

Nidify Subversion 使用手冊

中原資管 萬文傑 編寫

目 錄

| | |
|--|----|
| 第一章 Subversion簡介 | 2 |
| 第一節、什麼是Subversion | 2 |
| 第二節、安裝用戶端：TortoiseSVN..... | 2 |
| 第二章 使用Subversion..... | 3 |
| 第一節、使用TortoiseSVN 連線Subversion. | 3 |
| 第二節、Import 專案至Subversion | 4 |
| 第三節、上傳專案至Subversion | 5 |
| 第四節、執行Checkout 將Server檔案複製到本機並產生關連檔..... | 11 |
| 第五節、SVN Update | 13 |
| 第六節、檔案修改與更新 | 15 |
| 第七節、版本回溯 | 18 |
| 第八節、Clean up command | 20 |
| 第三章、專案的目錄結構 | 21 |
| 第四章、解決衝突 (合併他人的更動)..... | 26 |

第一章 Subversion 簡介

第一節、什麼是 Subversion

Subversion是一套使用開源碼開發的版本控制系統，它的設計目標就是取代目前的CVS。近年來由於資訊技術快速發展以致程式碼規模日亦複雜，舊的CVS 系統已經開始顯露疲態。相對於的RCS及CVS， Subversion採用了分支管理系統來接替 CVS。設計者以兩個方法來贏得 CVS 使用者的心，首先產生一個設計與外觀都類似 CVS 的開源碼系統，進而再修正 CVS 中最廣為人知的缺點。雖然不是版本控制設計的偉大變革，但是 Subversion 絕對是個強力可用性高而且深具彈性的工具，近來新建置的版本控制多已改用此套系統。

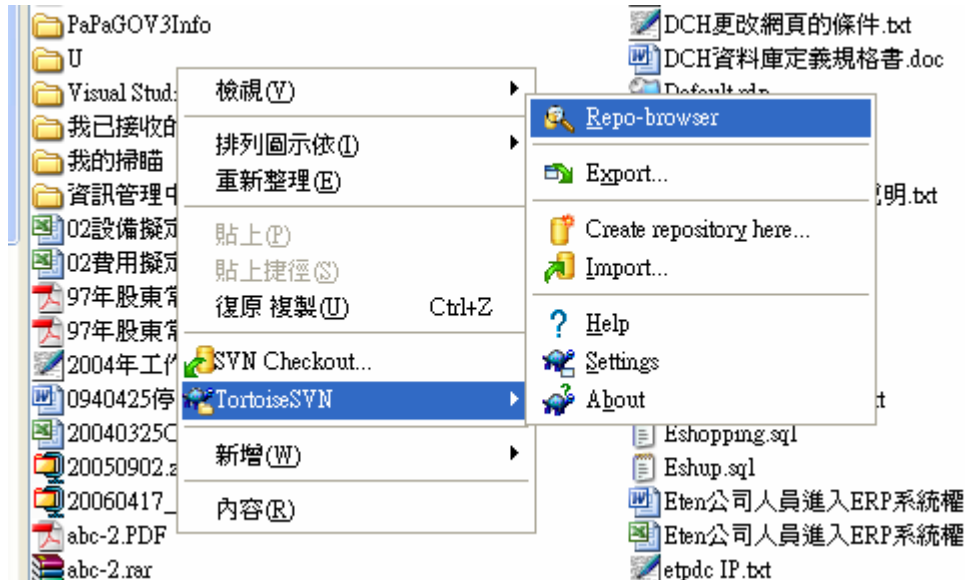
第二節、安裝用戶端：TortoiseSVN

點選[TortoiseSVN](#) 下載 TortoiseSVN client (依照CPU類別選擇 32 位元或 64 位元的msi安裝檔)安裝於使用者端。

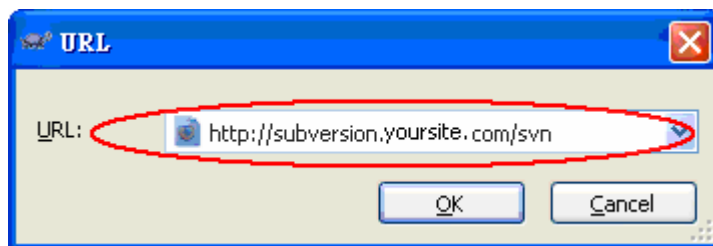
第二章 使用 Subversion

第一節、使用 TortoiseVNS 連線 Subversion.

2.1.1、檔案總管任何地方=>右鍵 =>TortoisSVN => Repo-browser



2.1.2、於跳出視窗中輸入 `http://subversion.yoursite.com/svn` (若無網域名稱可直接使用 IP)



2.1.3、連線成功出現詢問密碼視窗



2.1.4、登入成功之畫面，表示您已可以使用 Subversion

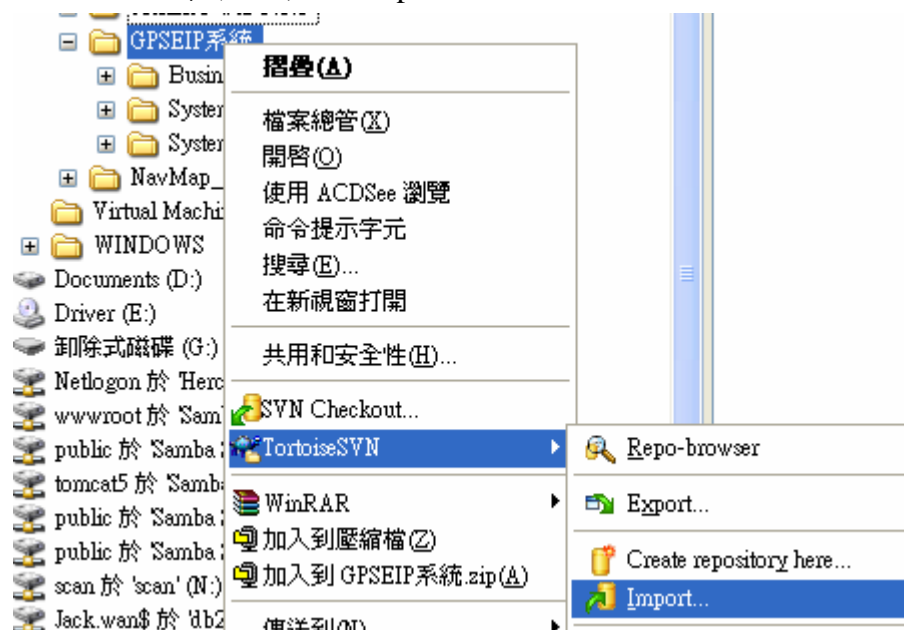


Revision 17: /

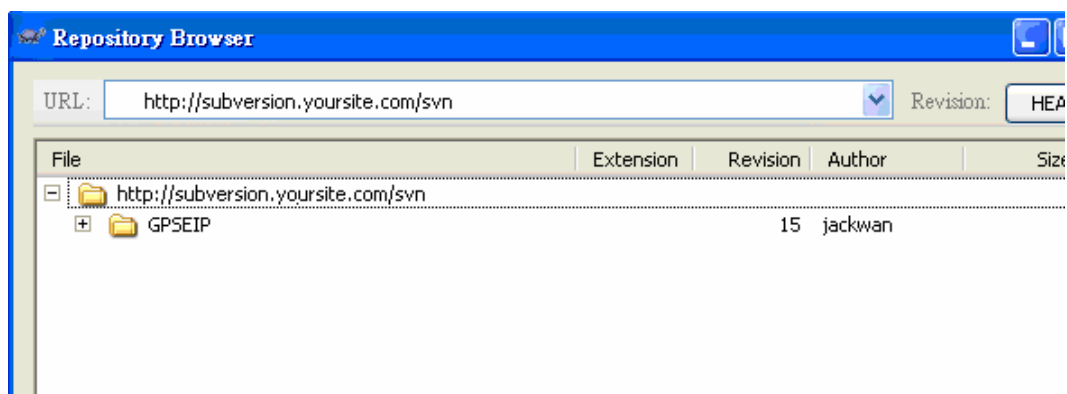
Powered by [Subversion](#) version 1.4.2 (r22196).

第二節、Import 專案至 Subversion

2.2.1、右鍵專案目錄 => Import



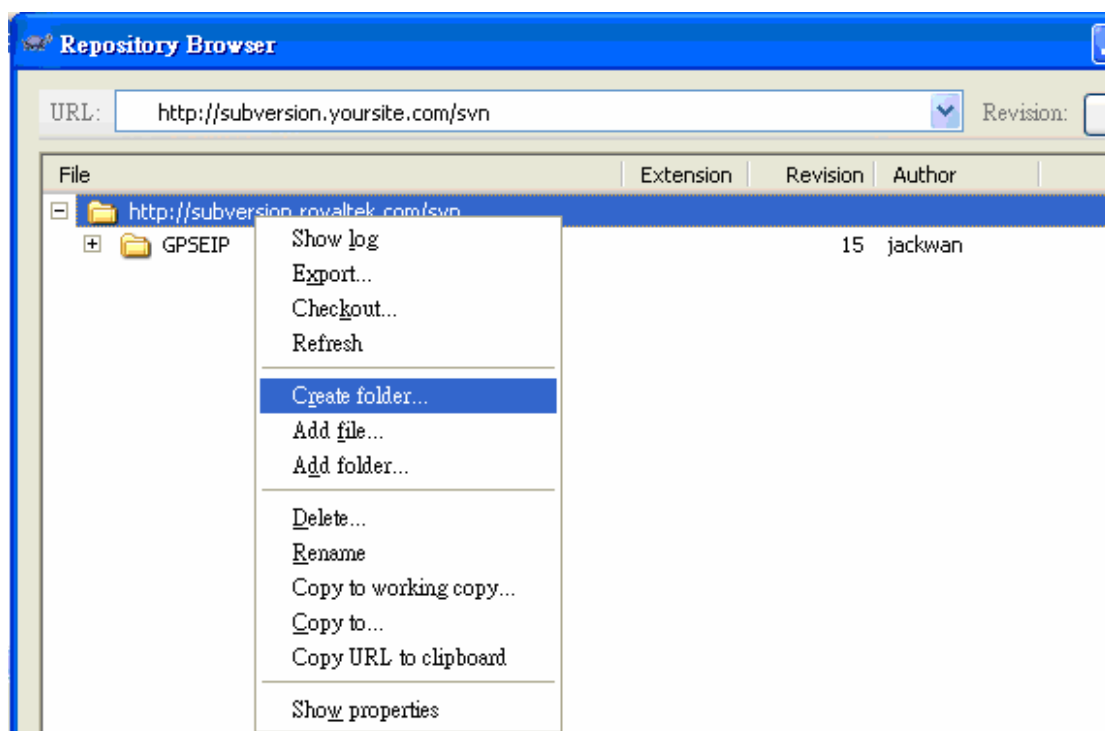
2.2.2、專案上傳完成。



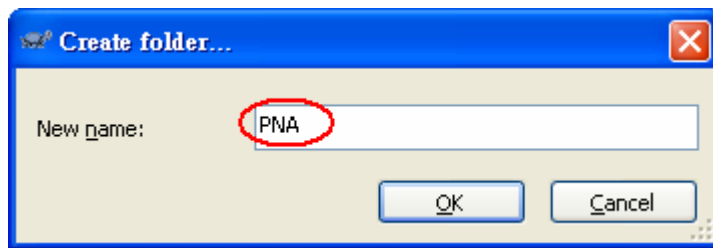
提示：使用 import 專案方式其下階目錄未必符合 Subversion 目錄的公開標準，必須再手動更改目錄名稱，建議使用第三節檔案上傳方式建立專案 (Subversion 的目錄結構詳見第三章)。

第三節、上傳專案至 Subversion

2.3.1、進入 Subversion 環境後 右鍵=> Create folder，建立專案目錄如圖。



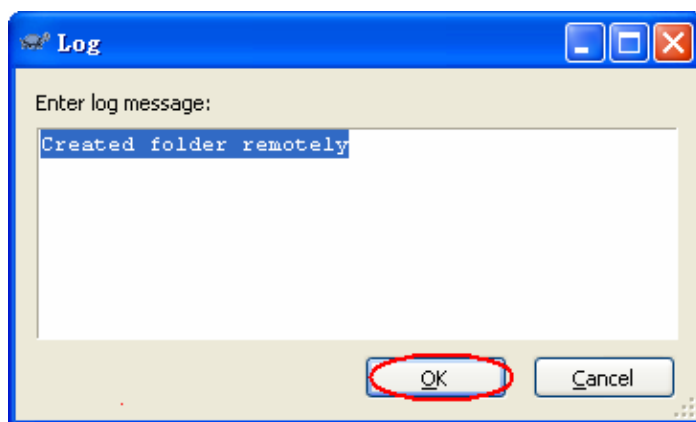
2.3.2、輸入專案名稱，第一層目錄必須以專案為名(exp: PNA)。



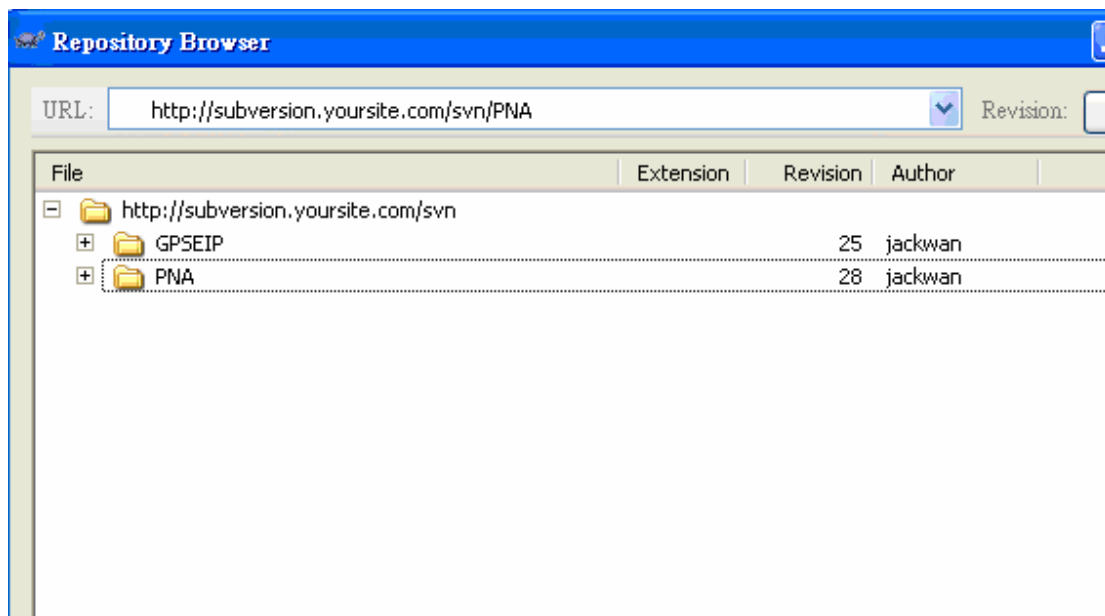
提示

目錄名稱請不要使用中文，這樣比較不容易出問題。

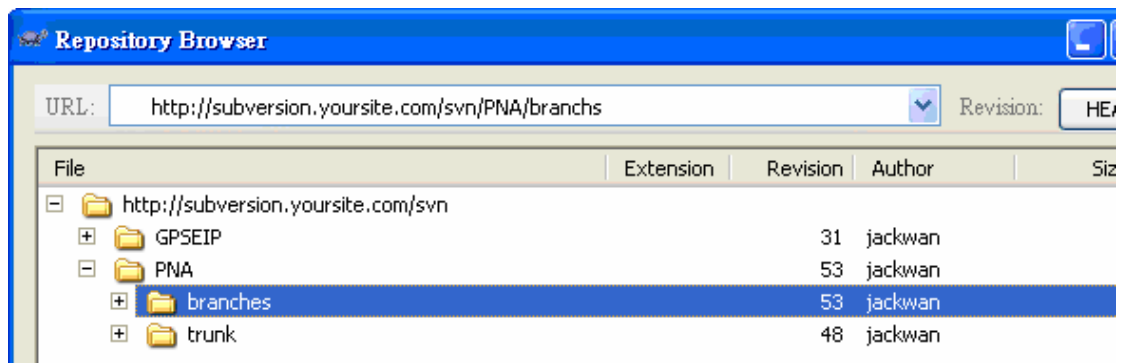
2.3.3、點 OK



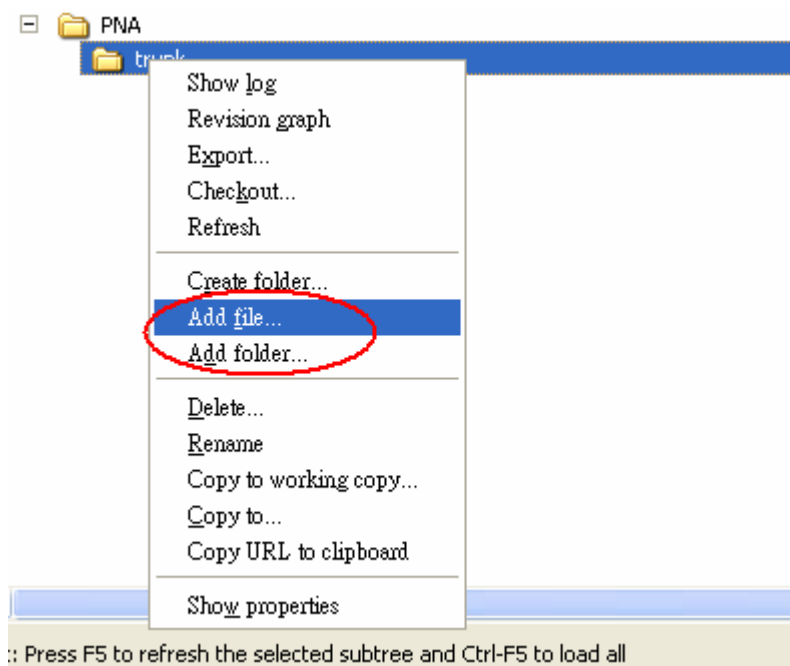
2.3.4、專案目錄建立完成如圖。



2.3.5、接著以同樣方式在 PNA 專案下建立主幹目錄 **trunk** 及分支目錄 **branches**。



2.3.6、上傳檔案到 PNA 目錄內(可選擇單一檔案上傳或是整個目錄上傳)如圖。

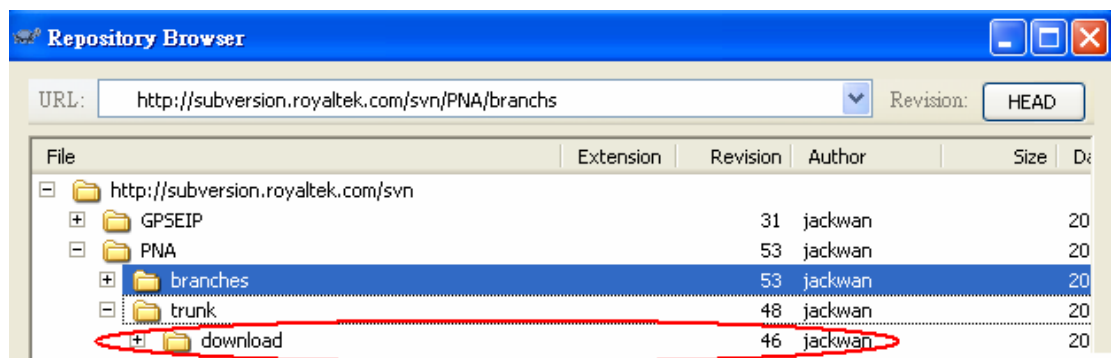


2.3.6.1、上傳目錄

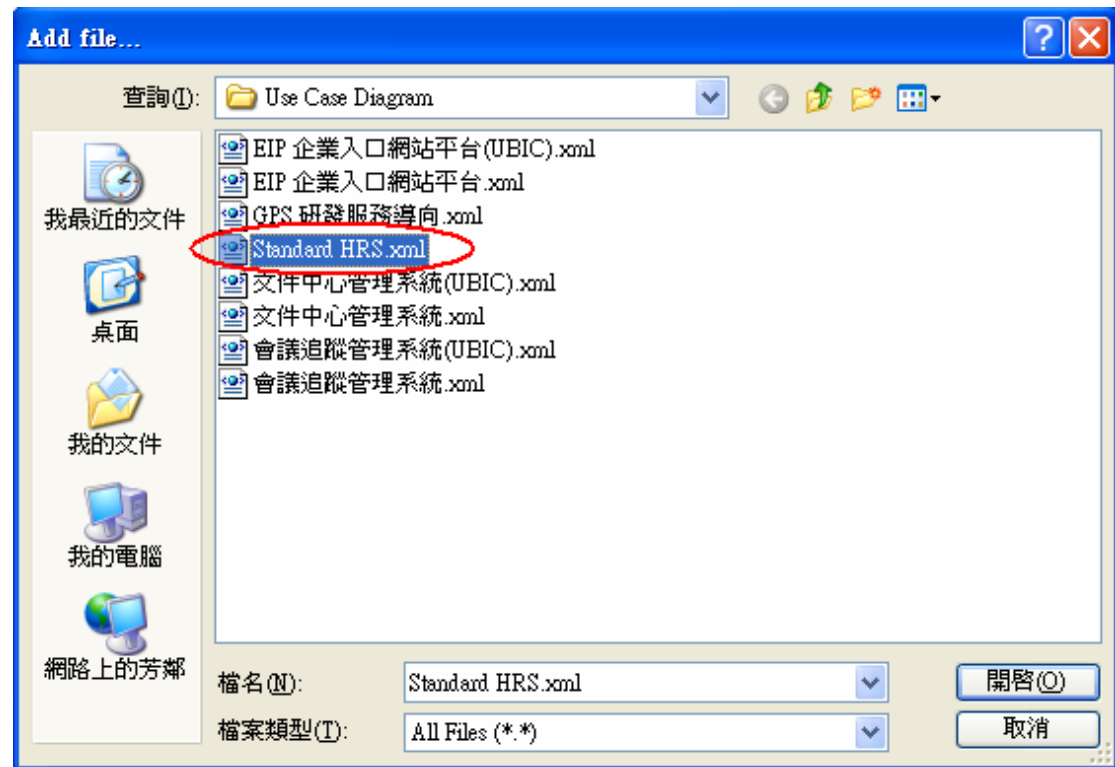
請選擇本機欲上傳之本目錄 => 確定。



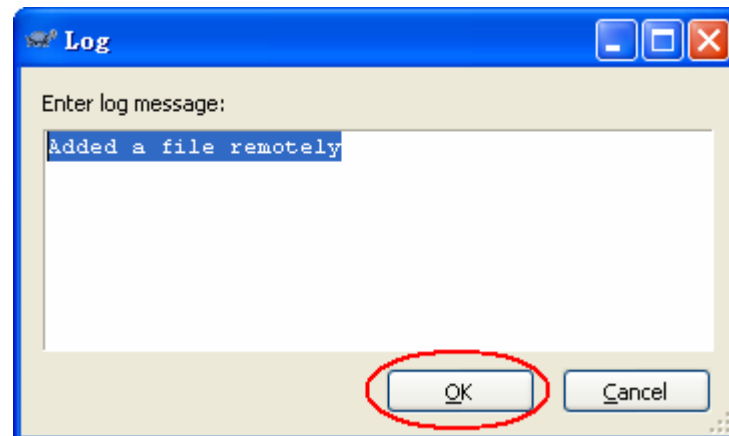
2.3.6.1.1、目錄上傳完成如圖。



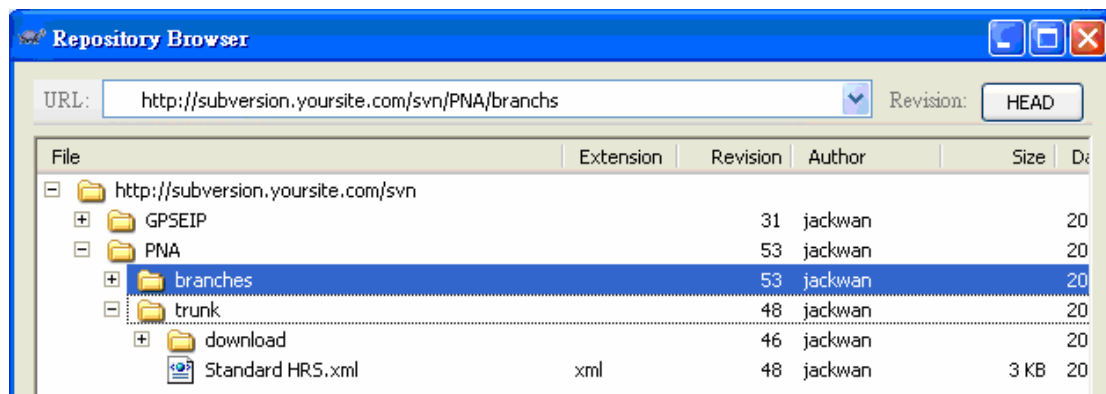
2.3.6.2、單一檔案上傳模式，
選擇本機之檔案=>開啟。



2.3.6.2.1、點選 OK 遠端上傳。



2.3.6.2.2、Subversion 正常顯示表示檔案上傳完成。



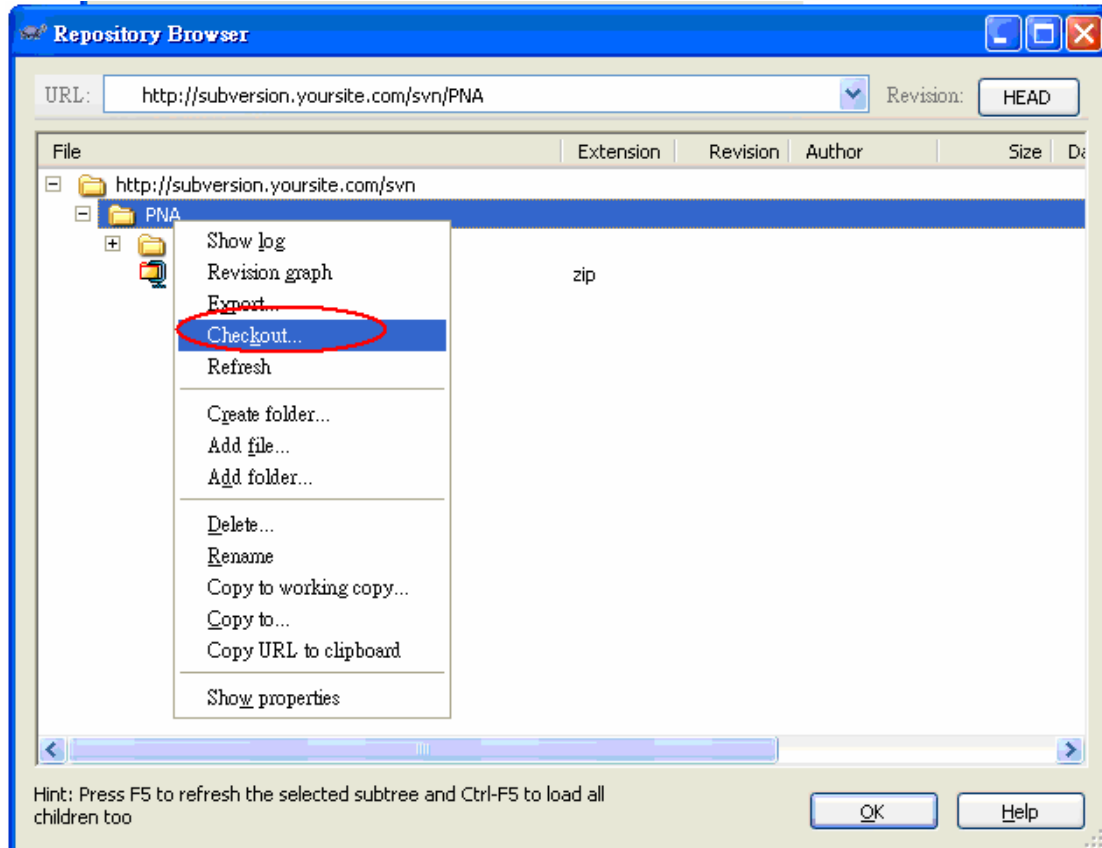
提示

檔名請第一次就決定好不可重複檔名，有些作業系統不區分檔名的大小寫，如果日後要改檔名，而且新的檔名和舊的檔名只有大小寫的不同，例如：myprg.java 和 MyPrg.java，在送交和更新時可能會碰到一些麻煩（檔名請勿使用中文）。

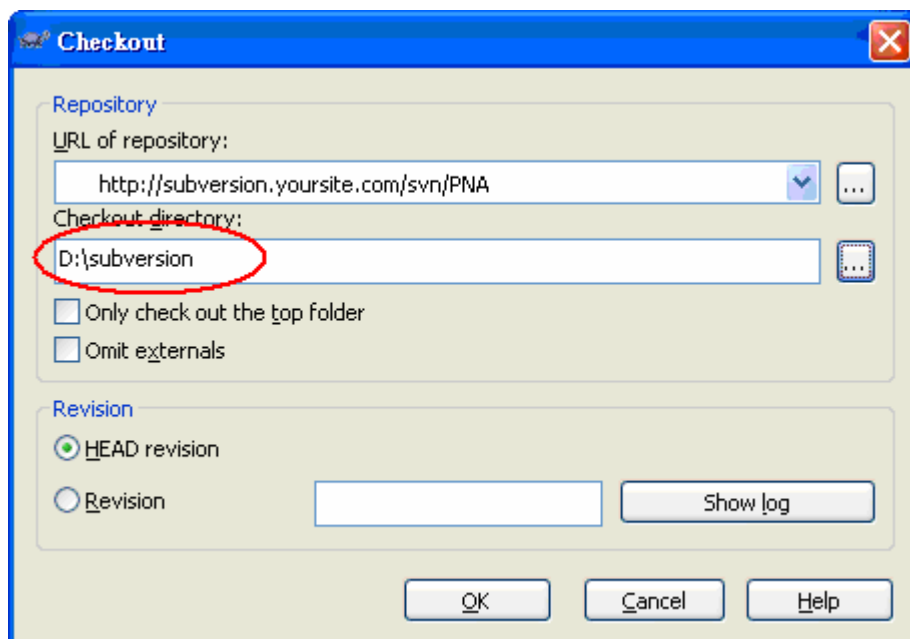
第四節、執行 Checkout 將 Server 檔案複製到本機並產生關連檔

如前述檔案雖然已上傳完成，但是那些檔案並未本機電腦產生關連，必須在 Server 上執行 Checkout 建立與本電腦的關連性。

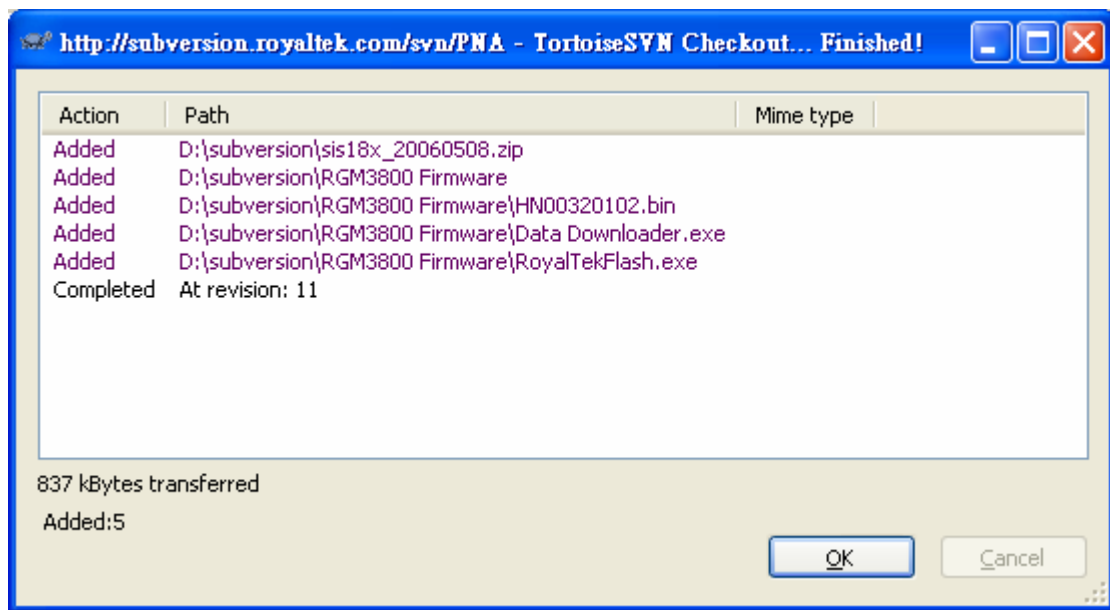
2.4.1、右鍵選擇目錄 => Checkout



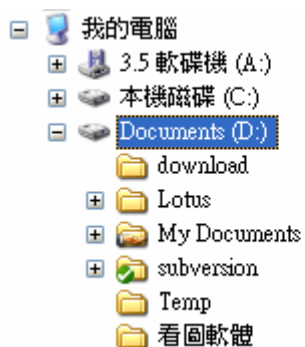
2.4.2、選擇與 Subversion 連結之本區目錄 => OK (請先建立 D:\Subversion 目錄)。



2.4.3、檔案 Checkout 完成。



2.4.4、在檔案總管檢視 D:\Subversion 中有綠色的🟢表示檔案下載完成並且與 Server 連結成功，目錄內可發現自動產生的.svn 子目錄。



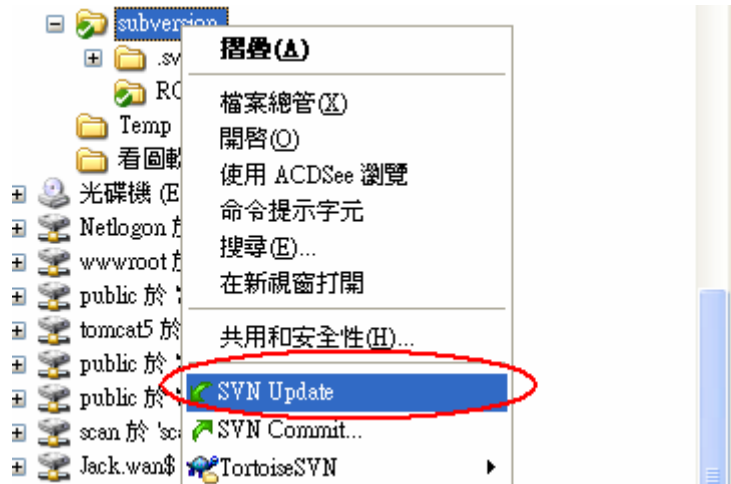
提示

.svn 目錄有什麼作用？

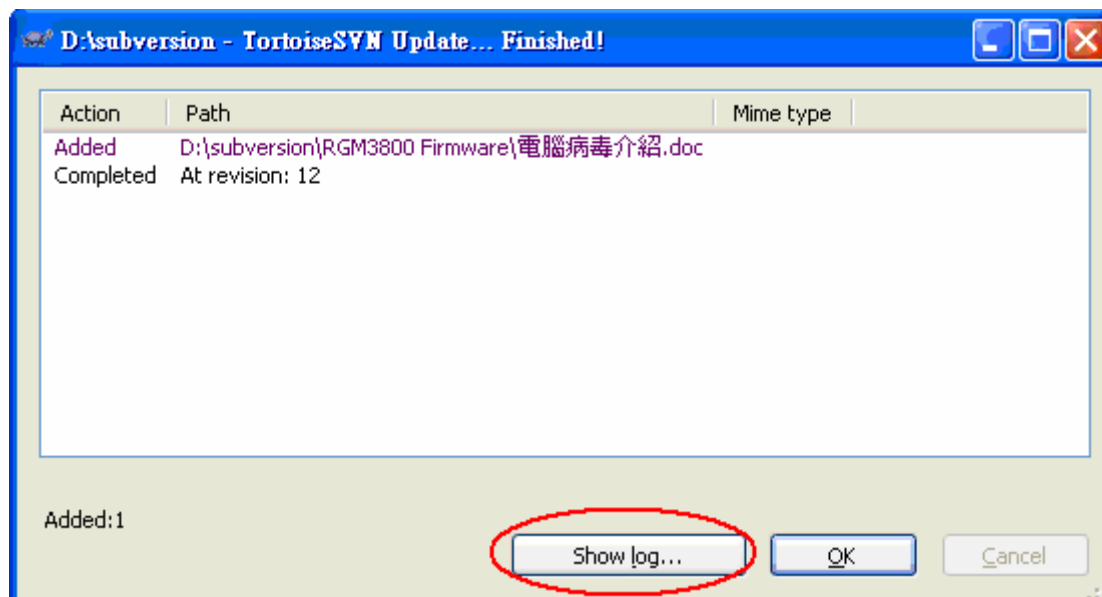
每一個工作複本的目錄都有一個名為 .svn 的子目錄(通常DOS指令不會顯示這個子目錄)。它是一個相當重要的目錄，千萬別刪除或是修改.svn裡的資料！Subversion 就是靠它來管理你的工作複本。

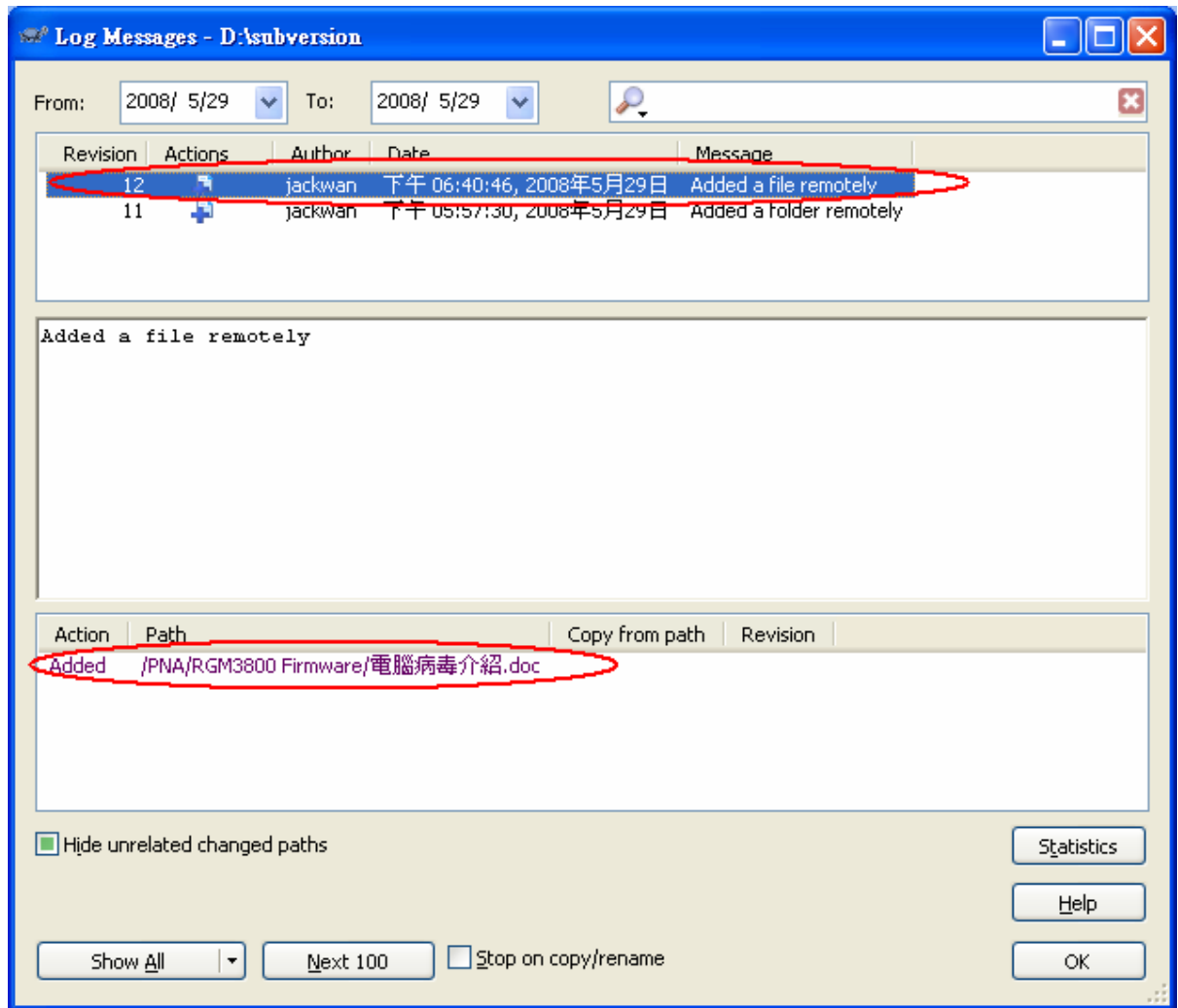
第五節、SVN Update

2.5.1、於 Subversion 按右鍵=>SVN Update Subversion 會將本機目錄內檔案之內容更新成 Server 上的最新版。



2.5.2、點 Show log 可以看到更新的 log 如下圖。



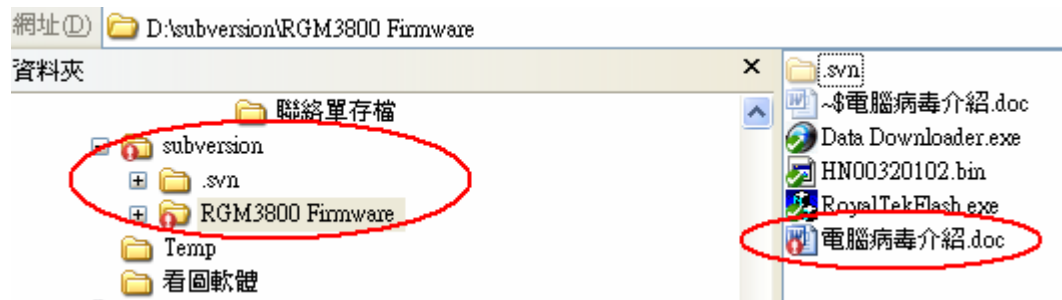


提示

團隊共同維護的專案在修改之前請先作資料更新 (svn update)，將本地電腦的工作複本更新為最新版，也就是說，修改前先取回其它研發人員所作的更動。

第六節、檔案修改與更新

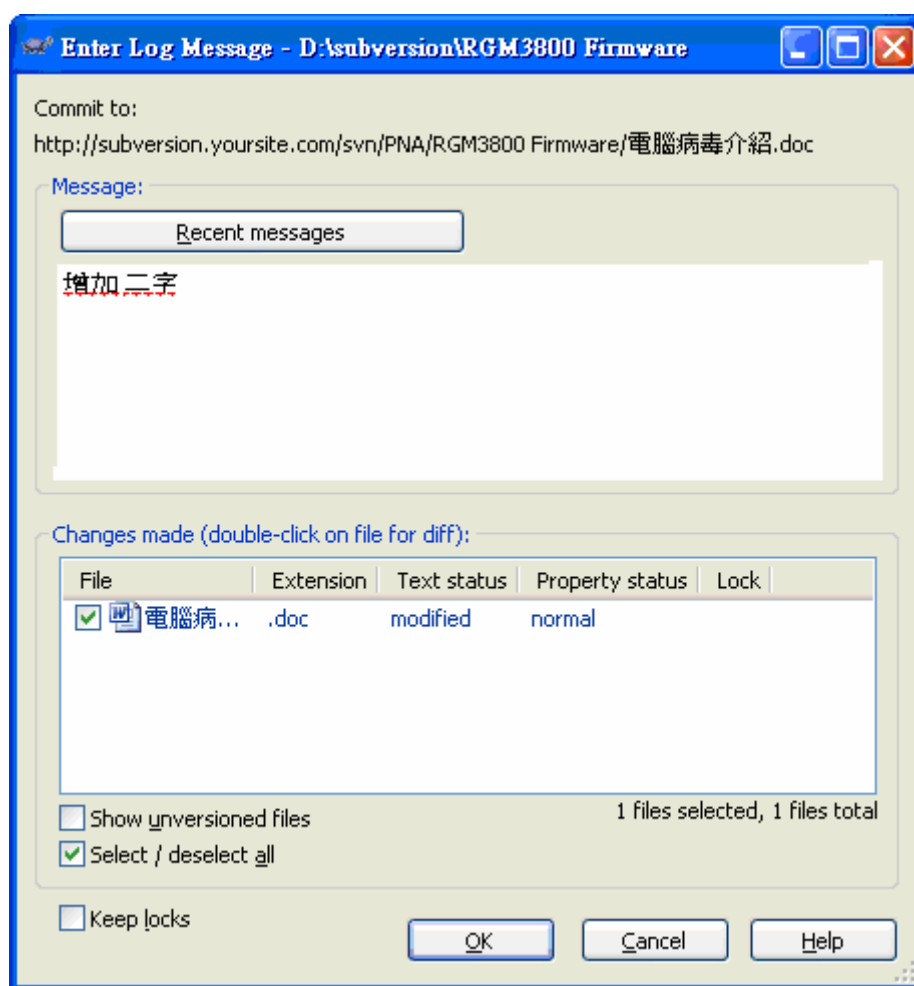
2.6.1、檔案被修改並存檔後，檔案的圖示就出現紅色驚嘆號如圖，表示檔案與已與 Server 不一致。



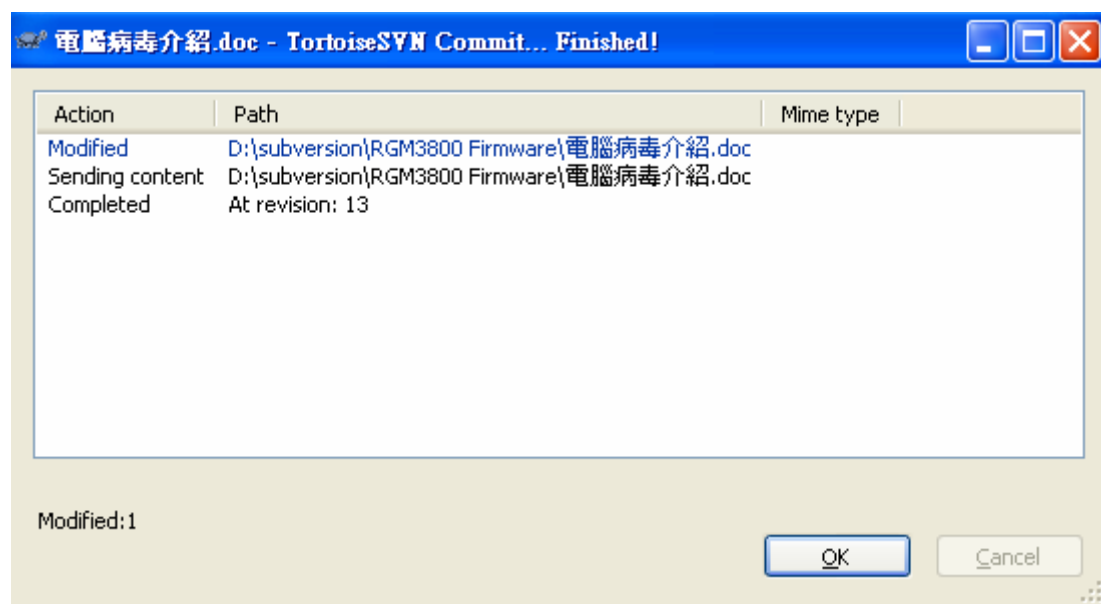
2.6.2、右鍵修改過之檔案 => SVN Commit 會將 Server 之檔案更新與目前正在修改的內容一致。




2.6.3、輸入修改之敘述 => OK



2.6.4、更新之完成畫面

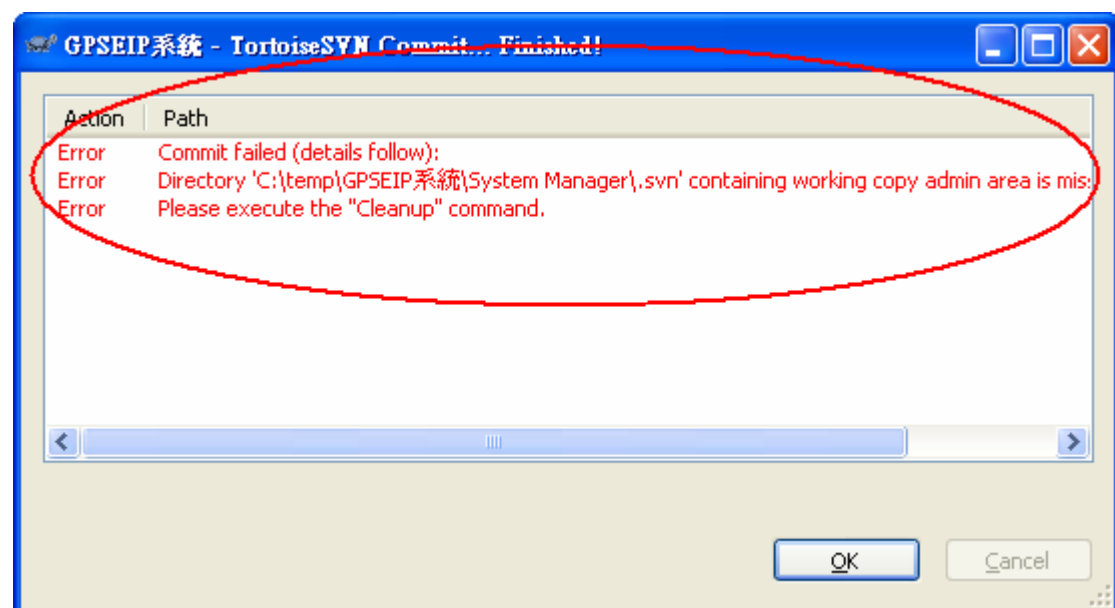


2.6.5、SVN Submit 後總管中該檔案已變成綠色的 。表示 Server 檔案已經與本機的檔案一致。



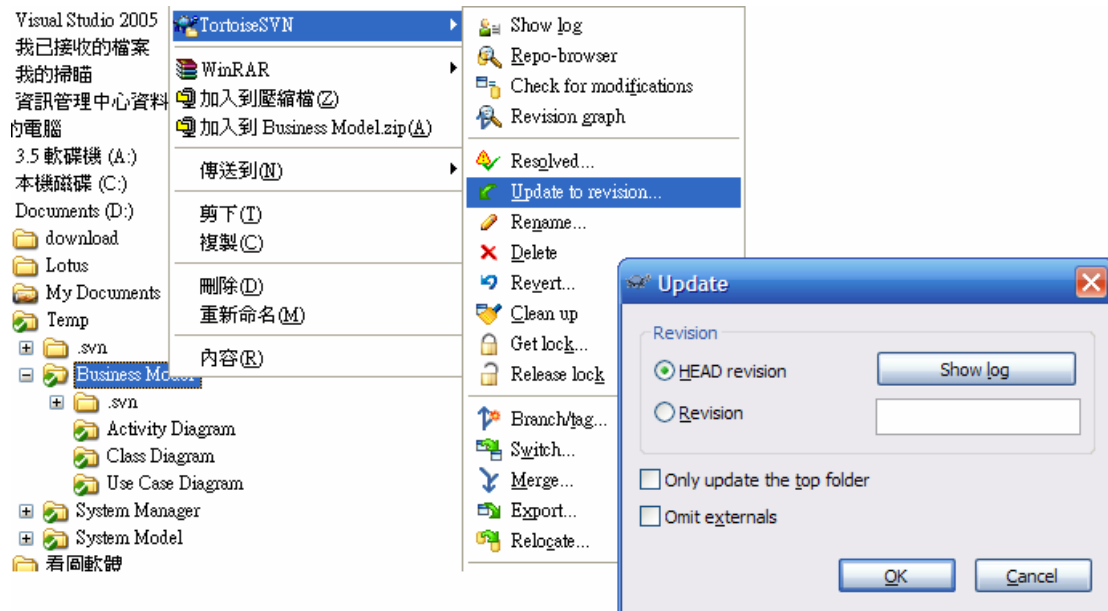
Note:有時候因為 Windows 本身的問題，您可能會看到有些 icon 沒有變成綠色。此時多按幾次 F5 應該就可以解決這個問題。

2.6.6、若 Submit 檔案時出現下圖，可能是未按照程序進行修改(例如:檔案遭直接覆蓋或檔案被刪除)，致本機的檔案無法對應到.SVN 的記錄資料而造成與 Server 同步失敗。

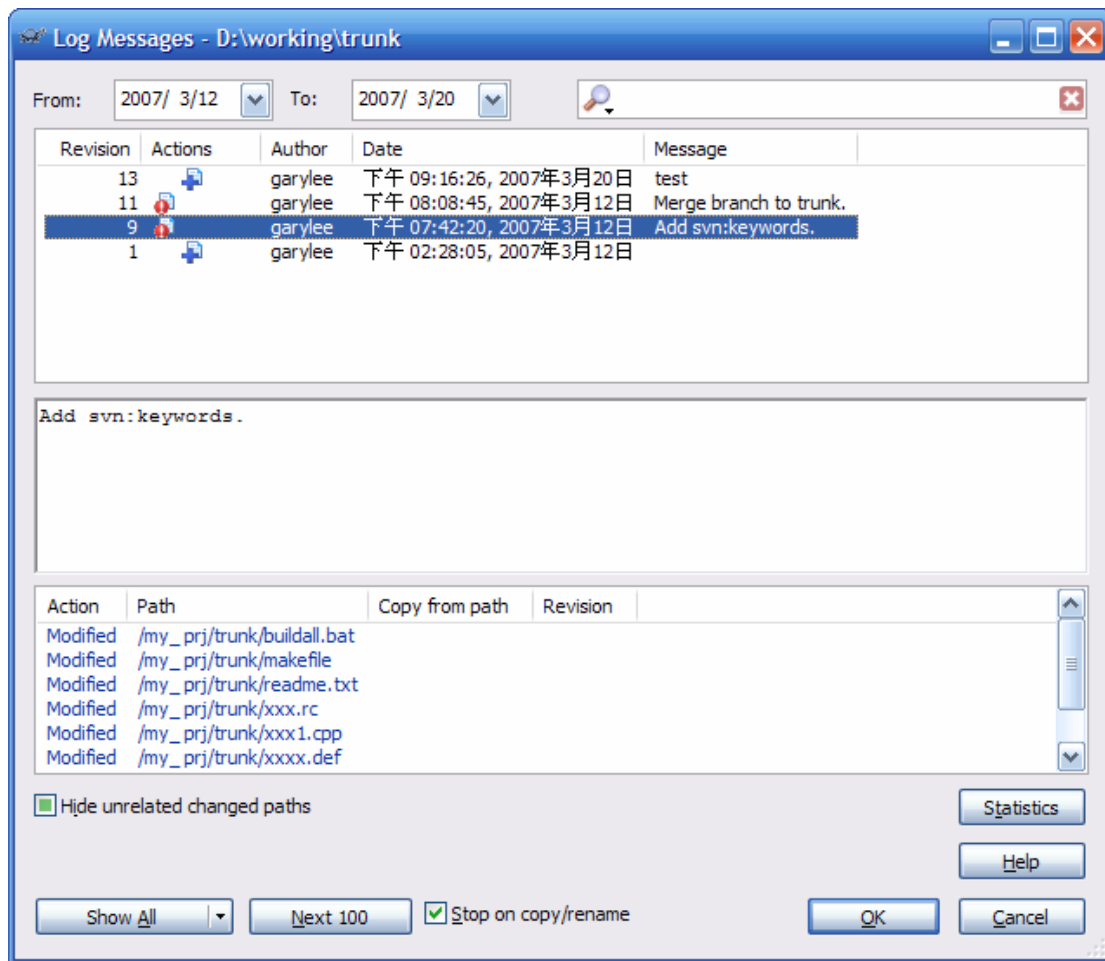


第七節、版本回溯

有時我們需要將檔案回溯至特定的日期或是版本，這時就可以利用 SVN 的 Update to revision 的功能。在想要更新的檔案或目錄上面按下滑鼠右鍵。並且選擇 TortoiseSVN => Update to revision。



在這個 Update 視窗中，您可以選擇更新到最新版本(HEAD)。也可以選擇更新到某個指定的版本(Revision)。當然，您可能早就記不起來正確的版本號碼。或者只隱約的記得大概在什麼時間。沒關係，按下 Show log 按鈕，您就可以回顧歷史了。所有您曾經做過的動作，及其日期與對應的版本都會列在這個視窗上面，只要在你想要的版上面點一下，讓他變成反白，然後按下 OK。這個版本就會自動填入 Update 視窗中的 Revision 欄位中。您只要再按下一次 OK，這個版本就會被取出來到您的硬碟中。



提示

svn commit 的動作，可以將不限數目的檔案與目錄的更動，視為單一不可分割的異動進行發表。在你的工作複本中，你可以修改檔案的內容，建立、刪除、更名、以及複製檔案與目錄，然後將所有的更動視為一個單位，一口氣送交回去。

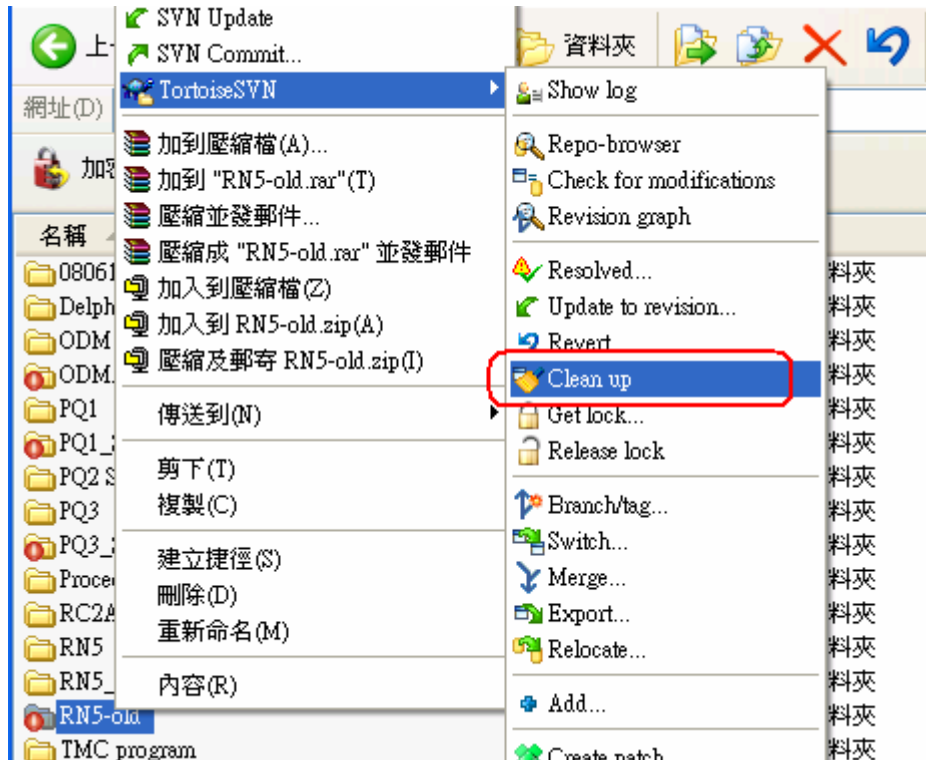
如果你在編輯器中寫了送交訊息，然後決定要取消這次的送交，只要不儲存更動，直接結束編輯器即可。如果你已經儲存了送交訊息，只要把文字都刪除然後再儲存一次即可。檔案庫不會管你的更動是否有意義；它只會檢查是否有人在你之前修改了你正在更動的檔案。如果檔案已被別人先修改過，那麼整個送交就會失敗並以一則訊息提示你，有一個或多個檔案已經過時了。

Transaction is out of date

此時，你需要處理產生的合併或衝突，然後再試著送交一次，詳見第四章解決衝突。

第八節、Clean up command

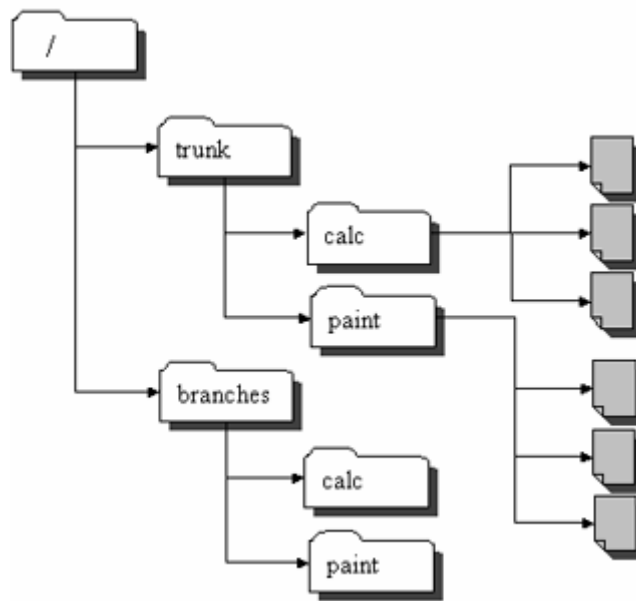
Subversion 使用工作日志機制來記錄所有的工作複本，如果 **svn** 工作意外中斷將會留下一個或多個尚未結束的鎖定檔案，存取工作遇到鎖定之檔案將會開啟失敗，請於該工作複本的上一層目錄執行 Clean up，繼續完成上述被中斷的工作後解除鎖定。



第三章、專案的目錄結構

Subversion 是一個用以分享資訊的中央系統，以檔案庫 (repository)作為儲存資料的核心。檔案庫儲存資料的形式是檔案系統樹 (filesystem tree)，也就是典型的目錄與檔案的架構。用戶需先連上檔案庫然後對這些檔案作讀取或寫入的動作。藉由共同資料庫觀念將資訊與他人共用。此節將說明目錄的結構。在專案的根目錄下建立的 trunk、branches、和 tags 這三個目錄是有特別意義的，它們的作用分別是：

- trunk 目錄用來存放目前專案正在進行開發的程式檔案和文件（又稱為主幹，即 mainline）。
- branches 用來存放主線的各個仍在發展中的分支。
- tags 則用來存放已經不再變動的分支，也就是其中的檔案不會再修改了。這是 Subverion 官方手冊建議的目錄結構安排方式。

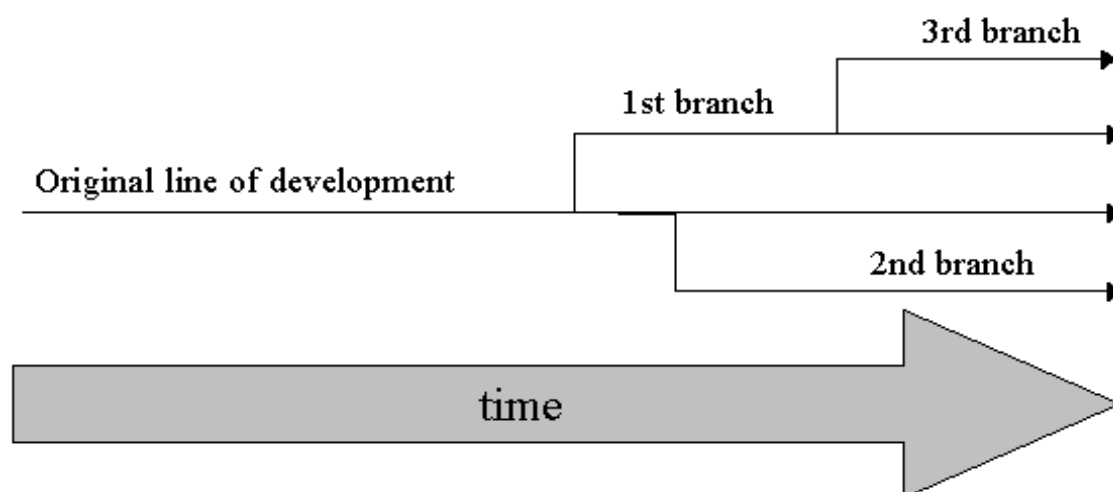


3.1、branches 分支說明：

Subversion 有一項功能可以幫助你維持檔案與目錄的平行分支。它讓你藉由複製資料來產生分支，並且會記住這些複本是彼此相關的，很多時候您會希望有另外一個複製的目錄來進行新的編修。等到確定這個分支的修改已經完畢了，再合併到原來的主要開發版本上。

分支的範例說明：

假設公司某部門有一份使用手冊，某天另一個部門被要求與你使用同一份手冊，由於部門不同內容也要有 '些微' 的修改，在這種情況下可使用分支將原手冊直接產生文件的第二份複本，然後兩個部門分別維護這兩份文件，如果未來兩個部門合併您也可以將分支文件併入原來的文件。換句話說，這兩條支線的發展途徑是各自獨立的，如果回溯的時間夠久你會發現它們的源頭都有共同的歷史紀錄。一個分支都是以某一文件的複本開始其生命週期，然後就自行發展下去有著自己的歷史紀錄，這就是分支的基本概念(如下圖)。



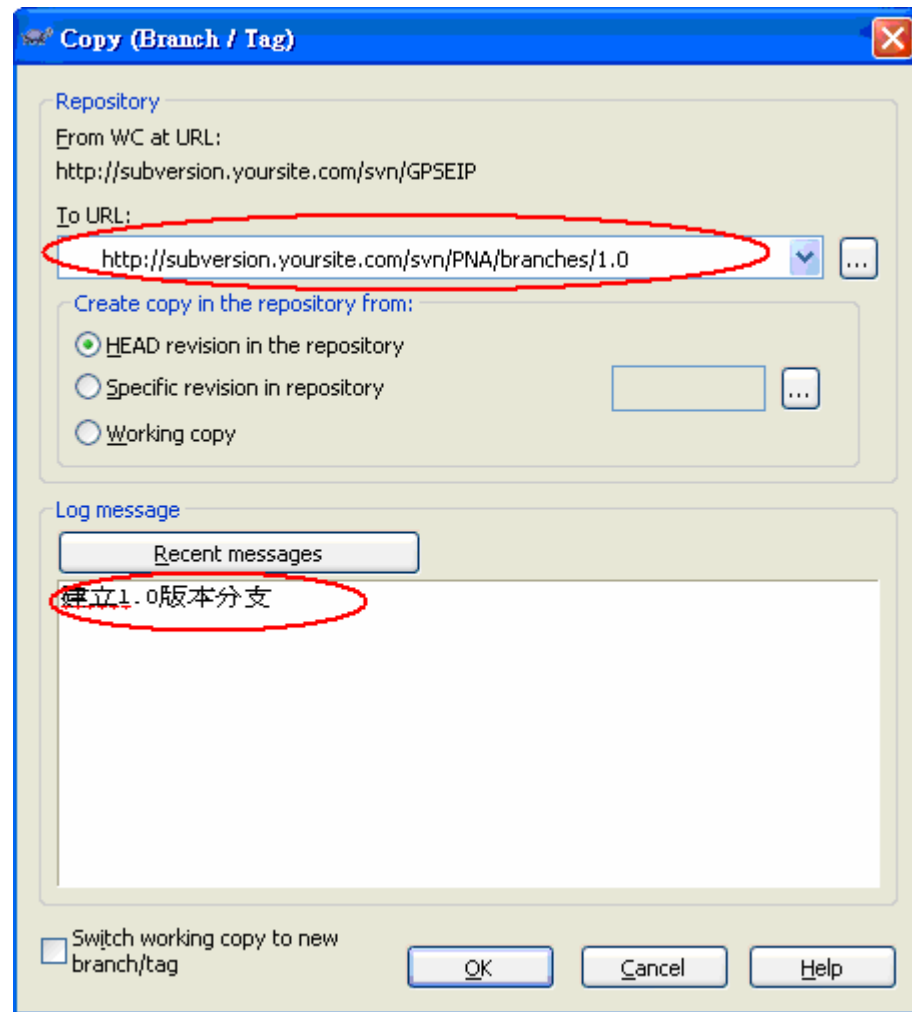
Subversion 的檔案庫設計相當地好。當你複製一個分支時，你不必擔心檔案庫會變得非常大，Subversion 不會真的複製資料，它只是建立一個指向現存檔案的連結。分支前的資料仍然使用同一檔案，只有分支後的修改才會分別記錄在各分支內。

3.2、建立分支

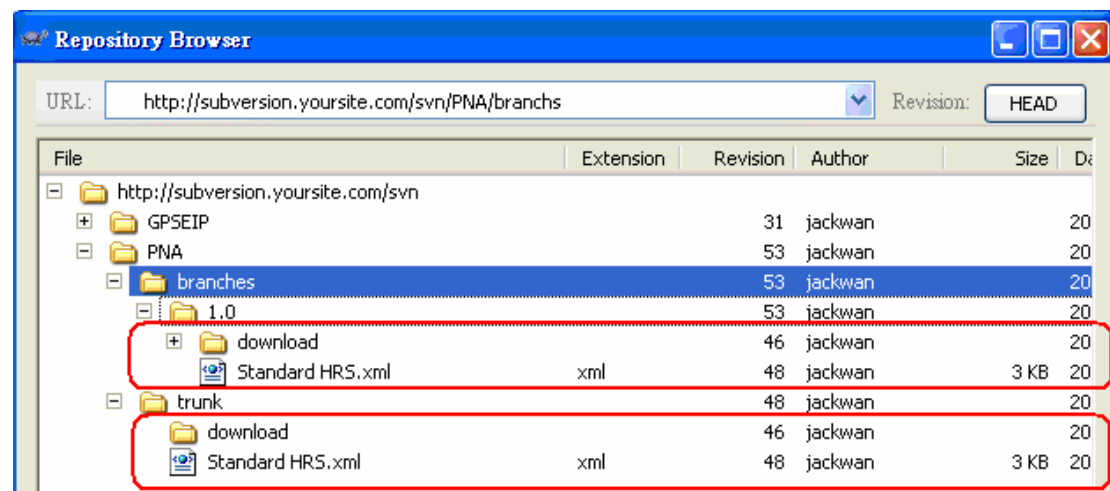


3.2.1、輸入分支目錄名稱及說明

專案分支請建立於 branches 目錄底下，以版本別做為目錄名稱。

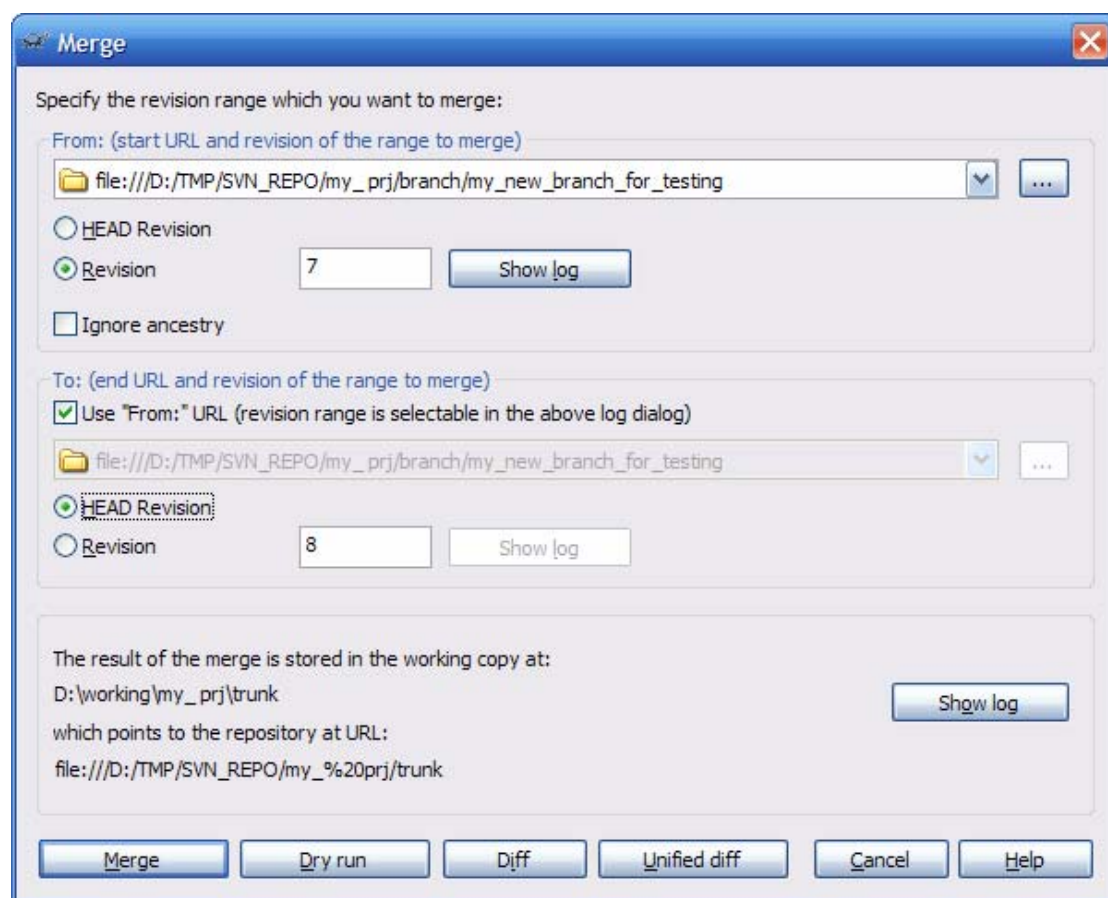


3.2.2、剛建立完成之分支內容與 trunk 相同。



3.3 合併分支

要將分支 merge 回原目錄中的方法很簡單。在分支目錄空白處，按下滑鼠右鍵，選擇 merge，接著可以看到如下的畫面：



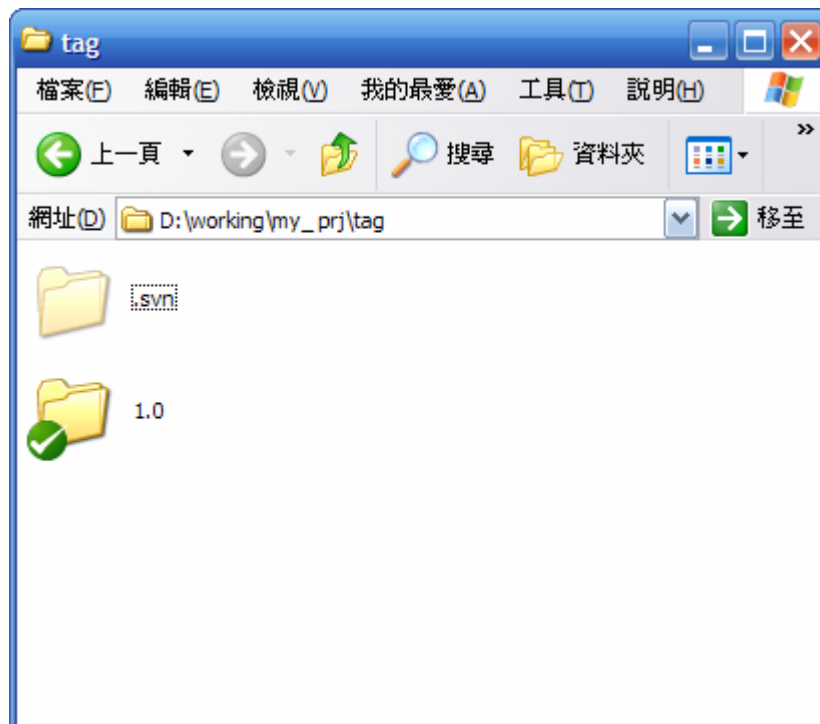
這個畫面主要分為三個部份，前面的 From: 與 To: 是要問您打算從 Branch 中的那個版本到那個版本，merge 回 Trunk 目錄中。因此，From 跟 To 的 URL 欄位應當都是指定原來 branch 的目錄下。剩下的就是指定要 merge 的 revision 範圍。以上面的例子而言，我們從 Branch 的 Revision 7 開始 merge 到 Branch 下面的最新版本。您可以透過，Dry run 按鈕，試作一次 merge。這個 merge 只會顯示一些訊息，不會真正的更新到 Trunk 的目錄去。只有按下 Merge 按鈕後，才會真正的將 branch 的檔案與 Trunk 檔案合併起來。

如果您確認這次的 merge 沒有問題，您可以直接使用 commit 來將這兩個被修改的檔案 commit 回 SVN repository 上。如果有問題，您可以直接修改這兩個檔案，直到確認 ok 了，再執行 commit。

3.4、製作 Tag 或是 Release

Tag 與 Release 的作法與 Branch 完全相同。只是 Branch 可能會需要 merge 回原來的 Trunk 中，而 tag 及 release 大部分都不需要 merge 回 Trunk 目錄中。舉例來說，今天我們完成了一版，這個版本被認定是軟體的 1.0 版。就開發而言 1.0 版

是非常重要的里程碑。所以我們要特別為他做一個標記，亦即 Tag。假設這個 1.0 版是要正式 release 給客戶或是相關 vendor，我們可以為他做一個 Release 的標記。



Note:製作 Release 的作法與 Tag 完全相同。只是把 Tag 的目錄換成 Release 而已。

相信大家都已經了解到無論是 Branch，Tag 或是 Release 都只是將指定的版本複製一份到另外一個目錄去。至於這個目錄要叫 Branch 還是叫 Release，SVN 根本就不管，您也可取其他的目錄名稱，不過 Trunk、Branch、Tag、或是 Release 都是 SVN 上面約定的名稱。除非您自己制定遊戲規則否則最好還是 follow 這個命名原則，以免後面新加入的人看不懂。

第四章、解決衝突（合併他人的更動）

假設因為你與協同工作人員 Sally 的溝通有誤，兩個人都同時編輯了 sandwich.txt. Sally 送交了她的更動，而你更新工作複本時就會得到一個衝突。此時我們要編輯 sandwich.txt 以解決衝突。首先，讓我們先看看這個檔案：

```
$ cat sandwich.txt
Top piece of bread
Mayonnaise
Lettuce
Tomato
Provolone
<<<<<<< .mine
Salami
Mortadella
Prosciutto
=====
Sauerkraut (德國泡菜)
Grilled Chicken
>>>>>>> .r2
Creole Mustard
Bottom piece of bread
```

這些小於符號，等於符號，以及大於符號，稱為衝突標記。夾在前兩種標記之間的文字，就是你在衝突區域所作的變更：

```
<<<<<<< .mine
Salami
Mortadella
Prosciutto
=====
```

夾在後兩組衝突標記的文字，來自於 Sally 的送交更動：

```
=====
Sauerkraut
Grilled Chickenn
>>>>>>> .r2
```

千萬不可以直接刪掉衝突標記與 Sally 作的更動，請拿起電話或是走過辦公室，跟她解釋在義大利食品店是買不到德國泡菜的。當你們都同意你將存入的變更，請編輯你的檔案將衝突標記移除。

Top piece of bread

Mayonnaise

Lettuce

Tomato

Provolone

Salami

Mortadella

Prosciutto

Creole Mustard

Bottom piece of bread

再執行 svn resolved， 你就可以送交你的更動。

