Maven

概述

1. Apache Maven是一个（特别是Java编程）项目管理及自动构建工具，由Apache软件基金会所提供。基于项目对象模型（缩写：POM）概念，Maven利用一个中央信息片断能管理一个项目的构建、报告和文档等步骤。
2. 项目管理工具：编译、测试、运行、打包（jar、war）、部署
3. 依赖管理
4. 下载

   地址：https://maven.apache.org/

	Link	Checksums	Signature
Binary tar.gz archive	apache-maven-3.8.1-bin.tar.gz	apache-maven-3.8.1-bin.tar.gz.sha512	apache-maven-3.8.1-bin.tar.gz.asc
Binary zip archive	apache-maven-3.8.1-bin.zip	apache-maven-3.8.1-bin.zip.sha512	apache-maven-3.8.1-bin.zip.asc
Source tar.gz archive	apache-maven-3.8.1-src.tar.gz	apache-maven-3.8.1-src.tar.gz.sha512	apache-maven-3.8.1-src.tar.gz.asc
Source zip archive	apache-maven-3.8.1-src.zip	apache-maven-3.8.1-src.zip.sha512	apache-maven-3.8.1-src.zip.asc

5. 安装、配置
   - 直接解压即可
   - 配置环境变量：
     - M2_HOME=>maven解压目录
     - 修改 path=》添加 %M2_HOME%\bin
   - 测试：mvn -v
6. maven仓库
   - 本地仓库：本地的一个文件夹
   - 中央仓库：世界唯一，由maven社区维护（网站）
   - 远程仓库：是位于web服务器上的一个私有仓库，由自己公司创建和维护
   - 镜像仓库：是中央仓库的镜像（副本），目的是加快依赖jar包的下载速度
7. 修改maven配置
   - 修改：maven安装目录下的 conf/settings.xml
     - 修改本地仓库位置：

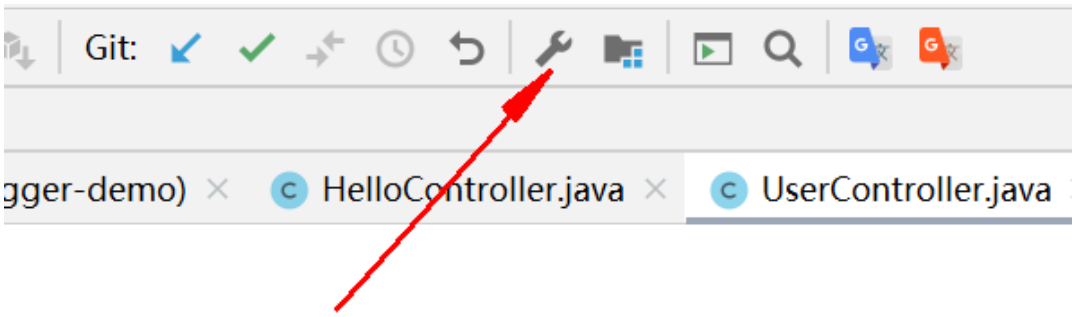
```
<localRepository>D:\mavenrepository</localRepository>
```

     - 配置阿里云镜像

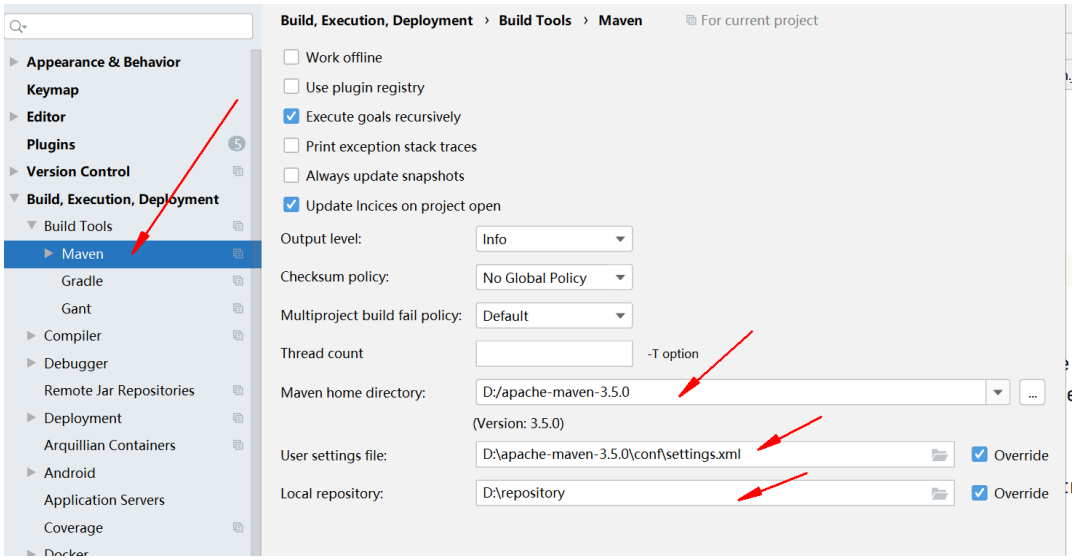
```
<mirrors>
    <mirror>
        <id>nexus-aliyun</id>
        <mirrorOf>central</mirrorOf>
        <name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>
    </mirror>
</mirrors>
```

8. 在 Idea 中配置

- 进入 settings

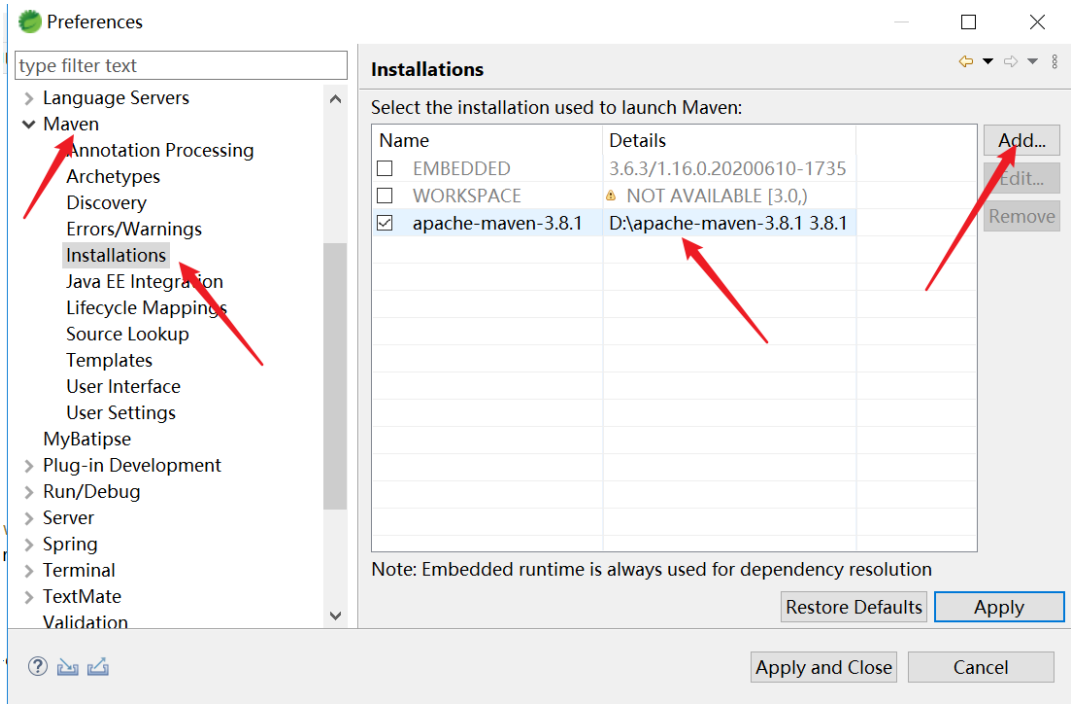


- 配置maven



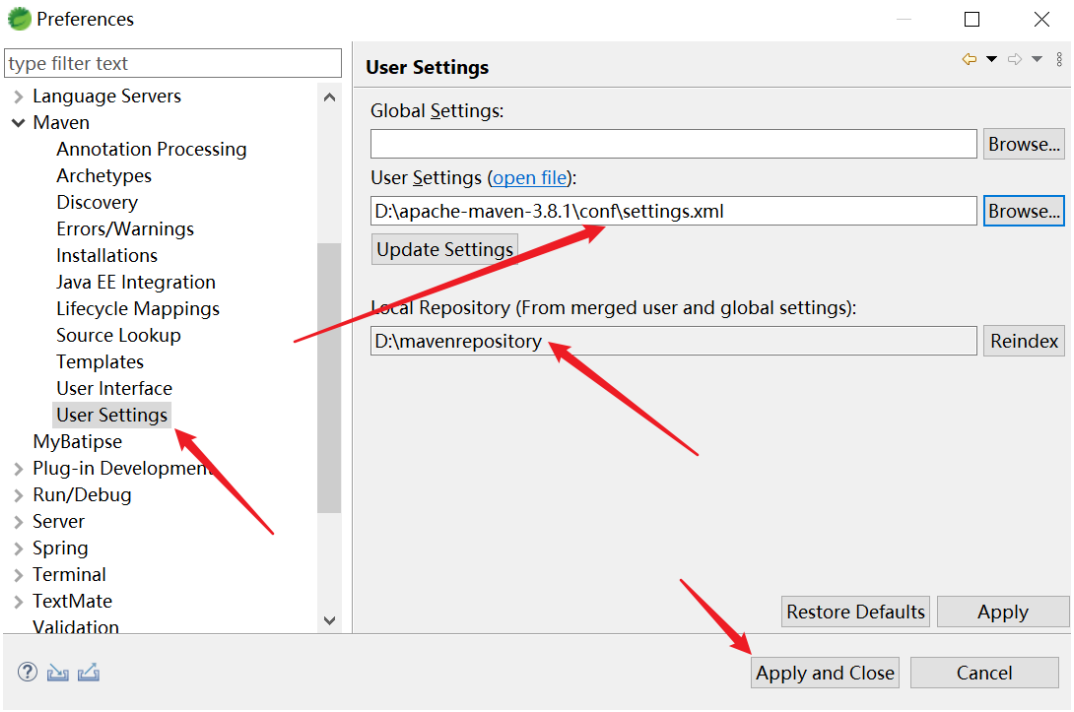
9. Eclipse配置配置

- 配置maven位置：



- 配置maven的配置文件：

  首选项=》maven=》User settings =》选择settings.xml文件



使用

1. maven的目录结构

   src：源程序目录

   main：源程序

   java：程序

   resources：资源文件(配置文件)

   webapp:web项目（不是web项目没有）

   test：测试代码

   target：项目生成的结果

2. 创建maven项目

   - maven project 向导
   - 配置三个坐标（定位唯一的jar包）
     - groupId：组织或公司的域名
     - artifactId：组件名（项目名）
     - version：版本号
   - 打包方式
     - jar（默认打包方式，控制台项目或window项目）
     - war（web项目）
     - pom（maven的管理项目）
   - 配置 pom.xml

```
1 <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
  https://maven.apache.org/xsd/maven-4.0.0.xsd">
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>com.neu</groupId>
4   <artifactId>test3</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
```

```
  <packaging>jar</packaging>

  <build>
        <plugins>
            <!-- 资源文件拷贝插件 -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-resources-plugin</artifactId>
                <version>2.7</version>
                <configuration>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
            <!-- 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>
            <!-- <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <path>/</path>
                    端口号
                    <port>8089</port>
                </configuration>
            </plugin> -->
            <!-- <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <version>2.1.1</version>
                <configuration>
                    <webResources>
                        <resource>
                            <excludes>
                                <exclude>**/WEB-
INF/web.xml</exclude>
                            </excludes>
                            <directory>src/main/webapp</directory>
                        </resource>
                    </webResources>
                   <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin> -->
        </plugins>
    </build>

    <!-- 依赖 -->
    <dependencies>
        <!-- 单元测试 -->
        <!-- https://mvnrepository.com/artifact/junit/junit -->
```

```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <!--
https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api
-->
        <!-- <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
            <scope>provided</scope>
        </dependency> -->
        <!--
https://mvnrepository.com/artifact/javax.servlet.jsp/jsp-api -->
        <!-- <dependency>
            <groupId>javax.servlet.jsp</groupId>
            <artifactId>jsp-api</artifactId>
            <version>2.2</version>
            <scope>provided</scope>
        </dependency> -->
    </dependencies>
</project>
```

3. 端口号冲突，解决方法

- 把原来使用该端口的程序关闭
  - 在eclipse中关闭
  - 在任务管理器中关闭 java 进程 或 javaw进程
- 修改当前tomcat服务的端口号，改成与之前冲突不一样即可

```
<port>8089</port>
```

4. web项目

- 打包方式

```
<packaging>war</packaging>
```

- idea中运行
  - maven面板中，选中项目
  - 在 plugins 中选择 tomcat7 =》tomcat7：run=》右键=》Run Maven Build
- eclipse中运行

  maven build

```
tomcat7:run
```

- 配置 tomcat插件

```xml
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
        <!-- 端口号 -->
        <port>8089</port>
    </configuration>
</plugin>
```

5. 从maven仓库中搜索并添加jar包
   - 进入maven中央仓库网站（https://mvnrepository.com/）
   - 在搜索栏中输入关键字
   - 在列表中查找需要的版本
   - 拷贝“maven”中内容到pom.xml的 中
6. 使用 JSTL

```xml
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>

<dependency>
    <groupId>taglibs</groupId>
    <artifactId>standard</artifactId>
    <version>1.1.2</version>
</dependency>

<dependency>
    <groupId>org.apache.taglibs</groupId>
    <artifactId>taglibs-standard-impl</artifactId>
    <version>1.2.5</version>
</dependency>
```

测试：

```xml
<%@ page contentType="text/html;charset=UTF-8" language="java"
%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
    <title>Title</title>
</head>
<body>
    你好!
    <c:forEach begin="1" end="10" var="pageNum">
        ${pageNum}
    </c:forEach>
</body>
</html>
```

生命周期

- clean：清除先前构建的artifacts（在maven中，把由项目生成的包都叫作artifact）。
- validate：验证工程是否正确，所有需要的资源是否可用。
- 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。这些测试不需要已打包和布署。
- Package：把已编译的代码打包成可发布的格式，比如jar。
- integration-test：如有需要，将包处理和发布到一个能够进行集成测试的环境。
- verify：运行所有检查，验证包是否有效且达到质量标准。
- install：把包安装在本地的repository中，可以被其他工程作为依赖来使用。
- deploy：在集成或者发布环境下执行，将最终版本的包拷贝到远程的repository，使得其他的开发者或者工程可以共享。
- site：为项目生成文档站点。