SVN中trunk,branches,tags用法详解

svn trunk branches tags

文章转载自：http://www.cnblogs.com/dafozhang/archive/2012/06/28/2567769.html

Subversion有一个很标准的目录结构，是这样的。
比如项目是proj，svn地址为svn://proj/，那么标准的svn布局是

svn://proj/|+-trunk+-branches+-tags
这是一个标准的布局，trunk为主开发目录，branches为分支开发目录，tags为tag存档目录（不允许修改）。
但是具体这几个目录应该如何使用，svn并没有明确的规范，更多的还是用户自己的习惯。

对于这几个开发目录，一般的使用方法有两种。我更多的是从软件产品的角度出发（比如freebsd），因为互联网的开发模式是完全不一样的。 1.第一种方法，使用trunk作为主要的开发目录
一般的，我们的所有的开发都是基于trunk进行开发，当一个版本/release开发告一段落（开发、测试、文档、制作安装程序、打包等）结束后，代码处于冻结状态（人为规定，可以通过hook来进行管理）。此时应该基于当前冻结的代码库，打tag。当下一个版本/阶段的开发任务开始，继续在trunk进行开发。
此时，如果发现了上一个已发行版本（Released Version）有一些bug，或者一些很急迫的功能要求，而正在开发的版本（Developing Version）无法满足时间要求，这时候就需要在上一个版本上进行修改了。应该基于发行版对应的tag，做相应的分支（branch）进行开发。
例如，刚刚发布1.0，正在开发2.0，此时要在1.0的基础上进行bug修正。
按照时间的顺序

1.0开发完毕，代码冻结
基于已经冻结的trunk，为release1.0打tag
此时的目录结构为
svn://proj/
+trunk/ (freeze)
+branches/
+tags/
+tag_release_1.0 (copy from trunk)
2.0开始开发，trunk此时为2.0的开发版
发现1.0有bug，需要修改，基于1.0的tag做branch
此时的目录结构为
svn://proj/
+trunk/ (dev 2.0)
+branches/
+dev_1.0_bugfix (copy from tag/release_1.0)
+tags/
+release_1.0 (copy from trunk)
在1.0 bugfix branch进行1.0 bugfix开发，在trunk进行2.0开发
在1.0 bugfix 完成之后，基于dev_1.0_bugfix的branch做release等
根据需要选择性的把dev_1.0_bugfix这个分支merge回trunk（什么时候进行这步操作，要根据具体情况）
这是一种很标准的开发模式，很多的公司都是采用这种模式进行开发的。trunk永远是开发的主要目录。

2.第二种方法，在每一个release的branch中进行各自的开发，trunk只做发布使用。

这种开发模式当中，trunk是不承担具体开发任务的，一个版本/阶段的开发任务在开始的时候，根据已经release的版本做新的开发分支，并且基于这个分支进行开发。还是举上面的例子，这里面的时序关系是：

1.0开发，做dev1.0的branch
此时的目录结构
svn://proj/
+trunk/ (不担负开发任务)
+branches/
+dev_1.0 (copy from trunk)
+tags/
1.0开发完成，merge dev1.0到trunk
此时的目录结构
svn://proj/
+trunk/ (merge from branch dev_1.0)
+branches/
+dev_1.0 (开发任务结束，freeze)
+tags/
根据trunk做1.0的tag
此时的目录结构
svn://proj/
+trunk/ (merge from branch dev_1.0)
+branches/
+dev_1.0 (开发任务结束，freeze)
+tags/
+tag_release_1.0 (copy from trunk)
1.0开发，做dev2.0分支
此时的目录结构
svn://proj/
+trunk/
+branches/
+dev_1.0 (开发任务结束，freeze)
+dev_2.0 （进行2.0开发）
+tags/
+tag_release_1.0 (copy from trunk)
1.0有bug，直接在dev1.0的分支上修复