01-08ssm-dao-->
    <dependency>
        <groupId>com.itheima</groupId>
        <artifactId>maven-day02-dao</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
```

```
        <artifactId>spring-context</artifactId>
    </dependency>
</dependencies>
```

com.itheima.maven.day02.service

```
package com.itheima.maven.day02.service;

import com.itheima.maven.day02.dao.com.itheima.maven.day02.dao.UserDao;
import com.itheima.maven.day02.domain.com.itheima.maven.day02.domain.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

/**
 * 用户业务中心.
 *
 * @author : Jason.lee
 * @version : 1.0
 * @date : 2019-5-3 14:20
 * @description : com.itheima.maven.day02.service.UserService
 * @modified : -
 */
@Service
public class com.itheima.maven.day02.service.UserService {

    @Autowired
    com.itheima.maven.day02.dao.UserDao userDao;

    public List<com.itheima.maven.day02.domain.User> findAll(){
        return userDao.findAll();
    }
}
```

6.3.5 创建web子项目

pom.xml

```
<!--配置依赖-->
<dependencies>
    <!--依赖maven-day01-09ssm-service-->
    <dependency>
        <groupId>com.itheima</groupId>
        <artifactId>maven-day02-service</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
    <!-- spring web包 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
    </dependency>
    <!-- spring mvc包 -->
    <dependency>
```

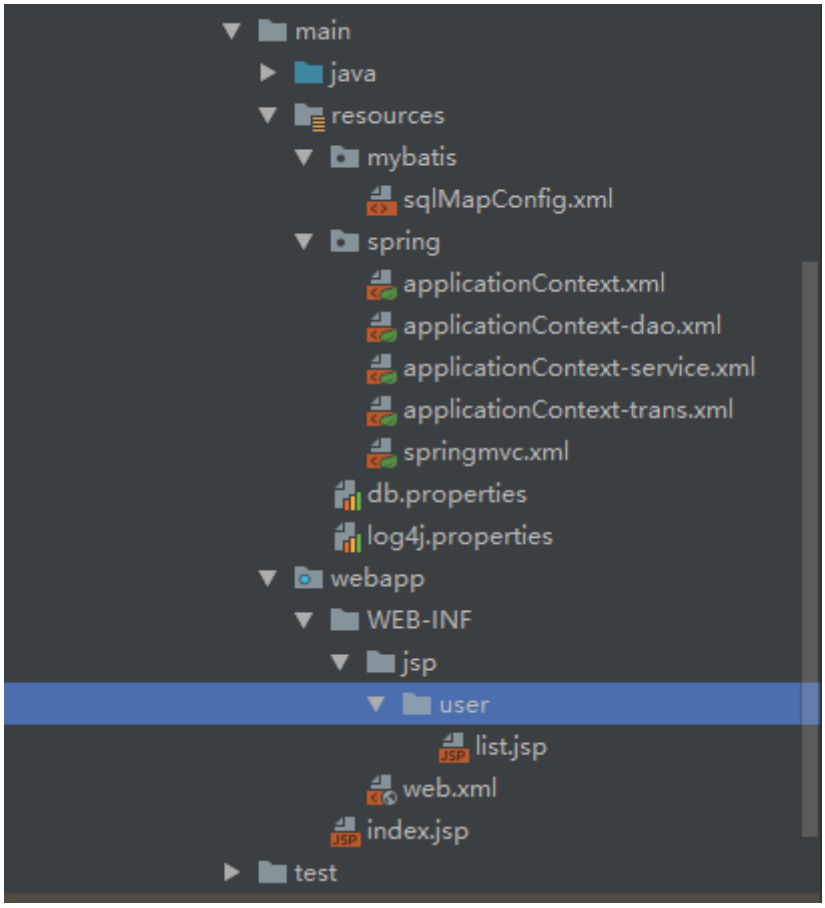
```
    <!--spring jdbc包-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
    </dependency>
    <!--spring tx包-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
    </dependency>
    <!--aspectj包-->
    <dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
    </dependency>
    <!-- jstl标签类 -->
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
    </dependency>
    <!--servlet依赖-->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <scope>provided</scope>
    </dependency>
    <!--jsp依赖-->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>jsp-api</artifactId>
        <scope>provided</scope>
    </dependency>
    <!-- log4j包 -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
    </dependency>
    <!-- mybatis-spring整合包 -->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
    </dependency>
</dependencies>
```

com.itheima.maven.day02.web

```
package com.itheima.maven.day02.web;
```

```
4  import
   com.itheima.maven.day02.service.com.itheima.maven.day02.service.UserService
   ;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.stereotype.Controller;
7  import org.springframework.ui.Model;
8  import org.springframework.web.bind.annotation.RequestMapping;
9
10 import java.util.List;
11
12 /**
13  * 用户控制器.
14  *
15  * @author : Jason.lee
16  * @version : 1.0
17  * @date : 2019-5-3 14:27
18  * @description : com.itheima.maven.day02.web.UserController
19  * @modified : -
20  */
21 @Controller
22 public class com.itheima.maven.day02.web.UserController {
23
24     // 注入service
25     @Autowired
26     private com.itheima.maven.day02.service.UserService userService;
27
28     @RequestMapping("list.do")
29     public String list(Model model){
30         List<com.itheima.maven.day02.domain.User> all =
   userService.findAll();
31         model.addAttribute("list",all);
32         return "user/list";
33     }
34 }
```

图示



注意: 数据库连接参数url对版本有要求, 往往需要添加如下参数:

```
db.url=jdbc:mysql:///mybatisdb?serverTimezone=UTC&useSSL=false
```

6.4 启动

- 在ssm父工程中运行tomcat7插件
  - 双击 tomcat7:run 菜单
  - mvn tomcat7:run 命令

6.5 访问

1	王五	6666666
10	张三	北京市
16	张小明	河南郑州
22	陈小明	河南郑州
24	张三丰	河南郑州
25	陈小明	河南郑州
26	王五	
76	第一个用户	本地人
77	第一个用户	本地人
79	666	数据库本地人
80	第一个用户	本地人
81	第一个用户	本地人