Subversion

Un article de WikiBR.

La SVN (petit nom de SubVersion) a pour but de remplacer la CVS. Pour l'utilisateur lambda, l'utilisation en est quasiment identique, mais les fonctionnalités supplémentaires sont tout de même non négligeables.

Sommaire

Qu'est-ce que c'est SVN ?

Gestion de versions concurrentes

Comme l'indique le nom d'un très cher ancêtre de SVN, Subversion est un système de gestion de version concurrente. C'est à dire un outil qui permet de programmer à plusieurs sur un projet. Chaque développeur peut travailler sereinement sur sa copie du projet car SVN se chargera de la fusion des modifications apportées par chacun des utilisateurs.

Un outil comme SVN s'avère donc être un très bon moyen de centraliser le développement d'un programme. Cela permet en effet de conserver de manière sûre une copie du projet sur le serveur avec la possibilité de structurer le développement, la certitude de pouvoir retrouver les versions antérieures grace au principe de versionning et de stockage des modifications sous forme de patchs.

Structure d'un projet sur la SVN

Les projets de développement (donc on exclut les modules de configuration) ont tous à peu près la même structure. Cette structure reflète le développement :

- trunk : la branche principale du projet
- branches : répertoire qui contient les autres branches sous la forme de sous répertoire.
- tags : répertoire que contient les versions tagguées sous la forme de sous répertoire. Une version tagguée est en fait une image instantannée du développement, c'est à dire que c'est l'état du projet à un moment donné et ce qu'il y a dans le tag n'est plus censé changé.

Prennons l'exemple concret de l'arborescence du projet qRezix :

```
svn://swl/qrezix/
  + trunk
  + branches
  |   + qrezix--release--1.2
  |   + qrezix--release--1.6.2
  |   + qrezix--release--1.6.3
  |   + qrezix--release--1.6.4
  |   \ qrezix--release--2.0.0
  \ tags
      + qrezix--1.3
      + qrezix--1.6.2
      + qrezix--1.6.2-rc1
      + qrezix--1.6.2-rc2
      + qrezix--1.6.3
      + qrezix--1.6.4
      + qrezix--2.0.0-beta1
      + qrezix--2.0.0-beta2
      + qrezix--2.0.0-rc1
      + qrezix--2.0.0
      \ qrezix--2.0.1
```

Utilisation de svn

SVN s'utilise aussi simplement que CVS avec cependant quelques différences notables, principalement la syntaxe différente pour un checkout.

La SVN gère également des branches qui permettent de continuer à développer sur une branche 'instable' pendant qu'on décide de 'releaser' une version stable.

Chaque projet est appelé un **module**.

Récupérer un module

La commande diffère selon que le module est accessible via le svnserver ou svn+ssh :

en svn :

```
svn co svn://[user@]skinwel/nom_du_projet/partie_a_co [dest_locale]
```

en http :

```
svn co http://[user@]skinwel/nom_du_projet/partie_a_co [dest_locale]
```

en svn+ssh :

```
svn co svn+ssh://user@skinwel/home/svn/nom_du_projet/partie_a_co [destination]
```

Pour la plupart des utilisateurs la branche à récupérer sera trunk, mais ça peut être aussi branches/nom_de_la_branche, ou Tag/nom_du_tag... ou n'importe quel sous répertoire ou fichier...

Récupérer les modifications des autres

```
svn up [$fichiers$]
```

Met à jour votre copie locale, en prenant en compte les dernières versions des fichiers spécifiés. Si aucun fichier n'est spécifié, le programme mettra à jour tout le répertoire et ce récursivement. En cas de modification conflictuelle entre celle du serveur et celle de la copie locale, il faut résoudre les conflits.

Lors de l'update, une liste de noms de fichier s'affiche précédés par une lettre. Ces fichiers sont les fichiers affectés par l'update, et chaque lettre a une signification particulière, indiquant l'effet de la commande sur la version locale :

- A : Le fichier a été ajouté à la version local
- M : Le fichier est localement modifié (le contenu n'est pas touché par SVN)
- U : Le fichier est mis à jour
- G : Le fichier est patché (le fichier est modifié à la fois localement et sur la SVN et les modifications de la SVN sont fusionnées à la copie locale).
- C : La copie locale du fichier entre en conflit avec la version du serveur.
- I : Le fichier est ignoré

- D : Le fichier est supprimé
- ! : Le fichier est manquant
- ? : Le fichier n'est pas géré par SVN

Valider ses modifications

```
svn status [$fichiers$]
```

Indique l'état de la version locale (ou d'un fichier précis de la version locale). Cela indique avec la même syntaxe que ci-dessus les fichiers qui ont étés localement modifiés par rapport à la dernière opération de 'synchronisation' avec le serveur. Cette opération étant locale, elle n'indique les modifications que par rapport à la révision actuelle. Il faut faire

```
svn status -u [$fichiers$]
```

Pour avoir les modifications par rapport à la dernière révision de la branche (c'est utile avant de faire un svn up, pour prendre ses précautions).

```
svn ci [$fichiers$]
```

Cette commande envoie au serveur les fichiers modifiés. Si aucun conflit n'est détecté la modification est validée.

Il faut toujours updater la copie locale avant de commiter ses modifications, et s'assurer donc que les modifications intervenues entre temps sont compatibles

Avant d'envoyer le commit, SVN vous demandera un message expliquant le travail effectué. Ce message est suivi de la liste des fichiers avec les tags comme expliqué dans la section précédente. Dans le pire des cas, l'utilisation de la commande *diff* permet de voir l'ensemble des modifications :

```
svn diff [$fichiers$]
```

Annuler ses modifications

```
svn revert [$fichiers$]
```

Supprime toutes les modifications locales apportées au fichier spécifié depuis le dernière update.

Cette commande est la seule qui ne soit pas récursive avec SVN. Pour utiliser la

commande sur un répertoire de façon récursive, il faut ajouter l'option -R

```
svn revert -R [$répertoire$]
```

Reverter des modifications déjà commitées

Il suffit d'appliquer un patch inverse correspondant aux modifications que l'on souhaite annuler. Ainsi pour annuler la révision 42, il suffit de faire

```
svn merge --commit -42 [$répertoire$]
```

ce qui indique à subversion qu'il faut appliquer à la version locale courante les changements opposés au commit 42. Il faut ensuite commiter ces changements.

Tu peux trouver plus d'informations dans "Version Control with Subversion" : Undoing Changes (http://svnbook.red-bean.com/en/1.4 /svn.branchmerge.commonuses.html#svn.branchmerge.commonuses.undo) .

Ajouter un fichier/répertoire

```
svn add $fichiers$
svn mkdir $repertoire$
```

Ajoute les fichiers spécifiés à la SVN. Les répertoires sont ajoutés immédiatement, les fichiers ajoutés par contre doivent être comités pour valider l'ajout. L'utilisation de mkdir nécessite que le répertoire n'existe pas encore. Il n'est donc pas suffisant de créer un répertoire ou un fichier dans l'arbre de la svn pour qu'il soit ultérieurement pris en compte par subversion.

Supprimer un fichier/répertoire

```
svn remove $fichiers$
svn rmdir $repertoire$
```

Supprime un fichier du module. La version locale est immédiatement effacée. La suppression sera effective après un commit (et il est toujours possible avec un 'merge' de revenir en arrière).

Déplacer un fichier/répertoire

```
svn move $fichier$ $destination$
```

Déplace le fichier ou répertoire vers sa destination. Se comporte comme mv.

Créer une version tagguée

Une version tagguée est un snapshot du projet, une photographie à un instant présent. Sur svn, un snapshot est donc une copie de l'instant présent dans l'arborescence. Le plus souvent on utilise un répertoire tag :

```
svn copy trunk tags/$montag$
```

Créer une branche

De la même manière que les tags, les branches sont une séparation du projet. C'est donc également une copie, mais celle-ci destinée à évoluer... On stock habituellement les branches dans un répertoire branches :

```
svn copy trunk branches/$mabranche$
```

Backporter

Backporter consiste à transposer les modifications d'une branche sur une autre.

```
svn merge -r vInitiale:vCible $brancheorigine$ [$branchedest$]
svn ci
```

Cette syntaxe applique les modifications de la branche $brancheorigine$ effectuées de la révision *vInitiale* à la révision *vCible* (dans cet ordre là, même si vInitiale>vCible) la branche *branchedest*. Si *branchedest* n'est pas spécifié, l'application sera réalisée sur le répertoire courant.

Attention à mettre *vInitiale* et *vCible* dans le bon sens, sinon on supprime les modifications au lieu de les appliquer.

Par exemple pour importer le commit 1783 de la branches qrezix--release--2.0.0 sur le trunk, il faut exécuter la commande :

```
cd trunk
svn merge -r1782:1783 svn://swl/qrezix/branches/qrezix--release--2.0.0 .
```

Résoudre les conflits

Après un update il est possible que certaines modifications locales soit en conflit

avec les modifications des autres développeurs (dans le cas en particuliers ou une même ligne a été modifiée). Il faut alors faire une fusion à la main, c'est la résolution du confit.

Pour ceci il faut éditer le fichier concerné. Les points conflictuels sont facilement identifiables car délimités par :

```
<<<<<<< .mine
... ma version du code
=======
... la version SVN du code
>>>>>>>.r849
```

Il suffit alors de choisir quelles modifications doivent être conservées... et d'enregistrer le fichier. Il ne faut tout de même pas oublier de marquer le conflit comme résolu :

```
svn resolved $fichier$
```

Installation de la SVN sous Windows

Un bon client SVN sous Windows est TortoiseSVN (http://tortoisesvn.tigris.org/) qui s'intègre à l'explorateur Windows et permet de réaliser toutes les commandes grace au menus contextuels associés aux répertoires.

Installation de SVN sous MacOS X

Subversion est installé en standard dans Mac OS X Leopard. Si vous souhaitez utiliser la toute dernière version de subversion, ou si vous n'êtes pas encore passés sous Leopard, plusieurs solutions s'offrent à vous :

- soit vous pouvez utiliser les packages 'tout fait' qui se trouvent sur le site de Martin Ott (http://www.codingmonkeys.de/mbo/) .
- soit vous pouvez utiliser fink (http://fink.sourceforge.net/) en tapant dans un Terminal :

```
fink install svn
```

- soit vous pouvez utiliser MacPorts (http://macports.org/) en tapant dans un Terminal :

```
port install subversion
```

Il y a ensuite un certain nombre de client graphiques pour svn (comme svnX (http://www.lachoseinteractive.net/en/community/subversion/svnx/download/) par exemple), si nécessaire, pour visualiser les évolutions du projet par exemple, ou bien voir sa structure.

Récupérée de « http://gwennoz.polytechnique.fr/wiki/Subversion »

- Cette page a été modifiée pour la dernière fois le 1 décembre 2008 à 00:45 par jean-philippe dugard. Basé sur le travail de Olivier et autres.