

原创

AhuntSun

于 2020-04-20 12:10:21 发布


1014

收藏 5

版权

分类专栏: git

文章标签: git

 git 专栏收录该内容

3 订阅

9 篇文章

订阅专栏

文章目录

前言

一、`submodule`

1.引入子库

`` git submodule add 子库地址 保存目录``

2.同步子库变化

方法一

方法二

3.复制父版本库

分步操作

合并操作

4.删除子版本库

二、`subtree`

1.简介

2.创建子库

3.建立关联

4.同步子库变化

5.参数`--squash`

6.修改子库

存在的问题

解决方案

7.抽离子库

``git subtree split``

前言

前情提要：[Git应用详解第九讲：Git cherry-pick与Git rebase](#)

一个中大型项目往往会依赖几个模块，`git` 提供了子库的概念。可以将这些子模块存放在不同的仓库中，通过 `submodule` 或 `subtree` 实现仓库的 **嵌套** 。本讲为 `Git` 应用详解的倒数第二讲，胜利离我们不远了！

文章目录

前言

一、 submodule

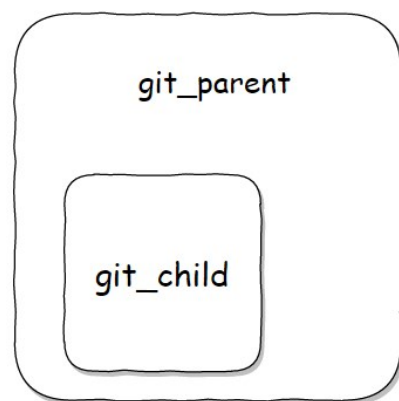
二、 subtree

	git	9篇
	前端	2篇
	性能优化	1篇
	图床	1篇
	浏览器	1篇
	C#	8篇



一、submodule

submodule：子模块的意思，表示将一个版本库作为子库引入到另一个版本库中：



1.引入子库

需要使用如下命令：

```
git submodule add 子库地址 保存目录
```

比如：

```
1 | git submodule add git@github.com:AhuntSun/git_child.git mymodule
```

执行上述命令会将地址对应的远程仓库作为子库，保存到当前版本库的 `mymodule` 目录下：

```
ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_parent (master)
$ git submodule add git@github.com:Ahuntsun/git_child.git mymodule
Cloning into 'D:/LearnGit/git_parent/mymodule'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), 269 bytes | 44.00 KiB/s, done.

ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_parent (master)
```

随后查看当前版本库的状态：

```
ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_parent (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   .gitmodules
        new file:   mymodule
```

可以发现新增了两个文件。查看其中的 `.gitmodules` 文件：

```
ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_parent (master)
$ cat .gitmodules
[submodule "mymodule"]
    path = mymodule
    url = git@github.com:Ahuntsun/git_child.git
```

可以看到当前文件的路径和子模块的 `url`，随后将这两个新增文件**添加**、**提交并推送**。在当前仓库 `git_parent` 对应的远程仓库中多出了两个文件：

其中 `mymodule` 文件夹上的 `3bd7f76` 对应的是**子仓库** `git_child` 中的**最新提交**：

点击 `mymodule` 文件夹，会自动跳转到**子仓库**中：

通过上述分析，可以得出结论：两个仓库已经关联起来了，并且仓库 `git_child` 为仓库 `git_parent` 的子仓库；

2.同步子库变化

当被依赖的子版本库发生变化时：在子版本库 `git_child` 中新增文件 `world.txt` 并提交到远程仓库：

```
ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_child (master)
$ vi world.txt

ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_child (master)
$ git add .
g
ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_child (master)
$ git commit -m 'add world.txt'
[master 565a6e9] add world.txt
 2 files changed, 2 insertions(+)
 create mode 100644 world.txt

ALLEN@DESKTOP-DRFJFS3 MINGW64 /d/LearnGit/git_child (master)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 841 bytes | 210.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To github.com:Ahuntsun/git_child.git
   3bd7f76..565a6e9  master -> master
```

这个时候依赖它的父版本库 `git_parent` 要如何感知这一变化呢？

方法一

这个时候 `git_parent` 只需要进入存放子库 `git_child` 的目录 `mymodule`，执行 `git pull` 就能将子版本库 `git_child` 的更新拉取到本地：

方法二

当父版本库 `git_parent` 依赖的多个子版本库都发生变化时，可以采用如下方法遍历更新所有子库：首先回到版本库主目录，执行以下指令：

```
1 | git submodule foreach git pull
```

该命令会遍历当前版本库所依赖的所有子版本库，并将它们的更新拉取到父版本库 `git_parent`：

拉取完成后，查看状态，发现 `mymodule` 目录下文件发生了变化，所以需要执行一次添加、提交、推送操作：

3.复制父版本库

如果将使用了 `submodule` 添加依赖了子库的父版本库 `git_parent` , 克隆一份到本地的话。在克隆出来的新版本库 `git_parent2` 中, 原父版本库存放依赖子库的目录虽在, 但是内容不在:

进入根据 `git_parent` 复制出来的仓库 `git_parent2` , 会发现 `mymodule` 目录为空:

****解决方法:****可采用多条命令的分步操作, 也可以通过参数将多步操作进行合并。

分步操作

这是在执行了 `clone` 操作后的额外操作, 还需要做两件事:

手动初始化 `submodule` :

```
1 | git submodule init
```

手动拉取依赖的子版本库; :

```
1 | git submodule update --recursive
```

执行完两步操作后, 子版本库中就有内容了。由此完成了 `git_parent` 的克隆;

合并操作

分步操作相对繁琐, 还可以通过添加参数的方式, 将多步操作进行合并。通过以下指令基于 `git_parent` 克隆一份 `git_parent3` :

```
1 | git clone git@github.com:AhuntSun/git_parent.git git_parent3 --recursive
```

`--recursive` 表示递归地克隆 `git_parent` 依赖的所有子版本库。

4.删除子版本库

`git` 没有提供直接删除 `submodule` 子库的命令, 但是我们可以通过其他指令的组合来达到这一目的, 分为三步:

将 `submodule` 从版本库中删除:

```
1 | git rm --cache mymodule
```

`git rm` 的作用为删除版本库中的文件, 并将这一操作纳入暂存区;

- 将 `submodule` 从工作区中删除;
- 最后将 `.gitmodules` 目录删除;

完成三步操作后, 再进行添加, 提交, 推送即可完成删除子库的操作:

二、`subtree`

1.简介

`subtree` 与 `submodule` 的作用是一样的, 但是 `subtree` 出现得比 `submodule` 晚, 它的出现是为了弥补 `submodule` 存在的问题:

- **第一:** `submodule` 不能在父版本库中修改子版本库的代码, 只能在子版本库中修改, 是单向的;
- **第二:** `submodule` 没有直接删除子版本库的功能;

而 `subtree` 则可以实现双向数据修改。官方推荐使用 `subtree` 替代 `submodule`。

2.创建子库

首先创建两个版本库: `git_subtree_parent` 和 `git_subtree_child` 然后在 `git_subtree_parent` 中执行 `git subtree` 会列出该指令的一些常见的参数:

3.建立关联

首先需要给 `git_subtree_parent` 添加一个子库 `git_subtree_child`:

****第一步: ****添加子库的远程地址:

```
1 | git remote add subtree-origin git@github.com:AhuntSun/git_subtree_child.git
```

添加完成后, 父版本库中就有两个远程地址了:

这里的 `subtree-origin` 就代表了远程仓库 `git_subtree_child` 的地址。

****第二步: ****建立依赖关系:

```
1 | git subtree add --prefix=subtree subtree-origin master --squash
2 | //其中的--prefix=subtree可以写成: --p subtree 或 --prefix subtree
```

该命令表示将远程地址为 `subtree-origin` 的, 子版本库上 `master` 分支的, 文件克隆到 `subtree` 目录下;

注意: 是在某一支 (如 `master`) 上将 `subtree-origin` 代表的远程仓库的某一支 (如 `master`) 作为子库拉取到 `subtree` 文件夹

中。可切换到其他分支重复上述操作，也就是说子库的实质就是子分支。

`--squash` 是可选参数，它的含义是**合并，压缩**的意思。

- 如果不增加这个参数，则会把远程的子库中指定的分支（这里是 `master`）中的提交一个一个地拉取到本地再去创建一个合并提交；
- 如果增加了这个参数，会将远程子库指定分支上的多次提交合并压缩成一次提交再拉取到本地，这样拉取到本地的，远程子库中的，指定分支上的，历史提交记录就没有了。

拉取完成后，父版本库中会增添一个 `subtree` 目录，里面是子库的文件，相当于把依赖的子库代码拉取到了本地：

此时查看一下父版本库的提交历史：

会发现其中没有子库李四的提交信息，这是因为 `--squash` 参数将他的提交压缩为一次提交，并由父版本库张三进行合并和提交。所以父版本库多出了两次提交。

随后，我们在父版本库中进行一次推送：

结果远程仓库中多出了一个存放子版本库文件的 `subtree` 目录，并且完全脱离了版本库 `git_subtree_child`，仅仅是属于父版本库 `git_subtree_parent` 的一个目录。而不像使用 `submodule` 那样，是一个点击就会自动跳转到依赖子库的**指针**：

- `subtree` 的远程父版本库：
- `submodule` 的远程父版本库：

即 `submodule` 与 `subtree` 子库的区别为：

4.同步子库变化

在子库中创建一个新文件 `world` 并推送到远程子库：

在父库中通过如下指令更新依赖的子库内容：

```
1 | git subtree pull --prefix=subtree subtree-origin master --squash
```

此时查看一下提交历史：

发现没有子库李四的提交信息，这都是 `--squash` 的作用。子库的修改交由父库来提交。

5. 参数 `--squash`

该参数的作用为：防止子库指定分支上的提交历史污染父版本库。比如在子库的 `master` 分支上进行了三次提交分别为： `a`、`b`、`c`，并推送到远程子库。

首先，复习一下合并分支时遵循的**三方合并**原则：

当提交 `4` 和 `6` 需要合并的时候，`git` 会先寻找二者的公共父提交节点，如图中的 `2`，然后在提交 `2` 的基础上进行 `2`、`4`、`6` 的三方合并，合并后得到提交 `7`。

父仓库执行 `pull` 操作时：如果添加参数 `--squash`，就会把远程子库 `master` 分支上的这三次提交合并为一次新的提交 `abc`；随后再与父仓库中子库的 `master` 分支进行合并，又产生一次提交 `X`。整个 `pull` 的过程一共产生了五次提交，如下图所示：

存在的问题：

由于 `--squash` 指令的合并操作，会导致远程 `master` 分支上的合并提交 `abc` 与本地 `master` 分支上的最新提交 `2`，找不到公共父节点，从而合并失败。同时 `push` 操作也会出现额外的问题。

最佳实践：要么全部操作都使用 `--squash` 指令，要么全部操作都不使用该参数，这样就不会出错。

错误示范：

为了验证，重新创建两个仓库 `A` 和 `B`，并通过 `subtree` 将 `B` 设置为 `A` 的子库。这次全程都没有使用参数 `--squash`，重复上述操作：

- 首先，修改子库文件；
- 然后，通过下列指令，在不使用参数 `--squash` 的情况下，将远程子库 `A` 变化的文件拉取到本地：

```
1 | git subtree pull --prefix=subtree subtree-origin master
```

此时查看提交历史：

可以看到子库 `儿子` 的提交信息污染了父版本库的提交信息，验证了上述的结论。

所以要么都使用该指令，要么都不使用才能避免错误；如果不需要子库的提交日志，推荐使用 `--squash` 指令。

补充: `echo 'new line' >> test.txt` : 表示在 `test.txt` 文件末尾追加文本 `new line` ; 如果是一个 `>` 表示替换掉 `test.txt` 内的全部内容。

6.修改子库

`subtree` 的强大之处在于, 它可以在父版本库中修改依赖的子版本库。以下为演示:

进入父版本库存放子库的 `subtree` 目录, 修改子库文件 `child.txt`, 并推送到远程父仓库:

此时远程父版本库中存放子库文件的 `subtree` 目录发生了变化, 但是独立的远程子库 `git_subtree_child` 并没有发生变化。

修改独立的远程子库:

可执行以下命令, 同步地修改远程子版本库:

```
1 | git subtree push --prefix=subtree subtree-origin master
```

如下图所示, 父库中的子库文件 `child.txt` 新增的 `child2` 内容, 同步到了独立的远程子库中:

修改独立的本地子库:

回到本地子库 `git_subtree_child`, 将对应的远程子库进行的修改拉取到本地进行合并同步:

由此无论是远程的还是本地的子库都被修改了。

实际上使用 `subtree` 后, 在外部看起来父仓库和子仓库是一个整体的仓库。执行 `clone` 操作时, 不会像 `submodule` 那样需要遍历子库来单独克隆。而是可以将整个父仓库和它所依赖的子库当做一个整体进行克隆。

存在的问题

父版本库拉取远程子库进行更新同步会出现的问题:

子仓库第一次修改:

经历了上述操作, 本地子库与远程子库的文件达到了同步, 其中文件 `child.txt` 的内容都是 `child~4`。在此基础上本地子库为该文件添加 `child5~6`:

然后推送到远程子库。

父仓库第一次拉取:

随后父版本库通过下述指令, 拉取远程子库, 与本地父仓库 `git_subtree_parent` 中的子库进行同步:

```
1 | git subtree pull --p subtree subtree-origin master --squash
```

结果出现了合并失败的情况：

我们查看冲突产生的文件：

发现父版本库中的子库与远程子库内容上并无冲突，但是却发生了冲突，这是为什么呢？

探究冲突产生的原因之前我们先解决冲突，先删除多余的内容：

随后执行 `git add` 命令和 `git commit` 命令标识解决了冲突：

解决完冲突后将该文件推送到独立的远程子库，发现文件并没有发生更新，也就是说 `git` 认为我们并没有解决冲突：

子仓库第二次修改与父仓库第二次拉取：

再次修改本地子库的文件并推送到对应的远程仓库，父版本库再次将远程子库更新的文件拉取到本地进行同步：

这次却成功了！为什么同样的操作，有的时候成功有的时候失败呢？

解决方案

原因出现在 `--squash` 指令中。实际上，`--squash` 指令把子库中的提交信息合并了，导致父仓库在执行 `git pull` 操作时找不到公共的父节点，从而导致即使文件没有冲突的内容，也会出现合并冲突的情况。其实不使用 `--squash` 也会有这种问题，问题的根本原因仍然是**三方合并时找不到公共父节点**。我们打开 `gitk`：

从图中不难看出，当使用 `subtree` 时，子库与父库之间是没有公共节点的，所以时常会因为找不到公共节点而出现合并冲突的情况，此时只需要解决冲突，手动合并即可。

不使用 `subtree` 时，普通的版本库中的各分支总会有一个公共节点：

****再次强调：使用 `--squash` 指令时一定要小心，要么都使用它，要么都不使用。**

7.抽离子库

git subtree split

当开发过程中出现某些子库完全可以复用到其他项目中时，我们希望将它独立出来。

- **方法一： **可以手动将文件拷贝出来。缺点是，这样会丢失关于该子库的提交记录；
- **方法二： **使用 `git subtree split` 指令，该指令会把关于独立出来的子库的每次提交都记录起来。但是，这样存在弊端：
 - 比如该独立子库为 `company.util`，当一次提交同时修改了 `company.util` 和 `company.server` 两个子库时。
 - 通过上述命令独立出来的子库 `util` 只会记录对自身修改的提交，而不会记录对 `company.server` 的修改，这样在别人看来这次提交就只修改了 `util`，这是不完整的。

以上就是本讲的全部内容，主要介绍了 `git` 子库的基本使用方法。下一讲将是 `Git` 应用详解系列的完结篇：`Git` 工作流 `Gitflow`。我们下一讲再见！

文章知识点与官方知识档案匹配，可进一步学习相关知识

CS入门技能树 Git入门 Git简介 1518 人正在系统学习中

为什么使用Git Subtree 05-29
分享作者关于版本控制工具Git插件的使用心得，主要针对Submodule与Subtree之间的比较，以及如何使用Subtree。

git subtree 不断增加的推送时间，解不玩的冲突！这篇文章应该能救你 walterlv - 吕毅 617
原生 git 对于公共组件那种类型的子仓库的支持并不怎么好，就是那种某个子文件夹是一个另外的 git 仓库，并被多个 git 父仓库使用的形式。实际使用的...

评论 1

请发表有价值的评论， 博客评论不欢迎灌水，良好的社区氛围需大家一起维护。

太阳与我肩并肩 2021.06.09
写的太好了，学习到了，也欢迎互关

java git subtree_ Git Subtree的使用 weixin_34413326的博客 40
背景项目A与项目B存在公用模块，在项目A中修改Bug或增加新功能需要同步到项目B中，由于存在区别所以还不能完全copy需求分析公用代码迁移出去独...

Git subtree和Git submodule weixin_30363981的博客 55
git submodule允许其他的仓库指定以一个commit嵌入仓库的子目录。 git subtree替代git submodule命令，合并子仓库到项目中的子目录。不用像submod...

Git submodule subtree 使用 区别 l_can_move_you的专栏 998
使用场景：多个项目代码在版本控制服务器，不同项目代码需要直接复用，依赖和被依赖项目能够双向同步代码。 环境要求： Git: https://git-scm.com/d...

java git subtree_ Git -- subtree与submodule weixin_42389421的博客 40
1 - 仓库共用(子仓库、子项目)两种子仓库使用方式git submodule(子模块)git subtree(子树合并)从1.5.2版本开始，官方新增Git Subtree并推荐使用这个功...

git子模块使用之git submodule与 git subtree比较 热门推荐 liusf1993的博客 1万+
简述在开发中，稍微复杂一点的项目都会有多个功能模块，大致结构可能是这样 project ├──moduleA ├──submoduleC ├──submoduleD ├──moduleB...

Git笔记7（ submodule与subtree） 学习的最好方法就是去做 1123
git init --bare 创建裸库（没有工作区，仅仅存放和中转开发者提交代码） 将一个git仓库引入到另一个git仓库git submodule add git@xxx mymodule 将远程...

git 子仓库(submodule)操作 whuzhang16的博客 1017
1、在主仓库添加子仓库： 直接在主仓库路径下clone子仓库代码，如我的主仓库是test_git_master， clone到src/perception路径， clone用完git status查看...

AhuntSun

码龄4年 暂无认证

20	47万+	162万+	8187	
原创	周排名	总排名	访问	等级
265	20	22	15	83
积分	粉丝	获赞	评论	收藏

私信

关注

搜博文文章

Git应用详解第十讲：Git子库： submodule与subtree 1007

从宏观到细节为你讲解前端性能优化 921

Git应用详解第九讲：Git cherry-pick与Git rebase 745

C#方法的定义、调用与调试 667

Git应用详解第四讲：版本回退的三种方式与stash 414

Git应用详解第五讲：远程仓库Github与G...

太阳与我肩并肩: 之前有看你的一篇文章，写的很好，期待您的回复与关注!

Git应用详解第十讲：Git子库： submodule...
太阳与我肩并肩: 写的太好了，学习到了，也欢迎互关

Git应用详解第八讲：Git标签、别名与Git...
Niklaus.ad: 第七讲咋没了

Git应用详解第九讲：Git cherry-pick与Git...
bugpool: 很全,感谢.

从宏观到细节为你讲解前端性能优化
AhuntSun 回复 极客日报: 感谢你的建议!









强烈不推荐 不推荐 一般般 推荐 强烈推荐

Git应用详解第九讲：Git cherry-pick与Git rebase

Git应用详解第八讲：Git标签、别名与Git gc

Git应用详解第六讲：Git协作与Git pull常见问题

2020年 12篇

2019年 8篇

git subtree

weixin_34226706的博客 8998

此文已由作者张磊授权网易云社区发布。欢迎访问网易云社区，了解更多网易技术产品运营经验。前言目前对 git 仓库拆分的已有实现之一。这里 git subtr...

Git Subtree 的介绍及使用

BingShuShu的专栏 8998

Git Subtree 的介绍及使用应用场景 有项目A、项目B。有LibraryC，为项目A的子项目子目录。项目B也想用LibraryC，作为自己的子项目子目录。希...

Git创建子仓库命令submodule的使用

u013463707的专栏 4032

使用Git submodule步骤： 一、先clone 一个父仓库： git clone "ssh://1632@ip:29418/test" cd test git submodule add <submodule_url> 例如： git submo...

git submodule subtree常用指令

anshaobiao6449的博客 51

submodule 官方文档 添加 git submodule add -b master git@git.xxx:xxx/xxx.git src/xxx 删除 git submodule deinit -f src/xxx // 取消注册 git rm -rf src/xxx // git...

git子仓库的理解，合并

最新发布

furfur-jiang的博客 370

git子仓库和主仓库是完全分离开的，主仓库通过保存子仓库的commit来链接。注意是子仓库的commit 可能出现的问题： 本地merge后，commit没有推送...

git子仓库的处理

daoliting5268的博客 537

https://www.jianshu.com/p/491609b1c426

git中子模块/子仓库的使用

罗伯特祥的博客 240

文章目录0. 常用git指令1. git 创建submodule2. 拉取submodule3. submodule的更新4. 删除 submodule 0. 常用git指令 # 拉取项目 git clone [地址] # 查看...

git submodule的坑

lishenglong666的专栏 642

前言 对于一些比较大的工程，为了便于复用，常常需要抽取子项目。例如我开发的猿题库客户端现在包括3门考试，客户端涉及的公共U、公共底层逻辑...

“相关推荐”对你有帮助么？



非常没帮助



没帮助



一般



有帮助

非常有帮助

©2022 CSDN 皮肤主题：数字20 设计师：CSDN官方博客 返回首页

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

AhuntSun 关注

3 1 5

专栏目录





举报

