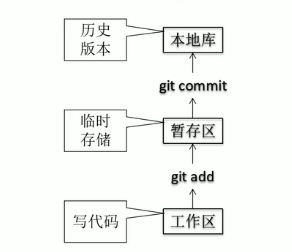
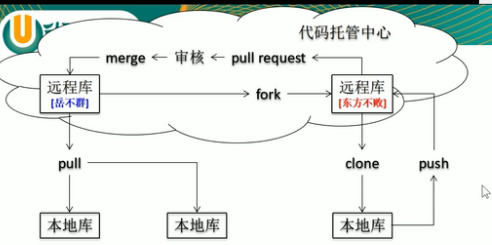
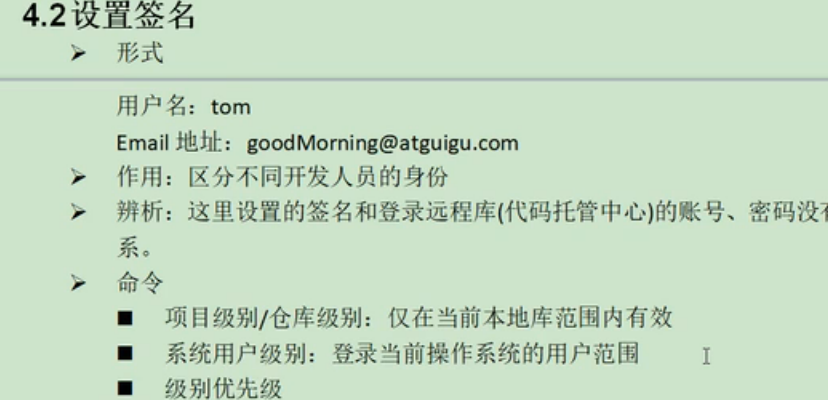
**Git 笔记**





初始化本地仓库命令：git init，该命令能产生 . git 的隐藏文件

签名设置：



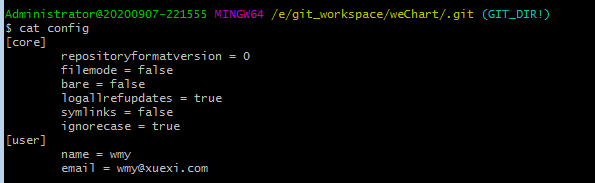


预备知识：

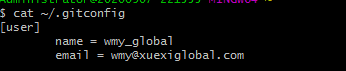
* . 开头的文件都是隐藏的
* cd ~ 打开home目录，home目录下有隐藏目录 需要用 $ ls -la|less 命令

Git 项目级别的签名存储在：

首先，找到本地仓库，然后打开.git文件夹，查看config



Git 系统级别的签名存储在：



**Git 基本操作：**

1. 状态查看操作 git status

查看工作去、暂存区状态

1. 添加操作 git add <file name>

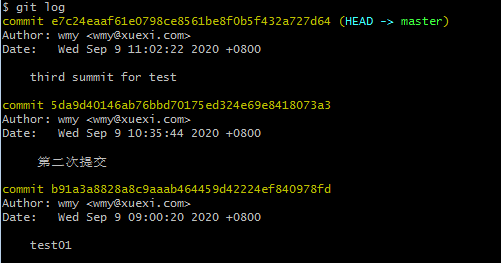
将工作区的“”新建/修改“”添加到暂存区

1. 提交操作 git commit -m “msg” <file name>

将暂存区的内容提交到本地库

1. 查看历史记录

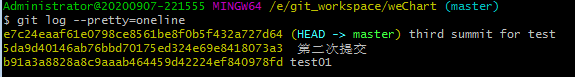
git log



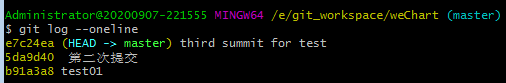
该方式显示详细，可能用到如下命令

多屏显示的控制方式：空格向下翻页、b向上翻页、q退出

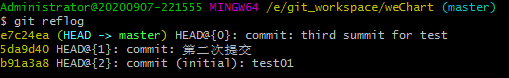
git log --pretty=oneline 简洁显示



git log --oneline 更简洁显示



git reflog



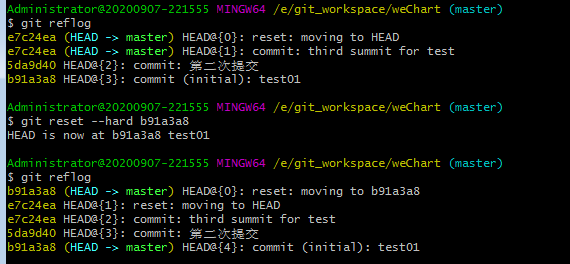
HEAD@{移动到当前版本需要几步}

1. 前进后退

本质：HEAD是一个指针，指向当前版本，通过移动HEAD指针就可以实现版本前进后退

方式一：基于索引操作（推荐）

Git reset --hard [局部索引值]



reset命令的三个参数对比

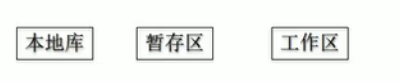
* -soft 参数
* 仅仅在本地库移动HEAD指针



* -mixed 参数
* 在本地库移动HEAD指针
* 重置暂存区



* -hard 参数
* 在本地库移动HEAD指针
* 重置暂存区
* 重置工作区

（一起动）

删除文件并找回

* 前提：删除前，文件存在时的状态提交到了本地库。
* 操作：git reset --hard [指针位置]
* 删除操作已经提交到本地库：指针位置指向历史记录
* 删除操作尚未提交到本地库：指针位置使用HEAD（当前状态）

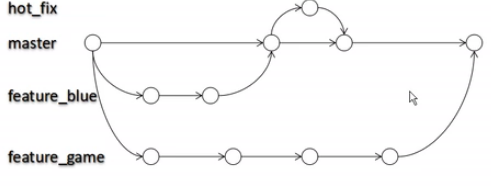
比较文件差异

* git diff [文件名]
* 将工作区中的文件和暂存区进行比较
* Git diff [本地库中历史版本] [文件名]
* 将工作区中的文件和本地库历史记录比较
* 不带文件名比较多个文件

分支管理

什么是分支？

在版本控制过程中，使用多条线同时推进多个任务



分支操作

* 创建分支

git branch [分支名]

* 查看分支

git branch -v

* 切换分支

git checkout [分支名]

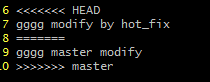
* 合并分支

第一步：切换到接受修改的分支（被合并，增加新内容）上

第二步：执行merge命令（git merge [有新内容分支名]）

* 解决冲突
* 冲突的表现

Hot\_fix的修改



master的修改

* 冲突的解决

第一步：编辑文件，删除特殊符号

第二步：把文件修改到满意的程度，保存退出

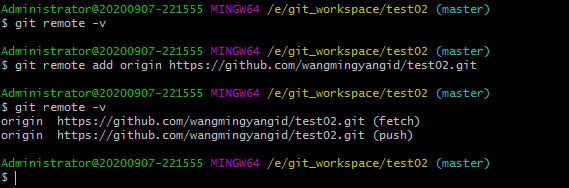
第三步：git add [文件名]

第四步：git commit -m “日志信息”

注意：此时commit一定不能带具体文件名

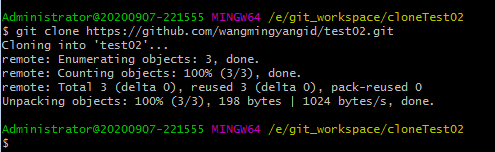
**推送给远程：**

Git remote add origin [远程库地址] 给远程库起个别名



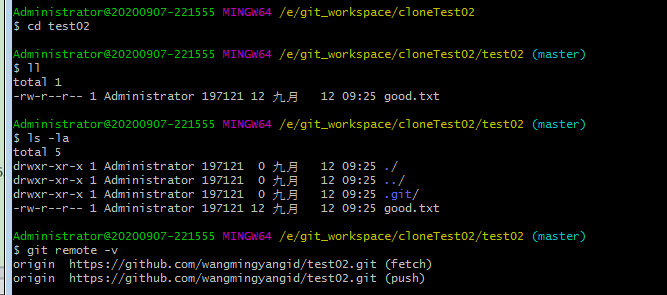
git push origin master 表示往origin所代表的地址进行推送，推送master分支

**Clone远程库**



**Clone后的文件特征**

* 完整的把远程库下载到本地
* 创建origin远程地址别名
* 初始化本地库

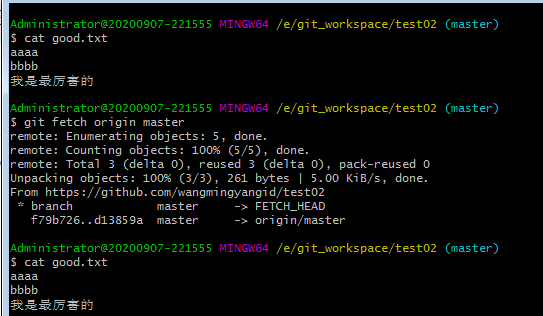


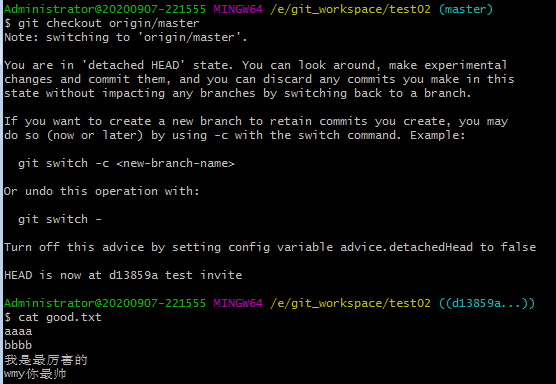
Windows中有个凭据管理器，会记住网站的地址，所以模拟协同开发时，要清除记录，否则永远都是当前用户在做操作。



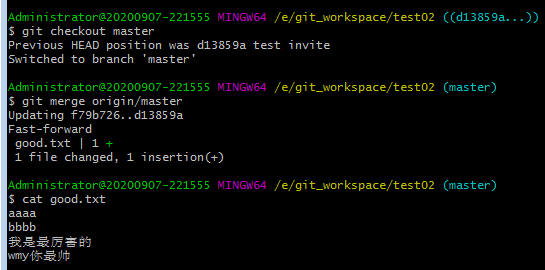
**git pull**

* 有两部分组成：git fetch、git merge
* Git fetch [远程库地址别名] [远程分支名] ( 只是拉取，但是不会合并，本地的分支没有远程分支的内容，通过git checkout [远程库别名]/[远程库分支名] 可以查看到远程的内容，如：git checkout origin/master )





* Git merge [远程库地址别名] [远程分支名]



**解决冲突**

* 要点：
* 如果不是基于GitHub远程库的最新版所做的修改，不能推送，必须先拉取
* 拉取下来后如果进入冲突状态，则按照“分支冲突解决”操作解决即可

**Eclipse 中忽略文件**

* 概念：Eclipse 特定文件

这些都是eclipse为了管理我们创建的工程而维护的文件，和开发的代码没有直接关系，最好不要在Git 中进行追踪，也就是把它们忽略。

. classpath 文件、. project文件、. settings目录下所有文件

* 为什么要忽略Eclipse特定文件呢？

同一个团队中很难保证大家使用相同的IDE工具，而IDE工具不同时，相关工程特定文件就有可能不同。如果这些文件加入版本控制，那么开发时很可能需要为了这些文件解决冲突。

* GitHub 官网样例文件

<https://github.com/github/gitignore>

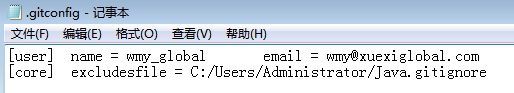
<https://github.com/github/gitignore/blob/master/Java.gitignore>

|  |
| --- |
| # Compiled class file  \*.class  # Log file  \*.log  # BlueJ files  \*.ctxt  # Mobile Tools for Java (J2ME)  .mtj.tmp/  # Package Files #  \*.jar  \*.war  \*.nar  \*.ear  \*.zip  \*.tar.gz  \*.rar  # virtual machine crash logs, see http://www.java.com/en/download/help/error\_hotspot.xml  hs\_err\_pid\*  .classpath  .project  .settings  target |

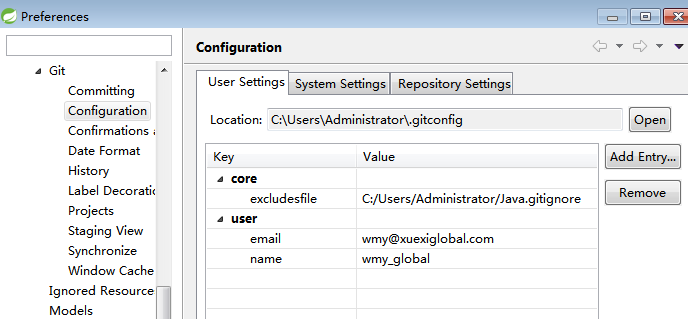
在git 的全局配置文件 (.gitconfig文件) 路径下，加入Java.gitignore 文件夹；

并在.gitconfig中配置[core] excludesfile = C:/Users/Administrator/Java.gitignore

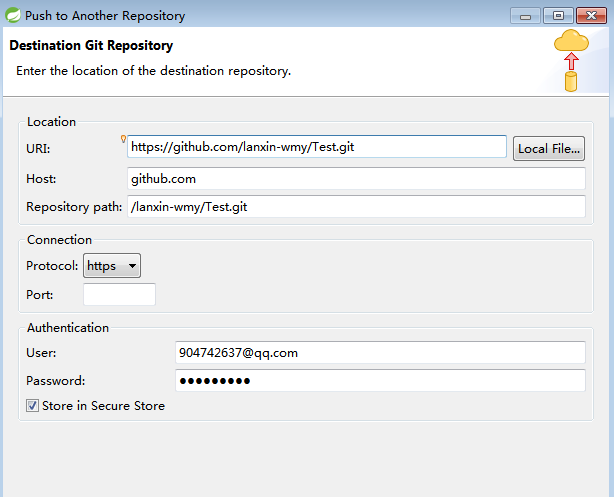
注意：这里路径中一定要使用“/”，不要使用“\”



在sts中会显示相关配置

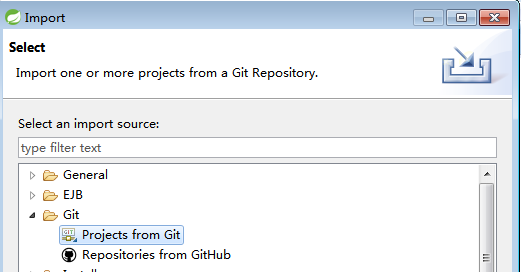


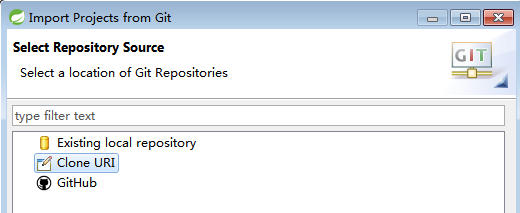
**Eclipse中进行push**

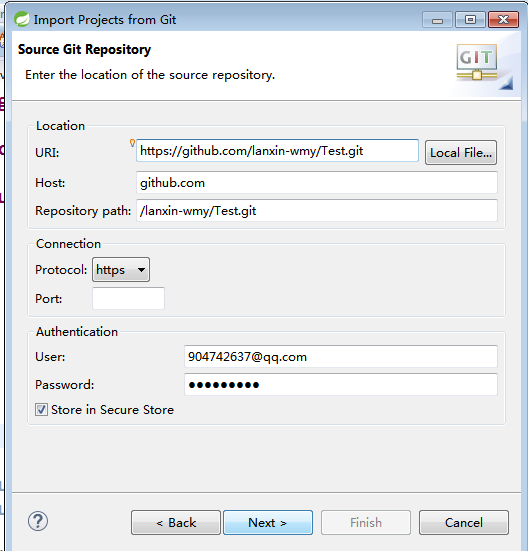


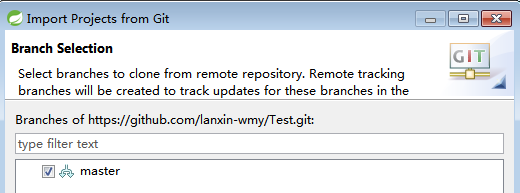
**从git导入工程**

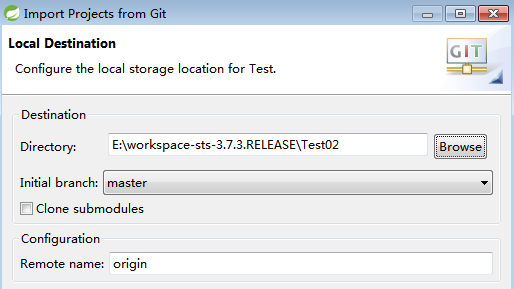
* **Import ...导入工程**

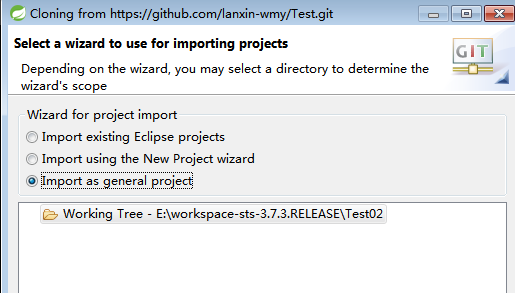




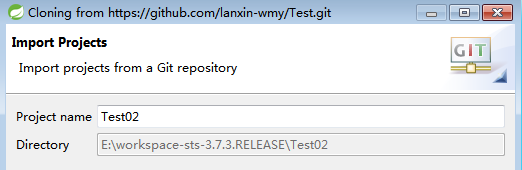


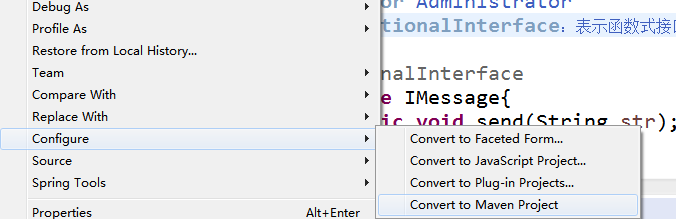




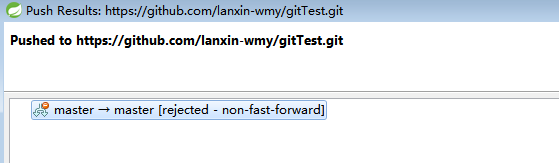


此处只能选择第三个，第一个代表已经存在的Eclipse project 但是我们当初push远程的时候忽略了很多eclipse文件，所以现在clone下来的已经不算是一个完整的工程；第二个会使目录结构改变。

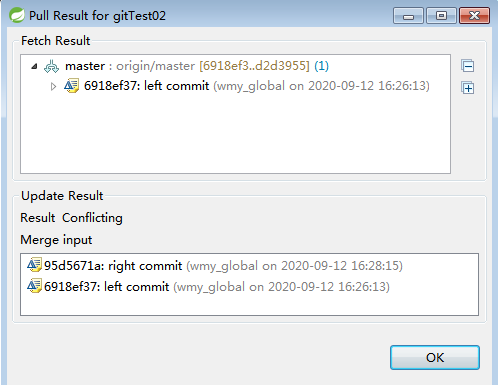




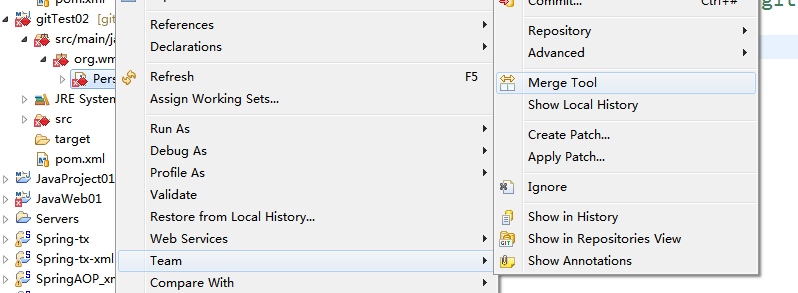
**Eclipse冲突解决**

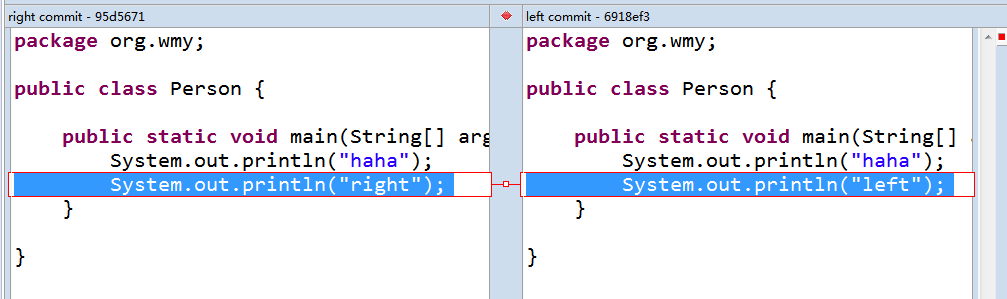


* 第一步：后提交造成冲突的，先进行拉取（pull）



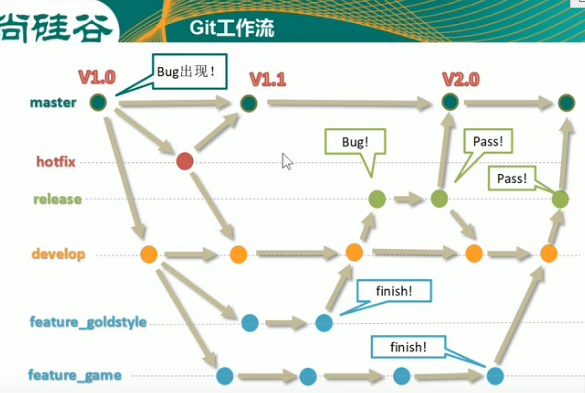
* 第二步：按如下方式进行点击





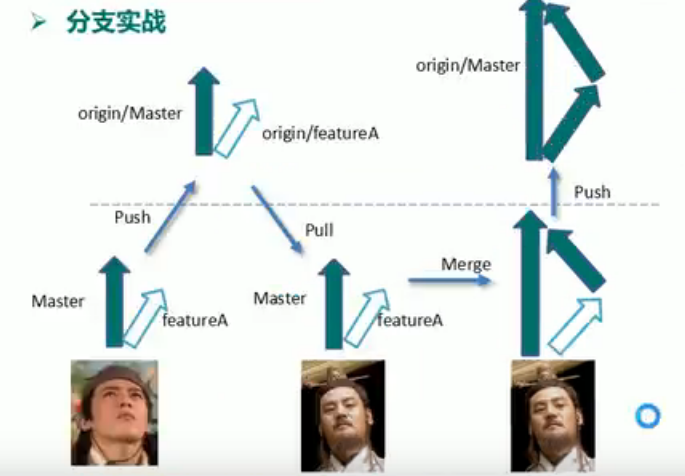
**Git工作流**

GitFlow工作流：通过为功能开发、发布准备和维护设立了独立的分支，让发布迭代过程更流畅。严格的分支模型也为大型项目提供了一些非常必要的结构。

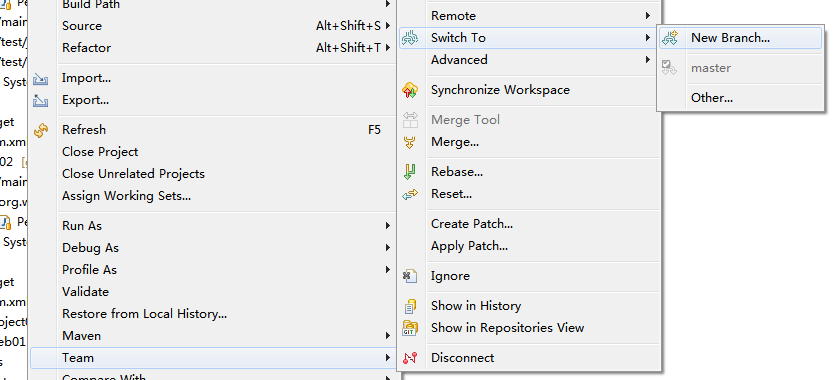


上图所示是比较全面的分支模型概述，实际工作中hotfix、release 可以进行取舍。

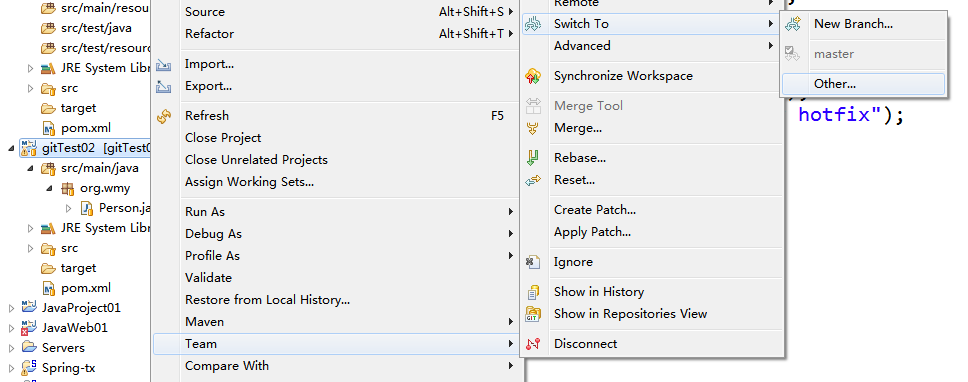
**分支实战图解**



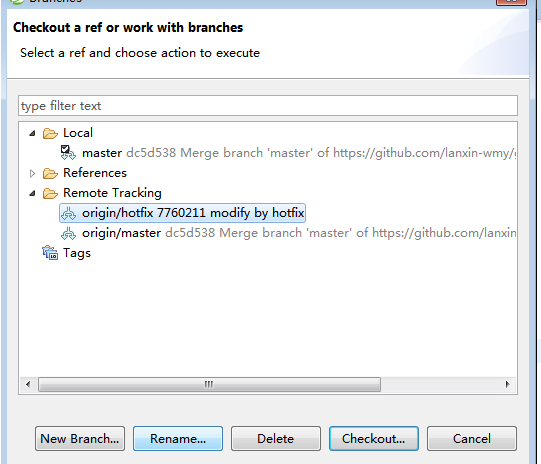
* 创建分支，然后操作，提交，推送

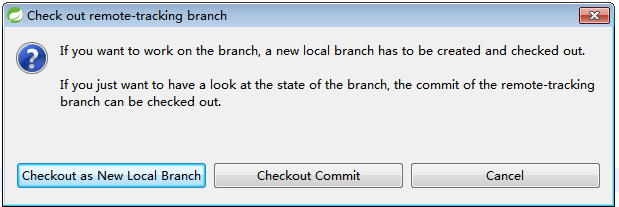


* 另一个工程直接pull就会拉取远程的分支，然后如图进行分支切换（其实本地此时，只是检测到远程新建了分支，并没有创建新分支，只有等到切换的时候才会创建分支并进行切换）

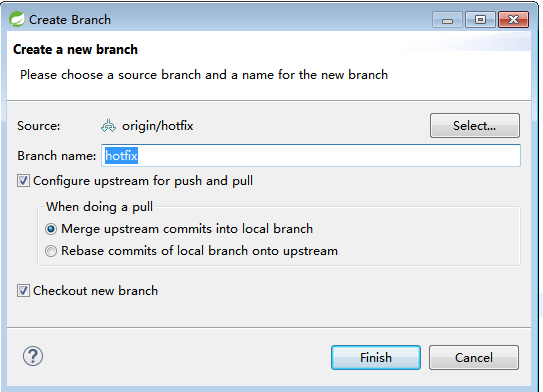


* 本地只有一个master，远程有hotfix，选择，点击checkout...

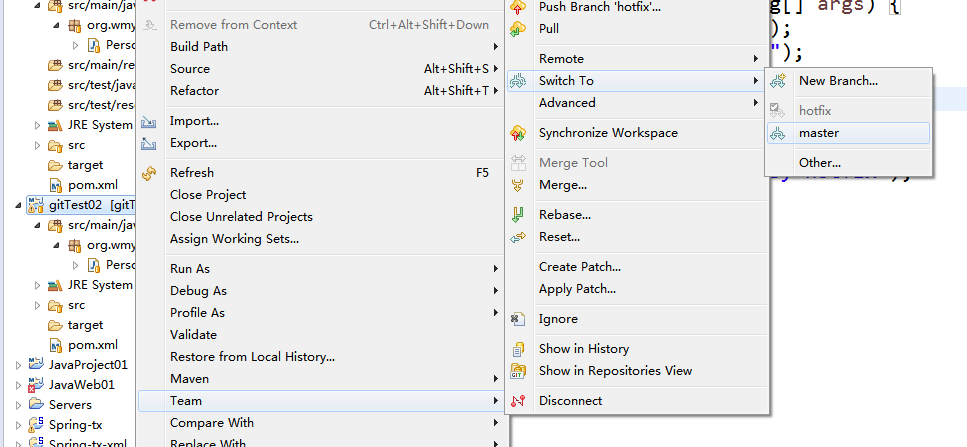




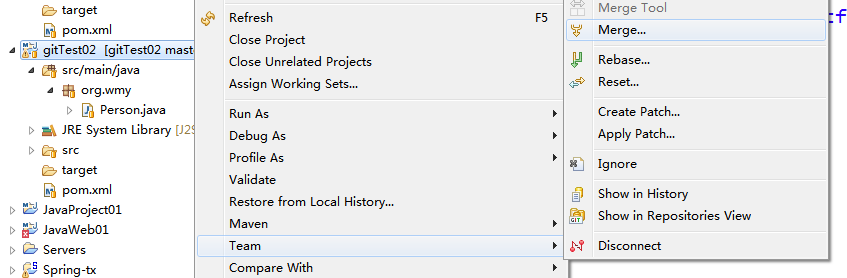
* 检出远程新分支

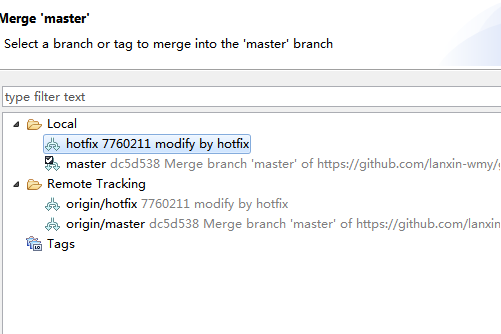


* 切换回主分支

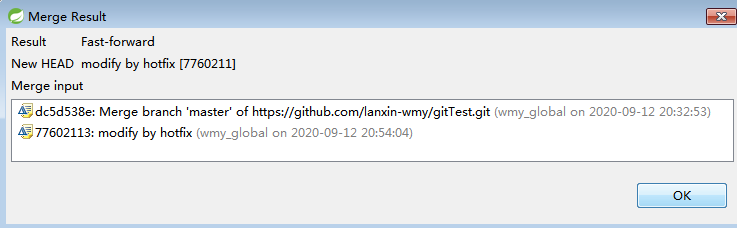


* 合并分支





* 合并结果



合并成功后把master推送到远程