**1.1.1 什么是 Maven**

Maven 的正确发音是[?mev?n]。

Maven 在美国是一个口语化的词语，代表专家、内行的意思。 一个对 Maven 比较正式的定义是这么说的：Maven 是一个项目管理工具，

它包含了一个项目对象模型 (POM：Project Object Model)，一组标准集合，一个项目生命周期(Project Lifecycle)，一个依赖管 理

系统(Dependency Management System)，和用来运行定义在生命周期阶段(phase)中插件(plugin)目标 (goal)的逻辑

**1.1.2 Maven 能解决什么问题**

能帮你构建工程，管理 jar 包，编译代码，还能帮你自动运行单元测试，打包，生成报表，甚至能帮你部署项目，生成 Web 站点。

**1.2.1 Maven 的依赖管理**

Maven 的一个核心特性就是依赖管理。当我们涉及到多模块的项目（包含成百个模块或者子项目），管理依赖就变成 一项困难的任务。

Maven 展示出了它对处理这种情形的高度控制。



通过分析发现：maven 工程中不直接将 jar 包导入到工程中，而是通过在 pom.xml 文件中添加所需 jar 包的坐标，这样就很好的避免了 jar 直接引入进来，在需要用到 jar 包的时候，只要查找 pom.xml 文 件，再通过 pom.xml 文件中的坐标，到一个专门用于”存放 jar 包的仓库”(maven 仓库)中根据坐标从 而找到这些 jar 包，再把这些 jar 包拿去运行.

**2.2 Maven 仓库**

maven 的工作需要从仓库下载一些 jar 包，如下图所示，本地的项目 A、项目 B 等都会通过 maven 软件从远程仓库（可以理解为互联网上的仓库）下载 jar 包并存在本地仓库，本地仓库 就是本地文 件夹，当第二次需要此 jar 包时则不再从远程仓库下载，因为本地仓库已经存在了，可以将本地仓库 理解为缓存，有了本地仓库就不用每次从远程仓库下载了.

本地仓库 ：用来存储从远程仓库或中央仓库下载的插件和 jar 包，项目使用一些插件或 jar 包， 优先从本地仓库查找

远程仓库：如果本地需要插件或者 jar 包，本地仓库没有，默认去远程仓库下载。 远程仓库可以在互联网内也可以在局域网内。

中央仓库 ：在 maven 软件中内置一个远程仓库地址 <http://repo1.maven.org/maven2> ，它是中 央仓库，服务于整个互联网，它是由 Maven 团队自己维护，里面存储了非常全的 jar 包，它包 含了世界上大部分流行的开源项目构件。

**2.2.3 全局 setting 与用户 setting**

maven 仓库地址、私服等配置信息需要在 setting.xml 文件中配置，分为全局配置和用户配置。 在 maven 安装目录下的有 conf/setting.xml 文件，此 setting.xml 文件用于 maven 的所有 project 项目，它作为 maven 的全局配置。

**2.3.1 Maven 工程的目录结构**

作为一个 maven 工程，它的 src目录和 pom.xml 是必备的。 进入 src目录后，我们发现它里面的目录结构如下

src/main/java —— 存放项目的.java 文件

src/main/resources —— 存放项目资源文件，如 spring, hibernate 配置文件 src/test/java —— 存放所有单元测试.java 文件，如 JUnit 测试类

src/test/resources —— 测试资源文件

target —— 项目输出位置，编译后的class 文件会输出到此目录

pom.xml——maven 项目核心配置文件



**第3章 Maven 常用命令**

**3.1.1 compile**

compile 是 maven 工程的编译命令，作用是将 src/main/java 下的文件编译为 class 文件输出到 target 目录下。

3.1.2 test

test 是 maven 工程的测试命令 mvn test，会执行src/test/java下的单元测试类。

3.1.3 clean

clean 是 maven 工程的清理命令，执行 clean 会删除 target 目录及内容。

3.1.4 package

package 是 maven 工程的打包命令，对于 java 工程执行 package 打成 jar 包，对于web 工程打成war 包。

3.1.5 install

install 是 maven 工程的安装命令，执行 install 将 maven 打成 jar 包或 war 包发布到本地仓库。

3.1.6 Maven 指令的生命周期

maven 对项目构建过程分为三套相互独立的生命周期，请注意这里说的是“三套”，而且“相互独立”， 这三套生命周期分别是：

Clean Lifecycle 在进行真正的构建之前进行一些清理工作。

Default Lifecycle 构建的核心部分，编译，测试，打包，部署等等。

Site Lifecycle 生成项目报告，站点，发布站点。

**3.2 坐标定义**

在 pom.xml 中定义坐标，内容包括：groupId、artifactId、version，详细内容如下：

<!--项目名称，定义为组织名+项目名，类似包名-->

<groupId>cn.itcast.maven</groupId>

<!-- 模块名称 -->

<artifactId>maven-first</artifactId>

<!-- 当前项目版本号，snapshot 为快照版本即非正式版本，release为正式发布版本 --> <version>0.0.1-SNAPSHOT</version>

<packaging > ：打包类型

jar：执行 package 会打成 jar 包

war：执行 package 会打成 war 包

pom ：用于 maven 工程的继承，通常父工程设置为 pom

**3.3 pom 基本配置**

pom.xml 是 Maven 项目的核心配置文件，位于每个工程的根目录，基本配置如下：

<project > ：文件的根节点 .

<modelversion > ： pom.xml 使用的对象模型版本

<groupId > ：项目名称，一般写项目的域名

<artifactId > ：模块名称，子项目名或模块名称

<version > ：产品的版本号 .

<packaging > ：打包类型，一般有 jar、war、pom 等 <name > ：项目的显示名，常用于 Maven 生成的文档。

<description > ：项目描述，常用于 Maven 生成的文档

<dependencies> ：项目依赖构件配置，配置项目依赖构件的坐标 <build> ：项目构建配置，配置编译、运行插件等