FreeSaber
在这个世界上，没有什么事儿是一行代码解决不了的，如果有，那就是两行！

Git本地仓库

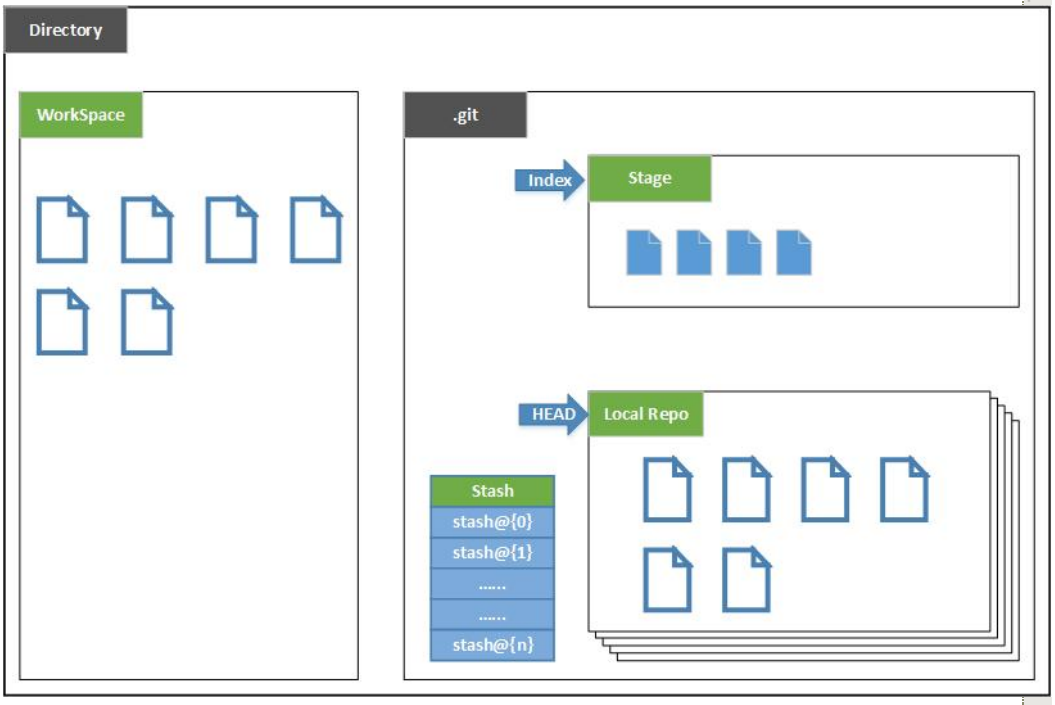
原文：http://www.cnblogs.com/wilber2013/p/4189920.html

Git基本概念

在Git中，我们将需要进行版本控制的文件目录叫做一个仓库（**repository**），每个仓库可以简单理解成一个目录，这个目录里面的所有文件都通过Git来实现版本管理，Git都能跟踪并记录在该目录中发生的所有更新。

现在我们已经知道什么是repository（缩写repo）了，假如我们现在建立一个仓库（repo），那么在建立仓库的这个目录中有一个“.git”的文件夹。这个文件夹非常重要，所有的版本信息，更新记录，以及Git进行仓库管理的相关信息全部保存在这个文件夹里面。所以，不要修改/删除其中的文件，以免造成数据的丢失。

进一步的讲解请参考下面一张图，大概展示出了我们需要了解的基本知识。



根据上面的图片，下面给出了每个部分的简要说明：

- Directory：使用Git管理的一个目录，也就是一个仓库，包含我们的工作空间和Git的管理空间。
- WorkSpace：需要通过Git进行版本控制的目录和文件，这些目录和文件组成了工作空间。
- .git：存放Git管理信息的目录，初始化仓库的时候自动创建。
- Index/Stage：暂存区，或者叫待提交更新区，在提交进入repo之前，我们可以把所有的更新放在暂存区。
- Local Repo：本地仓库，一个存放在本地的版本库；HEAD会只是当前的开发分支（branch）。
- Stash：是一个工作状态保存栈，用于保存/恢复WorkSpace中的临时状态。

有了上面概念的了解，下面就开始在本地repo上进行Git操作了。

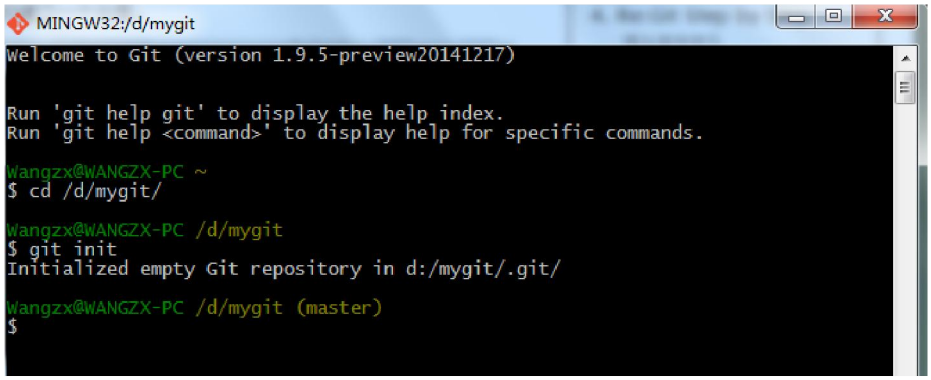
创建仓库

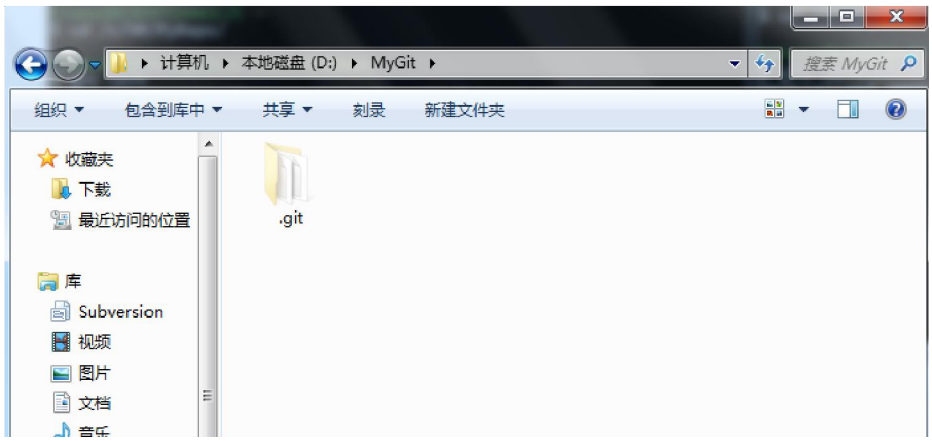
通过“Git Bash”命令行窗口进入到想要建立版本仓库的目录，通过“git init”就可以轻松的建立一个仓库。

5. C#调用BarTender .NET SDK 打印条码(5)

推荐排行榜

1. CSS浮动(24)
2. 数据库水平切分的实现原理解析——分库，分表，主从，集群，负载均衡器（转）(14)
3. Chrome 控制台console的用法(9)
4. MVC中的Repository模式(8)
5. 小白也能用Git管理团队项目了：百度云同步+Git Extensions+Git Source Control Provider(6)
6. 一个简单的图片监听和上传程序(4)
7. 发布在IIS上的Web程序，调用服务器的COM组件(4)
8. JavaScript事件基础知识总结【思维导图】(3)
9. Git本地仓库(3)
10. HTML中Meta标签详解以及meta property=og标签含义(3)



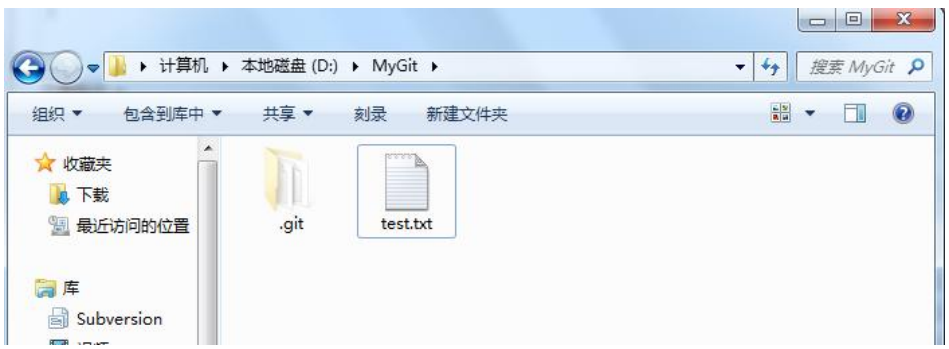


这时，我们的仓库中会自动的产生一个“.git”文件夹，这个就是我们前面提到的Git管理信息的目录。

添加

现在我们在仓库中新建一个“test.txt”的文本文件，内容如下。

```
123
123
```



通过"git status"可以查看WorkSpace的状态，看到输出显示"test.txt"没有被Git跟踪，并且提示我们可以使用"git add <file>..."把该文件添加到待提交区（暂存区）。

注意，如果添加到暂存区，这时的更新只是在**WorkSpace**中。

```
Wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test.txt

nothing added to commit but untracked files present (use "git add" to track)

Wangzx@WANGZX-PC /d/mygit (master)
$ _
```

使用"git add test.txt"或者"git add ."，然后继续查看WorkSpace的状态。这时发现文件已经被放到暂存区。

这时的更新已经从**WorkSpace**保存到**Stage**中。

```
Wangzx@WANGZX-PC /d/mygit (master)
$ git add .

Wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.txt


Wangzx@WANGZX-PC /d/mygit (master)
$
```

最后，我们就可以通过“git commit -m”来提交更新了。-m后面跟的是对commit的描述（message）。

这时的更新已经从**Stage**保存到了**Local Repo**中。

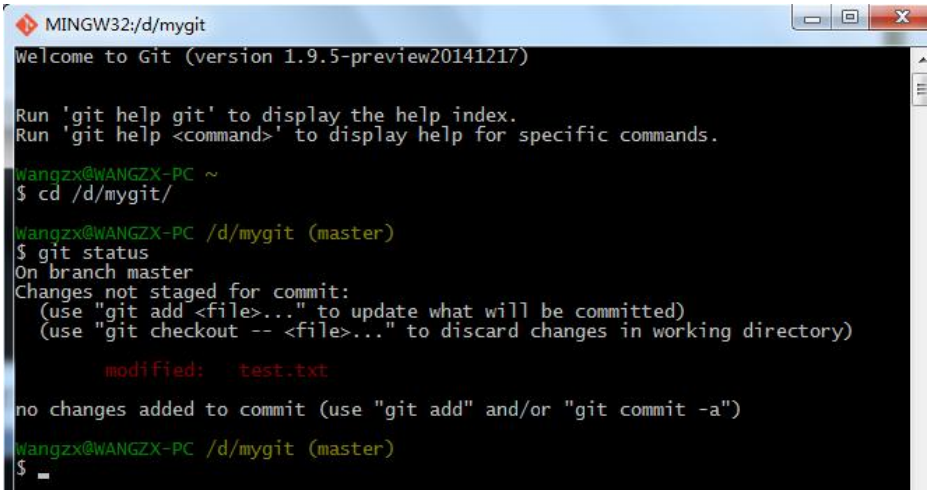
```
Wangzx@WANGZX-PC /d/mygit (master)
$ git commit -m "add test.txt into repo"
[master (root-commit) b259533] add test.txt into repo
 1 file changed, 2 insertions(+)
 create mode 100644 test.txt

Wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master
nothing to commit, working directory clean

Wangzx@WANGZX-PC /d/mygit (master)
$ _
```

通过上面的操作，文件"test.txt"就成功的被添加到了仓库中。

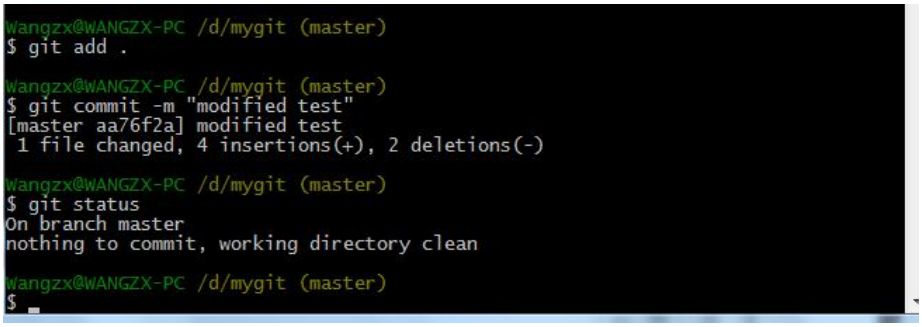
更新

假设现在需要对"test.txt"进行更新，修改文件后，查看WorkSpace的状态，会发现提示文件有更新，但是更新只是在WorkSpace中，没有到暂存区中。

```
莫语常言道知足，万事至终总是空。
理想现实一线隔，心无旁骛脚踏实。
谁无暴风劲雨时，守得云开见月明。
花开复见却飘零，残憾莫使今生留。
```



同样，通过add、commit的操作，我们可以把文件的更新先放到暂存区，然后从暂存区提交到repo中。



注意，只有被**add**到暂存区的更新才会被提交进入**repo**。提交前，如果对WorkSpace的文件进行修改，而没有被添加到暂存区，那么提交进repo中的只是暂存区的更新，WorkSpace修改的部分不会提交进repo中的。

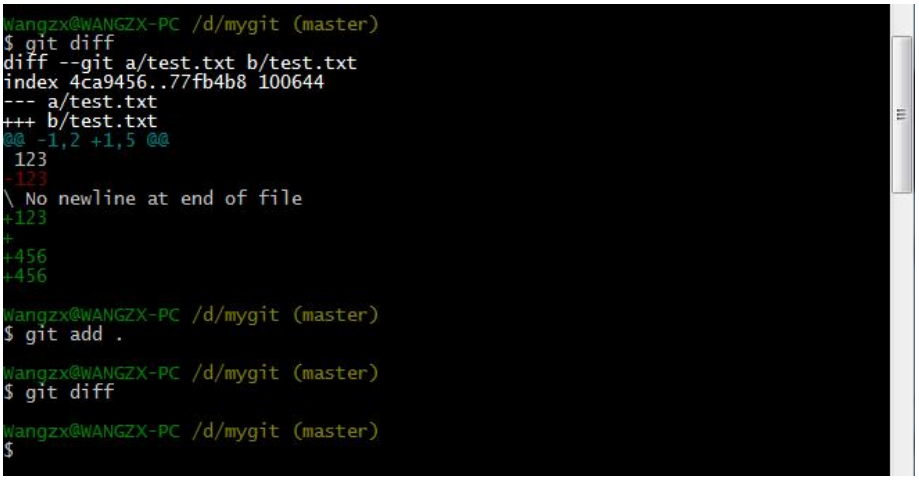
git diff

"git diff"是个很有用，而且会经常用到的命令，用于显示WorkSpace中的文件和暂存区文件的差异。先将我们之前的test.txt文件中的内容改成纯数字然后提交到**repo**中，中文字符运行该命令会显示乱码。

```
123
123
```

接下来修改成下面内容，我们通过"git diff"可以查看WorkSpace和Stage的diff情况，当我们把更新add到Stage中，diff就不会有任何输出了。

```
123
123

456
456
```



当然，我们也可以把WorkSpace中的状态和repo中的状态进行diff，命令如下。

```
git diff HEAD~n
```

撤销更新

根据前面对基本概念的了解，更新可能存在三个地方，WorkSpace中，Stage中和repo中。下面就分别介绍一下怎么撤销这些更新。

撤销**WorkSpace**中的更新

接着上面的例子，先将test.txt中的文本设置如下内容，并提交。

```
abc
abc
```

然后将test.txt中的内容进行修改：

```
abc
123
bca
321
```

```
Wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")

Wangzx@WANGZX-PC /d/mygit (master)
$
```

我们可以使用"git checkout --<file>..."来撤销WorkSpace中的更新。执行test.txt中的文本会变成两行"abc"。

```
Wangzx@WANGZX-PC /d/mygit (master)
$ git checkout -- test.txt

Wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master
nothing to commit, working directory clean

Wangzx@WANGZX-PC /d/mygit (master)
$
```

注意：使用这种方法撤销更新的时候一定要慎重，因为通过这种方式撤销后，更新将没有办法再找回。

撤销**Stage**中的更新

由于上个步骤，我们将test.txt修改撤销了。这里再次修改为下面内容，并且使用了"git add"把这个更新提交到了暂存区。这时，"git status"的输出中提示我们可以通过"git reset HEAD <file>..."把暂存区的更新移出到WorkSpace中。

如果想继续撤销WorkSpace中的更新，请参考上面一步。

```
abc
123
cba
321
```

```
wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")

wangzx@WANGZX-PC /d/mygit (master)
$ git add test.txt

wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   test.txt


wangzx@WANGZX-PC /d/mygit (master)
$ git reset HEAD test.txt
Unstaged changes after reset:
M       test.txt

wangzx@WANGZX-PC /d/mygit (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   test.txt

no changes added to commit (use "git add" and/or "git commit -a")

wangzx@WANGZX-PC /d/mygit (master)
$
```

撤销**repo**中的更新

介绍撤销repo中的更新之前，我们先看一下"git log"这个命令，通过这个命令我们可以查看commit的历史记录。

```
wangzx@WANGZX-PC /d/mygit (master)
$ git log
commit c2760c5512bc67a8b99b0c1da508d40cca623f23
Author: wangzx <saber.lover@qq.com>
Date:   Tue Jan 6 14:41:34 2015 +0800

    add

commit f75257012c00a79bd9974903b3fccf564c925e0f
Author: wangzx <saber.lover@qq.com>
Date:   Tue Jan 6 14:11:34 2015 +0800

    add

wangzx@WANGZX-PC /d/mygit (master)
$
```

撤销提交有两种方式：使用**HEAD**指针和使用**commit id**

在Git中，有一个HEAD指针指向当前分支中最新的提交。当前版本，我们使用"HEAD^"，那么再钱一个版本可以使用"HEAD^^"，如果想回退到更早的提交，可以使用"HEAD~n"。（也就是，HEAD^=HEAD~1，HEAD^^=HEAD~2）

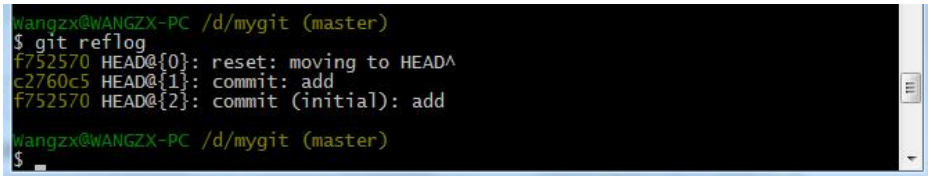
```
git rest --hard HEAD^
git rest --hard HEAD~1
git rest --c2760c5512bc67a8b990c1da508d40cca623f23
```

```
wangzx@WANGZX-PC /d/mygit (master)
$ git reset --hard HEAD^
HEAD is now at f752570 add

Wangzx@WANGZX-PC /d/mygit (master)
$ git log
commit f75257012c00a79bd9974903b3fccf564c925e0f
Author: wangzx <saber.lover@qq.com>
Date:   Tue Jan 6 14:11:34 2015 +0800

    add

Wangzx@WANGZX-PC /d/mygit (master)
$
```

再次查看，发现最新的提交已经被撤销了，查看test.txt中的文件，发现内容又变成两行"abc"文本。

那么问题就来了，我现在又想恢复被撤销的提交，当然Git是支持这样的操作。

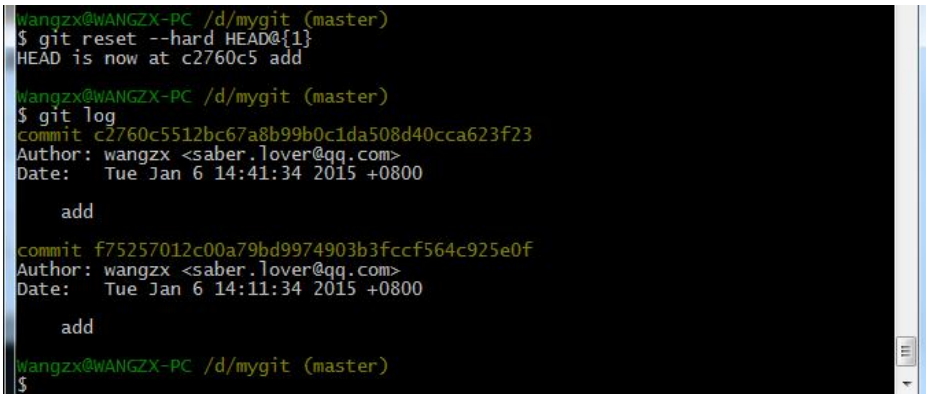
下面来看看"git reflog"这个命令。"git log"只是包括了当前分支中的commit记录，而"git reflog"中会记录这个仓库中所有的分支的所有更新记录，包括已经撤销的更新。



有了这个，我们就可以恢复撤销操作。

```
git reset --hard HEAD@{1}
git reset --hard f752570
```

再次查看，发现我们撤销的内容已经回来了。



--hard和--soft

前面在使用reset来撤销更新的时候，我们都是使用的"--head"选项，其实与之对应的还有一个"--soft"选项，区别如下：

--head：撤销并删除相应的更新

--soft：撤销相应的更新，把这些更新的内容放到Stage中

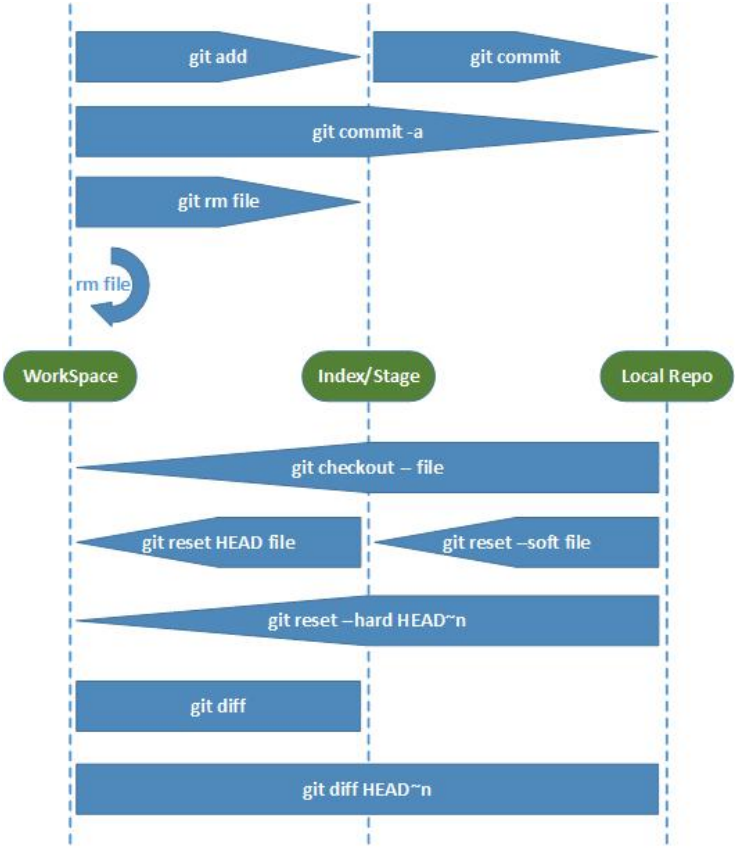
删除文件

在Git中，如果我们要删除一个文件，可以使用下面的命令，"git rm"相比"rm"只是多了一步，把这次删除的更新发到Stage中。

```
rm <file>
```

```
git rm <file>
```

总结



FreeSaber
关注 - 80
粉丝 - 187
+加关注

« 上一篇：Git简介
» 下一篇：触发器实现跨服务器

posted on 2015-01-06 10:04 FreeSaber 阅读(39286) 评论(2) 编辑 收藏

评论：

#1楼 2015-12-17 14:31 | pandajava
赞！

支持(0) 反对(0)

#2楼 2017-01-15 17:41 | lbwdev
很好的一篇文章，对我有帮助

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部