BUG分支

软件开发中，bug就像家常便饭一样。有了bug就需要修复，在Git中，由于分支是如此的强大，所以，每个bug都可以通过一个新的临时分支来修复，修复后，合并分支，然后将临时分支删除。

当你接到一个修复一个代号101的bug的任务时，很自然地，你想创建一个分支issue-101来修复它，但是，等等，当前正在dev上进行的工作还没有提交：

因为工作区和暂存区只有一个，如果dev分支中还有a.txt到z.txt没有提交，并且他们还处于测试中，不是完全形态，不能提交（如果提交了被别的队员下载了，就会出现隐患）。那么这些文件就在暂存区（在工作区的话，输入git stash会报错No local changes to save），所有分支都可以看到它们，如果我修复完bug以后又新添了A.txt到Z.txt文档，我想要提交的时候，就不能直接提交了，因为暂存区中还存在很多的a.txt到z.txt不是这次bug修复所需要的文件，而且如果我直接提交到bug分支，在dev分支中就不存在这些文件了（亲测）。

幸好，Git还提供了一个stash功能，可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作

我先给master分支上的masterfile.docx加上一句 git is fre并commit

然后新建一个dev分支，新建一个devStage!!!.docx,然后add到暂存区（只有在暂存区中才可以stash,不然会报错）

现在切换到master分支，因为需要是master分支需要debug

$ git checkout master

Switched to branch 'master'

然后新建并切换到issue-101分支

$ git checkout -b issue-101

Switched to a new branch 'issue-101'

把masterfile.docx中的内容改为git is free然后add并commit

$ git add masterfile.docx

$ git commit -m"fix bug 101"

[issue-101 d16d7ad] fix bug 101

1 file changed, 0 insertions(+), 0 deletions(-)

现在切换回master,准备合并分支

$ git checkout master

Switched to branch 'master'

$ git merge --no-ff -m "merged bug fix 101" issue-101

Merge made by the 'recursive' strategy.

masterfile.docx | Bin 11597 -> 11648 bytes

1 file changed, 0 insertions(+), 0 deletions(-)

好了，可以删除issue-101分支了

$ git branch -d issue-101

Deleted branch issue-101 (was d16d7ad).

现在切换回dev 分支

$ git checkout dev

Switched to branch 'dev'

工作区是干净的

$ git status

On branch dev

nothing to commit, working tree clean

使用git stash list命令查看，它被存储起来了，需要恢复一下

$ git stash list

stash@{0}: WIP on dev: 66eef56 devStage

有两个办法：

一是用git stash apply恢复，但是恢复后，stash内容并不删除，你需要用git stash drop来删除；

另一种方式是用git stash pop，恢复的同时把stash内容也删了：

$ git stash pop

On branch dev

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: devStage!!!.docx

再查看一下情况，

$ git status

On branch dev

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: devStage!!!.docx

文件又出现了！

再用git stash list查看，就看不到任何stash内容了：

$ git stash list

可以多次stash，恢复的时候，先用git stash list查看，然后恢复指定的stash，用命令：

$ git stash apply stash@{0}