



小D课堂 愿景: "让编程不在难学, 让技术与生活更加有趣" 更多教程请访问

[xdclass.net](http://xdclass.net)

---

## 第一章 maven的概述和课程介绍

### 第1集 maven概述和课程内容介绍

简介: 介绍maven的基本概述和课程内容章节



小D课堂 愿景: "让编程不在难学, 让技术与生活更加有趣" 更多教程请访问

[xdclass.net](http://xdclass.net)

---

## 第二章 0基础快速入门maven

### 第1集 如何配置maven环境变量

简介: 介绍在windows操作系统的maven环境变量配置方法

- 下载maven
- 配置环境变量
  - windows操作系统
    - MAVEN\_HOME : maven的安装目录
    - Path: ;%MAVEN\_HOME%\bin
- 重启电脑
- 检测是否安装成功
  - mvn --version

### 第2集 maven仓库的秘密

简介: maven仓库的介绍, 手把手教你如何设置本地仓库和远程仓库

- maven仓库的分类

- 本地仓库
- 远程仓库
  - 中央仓库
  - 私服
  - 其他公共库

- 本地仓库详解

- 本地仓库，顾名思义，就是Maven在本地存储的地方。
- maven的本地仓库，在安装maven后并不会创建，它是在第一次执行maven命令的时候才被创建
- maven本地仓库的默认位置：无论是Windows还是Linux，在用户的目录下都有一个.m2/repository/的仓库目录，这就是Maven仓库的默认位置

```
<settings>
  <localRepository>目录</localRepository>
</settings>
```

- 远程仓库-中央仓库详解

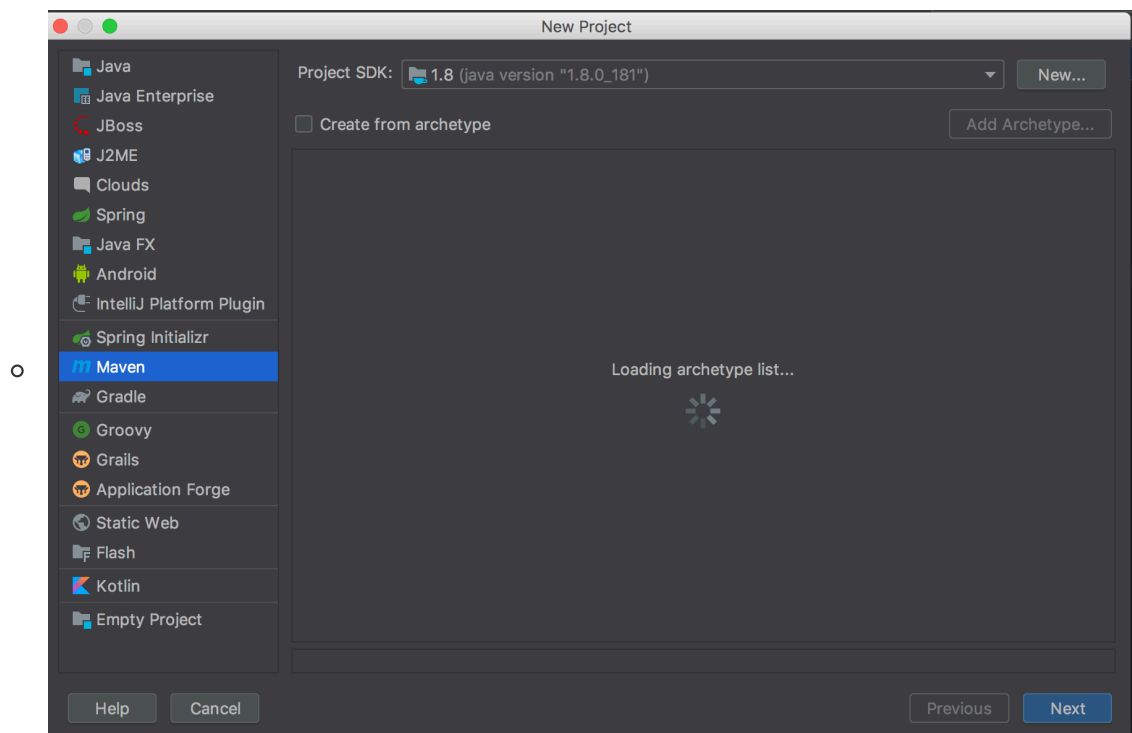
- 最核心的中央仓库开始，中央仓库是默认的远程仓库，maven在安装的时候，自带的就是中央仓库的配置，可以通过修改setting.xml文件来修改默认的中央仓库地址
- 中央仓库包含了绝大多数流行的开源Java构件，以及源码、作者信息、SCM、信息、许可证信息等。一般来说，简单的Java项目依赖的构件都可以在这里下载到

```
<repositories>
  <repository>
    <id>central</id>
    <name>Central Repository</name>
    <url>http://repo.maven.apache.org/maven2</url>
    <layout>default</layout>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
```

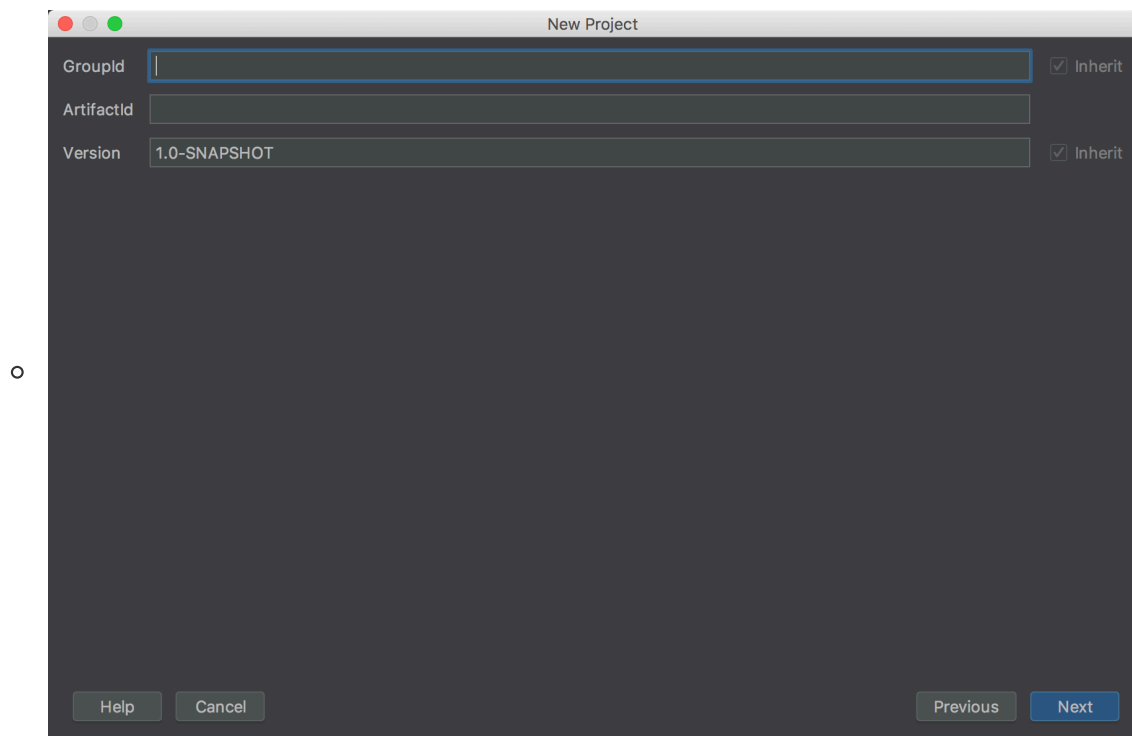
## 第3集 如何用idea建立一个简单的maven项目

简介：如何用idea建立一个简单的maven项目

- 点击创建项目，然后选择maven



- 填写项目信息



- 一直填写下去即可

## 第4集 maven项目的标准目录结构

简介：介绍maven项目目录结构和规范

- src
  - main
    - java -java源代码文件

- resources - 资源库
- webapp
  - WEB-INF
    - index.jsp
  - Css、js
- Bin - 脚本库
- config - 配置文件
- Filters - 资源过滤库
- test
  - java - java测试源代码文件
  - resources - 测试资源库
  - filters - 测试资源过滤库
- target - 存放项目构建后的文件和目录，比如jar包,war包,编译的class文件等

## 第5集 maven核心pom文件

简介：介绍maven的pom文件，分析它重要的组成部分

- 什么是pom
  - pom代表项目对象模型，它是Maven中工作的基本组成单位。它是一个XML文件，始终保存在项目的基本目录中的pom.xml文件中。pom包含的对象是使用maven来构建的，pom.xml文件包含了项目的各种配置信息，需要特别注意，每个项目都只有一个pom.xml文件。
- 项目配置信息
  - **project**: 工程的根标签
  - **modelVersion**: pom模型版本，maven2和3只能为4.0.0
  - **groupId**: 这是工程组的标识。它在一个组织或者项目中通常是唯一的。例如，一个银行组织 com.companyname.project-group 拥有所有的和银行相关的项目。
  - **artifactId**: 这是工程的标识。它通常是工程的名称。例如，消费者银行。groupId 和 artifactId 一起定义了 artifact 在仓库中的位置
  - **version**: 这是工程的版本号。在 artifact 的仓库中，它用来区分不同的版本
  - **packaging**: 定义 Maven 项目的打包方式，有 JAR、WAR 和 EAR 三种格式
- 最小pom

```
<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.xdclass</groupId>
  <artifactId>demo</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

- super pom

- 父（Super）POM是 Maven 默认的 POM。所有的 POM 都继承自一个父 POM（无论是否显式定义了这个父 POM）。父 POM 包含了一些可以被继承的默认设置。因此，当 Maven 发现需要下载 POM 中的 依赖时，它会到 Super POM 中配置的默认仓库。
- 使用以下命令来查看 Super POM 默认配置：

```
mvn help:effective-pom
```

- 依赖配置信息

1. dependencies

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
  </dependency>
</dependencies>
```

2. parent

```
<parent>
  <groupId>xd.class</groupId>
  <artifactId>demo-parent</artifactId>
  <relativePath>/</relativePath>
  <version>1.0</version>
</parent>
```

- groupId: 父项目的组Id标识符
- artifactId: 父项目的唯一标识符
- relativePath: Maven首先在当前项目中找父项目的pom，然后在文件系统的这个位置（relativePath），然后在本地仓库，再在远程仓库找。
- version: 父项目的版本

3. modules

- 有些maven项目会做成多模块的，这个标签用于指定当前项目所包含的所有模块。之后对这个项目进行的maven操作，会让所有子模块也进行相同操作。

```
<modules>
  <module>com-a</module>
  <module>com-b</module>
  <module>com-c</module>
</modules>
```

#### 4. properties

- 用于定义pom常量

```
<properties>
  <java.version>1.7</java.version>
</properties>
```

上面这个常量可以在pom文件的任意地方通过`${Java.version}`来引用

#### 5. dependencyManagement

- 应用场景
  - 当我们的项目模块很多的时候，我们依赖包的管理就会出现很多问题，为了项目的正确运行，必须让所有的子项目使用依赖项的同一版本，确保应用的各个项目的依赖项和版本一致，才能保证测试的和发布的是相同的结果。
- 使用的好处
  - 在父模块中定义后，子模块不会直接使用对应依赖，但是在使用相同依赖的时候可以不加版本号，这样的好处是，可以避免在每个使用的子项目中都声明一个版本号，这样想升级或者切换到另一个版本时，只需要在父类容器里更新，不需要任何一个子项目的修改

父项目：

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.2.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

子项目1：

```
<dependency>
  <groupId>junit</groupId>
```

```
        <artifactId>junit</artifactId>
    </dependency>

    子项目2:
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
    </dependency>

    子项目3:
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>5.0</version>
    </dependency>
```

- 和dependencies的区别
  - dependencies即使在子项目中不写该依赖项，那么子项目仍然会从父项目中继承该依赖项（全部继承）
  - dependencyManagement里只是声明依赖，并不实现引入，因此子项目需要显示的声明需要用的依赖。如果不在子项目中声明依赖，是不会从父项目中继承下来的；只有在子项目中写了该依赖项，并且没有指定具体版本，才会从父项目中继承该项，并且version和scope都读取自父pom;另外如果子项目中指定了版本号，那么会使用子项目中指定的jar版本。

## 第6集 maven的生命周期

简介：介绍和分析maven的整个生命周期

- 什么是生命周期
  - Maven的生命周期就是对所有的构建过程进行抽象和统一。包含了项目的清理、初始化、编译、测试、打包、集成测试、验证、部署和站点生成等几乎所有的构建步骤。
- maven的三个构建生命周期
  - clean
    - pre-clean 执行一些清理前需要完成的工作
    - clean 清理上一次构建生成的文件
    - post-clean 执行一些清理后需要完成的工作
  - default
    - validate: 验证工程是否正确
    - compile: 编译项目的源代码
    - test: 使用合适的单元测试框架来测试已编译的源代码。
    - package: 把已编译的代码打包成可以发布的格式，比如jar或者war
    - verify: 运行所有检查，验证包是否有效

- install: 安装到maven本地仓库
- deploy: 部署到远程的仓库, 使得其他开发者或者工程可以共享
- Site

## 第7集 常用的maven基本命令

简介: 介绍maven常用的基本命令, 并对个别命令重点剖析

- 常用命令
  - mvn validate 验证项目是否正确
  - mvn package maven打包
  - mvn generate-sources 生成源代码
  - mvn compile 编译
  - mvn test-compile 编译测试代码
  - mvn test 运行测试
  - mvn verify 运行检查
  - mvn clean 清理项目
  - mvn install 安装项目到本地仓库
  - mvn deploy 发布项目到远程仓库
  - mvn dependency:tree 显示Maven依赖树
  - mvn dependency:list 显示Maven依赖列表
- 常用参数
  - -D 指定参数, 如 -Dmaven.test.skip=true 跳过单元测试;
  - -P 指定 Profile 配置, 可以用于区分环境;
- web相关命令
  - mvn tomcat:run 启动tomcat
  - mvn jetty:run 启动jetty
  - mvn tomcat:deploy 运行打包部署



小D课堂 愿景: "让编程不在难学, 让技术与生活更加有趣" 更多教程请访问  
[xdclass.net](http://xdclass.net)

---

## 第三章 maven应用的高级实战之路

### 第1集 如何添加项目所需要的jar包

简介: 教你如何在maven使用你项目当中需要的依赖包

- 原理
  - 在本地, 指定一个文件夹, 便是maven的仓库, maven会从远程的中央仓库



中下载你需要的jar资源到你本地，然后通过maven关联，将jar包依赖到你的项目中，避免了你需要将jar包拷贝到lib中，并通过classpath引入这些jar包的工作。

- 步骤
  - 打开仓库网站
  - 选择你要jar包的信息和版本
  - 填写依赖信息到pom文件
  - 下载到本地仓库
  - 项目使用
- 实战

## 第2集 如何使用maven运行单元测试

简介：手把手教你怎么用maven运行一个单元测试

## 第3集 如何建立一个web应用

简介：从0到1用idea建立一个web项目

## 第4集 如何导入第三方jar包到本地仓库

简介：教你如何导入第三方jar包到本地仓库

- 进入cmd命令界面
- 输入指令如下：  
`mvn install:install-file -Dfile=xxxxxxx -DgroupId=com.alibaba -DartifactId=fastjson -Dversion=1.0 -Dpackaging=jar -DgeneratePom=true -DcreateChecksum=true`
- 参数说明
  - Dfile为jar包文件路径
  - DgroupId一般为jar开发组织的名称，也是坐标groupId
  - DartifactId一般为jar名称，也是坐标 artifactId
  - Dversion是版本号
  - Dpackaging是打包类型

## 第5集 常用的maven插件

简介：告诉你常用的maven插件，并且介绍怎么去更好的去用maven插件

- maven官方插件列表
  - groupId为org.apache.maven.plugins

- <http://maven.apache.org/plugins/index.html>
- 两种方式调用maven插件
  - 将插件目标与生命周期阶段绑定，例如maven默认将maven-compiler-plugin的compile与maven生命周期的compile阶段绑定。
  - 直接在命令行显示指定要执行的插件目标，例如mvn archetype:generate就表示调用maven-archetype-plugin的generate目标。
- 常用的maven插件
  - maven-antrun-plugin
    - maven-antrun-plugin能让用户在Maven项目中运行Ant任务。用户可以直接在该插件的配置以Ant的方式编写Target，然后交给该插件的run目标去执行。
  - maven-archetype-plugin
    - Archetype指项目的骨架，Maven初学者最开始执行的Maven命令可能就是mvn archetype:generate，这实际上就是让maven-archetype-plugin生成一个很简单的项目骨架，帮助开发者快速上手。
  - maven-assembly-plugin
    - maven-assembly-plugin的用途是制作项目分发包，该分发包可能包含了项目的可执行文件、源代码、readme、平台脚本等等。
  - maven-dependency-plugin
    - maven-dependency-plugin最大的用途是帮助分析项目依赖
    - dependency:list能够列出项目最终解析到的依赖列表
    - dependency:tree能进一步的描绘项目依赖树
  - maven-enforcer-plugin
    - maven-enforcer-plugin能够允许你创建一系列规则强制大家遵守，包括设定Java版本、设定Maven版本、禁止某些依赖、禁止SNAPSHOT依赖。
  - maven-help-plugin
    - maven-help-plugin是一个小巧的辅助工具。
    - 最简单的help:system可以打印所有可用的环境变量和Java系统属性。
  - maven-release-plugin
    - maven-release-plugin的用途是帮助自动化项目版本发布，它依赖于POM中的SCM信息。



小D课堂

愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问

[xdclass.net](http://xdclass.net)

---

## 第四章 maven的进阶之路

### 第1集 搭建你的第一个maven私人服务器

简介：从0到1手把手搭建属于自己的maven私人服务器

- 背景
  - 回顾下maven的构建流程，如果没有私服，我们所需的所有jar包都需要通过maven的中央仓库或者第三方的maven仓库下载到本地，当一个公司或者一个团队所有人都重复的从maven仓库下载jar包，这样就加大了中央仓库的负载和浪费了外网的带宽，如果网速慢的话还会影响项目的进程。
- 简介
  - 私服是在局域网的一种特殊的远程仓库，目的是代理远程仓库及部署第三方构件。有了私服之后，当Maven需要下载jar包时，先请求私服，私服上如果存在则下载到本地仓库。否则，私服直接请求外部的远程仓库，将jar包下载到私服，再提供给本地仓库下载。
- 安装
  - 我们可以使用专门的Maven仓库管理软件来搭建私服，这里我们使用Nexus
  - 下载地址：<https://help.sonatype.com/repomanager2/download>
  - Nexus专业版是需要付费的，这里我们下载开源版Nexus OSS，最新的是OSS3.x，我们选择稳定的版本2.x。
- 启动
  - 以管理员身份打开cmd，进入到bin目录，先执行nexus install命令，再执行nexus start。
  - 打开浏览器，访问<http://localhost:8081/nexus>
  - 点击右上角Log in，使用用户名：admin，密码：admin123登录

## 第2集 Nexus私服的秘密花园

简介：介绍nexus服务器预置的仓库

- 类型介绍
  - hosted：是本地仓库，用户可以把自己的一些jar包，发布到hosted中，比如公司的第二方库
  - proxy，代理仓库，它们被用来代理远程的公共仓库，如maven中央仓库。不允许用户自己上传jar包，只能从中央仓库下载
  - group，仓库组，用来合并多个hosted/proxy仓库，当你的项目希望在多个repository使用资源时就不需要多次引用了，只需要引用一个group即可
  - virtual，虚拟仓库基本废弃了。
- 预置仓库
  - Central：该仓库代理Maven中央仓库，其策略为Release，因此只会下载和缓存中央仓库中的发布版本构件。
  - Releases：这是一个策略为Release的宿主类型仓库，用来部署正式发布版本构件
  - Snapshots：这是一个策略为Snapshot的宿主类型仓库，用来部署开发版本构件。
  - 3rd party：这是一个策略为Release的宿主类型仓库，用来部署无法从maven中央仓库获得的第三方发布版本构件，比如IBM或者oracle的一些jar包（比如classe12.jar），由于受到商业版权的限制，不允许在中央仓库出

- 现，如果让这些包在私服上进行管理，就需要第三方的仓库。
- **Public Repositories**: 一个组合仓库

## 第3集 在nexus建立你的第一个仓库

简介：从0到1手把手建立专属于你的第一个仓库

- 建库，Add --> Hosted Repository
- 填写仓库信息
  - Repository ID 仓库编号
  - Repository NAME 仓库名称
  - Repository Type 仓库类型
  - Repository Policy 仓库策略
  - Default Local Storage Location 仓库路径
  - Deployment Policy 发布策略
- 然后选择Public Repositories, 打开configuration选项卡，将自己创建的仓库添加到group

## 第4集 如何将项目发布到maven私服

简介：手把手将项目发布到maven私服

## 第5集 maven私服发布和使用之实战

简介：结合例子从0到1发布和使用maven私服jar包

## 第6集 maven snapshot的秘密

简介：结合案例介绍和使用snapshot

- 产生背景
  - 假设一个团队工作，其中有个项目叫做data-use，同时他们使用数据服务工程（data-service.jar:1.0）。

现在负责数据服务的团队可能正在进行修 bug 或者更新迭代，每次发布都会发布工程到远程仓库中。

现在如果数据服务团队每天上传新的版本，那么就会有下面的问题：

  - 每次数据服务团队发布了一版更新的代码时，都要告诉应用接口团队。
  - 应用接口团队需要定期更新他们的 pom.xml 来得到更新的版本
- 什么是快照

- 快照是一个特殊的版本，它表示当前开发的一个副本。与常规版本不同，**Maven** 为每一次构建从远程仓库中检出一份新的快照版本。
- 快照 vs 版本
  - 对于版本，**Maven** 一旦下载了指定的版本（例如 `data-service:1.0`），它不会尝试从仓库里再次下载一个新的 `1.0` 版本。想要下载新的代码，数据服务版本需要被升级到 `1.1`。
  - 对于快照，每次用户接口团队构建他们的项目时，**Maven** 将自动获取最新的快照（`data-service:1.0-SNAPSHOT`）。
- maven快照延伸
  - updatePolicy
    - `always` 每次都去远程仓库查看是否有更新
    - `daily` 每天第一次的时候查看是否有更新
    - `interval` 允许设置一个分钟为单位的间隔时间，在这个间隔时间内只会去远程仓库中查找一次
    - `never` 从不会

## 第7集 maven的依赖管理

简介：依赖是maven最为用户熟知的特性之一，单个项目的依赖管理并不难，但是要管理几个或者几十个模块的时，那这个依赖应该怎么管理

- 依赖的传递性
  - 传递性依赖是在**maven2**中添加的新特征，这个特征的作用就是你不需要考虑你依赖的库文件所需要依赖的库文件，能够将依赖模块的依赖自动的引入。
- 依赖的作用范围
  - compile
    - 这是默认范围，编译依赖对项目所有的classpath都可用。此外，编译依赖会传递到依赖的项目
  - provided
    - 表示该依赖项将由JDK或者运行容器在运行时提供，也就是说由**Maven**提供的该依赖项我们只有在编译和测试时才会用到，而在运行时将由JDK或者运行容器提供。
  - runtime
    - 表明编译时不需要依赖，而只在运行时依赖
  - test
    - 只在编译测试代码和运行测试的时候需要，应用的正常运行不需要此类依赖。
  - system
    - 系统范围与**provided**类似，不过你必须显式指定一个本地系统路径的JAR，此类依赖应该一直有效，**Maven**也不会去仓库中寻找它。

<project>

```

...
<dependencies>
  <dependency>
    <groupId>sun.jdk</groupId>
    <artifactId>tools</artifactId>
    <version>1.5.0</version>
    <scope>system</scope>

    <systemPath>${java.home}/../lib/tools.jar</systemPath>

  </dependency>
</dependencies>
...
</project>

```

- import

- 范围只适用于部分。表明指定的POM必须使用部分的依赖。因为依赖已经被替换，所以使用import范围的依赖并不影响依赖传递。

- 依赖的两大原则

- 路径近者优先

```

A > B > C-1.0
A > C-2.0

```

- 第一声明优先

```

A > B > D-1.0
A > C > D-2.0

```

- 依赖的管理

- 依赖排除

- 任何可传递的依赖都可以通过 "exclusion" 元素被排除在外。举例说明，A 依赖 B，B 依赖 C，因此 A 可以标记 C 为 "被排除的"

- 依赖可选

- 任何可传递的依赖可以被标记为可选的，通过使用 "optional" 元素。例如：A 依赖 B，B 依赖 C。因此，B 可以标记 C 为可选的，这样 A 就可以不再使用 C。

## 第8集 如何解决jar包冲突

简介：当出现jar包冲突时，我们应该如何快速定位和处理jar包冲突问题

- 命令: mvn dependency:tree -Dverbose

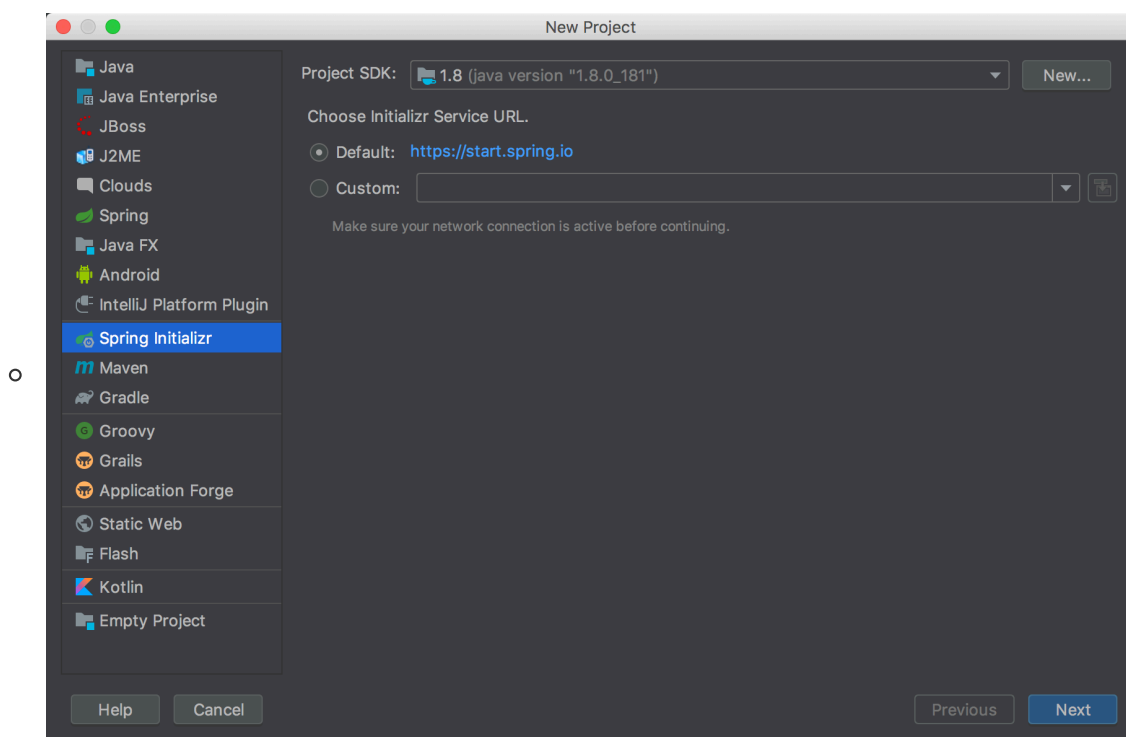


## 第五章 maven之项目实战进阶篇

### 第1集 实战进阶之使用idea建立Spring Boot项目

简介：手把手教大家怎么建立一个spring boot项目

- 下载企业版idea
- 选择Spring initializr，点击Next



- 填写项目信息

New Project

**Project Metadata**

Group:

Artifact:

Type:

Language:

Packaging:

Java Version:

Version:

Name:

Description:

Package:

Help Cancel Previous Next

- 选择依赖信息
  - 点击Web，勾选Spring Web Starter

New Project

**Dependencies**

Developer Tools

**Web**

Template Engines

Security

SQL

NoSQL

Messaging

I/O

Ops

Spring Cloud

Spring Cloud Security

Spring Cloud Tools

Spring Cloud Config

Spring Cloud Discovery

Spring Cloud Routing

Spring Cloud Circuit Breaker

Spring Cloud Tracing

Spring Cloud Messaging

Pivotal Cloud Foundry

Amazon Web Services

Microsoft Azure

☒ Spring Web Starter

☐ Spring Reactive Web

☐ Rest Repositories

☐ Spring Session

☐ Rest Repositories HAL Browser

☐ Spring HATEOAS

☐ Spring Web Services

☐ Jersey

☐ Spring REST Docs

☐ Vaadin

☐ Apache CXF

☐ Ratpack

☐ Spring Mobile

**Selected Dependencies**

**Web**

Spring Web Starter

**SQL**

MySQL Driver

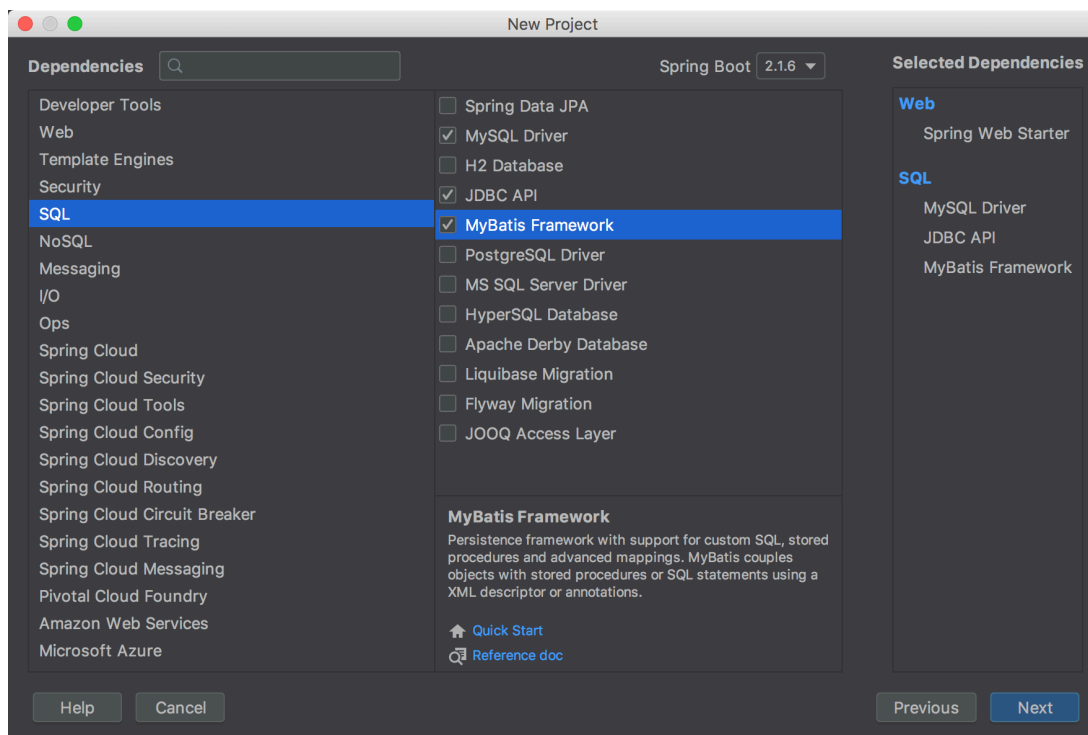
JDBC API

MyBatis Framework

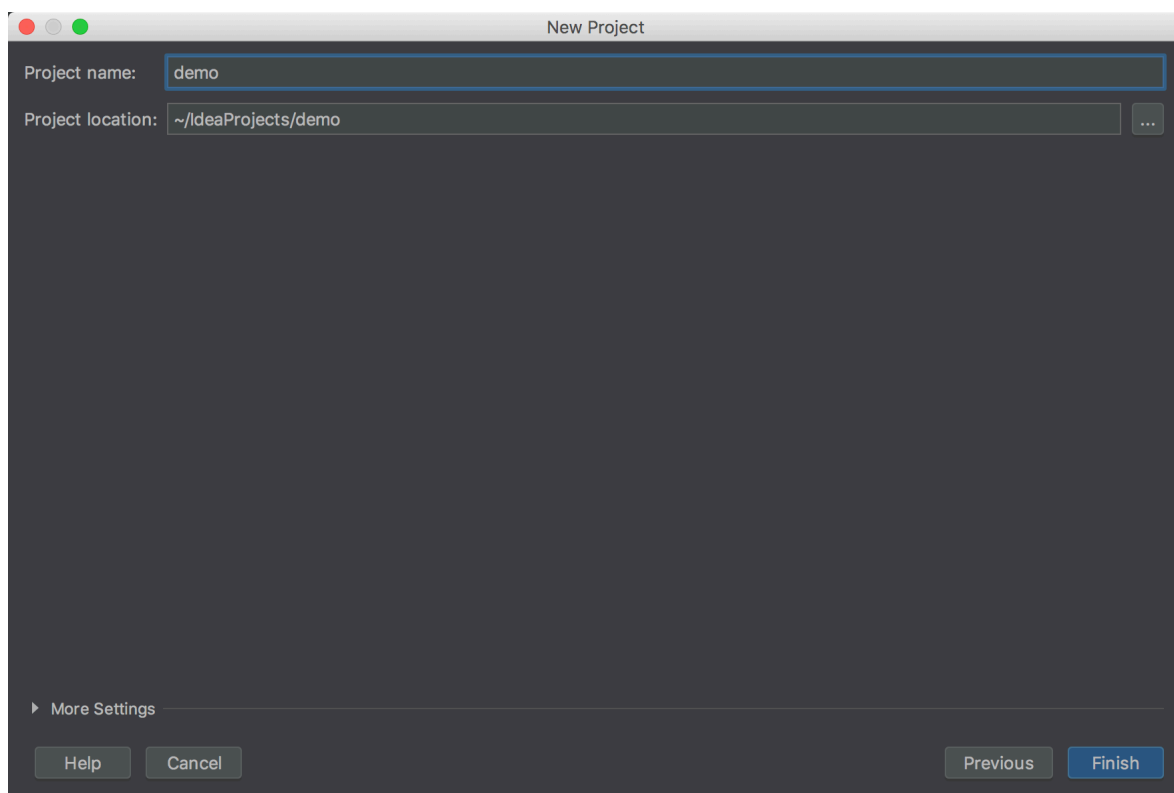
Help Cancel Previous Next

- 选择SQL，勾选Mysql Driver,JDBC API,MyBatis Framework





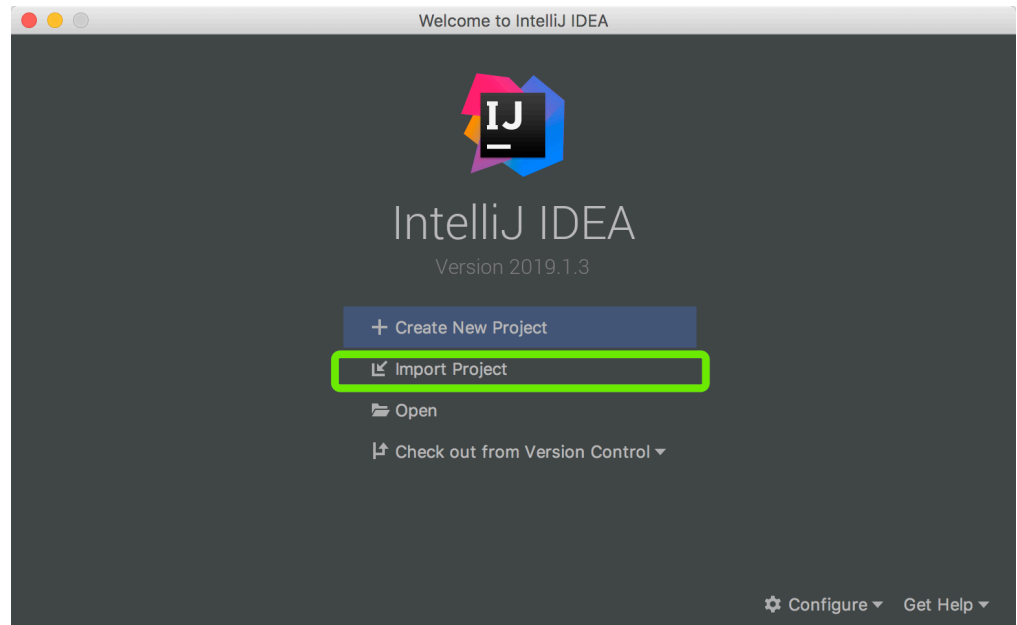
- 填写项目名点击finish



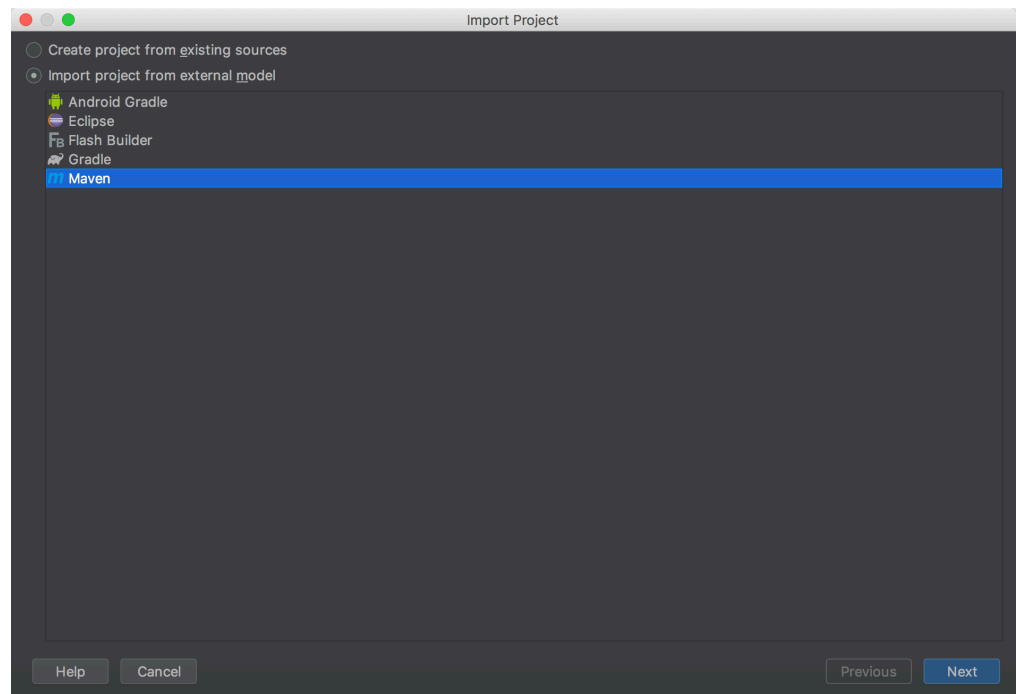
## 第2集 实战进阶之使用idea导入maven项目

简介：手把手教怎么导入一个maven项目，包括本地的项目和git上项目的导入

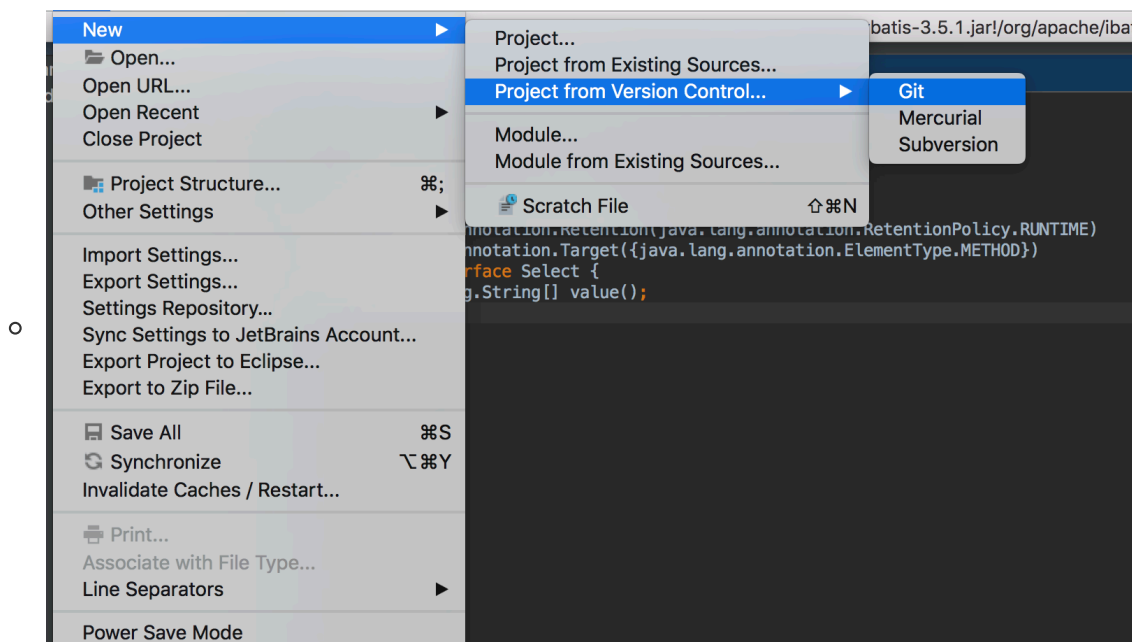
- 本地maven项目导入
  - import project



- 选择完项目以后勾选Maven



- 导入git的maven项目



### 第3集 Spring Boot整合Mysql


简介：手把手整maven+spring boot+mysql

### 第4集 Spring Boot整合Mybatis和Mysql

简介：手把手整maven+spring boot+mybatis+mysql

### 第5集 Spring Boot+Mybatis+Mysql 的增删改查

简介：结合例子，通过表的增删改查，熟悉maven

 小D课堂 愿景："让编程不在难学，让技术与生活更加有趣" 更多教程请访问 [xdclass.net](http://xdclass.net)

## 第六章 课程总结

### 第1集 maven总结和学习方法

简介：总结maven整个课程的内容，分享maven的学习方法

小D课堂，愿景：让编程不在难学，让技术与生活更加有趣

相信我们，这个是可以让你学习更加轻松的平台，里面的课程绝对会让你技术不断提升

欢迎加Louis讲师的微信：Mr-Datougege

我们官方网站：<https://xdclass.net>

千人IT技术交流QQ群： 718617859

**重点来啦：免费赠送你干货文档大集合，包含前端，后端，测试，大数据，运维主流技术文档（持续更新）**

<https://mp.weixin.qq.com/s/qYnjcDYGFDQorWmSfE7lpQ>