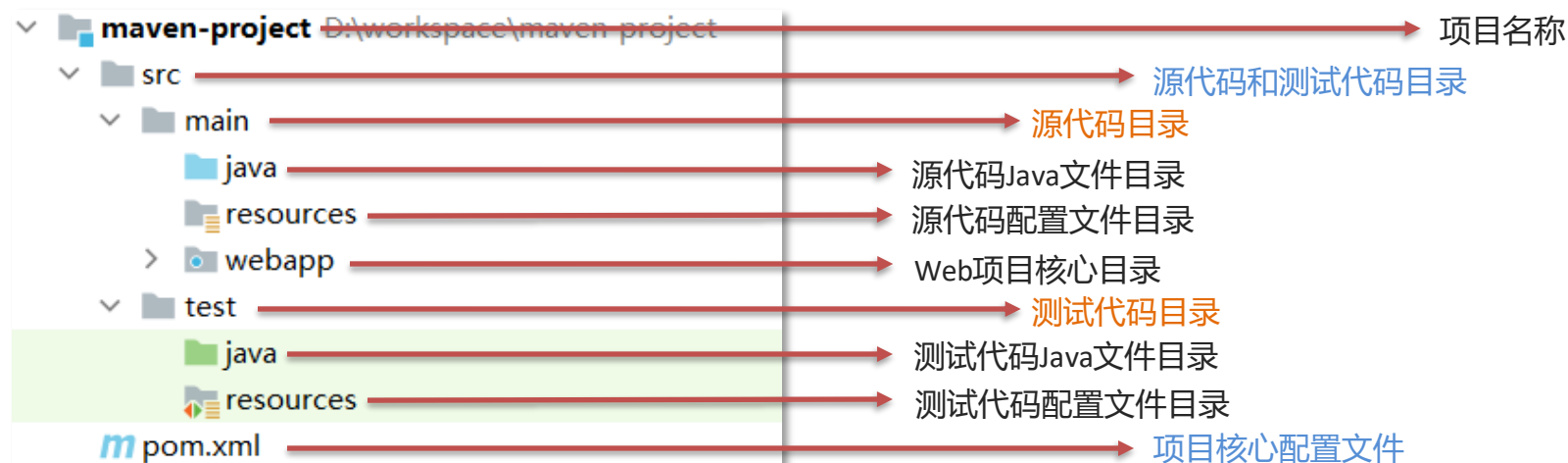


Maven

- Maven是专门用于管理和构建Java项目的工具，它的主要功能有：
 - 提供了一套标准化的项目结构
 - 提供了一套标准化的构建流程（编译，测试，打包，发布.....）
 - 提供了一套依赖管理机制
- 标准化的项目结构

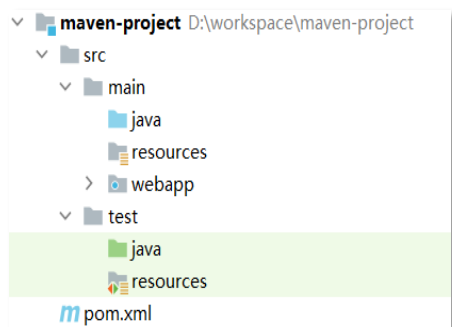


不同IDE之间，项目结构不一样，不通用



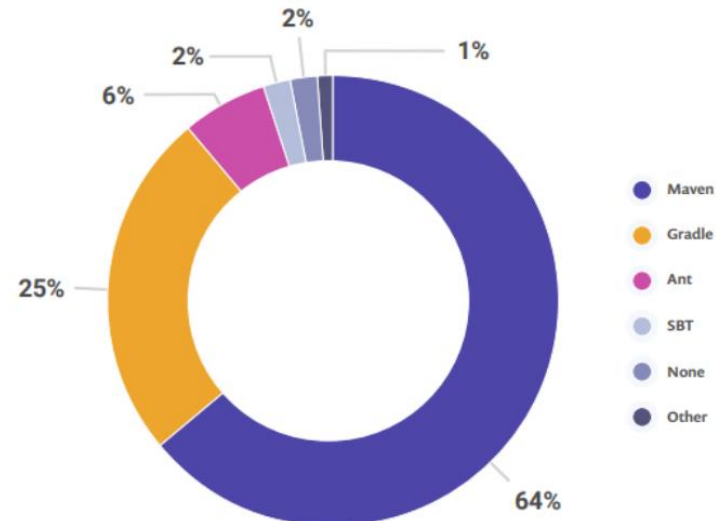
Maven提供了一套标准化的项目结构，所有IDE使用Maven构建的项目结构完全一样，所有IDE创建的Maven项目可以通用

- Maven是专门用于管理和构建Java项目的工具，它的主要功能有：
 - 提供了一套标准化的项目结构
 - 提供了一套标准化的构建流程（编译，测试，打包，发布.....）
 - 提供了一套依赖管理机制
- 标准化的构建流程

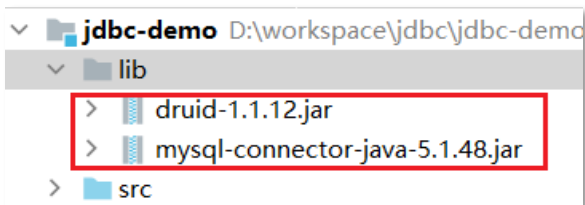


Maven提供了一套简单的命令来完成项目构建

- Maven是专门用于管理和构建Java项目的工具，它的主要功能有：
 - 提供了一套标准化的项目结构
 - 提供了一套标准化的构建流程（编译，测试，打包，发布.....）
 - 提供了一套依赖管理机制
- 依赖管理
 - 依赖管理其实就是管理你项目所依赖的第三方资源（jar包、插件...）



常见的项目构建工具



1. 下载jar 包
2. 复制jar 包到项目
3. 将jar 包加入工作环境



```
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.46</version>
</dependency>
```

1. Maven 使用标准的坐标配置来管理各种依赖
2. 只需要简单的配置就可以完成依赖管理



目录

Contents

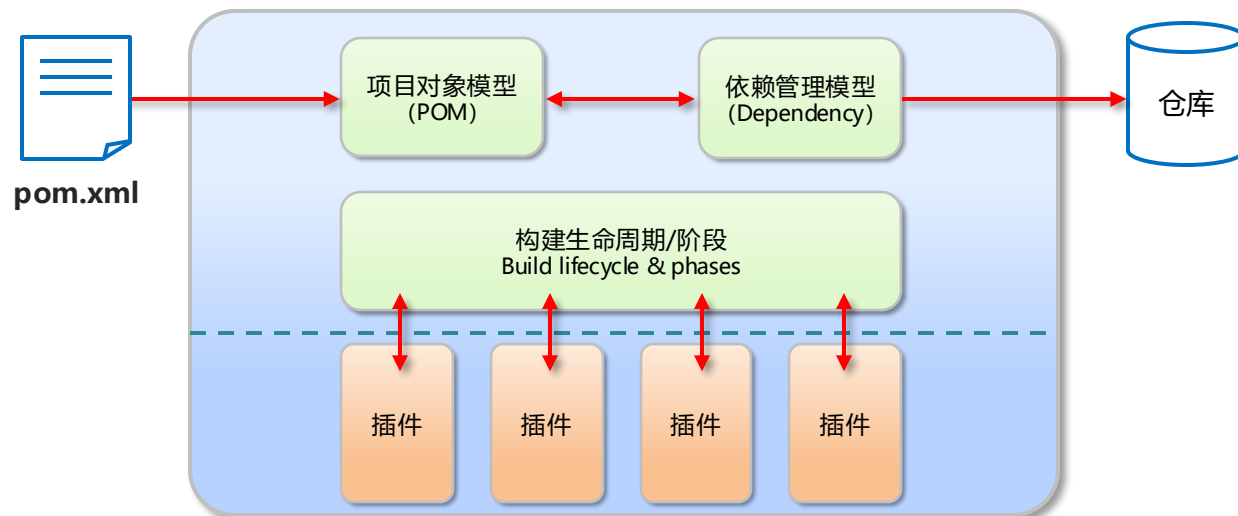
- ◆ Maven 简介
- ◆ Maven 安装配置
- ◆ Maven 基本使用
- ◆ IDEA 配置 Maven
- ◆ 依赖管理

- **Apache Maven** 是一个项目管理和构建**工具**，它基于项目对象模型 (POM) 的概念，通过一小段描述信息来管理项目的构建、报告和文档
- 官网: <http://maven.apache.org/>

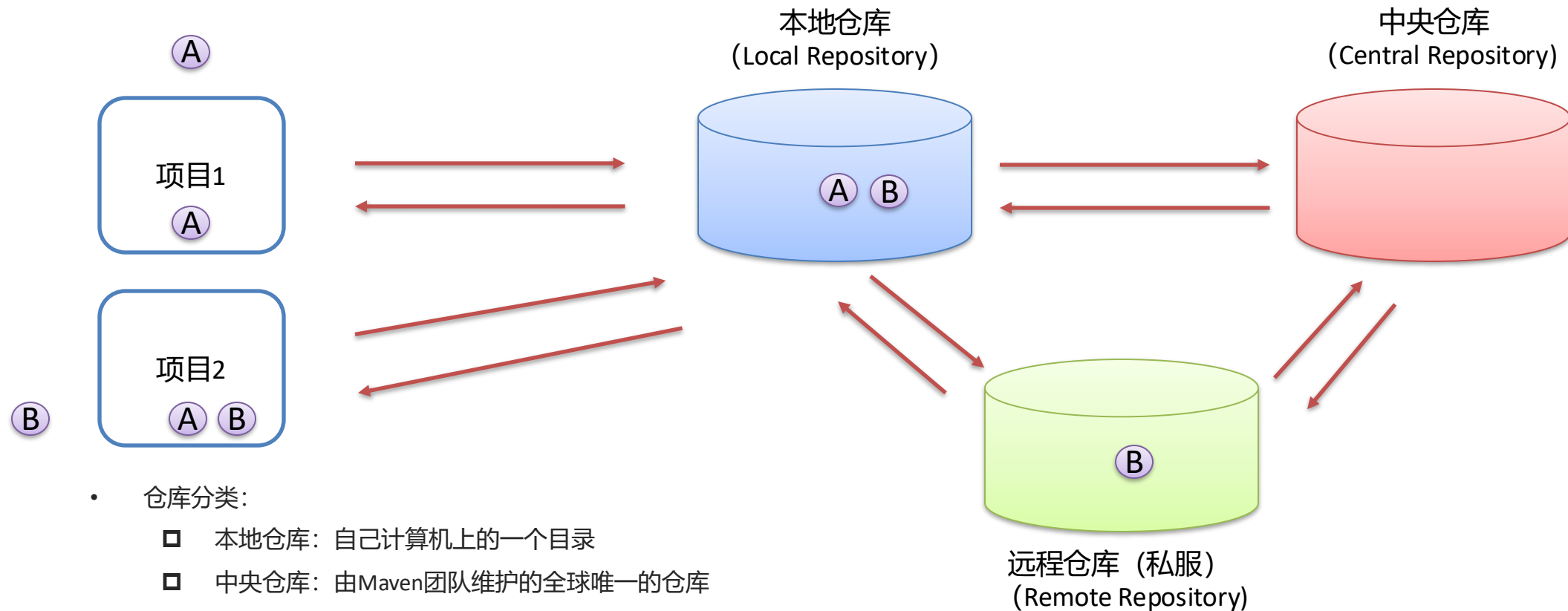
Maven 作用:

- 标准化的项目结构
- 标准化的构建流程
- 方便的依赖管理

Maven 模型:



- 项目对象模型 (Project Object Model)
- 依赖管理模型 (Dependency)
- 插件 (Plugin)



- 仓库分类：
 - ❑ 本地仓库：自己计算机上的一个目录
 - ❑ 中央仓库：由Maven团队维护的全球唯一的仓库
 - 地址：<https://repo1.maven.org/maven2/>
 - ❑ 远程仓库(私服)：一般由公司团队搭建的私有仓库
- 当项目中使用坐标引入对应依赖jar包后，首先会查找本地仓库中是否有对应的jar包：
 - ❑ 如果有，则在项目直接引用；
 - ❑ 如果没有，则去中央仓库中下载对应的jar包到本地仓库。
- 还可以搭建远程仓库，将来jar包的查找顺序则变为：
 - ❑ 本地仓库 → 远程仓库 → 中央仓库



目录

Contents

- ◆ Maven 简介
- ◆ **Maven 安装配置**
- ◆ Maven 基本使用
- ◆ IDEA 配置 Maven
- ◆ 依赖管理

步骤

Maven安装配置

1. 解压 apache-maven-3.6.1.rar 既安装完成
2. 配置环境变量 MAVEN_HOME 为安装路径的bin目录
3. 配置本地仓库：修改 conf/settings.xml 中的 <localRepository> 为一个指定目录
4. 配置阿里云私服：修改 conf/settings.xml 中的 <mirrors> 标签，为其添加如下子标签：

```
<mirror>
  <id>alimaven</id>
  <name>aliyun maven</name>
  <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
  <mirrorOf>central</mirrorOf>
</mirror>
```



目录

Contents

- ◆ Maven 概念模型
- ◆ Maven 安装配置
- ◆ **Maven 基本使用**
- ◆ IDEA 配置 Maven
- ◆ 依赖管理



Maven 基本使用

- Maven 常用命令
- Maven 生命周期

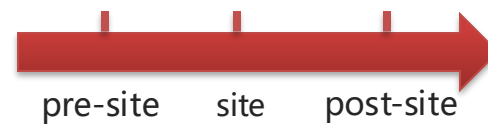
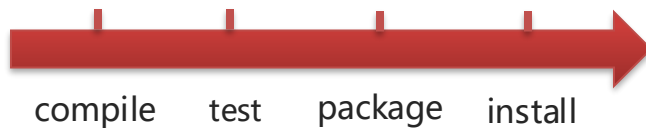
Maven 常用命令

- compile : 编译
- clean: 清理
- test: 测试
- package: 打包
- install: 安装

Maven 生命周期

- Maven 构建项目生命周期描述的是一次构建过程经历经历了多少个事件
- Maven 对项目构建的生命周期划分为3套
 - clean: 清理工作
 - default: 核心工作, 例如编译, 测试, 打包, 安装等
 - site: 产生报告, 发布站点等

同一生命周期内, 执行后边的命令, 前边的所有命令会自动执行



default 构建生命周期

- validate (校验)
校验项目是否正确并且所有必要的信息可以完成项目的构建过程。
- initialize (初始化)
初始化构建状态, 比如设置属性值。
- generate-sources (生成源代码)
生成包含在编译阶段中的任何源代码。
- process-sources (处理源代码)
处理源代码, 比如说, 过滤任意值。
- generate-resources (生成资源文件)
生成将会包含在项目包中的资源文件。
- process-resources (处理资源文件)
复制和处理资源到目标目录, 为打包阶段最好准备。
- **compile (编译)**
编译项目的源代码。
- process-classes (处理类文件)
处理编译生成的文件, 比如说对Java class文件做字节码改善优化。
- generate-test-sources (生成测试源代码)
生成包含在编译阶段中的任何测试源代码。
- process-test-sources (处理测试源代码)
处理测试源代码, 比如说, 过滤任意值。
- generate-test-resources (生成测试资源文件)
为测试创建资源文件。
- process-test-resources (处理测试资源文件)
复制和处理测试资源到目标目录。
- test-compile (编译测试源码)
编译测试源代码到测试目标目录。
- process-test-classes (处理测试类文件)
处理测试源码编译生成的文件。
- **test (测试)**
使用合适的单元测试框架运行测试 (JUnit是其中之一)。
- prepare-package (准备打包)
在实际打包之前, 执行任何的必要的操作为打包做准备。
- **package (打包)**
将编译后的代码打包成可分发格式的文件, 比如JAR、WAR或者EAR文件。
- pre-integration-test (集成测试前)
在执行集成测试前进行必要的动作。比如说, 搭建需要的环境。
- integration-test (集成测试)
处理和部署项目到可以运行集成测试环境中。
- post-integration-test (集成测试后)
在执行集成测试完成后进行必要的动作。比如说, 清理集成测试环境。
- verify (验证)
运行任意的检查来验证项目包有效且达到质量标准。
- **install (安装)**
安装项目包到本地仓库, 这样项目包可以用作其他本地项目的依赖。
- deploy (部署)
将最终的项目包复制到远程仓库中与其他开发者和项目共享。



目录

Contents

- ◆ Maven 概念模型
- ◆ Maven 安装配置
- ◆ Maven 基本使用
- ◆ IDEA 配置 Maven
- ◆ 依赖管理



IDEA 配置 Maven

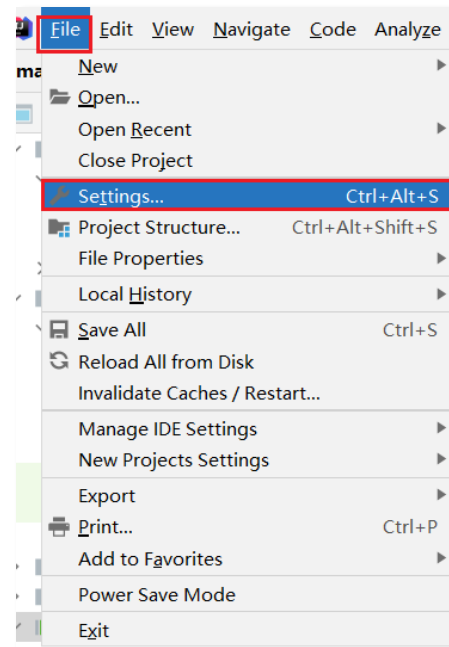
- IDEA 配置 Maven 环境
- Maven 坐标详解
- IDEA 创建 Maven 项目
- IDEA 导入 Maven 项目

步骤

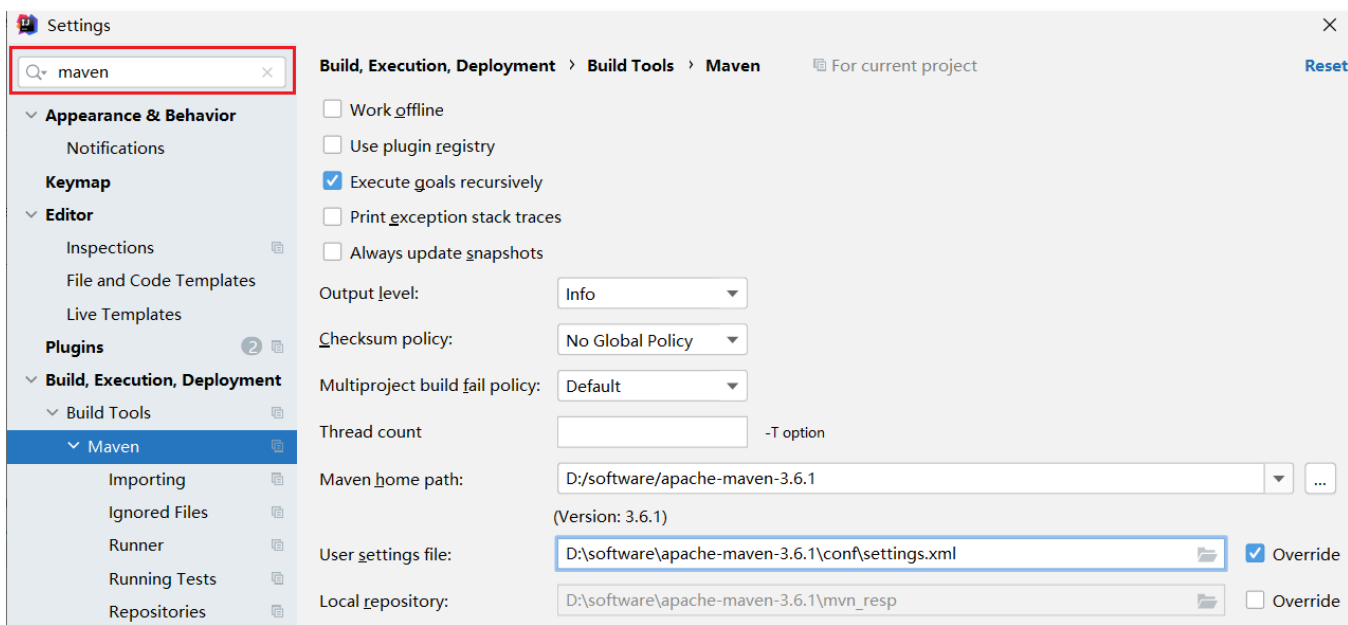
IDEA 配置 Maven 环境

1. 选择 IDEA 中 File --> Settings
2. 搜索 maven
3. 设置 IDEA 使用本地安装的 Maven，并修改配置文件路径

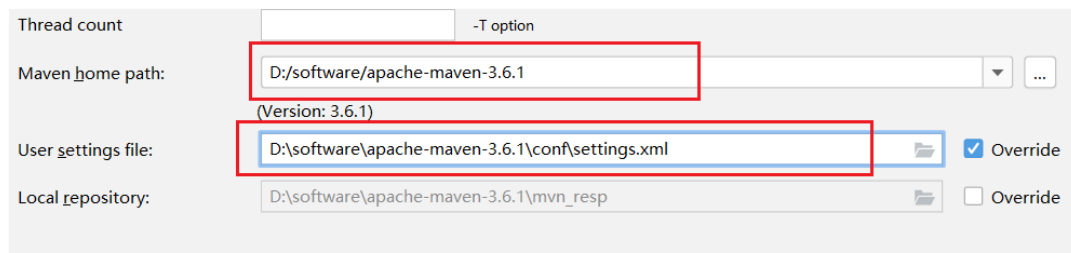
①



②



③





IDEA 配置 Maven

- IDEA 配置 Maven 环境
- **Maven 坐标详解**
- IDEA 创建 Maven 项目
- IDEA 导入 Maven 项目

Maven 坐标详解

- 什么是坐标?
 - Maven 中的坐标是资源的唯一标识
 - 使用坐标来定义项目或引入项目中需要的依赖
- Maven 坐标主要组成
 - groupId: 定义当前Maven项目隶属组织名称 (通常是域名反写, 例如: com.itheima)
 - artifactId: 定义当前Maven项目名称 (通常是模块名称, 例如 order-service、goods-service)
 - version: 定义当前项目版本号

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

```
<dependency>  
  <groupId>mysql</groupId>  
  <artifactId>mysql-connector-java</artifactId>  
  <version>5.1.46</version>  
</dependency>
```



IDEA 配置 Maven

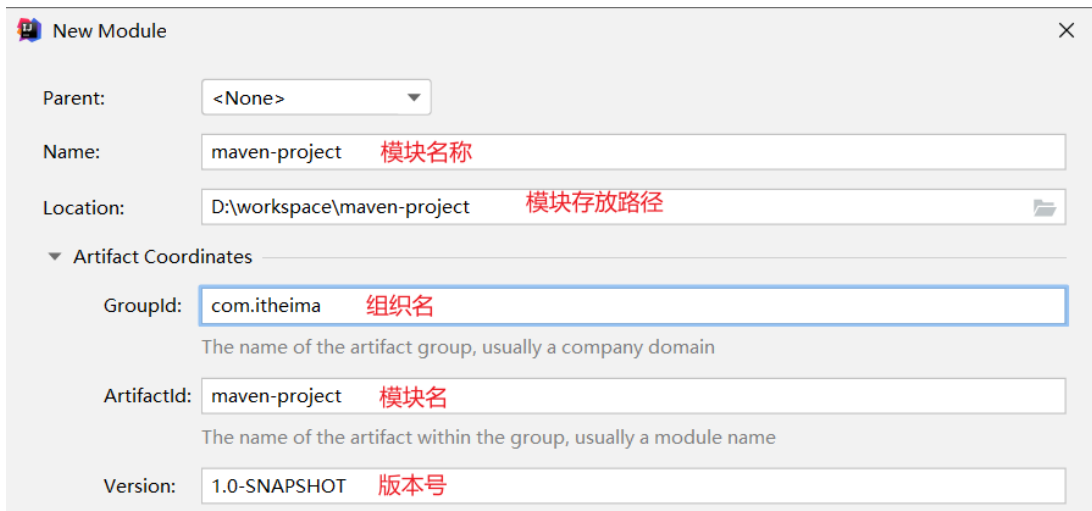
- IDEA 配置 Maven 环境
- Maven 坐标详解
- **IDEA 创建 Maven 项目**
- IDEA 导入 Maven 项目

步骤

IDEA 创建 Maven 项目

1. 创建模块，选择Maven，点击Next
2. 填写模块名称，坐标信息，点击finish，创建完成
3. 编写 HelloWorld，并运行

②



New Module

Parent: <None>

Name: maven-project 模块名称

Location: D:\workspace\maven-project 模块存放路径

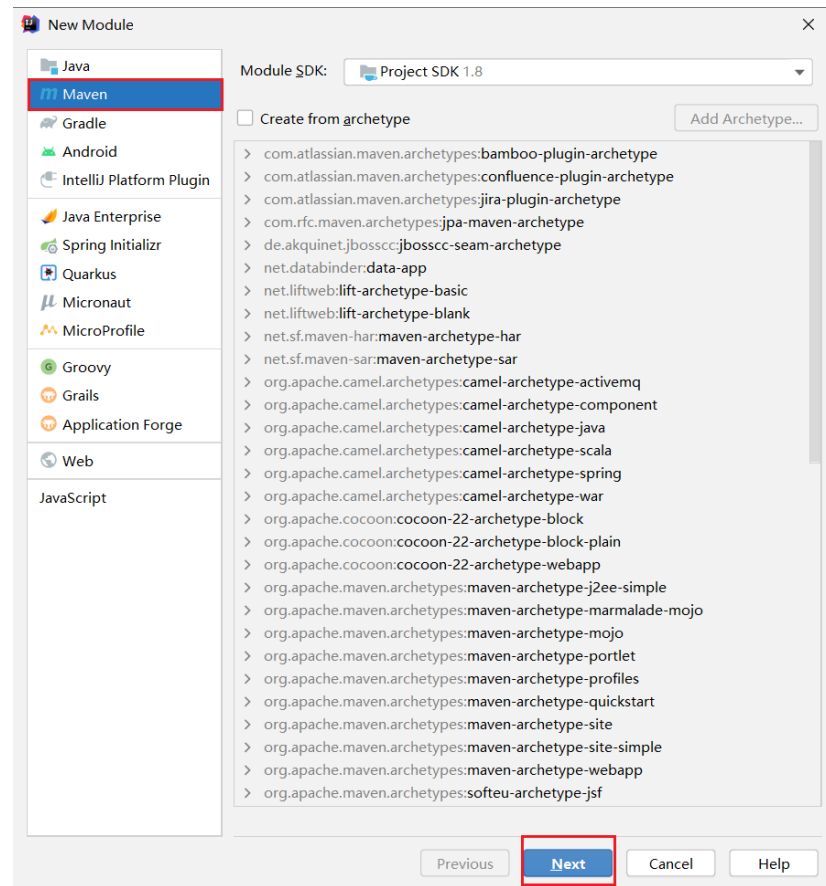
▼ Artifact Coordinates

GroupId: com.itheima 组织名
The name of the artifact group, usually a company domain

ArtifactId: maven-project 模块名
The name of the artifact within the group, usually a module name

Version: 1.0-SNAPSHOT 版本号

①



New Module

Module SDK: Project SDK 1.8

☐ Create from archetype Add Archetype...

> com.atlassian.maven.archetypes:bamboo-plugin-archetype
> com.atlassian.maven.archetypes:confluence-plugin-archetype
> com.atlassian.maven.archetypes:jira-plugin-archetype
> com.rfc.maven.archetypes:jpa-maven-archetype
> de.aquinet.jbosscc.jbosscc-seam-archetype
> net.databinder:data-app
> net.liftweb:lift-archetype-basic
> net.liftweb:lift-archetype-blank
> net.sf.maven-har:maven-archetype-har
> net.sf.maven-sar:maven-archetype-sar
> org.apache.camel.archetypes:camel-archetype-activemq
> org.apache.camel.archetypes:camel-archetype-component
> org.apache.camel.archetypes:camel-archetype-java
> org.apache.camel.archetypes:camel-archetype-scala
> org.apache.camel.archetypes:camel-archetype-spring
> org.apache.camel.archetypes:camel-archetype-war
> org.apache.cocoon:cocoon-22-archetype-block
> org.apache.cocoon:cocoon-22-archetype-block-plain
> org.apache.cocoon:cocoon-22-archetype-webapp
> org.apache.maven.archetypes:maven-archetype-j2ee-simple
> org.apache.maven.archetypes:maven-archetype-marmalade-mojo
> org.apache.maven.archetypes:maven-archetype-mojo
> org.apache.maven.archetypes:maven-archetype-portlet
> org.apache.maven.archetypes:maven-archetype-profiles
> org.apache.maven.archetypes:maven-archetype-quickstart
> org.apache.maven.archetypes:maven-archetype-site
> org.apache.maven.archetypes:maven-archetype-site-simple
> org.apache.maven.archetypes:maven-archetype-webapp
> org.apache.maven.archetypes:softeu-archetype-jsf

Previous Next Cancel Help



IDEA 配置 Maven

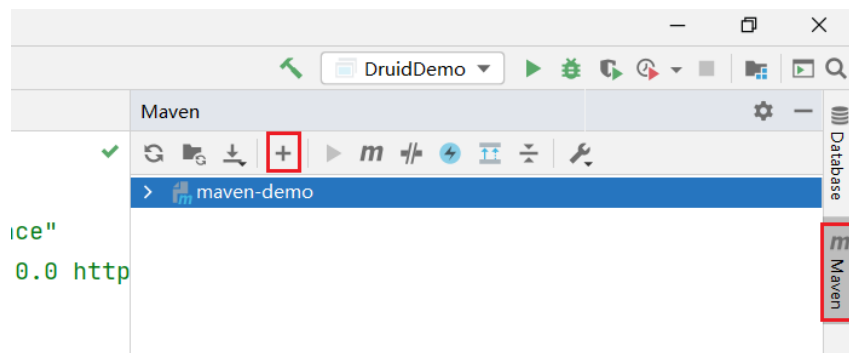
- IDEA 配置 Maven 环境
- Maven 坐标详解
- IDEA 创建 Maven 项目
- IDEA 导入 Maven 项目

步骤

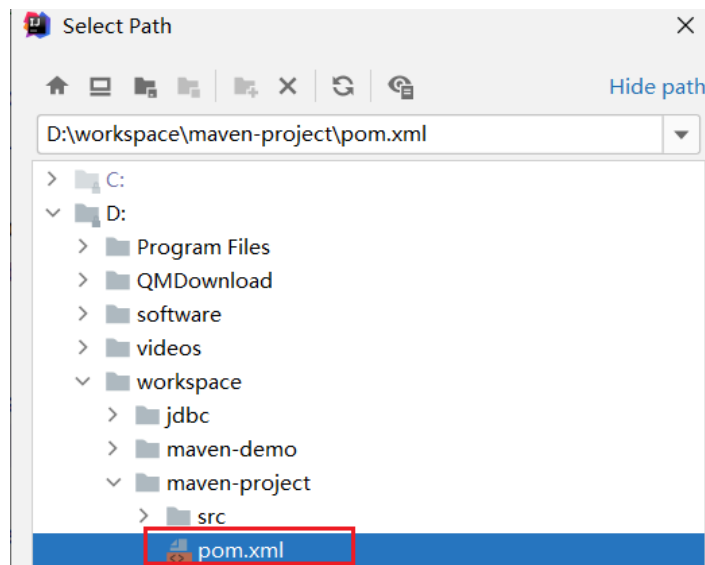
IDEA 导入 Maven 项目

1. 选择右侧Maven面板, 点击 + 号
2. 选中对应项目的pom.xml文件, 双击即可
3. 如果没有Maven面板, 选择
View → Appearance → Tool Window Bars

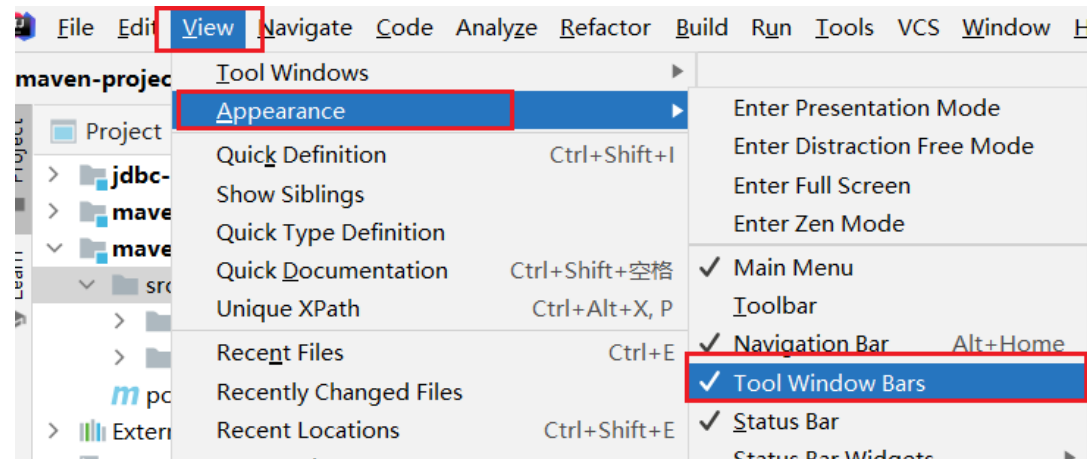
①



②



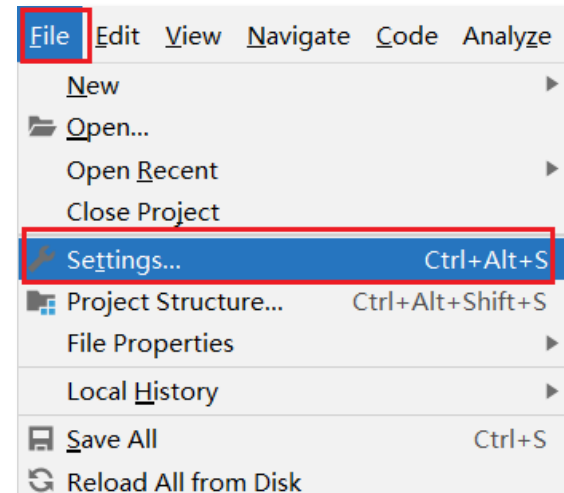
③



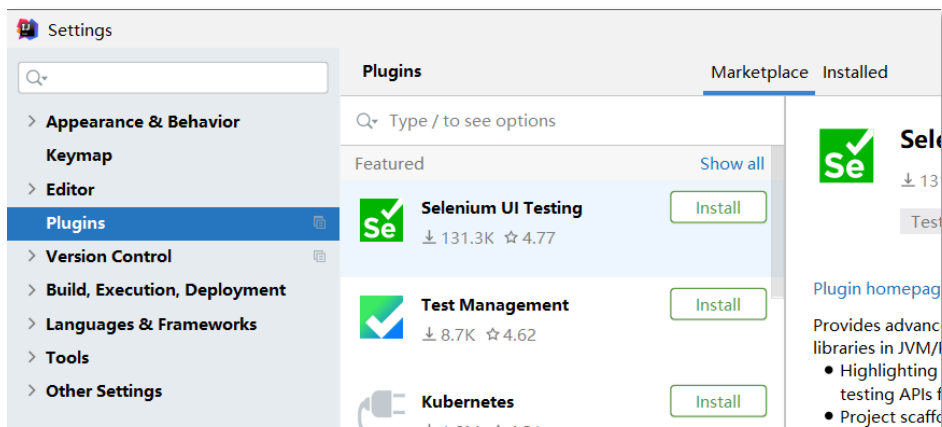
配置 Maven-Helper 插件

1. 选择 IDEA 中 File --> Settings
2. 选择 Plugins
3. 搜索 Maven, 选择第一个 Maven Helper, 点击 Install 安装, 弹出面板中点击 Accept
4. 重启 IDEA

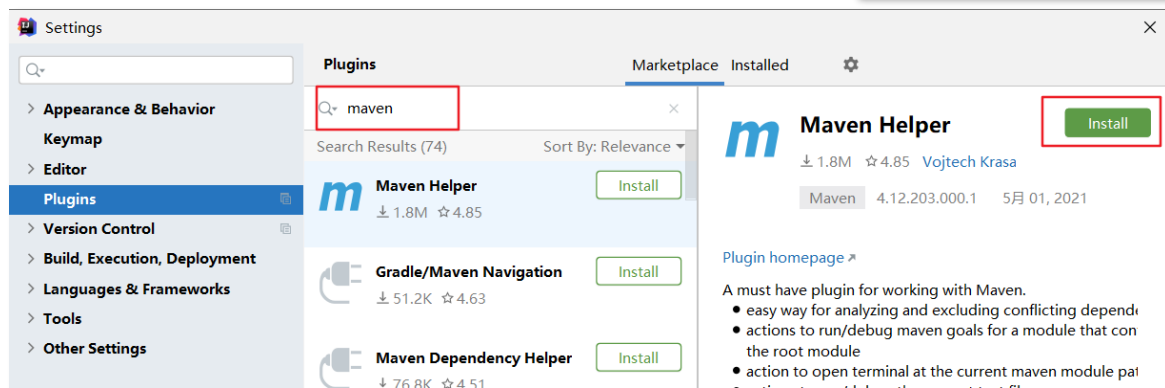
①



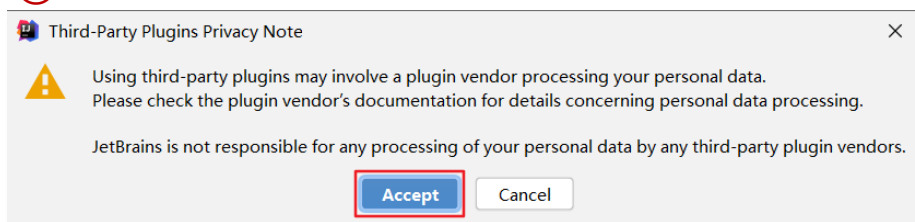
②



③



④





目录

Contents

- ◆ Maven 概念模型
- ◆ Maven 安装配置
- ◆ Maven 基本使用
- ◆ IDEA 配置 Maven
- ◆ 依赖管理

步骤

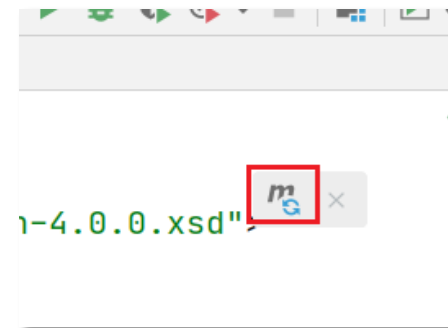
使用坐标导入 jar 包

1. 在 pom.xml 中编写 `<dependencies>` 标签
2. 在 `<dependencies>` 标签中 使用 `<dependency>` 引入坐标
3. 定义坐标的 `groupId`, `artifactId`, `version`
4. 点击刷新按钮, 使坐标生效

①

```
<dependencies>
  <!-- mysql 坐标 -->
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.32</version>
  </dependency>
</dependencies>
```

②

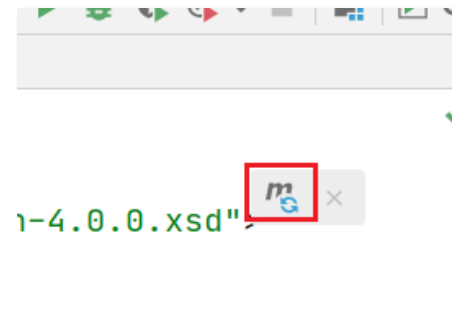


步骤

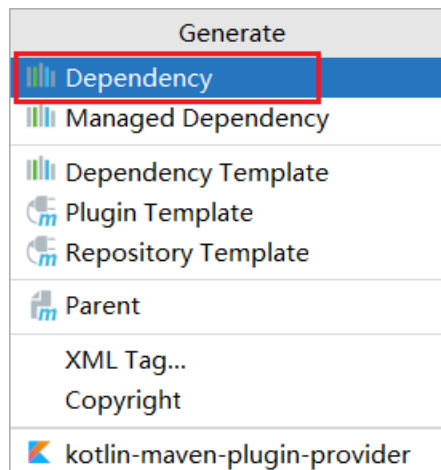
使用坐标导入 jar 包 – 快捷方式

1. 在 pom.xml 中按 alt + insert, 选择 Dependency
2. 在弹出的面板中搜索对应坐标, 然后双击选中对应坐标
3. 点击刷新按钮, 使坐标生效

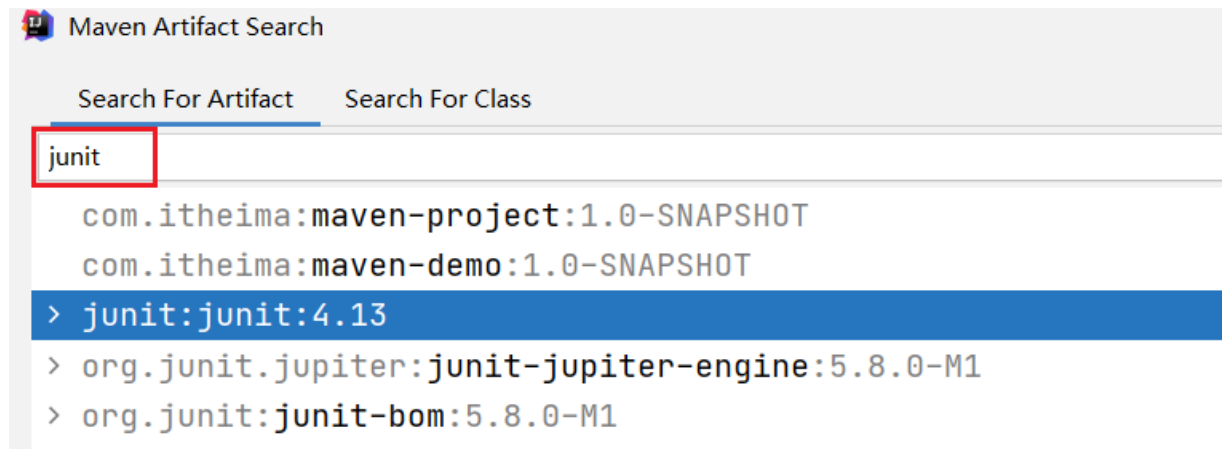
③



①



②

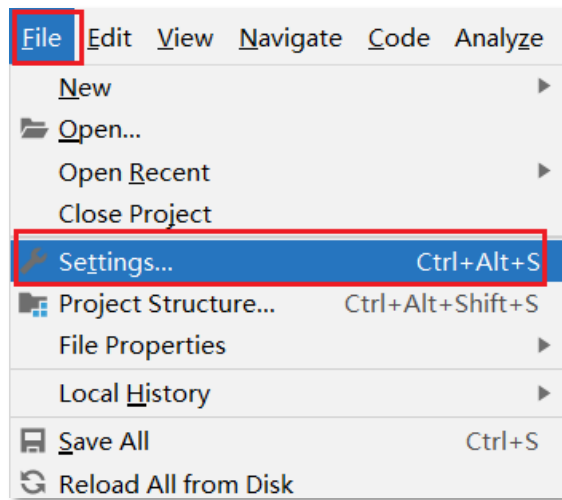


① 步骤

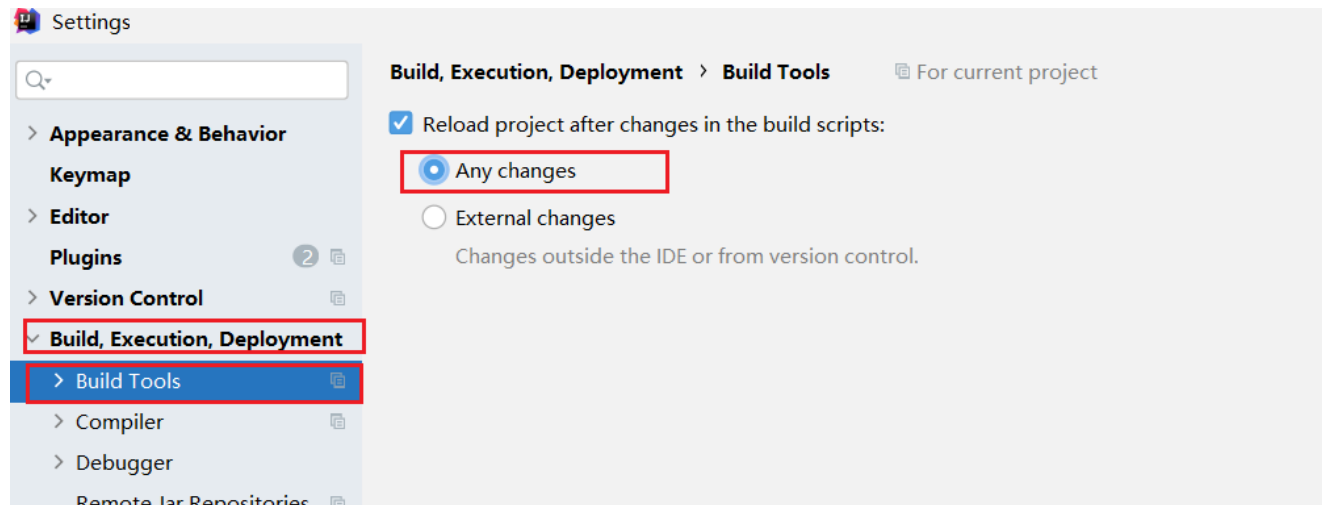
使用坐标导入 jar 包 – 自动导入

1. 选择 IDEA 中 File --> Settings
2. 在弹出的面板中找到 Build Tools
3. 选择 Any changes, 点击 ok 即可生效

①



②



依赖范围

- 通过设置坐标的依赖范围(scope)，可以设置 对应jar包的作用范围：编译环境、测试环境、运行环境

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13</version>
  <scope>test</scope>
</dependency>
```

依赖范围	编译classpath	测试classpath	运行classpath	例子
compile	Y	Y	Y	logback
test	-	Y	-	Junit
provided	Y	Y	-	servlet-api
runtime	-	Y	Y	jdbc驱动
system	Y	Y	-	存储在本地的jar包
import	引入DependencyManagement			

- <scope>默认值： compile



传智教育旗下高端IT教育品牌