# 使用eclipse与git 进行工程项目管理

### 01:项目托管选址及建立

常用的开源项目托管网站：

国外：GitHub

国内：oschina 开源中国 的码云 https://git.oschina.net/

由于GitHub 建立私有仓库是要收费的，同时界面都是纯英文的，从经济成本及操作的便利性采用 开源中国的码云

1: 注册oschina

用户名：A\_one2010 密码xxx

2: 建立项目



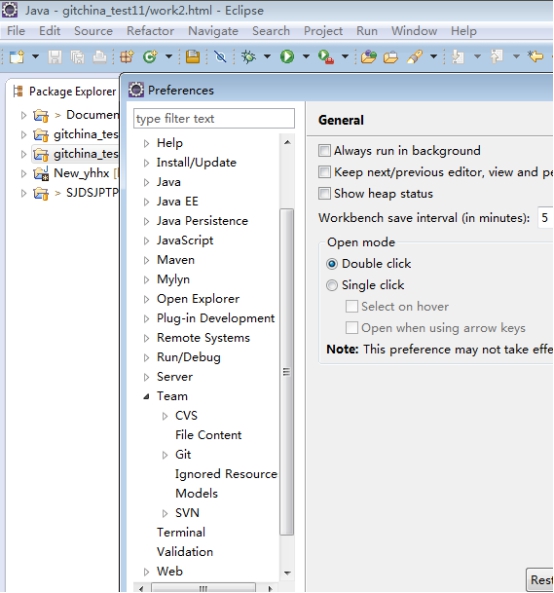
项目地址为：<https://git.oschina.net/A_one2010/gitchina_test.git>

项目名称： gitchina\_test

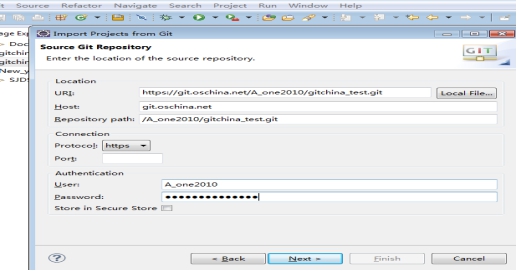
### 02：eslipse+git将项目导入本地

Eclipse已经集成了git,没有集成的见eclipse插件安装之git,此处略过。

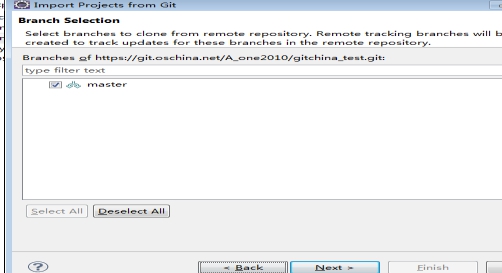
查看window-preference-team



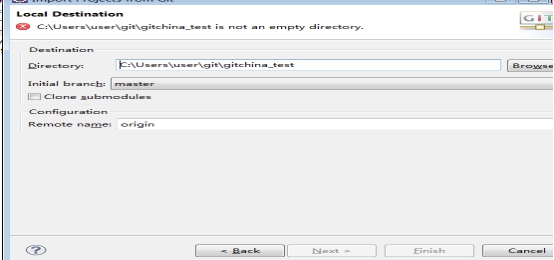
Import——git——projects from git——clone url——next——【输入工程地址、用户名及密码】——next——选择工程地址及名称



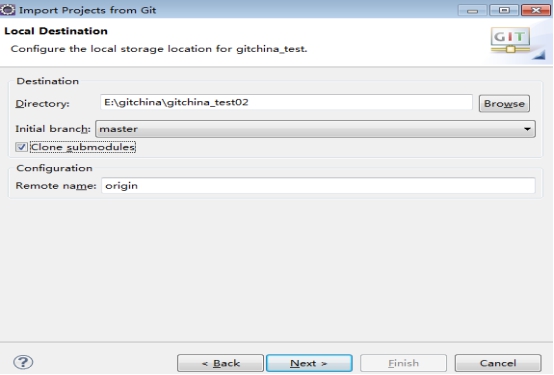
选择master——next



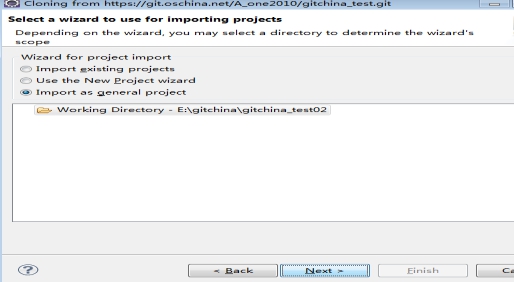
选择要存放的位置---及名称



选择clone submodules克隆子模块



选择 导入普通工程 import as general project



填写工程名称

Finished

### 03：cmd--git的相关命令

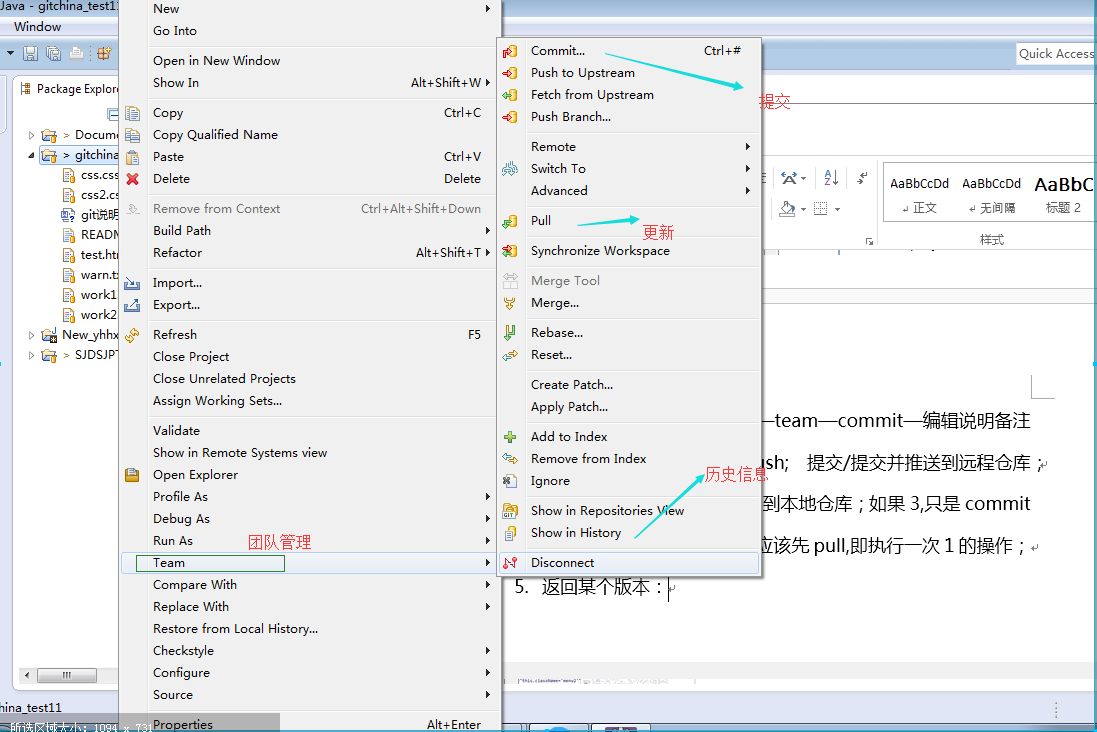
常用命令

* git clone url xxx 下载工程，并命名为xxx（如果想要重名了否则就是原名称）
* git status 查看有哪些文件改变（红色表示有改变且没添加到暂存区/绿色表示添加到了暂存区但未commit）
* git add xxx 将xxx添加到本地仓库【git add . 所有变化提交到暂存区(修改，新增)/git add -u （update）仅监控已经被add的文件，新文件不提交/git add -A】
* git commit -m xxxx 提交暂存区的文件及编辑的信息
* git branch 查看本地分支，git branch -a 查看远程分支
* git pull 更新为最新的工（推送前先拉远程代码防止冲突）
* git push origin master 提交的远程仓库 第一次提交要输入用户账号密码
* git pull 更新为最新的工程

### 04：Eclipse--git相关操作

工作中如下循环操作：

1. 更新工程：工程文件夹上--右击——team——pull——ok。
2. 编辑文件：修改、编辑工作的文件。
3. 本地提交：在文件上右击——team—commit—编辑说明备注—commit/commit and push; 提交/提交并推送到远程仓库；
4. 通常3只选择commit提交到本地仓库；如果3,只是commit并没有push，那么push前应该先pull,即执行一次1的操作；
5. 返回某个版本：



### 04：Git命令大全

[Git](http://gitref.org/index.html" \t "http://blog.csdn.net/a_one2010/article/details/_blank) 是一个很强大的分布式**[版本控制](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/a_one2010/article/details/_blank)**系统。它不但适用于管理大型开源软件的源代码，管理私人的文档和源代码也有很多优势。

**[Git](http://lib.csdn.net/base/git" \o "Git知识库" \t "http://blog.csdn.net/a_one2010/article/details/_blank)**常用操作命令：

1 ) 远程仓库相关命令

检出仓库：$ git clone git://github.com/**[jQuery](http://lib.csdn.net/base/jquery" \o "jQuery知识库" \t "http://blog.csdn.net/a_one2010/article/details/_blank)**/jquery.git

查看远程仓库：$ git remote -v

添加远程仓库：$ git remote add [name] [url]

删除远程仓库：$ git remote rm [name]

修改远程仓库：$ git remote set-url --push [name] [newUrl]

拉取远程仓库：$ git pull [remoteName] [localBranchName]

推送远程仓库：$ git push [remoteName] [localBranchName]

\*如果想把本地的某个分支test提交到远程仓库，并作为远程仓库的master分支，或者作为另外一个名叫test的分支，如下：

$git push origin test:master         // 提交本地test分支作为远程的master分支

$git push origin test:test              // 提交本地test分支作为远程的test分支

2）分支(branch)操作相关命令

查看本地分支：$ git branch

查看远程分支：$ git branch -r

创建本地分支：$ git branch [name] ----注意新分支创建后不会自动切换为当前分支

切换分支：$ git checkout [name]

创建新分支并立即切换到新分支：$ git checkout -b [name]

删除分支：$ git branch -d [name] ---- -d选项只能删除已经参与了合并的分支，对于未有合并的分支是无法删除的。如果想强制删除一个分支，可以使用-D选项

合并分支：$ git merge [name] ----将名称为[name]的分支与当前分支合并

创建远程分支(本地分支push到远程)：$ git push origin [name]

删除远程分支：$ git push origin :heads/[name] 或 $ gitpush origin :[name]

\*创建空的分支：(执行命令之前记得先提交你当前分支的修改，否则会被强制删干净没得后悔)

$git symbolic-ref HEAD refs/heads/[name]

$rm .git/index

$git clean -fdx

3）版本(tag)操作相关命令

查看版本：$ git tag

创建版本：$ git tag [name]

删除版本：$ git tag -d [name]

查看远程版本：$ git tag -r

创建远程版本(本地版本push到远程)：$ git push origin [name]

删除远程版本：$ git push origin :refs/tags/[name]

合并远程仓库的tag到本地：$ git pull origin --tags

上传本地tag到远程仓库：$ git push origin --tags

创建带注释的tag：$ git tag -a [name] -m 'yourMessage'

4 ) 子模块(submodule)相关操作命令

添加子模块：$ git submodule add [url] [path]

   如：$git submodule add git://github.com/soberh/ui-libs.git src/main/webapp/ui-libs

初始化子模块：$ git submodule init  ----只在首次检出仓库时运行一次就行

更新子模块：$ git submodule update ----每次更新或切换分支后都需要运行一下

删除子模块：（分4步走哦）

 1) $ git rm --cached [path]

 2) 编辑“.gitmodules”文件，将子模块的相关配置节点删除掉

 3) 编辑“ .git/config”文件，将子模块的相关配置节点删除掉

 4) 手动删除子模块残留的目录

5）忽略一些文件、文件夹不提交

在仓库根目录下创建名称为“.gitignore”的文件，写入不需要的文件夹名或文件，每个元素占一行即可，如

target

bin

\*.db

=====================

Git 常用命令

git branch 查看本地所有分支  
git status 查看当前状态   
git commit 提交   
git branch -a 查看所有的分支  
git branch -r 查看本地所有分支  
git commit -am "init" 提交并且加注释   
git remote add origin git@192.168.1.119:ndshow  
git push origin master 将文件给推到服务器上   
git remote show origin 显示远程库origin里的资源   
git push origin master:develop  
git push origin master:hb-dev 将本地库与服务器上的库进行关联   
git checkout --track origin/dev 切换到远程dev分支  
git branch -D master develop 删除本地库develop  
git checkout -b dev 建立一个新的本地分支dev  
git merge origin/dev 将分支dev与当前分支进行合并  
git checkout dev 切换到本地dev分支  
git remote show 查看远程库  
git add .  
git rm 文件名(包括路径) 从git中删除指定文件  
git clone git://github.com/schacon/grit.git 从服务器上将代码给拉下来  
git config --list 看所有用户  
git ls-files 看已经被提交的  
git rm [file name] 删除一个文件  
git commit -a 提交当前repos的所有的改变  
git add [file name] 添加一个文件到git index  
git commit -v 当你用－v参数的时候可以看commit的差异  
git commit -m "This is the message describing the commit" 添加commit信息  
git commit -a -a是代表add，把所有的change加到git index里然后再commit  
git commit -a -v 一般提交命令  
git log 看你commit的日志  
git diff 查看尚未暂存的更新  
git rm a.a 移除文件(从暂存区和工作区中删除)  
git rm --cached a.a 移除文件(只从暂存区中删除)  
git commit -m "remove" 移除文件(从Git中删除)  
git rm -f a.a 强行移除修改后文件(从暂存区和工作区中删除)  
git diff --cached 或 $ git diff --staged 查看尚未提交的更新  
git stash push 将文件给push到一个临时空间中  
git stash pop 将文件从临时空间pop下来  
---------------------------------------------------------  
git remote add origin git@github.com:username/Hello-World.git  
git push origin master 将本地项目给提交到服务器中  
-----------------------------------------------------------  
git pull 本地与服务器端同步  
-----------------------------------------------------------------  
git push (远程仓库名) (分支名) 将本地分支推送到服务器上去。  
git push origin serverfix:awesomebranch  
------------------------------------------------------------------  
git fetch 相当于是从远程获取最新版本到本地，不会自动merge  
git commit -a -m "log\_message" (-a是提交所有改动，-m是加入log信息) 本地修改同步至服务器端 ：  
git branch branch\_0.1 master 从主分支master创建branch\_0.1分支  
git branch -m branch\_0.1 branch\_1.0 将branch\_0.1重命名为branch\_1.0  
git checkout branch\_1.0/master 切换到branch\_1.0/master分支  
du -hs  
  
-----------------------------------------------------------  
mkdir WebApp  
cd WebApp  
git init  
touch README  
git add README  
git commit -m 'first commit'  
git remote add origin git@github.com:daixu/WebApp.git  
git push -u origin master