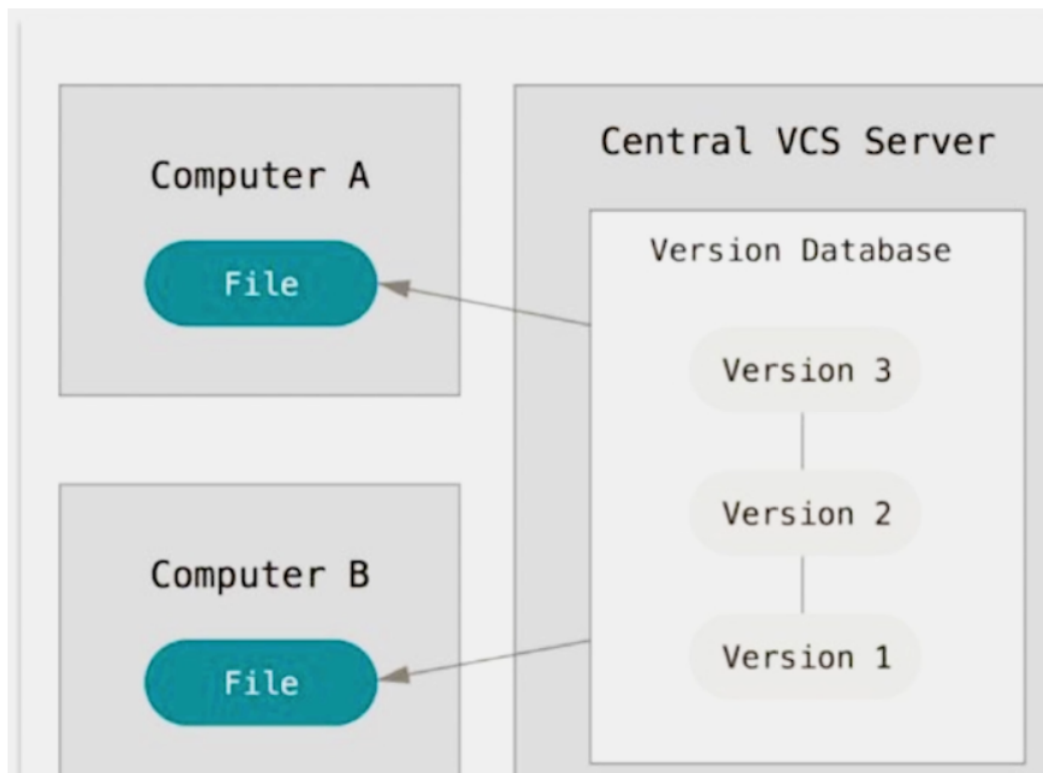


GIT基础功能培训

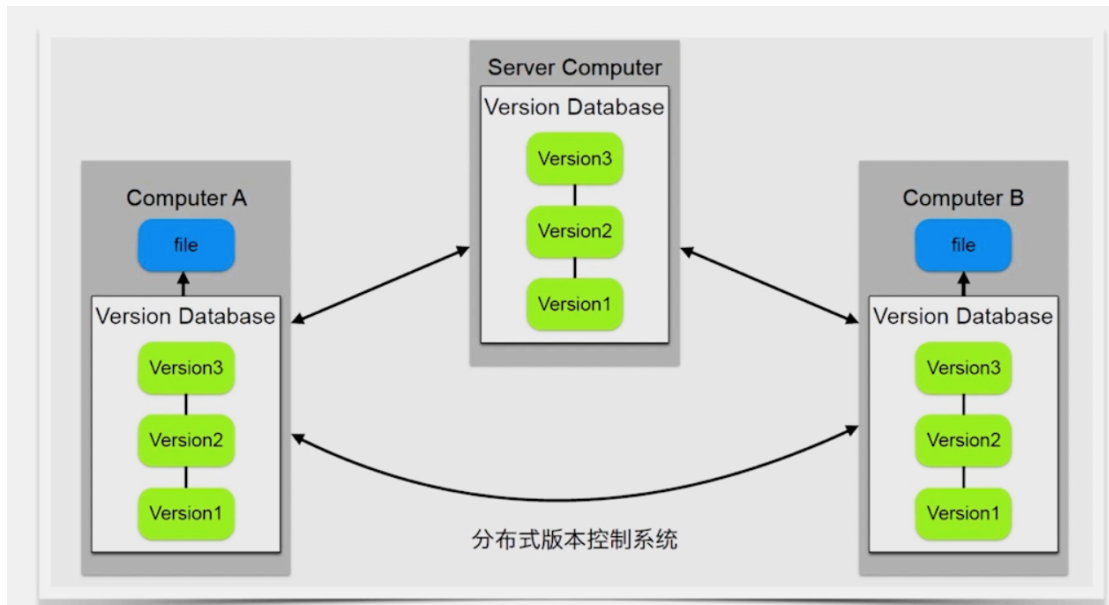
为什么要使用git,抛弃SVN?

SVN的特点:

- 有一个集中的版本管理服务器
- 具有文件版本和分支管理能力
- 但是客户端必须时刻和服务器相连



git的特点:



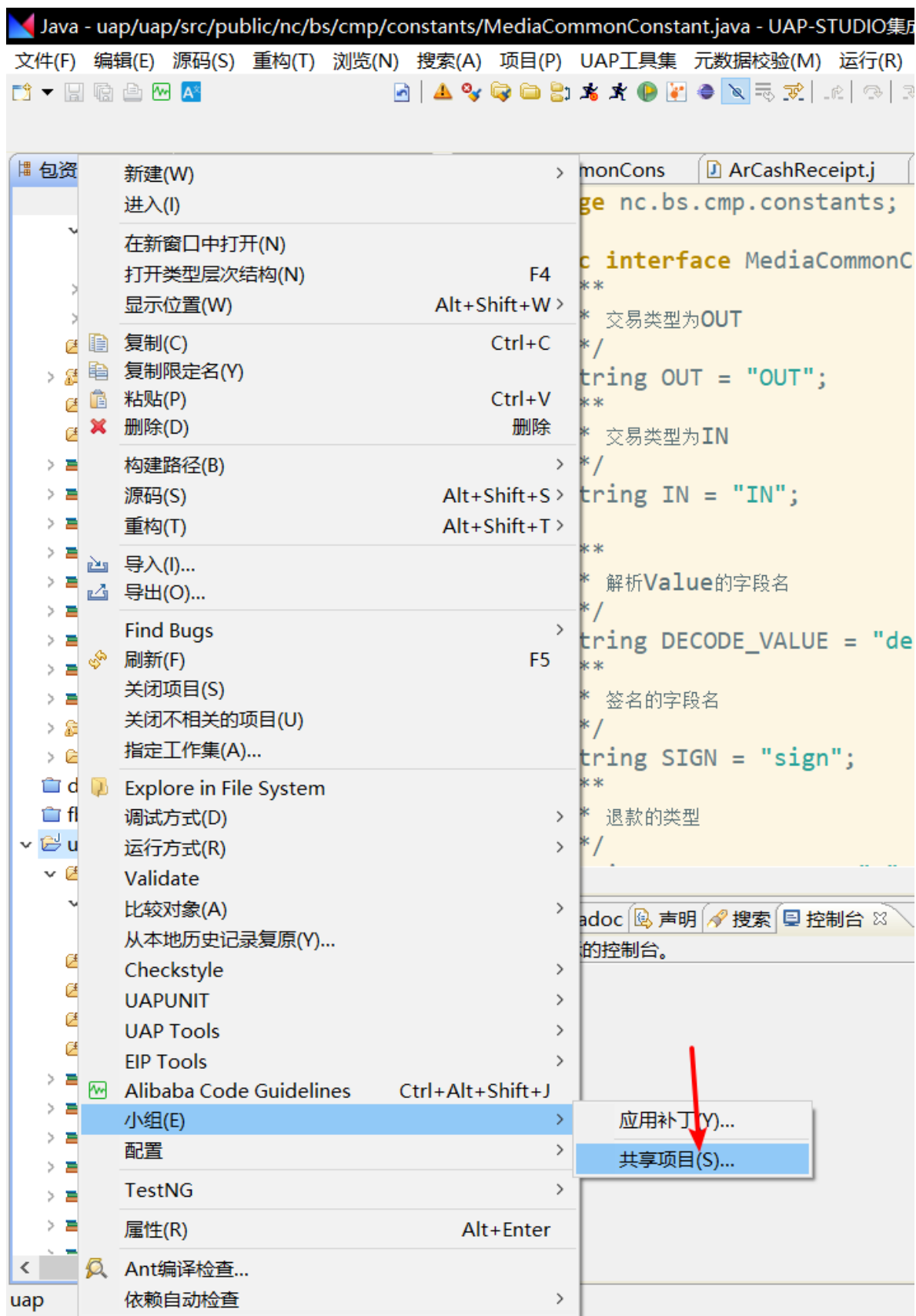
- 是分布式的版本管理服务器
- 脱离服务端,客户端照样可以管理版本
- 查看历史和版本比较等多数操作,都不需要访问服务器,比SVN这种集中式的版本控制更能提高版本管理效率
- 拥有优秀的存储能力

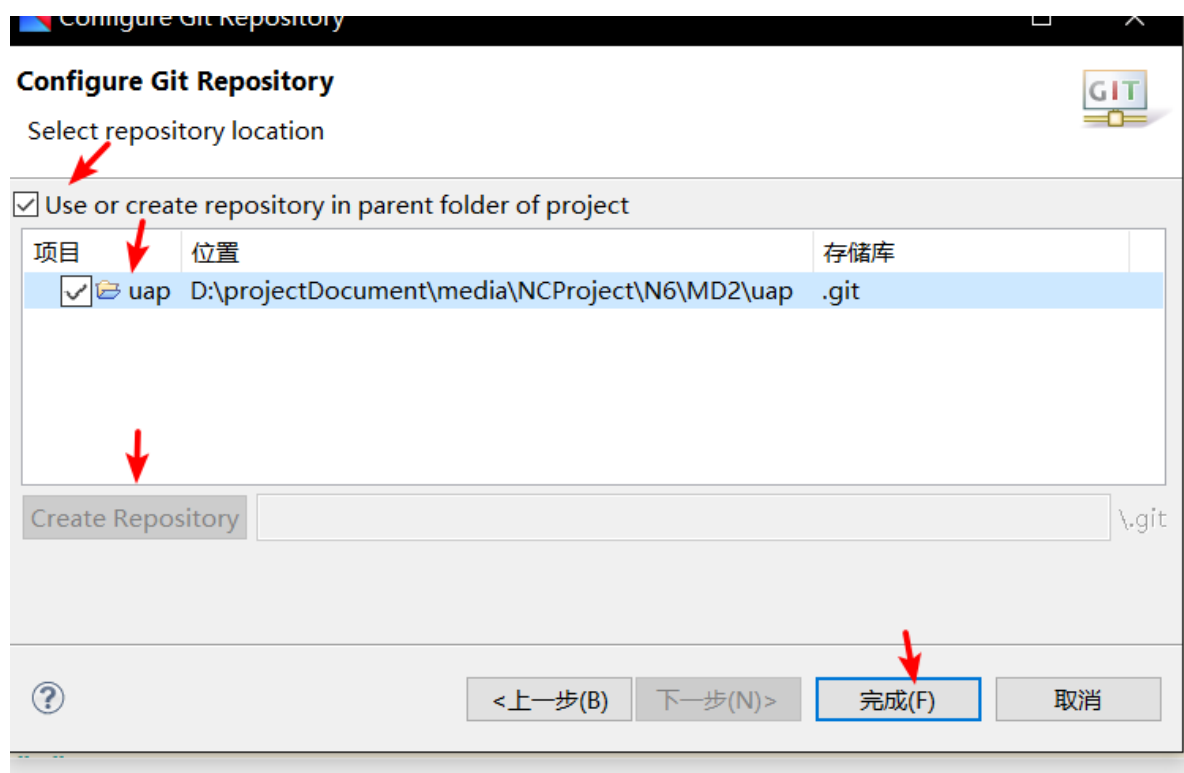
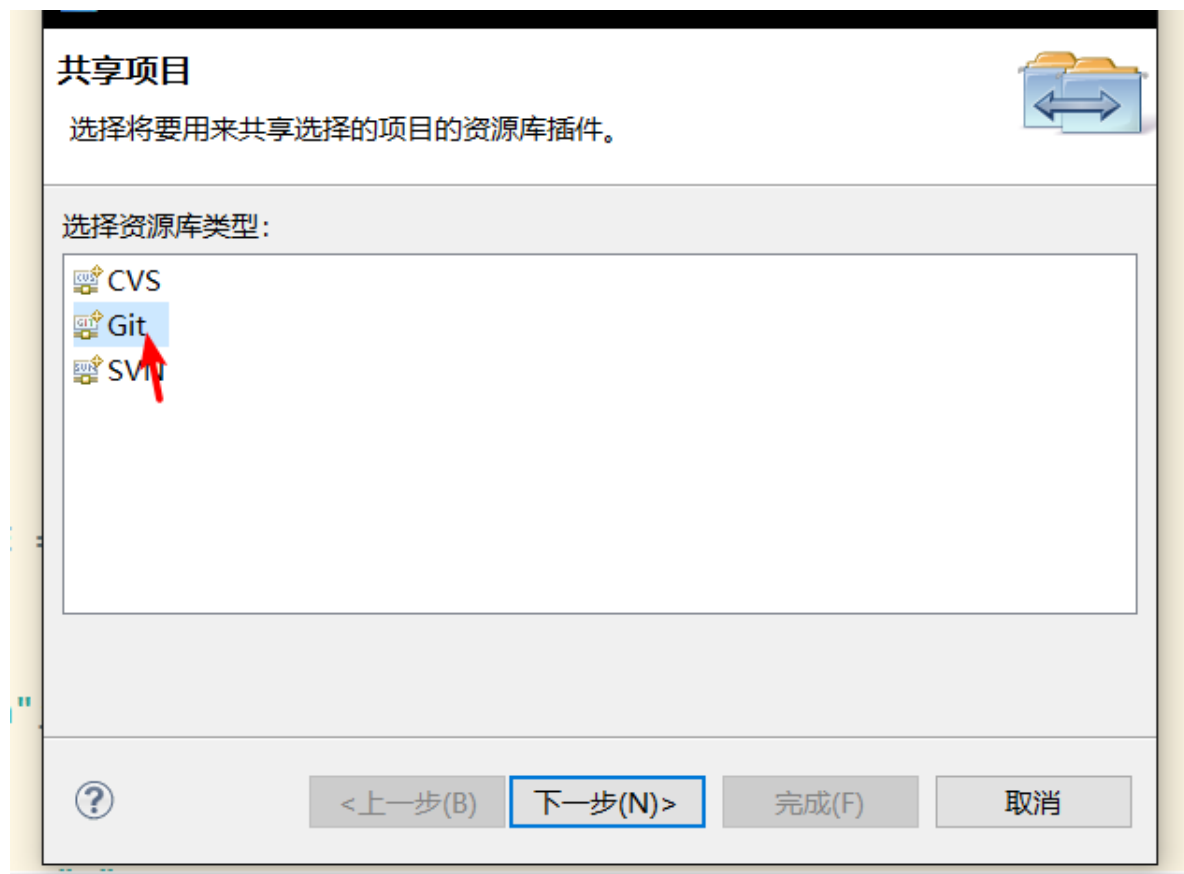
Git下载地址

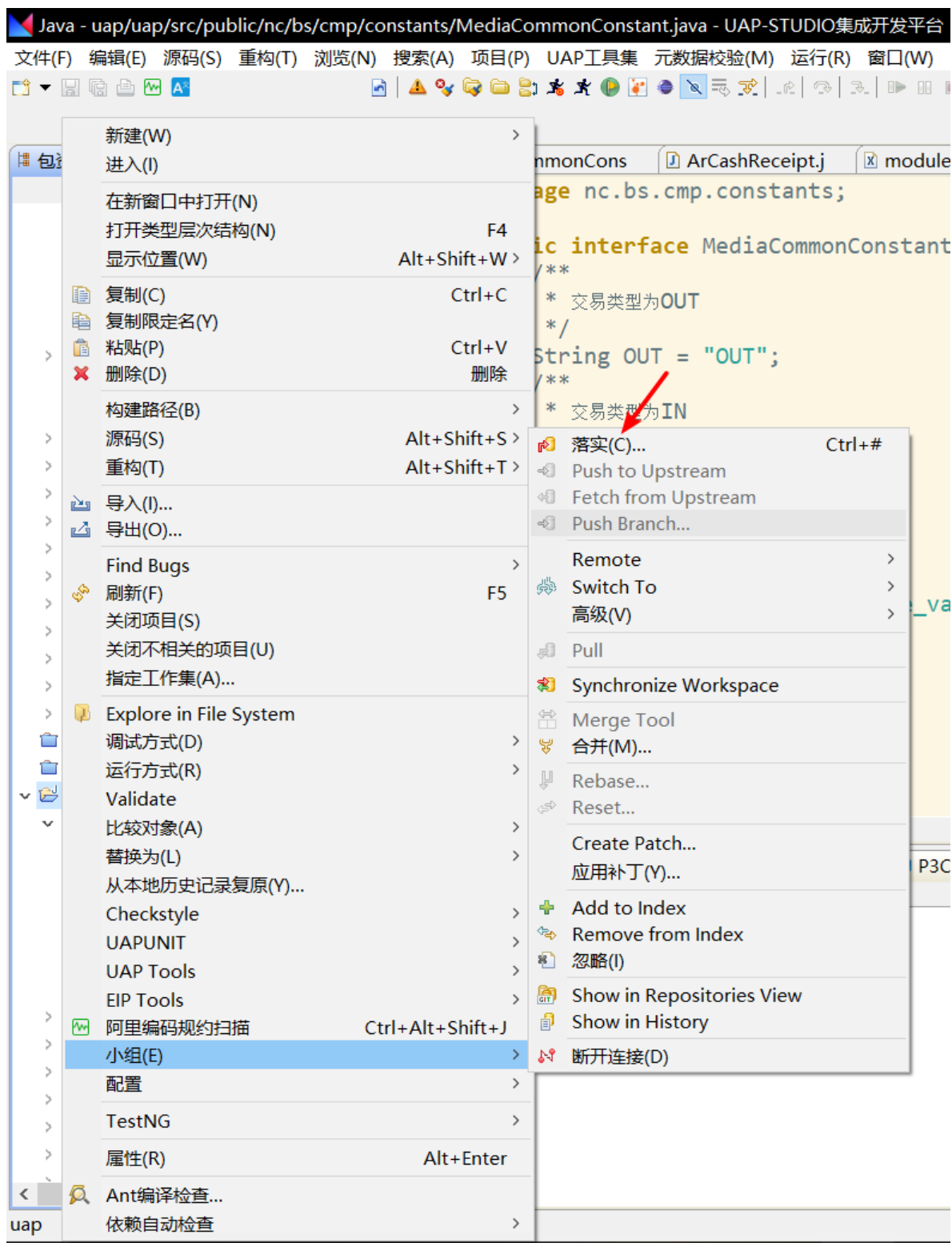
<https://git-scm.com/download/win>

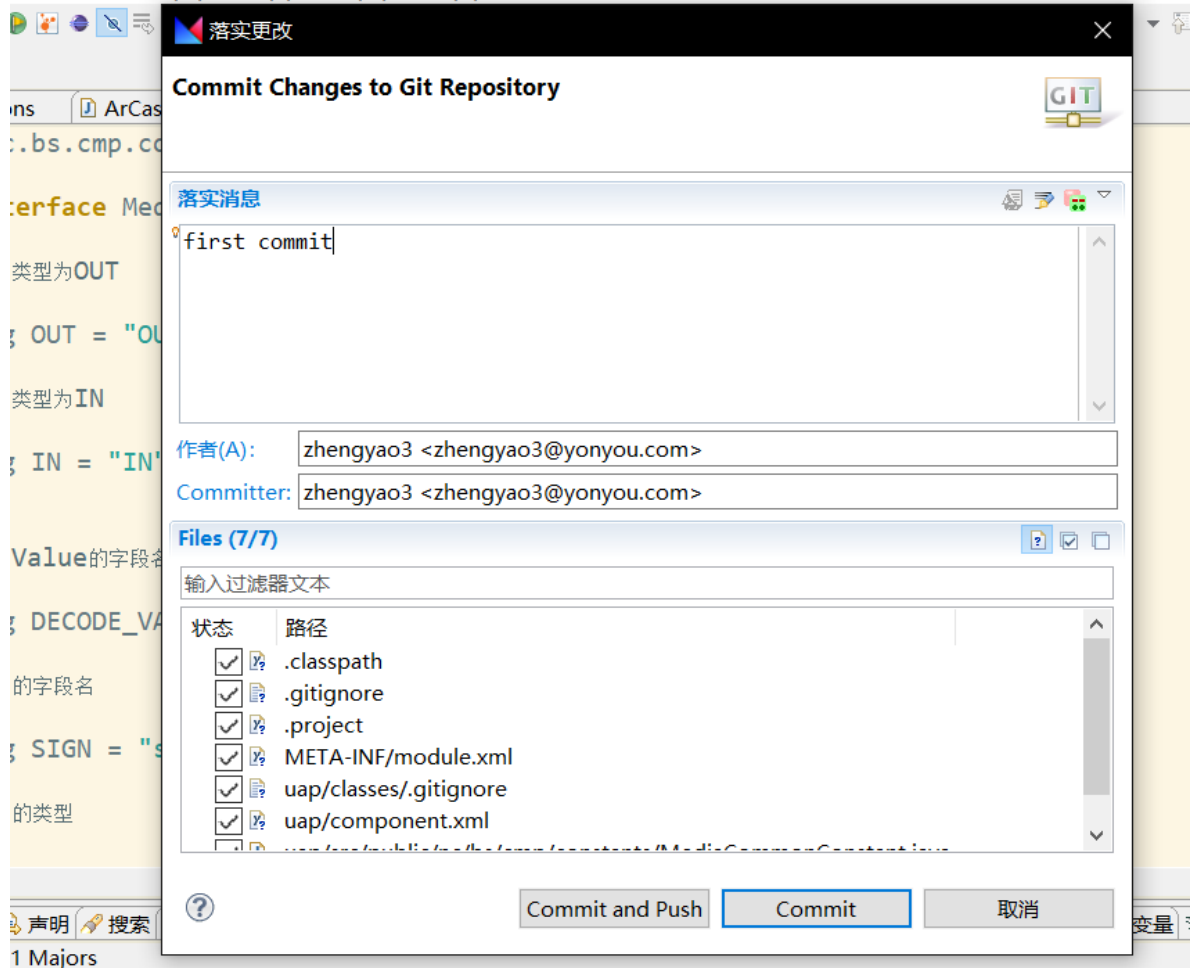
Eclipse下如何使用git

上传项目至git仓库









然后访问远程仓库: <http://218.13.1.186:3000/>, 输入账号密码, 都是各自的邮箱, 没有的找徐鹏哥帮忙注册一下

所有者 *

zhengyao3

仓库名称 *

BGY_NC633_NC_UAP

伟大的仓库名称一般都较短、令人深刻并且独一无二的。

可见性

☒ 该仓库为 私有的

仓库描述

请输入仓库描述, 最多为 512 个字符

剩余字符数: 512

.gitignore

Java x

授权许可

请选择授权许可文件

自述文档 ?

Default

☐ 使用选定的文件和模板初始化仓库

创建仓库
 取消

创建仓库

快速帮助

克隆当前仓库

不知道如何操作? 访问 [此处](#) 查看帮助!

HTTP

SSH

http://218.13.1.186:3000/zhengyao3/BGY_NC633_NC_UAP.git

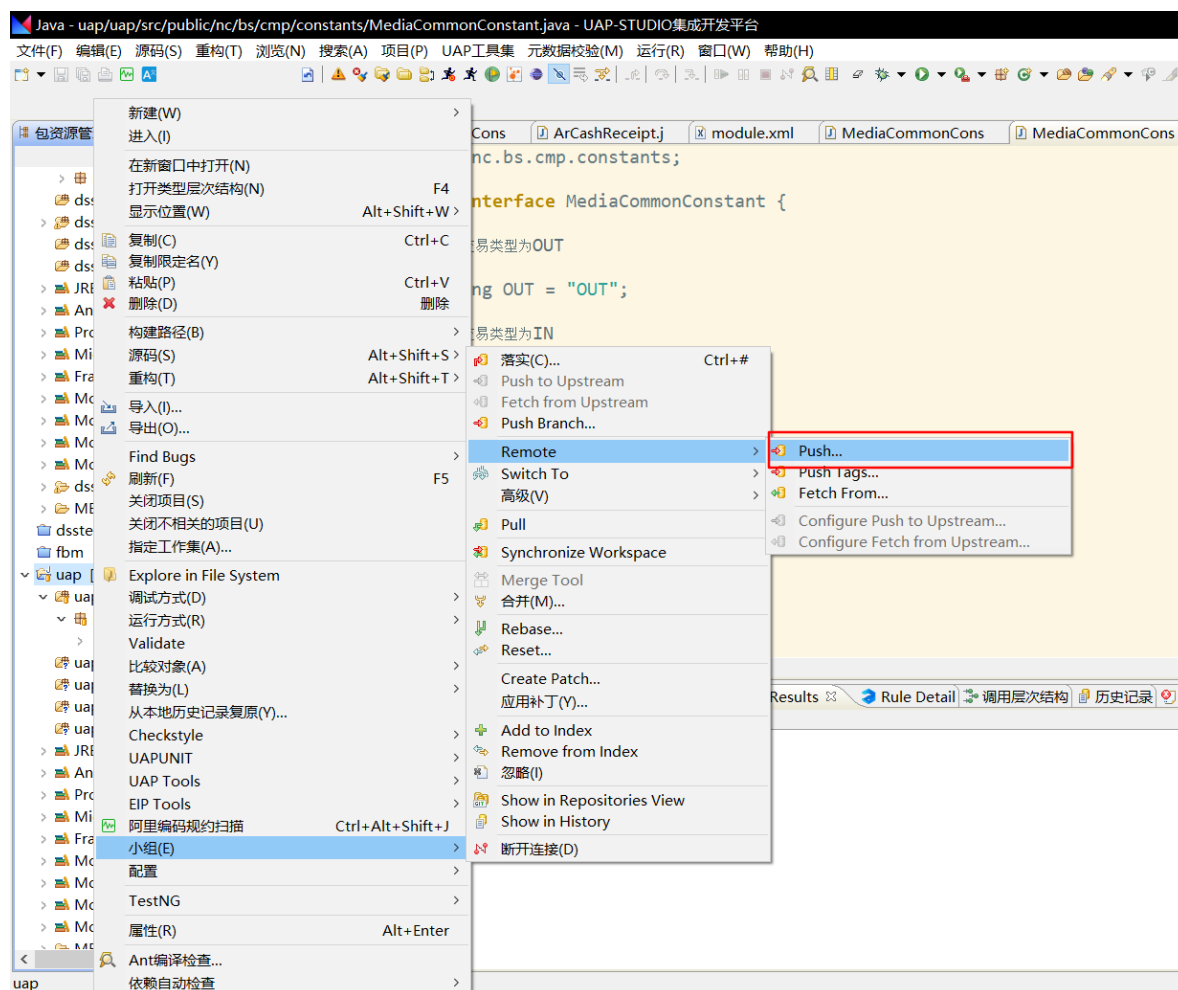
从命令行创建一个新的仓库

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin http://218.13.1.186:3000/zhengyao3/BGY_NC633_NC_UAP.git
git push -u origin master
```

从命令行推送已经创建的仓库

```
git remote add origin http://218.13.1.186:3000/zhengyao3/BGY_NC633_NC_UAP.git
git push -u origin master
```

此时复制生成的HTTP地址:http://218.13.1.186:3000/zhengyao3/BGY_NC633_NC_UAP.git



Push to Another Repository

Destination Git Repository

Enter the location of the destination repository.

位置

URI:

主机:

Repository path:

连接

Protocol:

Port:

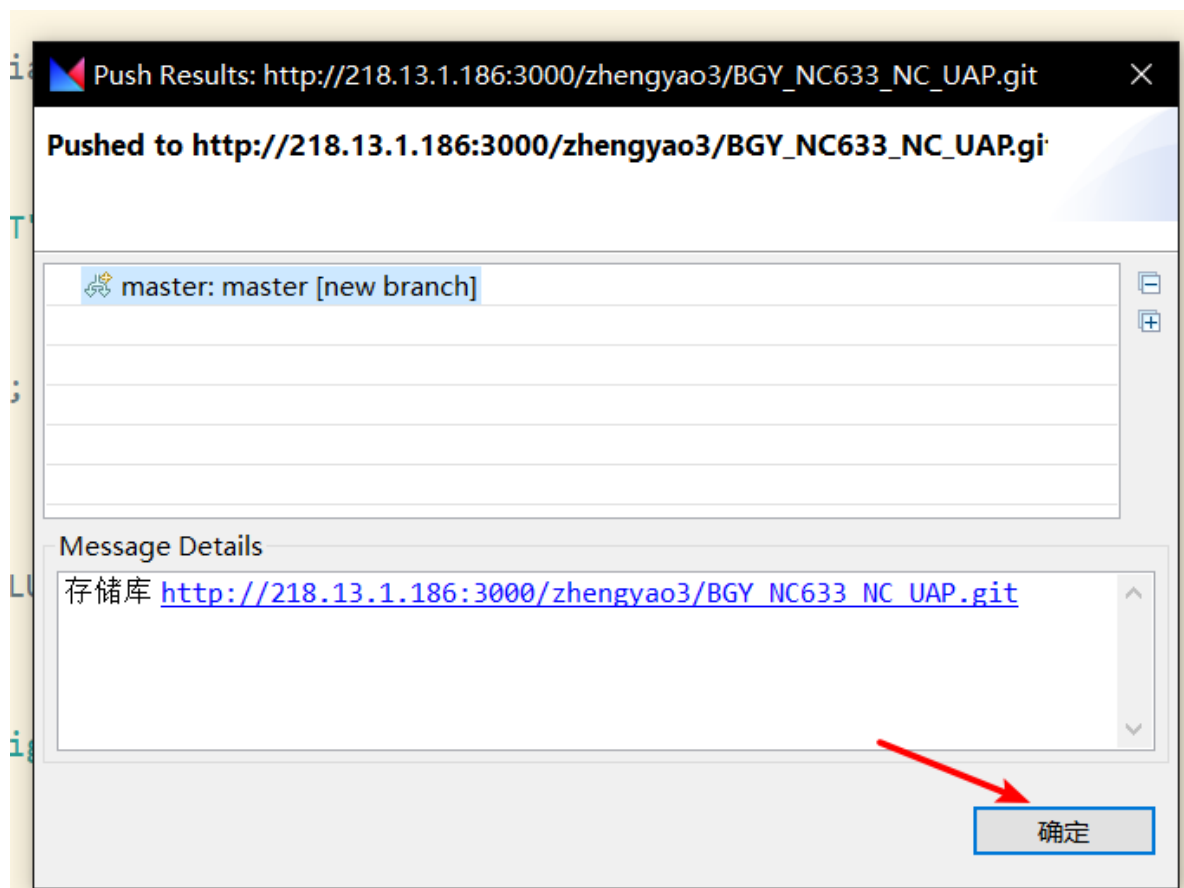
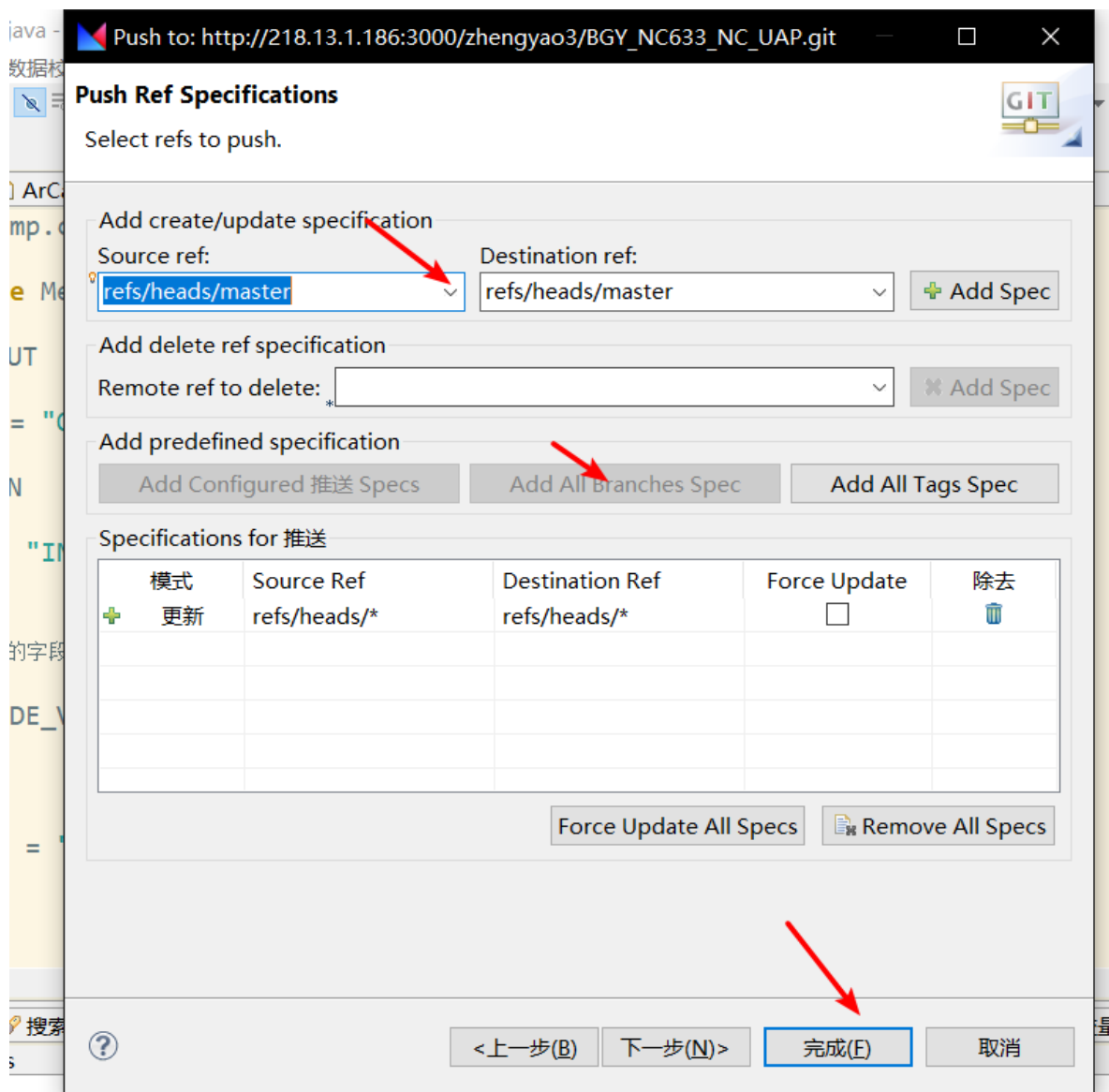
认证

用户(U):


Password:

Store in Secure Store ☒

点击下一步:



此时远程仓库的项目已经有了具体的项目了

 zhengyao3 / BGY_NC633_NC_UAP

取消关注1

点赞0

派生0

文件

工单管理0

合并请求0

Wiki

仓库设置

暂无描述

提交历史1

代码分支1

版本发布0

分支: master

BGY_NC633_NC...







新的文件 上传文件

HTTP SSH

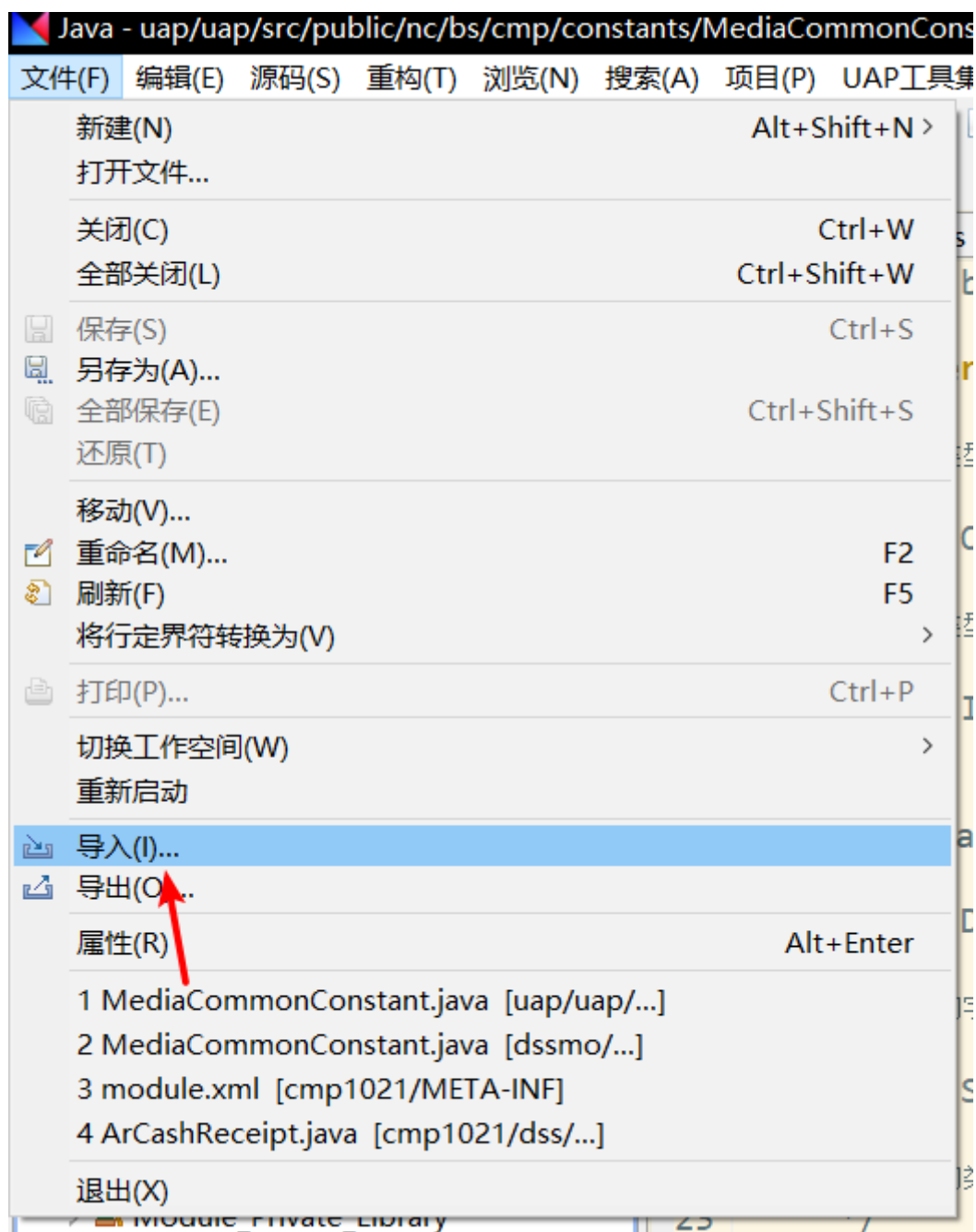
http://218.13.1.186:3000/zt

clone icon

download icon

 zhengyao3	9b52671200	first commit	5 分钟之前
 META-INF	9b52671200	first commit	5 分钟之前
 uap	9b52671200	first commit	5 分钟之前
 .classpath	9b52671200	first commit	5 分钟之前
 .gitignore	9b52671200	first commit	5 分钟之前
 .project	9b52671200	first commit	5 分钟之前

从远程数据库中下载项目




选择

从Git仓库中导入一个或多个项目



选择导入源(S):

输入过滤器文本

- > 文件夹 EIP资源
- ▼ 文件夹 Git
 -  来自Git项目
- > 文件夹 SVN
- > 文件夹 XML
- > 文件夹 安装
- > 文件夹 插件开发
- > 文件夹 小组
- > 文件夹 运行 / 调试

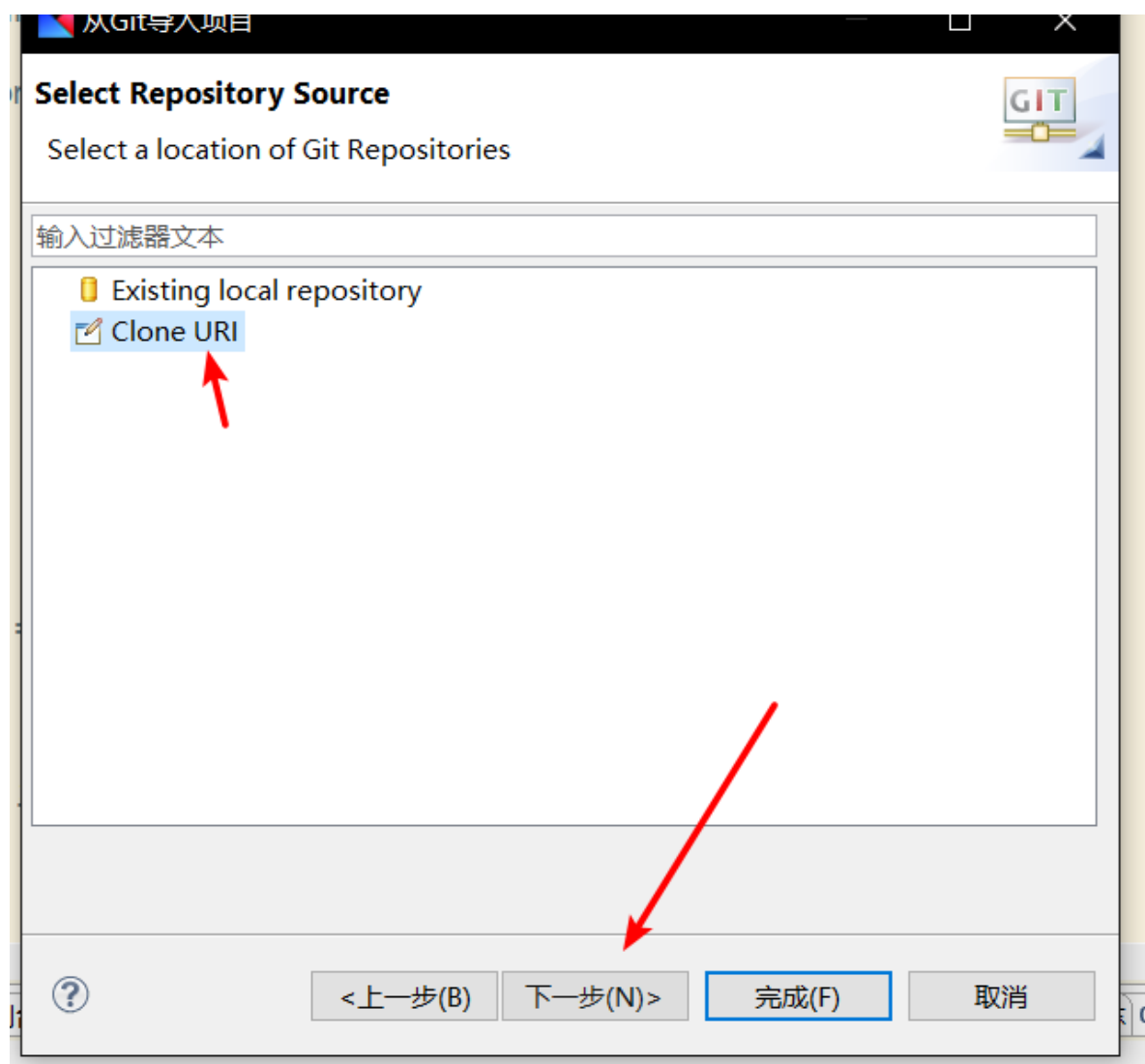


<上一步(B)

下一步(N)>


完成(F)

取消



从Git导入项目

Source Git Repository



Enter the location of the source repository.

位置

URI:

主机:

Repository path:

连接

Protocol:


Port:

认证

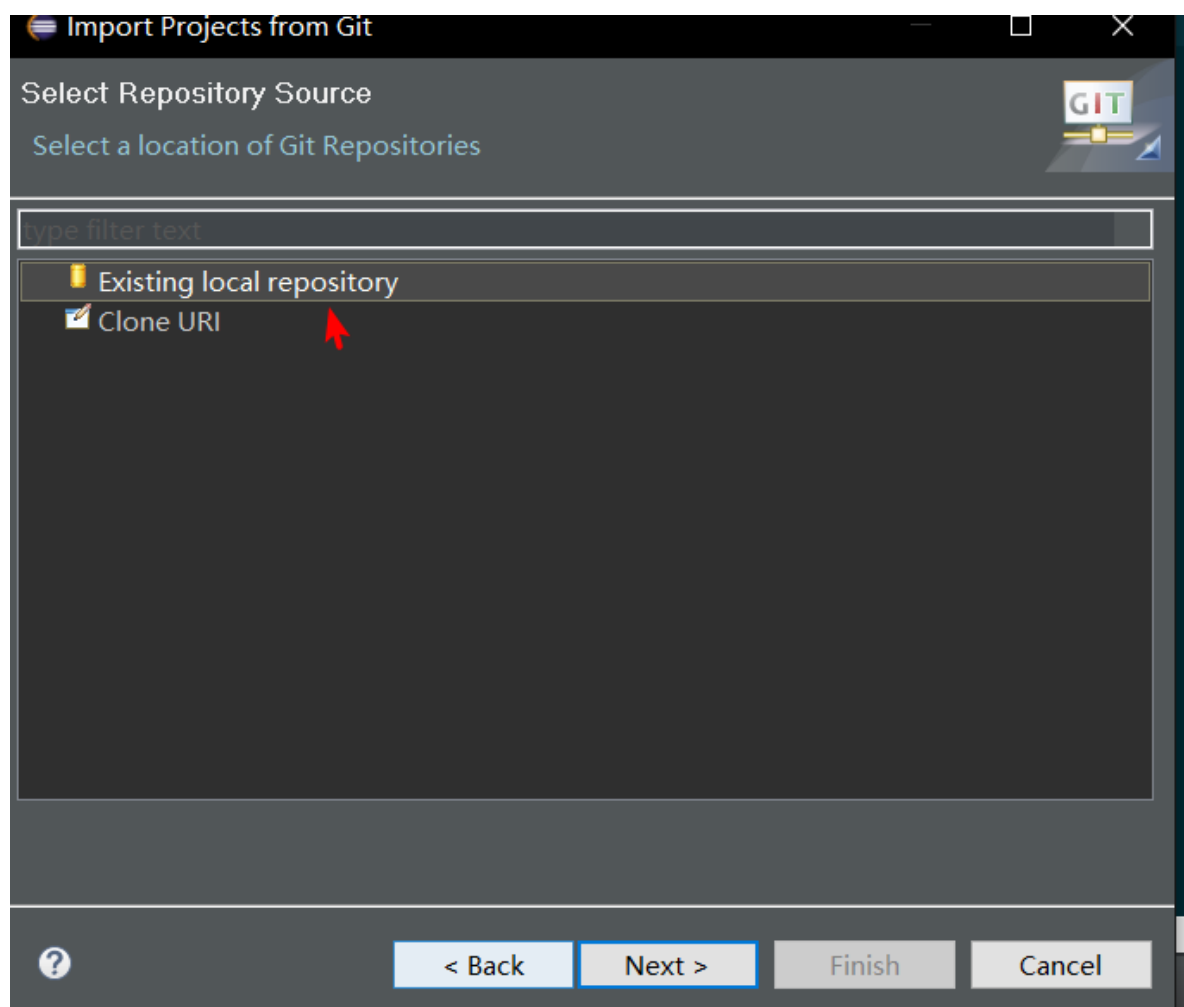
用户(U):

Password:

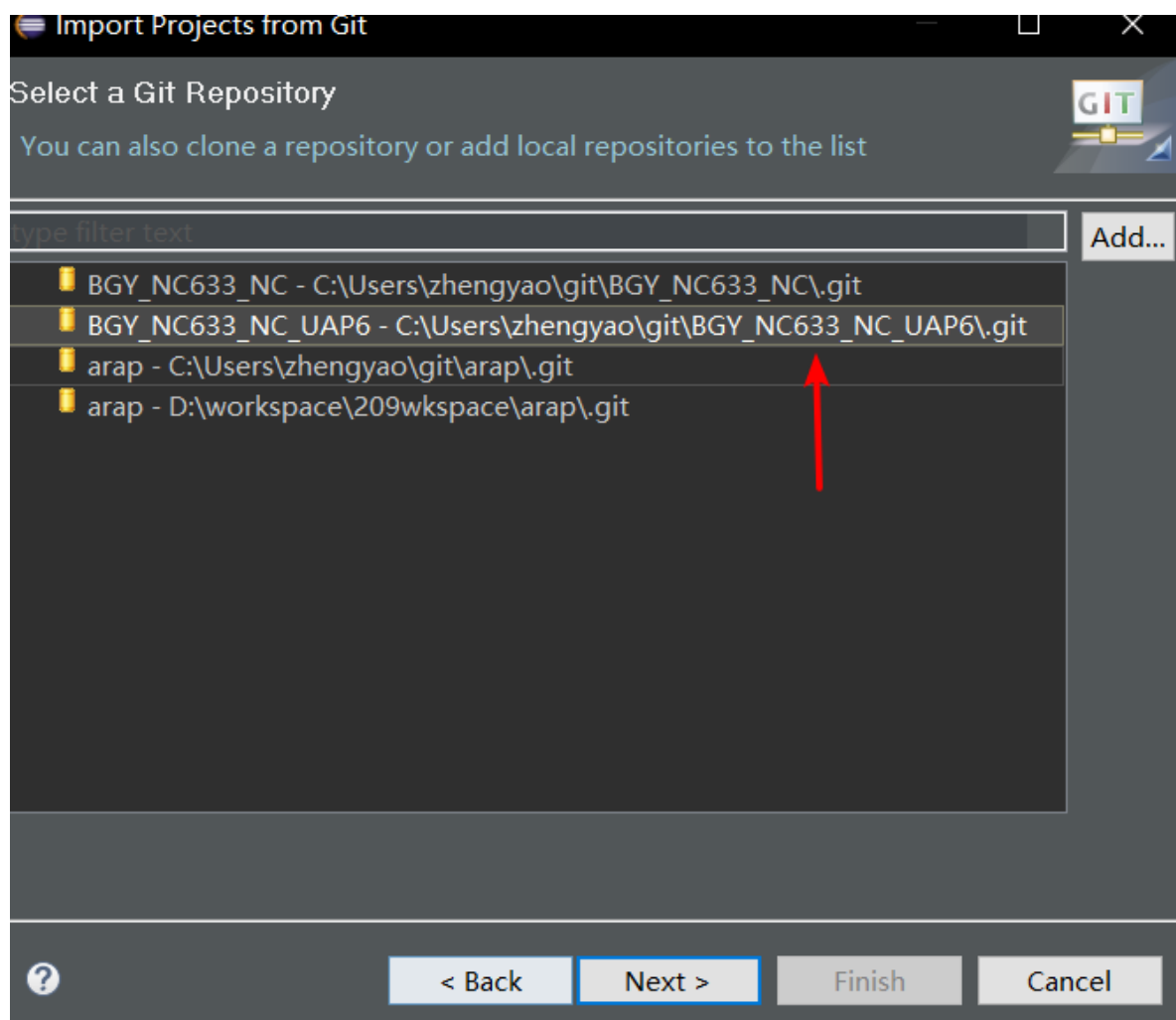
Store in Secure Store ☒

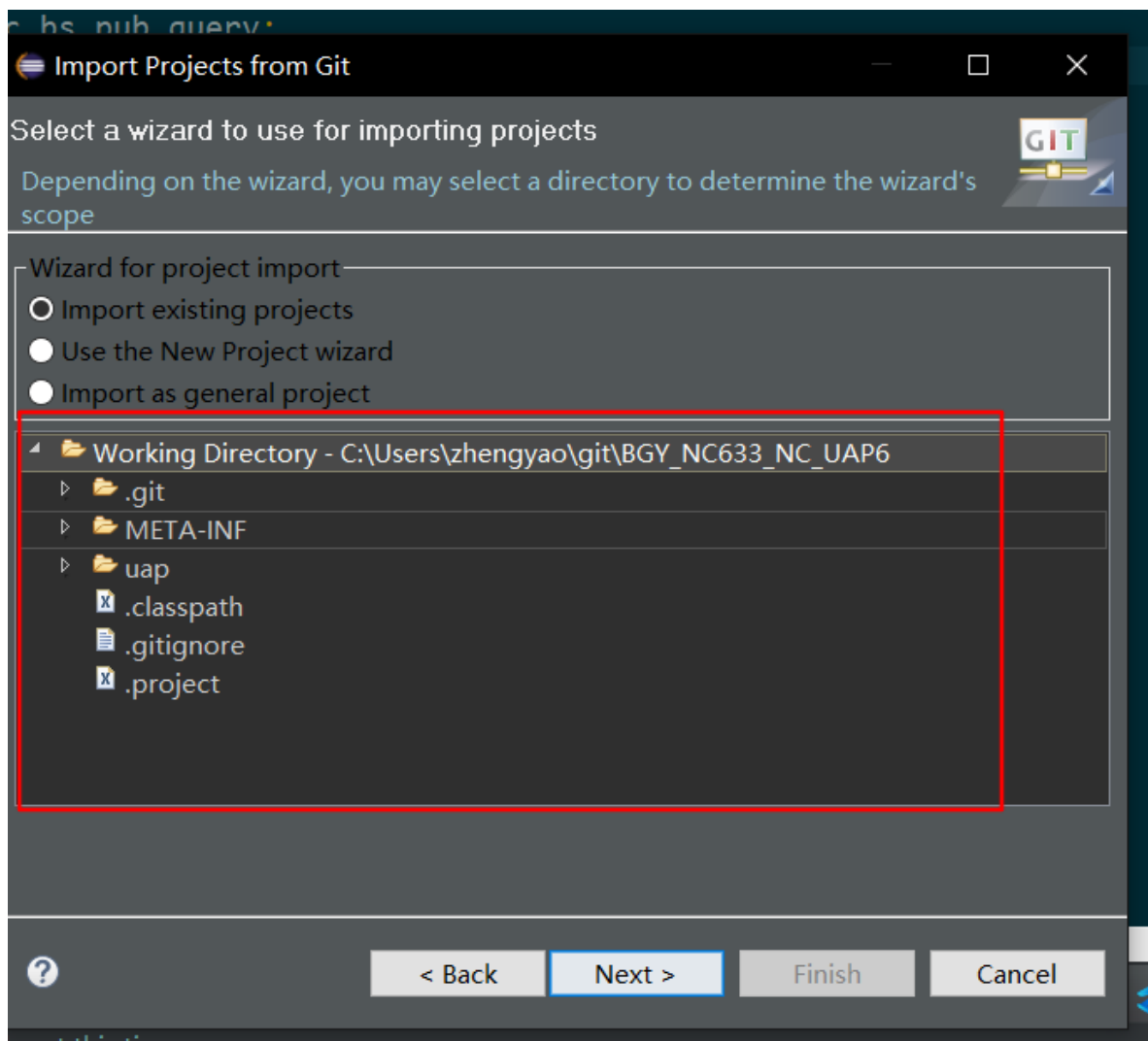


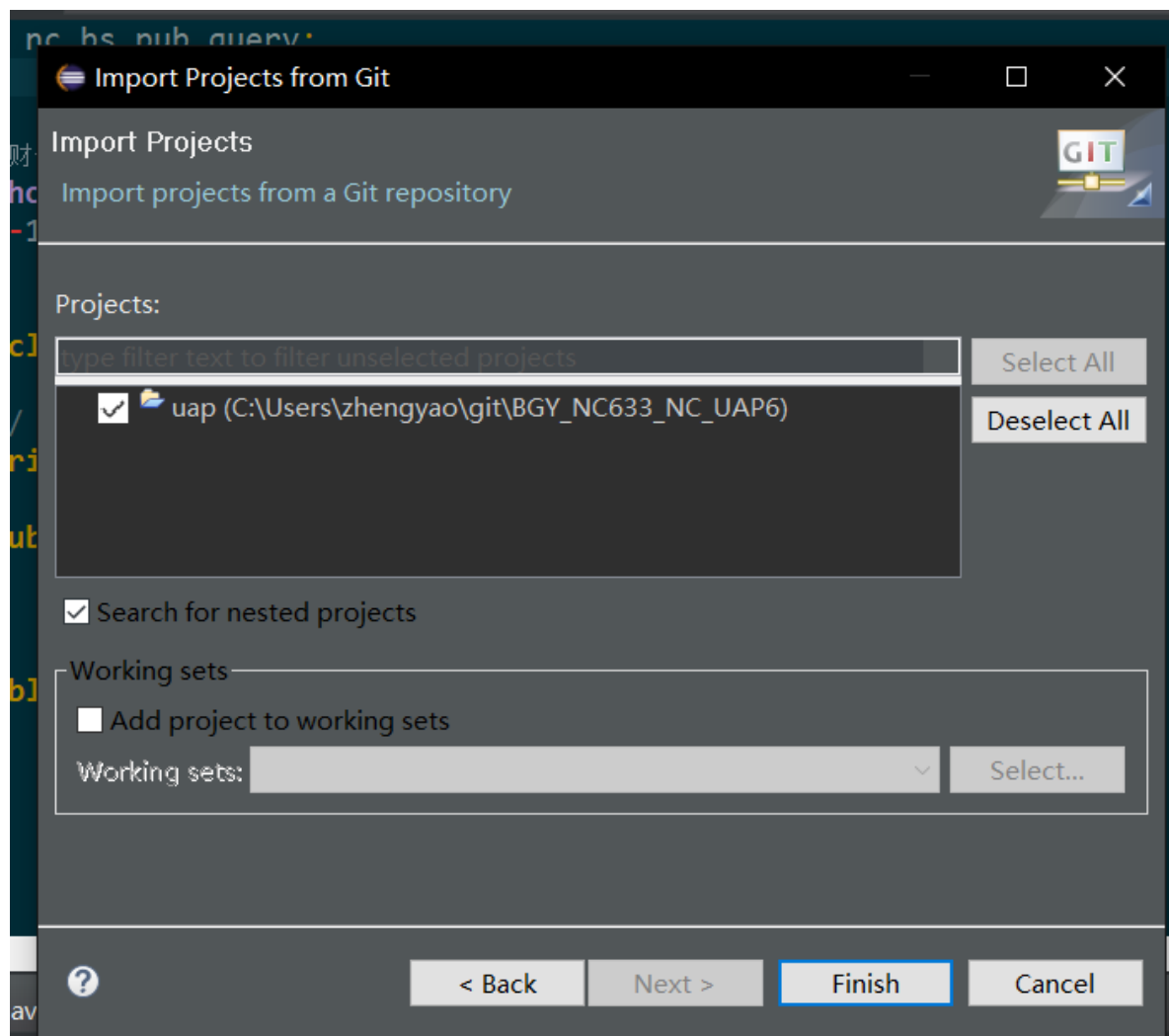
或者



选择对应路径:

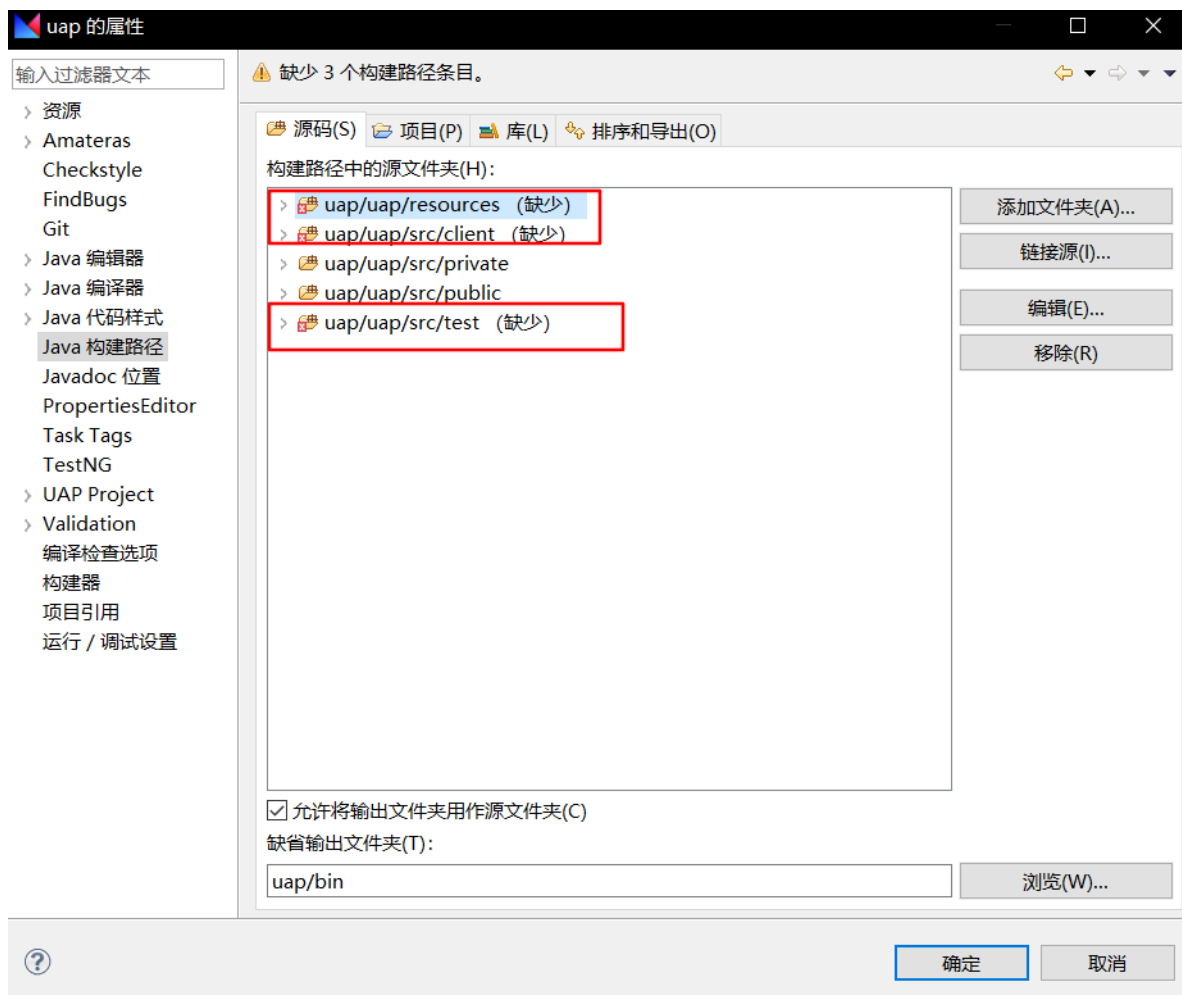






导入项目时,uapstudio导入的项目左上角会显示红色感叹号,发现是空白的文件夹未自动导入:

导入的项目有红色感叹号,查看之后还是缺少了文件夹,手工创建:



如果发现缺少对应文件夹,手工创建:

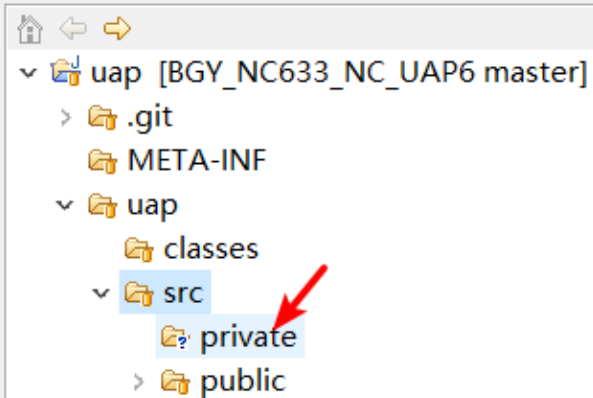
文件夹

“文件夹”名称是空的。



输入或选择父文件夹(E):

uap/uap/src



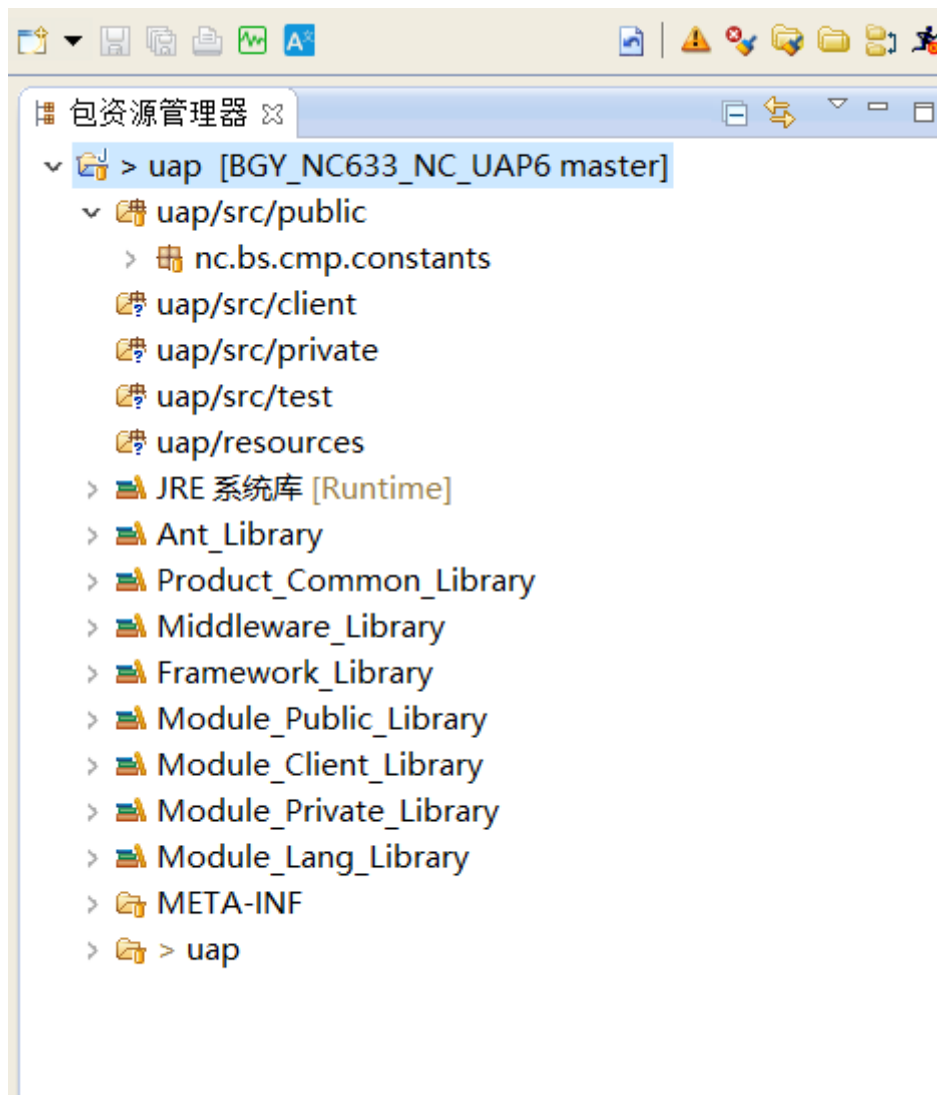
文件夹名(N):

高级(A) >>



完成(F)

取消

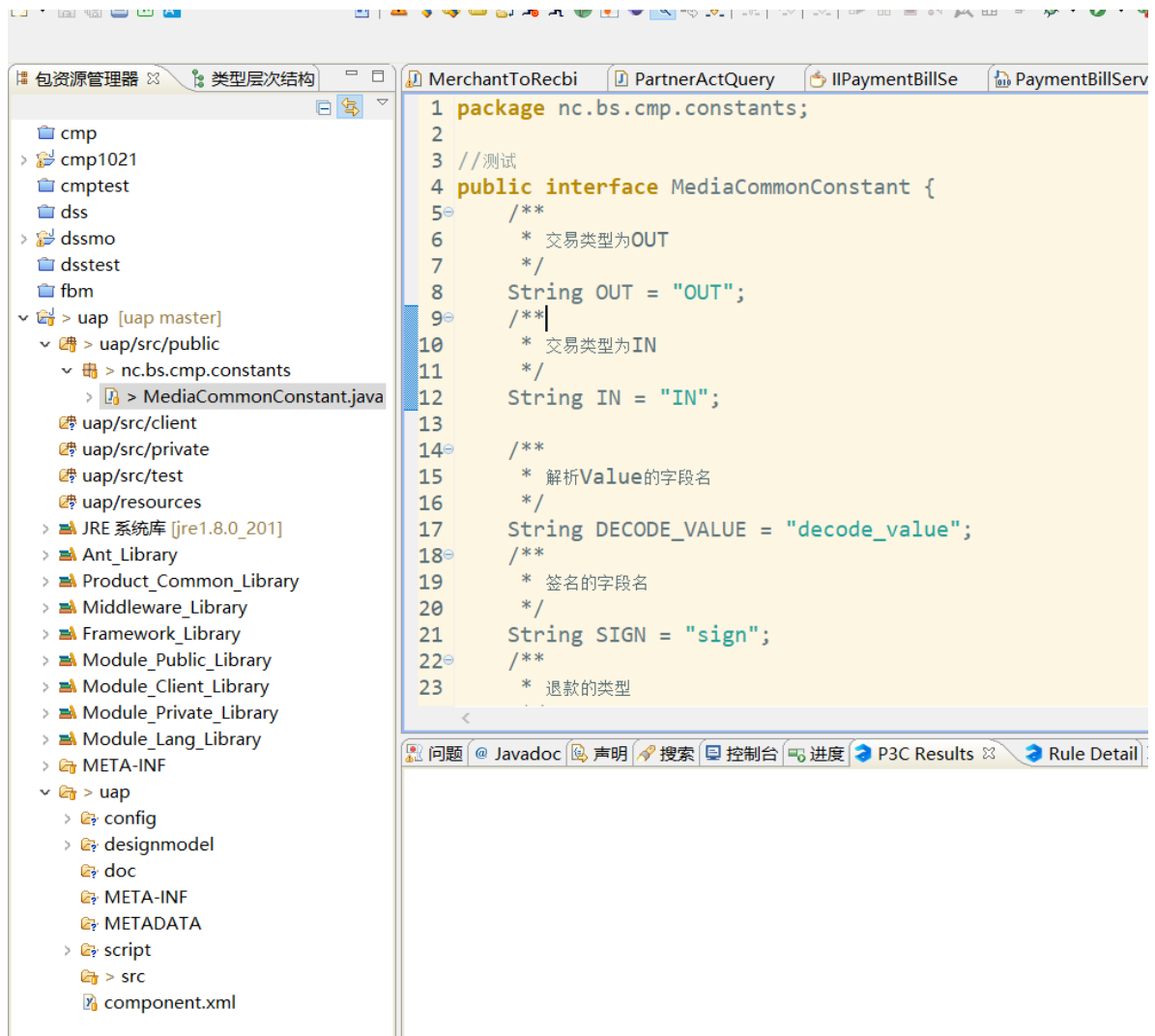


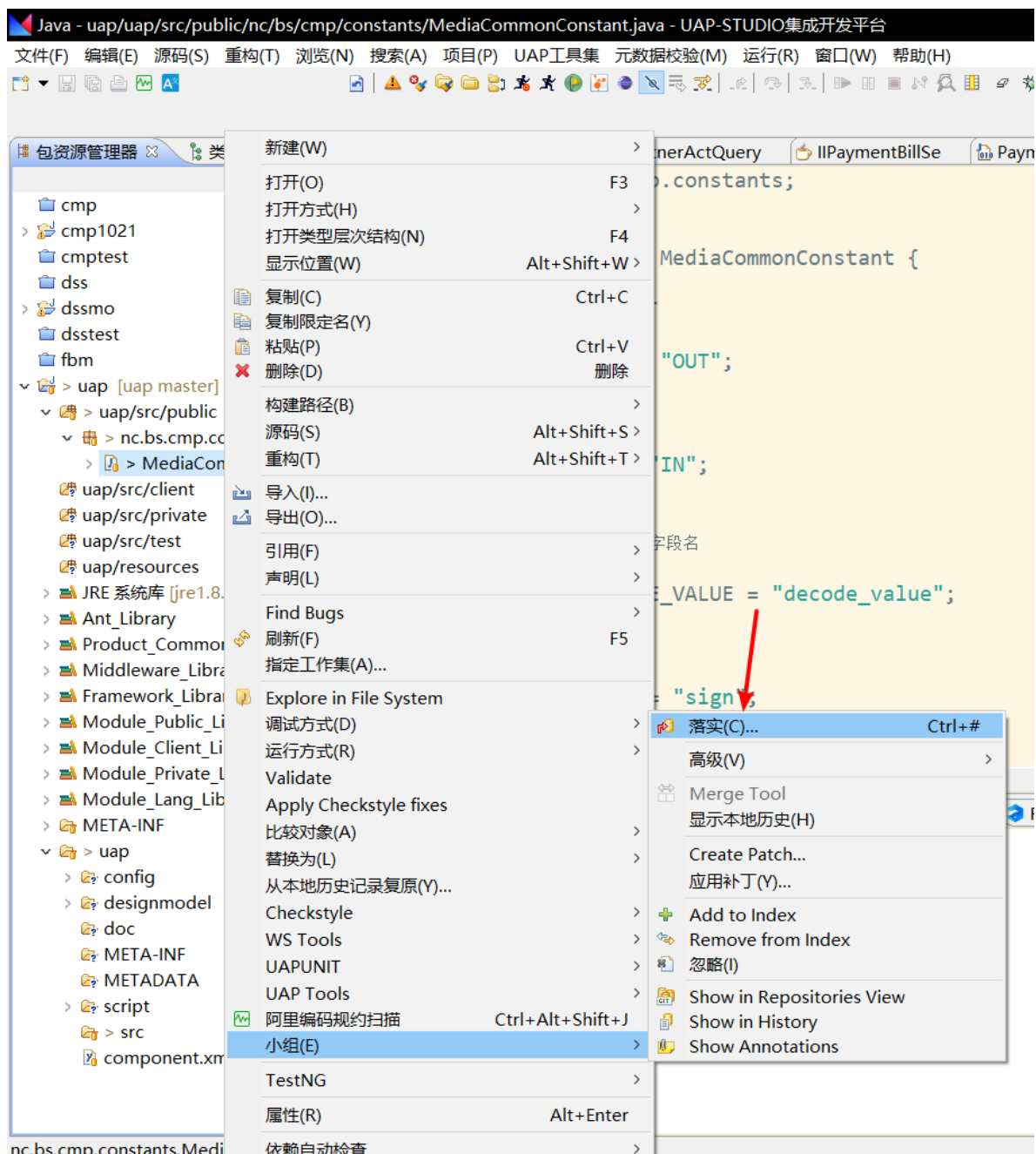
提交代码

单人提交push代码

修改文件内容后:

先进行commit到本地的仓库





commit的时候需要输入信息,是本次提交时候代码所具体完成的功能,例如修复某个bug,完成某个需求等

Commit Changes to Git Repository



落实消息

修复bug-XXXXXX, 禅道号:1134
完成需求-物业XXXXX, 禅道号:

作者(A): zhengyao3 <zhengyao3@yonyou.com>

Committer: zhengyao3 <zhengyao3@yonyou.com>

Files (1/1)

输入过滤器文本

状态	路径
<input checked="" type="checkbox"/>	uap/src/public/nc/bs/cmp/constants/MediaCommonConstant.java

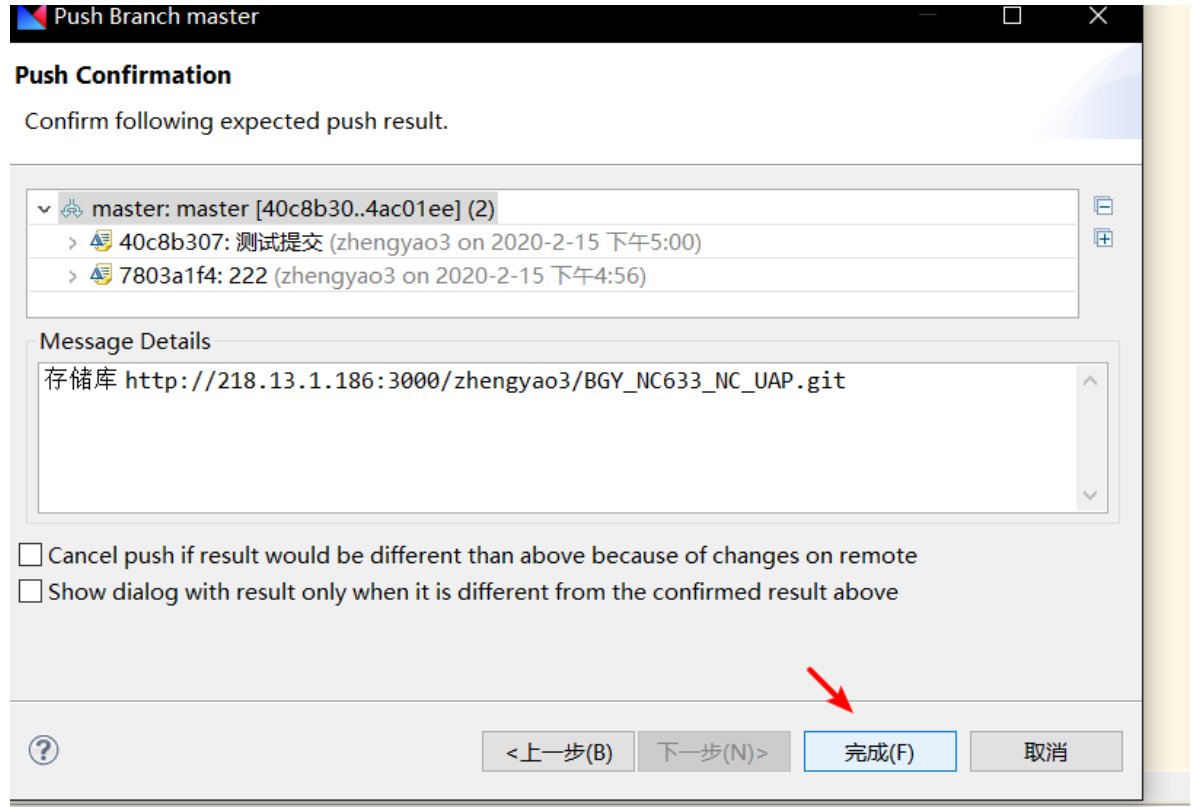
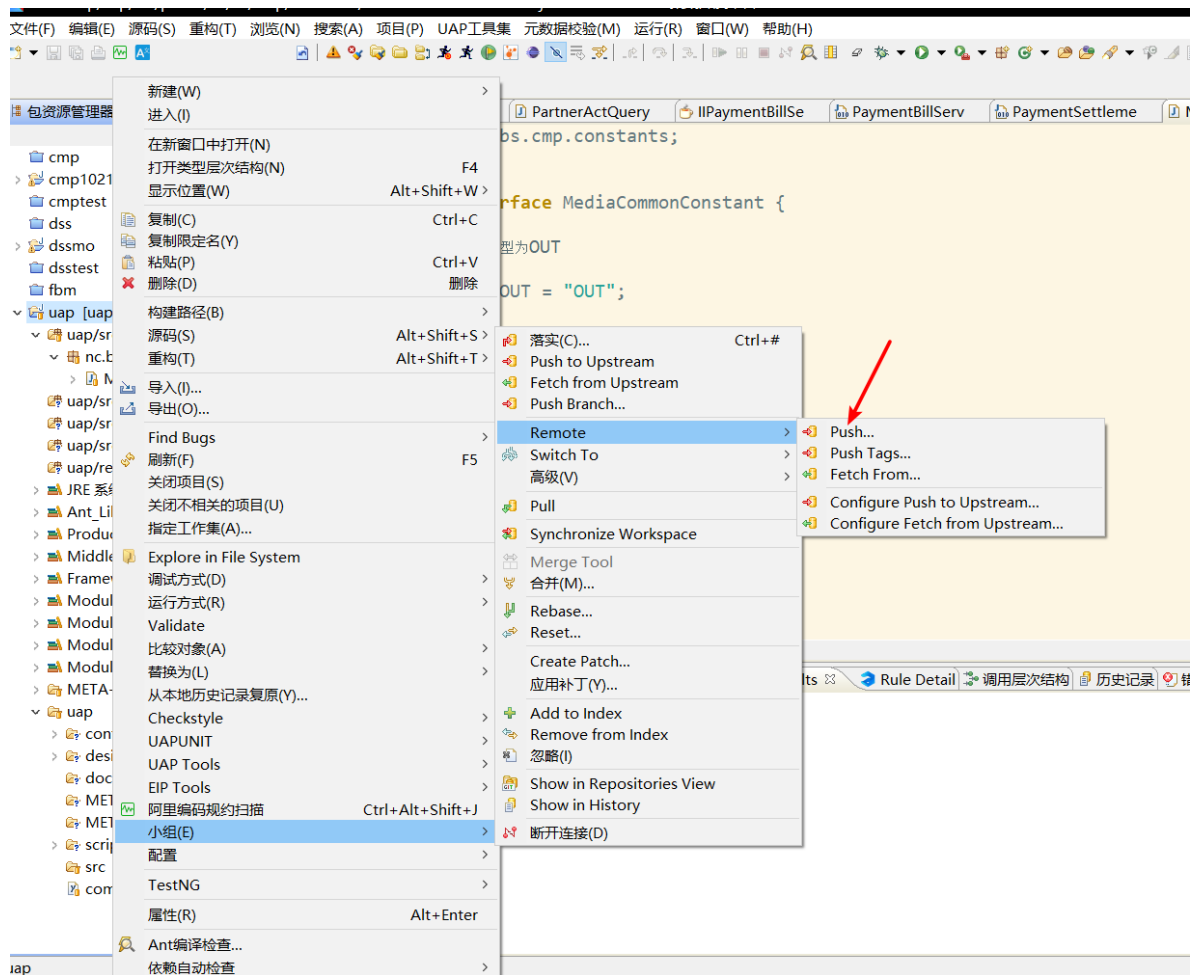


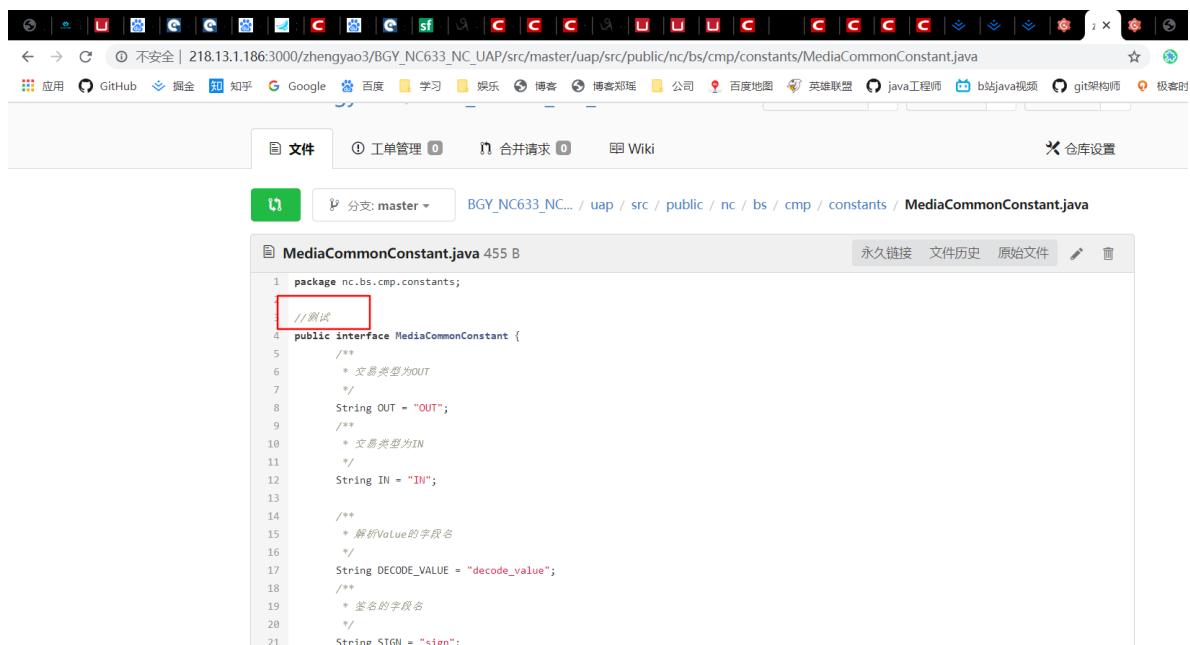
Commit and Push

Commit

取消

commit之后进行push:

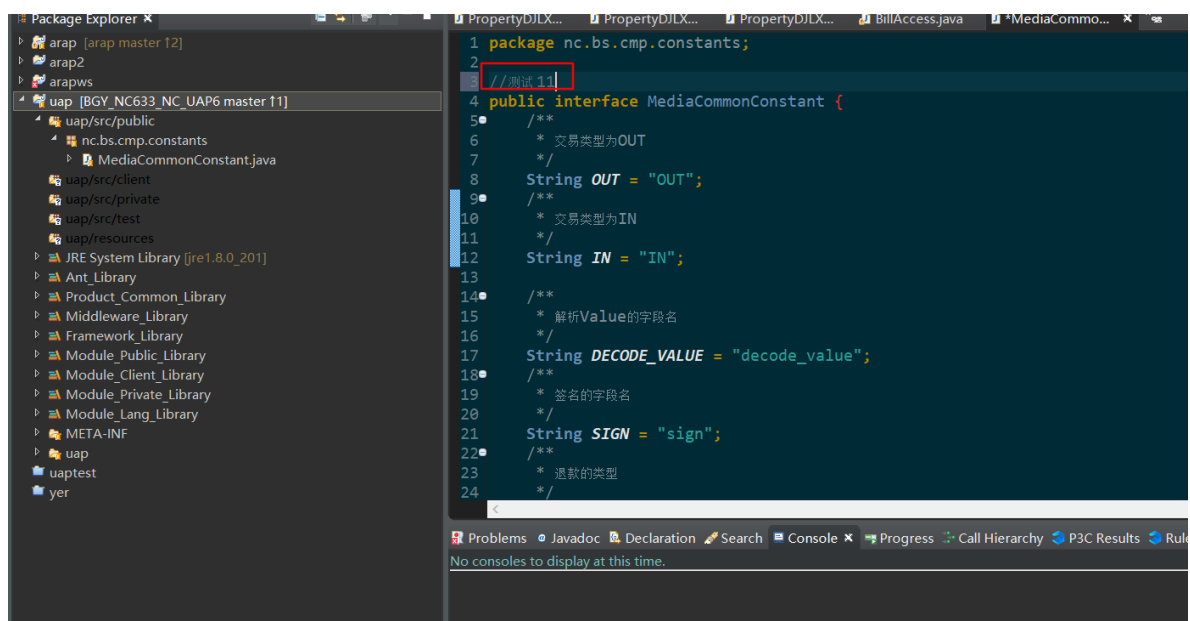




多人协作更新代码

改了相同环境下位置的代码

此时本地环境代码是://测试11

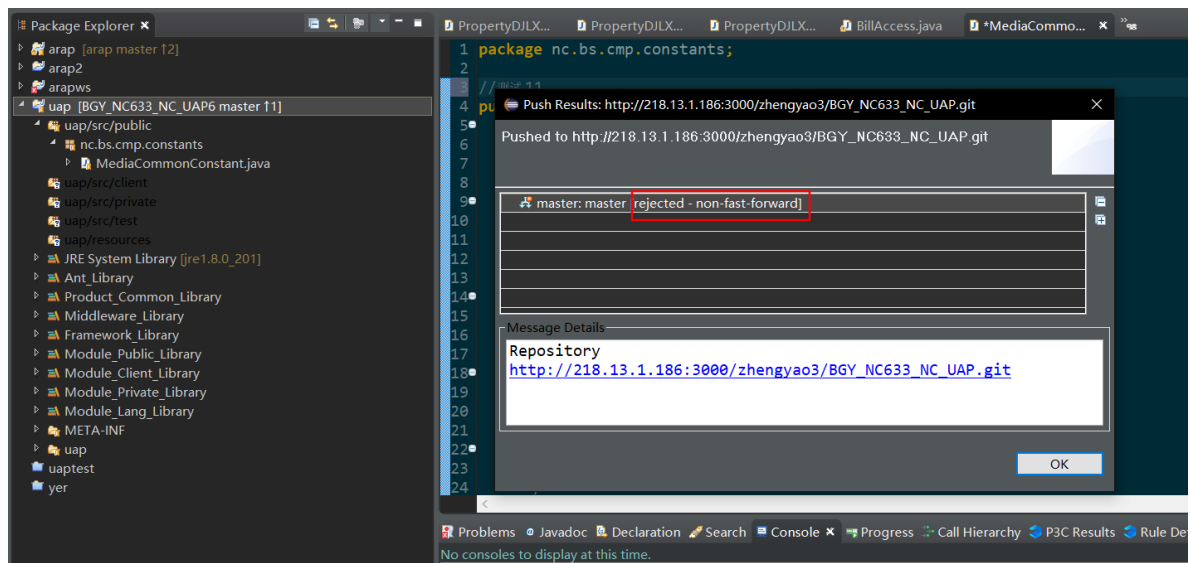


服务器的内容为:

```
1 package nc.bs.cmp.constants;
2
3 //测试 11 22
4 public interface MediaCommonConstant {
5     /**
6      * 交易类型为OUT
7      */
8     String OUT = "OUT";
9     /**
10      * 交易类型为IN
11      */
12     String IN = "IN";
13
14     /**
15      * 解析Value的字段名
16      */
17     String DECODE_VALUE = "decode_value";
18     /**
19      * 签名的字段名
20      */
21     String SIGN = "sign";
22     /**
23      * 退款的类型
24      */
25     String REFUND_TYPE = "5";
26 }
```

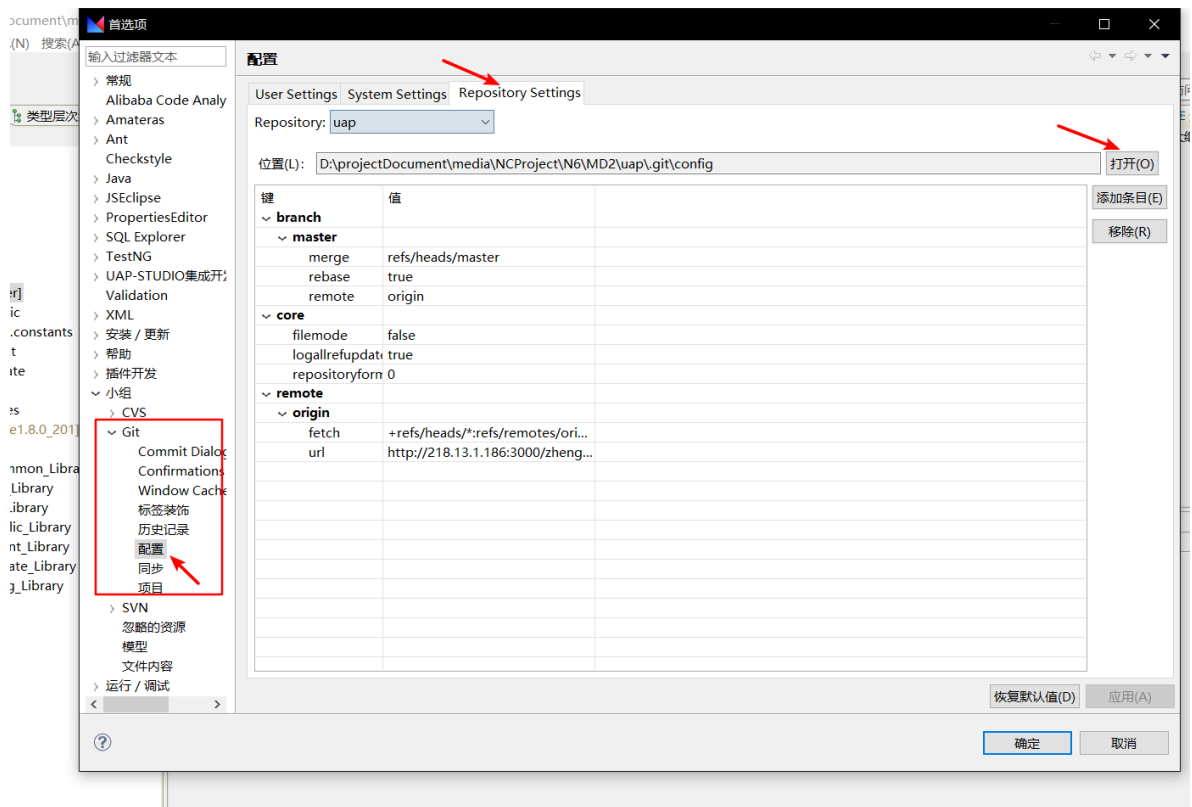
前面commit的步骤不变

此时git push代码到远程仓库:

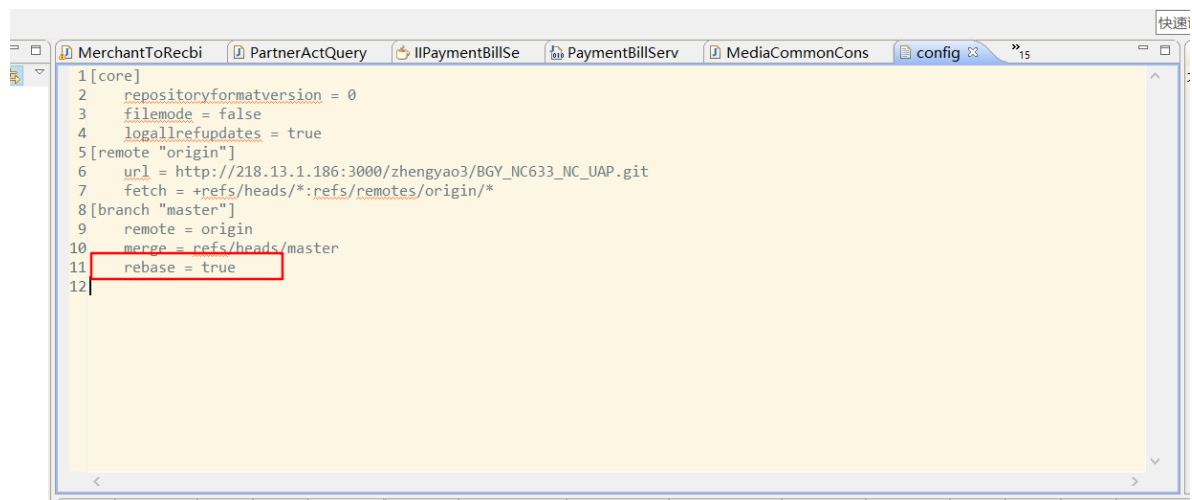


可以看到push失败了,因为本地仓库的最新的指针和远程仓库的版本对应不上,此时需要先进行pull:

首先配置pull的默认更新方式为rebase:



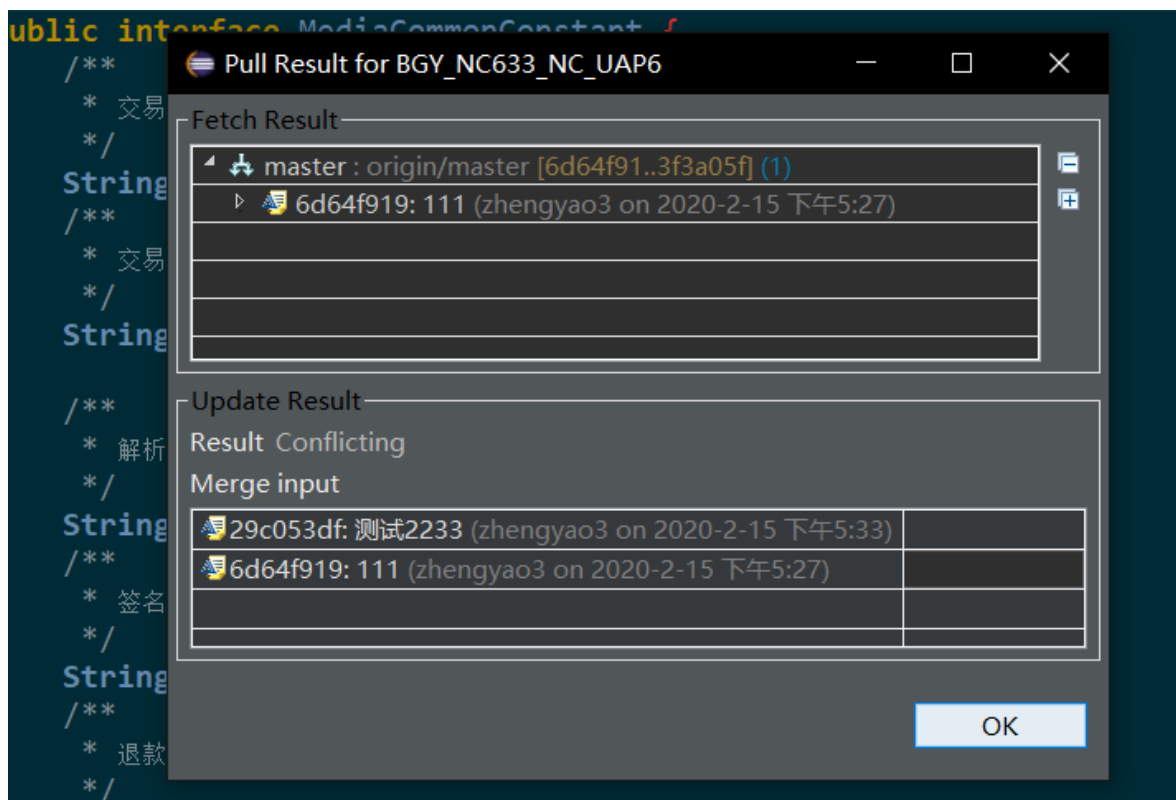
增加一行: rebase=true



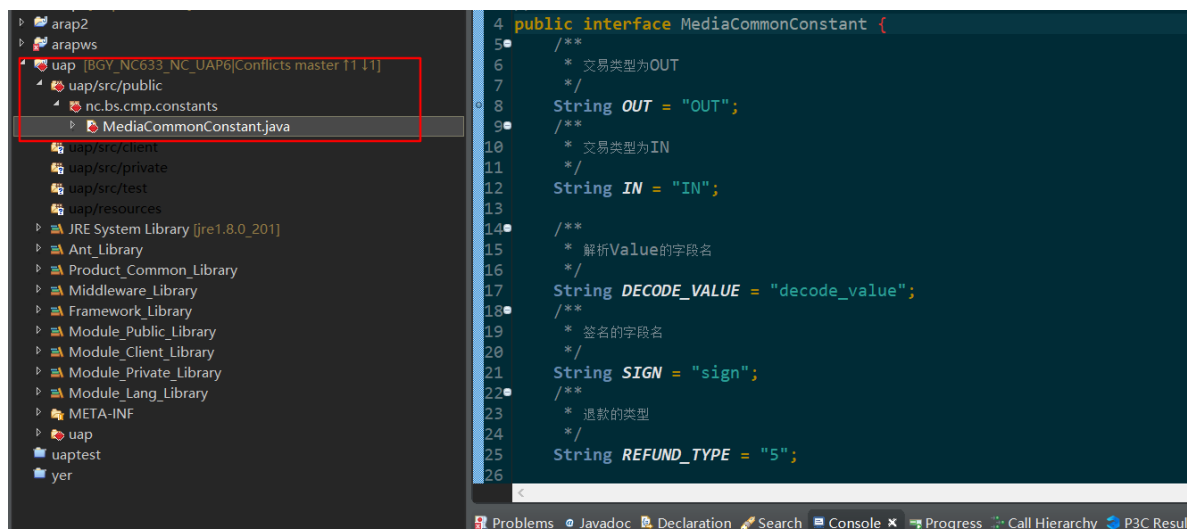
ps:为什么要使用rebase而不是默认的merge?

在开发过程中使用git rebase还是git merge, 优缺点分别是什么? - 王靖轩的回答 - 知乎
<https://www.zhihu.com/question/36509119/answer/131513261>

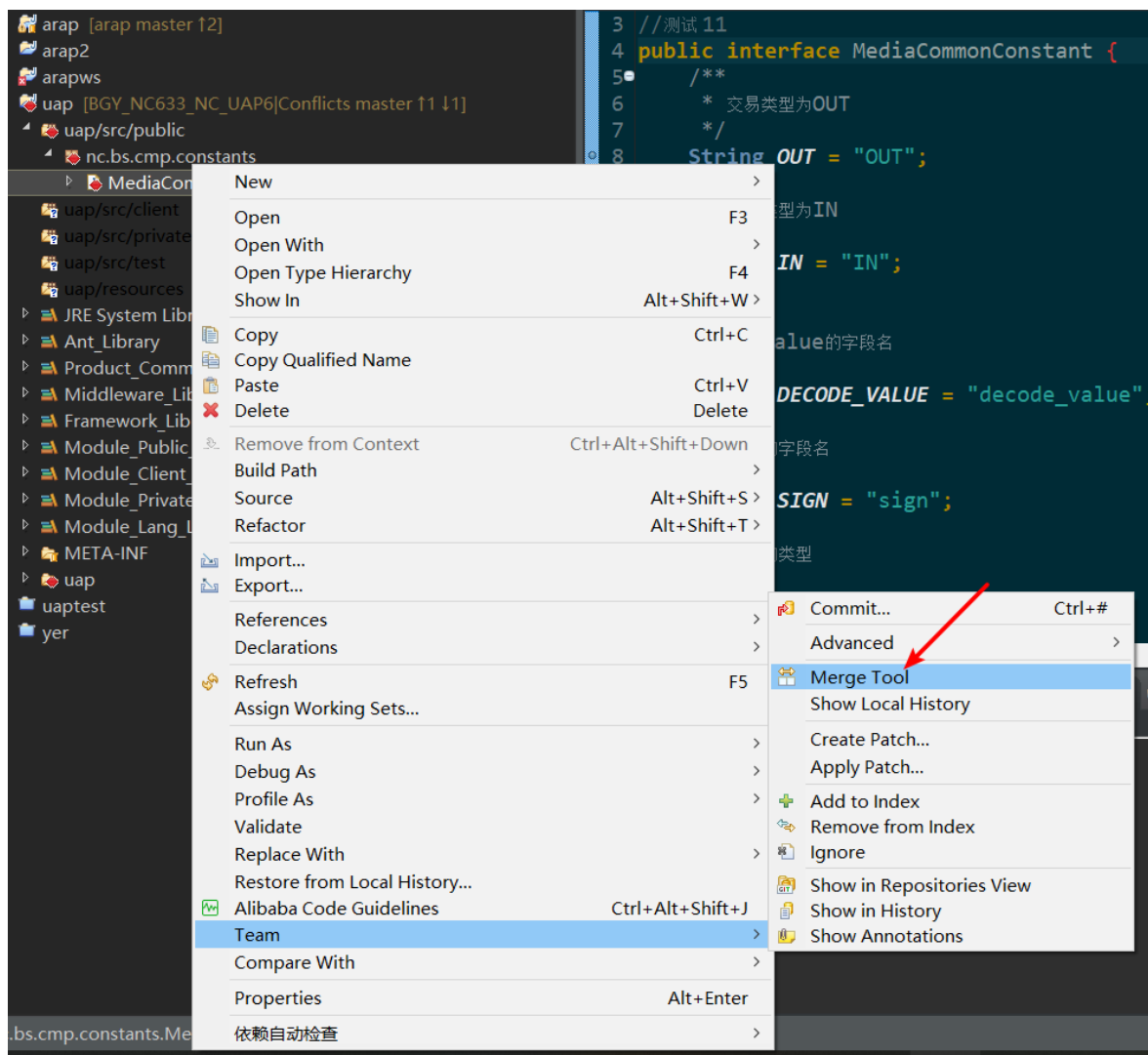
如果大家修改的地方不一样的话,git会自动进行合并,只需要单纯的pull更新代码即可,但是如何修改的是相同区域,git需要人工的解决冲突,例如本次



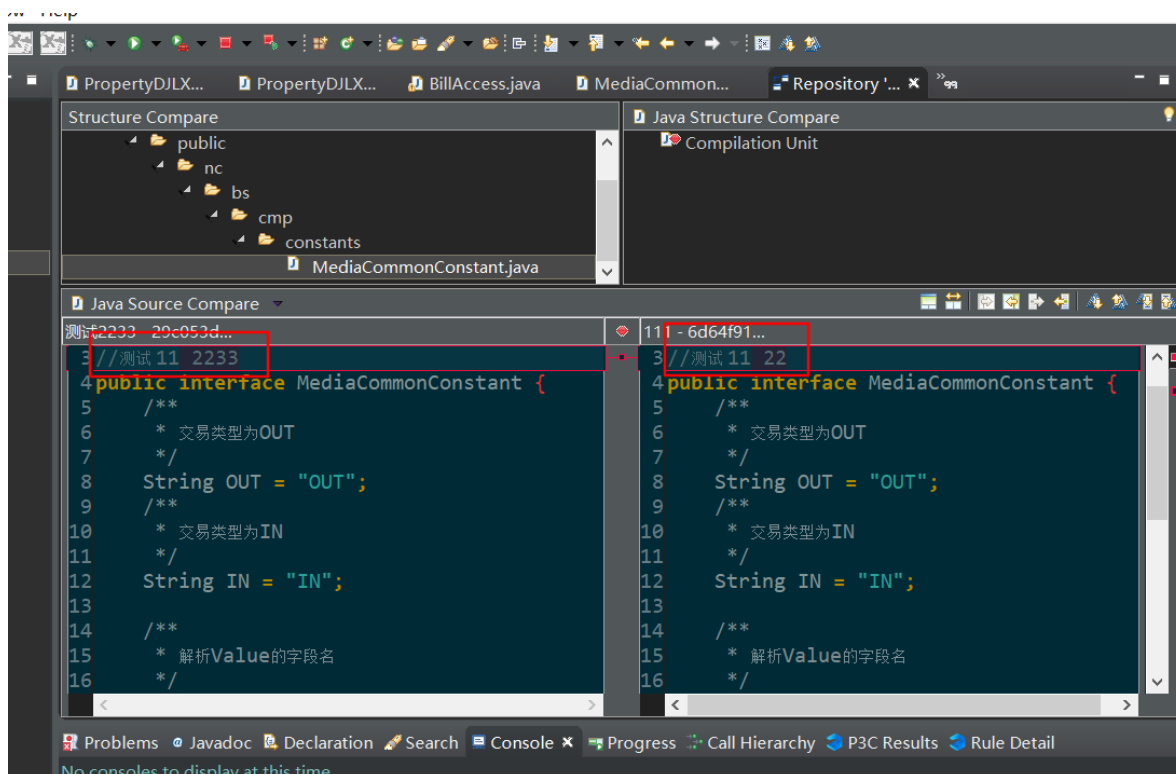
发生冲突后,文件会有提示:



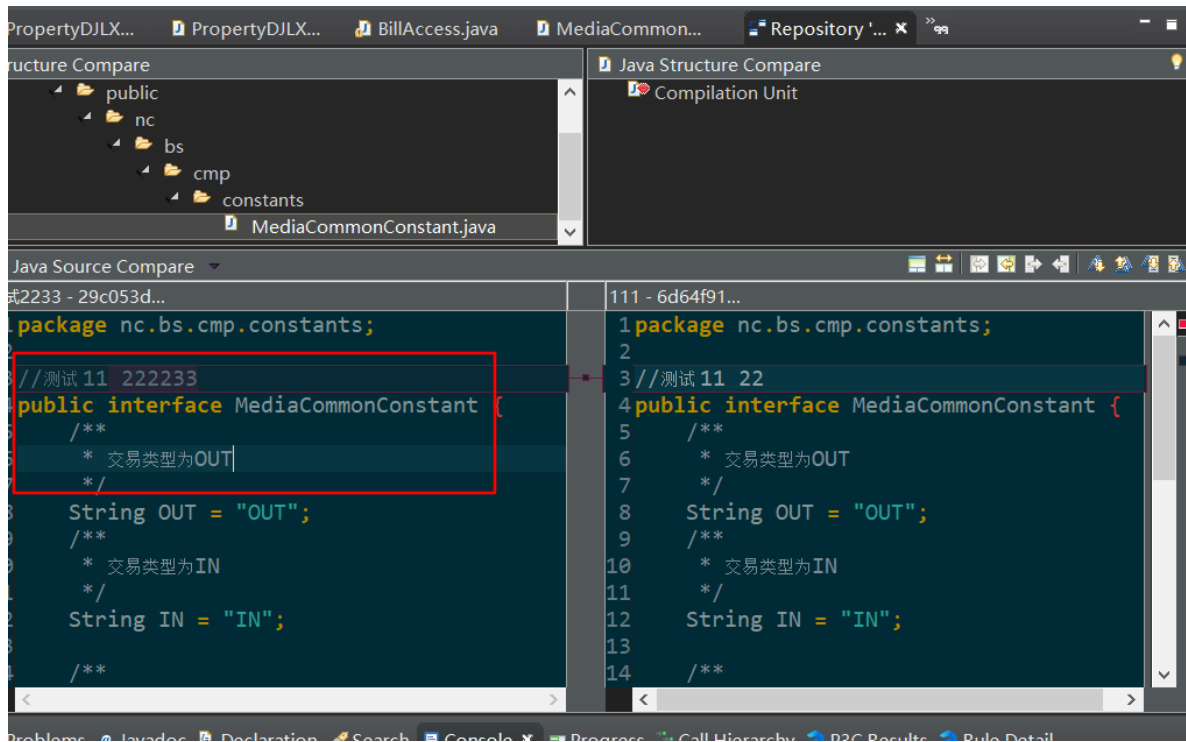
此时



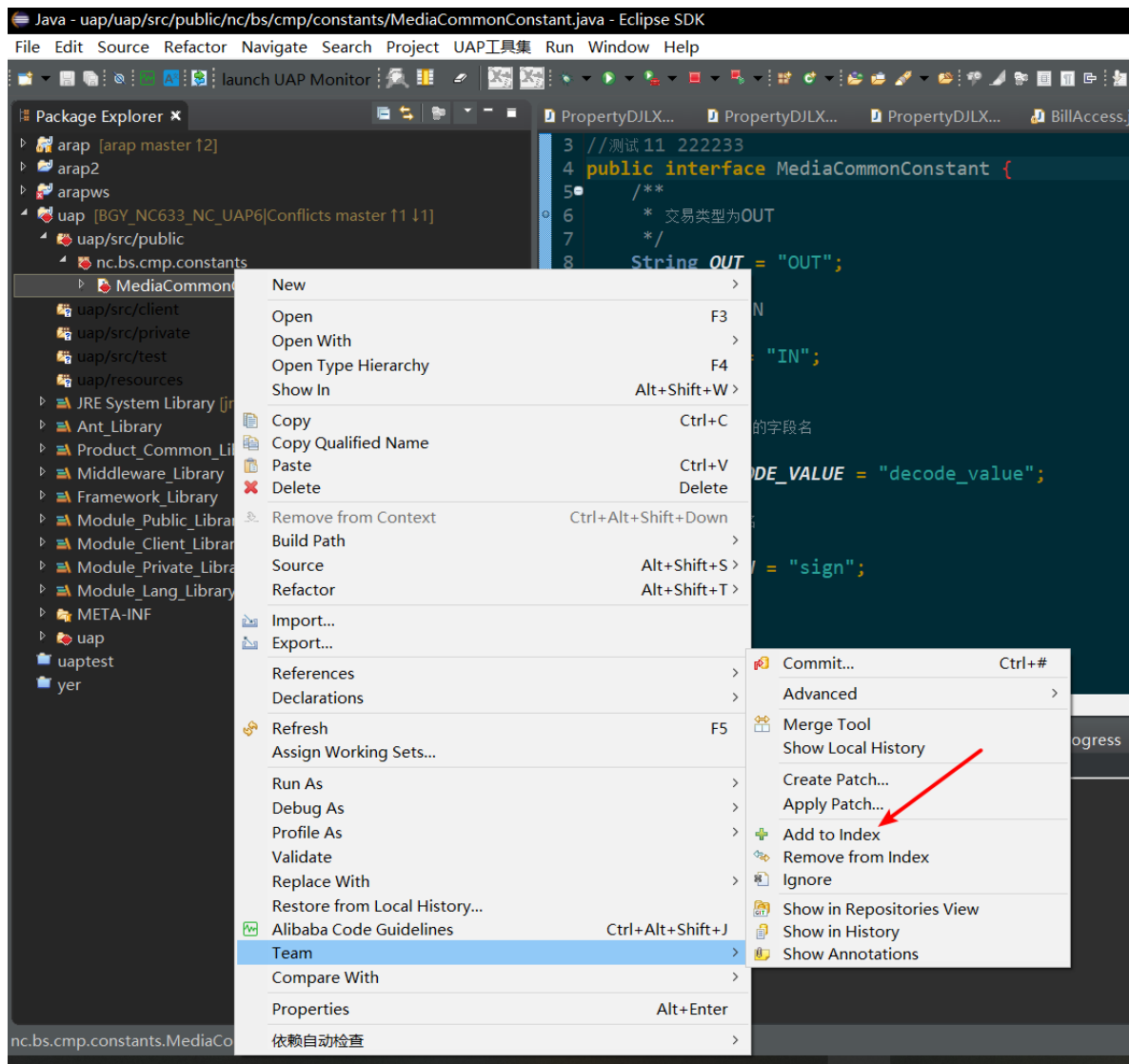
可以看到本地文件和远程仓库的文件产生了冲突,此时手工合并或者咨询对应开发看如何合并代码:

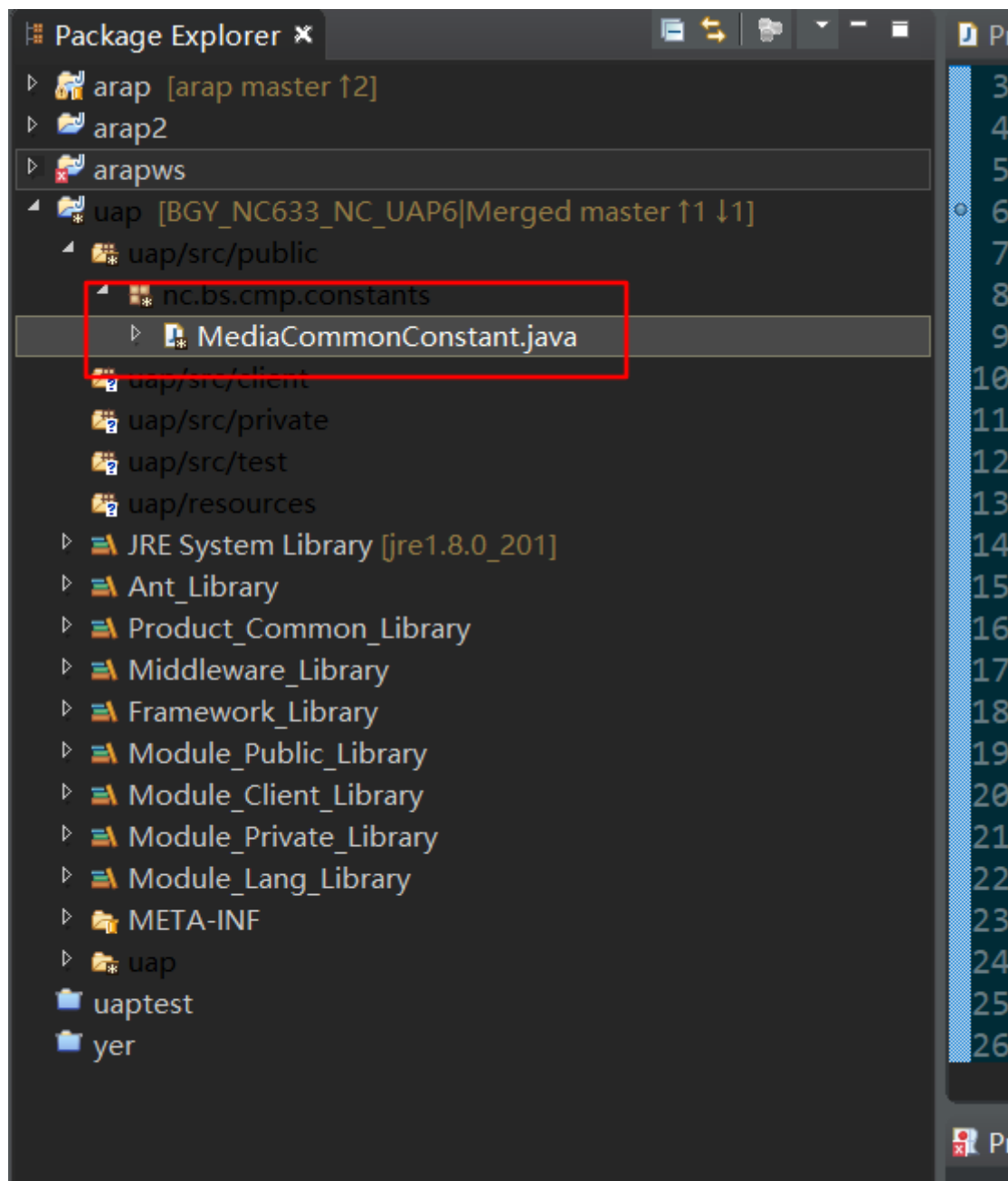


我这里将他修改为:

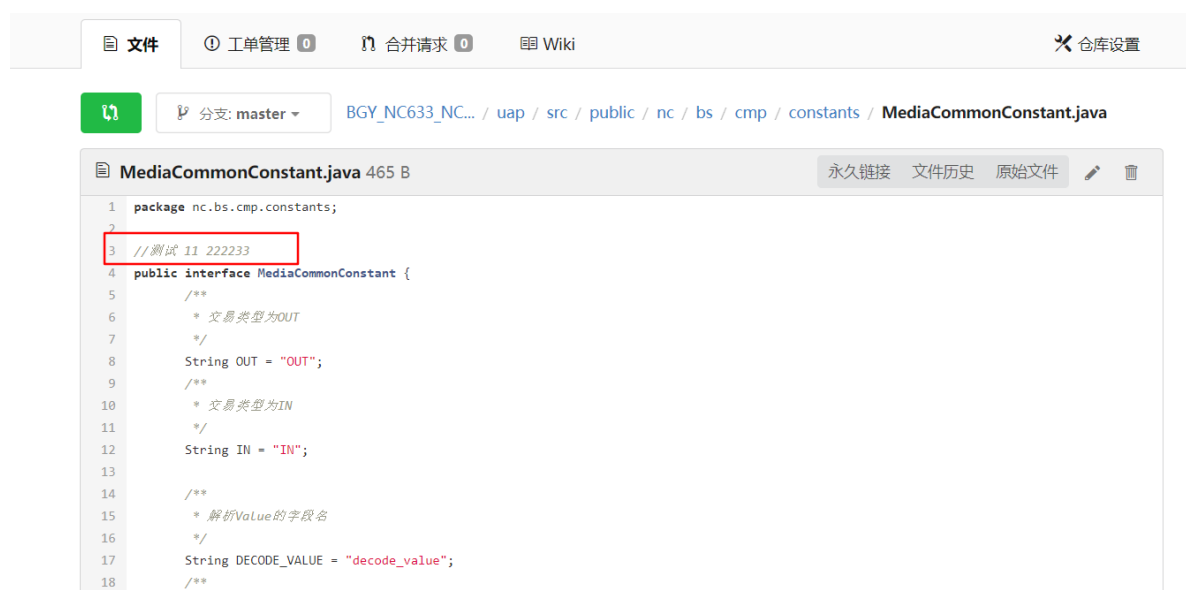


此时在点击:





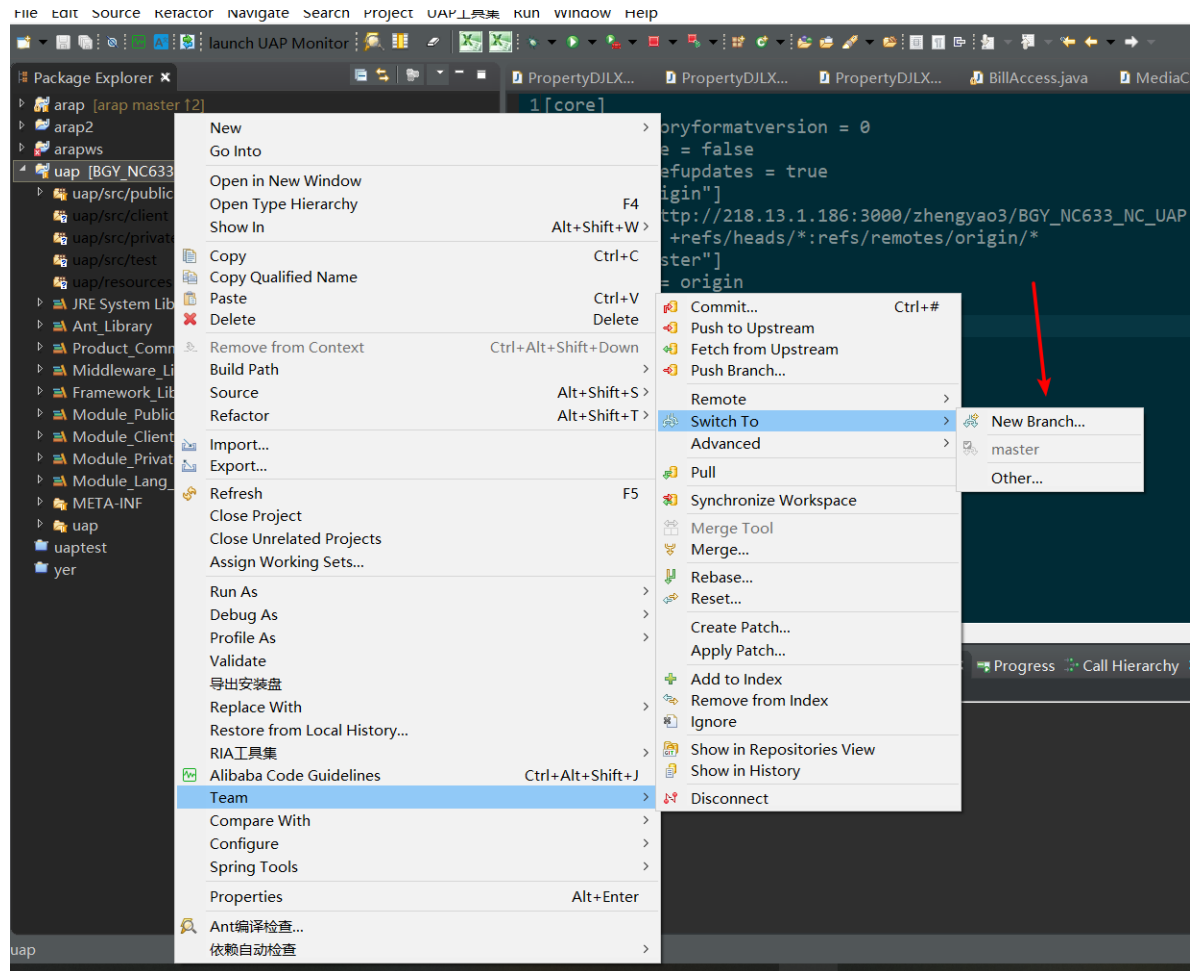
此时再重复commit-pull-push的步骤就可以正常提交代码了,可以看到远程仓库代码已经是最新的了:



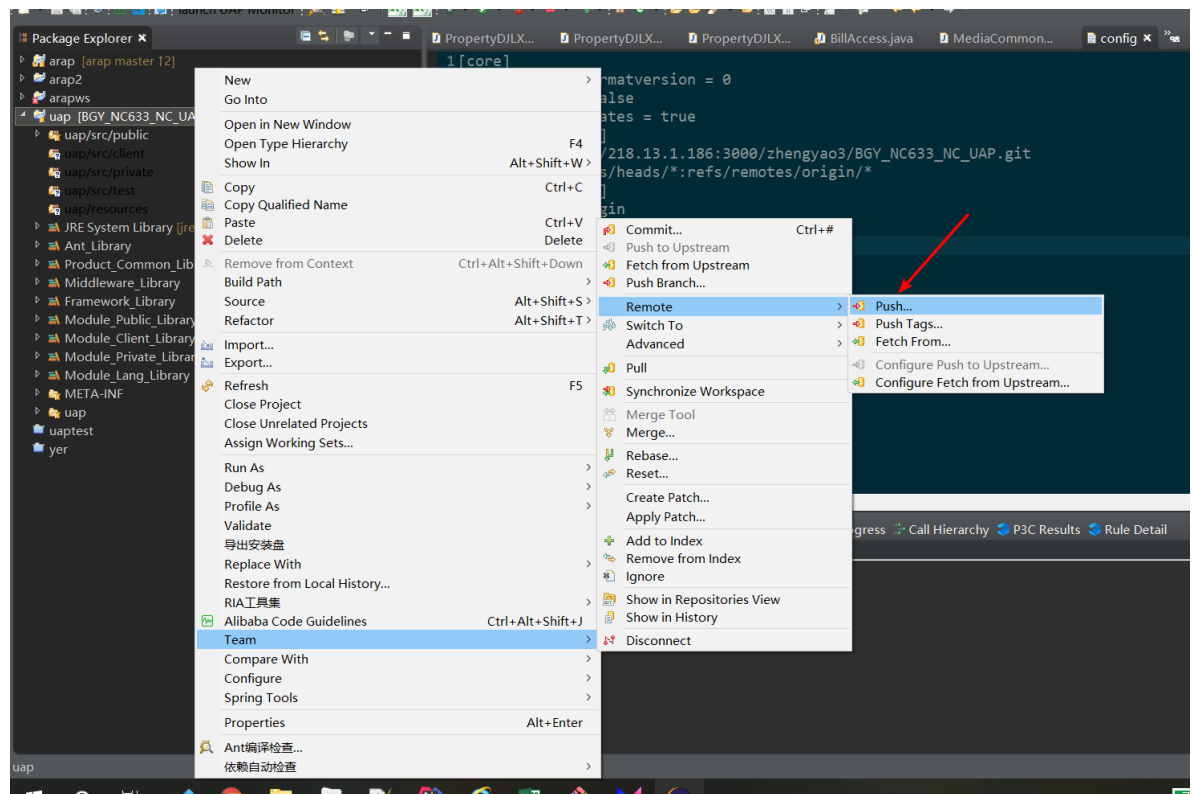
新建与切换分支

新建分支

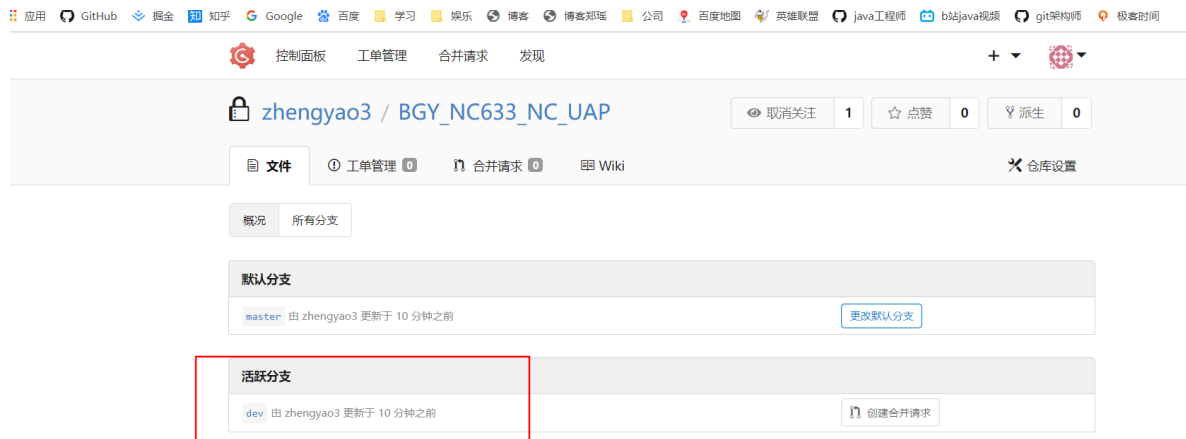
先新建一个分支:



将新分支推送到远程仓库:

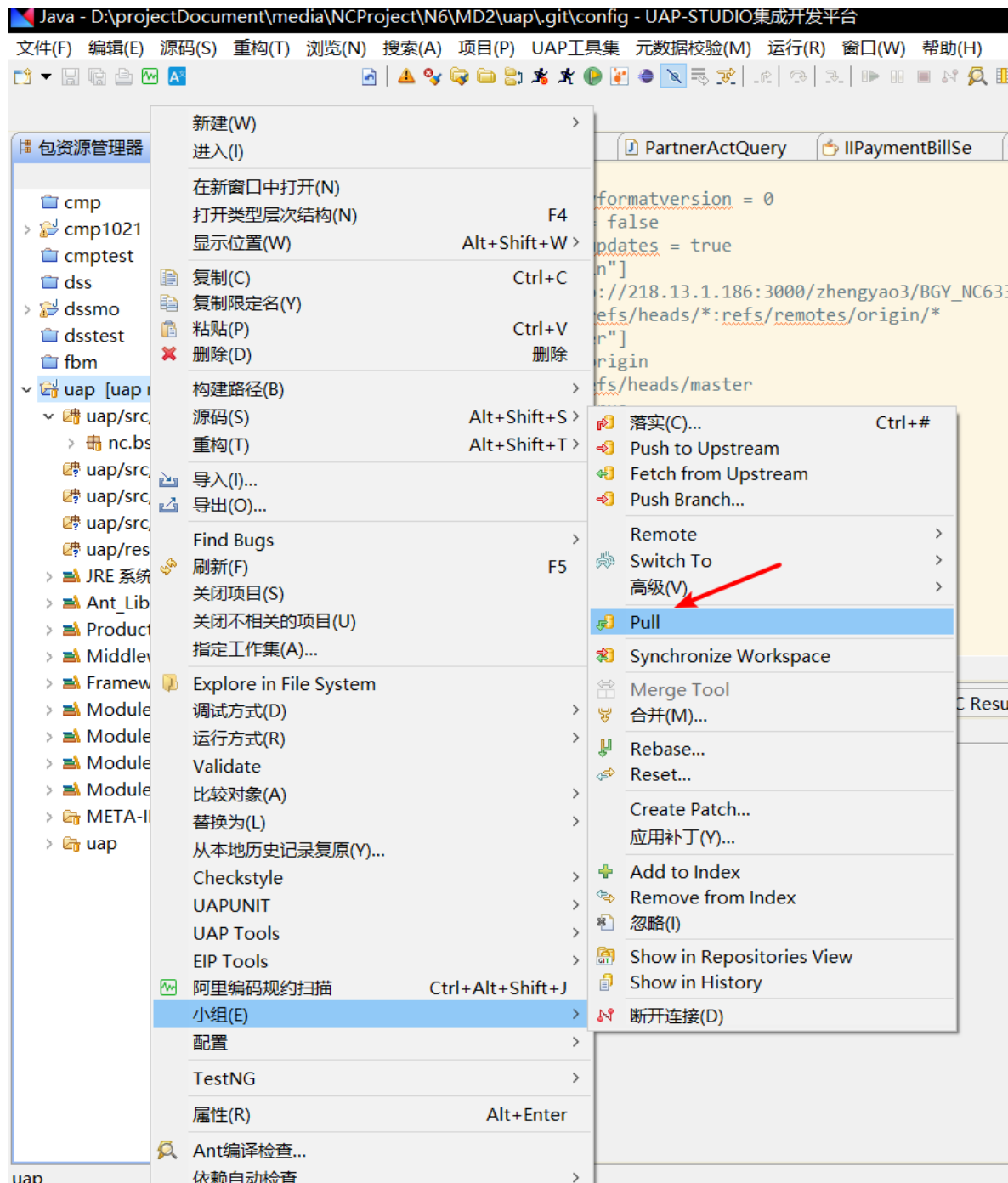


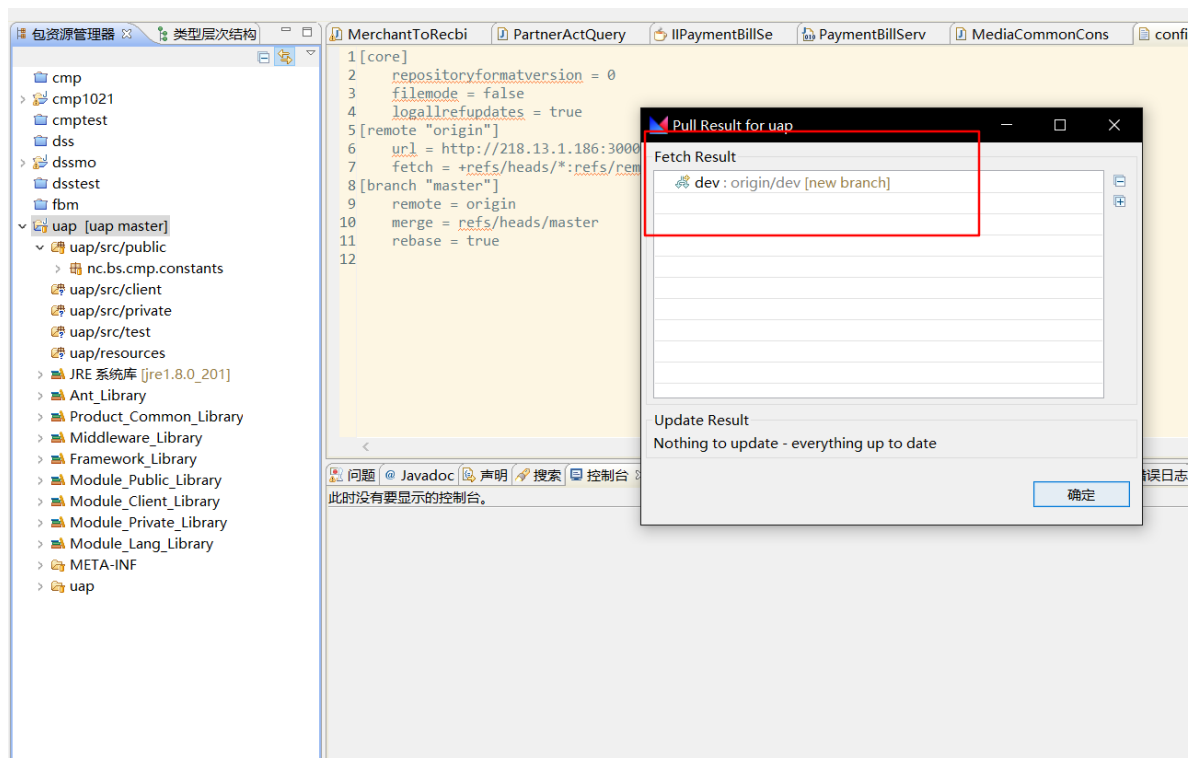
可以看到远程仓库下已经有了新的分支了:



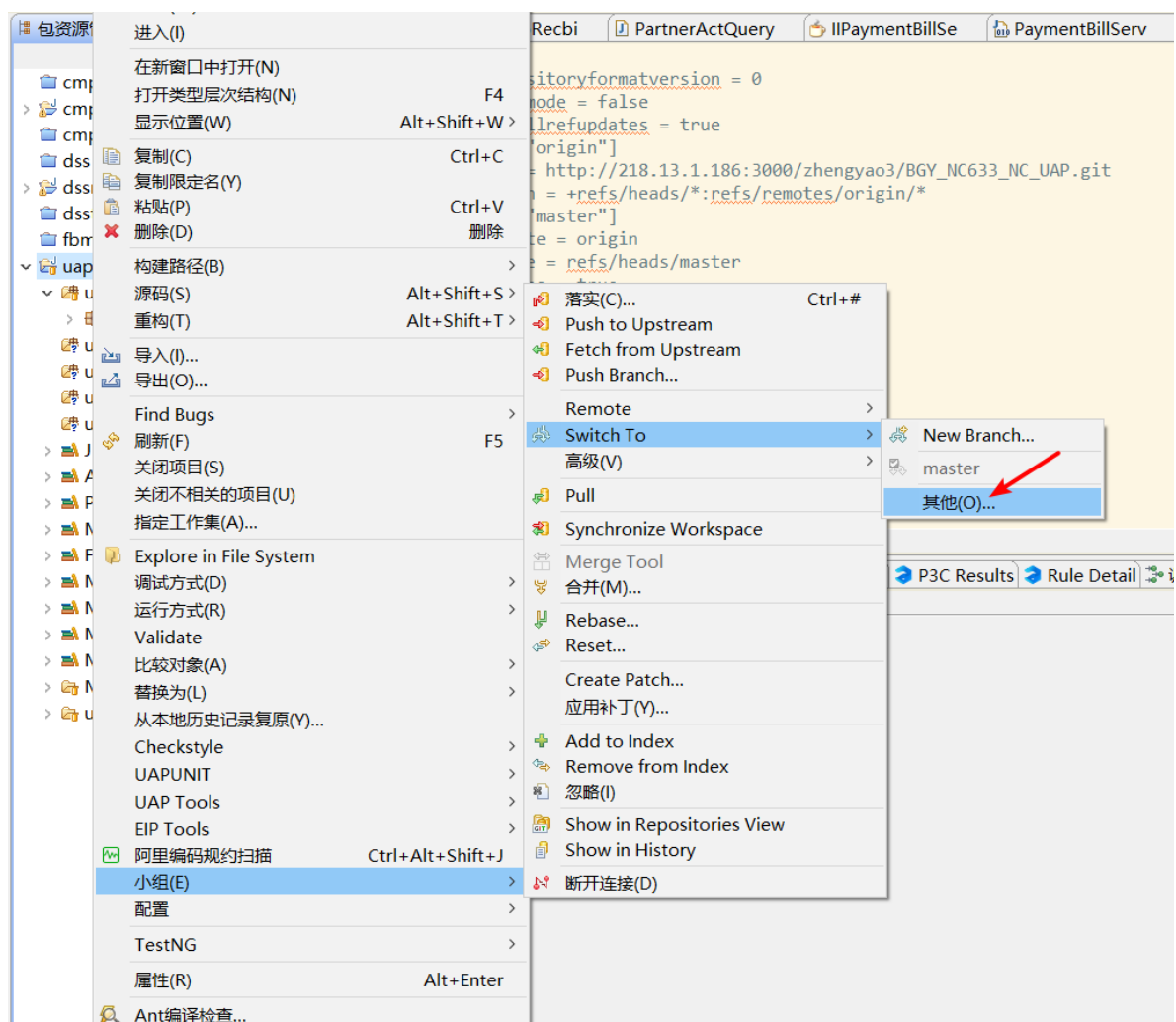
切换分支

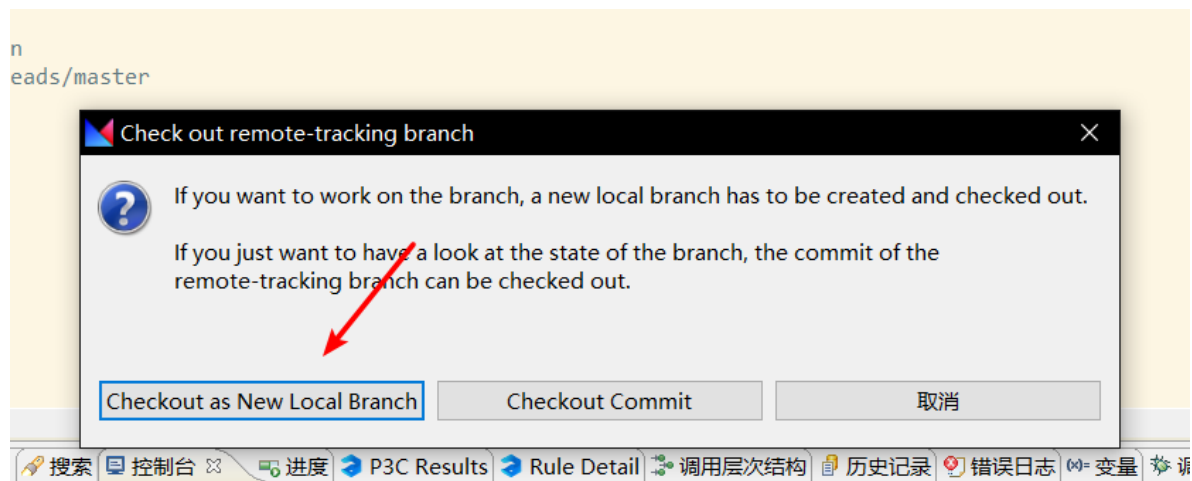
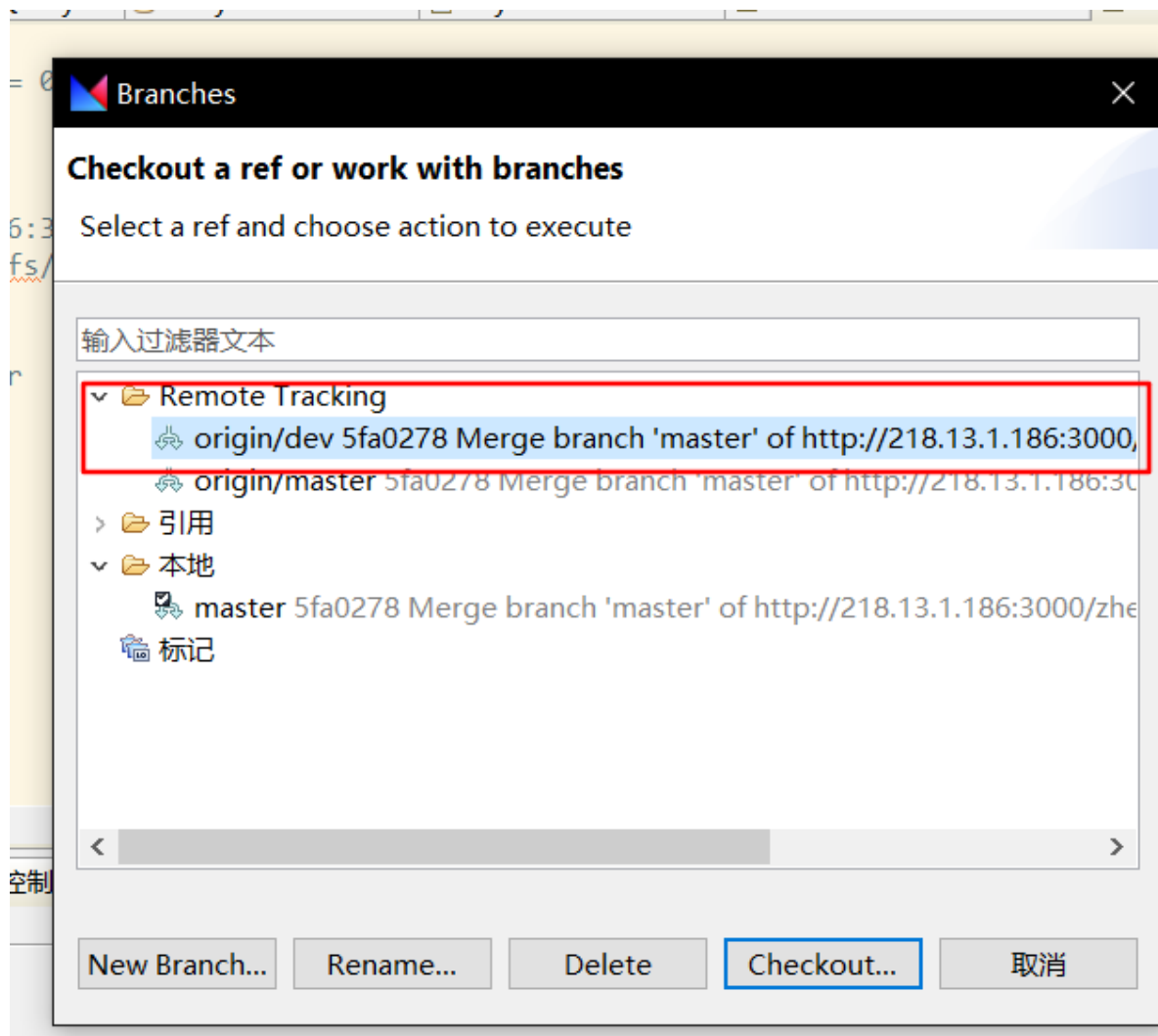
首先进行pull:

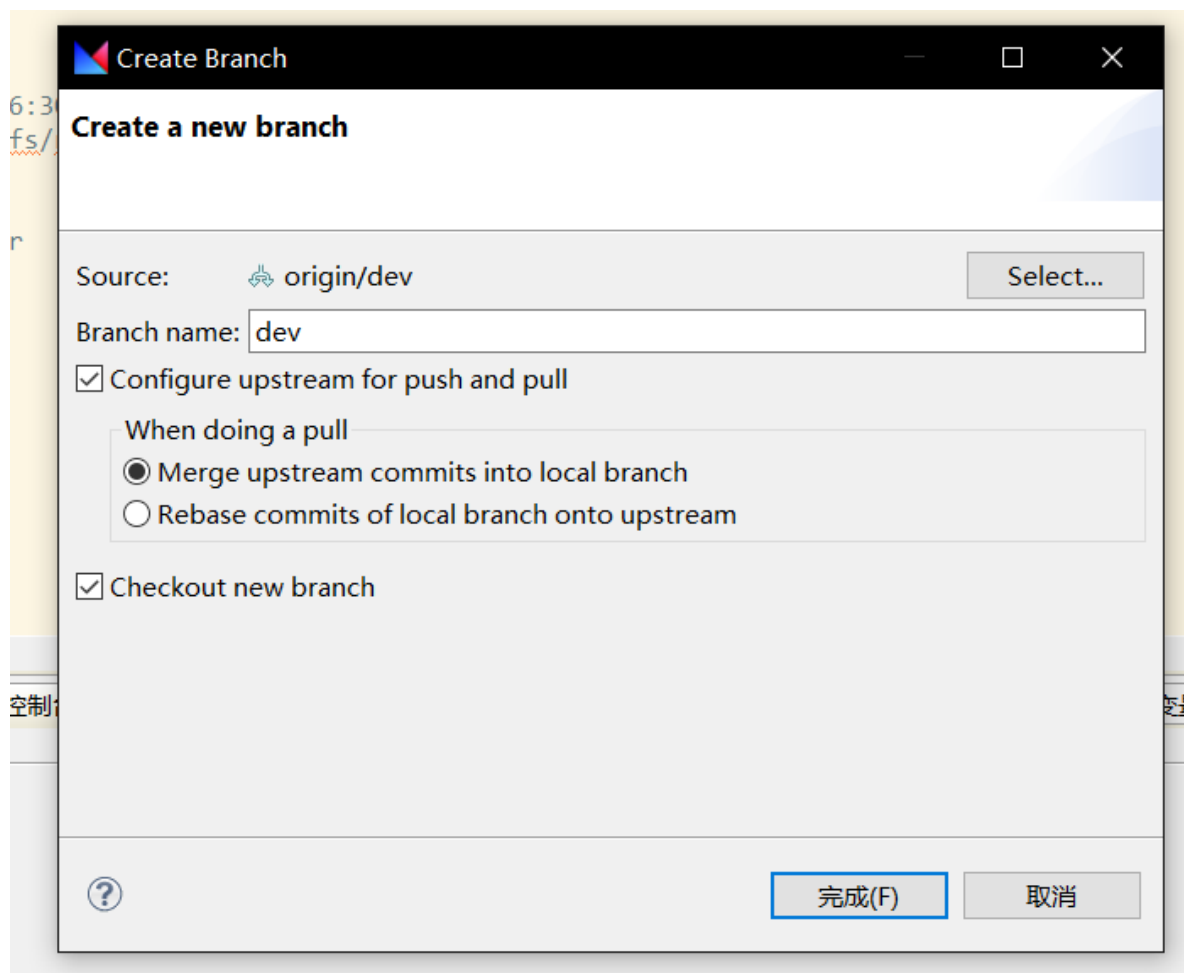




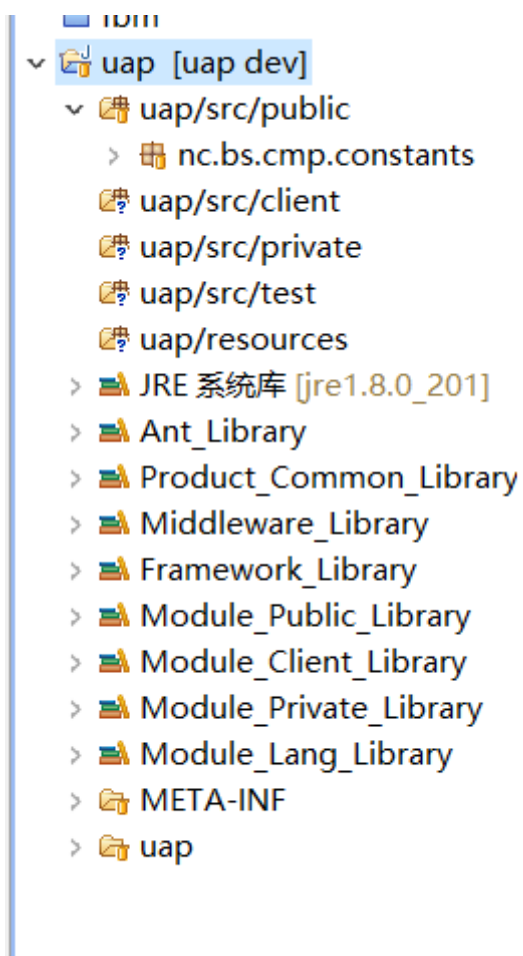
然后选择远程的分支拉取下来:







切换分支之后,项目上当前分支就变了:



多环境下开发流程

简单一点的开发步骤

项目拥有master分支和dev分支, 所有人往 dev 合并代码并出具补丁, 测试顾问进行测试, 测试通过直接把 dev 中测试通过的需求代码(考虑是否review)合并到 master 进行升级

复杂一点的开发步骤

dev 分支就是测试环境

这个分支, 不需要有权限控制, 研发根据 禅道 或者需求文档上的任务, 从 master `check` 出一份分支

以 禅道 bug <http://xx.104.96.233:60888/zentao/bug-view-11568.html> 为例

- 我们为分支命名为: fix-bug-11568
- 修复 BUG
- commit 代码写好 msg:fix xxx , merge 到 dev 「**fix-bug-11568** 分支此时不要删除」
- 通知相关人士进行测试
- 测试失败「打回重新提交」, 测试成功点击「关闭」通知研发可以了。
- 如有多个测试环境,此时将分支继续merge到其他测试环境,例如merge到联测环境
- 重复上述测试步骤,完全测试通过后
- 研发发起申请(考虑是否review)后由管理员或者自己合并.并出具生产环境补丁
- 生产环境代码验证通过后可删除**fix-bug-11568**分支

研发禁止的

- 禁止从 dev 分支 check 代码, 因为你要保证合并到 master 的只有你自己的, 没有其他人没测试的, 如果从 dev check 代码, 合并到 master 相当于 dev 合并到 master。
- 禁止 merge dev 到 master , 这样 dev 分支没意义。