# 9.4. Git和Hg面对面

## 9.4.1. 面对面访谈录 ¶

**Git：你好Hg，我发现我们真的很像。**

Hg：是啊，人们把我们都归类为分布式版本控制工具，所以我们之间的相似度，要比和CVS、SVN的相似度高的多了。

Hg：我是用Python和少部分的C语言实现的，你呢？

**Git：我的核心当然使用C语言了，因为Linus Torvalds最爱用C语言了。我的很多命令还使用了Shell脚本和Perl语言开发，Python用的很少。**

Hg：大量的使用C语言，是你的性能比我高的原因么？

**Git：当然不是了，你不也在核心模块使用C语言了么？问题的关键在于我的对象库设计的非常优秀。你不要忘了我是谁发明的，可是大名鼎鼎的Linux之父Linus Tolvars，他对Linux文件系统可是再熟悉不过的了，所以他能够以文件系统开发者的视角实现我的核心。**

**Git：还有我在网络传输过程中非常的直观，可以看到实时的进度显示，这也是你所没有的。**

Hg：是啊，非常的惭愧。当克隆一个比较大的Hg版本库时，会出现假死状态，用户不知道克隆操作的进展。对了，你为什么能够实现实时的进度显示呢？

**Git：之所以我能够有这样的实现，是因为我使用了“智能协议”。在网络传输的各自一端都启用了相应的辅助程序，实现差异传输及传输进度的计算和显示。**

Hg：我有一个特点是SVN用户非常喜欢的，就是我的顺序数字版本号。

**Git：你的顺序数字版本号只是在本地版本库中有效的。也就是说，你不能像SVN那样将顺序数字版本号作为项目本身的版本号，因为换做另外的一个版本库克隆，那个数字版本号就会不一样了。**

Hg：我觉得你的暂存区（stage）的概念太古怪了。我提交的时候，改动的文件会直接提交而不需要什么注册到暂存区的操作。

**Git：让读者来作评判吧。如果读者读过本书的第2篇，一定会说Git的暂存区帅呆了。**

Hg：我只允许用户对最近的一次提交进行回滚撤销，而你（Git）怎么能允许用户撤销任意多次历史提交呢？那样安全么？

**Git：这就是的我对象库和引用设计的强大之处，我可以使用:command:`git reset`命令将工作分支进行任意的重置，丢弃任意多的历史。至于安全性，我的重置命令有一个保险，就是reflog，我随时可以参照reflog的记录来弥补错误的重置。**

Hg：我们的**revert**命令好像不同？

**Git：你Hg的:command:`hg revert`命令和SVN的:command:`svn revert`命令相似，是取消本地修改，用原始拷贝覆盖。你的这个操作在我这里是用:command:`git checkout`命令实现的。我也有一个:command:`git revert`命令，但是这个命令是针对某个历史提交进行反向操作，以取消该历史提交的改动的。**

Hg：我执行日志查看能够看到文本显示的分支图，你呢？

**Git：我需要在日志显示时添加参数，即用命令:command:`git log –graph`。我支持通过建立别名实现简洁的调用，例如建立一个名为glog的别名。**

**Git：我听说你Hg不支持分支？**

Hg：坦白的说，是的。我虽然也有分支命令，但是分支不是一个独立显示的提交空间，而是各个分支都显示在一起，相当于在版本库中同时拥有多个头，选择哪个分支就相当于把帽子戴在哪个头上面而已。所以我尽量要求我的用户使用克隆来当做分支。

**Git：这也就是为什么使用Hg作为项目的版本控制工具，要为每一个分支创建一个单独的版本库的原因吧。实际上我的每一个克隆的版本库也相当于独立的分支，但是因为我有强大的分支功能，因此很多用户还没有意识到。使用Topgit的用户就应该使用版本库克隆做为Topgit本身的分支管理。**

**Git：还有，因为我对分支的完整支持，使得我可以和SVN很好的协同工作。我可以将整个SVN转换为本地的Git库，但是你Hg，显然只能每次转换一个分支。**

Hg：是的，我要向你多学习。

## 9.4.2. Hg和Git命令对照

| 比较项目 | Hg命令 | Git命令 |
|---|---|---|
| URL | http://host/path/to/repos | git://host/path/to/repos.git |
| | ssh://user@host/path/to/repos | ssh://user@host/path/to/repos.git |
| | file:///path/to/repos | user@host:path/to/repos.git |
| | /path/to/repos | file:///path/to/repos.git |
| | | /path/to/repos.git |
| 配置 | [ui]<br>username = Firstname Lastname <mail@addr> | [user]<br>name = Firstname Lastname<br>email = mail@addr |
| 版本库初始化 | hg init <path> | git init [–bare] <path> |
| 版本库克隆 | hg clone <url> <path> | git clone <url> <path> |
| 获取版本库更新 | hg pull –update | git pull |
| 更新至历史版本 | hg update -r <rev> | git checkout <commit> |
| 更新到指定日期 | hg update -d <date> | git checkout HEAD@'{<date>}' |
| 更新至最新提交 | hg update | git checkout master |
| 切换至里程碑 | hg update -r <tag> | git checkout <tag> |
| 切换至分支 | hg update -r <branch> | git checkout <branch> |
| 还原文件/强制覆盖 | hg update -C <path> | git checkout – <path> |
| 添加文件 | hg add <path> | git add <path> |
| 删除文件 | hg rm <path> | git rm <path> |
| 添加及删除文件 | hg addremove | git add -A |
| 移动文件 | hg mv <old> <new> | git mv <old> <new> |
| 撤消添加、删除等操作 | hg revert <path> | git reset – <path> |
| 清除未跟踪文件 | hg clean | git clean -fd |
| 获取文件历史版本 | hg cat -r<rev> <path> > <output> | git show <commit>:<path> > <output> |
| 反删除文件 | hg add <path> | git add <path> |
| 工作区差异比较 | hg diff | git diff |
| | | git diff –cached |
| | | git diff HEAD |
| 版本间差异比较 | hg diff -r <rev1> -r <rev2> <path> | git diff <commit1> <commit2> – <path> |
| 查看工作区状态 | hg status | git status -s |
| 提交 | hg commit -m "<msg>" | git commit -a -m "<msg>" |

| 比较项目 | Hg命令 | Git命令 |
| --- | --- | --- |
| 推送提交 | hg push | git push |
| 显示提交日志 | hg log \| less | git log |
| | hg glog \| less | git log –graph |
| 逐行追溯 | hg annotate | git annotate, git blame |
| 显示里程碑/分支 | hg tags | git tag |
| | hg branches | git branch |
| | hg heads | git show-ref |
| 创建里程碑 | hg tag [-m “<msg>”] [-r <rev>] <tagname> | git tag [-m “<msg>”] <tagname> [<commit>] |
| 删除里程碑 | hg tag –remove <tagname> | git tag -d <tagname> |
| 创建分支 | hg branch <branch> | git branch <branch> <commit> |
| | | git checkout -b <branch> <commit> |
| 删除分支 | hg commit –close-branch | git branch -d <branch> |
| 导出项目文件 | hg archive -r <rev> <output.tar.gz> | git archive -o <output.tar> <commit> |
| | | git archive -o <output.tar> –remote=<url> <commit> |
| 反转提交 | hg backout <rev> | git revert <commit> |
| 提交拣选 | - | git cherry-pick <commit> |
| 分支合并 | hg merge <rev> | git merge <commit> |
| 变基 | hg rebase | git rebase |
| 冲突解决 | hg resolve –tool=<tool> | git mergetool |
| | hg resolve -m <path> | git add <path> |
| 更改提交说明 | Hg + MQ | git commit –amend |
| 撤消最后一次提交 | hg rollback | git reset [ –soft \| –hard ] HEAD^ |
| 撤消多次提交 | Hg + MQ | git reset [ –soft \| –hard ] HEAD~<n> |
| 撤消历史提交 | Hg + MQ | git rebase -i <commit>^ |
| 启动Web浏览 | hg serve | git instaweb |
| 二分查找 | hg bisect | git bisect |
| 内容搜索 | hg grep | git grep |
| 提交导出补丁文件 | hg export | git format-patch |
| 工作区根目录 | hg root | git rev-parse –show-toplevel |
| 杂项 | .hgignore 文件 | .gitignore 文件 |
| | pager 扩展 | 内置分页器 |
| | color 扩展 | color.* 配置变量 |
| | mq 扩展 | StGit, Topgit |
| | graphlog 扩展 | git log –graph |
| | hgk 扩展 | gitk |