Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo:
Ant
Parte b

versione 1.0

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

Sommario

- Proprietà e Percorsi
  - ⇨Proprietà
  - ⇨Percorsi
  - ⇨Fileset
- Altri Task Importanti
  - ⇨Il Task junit
- Struttura del file di build

Proprietà e Percorsi

- Come tutti i linguaggi
  - ⇨anche Ant consente di utilizzare dati
- Dati di Ant
  - ⇨si tratta sostanzialmente di costanti simboliche
  - ⇨ovvero dati “immutabili” con un nome
  - ⇨sono sostanzialmente di due tipi: proprietà e percorsi

Proprietà e Percorsi

- Proprietà
  - ⇨coppie nome=valore (valore è una stringa)
  - ⇨nello spirito di java.util.properties
  - ⇨si utilizzano nel file di build con la sintassi ${nome}
- Obiettivo
  - ⇨rendere configurabile il processo di build stabilendo i valori delle proprietà in modo da poterli rapidamente cambiare

Proprietà e Percorsi

- Un esempio
  - ⇨i riferimenti alle cartelle
  - ⇨è opportuno definire una serie di proprietà, in modo che sia possibile riorganizzare rapidamente, all'occorrenza, l'albero delle cartelle senza dover cambiare i target
- Definizione delle proprietà
  - ⇨attraverso il task property

Proprietà e Percorsi

- Percorsi
  - ⇨un tipo di dato essenziale per la piattaforma Java
  - ⇨consente di definire classpath nel file di build
- Contenuto di un percorso
  - ⇨pathelement, ovvero nomi di file jar o cartelle
  - ⇨fileset, ovvero collezione di file in una cartella

Proprietà e Percorsi

- Definizione di un percorso
  - ⇨attraverso l'elemento path
  - ⇨prevede un sottoalbero fatto di elementi pathelement e fileset
  - ⇨può avere un id unico
  - ⇨in modo da poter fare riferimento al percorso in altri target/task del buildfile

Proprietà e Percorsi

>> indovinaIlNumero
>> build-semplice.xml

- Esempio
  - ⇨il buildfile di indovinaIlNumero
- Utilizzo delle proprietà
  - ⇨per definire tutti i riferimenti alle cartelle
- Utilizzo dei percorsi
  - ⇨per definire il classpath da utilizzare per la compilazione del codice, la compilazione dei test, l'esecuzione del codice e dei test

Proprietà

- Definire le proprietà
  - ⇨esistono vari modi
- I modi principali
  - ⇨acquisire dati da un file .properties
  - ⇨passare le proprietà ad Ant sulla linea di comando con –Dnome=valore
  - ⇨utilizzare il task property

Proprietà

- Il task property
  - ⇨consente di definire una proprietà all'interno del file di build
- Sintassi
  - ⇨<property name="<*nome*>" value="<*value*>" />
    es: <property name="build.sysclasspath" value="ignore" />
  - ⇨una volta attribuito il valore alla proprietà, questo è immutabile; altri task property eseguiti successivamente non hanno effetto sulla proprietà

Proprietà

- Una variante
  - ⇨l'attributo location per i percorsi
  - ⇨<property name="<*nome*>" location="<*percorso*>" />
    es: <property name="src.dir" location="./src" />
  - ⇨la versione con l'attributo location è preferibile per i riferimenti alle cartelle, perchè Ant espande il valore del riferimento rendendolo assoluto
    es: e:\codice\lavoro\java\indovinaIlNumero\src

Proprietà

- Proprietà predefinite accessibili
  - ⇨tutte le proprietà di sistema predefinite (>> java.util.properties)
- Le proprietà speciali di Ant
  - ⇨basedir (cartella base del progetto)
  - ⇨ant.project.name (nome del progetto)
  - ⇨ant.file (nome del file di build)
  - ⇨ant.version (versione di Ant)
  - ⇨ant.java.version (versione della JVM)

Percorsi

- Definizione di percorsi
  - ⇨<path id="<*idUnico*>">
  - ⇨all'interno: pathelement e/o fileSet
- Pathelement
  - ⇨location: rif. ad un singolo file o cartella
  - ⇨path: path complesso (i componenti possono essere separati con ; oppure con : indifferentemente)

Percorsi

```
<target name="init-classpath" depends="init-folders">
      <property name="build.sysclasspath" value="ignore" />
      <path id="compile.classpath">
         <pathelement location="${build.dir}" />
         <pathelement location="${mylib.dir}/utilita.jar" />
      </path>
      <path id="run.classpath">
         <path refid="compile.classpath" />
      </path>
      <path id="test.classpath">
         <path refid="compile.classpath" />
         <pathelement location="${test.build.dir}" />
         <pathelement location="${lib.dir}/junit.jar" />
      </path>
  </target>
```

Percorsi

- Di conseguenza
  - ⇨in un buildfile possono essere definiti vari classpath (es: per il codice, per i test ecc.)
  - ⇨il classpath è completamente controllabile
- La proprietà build.sysclasspath
  - ⇨stabilisce se considerare in un certo classpath gli elementi della variabile CLASSPATH
  - ⇨valori possibili: only, ignore, last, first

Percorsi

il classpath visibile per il task javac è composto da:
- ${build.dir}
- ${myLib.dir}\utilita.jar
- i jar di %ANT_HOME%\lib

il classpath di sistema è ignorato

```
<target name="compile" depends="prepare">
    <javac srcdir="${src.dir}"
           destdir="${build.dir}"
           classpathref="compile.classpath" />
</target>

<target name="compile-test" depends="compile">
    <javac srcdir="${test.src.dir}"
           destdir="${test.build.dir}"
           classpathref="test.classpath" />
</target>
```

il classpath visibile per il task javac è composto da:
- dal classpath precedente
- £{test.build.dir}
- ${lib.dir}\junit.jar

Fileset

- Fileset
  - definiti attraverso l'elemento fileSet
  - collezione di file contenuti in una cartella
  - consente di specificare pattern di inclusione ed esclusione
  - attraverso gli elementi include ed exclude
- Pattern
  - modello che descrive quali nomi di file includere o escludere

Fileset

- Caratteri speciali
  - *: qualsiasi stringa
  - **: qualsiasi cartella o sottocartella
- Esempio
  - identificare tutte le classi di test da eseguire attraverso il task junit

tutti i file nella cartella ${test.build.dir} o nelle sottocartelle il cui nome comincia per Test e finisce per .class

```
<fileset dir="${test.build.dir}">
      <include name="**/Test*.class" />
</fileset>
```

Fileset

- Esclusioni standard
  - ⇨da un fileset vengono sempre esclusi certi pattern corrispondenti tipicamente a file ausiliari
- File di backup creati dagli editor
  - ⇨**/*~, **/#*#, **/.#*, **/%*%
- File legati al controllo delle versioni
  - ⇨**/cvs, **/cvs/**
  - ⇨**/svn, **/svn/**

Fileset

- Un esempio complesso di path

```
<target name="init-classpath" depends="init-folders">
    <property name="build.sysclasspath" value="ignore" />
    <path id="test.classpath">
      <pathelement path="${build.dir};${test.build.dir}" />
      <pathelement location="${mylib.dir}/utilita.jar;" />
      <fileset dir="${project.lib.dir}">
        <include name="**/*.jar" />
        <exclude name="xerces*.jar" />
      </fileset>
    </path>
</target>
```

Altri Task Importanti

- Altri task importanti nell'esempio
  - ⇨il task zip
  - ⇨il task java
  - ⇨il task manifest
  - ⇨il task junit
- Il task zip
  - ⇨produce un file zip
  - ⇨sulla base di uno zipfileset

Altri Task Importanti

>> ant –f build-semplice.xml dist

- Zipfileset
  - ⇨simile ad un fileset; in più l'attributo prefix

```
<target name="dist" depends="rebuild, javadoc">
  <zip zipfile="${dist.dir}/${ant.project.name}-java.zip">
    <zipfileset dir="${src.dir}" prefix="${ant.project.name}-java" />
    <zipfileset dir="${build.dir}" prefix="${ant.project.name}-java" />
    <zipfileset dir="${test.src.dir}" prefix="${ant.project.name}-java/test/src“
                includes="**/*.java" />
    <zipfileset dir="${test.dir}" prefix="${ant.project.name}-java/test" />
    <zipfileset dir="${utilita.build.dir}" prefix="${ant.project.name}-java" />
    <zipfileset dir="${javadoc.dir}" prefix="${ant.project.name}-java/javadoc" />
  </zip>
</target>
```

Altri Task Importanti

>> ant –f build-semplice.xml java

- Il task java
  - ⇨serve ad eseguire l'applicazione
  - ⇨attenzione alla modalità di esecuzione
- La sintassi

```
<target name="run" depends="rebuild">
    <java fork="false" classname="${main.class}">
      <classpath refid="run.classpath" />
    </java>
</target>
```


Altri Task Importanti

- Attenzione all'attributo fork
  - ⇨se fork="true", viene avviata una nuova macchina virtuale, diversa da quella di Ant
  - ⇨il valore standard è false
- Vantaggio del fork
  - ⇨se l'applicazione va in crash questo non manda in crash anche Ant
- Svantaggio del fork
  - ⇨non è possibile catturare lo standard input

Altri Task Importanti

- Nota
  - ⇨Ant fornisce anche un task exec attraverso cui è possibile eseguire un'applicazione qualsiasi
  - ⇨specificando un comando da eseguire al prompt dei comandi
  - ⇨è però inopportuno utlizzare questo task perchè riduce la portabilità del file di build tra piattaforme diverse

Altri Task Importanti

- Il task manifest
  - ⇨crea un manifesto da utilizzare per il jar

```
<target name="create-manifest" depends="init">
    <manifest file="${build.dir}/MANIFEST.MF" >
      <attribute name="Built-by" value="${user.name}" />
      <attribute name="Main-Class" value="${main.class}" />
    </manifest>
</target>
<target name="jar" depends="rebuild, create-manifest">
    <jar destfile="${dist.dir}/${ant.project.name}.jar"
         manifest="${build.dir}/MANIFEST.MF">
      <fileset dir="${build.dir}" includes="**/*.class" />
      <fileset dir="${utilita.build.dir}" includes="**/*.class" />
    </jar>
</target>
```

Il Task junit

- Il task junit
  - ⇨un task opzionale, ma fondamentale
  - ⇨richiede che junit.jar sia raggiungibile attraverso il classpath della macchina virtuale in cui viene eseguito Ant
- Un modo per risolvere il problema
  - ⇨aggiungere junit.jar alla cartella %ANT_HOME%\lib

Il Task junit

- Attributo fondamentale del task junit
  - ⇨haltonfailure
  - ⇨stabilisce se il processo di build si deve interrompere in caso di fallimento dei test
- Sottoelementi
  - ⇨classpath
  - ⇨test o batchtest
  - ⇨formatter

Il Task junit

- Test e batchtest
  - ⇨test consente di eseguire i test di un'unica classe
  - ⇨batchtest consente di specificare un fileset di test da eseguire (consente di evitare di creare testSuite)
- Formatter
  - ⇨associa all'esecuzione dei test un formato per la produzione dei risultati

Il Task junit

- Il formattatore plain
  - ⇨risultati in formato testuale
- Il formattatore xml
  - ⇨risultati in formato xml
- Nota
  - ⇨per convenzione, il formattatore produce i risultati in un file in una cartella specificata, a meno che non venga specificato l'attributo usefile="false"

Il Task junit

>> ant –f build-semplice.xml
>> test report

```
<target name="test" depends="compile-test">
    <junit haltonfailure="false">
      <classpath refid="test.classpath" />
      <batchtest todir="${test.report.dir}">
        <fileset dir="${test.build.dir}">
          <include name="**/Test*.class" />
        </fileset>
      </batchtest>
      <formatter type="plain" usefile="false" />
      <formatter type="xml" />
    </junit>
</target>
```

in alternativa:

```
<test todir="${test.report.dir}"
      name="${test.build.dir}\**\TestTutto.class" />
```


Struttura del File di Build

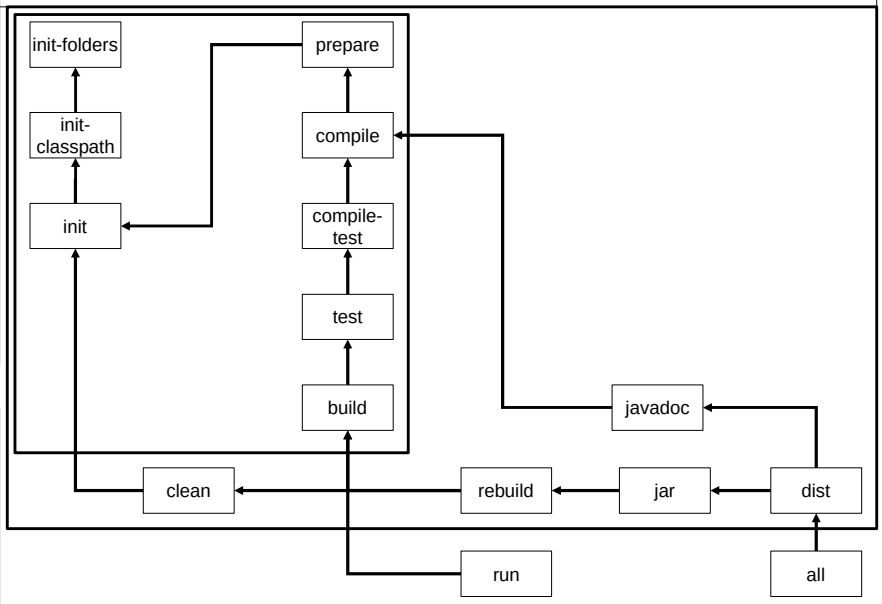
- Il file di build dell'esempio
  - ⇨ha una struttura abbastanza standard
- Il target di default
  - ⇨build
- Il target rebuild
  - ⇨<target name="rebuild" depends="clean, build" />
- Un target nuovo: all
  - ⇨<target name="all" depends="dist" />

Struttura del File di Build

>> build-semplice.xml

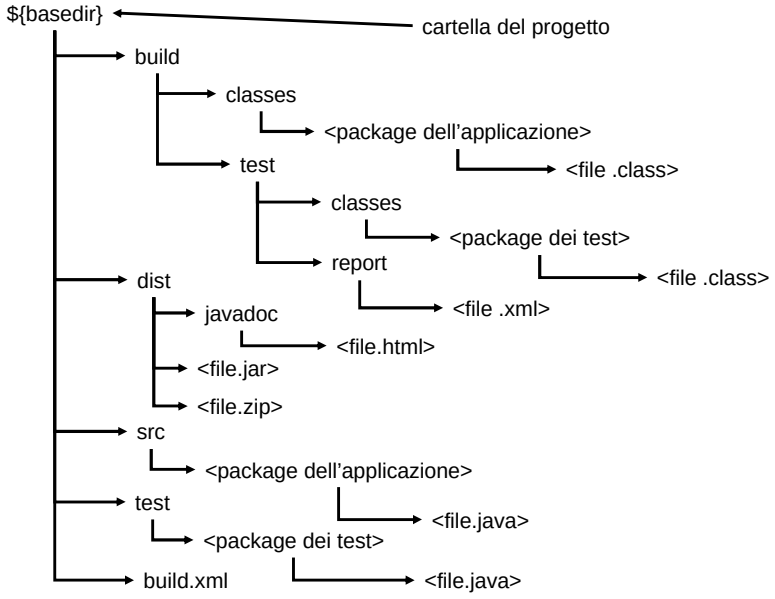
- Gli altri task: ant -projecthelp

```
all           costruisce e produce lo zip distribuibile
init          inizializza le proprieta'
clean         elimina le cartelle riproducibili
prepare       ricrea le cartelle riproducibili
compile       compila il codice sorgente
compile-test  compila i test
test          esegue i test
build         compila ed esegue i test
rebuild       ripulisce e ricostruisce il progetto
jar           produce il jar
javadoc       produce la documentazione
dist          produce lo zip distribuibile
run           esegue l'applicazione
```

Struttura del File di Build

- La struttura della cartella di progetto
  - ⇨rispecchia il principio di tenere separati i componenti di codice
  - ⇨codice sorgente (src)
  - ⇨codice oggetto (build/classes)
  - ⇨codice distribuibile (dist)
  - ⇨codice sorgente dei test (test)
  - ⇨codice oggetto dei test (build/test)



Struttura del File di Build

>> build-semplice.xml

```
<target name="init-folders" depends="">
    <property name="src.dir" location="./src" />
    <property name="test.src.dir" location="./test" />
    <property name="build.dir" location="./build/classes" />
    <property name="test.dir" location="./build/test" />
    <property name="test.build.dir" location="${test.dir}/classes" />
    <property name="test.report.dir" location="${test.dir}/report" />
    <property name="dist.dir" location="./dist" />
    <property name="javadoc.dir" location="${dist.dir}/javadoc" />
    <property name="utilita.src.dir" location="../utilita/src" />
    <property name="utilita.build.dir" location="../utilita/build/classes" />
    <property name="utilita.javadoc.dir" location="../utilita/dist/javadoc" />
    <property name="lib.dir" location="c:/Programmi/lib" />
    <property name="mylib.dir" location="e:/codice/lib" />
</target>
```


Riassumendo

- Proprietà e Percorsi
  - ⇨Proprietà
  - ⇨Percorsi
  - ⇨Fileset
- Altri Task Importanti
  - ⇨Il Task junit
- Struttura del file di build

Termini della Licenza