

Utilisation de SVN avec Eclipse

par Baptiste Wicht ([home](#))

Date de publication : Le 29 Juin 2007

Dernière mise à jour : Le 25 Octobre 2007

Subversion est un système de gestion de version, c'est-à-dire qu'il permet de gérer la version d'un fichier source et de garder un historique de toutes ces versions. Cela se révèle très pratique pour revenir en arrière par exemple. Cela permet aussi de travailler à plusieurs sur le même projet. On va voir comment intégrer **Subversion** dans **Eclipse**.

1 - Introduction.....	3
1.1 - Vocabulaire.....	3
2 - Installation.....	4
3 - Premier projet.....	5
4 - Utilisation de base.....	7
5 - Explorer le repository.....	10
6 - Conclusion.....	12
6.1 - Liens.....	12

1 - Introduction

Vous gérez des gros projets, vous avez de la peine à vous y retrouver et à gérer les versions de vos fichiers et en plus vous aimeriez travailler à plusieurs sur le même projet. Il vous faut donc un système de gestion de version.

Un tel système permet de gérer les versions des sources et de garder un historique de toutes ces versions. Ainsi aucun problème pour revenir à une ancienne version ni d'ailleurs pour travailler à plusieurs sur un projet puisqu'un tel système permet aussi de mixer vos modifications et celles d'une autre personne. Vous pouvez aussi mettre à jour une classe avant de faire des modifications pour bénéficier de la dernière version. Vous pouvez aussi comparer 2 versions d'un fichier.

Pour faire cela, nous allons donc utiliser **Subversion** qui est l'évolution de **CVS**, un autre système de gestion de version. **SVN** est un des systèmes de gestion de version les plus élaborés.

Malheureusement **Subversion** n'est pas installé par défaut dans **Eclipse**, au contraire de **CVS** qui est installé avec **Eclipse** (on y accède via le menu "team->share project"). Nous allons donc l'intégrer dans **Eclipse** pour faire encore plus simple.

1.1 - Vocabulaire

Avant d'aller plus loin, je vais décrire les différents mots de vocabulaire utilisé tout au long de cet article. Voici donc le vocabulaire **Subversion** :

- **Repository** ou **dépôt** : C'est simplement le répertoire, sur le serveur **Subversion**, qui va accueillir les fichiers du projet ainsi que les données sur les versions.
- **Checkout** : C'est l'opération qui consiste à obtenir une copie locale de la dernière version d'un projet sur le *repository*.
- **Commit** : C'est l'opération qui consiste à envoyer les changements d'un fichier sur le serveur **Subversion**.
- **Update** : C'est l'opération qui permet de mettre à jour la version locale d'un fichier avec la dernière version du serveur.
- **Merge** : C'est l'opération qui permet de mettre à jour un fichier local qui a été modifié avec les mises à jour du *repository*. Cette opération est déclenchée quand le fichier a été modifié en local et qu'il y a également des modifications sur la version du dépôt.
- **Branche** : C'est en fait une sous-version (dérivation) de la version principale. Si on a une seule version tout au long du développement, on n'aura pas de branches, mais si on doit faire une sous-version d'une version spécifique, on devra alors créer une nouvelle branche qui évoluera alors indépendamment de la branche principale
- **Tag** : C'est une capture du repository à un moment précis. Par exemple, pour la beta2 de notre projet, on va faire un tag "beta2" et y copier le contenu de la branche sur laquelle on travaille. Un tag n'est pas destiné à évoluer, c'est une version figée.

2 - Installation

Cette installation a été réalisée sur la version 3.2 d'**Eclipse**.

Pour commencer, il vous faudra un serveur **Subversion**. Vous pouvez soit l'avoir directement sur votre machine soit sur Internet, ce qui est plus pratique au niveau du partage. Vous pouvez par exemple en avoir un sur Sourceforge.net avec l'hébergement de votre projet, mais plusieurs autres hébergeurs donnent aussi l'accès à un serveur Subversion en plus de l'hébergement. Voici par exemple deux endroits où trouver gratuitement un serveur Subversion :

- <https://opensvn.csie.org/>
- <http://unfuddle.com/home>

Ensuite, nous allons installer le plugin **Subclipse** qui permet d'utiliser **SVN** directement dans **Eclipse**. Pour installer ce plugin, c'est tout ce qu'il y a de plus simple :

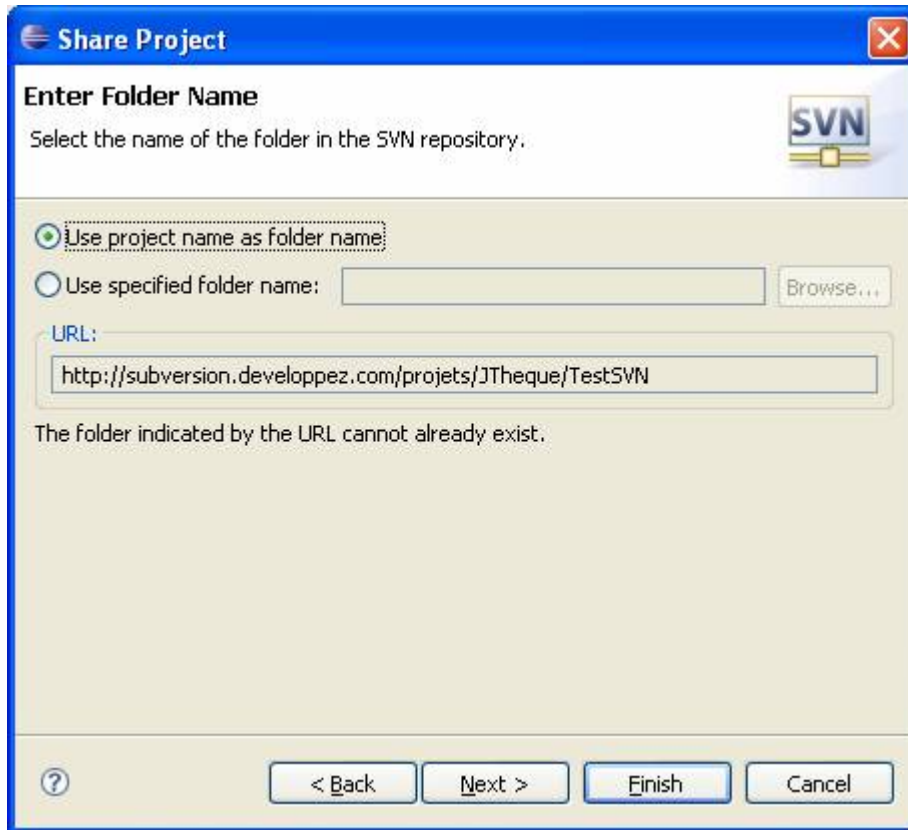
- Aller dans le menu "Help>Software updates>Find and Install"
- Cliquer sur "search for new features"
- Ajouter un nouveau site distant du nom de Subclipse et avec l'URL <http://subclipse.tigris.org/update>
- Sélectionner le site puis cliquer sur "finish"
- Sélectionner tous les éléments de **Subclipse** dans la liste qui devrait arriver
- Et cliquer sur "next"
- Accepter les termes de la licence
- Appuyer sur "next" puis sur "finish"
- Redémarrer le workspace

Subclipse est désormais installé.

3 - Premier projet

Maintenant que **Subclipse** est installé, nous allons créer un projet qui va utiliser **SVN**. Créons donc un projet que l'on va appeler **TestSVN**. Ensuite pour relier votre projet à un *repository* **SVN**, cliquez droit sur votre projet, et choisissez "share project", dans le menu "team". Choisissez **Subversion** comme type de *repository* et choisissez ensuite "Create a new repository location". Choisissez ensuite l'URL de votre repository.

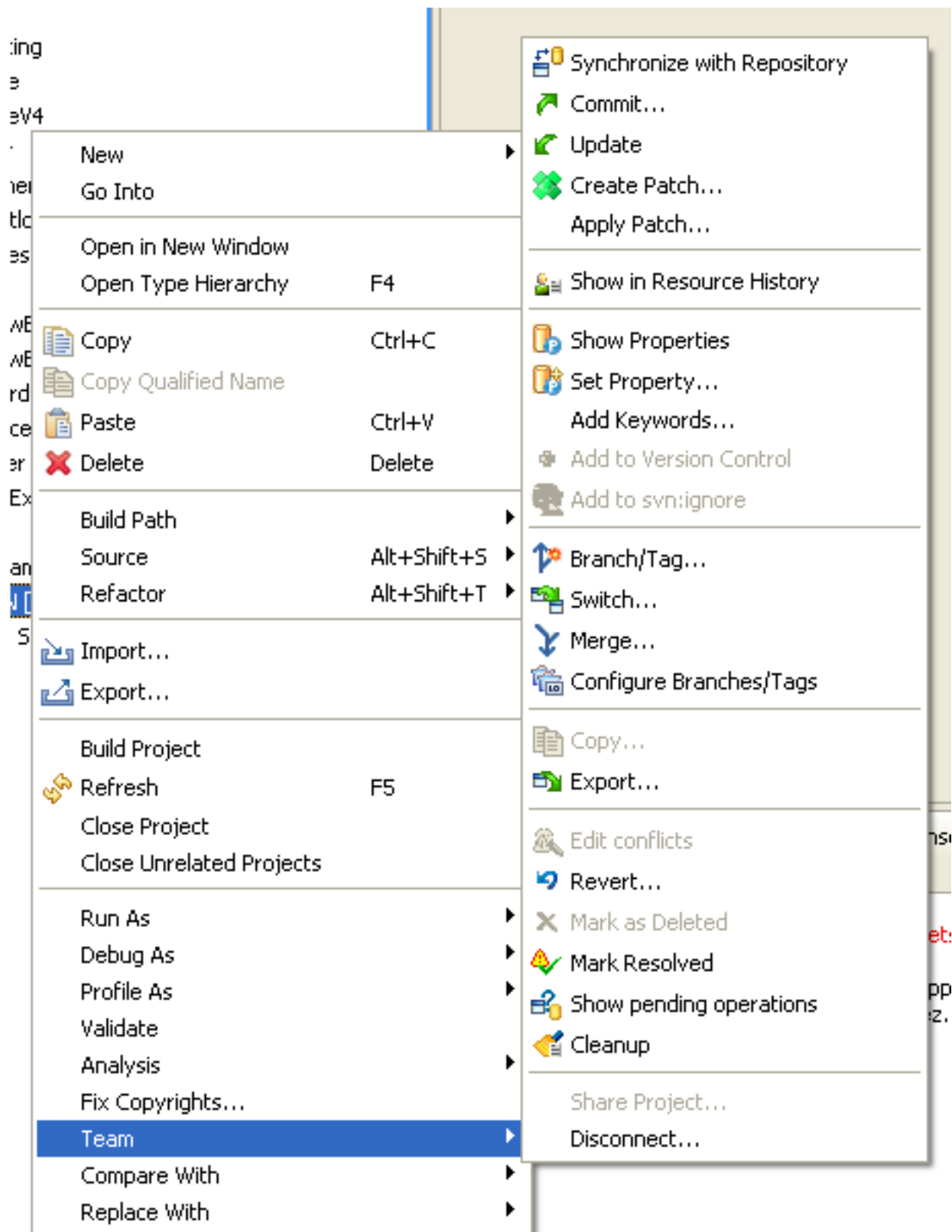
Vous devriez donc avoir quelque chose comme ça comme résultat à cette étape :



Partager un projet

Cliquez ensuite sur "finish" et entrez votre mot de passe dans la fenêtre de connexion qui devrait arriver. Vous pouvez choisir d'enregistrer votre mot de passe pour ne pas avoir à l'entrer plusieurs fois.

Ensuite, le programme va automatiquement exécuter un *checkout* du projet. Voilà, vous êtes donc prêts à travailler avec votre projet de manière collaborative. Pour vérifier que **SVN** est bien configuré pour votre projet, cliquez droit sur le projet et dans le menu "team", vous devriez maintenant voir apparaître plusieurs nouvelles fonctions.



Menu Team d'un projet

Nous allons voir l'utilité d'une partie de ces fonctions dans les prochains chapitres de ce tutoriel.

4 - Utilisation de base

Créez maintenant une nouvelle classe dans un nouveau package. La première chose que l'on remarque, c'est un petit point d'interrogation sur l'icône du fichier et la même sur celle du package, cela veut tout simplement dire que ces fichiers ne se trouvent pas sur le *repository*. Pour commencer, nous allons juste ajouter un constructeur à notre classe. Maintenant, nous allons *committer* la version de ce fichier sur le *repository*. Pour faire cela, rien de plus simple, il vous suffit de cliquer droit sur votre fichier, d'aller dans le menu team et de choisir "commit". Ensuite, il vous suffit d'entrer un commentaire de *commit* (c'est très pratique ensuite pour s'y retrouver avec des nombreuses versions) et de sélectionner le fichier dans la liste.

Cliquez ensuite sur "OK" et le fichier va être *committé*. Vous voyez tout de suite que l'icône associée au fichier a changée. Celle-ci indique que le fichier est bien dans le *repository*. A chaque fois que vous faites une opération **SVN**, vous aurez les indications dans la console qui vous donnerons des informations, par exemple, voici la trace pour l'opération que l'on vient de faire :

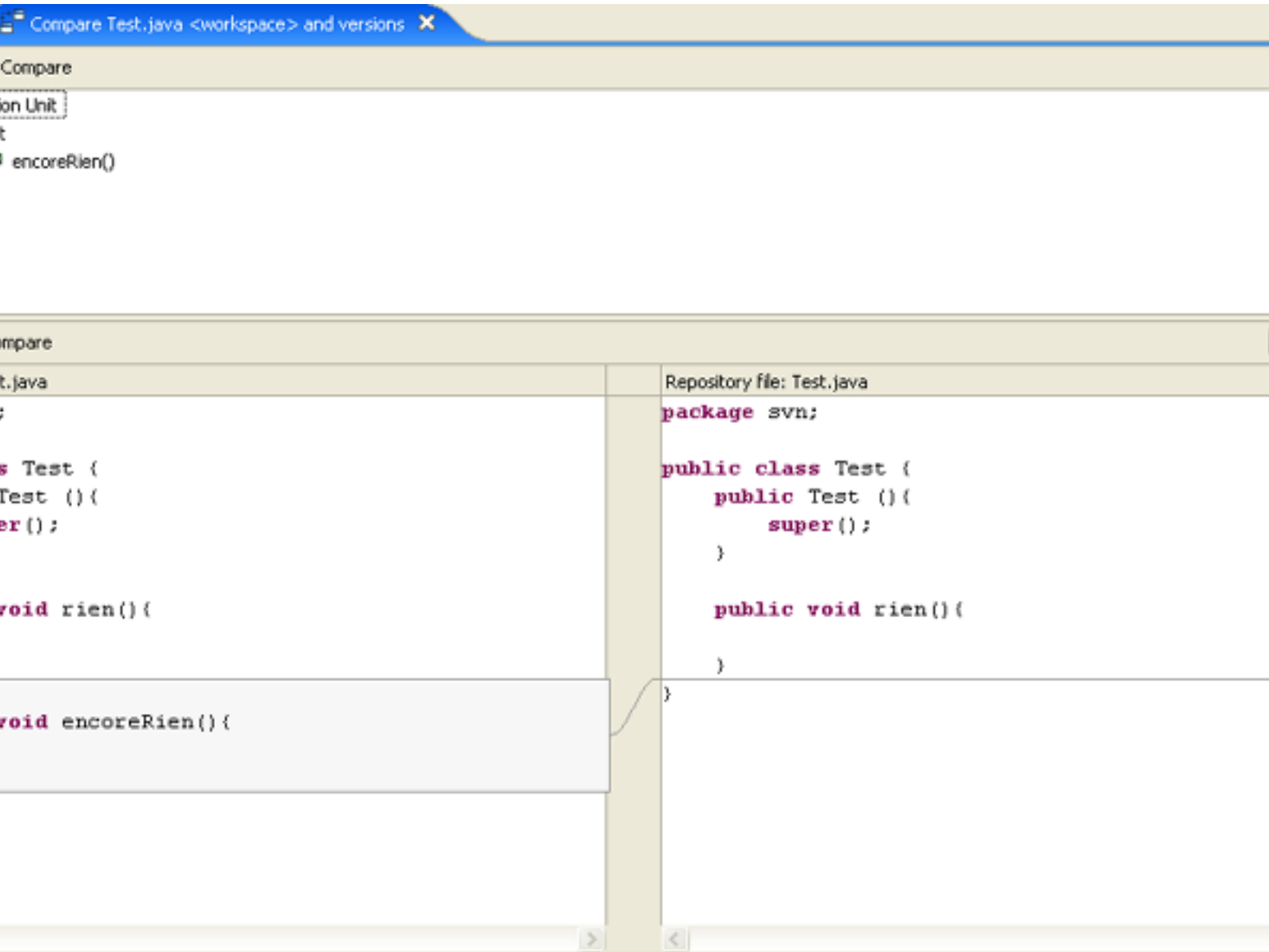
```
add-N M:\Programmation\Workspaces\Java - Workspace - Propre\TestSVN\svn
  A      M:/Programmation/Workspaces/Java - Workspace - Propre/TestSVN/svn
add -N M:\Programmation\Workspaces\Java - Workspace - Propre\TestSVN\svn\Test.java
  A      M:/Programmation/Workspaces/Java - Workspace - Propre/TestSVN/svn/Test.java
commit -m "First revision" -N M:/Programmation/Workspaces/Java - Workspace - Propre/TestSVN/svn
Adding      M:/Programmation/Workspaces/Java - Workspace - Propre/TestSVN/svn
Committed revision 2.
commit -m "First revision" M:/Programmation/Workspaces/Java - Workspace - Propre/TestSVN/svn/Test.java
Adding      M:/Programmation/Workspaces/Java - Workspace - Propre/TestSVN/svn/Test.java
Transmitting file data ...
Committed revision 3.
```

Il a donc commencé par ajouter le package `svn` et le fichier `Test.java` sur le serveur et ensuite il a *committé* les 2 éléments. Si vous regardez à côté du nom du fichier, vous verrez apparaître le numéro de révision, la date de *commit* ainsi que le nom de l'utilisateur qui l'a committé.

Maintenant, nous allons modifier notre classe. On va y ajouter par exemple une méthode ou un champ et on va sauvegarder nos modifications. Vous verrez qu'encore une fois l'icône a changée, cette fois, elle indique que le fichier est bien sur le serveur, mais que la version en local a changée par rapport à la version du serveur. Vous pouvez voir que cela s'applique aussi à tous les objets parents du fichier modifié, c'est-à-dire le package et le projet. Nous allons donc *committer* les changements de notre classe et l'icône du fichier va reprendre son état d'origine, indiquant que le fichier est à jour.

Vous pouvez aussi bien entendu *committer* plusieurs fichiers en même temps. Il vous suffit de sélectionner tous les fichiers que vous voulez et de faire un *commit*.

Une fonction intéressante est de pouvoir comparer notre version avec une autre. Nous allons encore modifier notre classe. Ensuite, utiliser le menu "compare with" du projet et choisissez "base revision", c'est-à-dire la dernière révision actuelle de ce fichier. Vous pouvez aussi choisir une autre révision voire un fichier d'une autre branche, mais nous n'allons pas voir cela dans ce tutoriel. Vous avez donc une interface qui vous compare tout d'abord, la structure des deux fichiers si elle a changé et ensuite le fichier en entier et vous verrez très clairement les différences entre les deux versions.



Comparaison de deux versions

Admettons maintenant que vous n'êtes pas content des changements apportés à votre fichier et que vous aimeriez revenir à la version précédente. Vous avez deux manières de faire. Soit tout simplement "Team->Revert" qui vous fait revenir à la dernière version, mais vous pouvez aussi spécifier à quelle version vous voulez revenir en faisant "replace with" et en choisissant précisément le moment auquel vous voulez revenir.

Il faut maintenant penser que vous n'êtes pas seul sur ce projet, il faut donc récupérer aussi les modifications de vos collègues. Pour cela, rien de plus simple, il vous suffit de faire un *update* d'un fichier. S'il y a eu des modifications, le programme va directement les intégrer dans votre fichier. N'ayez pas peur de perdre vos propres modifications non encore sur le repository, s'il y a des modifications des deux côtés, le programme va automatiquement faire un *merge* des deux fichiers pour en faire une nouvelle version qui contient les modifications des 2 personnes. S'il y a conflit, c'est-à-dire que le programme n'arrive pas à *merger* correctement les 2 révisions, le fichier va être marqué comme étant en conflit. Vous avez dans le menu "team" une fonction "edit conflicts" qui va vous aider à résoudre les conflits, une fois le conflit résolu, il vous suffit d'utiliser la fonction "mark resolved" sur le fichier pour que le conflit soit marqué comme résolu. Mais rassurez vous ça n'arrive pas très souvent, c'est surtout dans les gros projets avec un grand nombre de personnes travaillant dessus et faisant beaucoup de modifications que le risque augmente.

Vous pouvez aussi visualiser l'historique des révisions d'un fichier. Pour cela, dans le menu "team", choisissez "show in resource history". Vous verrez ainsi apparaître une vue contenant la liste des révisions avec à chaque fois l'auteur

de la révision, la date et le commentaire. Il vous suffit de cliquer sur une des révisions pour avoir le commentaire en entier et le path du fichier dans l'affichage du bas. Si vous cliquez droit sur une révision, vous aurez accès à d'autres actions directement en rapport avec la révision, vous pouvez revenir depuis ici à une ancienne version ou voir le fichier tel qu'il était lors de la révision. Vous pouvez aussi rafraîchir pour vérifier si d'autres changements ont eu lieu.

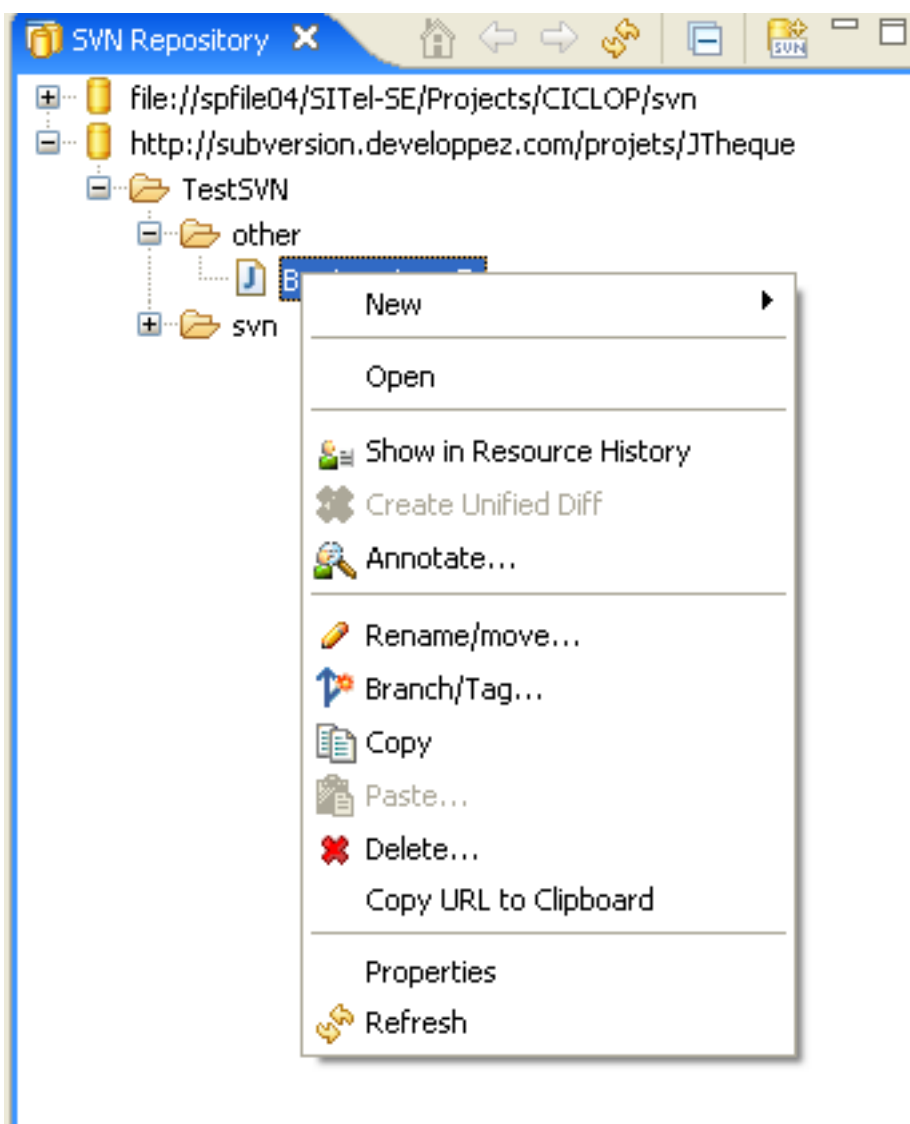
5 - Explorer le repository

Une autre possibilité très intéressante est de pouvoir explorer le *repository* **SVN**. Pour cela, rien de bien compliqué, il vous suffit d'aller dans le menu "Window" et d'ouvrir la perspective "SVN Repository Exploring".

Dans le panneau de gauche, choisissez le *repository* que vous voulez explorer. Depuis là, vous verrez l'arborescence de tous vos fichiers et packages. Vous pouvez bien sûr ouvrir des fichiers et cela va vous ouvrir le fichier qui se trouve actuellement sur le *repository* en lecture seule.

En cliquant droit sur le fichier de votre choix et en choisissant "show in resource history", vous avez accès à l'historique du fichier dans la vue du bas.

C'est aussi depuis cette perspective que vous pourrez supprimer des fichiers et des dossiers que vous auriez ajouté par erreur sur votre *repository*. Vous pouvez aussi renommer et déplacer les fichiers.



Exploration d'un repository

C'est également depuis là que vous pouvez copier vos sources vers une nouvelle *branche* ou un *tag*. Il vous suffit pour cela d'utiliser la fonction "Branch/Tag".

Enfin, vous avez la possibilité de faire un *checkout* d'un dossier précis depuis cette vue. Le checkout est le fait de récupérer tous les fichiers du repository ou d'un dossier de celui-ci. Vous pouvez donc faire un *checkout* d'une *branche* en particulier ou alors d'un dossier spécifique au lieu de faire un *checkout* complet. Pour ce faire, il vous suffit de choisir le dossier source de votre choix et d'y cliquer droit puis d'utiliser la fonction "checkout". Vous pouvez ensuite choisir entre effectuer le *checkout* dans un nouveau projet ou alors dans un projet existant.

6 - Conclusion

Comme vous le voyez, l'utilisation de **Subversion** avec **Eclipse** est très simple et très intuitive. Toutes les options sont à portée de main et vous pouvez faire tout ce que propose **SVN** avec une interface graphique claire.

Si ce plugin ne vous a pas conquis, vous pouvez aussi vous diriger dans la direction de **Subversive**, un autre plugin permettant de manipuler **Subversion** avec **Eclipse**. Pour plus d'informations, je vous invite à consulter le [site officiel](#).

Je tiens à remercier [kimz](#) pour sa relecture attentive de cet article.

6.1 - Liens

Pour plus d'informations sur Subversion, je vous invite à consulter le site officiel : <http://subversion.tigris.org>. Vous pouvez aussi trouver ici (<http://svnbook.red-bean.com/>) un livre gratuit en anglais sur Subversion.

Je vous invite aussi à consulter les ressources suivantes :

- [Gestion de projet avec Subversion](#)
- [FAQ Subversion](#)
- [Définition de Subversion](#)
- [Installation de Subversion sur Windows](#)
- [Subversion, par Hugo](#)
- [La FAQ des outils de versioning](#)