EGit（Git Eclipse Plugin）使用

EGit（Git Eclipse Plugin）使用

博客分类：
- Develop

GitEGit版本控制Git Eclipse plugin

本文章已收录于： Git

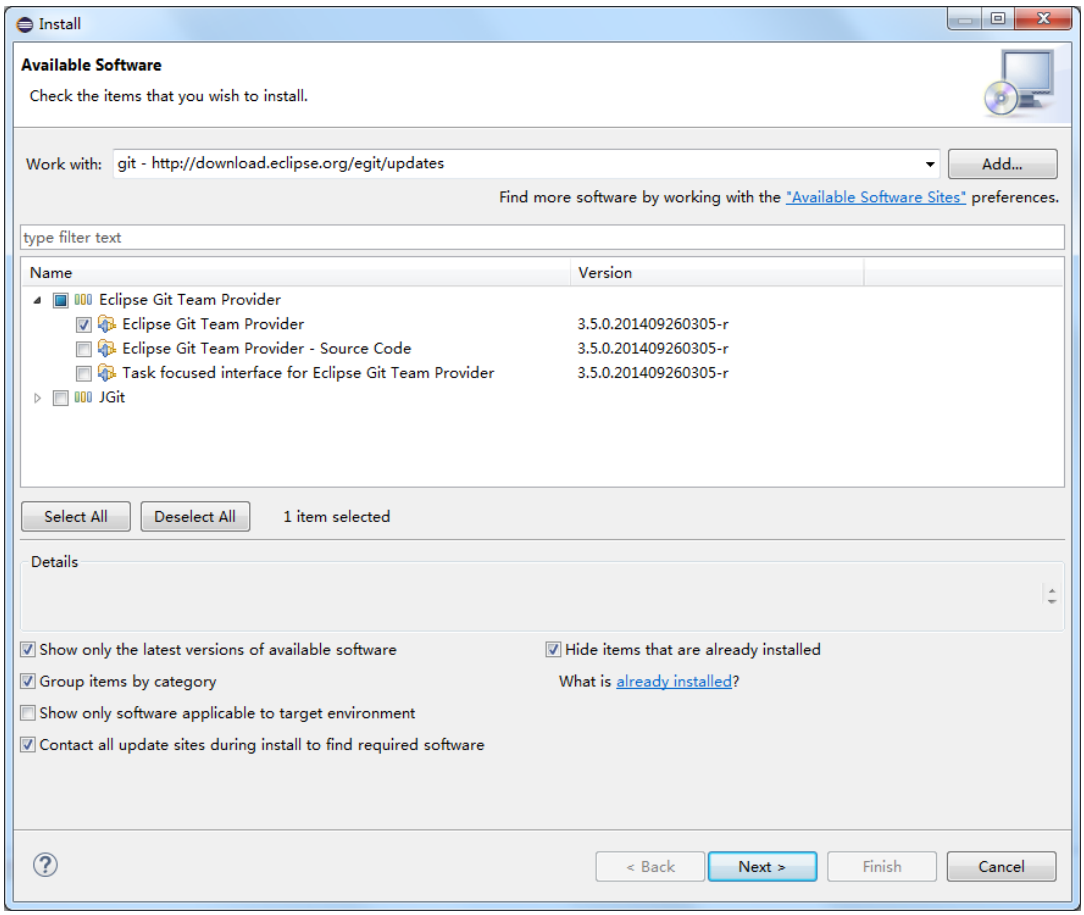
编写不易，转载请注明（http://shihlei.iteye.com/blog/2124411）
最近跳槽，新公司使用Git，快速学习了下，开发主要用EGit插件，总结下。
前言：
1）Git于SVN的不同
Git是分布式数据库，本地创建仓库，即可在本地完成版本控制（等价于SVN在本地安装服务器和客户端，SVN服务器如果在远程，断网情况将无法完成提交及版本维护）。
Git协作开发，大家可以互相克隆版本库（相当于SVN下载项目），进行开发，每人都有完整的库（分布式）。通常为了方便，远程还是会建立一个共享库，如GitHub，方便大家同步和共享，不用互相在线，点对点同步修改。

2）Git元素概念
工作区（Working Directory）：代码开发和修改的区域，Eclipse将Workspace区域的文件显示给用户，用于操作。
暂存区（Index）：修改不同文件，通过Add to Index，添加到暂存区，暂存该批次的多个修改。
注：在最初的Git,文件提交前必须提交到暂存区。EGit这不是必要的,Team => Commit可以提交unstaged变化。可以和暂存区的状态比较和回退暂存区修改。
（状态参见二）
版本库（Repository）：该到一定程度时，可以提交一批次暂存区的修改，操作后修改提交版本库，并标记版本，是后续分享和回退的批次。

3）Git教程：
http://www.liaoxuefeng.com/wiki/001373951630592960 6dd18361248578c67b8067c8c017b000

4）文章概述：
1> EGit 安装配置
2> EGit 文件状态及图标标志
2> EGit 本地开发
创建本地库；提交；回退；创建分支；合并；
3> EGit 协作开发
项目推送远程仓库；克隆远程仓库导入项目；提交修改到远程仓库程；更新远程仓库修改到本地；解决冲突。
4> EGit 视图
仓库视图；同步视图；历史视图；
5）参考
http://eclipsesource.com/blogs/tutorials/egit-tutorial/

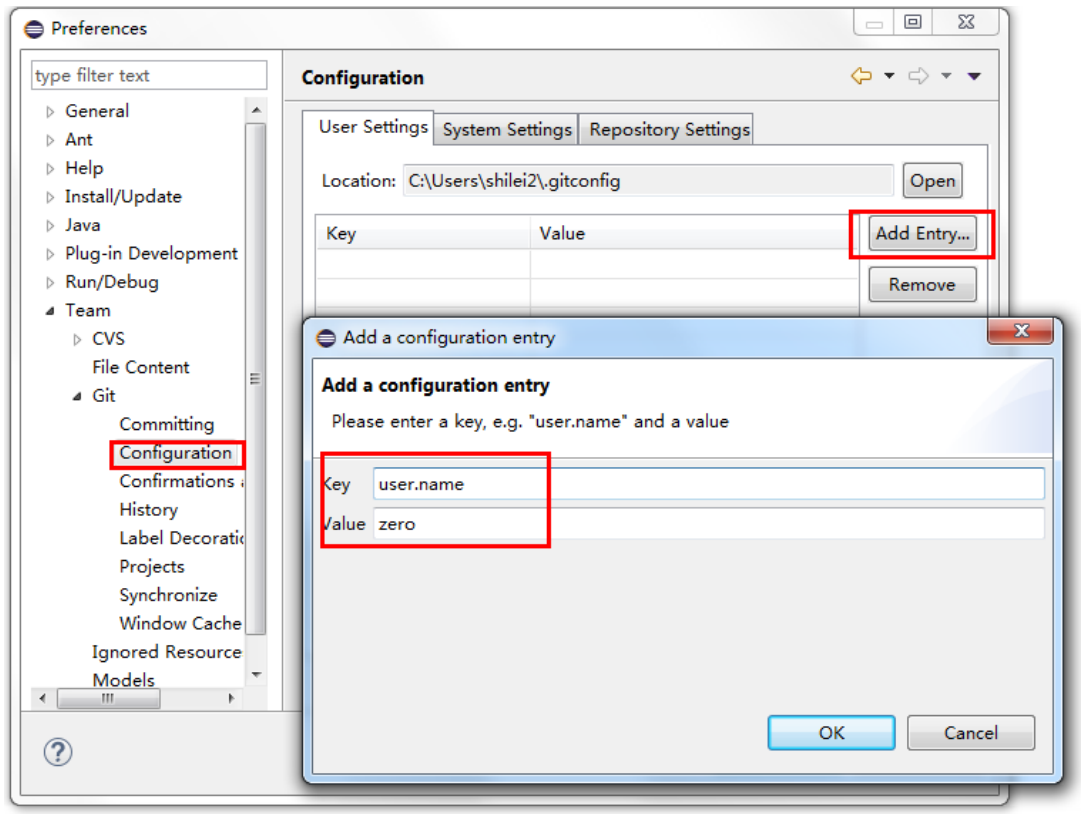
一 Eclipse 安装EGit
1）安装
Eclipse Juno 版后已经集成了EGit，如果使用早些版本，可以通过如下地址安装：http://download.eclipse.org/egit/updates ，选择“Eclipse Git Team Provider”，不需要安装其他插件，直到结束。



2）配置

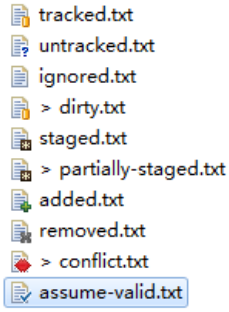
EGit每次提交都将包含用户名和邮件，可以通过“Window => Preferences => Team => Git => Configuration”配置，通过Add Entry 按钮添加信息，

key： user.name，value：用户名；key：user.emai， value： 邮箱，账户信息通常GitHub相同。



二 EGit文件状态及图标展示

EGit会出现如下图标，其对应状态及意义如下：



1）忽略[ignored]：仓库认为该文件不存在（如bin目录，不需要关注）。通过右键Team => Ignore 添加忽略文件

2）未跟踪[untracked]：仓库未跟踪，通常是新建的文件，要接入版本管理可以通过“Add to Index”或直接“Commit”操作。

3）已跟踪[tracked]：文件已被仓库记录。

4）已添加[added]：untracked 状态的文件，通过“Add to Index”被仓库已知，但是没有“Commit”，“Commit”后可变为“已跟踪[tracked]”状态。

5）已删除[removed]：从工作区中删除文件，文件会消失，也就没有图标出现，下一次提交时被删除。Team => Untrack可以触发本图标，在“Commit”对话框中可以看到图标。

6）已修改[dirty]: 修改“已跟踪[tracked]”的文件，未添加到暂存区Index（未“Add to Index”或“Commit”）的文件，标志与本地库不一致。

7）已暂存[staged]：修改“已跟踪[tracked]”的文件，并添加到暂存区Index（即执行“Add to Index”）；

8）冲突[conflict] : 进行Marge合并操作会引起冲突，需要人工解决并添加到索引区修改状态。

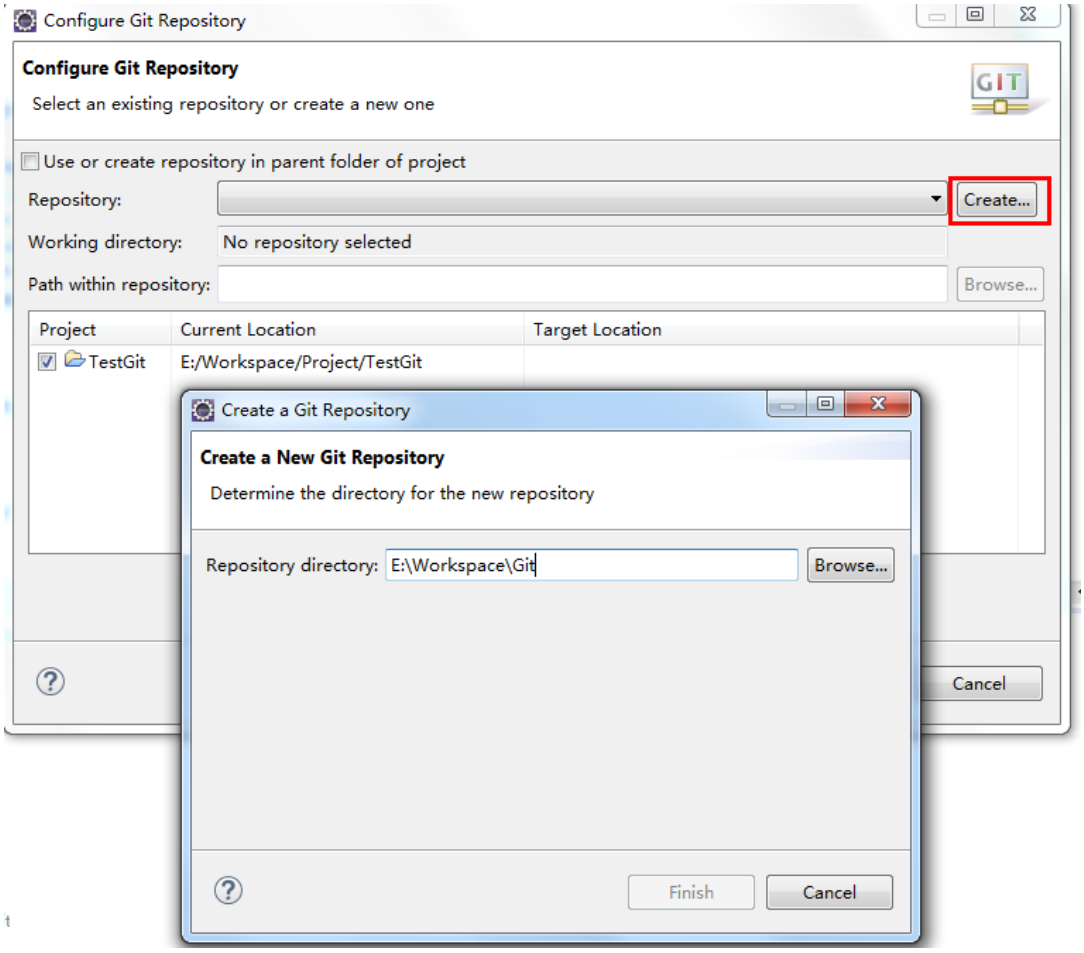
9）已部分暂存[partially-staged]：修改“已跟踪[tracked]”的文件，部分修改已添加暂存区Index，部分未添加。
相当于：已跟踪的文件修改，Add to Index，Commit前又修改了文件。

10）假设有效[assume-valid]：一些修改未被Git检查。右键Team => Assume unchanged可产生该状态。

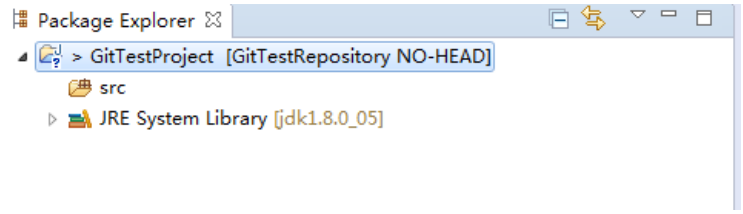
三 EGit使用

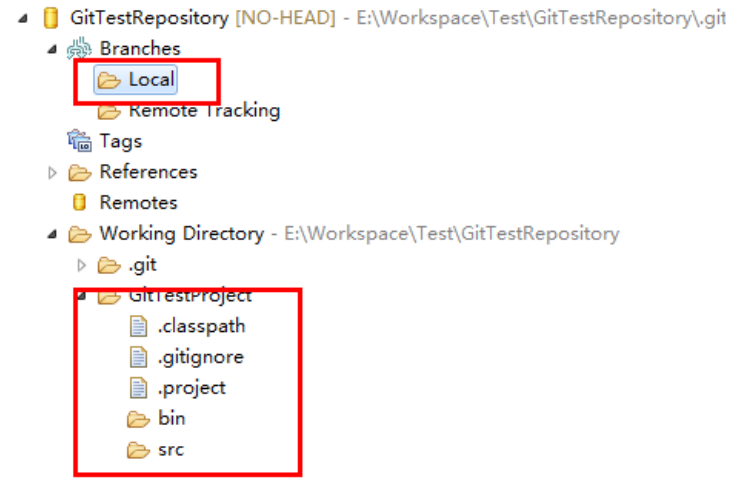
（一）创建本地仓库

Git优势是容易创建本地仓库，将工作本地化，待需要的时候推送到远程仓库，因此所有修改可以在本地版本化。
方法：本地见一个工程，添加一些文件，然后在工程上右键Team => Share Project ，点击 create按钮创建仓库。



尽管我们创建了本地仓库，并share 项目，但库是空的。我们可以向工程添加文件，并提交到本地仓库。

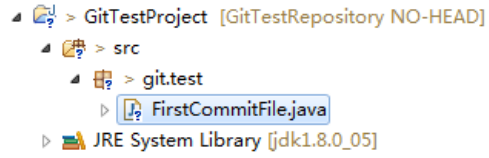




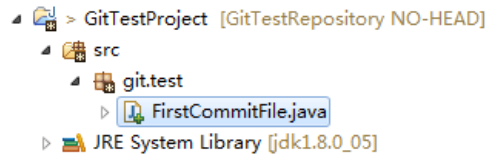
（二） 提交（Commit）

1）创建文件提交

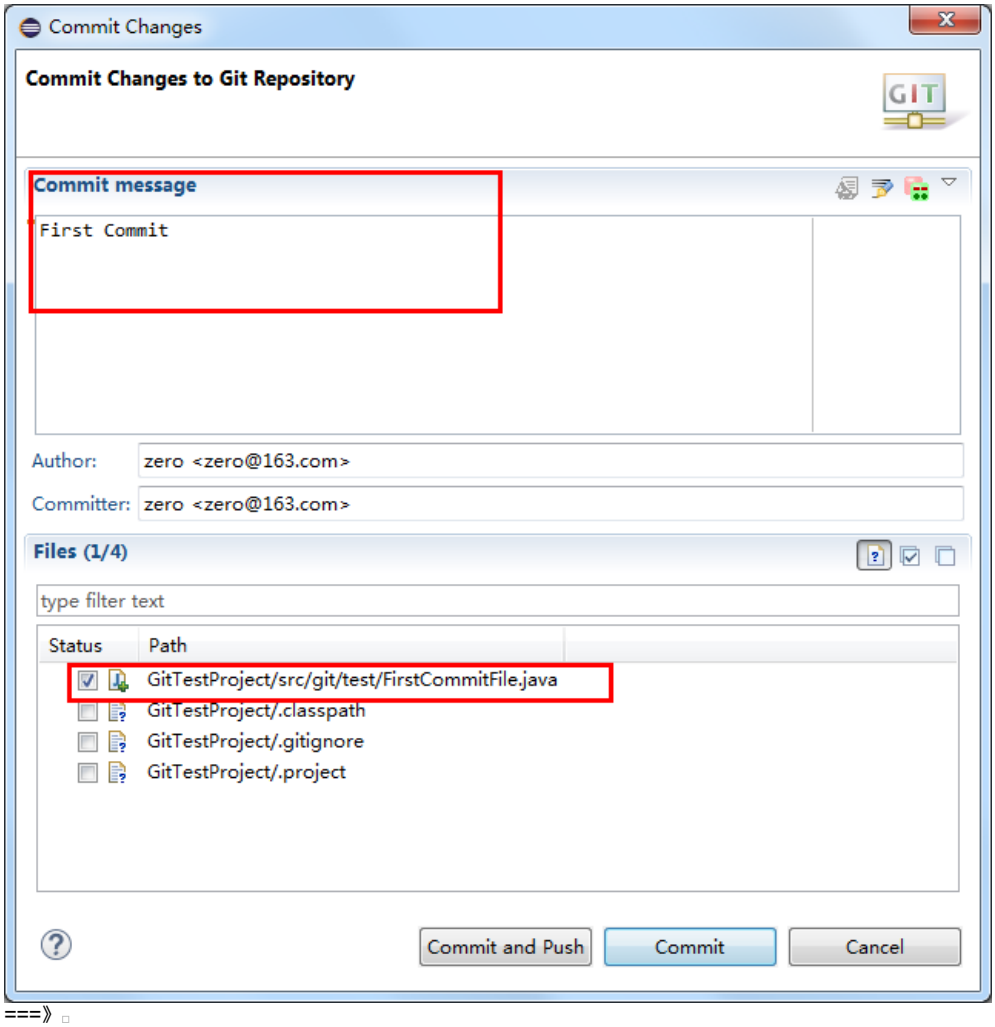
创建新文件，由“？”标记，想提交倒本地仓库进行版本管理，需要右键Team => Add to Index,操作后“？”变成“+”。然后项目右键Team => Commit，输入本次提交信息（注：输入的信息会展示在历史页面），成功后标记由“+”变成“仓库”符号。



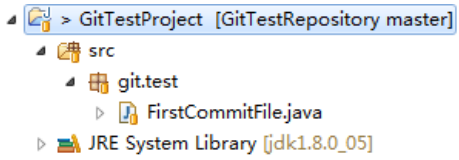
=And to Index=>



=commit=>

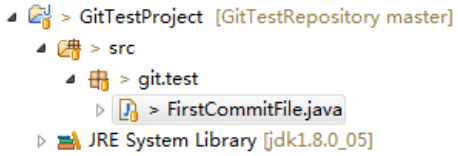


===》



2）产生修改提交

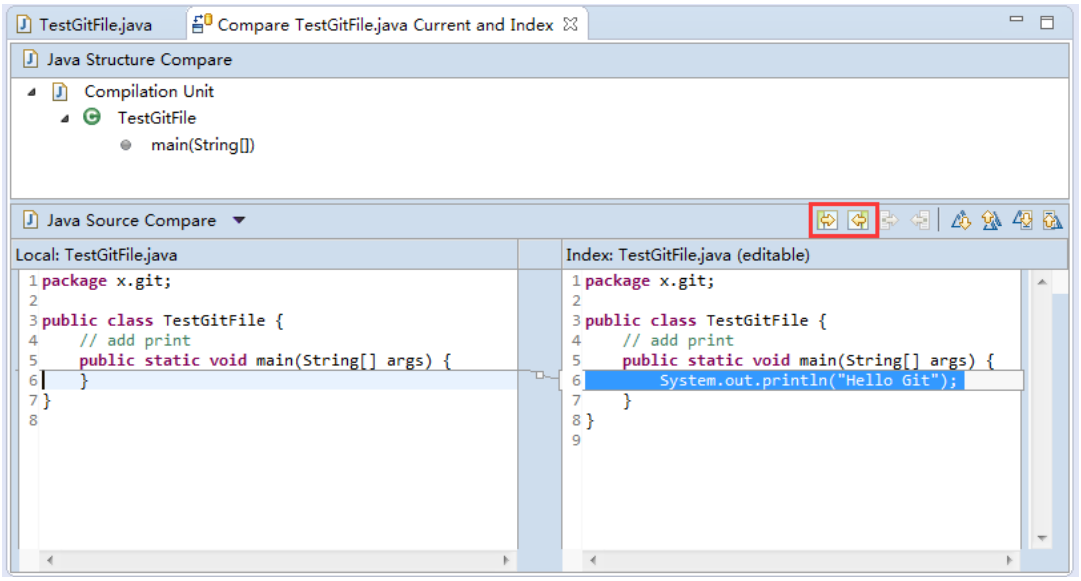
修改项目文件，文件会由“>”标记，如果需要同步到本地库，我们需要提交，过程同上commit



（三） 撤销修改

方法1：通过和上一次暂存区（Index）文件比较——用于单个文件，一般文件状态为“已部分暂存[partially-staged]”
文件上右键Compare With => Git index

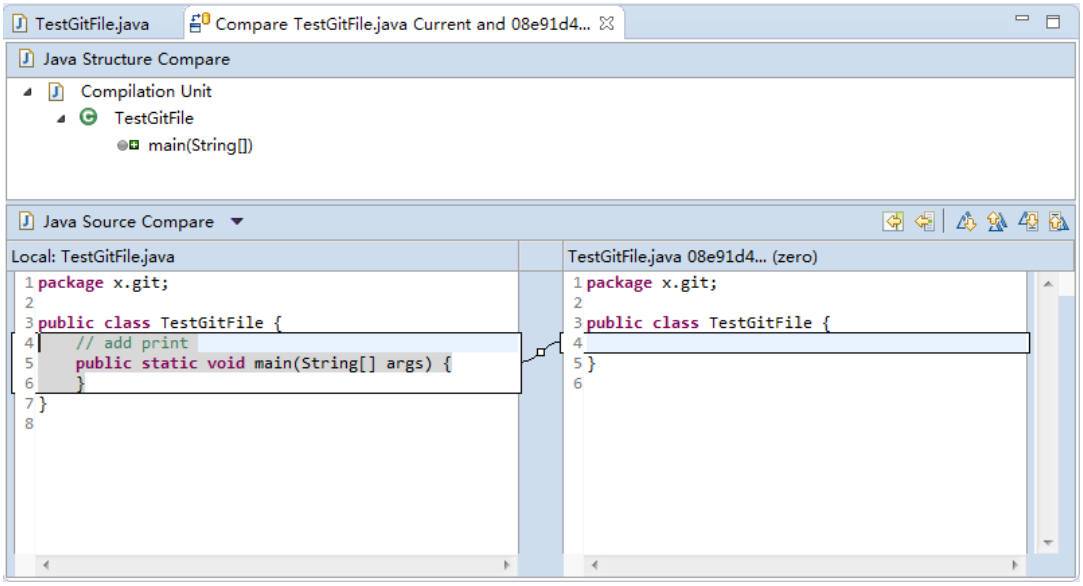
> 如果想修改暂存区Index，点击Copy All Non-Conflicting Changes from Right to Left-button，修改后文件变为“已暂存[staged]”。
> 如果向回退上次Add to Index的情况，点击Copy All from Left to Right，修改后文件变为“已暂存[staged]”。



方法1：通过和上一次Commit版本比较回退——用于单文件

文件上右键Compare With => HEAD Revision

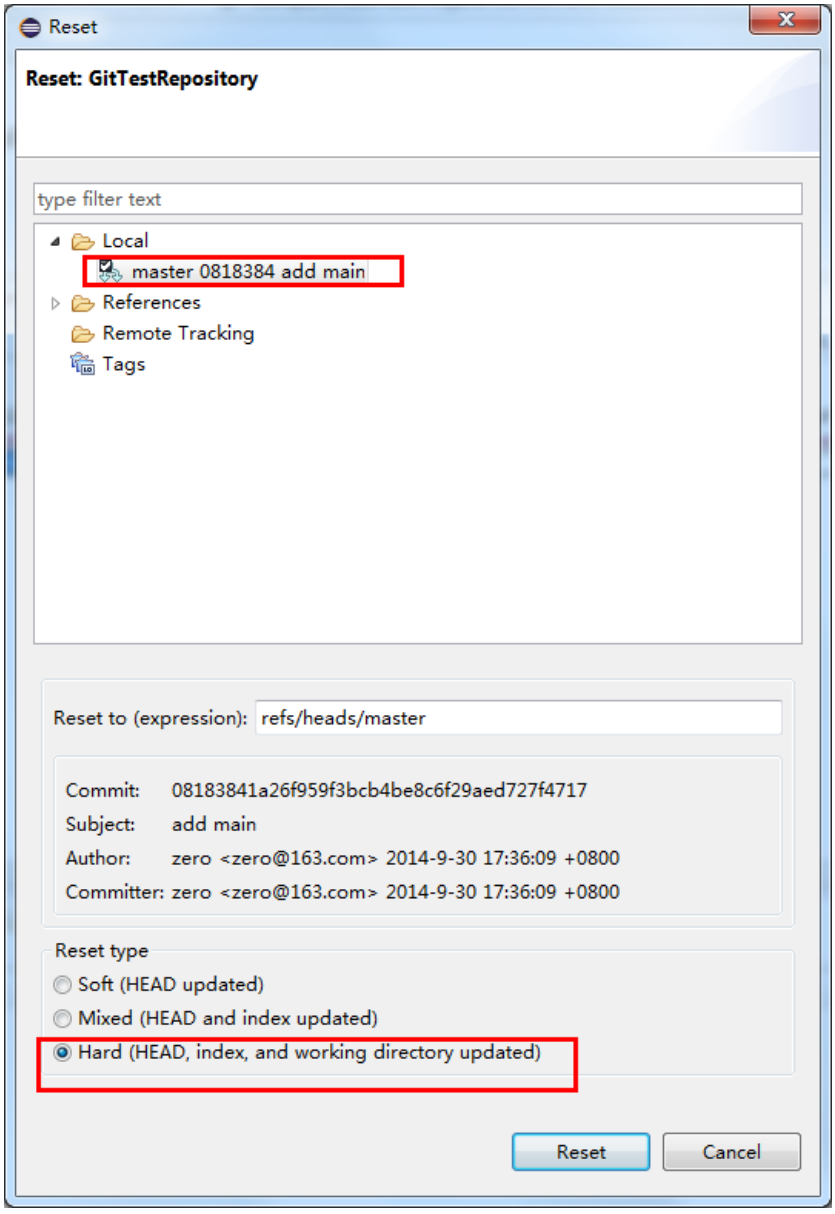
如果想完全恢复文件，点击Copy All Non-Conflicting Changes from Right to Left-button
如果向回退某几行，选择单独每一行，点击the Copy Current Change from Right to Left button。



手工完成修改合并后再提及。

方法3：通过重置回退（Revert via Reset）——用于整个工程

在项目上回退所有修改，项目上右键Team => Reset…；选择要回退到的分支（如果没有创建分支，只有有一个Master），Reset Type选择Hard，点击“Reset”确认。所有修改回退到该分支的最后一次提交，包括工作区所有完成的修改。



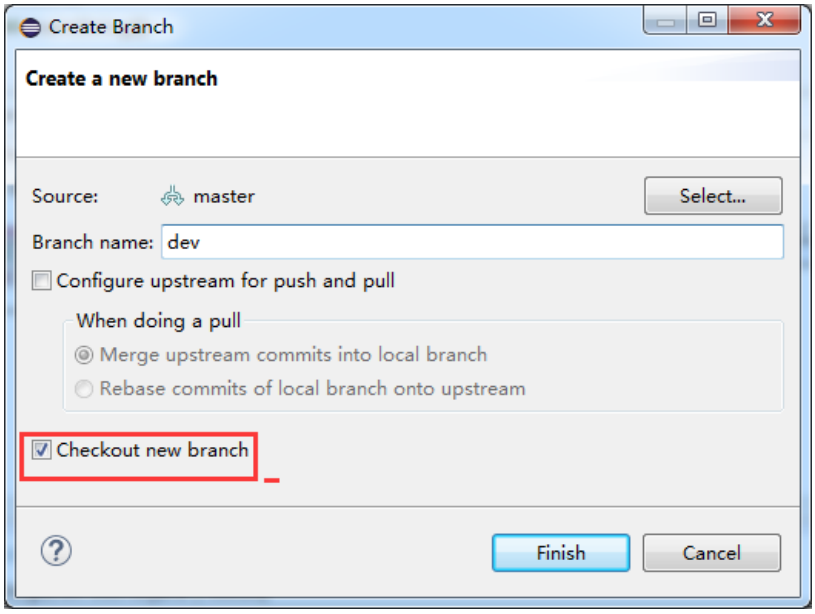
注：重置类型（Reset Type）：
1）Soft：
只回退commit信息（HEAD 指针），不回退暂存区（Index）和工作区（Working Directory 文件）源码，如果需要可再次提交，回到上次commit情况。

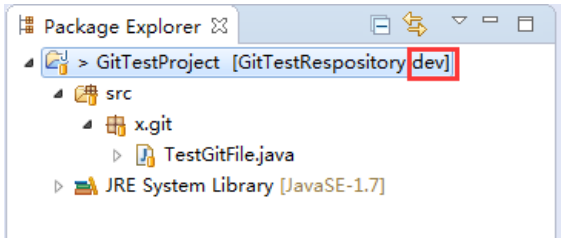
2）Mixed：
回退commit信息（HEAD 指针）和暂存区（Index），只保留工作区（Working Directory 文件）源码。

3）Hard:
所有更改都将恢复到所选分支/标记/Commit版本。工作区（Working Directory 文件）源码会变为上一个Commit版本的内容，未提交的更改都将丢失，因此该操作必须确认。

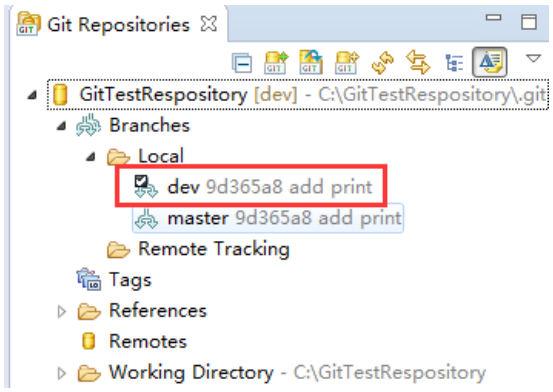
（三） 创建分支

项目上右键Team => Switch to => New Branch…，选择一个新分支的来源，点击创建分支，输入分支名。
新分支会出现在分支选择窗口，如果向检出一个新创建的分支，选择然后点击checkout





》》》》》》》



注：

Git的一大优势是很容易创建分支，根据需要可以快速创建Bug修复分支，新功能分支，几条分支独立开发，最后合并到主分支。

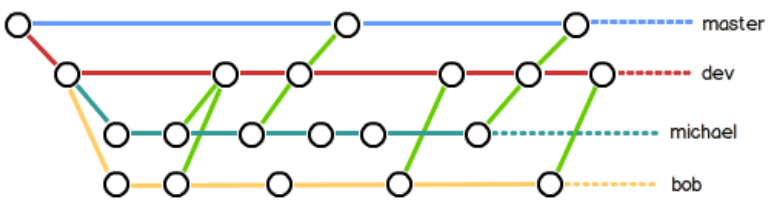
分支管理策略（引用：http://www.liaoxuefeng.com/wiki/001373951630592960 6dd18361248578c67b8067c8c017b000/0013758410364457b9e3d821f4244beb0fd69c61a185ae0000）：

原则：

a）master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面开发；

b）开发都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；

c）每个开发人员都在自己dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

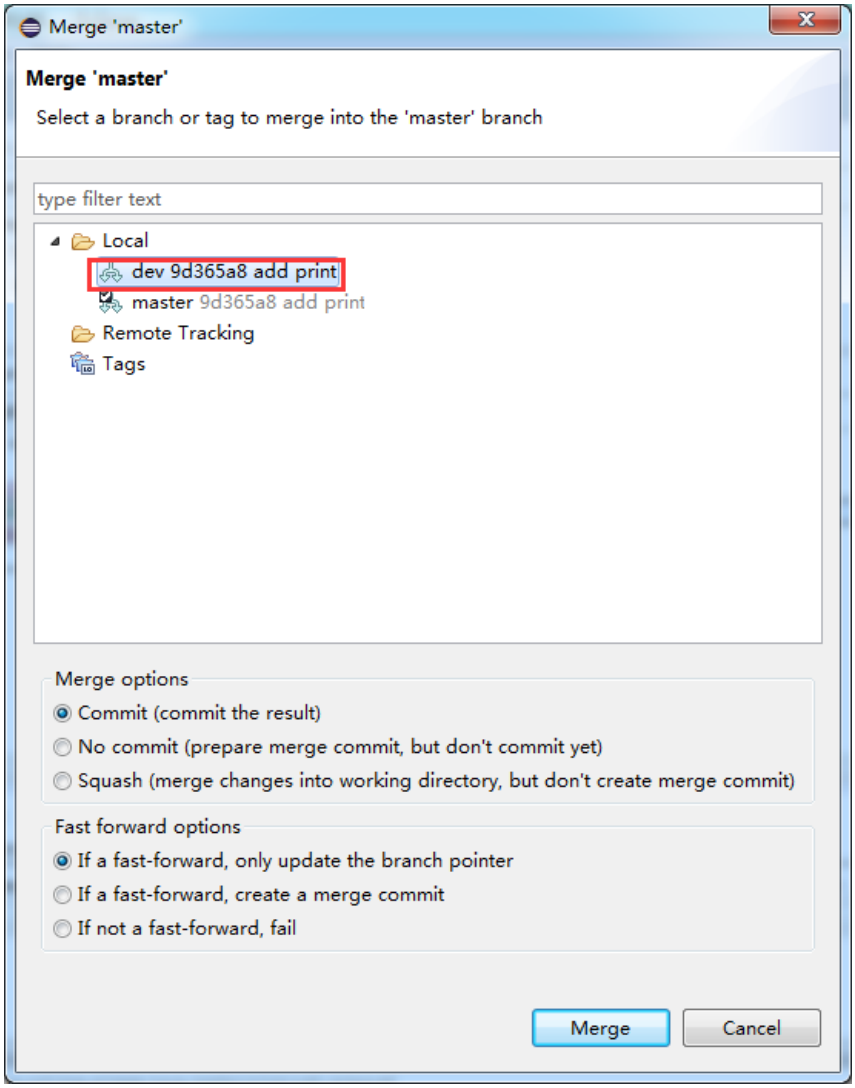


（四）合并和解决冲突

1）合并

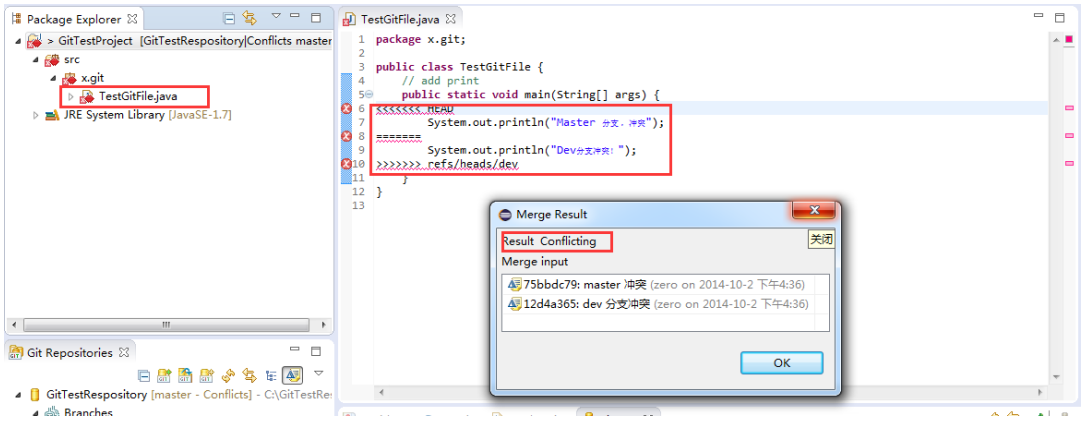
将一个分支合并到另一个分支，首先切换到要合并的分支，项目邮件Team => Switch to => 分支名（或仓库视图，Branches => Local =>指定分支右键Checkout），右键 Team => Merge… **选择要合并到当前分支的分支**，点击Merge。

合并开始并弹出结果，结果有如下几种：Already-up-to-date, Fast-forward, Merged, Conflicting, Failed.



2）解决冲突

冲突需要手动解决。打开冲突文件，找到冲突修改标志“<<<<<<<”，手动合并后后，需要通知Git冲突解决，**Add to index 和commit 完成合并**

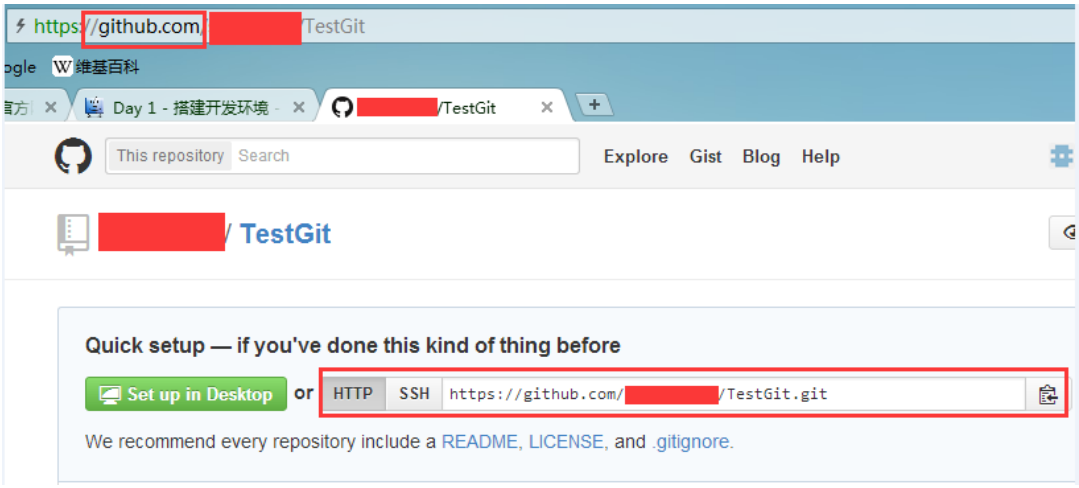


四 远程仓库

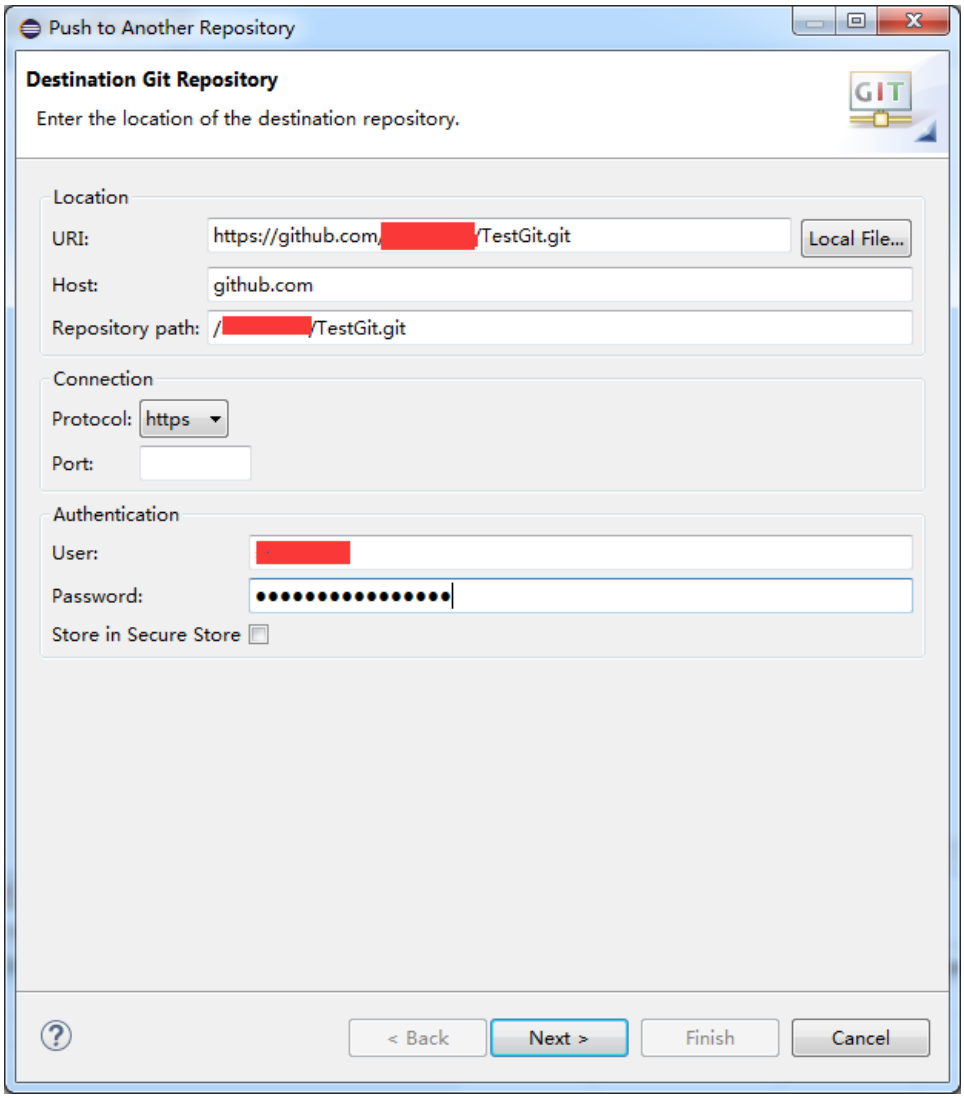
远程仓库一般用于多人共享，GitHub就提供免费的共享空间，注册帐号，在GitHub上创建仓库，就可以推送项目。

（一）项目提交远程仓库

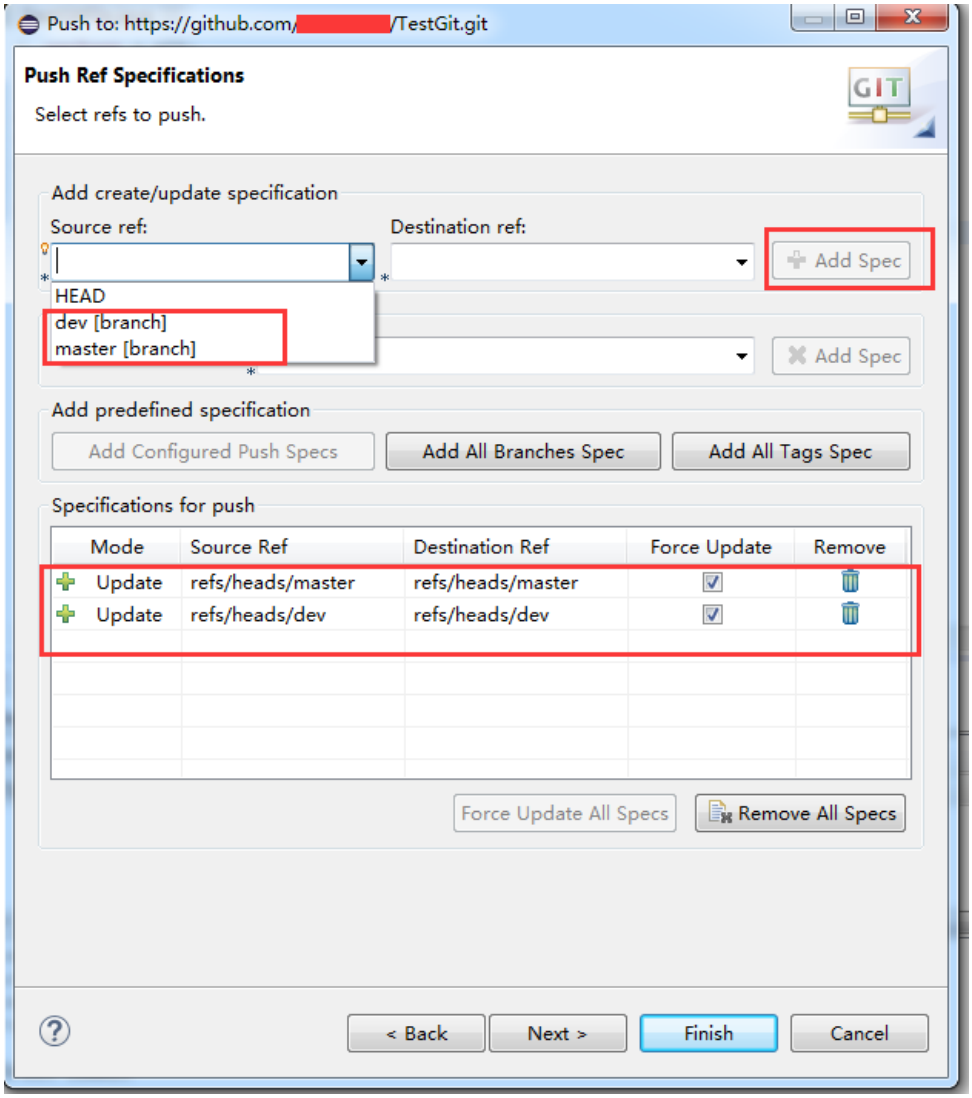
1）GitHub上创建项目



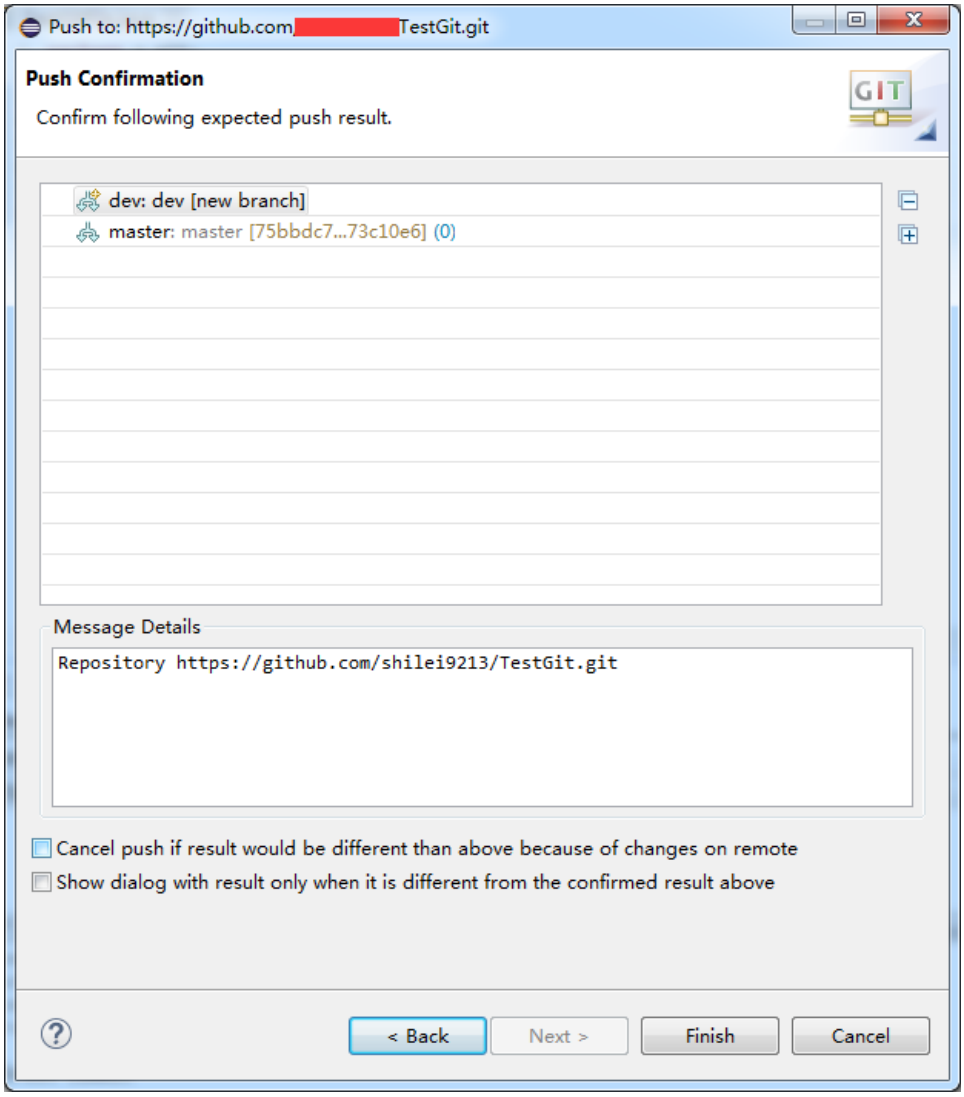
2）项目：右键 Team => Remote => Push，填写远程仓库信息



3）选择推送的分支信息，这里由于建立了master和dev分支，选择并通过"Add Spec"按钮添加，选中复选框“Force Update”，已避免冲突，否则需要现更新再推送。



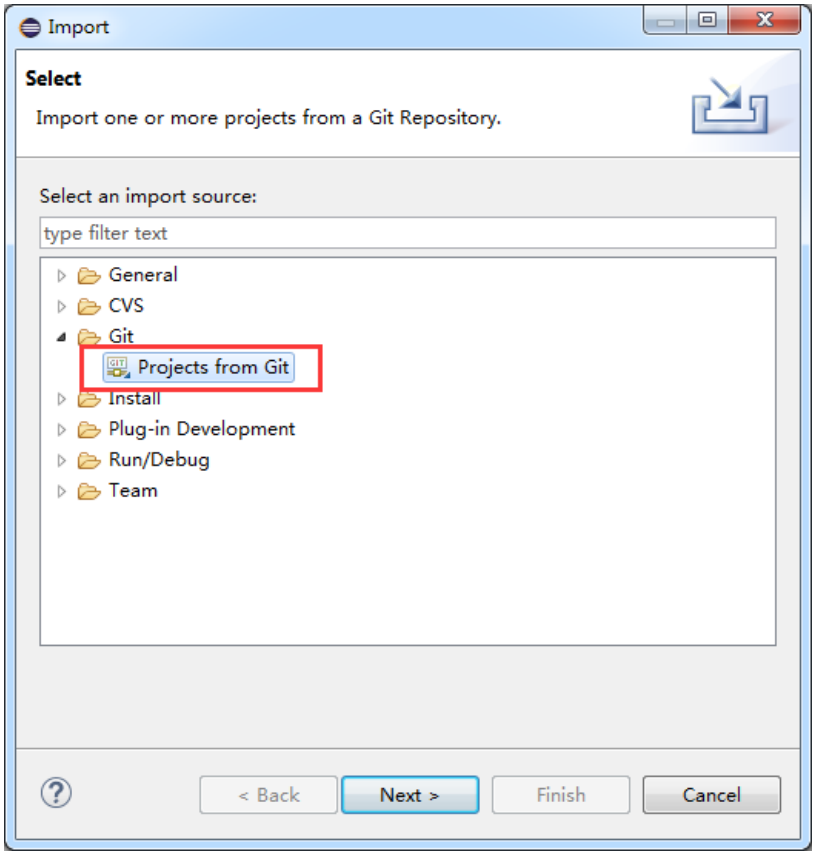
4）由于Force Update所以需要确认。



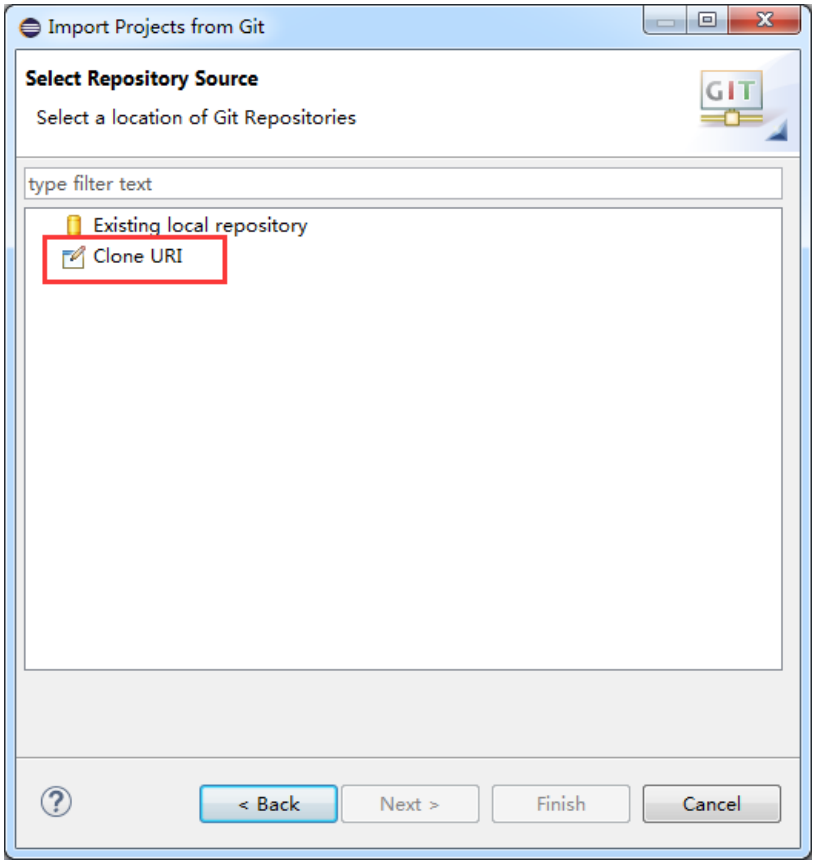
5）推送完成

（二）克隆远程仓库导入项目

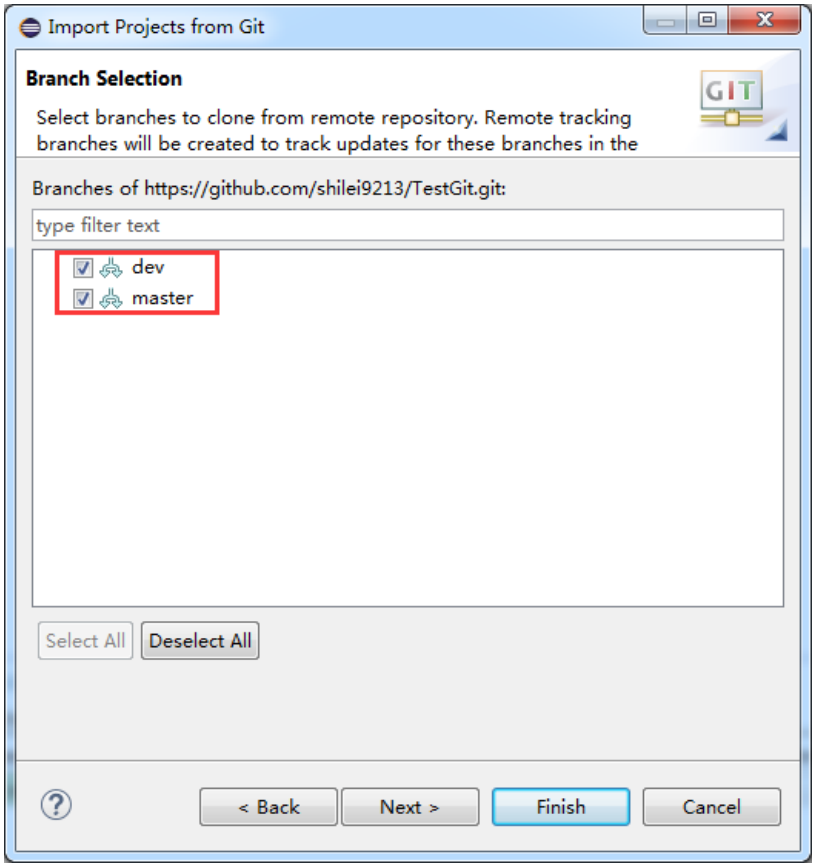
1）File => Import => Git => Project from Git



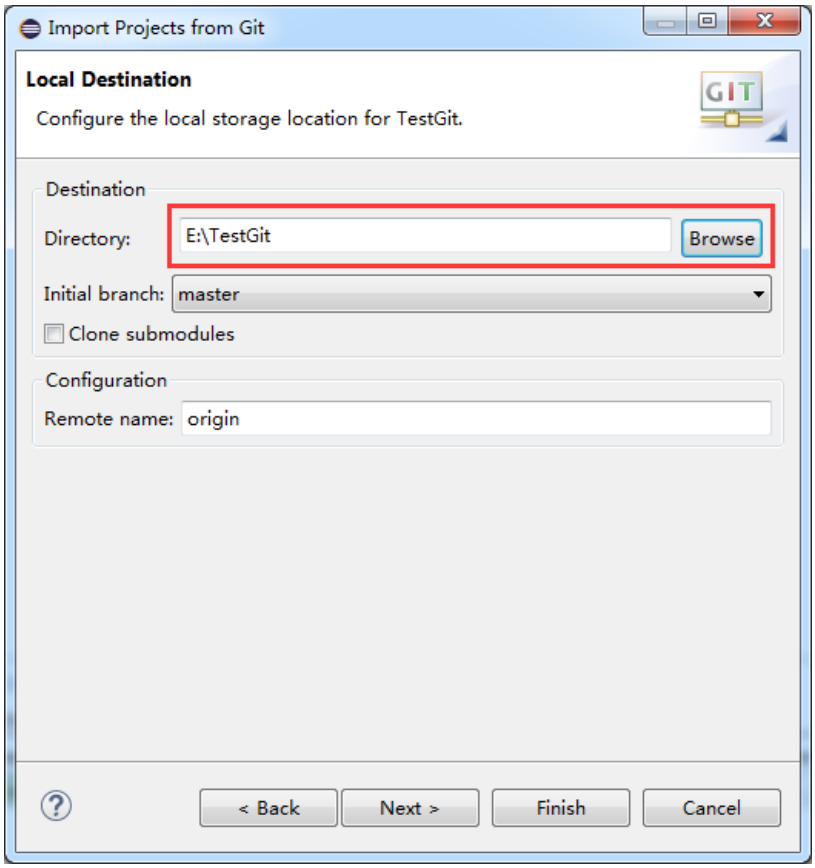
2）仓库来源选择：Clone from uri 填写账户信息



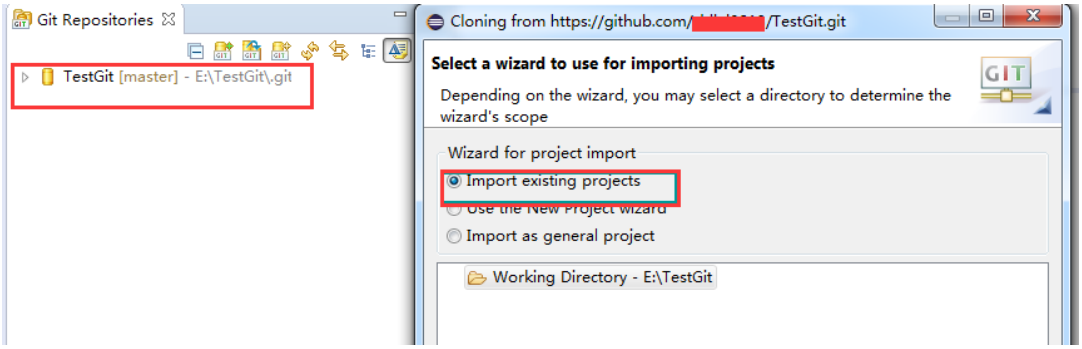
3）分支选择也没：选择要导入的分支



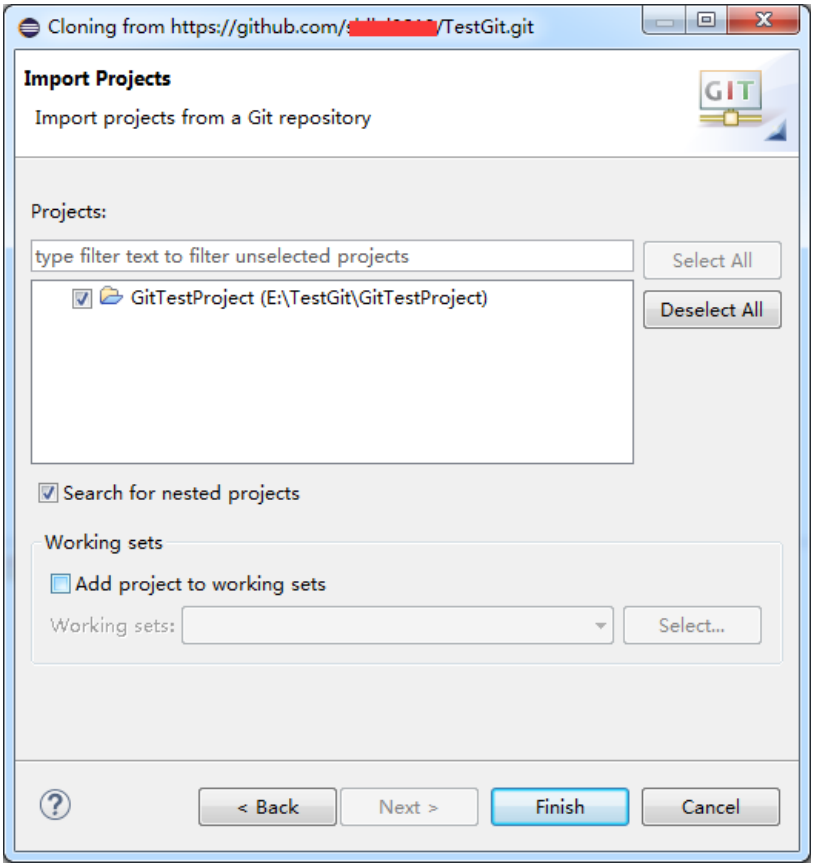
4）本地路径指定地址



5）完成后本地仓库出现，克隆完成，可以直接选择“Import as existing project”导入



6）导入



（三）提交修改到远程仓库程

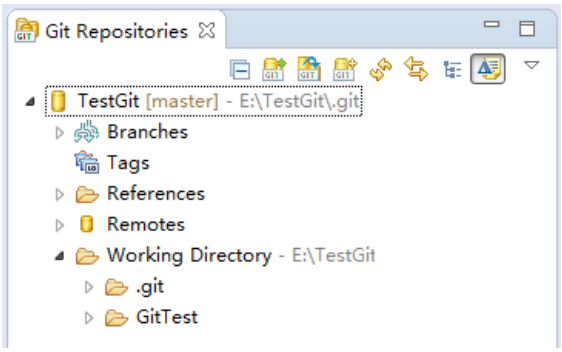
方法1：正常的提交操作：
项目右键 => Team => Remote => Push 可以选择不同的远程仓库进行推送。包括其他同事的仓库推送。

方法2：向默认远程仓库推送：
项目右键 => Team => Remote =>Push to upstream

（四）更新远程仓库修改到本地
方法1：
1）项目右键 => Team => Remote =>Fetch form upstream
注：该操作首先将远程的分支同步到本地的origin/master分支
2）项目右键 => Team => Marge
注：合并，可以查看和本地是否有冲突，可以通过合并解决冲突。

方法2： 项目右键 => Team => Pull
注：Pull 合并了Fetch from upstream 和marge；强制Marge，但上面的方法更安全。

五 EGit 视图
（一）仓库视图
库视图是有用的在处理分支/标记和执行操作,以及处理远程存储库,让你所了解概况。
右键Team => Show in Repositories View可以看到

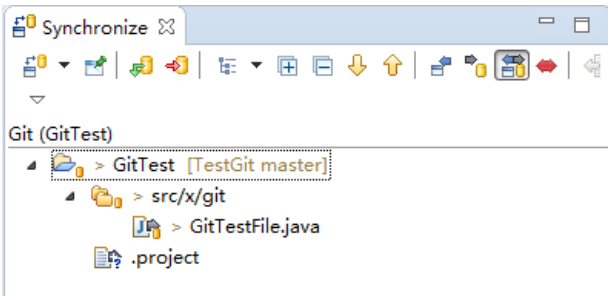


（二）同步视图

工作区和本地库的对比或者当前分支和其他分支的对比是通过同步操作完成的。

右键Team => Sychronize Workspace ，工作区会和当前分支比较并展示出不同。

右键Team => Advanced => Synchronize… 可以查看当前分支和其他分支的比较。这包括所有本地未提交的变化。



（三）历史视图

查看已共享文件的历史，右键Team => Show in History.

可以进行比较，标签，回退。