Git视频教程\_颜群

1. SVN的冲突问题如何避免和解决：

（1）及时提交，及时更新；

（2）每次提交更新前，右击项目-》team-》与资源库同步（可以看到资源库与本地的情况），具体如下：

a.灰色，说明本地比服务器更新，可以执行提交；

b.蓝色，说明服务器比本地更新，可以执行更新；

c.红色，说明服务器与本地的该文件有冲突，因此需先解决冲突；

1. GIT与SVN的区别：

（1）SVN：

a.是集中式版本控制系统；

b.所有的文件都集中放在中央仓库，因此使用SVN时必须确保网络正常；

（2）GIT：

a.是分布式版本控制系统；

b.每个本地都有一个版本库，说明如下：

a)本地工作目录

b)暂存区

c)版本库（本地仓库）

注意：本地工作目录可以多次添加进暂存区，然后再从暂存区一并提交到版本库，这样可以减少版本库的版本号；

1. GIT常用操作命令如下：

（1）从本地工作目录添加进暂存区，使用add -- git add . (代表把当前目录的文件添加进暂存区)；

（2）从暂存区添加进版本库，使用commit -- git commit -m "备注说明"；

（3）把本地版本库推送到其他的版本库，使用push；从其他版本库拉取到本地仓库，使用pull；

**注意：本地仓库和远程仓库都是通过分支进行管理的，从暂存区提交到本地仓库时，是添加到本地仓库的某个分支；**

**从本地仓库提交到远程仓库时，是把本地仓库的某个分支推送到远程仓库的某个分支；**

**从远程仓库拉取到本地仓库时，是把远程仓库的某个分支拉取到本地仓库的某个分支；**

**如果没有指定分支，则默认为主分支，即master**

1. GIT安装：

（1）进入msysgit.github.io -> 点击 Download执行下载即可（一般默认下载的是 Git-2.20.1-64-bit.exe）;

（2）安装时，只有“Adjusting your PATH environment”需要修改选择“Use Git from Git Bash only”，其他的都默认下一步即可；

（3）配置PATH：（将“D:\Program Files\Git\bin”添加进环境变量PATH中）；

（4）配置GIT：任意页面右击 -》选择Git Bash Here（此时只要能够识别唯一，因此只需配置用户名和邮箱即可，不用配置密码）

a.配置用户名：git config --global user.name "15338066274" 回车；

b.配置邮箱：git config --global user.email "401934990@qq.com" 回车；

c.配置好后，可以到“C:\Users\Administrator\.gitconfig”文件里面查看刚刚配置的内容，如果存在则说明配置成功；

1. 搭建GIT服务器（远程仓库）：

（1）使用统一的官方托管网站（https://github.com/） -- 注册账户，如果已存在则直接登录即可；

（2）配置免密钥登录（这样可以在本地仓库和远程仓库之间进行免密钥登录）：先在本地配置ssh key，然后发送给远程配置 ，具体如下：

a.打开“Git Bash Here”窗口，输入 ssh-keygen -t rsa -C "401934990@qq.com"回车即可；然后执行三次回车，若看到“+---[RSA 2048]---+”信息则说明生成成功；

b.所生成的rsa文件存放在“c:\Users\Administrator\.ssh\id\_rsa”，Bash窗口有显示路径，可以找到该文件并打开查看；

a)id\_rsa：私钥

b)id\_rsa.pub：公钥

c.进入github.com -》个人中心 -》settings -》SSH and GPG keys -》New SSH key -》

Title:2019studygit -- 任意命名

Key: -- key中输入刚才在本地生成的ssh公钥（即“c:\Users\Administrator\.ssh\id\_rsa.pub”里面的内容），注意去掉最后一行的回车符，然后执行“Add SSH key”即可；

（3）测试连通性：在“Git Bash Here”窗口输入“ssh -T git@github.com”回车，然后选择yes回车，如果看到“successfully”则说明测试成功；此时在“c:\Users\Administrator\.ssh\”里面可以看到一个文件“known\_hosts”，则说明本地仓库和远程仓库已成功通信；如果失败，多尝试几次，以及检查回车符或换行符等；

1. 本地仓库与远程仓库建立项目关联需要执行以下三步：

（1）在本地新建git项目（如D:\software\GIT\mygit项目\hello.txt），并在当前页打开“bash”窗口，执行“git init”即可；

（2）在远程仓库建立git项目：进入github.com -》个人中心 -》Your profile -》Repositories -》New ,具体如下：

A. Repository name：mygitremote

B. Public：公开项目（此时选择Public）

C. Private：私有项目

然后点击“Create repository”即可；

（3）建立本地仓库项目和远程仓库项目之间的关联：

a.拷贝远程仓库项目地址“https://github.com/15338066274/mygitremote.git”

进入github.com -》个人中心 -》Your profile -》Overview，点击项目名称进去拷贝HTTPS地址即可（SSH地址也行）；

b.在“bash”窗口，输入git remote add origin https://github.com/15338066274/mygitremote.git -- 执行后将本地仓库的master分支与运程仓库的master分支建立关联（默认都是master分支）；

1. 使用GIT对项目的操作包括以下四种：

（1）第一次发布项目（本地-》远程）

a.git add . //将当前目录下的所有文件传到暂存区（注意add与.之间有一个空格）；

b.git commit -m "注释内容" //将暂存区的文件传到本地仓库分支（默认为master）；

c.git push -u origin master //将本地master分支的文件上传到远程仓库的master分支；

d.到远程仓库查看并确认：进入github.com -》个人中心 -》Your profile -》Overview -》mygitremote(项目名称) -- 此时可以看到刚刚上传的文件；

（2）第一次下载项目（远程-》本地）

a.git clone https://github.com/15338066274/mygitremote.git -- 项目地址HTTS或SSH

（3）提交

a.git add .//将当前目录下的所有修改过的文件添加到暂存区（注意add与.之间有一个空格）；

b.git commit -m "注释内容" //将暂存区的文件提交到本地仓库分支（默认为master）；

c.git push origin master //将本地master分支的文件提交到远程仓库的master分支； -- 与第一次发布的区别就是少了“-u”

（4）更新

a.git pull

1. 在Eclipse中操作GIT前的准备工作，具体如下：

（1）Help -> About Spring Tool Suite，如果能找到EGit图标，说明已经支持EGit，无需安装，若没找到，则利用第二点安装EGit插件；

（2）在Eclipse中安装EGit插件：Help -> Eclipse Marketplace... -> Find(输入git) -> Go -> 找到EGit，点击Installed安装即可；

（3）配置GIT：

a. Windows -> Preferences -> Team -> Git -> Default repository folder：用来指定从GIT远程仓库clone文件到本地仓库的存储位置；

b. Windows -> Preferences -> Team -> Git -> Configuration：可以看到所配置的用户名和邮箱，也可以进行修改；

c. Windows -> Preferences -> General -> Network Connections -> SSH2 -> SSH2 home ：指定所生成的私钥路径如“c:\Users\Administrator\.ssh”，此时Private keys 显示内容为“id\_dsa,id\_rsa”，若没有显示，则手动输入即可；

1. 在Eclipse中使用GIT包括以下四点，具体如下：

（1）第一次发布项目（本地-》远程）

A. 创建或指定本地仓库：右击项目 -> Team -> Share Project... -> Repository(点击Create...)创建本地仓库 --> Finish；

B. 添加进暂存区：右击项目 -> Team -> Add to Index；

C. 提交到本地分支(本地仓库)：右击项目 -> Team -> Commit... 打开提交对话框：

A) Commit按钮：从暂存区提交到本地分支；Commit后执行Push时，不能单独Push某一个文件，只能Push整个项目；

B) Commit and Push...按钮：提交到本地分支同时推送到远程分支；这种情况下可以单独Push某一个文件；

D. 将项目推送到远程分支(远程仓库)：右击项目 -> Team -> Remote -> Push...(打开Push对话框) ,操作如下：

URI：远程仓库项目HTTPS地址

Host：域名或地址（添加URI后Host自动填入并显示）

Repository path：远程仓库路径（添加URI后自动填入并显示）

Protocol：https（选择时应与URI相对应）

User：远程仓库用户名

Password：远程仓库密码

Store in Secure Store：勾选该复选框后下次提交时不用提供用户名及密码

-> Next -> 选分支，具体如下：

Source ref：本地分支

Destination ref：远程分支

选好分支后点击“Add Spec”-> Next -> Finish -> 推送完成后点击OK即可。

（2）第一次下载项目（远程-》本地）

A. 右键菜单 -> Import... -> Git -> Projects from Git -> Clone URI ,具体如下：

URI：远程仓库项目HTTPS地址

Host：域名或地址（添加URI后Host自动填入并显示）

Repository path：远程仓库路径（添加URI后自动填入并显示）

Protocol：https（选择时应与URI相对应）

User：远程仓库用户名

Password：远程仓库密码

Store in Secure Store：勾选该复选框后下次提交时不用提供用户名及密码

-> Next -> Next -> 在“Directory”里面指定本地仓库的路径（如“L:\git\_repository”） -> Next -> Back -> Next -> 选择“Import as general project”,以及选中“Working Tree”里面的项目 -> Next -> Finish . --> 然后将已导入的项目转成Maven项目即可，具体如下：

右击项目 -> Configure -> Convert to Maven project 即可。

B. 右键菜单 -> Import... -> Maven -> Check out Maven Projects from SCM -> Next -> 提供远程仓库项目地址 -> Next -> Finish .

（3）提交

A. 添加进暂存区：右击项目 -> Team -> Add to Index；

B. 提交到本地分支(本地仓库)：右击项目 -> Team -> Commit... 打开提交对话框执行commit操作；

C. 提交项目到远程仓库：

a)推送单个文件：右击需提交的文件 -> Team -> Repository -> Push Branch ‘master’...(打开Push对话框)， 具体如下：

Remote name：远程仓库分支名称，默认为origin;其余选项与推送项目一致

-> Preview -> Preview -> Push -> 推送完成后点击Close即可

b)推送整个项目：右击项目 -> Team -> Remote -> Push...(打开Push对话框)，操作同第一次发布项目一致；

（4）更新

更新项目：右击项目 -> Team -> Pull

更新文件：右击文件 -> Team -> Repository -> Pull

1. 如何解决本地仓库与远程仓库之间的冲突：

（1）把修改过的代码提交到暂存区，再提交到本地仓库；

（2）把远程仓库的代码pull到本地仓库；

（3）把存在的冲突在本地解决掉；

（4）再重新把调整过的代码提交到暂存区-》本地仓库-》远程仓库即可；

**注意：a. 如果没有解决冲突前直接提交到远程仓库，则提交失败并报错。**

**b. 解决冲突前可以进入同步视图：右击项目 -> Team -> Synchronize Workspace -> 如果有红色标记则说明有冲突产生**

1. 如何使用github进行团队协作：

（1）进入需要协作的项目，如“study”-> Settings -> Collaborators -> 输入用户名(github里面的用户,全名或邮箱),点击“Add collaborator” -> 复制邀请链接,点击“Copy invite link”，发给对方；

（2）被添加的协作者在自己的github账户已登录的情况下，在地址栏输入邀请链接并回车 -> 点击“Accept invitation”接受邀请；

（3）协作者可以登录进github后，点击左上角的cat图标进入“https://github.com/”，可看到自己有权操作的所有远程仓库；

（4）协作者不用配置SSH key，只需拿到远程仓库的项目链接就可以访问；

（5）剩下的操作跟上面讲到的内容一致；

1. 问题处理：（error: RPC failed; result=55, HTTP code = 200）

（1）克隆远程仓库时，用env命令设置GIT\_SSL\_NO\_VERIFY环境变量为”ture”，并同时调用正常的git clone命令。完整的命令如下：

**env GIT\_SSL\_NO\_VERIFY=true git clone https://<host\_name/git/project.git**

（2）在克隆完毕的仓库中将http.sslVerify设置为”false”。完整的命令如下：

**git config http.sslVerify "false"**

（3）参考资料：https://blog.csdn.net/m0\_37052320/article/details/77799413