The first time beginning to use GIT

Part One: Initial

1. In local project. - do initialization

**git init**

qians-mbp:learn qianqian$ git init

Initialized empty Git repository in /Users/qianqian/IdeaProjects/learn/.git/

1. Check file status

**git status**

qians-mbp:learn qianqian$ git status

On branch master

Initial commit

Untracked files:

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

.gradle/

.idea/

build.gradle

build/

gradle/

gradlew

gradlew.bat

settings.gradle

src/

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

1. Add git ignore file in the project root path

.gitignore

.idea/

.gradle/

build/

1. Add files to local git

**git add .**

qians-mbp:learn qianqian$ git add .

qians-mbp:learn qianqian$ git status

On branch master

Initial commit

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: .gitignore

new file: build.gradle

new file: gradle/wrapper/gradle-wrapper.jar

new file: gradle/wrapper/gradle-wrapper.properties

new file: gradlew

new file: gradlew.bat

new file: settings.gradle

new file: src/main/java/Fibonacci.java

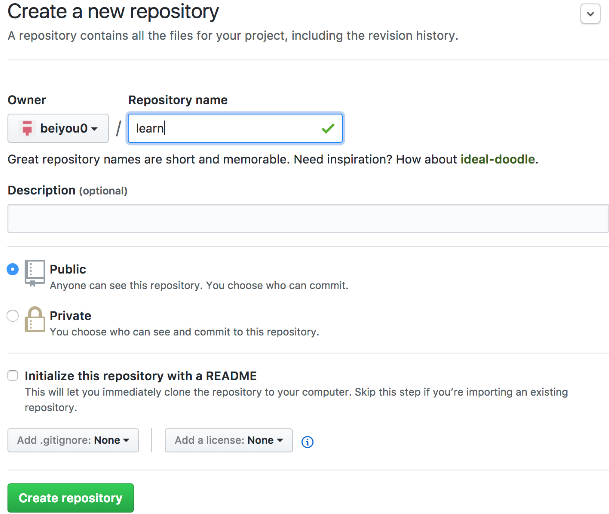
new file: src/main/java/Searching.java

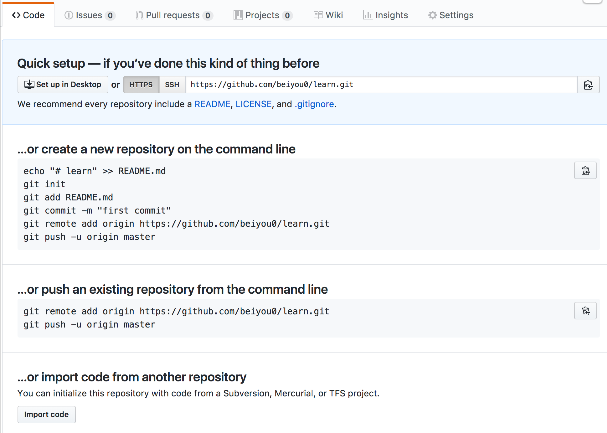
new file: src/main/java/Sorting.java

1. 在远程仓库创建repository

<https://github.com/>

[beiyou0@163.com](mailto:beiyou0@163.com)/\*\*\*\*\*5r\*





1. 本地工程目录下，创建远程代码仓库链接

git remote add origin [git@github.ibm.com:DSWESB/kafkaTesting.git](https://github.com/beiyou0/learn.git)

git remote -v // 查看remote git repository

git push -u origin master

Part Two: Git delete file

Git删除远程文件/文件夹（本地文件保留）

<http://blog.csdn.net/github_32658299/article/details/54137990>

<https://stackoverflow.com/questions/2047465/how-can-i-delete-a-file-from-git-repo>

先声明一点，如果要同时删除本地的和**github**上的文件，直接删除本地的再**push**就行了，比较简单。这里的要求是不能删除本地的文件，而要删除**github**里，就是网页上的文件。

其实质就是删除缓冲区里的文件，再提交给服务器端。

1.首先进入要删除的文件夹或文件的根目录下，如F:\myprojects\supermarketmanager1115

2.执行下面的语句”some-directory”是相对于本地根目录下的文件夹/文件路径

git ls-files (list all files which added to version control)

git rm -r --cached some-directory

git commit -m 'Remove the now ignored directory "some-directory"'

git push origin master

另外[**Git**](http://lib.csdn.net/base/git) pull --rebase origin master这句挺有用的，记录一下。这句的意思是：把github上最新的文件下载下来。

Part Three: Git ignore

英文官网： <https://git-scm.com/docs/gitignore>

<http://blog.csdn.net/hustpzb/article/details/8649545>

在[**Git**](http://lib.csdn.net/base/git)的[版本控制](http://lib.csdn.net/base/git)中，可能有些文件是不需要加入控制的，那我们在提交代码时就需要忽略这些文件，下面讲讲应该怎么给[**git**](http://lib.csdn.net/base/git)配置一些忽略规则。

有三种方法可以忽略掉这些文件，这三种方法都能达到目的，只不过适用情景不一样。

(1).针对单一工程排除文件，这种方式会让这个工程的所有修改者在克隆代码的同时，也能克隆到过滤规则，而不用自己再写一份，这就能保证所有修改者应用的都是同一份规则，而不是张三自己有一套过滤规则，李四又使用另一套过滤规则，个人比较喜欢这个。配置步骤如下：

在工程根目录下建立.gitignore文件，将要排除的文件或目录 写到.gitignore这个文件中，有两种写入方法。

(a)使用命令行增加排除文件    
   
排除以.class结尾的文件  echo “\*.class” >.gitignore (>> 是在文件尾增加,> 是删除已经存在的内容再增加)，之后会在当前目录下生成一个.gitignore的文件。  
排除bin目录下的文件     echo “bin/” >.gitignore  
  
(b)最方便的办法是，用记事本打开，增加需要排除的文件或目录，一行增加一个，如：

  \*.class  
  \*.apk  
  bin/  
  gen/  
  .settings/  
  proguard/  
    
(2).全局设置排除文件，这会在全局起作用，只要是Git管理的工程，在提交时都会自动排除不在控制范围内的文件或目录。这种方法对开发者来说，比较省事，只要一次全局配置，不用每次建立工程都要配置一遍过滤规则。但是这不保证其他的开发者在克隆你的代码后，他们那边的规则跟你的是一样的，这就带来了代码提交过程中的各种冲突问题。配置步骤如下：

(a）像方法（1）一样，也需要建立一个.gitignore文件，把要排除的文件写进去。

        (b) 但在这里，我们不规定一定要把.gitnore文件放到某个工程下面，而是任何地方，比如我们这里放到了Git默认的Home路径下，在我的windows上就是C:\Users\zhbpeng

        (c)使用命令方式可以配置全局排除文件 git config --global core.excludesfile ~/.gitignore，你会发现在~/.gitconfig文件中会出现excludesfile = c:/Users/zhbpeng/.gitignore。说明Git把文件过滤规则应用到了Global的规则中。  
  
  
(3). 单个工程设置排除文件，在工程目录下找到.git/info/exclude，把要排除的文件写进去：  
  \*.class  
  \*.apk  
  bin/  
  gen/  
  .settings/  
  proguard/

这种方法就不提倡了，只能针对单一工程配置，而且还不能将过滤规则同步到其他开发者，跟方法（1）（2）比较起来没有一点优势。

Part Four: Git config

1. Git 配置文件

（1) 仓库级配置文件—— config, 该文件位于当前仓库下, 路径.git/。这个配置中的设置只对当前所在仓库有效。

qians-mbp:.git qianqian$ pwd

/Users/qianqian/IdeaProjects/iibtesting/.git

qians-mbp:.git qianqian$ cat config

[core]

bare = false

filemode = true

ignorecase = true

precomposeunicode = true

logallrefupdates = true

[core]

repositoryformatversion = 0

[remote "origin"]

url = git@github.ibm.com:DSWESB2/iibtesting.git

fetch = +refs/heads/\*:refs/remotes/origin/\*

(2) 全局级别配置文件：/Users/qianqian/.gitconfig

qians-mbp:~ qianqian$ pwd

/Users/qianqian

qians-mbp:~ qianqian$ cat .gitconfig

[user]

name = Qian Qian Liu

email = [liuqqbj@cn.ibm.com](mailto:liuqqbj@cn.ibm.com)

我们可以发现，有些配置项的名称是一样的，比如“core”，在git中，每一项的配置名称又叫做“section”，“section”下面的每一行对应着一个key和一个value。

1. 配置文件如何生效

       用一个我们大家都熟悉的例子来比较，就是css。一般来说，在一个页面中引入css文件的方法有三种，见：<http://www.cnblogs.com/hutaoer/archive/2010/12/07/1899105.html>，

1） 通俗的讲就是行内样式“<div style=“width:200px”>”，

2） 内嵌样式“<style type=“text/css”>div{width:200px}</style>”

3） 和外部样式“<link rel=“stylesheet” href=“div.css”>”。

       这三种样式的权重是行内>内嵌>外部。类似的，对于git来说，配置文件的权重是仓库>全局>系统。Git会使用这一系列的配置文件来存储你定义的偏好。

1. 用 git config命令查看配置文件

**qians-mbp:etc qianqian$ git config -h**

**usage: git config [<options>]**

**Config file location**

**--global              use global config file**

**--system              use system config file**

**--local               use repository config file**

**-f, --file <file>     use given config file**

**--blob <blob-id>      read config from given blob object**

**…**

qians-mbp:etc qianqian$ git config --global -l

user.name=Qian Qian Liu

[user.email=liuqqbj@cn.ibm.com](mailto:user.email=liuqqbj@cn.ibm.com)

qians-mbp:iibtesting qianqian$ pwd

/Users/qianqian/IdeaProjects/iibtesting

:iibtesting qianqian$ git config --local -l

core.bare=false

core.filemode=true

core.ignorecase=true

core.precomposeunicode=true

core.logallrefupdates=true

core.repositoryformatversion=0

remote.origin.url=git@github.ibm.com:DSWESB2/iibtesting.git

remote.origin.fetch=+refs/heads/\*:refs/remotes/origin/\*

1. 使用 git config命令编辑配置文件

编辑的英文单词是什么，没错，edit

命令参数 –edit, 简写 -e

格式：git config [--local|--global|--system] -e

1. 使用 git config 增加一个配置项

参数

 --add

格式: git config [--local|--global|--system] --add section.key value(默认是添加在local配置中)

e.g.   git config --add cat.name tom

1. 获取一个配置项

有时候，我们并不需要查看所有配置的值，而是查看某个配置项的值，怎么做呢？

命令参数 --get

格式：git config [--local|--global|--system] --get section.key(默认是获取local配置中内容)

我们先往global配置中写入一个cat.name=Tomcat的配置项，再使用git config --get cat.name看看得到的是什么

1. 删除一个配置项

命令参数

 --unset

格式：git config [--local|--global|--system] --unset section.key

相信有了前两个命令的使用基础，大家举一反三就知道改怎么用了，来，我们试试删除local配置中的cat.name