GL02 TD#6 : Subversion et développement coopératif – This is not Rocket Science!

!!! Ce tp est à réaliser par groupe projet (2 x 2 pers) avec quelques consignes initiales légèrement différentes par binômes.

Préparation et configuration

Pour les machines Mac de M102, aucune configuration particulière n'est nécessaire.

Pour installer SVN sur PC, consultez le guide d'installation et de configuration disponible sur Moodle après avoir créé votre compte sur SourceForge (étape 1) :

https://elearning.utt.fr/mod/resource/view.php?id=44575

0. Document de références et aide sur SVN

Les commandes suivantes vous permettent d'accéder à la documentation de SVN.

svn help

svn help <cmd>

A toute fin utile : http://svnbook.red-bean.com/fr/1.8/

http://svnbook.red-bean.com/en/1.8/

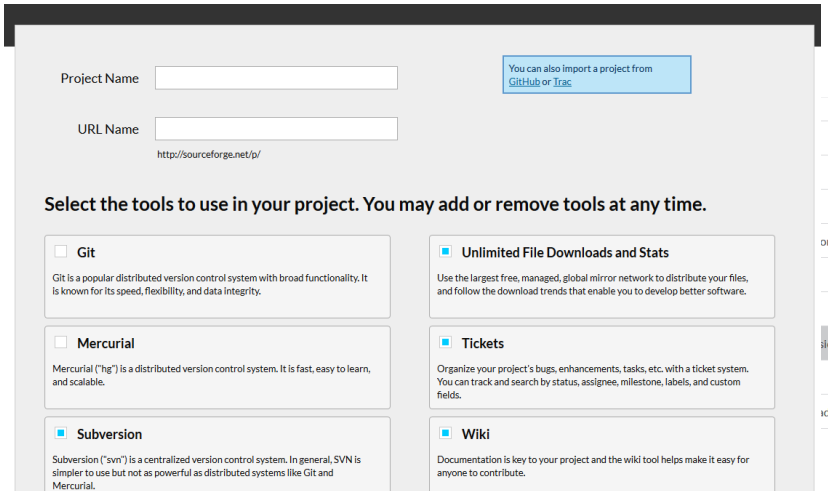
1. Créer un compte sur SourceForge

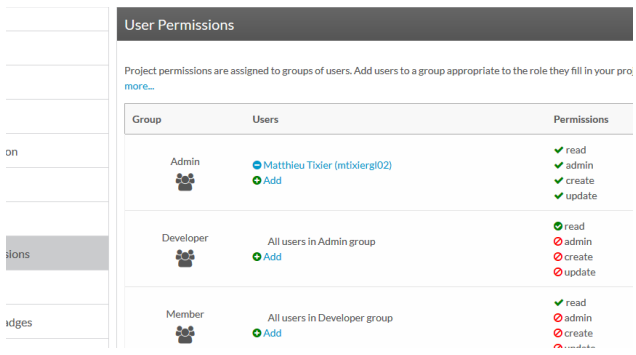
Rendez-vous sur http://sourceforge.net et créez un compte si vous n'en disposez pas déjà d'un.

2.a Création du dépôt (charge du binôme du responsable de projet)

Créez un projet sur SourceForge.

- Allez sur votre page « profile » et cliquez sur le lien « Add project » (il est possible que l'on vous demande votre numéro de téléphone pour activer le projet).
- Choisissez Subversion (SVN) comme système de gestion de version. Prenez en plus au moins les modules Wiki, Ticket, Files & Stats





- (figure de gauche) Renseignez le nom du projet (<nom équipe>_gl02, soit le nom d'équipe suffixé par "_gl02") ainsi qu'une description minimale à propos du sujet **sur lequel votre équipe ne fait pas le cahier des charges.**

- (figure de droite) Invitez les membres du projet (identifiants source forge) ainsi que votre chargé de TP avec les droits admin.

- Réalisez les commandes suggérées par Sourceforge pour créer les dossiers par défaut (trunk, branches, tags) à la racine de votre dépôt.

```
svn checkout --username=<sf_username> svn+ssh://<user>@svn.code.sf.net/p/<project>/code/
<project>

cd <project>
mkdir trunk branches tags
svn add trunk branches tags
svn commit -m "Add initial directories"
```

2.b This is not Rocket Science! (pour tous et en parallèle de 2.a)

Comme exercice de prise en main de SVN, vous allez avec votre équipe construire une fusée (virtuelle en javascript). Chaque membre de l'équipe aura pour responsabilité d'implémenter un composant de la fusée :

engine.js (pour le responsable d'équipe)
command.js
radio.js
(selon le nombre de membre de l'équipe, également : satellite1.js et satellite2.js)

Si chacun réalise correctement son composant et le met sur le dépôt, vous devriez réussir à faire décoller la fusée de votre équipe.

Chaque membre doit créer le fichier de son composant (ie : `<composant>.js` avec l'éditeur nano du terminal ou tout autre que vous préférez.

Le squelette du code vous est donné ci-après :

```
var Engine = {
	check: function(){
	console.log("Engine Ready !")
	return true
	}
}
module.exports = Engine
```

3. Création de l'espace local de travail – svn checkout (chaque membre)

- Rendez-vous sur Sourceforge dans l'espace du projet de votre équipe.

- Chaque membre de l'équipe va créer son espace de travail local sur la machine à partir de l'adresse du dépôt. Ouvrez le terminal (bash) et placez-vous dans ~/Documents

- Vous pouvez utiliser les commandes générées dans l'onglet « SVN », assurez-vous seulement d'être bien identifié sur Sourceforge et d'être en **mode RW (Read and Write) et non en RO (Read Only)**.

  ```
  svn checkout <URI_projet_sf> <nom_TP6>
  ```

  Vous devriez constater sur votre machine la création du répertoire <nom_TP6>. Si votre responsable d'équipe a été efficace, vous devriez trouver un dossier "trunk". Dans votre copie locale se trouve également un répertoire caché ".svn" qui confirme qu'il s'agit d'une copie locale de travail SVN.

- Conservez un navigateur ouvert sur la page du projet pour observer les effets des manipulations qui suivront sur le dépôt. Vous pouvez voir SVN comme un système de fichiers distant auquel vous avez accès et devez demander régulièrement de se synchroniser suivant les évolutions du travail de chaque membre de l'équipe.

4. Ajouter un fichier à l'espace de travail – svn add (chaque membre)

- Une fois que votre fichier js décrivant le composant de fusée que vous avez réalisé est prêt, copiez le dans le dossier trunk de votre espace de travail local (la commande bash est : `cp <source> <destination>`. La commande sous DOS est : `copy <source> <destination>`).

- Afin qu'il soit pris en compte pour le projet et potentiellement envoyé sur le dépôt, vous devez l'ajouter explicitement avec la commande add. Placez-vous dans le dossier trunk de votre espace de travail :

  ```
  svn add <composant.js>
  ```

Votre fichier fait désormais parti du *cycle de révisions* **de votre copie du projet. Vous ne devez plus le supprimer ou le renommer sans passer par les commandes SVN au risque de créer un état inconsistant du projet (générateur de nombreuses erreurs).**

De la même façon que vous venez d'ajouter un fichier au cycle de révisions, vous pouvez également le supprimer en utilisant la commande : `svn delete --keep-local <composant.js>`

- Suite à l'ajout du fichier, vous pouvez observer l'état des écarts entre votre copie locale de travail et le dépôt avec (? = fichier non associé, A = fichier à ajouter, D = fichier à supprimer):
  `svn status`

- Après avoir constaté que le fichier <composant.js> est un ajout par rapport au dépôt, supprimer le avec :
  `svn delete --keep-local <composant.js>`

  Il est important d'utiliser l'option `keep-local` sinon le fichier sera vraiment supprimer en plus d'être enlevé du cycle de révision. Par ailleurs SVN vous indiquera un risque d'erreur de commande pouvant entraîner la suppression d'un fichier.

- Constater le résultat avec `svn status` et ajoutez-le pour de bon. Notez au passage que le dépôt distant n'a été aucunement affecté par les commandes que nous venons de faire.

5. S'assurer d'avoir un espace de travail à jour – svn update (chaque membre)

Il est temps maintenant de déposer votre composant sur le dépôt distant. Pour se faire vous allez demander à transférer les changements de votre copie locale de travail sur le dépôt.

Cependant avant de pouvoir effectuer ce transfert vous devez vous assurez que votre copie locale de travail est à jour. C'est-à-dire qu'elle comprend bien les derniers changements qui auraient pu être réalisé par d'autres entre temps.

- Vous pouvez vérifier la différence de numéro de révision entre votre copie et le dépôt.

  `svn info` (vous indique le numéro de revision de votre copie de travail)

  `svn info –r HEAD` (vous indique le numéro de revision de la dernière version sur le dépôt)

  `svn diff –r HEAD` (vous montre en détail les différence entre votre copie locale et le dépôt)

- Pour mettre à jour votre copie de travail par rapport au dépôt distant utilisez :

  ```
  svn update
  ```

Comme jusqu'ici pour l'exercice chacun travaille sur un fichier à part aucun conflit ne devrait être généré. Cependant faîtes attention lors des mises à jour en cas de fichier renommé ou d'édition concurrente d'un même fichier, des conflits peuvent être remontés. Vous ne pourrez rien déposer tant qu'ils ne seront pas résolus.

6. Déposer votre espace de travail sur le dépôt distant – svn commit (chaque membre)

- Une fois votre copie locale de travail à jour, déposez là sur le dépôt distant avec :
  `svn commit –m "Ajout du composant <composant.js> dans trunk"`

  Le paramètre m, pour message, est **obligatoire** et doit contenir une courte explication sur le changement apporté. Plus qu'une nécessité technique, il s'agit d'une bonne pratique car il devient très complexe de suivre l'historique des changements d'un projet sans la documenter au fur et à mesure. Le message doit être informatif, clair et concis.

- Félicitations, vous avez déposé votre composant sur le dépôt. Maintenant il vous faut mettre à jour votre copie locale de travail pour récupérer également le travail des autres (observez que sinon aucun autre fichier n'est ajouté à votre copie de travail suite au commit).
  `svn update`

7. Procédure de lancement – svn import (responsable d'équipe seul, les autres membres observent)

Une fois que chacun a bien déposer son composant de fusée il est temps de voir si elle décolle.

- Télécharger la procédure de décollage avec l'archive « takeOffTools.zip » disponible sur Moodle : https://elearning.utt.fr/mod/resource/view.php?id=44577

- Décompressez l'archive quelque part dans ~/Documents

- Plutôt que de copier et déposer les fichiers un à un sur le dépôt, utilisez la commande svn import afin de transférer directement ces fichiers dans le trunk sur le dépôt :

  `svn import <chemin du dossier takeOffTools décompressé> <URI_projet_sf/`**`trunk`**`> -m "Ajout de la procédure de lancement"`

  Remarques : Peu importe que vous soyez dans votre copie locale de travail pour effectuer la commande, l'ajout est directement réalisée sur le dépôt. Notez que dans la mesure où l'import modifie directement le dépôt, vous devez, comme lors d'un commit ajouter un message. Le numéro de révision sera incrémenté.

- Vous devriez constater sur Sourceforge que les fichiers ont bien été ajoutés. Cependant, en procédant ainsi les fichiers ont été directement mis sur le dépôt et plus aucune copie locale de travail de l'équipe n'est à jour…

8. Resynchronisation des copies locales et décollage – svn update (chaque membre)

- Mettez à jour votre copie locale de travail avec : `svn update`

- Vous pouvez maintenant depuis votre copie locale de travail tester avec Node.js, si votre fusée décolle avec l'intégration de l'ensemble des composants de l'équipe :
  (il peut être nécessaire d'installer le module colors avant : `npm install colors`)

  `node takeoff.js`

- Félicitations si votre fusée a décollé, sinon il y a certainement quelques corrections à apporter sur un composant ou l'autre.
  Corrigez le problème en équipe sur une copie locale de travail d'un membre, faîtes un commit et synchronisez de nouveau les copies locales de travail.

9. Vous avez une promotion – Créer une branche (chaque membre)

Fort de votre premier succès aéronautique, votre équipe est promue au service qualité. Désormais au lieu de travailler chacun sur un composant, vous êtes charges de contrôler le respect de norme de qualité pour chaque composants des fusées qui sont désormais produites en série.

Chaque membre de l'équipe est en charge d'une norme portant le nom d'une couleur (blue, magenta , cyan, grey, gray). Il devra ajouter la vérification de la norme dans la méthode check de chaque composant.

```
console.log("Blue level checked".blue)
```

- Afin de pouvoir travailler sans se gêner à l'implantation de la norme il vous faut créer une ligne de code parallèle, c'est-à-dire une branche. Concrètement il s'agit de copier le contenu du répertoire dans un autre dossier de même niveau. Pour cela utilisez :

  ```
  svn copy <URI_projet_sf/trunk> <URI_projet_sf/branches/ma_branche> -m "Création de la branche ma_branche."
  ```

  Notez que cette opération impliquant un changement direct dans le dépôt, elle doit être accompagné d'un message (tout comme `svn import`).

- Mettez à jour vos copies de travail respectives pour récupérer les branches :

  ```
  svn update
  ```

- Réalisez l'implantation de votre norme et testez là avec Node.js.

- Une fois que vous êtes satisfait, partagez votre branche sur le dépôt avec la commande svn commit. Si votre commit est refusé, il est possible que ce soit dû au fait que le numéro de révision du projet soit avancé entre temps (ie vos collègues ont fait des commit), auquel cas faîtes appel à `svn update`.

10. Fusionner une branche -- svn merge (en équipe)

- Une fois que chaque membre a pu déposer sa branche, il s'agit de les passer en revue et de décider laquelle fusionner dans le tronc (on pourrait le faire avec toutes). Sur le poste du responsable d'équipe, mettez à jour une copie de travail et testez les différentes branches.

- Choisissez-en une pour la fusionner dans la ligne de code principal, trunk. Dans une copie de travail, placez-vous dans le répertoire trunk et lancez.

  ```
  svn merge <URI_branche> <chemin_trunk_copie_locale>
  ```

  Si vous ne renseignez pas le second argument c'est le répertoire courant qui sera utilisé.

  Gardez bien en tête qu'une fusion s'opère d'**une branche sur le dépôt** vers **un trunk en copie locale**. Si le résultat est satisfaisant, alors on peut faire un commit vers le dépôt de la nouvelle version de référence.

- Observez ce qui sera transmis sur le dépôt avec commit à l'aide de `svn status`.

- Si vous êtes satisfait, réaliser le commit (sinon, annulez les changements fichier par fichier, `svn revert <file>`, ou au niveau du dossier courant avec `svn revert -R .`)

- Mettez à jour votre copie de travail pour tester la nouvelle version de référence avec Node.js.

11. Synchroniser une branche avec le trunk – svn merge, revert (en équipe)

Si l'on souhaite implanter une norme supplémentaire parmi celles développées par les membres de l'équipe dans le trunk, nous nous trouvons face à un problème. Ces branches ne sont plus à jour par rapport à la version de référence dans trunk. Il ne sera donc pas possible d'utiliser merge, il vous faut les synchroniser de nouveau avec le trunk.

- Pour synchroniser une branche suite à des changements dans trunk, on utilise de nouveau la commande merge mais dans l'autre sens. Il s'agit maintenant de fusionner les dernières modifications sur trunk dans la branche.

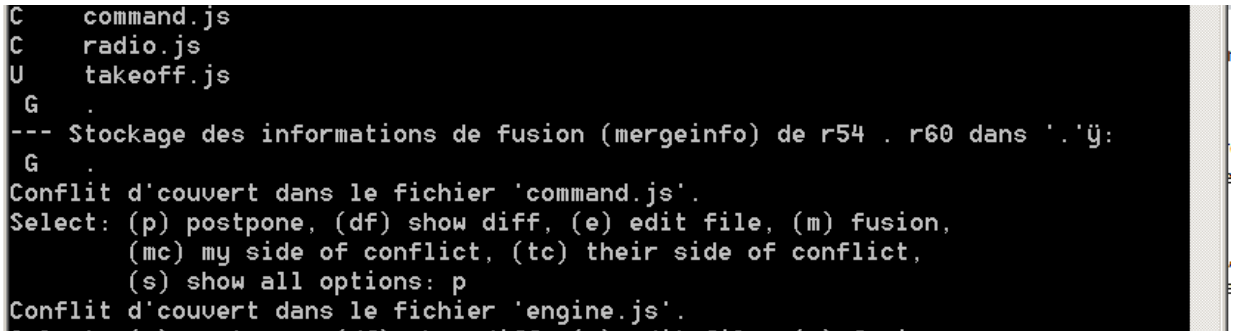
- Cependant vous devrez le faire sélectivement sur les changements qui sont intervenus depuis la dernière version déposées de la branche. Pour retrouver ce numéro de version, placez-vous dans la branche concernée :

  svn log .

- Une fois le numéro retrouvé (par exemple 49) nous pouvons demander la fusion du trunk vers la branche.

  svn merge -r 49:HEAD <URI_trunk>

- Il fort probable que des conflits soient détectés. SVN va vous proposer de les résoudre interactivement. En ce qui concerne l'exercice courant ils sont assez simples à résoudre dans la mesure où l'on souhaite mettre en application toutes les normes de niveau de couleur.



  Choisissez l'option (p) postpone pour corriger le conflit manuellement. Les fichiers en conflits seront isolé pour backup et le conflits mis en évidence dans chaque fichier.

  Corriger le fichier de sorte à ce qu'il soit dans l'état que vous le souhaitez (en l'occurrence nous voulons toute les certifications).

  La modification appliquée, nous devons indiquée à SVN que le conflit est résolu.

```
svn resolve --accept working ./engine.js
```

A l'issue de cette commande vous pouvez observer que les fichiers supplémentaire qui gardaient trace des révisions en conflits ont disparu.

- Une fois les conflits résolus, observer l'état des fichiers de votre copie de travail et testez la nouvelle version de référence avec Node.js
- Observez ce qui sera transmis sur le dépôt avec commit à l'aide de `svn status`.
- Si vous êtes satisfait, réaliser le commit (sinon, annulez les changements fichier par fichier, `svn revert <file>`, ou au niveau du dossier courant avec `svn revert -R .`)

12. Nettoyer les branches superflues.

- Une fois que votre branche a été fusionnée dans la ligne de code principale, il est préférable de la supprimer. Vous recréerez une nouvelle branche si vous avez d'autres changements conséquents à réaliser. Cela simplifie la compréhension de l'historique du projet et évite les conflits avec SVN.

  svn delete <URI_branche> `-m "Suppression de la branche ma_branche"`
- Par ailleurs votre branche n'est pas perdue. S'il y avait besoin pour une raison ou une autre de la rétablir. Il vous faut identifier le numéro de révision auquel le dossier (fonctionne aussi pour un fichier seul) était dans l'état que vous souhaitez récupérer (svn log, à l'intérieur du dossier qui contenait la ressource). Admettons, au le numéro de révision 38.
- De là le plus simple est de faire une copie de la ressource à la version identifiée vers une copie locale de travail.

  svn copy <URI_ressource>@38 <chemin_copie_locale>
- Une fois satisfait de votre copie locale de travaille vous pouvez faire un commit sur le dépôt.

13. Restaurer

Afin de nettoyer votre espace projet de l'exercice de TP, vous pouvez demander au dépôt d'être remis à n'importe quel numéro de révision précédent.

```
svn merge . -r HEAD:0
```

Notez bien le « . » pour signifier la référence au répertoire courant dans lequel on se trouve. Il faut bien entendu faire attention pour quel dossier l'on demande le merge. Les répertoires ayant subi un merge peuvent nécessiter de répéter la commande spécifiquement pour eux.

Vous pouvez constater l'ampleur des changements qui auront lieu sur le dépôt avec : `svn status`

Afin de remettre le dépôt distant à l'état initial, il vous faut bien entendu faire un commit.

```
svn commit –m "Retour à l'état initial (-r HEAD:0 racine du dépôt)"
```

Notez que l'opération ne réalise pas vraiment un retour à l'état initial, mais plutôt crée une nouvelle révision ayant les mêmes fichiers que l'état initial. Ce commit entraînera l'incrément du numéro de révision et il vous faudra donc également synchroniser de nouveau votre copie de travail avec update.

HEAD et 0 marquent respectivement la dernière et la première révision sur le dépôt, leur numéro de version. Il bien entendu très utile d'utiliser cette procédure sur des numéros de version précis. Par exemple, lorsque l'on souhaite revenir en arrière sur des suites de changements maladroits.

```
svn merge . -r 38:33

svn commit –m "Retour à l’état de la révision 33"
```