Evan
Only let oneself become strong enough, good enough, can afford the life that you want to.

svn工具的使用问题总结

前言：

最近在开发的时候，由于需求太多，开发周期长短不一，从主线上切了多个分支（一般不在主线trunk上开发，万一线上出问题可及时修改代码上线），在部分功能上线后，想把代码同步到新的分支上去，最开始的想法是人工去合并代码，把两个分支down下来把修改的文件比较并进行合并，然后最后提交；但是文件修改的很多很杂，人工去合并肯定出现误差，比如代码合并少了；其实这些都是可以通过svn工具进行处理的，于是回顾了下开发过程中经常用到的svn功能。

svn有客户端和服务端两个，我们一般开发的时候，不会涉及到服务端svn的管理，服务端一般是配管来进行管理的。

我们平时开发接触到的都是svn的客户端tortoise svn，还有开发工具eclipse的svn插件.

最常用的svn功能有：检出工程 ，提交文件，更新文件，合并代码 ，show revision graph，切换（切换到历史版本或者是当前最新版本），显示资源历史记录；

在大多数情况下我们使用开发工具的svn插件，方便，但是插件有时候不是很稳定，经常抽风，这时候小乌龟就该上场了。

小乌龟的基本使用：

1，查看资源库内容，右键-tortoiseSvn-repo browser 输入url就可以查看服务器上的代码

2，检出资源的时候，右键-tortoiseSvn-export/svn check out[export和svn check out 是有区别的，export把源码down到本地，然后，然后，就和服务器端的svn没联系了，所以开发的时候建议用check out]

3，提交新增文件，右键-tortoiseSvn-Add

4，提交修改文件，右键-tortoiseSvn-commit

5. 比较历史文件，右键-tortoiseSvn-diff with previous version

6. 把文件切换到历史一个版本 ，在文件右键-tortoiseSvn-switch

使用中常见问题总结：

1，代码冲突如何解决？

冲突原因： A 、 B 两个用户都在版本号为 1 的时候，更新了1.txt 这个文件， A 用户在修改完成之后提交1.txt 到服务器，这个时候提交成功，这个时候 1.txt 文件的版本号已经变成 2 了。同时 B 用户在版本号为 1 的1.txt 文件上作修改，修改完成之后提交到服务器时，由于不是在当前最新的 2 版本上作的修改，所以导致提交失败。

版本冲突现象：冲突发生时， subversion 会在当前工作目录中保存所有的目标文件版本 [上次更新版本、当前获取的版本（即别人提交的版本）、自己更新的版本、目标文件] 。假设文件名是 1.txt

对应的文件名分别是：1.txt.r101，1.txt.r102，1.txt.mine，1.txt 。同时在目标文件中标记来自不同用户的更改。

解决冲突有三种选择：

A 、放弃自己的更新，使用 svn revert （回滚），然后提交。在这种方式下不需要使用 svn resolved （解决）

B 、放弃自己的更新，使用别人的更新。使用最新获取的版本覆盖目标文件，执行 resolved filename 并提交（选择文件 —[illegible] 解决）。

C 、手动解决：冲突发生时，通过和其他用户沟通之后，手动更新目标文件。然后执行 resolved filename 来解除冲突，最后[illegible]式比较多用)

使用tortoiseSvn解决冲突图形方法：

在冲突的文件上（选中文件 -- 右键菜单 —TortoiseSVN—Edit conflicts （解决冲突）），Theirs 窗口为服务器上当前最新版本，Mine 窗口为本地修改后的版本，Merged 窗口为合并后的文件内容显示 ；

修改完成后，保存 1.txt 文件内容;

在冲突目录下，选中文件 -- 右键菜单 —TortoiseSVN—Resolved （解决）。会列出冲突的文件列表，如果确认已经解决，点 OK；

提交解决冲突后的文件。

命令行操作 ：

1、svn update后，1.txt文件出现冲突，选择base版本，即1.txt.rOld作为最后提交的版本

$ svn resolve –accept base

2.手工修改1.txt文件，然后将当前拷贝即1.txt作为最后提交的版本

$ svn resolve –accept working 1.txt

3.使用1.txt.rNew作为最后提交的版本

$ svn resolve –accept theirs-full 1.txt

4.使用1.txt.mine作为最后提交的版本

$ svn resolve –accept mine-full 1.txt

5.使用1.txt.mine作为最后提交的版本

$ svn resolve –accept theirs-conflict 1.txt

```
resolve: 解决工作副本中目录或文件的冲突。
用法: resolve --accept=ARG [PATH...]

  注意:  当前需要选项 --accept 。

有效选项:
  --targets ARG           : 传递文件 ARG 内容为附件参数
  -R [--recursive]        : 向下递归，与 --depth=infinity 相同
  --depth ARG             : 受深度参数 ARG(“empty”， “files”， “immediates”，或“infinity”) 约束的操作
  -q [--quiet]            : 不打印信息，或只打印概要信息
  --accept ARG            : 指定自动解决冲突动作的源
                            ('base', 'working', 'mine-conflict',
                             'theirs-conflict', 'mine-full', 'theirs-full')

全局选项:
  --username ARG          : 指定用户名称 ARG
  --password ARG          : 指定密码 ARG
```

--no-auth-cache : 不要缓存用户认证令牌

--non-interactive : 不要交互提示

--trust-server-cert : 不提示的接受未知的 SSL 服务器证书(只用于选项 “--non-interactive”)

--config-dir ARG : 从目录 ARG 读取用户配置文件

--config-option ARG : 以下属格式设置用户配置选项：

FILE:SECTION:OPTION=[VALUE]

例如：

servers:global:http-library=serf

2，TortoiseSVN branch/tag switch Relocate

Switch是转换当前工作副本对应的工作目录，一般是从trunk工作目录转向tag工作目录，或者从tag转回来，switch的类似update，将switch的目标工作目录的文件更新到本地，一般会产生很多冲突。【像在前言提到的需要合并或切换的分支的时候，就可以用此功能】

Relocate是当代码仓库的访问路径（服务器的计算机名称修改，或IP地址变更，URL变更），而此时已检出修改的工作副本（working copy）没有变更，若此时直接提交（commit），肯定不能成功，因为此提交地址对应的svn服务器不存在了。TortoiseSVN为我们提供了重定位工作副本的功能（TortoiseSVN → Relocate），此指令扫描.svn文件夹中的所有条目，改变条目的url（服务器地址）为新输入的地址。

重定位操作可能的原因：

a) 服务器的IP地址已更改

b) 协议已更改(比如从http://改为 https://)

c) 版本库在服务器的路径已更改

3.svn switch的使用（看完merge的功能后，对switch的理解就是切换分支的作用）

a) 分支内的switch命令使用，右键A文件swith B文件，实际是将 B 的内容更新到了 A。

b)分支A和B，A本地文件有修改未提交，然后swtih B分支时，若A本地修改文件不会丢失；

分支A和B，A分支有文件修改并已提交，然后switch B分支，此时A分支代码被覆盖，这个时候就相当于update B 文件到A分支，所以慎用！！

分支A和B，A本地文件有修改，B分支同一文件也有修改并已提交，然后swtih B分支时，此时会文件冲突；

分支A和B，B分支有文件修改并已提交，然后swtih B分支时，B分支文件被更新到A；

4.svn创建分支

```
svn copy -m "1.7.2 - theme" svn://localhost/www/trunk svn://localhost/www/branches/branch
svn co svn://localhost/www/branches/branch
```

5.合并代码

a)从trunk中merge到分支。**(如果使用小乌龟选择第一项)**

```
#前面的1是开分支之前trunk的版本号，后面的2是merge时trunk的版本号
svn merge -r 1:2svn://localhost/www/trunk
```

b)从分支merge到trunk。**(如果使用小乌龟选择第二项)**

```
#先从trunk checkout一份新鲜的代码，然后cd到该版本目录下
svn co svn://localhost/www/trunk
cd trunk
```

#12973是分支开始的版本号，13006是分支结束的版本号

```
svn merge -r 12973:13006 svn://localhost/www/branches/branch
```

c)分支合并（小乌龟第三项）

合并的时候，一定配上起始版本号

终上所述：前言部分所遇到的问题，应该已经有比较好的解决方法，有效的使用svn工具会为我们的开发效率带来很大的提升。

备注：很好的一份svn使用手册http://www.bsdmap.com/manuals/subversion/index.html

- 上一篇 Maven 本地仓库，远程仓库，中央仓库，Nexus私服，镜像 详解

相关文章推荐

- SVN的使用方法总结/如何使用TortoiseSVN工具...
- SVN常见问题及解决方案
- xcode代码同步的问题--使用自带的svn工具
- svn使用
- SVN版本管理工具使用中常见的代码提交冲突问题...
- #版本管理工具使用总结（git,svn,hg）
- 工具集合
- git ,Maven，SVN工具使用总结;
- Java开发工具Eclipse使用上的问题总结
- 工具之SVN使用教程总结

猜你在找

【直播】机器学习&深度学习系统实战（唐宇迪）
【直播回放】深度学习基础与TensorFlow实践（王琛）
【直播】机器学习之凸优化（马博士）
【直播】机器学习之概率与统计推断（冒教授）
【直播】TensorFlow实战进阶（智亮）
【直播】Kaggle 神器：XGBoost 从基础到实战（冒教授）
【直播】计算机视觉原理及实战（屈教授）
【直播】机器学习之矩阵（黄博士）
【直播】机器学习之数学基础
【直播】深度学习30天系统实训（唐宇迪）

查看评论

暂无评论

发表评论

用 户 名： molashaonian
评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场