Génie logiciel pour (l'entreprise) la modélisation: Compilation automatique et Gestion de version

Jean-Baptiste Apoung Kamga

Cours M2 IM
Université Paris-sud XI

Plan

1. Gestion de version: SVN, CVS , GIT
   - Les grandes commandes de base
   - Exemple avec SVN
   - Exemple avec CVS
   - Exemple avec GIT
   - Exemple avec Hg (Mercurial)
2. Compilation Automatique: Autotools, Qmake, Cmake, Scons
   - Les grandes classes d'approches
   - Mannuels: Autotools, Qmake, Cmake, Scons, Bjam
   - Environnement intégrés: Kdevelop, Qtcreator, Eclipse, Anjuta, NetBeans
3. Gestion de Version dans les environnements intégrés

- La gestion de version est un outil indispensable dans la réalisation des projets logiciels.
- Elle soutien les points de l'XP suivants :
  - le point 2 de pratique de développement
  - le point 4 de pratique de développeur
  - le point 3 de pratique métier
- Dans son principe de fonctionnement, elle permet :
  - De stocker des états particuliers d'un projet
  - D' accéder à un état antérieur du projet en cours de développement.

Outline

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Les grandes commandes de base

L'administration ou l'utilisation d'un dépôt de gestion de versions peut se faire au moyen des grandes commandes suivantes:

- Création du dépôt.
- Importation d'un projet existant.
- Récupération d'un projet dans un dépôt.
- Travail sur un projet (1): (*édition*, *ajout* de fichiers).
- Accession à une version particulière.
- Création d'une branche.
- Travail sur un projet (2): (*suppression*, *renommage* de fichier).

Quelques outils de gestion de version

- Concurrent Versions System (**CVS**): fondée en 1986 par Dick Grune et Brian Berliner
- Subversion (**SVN**): fondée en 2000 par l'entreprise CollabNet et distribué sous licence Apache
- **GIT**: créé en 2005 par Linus Torvalds (créateur de Linux)
- **Hg**: créé en 2005 par Mackall (19 avril trois jours après GIT)

Outline

Création de Dépôt

```
svnadmin create --fs-type fsfs /home/user/svn
```

Il est vivement conseiller de définir une variable qui pointe sur le lieu du dépôt

```
export SVNROOT=file:///home/user/svn
```

Observation de la structure du dépôt

```
svn ls ${SVNROOT}
```

Observation des différentes modifications du dépôt

```
svn log ${SVNROOT}
```

Importation d'un projet existant dans une base

```
cd /path/to/project

//"Supprimer tous les fichiers binaire ou executables et importer"
svn import /path/to/project/ ${SVNROOT}/project/trunk -m 'Initial import'
```

Récupérer un projet. On se Déplace dans le projet pour tout effacer avant la récupération d'une version

```
cd /path/to/project
//"On monte d'un cran"
cd ..
//"On efface le repertoire importer dans le depot"
rm -rf project
//"On recupere une fichier la version actuel du projet dans le depot"
svn checkout ${SVNROOT}
```

On peut lister le répertoire crée par la commande précédente

```
//"On peut lister le repertoire cree"
cd project
ls -a
svn info
```

Travail avec un projet partie 1 : ajout de fichier

```
cd /path/to/project/trunk/
//"Suppose qu'on cree un fichier"
touch MaClass.hpp et MaClass.cpp
//"On observe le contenu du reportoire avec"
svn status
cd /path/to/project/trunk/
svn add MaClass.hpp MaClass.cpp
//"Ajouter un commentaire a l'operation realisee"
svn commit -m "Ajout de MaClass et .... "
```

Creation d'une branche (ici avant-gui)

```
//"#Creation de branche dite avant-gui"
cd /path/to/project
//"#Les branche sont generalement placee dans un repertoire tags"
svn mkdir tags
//"#trunk etant le repertoire de travail"
//"#On le copie comme la version avant modification"
svn copy trunk/ tags/avant-gui
//"#Commit "
svn commit -m "Tagged version before switching to gui" tags/
//"#Recuperation d'une branche"
cd /tmp/
svn checkout file:///home/user/svn/project/tags/avant-gui
```

```
//"On peut lister le repertoire cree"
cd project
ls -a
svn info
```

Travail avec un projet partie 2 : de fichier

```
//"#Supposons vouloir supprimer le Makefile pour passer au  "
cd /path/to/project/trunk/src
svn rm Makefile
svn add src.pro
//"# rempacer doc par html"
cd ..
svn rename html
//"#transmttre les mofications"
svn commit -m 'Switched to qmake. Renamed doc -> html'
```

D'autres commandes

- **revert** annule toutes les modifications dans le répertoire courant.
- **update** met à jour les fichiers du répertoire relativement aux modifications soumise.
- **diff** affiche les différences entre les fichiers du répertoire courant et ceux de la dernière version transmis au dépôt.

Acceder à la doc de ces commande avec
svn help nom_de_commande

Travailler sur une machine distante

Il suffit de remplacer
SVNROOT=file:///home/svn/ par
SVNROOT = svn+ssh://url.of.desktop/home/user/svn

Outline

Gestion avec CVS

- CVS : Current Version System. Fondée en 1986, par Dick Grune et Brian Berliner.
- `http://www.nongnu.org/cvs/`

Exercice

Retrouver l'équivalent les commandes ci-dessus, pour l'outil CVS

Outline

Gestion avec GIT

- C'est un outil de gestion de version efficace et facile d'utilisation
- Page web:http://git-scm.com/
- Prise en main rapide : http://gitref.org/creating/
- Consulter la FAQ https://git.wiki.kernel.org/index.php/GitFaq

Exercice

Retrouver l'équivalent des commandes de base ci-dessus, pour l'outil GIT.

Outline

Gestion avec Hg

- C'est un outil de gestion de version efficace et facile d'utilisation
- Page web:http://mercurial.selenic.com/
- Prise en main rapide : http://mercurial.selenic.com/quickstart/
- Consulter la page Wiki http://mercurial.selenic.com/wiki/

Exercice

Retrouver l'équivalent des commandes de base ci-dessus, pour l'outil Hg.

Outline

Les grandes classes d'approches

- Une approche manuelle utilisant par exemple terminal (Autotools, Qmake, Cmake, Scons, Bjam)
- Une approche basée sur des environnements intégrés (Kdevelop, Qtcreator, Eclipse, Anjuta, NetBeans)

Outline

Générer un projet avec les autotools

Si l'on dispose de l'outil GNU appelé autoproject

Si l'on dispose de autoproject, l'utiliser en ligne de commande et répondre aux questions posées afin de générer un squelette de projet.

Si l'on ne dispose pas de autoproject

- Exécuter autoscan et renommer configure.scan en configure.ac et y ajouter après AC_INIT la ligne AM_INIT_AUTOMAKE.
- Exécuter autoheader pour générer config.h.in
- Rendre le code portable en modifiant si possible config.h.in
- Créer le fichier Makefile.am et le remplir à votre guise.
- Exécuter automake -add-missing
- Exécuter aclocal
- Exécuter autoconf
- Exécuter configure puis make

Applications

Voir en séance de TP

Génération de librairie dynamique avec Autotools

Les fichiers à modifier: **configure.ac, Makefile.am** *(on fournit aussi une interface pour pkg-config)*

configure.ac

- Ajouter, AC_CONFIG_MACRO_DIR([m4]) avant AM_INIT_AUTOMAKE
- Remplacer AC_RANDLIB par AC_PROG_LIBTOOL suivi de AC_PROG_INSTALL
- Ajouter dans le champ AC_CONFIG_FILES le fichier pour pkg-config (ex. mirabelle_dg.pc)

Makefile.am

Ajouter les lignes:

pkgconfigdir = $(libdir)/pkgconfig	⟶	si l'on souhaite utiliser pkgconfig
pkgconfig_DATA = mirabelle_dg.pc	⟶	si l'on souhaite utiliser pkgconfig
pkgconfig_DATA: config.status	⟶	si l'on souhaite utiliser pkgconfig
EXTRA_DIST = mirabelle_dg.pc.in	⟶	si l'on souhaite utiliser pkgconfig
DISTCLEANFILES = mirabelle_dg.pc	⟶	si l'on souhaite utiliser pkgconfig
ACLOCAL_AMFLAGS = -I m4		

Remplacer dans les Makefile.am de chaque sous-répertoire l'instruction
lib_LIBRARIES = libxxx.a par
lib_LTLIBRARIES = libxxx.la et
libxxx_a_SOURCES par

libxxx_la_SOURCES

Exemple de fichier pour pkg-config

mirabelle_dg.pc.in

```
prefix=@prefix@
exec_prefix=@exec_prefix@
libdir=@libdir@
includedir=@includedir@

Name: Mirabelle_dg
Description: C++ library for discontinuous Galerkin method
Version: @VERSION@
#Requires: umfpack
Libs: -L${libdir} -lgeometry -lgraphic -lalgebra -lutils -lfem -lcommon -lnewformalism -↩
      lclassicformalism
Cflags: -I${includedir}
```

Compilation automatique avec **boost.build**

Boost http//:www.boost.org est une collection de librairies utiles pour le développement d'applications en C++.
Boost fournit aussi un outil de compilation automatique: **boost.build** Pour compiler du code présent dans le fichier *essai.cpp*, il faut :

1. Créer un fichier JamRoot

fichier: Jamroot.jam

```
exe essai : essai.cpp : <variant>release ;
```

2. Exécuter bjam ou b2 sur ce fichier

```
chemin/vers-bjam/b2 -sBOOST_ROOT=ccd /home/apoung/MC-2010-2011/teaching/MC-2011-2012/↩
      genielogiciel/semaine2/cours
```

Outline

Importer un projet sous eclipse-CDT

Changer le type de perspective

Cliquer sur **window -> Open perspective -> Others** et choisir C/C++ et valider

Charger un projet (il est supposer se reposer sur un Makefile)

- **File -> New -> C++ Project**
- Donner un nom de projet et dans **Project type**: sélectionner **Empty Project**
- Pour **Location** cliquer sur **Browse** et se reporter au répertoire où se trouve le code.

Ajouter l'exécutable

- Cliquer sur **Run**
- Sélectionner **Run Configuration**
- Choisir **C/C++ Application** puis New.
- Remplacer **Default** par **Use Active**. Utiliser **Browse** pour sélectionner le projet
- Remplir **C/C++ Application** en utilisant **Browse** pour sélectionner l' exécutable. Puis fermer la fenêtre.

Importation d'un projet SVN a partir de Anjuta

Il faut au préalable activer SVN à l'aide
Edition -> Préférences->Général->Greffons installés
puis cocher la case subversion.

Portage de projet depuis le dépôt

Il faut suivre les étapes suivantes:

- Lancer anjuta
- Sélectionner **Nouveau -> Fichier basé sur les sources existants**
- Donner le nom du fichier et cocher l'option **Importer à partir d'un système de gestion de versions**
- Dans le champ **Emplacement**, indiquer le chemin vers le projet dans la base (par exemple *file:///path/to/svn/projet*)
  puis choisir dans le menu déroulant, l'option **subversion**
- Cliquer sur l'onglet **Importer** qui est maintenant actif.

Utilisation (accéder au journal de fichiers)

Vous pouvez cliquer sur **Journal de Subversion** pour observer le journal des fichiers du dossier.

Importation d'un projet SVN a partir de qtcreator

Portage de projet depuis le dépôt

Il faut suivre les étapes suivantes:

- Lancer qtcreator
- Sélectionner **Fichier->Nouveau fichier ou projet**
- Choisir l'option **Gestion de versions -> Checkout Subversion**
- Dans le champ **Dépôt**, indiquer le chemin vers le projet dans la base (par exemple *file:///path/to/svn/projet*)
  puis cliquer sur **Parcourir** pour rentrer le lieu où mettre le dossier projet.
- Cliquer sur **Suivant**. La récupération est donc effectuée.
- Cliquer sur **Terminer**.
- Si le projet n'est un projet **qmake** (contient un fichier ***.pro**) ou **cmake** (contient **CMakeLists.txt**) il y'aura un échec.

Importation d'un projet SVN a partir de eclipse

Autoriser des commentaires Doxygen dans Eclipse

Configuration

- Sélectionner le projet et cliquer sur le bouton droit de la souris.
- Cliquer sur **Properties**
- Choisir **C/C++ General** et activer doxygen

Utilisation

- Se placer au dessus d'une entité à documenter
- Saisir le début de commentaire : /** et valider, eclipse fera le reste.

Il faut noter que la documentation de classes n'est pas efficace.

Intégration automatique de Doxygen à Qtcreator

Le script suivant récupère qtcreator sur la toile, le compile et l'installe; puis télécharge un *plugging* doxygen (qtcreator-doxygen) le compile et l'installe et le rend accessible à qtcreator.

```
#!/bin/sh #"written by JB APOUNG jean-baptiste.apoung@math.u-psud.fr"
# "Mettre ici la function regenerate_doxygen_pro() vois ci dessous"
#######################################################
## "qtcreator part"
#######################################################
echo " Enter the working directory :" ;
rm -rf qtcreator_1.3.1.orig.tar.gz* ;
echo " fetching for the qtcreator cource" ;
wget http://ftp.de.debian.org/debian/pool/main/q/qtcreator/qtcreator_1.3.1.orig.tar.gz . ;
echo " unpacking qt_creator version 1.3.1";
tar -zxvf qtcreator_1.3.1.orig.tar.gz;
echo "creating the build directory" ;
mkdir qtcreator-1.3.1_build
cd qtcreator-1.3.1_build;
echo "Compiling " ;
qmake ../qtcreator-1.3.1/qtcreator.pro ;
make -j8 ;
#######################################################
## "qtcreator-doxygen  part"
#######################################################
echo " feching for the the qtcreator-doxygen " ;
svn co http://svn.kofee.org/svn/qtcreator-doxygen/trunk@20 qtcreator-doxygen ;
cd qtcreator-doxygen;
echo "changing the required version 1.3.0 to 1.3.1";
sed -e 's/1.3.0/1.3.1/' \
    < Doxygen.pluginspec > Doxygen.pluginspec2 ;
/bin/mv Doxygen.pluginspec2 Doxygen.pluginspec ;
echo "performing appropriate replacement in the doxygen.pro";
regenerate_doxygen_pro; #"this is a call to function"
echo "generatint the Makefile";
qmake -makefile;
echo "running the make script";
make -j8;
```

Intégration automatique de Doxygen à Qtcreator

La fonction regenerate_doxygen_pro appellée ci-dessus (1/2)

```
regenerate_doxygen_pro(){
echo "TEMPLATE = lib " > doxygen.pro ;
echo "TARGET = Doxygen" >> doxygen.pro ;
echo "DEFINES += DOXYGEN_LIBRARY" >> doxygen.pro ;
echo "PROVIDER = Kofee">> doxygen.pro ;
# Define QTC_SOURCE_DIR to the location of Qt Creator
#sources (i.e: ~/dev/qtcreator/qt-creator-src/)
echo "unix:QTC_SOURCE_DIR =../../qtcreator-1.3.1/ " >> doxygen.pro ;
#/home/kofee/dev/qtcreator/qt-creator-1.3.0/
echo "win32:QTC_SOURCE_DIR = C:/Qt/dev/qt-creator-1.3.0/">> doxygen.pro ;
echo "IDE_SOURCE_TREE =\$\$QTC_SOURCE_DIR" >> doxygen.pro ;
# Define QTC_BUILD_DIR to the location of Qt Creator
# build dir for the plugin (i.e ~/dev/qtcreator/doxygen/)
echo "unix:QTC_BUILD_DIR =../.">> doxygen.pro ;
#/home/kofee/dev/qtcreator/qtcreator-doxygen/
echo "win32:QTC_BUILD_DIR = C:/Qt/dev/qtcreator-doxygen/">> doxygen.pro ;
echo "IDE_BUILD_TREE = \$\$QTC_BUILD_DIR">> doxygen.pro ;
echo "DESTDIR =../lib/qtcreator/plugins/\$\$(PROVIDER)">> doxygen.pro ;
echo "unix:LIBS += -L../lib/qtcreator \ ">> doxygen.pro ;
echo "-L../lib/qtcreator/plugins/Nokia \ ">> doxygen.pro ;
echo "-L/usr/lib/qt4" >> doxygen.pro ;
echo "win32:LIBS += -LC:/Qt/4.6.0/lib \ ">> doxygen.pro ;
echo "-LC:/Qt/dev/qt-creator-1.3.0/lib/qtcreator/plugins/Nokia \ ">> doxygen.pro ;
echo "-LC:/Qt/dev/qt-creator-1.3.0/lib/qtcreator ">> doxygen.pro ;
echo "include( \$\$IDE_SOURCE_TREE/src/qtcreatorplugin.pri ) " >> doxygen.pro ;
echo "include( \$\$IDE_SOURCE_TREE/src/plugins/coreplugin/coreplugin.pri ) ">> doxygen.pro ;
echo "include( \$\$IDE_SOURCE_TREE/src/plugins/texteditor/texteditor.pri ) ">> doxygen.pro ;
echo "include( \$\$IDE_SOURCE_TREE/src/plugins/cppeditor/cppeditor.pri ) ">> doxygen.pro ;
echo "HEADERS += doxygenplugin.h \ ">> doxygen.pro ;
echo "doxygen_global.h \ ">> doxygen.pro ;
echo "doxygenconstants.h \ ">> doxygen.pro ;
echo "doxygen.h \ ">> doxygen.pro ;
echo "doxygensettings.h \ ">> doxygen.pro ;
```

Intégration automatique de Doxygen à Qtcreator

La fonction regenerate_doxygen_pro suite (2/2)

```
echo "doxygensettingswidget.h \ ">> doxygen.pro ;
echo "doxygensettingsstruct.h ">> doxygen.pro ;
echo "SOURCES += doxygenplugin.cpp \ ">> doxygen.pro ;
echo "doxygen.cpp \ ">> doxygen.pro ;
echo "doxygensettings.cpp \ ">> doxygen.pro ;
echo "doxygensettingswidget.cpp \ ">> doxygen.pro ;
echo "doxygensettingsstruct.cpp ">> doxygen.pro ;
echo "FORMS += doxygensettingswidget.ui ">> doxygen.pro ;
echo "OTHER_FILES += Doxygen.pluginspec ">> doxygen.pro ;
echo "INCLUDEPATH += \$\$QTC_SOURCE_DIR/src \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/plugins \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/libs \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/libs/cplusplus \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/libs/extensionsystem \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/libs/utils \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/shared \ ">> doxygen.pro ;
echo "\$\$QTC_SOURCE_DIR/src/shared/cplusplus ">> doxygen.pro ;
echo "message(QTC_SOURCE_DIR =\$\$QTC_SOURCE_DIR) ">> doxygen.pro ;
echo "message(IDE_SOURCE_TREE=\$\$IDE_SOURCE_TREE) ">> doxygen.pro ;
echo "message(QTC_BUILD_DIR =\$\$QTC_BUILD_DIR) ">> doxygen.pro ;
echo "message(IDE_BUILD_TREE =\$\$IDE_BUILD_TREE) ">> doxygen.pro ;
echo "message(DESTDIR =\$\$DESTDIR) ">> doxygen.pro ;
echo "message(Good luck with make... :-D) ">> doxygen.pro ;
}
```

- Une fois installé lancer qtcreator et vérifier que le pluging doxygen est chargé:
  Aide -> A propos de plugings
- Ouvrer un fichier, se placer après une déclaration de fonction et exécuter `Ctrl-Maj-F3`.