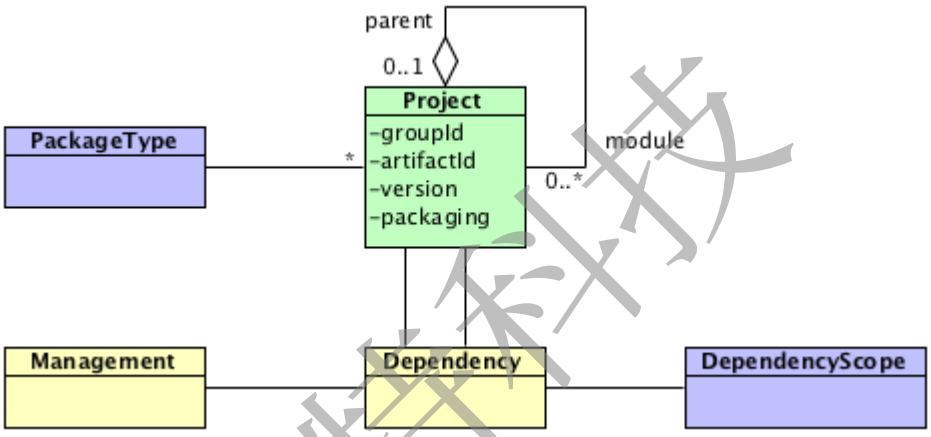
Maven核心概念

> ## 本节目标
>
> 1.了解Maven核心概念 2.熟悉Maven的常用操作

1. Maven管理目标

Maven的管理目标是工程，它是一个软件工程管理工具。 对于maven来说，一个软件工程的唯一标识是由组编号（groupId)、构件编号（artifactId)、版本信息（version） 共同决定的。每个工程都有一个打包类型，可以是jar, war, ear 或 pom。打包类型决定了工程最终产物的类型， 其中pom类型用于构建多模块工程。工程之间有两种关系：继承和聚合。



1.1 依赖关系

依赖关系的管理是maven最精髓的地方。一个工程可以依赖多个其它工程，通过工程的唯一标识（groupId+artifactId+version)可以明确指明依赖的库及版本，而且能够处理 依赖关系的传递。maven可以指定依赖的作用范围（scope），包括以下几种：

scope	编译阶段	测试阶段	运行阶段	备注
compile	V	V	V	默认scope
test		V		只在测试期依赖，如junit包
provided	V	V		运行期由服务器提供，如servlet-api包
runtime		V	V	编译期间不需要直接引用
system	V	V		编译和测试时由本机环境提供

maven的依赖关系管理解决了以下重要问题：

- 工程依赖的版本管理
- 工程依赖中同一个工程的多版本依赖的冲突问题
- 提供了标准的依赖关系定义

1.2 聚合与继承

1.2.1 聚合

聚合：上文提到的pom类型的工程用于构建多模块的工程，这体现了project之间的一种聚合关系， 将一系列小的模块聚合成整个产品。通过聚合后的工程可以同时管理每个相关模块的构建、清理、文档生成等工作。

1.2.2 继承

继承：上文提到的pom类型的工程同样可以定义多模块的工程配置信息，其作为多模块工程的父模块被多模块继承，此时多模块就是该pom类型工程的子工程，子工程可以复用父工程的配置信息。

1.2.3 聚合vs继承

- 区别
  - 对于聚合模块来说，它知道有哪些被聚合的模块，但被聚合模块相互独立。
  - 对于继承关系的父POM来说，它不知道有哪些子模块继承它，但那些子模块都必须知道自己的父POM。
- 共同点
  - 聚合 POM与继承关系中的父POM的packaging都是pom。
  - 聚合模块与继承关系中的父模块除了POM之外都没有实际的内容。

> 备注：课堂上通过创建项目案例讲述

2. Maven核心概念

2.1 POM

POM是指`project object Model`。pom是maven工作的基础，在执行goal时，maven会去项目根目录下读取pom.xml，获得需要的配置信息。pom文件中包含了项目管理和构建的所有信息。

2.2 构件

构建就是一个项目将要产生的文件，可以是jar文件，源文件，二进制文件，war文件，甚至是pom文件。每个构件都由`groupId:artifactId:version`组成的标识符唯一识别。

- groupId：通常使用创建构件的组织名称或者域名的反转（比如：org.springframework）
- artifactId：是构件的名称，通常在一个组织创建的构件中是唯一的（比如：spring-core）
- version：就比较好理解了，构件的版本信息，随着构件的升级版本号依次升级（比如：4.0.9.RELEASE）

同一个构件的不同版本：

```
<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>4.0.9.RELEASE</version>

<groupId>org.springframework</groupId>
<artifactId>spring-core</artifactId>
<version>4.1.0.RELEASE</version>
```

不同构件：

```
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.4.0</version>

<groupId>com.google.code.gson</groupId>
<artifactId>gson</artifactId>
<version>2.6.2</version>
```

2.3 仓库

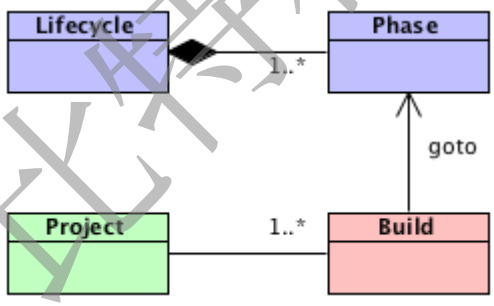
仓库主要用于获取工程依赖的其它工程的生成物，也可用来部署maven工程的生成物。 生成物包括各种打包的生成物以及pom文件。如果有必要一个工程可以部署到多个仓库。

仓库可以分为本地库（local）和远程库（remote）。本地库通常位于本机的~/.m2/repository文件夹(可以通过settings.xml中的localRepository进行修改)， 远程库最常见的是maven中央库，此外也会有一些私服库用于企业内部。

常用的仓库列表

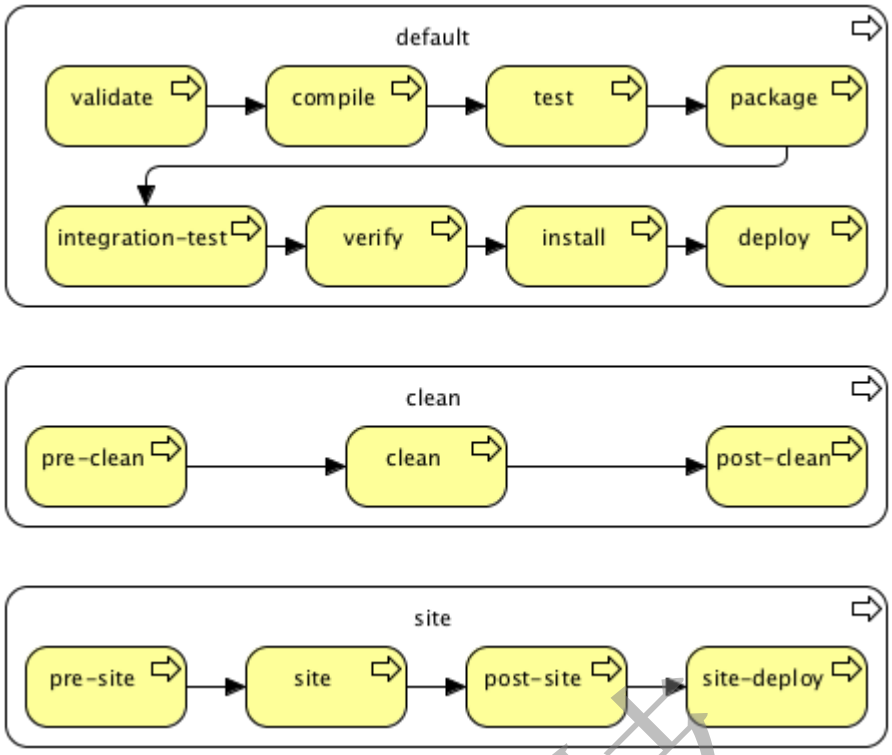
- Maven在线检索： http://search.maven.org/
- 中央仓库: http://repo1.maven.org/maven2
- 阿里: http://maven.aliyun.com/nexus/content/groups/public/ (国内访问速度快)
- 开源中国: http://maven.oschina.net/content/groups/public/
- Spring社区: http://maven.springframework.org/release

2.4 构建生命周期



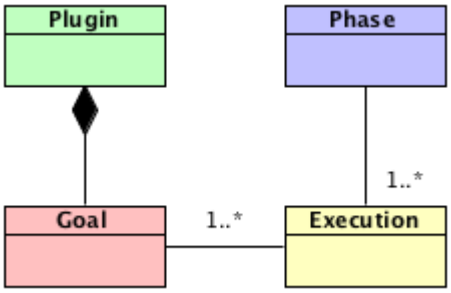
maven将工程（Project）的构建过程理解为不同的生命周期(LifeCycle)和阶段（Phase）。 在工程的构建过程中，存在着不同的生命周期，这些生命周期互相独立，之间也没有一定的顺序关系。 每个生命周期又划分为不同的阶段（Phase）。阶段之间有明确的顺序关系， 同一生命周期内的阶段必须按顺序依次执行。

maven内置了三个生命周期，并为每个生命周期内置了一些阶段。 下面列举出maven内置的生命周期及主要的阶段:



- default：构建（Build）
  - validate：验证项目是否正确，所有必需的信息是否可用。
  - compile：编译项目中的代码。
  - test：用相关的单元测试框架测试编译后的代码，这些运行的测试并不会随项目打包和布署。
  - package：将编译后的代码打包成相应的格式文件，如jar包。
  - integration-test： 如果需要在一个综合环境中运行我们的测试，这个阶段将会运行和布署项目到该环境中。
  - verify： 检查项目的包是否正确和符合要求。
  - install：将包安装到本地maven仓库，可以让其它项目作为依赖使用该包。
  - deploy：将包发布到远程的maven仓库，并提供给其它开发者使用。
- clean：清理
  - pre-clean 准备清理
  - clean 执行清理工作
  - post-clean 执行清理后的后续工作
- site：生成项目文档和站点
  - pre-site 准备生成
  - site 生成项目站点和文档
  - post-site 执行生成文档后的后续工作
  - site-deploy 发布项目文档

2.5 目标和插件



Maven中定义的工程周期和阶段只是抽象的概念，不涉及具体的功能。具体的功能由插件（Plugin）实现。目标代表一个特定的任务，一个插件可以实现多个目标（Goal）。

为了解耦插件的功能和工程阶段，实现高度的可配置性，maven规定插件只是实现目标的功能， 通过配置来决定在哪个阶段执行（Execution)哪些目标操作。 甚至可以把一个Goal绑定到多个Phase，以实现复用。

maven内置了一些默认的插件，并根据不同的工程packing类型在各个phase中默认绑定了一些goal。 下表中列出default生命周期中各阶段默认绑定的goal，其中goal按照惯例使用pluginname:goalname的方式标记：

Phase(阶段)	**Phase:Goal(阶段:目标)**
process-resources	resources:resources
compile	compiler:compile
process-test-resources	resources:testResources
test-compile	compiler:testCompile
test	surefire:test
package	ejb:ejb/ejb3:ejb3/jar:jar/par:par/rar:rar/war:war
install	install:install
deploy	deploy:deploy

maven的插件是一种packaging类型为maven-plugin的特殊类型的工程，它和普通的maven工程在使用maven的方法和特性是一样的，不同之处是它服务与maven工具，可以通过创建maven插件来完成定制化的工程构件，管理的功能。

3. Maven常用操作

- `mvn clean` 清理构建目录
- `mvn compile` 编译源代码
- `mvn test-compile` 编译测试代码
- `mvn test` 运行单元测试
- `mvn package` 运行打包
- `mvn install` 运行安装，生成的构建存储到本地仓库
- `mvn clean package` 清理然后打包
- `mvn clean package -Dmaven.test.skip=true` 清理然后打包，但是跳过测试阶段
- `mvn site` 生成项目站点