jsp-api导入：scope只能是provided，否则会出现冲突。

```
<dependency>
                <groupId>javax.servlet.jsp</groupId>
                <artifactId>jsp-api</artifactId>
                <version>2.1.3-b06</version>
                <scope>provided</scope>
  </dependency>
```

1，导入工程

手动创建maven工程需要将其放在工作路径下，而且还需要使用maven方式进行导入。

大多数情况下开发过程中不需要进行安装命令的执行。

2，依赖的高级

（1）依赖的传递性

①传递性的好处：可以传递的依赖不必在每个慕课中都重复声明，在“最下面”的工程中依赖

一次即可。

②test和provided只能在本maven工程引用，是不能被传递的。只有compile的才能进行传

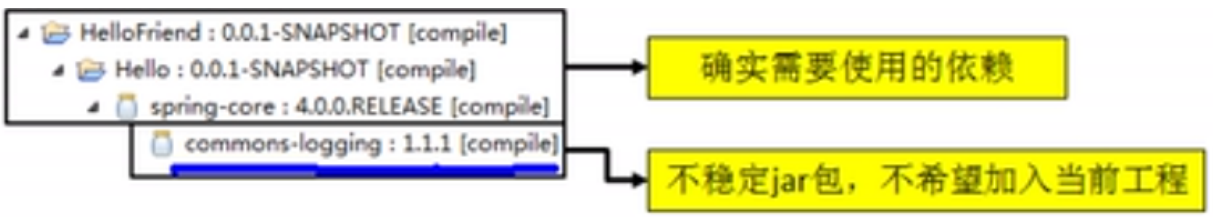
递。

③注意：非compile范围的依赖不能传递，所以在各个工程模块中，如果有需要就得重复声

明依赖。

（2）依赖的排出

①需要设置依赖排出的场合



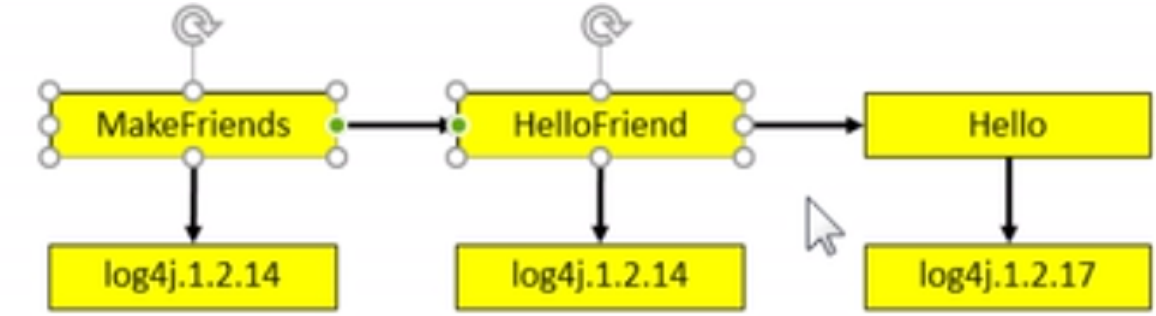
②依赖排除的设置

```
<dependency>
   <exclusions>
          <exclusion>
               <groupId>g值 </groupId>
               <artifactId>a值</artifactId>
          </exclusion>
      </exclusions>
 </dependency>
```

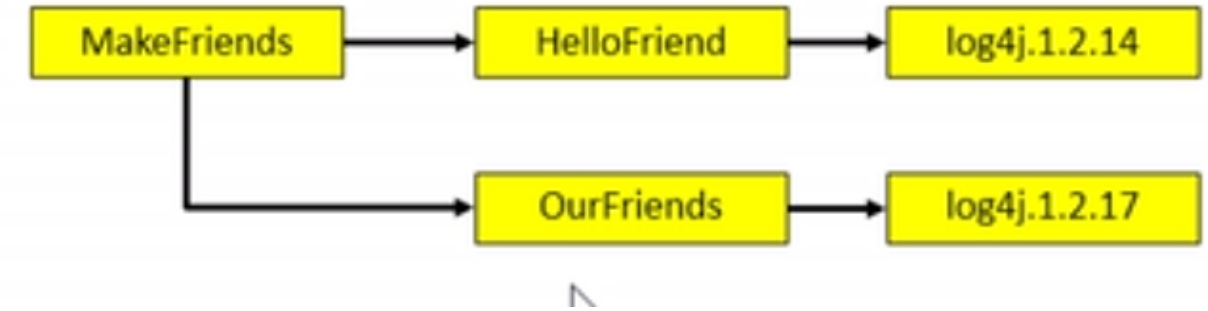
3，依赖的原则

（1）作用：解决模块工程质检的jar包冲突问题。

（2）情景设定1：路径最短者优先原则。

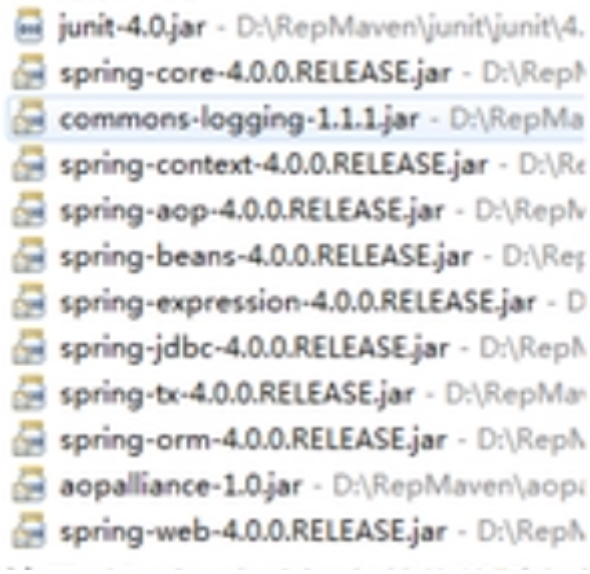


（3）情景设定2：验证路径相同时，先声明者优先。先声明的是dependency标签的声明顺序。



4，统一管理依赖的版本

（1）情景举例：



这里对spring各个jar的依赖版本都是4.0.0.如果需要统一升级为4.1.1，怎么办？手动注意修改不可靠。

（2）建议的配置方式

①使用properties标签内使用标签统一声明版本号。

```
<properties>
    <atguigu.spring.version>4.0.0.RELEASE</atguigu.spring.version>
</properties>
```

②在需要统一版本的位置，使用${自定义标签名}引用声明的版本号。

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

（3）其实properties标签配合自定义标签声明数据的配置并不是只能用于声明版本号，也可以用来声明其它的东西。凡是需要统一声明后在引用的场合都可以使用。

5，maven的继承（注意：配置继承后执行安装命令时，要先安装父工程）

（1）现状：test范围内的依赖不能传递，所以必然会分散在各个模块工程中，很有可能会造成各个版本不一致。

（2）需求：统一管理各个模块工程中对Junit依赖的版本。

（3）解决思路：将Junit依赖统一提取到父工程中，在子工程中声明Junit依赖时不指定版本，以父工程中统一设定为准，同时也便于修改。

（4）操作步骤：

①创建一个maven工程作为父工程，注意：打包为pom格式。

```
<groupId>com.atguigu.maven</groupId>
<artifactId>Parent</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>pom</packaging>
```

②在子工程中声明对父工程中的引用。

```
<!-- 子工程中声明父工程 -->
<parent>
    <groupId>com.atguigu.maven</groupId>
    <artifactId>Parent</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <!-- 以当前文件为基准的父工程pom.xml文件的相对路径 -->
    <relativePath>../Parent/pom.xml</relativePath>
</parent>
```

③将子工程的坐标中与父工程坐标中重复的内容删除。

```
<groupId>com.atguigu.maven</groupId>
```

GroupId is duplicate of parent groupId

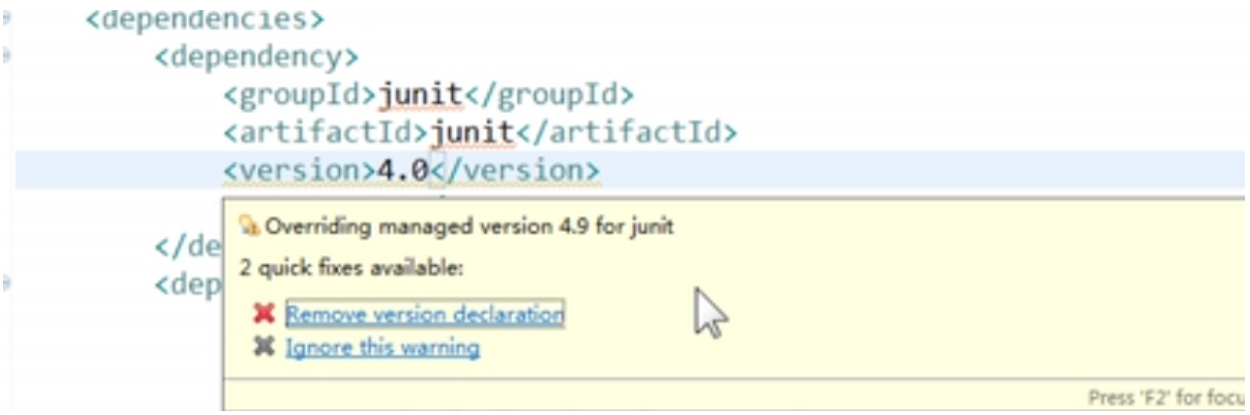
1 quick fix available:

Remove groupId declaration

④在父工程中统一Junit依赖。

```
<!-- 配置依赖的管理 -->
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.0</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

⑤在子工程中删除Junit依赖的版本号部分。



6，聚合

（1）作用：一键安装各个模块工程。

（2）配置方式：在一个总的聚合工程中国配置各个参与聚合的模块。（其中的内容位置没有特定的要求）

```
<!-- 配置聚合 -->
<modules>
    <!-- 指定各个子工程的相对路径 -->
    <module>../Hello</module>
    <module>../HelloFriend</module>
    <module>../MakeFriends</module>
</modules>
```

（3）使用方式：在聚合工程的pom.xml上点右键--->run as ---->maven install

7，maven的web工程自动部署

（1）将项目放在Tomcat的webapps下才可以运行。

（2）在pom中配置的配置文件：

```
<!-- 配置当前工程构建构成中的特殊设置 -->
    <build>
        <!-- 当前项目最终的名字 -->
        <finalName>hello</finalName>
        <!-- 配置构建构成中需要使用的插件 -->
        <plugins>
            <!-- 配置插件的坐标 -->
            <plugin>
                <!-- cargo是一家专门从事启动servlet容器的组织 -->
                <groupId>org.codehaus.cargo</groupId>
                <artifactId>cargo-maven2-plugin</artifactId>
                <version>1.2.3</version>
                <configuration>
                    <!-- 配置当前系统中容器的位置 -->
                    <container>
```

```
                        <containerId>tomcat9x</containerId>
                        <home>C:\Users\Anly\Desktop\apache-tomcat-
9.0.20</home>
                        <!-- 如果Tomcat端口默认值8080则不必设置该属性 -->
                        <!-- <properties>
                            <cargo.servlet.port>8080</cargo.servlet.port>
                        </properties> -->
                    </container>
                </configuration>
                <!-- 配置插件在什么情况下运行 -->
                <executions>
                    <execution>
                        <id>cargo-run</id>
                        <!-- 生命周期的阶段 -->
                        <phase>install</phase>
                        <goals>
                            <!-- 插件的目标 -->
                            <goal>run</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

8，maven查找信息的网址

（1）我们可以到：http://mvnrepository.com/搜索需要的jar包的依赖信息。