**淺談Subversion(SVN) Hook Script及其應用**

**前言**

版本控制系統(Version Control System)是一個開發團隊在開發程式的過程中不可或缺的工具。它可以有效地處理多人共同開發時會遇到的問題(例如：追蹤程式碼變動、確保程式碼同步性)，也能有效的管理產品開發進度、產品版本的控管。

根據管理的方式，大致可將版本控制系統分成集中式版本控管系統及分散式版本控管系統。

集中式版本控管系統主流的代表有**Concurrent Versions System(CVS)**、**Subversion(SVN)**。

分散式版本控管系統主流的代表有**Git**、**Mercurial(Hg)**。

接下來所要介紹的Hook Script，便是Subversion(以下簡稱SVN)底下的一個工具。

**Hook Script是什麼？**

根據特定的SVN操作(例如：提交版本**(commit)**，變更版本屬性**(revision property change)**、檔案上鎖**(lock)**、檔案解鎖**(unlock)**，SVN伺服器會在處理這些操作時，呼叫對應的Hook Script。

這個Script可由使用者自行撰寫，在Window環境下為.bat檔，在Linux環境下則為.sh檔。

依照不同的操作，可以將其分為四大類：

1. Commit Hook Script 提交版本時被呼叫
2. Revprop Change Hook Script 變更版本屬性時被呼叫
3. Lock Hook Script 檔案上鎖時被呼叫
4. Unlock Hook Script 檔案解鎖時被呼叫

**svnlook指令**

為了介紹Hook Script的應用，在詳述四個不同的Hook Script之前，必須先簡介svnlook這個指令。

svnlook是一個SVN命令列工具，在這個工具之下還有許多子命令，這裡只概述幾個常用的子命令，餘下的請參閱參考文獻。

在這裡以本機的SVN檔案庫(C:\\Repositories\gt)為例：

1. svnlook author 列出檔案庫檔案指定版本號或是交易序號的作者

svnlook author –r 100 C:\Repositories\gt

>> Vincent Huang

1. svnlook changed 列出檔案庫檔案指定版本號或是交易序號的變更檔案列表

svnlook changed –r 101 C:\Repositories\gt

>> U Branches/B\_20140123/Rule118.java

U Branches/B\_20140123/Rule200.java

U Branches/B\_20140123/Rule350.java

1. svnlook date 列出檔案庫檔案指定版本號或是交易序號的時間戳記

svnlook date –r 102 C:\Repositories\gt

>> 2014-09-14 11:15:33 +0800 (星期日, 14 九月 2014)

1. svnlook log 列出檔案庫檔案指定版本號或是交易序號的提交日誌

svnlook log –r 102 C:\Repositories\gt

>> [X][新增處理 Esm 格式的 method]

1. svnlook youngest 列出檔案庫檔案目前最新的版本號

svnlook youngest C:\Repositories\gt

>> 984

**Commit Hook Script**

SVN將提交的過程視為類似資料庫的交易行為，每一次提交的整個過程是不可分割的。這過程首先會產生交易序號，接著會再產生此次提交的版本號，依據產生交易序號跟產生版本號這兩個時間點將提交程序分成三個階段，這三個階段分別對應到三個不同的commit script(**start-commit**, **pre-commit**, **post-commit**)。

1. start-commit

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] USER (the authenticated user attempting to commit)

觸發時間點：開始提交之後，產生交易序號之前。

回傳成功：繼續提交程序，產生交易序號。

回傳失敗：終止程序。

常見應用：驗證USER是否有權限提交版本至REPOS-PATH。

範例腳本片段：

**SET** REPOS**=%1**

**SET** USER**=%2**

java VerifyAuthor %REPOS% %USER%

**if** %errorlevel% gtr 0 (goto err) **else** (goto success)

**:success**

**exit** 0

**:err**

**echo** Wrong Author! 1**>**&2

**exit** 1

1. pre-commit

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] TXN-NAME (the name of the txn about to be committed)

觸發時間點：產生交易序號之後，產生版本序號之前。

回傳成功：繼續提交程序，產生版本序號。

回傳失敗：終止程序。

常見應用：根據TXN-NAME，利用svnlook工具取得提交版本其餘資訊(author,date,changed,log,youngest)，驗證此次提交是否符合團隊規則。

範例腳本片段：

**SET** REPOS**=%1**

**SET** TXT**=%2**

svnlook author -t **%TXT% %REPOS% >** D:\CommitLog\author\_**%TXT%**.txt

svnlook log -t **%TXT% %REPOS% >** D:\CommitLog\log\_**%TXT%**.txt

java PreCommitVerify D:\CommitLog\author\_**%TXT%**.txt D:\CommitLog\log\_**%TXT%**.txt

**if** **%errorlevel%** gtr 0 (goto err) **else** (goto success)

**:success**

**exit** 0

**:err**

**echo** Wrong Commit Log Format! 1**>**&2

**exit** 1

1. post-commit

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] REV (the number of the revision just committed)

觸發時間點：產生版本序號之後，此時版本已為提交狀態。

注意：由於此時已為完成提交的狀態，無論腳本回傳成功失敗皆無法回溯(rollback)此次操作。

常見應用：根據REV利用svnlook工具取得提交版本其餘資訊(author,date,changed,log,youngest)，發送提交版本的通知，紀錄提交版本資訊。

範例腳本片段：

**SET** REPOS**=%1**

**SET** REV**=%2**

svnlook author -r **%REV% %REPOS%** **>** D:\CommitLog\author\_**%REV%**.txt

svnlook changed -r **%REV% %REPOS%** **>** D:\CommitLog\changed\_**%REV%**.txt

svnlook date -r **%REV% %REPOS%** **>** D:\CommitLog\date\_**%REV%**.txt

svnlook log -r **%REV% %REPOS%** **>** D:\CommitLog\log\_**%REV%**.txt

svnlook youngest **%REPOS%** **>** D:\CommitLog\youngest\_**%REV%**.txt

java RecordAndNotify D:\CommitLog\author\_**%REV%**.txt D:\CommitLog\log\_**%REV%**.txt

D:\CommitLog\date\_**%REV%**.txt D:\CommitLog\log\_**%REV%**.txt

D:\CommitLog\youngest\_**%REV%**.txt

**Revprop Change Hook Script**

在團隊開發的過程中，可能需要修正SVN伺服器上某個版本的屬性，例如：作者、提交日誌等等。

這個版本屬性變更的過程分成處理變更之前跟處理變更之後，分別對應到兩個不同的revprop change script(**pre-revprop-change-commit**, **post-revprop-change-commit**)。

1. pre-revprop-change-commit

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] REVISION (the revision being tweaked)

[3] USER (the username of the person tweaking the property)

[4] PROPNAME (the property being set on the revision)

[5] ACTION (the property is being 'A'dded, 'M'odified, or 'D'eleted)

觸發時間點：開始變更屬性之後，處理變更之前。

回傳成功：繼續版本屬性變更程序，開始處理變更。

回傳失敗：終止程序。

常見應用：驗證USER是否有權限變更PROPNAME。

範例腳本片段：

**SET** REPOS**=%1**

**SET** REV**=%2**

**SET** USER**=%3**

**SET** PROPNAME**=%4**

**SET** ACTION**=%5**

**if** **%PROPNAME%==**"svn:log" (goto MODIFY) **else** (goto CANNOT\_DOIT)

**:MODIFY**

**if** **%ACTION%==**"M" (goto DOIT) **else** (goto CANNOT\_DOIT)

**:DOIT**

**exit** 0

**:CANNOT\_DOIT**

**echo** Changing revision properties other than svn:log is prohibited 1**>**&2

**exit** 1

1. post-revprop-change-commit

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] REVISION (the revision being tweaked)

[3] USER (the username of the person tweaking the property)

[4] PROPNAME (the property being set on the revision)

[5] ACTION (the property is being 'A'dded, 'M'odified, or 'D'eleted)

觸發時間點：處理變更之後，此時已成功變更版本屬性。

注意：由於此時已成功變更版本屬性，無論腳本回傳成功失敗皆無法回溯(rollback)此次操作。

常見應用：變更屬性通知，紀錄變更屬性。

範例腳本片段：

**SET** REPOS**=%1**

**SET** REV**=%2**

**SET** USER**=%3**

**SET** PROPNAME**=%4**

**SET** ACTION**=%5**

svnlook log -r **%REV% %REPOS%** **>** D:\CommitLog\modify\_log\_**%REV%**.txt

java UpdateLogToDB **%REV%** D:\CommitLog\modify\_log\_**%REV%**.txt

**Lock Hook Script**

現今的SVN在處理衝突時，所使用的鎖定方法可以分為悲觀鎖定及樂觀鎖定。

這邊的鎖定指的是悲觀鎖定，是將要編輯的檔案上鎖，避免其他的使用者提交更改其內容或屬性。直到編輯完成再解鎖，所以當其他使用者提交已經被上鎖的檔案便會提交失敗。

這個鎖定的過程可以分成檔案上鎖之前跟檔案上鎖之後，分別對應到不同的lock script(**pre-lock**, **post-lock**)。

1. pre-lock

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] PATH (the path in the repository about to be locked)

[3] USER (the user creating the lock)

[4] COMMENT (the comment of the lock)

[5] STEAL-LOCK (1 if the user is trying to steal the lock, else 0)

觸發時間點：開始將檔案上鎖，處理檔案上鎖之前。

回傳成功：繼續檔案上鎖程序，開始上鎖檔案。

回傳失敗：終止程序。

常見應用：驗證USER是否有權限上鎖檔案。

範例腳本片段：

**SET** REPOS**=%1**

**SET** PATH**=%2**

**SET** USER**=%3**

**SET** COMMENT**=%4**

**SET** STEAL**=%5**

svnlook lock **%REPOS% %PATH%** **>** D:\CommitLog\lock\_**%USER%**.txt

java VerifyLock D:\CommitLog\lock\_**%USER%**.txt

**if** **%errorlevel%** gtr 0 (goto err) **else** (goto success)

**:success**

**exit** 0

**:err**

**echo** Lock File Fail! 1**>**&2

**exit** 1

1. post-lock

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] USER (the user who created the lock)

觸發時間點：處理檔案上鎖之後，此時已成功將檔案上鎖。

注意：由於此時已成功將檔案上鎖，無論腳本回傳成功失敗皆無法回溯(rollback)此次操作。

常見應用：檔案上鎖通知，檔案上鎖紀錄。

範例腳本片段：

**SET** REPOS**=%1**

**SET** USER**=%2**

java RecordAndNotifyLock **%REPOS% %USER%** D:\CommitLog\lock\_**%USER%**.txt

**Unlock Hook Script**

依照上述的Lock Hook，有上鎖就要解鎖，這邊的Unlock Hook便是與檔案解鎖相關的腳本。

這個解鎖的過程可以分成檔案解鎖之前跟檔案解鎖之後，分別對應到不同的unlock script(**pre-unlock**, **post-unlock**)。

1. pre-unlock

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] PATH (the path in the repository about to be unlocked)

[3] USER (the user destroying the lock)

[4] TOKEN (the lock token to be destroyed)

[5] BREAK-UNLOCK (1 if the user is breaking the lock, else 0)

觸發時間點：開始將檔案解鎖，處理檔案解鎖之前。

回傳成功：繼續檔案解鎖程序，開始解鎖檔案。

回傳失敗：終止程序。

常見應用：驗證USER是否有權限解鎖。

範例腳本片段：

**SET** REPOS**=%1**

**SET** PATH**=%2**

**SET** USER**=%3**

**SET** TOKEN**=%4**

**SET** BREAK**=%5**

svnlook lock **%REPOS% %PATH%** **>** D:\CommitLog\lock\_**%USER%**.txt

java VerifyUnLock D:\CommitLog\lock\_**%USER%**.txt

**if** **%errorlevel%** gtr 0 (goto err) **else** (goto success)

**:success**

**exit** 0

**:err**

**echo** UnLock File Fail! 1**>**&2

**exit** 1

1. post-unlock

預設帶入參數：

[1] REPOS-PATH (the path to this repository)

[2] USER (the user who destroyed the lock)

觸發時間點：處理檔案解鎖之後，此時已成功將檔案上鎖。

注意：由於此時已成功將檔案解鎖，無論腳本回傳成功失敗皆無法回溯(rollback)此次操作。

常見應用：檔案解鎖通知，檔案解鎖紀錄。

範例腳本片段：

**SET** REPOS**=%1**

**SET** USER**=%2**

java RecordAndNotifyUnLock **%REPOS% %USER%** D:\CommitLog\lock\_**%USER%**.txt

**結論**

本文快速地對SVN Hook Script主要內容做一個介紹。透過幾個腳本的整合應用，可以有效地控管開發進度及產生數據報表。文中範例是介接java來做邏輯判斷，就過程而言，是在特定的時間點由不同的Hook Script呼叫svnlook將需要的檔案庫資訊導出，接著再呼叫java程式來處理這些資訊。因此若是需要，也可使用其他語言(例如：C, Python)來替換java在範例中的角色。

**參考文獻**

1. <http://svnbook.red-bean.com/en/1.7/>
2. http://folk.uib.no/dpu081/svn/ecosmo\_repos/hooks/