

后端

Java01Plus

task5.Github Flow 工作流实践

.git/ 目录是什么，.git目录里放了什么？

.git/目录是Git用来存放版本控制相关的信息的文件夹，里面包括有，仓库的配置文件，对象数据库（储存所有提交了的对象），引用文件（如.git/HEAD指向当前的分支），当然还有缓存区的一些信息

在IDEA的git工具中，「添加」，「提交」，「回滚」，

「签出」，「删除」，「合并」，「变基」，「克隆」，「提取」，「更新」，「将传入更改合并到当前分支」，「在传入更改上变基当前分支」，「推送」。这些分别表示git的什么命令，有什么作用？

「添加」(git add):将工作区的新的变动提交到缓存区。

「提交」(git commit):将缓存区的所用内容提交到本地仓库。

「回滚」(git reset):可以撤销提交，以及回到之前的某个版本。

「签出」(git checkout):用于切换分支。

「删除」(git rm):从版本中删除指定文件。

「合并」(git merge):将一个分支合并到当前分支。

「变基」(git rebase):将一条分支上的一长串的提交并入另一个分支(一般开发会用到)

「克隆」(git clone):将远程仓库复制到本地

「提取」(git fetch):从远程仓库获取最新的一次提交（但不会合并到当前的分支上）

「更新」(git pull):这个是更新并合并，也就是fetch+merge

「将传入更改合并到当前分支」:这个与下一个的区别在于它会把远程分支合并到本地并创建一个新的合并提交。多人开发时比较适用。

「在传入更改上变基当前分支」:这个会将当前分支的更改提交给远程分支，再把远程的分支合并后拉回本地。（这两个结果差不多，过程相反）

「推送」(git push):直接粗暴的把本地仓库推给远程仓库

fork和clone有什么区别，Pull Request和push有什么区别？

clone:是把已经存在的一个远程仓库下载到本地

fork:是在平台上创建一个副本，我们可以在副本里进行修改，也可以把它再次合并到原来的仓库里

Pull Request:对**副本**仓库操作

Pull:对**本地**仓库操作

git管理的文件存在于四个区：工作区、暂存区、本地仓库、远程仓库，这四个区分别指什么，git管理的文

件如何在这四个区“移动”？

工作区:就是你本地的文件夹

缓存区:也不太讲得清，就是你用add指令把文件提交到的地方就叫缓存区

本地仓库:你用commit指令就会把缓存区的文件提交到本地仓库，这个仓库存在于本地（废话），里面会有记录

远程仓库:push指令会把本地仓库的东西推给远程仓库，需要去类似github这样的网站上创建。

工作区——>>缓存区——>>本地仓库——>>远程仓库

add commit push

什么是git冲突？冲突发生的条件是什么，有哪些操作会引发冲突？应该如何处理？

git冲突是什么:多个分支对同一文件进行了修改,而git并不能确定该采用哪个修改

条件好像已经说了

这些操作有:git merge,git rebase(当然背地里肯定有push啥的)

处理方法:挺多这里列出一些

- 手动编辑解决冲突(两个部分会给你标识出来,把不要的部分手动删掉)
 - 很多开发环境有自己的解决工具
 - 使用git mergetool
 - 当然也可以直接用命令告诉git该使用哪个分支
-

[拓展]如果你在某一个分支上进行了一些开发（修改），但没有提交，切换到另一个分支，你的工作区

文件会如何变化呢？注意，这个情况比较复杂，不用太过深究。

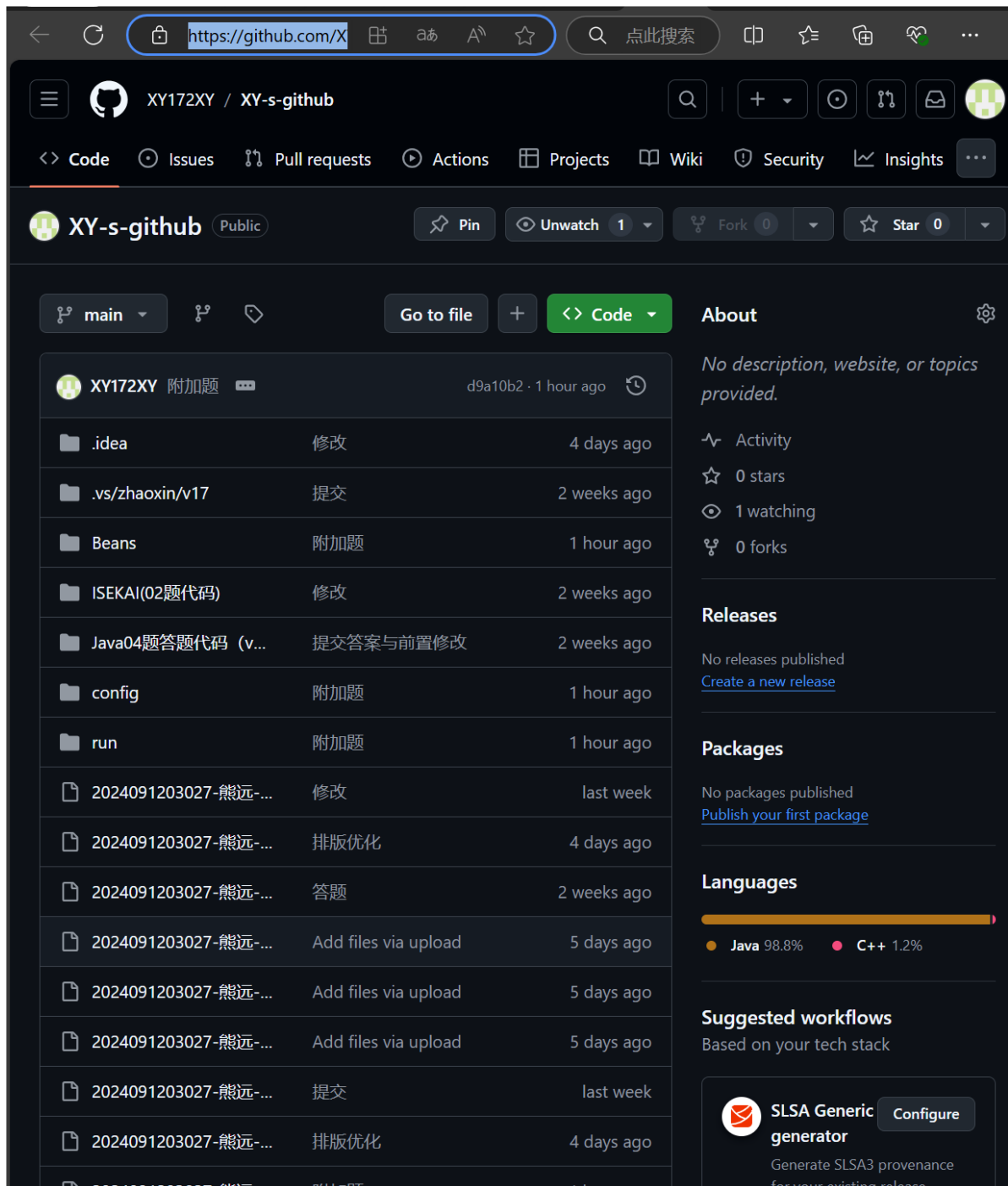
欸,这个好像是比较复杂的一个事,据文章所说一般来讲,工作区的文件会保持不变,但如果修改的文件在两个分支中有不同版本的话会给你提示。

反正就一句话，最好不要这样。

task6.请完成一个简单的Github Flow workflow实践，并记录下你的实践过程

- \1. 安装并配置Git（若之前已完成可跳过）
- \2. 在一个远程仓库（如github,gitee）新建代码库
- \3. 克隆远程仓库至本地（加分项：配置SSH进行克隆）
- \4. 在本地仓库进行开发工作

1. 这个已经搞好了
2. 其实这个也...还是贴张图记录一下吧



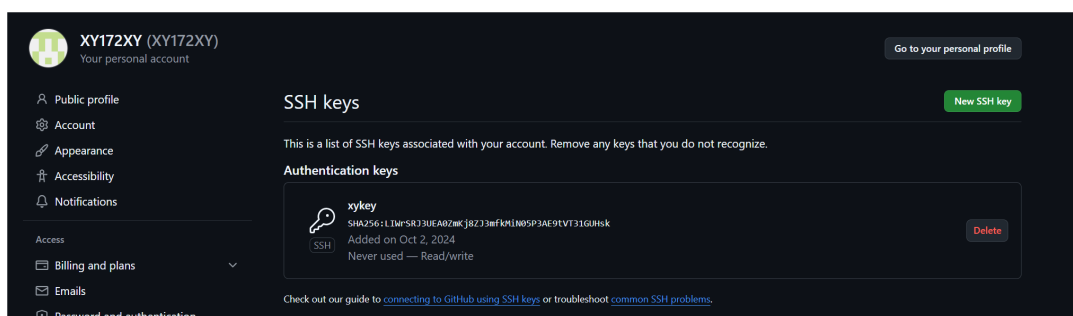
3. 好的,必须试试配置ssh,以下是步骤
 1. 生成密钥

```

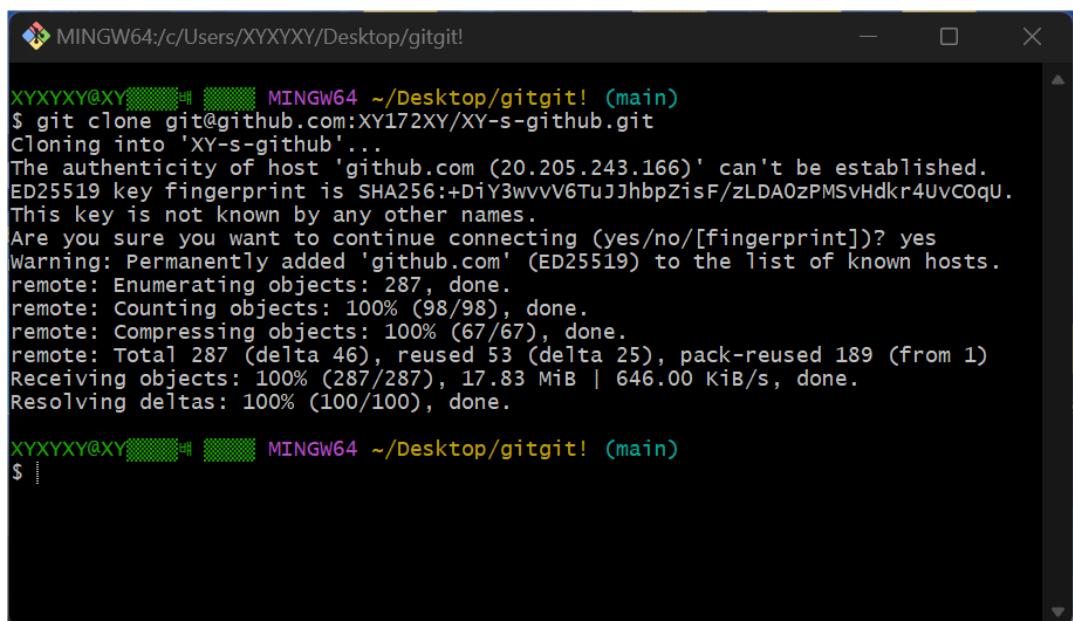
PS C:\Users\XYXYXY\Desktop> ssh-keygen -t rsa -C "3303771708@qq.com"
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\XYXYXY\.ssh/id_rsa):
Created directory 'C:\Users\XYXYXY\.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\XYXYXY\.ssh/id_rsa
Your public key has been saved in C:\Users\XYXYXY\.ssh/id_rsa.pub
The key fingerprint is:
SHA256:LIWwSRJ3UEA0ZmKj8ZJ3mfkMiN05P3AE9tVT31GUHsk 3303771708@qq.com
The key's randomart image is:
+---[RSA 3072]-----+
| . +o@+o .. .. .o*|
| 0 0 X o o . E.|
| = * % = . . o o|
| o + X + .|
| . . B S|
| o o o|
| o|
+-----[SHA256]-----+
PS C:\Users\XYXYXY\Desktop>

```

2. 添加密钥到github



3. 使用SSH克隆



4. 模拟

```

XYXYXY@XY:~$ git checkout -b develop
Switched to a new branch 'develop'

XYXYXY@XY:~$ git add .

XYXYXY@XY:~$ git commit -m"假装是一个开发"
[develop 218cae5] 假装是一个开发
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "\\345\\201\\207\\350\\243\\205\\346\\230\\257\\344\\270\\200\\344\\270\\252\\345\\274\\200\\345\\217\\221.txt"

```

```

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$ git push origin develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 32 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 304 bytes | 152.00 KiB/s, done.
Total 3 (delta 1), reused 1 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/XY172XY/XY-s-github/pull/new/develop
remote:
To github.com:XY172XY/XY-s-github.git
 * [new branch]      develop -> develop

```

创建feature:对应 创建,传入本地仓库,合并,删除操作

```

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$ git checkout -b feature
Switched to a new branch 'feature'

```

```

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (feature)
$ git add .

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (feature)
$ git commit -m'假装再次进行开发'
[feature 71b6848] 假装再次进行开发
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 "\\345\\201\\207\\350\\243\\205\\345\\217\\210\\350\\277\\233\\350\\241\\214\\344\\272\\206\\345\\274\\200\\345\\217\\221.txt"

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (feature)
$ git checkout develop
Switched to branch 'develop'

```

```

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$ git merge feature
Updating 218cae5..71b6848
Fast-forward
 ...7\\210\\350\\277\\233\\350\\241\\214\\344\\272\\206\\345\\274\\200\\345\\217\\221.txt" | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 "\\345\\201\\207\\350\\243\\205\\345\\217\\210\\350\\277\\233\\350\\241\\214\\344\\272\\206\\345\\274\\200\\345\\217\\221.txt"

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$ git push origin develop
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 32 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 274 bytes | 274.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:XY172XY/XY-s-github.git
 218cae5..71b6848 develop -> develop

```

```

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$ git branch -d feature
Deleted branch feature (was 71b6848).

```

将develop合并到main

```

XYXYXY@XY MINGW64 ~/Desktop/XY-s-github (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

XYXYXY@XY MINGW64 ~/Desktop/XY-s-github (main)
$ git merge develop
Updating d9a10b2..71b6848
Fast-forward
...7\210\350\277\233\350\241\214\344\272\206\345\274\200\345\217\221.txt" | 0
...3\205\346\230\257\344\270\200\344\270\252\345\274\200\345\217\221.txt" | 0
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "\345\201\207\350\243\205\345\217\210\350\277\233\350\241\214\344\272\206\345\274\200\345\217\221.txt"
create mode 100644 "\345\201\207\350\243\205\346\230\257\344\270\200\344\270\252\345\274\200\345\217\221.txt"

```

操作重复了,我就不贴出来了,这里的原因应该是:

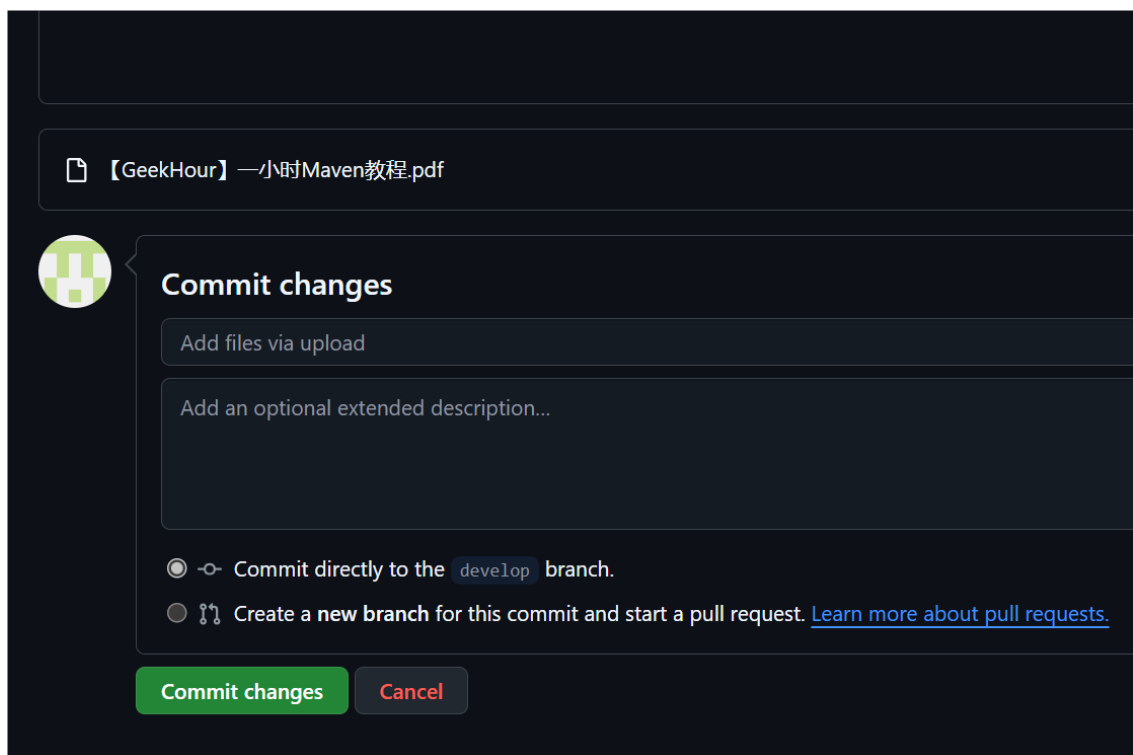
要保证develop中正在开发的代码中的bug也被修复了,直接合并到main中就不能修复develop中的bug了

5. 模拟多人协作开发

因为你的本地仓库中的develop与远程仓库不一致,为了防止出现不必要的数据丢失,我们需要先更新本地仓库再进行push

下面是设计的案例:

在远程仓库develop分支直接提交一个文件



【GeekHour】一小时Maven教程.pdf

Commit changes

Add files via upload

Add an optional extended description...

☒ Commit directly to the `develop` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

此时本地无法进行推送

```

XYXYXY@XY MINGW64 ~/Desktop/XY-s-github (develop)
$ git push origin develop
To github.com:XY172XY/XY-s-github.git
! [rejected]        develop -> develop (fetch first)
error: failed to push some refs to 'github.com:XY172XY/XY-s-github.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

```

我们执行pull操作(不小心把窗口关了,没截到图)

最后push提交

```

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$ git push origin develop
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 32 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 492 bytes | 492.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:XY172XY/XY-s-github.git
18efb51..73e0991 develop -> develop

XYXYXY@XY: MINGW64 ~/Desktop/XY-s-github (develop)
$

```

6. 假设你和另一名开发者同时开发一个项目，该项目使用github作为远程仓库。请描述你们的开发过程应该如何使用git工具进行项目管理，以及中途可能遇到的问题。

开发过程：

1. 初始化项目：

一名开发者在本地创建项目，并初始化 Git 仓库，执行 `git init`。

将项目推送到 GitHub 远程仓库，执行 `git remote add origin <remote_repository_url>`，然后 `git push -u origin master`。

2. 其他开发者克隆项目：

另一名开发者执行 `git clone <remote_repository_url>` 克隆项目到本地

3. 开发

团队里的每个开发者在本地创建develop分支经行开发,并进行add和commit

当开发完成一定阶段后push

4. 合并分支：

当一个开发者的分支需要合并到主分支,发起 Pull Request

负责人审查后，将代码合并到主分支

可能遇到的问题：

1. 冲突问题：当多个开发者对同一个文件的同一部分进行了不同的修改，在合并分支时可能会出现冲突。解决方法就是上述提到的手动解决冲突或使用工具辅助解决。
2. 也许还有什么兼容问题吧...这个我就不太知道了