TD n° 2
Gestion de versions
Subversion

1 Introduction

Subversion est un système de contrôle de versions (ou gestionnaire de sources). Il autorise plusieurs personnes à travailler sur des documents communs, chacun ayant une copie locale, et permet :

- des synchronisations entre les différentes versions de ces documents,
- des retours en arrière vers des versions plus anciennes,
- un suivi des modifications au cours du temps.

Il fonctionne selon un modèle copie-modifie-fusionne, c'est à dire qu'un utilisateur :

- sort une copie des fichiers du dépôt dans un espace de travail local ('working copy'),
- modifie ces fichiers localement même hors connexion, puis
- demande lorsqu'il le veut que ses modifications soient reportées dans le dépôt par un commit.
- Il peut aussi demander que sa version locale de travail soit re-synchronisée avec la dernière version du dépôt pour récupérer les modifications des collègues.

2 Installation serveur SVN

Il y a 2 moyens d'accéder à vos sources avec SVN :

- A travers le service **svnserve** (à l'écoute sur le port 3690) que vous pouvez lancer en ligne de commande ou en tant que service Windows
- A travers le module **mod_dav_svn** du serveur Apache, préconfiguré et déja opérationnel sur le port 80, qui rendra vos sources accessible après authentification à travers un navigateur web.

1. Ouvrez un invite de commande MS-DOS etentrez la commande : svn
   La commande devrait être prise en compte sinon vous devrez ajouter le répertoire d'installation de SVN dans la variable d'environnement path.
2. Vérifiez la création du répertoire de stockage des sources (exemple c:\svn_repository ou c:\csvn). Le répertoire est vide car aucun stockage (encore apellé 'repository') n'a été créé.
3. Analysez le contenu du répertoire d'installation :
   Observez en particulier les exécutables suivant :
   - **svn.exe** : exécutable qui sera le principal utilisé pour travailler avec le serveur. Il s'agit d'un **client** subversion, utile pour récupérer sources, valider changement...
   - **svnadmin.exe** : exécutable qui servira à créer les répertoires de stockage des sources de chacun de vos projets
   - **svnserve.exe** : l'exécutable qui lance le **processus** svn (démon) qui rendra vos sources accessibles après authentification sur un port de l'ordinateur (par défaut 3690)

3 Création d'un repository

Vous allez ranger vos projets dans des **dépôts** appelés 'repository'. Un repository est physiquement un **répertoire** possédant une arborescence précise imposée par Subversion. La commande **svnadmin**, vous permet de créer un repository Subversion.

1. Créez avec la commande svnadmin un repository Subversion nommé 'repository1'
   Ouvrez une Invite de commande, lancez la commande :
   svnadmin create c:\svn_repository\repository1
2. Notez que cette commande a créé un répertoire 'repository1' dans le répertoire c:\svn_repository
   Analysez l'arborescence et identifiez le répertoire de configuration de l'authentification SVN.

- ***répertoire conf :*** Chaque fois qu'une personne essaie d'accéder un de vos repository (via **svnserve** ou via **Apache**), Subversion examine le fichier **svnserve.conf** pour connaître votre **stratégie** d'accès en lecture / écriture. Plus loin dans la partie authentification Subversion vous allez mettre en œuvre une stratégie d'authentification qui suffit pour la plupart des projets.

- ***répertoire hooks***

  Ici sont stockés les scripts d'administration , permettant par exemple de déclencher des actions (Ex : envoi de mail) en fonction d'événements du cycle de vie du projet. Vérifiez l'existence de 3 scripts correspondant à des actions à déclencher avant (pre-commit), pendant (start-commit) et après (post-commit) chaque modification du repository :
  - pre-commit.tmpl
  - start-commit.tmp
  - post-commit.tmpl

4 Authentification Subversion

4.1 Stratégie basée sur le fichier passwd

vous allez configurer une **stratégie d'authentification** Subversion par défaut, basée sur un fichier **passwd**. C'est le fichier **svnserve.conf** qui va vous permettre de fixer cette stratégie d'authentification, aussi bien pour les accès **anonymes** que pour les utilisateurs **authentifiés**.

1. Allez dans le répertoire conf de l'arborescence du repository SVN 'repository1'.
2. Ouvrez le fichier svnserve.conf
3. Le contenu par défaut de svnserve.conf est le suivant.

```
[general]
### The anon-access and auth-access options control access to the
### repository for unauthenticated (a.k.a. anonymous) users and
### authenticated users, respectively.
### Valid values are "write", "read", and "none".
### Setting the value to "none" prohibits both reading and writing;
### "read" allows read-only access, and "write" allows complete
### read/write access to the repository.
### The sample settings below are the defaults and specify that anonymous
### users have read-only access to the repository, while authenticated
```

```
### users have read and write access to the repository.
# anon-access = read
# auth-access = write
### The password-db option controls the location of the password
### database file.  Unless you specify a path starting with a /,
### the file's location is relative to the directory containing
### this configuration file.
### If SASL is enabled (see below), this file will NOT be used.
### Uncomment the line below to use the default password file.
# password-db = passwd
### The authz-db option controls the location of the authorization
### rules for path-based access control.  Unless you specify a path
### starting with a /, the file's location is relative to the the
### directory containing this file.  If you don't specify an
### authz-db, no path-based access control is done.
### Uncomment the line below to use the default authorization file.
# authz-db = authz
### This option specifies the authentication realm of the repository.
### If two repositories have the same authentication realm, they should
### have the same password database, and vice versa.  The default realm
### is repository's uuid.
# realm = My First Repository
```

1. Décommentez les 3 lignes suivantes:

— **anon-access = read** (accès en lecture pour utilisateurs anonymes)
— **auth-acces = write** (accès en écriture pour utilisateurs authentifiés)
— **passwd-db = passwd** (validation des login/pwd dans fichier conf\passwd)

2. Ouvrez le fichier passwd et identifiez la section 'users' du fichier

```
### This file is an example password file for svnserve.
### Its format is similar to that of svnserve.conf. As shown in the
### example below it contains one section labelled [users].
### The name and password for each user follow, one account per line.

[users]
# harry = harryssecret
# sally = sallyssecret
```

3. Modifiez les utilisateurs et leurs login/pwd de la façon suivante.
   Veillez à ne pas mettre d'espace en début de ligne.

```
saida = saida
ali = ali
```

Vous venez de configurer deux utilisateurs ('saida' et 'ali') ayant accès en écriture à votre repository Subversion nommé 'repository1'.
Ils auront accès en écriture à tous les projets que vous allez créer dans votre repository.

5 Affiner la sécurité d'accès

Il y a possibilité d'affiner à une personne l'accès en lecture / écriture sur un fichier / répertoire spécifique.

1. Editez le fichier conf\authz

```
### (optional) repository specified by the section name.
### The authorizations follow. An authorization line can refer to:
###  - a single user,
###  - a group of users defined in a special [groups] section,
###  - an alias defined in a special [aliases] section,
###  - all authenticated users, using the '$authenticated' token,
###  - only anonymous users, using the '$anonymous' token,
###  - anyone, using the '*' wildcard.
###
### A match can be inverted by prefixing the rule with '~'. Rules can
### grant read ('r') access, read-write ('rw') access, or no access
### (").

[aliases]
# joe = /C=XZ/ST=Dessert/L=Snake City/O=Snake Oil, Ltd./OU=Research Institute/CN=Joe Average

[groups]
# harry_and_sally = harry,sally
# harry_sally_and_joe = harry,sally,&joe

# [/foo/bar]
# harry = rw
# &joe = r
# * =

# [repository:/baz/fuz]
# @harry_and_sally = rw
# * = r
```

Expliquez.

6 Import de votre projet

Créez un répertoire 'monprojet1' dans un répertoire temporaire, ex : c:\tmp. Nous allons maintenant importer ce projet dans subversion, en utilisant l'utilisateur '**saida**'.

Démarrage service svnserve

1. Vérifiez que le service subversion svnserve est bien démarré (Services Windows).

Import

1. Ouvrez un nouvel invite de commande et déplacez-vous jusqu'au répertoire 'tmp'.
2. Lancez la commande :

svn import —username saida —password saida monprojet1 svn://localhost/repository1/monprojet1 -m "Import initial projet 1"

Voici l'explication de la commande svn import ci-dessus :

- Vous vous authentifiez (**username/password**) pour pouvoir accéder au repository en écriture
- Vous utilisez la commande **svn import**, dédiée à cette tache d'import dans svn
- Vous recherchez un repository SVN sur la machine locale (**localhost**)
- Vous utilisez le protocole **svn://** en vous connectant au **port par défaut** de svn sur la machine locale
- Vous souhaitez importer l'arborescence de monprojet1 dans le repository **repository1**
- Vous spécifiez un message (**-m** "Import initial projet 1") qui permettra à toute personne

ayant accès au répository de savoir le contexte de création de cette modification du repository.
3. Vérifiez la sortie produite par la commande.

7 Récupération des sources avec plusieurs clients

7.1 Client ligne de commande : utilisateur ali

Avec l'invite de commande, positionnez-vous dans le répertoire c:\utilisateurs\ali , afin de simuler que nous sommes sur l'ordinateur de Morad et que celui-ci souhaite récupérer les sources déposées par l'utilisateur saida.
Compte-tenu de notre stratégie d'authentification, nous pouvons accéder aux sources sans s'authentifier (**accès anonyme**).

1. Lancez le commande :
   svn checkout svn://localhost/repository1/monprojet1 projet1
2. Vérifiez la sortie de la commande.
   La lettre A au début de chaque ligne signifie 'Ajout'.
   Vérifiez que vous avez récupéré une 'copie de travail' du monprojet1.
3. Modifiez la configuration de votre explorateur windows afin de voir les fichiers cachés.
4. Vérifiez que subversion a créé un fichier .svn dans le projet récupéré. C'est ce fichier qui va permette à Subversion d'identifier les modifications faites en local et les comparer avec celles du serveur.

7.2 client eclipse (plugin subclise) utilisateur ahmed

Pour gagner du temps et simplifier la gestion des sources, Ali a décidé de récupérer les sources directement à partir d'Eclipse, en utilisant le plugin subclipse.

1. Installer le plugin SVN pour eclipse si ce n'est pas déjà fait
2. Réalisez la manipulation de récupération de la copie de travail pour ahmed :
   File → New → Project → SVN → Check out project from SVN → reseignez l'url →
   Sélectionnez le dossier monprojet → include in workspace
3. Vérifiez ici le résultat de la récupération des sources dans son environnement de développement.
   Quelle est la valeur ajouter de l'intégration du client SVN à eclipse ?

8 Suivi des modifications et synchronisation

On va comprendre comment suivre les modifications effectuées dans un repository. On va utiliser les éléments visuels nous permettant avec Eclipse de comprendre si un fichier de la copie de travail n'est pas dans le même état que la version courante sur le serveur et faire une synchronisation .

On va modifier et analyser l'historique des modifications d'un fichier

1. Cliquez-droit sur file.txt → team → show history
   Notez le numéro de révision, la date de révision, l'auteur et le commentaire
2. Modifiez le contenu fichier file.txt
3. clique-droit sur file.txt → team → commit
4. Vérifiez le nouveau historique
5. Refaites une deuxième modification
6. Cliquez-droit sur file.txt → team → synchronize with repository → cliquez Yes

7. Notez la fenêtre qui affiche les comparaisons
   Quelle est la valeur ajoutée d'une synchronisation ?

9 Exercices

1. **Conflits.** Utilisez deux clients séparés sur deux machines, ou une machine avec deux workspaces, pour créer une situation de conflits solvable automatiquement et une autre situation solvable manuellement ; et résoudre ces conflits.
2. **Diff.** Utilisez svn diff -r pour voir les différences entre deux versions.
3. **Releases.** Au lieu d'utiliser les numéros de révision (qui sont individuels pour chaque fichier géré par
   SVN), il est possible d'étiqueter un ensemble de données avec un nom de version commun, et d'utiliser ensuite cette étiquette pour récupérer les données. Etiquetez votre version courante avec le tag V2015-VERSION-COMMERCIALISE, puis committez.
   svn copy tagname fichiers
   Ensuite refaites quelques modications (avec un commit), puis récuperez la version du tag.
4. **Branches.** La commande copy permet aussi de faire des branchements.
   svn copy -r tagname fichiers
   Revenez a une version antérieure de votre projet et créez une nouvelle branche, avec des modifications nouvelles que vous enregistrerez avec commit. Constatez avec svn status que les numéros de révisions ont été allongés.
5. Comment récupérer un fichier du dépôt, détruit localement par erreur ?
6. Comment récupérer un fichier supprimé du dépôt ?
7. Faut-il toujours versionner tous les fichiers d'un projet ?
8. Conseillez-vous de faire des commit par fichiers ou des commit logiques ?
9. Comment éviter le plus possible les conflits lors de propagation de données vers le serveur ?
10. Quelle est la différence entre checkout et update ?
11. Quelles sont les différences entre le tronc d'un projet (trunc), les branches d'un projet (branch) et les distributions d'un projets (tag ou releases) ?