

# Java基础

## 1.1

# 设计模式

# 数据结构

# Android基础

# Git使用

## Git基本概念

版本控制：记录若干文件变化，以便将来查阅特定版本修订情况的系统。

文件的三种状态：对任何一个文件，在git内都有三种状态：已提交（committed），已修改（modified），已暂存（staged）。已提交：表示该文件已经被安全的保存到了本地数据库中；已修改：表示已经修改了某个文件，但是还没有提交保存；已暂存：表示把已修改的文件放在下次提交时要保存的清单中

每个项目都有一个Git目录（.git）它是Git用来保存元数据和对象数据的地方，每次克隆镜像仓库的时候，实际拷贝的就是这个目录里面的数据

Git工作流程：在工作目录中修改某些文件；对修改后的文件进行快照，然后保存到暂存区域；提交更新，将保存在暂存区域的文件快照永久转存到Git目录中。

对现有的某个项目开始使用git管理：git init；.git文件是隐藏的，用于保存下载下来的所有版本记录想要看里面的内容需要在文件夹下显示。

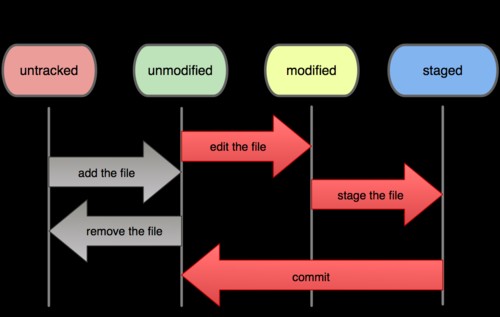
如果当前目录有几个文件想要纳入版本控制，需要git add命令告诉git开始追踪这些文件，然后提交

从现有仓库克隆：git clone [url]；

$ git clone git://github.com/schacon/grit.git mygrit

git支持许多数据传输协议，git：//；http(s)://；user2server：/path.git表示的ssh传输协议

文件状态生命周期：已跟踪和未跟踪，已跟踪的文件时指本来就纳入版本控制管理的文件。



git status命令：用于检查当前文件的状态

git add命令：用于跟踪新文件

git commit命令：用于提交文件

git diff命令：查看不同（difference）

git log命令：显示从最近到最远的提交日志

git reset --hard HEAD^：退回到上一个版本，如果想在退回到上一个版本之后再回到退回之前的版本，git reset --hard [commit id](退回之前的版本的commit ID的前几位即可，通过git log可以查看)，HEAD表示当前版本，HEAD^表示上一个版本，上上一个版本是HEAD^^表示，往上100个版本用HEAD~100表示

git reflog查看提交历史，查看记录每一次命令，前面显示的是commit id，可以通过commit id恢复到以往的任意一个版本

git checkout -- file：恢复到最近一次git commit或者git add之前的状态

cat：查看文档内容

## 建立本地git和远程GitHub仓库的联系

具体怎么创建联系的参照《Git教程By廖雪峰》

git仓库和GitHub之间的传输是通过SSH加密的

git push -u origin master：第一次推送master分支所有内容

git push origin master：推送最新修改

要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快

.gitignore文件：是列出要忽略的文件模式

所有空行或者以注释符号#开头的行都会被Git忽略

/说明要忽略的是目录；

\*匹配零个或多个任意字符

[abc]匹配任何一个列在方括号中的内容，[0-9]表示所有在这两个字符范围内的都可以匹配

？只匹配一个任意字符

## 分支管理

查看分支：git branch

创建分支：git branch name

切换分支：git checkout name

创建+切换分支：git checkout -b name

合并某分支到当前分支：git merge name

删除分支：git branch -d name

解决冲突

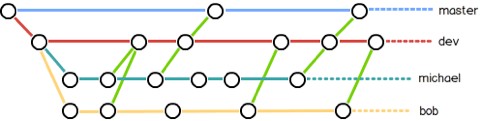
当git无法自动合并分支时，就必须首先解决冲突，解决冲突后，再提交，合并完成

git log --graph命令可以看到分支合并图

## 分支策略

master分支是非常稳定的，仅仅用来发布新版本，平时不能在上面干活，

干活都在dev分支上，dev分支是不稳定的，到某个版本发布时，再把dev分支合并到master分支上，在master分支发布11.0版本



合并分支，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

## Bug分支

修复bug时，通过创建新的bug分支进行修复，然后合并，最后删除

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场

## Feature分支

开发新功能，最好建立一个新分支，如果要丢弃一个没有被合并过的分支，可以通过git branch -D name强行删除

## 多人协作

查看远程库信息，使用git remote -v

本地新建的分支如果不推送到远程，对其他人就是不可见的

从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交

在本地创建和远程分支对应的分支，使用git checkout -b branch-name origin/branch-name，本地分支和远程分支的名字最好一致

建立本地分支和远程分支的关联，使用 gitbranch --set-upstream branch-name origin/branch-name

从远程抓取分支，使用git pull，如果有冲突，要首先处理冲突。

## 创建标签

命令git tag name 用于新建一个标签，默认HEAD，也可制定一个commit id

-a tagname -m "bababababab"可以指定标签信息

命令git tag可以查看所有标签

## 操作标签

命令git push origin tagname可以推送一个本地标签

命令git push origin --tags可以推送全部未推送过的本地标签

命令git tag -d tagname 可以删除一个本地标签

命令git push origin ：refs/tags/tagname可以删除一个远程标签

## 使用github

github上可以任意Fork开源仓库

自己拥有Fork后的仓库的读写权限

可以推送pull request给官方仓库来贡献代码

# adb使用

# 待学习

# 专利