853769620

在maven中的版本如何管理

1、版本到底是什么

版本分为两种，一种是快照（snapshot）版本，一种是发布（release）版本

快照版本就是版本号后面加上SNAPSHOT，比如1.0.0-SNAPSHOT

发布版本就是版本号上不加SNAPSHOT的，比如1.0.0

而且版本通常而言是三位的，特别是我们自己公司里的项目，就是比如1.0.0

1.0.0中的第一个1指的是一个重大版本，比如已经基本成型的一个系统架构

也有不少项目一开始是0.0.1这样的版本，这个指的就是这个系统刚开始起步，甚至都没能形成一个完整的东西出来，只是少数核心功能也出来了

1.0.0中的第二个0指的是一个次要版本，比如1.1.0，那么就是加了一些新的功能或者开发了一些新的模块，或者做了一些较大的代码重构，技术升级改造

1.0.0中的第三个0指的是日常迭代的一个增量版本，比如1.1.1，一般对应着修复了一个bug，或者对某些代码做了轻微的优化或者调整

形象一些：

假设，咱们现在有一个电商系统，一开始比较low，就是spring boot快速做出来的，1.0.0-SNAPSHOT，内部开发和测试，1.0.0。一开始1.0.0这个版本包含了商城首页、购物车、订单系统、物流系统，就这么几个模块。

接下来开始对第一个版本进行迭代，比如说这次要新增一个用户评论功能，此时就会进入一个版本，叫做1.1.0-SNAPSHOT，在内部先开发和测试。接着发布1.1.0版本，此时这个版本就新增了一个用户评论的模块。

不巧，突然发现某天，购物车这块有个bug，就是说，5分钟没有支付，就莫名奇妙session失效，导致购物车里的东西没了。但是产品需求是说购物车里的东西默认保留7天。修复这个bug，1.1.1-SNAPSHOT，这个bug的修复在内部先开发和测试，让产品经理来验收。最后发布，此时版本变成了1.1.1。此时代码就是包含了一个bug的修复。

后面就新增了抽奖模块，1.2.0。新增了个性化推荐功能，1.3.0，修复了几个bug，1.3.5，版本。

接着，咱们发现，这个现有系统架构有点问题，主要是多人写作开发，一个30人的团队，全部对一块代码在各种修改，很恶心。所以开始改造和升级到微服务架构。

整个系统架构出现了重大的变化，就开始进入2.0.0-SNAPSHOT的开发，这个过程，比如说先把核心的几个模块改造了spring cloud的微服务化。2.0.0代码发布。

接着，开始陆续对几个非核心的模块进行微服务的改造，比如评论模块，单拉出来作为一个服务，此时进入2.1.0版本。修复了几个bug，变成了2.1.9版本。

然后又开始，用户量变成了1000万，现有架构无法支撑高并发的访问，高可用的保障。此时又要升级系统架构，高并发高可用的架构，基于各种外部设施以及系统架构的改造来做。此时版本会进入3.0.0-SNAPSHOT版本。3.0.0，3.1.0，3.5.15。

上面的是互联网公司里，比较通行的一个版本迭代演进的方式，而且也是比较简单的一种。

如果有的项目，你要拆分的更细，那么可能还可以有4位，1.2.3.15。具体根据你的项目来定。

此外maven的标准版本规范里，还包含了一个1.0.0-beta-1，最后一个1代表的是增量版本的一个里程碑，就是在进行某个bug修复，或者功能调整的时候，是分为多个步骤，也就是多个里程碑的，那么此时可能就会对应了1.1.1-beta-1，1.1.1-beta-2，1.1.1-beta-3，这样好几个里程碑版本，每个里程碑版本代表完成了一个小阶段

这里的beta代表的是公开测试版，就是对外提供试用的

也有的项目用的时alpha来替代beta，alpha也可以理解为实验版本，也就是内部测试版本，就是给自己公司内部用用的

也有rc版本，就是预发布版本，基本就比较稳定了，但是还不是最终发布版，可以尝试下载使用了

带你看看spring boot的版本

并不是这几位的数字都需要使用的，一般我们自己的项目里，用3位就可以了，通常我们的经验是这样的：

除非进行大的架构升级改造，才会增加大版本，这个很少很少，一般几年才一次

次要版本，一般就是对应各种需求，比如一个持续几周，或者一个月或者了几个月的大需求，对应一个次要版本

增量版本，一般就是一个大需求中的多个小需求，每个小需求可以给一个增量版本，或者是紧急fix了一个bug，那么就增加一个增量版本

2、版本是如何变更的

通常来说，在开发的时候，版本都是SNAPSHOT版本，也就是快照版本，比如说1.0.0-SNAPSHOT此时代码还在不断开发和修改，或者进行测试中，还没有完成所有的测试

这个时候的版本通常用于多个系统间协调开发，给其他系统去引用你开发的SNAPSHOT版本，让人家可以测试或者联调

然后在一个SNAPSHOT版本开发完之后，同时通过了完善的测试，版本就会升级到release版本，也就是不带snapshot后缀的版本，比如说1.0.0，此时就是让依赖方也改为这个发布版本，然后就可以上线了

比如1.0.0-SNAPSHOT在开发中，接着正式发布了1.0.0；然后又开始了1.1.0-SNAPSHOT的开发了，接着正式发布了1.1.0；然后又开始了1.2.0-SNAPSHOT的开发了，接着正式发布了1.2.0，以此类推

如果要将snapshot发布发布为release版本，至少需要满足几个条件：

（1）所有的单元测试全部通过

（2）pom.xml中没有任何snapshot版本的依赖

（3）pom.xml中没有任何snapshot版本的插件

（4）这个版本的代码已经全部提交到对应的git分支上了，同时一定从分支merge到了master主干分支上去，并且给master分支打上了一个tag，这个tag就是对主干分支这一时刻代码的一个标签，这样任何时候都可以回退到主干分支的任何一个tag对应的版本的代码上去

此时一个release版本就完成了，然后就继续升级到下一个版本的snapshot版本上去，继续进行开发和测试

3、版本与主干、分支以及标签

其实一般就是一个版本对应一个分支，各种开发+测试

搞定之后就会merge到主干

然后就会给这个时候的主干打一个tag作为主干在这个版本的代码，以后任何时候都可以使用主干的某个tag，也就是某个时刻的某个版本的代码