第 07 天：解析 Git 資料結構 - 索引結構

我們知道在 Git 裡兩個重要的資料結構，分別是「物件」與「索引」，這篇文章主要用來解說「索引」的細節。使用 Git 版本控管的過程中，或許你可以很輕易的了解 git 指令的用法，不過那很容易流於死記，無法靈活運用，連 Linus Torvalds 都在郵件清單(Mailing List)中提到：「在使用者了解索引的意義之前，是無法完整了解 Git 的能力的」，因此，了解「索引」的用途十分重要。

關於索引

簡單來說，「索引」的目的主要用來紀錄「有哪些檔案即將要被提交到下一個 commit 版本中」。

換句話說，「如果你想要提交一個版本到 Git 儲存庫，那麼你一定要先更新索引狀態，變更才會被提交出去。」

這裡的「索引」其實在國外很多文章裡曾經出現過很多別名，但其意思都是相同的，各位以後看到相關單字千萬不要被混淆了。

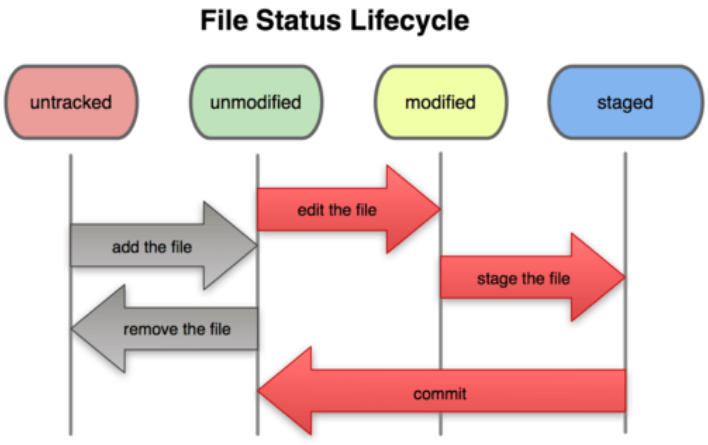
- Index (索引)
- Cache (快取)
- Directory cache (目錄快取)
- Current directory cache (當前目錄快取)
- Staging area (等待被 commit 的地方)
- Staged files (等待被 commit 的檔案)

舉個例子來說，指令 `git diff --cached` 就與 `git diff --staged` 是完全同義的。

操作索引的指令

由於「索引」對 Git 來說十分重要，在大多數的指令中都會有跟 Git 索引相關的參數可用，不過我們大致列出幾個直接與「索引」相關的指令來解說。

在解說指令之前，各位可以先看看以下示意圖，這說明了透過指令改變狀態的生命週期，事實上，這些改變的過程，都是在更新「索引檔」的過程：



首先，先介紹四種檔案狀態：

- untracked (未追蹤的，代表尚未被加入 Git 儲存庫的檔案狀態)
- unmodified (未修改的，代表檔案第一次被加入，或是檔案內容與 HEAD 內容一致的狀態)
- modified (已修改的，代表檔案已經被編輯過，或是檔案內容與 HEAD 內容不一致的狀態)
- staged (等待被 commit 的，代表下次執行 git commit 會將這些檔案全部送入版本庫)

git status

取得 **工作目錄** (working tree) 下的狀態。

由於先前已經講過儲存庫、工作目錄、物件與索引之間的關係，我們用一句話說明這關係：

> `Git` 儲存庫的運作，是將工作目錄裡的變化，透過更新索引的方式，將資料寫入成 `Git` 物件。

這裡的 `git status` 指令，目的是顯示出 **目前最新版** 與 **索引檔** 之間的差異，這當中的差異包含了一些微妙的關係，我們用一個例子來解釋這層關係。

以下是執行 git status 的結果：

```
G:\git-demo>git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   c.txt
#
# 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:   a.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       b.txt
```

這裡你會看到有三種不同的分組，分別是：

- Changes to be committed (準備提交的變更)
  - 這區有個 `new file: c.txt` 檔案，代表 c.txt 是一個新檔案，而且已經被標示可提交。
  - 這代表著幾件事：
    a. **目前最新版** 並沒有 c.txt 這個檔案
    b. **索引檔** 已經加入了這個 c.txt 檔案
    c. 所以該檔案會在執行 git commit 之後被存入下一個版本
- Changes not staged for commit (尚未準備提交的變更)
  - 這區有個 `modified: a.txt` 檔案，代表 a.txt 已經被變更，但尚未標示可提交。 (not staged)
  - 這代表著幾件事：
    a. **目前最新版** 也有 a.txt 這個檔案
    b. **索引檔** 尚未加入 a.txt 這個檔案
    c. 所以該檔案就算執行了 git commit 也不會在下一版中出現
- Untracked files (未追蹤的變更)
  - 這區有個 `b.txt` 檔案，代表 b.txt 尚未被追蹤。(untracked)
  - 這代表著幾件事：
    a. **目前最新版** 沒有 b.txt 這個檔案
    b. **索引檔** 也沒有 b.txt 這個檔案
    c. 所以該檔案就算執行了 git commit 也不會在下一版中出現

所以你可以看到，執行 git status 就是為了查出 **目前最新版** 與 **索引檔** 之間的差異，最終只有 **目前最新版** 與 **索引檔** 之間有差異的變更，才會真正儲存到下一個 commit 物件裡。

git add

`git add` 指令，是為了將目前「工作目錄」的變更寫入到「索引檔」裡。

使用 `git add -u` 則可以僅將「更新」或「刪除」的檔案變更寫入到「索引檔」中。

git rm

我們以 `git rm` 為例，當你直接在檔案系統中刪除一個檔案，這只是從「工作目錄」中刪除而已，並沒有更新到索引檔，你可以利用 git status 看到這層改變，不過若要真正把「刪除」的狀態寫進索引檔的話，則要靠 `git rm filename` 更新索引檔。

在執行 `git rm filename` 的時候，除了更新索引檔之外，連工作目錄下的檔案也會一併被刪除。若你只想刪除索引檔中的該檔，又要保留工作目錄下的實體檔案，那麼你可以在指令列加上 `--cached` 參數，就能做到，例如：

```
git rm --cached a.txt
```

git mv

使用 `git mv oldname newname` 可以將檔案更名，執行此命令會同時更新索引與變更工作目錄下的實體檔案。

git commit

這個指令，則是把「索引檔」與「目前最新版」中的資料比對出差異，然後把差異部分提交變更成一個 commit 物件。

git ls-files

在索引檔之中，預設就包含了 **目前最新版** 的所有檔案，外加你在工作目錄中新增檔案且透過 `git add` 更新索引檔後的那些檔案。透過 `git ls-files` 命令，可以列出所有目前已經儲存在「索引檔」中的那些檔案路徑。

從如下圖範例，你應該可以看出這幾個指令之間的關係：

```
G:\git-demo>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:   a.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       b.txt
#       c.txt
no changes added to commit (use "git add" and/or "git commit -a")

G:\git-demo>git ls-files
a.txt

G:\git-demo>git add .

G:\git-demo>git ls-files
a.txt
b.txt
c.txt

G:\git-demo>git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   a.txt
#       new file:   b.txt
#       new file:   c.txt
#
```

今日小結

Git 裡的「索引」是 Git 版控中最重要的觀念，有了這層觀念，也自然能得知，為什麼每次提交變更都要打一些指令把變更給加進去。當然，也有許多好用的 GUI 工具可以幫你少打許多指令，不過在我們正式開始使用 Git 的 GUI 工具之前，我們還是多靠指令把觀念給建立再說吧！

參考連結

- What is the deal with the Git Index? What is the Git Index? - GitGuys
- Git - Recording Changes to the Repository

- Pro Git Book
- Git Magic - 繁體中文版
- Git (software) - Wikipedia, the free encyclopedia

- HOME
- 回目錄
- 前一天：解析 Git 資料結構 - 物件結構
- 下一天：關於分支的基本觀念與使用方式