

TP - Génie Logiciel (1)

Construction et gestion de sources

Romain PELISSE

ESME Sudria

1^{er} août 2008

1 Introduction

- Gérer un projet logiciel code
- Les outils du Génie Logiciel

2 Outils de construction

- Quel outil prendre et dans quel but ?
- Ant
- Pour aller plus loin...
Maven

3 Gestionnaire de sources

- Les fonctionnalités des SCM
- Lexique

4 Livraison logicielle

- Les produits et l'avenir
- TD :Utilisation de CVS avec Eclipse

5 Fin

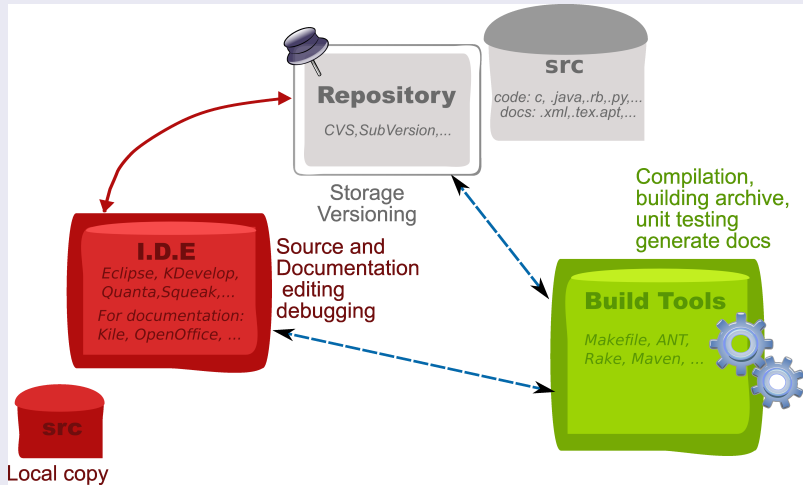
Autour d'un projet de développement...

- La procédure de **compilation** et d'*éditions de liens*
- Gérer les *dépendances* du projets.
- Disposer d'une **documentation** synchronisée avec la version
- Le **packaging** :
 - format exécutable (elf, exe, .app)
 - embarquer (ou non) les dépendances
 - version
- ...

Objectifs

- Pouvoir **reconstruire** le projet facilement
- **Abstraire l'environnement** de développement (OS, IDE,...)
- Reproduire de manière exacte **une version**, conserver toutes les versions
- Automatisation de la **non-régression**
- Effectuer des tâches redondantes **contrôle qualité, documentation,...**
- **Assurer une livraison** de bonne qualité

Tout ceci requiert des outils !



Objectif

- Automatiser les tâches redondantes du projet :
 - Exécution des tests unitaires
 - Génération de la documentation
 - Produire les artefacts
 - ...
- Permettre de facilement 'construire' le projet (compilation, packaging,...)
- Externaliser les paramètres (adresse IPs, paramètres par défaut,...)
- Automatiser les déploiements (serveur de pré-prod, prod)

Le nerf du projet : le build

Le **build** est donc un point essentiel du projet car il est le seul garant de sa **maintenabilité**. *Un projet sans build peut être impossible à reprendre, à maintenir ou à faire évoluer.*

La relation technologie/build

| Techno | Build |
|---------|------------------------|
| OpenVMS | MMS |
| C/C++ | CMake, Makefile |
| Java | Ant , Maven |
| C# | NAnt |
| Ruby | Raven |

Historique du projet Ant

- Outil de Build en Java, alternative à Makefile :
 - Trop complexe
 - Peu adapté à Java
- Fichier de description de tâche en XML
- Conçu en Java, donc portable sur tout les OS
- Appel : `ant + série de tâches (ex : ant clean compile install)`

Structure du fichier

- Tâche
- Interdépendance

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="tpbuild" default="init" basedir=".">

    <!-- La tache 'init' depend de 'clean' -->
    <target name="init" depends="clean">
<!-- Si 'clean' n'a pas ete appelee au moment ou
en Ant execute 'init', Ant appelle automatiquement
'clean'. -->
    </target>

    <target name="clean">
    </target>
</project>
```

Ligne de commande

- ① Télécharger Ant et dézipper Ant
- ② Définir la variable ANT_HOME qui contiendra l'adresse du répertoire Ant
- ③ Ajouter dans le PATH
 - Windows : %ANT_HOME% \bin
 - Unix et Linux : \${ANT_HOME}/bin : {PATH}
- ④ La commande ant sera désormais reconnue.

Au sein d'Eclipse

Etudiez après en TD...

Définir des propriétés

- Variables remplacées à l'exécution du script par leur valeur
- Externalisation de paramètres (numéro de version, adresse ip,...)
- Définir une valeur à un seul endroit

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="tpbuild"
         default="echo-properties">

  <property name="version"          value="1.0"/>
  <property name="src"              value="src"/>
  <property name="build"            value="build"/>

  <target name="echo-properties">
    <echo>Project version:${version}</echo>
    <echo>Source directory:${src}</echo>
```

Ant task

Il existe de nombreuses tâches associées aux actions :

- **echo**, pour afficher du texte
- **mkdir**, pour créer des répertoires
- **delete**, pour effacer des fichiers ou des répertoires
- **javac**, pour compiler du code java
- **javadoc**, pour générer la documentation à partir des commentaires javadoc
- **jar**, pour fabriquer un jar à partir de classes
- ...

Liste complète : <http://ant.apache.org/manual/coretasklist.html>

Mise en place

- Récupérer le projet 'tpbuild'
- Dézippez le, et placez le répertoire 'tpbuild' dans le workspace d'Eclipse
- Avec Eclipse, faite *Fichier > Nouveau Projet- > Java Project*
- Appeler le projet comme le répertoire présent dans le workspace, soit 'tpbuild'
- Eclipse reconnaît la présence d'un projet existant. Valider.

Utilisation de Ant

- Dans le menu *Fenêtres*, sélectionner l'option *Vue*, et choisissez la vue *Ant*
- Dans la fenêtre qui apparaît, sélectionnez le premier bouton *Add build files*
- Sélectionner le fichier *build.xml* déjà présent dans le répertoire 'tpbuild'

Conception du build

Ouvrez le fichier **build.xml** avec l'éditeur Ant de Eclipse et commencer par le **TODO FIRST**.

Maven 2

- Gestion automatique des dépendances
- Intégration avec le SCM
- Architecture évolutif à base de plugin

Pour les projets de dernière année en Java, à considérer très sérieusement ! [http ://maven.apache.org/](http://maven.apache.org/)

SCM

Source Configuration Management :

- Centraliser les sources
- Versionner les fichiers
- Travail concurrent, 2 philosophies
 - lock
 - merge
- Tagger une version

Lexique

- checkout - récupérer une copie local du projet
- commit - appliquer ses modifications sur le projet
- diff - Etudier les différences entre 2 versions
- update - mettre à jour son projet local
- patch - proposer une modification, sans l'appliquer
- override & commit - comitter de 'force' (attention !)
- override & update - remplacer les copies locales modifié par celle du scm

Lexique

- La livraison !

Etat de l'existant

- Solutions libres comme CVS de plus en plus remplacé par SVN
- Solutions propriétaires multiples (ClearCase,SourceSafe)

Pour aller plus loin

- Les Forges
- L'intégration continue

Enoncé

Avec le plugin CVS de Eclipse ou TortoiseCVS, réalisez les actions suivantes :

- ① **Import** du projet fourni dans votre espace projet
 - Attention à ne pas importer des fichiers binaires ou des artefacts !
- ② Faire un **checkout** du projet, modifier un source puis utiliser *Synchronize with Repository*
- ③ Réaliser un patch avec **Create Patch** et étudier le fichier généré
- ④ **commiter** le source modifié
- ⑤ Faites un **update** du projet pour prendre en compte mes modifications

Latex

Ce slideware a été réalisé à l'aide du package **beamer** pour

The logo for LATEX, featuring the word "LATEX" in a serif font, with the "T" and "E" having a small "X" and "A" respectively, all enclosed in a thin rectangular border.

Licence associée

Cette présentation et son contenu est placé sous licence Creative Commons, vous pouvez réutiliser cette dernière en respectant les clause suivantes :

- **Paternité** : Citer le nom de l'auteur original.
- **Pas d'Utilisation Commerciale** : Vous n'avez pas le droit d'utiliser cette présentation à des fins commerciales.