Maven01

笔记本:	Maven		
创建时间:	2018/10/12 17:10	**更新时间:**	2018/10/14 22:27
作者:	155307642@qq.com		
URL:	file:///C:/Users/15530/AppData/Local/Temp/Rar$DIa3140.7514/2-Hello%20Maven.doc		

Maven项目管理工具

一、Maven简介

1、什么是Maven
Maven是跨平台的项目管理工具。主要服务鱼基于Java平台的项目构建，依赖管理和项目信息管理。
2、什么是理想的项目构建
清理--->编译--->测试--->报告--->打包--->部署
高度自动化，跨平台，可重用的组件，标准化的
3、什么是依赖？为什么要进行依赖管理?
自动下载，统一依赖管理
4、有哪些项目信息
项目名称描述等，开发人员信息，开发者信息
5、项目构建过程

二、为什么使用Maven

不使用Maven:
1、手工操作较多，编译、测试、部署等工作都是独立的，很难一步完成
2、每个人的IDE配置都不同，很容易出现本地代码换个地方编译就出错
使用Maven:
1、拥有约定，知道你的代码在哪里，放到哪里去
2、拥有一个生命周期，例如执行 mvn install就可以自动执行编译，测试，打包等构建过程
3、只需要定义一个pom.xml，然后把源代码放到默认目录，Maven帮你处理其他事情
4、拥有依赖管理，仓库管理

三、Maven的安装与配置

1、确认jdk是否安装
http://maven.apache.org/docs/history.html

Maven3.3.X及以上版本至少需要jdk1.7的支持。
2、去官网http://maven.apache.org/download.cgi
3、下载 apache-maven-3.5.3-bin.zip
4、解压
目录结构:
bin：命令
boot：maven启动所需
conf：配置文件（常用的setting.xml就在其中）
lib：maven所依赖的jar包
5、环境变量
环境变量配置和JDK配置一样，如：M2_HOME/MAVEN_HOME和Path

四、第一个Maven项目

1、Maven规定了一套默认的项目格式:
src/main/java —— 存放项目的.java文件
src/main/resources —— 存放项目资源文件，如spring、struts2配置文件，db.properties
src/main/webapp —— 存放jsp，css，image等文件

src/test/java —— 存放所有测试.java文件，如JUnit测试类
src/test/resources —— 测试资源文件
pom.xml——主要要写的maven配置文件
target —— 项目由maven自动输出位置

2、按照下面的目录结构创建一个目录结构：

--src
-----main
----------java
----------resources
-----test
----------java
----------resources
--target
--pom.xml

3、编写下面的Java类：

1）在src/main/java/com/luban/maven目录下新建文件Hello.java：

```
package com.luban.maven;
public class Hello {
	public String sayHello(String name){
		return "Hello"+name;
	}
}
```

2）在src/test/java/com/luban/maven目录下新建文件HelloTest.java：

```
package com.luban.maven;
import org.junit.Test;
import static junit.framework.Assert.*;
public class HelloTest {
	@Test
	public void testHello(){
		Hello hello = new Hello();
		String results = hello.sayHello("World");
		assertEquals("HelloWorld!",results);
	}
}
```

4，编辑pom.xml

```
<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/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.luban.maven</groupId>
	<artifactId>HelloWorld</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>HelloWorld</name>
	<dependencies>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>4.12</version>
			<scope>test</scope>
		</dependency>
	</dependencies>
</project>
```

5，切换到项目根目录下，分别执行下列命令并观察：
1）执行 mvn compile命令
2）执行mvn clean命令
3）执行mvn clean compile命令
4）执行mvn clean test命令
5）执行mvn clean package命令
6）执行mvn clean site命令
7）执行mvn install命令

mvn -v：查看版本
mvn compile：编译
mvn test：测试
mvn package：打包
mvn install：打包并拷贝到本地仓库

五、第二个Maven项目

1、新建第二个项目模块HelloFriend目录及约定的目录结构：

HelloFriend
--src
-----main
----------java
----------resources
-----test
----------java
----------resources
--target
--pom.xml

3、编写下面的Java类：

1）在src/main/java/com/luban/maven目录下新建文件HelloFriend.java：

```
package com.luban.maven;
import com.luban.maven.Hello;

public class HelloFriend{
    public String sayHelloToFriend(String name){
        Hello hello = new Hello();
        String str = hello.sayHello(name)+" I am "+this.getMyName();
        System.out.println(str);
        return str;
    }
    public String getMyName(){
        return "Fred";
    }
}
```

2）在src/test/java/com/luban/maven目录下新建文件HelloFriendTest.java：

```
package com.luban.maven;
import static junit.framework.Assert.assertEquals;
import org.junit.Test;
import com.luban.maven.Hello;

public class HelloFriendTest{
    @Test
    public void testHelloFriend(){
        HelloFriend helloFriend = new HelloFriend();
        String results = helloFriend.syaHelloToFriend("jack");
        assertEquals("Hellojack I am Fred",results);
```

```
    }
}
```

3）编辑pom.xml

```
<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/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.luban.maven</groupId>
    <artifactId>HelloWorld</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>HelloWorld</name>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.luban.maven</groupId>
            <artifactId>Hello</artifactId>
            <version>0.0.1-SNAPSHOT</version>
            <scope>compile</scope>
        </dependency>
    </dependencies>
</project>
```

六、Maven相关概念介绍

1、项目对象模型（POM）

2、坐标

- 什么是坐标
  - 在平面几何中坐标（x，y）可以标识平面中唯一的一点
- Maven坐标主要组成
  - groupId：定义当前Maven项目隶属项目
  - artifactId：定义实际项目中的一个模块
  - version：定义当前项目的当前版本
  - packaging：定义该项目的打包方式
- Maven为什么使用坐标
  - Maven世界拥有大量构建，我们需要找一个用来标识

3、依赖管理

1、依赖声明

```
<dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
 </dependency>
```

2、依赖范围

依赖范围scope用来控制依赖和编译、测试、运行的classpath的关系

compile：默认编译依赖范围。对于编译、测试、运行三种classpath都有效。

test：测试依赖范围。只对于测试classpath有效。

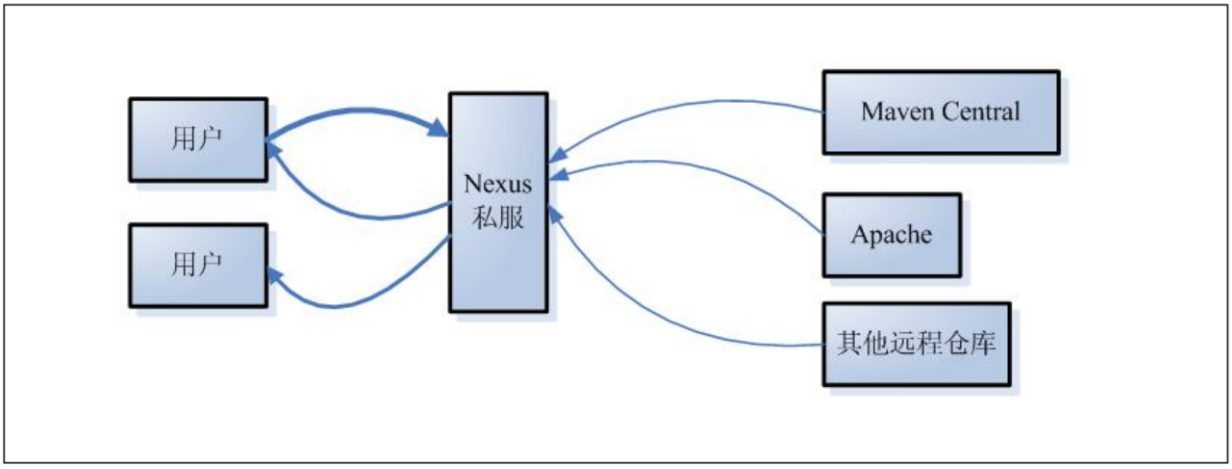
provided：已提供依赖访问。对于编译、测试的classpath有效，但对于运行无效，因为容器已经提供，例如servlet-api

runtime：运行时提供。例如jdbc驱动。

依赖范围（Scope）	对于主代码classpath有效	对于测试代码classpath有效	被打包，对于运行时classpath有效	例子
compile	Y	Y	Y	log4j
test	-	Y	-	junit
provided	Y	Y	-	servlet-api
runtime	-	-	Y	JDBC Driver Implementation

4、仓库管理

- 何为Maven仓库
  - 用来统一存储所有Maven共享构建的位置就是仓库
- Maven仓库布局
  - 根据Maven坐标定义每个构建在仓库中唯一存储路径
  - 大致为：groupId/artifactId/version/artifactId-version.packaging
- 仓库的分类
  - 本地仓库
    - ~/.m2/repository/
    - 每个用户只有一个本地仓库
  - 远程仓库
    - 中央仓库：Maven默认的远程仓库http://repo1.maven.org/maven2/
    - 私服：是一种特殊的远程仓库，它是架设在局域网内的仓库
    - 镜像：用来替代中央仓库，速度一般比中央仓库



5、生命周期

- 何为生命周期
  - Maven生命周期就是为了对所有的构建过程进行抽象和统一
  - 包括项目清理、初始化、编译、打包、测试、部署等几乎所有构建步骤
- Maven三大生命周期
  - clean：清理项目的
  - default：构建项目的
  - site：生成项目站点的

6、仓库

1、本地仓库

设置本地仓库：conf/setting.xml

```
<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
   | The path to the local repository maven will use to store artifacts.
   |
   | Default: ${user.home}/.m2/repository
  <localRepository>/path/to/local/repo</localRepository>
  -->
    <localRepository>D:/soft/repo</localRepository>
  <!-- interactiveMode
   | This will determine whether maven prompts you when it needs input. If set to false,
   | maven will use a sensible default value, perhaps based on some other setting, for
   | the parameter in question.
   |
   | Default: true
  <interactiveMode>true</interactiveMode>
  -->
```

作业：

使用手动方式搭建一个Maven项目，并执行编译、清理、测试、打包、安装命令。