The APACHE ANT Project

"Another Neat Tool" (un autre chouette outil)

ant.apache.org

Nicolas Hernandez

IUT de Nantes – Département Informatique
LINA - Laboratoire d'Informatique de Nantes Atlantique
Cours de Licence Professionnelle
2009 – 20..

Nantes, le 8 septembre 2014

Sommaire

1. Motivations, principe de l'outil `ant`, installation, exécution
2. Structure du fichier de configuration `build.xml`
3. Mises en oeuvres des tâches classiques investies dans des cibles habituelles (compilation, génération de documentation, test et versionning)
4. Les catégories de tâches
5. Définir ses propres tâches
6. `ant` dans Eclipse
7. Gestion des dépendances à l'aide de Apache Ivy `http://ant.apache.org/ivy`

Introduction – Sommaire

Makefile

Fichier nommé...

Makefile

```
# top-level rule to create the program
all : main

# compiling the source code
main.o : main.c
	gcc -g -Wall -c main.c

# linking the compiled files
main : main.o
	gcc -g main.o -o main

# remove generated files
clean :
	/bin/rm -f main main.o
```

Appelé avec `make [cible]`

Adapté pour les programmes `C` et `C++` mais les autres comme `Java` ? Portabilité (IDE, OS) ? La manière de fédérer différents outils en un seul processus cohérent ?

Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (compilation, test, génération de documentation, déploiement, versionning, ...) d'une application
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (compilation, test, génération de documentation, déploiement, versionning, ...) d'une application
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (compilation, test, génération de documentation, déploiement, versionning, ...) d'une application
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (compilation, test, génération de documentation, déploiement, versionning, ...) d'une application
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (compilation, test, génération de documentation, déploiement, versionning, ...) d'une application
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

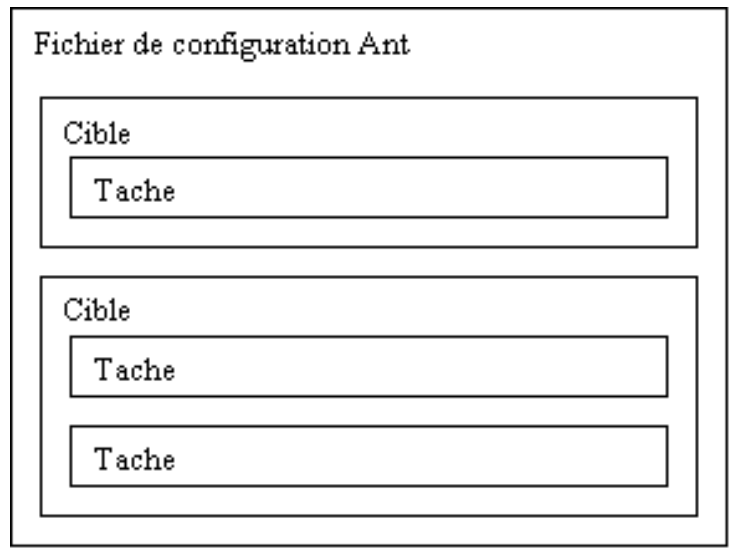
Motivations

"L'objectif du projet `ant.apache.org` *est de fournir un outil pour permettre "*

- la **construction d'applications**
- l'**automatisation les opérations répétitives** du cycle du développement (compilation, test, génération de documentation, déploiement, versionning, ...) d'une application
- l'**indépendance envers toute plate-forme** (écrit en Java)
- la **configuration** à l'aide d'un fichier de XML qui décrit les tâches à exécuter
- l'**extension** en permettant l'écriture de nouvelles tâches

Principes et concepts

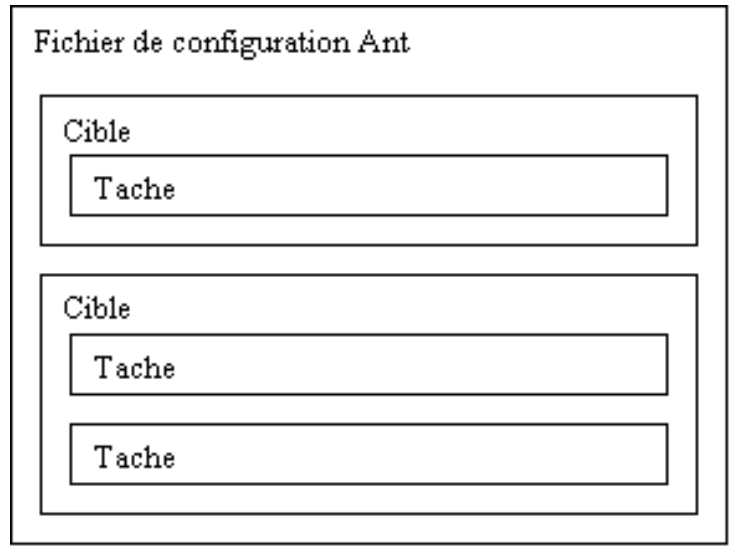
- la commande `ant` repose sur un fichier de configuration `build.xml`
- le `build.xml` contient un ensemble de cibles (`target`), qui constituent les étapes du projet de construction
- chaque cible contient une ou plusieurs tâches (`task`) ordonnées, qui constituent des traitements unitaires à réaliser
- chaque cible peut être dépendante (`depends`) de l'exécution d'une ou plusieurs autres cibles



Principes et concepts

`build.xml`

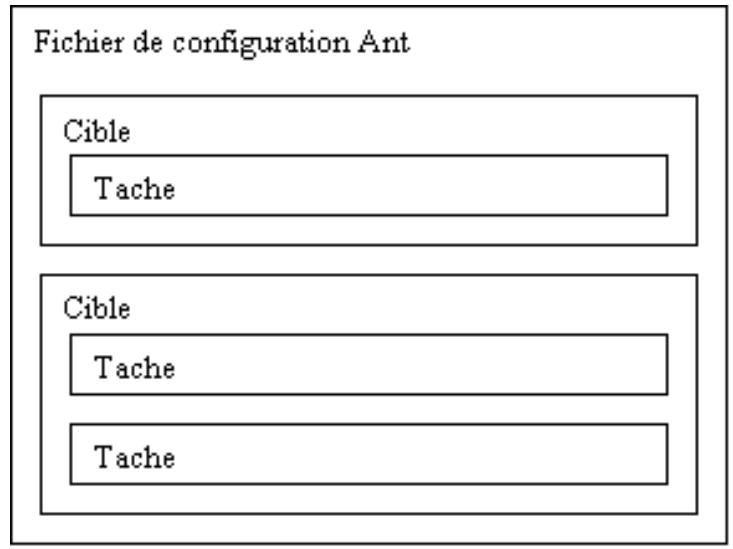
- la commande `ant` repose sur un fichier de configuration `build.xml`
- le `build.xml` contient un ensemble de cibles (`target`), qui constituent les étapes du projet de construction
- chaque cible contient une ou plusieurs tâches (`task`) ordonnées, qui constituent des traitements unitaires à réaliser
- chaque cible peut être dépendante (`depends`) de l'exécution d'une ou plusieurs autres cibles



Principes et concepts

- la commande `ant` repose sur un fichier de configuration `build.xml`
- le `build.xml` contient un ensemble de cibles (`target`), qui constituent les étapes du projet de construction
- chaque cible contient une ou plusieurs tâches (`task`) ordonnées, qui constituent des traitements unitaires à réaliser
- chaque cible peut être dépendante (`depends`) de l'exécution d'une ou plusieurs autres cibles

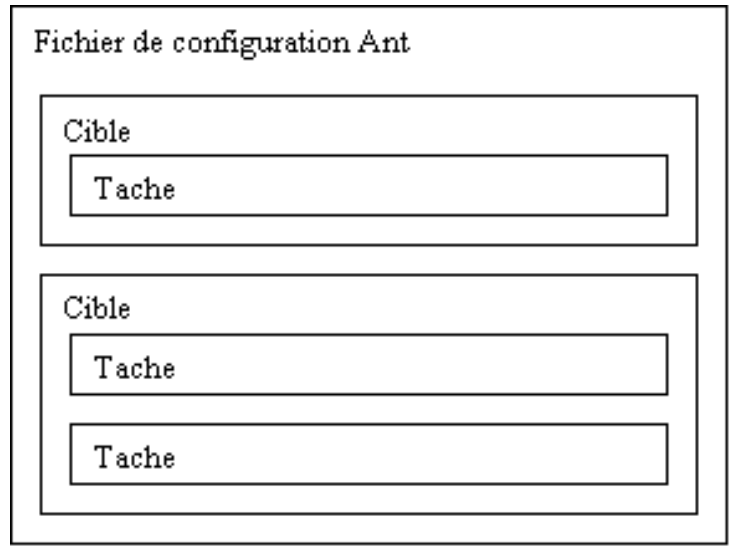
`build.xml`



Principes et concepts

- la commande `ant` repose sur un fichier de configuration `build.xml`
- le `build.xml` contient un ensemble de cibles (`target`), qui constituent les étapes du projet de construction
- chaque cible contient une ou plusieurs tâches (`task`) ordonnées, qui constituent des traitements unitaires à réaliser
- chaque cible peut être dépendante (`depends`) de l'exécution d'une ou plusieurs autres cibles

`build.xml`



Un exemple simple de build file

build.xml

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

  <target name="init">
    <mkdir dir="build/classes"/>
    <mkdir dir="dist"/>
  </target>

  <target name="compile" depends="init" description="Compile Source Code">
    <javac srcdir="src" destdir="build/classes"/>
  </target>

  <target name="package" depends="compile" description="Generate jar file">
    <jar destfile="dist/monProjet.jar" basedir="build/classes"/>
  </target>

  <target name="clean" description="Delete generated directories">
    <delete dir="build"/>
    <delete dir="dist"/>
  </target>
</project>
```

`javac -nowanr -d build/classes src/main`
`ant -projecthelp` pour connaître les cibles et leur description

Un exemple simple de build file

`Ant` n'impose aucune arborescence particulière. Il est nécessaire de définir la sienne à chaque projet.
flexibilité mais coût de prise en main.
Néanmoins il y a certaines pratiques qui sont plus communes que d'autres...

src	code source de l'application
test	code des tests unitaires
lib	dépendances (bibliothèques requises) du projets
build	tout fichier généré par le processus de construction
build/classes	les classes sources compilées
build/test-classes	les tests unitaires compilés
dist	fichier de distribution tel que `jar` ou `war`

Core, optional and third-party tasks

core task construit dans `Ant`, ne requiert pas de configuration spéciale

optional task maintenu par l'équipe `Ant` et livré avec `Ant`, mais nécessitant des bibliothèques externes
Exemple : `xalan.jar` (XSL transformer), `junit.jar`, `mail.jar`, `Groovy jars` (scripts Java), `jdepend.jar` ...

third party task écrit et maintenu hors du projet `Ant`
Exemple : `Subversion` fourni par Tigris (l'équipe qui maintient Subversion)

`Ant 1.8` vient avec environ 100 core tasks et 50 optionnelles
`Ant 1.9` Java 1.5 is now required...

Installation

Download

`ant.apache.org`

Requirement

un JDK plutôt qu'un JRE (sans quoi des tâches indisponibles)

Setup (Linux/Unix bash)

```
export ANT_HOME=/mon/local/ant
export JAVA_HOME=/usr/java/jdk1.6.0_20
export PATH=${PATH}:${ANT_HOME}/bin
```

Si des `optional task` *sont requises, il faut spécifier dans le classpath les bibliothèques externes qui les implémentent*

- Pour cela utiliser l'élément `<classpath>` de la tâche si disponible
- ne pas modifier la variable d'environnement `CLASSPATH` car rend jar accessible à toutes applications

Pour information, le répertoire `ANT_HOME/lib` de la version binaire téléchargeable de *ant* contient la plupart de ces dépendances,
`http://ant.apache.org/manual/install.html#installing` indique comment les installer

Exécution

Utilisation en ligne de commande selon la syntaxe

```
ant [options] [cible]
```

Comportement par défaut

- recherche un fichier `build.xml` dans le répertoire courant
- si aucune cible n'est spécifiée, il prendra celle déclarée par défault dans le fichier

```
ant
```

Spécification d'un fichier de configuration

```
ant -buildfile monbuild.xml
```

Exécution de la cible `clean` et toutes les cibles dont elle dépend

```
ant clean
```

ant interfacé dans de nombreux IDEs

Existence de plugins pour Eclipse, NetBeans, IntelliJ IDEA...

Exécution

Utilisation en ligne de commande selon la syntaxe

```
ant [options] [cible]
```

Comportement par défaut

- recherche un fichier `build.xml` dans le répertoire courant
- si aucune cible n'est spécifiée, il prendra celle déclarée par défault dans le fichier

```
ant
```

Spécification d'un fichier de configuration

```
ant -buildfile monbuild.xml
```

Exécution de la cible `clean` et toutes les cibles dont elle dépend

```
ant clean
```

ant interfacé dans de nombreux IDEs

Existence de plugins pour Eclipse, NetBeans, IntelliJ IDEA...

Exécution

Utilisation en ligne de commande selon la syntaxe

```
ant [options] [cible]
```

Comportement par défaut

- recherche un fichier `build.xml` dans le répertoire courant
- si aucune cible n'est spécifiée, il prendra celle déclarée par défault dans le fichier

```
ant
```

Spécification d'un fichier de configuration

```
ant -buildfile monbuild.xml
```

Exécution de la cible `clean` et toutes les cibles dont elle dépend

```
ant clean
```

ant interfacé dans de nombreux IDEs

Existence de plugins pour Eclipse, NetBeans, IntelliJ IDEA...

Exécution

Utilisation en ligne de commande selon la syntaxe

```
ant [options] [cible]
```

Comportement par défaut

- recherche un fichier `build.xml` dans le répertoire courant
- si aucune cible n'est spécifiée, il prendra celle déclarée par défault dans le fichier

```
ant
```

Spécification d'un fichier de configuration

```
ant -buildfile monbuild.xml
```

Exécution de la cible `clean` et toutes les cibles dont elle dépend

```
ant clean
```

ant interfacé dans de nombreux IDEs

Existence de plugins pour Eclipse, NetBeans, IntelliJ IDEA...

Exécution

Utilisation en ligne de commande selon la syntaxe

```
ant [options] [cible]
```

Comportement par défaut

- recherche un fichier `build.xml` dans le répertoire courant
- si aucune cible n'est spécifiée, il prendra celle déclarée par défault dans le fichier

```
ant
```

Spécification d'un fichier de configuration

```
ant -buildfile monbuild.xml
```

Exécution de la cible `clean` et toutes les cibles dont elle dépend

```
ant clean
```

ant interfacé dans de nombreux IDEs

Existence de plugins pour Eclipse, NetBeans, IntelliJ IDEA...

Le corps du fichier build.xml

Le "corps" et les définitions des

1. **propriétés** (properties) : variables qui contiennent des valeurs utilisables par des cibles ou tâches
2. **patrons décrivant des ensembles** de fichiers ou de répertoires (patternset), et **listes explicites de fichiers** (filelist) utiles pour définir des path
3. **path** chemins vers des fichiers/répertoires utiles pour la définition globale de classpath
4. **cibles** (targets), étapes du projet de construction qui mettent en oeuvre des **tâches**, traitements unitaires

(habituellement présentées dans cet ordre)

Le corps du fichier build.xml

Le "corps" et les définitions des

1. **propriétés** (properties) : variables qui contiennent des valeurs utilisables par des cibles ou tâches
2. **patrons décrivant des ensembles** de fichiers ou de répertoires (patternset), et **listes explicites de fichiers** (filelist) utiles pour définir des path
3. **path** chemins vers des fichiers/répertoires utiles pour la définition globale de classpath
4. **cibles** (targets), étapes du projet de construction qui mettent en oeuvre des **tâches**, traitements unitaires

(habituellement présentées dans cet ordre)

L'élément racine `project` et les commentaires

Le fichier `build.xml` contient la description du processus de construction de l'application

Le prologue

`<?xml version="1.0" encoding="UTF-8">` (ou bien `"ISO-8859-1"`)

`project`, *élément racine du document et ses attributs*

- `name` : nom du projet
- `default` : cible par défaut à exécuter si aucune cible précisée
- `basedir` : répertoire de référence pour la localisation relative des autres répertoires

```
<project name="mon projet" default="compile" basedir=".">
```

Les commentaires

```
<!-- Ceci est un exemple de commentaire -->
```

L'élément racine project et les commentaires

Le fichier build.xml contient la description du processus de construction de l'application

Le prologue

<?xml version="1.0" encoding="UTF-8"> (ou bien "ISO-8859-1")

project, élément racine du document et ses attributs

- name : nom du projet
- default : cible par défaut à exécuter si aucune cible précisée
- basedir : répertoire de référence pour la localisation relative des autres répertoires

```
<project name="mon projet" default="compile" basedir=".">
```

Les commentaires

```
<!-- Ceci est un exemple de commentaire -->
```

L'élément racine `project` et les commentaires

Le fichier `build.xml` contient la description du processus de construction de l'application

Le prologue

`<?xml version="1.0" encoding="UTF-8">` (ou bien `"ISO-8859-1"`)

`project`, élément racine du document et ses attributs

- `name` : nom du projet
- `default` : cible par défaut à exécuter si aucune cible précisée
- `basedir` : répertoire de référence pour la localisation relative des autres répertoires

```
<project name="mon projet" default="compile" basedir=".">
```

Les commentaires

```
<!-- Ceci est un exemple de commentaire -->
```

L'élément racine `project` et les commentaires

Le fichier `build.xml` contient la description du processus de construction de l'application

Le prologue

`<?xml version="1.0" encoding="UTF-8">` (ou bien `"ISO-8859-1"`)

`project`, élément racine du document et ses attributs

- `name` : nom du projet
- `default` : cible par défaut à exécuter si aucune cible précisée
- `basedir` : répertoire de référence pour la localisation relative des autres répertoires

```
<project name="mon projet" default="compile" basedir=".">
```

Les commentaires

```
<!-- Ceci est un exemple de commentaire -->
```

Détails du fichier `build.xml` – *Sommaire*

Les propriétés (définition)

Utilité

définir une seule fois une valeur qui est utilisée plusieurs fois dans le projet

Définition des variables

- avec l'option `-D nom=valeur` en ligne de commande
- avec la balise `property` dans le `build.xml` avec au choix
  - soit attribut `file`, fichier contenant liste de lignes `nom=valeur`
  - soit couple d'attributs `name/value`
  - soit couple d'attributs `name/location`, fichier dont le contenu désigne la valeur

Dans le build.xml

```
<property file="mesproprieteslocales.properties" />
<property name="projet.nom" value="mon_projet" />
<property name="projet.version" value="0.0.10" />
<property name="projet.license" location="doc/LICENSE" />
<property name="src.dir" value="src" />
<property name="build.dir" value="build" />
```

Les propriétés (définition)

Utilité

définir une seule fois une valeur qui est utilisée plusieurs fois dans le projet

Définition des variables

- avec l'option `-D nom=valeur` en ligne de commande
- avec la balise `property` dans le `build.xml` avec au choix
  - soit attribut `file`, fichier contenant liste de lignes `nom=valeur`
  - soit couple d'attributs `name/value`
  - soit couple d'attributs `name/location`, fichier dont le contenu désigne la valeur

Dans le build.xml

```
<property file="mesproprieteslocales.properties" />
<property name="projet.nom" value="mon_projet" />
<property name="projet.version" value="0.0.10" />
<property name="projet.license" location="doc/LICENSE" />
<property name="src.dir" value="src" />
<property name="build.dir" value="build" />
```

Les propriétés (utilisation)

Utilisation à l'aide de `${projet.name}`

Ordre de définition des propriétés

seule la première définition d'une propriété compte, les suivantes sont ignorées

Propriétés prédéfinies

- `basedir`, chemin absolu du répertoire de travail ;
- `ant.file`, chemin absolu du fichier build en cours de traitement ;
- `ant.java.version`, version de la JVM qui exécute ant ;
- `ant.project.name`, nom du projet en cours d'utilisation
- `user.home`, le `user home directory` mais sa valeur dépend de la version de l'OS et de l'implémentation de la JVM

`classpath` n'est pas une propriété prédéfinie !

Les propriétés (utilisation)

Utilisation à l'aide de

`${projet.name}`

Ordre de définition des propriétés

seule la première définition d'une propriété compte, les suivantes sont ignorées

Propriétés prédéfinies

- `basedir`, chemin absolu du répertoire de travail ;
- `ant.file`, chemin absolu du fichier build en cours de traitement ;
- `ant.java.version`, version de la JVM qui exécute ant ;
- `ant.project.name`, nom du projet en cours d'utilisation
- `user.home`, le `user home directory` mais sa valeur dépend de la version de l'OS et de l'implémentation de la JVM

`classpath` n'est pas une propriété prédéfinie !

Patrons et listes identifiés de fichiers/répertoires

`filelist`, *liste de fichiers explicitement nommés*

id Identifiant pour l'ensemble qui pourra ainsi être réutilisé

dir Répertoire de départ de l'ensemble de fichiers

files Liste des fichiers séparés par une virgule

refid Demande réutilisation d'une liste ayant l'identifiant fourni comme valeur

```
<filelist id="optional_jars" dir="lib" files="junit.jar,xalan.jar,mail.jar" />
<filelist id="third-party_jars" dir="lib" files="svnant.jar" />
```

`patternset`, *patrons définissant des ensembles de fichiers/répertoires*

id Identifiant pour l'ensemble qui pourra ainsi être réutilisé

includes et excludes liste de patrons de fichiers/répertoires séparés par virgule/espace

refid Demande réutilisation d'un ensemble ayant l'identifiant fourni comme valeur

```
  <patternset id="sources.code">
    <include name="**/*.java"/>
    <exclude name="**/*Test*"/>
<!--    <exclude name="**/*~"/> inutile ?-->
  </patternset>
```

Les ensembles de fichiers/répertoires

`fileset`, *groupes de fichiers et* `dirset`, *groupes de répertoires*

dir Répertoire de départ de l'ensemble de fichiers

includes et excludes Liste des fichiers à inclure/exclure

non identifié ! Utilisés pour définir des path

```
<fileset dir="${server.src}" casesensitive="yes">
  <include name="**/*.java"/>
  <exclude name="**/*Test*"/>
</fileset>

<fileset dir="${client.src}" > (équivalent avec patternset)
  <patternset refid="sources.code"/>
</fileset>

<fileset dir="src" includes="**/*.java"> (forme abrégée)

<dirset dir="${build.dir}">
  <include name="apps/**/classes"/>
  <exclude name="apps/**/*Test*"/>
</dirset>
```

Généralités

Spécification d'un chemin

- Quelle que soit la plate-forme, un chemin utilise la caractère slash '`/`' comme séparateur ;`ant` convertit selon l'OS
- de même, '`:`' et '`;`' peuvent être utilisés indifféremment pour spécifier une liste de chemin
- L'expression `**/` permet de désigner tous les sous répertoires du répertoire défini dans l'attribut dir répertoire

Liste traditionnelle de fichiers à exclure

```
**/*~
**/#*#
**/.#*
**/%*%
**/._*
**/CVS
**/CVS/**
**/.cvsignore
**/SCCS
**/SCCS/**
**/vssver.scc
**/.svn
**/.svn/**
**/.DS_Store
```

Les classpath

Un `classpath` *, élément utile pour certaines tâches*

se définit à l'aide d'éléments `pathelement`. Deux attributs exclusifs :

location spécifie un seul fichier ou répertoire de manière absolue/relative

path spécifie une liste de `location` séparée par un ';' ou un ':'

En général, on utilise `path` avec des chemins prédéfinis et de multiples éléments `locations` dans les autres cas

```
<classpath>
  <pathelement path="${classpath}"/>
  <pathelement location="lib/helper.jar"/>
</classpath>
```

```
<classpath>
  <pathelement path="${classpath}"/>
  <fileset dir="lib">
    <include name="**/*.jar"/>
  </fileset>
  <pathelement location="classes"/>
  <dirset dir="${build.dir}">
    <include name="apps/**/classes"/>
    <exclude name="apps/**/*Test*"/>
  </dirset>
  <filelist refid="third-party_jars"/>
</classpath>
```

Les path

Si l'on souhaite utiliser un même chemin dans plusieurs tâches alors on peut les définir en tant que

Un `path` *élément*

déclaré au niveau des `target`

référencé avec `id/refid`

```
<path id="base.path">
  <pathelement path="${classpath}"/>
  <fileset dir="lib">
    <include name="**/*.jar"/>
  </fileset>
  <pathelement location="classes"/>
</path>

<path id="tests.path">
  <path refid="base.path"/>
  <pathelement location="testclasses"/>
</path>
```

On peut ensuite imaginer un `classpath` d'une tâche qui y réfère

```
<classpath refid="base.path"/>
```

Formes abrégées de classpath et path

Avec `classpath`

```
<classpath>
  <pathelement path="${classpath}"/>
</classpath>
```

peut se réécrire en

```
<classpath path="${classpath}"/>
```

de même avec `path`

```
<path id="base.path">
  <pathelement path="${classpath}"/>
</path>
```

peut se réécrire en

```
<path id="base.path" path="${classpath}"/>
```

Les cibles

Les cibles `<target>`

ensemble de tâches à réaliser dans l'ordre de présentation

- *name* : le nom de la cible. obligatoire
- *description* : brève description de la cible. optionnel (utile pour les IDE)
- *depends* : liste des cibles dont dépend la cible. optionnel
- *if* : conditionne l'exécution par l'existence d'une propriété. optionnel
- *unless* : conditionne l'exécution par l'inexistance de la définition d'une propriété. optionnel

Exemples de mise en oeuvre de tâches – Sommaire

Tâche echo

<echo> permet d'écrire dans un fichier ou d'afficher un message durant l'exécution des traitements

message	the message to echo. Optional
file	the file to write the message to. Optional
append	Append to an existing file (or open a new file / overwrite an existing file) ? Optional - default is false.
level	Control the level at which this message is reported. Optional of "error", "warning" (-quiet, -q), "info" (no statement), "verbose" (-verbose, -v), "debug" (-debug, -d) (decreasing order) Optional - default is "warning".
encoding	encoding to use, default is "" ; the local system encoding. since Ant 1.7 Optional

http://ant.apache.org/manual/CoreTasks/echo.html

Tâche echo

build.echo.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test echo avec Ant" default="displayProperties" basedir=".">

  <target name="displayProperties">
    <echo message="Debut des traitements" />
    <echo>
      Fin des traitements du projet ${ant.project.name}
    </echo>
    <echo message="Ceci est un message warning" level="warning" />
    <echo message="Ceci est un message debug" level="debug"/>
    <echo file="${basedir}/log.txt" append="false" message="Debut Traitement"/>
    <echo file="${basedir}/log.txt" append="true" >
Fin Traitement
    </echo>
    <echoproperties/>
  </target>
</project>
```

Tâche echo

ant -quiet -buildfile build.echo.xml

```
Buildfile: build.echo.xml

init:
     [echo] Debut des traitements
     [echo]
     [echo]       Fin des traitements du projet Test echo avec Ant
     [echo]
     [echo] Ceci est un message warning
***
BUILD SUCCESSFUL
Total time: 0 seconds
```

Tâche echo ***

sélection de lignes affichées :

```
[echoproperties] #Ant properties
[echoproperties] #Wed Oct 22 01:37:44 CEST 2008
[echoproperties] ant.core.lib=/home/hernandez/applications/ant/lib/ant.jar
[echoproperties] ant.file=/media/MyPassport/current/public/teaching/3a.LP.TdD/TD
[echoproperties] ant.home=/home/hernandez/applications/ant
[echoproperties] ant.java.version=1.6
[echoproperties] ant.library.dir=/home/hernandez/applications/ant/lib
[echoproperties] ant.project.name=Test echo avec Ant
[echoproperties] ant.version=Apache Ant version 1.7.0 compiled on November 20 20
[echoproperties] basedir=/media/MyPassport/current/public/teaching/3a.LP.TdD/TD0
[echoproperties] file.encoding=UTF-8
[echoproperties] file.separator=/
[echoproperties] java.class.path=(... trop long ...)
[echoproperties] java.home=/media/MyPassport/applications/jsdk-u5/jdk/jre
[echoproperties] java.io.tmpdir=/tmp
[echoproperties] java.library.path=(... trop long ...)
[echoproperties] java.runtime.name=Java(TM) SE Runtime Environment
[echoproperties] java.runtime.version=1.6.0_06-b02
[echoproperties] line.separator=\n
[echoproperties] os.arch=i386
[echoproperties] os.name=Linux
[echoproperties] os.version=2.6.24-19-generic
[echoproperties] path.separator=\:
[echoproperties] sun.desktop=gnome
[echoproperties] user.country=FR
[echoproperties] user.dir=/media/MyPassport/current/public/teaching/3a.LP.TdD/TD
```

La tâche tstamp

<tstamp> définit trois propriétés :

- *DSTAMP* : la date du jour au format AAAMMJJ
- *TSTAMP* : l'heure actuelle sous la forme HHMM
- *TODAY* : la date du jour au format long

build.tstamp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test tstamp avec Ant" default="init" basedir=".">
  <target name="init">
    <tstamp/>
    <echo message="Nous sommes le ${TODAY}" />
    <echo message="DSTAMP = ${DSTAMP}" />
    <echo message="TSTAMP = ${TSTAMP}" />
  </target>
</project>
```

ant -buildfile build.tstamp.xml

```
Buildfile: build.tstamp.xml
init:
     [echo] Nous sommes le November 21 2007
     [echo] DSTAMP = 20071121
     [echo] TSTAMP = 1120
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche tstamp

<tstamp> définit trois propriétés :

DSTAMP : la date du jour au format AAAMMJJ

TSTAMP : l'heure actuelle sous la forme HHMM

TODAY : la date du jour au format long

build.tstamp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test tstamp avec Ant" default="init" basedir=".">
  <target name="init">
    <tstamp/>
    <echo message="Nous sommes le ${TODAY}" />
    <echo message="DSTAMP = ${DSTAMP}" />
    <echo message="TSTAMP = ${TSTAMP}" />
  </target>
</project>
```

ant -buildfile build.tstamp.xml

```
Buildfile: build.tstamp.xml
init:
     [echo] Nous sommes le November 21 2007
     [echo] DSTAMP = 20071121
     [echo] TSTAMP = 1120
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche tstamp

<tstamp> définit trois propriétés :

DSTAMP : la date du jour au format AAAMMJJ
TSTAMP : l'heure actuelle sous la forme HHMM
TODAY : la date du jour au format long

build.tstamp.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test tstamp avec Ant" default="init" basedir=".">
  <target name="init">
    <tstamp/>
    <echo message="Nous sommes le ${TODAY}" />
    <echo message="DSTAMP = ${DSTAMP}" />
    <echo message="TSTAMP = ${TSTAMP}" />
  </target>
</project>
```

ant -buildfile build.tstamp.xml

```
Buildfile: build.tstamp.xml
init:
     [echo] Nous sommes le November 21 2007
     [echo] DSTAMP = 20071121
     [echo] TSTAMP = 1120
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche mkdir

<mkdir> Creates a directory. Also non-existent parent directories are created, when necessary. Does nothing if the directory already exist.
http://ant.apache.org/manual/CoreTasks/mkdir.html

build.mkdir.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test mkdir avec Ant" default="init" basedir=".">

  <property name="classes.dir" value="build/classes" />
  <property name="dist.dir" value="dist" />

  <target name="init">
    <mkdir dir="${classes.dir}" />
    <mkdir dir="${dist.dir}" />
  </target>
</project>
```

Avec dir, le chemin et le nom du répertoire à créer

ant -buildfile build.mkdir.xml

```
Buildfile: build.mkdir.xml
init:
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/build
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/dist

BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche mkdir

<mkdir> Creates a directory. Also non-existent parent directories are created, when necessary. Does nothing if the directory already exist.
http://ant.apache.org/manual/CoreTasks/mkdir.html

build.mkdir.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test mkdir avec Ant" default="init" basedir=".">

  <property name="classes.dir" value="build/classes" />
  <property name="dist.dir" value="dist" />

  <target name="init">
    <mkdir dir="${classes.dir}" />
    <mkdir dir="${dist.dir}" />
  </target>
</project>
```

Avec dir, le chemin et le nom du répertoire à créer
ant -buildfile build.mkdir.xml

```
Buildfile: build.mkdir.xml
init:
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/build
    [mkdir] Created dir: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/dist

BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche delete

<delete> supprime des fichiers ou des répertoires

build.delete.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test delete avec Ant" default="clean" basedir=".">

  <target name="clean">
    <delete dir="${basedir}/dist"  includeEmptyDirs="true"/>
    <delete file="${basedir}/log.txt" />
    <delete>
      <fileset dir="${basedir}/build" includes="**/*.class" />
    </delete>
     <delete>
      <fileset dir="${basedir}/src" excludes="**/.svn"/>
    </delete>
  </target>
</project>
```

ant -buildfile build.delete.xml

```
Buildfile: build.delete.xml
init:
   [delete] Deleting: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/log.txt
   [delete] Deleting directory /home/hernandez/teaching/TdD/05_NH_CM_ant/test/di
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche delete

<delete> supprime des fichiers ou des répertoires

build.delete.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test delete avec Ant" default="clean" basedir=".">

  <target name="clean">
    <delete dir="${basedir}/dist"  includeEmptyDirs="true"/>
    <delete file="${basedir}/log.txt" />
    <delete>
      <fileset dir="${basedir}/build" includes="**/*.class" />
    </delete>
     <delete>
      <fileset dir="${basedir}/src" excludes="**/.svn"/>
    </delete>
  </target>
</project>
```

ant -buildfile build.delete.xml

```
Buildfile: build.delete.xml
init:
   [delete] Deleting: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/log.txt
   [delete] Deleting directory /home/hernandez/teaching/TdD/05_NH_CM_ant/test/di
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche delete

<delete> supprime des fichiers ou des répertoires

build.delete.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test delete avec Ant" default="clean" basedir=".">

  <target name="clean">
    <delete dir="${basedir}/dist"  includeEmptyDirs="true"/>
    <delete file="${basedir}/log.txt" />
    <delete>
      <fileset dir="${basedir}/build" includes="**/*.class" />
    </delete>
     <delete>
      <fileset dir="${basedir}/src" excludes="**/.svn"/>
    </delete>
  </target>
</project>
```

ant -buildfile build.delete.xml

```
Buildfile: build.delete.xml
init:
   [delete] Deleting: /home/hernandez/teaching/TdD/05_NH_CM_ant/test/log.txt
   [delete] Deleting directory /home/hernandez/teaching/TdD/05_NH_CM_ant/test/di
BUILD SUCCESSFUL
Total time: 0 seconds
```

La tâche copy

<copy> Copies a file or resource collection to a new file or directory. By default, files are only copied if the source file is newer than the destination file, or when the destination file does not exist.

build.copy.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test de copy avec ant" default="init" basedir=".">

  <!-- Definition des proprietes du projet -->
  <property name="sources.dir"    value="src"/>
  <property name="build.dir"      value="bin"/>

  <!-- Initialisation des traitements -->
  <target name="init" description="Initialisation">
    <!-- Copie des fichiers de configuration et parametrage -->
    <copy todir="${build.dir}" >
      <fileset dir="${sources.dir}" >
        <include name="**/*.properties"/>
        <include name="**/*.cfg.xml"/>
      </fileset>
    </copy>
  </target>
</project>
```

D'autres options sont disponibles tofile, overwrite

La tâche copy

<copy> Copies a file or resource collection to a new file or directory. By default, files are only copied if the source file is newer than the destination file, or when the destination file does not exist.

build.copy.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Test de copy avec ant" default="init" basedir=".">

  <!-- Definition des proprietes du projet -->
  <property name="sources.dir"    value="src"/>
  <property name="build.dir"      value="bin"/>

  <!-- Initialisation des traitements -->
  <target name="init" description="Initialisation">
    <!-- Copie des fichiers de configuration et parametrage -->
    <copy todir="${build.dir}" >
      <fileset dir="${sources.dir}" >
        <include name="**/*.properties"/>
        <include name="**/*.cfg.xml"/>
      </fileset>
    </copy>
  </target>
</project>
```

D'autres options sont disponibles `tofile`, `overwrite`

La tâche `javac`

`<javac>` permet la compilation de fichiers source contenus dans une arborescence de répertoires

srcdir	répertoire racine de l'arborescence du répertoire contenant les sources
destdir	répertoire où les résultats des compilations seront stockés
classpath	classpath pour l'exécution. Il est aussi possible d'utiliser un tag fils <classpath> pour le spécifier
classpathref	utilisation d'un classpath précédemment défini dans le fichier de build
fork	lance la compilation dans une JVM dédiée au lieu de celle ou s'exécute Ant. défaut est false
source	version des sources java 1.4, 1.5, ...
deprecation	avertissements du compilateur concernant l'usage d'éléments deprecated. défaut est off
target	précise la version de la plate-forme Java cible (1.1, 1.2, 1.3, 1.4, ...)
...	nowarn, debug, optimize, failonerror

La tâche `javac`

`<javac>` permet la compilation de fichiers source contenus dans une arborescence de répertoires

srcdir	répertoire racine de l'arborescence du répertoire contenant les sources
destdir	répertoire où les résultats des compilations seront stockés
classpath	classpath pour l'exécution. Il est aussi possible d'utiliser un tag fils <classpath> pour le spécifier
classpathref	utilisation d'un classpath précédemment défini dans le fichier de build
fork	lance la compilation dans une JVM dédiée au lieu de celle ou s'exécute Ant. défaut est false
source	version des sources java 1.4, 1.5, ...
deprecation	avertissements du compilateur concernant l'usage d'éléments deprecated. défaut est off
target	précise la version de la plate-forme Java cible (1.1, 1.2, 1.3, 1.4, ...)
...	nowarn, debug, optimize, failonerror

build.javac.xml

```
<xml version="1.0" encoding="UTF-8"?>

<project name="Test javac task" default="compile" basedir=".">
  <!-- Definition des proprietes du projet -->
  <property name="sources.dir" value="src"/>
  <property name="classes.dir" value="build/classes"/>
  <property name="lib.dir" value="lib"/>

  <!-- Definition du classpath pour compiler les sources -->
  <path id="compile.classpath">
    <fileset dir="${lib.dir}">
<patternset>  <include name="*.jar"/> </patternset>
    </fileset>
  </path>

  <!-- Compilation des sources du projet -->
  <target name="compile" description="Compilation des classes">
    <javac srcdir="${sources.dir}"
           destdir="${classes.dir}"
  debug="on"
  optimize="off"
  deprecation="on">
      <classpath refid="compile.classpath"/>
    </javac>
  </target>
</project>
```

La tâche `java`

`<java>` permet de lancer une machine virtuelle pour exécuter une application compilée.

classname	nom pleinement qualifié de la classe à exécuter
jar	nom du fichier de l'application à exécuter
classpath	classpath pour l'exécution.
classpathref	utilisation d'un classpath précédemment défini
fork	lancer l'exécution dans une JVM dédiée au lieu de celle ou l'exécute Ant
output	enregistrer les sorties de la console dans un fichier

La tâche `java`

`<java>` permet de lancer une machine virtuelle pour exécuter une application compilée.

classname	nom pleinement qualifié de la classe à exécuter
jar	nom du fichier de l'application à exécuter
classpath	classpath pour l'exécution.
classpathref	utilisation d'un classpath précédemment défini
fork	lancer l'exécution dans une JVM dédiée au lieu de celle ou l'exécute Ant
output	enregistrer les sorties de la console dans un fichier

La tâche java

build.java.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project name="Test java task" default="run" basedir=".">
  <!-- Definition des proprietes du projet -->
  <property name="sources.dir" value="src"/>
  <property name="classes.dir" value="build/classes"/>
  <property name="lib.dir" value="lib"/>

  <!-- Definition du classpath pour compiler les sources -->
  <path id="run.classpath">
    <fileset dir="${lib.dir}">
<patternset>  <include name="*.jar"/> </patternset>
    </fileset>
    <pathelement location="${classes.dir}"/>
  </path>

  <!-- Execution de HelloWorld -->
  <target name="run" description="Execution de HelloWorld" >
    <java classname="HelloWorld" fork="true">
      <classpath refid="run.classpath"/>
    </java>
  </target>
</project>
```

La tâche javadoc

<javadoc> génération de la documentation au format javadoc des classes incluses dans une arborescence de répertoires

sourcepath le répertoire de base qui contient les sources dont la documentation est à générer

destdir le répertoire qui va contenir les fichiers de documentation générés

build.javadoc.xml

```
<target name="javadoc">
  <mkdir dir="build/doc"/>
  <javadoc sourcepath="src"
              destdir="build/doc" >
   <classpath >
    <path refid="compile.classpath"/>
           <pathelement path="${classes.dir}"/>
       </classpath>
  </javadoc>
</target>
```

La tâche javadoc

<javadoc> génération de la documentation au format javadoc des classes incluses dans une arborescence de répertoires

sourcepath le répertoire de base qui contient les sources dont la documentation est à générer

destdir le répertoire qui va contenir les fichiers de documentation générés

build.javadoc.xml

```
<target name="javadoc">
  <mkdir dir="build/doc"/>
  <javadoc sourcepath="src"
              destdir="build/doc" >
   <classpath >
    <path refid="compile.classpath"/>
           <pathelement path="${classes.dir}"/>
       </classpath>
  </javadoc>
</target>
```

La tâche javadoc

<javadoc> génération de la documentation au format javadoc des classes incluses dans une arborescence de répertoires

sourcepath le répertoire de base qui contient les sources dont la documentation est à générer

destdir le répertoire qui va contenir les fichiers de documentation générés

build.javadoc.xml

```
<target name="javadoc">
  <mkdir dir="build/doc"/>
  <javadoc sourcepath="src"
              destdir="build/doc" >
 <classpath >
  <path refid="compile.classpath"/>
          <pathelement path="${classes.dir}"/>
      </classpath>
  </javadoc>
</target>
```

La tâche jar

<jar> la création d'une archive de type jar

jarfile nom du fichier .jar à créer

basedir répertoire qui contient les éléments à ajouter dans l'archive

compress sprécise si le contenu de l'archive doit être compressé ou non. Par défaut est true

manifest le fichier manifest qui sera utilisé dans l'archive

La tâche jar crée un MANIFEST.MF par défaut qui ne précise pas de main...

build.jar.xml

```
<property name="project.name" value="${ant.project.name}"/>
<property name="project.version" value="1.0"/>

<target name="package" depends="compile" description="Generate jar file">
<jar destfile="${dist.dir}/$[project.name}-${project.version}.jar"
basedir="${classes.dir}">
<manifest>
<attribute name="Built-By" value="${user.name}"/>
<attribute name="Main-Class" value="monPackage.HelloWorld"/>
</manifest>
</jar>
</target>
```

La tâche jar

<jar> la création d'une archive de type jar

jarfile nom du fichier .jar à créer

basedir répertoire qui contient les éléments à ajouter dans l'archive

compress sprécise si le contenu de l'archive doit être compressé ou non. Par défaut est true

manifest le fichier manifest qui sera utilisé dans l'archive

La tâche jar crée un MANIFEST.MF par défaut qui ne précise pas de main...

build.jar.xml

```
<property name="project.name" value="${ant.project.name}"/>
<property name="project.version" value="1.0"/>

<target name="package" depends="compile" description="Generate jar file">
<jar destfile="${dist.dir}/$[project.name}-${project.version}.jar"
basedir="${classes.dir}">
<manifest>
<attribute name="Built-By" value="${user.name}"/>
<attribute name="Main-Class" value="monPackage.HelloWorld"/>
</manifest>
</jar>
</target>
```

La tâche `jar`

`<jar>` la création d'une archive de type jar

jarfile nom du fichier .jar à créer

basedir répertoire qui contient les éléments à ajouter dans l'archive

compress sprécise si le contenu de l'archive doit être compressé ou non. Par défaut est true

manifest le fichier manifest qui sera utilisé dans l'archive

La tâche `jar` crée un `MANIFEST.MF` par défaut qui ne précise pas de main...

build.jar.xml

```
<property name="project.name" value="${ant.project.name}"/>
<property name="project.version" value="1.0"/>

<target name="package" depends="compile" description="Generate jar file">
<jar destfile="${dist.dir}/$[project.name}-${project.version}.jar"
basedir="${classes.dir}">
<manifest>
<attribute name="Built-By" value="${user.name}"/>
<attribute name="Main-Class" value="monPackage.HelloWorld"/>
</manifest>
</jar>
</target>
```

La tâche JUnit

Installation

- Récupérer JUnit.jar `http://www.junit.org/`
- `export CLASSPATH=`*CLASSPATH* `:APPLI/JUnit/junit-4.4.jar`

Description des tâches

<junit> This task runs tests from the JUnit testing framework

<formater> print results of tests in different formats (`plain, xml`)

<test> Defines a single test class

<batchtest> Define a number of tests based on pattern matching

`http://ant.apache.org/manual/OptionalTasks/junit.html`

La tâche JUnit

Installation

- Récupérer JUnit.jar `http://www.junit.org/`
- `export CLASSPATH=`*CLASSPATH* `:APPLI/JUnit/junit-4.4.jar`

Description des tâches

<junit>	This task runs tests from the JUnit testing framework
<formater>	print results of tests in different formats (`plain, xml`)
<test>	Defines a single test class
<batchtest>	Define a number of tests based on pattern matching

`http://ant.apache.org/manual/OptionalTasks/junit.html`

La tâche JUnit

Installation

- Récupérer JUnit.jar `http://www.junit.org/`
- `export CLASSPATH=`*CLASSPATH* `:APPLI/JUnit/junit-4.4.jar`

Description des tâches

<junit>	This task runs tests from the JUnit testing framework
<formater>	print results of tests in different formats (`plain, xml`)
<test>	Defines a single test class
<batchtest>	Define a number of tests based on pattern matching

`http://ant.apache.org/manual/OptionalTasks/junit.html`

La tâche JUnit

build.junit.xml

```
<junit printsummary="yes" haltonfailure="yes">
  <classpath refid="${test.classpath}"/>

  <formatter type="plain"/>

  <test name="my.test.TestCase" haltonfailure="no" outfile="result">
    <formatter type="xml"/>
  </test>

  <batchtest fork="yes" todir="${reports.tests}">
    <fileset dir="${src.tests}">
      <include name="**/*Test*.java"/>
      <exclude name="**/AllTests.java"/>
    </fileset>
  </batchtest>
</junit>
```

printsummary	Print one-line statistics for each testcase.
fork	Run the tests in a separate VM.
haltonfailure	Stop the build process if a test fails
timeout	Cancel the individual tests if they don’t finish in the given time
todir	Directory to write the reports to

La tâche svn

Installation

- Récupérer svnant.jar, svnClientAdapter.jar et svnjavahl.jar à partir de l'archive svnant http://subclipse.tigris.org/svnant.html
- Les mettre dans le classpath...

Description de la tâche svn

- http://subclipse.tigris.org/svnant/svn.html
- Exemple de build.xml mettant en oeuvre svnant dans l'archive récupérée ci-dessus
- Autres exemples d'utilisation subversion.open.collab.net/articles/IntegratingSubversionIntoYourAntBuild.html

La tâche svn

Un exemple de build.properties

```
# build.properties
svnant.version=1.0.0

lib.dir=lib
svnant.jar=${lib.dir}/svnant.jar
svnClientAdapter.jar=${lib.dir}/svnClientAdapter.jar
svnjavahl.jar=${lib.dir}/svnjavahl.jar

svnant.latest.url=http://subclipse.tigris.org/svn/subclipse/trunk/svnant/
svnant.this.url=http://subclipse.tigris.org/svn/subclipse/tags/svnant/${svnant.v
ersion}/

svnant.repository.user=guest
svnant.repository.passwd=""
```

build.xml (part 1/2)

```
  <!--  all properties are in build.properties -->
  <property file="build.properties" />

  <!-- path to the svnant libraries. Usually in ANT_HOME/lib -->
  <path id="project.classpath">
    <pathelement location="${svnjavahl.jar}" />
    <pathelement location="${svnant.jar}" />
    <pathelement location="${svnClientAdapter.jar}" />
  </path>
```

La tâche svn

build.xml (part 2/2)

```
...
  <!-- load the svn task -->
  <taskdef resource="svntask.properties" classpathref="project.classpath"/>

  <target name="clean">
    <delete dir="src_latest"/>
    <delete dir="src_${svnant.version}"/>
  </target>

  <target name="checkoutLatest">
    <svn username="${svnant.repository.user}" password="${svnant.repository.pass
wd}">
      <checkout url="${svnant.latest.url}" revision="HEAD" destPath="src_latest"
 />
    </svn>
  </target>

  <target name="checkoutThis">
    <svn username="${svnant.repository.user}" password="${svnant.repository.pass
wd}">
      <checkout url="${svnant.this.url}" revision="HEAD" destPath="src_${svnant.
version}" />
    </svn>
  </target>
```

Catégories de tâches – Sommaire

File / Directory / Archive tasks
- File / Directory tasks
- Archive Tasks

Development tasks
- Compile Tasks
- Documentation, Logging and Testing Tasks

Execution Tasks
- Execution Tasks
- Remote Tasks

Misc and Property tasks
- Misc tasks
- Property Tasks

Un aperçu complet des tâches standardes disponibles :
`http://ant.apache.org/manual/tasksoverview.html`

File / Directory tasks

File / Directory tasks

Copy	Copies a file or Fileset to a new file or directory.
Delete	Deletes either a single file, all files and sub-directories in a specified directory, or a set of files specified by one or more FileSets.
Mkdir	Creates a directory. Non-existent parent directories are created, when necessary.
Move	Moves a file to a new file or directory, or a set(s) of file(s) to a new directory.
Get	Gets a file from a URL.
Patch	Applies a "diff" file to originals.
...	FixCRLF, Replace, ReplaceRegExp, Sync, Tempfile, Touch, Checksum, Chgrp, Chmod, Chown, Concat

Archive Tasks

Zip / Unzip	Zips a set of files. / Expands a Zip file.
Jar/Unjar	Jars a set of files./ Unzips a jarfile.
Manifest	Creates a manifest file.
Rpm	Invokes the rpm executable to build a Linux installation file. This task currently only works on Linux or other Unix platforms with RPM support.
Tar/Untar	Creates a tar archive / Untars a tarfile.
...	BUnzip2, BZip2, Cab, Ear, GZip, GUnzip War, Unwar

Compile Tasks

Compile Tasks

Javac	Compiles the specified source file(s) within the running (Ant) VM, or in another VM if the fork attribute is specified.
Depend	Determines which classfiles are out-of-date with respect to their source, removing the classfiles of any other classes that depend on the out-of-date classes, forcing the re-compile of the removed classfiles. Typically used in conjunction with the Javac task.
JspC	Runs the JSP compiler. It can be used to precompile JSP pages for fast initial invocation of JSP pages, deployment on a server without the full JDK installed, or simply to syntax-check the pages without deploying them. The Javac task can be used to compile the generated Java source. (For Weblogic JSP compiles, see the Wljspc task.)

Documentation, Logging and Testing Tasks

Documentation Tasks

Javadoc Generates code documentation using the javadoc tool.

Logging Tasks

Record Runs a listener that records the logging output of the build-process events to a file. Several recorders can exist at the same time. Each recorder is associated with a file.

Testing Tasks

Junit Runs tests from the Junit testing framework. This task has been tested with JUnit 3.0 up to JUnit 3.7 ; it won't work with versions prior to JUnit 3.0.

JunitReport Merges the individual XML files generated by the Junit task and applies a stylesheet on the resulting merged document to provide a browsable report of the testcases results.

Execution Tasks

Execution Tasks

Ant	Runs Ant on a supplied buildfile,
AntCall	Runs another target within the same buildfile,
Apply/ExecOn ; Exec	Executes a system command.
Java	Executes a Java class within the running (Ant) VM, or in another VM if the fork attribute is specified.
Parallel	A container task that can contain other Ant tasks. Each nested task specified within the <parallel> tag will be executed in its own thread.
Sequential	A container task that can contain other Ant tasks. The nested tasks are simply executed in sequence. Its primary use is to support the sequential execution of a subset of tasks within the <parallel> tag.
Sleep	A task for suspending execution for a specified period of time. Useful when a build or deployment process requires an interval between tasks.

Remote Tasks

FTP	Implements a basic FTP client that can send, receive, list, and delete files, and create directories.
Scp	Copy files to or from a remote server using SSH.
setproxy	Sets Java's web proxy properties, so that tasks and code run in the same JVM can have through-the-firewall access to remote web sites.
Sshexec	Execute a command on a remote server using SSH.
Telnet	Task to automate a remote telnet session. This task uses nested <read> and <write> tags to indicate strings to wait for and specify text to send.

Misc tasks

Mail	A task to send SMTP email
Echo	Echoes text to System.out or to a file.
Fail	Exits the current build by throwing a BuildException, optionally printing additional information.
Input	Allows user interaction during the build process by displaying a message and reading a line of input from the console.
Sound	Plays a sound file at the end of the build, according to whether the build failed or succeeded.
Sql	Executes a series of SQL statements via JDBC to a database. Statements can either be read in from a text file using the src attribute, or from between the enclosing SQL tags.
TStamp	Sets the DSTAMP, TSTAMP, and TODAY properties in the current project, based on the current date and time.
XmlValidate	Checks that XML files are valid (or only well-formed). This task uses the XML parser that is currently used by Ant by default, but any SAX1/2 parser can be specified, if needed.

Property Tasks

Property Tasks

Available	Sets a property if a specified file, directory, class in the classpath, or JVM system resource is available at runtime.
Basename	Sets a property to the last element of a specified path.
Dirname	Sets a property to the value of the specified file up to, but not including, the last path element.
Condition	Sets a property if a certain condition holds true ; this is a generalization of Available and Uptodate.
XmlProperty	Loads property values from a well-formed XML file.
...	Whichresource, Echoproperties, LoadFile, LoadProperties, MakeURL, PathConvert, Property, PropertyFile, Uptodate,

Définir ses propres tâches – Sommaire

Définir une tâche `ant` *dans le* `build.xml`

Ant permet de définir ces propres tâches

```
<?xml version="1.0"?>
<project name="ExempleNotreTache" default="main" basedir=".">

<taskdef name="matache" classname="tdd.MaTacheAMoi"/>

<target name="main">
<matache message="Ant is Great !"/>
</target>
</project>
```

La classe implémentant cette tâche

- La classe implémentant cette tâche doit **étendre** `org.apache.tools.ant.Task`
- Un **accesseur en écriture** doit être défini pour chaque attribut. Il aura la forme traditionnelle en Java : `setNomAttribut`.
  Le type recu en paramètre de cette méthode peut être String, ou n'importe quel des types de base, Ant se chargeant des conversions.
- **Chaque sous-élément supporté par la tâche** devra de la même manière être traité par le biais de méthodes `createNomElement` **ou** `addNomElement`
- La classe doit enfin comporter **une méthode** `public void execute() throws BuildException`

Code de la classe implémentant la tâche

```
Package tdd;

import org.apache.tools.ant.BuildException;
import org.apache.tools.ant.Task;

public class MaTacheAMoi extends Task {
private String msg;

// La méthode appelée par Ant pour l'execution de la tache
public void execute() throws BuildException {
System.out.println(msg);
}

// Accesseur pour l'attribut message
public void setMessage(String msg) {
this.msg = msg;
}
}
```

Conclusion – Sommaire

ant *et Eclipse*

Documentation
`http://help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.user/`

- puis `gettingStarted/qs-81_basics.htm`
- ou `concepts/concepts-antsupport.htm`

En bref...

Créer un projet à partir d'un `ant buildfile` *ou en ajouter un*

- soit `File > New > Java Project > Create a new project java from an existing ant buildfile`
- soit `File > New > File`

Exécuter

- dans la vue `Package`, bouton droit sur le fichier build.xml sélectionné
- ou bien Icone *Run avec la malette* de la barre de menu horizontale
- ou bien dans la vue *Outline*, bouton droit sur une des cibles

Puis `Run as > Ant Build`

Conclusion

Synthèse

- multi-plate-forme
- configurable grâce à un fichier XML
- open-source
- extensible

Perspective

- actuellement la version 1.9.4 (05-May-2014) ; sur Ubuntu 12.04 la version 1.8.2 (03-Dec-2011) ; possibilité de problèmes de compatibilité suivant la version installée avec java/doc en ligne et vos build.xml ...
- `make` le passé et `maven`, le futur ?
- Apache Ivy is a very powerful dependency manager oriented toward Java dependency management, even though it could be used to manage dependencies of any kind.
  `http://ant.apache.org/ivy/features.html`

Bibliographie

Gestion de Version

- Site officiel `http://ant.apache.org/`
- Manuel d'utilisateur `http://ant.apache.org/manual/index.html` notamment
  - Using Ant > Properties, Path-like Structures...
  - Ant Tasks > Overview of Ant Tasks, Core Tasks, Optional Tasks...
  - Concepts et types > filelist, patternset, dirset, fileset...
  - Tutoriels > Hello World with Ant
- Ant et Eclipse
  http ://`help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.user/` puis `gettingStarted/qs-81_basics.htm` ou `concepts/concepts-antsupport.htm`
- Manuels non-officiel et non-complet de `ant` et `maven` (en français)
  `http://www.jmdoudoux.fr/java/dej/index.htm`
- Ant et JUnit
  `http://ant.apache.org/manual/OptionalTasks/junit.html`
- Ant et svn `http://subclipse.tigris.org/svnant.html`