Show

第 05 天：了解儲存庫、工作目錄、物件與索引之間的關係

在使用 Git 版本控管的過程中，有些很基本的觀念必須被建立，這樣才能更有效率也更有意義的學下去。有清楚且正確的觀念支持，不但有助於你學習 Git 指令操作，更重要的是，學習 Git 相關知識也會更加上手。

了解儲存庫

我們要使用 Git 進行版本控管，很自然的，我們需要一個「版本庫」來儲存這些版本資訊，而英文的 Repository 就是這個意思，筆者習慣將這個英文翻譯成「儲存庫」，代表用來儲存所有版本的一個空間或一個資料夾與一堆檔案。

如果有 Git 使用經驗的人，應該很清楚，建立儲存庫有很多方法，如果你要在任意一個資料夾建立一個 Git 儲存庫，只要輸入以下指令就可以建立完成：

```
git init
```

我們透過下圖建立 Git 儲存庫的過程來說明，透過這張圖我們可以很清楚的知道，當我們在 G:\git-demo 目錄下執行 git init 之後，Git 會自動幫我們建立一個所謂的 Git repository 在該目錄的 .git 目錄下，各位不用懷疑，這個 .git 資料夾，就是一個完整的 Git 儲存庫，未來所有版本的變更，都會自動儲存在這個資料夾裡面。

了解工作目錄

在上述這個例子裡，目錄 G:\git-demo 此時就會自動成為我們的「工作目錄」 (working directory)。所謂「工作目錄」的意思，就是我們正在準備開發的專案檔案，未來都會在這個目錄下進行編輯，無論是新增檔案、修改檔案、刪除檔案、檔案更名、...以及所有其他 Git 相關的操作，都會在這個目錄下完成，所以才稱為「工作目錄」。

我們在操作 Git 相關指令參數的時候，也通常都是在「工作目錄」下執行的。

由於在使用 Git 版本控管時，會遭遇到很多分支的狀況，所以工作目錄很有可能會在不同的分支之間進行切換，有些 git 指令在執行的時候，會一併更新工作目錄下的檔案。例如當你使用 git checkout 切換到不同分支時，由於目前分支與想要切換過去的分支的目錄結構不太一樣，所以很有可能會將你目前工作目錄下的檔案進行更新，好讓目前的工作目錄下的這些目錄與檔案，都與另一個要切換過去的分支下的目錄與檔案一樣。

所以，適時的保持工作目錄的乾淨，是版本控管過程中的一個基本原則，更尤其是日後要進行合併的時候，這點尤其重要，相關知識我會在日後的文章中進一步說明。

了解 Git 的資料結構

在 Git 裡有兩個重要的資料結構，分別是「物件」與「索引」。

「物件」用來保存版本庫中所有檔案與版本紀錄，「索引」則是用來保存當下要進版本庫之前的目錄狀態。

關於物件

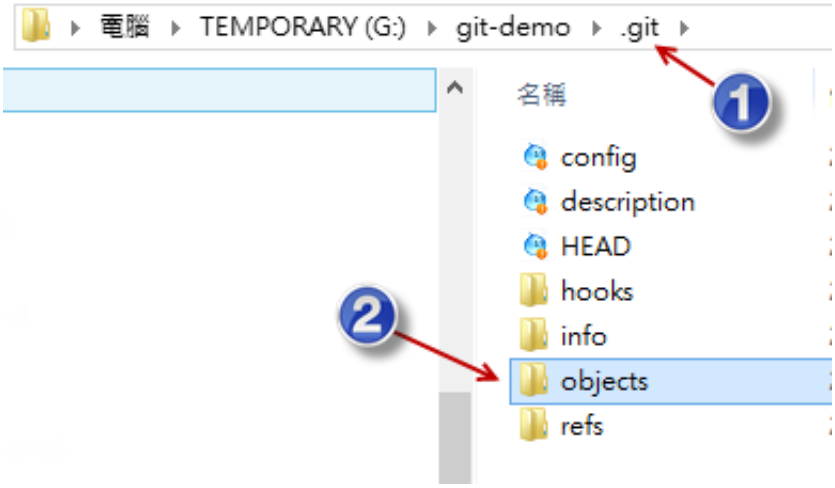
所謂的「物件」是一個「特別的檔案」，該檔案的產生過程很有趣，是將一個檔案的內容中取出，透過內容產生一組 SHA1 雜湊值，然後依照這個 SHA1 雜湊值命名的一個檔案。

在使用 Git 進行版本控管的過程中，所有要進行控管的目錄與檔案，都會先區分「目錄資訊」與「檔案內容」，我們稱為 tree 物件與 blob 物件。

其中 blob 物件就是把原本的「檔案內容」當成 blob 檔案的內容 (注意: blob 物件其實就是一個實體檔案)，然後再將其內容進行 SHA1 雜湊運算後產生的一個 hash id，再把這個 hash id 當成 blob 檔案的檔名。由此可知，blob 物件是一個「只有內容」的檔案，其檔名又是由內容產生的，所以，任何一的單獨存在的 blob 檔案通常對版本控管沒有任何幫助。

另一個 tree 物件，則是用來儲存特定資料夾下包含哪些檔案，以及該檔案對應的 blob 物件的檔名為何。在 tree 物件中，除了可以包含 blob 物件的檔名與相關資訊，還可以包含其他的 tree 物件。所以 tree 物件其實就是「資料夾」的代名詞。

無論 blob 物件與 tree 物件，這些都算是物件，這些物件都會儲存在一個所謂的「物件儲存區」(object storage) 之中，而這個「物件儲存區」預設就在「儲存庫」的 objects 目錄下，如下圖示：



詳細的物件結構，我們會在接下來的文章談到。

關於索引

所謂的「索引」是一個經常異動的暫存檔，這個檔案通常位於 .git 目錄下的一位名為 index 的檔案。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。

這個索引檔，通常保存著 Git 儲存庫中特定版本的狀態，這個狀態可以由任意一個 commit 物件，以及 tree 物件所表示。

我們通常不會直接去編輯 .git\index 這個二進位檔，而是透過標準的 git 指令去操作這個索引檔，對於索引檔的操作指令大概有以下幾個：

- git add
- git mv
- git rm
- git status
- git commit
- git ls-files

Git 的「索引」是一個介於「物件儲存區」 (object storage) 與「工作目錄」 (working directory) 之間的媒介。

各位也許已經可以猜到，本篇文章想闡述的這幾個觀念之間的關係，可以用以下 5 個步驟解釋：

- 要使用 Git 版本控管，你必須先建立「工作目錄」與「版本庫」。(mkdir, git init)
- 你要先在「工作目錄」進行開發，你可能會建立目錄、建立檔案、修改檔案、刪除檔案、... 等操作。
- 然後當你想提交一個新版本到 Git 的「儲存庫」裡，一定要先更新「索引」狀態。(git add, git mv, ...)
- 然後 Git 會依據「索引」當下的狀態，決定要把那些檔案提交到 Git 的「儲存庫」裡。(git status)
- 最後提交變更時 (git commit)，才會把版本資訊寫入到「物件儲存區」當中 (此時將會寫入 commit 物件)。

詳細的索引結構與指令操作，我們會在接下來的文章談到。

> 註: 由於 `tree` 的概念跟 `directory` 很像，所以在看國外原文時，`working directory` 也經常被寫成 `working tree`！

今日小結

今天探討的 Git 架構，最重要的還是在「物件」與「索引」之間的關係，因為沒有「索引」資訊，Git 就無法建立版本。

而基於「物件」與「索引」的差異，你應該可以發現，「物件」是屬於一種「不可變的」 (immutable) 檔案類型，任何寫入到「物件儲存區」的物件，原則上都不會再發生異動，因為所有的物件都是從原本的檔案內容產生的。我們也可以說這是一個「物件資料庫」 (object database)，且這個資料庫通常只會增加內容，比較不會有「刪除內容」或「異動內容」的情況，只有在執行 git gc 清除垃圾資料時才會刪除資料。「索引」則是屬於一種「可變的」 (mutable) 索引檔，用來記錄目前工作目錄準備要 commit 的內容。

當你一步一步的接近 Git 核心，慢慢地將模糊不清的抽象概念，轉變成具象的觀念知識，你就不會再對 Git 感到不安，請繼續努力學習，成功就在前方。

參考連結

- Git Internals - Git Objects (http://git-scm.com/book/en/Git-Internals-Git-Objects)
- Pro Git Book (http://progit.org/)
- Git Magic - 繁體中文版 (http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/zh_tw/)
- Git (software) - Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Git_(software))