咖啡兔

27 Mar 2012 Comments

Git Submodule使用完整教程

自从看了蒋鑫的《Git权威指南》之后就开始使用Git Submodule功能，团队也都熟悉了怎么使用，多个子系统（模块）都能及时更新到最新的公共资源，把使用的过程以及经验和容易遇到的问题分享给大家。

`Git Submodule`功能刚刚开始学习可能觉得有点怪异，所以本教程把每一步的操作的命令和结果都用代码的形式展现给大家，以便更好的理解。

1.对于公共资源各种程序员的处理方式

每个公司的系统都会有一套统一的系统风格，或者针对某一个大客户的多个系统风格保持统一，而且如果风格改动后要同步到多个系统中；这样的需求几乎每个开发人员都遇到，下面看看各个层次的程序员怎么处理：

假如对于系统的风格需要几个目录：css、images、js。

- **普通**程序员，把最新版本的代码逐个复制到每个项目中，如果有N个项目，那就是要复制**N x 3**次；如果漏掉了某个文件夹没有复制...@（&#@#。
- **文艺**程序员，使用Git Submodule功能，执行：**git submodule update**，然后冲一杯咖啡悠哉的享受着。

引用一段《Git权威指南》的话： 项目的版本库在某些情况虾需要引用其他版本库中的文件，例如公司积累了一套常用的函数库，被多个项目调用，显然这个函数库的代码不能直接放到某个项目的代码中，而是要独立为一个代码库，那么其他项目要调用公共函数库该如何处理呢？分别把公共函数库的文件拷贝到各自的项目中会造成冗余，丢弃了公共函数库的维护历史，这显然不是好的方法。

2.开始学习Git Submodule

“工欲善其事，必先利其器”！

既然文艺程序员那么轻松就搞定了，那我们就把过程一一道来。

说明：本例采用**两个项目**以及**两个公共类库**演示对submodule的操作。因为在一写资料或者书上的例子都是一个项目对应1～N个lib，但是实际应用往往并不是这么简单。

2.1 创建Git Submodule测试项目

2.1.1 准备环境

```
1 ➜ henryyan@hy-hp ~ pwd ?
```

```
/home/henryyan
mkdir -p submd/repos
```

创建需要的本地仓库：

```
cd ~/submd/repos
git --git-dir=lib1.git init --bare
git --git-dir=lib2.git init --bare
git --git-dir=project1.git init --bare
git --git-dir=project2.git init --bare
```

初始化工作区：

```
mkdir ~/submd/ws
cd ~/submd/ws
```

2.1.2 初始化项目

初始化project1：

```
➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/project1.git
Cloning into project1...
done.
warning: You appear to have cloned an empty repository.

➜ henryyan@hy-hp  ~/submd/ws  cd project1
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) echo "project1" > project-in
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ ls
project-infos.txt

➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git add project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:   project-infos.txt
#
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git commit -m "init projec
[master (root-commit) 473a2e2] init project1
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 232 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/ws/../repos/project1.git
 * [new branch]      master -> master
</pre>

初始化project2:
<pre class="brush: shell">
➜ henryyan@hy-hp  ~/submd/ws/project1 cd ..
➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/project2.git
Cloning into project2...
done.
warning: You appear to have cloned an empty repository.

➜ henryyan@hy-hp  ~/submd/ws  cd project2
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) echo "project2" > project-in
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ ls
project-infos.txt

➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git add project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status
# On branch master
```

```
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:   project-infos.txt
#
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -m "init projec
[master (root-commit) 473a2e2] init project2
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 232 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/ws/../repos/project2.git
 * [new branch]      master -> master
</pre>

#### 2.1.3 初始化公共类库

初始化公共类库lib1:
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/lib1
Cloning into lib1...
done.
warning: You appear to have cloned an empty repository.
➜ henryyan@hy-hp  ~/submd/ws  cd lib1
➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) echo "I'm lib1." > lib1-features
➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) ✗ git add lib1-features
➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) ✗ git commit -m "init lib1"
[master (root-commit) c22aff8] init lib1
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 lib1-features
➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 227 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/ws/../repos/lib1.git
 * [new branch]      master -> master
</pre>

初始化公共类库lib2:
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/lib1 git:(master) cd ..
➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/lib2.git
Cloning into lib2...
done.
warning: You appear to have cloned an empty repository.
➜ henryyan@hy-hp  ~/submd/ws  cd lib2
➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) echo "I'm lib2." > lib2-features
➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) ✗ git add lib2-features
➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) ✗ git commit -m "init lib2"
[master (root-commit) c22aff8] init lib2
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 lib2-features
➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 227 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/ws/../repos/lib2.git
 * [new branch]      master -> master
</pre>

### 2.2 为主项目添加Submodules

#### 2.2.1 为project1添加lib1和lib2
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/lib2 git:(master) cd ../
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ls
```

```
project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git submodule add ~/submd/re
Cloning into libs/lib1...
done.
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git submodule add ~/submd/
Cloning into libs/lib2...
done.
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ ls
libs  project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ ls libs
lib1  lib2

➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   .gitmodules
#   new file:   libs/lib1
#   new file:   libs/lib2
#

# 查看一下公共类库的内容

➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat libs/lib1/lib1-features
I'm lib1.
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat libs/lib2/lib2-features
I'm lib2.
</pre>
好了，到目前为止我们已经使用**git submodule add**命令为**project1**成功添加了两个公共类库
<pre class="brush: shell">n@hy-hp  ~/submd/ws/project1 git:(master) ✗ cat .gitmc
[submodule "libs/lib1"]
    path = libs/lib1
    url = /home/henryyan/submd/repos/lib1.git
[submodule "libs/lib2"]
    path = libs/lib2
    url = /home/henryyan/submd/repos/lib2.git
</pre>
原来如此，**.gitmodules**记录了每个submodule的引用信息，知道在当前项目的位置以及仓库的所在。

好的，我们现在把更改提交到仓库。
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗
[master 7157977] add submodules[lib1,lib2] to project1
 3 files changed, 8 insertions(+), 0 deletions(-)
 create mode 100644 .gitmodules
 create mode 160000 libs/lib1
 create mode 160000 libs/lib2

➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 491 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
To /home/henryyan/submd/ws/../repos/project1.git
   45cbbcb..7157977  master -> master
</pre>

    假如你是第一次引入公共类库的开发人员，那么项目组的其他成员怎么Clone带有Submodule的项目呢

### 2.3 Clone带有Submodule的仓库

模拟开发人员B……
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cd
➜ henryyan@hy-hp  ~/submd/ws  git clone ../repos/project1.git project1-b
Cloning into project1-b...
done.
➜ henryyan@hy-hp  ~/submd/ws  cd project1-b
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git submodule
-c22aff85be91eca442734dcb07115ffe526b13a1 libs/lib1
```

```
-7290dce0062bd77df1d83b27dd3fa3f25a836b54 libs/lib2
</pre>
```

看到submodules的状态是hash码和文件目录，但是注意前面有一个减号：**-**，含义是该子模块还没有

OK，检出project1-b的submodules……

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path
Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git submodule update
Cloning into libs/lib1...
done.
Submodule path 'libs/lib1': checked out 'c22aff85be91eca442734dcb07115ffe526b13a
Cloning into libs/lib2...
done.
Submodule path 'libs/lib2': checked out '7290dce0062bd77df1d83b27dd3fa3f25a836b5
</pre>
```

> 读者可以查看：.git/config文件的内容，最下面有submodule的注册信息！

验证一下类库的文件是否存在：

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
I'm lib1.
I'm lib2.
</pre>
```

> 上面的两个命令(git submodule init & update)其实可以简化，后面会讲到！

2.3 修改Submodule

我们在开发人员B的项目上修改Submodule的内容。

先看一下当前Submodule的状态：

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  git status
# Not currently on any branch.
nothing to commit (working directory clean)
</pre>
```

为什么是**Not currently on any branch**呢？不是应该默认在**master**分支吗？别急，一一解

Git对于Submodule有特殊的处理方式，在一个主项目中引入了Submodule其实Git做了3件事情：

* 记录引用的仓库

* 记录主项目中Submodules的目录位置

* 记录引用Submodule的**commit id**

在**project1**中push之后其实就是更新了引用的commit id，然后project1-b在clone的时候获取到

查看~/submd/ws/project1-b/libs/lib1的引用信息：

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  ca
c22aff85be91eca442734dcb07115ffe526b13a1
➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  cat .git/refs/heads/master
c22aff85be91eca442734dcb07115ffe526b13a1
</pre>
```

现在我们要修改lib1的文件需要先切换到**master**分支：

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1  gi
Switched to branch 'master'
➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) echo "add by dev
➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) ✗ git commit -a
[master 36ad12d] update lib1-features by developer B
 1 files changed, 1 insertions(+), 0 deletions(-)
</pre>
```

在主项目中修改Submodule提交到仓库稍微繁琐一点，在**git push**之前我们先看看**project1-b**

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
```

```
#
#   modified:   libs/lib1 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
</pre>

**libs/lib1 (new commits)**状态表示**libs/lib1**有新的提交，这个比较特殊，看看**projec
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
diff --git a/libs/lib1 b/libs/lib1
index c22aff8..36ad12d 160000
--- a/libs/lib1
+++ b/libs/lib1
@@ -1 +1 @@
-Subproject commit c22aff85be91eca442734dcb07115ffe526b13a1
+Subproject commit 36ad12d40d8a41a4a88a64add27bd57cf56c9de2
</pre>

从状态中可以看出**libs/lib1**的commit id由原来的**c22aff85be91eca442734dcb07115ffe52

<pre>注意：如果现在执行了git submodule update操作那么libs/lib1的commit id又会还原到c22

这样的话刚刚的修改是不是就丢死了呢？不会，因为修改已经提交到了master分支，只要再git checkout
</pre>

现在可以把**libs/lib1**的修改提交到仓库了：
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git:(master) git push
Counting objects: 5, done.
Writing objects: 100% (3/3), 300 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/repos/lib1.git
   c22aff8..36ad12d  master -> master
</pre>

现在仅仅只完成了一步，下一步要提交**project1-b**引用submodule的commit id：
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b/libs/lib1 git
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git add -u
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git commit -m "update li
[master c96838a] update libs/lib1 to lastest commit id
 1 files changed, 1 insertions(+), 1 deletions(-)
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 395 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/ws/../repos/project1.git
   7157977..c96838a  master -> master
</pre>

OK，大功高成，我们完成了Submodule的修改并把**libs/lib1**的最新commit id提交到了仓库。

接下来要看看**project1**怎么获取submodule了。

### 2.4 更新主项目的Submodules

好的，让我们先进入**project1**目录同步仓库：
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/ws/../repos/project1
   7157977..c96838a  master     -> origin/master
Updating 7157977..c96838a
Fast-forward
 libs/lib1 |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)
```

```
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   libs/lib1 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
</pre>

我们运行了**git pull**命令和**git status**获取了最新的仓库源码，然后看到了状态时**modifie

我们用**git diff**比较一下不同：
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗
diff --git a/libs/lib1 b/libs/lib1
index 36ad12d..c22aff8 160000
--- a/libs/lib1
+++ b/libs/lib1
@@ -1 +1 @@
-Subproject commit 36ad12d40d8a41a4a88a64add27bd57cf56c9de2
+Subproject commit c22aff85be91eca442734dcb07115ffe526b13a1
</pre>

从diff的结果分析出来时因为submodule的commit id更改了，我们前面刚刚讲了要在主项目更新submod

follow me……

<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   libs/lib1 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
</pre>

泥马，为什么没有更新？**git submodule update**命令不是更新子模块仓库的吗？

别急，先听我解释；因为子模块是在**project1**中引入的，**git submodule add ~/submd/repos
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd2/ws/project1 git:(master) ✗
[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = /home/henryyan/submd/ws/../repos/project1.git
[branch "master"]
    remote = origin
    merge = refs/heads/master
</pre>

我们说过**git submodule init**就是在**.git/config**中注册子模块的信息，下面我们试试注册之
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗
Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path
Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git submodule update
remote: Counting objects: 5, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/repos/lib1
   c22aff8..36ad12d  master     -> origin/master
Submodule path 'libs/lib1': checked out '36ad12d40d8a41a4a88a64add27bd57cf56c9de

➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat .git/config
[core]
```

```
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = /home/henryyan/submd/ws/../repos/project1.git
[branch "master"]
    remote = origin
    merge = refs/heads/master
[submodule "libs/lib1"]
    url = /home/henryyan/submd/repos/lib1.git
[submodule "libs/lib2"]
    url = /home/henryyan/submd/repos/lib2.git

➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cat libs/lib1/lib1-features
I'm lib1.
add by developer B
</pre>

上面的结果足以证明刚刚的推断，所以记得当需要更新子模块的内容时请先确保已经运行过**git submodu

### 2.5 为project2添加lib1和lib2

这个操作对于读到这里的你来说应该是**轻车熟路**了，action:
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cd
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git submodule add ~/submd/re
Cloning into libs/lib1...
done.
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git submodule add ~/submd/
zsh: correct 'libs/lib2' to 'libs/lib1' [nyae]? n
Cloning into libs/lib2...
done.
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ ls
libs  project-infos.txt
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git submodule init
Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path
Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path

➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   .gitmodules
#   new file:   libs/lib1
#   new file:   libs/lib2
#
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -a -m "add lib1
[master 8dc697f] add lib1 and lib2
 3 files changed, 8 insertions(+), 0 deletions(-)
 create mode 100644 .gitmodules
 create mode 160000 libs/lib1
 create mode 160000 libs/lib2
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 471 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
To /home/henryyan/submd/ws/../repos/project2.git
   6e15c68..8dc697f  master -> master

</pre>

我们依次执行了添加submodule并commit和push到仓库，此阶段任务完成。

### 2.6 修改lib1和lib2并同步到project1和project2

假如开发人员C同时负责**project1**和**project2**，有可能在修改**project1**的某个功能的时候
```

假如我的需求如下：

* 在lib1中添加一个文件：README，用来描述lib1的功能

* 在lib2中的lib2-features文件中添加一写文字：学习Git submodule的修改并同步功能

2.6.1 在lib1中添加一个文件：README

<pre class="brush: shell">➜ henryyan@hy-hp ~/submd/ws/project2 git:(master) cd
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib1 git:(master) echo "lib1 readme
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib1 git:(master) ✗ git add README
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib1 git:(master) ✗ git commit -m "a
[master 8c666d8] add file README
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 README
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib1 git:(master) git push
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 310 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/repos/lib1.git
36ad12d..8c666d8 master -> master
</pre>

前面提到过现在仅仅只完成了一部分，我们需要在**project2**中再更新lib1的commit id：

<pre class="brush: shell">➜ henryyan@hy-hp ~/submd/ws/project2 git:(master) ✗
On branch master
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
#
modified: libs/lib1 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
➜ henryyan@hy-hp ~/submd/ws/project2 git:(master) ✗ git add libs/lib1
➜ henryyan@hy-hp ~/submd/ws/project2 git:(master) ✗ git commit -m "update lib1
[master ce1f3ba] update lib1 to lastest commit id
1 files changed, 1 insertions(+), 1 deletions(-)
</pre>

我们暂时不push到仓库，等待和lib2的修改一起push。

2.6.2 在lib2中的lib2-features文件添加文字

<pre class="brush: shell">➜ henryyan@hy-hp ~/submd/ws/project2 git:(master) cd
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) echo "学习Git submc
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) ✗ git add lib2-fea
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) ✗ git commit -m "添
[master e372b21] 添加文字：学习Git submodule的修改并同步功能
1 files changed, 1 insertions(+), 0 deletions(-)
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 376 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/repos/lib2.git
7290dce..e372b21 master -> master

➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) echo "学习Git submc
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) ✗ git add lib2-fea
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) ✗ git commit -m "添
[master e372b21] 添加文字：学习Git submodule的修改并同步功能
1 files changed, 1 insertions(+), 0 deletions(-)
➜ henryyan@hy-hp ~/submd/ws/project2/libs/lib2 git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
</pre>

```
Writing objects: 100% (3/3), 376 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/repos/lib2.git
   7290dce..e372b21  master -> master
➜ henryyan@hy-hp  ~/submd/ws/project2/libs/lib2 git:(master) cd -
~/submd/ws/project2
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   libs/lib2 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git add libs/lib2
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) ✗ git commit -m "update lib2
[master df344c5] update lib2 to lastest commit id
 1 files changed, 1 insertions(+), 1 deletions(-)
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git status
# On branch master
# Your branch is ahead of 'origin/master' by 2 commits.
#
nothing to commit (working directory clean)
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git push
Counting objects: 8, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 776 bytes, done.
Total 6 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (6/6), done.
To /home/henryyan/submd/ws/../repos/project2.git
   8dc697f..df344c5  master -> master
</pre>

### 2.7 同步project2的lib1和lib2的修改到project1

现在project2已经享受到了最新的代码带来的**快乐**，那么既然project1和project2属于同一个风格

<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) cd
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git pull
Already up-to-date.
</pre>

看看上面的结果对吗？为什么lib1和lib2更新了但是没有显示**new commits**呢？说到这里我记得刚刚

帮大家分析一下问题，不过在分析之前先看看当前(project1和project2)的submodule状态：
<pre class="brush: shell"># project2 的状态，也就是我们刚刚修改后的状态
➜ henryyan@hy-hp  ~/submd/ws/project2 git:(master) git submodule
 8c666d86531513dd1aebdf235f142adbac72c035 libs/lib1 (heads/master)
 e372b21dffa611802c282278ec916b5418acebc2 libs/lib2 (heads/master)

# project1 的状态，等待更新submodules
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git submodule
 36ad12d40d8a41a4a88a64add27bd57cf56c9de2 libs/lib1 (remotes/origin/HEAD)
 7290dce0062bd77df1d83b27dd3fa3f25a836b54 libs/lib2 (heads/master)
</pre>

两个项目有两个区别：

* commit id各不相同

* **libs/lib1**所处的分支不同

#### 2.7.1 更新project1的lib1和lib2改动

我们还记得刚刚在**project2**中修改的时候把lib1和lib2都切换到了master分支，目前project1中的
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) cd
```

```
➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1  git checkout master
Previous HEAD position was 36ad12d... update lib1-features by developer B
Switched to branch 'master'
Your branch is behind 'origin/master' by 1 commit, and can be fast-forwarded.
➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1 git:(master) git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/repos/lib1
   36ad12d..8c666d8  master     -> origin/master
Updating c22aff8..8c666d8
Fast-forward
 README        |    1 +
 lib1-features |    1 +
 2 files changed, 2 insertions(+), 0 deletions(-)
 create mode 100644 README
➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1 git:(master)
</pre>

果不其然，我们看到了刚刚在project2中修改的内容，同步到了project1中，当然现在更新了**project
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib1 git:(
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   libs/lib1 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git diff
diff --git a/libs/lib1 b/libs/lib1
index 36ad12d..8c666d8 160000
--- a/libs/lib1
+++ b/libs/lib1
@@ -1 +1 @@
-Subproject commit 36ad12d40d8a41a4a88a64add27bd57cf56c9de2
+Subproject commit 8c666d86531513dd1aebdf235f142adbac72c035
</pre>

现在最新的commit id和project2目前的状态一致，说明真的同步了；好的，现在可以使用相同的办法更新
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗
➜ henryyan@hy-hp  ~/submd/ws/project1/libs/lib2 git:(master) git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/repos/lib2
   7290dce..e372b21  master     -> origin/master
Updating 7290dce..e372b21
Fast-forward
 lib2-features |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
</pre>

#### 2.7.2 更新project1的submodule引用

在**2.7.1**中我们更新了**project1**的**lib1**和**lib2**的最新版本，现在要把最新的commit
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   libs/lib1 (new commits)
#   modified:   libs/lib2 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) ✗ git commit -a -m "update l
[master 8fcca50] update lib1 and lib2 commit id to new version
```

```
 2 files changed, 2 insertions(+), 2 deletions(-)
➜ henryyan@hy-hp  ~/submd/ws/project1 git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 397 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
To /home/henryyan/submd/ws/../repos/project1.git
   c96838a..8fcca50  master -> master
</pre>
```

2.8 更新project1-b项目的子模块(使用脚本)

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/ws/../repos/project1
   c96838a..8fcca50  master     -> origin/master
Updating c96838a..8fcca50
Fast-forward
 libs/lib1 |    2 +-
 libs/lib2 |    2 +-
 2 files changed, 2 insertions(+), 2 deletions(-)
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   libs/lib1 (new commits)
#   modified:   libs/lib2 (new commits)
#
no changes added to commit (use "git add" and/or "git commit -a")
</pre>
```

Git提示lib1和lib2有更新内容，这个判断的依据来源于submodule commit id的引用。

现在怎么更新呢？难道还是像project1中那样进入子模块的目录然后**git checkout master**，接着

而且现在仅仅才两个子模块、两个项目，如果在真实的项目中使用的话可能几个到几十个不等，再加上N个su

例如笔者现在做的一个项目有**5**个web模块，每个web模块引用公共的css、js、images、jsp资源，这

工欲善其事，必先利其器，写一个脚本代替手动任务。

2.8.1 牛刀小试

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ cat /tmp/study-git-submo
 libs/lib1
 libs/lib2
</pre>
```

我们通过分析**.gitmodules**文件得出子模块的路径，然后就可以根据这些路径进行更新。

2.8.2 上路

```
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ vi bin/update-submodules
</pre>
```

把下面的脚本复制到**bin/update-submodules.sh**中:

```
<pre class="brush: shell">#!/bin/bash
grep path .gitmodules | awk '{ print $3 }' > /tmp/study-git-submodule-dirs

# read
while read LINE
do
```

```
    echo $LINE
    (cd ./$LINE && git checkout master && git pull)
done < /tmp/study-git-submodule-dirs
</pre>

稍微解释一下上面的脚本执行过程:

* 首先把子模块的路径写入到文件**/tmp/study-git-submodule-dirs**中;

* 然后读取文件中的子模块路径，依次切换到master分支(修改都是在master分支上进行的)，最后更新最

#### 2.8.3 <font color="red">2013-01-19更新</font>

网友**@紫煌**给出了更好的办法，一个命令就可以代替上面的**bin/update-submodules.sh**的功能

<pre class="brush:shell">git submodule foreach git pull
</pre>

此命令也脚本一样，循环进入（enter）每个子模块的目录，然后执行**foreach**后面的命令。

> 该后面的命令可以任意的，例如 git submodule foreach ls -l 可以列出每个子模块的文件列表

#### 2.8.3 体验工具带来的便利

<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) 
+36ad12d40d8a41a4a88a64add27bd57cf56c9de2 libs/lib1 (heads/master)
+7290dce0062bd77df1d83b27dd3fa3f25a836b54 libs/lib2 (heads/master)

# 添加执行权限
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ chmod +x ./bin/update-su
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ ./bin/update-submodules.
libs/lib1
Already on 'master'
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/repos/lib1
   36ad12d..8c666d8  master     -> origin/master
Updating 36ad12d..8c666d8
Fast-forward
 README |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 README
libs/lib2
Switched to branch 'master'
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From /home/henryyan/submd/repos/lib2
   7290dce..e372b21  master     -> origin/master
Updating 7290dce..e372b21
Fast-forward
 lib2-features |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git submodule
 8c666d86531513dd1aebdf235f142adbac72c035 libs/lib1 (heads/master)
 e372b21dffa611802c282278ec916b5418acebc2 libs/lib2 (heads/master)

 ➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   bin/
nothing added to commit but untracked files present (use "git add" to track)
</pre>

更新之后的两个变化:
```

```
* git submodule的结果和project2的submodule commit id保持一致;

* **project1-b**不再提示**new commits**了。

现在可以把工具添加到仓库了，当然你可以很骄傲的分享给其他项目组的同事。
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master)
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) ✗ git commit -m "添加自动更新
[master 756e788] 添加自动更新submodule的快捷脚本^_^
 1 files changed, 9 insertions(+), 0 deletions(-)
 create mode 100755 bin/update-submodules.sh
➜ henryyan@hy-hp  ~/submd/ws/project1-b git:(master) git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 625 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (4/4), done.
To /home/henryyan/submd/ws/../repos/project1.git
   8fcca50..756e788  master -> master
</pre>

### 2.9 新进员工加入团队，一次性Clone项目和Submodules

一般人使用的时候都是使用如下命令:
<pre class="brush: shell">git clone /path/to/repos/foo.git
git submodule init
git submodule update
</pre>

    新员工不耐烦了，嘴上不说但是心里想：怎么那么麻烦?

上面的命令简直弱暴了，直接一行命令搞定:
<pre class="brush: shell">git clone --recursive /path/to/repos/foo.git
</pre>

-**recursive**参数的含义：可以在clone项目时同时clone关联的submodules。

git help 对其解释:
<pre>--recursive, --recurse-submodules
   After the clone is created, initialize all submodules within, using their def
   submodule update --init --recursive immediately after the clone is finished.
   does not have a worktree/checkout (i.e. if any of --no-checkout/-n, --bare, c
</pre>

#### 2.9.1 使用一键方式克隆project2

<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws  git clone --recursive .
Cloning into project2-auto-clone-submodules...
done.
Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path
Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path
Cloning into libs/lib1...
done.
Submodule path 'libs/lib1': checked out '8c666d86531513dd1aebdf235f142adbac72c03
Cloning into libs/lib2...
done.
Submodule path 'libs/lib2': checked out 'e372b21dffa611802c282278ec916b5418acebc
</pre>

舒服……

## 3.移除Submodule

**牢骚**: 搞不明白为什么git不设计一个类似: git submodule remove的命令呢?

我们从project1.git克隆一个项目用来练习移除submodule:
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws  git clone --recursive .
Cloning into project1-remove-submodules...
done.
Submodule 'libs/lib1' (/home/henryyan/submd/repos/lib1.git) registered for path
Submodule 'libs/lib2' (/home/henryyan/submd/repos/lib2.git) registered for path
```

```
Cloning into libs/lib1...
done.
Submodule path 'libs/lib1': checked out '8c666d86531513dd1aebdf235f142adbac72c03
Cloning into libs/lib2...
done.
Submodule path 'libs/lib2': checked out 'e372b21dffa611802c282278ec916b5418acebc
➜ henryyan@hy-hp  ~/submd/ws  cd !$
➜ henryyan@hy-hp  ~/submd/ws  cd project1-remove-submodules
</pre>

### 3.1 Step by

1、删除git cache和物理文件夹
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodul
rm 'libs/lib1'
rm 'libs/lib2'
➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ rm -rf l
</pre>

2、删除.gitmodules的内容（或者整个文件）
因为本例只有两个子模块，直接删除文件：
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodul
</pre>

    如果仅仅删除某一个submodule那么打开.gitmodules文件编辑，删除对应submodule配置即可。

3、删除.git/config的submodule配置
源文件：
<pre>[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = /home/henryyan/submd/ws/../repos/project1.git
[branch "master"]
    remote = origin
    merge = refs/heads/master
[submodule "libs/lib1"]
    url = /home/henryyan/submd/repos/lib1.git
[submodule "libs/lib2"]
    url = /home/henryyan/submd/repos/lib2.git
</pre>

删除后：
<pre>[core]
    repositoryformatversion = 0
    filemode = true
    bare = false
    logallrefupdates = true
[remote "origin"]
    fetch = +refs/heads/*:refs/remotes/origin/*
    url = /home/henryyan/submd/ws/../repos/project1.git
[branch "master"]
    remote = origin
    merge = refs/heads/master
</pre>

4、提交更改
<pre class="brush: shell">➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodul
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   deleted:    libs/lib1
#   deleted:    libs/lib2
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
```

```
#
#   deleted:    .gitmodules
#
➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ git add
➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) ✗ git comm
[master 5e2ee71] 删除子模块lib1和lib2
 3 files changed, 0 insertions(+), 8 deletions(-)
 delete mode 100644 .gitmodules
 delete mode 160000 libs/lib1
 delete mode 160000 libs/lib2
➜ henryyan@hy-hp  ~/submd/ws/project1-remove-submodules git:(master) git push
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 302 bytes, done.
Total 2 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (2/2), done.
To /home/henryyan/submd/ws/../repos/project1.git
   756e788..5e2ee71  master -> master
</pre>

## 4.结束语

对于Git Submodule来说在刚刚接触的时候多少会有点凌乱的赶紧，尤其是没有接触过**svn:externals

本文从开始创建项目到使用git submodule的每个步骤以及细节、原理逐一讲解，希望到此读者能驾驭它。

学会了Git submdoule你的项目中再也不会出现大量重复的资源文件、公共类库，更不会出现多个版本，甚

本文的写作来源于工作中的实践，如果您对于某个做法有更好的办法还请赐教，希望能留下您宝贵的意见。

<em>原创文章，转载请注明出处！</em>
<em>[Git Submoudle使用完整教程--咖啡兔](http://www.kafeitu.me/git/2012/03/27/git-su
</file></file></file></pre></file></pre></file></file></pre></file></file></pre>
```

原创文章，转载请注明：转载自：Git Submodule使用完整教程