Maven and Ant

Ant是软件构建工具，Maven的定位是软件项目管理和理解工具。Maven除了具备Ant的功能外，还增加了以下主要的功能：

1）使用Project Object Model来对软件项目管理；

2）内置了更多的隐式规则，使得构建文件更加简单；

3）内置依赖管理和Repository来实现依赖的管理和统一存储；

4）内置了软件构建的生命周期；

一 POM（Project Object Model）与项目管理

每一个Maven工程都包含一个pom.xml文件，其他存储了该工程相关的信息，从而达到一定的项目管理的功能。例如包含了工程的配置，缺陷跟踪系统信息，工程的组织，许可协议，工程的路径，依赖等信息。

典型的pom.xml如下：

<project … >  
<modelVersion>4.0.0</modelVersion>

<!-- The Basics -->  
<groupId>...</groupId>  
< artifactId>...</artifactId>  
< version>...</version>  
< packaging>...</packaging>  
< dependencies>...</dependencies>  
< parent>...</parent>  
< dependencyManagement>...</dependencyManagement>  
< modules>...</modules>  
< properties>...</properties>

<!-- Build Settings -->  
<build>...</build>  
< reporting>...</reporting>

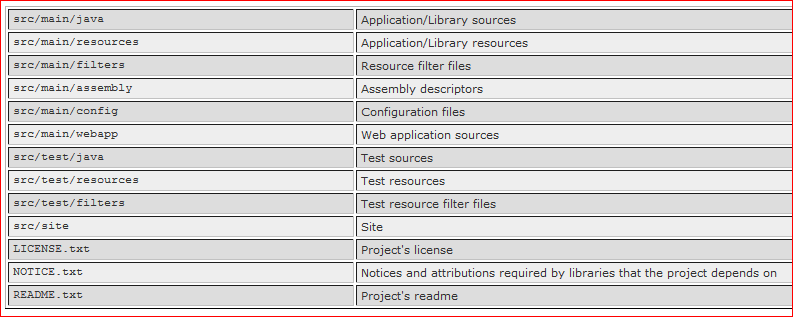
<!-- Project Meta Data -->  
<name>...</name>  
< description>...</description>  
< url>...</url>  
< inceptionYear>...</inceptionYear>  
< licenses>...</licenses>  
< organization>...</organization>

<developers>...</developers>  
< contributors>...</contributors>

<!-- Environment -->  
<issueManagement>...</issueManagement>  
< ciManagement>...</ciManagement>  
< mailingLists>...</mailingLists>  
< scm>...</scm>  
< prerequisites>...</prerequisites>  
< repositories>...</repositories>  
< pluginRepositories>...</pluginRepositories>  
< distributionManagement>...</distributionManagement>  
< profiles>...</profiles>  
< /project>

二 隐形的规则和简单的构建文件

Maven工程的目录结构必须为如下的结构



Maven还有内置的构建生命周期，内置定义了build，test，package，deploy等task。

由于Maven工程目录的规则和内置的构建生命周期，从而使得构建文件简单，例如如下的构建文件中甚至没有出现build，package等task的定义，但是我们已经可以调用wvm package等内置的task了：

<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>com.mycompany.app</groupId> <artifactId>my-app</artifactId> <packaging>jar</packaging> <version>1.0-SNAPSHOT</version> <name>Maven Quick Start Archetype</name> <url>http://maven.apache.org</url> <dependencies> <dependency> <groupId>junit</groupId> <artifactId>junit</artifactId> <version>3.8.1</version> <scope>test</scope> </dependency> </dependencies> </project>

三 依赖管理和Repository

Maven的dependence management用来管理所有此project的dependences，且在dependence repository中自动查找和下载dependence。

1） 依赖管理

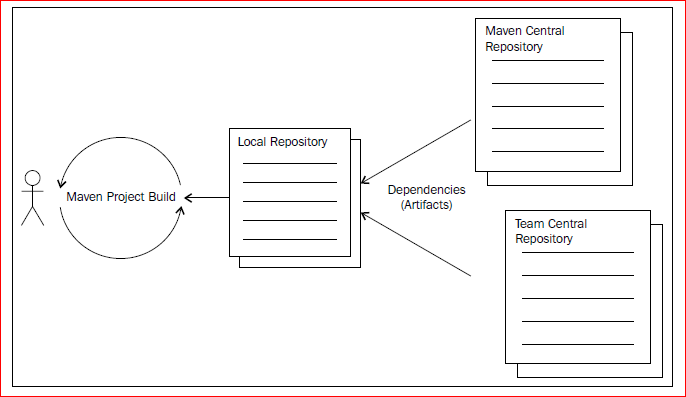
例如工程以来MySQL如下：

<dependencyManagement>  
< dependencies>  
< dependency>  
< groupId>mysql</groupId>  
< artifactId>mysql-connector-java</artifactId>  
< version>5.1.2</version>  
< /dependency>  
< dependencies>  
< /dependencyManagement>

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

2）dependence repository

更好的办法是公司或每个team有自己的repository，例如下图：



四 构建生命周期的定义

构建生命周期显式地定义了构建，测试，和发布的过程，是每个Maven工程的核心。Maven包含了3个内置的生命周期：default，clean和site。

1）default生命周期处理了工程的编译，测试和部署，他一共包含20多个阶段，主要的阶段如下：

Validate: 验证所有的工程信息是否可用且正确

Compile: 编译源代码

Test: 在一套framework下运行单元测试

Package: 以发布的格式打包编译的代码

Integration-test: 在集成测试环境中处理（部署）发布包

Verify: 检测发布包是否正确可用  
Install: 在本地的repository上安装发布包  
Deploy: 在远程的repository上安装发布包

以上的阶段具有先后顺序，执行某个阶段时，此阶段前的所有阶段都会被自动地执行。

2）clean生命周期处理工程的清理工作，包含3个阶段：pre-clean, clean, post-clean。

3）site生命周期处理工程site文档的生成和部署，包含下列阶段：

pre-site， site， post-site 和site-deploy，其中site-deploy用来将site文档部署到指定的web server上。

完！