

# 一、初始 Maven

---

## 1、项目遇到的问题

---

- 构建：编译代码，运行测试，打包，部署应用，运行服务器等；
- 依赖：项目依赖大量的第三方包，第三方包又依赖另外的包，对依赖包的管理非常麻烦。

## 2、Maven 定义和作用

---

- Maven 翻译为“知识的积累”，“专家”，“行家”，是一个跨平台的项目管理工具；
- Maven 主要用作基于 Java 平台的项目（Maven 本身也是 Java 编写的）的构建、依赖包管理和项目信息管理；
- Maven 能提供**一种项目的配置**，配置好的项目，只需要运行一条简单的命令，就能完成重复的，繁琐的构建动作；
- Maven 能提供一种项目的依赖配置，配置好的项目，Maven 能自动的**从 Maven 的中央仓库中帮我们下载并管理项目依赖的 jar 包**，并且还能自动的管理这些 jar 包依赖的其他 jar 包；
- Maven 提供了一种**标准的项目目录结构**，测试命名规则等项目的最佳实践方案，统一了不同项目的学习成本（约定大于配置）。

# 二、Maven 安装

---

## 1、检查 JDK 的版本

---

因为 Maven 不同的版本对 JDK 是有要求的，具体请看[官网](#)。

在命令行输入 `java -version`，查询安装 JDK 是否正确并查询安装的版本。若没有配置好的话，按照一下步骤配置好：

- 添加 JAVA\_HOME，需要指向 JDK 安装目录；
- 添加 PATH，添加 `%JAVA_HOME%\bin`。

## 2、安装 Maven

---

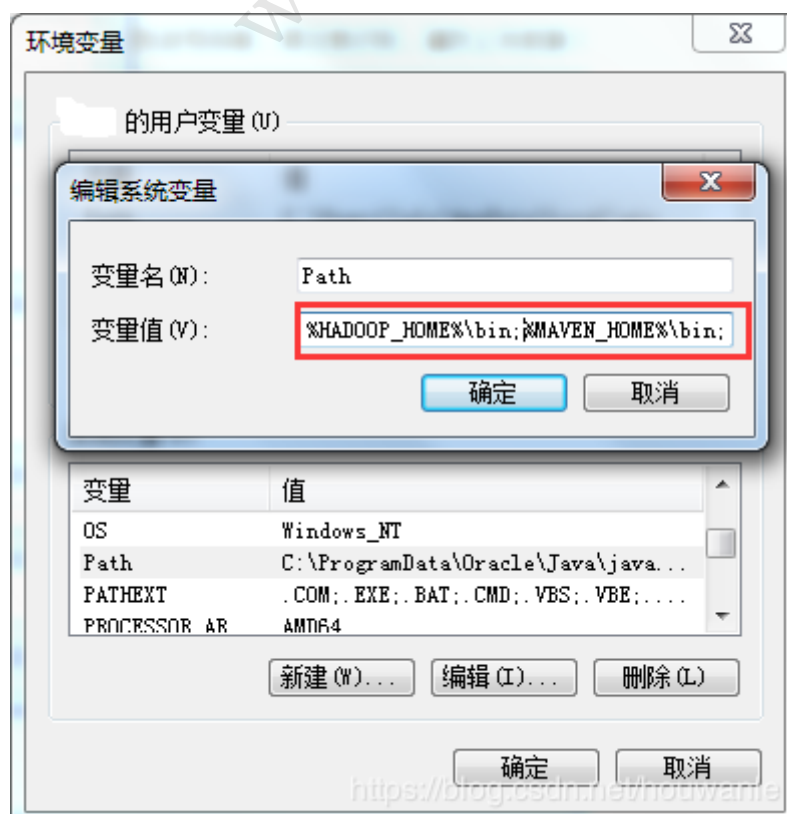
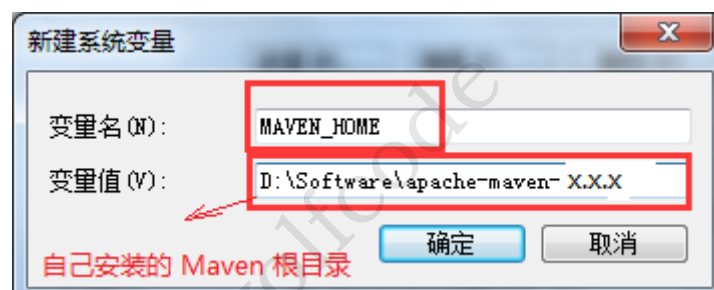
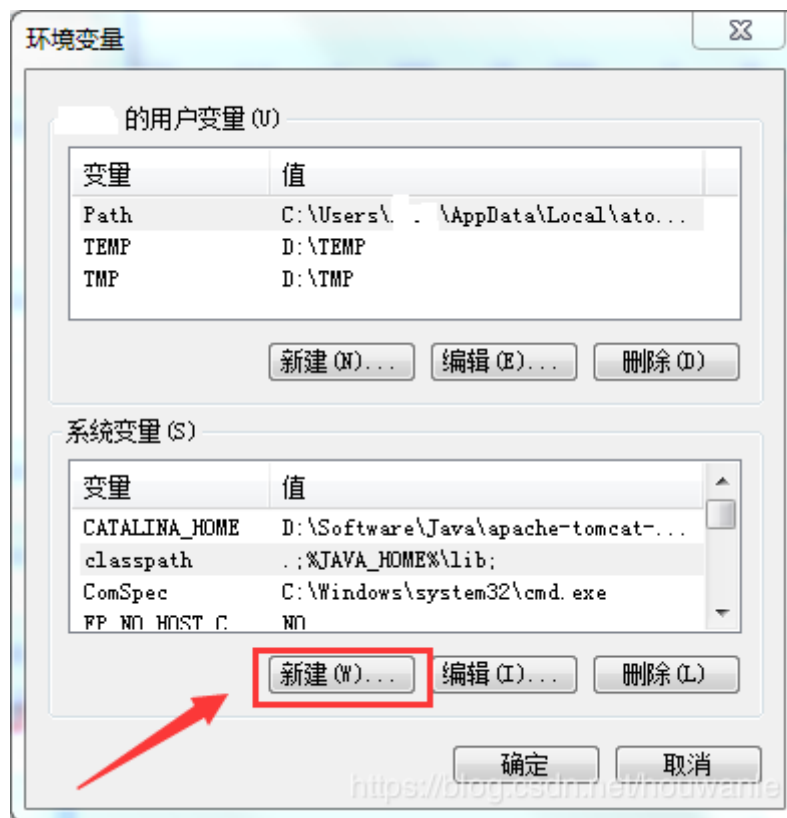
把 `apache-maven-x.x.x-bin.zip` 解压到目录中（**目录路径最好不要有空格和中文**）。

## 3、配置环境变量

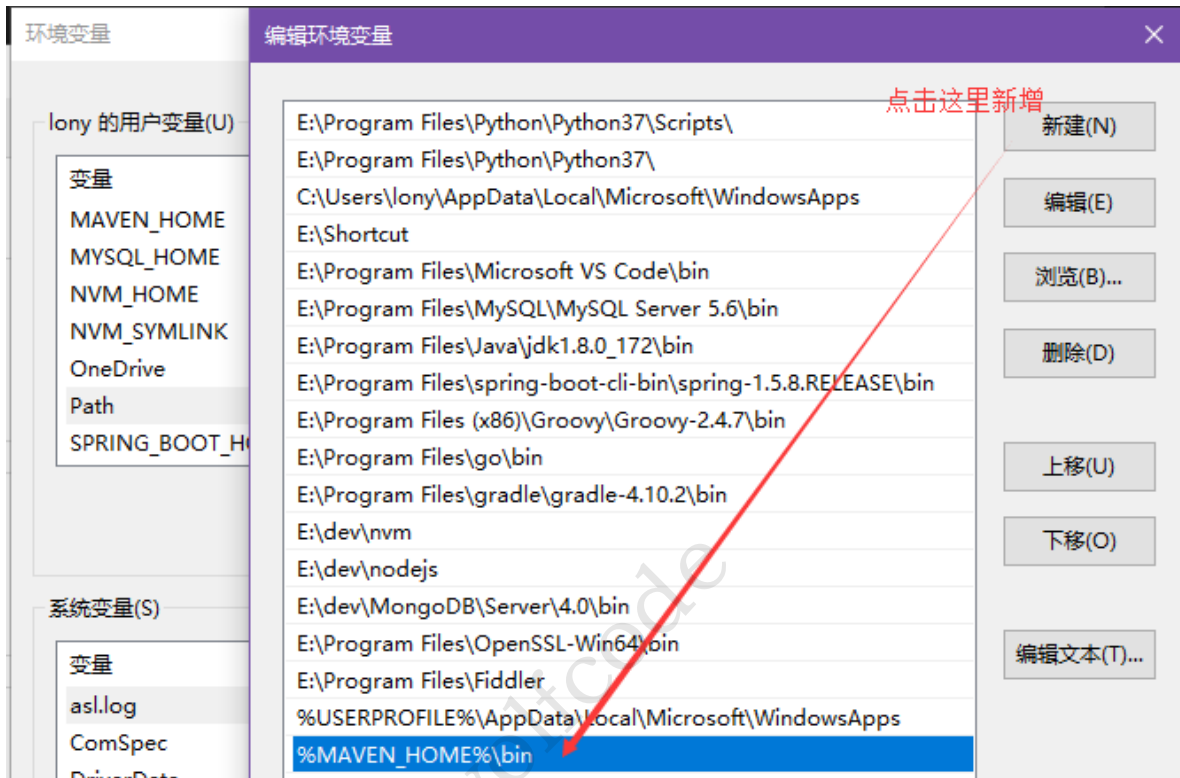
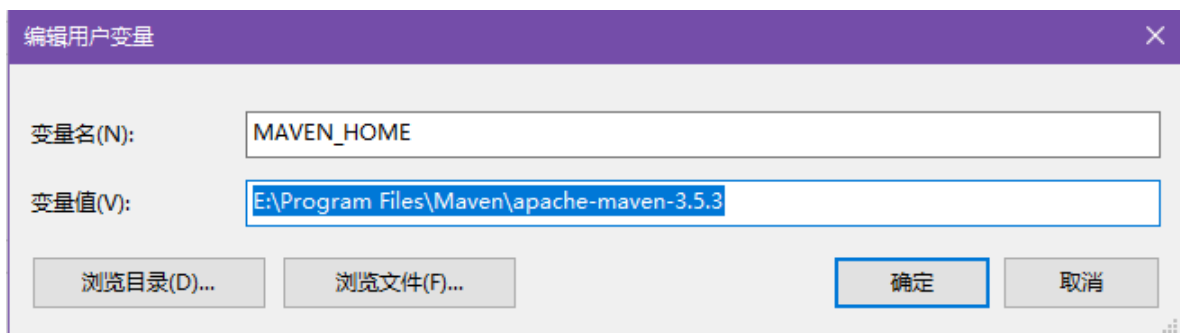
---

- 配置环境变量，MAVEN\_HOME，指向 maven 的根目录；
- 配置环境变量 Path，将 `%MAVEN_HOME%\bin` 追加到 Path 中。

### 3.1、Win7 配置



### 3.2、Win10 配置



## 4、验证 Maven 安装

打开命令行窗口，输入 `mvn -v` 执行，若执行类似如下效果即代表安装成功。

```
C:\Users\lony>mvn -v
Apache Maven 3.5.3 (3383c37e1f9e9b3bc3df5050c29c8aff9f295297; 2018-02-25T03:49:05+08:00)
Maven home: E:\Program Files\Maven\apache-maven-3.5.3\bin\..
Java version: 1.8.0_172, vendor: Oracle Corporation
Java home: E:\Program Files\Java\jdk1.8.0_172\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

## 三、Maven Hello World



把事先准备好的 Maven 项目，在命令行下输入命令试下：

- `mvn compile`：编译

- mvn clean: 清除
- mvn test: 测试
- mvn package: 打包
- mvn install: 打包, 本地仓库也放一份

## 四、Maven 设置

### 1、本地仓库

运行过 Maven 命令才会在你的用户目录生成一个 .m2 目录, 该目录下有个目录名 repository, 这个就是所谓的本地仓库, 主要存放是一些下载的 jar 文件。

本地仓库默认位置在用户目录/.m2/repository, 本地仓库不建议在系统盘, 最佳实践:

- 将 Maven 根目录的 conf/settings.xml 拷贝到 .m2 中, 作为个人 Maven 配置文件;
- 在 settings.xml 文件中 修改本地仓库的位置。

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- 本地仓库地址 -->
  <localRepository>你本地仓库的目录位置</localRepository>
</settings>
```

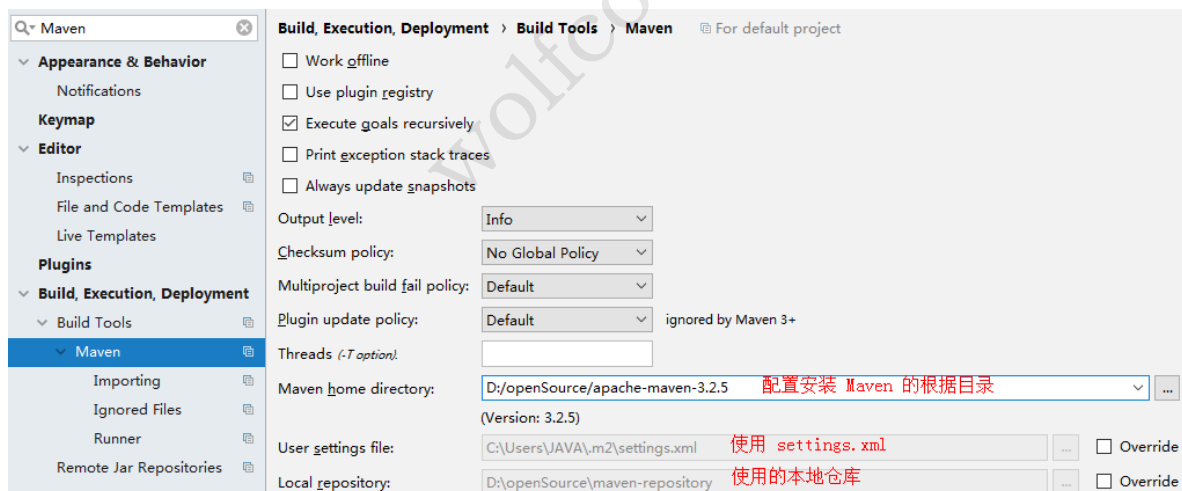
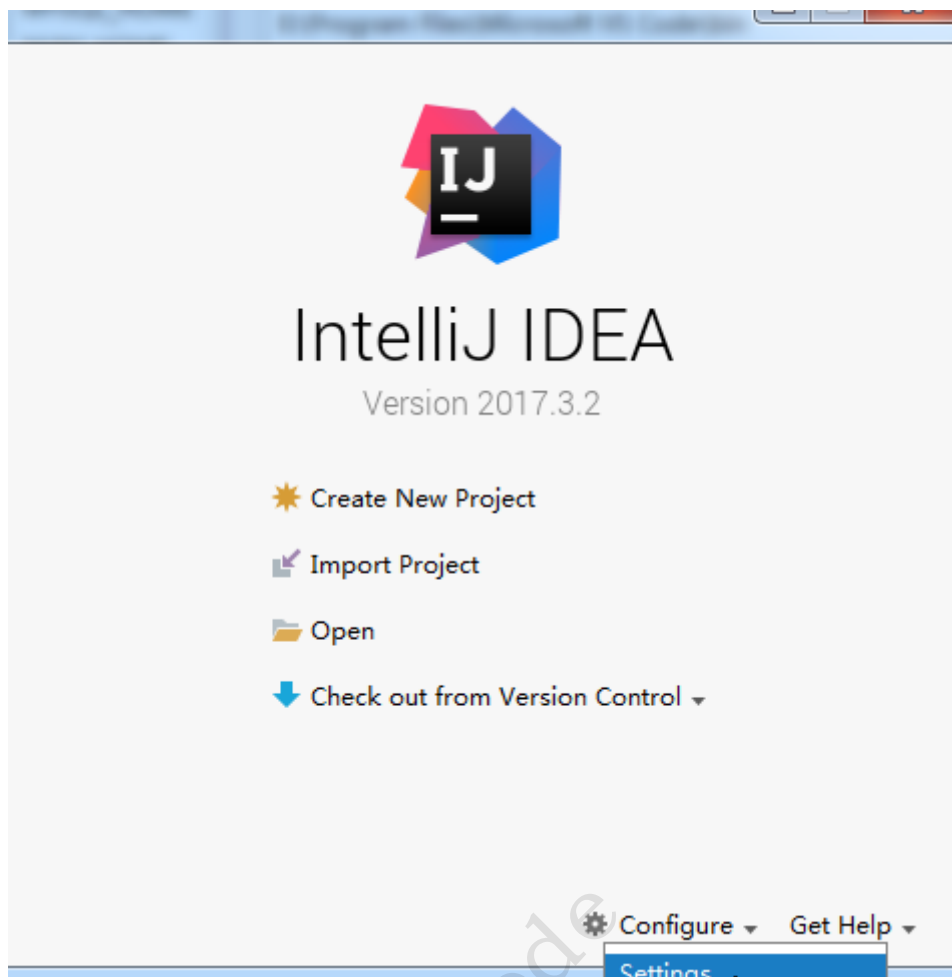
### 2、配置中央仓库镜像

在有网情况下, 官方中央仓库服务器在国外, 所以很大可能出现下载失败的问题, 所以我们配置一个中央仓库的镜像来降低下载失败的概率。

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <!-- 本地仓库地址 -->
  <localRepository>你本地仓库的目录位置</localRepository>
  <mirrors>
    <!-- 阿里仓库镜像 -->
    <mirror>
      <id>alimaven</id>
      <name>aliyun maven</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
</settings>
```

## 五、IDEA 中 Maven 的配置

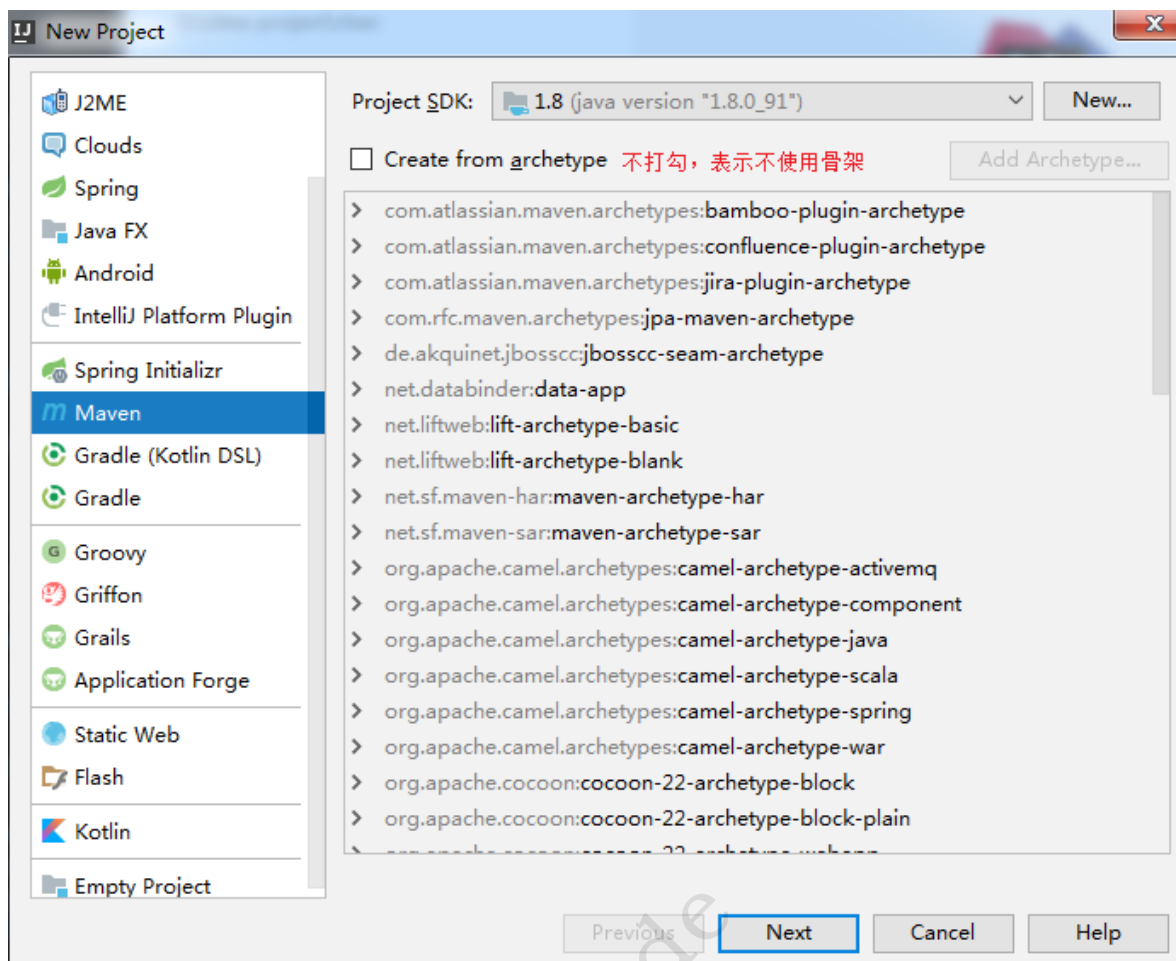
配置安装的 Maven, 设置 settings.xml 和本地仓库。



## 六、搭建基于 Maven 构建的 JavaSE 项目

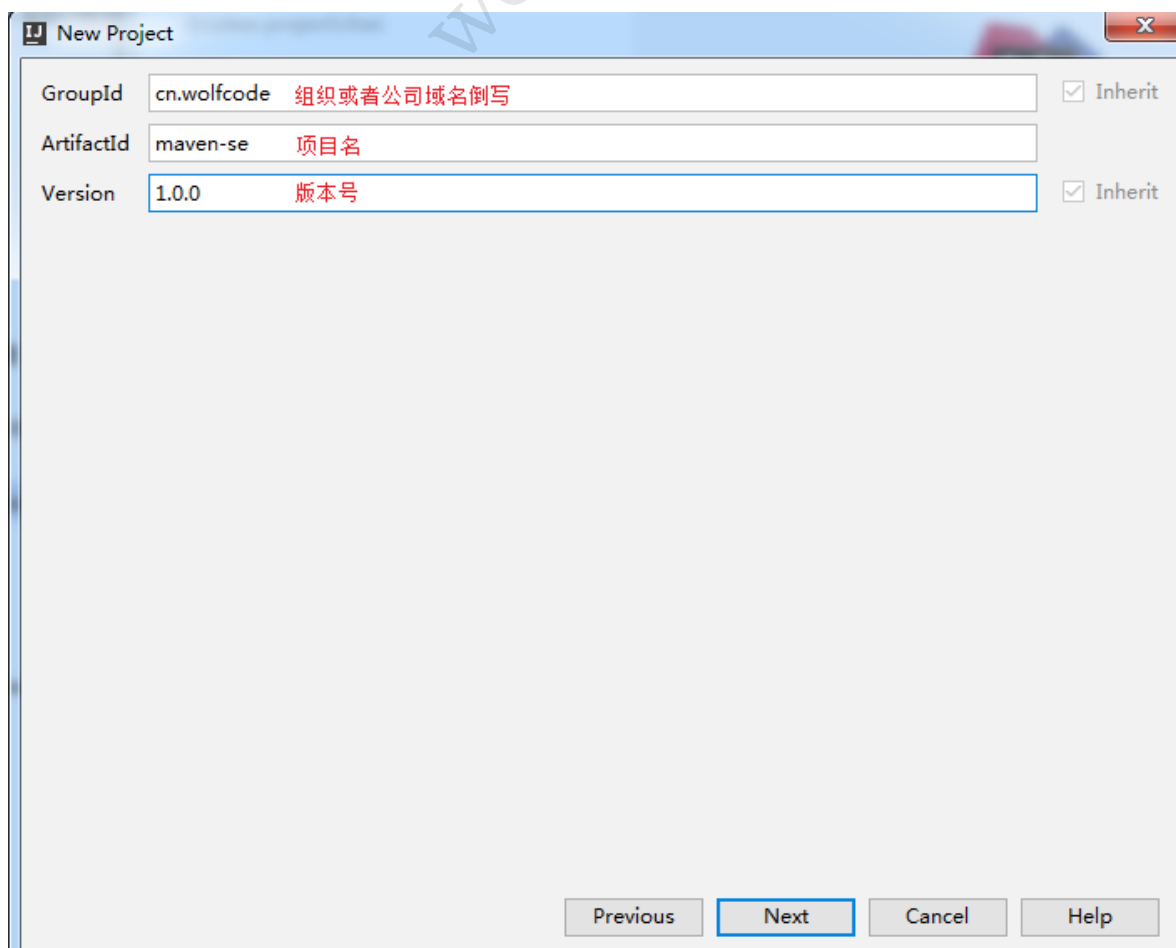
### 1、创建项目

#### 1.1、选择建的是 Maven 项目



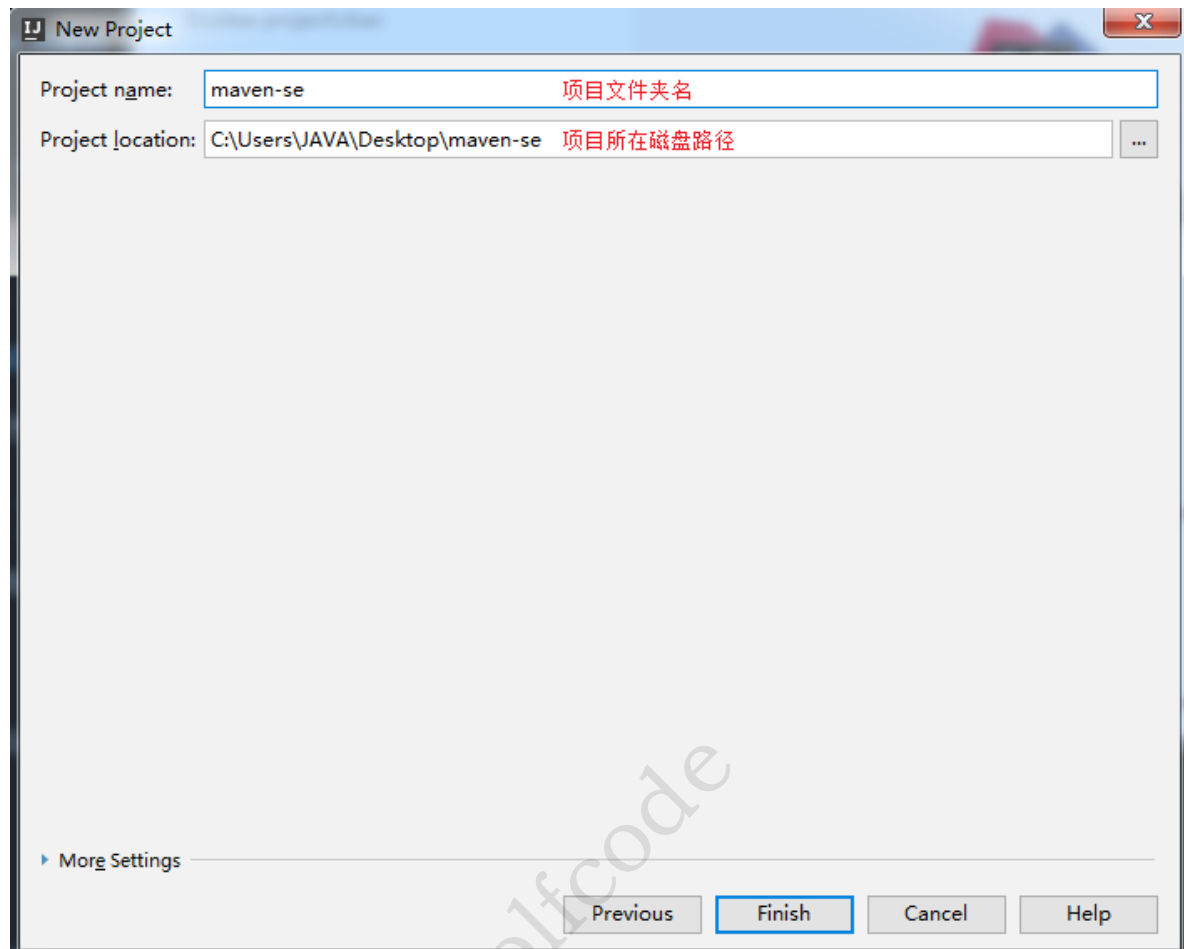
确定后之后点击 Next。

## 1.2、填写对应项目信息



填完之后点击 Next。

## 1.3、确定项目存储位置



确定完之后点击 Finish，之后工具会打开解析项目，要等一会儿。

## 2、pom.xml 文件

用于填写项目信息，打包方式，添加依赖及插件等。

```
<?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>cn.wolfcode</groupId>
  <artifactId>maven-se</artifactId>
  <version>1.0.0</version>

</project>
```

## 3、添加依赖

### 3.1、利用索引添加依赖

所谓建索引，就对本地仓库的依赖建立索引，若本地仓库存在 10 个依赖，那么在添加的时候可以搜索得出来，很方便在 pom.xml 添加。但注意搜索得出来是本地仓库已有的前提下。而这个索引在 IDEA 开发工具会自动建好，直接使用即可。

### 3.1.1、添加 dependencies 元素

```
<?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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.wolfcode</groupId>
    <artifactId>maven-se</artifactId>
    <version>1.0.0</version>
```

I

```
<d
</pr dependencies      http://maven.apache.org/POM/4.0.0
dependencyManagement  http://maven.apache.org/POM/4.0.0
description            http://maven.apache.org/POM/4.0.0
developers             http://maven.apache.org/POM/4.0.0
distributionManagement http://maven.apache.org/POM/4.0.0
artifactId             http://maven.apache.org/POM/4.0.0
build                 http://maven.apache.org/POM/4.0.0
groupId               http://maven.apache.org/POM/4.0.0
modelVersion          http://maven.apache.org/POM/4.0.0
modules               http://maven.apache.org/POM/4.0.0
Press Ctrl+句点 to choose the selected (or first) suggestion and insert a dot afterwards >> π
```

### 3.1.2、添加 dependency 元素

```
<?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/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.wolfcode</groupId>
    <artifactId>maven-se</artifactId>
    <version>1.0.0</version>

    <dependencies>
        <de
    </d dep dependency
</proj> Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor >>
```

### 3.1.3、填选依赖名称



```
<?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.apac
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.wolfcode</groupId>
    <artifactId>maven-se</artifactId>
    <version>1.0.0</version>

    <dependencies>
      <dependency>
        <groupId></groupId>
        <artifactId>my</artifactId>
        <version></
      </dependency>
    </dependencies>
  </project>
```

mybatis  
mybatis-parent  
mybatis-spring  
mysql-connector-java

Ctrl+向下箭头 and Ctrl+向上箭头 will move caret down and up in the editor >>>

### 3.1.4、选择依赖的版本

```
<?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
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.wolfcode</groupId>
    <artifactId>maven-se</artifactId>
    <version>1.0.0</version>

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

5.1.45  
RELEASE  
LATEST

### 3.1.5、填写依赖的作用域

```

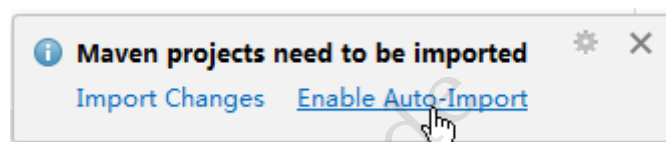
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4
    <modelVersion>4.0.0</modelVersion>

    <groupId>cn.wolfcode</groupId>
    <artifactId>maven-se</artifactId>
    <version>1.0.0</version>

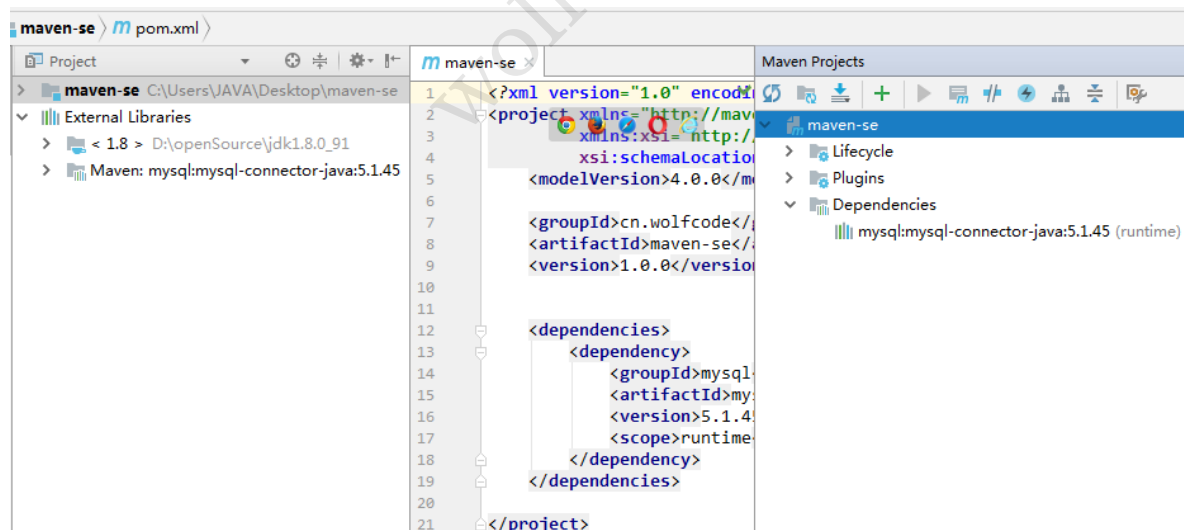
    <dependencies>
      <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.45</version>
        <scope>runtime</scope>
      </dependency>
    </dependencies>
  </project>

```

### 3.1.6、让工具自动解析修改的内容



### 3.1.7、验证依赖添加成功



## 3.2、网站搜索拷贝添加依赖

[搜索依赖的网站。](#)

### 3.2.1、在搜索栏中输入要添加的依赖

Found 287 results

## 搜索到的结果

Sort: **relevance** | popular | newest



## 1. MyBatis

[org.mybatis](#) » [mybatis](#)

367 usages

Apache

The MyBatis SQL mapper framework makes it easier to use a relational database with object-oriented applications. MyBatis couples objects with stored procedures or SQL statements using a XML descriptor or annotations. Simplicity is the biggest advantage of the MyBatis data mapper over object relational mapping tools.

Last Release on Mar 11, 2018

## 2. MyBatis Spring

[org.mybatis » mybatis-spring](#)

158 usages

Apache

An easy-to-use Spring bridge for MyBatis sql mapping framework.

Last Release on Mar 14, 2018

### 3. MyBatis Generator Core

[org.mybatis.generator](#) » [mybatis-generator-core](#)

66 usages

Apache

MyBatis Generator - a code generator for MyBatis and iBatis.

Last Release on Dec 21, 2017

### 3.2.2、选择所要的依赖和其版本



## 1. MyBatis [点击](#)

[org.mybatis](#) » mybatis

367 usages

Apache

The MyBatis SQL mapper framework makes it easier to use a relational database with object-oriented applications. MyBatis couples objects with stored procedures or SQL statements using a XML descriptor or annotations. Simplicity is the biggest advantage of the MyBatis data mapper over object relational mapping tools.

Last Release on Mar 11, 2018

Version		Repository	Usages	Date
3.4.x	<a href="#">3.4.6</a>	Central	<a href="#">16</a>	(Mar, 2018)
	<a href="#">3.4.5</a> 点击对应的版本	Central	<a href="#">67</a>	(Aug, 2017)
	<a href="#">3.4.4</a>	Central	<a href="#">56</a>	(Apr, 2017)
	<a href="#">3.4.3</a>	Central	<a href="#">0</a>	(Apr, 2017)
	<a href="#">3.4.2</a>	Central	<a href="#">27</a>	(Jan, 2017)
	<a href="#">3.4.1</a>	Central	<a href="#">74</a>	(Jun, 2016)
	<a href="#">3.4.0</a>	Central	<a href="#">35</a>	(Apr, 2016)

### 3.2.3、拷贝对应配置到 pom.xml 中

Maven   Gradle   SBT   Ivy   Grape   Leiningen   Buildr   构建工具

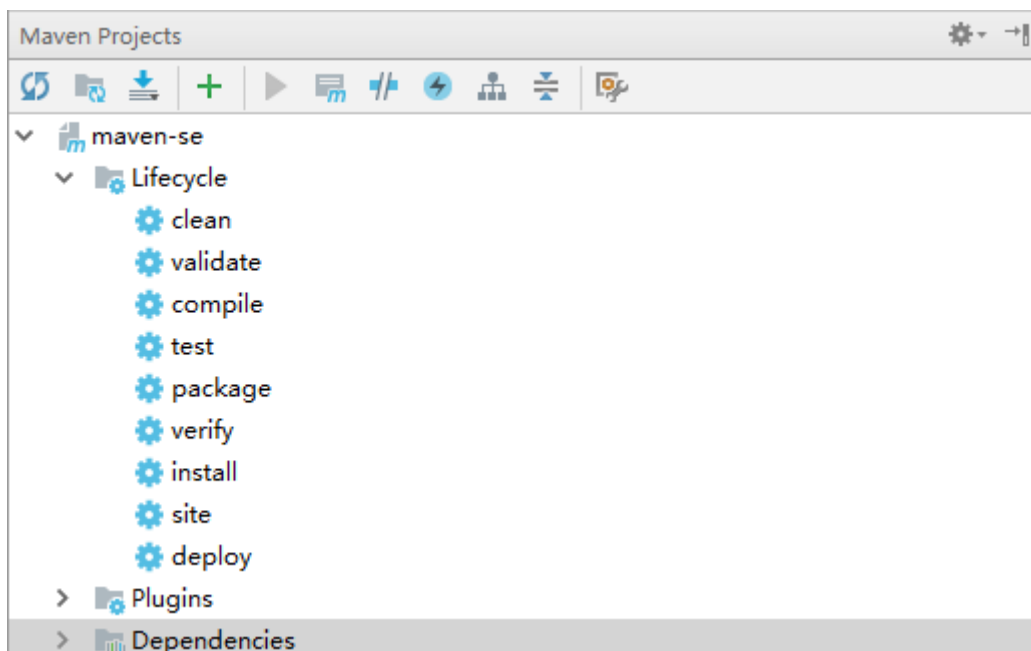
```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
```

```
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.4.5</version>
</dependency>
```

拷贝这里

#### 4、使用 Maven 命令

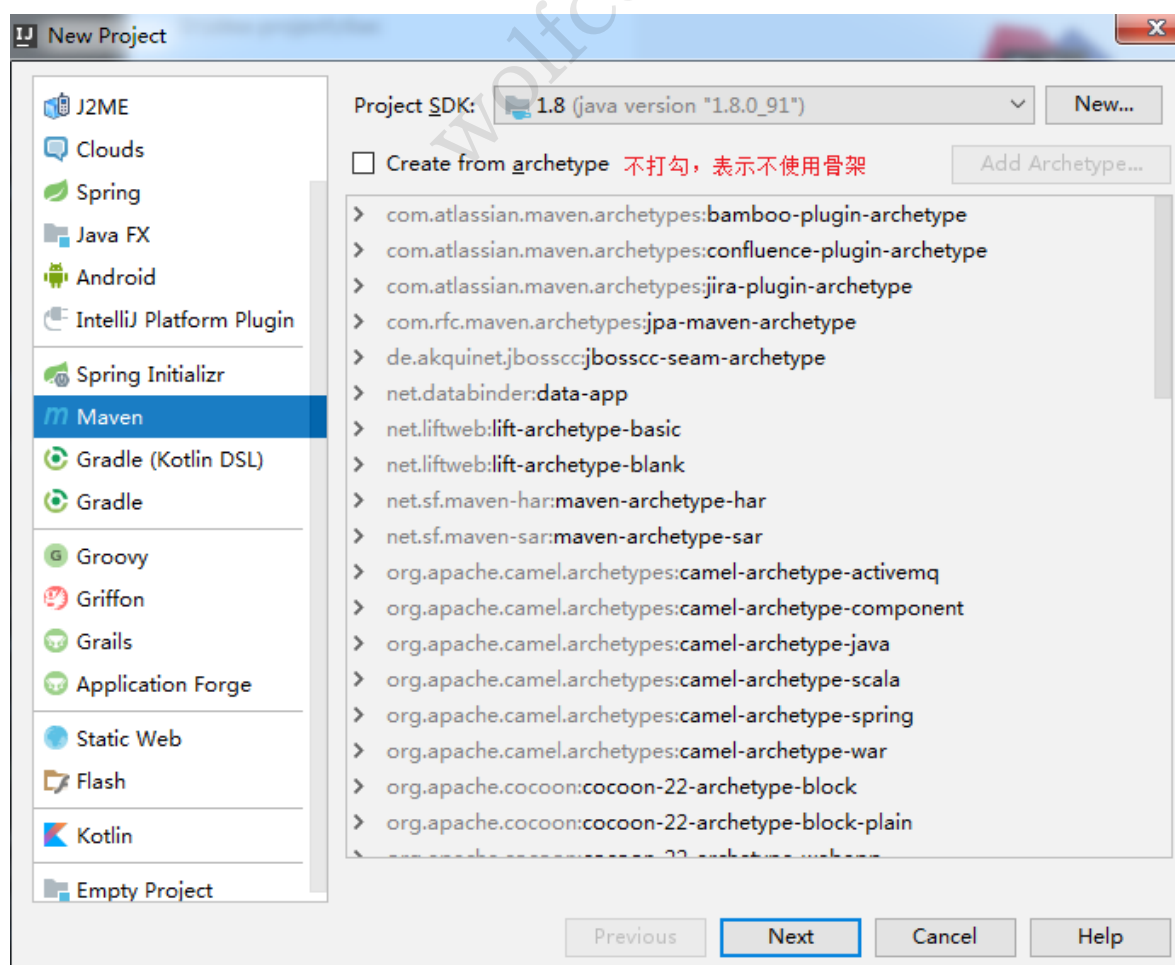
在 Maven Projects 窗口，双击对应命令即可：



## 七、搭建基于 Maven 构建的 JavaWeb 项目

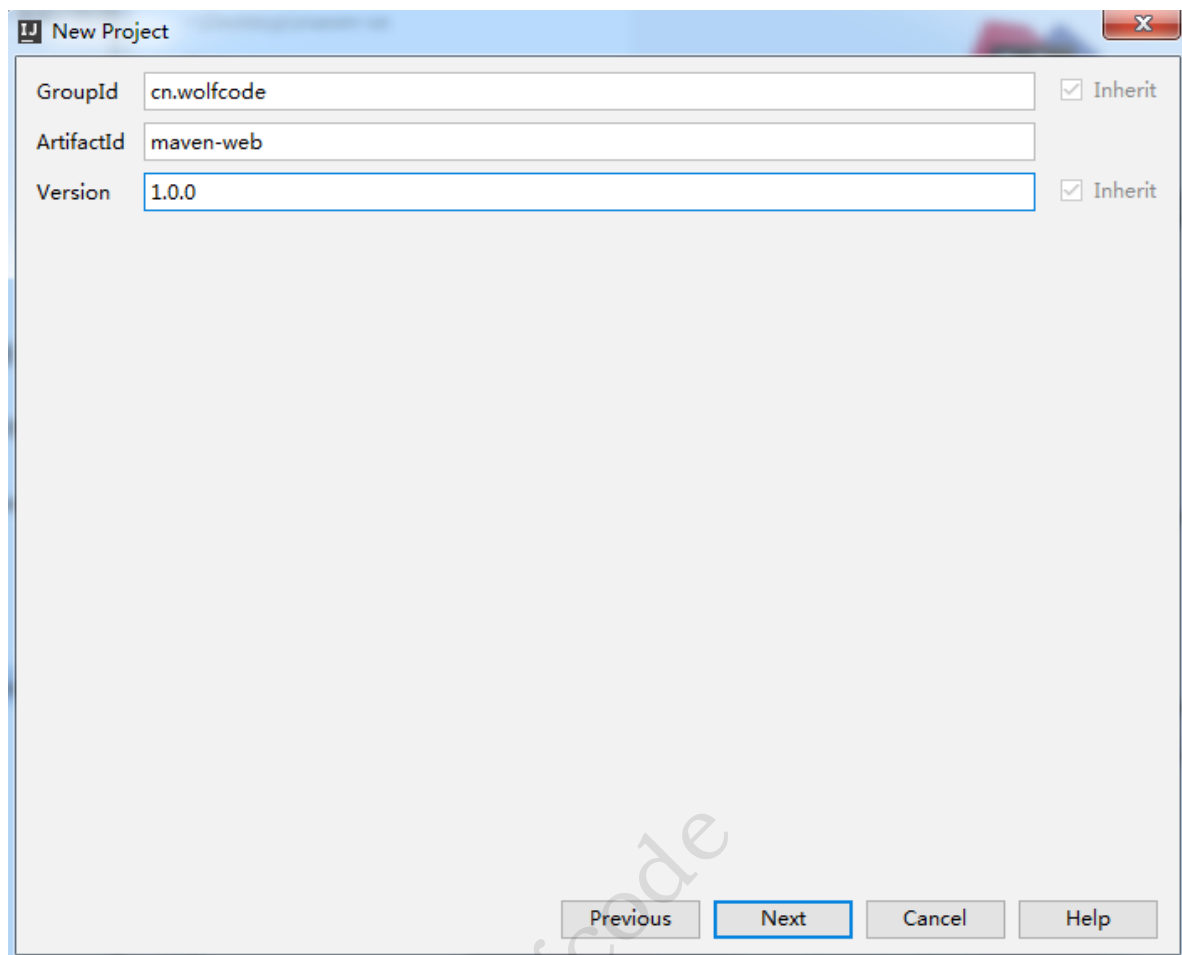
### 1、创建项目

#### 1.1、选择建的是 Maven 项目



确定后之后点击 Next。

## 1.2、填写对应项目信息



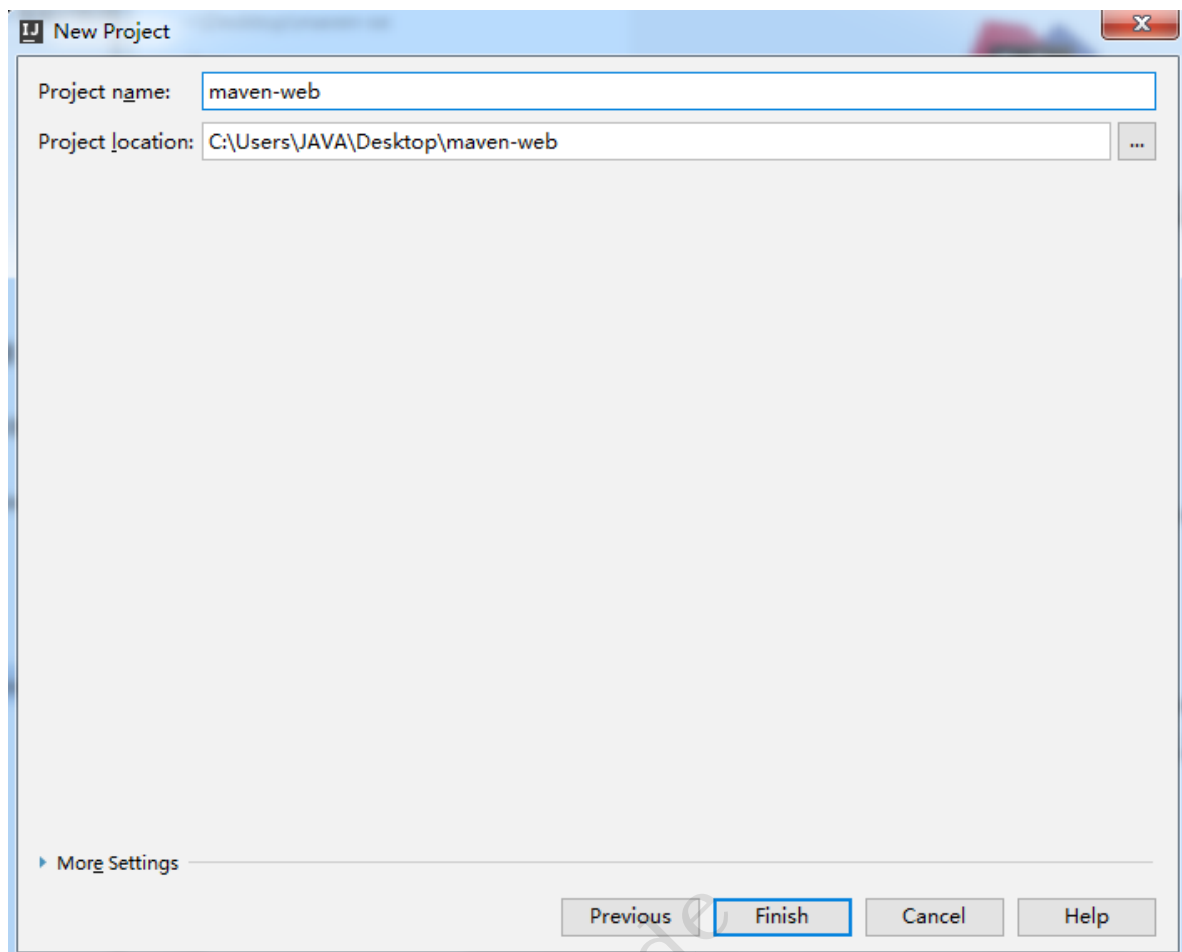
The image shows a 'New Project' dialog box with the following fields and options:

Field	Value	Inherit
GroupId	cn.wolfcode	<input checked="" type="checkbox"/>
ArtifactId	maven-web	<input checked="" type="checkbox"/>
Version	1.0.0	<input checked="" type="checkbox"/>

At the bottom of the dialog, there are four buttons: 'Previous', 'Next' (highlighted with a blue border), 'Cancel', and 'Help'.

填完之后点击 Next。

## 1.3、确定项目存储位置



确定完之后点击 Finish，之后工具会打开解析项目，要等一会儿。

## 1.4、修改项目打包方式

在 pom.xml 修改：

```
<?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>cn.wolfcode</groupId>
    <artifactId>maven-web</artifactId>
    <version>1.0.0</version>
    <!-- 修改项目打包方式 -->
    <packaging>war</packaging>
</project>
```

## 2、添加 web.xml

自己手动在项目的 main 目录下建 webapp/WEB-INF/web.xml 文件，拷贝如下内容：

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
</web-app>
```

### 3、修改项目编译运行版本

在 pom.xml 配置如下:

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>
```

### 4、添加依赖

在 pom.xml 文件添加如下依赖:

```
<dependencies>
  <!-- 配置 servlet-api 依赖 -->
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.0.1</version>
    <!-- 记得一定配置 -->
    <scope>provided</scope>
  </dependency>
</dependencies>
```

### 5、编写 Servlet 与 JSP

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
        req.setAttribute("msg", "你好 Maven");
        req.getRequestDispatcher("/WEB-INF/a.jsp").forward(req, resp);
    }
}
```

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Maven</title>
</head>
<body>
    ${msg}
</body>
</html>
```

## 6、配置 Tomcat 插件

---

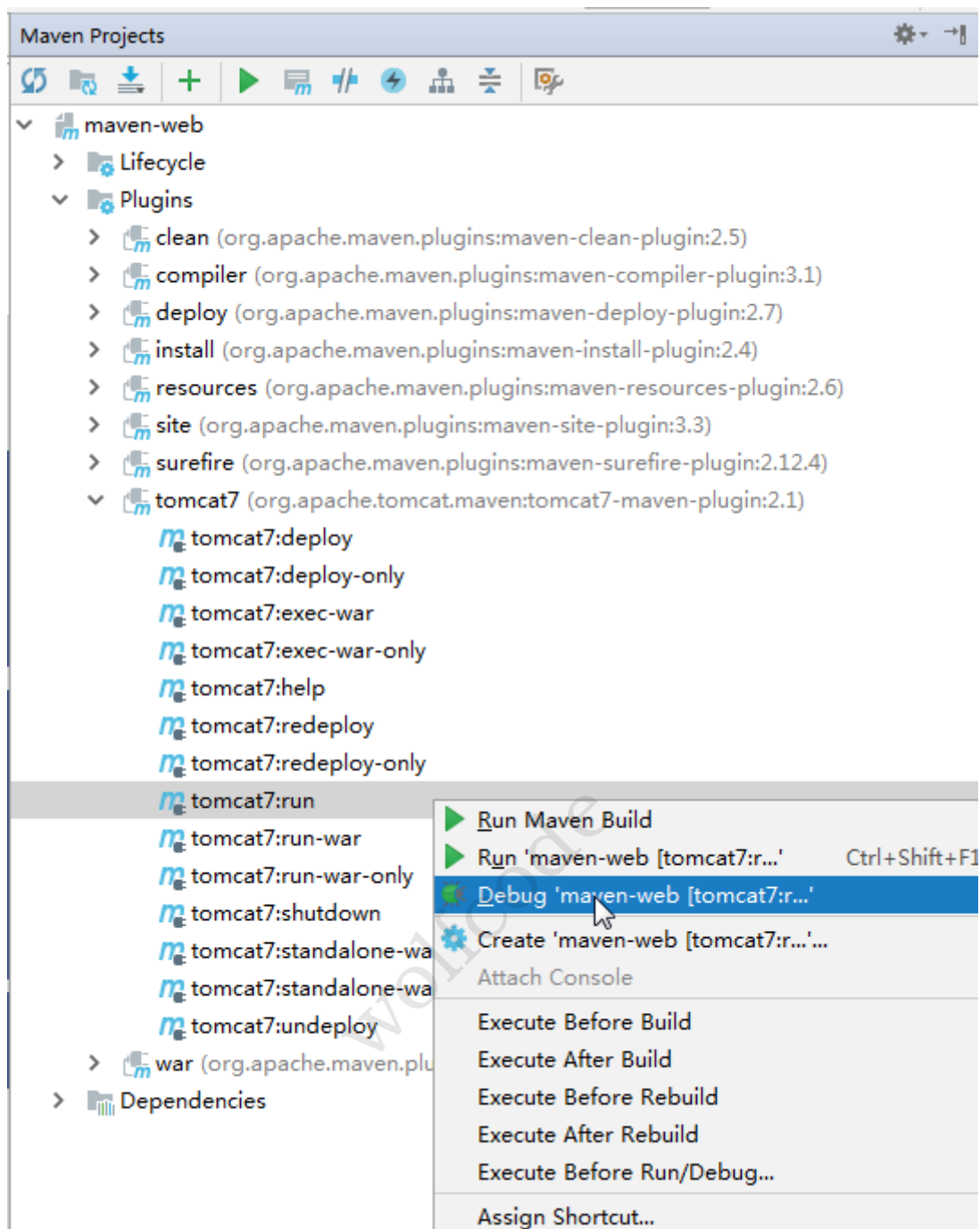
在 pom.xml 添加如下插件：

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.1</version>
            <configuration>
                <port>80</port> <!-- 端口 -->
                <path>/</path> <!-- 上下路径 -->
                <uriEncoding>UTF-8</uriEncoding> <!-- 针对 GET 方式乱码处理 -->
            </configuration>
        </plugin>
    </plugins>
</build>
```

## 7、启动项目

---





## 八、概念解释

### 1、坐标

#### 1.1、数学中的坐标

在空间中，使用 X、Y、Z 三个向量可以唯一的定位空间中的任意一个点。

#### 1.2、Maven 中的坐标

标识项目或者依赖在仓库中的位置。主要由下面构成：

- groupId：项目名称，一般使用公司或组织域名倒写，域名具有唯一性，区分不同公司；
- artifactId：项目中 Maven 项目/模块名称，区分同一个公司的项目；
- version：项目版本，区分同一个项目不同时期的特点。

## 2、依赖

配置项目所要依赖的 jar，在 pom.xml 使用 dependency 元素配置，通过坐标先从本地仓库找，若没有再从远程仓库找。

### 2.1、scope

Maven 在编译，测试，运行的时候，各需要一套 classpath（比如编译的时候会有编译的 classpath，测试的时候会有测试的 classpath）。scope 表明该依赖的项目和三个 classpath 的关系，即表示依赖什么阶段起作用，什么阶段不起作用。

scope 主要可选值：

- compile：默认，适用于所有阶段，会随着项目一起发布，在编译，测试，运行时都有效；
- provided：编译和测试阶段使用；典型的如 servlet-api.jar，打包时不需要，容器来提供；
- runtime：测试和运行阶段使用，用于接口和实现分离，典型的如 jdbc 具体驱动实现；
- test：测试阶段使用，不会随项目发布，如 junit。

	Compile classpath	Runtime classpath	Test classpath
compile	Y	Y	Y
provided	Y	X	Y
runtime	X	Y	Y
test	X	X	Y

### 2.2、传递性

Maven 的依赖是具有传递性的，比如 A -> B，B -> C，那么 A 间接的依赖于 C，这就是依赖的传递性，其中 A 对于 B 是第一直接依赖，B 对于 C 是第二直接依赖，C 为 A 的传递性依赖。依赖的传递性可以解决依赖繁琐的问题。