关于git目录进行代码管理说明

由于实验室开发初步框架已经出现，因此我提出使用git和gitblit对代码版本进行控制。以下为git的介绍以及使用说明：

**1.关于版本控制**

什么是版本控制？我为什么要关心它呢？版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。在本书所展示的例子中，我们仅对保存着软件源代码的文本文件作版本控制管理，但实际上，你可以对任何类型的文件进行版本控制。

如果你是位图形或网页设计师，可能会需要保存某一幅图片或页面布局文件的所有修订版本（这或许是你非常渴望拥有的功能）。采用版本控制系统（VCS）是个明智的选择。有了它你就可以将某个文件回溯到之前的状态，甚至将整个项目都回退到过去某个时间点的状态。你可以比较文件的变化细节，查出最后是谁修改了哪个地方，从而找出导致怪异问题出现的原因，又是谁在何时报告了某个功能缺陷等等。使用版本控制系统通常还意味着，就算你乱来一气把整个项目中的文件改的改删的删，你也照样可以轻松恢复到原先的样子。但额外增加的工作量却微乎其微。

**2.版本控制工具----git介绍**

Git 基础

那么，简单地说，Git 究竟是怎样的一个系统呢？请注意，接下来的内容非常重要，若是理解了 Git 的思想和基本工作原理，用起来就会知其所以然，游刃有余。在开始学习 Git 的时候，请不要尝试把各种概念和其他版本控制系统（诸如 Subversion 和 Perforce 等）相比拟，否则容易混淆每个操作的实际意义。Git 在保存和处理各种信息的时候，虽然操作起来的命令形式非常相近，但它与其他版本控制系统的做法颇为不同。理解这些差异将有助于你准确地使用 Git 提供的各种工具。

Git 和其他版本控制系统的主要差别在于，Git 只关心文件数据的整体是否发生变化，而大多数其他系统则只关心文件内容的具体差异。这类系统（CVS，Subversion，Perforce，Bazaar 等等）每次记录有哪些文件作了更新，以及都更新了哪些行的什么内容，

其他系统在每个版本中记录着各个文件的具体差异

Git 并不保存这些前后变化的差异数据。实际上，Git 更像是把变化的文件作快照后，记录在一个微型的文件系统中。每次提交更新时，它会纵览一遍所有文件的指纹信息并对文件作一快照，然后保存一个指向这次快照的索引。为提高性能，若文件没有变化，Git 不会再次保存，而只对上次保存的快照作一链接。

这是 Git 同其他系统的重要区别。它完全颠覆了传统版本控制的套路，并对各个环节的实现方式作了新的设计。Git 更像是个小型的文件系统，但它同时还提供了许多以此为基础的超强工具，而不只是一个简单的 VCS。稍后在第三章讨论 Git 分支管理的时候，我们会再看看这样的设计究竟会带来哪些好处。

**3.Git使用常用命令**

|  |
| --- |
| 0、git init 初始化    1、git add filename.txt  添加要往远程传的文件到一个临时列表    2、git commit -m "xxx"  # 说明为啥上传    3、git push origin master  # 上传    4、git clone https://github.com/triaquae/Weibo  # 克隆远程的项目    5、git status  查看修改结果    6、git log  查看修改过几次    6.1、git reflog  查看修改过后所有历史操作    7、git log --pretty=oneline  查看修改过的所有版本    8、git reset --hard 加版本  回滚到哪个版本    9、git reset --hard head^  回滚到上个版本    10、git reset --hard head^^  回滚到上上（回滚到上两个）个版本    11、git reset HEAD filename.txt  撤回提交到暂存区的操作    12、git checkout --filename.txt  撤销工作区操作（可以多个文件）    13、git rm filename.txt  删除暂存区    14、git checkout -b "dev"  创建分支（引号里是分支名）    15、git push origin xxx  提交到分支库（xxx表示分支名）    16、git branch  查看分支    16.1、git checkout dev  切换到分支库    17、git merge dev  合并分支库到主库（dev表示分支库）    18、git pull  拉远程库的文件    19、git stash  缓存工作区的内容    20、git stash list  查看暂存区的内容    21、git stash apply  把暂存区的内容恢复到工作区    22、git stash pop  把暂存区的内容恢复到工作区并且把暂存区的删除    23、版本回退：  git reset --hard HEAD : 回退到当前的版本  git reset --hard HEAD^ :  回退到上一个版本  git reset --hard HEAD^^ : 回退到上上个版本  git reset --hard HEAD~100 : 回退到前100版本  git reset --hard 版本号(前5位) |

4.gitblib 代码服务器的安装和配置

windows: <https://www.cnblogs.com/jeremylee/p/5626240.html>

linux:

<https://blog.csdn.net/fyc__ios/article/details/52262296>

osx:

<https://blog.csdn.net/pony_maggie/article/details/50880142>

5.gitblib客户端的安装

三种系统对应方法如下

https://git-scm.com/book/zh/v1/%E8%B5%B7%E6%AD%A5-%E5%AE%89%E8%A3%85-Git

**6.gitblib 和 git的使用**

6.1**本地初始化代码仓库以及使用**

1） ：新建文件夹；并使用命令行进入对应文件夹

2 ）：使用命令git init 初始化文件夹，即可生成代码仓库。

**6.2代码提交仓库：**

1）：使用git add +文件夹路径/文件路径 的命令；将文件添加缓存

2）：使用git commit –m “备注，XXXXXX” 的命令；将文件变化写在.git目录中，

3）：使用git log 命令；即可看见提交的记录。

**6.3 代码回溯**

由于版本提交的可能存在问题，需要回归到历史版本，因此使用如下流程进行

1）：使用命令行进入代码仓库文件夹的根目录

2）：使用git log命令；查看历史提交记录



3）：使用git checkout 加上历史commit代码；例如图中回溯到0.8骗人版；则使用



命令；这时版本将回溯到0.8骗人版，同时分支将指向master，就是分支主干。

**6.4 分支管理**

一般情况下，分支是用于测试，预发布，新功能实现的情况；为了防止将主干代码破坏，因此使用分支代码进行测试编写。

分支的建立如下：

1）：命令台/终端进入对应代码仓库文件夹

2）：使用git branch 分支名 建立新分支

3）：使用git switch 分支名 切换至对应分支

4）：使用git branch 就可以看到当前分支所在的地方

5）：之后的代码的提交和添加和之前的一样，这里就不细数了。

**6.5 代码发送到远程仓库以及下载代码**

发送代码：

1：

git remote add origin 你的远程库地址 // 把本地库与远程库关联

git push -u origin master // master为分支名，是主干，如果有其他的可以用其它

git push origin 分支名 // 第一次推送后，直接使用该命令即可推送修改

下载代码 ：

1：git clone你的远程库地址

推送分支至远程库

1：git checkout 分支名

2：git remote add origin 推送地址

3：git push origin 分支名

下载分支代码

1：进入工程根目录文件夹

2：执行命令： git clone xxx.git

3. 使用命令行进入工程目录

cd xxx

4. 切换到分支并下载代码

git checkout -b 远程仓库分支名字 origin/远程仓库分支名字

git pull