Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo:
Ant
Parte a

versione 1.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

Introduzione

- Ant (“Another Neat Tool”)
  - un linguaggio
  - con una sintassi ed una semantica
  - non si tratta di un linguaggio di programmazione, ma di un linguaggio per la costruzione del codice
- Altri esempi di linguaggi
  - linguaggi di interrogazione (es: SQL)
  - linguaggi di composizione (es: HTML)

Introduzione

- Perché serve uno sistema di costruzione?
  - consideriamo una operazione fondamentale: la compilazione
  - in Java richiede di effettuare tipicamente varie operazioni
  - impostare il classpath
  - compilare i vari package separatamente
  - il tutto deve essere ripetuto frequentemente
  - sarebbe ideale automatizzare il processo

Introduzione

- Lo strumento di costruzione tradizionale
  - ⇨Make
  - ⇨utilizzato nella stragrande maggioranza di progetti basati su C/C++
- Idea
  - ⇨un makefile consente di specificare sequenze di chiamate a strumenti del S.O.
  - ⇨es: bcc32, ilink32, copy, del ecc.

Introduzione

- Le caratteristiche negative di Make
  - ⇨ovvero le sue rughe (“make’s wrinkes”)
- Complesso da utilizzare
  - ⇨la sintassi è complicata
- Dipendente dalla piattaforma
  - ⇨i makefile sono difficilmente portabili
- Non estensibile
  - ⇨non è possibile aggiungere nuove operazioni

Introduzione

- Ant: la formica
  - ⇨“the only bug in your software”
  - ⇨“ants find the shortest distance around obstacles”
  - ⇨“ants carry 50 times their weight”
  - ⇨“ants work around the clock”
- La definizione più nota
  - ⇨“make without make’s wrinkles”

Introduzione

- Le caratteristiche di Ant
  - ⇨semplice: basato su XML, con una sintassi semplice da scrivere e da interpretare
  - ⇨indipendente dalla piattaforma: completamente basato su Java
  - ⇨estensibile: consente di definire nuove operazioni di costruzione
  - ⇨scalabile: consente di gestire progetti piccoli come anche progetti enormi

Introduzione

- La storia di Ant
  - la versione 1.0 è del 2000
  - è rapidamente diventato uno strumento diffusissimo
  - utilizzato praticamente da tutti i progetti open source della comunità Java, e da moltissimi progetti commerciali
  - attualmente alla versione 1.6
  - in vista la versione 2.0

Installazione

- Installazione di Ant
  - scaricare il file .zip da http://ant.apache.org
  - definire la variabile di ambiente ANT_HOME
  - aggiornare il PATH aggiungendo la cartella %ANT_HOME%\bin che contiene ant.bat
  - nota: non è necessario aggiornare il CLASSPATH
  - per verificare: dal prompt di dos digitare ant -version

Installazione

- Nota
  - Anti richiede che sulla macchina sia installato l'SDK (non basta il JRE)
  - Ant richiede un parser conforme con JAXP disponibile sul CLASSPATH
  - include nella distribuzione Xerces (in %ANT_HOME%\lib), compatibile con JAXP 1.3
  - nel caso sia necessario utilizzare parser diversi con JAXP (es: JAXP 1.1) possono verificarsi conflitti sul classpath

Preliminari

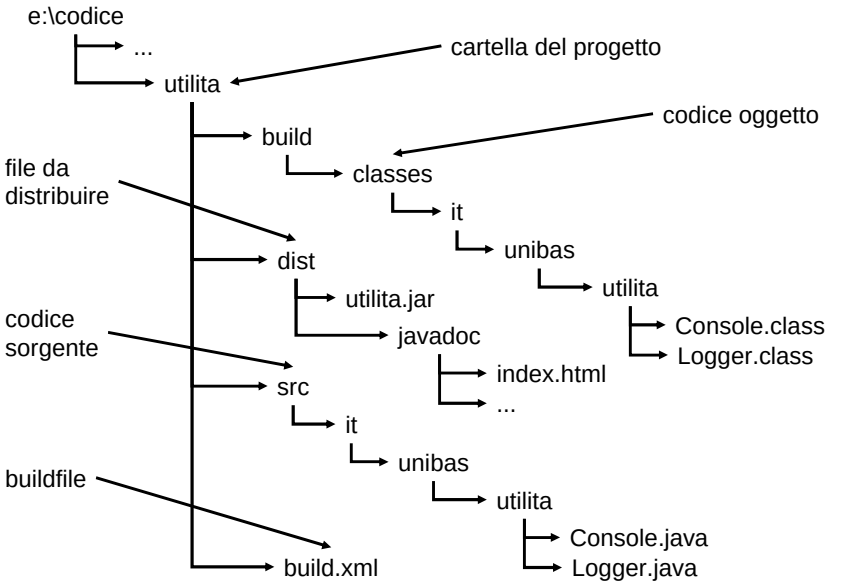
- File di build ("build file")
  - "programmi" per ant
  - ciascun file corrisponde ad un progetto soft. specifica le operazioni di build relative
- Operazioni tipiche
  - compilazione
  - distribuzione
  - deployment

Preliminari

- Finora
  - ⇨le cartelle corrispondenti ai progetti erano organizzate secondo i package
  - ⇨il codice sorgente ed il codice oggetto erano mischiati
- Avendo a disposizione un sistema di build
  - ⇨è possibile riorganizzare completamente la struttura della cartella di progetto

Preliminari

- Nuova struttura della cartella di progetto
  - ⇨le sottocartelle sono organizzate in modo da separare i componenti
  - ⇨codice sorgente, codice oggetto, jar, javadoc
- Organizzazione standard
  - ⇨codice sorgente nella cartella src
  - ⇨codice oggetto nella cartella build/classes
  - ⇨jar e documentazione nella cartella dist



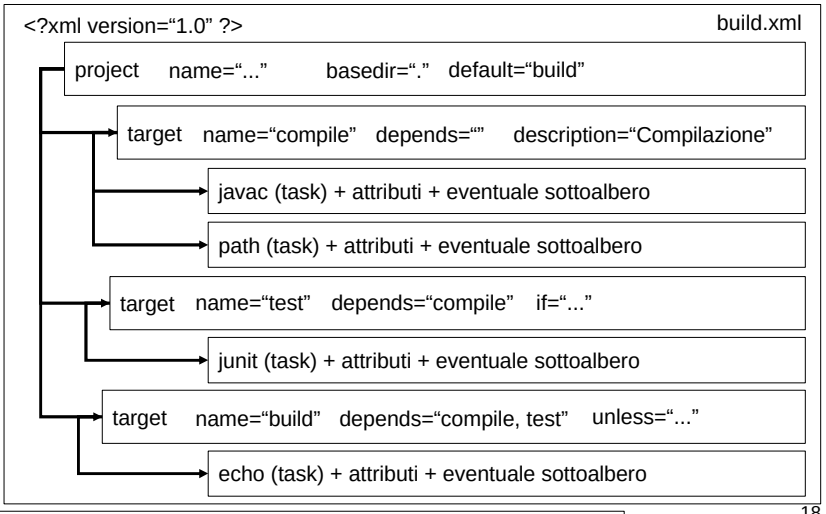
Preliminari

- Una annotazione
  - ⇨l'unica cartella di materiale indispensabile è la cartella src
  - ⇨le altre due cartelle contengono materiale generato a partire dal codice
  - ⇨possono essere eliminate e ricreate nel caso in cui si voglia "ricostruire" da zero il progetto
  - ⇨si tratta di "cartelle riproducibili"

Struttura di un File di Build

- File di build
  - ⇨documento XML
  - ⇨costruito secondo uno schema opportuno
- Contenuto di un file di build
  - ⇨ciascun file corrisponde ad un progetto
  - ⇨il progetto contiene uno o più target
  - ⇨ciascun target contiene uno o più task

Struttura di un File di Build



Struttura di un File di Build

- Esempio: il progetto it.unibas.utilita
  - ⇨file di build build-semplice.xml
- Target principali
  - ⇨compile: compila il codice
  - ⇨javadoc: produce la documentazione
  - ⇨jar: produce il jar
  - ⇨deploy: rende la libreria visibile nel classpath copiandola nella cartella e:\codice\lib

Struttura di un File di Build

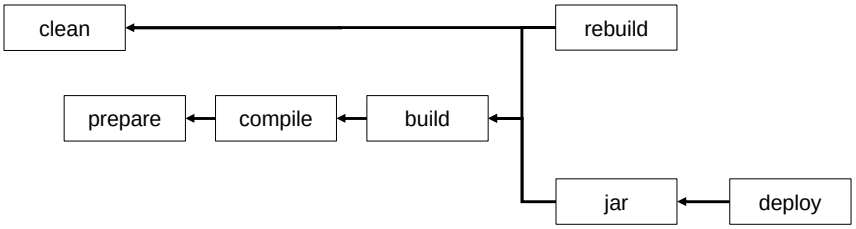
>> utilita

>> utilita\build-semplice.xml

- Target ausiliari
  - ⇨prepare: crea le cartelle build/classes e dist
  - ⇨clean: elimina le cartelle riproducibili
- Due target interessanti
  - ⇨build: in questo caso semplice coincide con compile; è il target predefinito
  - ⇨rebuild: build preceduto da clean (ricostruisce il codice da zero)

Struttura di un File di Build

- Il grafo delle dipendenze
  - ⇨le dipendenze tra i task sono descrivibili attraverso un grafo

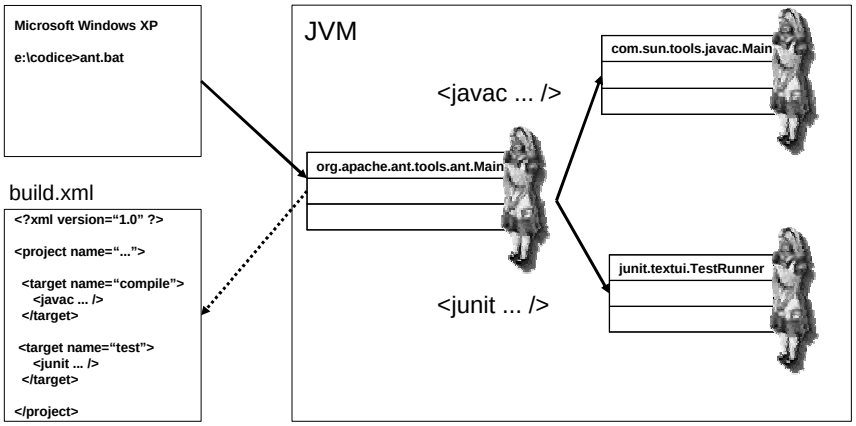


Esecuzione di Ant

- Esecuzione di Ant
  - ⇨Ant è un'applicazione Java
  - ⇨eseguendo Ant viene avviata la macchina virtuale ed eseguita la classe org.apache.ant.tools.ant.Main
- Per ciascun target
  - ⇨Ant crea oggetti ed esegue metodi di opportune classi Java che implementano il task

Esecuzione di Ant

java org.apache.ant.tools.ant.Main



Esecuzione di Ant

- Sulla base di questo meccanismo
  - ⇨si capiscono le caratteristiche di Ant
  - ⇨semplicità: i file di build sono documenti XML
  - ⇨portabilità: tutte le operazioni sono scritte in Java
  - ⇨estensibilità: per scrivere nuovi task è sufficiente sviluppare le classi corrispondenti

Esecuzione di Ant

- Comando da eseguire
  - ⇨comando ant dal prompt dei comandi
  - ⇨cerca nella cartella corrente un file build.xml ed esegue il target di default
- Varianti
  - ⇨consentono di specificare un target diverso
  - ⇨e/o un file di build diverso da build.xml

Esecuzione di Ant

- Sintassi delle varianti
  - ⇨ant <target> es: ant rebuild
    esegue il target specificato
  - ⇨ant –buildfile <file>.xml oppure ant –f <file>.xml
    es: ant –buildfile build-semplice.xml
    egue il target di default del file specificato
  - ⇨ant –f <file>.xml <target>
    es: ant –f build-semplice.xml dist
    esegue il target specificato del file specificato

Esecuzione di Ant

- Semantica dell'esecuzione
  - ⇨ant individua il file di build da eseguire
  - ⇨individua il target da eseguire
  - ⇨ricostruisce il grafo delle dipendenze e costruisce una sequenza di build ("build sequence") che soddisfi le dipendenze
- Esempio: build sequence per rebuild
  - ⇨depends="clean, build"
  - ⇨clean, prepare, compile, build, rebuild

Esecuzione di Ant

- Semantica dell'esecuzione (continua)
  - ⇨ant esegue i target secondo la sequenza individuata
  - ⇨nell'esecuzione dei target, evita di ripetere operazioni inutili
- Esempio
  - ⇨non ricrea le cartelle che esistono
  - ⇨non ricompila le classi già compilate

Esecuzione di Ant

- Per analizzare l'esecuzione
  - ⇨opzione –verbose
  - ⇨per vedere tutte le opzioni: ant –help
- Esempi
  - ⇨ant –verbose –f build-semplice.xml
  - ⇨ant –verbose –f build-semplice.xml (ancora)
  - ⇨ant –verbose –f build-semplice.xml rebuild

Principali Task

- Task di Ant
  - ⇨due categorie
- Task principali ("Core tasks")
  - ⇨task predefiniti forniti a corredo di Ant e pronti per l'uso
- Task opzionali ("Optional tasks")
  - ⇨il codice dei task è fornito a corredo di Ant, ma l'esecuzione dei task richiede tipicamente librerie aggiuntive (es: JUnit)

Principali Task

- Nel complesso
  - ⇨83 task principali
  - ⇨60 task opzionali
- In queste lezioni
  - ⇨descriveremo solo i task principali
  - ⇨per una descrizione completa è possibile fare riferimento al manuale di Ant
  - ⇨%ANT_HOME%\docs\manuale

Principali Task

- Nel file di build che abbiamo visto
  - ⇨task che operano sui file e sulle cartelle
  - ⇨task che eseguono strumenti dell'SDK
  - ⇨il task echo
- Task che operano su file e cartelle
  - ⇨<delete dir="<*cartella*>" />
    es: <delete dir="./build/classes" />
  - ⇨<mkdir dir="<*cartella*>" />
    es: <mkdir dir="./dist" />
  - ⇨<copy file="<*file*>" todir="<*cartella*>" />
    es: <copy file="./dist/utilita.jar" todir="e:/codice/lib" />

Principali Task

- Riferimenti ai file
  - riferimenti assoluti; es: e:\codice\lib
  - riferimenti relativi; es: ./build/classes
- Regole sui riferimenti
  - possono essere utilizzati entrambi i tipi di separatori (\ oppure /)
  - i riferimenti relativi vengono espansi rispetto alla "basedir" del progetto (per convenzione coincide con ".", cioè la cartella del buildfile)

Principali Task

- Task che eseguono strumenti dell'SDK
  - il task javac
  - il task jar
  - il task javadoc
- Il task javac
  - è probabilmente il task principale di Ant
  - struttura molto complessa
  - funzionamento sofisticato

Principali Task

ATTENZIONE
al funzionamento
del task javac

- Esempio semplice di javac
  - ⇨<javac srcdir="./src" destdir="./build/classes" />
- Caratteristiche del task javac
  - ⇨è ricorsivo, ovvero considera tutte le sottocartelle della srcdir
  - ⇨analizza solo i file .java e ignora gli altri
  - ⇨riproduce la struttura dei package in destdir
  - ⇨effettua un controllo sui .class e ricompila il codice solo se il .java è più recente del .class

Principali Task

ATTENZIONE
al classpath
per il task javac

- Il classpath visibile nel task javac
  - ⇨è completamente configurabile (>>)
- Il classpath standard
  - ⇨include il classpath di sistema (valore della variabile CLASSPATH o classpath predefinito)
  - ⇨include tutti i jar contenuti nella cartella %ANT_HOME%\lib

>> %ANT_HOME%\lib

Principali Task

- Il task jar
  - ⇨<jar destfile="<*file.jar*>" basedir="<*cartella*>" />
    es: <jar destfile="./dist/utilita.jar" basedir="./build/classes" />
- Il task javadoc
  - ⇨<javadoc destdir="<*cartellaDoc*>"
    sourcepath="<*cartellaSrc*>"
    packagenames="<*pattern*>" />
    es: <javadoc destdir="./dist/javadoc"
    sourcepath="./src"
    packagenames="*" />

Principali Task

- Il task echo
  - ⇨Ant include un proprio sistema di logging
  - ⇨il task per produrre messaggi di logging è echo
- Livelli di logging (in ordine di priorità)
  - ⇨error, warning, info, verbose, debug
  - ⇨il livello standard è warning
- L'handler standard: console

Principali Task

- Esempi

```
<echo message="Sto effettuando il deploy" />
<echo>Sto effettuando il deploy"</echo>
<echo message="Sto effettuando il deploy"
        level="verbose" />
<echo message="Sto effettuando il deploy"
        level="debug" />
<echo file="log.txt" level="error">Deploy fallito</echo>
```


Principali Task

- Esecuzione di Ant
  - ⇨ant : livello: warning
    messaggi: error e warning
  - ⇨ant –verbose : livello: verbose
    messaggi: verbose, info, warning, error
  - ⇨ant –debug : livello: debug
    messaggi: tutti
  - ⇨ant –quiet : livello: error
    messaggi: solo error

Principali Task

- Nota
  - ⇨tipicamente i messaggi di log sono stampati sulla console con un “prefisso” (“adornment”)
  - ⇨es: [echo] Build di debug
  - ⇨es: [mkdir] Created dir: E:\codice\...\build\classes
- Per evitarlo
  - ⇨eseguire ant con l’opzione –e (-emacs)

Riassumendo

- Introduzione
- Installazione
- Preliminari
- Struttura di un File di Build
- Esecuzione di Ant
- Principali Task

Termini della Licenza