陽芫眞薀暖，筷楽ぬ簡單

suí xīn suí yì

SVN tag和branch的选择及应用

默认分类 2009-11-26 11:08:44 阅读242 评论0

trunk：表示开发时版本存放的目录，即在开发阶段的代码都提交到该目录上。

branches：表示发布的版本存放的目录，即项目上线时发布的稳定版本存放在该目录中。

tags：表示标签存放的目录。

在这需要说明下分三个目录的原因，如果项目分为一期、二期、三期等，那么一期上线时的稳定版本就应该在一期完成时将代码copy到branches 上，这样二期开发的代码就对一期的代码没有影响，如新增的模块就不会部署到生产环境上。而branches上的稳定的版本就是发布到生产环境上的代码，如 果用户使用的过程中发现有bug，则只要在branches上修改该bug，修改完bug后再编译branches上最新的代码发布到生产环境即可。 tags的作用是将在branches上修改的bug的代码合并到trank上时创建个版本标识，以后branches上修改的bug代码再合并到 trunk上时就从tags的version到branches最新的version合并到trunk，以保证前期修改的bug代码不会在合并。

一直以来用svn只是当作cvs，也从来没有仔细看过文档，直到今天用到，才去翻看svn book文档，惭愧

需求一：
有一个客户想对产品做定制，但是我们并不想修改原有的svn中trunk的代码。

方法：
用svn建立一个新的branches，从这个branche做为一个新的起点来开发

代码

□□□ svn copy svn://server/trunk svn://server/branches/ep -m "init ep"

簡單diǎ

微小的幸福就在身边,容易满足就是天堂。

加博友 关注他

最新日志

字节与比特
2011年01月18日
WebLogic Server应用部□
网站静态化
揭秘淘宝286亿海量图片□
svn开发模式目录结构

博主推荐

博客首页酷起来！8种摘□
西南大旱考问中国的城□

首页推荐

ÖÐ¹Ø´åÑÞÕÕÃÅÊÇ³´×÷
Ôâ»ð»ÙÈÝÉÙÅ®µÄ×Ô°×
ÇéÈË½Ú£¬ÈÃ×Óµ¯·É
¶¬Ó¾·¢ÏÖµÄ³¬Õð⁰³Ò»Ä»
·ÇÖÞÒ°Ïó±Æ½üÓÎÀÀ³µ
Ð¡Ò×ÓÎÌ©¹ú°ìµÆÇø

更多>>

Tip:
如果你的svn中以前没有branches这个的目录，只有trunk这个，你可以用

代码

□□□ svn mkdir branches

新建个目录

需求二:
产品开发已经基本完成，并且通过很严格的测试，这时候我们就想发布给客户使用，发布我们的1.0版本

代码

□□□ svn copy svn://server/trunk svn://server/tags/release-1.0 -m "1.0 released"

咦，这个和branches有什么区别，好像啥区别也没有?
是的，branches和tags是一样的，都是目录，只是我们不会对这个release-1.0的tag做修改了，不再提交了，如果提交那么就是branches

需求三:
有一天，突然在trunk下的core中发现一个致命的bug,那么所有的branches一定也一样了，该怎么办?

代码

□□□ svn -r 148:149 merge svn://server/trunk branches/ep

其中148和149是两次修改的版本号。

在SVN中Branch/tag在一个功能选项中，在使用中也往往产生混淆。

在实现上，branch和tag，对于svn都是使用copy实现的，所以他们在默认的权限上和一般的目录没有区别。至于何时用tag，何时用branch，完全由人主观的根据规范和需要来选择，而不是强制的（比如cvs）。

一般情况下，
tag，是用来做一个milestone的，不管是不是release，都是一个可用的版本。这里，应该是只读的。更多的是一个显示用的，给人一个可读（readable）的标记。
branch，是用来做并行开发的，这里的并行是指和trunk进行比较。

比如，3.0开发完成，这个时候要做一个tag，tag_release_3_0，然后基于这个tag做release，比如安装程序等。trunk进入 3.1的开发，但是3.0发现了bug，那么就需要基于tag_release_3_0做一个branch，branch_bugfix_3_0，基于这 个branch进行bugfix，等到bugfix结束，做一个tag，tag_release_3_0_1，然后，根据需要决定 branch_bugfix_3_0是否并入trunk。

对于svn还要注意的一点，就是它是全局版本号，其实这个就是一个tag的标记，所以我们经常可以看到，什么什么release，基于xxx项目的 2xxxx版本。就是这个意思了。但是，它还明确的给出一个tag的概念，就是因为这个更加的可读，毕竟记住tag_release_1_0要比记住一个 很大的版本号容易的多。

这只是个说明，你不按说明来办事也不会怎么样，svn照样工作得很好。

branches：分枝

当多个人合作（Sally和John）时，可能有这样的情况出现：John突然有个想法，跟原先的设计不太一致，可能是功能的添加或者日志格式的改 进等等，总而言之，这个想法可能需要花一段时间来完成，而这个过程中，John的一些操作可能会影响Sally的工作，John从现有的状态单独出一个 project的话，又不能及时得到Sally对已有代码做的修正，而且独立出来的话，John的尝试成功时，跟原来的合并也存在困难。这时最好的实践方 法是使用branches。John建立一个自己的branch，然后在里面实验，必要的时候从Sally的trunk里取得更新，或者将自己的阶段成果 汇集到trunk中。（svn copy SourceURL/trunk \ DestinationURL/branchName \ -m “Creating a private branch of xxxx/trunk.” ）

trunk：主干

主干，一般来说就是开发的主要呆的地方，

tag:

在经过了一段时间的开发后，项目到达了一个里程碑阶段，你可能想记录这一阶段的代码的状态，那么你就需要给代码打上标签。(svn cp file:///svnroot/mojavescripts/trunk \ file:///svnroot/mojavescripts/tags/mirrorutils_rel_0_0_1 \ -m “taged mirrorutils_rel_0_0_1″)

版本库布局
在将你的数据导入到版本库之前，首先你得考虑如何组织你的数据。如果你使用一种推荐的布局，你在后面的操作将会更容易许多。

有一些标准的、推荐的方式来组织一个版本库。大多数人建一个trunk目录来存放开发的“主线”、一个branches目录来容纳分支副本、以及一个tags目录来容纳标签复制。如果一个版本库只存放一个项目，人们通常创建三个这样的顶层目录:

/trunk
/branches
/tags

如果一个版本库包含多个项目，人们通常按分支来安排布局:

/trunk/paint
/trunk/calc
/branches/paint
/branches/calc
/tags/paint
/tags/calc

……或者按项目:

/paint/trunk
/paint/branches
/paint/tags
/calc/trunk
/calc/branches
/calc/tags

如果项目不是密切相关，而且每一个是单独被检出，那么按项目布局是合理的。对于那些你想一次检出所有项目，或需要将它们打成一个分发包的相关项目，按分支来布局通常比较好。这种方式你只要检出一个分支，而且子项目之间的关系也比较清楚。

如果你采用顶层/trunk /tags /branches这种方式，并不意味着你必须复制整个主线为分支或标签，而且某些情况下这种结构更具灵活性。

对于不相关的项目，你可能更愿意使用不同的版本库。当你提交时，改变的是整个版本库的修订号，而不是项目的。让两个不相关的项目共用一个版本库，会导致修 订号出现较大的跳跃。Subversion和TortoiseSVN项目看起来是在同一个主机地址，但是它们是在完全独立的版本库中开发着，并且版本号也 不相干。

当然，你完全可以不理会上面提及的通用布局。你可以自由改变，来满足你和你团队的需要。请记住，不管你选择哪种布局，它都不是永久的。你可以在随时重新组织你的版本库。因为分支和标签是普通的目录，只要你愿意，TortoiseSVN可以将它们移动或重命名。