1，POM文件详解

（1）含义：project object model项目对象模型

DOM：document object model文档对象模型

（2）pom.xml对于maven工程是核心的配置文件，与构建过程相关的一切设置都在这个文件中

进行配置。重要程度相当于web.xml对于web工程。

2，坐标

（1）数学中的坐标

①在平面中，使用x、y俩个向量可以唯一定位平面中的任何一个点。

②在空间中，使用x、y、z三个向量可以唯一的定位空间中的任何一个点。

（2）maven的坐标

（gav表示maven工程中唯一定位一个maven工程的三个坐标）

使用下面三个向量在仓库中唯一定位一个maven工程。

①groupid：公司或组织域名倒序+项目名

如：<groupId>com.baidu.maven1</groupId>

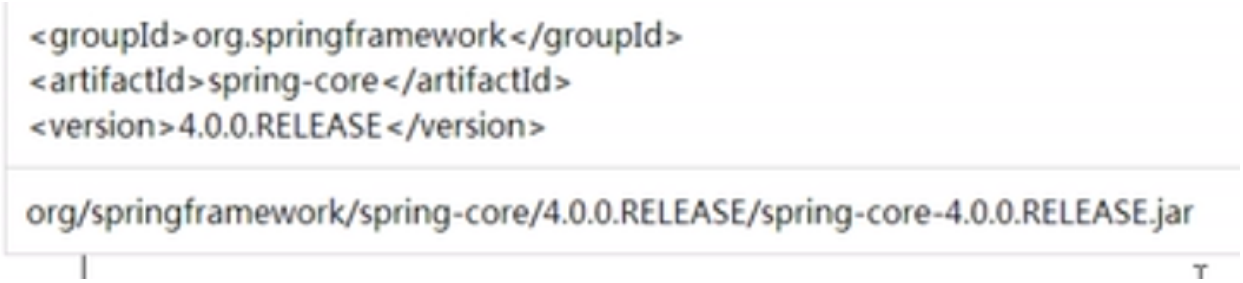
②artifactid：模块的名称

如：<artifactId>hello</artifactId>

③version：版本

如：<version>1.0.0</version>

（3）maven工程中的坐标与仓库中路径的对应关系



①groupId和artifactId中将点换成/，但是version中版本号的点却不需要换成/。

②坐标与路径的对应关系：

groupId/artifactId/version/artifactId-version.jar

3，仓库

（1）仓库的分类

①本地仓库：当前电脑上部署的仓库目录，为当前电脑上所有maven工程服务。

②远程仓库

a，私服nexus（局域网范围内）：搭建在局域网环境中，为局域网范围内的所有maven

工程服务。

b，中央仓库：假设在Internet上，为全世界所有的maven工程服务。

c，中央仓库的镜像：为了分担中央仓库的流量，提升用户访问的速度。

（2）仓库中保存的内容：maven工程

①maven自身所需要的插件

②第三方框架或者工具的jar包

③我们自己开发的maven工程

第一方是JDK，第二方是我们自己，第三方是jar包。

4，依赖

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>
  </dependencies>
```

（下载第三方jar包的话，尽量使用release稳定版的，而不要用快照形式的）

（1）maven解析依赖信息时，会到本地仓库中查找被依赖的jar包。

对于我们自己开发的maven工程，使用install命令安装后就可以进入仓库。

安装命令：mvn install（在需要被依赖的项目里面，和src同级）

（2）依赖的范围：主程序看不见测试程序，测试程序可以看见主程序

①compile

a，对主程序是否有效：有效

b，对测试程序是否有效：有效

c，是否参与打包：参与

d，是否参与部署：参与

②test

a，对主程序是否有效：无效

b，对测试程序是否有效：有效

c，是否参与打包：不参与

d，典型例子：Junit

③provided

a，对主程序是否有效：有效（开发过程）

b，对测试程序是否有效：有效（开发过程）

c，是否参与打包：不参与

d，是否参与部署：不参与

e，典型例子，servlet-api.jar

5，生命周期

（1）各个构建环节执行的顺序：不能打乱顺序，必须按照既定的正确顺序来执行。

（2）maven的核心程序中定义了抽象的生命周期。生命周期中，各个阶段的具体任务是由插件

来完成的。

（3）maven有三套相互独立的生命周期，分别是：

①Clean Lifecycle在进行真正的构建之前进行一些清理工作。

②Default Lifecycle构建的核心部分，编译、测试、打包、安装和部署等。

③Site Lifecycle生成项目报告，站点，发布站点。

他们是相互独立的，你可以仅仅调用clean来清理工作目录，仅仅调用site来生成站点，当

然也可以直接运行mvn clean install site 运行所有这三套生命周期。

每套生命周期都由一组阶段（phase）组成，我们平时在命令行输入的命令总会对应于一

个特定的阶段，比如：运行 mvn clean这个是clean生命周期的一个阶段，有Clean生命周

期，也有clean阶段。

（4）Clean生命周期（mvn clean）

Clean生命周期一共包含了三个阶段：

①pre-clean执行一些需要在clean之前完成的工作。

②clean移除所有上一次构建生成的文件。

③post-clean执行一些需要在clean之后完成的 工作。

（5）Site生命周期（mvn site）

①pre-site执行一些需要在生成站点文档之前完成的工作。

②site生成项目的站点文档。

③post-site执行一些需要在生成站点文档之后完成的工作，并且为部署做准备。

④site-deploy将生成的站点文档发送到特定的服务器上。

这里经常用到的是site阶段和site-deploy阶段，用以生成和发布maven站点，这可是

maven相当强大的功能，manager比较喜欢，文档及统计数据自动生成。

（6）Default生命周期

Default生命周期是maven生命周期中最重要的一个，绝大部分工作都发生在这个生命周期

中，这里，只解释一些比较重要和常用的阶段。

①process-resources复制并处理资源文件，至目标目录，准备打包。

②compile编译项目的源代码。

③process-test-resources复制并处理资源文件，至目标测试目录。

④test-compile编译测试源代码。

⑤test使用合适的单元测试框架运行测试，这些测试代码不会被打包或部署。

⑥package接受编译好的代码，打包成可发布的格式，如jar。

⑦install将包安装至本地仓库，以让其他项目依赖。

⑧deploy将最终的包复制到远程仓库，以让其他开发人员与此昂吗共享或部署到服务器运

行。

（7）maven核心程序为了更好的实现自动化构建，按照这一特点执行声明周期中的各个阶段，

不论现在要执行生命周期的哪一个阶段。都是从这个生命周期最初的位置开始执行。

（8）插件和目标

①声明周期的各个阶段仅仅定义了要执行的任务是什么。

②各个阶段和插件的目标是对应的。

③相似的目标由特定的插件来完成。

生命周期阶段	插件目标	插件
compile	compile	maven-compiler-plugin
test-compile	testCompile	maven-compiler-plugin

④可以将目标看做“调用插件功能的命令”。