Formation SUBVERSION

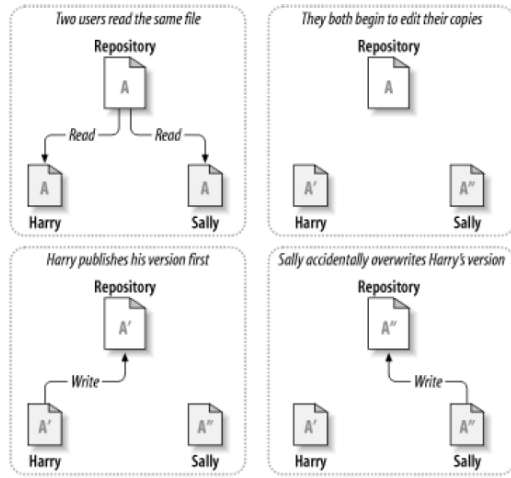
Gérer les sources avec SUBVERSION

- Introduction et problème à résoudre
- Concepts fondamentaux
- Subversion au jour le jour
- Branches et Tags
- Administration
- Client AKHSVN
- Ressources et liens utiles

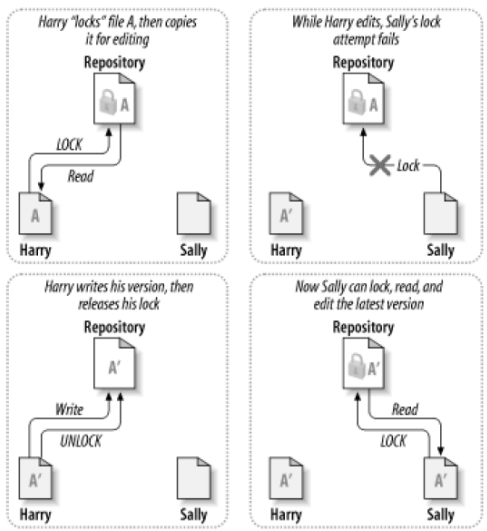
Introduction

- Subversion est un système de contrôle de versions (ou gestionnaire de sources).
- Subversion autorise plusieurs personnes à travailler sur des documents communs (chacun en ayant une copie locale)
- Subversion permet :
  - synchronisations entre les différentes versions de ces documents
  - retours arrière (undo) vers versions plus anciennes,
  - suivi modifications au cours du temps.

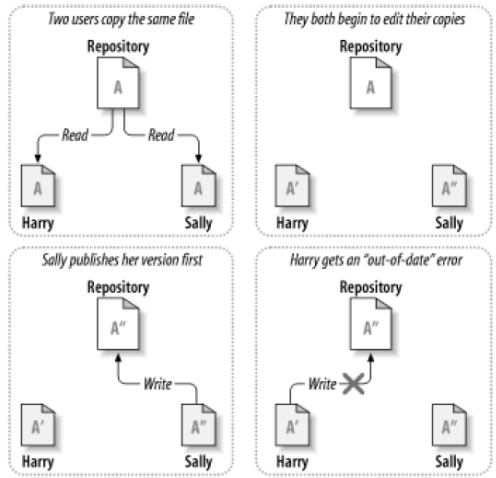
Problème à résoudre



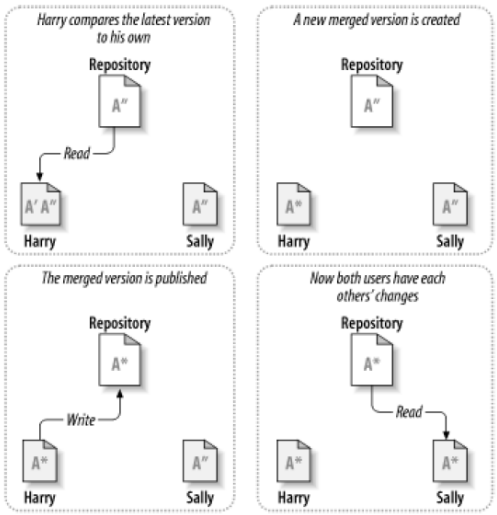
Solution 1 : le lock



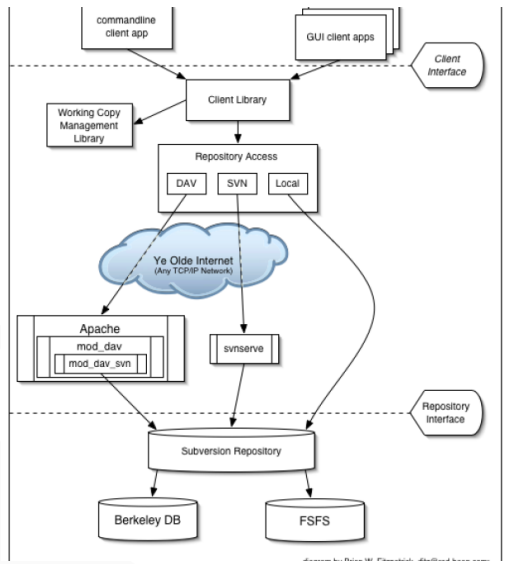
Solution 2 : copy-modify-merge (phase1)



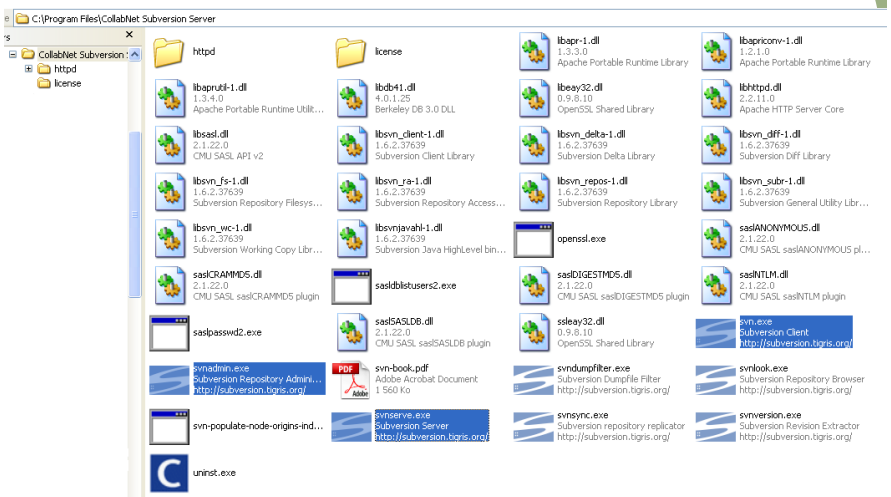
Solution 2 : copy-modify-merge (phase2)



Architecture SVN



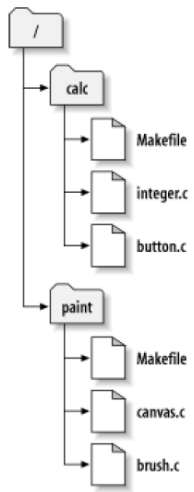
Arborescence



URL accès Repository

- svn checkout http://monsite.com:4567/repos
- svn checkout https://monsite.com:3798/repos
- svn checkout file:///var/svn/repos
- svn checkout file:///C:/var/svn/repos
- **svn checkout svn://localhost/repository1**
- svn checkout ssh+svn://localhost/repository/monprojet

Copie de travail (svn checkout)



Subversion au jour le jour

- Mettre à jour votre copie de travail
  - svn update
- Réaliser des modifications
  - svn add
  - svn delete
  - svn copy
  - svn move
- Examiner vos changements
  - svn status
  - svn diff
- Publier vos changements
  - svn commit

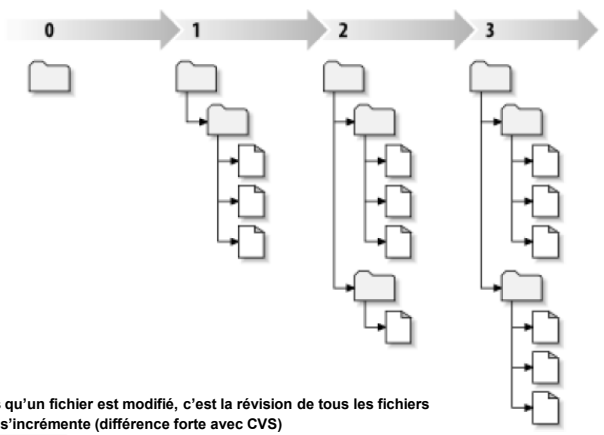
Désactivation cache login/passwd

- Par défaut, et pour vous éviter à chaque fois de fournir login/pwd, Subversion met en cache version criptée login/pwd
- Désactivation temporaire (par commande) : param `-no-auth-cache`
- Vider cache :
  - Linux : $HOME/ .subversion/auth
  - Windows : %USER_HOME%/Appli_Data/Subversion/auth
- Fichier de configuration SVN () : ajouter
  - store-auth-creds = no

Repository & projets

- Création :
  - svnadmin create c:\svn_repository\repository1
- Import d'un projet :
  - **svn** —username douglas —password java **import** monprojet1 **svn://localhost/repository1/monprojet1** -m "Import initial projet 1"
- Contrôle d'accès au repository
  - conf\svnserve.conf et conf\passwd
- Organisation repository & projet
  - 1 repository par projet ?

Révisions



Dès qu'un fichier est modifié, c'est la révision de tous les fichiers qui s'incrémente (différence forte avec CVS)

Révisions : mots clés
HEAD et BASE

- HEAD : Numéro dernière révision (la + récente)
  - svn diff -r HEAD
    - Compare votre copie de travail (dont les modifs) avec la dernière révision serveur
  - svn log -r HEAD
    - Montre message pour le dernier commit
- BASE : Numéro révision dans la copie de travail. Si fichier modifié, représente num version avant modif.
  - svn log -r BASE:HEAD test.java
    - Montre messages de commit du fichier depuis la dernière mise à jour
  - Svn diff -r BASE:15 test.java
    - Compare version non modifiée de test.java avec la version de test.java en rev 15

Lock

- Locker :
  - **svn lock test.java** (Status 'O' → Other)
- Investiguer sur un lock (suite à echec commit) :
  - svn info test.java
  - svn status -u
- Delocker (quiconque) : svn unlock --force
- Lister les locks (admin) : svnadmin lslocks /svn/repos
- Supprimer lock : svnadmin rmlock test.java

Révisions : les dates

1

```
$ svn log -r {2006-11-28}
------------------------------------------------------------------------
r12 | ira | 2006-11-27 12:31:51 -0600 (Mon, 27 Nov 2006) | 6 lines
```

2

```
$ svn checkout -r {2006-02-17}
$ svn checkout -r {15:30}
$ svn checkout -r {15:30:00.200000}
$ svn checkout -r {"2006-02-17 15:30"}
$ svn checkout -r {"2006-02-17 15:30 +0230"}
$ svn checkout -r {2006-02-17T15:30}
$ svn checkout -r {2006-02-17T15:30Z}
$ svn checkout -r {2006-02-17T15:30-04:00}
$ svn checkout -r {20060217T1530}
$ svn checkout -r {20060217T1530Z}
$ svn checkout -r {20060217T1530-0500}
```

3

```
$ svn log -r {2006-11-20}:{2006-11-29}
```

Révisions : les 'propriétés'

- Vous pouvez ajouter, supprimer, modifier métadonnées (clé=val) associées à fichier / répertoire
- Exemple : associer un numéro de bug tracking

1
```
$ svn propset copyright '(c) 2006 Red-Bean Software' calc/button.c
property 'copyright' set on 'calc/button.c'
$
```

2
```
$ svn propset license -F /path/to/LICENSE calc/button.c
property 'license' set on 'calc/button.c'
$
```

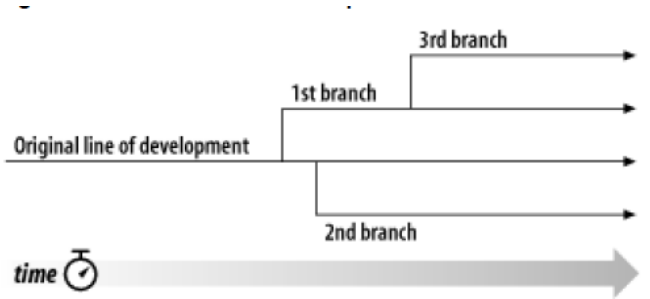
3
```
$ svn proplist calc/button.c
Properties on 'calc/button.c':
  copyright
  license
$ svn propget copyright calc/button.c
(c) 2006 Red-Bean Software
```

4
```
$ svn propdel license calc/button.c
property 'license' deleted from 'calc/button.c'.
$ svn proplist -v calc/button.c
Properties on 'calc/button.c':
  copyright : (c) 2006 Red-Bean Software
$
```

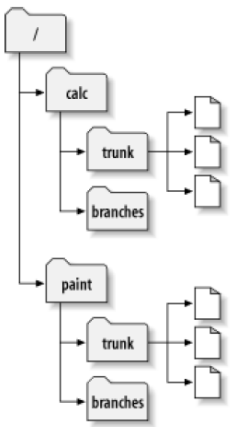

Révisions : mots clés HEAD et BASE

- HEAD : Numéro dernière révision (la + récente)
  - svn diff -r HEAD
    - Compare votre copie de travail (dont les modifs) avec la dernière révision serveur
  - svn log -r HEAD
    - Montre message pour le dernier commit
- BASE : Numéro révision dans la copie de travail. Si fichier modifié, représente num version avant modif.
  - svn log -r BASE:HEAD test.java
    - Montre messages de commit du fichier depuis la dernière mise à jour
  - Svn diff -r BASE:15 test.java
    - Compare version non modifiée de test.java avec la version de test.java en rev 15

Branches et Tags

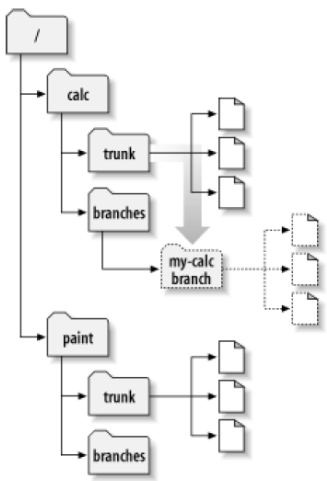


Branches et Tags



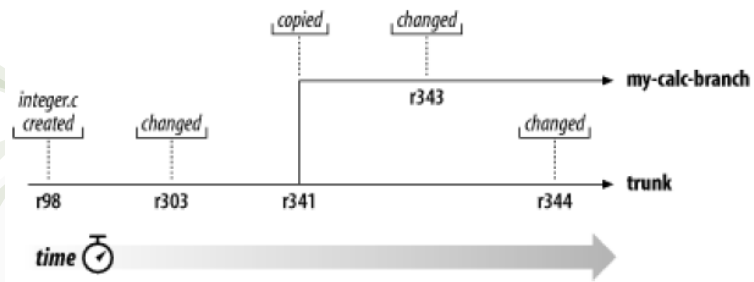
Bonne pratique : répertoires trunk, tags, branches pour chaque projet

Branche



Création d'une branche

svn copy http://svn.example.com/repos/calc/trunk
http://svn.exeample.com/repos/calc/branches/mabranche-calc
-m « création d'une branche de trunk»



Branches et tags

- Création d'un tag
  - svn copy monprojet/trunk monprojet/tags/**release-1.0-snapshot**
- Création d'une branche de développement
  - svn copy monprojet/trunk monprojet/branches/**prepa-dev-1.1**
- Se mettre sur la branche :
  - svn switch monprojet/branches/**prepa-dev-1.1**
- Mise à jour branche (en étant sur la branche)
  - svn merge monprojet/trunk (ajouter - - dry-run pour simulation)
  - svn status
  - svn commit -m 'Merge dernères modifs du trunk'
- Réintégration vers trunk (en étant sur trunk)
  - svn merge - -reintegrate monprojet/branches/prepa-dev-1.1
  - Svn commit -m 'réintégration de la branche vers le trunk !'

Branches et tags

- Bonne pratique : synchroniser régulièrement trunk vers branche
  - Evite conflit lors reintegration branche → trunk
- Supprimer la branche après la réintegration
  - svn delete monprojet/trunk monprojet/branches/**prepa-dev-1.1**
- Retrouver informations branches
  - svn log /branches
- Informations de merges (à partir d'une branche)
  - svn propget svn:mergeinfo

Exemple HOOK 1 : envoi mail après commit

```
#!/bin/sh

REPOS="$1"
REV="$2"

SUBJECT=`\
  echo -n "[SVN commit] R-$REV Log:";\
  svnlook log --revision \
        $REV /home/webadmin/htdocs/repository | head -n1`
MESSAGE=`\
  echo -n "REPOS:$REPOS ; REV:$REV ; AUTHOR:";\
  svnlook author --revision $REV /home/webadmin/htdocs/repository; \
  echo ; echo "LOG:";\
  svnlook log --revision $REV /home/webadmin/htdocs/repository;\
  echo;echo "LISTE DES FICHIERS:";echo "-------------";\
  svnlook changed --revision $REV /home/webadmin/htdocs/repository`

/usr/local/bin/sendEmail.pl -f subversion@kitpages.com \
  -t webmaster@kitpages.com \
  -u "$SUBJECT" -m "$MESSAGE" >> /dev/null
```


Exemple HOOK 2 : Interdire commentaires vides

```
#!/bin/sh

REPOS="$1"
TXN="$2"

SVNLOOK=/usr/local/bin/svnlook
$SVNLOOK log -t "$TXN" "$REPOS" | \
   grep "[a-zA-Z0-9]" > /dev/null
if [ $? -ne 0 ] ; then
  echo "Mettez un message dans le commit.">/dev/stderr
  exit 1
fi

exit 0
```

Exemple HOOK 3 :
Merge d'une branche sur une autre

* On travaille sur la branche 2.7 d'un produit (http://svn.xxx.com/myProduct/2.7)

* la branche 2.7 a été créé par copie de la branche 2.6 lors de la révision 5234

* La version en production est la 2.6

* Un bug est repéré sur la version 2.6 et corrigé par un développeur sur la branche 2.6

* L'idée est de réintégrer cette modification sur la branche 2.7,
  ainsi que toutes les modifications de la branche 2.6 depuis la création de la branche 2.7

```
# aller dans le répertoire de la branche 2.7 sur sa machine locale
cd ..../myProduct/2.7
# appliquer les modifications depuis la création de la branche
svn merge -r 5234:HEAD http://svn.xxx.com/myProduct/2.6 .
# s'il n'y a pas de conflit, vérifier que tout
# marche bien et commiter le tout sur la branche 2.7
svn commit -m "merge [Rev:5234:5xxx] myProduct/2.6->myProduct/2.7"
```


Administration Subversion

- **svnadmin**
  - Création de repository
  - Opérations de maintenance (dump / load)
    - svnadmin dump
    - svnadmin load
- **svnlook**
  - Examiner révisions (-r) & transactions (-t)
  - Utilisé typiquement par hooks (pré/post commit)
  - svnlook - -revision, svnlook –t
  - **svnlook info /var/svn/repos –r 19**
    - auteur, date , nb caractère dans log

Administration Subversion

1

```
$ svnlook youngest myrepos
26
$ svnadmin dump myrepos > dumpfile
* Dumped revision 0.
* Dumped revision 1.
* Dumped revision 2.
…
* Dumped revision 25.
* Dumped revision 26.
```

2

```
$ svnadmin load newrepos < dumpfile
<<< Started new txn, based on original revision 1
     * adding path : A ... done.
     * adding path : A/B ... done.
     …
------- Committed new rev 1 (loaded from original rev 1) >>>

<<< Started new txn, based on original revision 2
     * editing path : A/mu ... done.
     * editing path : A/D/G/rho ... done.

------- Committed new rev 2 (loaded from original rev 2) >>>

…

<<< Started new txn, based on original revision 25
     * editing path : A/D/gamma ... done.

------- Committed new rev 25 (loaded from original rev 25) >>>

<<< Started new txn, based on original revision 26
     * adding path : A/Z/zeta ... done.
     * editing path : A/mu ... done.

------- Committed new rev 26 (loaded from original rev 26) >>>
```

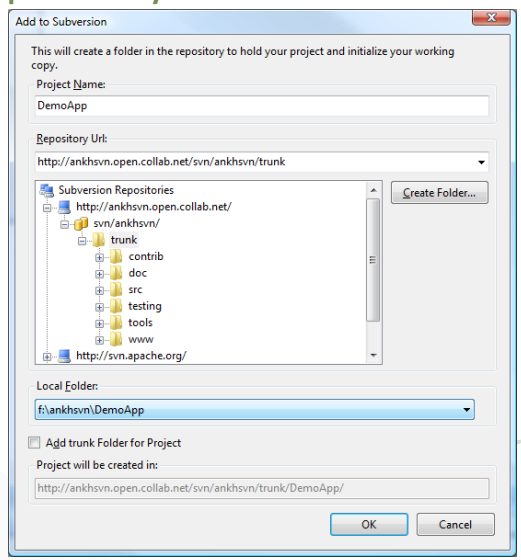

SUBVERSION dans une chaine d' intégration continue

- Technique puissante permettant dans le cadre du développement d'un logiciel en équipes:
  - Garder en phase les équipes de dév.
  - Limiter risques
  - Limiter la complexité
- A intervalles réguliers, vous allez construire (build) et tester la dernière version de votre logiciel
- Parrallèlement, chaque développeur teste et valide (commit) son travail en ajoutant son code dans un lieu de stockage unique.

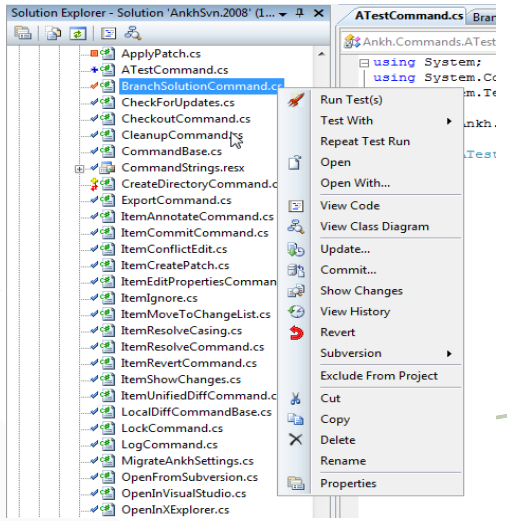
Fonctionalités clés d'un client Subversion

- Navigation dans un le repository
- Historique des modifications d'un document
- Statut d'un document
- Différentiel entre 2 versions d'un document
- Merge entre 2 documents (gestion conflit)
- Branchement sur une version révision donnée

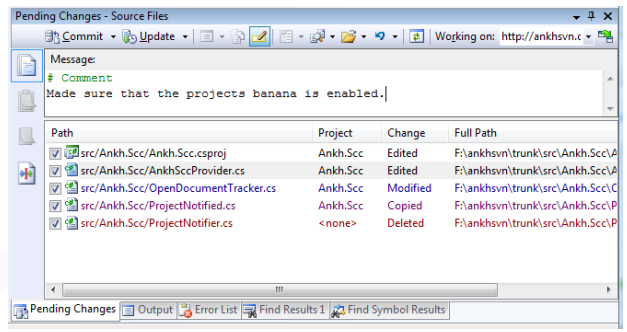
Client Visual studio ANKHSVN : Repository SUBVERSION



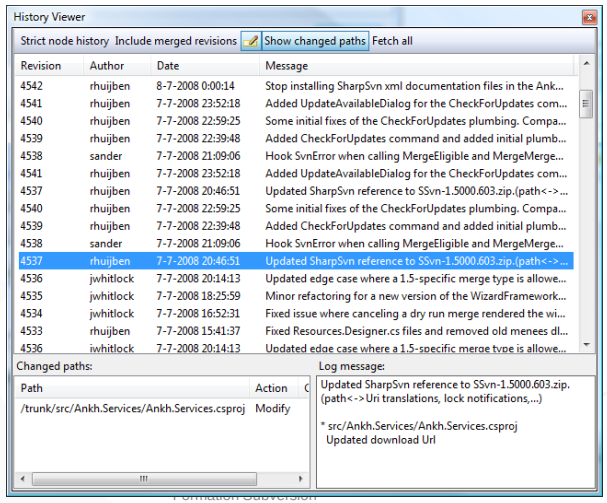
Client Visual studio ANKHSVN
Explorateur de Solution / Menu SVN



Client Visual studio ANKHSVN
Voir les changements

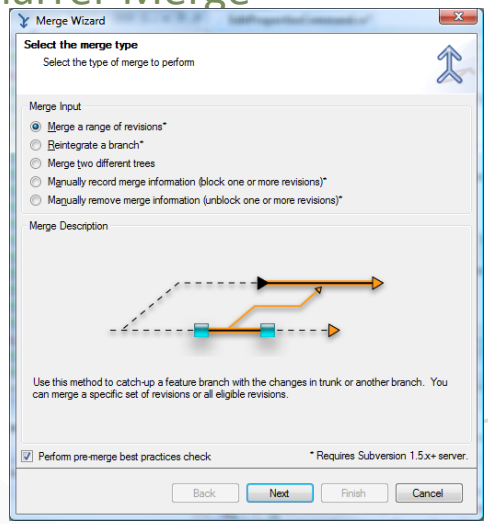


Client Visual studio ANKHSVN
Voir historique modifications

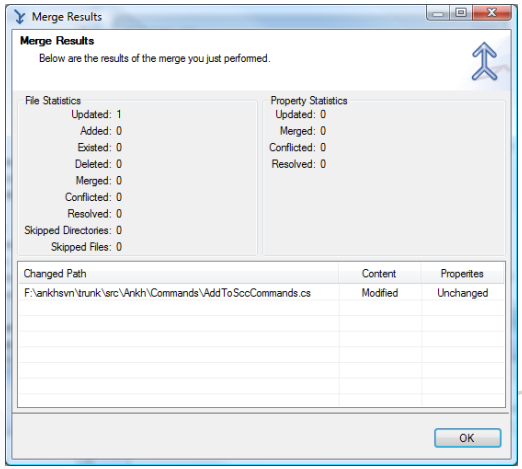


Client Visual studio ANKHSVN

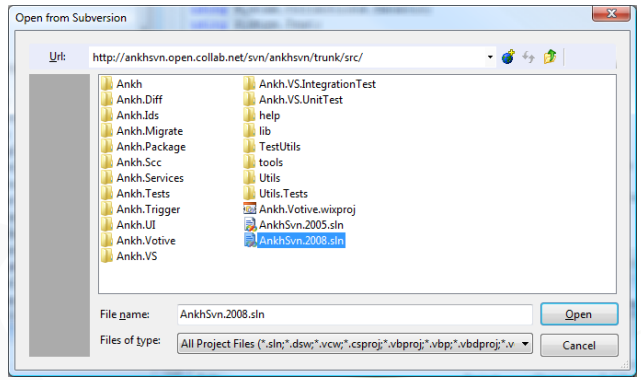
démarrer Merge



bilan Merge

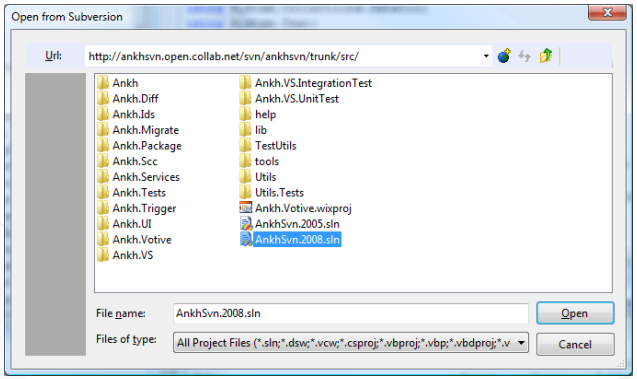


Ouvrir à partir de Subversion



Client Visual Studio ANKHSVN

Explorateur de repository



Liens utiles SUBVERSION

- PDF SVN BOOK : http://svnbook.red-bean.com/
  - Livre éditions O'Reilly – 420 pages

- Site Subversion : http://subversion.tigris.org/
- Client SVN pour Visual Studio :
  - Visual SVN (client et serveur) : http://www.visualsvn.com/
  - ANKHSVN http://ankhsvn.open.collab.net/
- Client svn windows Tortoise SVN
  - http://tortoisesvn.tigris.org/