Maven

主讲：崔译

一、简介

Apache Maven 是一个 项目的 构建、管理工具，基于 POM (Project Object Model 项目对象模型) 的概念，管理整个项目的生命周期 (从创建到打包)

- 项目的构建
- 管理依赖 (jar 包)
- 打包、发布、部署、运行

二、Maven 的下载和安装

1、下载Maven

maven官网下载地址

注意：不同的Maven 版本对JDK版本有要求，3.5.x 要求jdk7+

2、安装

将下载的压缩包 解压缩到 **无中文、无空格的目录下**

建议放在 主目录中

```
# 1. 将安装包放在主目录下
# 2. 在主目录 右键  在此处打开终端
# 3. 进入主目录
cd
# 4. 创建文件夹
mkdir maven
# 5. 将压缩包拷贝到文件夹中
mv apache-maven-3.2.5-bin.tar.gz  maven
# 6. 解压缩文件夹
tar -zxvf apache-maven-3.2.5-bin.tar.gz
```

3、配置环境变量

3-1 依赖的环境变量

JAVA_HOME / classpath

3-2 配置Maven 环境变量

在 `~/.bashrc` 中末尾追加

```
# export PATH=~/maven/apache-maven-3.2.5/bin
export PATH=[maven安装目录]/bin:$PATH
export M2_HOME=[maven安装目录]
```

在终端中执行以下命令

```
source .bashrc
```

4、测试

```
mvn -version
```

三、相关概念

1、关于依赖管理

我们可以使用Maven 关系 项目依赖，在项目中 添加相关依赖的坐标，

Maven ：

1. 从`本地仓库` 查找 看是否存在 该依赖包
2. 如果不存在，并且配置了镜像仓库 ， 到`镜像仓库`查找该依赖包
3. 如果不存在，没有配置了镜像仓库，到`中央仓库`查找该依赖包
4. 如果找到了，将该依赖包下载到 `本地仓库`,并且添加到项目中

2、本地仓库

2-1 简介

是一个文件夹，用于存放maven 相关的所有的依赖和 插件

2-2 默认位置

- windos : `C:\Users\User\.m2`
- linux : `~/.m2`

2-3 修改本地仓库

在 M2_HOME/conf/settings.xml 中修改

```
<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">
  <localRepository>D:\maven\apache-maven-repositories</localRepository>
</settings>
```

3、中央仓库

是一个网络仓库，中央仓库地址，用于存放的是Maven 的相关资源（Maven 所有插件以及几乎所有的jar包）

4、镜像仓库

4-1 简介

是 中央仓库的 镜像， 镜像： 中央仓库的复制（代理）

作用： 提高下载速度

4-2 配置方式

在 M2_HOME/conf/settings.xml 中修改

```
<mirrors>
     <mirror>
      <id>alimaven</id>
      <mirrorOf>*</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public</url>
    </mirror>
</mirrors>
```

```
<mirrors>
     <mirror>
      <id>itanymaven</id>
      <mirrorOf>*</mirrorOf>
      <name>Human Readable Name for this Mirror.</name>
      <url>http://maven.itany.com/repository/maven-public/</url>
    </mirror>
</mirrors>
```

四、HelloWorld

1、创建java项目

1. 创建项目文件夹

   ```
   cd ~/maven
   mkdir project
   cd project
   ```

2. 创建项目

   ```
   mvn archetype:generate
   ```

3. 选择archetype

   选择的是7 `maven-archetype-quickstart` --> java 项目

4. 输入groupId

   输入 公司域名反向（包名）

5. 输入artifactId

   输入的是 项目名(模块名)

6. 输入version

   输入的是版本,默认`1.0-snapshot`

7. 打包、发布

```
#进入到项目根目录（pom.xml所在目录）
cd hello
# 修改App.java
# 将项目打包发布
mvn package
```

8. 运行测试代码

```
cd target/classes
java com.itany.App
```

2、创建web项目

1. 输入命令

```
# 命令是不允许有换行的，此处换行是为了方便阅读和加注释
mvn archetype:generate
-DarchetypeArtifactId=maven-archetype-webapp #指定使用的模板为web项目
-DgroupId=com.itany
-DartifactId=helloWeb
-DinteractiveMode=false #互动模式，设置为false
```

2. 打包，发布

```
#进入到项目根目录（pom.xml所在目录）
cd helloWeb
# 将项目打包发布
mvn package
```

3. 启动tomcat

```
#将war包放到tocmat/webapps
#进入到tomcat bin目录
cd apache-tomcat/bin
#启动tomcat
startup.bat 或者 ./startup.sh
```

4. 访问：`localhost:8080/helloWeb`

五、使用IDEA创建Maven项目

1、为IDEA配置Maven

`settings -- maven`

- Maven Home Directory : 指向自己安装的Maven
- User settings File ：勾选override，指向Maven/conf/settings.xml
- Local Repository : 勾选override

2、使用Maven创建Web项目

`project structer ---> 点击加号 --> Maven ---> 收集相关信息 --> OK`

3、配置Maven 的tocmat 插件

```
<!--在项目的pom.xml文件中-->
<!--删除pluginManagement标签-->
<!--为maven添加tomcat 插件-->
<plugin>
  <groupId>org.apache.tomcat.maven</groupId>
  <artifactId>tomcat7-maven-plugin</artifactId>
  <version>2.2</version>
  <configuration>
    <!--Application Context / pageContext.request.contextPath-->
    <path>/</path>
    <!--指定tomcat端口号-->
    <port>8080</port>
  </configuration>
</plugin>
```

4、修改Maven的编译版本

4-1 修改特定项目的编译版本

```
<!--在项目的pom.xml文件中-->
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

4-2 修改全局的编译版本

```
<!--在settiongs.xml文件中-->
<profile>
  <id>jdk-1.8</id>
  <activation>
    <activeByDefault>true</activeByDefault>
    <jdk>1.8</jdk>
  </activation>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>
  </properties>
</profile>
```

5、运行web项目

右上角Edit Configuration ---》 点击加号 ---》 选择 Maven --》 改名字

---》 Command Line --》 输入命令 `clean compile package tomcat7:run`

六、Maven的常用命令

所有命令必须在Maven项目根目录（POM文件所在目录）下执行

命令	作用	解释
mvn compile	编译	生成target目录，编译项目
mvn package	打包	生成 jar / war （打包前会自动执行compile）
mvn clean	清理	删除target目录
mvn install	安装	将jar/war 安装到 本地仓库中

七、pom.xml

1、坐标

```
<!--
     Maven 项目（模块）的 坐标
        三者每一项叫做一个 向量
        三项合在一起叫做一个坐标
        一个坐标唯一(全球唯一)的确定一个项目（模块）
                （指的是项目的最终形态 jar / war / pom）

     groupId: 项目组的id 一般情况下是 公司域名反向.项目名
     artifactId: 项目的模块名
            一个真实的完整的项目 会被拆分为多个模块(子项目)

            每个模块 有自己的 "任务"
```

```
      version  版本号

      如何通过坐标找到对应的jar包？
         仓库地址/groupid/artifactid/version/xxx-version.jar
   -->
<groupId>com.itany</groupId>
<artifactId>mavenweb</artifactId>
<version>1.0-SNAPSHOT</version>
```

坐标查询方式：[中央仓库的查询网站]

2、打包方式

```
<!--项目的打包方式
        1. war  ： java web 项目的打包方式
        2. jar  ： java 项目的打包方式
        3. pom  ： 该打包方式用于 继承 和  聚合，
                   以该方式打包的项目中不存在java代码，只有pom.xml
      默认打包方式为 jar
-->
<packaging>war</packaging>
```

3、properties

```
<!--
      定义全局的常量值，一般用于定义：
        1. 项目字符集
        2. 编译版本
        3. 依赖的jar包版本（手动定义，标签名任意）
   -->
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
  <junit.version>4.11</junit.version>
</properties>
```

4、dependencies

```
<!--项目的依赖（添加的jar包）的坐标-->
<dependencies>
  <dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.16</version>
  </dependency>

  <dependency>

    <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-webmvc</artifactId>
    <version>4.3.9.RELEASE</version>
  </dependency>
</dependencies>
```

5、dependency-scope

作用域：jar包的作用域

值	作用域	备注
test	作用于测试类，不参与打包和部署	junit
compile	默认值，作用于整个项目，并且参与打包部署	正常的jar
provided	编写源代码时有效，但是不参与打包部署	servlet-api.jar
runtime	运行时范围，在运行时有效，在编译时不需要，参与打包部署	mysql-connector.jar
system	系统，和SystemPath一起使用，指定本地路径下的jar包	oracle驱动包

```
<dependency>
  <groupId>oracle</groupId>
  <artifactId>ojdbc</artifactId>
  <version>14</version>
  <scope>system</scope>
  <systemPath>C:/Users/User/Desktop/ojdbc-14.jar</systemPath>
</dependency>
```

6、dependencyManagement

```
<!--父项目的pom-->
<!--依赖管理，只用来定义依赖
        需要在子项目中导入依赖
    -->
<dependencyManagement>
  <dependencies>

    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>5.1.16</version>
    </dependency>

  </dependencies>
</dependencyManagement>
```

```
<!--子项目的pom-->
 <dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
</dependency>
```

八、Maven项目目录结构

1、java项目

```
项目
|---src
    |---main
        |---java （相当于MyEclipse的src，放的是java类）
              1. 名字必须叫java（Maven规定的）
              2. 必须是 Sources Root （IDEA规定的）
        |---resources (相当于MyEclipse的src，放的是配置文件)
              1. 名字必须叫resources（Maven规定的）
              2. 必须是 Resources Root （IDEA规定的）
    |---test (用于放测试类和测试代码，可以删除)
        |---java
|---pom.xml
```

2、java web 项目

```
项目
|---src
    |---main
        |---java （相当于MyEclipse的src，放的是java类）
              1. 名字必须叫java（Maven规定的）
              2. 必须是 Sources Root （IDEA规定的）
        |---resources (相当于MyEclipse的src，放的是配置文件)
              1. 名字必须叫resources（Maven规定的）
              2. 必须是 Resources Root （IDEA规定的）
        |---webapp (相当于MyEclipse的 WebRoot)
            |---css
            |---js
            |---images
            |---各种静态资源
            |---WEB-INF
                |---pages
                |---web.xml
    |---test (用于放测试类和测试代码，可以删除)
        |---java
|---pom.xml
```

九、Maven大型项目项目架构

1、项目需求

> OA 系统 - 行政管理模块
>
> 员工：
>
> 登录，查看自己的考勤信息，查看工资，请假，调休....
>
> 管理员
>
> 登录，查看自己的考勤信息，查看他人的考勤信息，查看薪资，请假，调休，审核

该系统只是整个OA的 子系统，今后会添加更多系统功能

2、项目结构

> oa
>
> oa-sso (单点登录，统一身份认证)
>
> oa-common
>
> oa-entity (实体类)
>
> oa-dao （ 数据库访问 ）
>
> oa-utils (通用类 通用代码)
>
> oa-dept
>
> oa-front
>
> oa-front-page
>
> oa-front-rest (controller)
>
> oa-front-service
>
> oa-back
>
> oa-back-web (controller , page)
>
> oa-back-service

十、继承和聚合

1、继承

1-1 概念和要求

Maven 项目 可以被另一个Maven项目继承，**继承的是 POM 文件**

- 父项目的打包方式必须是 `pom`
- 子项目中使用`parent` 标签指定父项目的坐标

- 如果子项目的坐标向量值 和 父项目一致， 可以省略不写

```
<parent>
  <artifactId>oa-parent</artifactId>
  <groupId>com.itany.oa</groupId>
  <version>1.0</version>
  <relativePath>../oa-parent/pom.xml</relativePath>
</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>oa-common</artifactId>
```

1-2 特性

依赖的传递发现

> 假如： C 项目 继承 B 项目 ， B 项目 继承了 A 项目。
>
> 在 A 项目 添加一个依赖
>
> 此时，在B 项目 和 C 项目中都可以使用 该依赖

2、聚合

将多个项目，聚合到同一个项目A 中， 通过对A的操作，自动的批量的对其他所有聚合项目进行相同操作

- 对于A 项目，打包方式 必须是 `pom`