何もしてないのにサブモジュールの差分が出て悩んだら読む本

mochikoAsTech　著

2020-09-12 版　　mochikoAsTech 発行

はじめに

2020 年 9 月 mochikoAsTech

本書を手に取ってくださったあなた、こんにちは！ あるいは、はじめまして。「何もしてないのにサブモジュールの差分が出て悩んだら読む本」の筆者、mochikoAsTech です。

想定する読者層

本書は、こんな人に向けて書かれています。

- 仕事で使っているリポジトリにサブモジュールがいる
- なぜか git pull しただけなのにサブモジュールの差分が出て納得いかない
- git checkout して別ブランチに移動しただけなのにサブモジュールの差分がでて納得いかない
- ちゃんと git pull したのにサブモジュールが更新されなくて困っている
- サブモジュールのフォルダまで移動して git pull しただけなのに差分が出てわけが分からない

マッチしない読者層

本書は、こんな人が読むと恐らく「not for me だった…（私向けじゃなかった）」となります。

- Git について何も知らないので 1 から学びたい
- プロジェクトでサブモジュールを導入すべきか迷っていて判断材料が欲しい
- いまからリポジトリにサブモジュールを入れようとしているので方法が知りたい

本書のゴール

本書を読み終わると、あなたはこのような状態になっています。

- サブモジュールの仕組みが分かっている
- 意図せぬサブモジュールの差分が出た時に対処できる
- 読む前よりサブモジュールがちょっと好きになっている

免責事項

本書に記載されている内容は筆者の所属する組織の公式見解ではありません。

また本書はできるだけ正確を期すように努めましたが、筆者が内容を保証するものではありません。よって本書の記載内容に基づいて読者が行なった行為、及び読者が被った損害について筆者は何ら責任を負うものではありません。

不正確あるいは誤認と思われる箇所がありましたら、必要に応じて適宜改訂を行いますので GitHub の Issue や Pull request で筆者までお知らせいただけますと幸いです。

`https://github.com/mochikoAsTech/SubmoduleUpdated`

目次

第 1 章

ああサブモジュール、君のこと（挙動）が分からない

1.1 Git とは

本書は Git のサブモジュールの挙動が分からずに苦しむ人のための本なので、Git そのものについては解説しません。「Git について何も知らないので 1 から学びたい」という方には、湊川あいさんの書籍がお勧めです。

- わかばちゃんと学ぶ Git 使い方入門
  - `https://www.amazon.co.jp/dp/4863542178`

「お金をかけずにまずは無料で学びたい」という場合は、先ほどの本の元となったウェブ連載を読みましょう。

- マンガでわかる Git 第 1 話「Git ってなあに？」
  - `https://next.rikunabi.com/journal/20160526_t12_iq/`

なんと最近は Git をコマンドで使う方法を学ぶ「コマンド編」も連載されているそうです。

- マンガでわかる Git ～コマンド編～ 第 1 話「リポジトリを作ってコミットしてみよう」
  - `https://www.r-staffing.co.jp/engineer/entry/20190621_1`

Git は付け焼き刃の操作だけを学ぶよりも、どういう仕組みで、どんな理屈で動いているのかをしっかり学んだ方が、結果としては理解の速度が上がります。わかばちゃんと一緒にたくさん転んで、Git を楽しく学んでみてください。

1.2 Git のサブモジュールとは

プロジェクト A とプロジェクト B の両方で同じライブラリを使いたい！ そのライブラリをメンテナンスするのはプロジェクト C のみなさんだとします。

このときプロジェクト A のリポジトリにも、プロジェクト B のリポジトリにもライブラリのソースコードを直接置いてしまう（プロジェクトのツリーに取り込んでしまうと）と、プロジェクト C のメンバーがライブラリをメンテナンスしたいときに、両方のリポジトリでそれぞれ更新しなければなりません。

Git のサブモジュールを使うと、プロジェクト C のリポジトリを、別のリポジトリのサブディレクトリとして扱えるようになります。プロジェクト A やプロジェクト B を親（メイン）としたとき、子（サブ）にあたるプロジェクト C のコミットは、親のコミットとは別で管理できます。

1.3 サブモジュールの作り方

まず親を作ります。

```
メインのリポジトリ（main_project）を作る
$ git init main_project
```

続いてメインのリポジトリのサブモジュールとして、既に存在している別のリポジトリを追加します。

```
作ったリポジトリのディレクトリに移動する
$ cd main_project

サブモジュールとして「SubmoduleUpdated」というリポジトリを追加する
$ git submodule add https://github.com/mochikoAsTech/SubmoduleUpdated
```

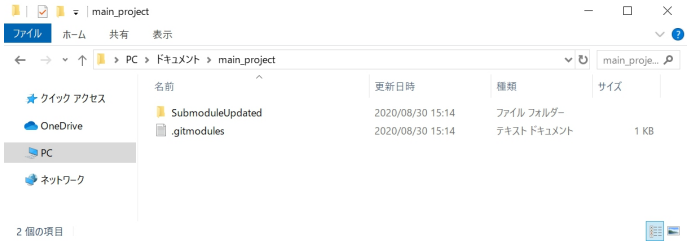
今回はサブモジュールとして、本書の原稿リポジトリを追加してみました。[*1]追加した SubmoduleUpdated をクローンしてくるため、こんな表示がされたと思いま

[*1] 今回は GitHub の URL を指定しましたが、このような絶対の URL に限らず、`git submodule add ../SubmoduleUpdated` のような相対パス、サブモジュールとしてローカルのリポジトリを指定することも可能です。ただし相対パスで追加すると、サブモジュールのリモートリポジトリの URL（`remote.origin.url`）が `C:/Users/mochikoAsTech/Documents/SubmoduleUpdated` や `../SubmoduleUpdated` のようになってしまうので、特に理由が無ければリモートの URL で指定する方がお勧めです。

す。このときサブモジュールの中身はクローンしてきますが、サブモジュールのさらにサブモジュール以下については再帰的にはクローンしてきてくれないので注意が必要です。

```
$ git submodule add https://github.com/mochikoAsTech/SubmoduleUpdated
Cloning into 'C:/Users/mochikoAsTech/Documents/main_project/SubmoduleUpdated'..
remote: Enumerating objects: 162, done.
remote: Counting objects: 100% (162/162), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 162 (delta 43), reused 26 (delta 7), pack-reused 0 eceiving objec
objects: 100% (162/162), 518.33 KiB | 772.00 KiB/s, done.
Resolving deltas: 100% (43/43), done.
```

サブモジュールを追加するとき、先ほどのように特にディレクトリ名を指定しないと、サブモジュールのリポジトリ名（SubmoduleUpdated）がそのままディレクトリ名となります。（図 1.1）



▲図 1.1　SubmoduleUpdated がサブモジュールとして追加された

ディレクトリ名を変えたいときは、次のように末尾でディレクトリ名を指定してやれば、たとえばディレクトリ名を「sub」にした状態でサブモジュールを追加できます。

```
git submodule add https://github.com/mochikoAsTech/SubmoduleUpdated submodule
```

サブモジュールを追加してどうなったのか、git status でメインリポジトリの状態を確認してみましょう。

```
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitmodules
        new file:   SubmoduleUpdated
```

.gitmodules と SubmoduleUpdated の 2 つが、新しいファイルとして認識されています。

サブモジュールを追加すると、このように親のリポジトリに.gitmodules というファイルが生まれます。これはテキスト形式の設定ファイルで、テキストエディタで開くとこんなふうにサブモジュールのディレクトリパスと、リモートの URL が書かれています。サブモジュールを複数追加すると、このファイルにその分いくつも追記されていきます。

```
[submodule "SubmoduleUpdated"]
        path = SubmoduleUpdated
        url = https://github.com/mochikoAsTech/SubmoduleUpdated
```

他の人がメインリポジトリをクローンすると、このファイルに書かれた内容を元に、サブモジュールの取得元を把握することになります。

続いて git diff で SubmoduleUpdated を見てみましょう。コミット前のファイルの差分を見たいので、--cached オプションを付ける必要があります。

変更前の--- /dev/null は、このファイルが新たに作られたものであることを表しています。

```
$ git diff --cached SubmoduleUpdated
diff --git a/SubmoduleUpdated b/SubmoduleUpdated
new file mode 160000
index 0000000..6f47087
--- /dev/null
+++ b/SubmoduleUpdated
@@ -0,0 +1 @@
+Subproject commit 6f47087f1c9079ea6c677702da23ca040d0a13ed
```

実際は SubmoduleUpdated はディレクトリであり、その中にはたくさんの原稿ファイルがあります。ですがメインのリポジトリからは、サブモジュールの中身を 1 つ 1 つ追跡するようなことはおこないません。代わりにこのサブディレクトリを、親

から見た「子の年齢」のような +Subproject commit 6f47087f1c9079ea6c67770 2da23ca040d0a13ed という 1 つのコミットとして記録していることが分かります。

1.4 サブモジュールを含むリポジトリをクローンしてこよう

サブモジュールを含むメインのリポジトリをクローンすると、最初は「サブモジュールが入っているはずのディレクトリ」は取得できるのですが、その中身は空っぽです。

```
ローカルの設定ファイルを初期化する
$ git submodule init

親から見た年齢（コミット）のサブモジュールを連れてくる
$ git submodule update
```

1.5 最初からサブモジュールの中身も含めて全部連れてきたかった

最初からサブモジュールの中身も含めて全部まるっと持ってきたい場合は、--recursive オプションを付けてクローンしてきます。

```
サブモジュールも含めて全部まるっとクローンしてくる
$ git clone --recursive https://github.com/mochikoAsTech/SubmoduleUpdated
```

1.6 親子の関係

Git のメインリポジトリを親、サブモジュールを子だとします。

サブモジュールのリポジトリではコミットが「A → B → C」で進んでいて、けれど親のリポジトリではまだ B を参照しているとします。この場合、親のリポジトリで git submodule update したら、サブモジュールは B を引っ張ってきます。

git submodule update とは、「子のサブモジュールを最新にしておくれ」ではなく、「"親が指定している子のコミット"に更新しておくれ」なのです。

1.7 git pull しただけなのに差分が出た

1.8 git checkout しただけなのに差分が出た

1.9 サブモジュールで git pull しただけなのに差分が出た

1.10 正しい差分の無くし方

あとがき

この原稿を書いたことで、「prh-rules（校正のルールを書いたファイルたち）をサブモジュールにすれば、本のリポジトリを作るたびに前回のリポジトリから prh-rules 持ってなくてもいいんじゃん！」と気づいてしまったので、本当に本を書くのは自分にとっていいことだなぁ、と思いました。まる。

数ある技術書の中から「Submodule 本（仮題）」を手に取ってくださったあなたに感謝します。

2020 年 9 月
mochikoAsTech

PDF 版のダウンロード

本書はどなたでも PDF 版を無料ダウンロードできます。ぜひあなたの周りにいるサブモジュールでお悩みの方に知らせてあげてください。

- ダウンロード URL
  - `https://mochikoastech.booth.pm/items/xxxxxxx`

Special Thanks:

- ねこ

レビュアー

- かしこいねこ

参考文献・ウェブサイト

- git-scm.com
  - `https://git-scm.com/`

著者紹介

mochiko / @mochikoAsTech

テクニカルライター。元 Web 制作会社のインフラエンジニア。ねこが好き。「分からない気持ち」に寄り添える技術者になれるように日々奮闘中。技術書典で頒布した「DNS をはじめよう」「AWS をはじめよう」「技術をつたえるテクニック」「技術同人誌を書いたあなたへ」は累計で 9,700 冊を突破。

- https://twitter.com/mochikoAsTech
- https://mochikoastech.booth.pm/
- https://note.mu/mochikoastech
- https://mochikoastech.hatenablog.com/
- https://www.amazon.co.jp/mochikoAsTech/e/B087NBL9VM

Hikaru Wakamatsu

表紙デザインを担当。

Shinya Nagashio

挿絵デザインを担当。

何もしてないのにサブモジュールの差分が出て悩んだら読む本

2020 年 9 月 12 日　技術書典 9 初版

著　者　　mochikoAsTech
デザイン　Hikaru Wakamatsu / Shinya Nagashio
発行所　　mochikoAsTech
印刷所　　日光企画