子树合并

现在你已经看到了子模块系统的麻烦之处，让我们来看一下解决相同问题的另一途径。当 Git 归并时，它会检查需要归并的内容然后选择一个合适的归并策略。如果你归并的分支是两个，Git使用一个*递归*策略。如果你归并的分支超过两个，Git采用*章鱼*策略。这些策略是自动选择的，因为递归策略可以处理复杂的三路归并情况——比如多于一个共同祖先的——但是它只能处理两个分支的归并。章鱼归并可以处理多个分支但是但必须更加小心以避免冲突带来的麻烦，因此它被选中作为归并两个以上分支的默认策略。

实际上，你也可以选择其他策略。其中的一个就是*子树*归并，你可以用它来处理子项目问题。这里你会看到如何换用子树归并的方法来实现前一节里所做的 rack 的嵌入。

子树归并的思想是你拥有两个工程，其中一个项目映射到另外一个项目的子目录中，反过来也一样。当你指定一个子树归并，Git可以聪明地探知其中一个是另外一个的子树从而实现正确的归并——这相当神奇。

首先你将 Rack 应用加入到项目中。你将 Rack 项目当作你项目中的一个远程引用，然后将它检出到它自身的分支：

$ git remote add rack\_remote git@github.com:schacon/rack.git

$ git fetch rack\_remote

warning: no common commits

remote: Counting objects: 3184, done.

remote: Compressing objects: 100% (1465/1465), done.

remote: Total 3184 (delta 1952), reused 2770 (delta 1675)

Receiving objects: 100% (3184/3184), 677.42 KiB | 4 KiB/s, done.

Resolving deltas: 100% (1952/1952), done.

From git@github.com:schacon/rack

\* [new branch] build -> rack\_remote/build

\* [new branch] master -> rack\_remote/master

\* [new branch] rack-0.4 -> rack\_remote/rack-0.4

\* [new branch] rack-0.9 -> rack\_remote/rack-0.9

$ git checkout -b rack\_branch rack\_remote/master

Branch rack\_branch set up to track remote branch refs/remotes/rack\_remote/master.

Switched to a new branch "rack\_branch"

现在在你的rack\_branch分支中就有了Rack项目的根目录，而你自己的项目在master分支中。如果你先检出其中一个然后另外一个，你会看到它们有不同的项目根目录：

$ ls

AUTHORS KNOWN-ISSUES Rakefile contrib lib

COPYING README bin example test

$ git checkout master

Switched to branch "master"

$ ls

README

要将 Rack 项目当作子目录拉取到你的master项目中。你可以在 Git 中用git read-tree来实现。你会在第9章学到更多与read-tree和它的朋友相关的东西，当前你会知道它读取一个分支的根目录树到当前的暂存区和工作目录。你只要切换回你的master分支，然后拉取rack\_branch到你主项目的master分支的rack子目录：

$ git read-tree --prefix=rack/ -u rack\_branch

当你提交的时候，看起来就像你在那个子目录下拥有Rack的文件——就像你从一个tarball里拷贝的一样。有意思的是你可以比较容易地归并其中一个分支的变更到另外一个。因此，如果 Rack 项目更新了，你可以通过切换到那个分支并执行拉取来获得上游的变更：

$ git checkout rack\_branch

$ git pull

然后，你可以将那些变更归并回你的 master 分支。你可以使用git merge -s subtree，它会工作的很好；但是 Git 同时会把历史归并到一起，这可能不是你想要的。为了拉取变更并预置提交说明，需要在-s subtree策略选项的同时使用--squash和--no-commit选项。

$ git checkout master

$ git merge --squash -s subtree --no-commit rack\_branch

Squash commit -- not updating HEAD

Automatic merge went well; stopped before committing as requested

所有 Rack 项目的变更都被归并并且可以进行本地提交。你也可以做相反的事情——在你主分支的rack目录里进行变更然后归并回rack\_branch分支，然后将它们提交给维护者或者推送到上游。

为了得到rack子目录和你rack\_branch分支的区别——以决定你是否需要归并它们——你不能使用一般的diff命令。而是对你想比较的分支运行git diff-tree：

$ git diff-tree -p rack\_branch

或者，为了比较你的rack子目录和服务器上你拉取时的master分支，你可以运行

$ git diff-tree -p rack\_remote/master