



Presentazione Ant

a cura di

Marco Pugliese





Agenda

- Cos'è 'Ant'?
- Ant tasks
- MacroDef
- Estendere Ant
- Ant Contrib
- Esempi Ant

Cos'è ANT?





Panoramica

- Apache Jakarta Project
- Open Source, Community Development
 - <http://jakarta.apache.org/ant/>
- Tool Basato su Java
 - Estendibile con l'uso di classi Java
- Configurazione del File 'Build' in XML
 - Facile da leggere e da modificare
- Cross Platform
 - Ma consente anche di eseguire comandi di shell



I vantaggi di ANT

- Scritto IN Java, USANDO Java, and PER Java
- Supporta i Java Tools (javac, javadoc, etc.)
- Il File Build XML è più facile da creare, leggere e mantenere di un file bat/sh.
- ANT è facile da estendere
- Supporta lo sviluppo Cross Platform Java



...i vantaggi di ANT

- Ant è molto più veloce di un MAKE
 - MAKE è basato sulla shell
 - Ogni comando è un nuovo processo
 - Ant viene eseguito nella JVM
 - Ogni com viene eseguito nella JVM
 - Tools come javac sono thread, non processi
 - Con Ant la compilazione di numerosi file sorgenti è molto più veloce
- Le opzioni di debug di Ant sono molto utili
 - '-verbose' e '-debug'



Ant Base

- Ogni 'Project' ha un Build File
 - Il build file di Default si chiama 'build.xml'
 - Si può specificare con l'opzione da linea di comando '-buildfile' un file specifico
- Ogni Build File è composto da Targets
 - 'compile', 'test', 'install', 'clean', etc.
- Ogni Target è composto da Tasks
 - Invoca un comando o un altro programma



...ant Base

- I Targets possono avere delle Dependencies
 - Ad esempio: 'install' può dipendere da 'compile'
 - Si possono gestire le dipendenze a cascata
 - La dipendenza viene eseguita solo se necessario



Ant Tasks





Ant Tasks

- Property Tasks
- Archive tasks
- Compile tasks
- File Tasks
- Execution tasks
- Mail Tasks
- Molti altri



Property Tasks





property

- Consente di settare una property (con name e value insieme), o rendere disponibili delle properties (da un file o risorsa) nel progetto
- Le Properties sono case sensitive
- Properties sono immutabili: chiunque setti per primo il valore per la property la congela per la restante esecuzione del build



property

Imposta la property `prop.welcome` con il valore `"benvenuto"`.

```
<property name="prop.welcome" value="benvenuto"/>
```

Carica un set di properties da un file chiamato `"cant.properties"` che si trova nella stessa directory .

```
<property file="cant.properties"/>
```

carica un set di properties da un indirizzo URL

```
"http://localhost:8080/corsoAnt/props/cant.properties".
```

```
<property url="http://localhost:8080/corsoAnt/props/cant.properties"/>
```



Esempio di file property

```
tomcat.homedir=C:\\Programmi\\sviluppo\\Tomcat 5.5  
tomcat.deploydir=webapps  
jboss.homedir=C:\\Programmi\\sviluppo\\jboss-4.0.2  
jboss.deploydir=server\\default  
jboss.deploy.path=${jboss.homedir}\\${jboss.deploydir}  
tomcat.deploy.path=${tomcat.homedir}\\${tomcat.deploydir}
```



System Properties

<code>java.version</code>	JRE version
<code>java.vendor</code>	JRE vendor
<code>java.vendor.url</code>	Java vendor URL
<code>java.home</code>	Java installation directory
<code>java.vm.specification.version</code>	JVM specification version
<code>java.vm.specification.vendor</code>	JVM specification vendor
<code>java.vm.specification.name</code>	JVM specification name
<code>java.vm.version</code>	JVM implementation version
<code>java.vm.vendor</code>	JVM implementation vendor
<code>java.vm.name</code>	JVM implementation name
<code>java.specification.version</code>	JRE specification version
<code>java.specification.vendor</code>	JRE specification vendor
<code>java.specification.name</code>	JRE specification name



...system Properties

<code>java.class.version</code>	Java class format version number
<code>java.class.path</code>	Java class path
<code>java.ext.dirs</code>	Path of extension directory or directories
<code>os.name</code>	Operating system name
<code>os.arch</code>	Operating system architecture
<code>os.version</code>	Operating system version
<code>file.separator</code>	File separator ("/" on UNIX)
<code>path.separator</code>	Path separator (":" on UNIX)
<code>line.separator</code>	Line separator ("\n" on UNIX)
<code>user.name</code>	User's account name
<code>user.home</code>	User's home directory
<code>user.dir</code>	User's current working directory



Property

esempi\ES01 - properties

Archiving Tasks



Jar

Crea un jar contenente i files di `${build}/classes`. Il jar si chiamerà `app.jar` e verrà inserito nella directory `${dist}/lib`

```
<jar destfile="${dist}/lib/app.jar" basedir="${build}/classes"/>
```

Stessa cosa che nel precedente caso salvo che vengono esclusi i files con il nome `Test.class`.

```
<jar destfile="${dist}/lib/appNoTest.jar"  
  basedir="${build}/classes"  
  excludes="**/Test.class"  
>
```



Jar

esempi\ES02 - jars

war

- Un'estensione del task jar con un trattamento speciale per i file destinati alle WEB-INF/lib, WEB-INF/classes o WEB-INF directories del Web Application Archive
- Il task War è una scorciatoia per creare un layout per il WAR file. La stessa cosa può essere ottenuta con l'uso degli attributi **prefix** e **fullpath** di **zipfilesets** nei task **Zip** o **Jar**.



.war

Supponiamo di avere questa struttura di directories:

terzeparti/libs/**log4j1.jar**

terzeparti/libs/**log4j2.jar**

build/esempio02/jar/pckg1/**Classe.class**

build/esempio02/jar/pckg1/**Test.class**

build/esempio02/jar/pckg2/**Classe.class**

build/esempio02/jar/pckg2/**Test.class**

build/esempio02/jar/pckg3/**Classe.class**

build/esempio02/jar/pckg3/**Test.class**

src/metadata/**myapp.xml**

src/html/myapp/**index.html**

src/jsp/myapp/**front.jsp**

src/graphics/images/gifs/small/**logo.gif**

src/graphics/images/gifs/large/**logo.gif**



..war

Quindi il file war myapp.war viene creato con:

<fileset> specifica i files che andranno nella root (jsp,html)

<lib> specifica il set delle librerie, che andranno in ./WEB-INF/lib

<classes> specifica il set delle classi java, che andranno in ./WEB-INF/classes

<zipfileset> specifica un set di risorse, che andranno in images

```
<war destfile="${dist}/myapp.war" webxml="src/metadata/mymap.xml">
```

```
  <fileset dir="src/html/myapp"/>
```

```
  <fileset dir="src/jsp/myapp"/>
```

```
  <lib dir="terzeparti/libs"/>
```

```
  <classes dir="build"/>
```

```
  <zipfileset dir="src/graphics/images/gifs" prefix="images"/>
```

```
</war>
```



...war

Il war consisterà di:

WEB-INF/web.xml

WEB-INF/liblog4j1.jar

WEB-INF/liblog4j2.jar

WEB-INF/classes/esempio02/jar/pckg1/Classe.class

WEB-INF/classes/esempio02/jar/pckg1/Test.class

WEB-INF/classes/esempio02/jar/pckg2/Classe.class

WEB-INF/classes/esempio02/jar/pckg2/Test.class

WEB-INF/classes/esempio02/jar/pckg3/Classe.class

WEB-INF/classes/esempio02/jar/pckg3/Test.class

META-INF/MANIFEST.MF

index.html

front.jsp

images/small/logo.gif

images/large/logo.gif



....war

Quindi il file war myappExclude.war viene creato con:

- <fileset>** specifica i files che andranno nella root (jsp,html)
- <lib>** specifica il set delle librerie, che andranno in ./WEB-INF/lib
- <exclude>** specifica le librerie da escludere
- <classes>** specifica il set delle classi java, che andranno in ./WEB-INF/classes
- <exclude>** specifica le classi da escludere
- <zipfileset>** specifica un set di risorse, che andranno in images

```
<war destfile="${dist}/myappExclude.war" webxml="src/metadata/mymap.xml">  
  <fileset dir="src/html/myapp"/>  
  <fileset dir="src/jsp/myapp"/>  
  <lib dir="terzeparti/libs">  
    <exclude name="log4j2.jar"/>  
  </lib>  
  <classes dir="build">  
    <exclude name="**/Test.class"/>  
  </classes>  
  <zipfileset dir="src/graphics/images/gifs" prefix="images"/>  
</war>
```



.....war

Il war consisterà di:

WEB-INF/web.xml

WEB-INF/liblog4j1.jar

WEB-INF/classes/esempio02/jar/pckg1/Classe.class

WEB-INF/classes/esempio02/jar/pckg2/Classe.class

WEB-INF/classes/esempio02/jar/pckg3/Classe.class

META-INF/MANIFEST.MF

index.html

front.jsp

images/small/logo.gif

images/large/logo.gif



War

esempi\ES03 - war



Compiling Tasks





javac

Vengono compilate tutte le classi della directory `${src}`, i files compilati vengono messi in `${build}`. Il classpath usato include la libreria `log4j2.jar`, le informazioni di debug sono attivate. Il source level è impostato a 1.4.

```
<javac srcdir="${src}" destdir="${build}"  
      classpath="terzeparti\libs\log4j2.jar"  
      debug="on" source="1.4"  
>
```



javac

Fa la stessa cosa del precedente esempio ma esclude i files Test.java dalla compilazione e include solo i file dei package con nome p*1 e p*3.

```
<javac srcdir="{src}" destdir="{build}"  
    excludes="**/Test.java"  
    includes="**/p*1/*.java,**/p*3/*.java"  
    classpath="terzeparti\\libs\\log4j2.jar"  
    debug="on" source="1.4"
```

```
/>
```



javac

esempi\ES04 - javac



File Tasks





fileset

- Il FileSets rappresenta un gruppo di files
- Questi file vengono selezionati a partire da una directory di base, attraverso diversi PatternSets e Selectors.

.fileset

Seleziona tutti i files contenuti nella directory `${server.src}` purché abbiano l'estensione `.java` e non contengano nel nome "Test"

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

Si può ottenere lo stesso risultato usando il selector "filename".

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



..fileset

Un altro modo per ottenere lo stesso risultato, l'uso dei “patternset” ha però un vantaggio importante, è cioè possibile assegnare un id al patternset in modo tale da poterlo usare anche in altri fileset su altre directories.

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

Come pocanzi detto ecco un fileset che usa lo stesso patternset ma lo applica su una diversa directory: `${client.src}`.

```
<fileset dir="${client.src}" >  
<patternset refid="non.test.sources"/>  
</fileset>
```



copy

- Copia un file o un FileSet in un novo file o in una directory.
- Di default, i files vengono copiati solo se la sorgente del file è diversa dal file destinazione, o quando il file destinazione non esiste.



.copy

Copia una directory in un'altra directory

```
<target name = "simpleCopy" depends = "init">  
  <copy todir = "${dest}">  
    <fileset dir = "${src}" />  
  </copy>  
</target>
```

Copia tutti i files non java nella directory “/dest/dir”

```
<target name = "excludeCopy" depends = "init">  
  <copy todir = "${dest}">  
    <fileset dir = "${src}">  
      <exclude name = "**/*.html" />  
    </fileset>  
  </copy>  
</target>
```



..copy

Copia un set di files in una directory, appendendo al volo l'estensione bak a tutti i files.

```
<target name="renameCopy" depends="init">  
  <copy todir="{bck}">  
    <fileset dir="{src}">  
      <include name="**/*.java" />  
      <include name="**/*.jar" />  
    </fileset>  
    <globmapper from="*" to="*.bak" />  
  </copy>  
</target>
```



delete

Il seguente task cancella il file “/libs/log4j1.jar”

```
<target name="simpleFileDelete" depends="excludeCopy">  
<delete file="${dest}/libs/log4j1.jar" />  
</target>
```

Cancella la directory libs e tutti i files e le subdirectory contenute

```
<target name="simpleDirDelete" depends="excludeCopy">  
<delete dir="${dest}/libs" />  
</target>
```

Cancella tutti i files con estensione .html dalla directory corrente e dalle subdirectory

```
<target name="fileSetDelete" depends="simpleCopy">  
<delete>  
<fileset dir="${dest}" includes="**/*.html" />  
</delete>  
</target>
```



files

esempi\ES05 - files

Execution Tasks





java

- Consente di eseguire una classe nella stessa JVM su cui gira ANT o di forzare l'uso di una JVM diversa.
- Se si verificano dei problemi nell'uso di questo task, settate `fork="true"` per usare una nuova JVM.



.java

Si può lanciare una classe aggiungendo una libreria al classpath

```
<java classname="test.Main">  
  <arg value="-h"/>  
  <classpath>  
    <pathelement location="dist/test.jar"/>  
    <pathelement path="{java.class.path}"/>  
  </classpath>  
</java>
```



exec

- Esegue un comando di shell
- Se viene specificato l'attributo os, il task viene eseguito solo se ant sta girando su uno dei sistemi specificati nel parametro.



.exec

Aggiunge `${basedir}/bin` al PATH di sistema per il comando.

```
<property environment="env"/>  
<exec ... >  
  <env key="PATH" path="${env.PATH}:${basedir}/bin"/>  
</exec>
```

Avvia il `${browser}` con il file specificato `${file}` e chiude il processo ant.

```
<property name="browser" location="C:/Programmi/Internet Explorer/iexplore.exe"/>  
<property name="file" location="ant/docs/manual/index.html"/>  
<exec executable="${browser}" os="Windows 2000" spawn="true">  
  <arg value="${file}"/>  
</exec>
```

Se il parametro “spawn” è abilitato, allora l'applicazione lanciata sopravviverà al task ant. Richiede `fork=true`, e non è compatibile con gli attributi `timeout`, `input`, `output`, `error`, `result`.

dependset

- Il dependset è un task che compara un set di files sorgenti con un set di files *target*. Se almeno uno dei files sorgenti risulta più recente dei files *target*, questi ultimi saranno cancellati.
- I files da confrontare vengono selezionati con i tag annidati filesets e filelists. Sia per i sorgenti che per i *target* deve essere specificato almeno un filelist/fileset.



.dependset

I files HTML che si trovano nella `${output.dir}`, verranno rimossi se almeno una dei seguenti files è più recente:

- 1)il paper.dtd
- 2)il common.dtd
- 3)il common.xsl
- 4)il buildfile

```
<dependset>  
  <srcfilelist dir= "${dtd.dir}" files="paper.dtd,common.dtd"/>  
  <srcfilelist dir= "${xsl.dir}" files="common.xsl"/>  
  <srcfilelist dir= "${basedir}" files="build.xml"/>  
  <targetfileset dir= "${output.dir}" includes="**/*.html"/>  
</dependset>
```



Mail Tasks





mail

Manda un' e-mail da pugliese@zerob.it a bobbuley@gmail.com con soggetto “prova”. La risposta a questa e-mail andrà a bobbuley@tin.it. Ogni file zip che si trovi nella dir dist verrà allegato. Il task cercherà di usare JavaMail.

```
<mail mailhost="smtp.myisp.com" mailport="1025" subject="Prova">  
  <from address="pugliese@zerob.it"/>  
  <replyto address="bobbuley@tin.it"/>  
  <to address="bobbuley@gmail.com"/>  
  <message>The ${buildname} nightly build has completed</message>  
  <fileset dir="dist">  
    <include name="**/*.zip"/>  
  </fileset>  
</mail>
```



Tasks Vari



echo

- Stampa un messaggio sul logger corrente.
- Può essere specificato un level, che gestisce il livello del filtro da applicare ai messaggi.
- Il task può anche loggare su file, in questo caso è possibile specificare se *appendere* o *sovrascrivere* il file.



.echo

Qualche esempio:

```
<echo message="ciao, mondo"/>
```

```
<echo message="Fine linea $$\{line.separator}=\${line.separator}"/>
```

Un messaggio che può apparire solo in modalità -debug.

```
<echo message="Cancello il disco C:" level="debug"/>
```

Genera un file di shell. Il doppio simbolo \$ serve da escape per il carattere \$

```
<echo file="runner.csh" append="false">
```

```
#!/bin/tcsh java-1.3.1 -mx1024m ${project.entrypoint} $$*
```

```
</echo>
```



input

- Abilita l'interazione con l'utente durante il processo di building attraverso l'input da console.
- L'attributo message consente di specificare un messaggio
- L'attributo validargs permette di elencare i possibili valori di input
- Defaultvalue indica il valore di default
- Addproperty è la variabile che conterrà il valore di input dell'utente



MacroDef





MacroDef

- Permette di creare dei template per operazioni complesse
- L'attributo name è il nome con il quale la MacroDef verrà invocata
- I tag annidati disponibili sono:
 - Attribute
 - Element
 - Text
 - Sequential

MacroDef:Attribute

- Esprime un attributo che la MacroDef può avere
- Name è il nome dell'attributo
- L'attributo risulta obbligatorio se non viene specificato un default
- Nel codice della MacroDef, ci si potrà riferire al valore dell'attributo usando la notazione `@{nome_attributo}`. Nella notazione `@@` il primo simbolo `@` ha funzione di escape.

MacroDef:Element

- Rappresenta un elemento (tag) annidato al task che chiama la MacroDef
- name è il nome dell' element
- Optional se vale yes o true rende opzionale l'element in ogni altro caso vale false(default)
- Implicit considera implicito l'elemento vincolandolo a essere presente una sola volta

MacroDef:Esempi

```
<macrodef name="localizedParam">
  <attribute name="nome" />
  <attribute name="language" />
  <attribute name="propertyToSet" />
  <attribute name="propertyFile" />
  <sequential>
    <property name="path" value="@{propertyFile}_@{language}.properties" />
    <property file="{path}" />
    <property name="@{propertyToSet}" value="{@{nome}}" />
    <echo>benvenuto ${@{propertyToSet}}</echo>
  </sequential>
</macrodef>
```

La macro ha quattro parametri obbligatori, usa il codice del *locale* per localizzare il properties giusto, poi setta una **property** con **name** uguale al valore dell'attributo **propertyToSet** per consentire di gestire il valore dal chiamante. Notare le notazioni **`{@{nome}}`** e **`{@{propertyToSet}}`**.



.macroDef: Esempi

```
<import file="myMacro.macrodef" />  
  
<target name="compAttr" depends="datilInsert">  
  <localizedParam nome="{nomeUtente}" language="{local}"  
    propertyToSet="locNome" propertyFile="nomi" />  
  <echo message="{locNome}" />  
</target>
```

Questo codice carica il file `myMacro.macrodef` nel quale si trova la macro presentata prima.

Il target `compAttr` usa la macroDef precedente `localizedParam`. Le properties `{nomeUtente}` e `{local}` vengono settate nel task specificato in `depends`.

MacroDef: Esempio completo

```
<project name="simpleAttr" default="simpleAttr" basedir=". ">
```

```
<import file="myMacro.macrodef" />
```

```
<target name="simpleAttr">
```

```
  <input message="inserisci il nome:" addproperty="nomeUtente" />
```

```
  <simpleAttribute nome="{nomeUtente}" />
```

```
</target>
```

```
<target name="setAttr">
```

```
  <input message="inserisci il nome:" addproperty="nomeUtente" />
```

```
  <setAttribute nome="{nomeUtente}" propertyToSet="result" />
```

```
  <echo>${result}</echo>
```

```
</target>
```

Questa istruzione rende disponibili nel corrente file le macrodef dichiarate in MyMacro.macrodef

MacroDef: Esempio completo

Richiama una MacroDef di nome simpleAttribute e le
passa il parametro `${nomeUtente}`

```
<project name="simpleAttr" default="simpleAttr" basedir=".">
  <import file="myMacro.xml" />
  <target name="simpleAttr">
    <input message="inserisci il nome:" addproperty="nomeUtente" />
    <simpleAttribute nome="${nomeUtente}" />
  </target>
  <target name="setAttr">
    <input message="inserisci il nome:" addproperty="nomeUtente" />
    <setAttribute nome="${nomeUtente}" propertyToSet="result" />
    <echo>${result}</echo>
  </target>
</project>
```

MacroDef: Esempio completo

```
<project name="simpleAttr" default="simpleAttr" basedir=".">
```

```
<import file="myMacro.macrotdef" />
```

```
<target name="simpleAttr">
```

```
<input message="inserisci il nome:" addproperty="nomeUtente" />
```

```
<simpleAttribute nome="{nomeUtente}" />
```

```
</target>
```

```
<target <macrodef name="simpleAttribute">
  <attribute name="nome" default="ANONIMO"/>
  <input message="inserisci il nome:" addproperty="nomeUtente" />
  <sequential>
    <setAttribute name="{nomeUtente}" propertyToSet="result" />
    <length property="nome.length" trim="true" string="@{nome}" />
    <echo>Ciao @{nome}</echo>
  </sequential>
  <echo>Il tuo nome contiene ${nome.length} lettere</echo>
</macrodef>
```

MacroDef: Esempio completo

```
<project name="simpleAttr" default="simpleAttr" basedir=". ">
```

```
<import file="myMacro.macrodéf" />
```

```
<target name="simpleAttr">
```

```
<input message="inserisci il nome:" addproperty="nomeUtente" />
```

```
<simpleAttribute nome="{nomeUtente}" />
```

```
</target>
```

```
<target <macrodef name="simpleAttribute">
  <attribute name="nome" default="ANONIMO"/>
  <input message="inserisci il nome:" addproperty="nomeUtente" />
  <sequential>
    <setAttribute name="{nomeUtente}" propertyToSet="result" />
    <length property="nome.length" trim="true" string="@{nome}" />
    <echo>Ciao @{nome}</echo>
  </sequential>
  <echo>Il tuo nome contiene ${nome.length} lettere</echo>
</target>
```



MacroDef: Esempio completo

```
<target name="datiInsert">  
  <input message="inserisci il nome:" addproperty="nomeUtente"  
    validargs="marco,giovanni,luca,andrea,maria,giovanna" />  
  <input message="inserisci la nazione:" addproperty="local" validargs="it,en,fr" />  
</target>
```

```
<target name="compAttr" depends="datiInsert">  
  <localizedParam nome="{nomeUtente}" language="{local}"  
    propertyToSet="locNome" propertyFile="nomi" />  
  <echo message="{locNome}" />  
</target>
```

```
<target name="conTesto" depends="datiInsert">  
  <textIssue language="{local}" propertyToSet="locNome" propertyFile="nomi">  
    {nomeUtente}  
  </textIssue>  
  <echo message="{locNome}" />  
</target>  
</project>
```

MacroDef: Esempio completo

```
<target name="datiInsert">
```

```
  <input message="inserisci il nome:" addproperty="nomeUtente"
    validargs="marco,giovanni,luca,andrea,maria,giovanna" />
  <input message="inserisci la nazione:" addproperty="local" validargs="it,en,fr" />
```

```
</target>
```

```
<target name="compAttr" depends="datiInsert">
```

```
  <localizedParam nome="${nomeUtente}" language="${local}"
    propertyToSet="locNome" propertyFile="nomi" />
```

```
  <echo message="${locNome}" />
```

```
</target>
```

```
<target name="conT" depends="compAttr">
```

```
  <textIssue language="${local}" propertyToSet="locNome" propertyFile="nomi">
```

```
    ${nomeUtente}
```

```
  </textIssue>
```

```
  <echo message="${locNome}" />
```

```
</target>
```

```
</project>
```

Con i due task INPUT si richiedono all'utente il nome utente e il locale scelti in un range di valori. I valori immessi verranno salvati in due properties, rispettivamente di nome nomeUtente e local.

MacroDef: Esempio completo

```
<target name="datiIns"
  <input message="ins
  <input message="ins
</target>
```

Questo task chiama la MacroDef localizedParam, descritta di seguito, passandole i quattro attributi richiesti.

```
<target name="compAttr" depends="datiInsert">
  <localizedParam nome="{nomeUtente}" language="{local}"
    propertyToSet="locNome" propertyFile="nomi" />
  <echo message="{locNome}" />
</target>
```

```
<target name="conTesto" depends="datiInsert">
  <textIssue language="{local}" propertyToSet="locNome" propertyFile="nomi">
    {nomeUtente}
  </textIssue>
  <echo message="{locNome}" />
</target>
</project>
```

MacroDef: Esempio completo

```
<target name="datiInsert">
```

```
  <input message="inserisci il nome:" addproperty="nomeUtente"  
           validargs="marco,giovanni,luca,andrea,maria,giovanna" />
```

```
  <input message="inserisci la nazione:" addproperty="local" validargs="it,en,fr" />
```

```
</target>
```

```
<target name="compAttr" depends="datiInsert">
```

```
  <localizedParam nome="${nomeUtente}" language="${local}"  
                 propertyToSet="locNome" propertyFile="nomi" />
```

```
  <echo message="${locNome}" />
```

```
</target>
```

Notare che in virtù della dipendenza di questo task da

```
<target name="compAttr" depends="datiInsert">  
  <text message="${local}" propertyToSet="locNome" propertyFile="nomi">
```

```
    ${nomeUtente}
```

```
</text>
```

```
  <echo message="${locNome}" />
```

```
</target>
```

```
</project>
```

MacroDef: Esempio completo

```
<target name="datiInsert">
  <input message="inserisci il nome:" addproperty="nomeUtente"
    validargs="marco,giovanni,luca, andrea, maria, giovanna" />
  <input message="inserisci la nazione:" addproperty="local" validargs="it,en,fr" />
</target>
```

```
<target name="compAttr" depends="datiInsert">
  <localizedParam nome=" ${nomeUtente} " language="${local}"
    propertyToSet="locNome" propertyFile="nomi" />
  <echo message=" ${locNome} " />
</target>
```

```
<target name="compEs" depends="datiInsert">
  <textIssue language="${local}"
    message="nome: ${nomeUtente}, nazione: ${local}"
    propertyFile="nomi" />
  <echo message=" ${locNome} " />
</target>
</project>
```

...è possibile accedere ai valori settati, usando i nomi assegnati nel task precedente.

MacroDef: Esempio completo

```
<target name="datiInsert">  
  <input message="inserisci il nome:" addproperty="nomeUtente"  
    validargs="marco,giovanni,luca,andrea,maria,giovanna" />  
  <input message="inserisci la nazione:" addproperty="local" validargs="it,en,fr" />  
</target>
```

```
<target name="compAttr" depends="datiInsert">  
  <localizedParam nome="${nomeUtente}" language="${local}"  
    propertyToSet="locNome" propertyFile="nomi" />  
  <echo message="${locNome}" />  
</target>
```

...è possibile accedere ai valori settati, usando i nomi assegnati nel task precedente.

```
<target name="compAttr" depends="datiInsert">  
  <text>  
    ${nomeUtente}  
  </text>  
  <echo message="${locNome}" />  
</target>
```



MacroDef: Esempio completo

```
<target name="datiInsert">
  <input message="inserisci il nome:" addproperty="nomeUtente"
           validargs="marco,giovanni,luca,andrea,maria,giovanna" />
  <input message="inserisci la nazione:" addproperty="local" validargs="it,en,fr" />
</target>
```

```
<target name="compAttr" depends="datiInsert">
  <localizedParam nome="${nomeUtente}" language="${local}"
                 propertyToSet="locNome" propertyFile="nomi" />
  <echo message="${locNome}" />
</target>
```

```
<target name="con" depends="compAttr">
  <textIssue language="${local}">
    <attribute name="propertyToSet" />
    <attribute name="propertyFile" />
    <sequential>
      <property name="path" value="@{propertyFile}_@{language}.properties" />
      <property file="${path}" />
      <property name="@{propertyToSet}" value="${@{nome}}" />
      <echo>benvenuto ${@{propertyToSet}}</echo>
    </sequential>
  </textIssue>
</target>
</project>
```

```
<macrodef name="localizedParam">
  <attribute name="nome" default="NN"/>
  <attribute name="language" />
  <attribute name="propertyToSet" />
  <attribute name="propertyFile" />
  <sequential>
    <property name="path" value="@{propertyFile}_@{language}.properties" />
    <property file="${path}" />
    <property name="@{propertyToSet}" value="${@{nome}}" />
    <echo>benvenuto ${@{propertyToSet}}</echo>
  </sequential>
</macrodef>
```

MacroDef: Esempio completo

```

<macrodef name="textIssue">
  <attribute name="language" />
  <target name="dati" />
  <input message="nomeUtente" />
  <input message="nome" />
  <input message="path" />
  </target>
  <target name="completo" depends="dati" />
  <localizedName="nomeUtente" language="{local}" />
  <localizedName="locNome" language="{local}" />
  <echo message="{locNome}" />
</target>
</macrodef>

<target name="conTesto" depends="datiInsert">
  <textIssue language="{local}" propertyToSet="locNome" propertyFile="nomi">
    {nomeUtente}
  </textIssue>
  <echo message="{locNome}" />
</target>
</project>

```

MacroDef: Esempio completo

```

<macrodef name="textIssue">
  <attribute name="language" />
  <target name="dati" >
    <input message="nomeUtente" />
    <input message="nome" />
    <input message="indirizzo" />
  </target>
  <target name="completa" >
    <sequential>
      <property name="path" value="@{propertyFile}_@{language}.properties" />
      <property file="$ {path}" />
      <property name="@{propertyToSet}" value="@{nome}" />
    </sequential>
    <echo>benvenuto $ {@{propertyToSet}}</echo>
  </target>
</macrodef>
<target name="conTesto" depends="datiInsert">
  <textIssue language="$ {local}" propertyToSet="locNome" propertyFile="nomi">
    $ {nomeUtente}
  </textIssue>
  <echo message="$ {locNome}" />
</target>
</project>

```



MacroDef

esempi\ES06 - macroDef



Estendere ant





Estendere Ant

- La classe deve estendere `org.apache.tools.ant.Task` o una sua sottoclasse
- A ciascun attributo deve corrispondere un metodo setter, che rispetti le comuni naming conventions
- Se il task contiene tag annidati deve implementare `org.apache.tools.ant.TaskContainer`.
- Se si vuole supportare del testo nel corspo del tag, si deve implementare il metodo `public void addText(String)`
- Implementare un metodo `public void execute()` che può rilanciare una `BuildException`.



Estendere Ant

Tipi supportati

- String
- boolean, vengono accettati i valori true e yes per “true”, ogni altro vale “false”
- any other primitive type
- `java.io.File`, se il nome rappresenta un path assoluto lo usa, altrimenti prima cerca di risolverlo come path relativo
- `org.apache.tools.ant.types.Path`
- `java.lang.Class`
- `org.apache.tools.ant.types.EnumeratedAttribute`



.estendere Ant

```
package it.bobpuley.antext.stringutils;  
import org.apache.tools.ant.Task;  
import org.apache.tools.ant.taskdefs.Property;  
import org.apache.tools.ant.BuildException;
```

```
public class Capitalize extends Task {  
    private String property;  
    private String input;  
    public void setInput(String input) { this.input = input; }  
    public void setProperty(String property) { this.property = property; }  
    public void execute() throws BuildException {  
        if (this.property == null) {  
            getProject().setNewProperty(property, capitalize(input));  
        }  
    }  
    private String capitalize(String old) {  
        if (old == null || "".equals(old)) return old;  
        char[] tmpArr = old.toCharArray();  
        tmpArr[0] = Character.toUpperCase(tmpArr[0]);  
        return new String(tmpArr);  
    }  
}
```

Ogni “custom task” deve estendere la classe `org.apache.tools.ant.Task`, oppure `org.apache.tools.ant.TaskContainer` se il task ne contiene altri.



.estendere Ant

```
package it.bobpuley.antext.stringutils;  
import org.apache.tools.ant.Task;  
import org.apache.tools.ant.taskdefs.Pr  
import org.apache.tools.ant.BuildExcep
```

```
public class Capitalize extends Task{
```

```
    private String property;
```

```
    private String input;
```

```
    public void setInput(String input){this.input=input;}
```

```
    public void setProperty(String property){this.property=property;}
```

```
    public void execute() throws BuildException{
```

```
        if(this.property==null)throw new BuildException(  
            "you must valorize property attribute!");
```

```
        getProject().setNewProperty(property,capitalize(input));
```

```
    }
```

```
    private String capitalize(String old){
```

```
        if(old == null || "".equals(old))return "";
```

```
        char[] tmpArr = old.toCharArray();
```

```
        tmpArr[0]=Character.toUpperCase(tmpArr[0]);
```

```
        return new String(tmpArr);
```

```
    }
```

```
}
```

e implementare il metodo

```
public void execute()throws BuildException{}
```

che verrà gestito all'interno del ciclo di vita del project ant.



.estendere Ant

```
package it.bobpuley.antext.stringutils;  
import org.apache.tools.ant.Task;  
import org.apache.tools.ant.taskdefs.Property;  
import org.apache.tools.ant.BuildException;
```

```
public class Capitalize extends Task{  
    private String property;  
    private String input;  
    public void setInput(String input){this.input=input;}  
    public void setProperty(String property){this.property=property;}  
    public void execute() throws BuildException{  
        if(this.property==null)throw new BuildException(  
            "you must valorize property attribute!");  
        getProject().setNestedProperty("property.capitalize",  
            getProperty().capitalize(input));  
    }  
    private String capitalize(String old){  
        if(old == null || old.isEmpty())return "";  
        char[] tmpArr = old.toCharArray();  
        tmpArr[0]=Character.toUpperCase(tmpArr[0]);  
        return new String(tmpArr);  
    }  
}
```

Gli attributi del task sono rappresentati da attributi private di classe con il corrispondente setter method.



.estendere Ant

```
package it.bobpuley.antext.stringutils;  
import org.apache.tools.ant.Task;  
import org.apache.tools.ant.taskdefs.Prop  
import org.apache.tools.ant.BuildExcepti
```

```
public class Capitalize extends Task{
```

```
    private String property;
```

```
    private String input;
```

```
    public void setInput(String input){this.input=input;}
```

```
    public void setProperty(String property){this.property=property;}
```

```
    public void execute() throws BuildException{
```

```
        if(this.property==null)throw new BuildException(  
            "you must valorize property attribute!");
```

```
        getProject().setNewProperty(property,capitalize(input));
```

```
    }
```

```
    private String capitalize(String old){
```

```
        if(old == null || "".equals(old))return "";
```

```
        char[] tmpArr = old.toCharArray();
```

```
        tmpArr[0]=Character.toUpperCase(tmpArr[0]);
```

```
        return new String(tmpArr);
```

```
    }
```

```
}
```

Gli attributi del task sono rappresentati da attributi private di classe con il corrispondente setter method. Si può facilmente notificare che l'attributo è required.



.estendere Ant

```
package it.bobpuley.antext.stringutils;  
import org.apache.tools.ant.Task;  
import org.apache.tools.ant.taskdefs.Pr  
import org.apache.tools.ant.BuildExcept
```

```
public class Capitalize extends Task{
```

```
private String property;
```

```
private String input;
```

```
public void setInput(String input){this.input=input;}
```

```
public void setProperty(String property){this.property=property;}
```

```
public void execute() throws BuildException{
```

```
    if(this.property==null)throw new BuildException(
```

```
        "you must valorize property attribute!");
```

```
    getProject().setNewProperty(property,capitalize(input));
```

```
}
```

```
private String capitalize(String old){
```

```
    if(old == null || "".equals(old))return "";
```

```
    char[] tmpArr = old.toCharArray();
```

```
    tmpArr[0]=Character.toUpperCase(tmpArr[0]);
```

```
    return new String(tmpArr);
```

```
}
```

```
}
```

In questo modo si aggiunge al project ant una property che ha per nome il valore dell'attributo property del nostro task. Equivale a scrivere nel build.xml

```
<property name="il valore assegnato all'attributo property"  
...
```



.estendere Ant

```
package it.bobpuley.antext.stringutils;  
import org.apache.tools.ant.Task;  
import org.apache.tools.ant.taskdefs.Pr  
import org.apache.tools.ant.BuildExcep
```

```
public class Capitalize extends Task{
```

```
    private String property;
```

```
    private String input;
```

```
    public void setInput(String input){this.input=input;}
```

```
    public void setProperty(String property){this.property=property;}
```

```
    public void execute() throws BuildException{
```

```
        if(this.property==null)throw new BuildException(
```

```
            "you must valorize property attribute!");
```

```
        getProject().setNewProperty(property,capitalize(input));
```

```
    }
```

```
    private String capitalize(String old){
```

```
        if(old == null || "".equals(old))return "";
```

```
        char[] tmpArr = old.toCharArray();
```

```
        tmpArr[0]=Character.toUpperCase(tmpArr[0]);
```

```
        return new String(tmpArr);
```

```
    }
```

```
}
```

Il valore della property sarà la capitalizzazione dell'attributo input e equivale a

```
<property name="il valore assegnato all'attributo property"  
value="il valore dell'attributo input capitalizzato" />
```

Estendere Ant

```
<antlib>  
  <typedef  
    name="capitalize"  
    classname="it.bobpuley.antext.test.Capitalize"/>  
</antlib>
```

Il file antib.xml si può trovare in un qualsiasi punto del jar contenente la classe Capitalize e definisce il nome completo della classe...

Estendere Ant

```
<antlib>  
  <typedef  
    name="capitalize"  
    classname="it.bobpuley.antext.test.Capitalize"/>  
</antlib>
```

...e il nome con il quale ci si riferirà al task nel file ant (build.xml).

Estendere Ant

```
<project name ="property" default ="test" basedir =".">  
  <taskdef resource="antlib.xml">  
    <classpath>  
      <pathelement location="antExtProj/lib/stringutil.jar" />  
    </classpath>  
  </taskdef>  
  
  <target name="test">  
    <capitalize property="prova" input="testo_da_capitalizzare" />  
    <echo>${prova}</echo>  
  </target>  
</project>
```

Se il file antlib.xml si trova nella root del jar contenente la nostra classe, allora ci si riferirà in questo modo alla risorsa: con il tag `<taskdef` si richiama l'xml...

Estendere Ant

```
<project name ="property" default ="test" basedir =".">
```

```
  <taskdef resource="antlib.xml">
```

```
    <classpath>
```

```
      <pathelement location="antExtProj/lib/stringutil.jar" />
```

```
    </classpath>
```

```
  </taskdef>
```

```
  <target name="test">
```

```
    <capitalize property="prova" input="testo_da_capitalizzare" />
```

```
    <echo>${prova}</echo>
```

```
  </target>
```

```
</project>
```

... CON <classpath><pathelement si indica la posizione relativa sul file system, del jar nella quale root dovrà trovarsi antlib.xml.

Estendere Ant

```
<project name="proper"
  <taskdef resource="antExtProj/lib/stringutil.jar" />
    <classpath>
      <pathelement location="antExtProj/lib/stringutil.jar" />
    </classpath>
  </taskdef>

  <target name="test">
    <capitalize property="prova" input="testo_da_capitalizzare" />
    <echo>${prova}</echo>
  </target>
</project>
```

```
<antlib>
  <typedef name="capitalize"
    classname="it.bobpuley.antext.test.Capitalize" />
</antlib>
```

Estendere Ant

```
public class Capitalize extends Task{
    private String property;
    private String input;
    public void setInput(String input){this.input=input;}
    public void setProperty(String property){this.property=property;}
    public void execute() throws BuildException{
        if(this.property==null)throw new BuildException(
            "you must valorize property attribute!");
        getProject().setNewProperty(property,capitalize(input));
    }
}
<target name="test">
    <capitalize property="prova" input="testo_da_capitalizzare" />
    <echo>${prova}</echo>
</target>
</project>
```

Estendere Ant

```
public class Capitalize extends Task{  
    private String property;  
    private String input;  
    public void setInput(String input){this.input=input;}  
    public void setProperty(String property){this.property=property;}  
    public void execute() throws BuildException{  
        if(this.property==null)throw new BuildException(  
            "you must valorize property attribute!");  
        getProject().setNewProperty(property,capitalize(input));  
    }  
}  
  
<target name="test">  
    <capitalize property="prova" input="testo_da_capitalizzare" />  
    <echo>${prova}</echo>  
</target>  
</project>
```

Estendere Ant

```
public class Capitalize extends Task{  
    private String property;  
    private String input;  
    public void setInput(String input){this.input=input;}  
    public void setProperty(String property){this.property=property;}  
    public void execute() throws BuildException{  
        if(this.property==null)throw new BuildException(  
            "you must valorize property attribute!");  
        getProject().setNewProperty(property,capitalize(input));  
    }  
}
```

```
<target name="test">  
    <capitalize property="prova" input="testo_da_capitalizzare" />  
    <echo>${prova}</echo>  
</target>  
</project>
```



Estendere Ant

esempi\ES07 – AntExt



AntContrib





AntContrib

- Logic Tasks (AntCallback, AntFetch, Assert, Foreach, For, If, Outofdate, RunTarget, Switch, Throw, TimestampSelector, Trycatch)
- Network Tasks(HTTP Post, AntServer / RemoteAnt)
- Performance Monitoring and Tasks
- Platform Tasks
- Property Tasks(Math, Propertycopy, PropertySelector, Pathtofileset, PropertyRegex, SortList, URLEncode, Variable)
- Process Tasks
- Other Tasks



Installazione

Si può installare questa estensione in due modi:

1) Copiare ant-contrib-0.3.jar nella lib directory della tua installazione di ant. Se vuoi usare uno dei task di contrib, includi la seguente riga al tuo progetto.

```
<taskdef resource="net/sf/antcontrib/antcontrib.properties"/>
```

2) Se si vuole caricare contrib da una directory arbitraria, devi esplicitamente indicare il path della libreria (es: /commonslib/ant-contrib-0.3.jar):

```
<taskdef resource="net/sf/antcontrib/antcontrib.properties">  
  <classpath>  
    <pathelement location="/commonslib/ant-contrib-0.3.jar"/>  
  </classpath>  
</taskdef>
```

For

Il task for ha diversi attributi :

- List, la lista dei valori da processare, con un carattere separatore, indicato dall'attributo "delimiter".
 - Param, nome della variabile che conterrà di volta in volta l'elemento corrente.
 - Delimiter, il separatore che divide i tokens nell'attributo "list". Si può usare qualsiasi carattere come separatore, un po' come in StringTokenizer.Default ','
 - Parallel, se impostato a true, tutte le iterazioni del nested <sequential> verranno eseguite in parallelo. Il valore di default è false, che forza l'esecuzione sequenziale dell'iterazione.
 - Keepgoing, se true, tutte le iterazioni chiamate in <sequential> saranno eseguite, anche se uno o più task falliscono. Alla fine, se tutte le iterazioni sono fallite, il task <for> fallirà, diversamente avrà successo. Il valore di default **false**, forza l'esecuzione a interrompersi se almeno un task fallisce.
 - ThreadCount il numero massimo di threads allocabili quando si usa l'esecuzione parallela. Default a 5.
- Trim, se true, ogni spazio sarà rimosso dall'item prima di passarlo al sequential.



.for

Il task seguente mostra come stampare una lista di valori divisi da virgola:

```
<for list="a,b,c,d,e" param="lettera">  
  <sequential>  
    <echo>Lettera @{{lettera}}</echo>  
  </sequential>  
</for>
```

Il task seguente mostra come caricare e ciclare su una lista di files selezionati con filset:

```
<for param="file">  
  <path>  
    <fileset dir="{{test.dir}}/mains" includes="*.java"/>  
  </path>  
  <sequential>  
    <propertyregex override="yes"  
      property="program" input="@{{file}}"  
      regexp=".*(?:[^\.]*)\.java" replace="\1"/>  
  </sequential>  
</for>
```

If

Il task seguente mostra come controllare l'uguaglianza tra due valori:

```
<if>  
  <equals arg1="{foo}" arg2="bar" />  
  <then>  
    <echo message="il valore è bar" />  
  </then>  
  <else>  
    <echo message="il valore non è bar" />  
  </else>  
</if>
```



.if

Il task seguente mostra come usa l'elemento elseif:

```
<if>  
  <equals arg1="{foo}" arg2="bar" />  
  <then>  
    <echo message="The value of property foo is 'bar'" />  
  </then>  
  <elseif>  
    <equals arg1="{foo}" arg2="foo" />  
    <then>  
      <echo message="The value of property foo is 'foo'" />  
    </then>  
  </elseif>  
  <else>  
    <echo message="The value of property foo is not 'foo' or 'bar'" />  
  </else>  
</if>
```



Propertycopy

Il task Propertycopy ha diversi attributi :

- Property, il nome della variabile che conterrà il valore risultante.
- Override, se true consente di sovrascrivere il valore della property.
- From, il nome della proprietà da cui si copia il valore.

L' uso di questo non pone particolari problemi, ma la sua utilità è enorme, dal momento che, in ant ogni property può essere valorizzata una volta sola per ogni processo di build. In pratica in un ciclo for può essere utile per varie ragioni usare una variabile di appoggio, in tal caso l'opzione override è di grande aiuto.



Propertyregex

Il task PropertyRegex ha i seguenti attributi :

- Property, il nome della variabile che conterrà il valore risultante.
- Override, se true consente di sovrascrivere il valore della property.
- Input, la stringa da elaborare.
- Regex l'espressione regolare che verrà applicata all'input string
- Select consente di selezionare un valore di ritorno. Usa la substitution pattern syntax per indicare dove inserire il valore risultante dall'applicazione di una espressione regolare.
- Replace, consente di effettuare una replace con l'uso di regular expression
- Casesensitive ...
- Global in accoppiata con replace, indica che devono essere rimpiazzate tutte le occorrenze, invece che la prima e basta.
- DefaultValue...

AntContrib esempio ⁽¹⁾

esempi/ES08 - antContrib/bobby.macrodef

```
<?xml version="1.0"?>
```

```
<project name="DTO Maker"
```

Il task var, in questo frammento, garantisce che, tra una chiamata e la successiva alla macrodef, ...

```
<macrodef name="getPackages">
```

```
  <attribute name="propertiesPath" />
```

```
  <attribute name="delimiter" default="," />
```

```
  <attribute name="property" />
```

```
  <sequential>
```

```
    <property file="@{propertiesPath}" />
```

```
    <var name="@{property}" unset="true" />
```

```
    <property name="@{property}" value="{packages.nomi}" />
```

```
  </sequential>
```

```
</macrodef>
```

AntContrib esempio ⁽¹⁾

esempi/ES08 - antContrib/bobby.macrodef

```
<?xml version="1.0"?>
```

```
<project name="DTO Maker" default="nothing" basedir=" ">
```

... la property di nome `@{property}` venga
“sovrascritta”.

```
<macrodef name="getPackages">
```

```
  <attribute name="propertiesPath" />
```

```
  <attribute name="delimiter" default="," />
```

```
  <attribute name="property" />
```

```
  <sequential>
```

```
    <property file="@{propertiesPath}" />
```

```
    <var name="@{property}" unset="true" />
```

```
    <property name="@{property}" value="{packages.nomi}" />
```

```
  </sequential>
```

```
</macrodef>
```

AntContrib esempio (1)

esempi/EFacendo in questo modo, creiamo una

property che avrà come nome la stringa che è stata assegnata all'attribute `property` nella chiamata alla macrodef `getPackages`.

```
<?xml version="1.0"?>
```

```
<project name="DTO M
```

```
<macrodef name="getPackages">
```

```
  <attribute name="propertiesPath" />
```

```
  <attribute name="delimiter" default="," />
```

```
  <attribute name="property" />
```

```
  <sequential>
```

```
    <property file="@{propertiesPath}" />
```

```
    <var name="@{property}" unset="true" />
```

```
    <property name="@{property}" value="{packages.nomi}" />
```

```
  </sequential>
```

```
</macrodef>
```

AntContrib esempio (1)

esempi/ES08 - antContrib/esempi/antContrib/bobby.macrodef

file:ES08 - antContrib/esempi/antContrib/bobby.macrodef

```
...
<getPackages propertiesPath="{propertiesFile}"
property="lista.packages" />
```

...
 \${propertieFile} contiene il path del file .properties da

<macrodef name="gcaricare, fatto questo..."

<attribute name="propertiesPath" />

<attribute name="delimiter" default="," />

<attribute name="property" />

<sequential>

<property file="@{propertiesPath}" />

<var name="@{property}" unset="true" />

<property name="@{property}" value="{packages.nomi}" />

</sequential>

</macrodef>

AntContrib esempio (1)

esempi/E...

file:ES08 - antContrib/esempi/antContrib/bobby.macrodef

... ES08 - antContrib/bobby.macrodef

```
<getPackages propertiesPath="{propertiesFile}"  
              property="lista.packages" />
```

```
<?xml version="1.0"?>
```

```
<project name="DTO M...
```

```
...er" default="nothing" basedir=". ">
```

... leggiamo il valore della chiave `{packages.nomi}` e la salviamo in una property di nome `lista.packages`.

```
<macrodef name="getPackages">
```

```
  <attribute name="propertiesPath" />
```

```
  <attribute name="delimiter" default="," />
```

```
  <attribute name="property" />
```

```
  <sequential>
```

```
    <property file="@{propertiesPath}" />
```

```
    <var name="@{property}" unset="true" />
```

```
    <property name="@{property}" value="{packages.nomi}" />
```

```
  </sequential>
```

```
</macrodef>
```

AntContrib esempio (1)

esemp

```
file:ES08 - antContrib/esempi/antContrib/bobby.macrodef
...
<getPackages propertiesPath="{propertiesFile}"
property="lista.packages" />
```

<?xml version="1.0"?> ...
<project name="DTO" ...
nella chiamata passiamo il valore "lista.packages" all' attributo
property

```
<macrodef name="getPackages">
  <attribute name="propertiesPath" />
  <attribute name="delimiter" default="," />
  <attribute name="property" />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="@{property}" unset="true" />
    <property name="@{property}" value="{packages.nomi}" />
  </sequential>
</macrodef>
```

AntContrib esempio (1)

esemp

```
file:ES08 - antContrib/esempi/antContrib/bobby.macrodef
...
<getPackages propertiesPath="{propertiesFile}"
  property="lista.packages" />
```

```
<?xml version="1.0"?>
<project name="DTO"
  useremo questo valore come nome di una nuova property al cui
  valore accederemo, nel project ant, con la sintassi
  ${lista.packages}
```

```
<macrodef name="getPackages">
  <attribute name="propertiesPath" />
  <attribute name="delimiter" default="," />
  <attribute name="property" />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="@{property}" unset="true" />
    <property name="@{property}" value="{packages.nomi}" />
  </sequential>
</macrodef>
```

AntContrib esempio ⁽¹⁾

esempi/ES08 - antContrib/bobby_macrodef

file:ES08 - antContrib/esempi/antContrib/classes.properties

```
<?xml version="1.0"?>
...
packages.nomi=it.bobpuley,it.bobpuley.user,it.bobpuley.prod
...
</project name="DTO" />
```

il valore assegnato sarà quello corrispondente alla chiave
packages.nomi

```
<macrodef name="getPackages">
  <attribute name="propertiesPath" />
  <attribute name="delimiter" default="," />
  <attribute name="property" />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="@{property}" unset="true" />
    <property name="@{property}" value="{packages.nomi}" />
  </sequential>
</macrodef>
```

antContrib esempio (2)

```
<macrodef name="getAttributesByClass">
  <attribute name="package" value="it.bobpuley.user" />
  <attribute name="className" value="Utente" />
  <attribute name="packageName" value="it.bobpuley.user" />
  <attribute name="propertyName" value="nome" />
  <sequential>
    <property file="it.bobpuley.user.Utente.properties" />
    <var name="@{propertyName}" unset="true" />
    <propertyselector property="@{propertyName}" delimiter=","
      match="^{@{packageName}}.@{className}.([\D]*)" select="\1"
      casesensitive="false" />
  </sequential>
</macrodef>
```

Usiamo `propertyselector` per avere una lista di suffissi di chiavi corrispondenti alle chiavi che hanno come prefisso il nome completo della classe (`package.nomeClasse`)

antContrib esempio (2)

```
<macrodef name="getAttributesByClass">
  <attribute name="package" value="it.bobpuley.user" />
  <attribute name="className" value="Utente" />
  <attribute name="propertyName" value="nome" />
  <attribute name="propertyType" value="String" />
  <sequential>
    <property file="it.bobpuley.user" name="it.bobpuley.user.Utente.nome" value="${packageName}.${className}.${propertyName}" type="${propertyType}" />
    <property file="it.bobpuley.user" name="it.bobpuley.user.Utente.cognome" value="${packageName}.${className}.cognome" type="String" />
    <property file="it.bobpuley.user" name="it.bobpuley.user.Utente.gruppo" value="${packageName}.${className}.gruppo" type="Integer" />
    <var name="@{propertyName}" unset="true" />
    <propertyselector property="@{propertyName}" delimiter=","
      match="^@{packageName}\\.@{className}\\.([\\D]*)" select="\1"
      casesensitive="false" />
  </sequential>
</macrodef>
```

Usiamo `propertyselector` per avere una lista di suffissi di chiavi corrispondenti alle chiavi che hanno come prefisso il nome completo della classe (`package.nomeClasse`)

antContrib esempio (2)

```
<macrodef name="getAttributesByClass">
  <attribute name="package" value="it.bobpuley.user" />
  <attribute name="className" value="Utente" />
  <attribute name="packageName" value="it.bobpuley.user" />
  <attribute name="propertyName" value="nome=String" />
  <sequential>
    <property file="it.bobpuley.user.Utente.properties" />
    <property file="it.bobpuley.user.Utente.properties" />
    <var name="@{propertyName}" unset="true" />
    <propertyselector property="@{propertyName}" delimiter=","
      match="^{packageName}.{className}([\D]*)" select="\1"
      casesensitive="false" />
  </sequential>
</macrodef>
```

Usiamo `propertyselector` per avere una lista di suffissi di chiavi corrispondenti alle chiavi che hanno come prefisso il nome completo della classe (`package.nomeClasse`)

antContrib esempio (2)

```
<macrodef name="getAttributesByClass">
  <attribute name="package" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="propertyName" />
  <sequential>
    <property file="pit.bobpuley.user.Utente.properties" name="nome" value="nome" />
    <property file="pit.bobpuley.user.Utente.properties" name="cognome" value="cognome" />
    <property file="pit.bobpuley.user.Utente.properties" name="gruppo" value="gruppo" />
    <var name="@{propertyName}" unset="true" />
    <propertyselector property="@{propertyName}" delimiter=","
      match="^{@{packageName}}.@{className}.([\D]*)" select="\1"
      casesensitive="false" />
  </sequential>
</macrodef>
```

Usiamo `propertyselector` per avere una lista di suffissi di chiavi corrispondenti alle chiavi che hanno come prefisso il nome completo della classe (`package.nomeClasse`)

pit.bobpuley.user.Utente.**nome**=String
pit.bobpuley.user.Utente.**cognome**=String
pit.bobpuley.user.Utente.**gruppo**=Integer

match="^{@{packageName}}.@{className}.([\D]*)" select="\1"

antContrib esempio (2)

```
<macrodef name="getAttributesByClass">
```

```
  <attribute name="{propertiesPath}" />
```

Poiché abbiamo scelto come delimiter la virgola, il risultato che verrà salvato sarà: *“nome,cognome,gruppo”*

```
  <attribute name="{packageName}" />
```

```
  <attribute name="{className}" />
```

```
  <sequential>
```

```
    <property file="@{propertiesPath}" />
```

```
    <var name="@{property}" unset="true" />
```

```
    <propertyselector property="@{property}" delimiter=","  
      match="^@{packageName}.@{className}([\D]*)" select="\1"  
      casesensitive="false" />
```

```
  </sequential>
```

```
</macrodef>
```

antContrib esempio ⁽³⁾

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="$ {lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="$ {lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass propertiesPath="@{propertiesPath}"
              className="@{classe}" property="lista.attributes"
              packageName="@{pack}" />
          </sequential>
        </for>
      </sequential>
    </for>
  </sequential>
</macrodef>
```

MakeBeans è la macrodef centrale dell'esempio, essa richiama le altre passando gli opportuni parametri.

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="$ {lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@ {propertiesPath}"
          property="lista.classi" packageName="@ {pack}" />
        <for list="$ {lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass propertiesPath="@ {propertiesPath}"
              className="@ {classe}" property="lista.attributes"
              packageName="@ {pack}" />
          </sequential>
        </for>
      </sequential>
    </for>
  </sequential>
</macrodef>
```

`propertiesPath` è l'attributo a cui dovremo passare il path del file properties contenente le informazioni sui dto da creare. Nell'esempio sarà: `classes.properties`

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="{ ${lista.packages} " param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="{ ${lista.classi} " param="classe">
          <sequential>
            <getAttributesByClass file:ES08 – antContrib/esempi/antContrib/build.xml
```

```
      cla...Name="@{classe}" property="lista.attributes"
    <makeBeans propertiesPath="{ ${propertiesFiles} "
      srcPath="{ ${sourcesPath} "
      templatePath="dtoTempl" />
    ...
```

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="\${lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="\${lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass file:ES08 – antContrib/esempi/antContrib/build.xml
```

```
      cla...Name="@{classe}" property="lista.attributes"
    <makeBeans propertiesPath="\${propertiesFiles}"
      srcPath="\${sourcesPath}"
      templatePath="dtoTempl" />
    ...
  </for>
</for>
</sequential>
</macrodef>
```

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="{ $ {lista.packages} " param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="{ $ {lista.classi} " param="classe">
          <sequential>
            <getAttributesByClass file:ES08 – antContrib/esempi/antContrib/build.xml
```

```
      cla...Name="@{classe}" property="lista.attributes"
    <makeBeans propertiesPath="{ $ {propertiesFiles} "
      srcPath="{ $ {sourcesPath} "
      templatePath="dtoTempl" />
    ...
  </for>
</for>
</sequential>
</sequential>
</macrodef>
```

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="$ {lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="$ {lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass propertiesPath="@{propertiesPath}"
              className="@{classe}" property="lista.attributes"
              packageName="@{pack}" />
          </sequential>
        </for>
      </sequential>
    </for>
  </sequential>
</macrodef>
```

Per prima cosa invoca la macrodef `getPackages` che restituisce un elenco di packages, separati da virgola, dal file di properties `propertiesPath`

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="$ {lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="$ {lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass propertiesPath="@{propertiesPath}"
              className="@{classe}" property="lista.attributes"
              packageName="@{pack}" />
          </sequential>
        </for>
      </sequential>
    </for>
  </sequential>
</macrodef>
```

Quindi cicla sull'elenco ottenuto, usando la variabile pack per incapsulare i valori singoli della lista.

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="$ {lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="$ {lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass propertiesPath="@{propertiesPath}"
              className="@{classe}" property="lista.attributes"
              packageName="@{pack}" />
          </sequential>
        </for>
      </sequential>
    </for>
  </sequential>
</macrodef>
```

Per ogni package, invoca la macrodef `getClassesByPackage` che a partire dal nome del package carica le classi che ne fanno parte. Renderà disponibili le classi come elenco separato da virgole.

antContrib esempio (3)

```
<macrodef name="makeBeans">
  <attribute name="propertiesPath" />
  <attribute name="srcPath" default="src"/>
  <attribute name="templatePath" default="." />
  <sequential>
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />
    <for list="$ {lista.packages}" param="pack">
      <sequential>
        <getClassesByPackage propertiesPath="@{propertiesPath}"
          property="lista.classi" packageName="@{pack}" />
        <for list="$ {lista.classi}" param="classe">
          <sequential>
            <getAttributesByClass propertiesPath="@{propertiesPath}"
              className="@{classe}" property="lista.attributes"
              packageName="@{pack}" />
          </sequential>
        </for>
      </sequential>
    </for>
  </sequential>
</macrodef>
```

Passiamo quindi a ciclare sull'elenco delle classi del package corrente (@{pack})

antContrib esempio (3)

```
<macrodef name="makeBeans"  
  <attribute name="propertiesPath" default="." />  
  <attribute name="srcPath" default="." />  
  <attribute name="templatePath" default="." />  
  <sequential>  
    <getPackages propertiesPath="@{propertiesPath}" property="lista.packages" />  
    <for list="$ {lista.packages}" param="pack">  
      <sequential>  
        <getClassesByPackage propertiesPath="@{propertiesPath}"  
          property="lista.classi" packageName="@{pack}" />  
        <for list="$ {lista.classi}" param="classe">  
          <sequential>  
            <getAttributesByClass propertiesPath="@{propertiesPath}"  
              className="@{classe}" property="lista.attributes"  
              packageName="@{pack}" />  
          </sequential>  
        </for>  
      </sequential>  
    </for>  
  </sequential>  
</macrodef>
```

Per ogni classe trovata, attraverso la macrodef `getAttributesByClass`, carichiamo l'elenco degli attributi.

antContrib esempio (4)

```
<makeBean srcPath="@{srcPath}" propertiesPath="@{propertiesPath}"  
  className="@{classe}" packageName="@{pack}"  
  attributeList="$ {lista.attributes}" templatePath="@{templatePath}" />
```

```
</sequential>  
</for>  
</sequential>  
</for>  
</sequential>  
</macrodef>
```

A questo punto abbiamo tutte le informazioni necessarie per chiamare la macrodef `makeBean` che creerà la classe richiesta.

antContrib esempio (4)

```
<makeBean srcPath="@{srcPath}" propertiesPath="@{propertiesPath}"  
  className="@{classe}" packageName="@{pack}"  
  attributeList="${lista.attributes}" templatePath="@{templatePath}" />
```

```
</sequential>  
</for>  
</sequential>  
</for>  
</sequential>  
</macrodef>
```

Chiude il ciclo che itera sull'elenco delle classi del package corrente.

antContrib esempio (4)

```
<makeBean srcPath="@{srcPath}" propertiesPath="@{propertiesPath}"  
  className="@{classe}" packageName="@{pack}"  
  attributeList="${lista.attributes}" templatePath="@{templatePath}" />
```

```
</sequential>  
</for>  
</sequential>  
</for>  
</sequential>  
</macrodef>
```

Chiude il ciclo che itera sull'elenco dei package.

antContrib esempio (5)

```
<macrodef name="makeBean">
```

```
<attribute name="propertiesPath" />
```

```
<attribute name="className" />
```

```
<attribute name="packageName" />
```

```
<attribute name="attributeList" />
```

```
<attribute name="srcPath" default="src" />
```

```
<attribute name="templatePath" default="." />
```

```
<sequential>
```

```
<property file="@{propertiesPath}" />
```

```
<var name="filePath" unset="true" />
```

```
<propertyregex property="filePath" input="@{packageName}"
```

```
  regexp="([a-z])" replace="/" casesensitive="false" />
```

Infine la macroDef principale `makeBean` che ha la funzione di creare i files java conformemente ai dati ricevuti dal chiamante. Ricordate che il chiamante nell'esempio è `makeBeans`.

antContrib esempio (5)

```
<macrodef name="makeBean">  
  <attribute name="propertiesPath" />  
  <attribute name="className" />  
  <attribute name="packageName" />  
  <attribute name="attributeList" />  
  <attribute name="srcPath" default="src" />  
  <attribute name="templatePath" default="template" />  
  <sequential>  
    <property file="@{propertiesPath}" />  
    <var name="filePath" unset="true" />  
    <propertyregex property="filePath" input="@{packageName}"  
      regexp="([\^a-z]" replace="/" casesensitive="false" />
```

L'attributo `propertiesPath` deve contenere il path del file di properties contenente le info sulle classi da creare, Nell'esempio `classes.properties`.

antContrib esempio (5)

```
<macrodef name="makeBean">
  <attribute name="propertiesPath" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="attributeList" />
  <attribute name="srcPath" default="src" />
  <attribute name="templatePath" default="." />
  <sequential>
    <property file="@{propertiesPath}/@{className}.properties" />
    <var name="filePath" unset="true" />
    <propertyregex property="filePath" input="@{packageName}"
      regexp="([\^a-z])" replace="/" casesensitive="false" />
  </sequential>
</macrodef>
```

L'attributo `className` deve contenere il nome della classe da creare.

antContrib esempio (5)

```
<macrodef name="makeBean">
  <attribute name="propertiesPath" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="attributeList" />
  <attribute name="srcPath" default="src" />
  <attribute name="templatePath" default="" />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="filePath" unset="true" />
    <propertyregex property="filePath" input="@{packageName}"
      regexp="([^a-z])" replace="/" casesensitive="false" />
```

L'attributo packageName deve contenere il nome del package della classe da creare.

antContrib esempio (5)

```
<macrodef name="makeBean">
  <attribute name="propertiesPath" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="attributeList" />
  <attribute name="srcPath" default="src" />
  <attribute name="templatePath" default="." />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="filePath" unset="true" />
    <propertyregex property="filePath" file="<var name="filePath" unset="true" />
```

L'attributo `attributeList` deve contenere l'elenco dei nomi degli attributi della classe che si vuole creare.

antContrib esempio (5)

```
<macrodef name="makeBean">
  <attribute name="propertiesPath" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="attributeList" />
  <attribute name="srcPath" default="src" />
  <attribute name="templatePath" default="." />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="filePath" unset="true" />
    <propertyregex property="filePath"
      regexp="
```

L'attributo `srcPath` rappresenta la directory nella quale verranno creati i files `.java`. Se non specificata dal chiamante varrà `src`.

antContrib esempio (5)

```
<macrodef name="makeBean">
  <attribute name="propertiesPath" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="attributeList" />
  <attribute name="srcPath" default="src" />
  <attribute name="templatePath" default="." />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="filePath" unset="true" />
    <propertyregex property="filePath"
      regex="<math>^</math>L'attributo templatePath rappresenta la
      directory nella quale la macrodef cercherà i
      template per generare i sorgenti.
    </propertyregex>
  </sequential>
</macrodef>
```

antContrib esempio (5)

```
<macrodef name="makeBean">
  <attribute name="propertiesPath" />
  <attribute name="className" />
  <attribute name="packageName" />
  <attribute name="attributeList" />
  <attribute name="srcPath" default="src" />
  <attribute name="templatePath" default="." />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="filePath" unset="true" />
```

Covertite il nome del path sostituendo i . con /.
es: prova.conv.test -> prova/conv/test

```
<propertyregex property="filePath" input="@{packageName}"
  regexp="([^a-z])" replace="/" casesensitive="false" />
```

antContrib esempio (6)

```
<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/{filePath}/{fileName}" />
<copy tofile="@{srcPath}/{filePath}/{fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmpl" />
  <filterset begintoken="bobpuley.replace("" endtoken="")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="$ {extends.@{packageName}.@{className}} " />
  </filterset>
</copy>
```

Crea la prima porzione di codice della classe comprendente l'intestazione e la dichiarazione della classe.

```
<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/${filePath}/${fileName}" />
<copy tofile="@{srcPath}/${filePath}/${fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmplt" />
  <filterset begintoken="bobbuley.replace(" endtoken=")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="\${extends @\{packageName\} @\{className\}}" />
  </filterset>
</copy>
```

```
file:ES08 - antContrib/esempi/antContrib/dtoTempl/prefix.tmplt
package bobbuley.replace('CUSTOM_PACK');

import java.io.Serializable;

public class bobbuley.replace('NOME_CLASSE')
extends bobbuley.replace('EXTENDS') implements
Serializable{
```

antContrib esempio (6)

```
<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/${filePath}/${fileName}" />
<copy tofile="@{srcPath}/${filePath}/${fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmplt" />
  <filterset begintoken="bobbuley.replace(" endtoken=")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="\${extends @{packageName} @{className}}"/>
  </filterset>
</copy>
package bobbuley.replace('CUSTOM_PACK');

import java.io.Serializable;

public class bobbuley.replace('NOME_CLASSE')
  extends bobbuley.replace('EXTENDS') implements
  Serializable{
```

antContrib esempio (6)

```
<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/${filePath}/${fileName}" />
<copy tofile="@{srcPath}/${filePath}/${fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmpl" />
  <filterset begintoken="bobbuley.replace(" endtoken=")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="\$ extends @{packageName} @{className}" />
  </filterset>
</copy>
```

file:ES08 - antContrib/esempi/antContrib/dtoTempl/prefix.tmpl

```
package bobbuley.replace('CUSTOM_PACK');

import java.io.Serializable;

public class bobbuley.replace('NOME_CLASSE')
  extends bobbuley.replace('EXTENDS') implements
  Serializable{
```

antContrib esempio (6)

```
<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/{filePath}/{fileName}" />
<copy tofile="@{srcPath}/{filePath}/{fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmpl" />
  <filterset begintoken="bobbuley.replace(" endtoken=")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="\${extends @{packageName} @{className}}"/>
  </filterset>
</copy>
```

file:ES08 - antContrib/esempi/antContrib/dtoTempl/prefix.tmpl

```
package bobbuley.replace('CUSTOM_PACK');

import java.io.Serializable;

public class bobbuley.replace('NOME_CLASSE')
  extends bobbuley.replace('EXTENDS') implements
  Serializable{
```

antContrib esempio (6)

```

<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/{filePath}/{fileName}" />
<copy tofile="@{srcPath}/{filePath}/{fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmpl" />
  <filterset begintoken="bobbuley.replace(" endtoken=")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="\$ extends @{packageName} @{className}" />
  </filterset>
</copy>

```

file:ES08 - antContrib/esempi/antContrib/dtoTempl/prefix.tmpl

```

package bobbuley.replace('CUSTOM_PACK');

import java.io.Serializable;

public class bobbuley.replace('NOME_CLASSE')
extends bobbuley.replace('EXTENDS') implements
Serializable{

```

antContrib esempio (6)

```
<var name="fileName" unset="true" />
<property name="fileName" value="@{className}.java" />
<delete file="@{srcPath}/${filePath}/${fileName}" />
<copy tofile="@{srcPath}/${filePath}/${fileName}" overwrite="true">
  <fileset dir="@{templatePath}" includes="prefix.tmplt" />
  <filterset begintoken="bobbuley.replace(" endtoken=")">>
    <filter token="CUSTOM_PACK" value="@{packageName}" />
    <filter token="NOME_CLASSE" value="@{className}" />
    <filter token="EXTENDS"
      value="\$ extends @{packageName} @{className}" />
  </filterset>
</copy>
```

```
file:ES08 - antContrib/esempi/antContrib/dtoTempl/prefix.tmplt
package bobbuley.replace('CUSTOM_PACK');

import java.io.Serializable;

public class bobbuley.replace('NOME_CLASSE')
  extends bobbuley.replace('EXTENDS') implements
  Serializable{
```

antContrib esempio (7)

```
<for list="@{attributeList}" param="attributo">
  <sequential>
    <concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">
      <fileset dir="@{templatePath}" includes="dichiarazioni.tmplt" />
      <filterchain>
        <replacetokens>
          <token key="TIPO_PROP"
            value="{@packageName}.{className}.{attributo}" />
          <token key="NOME_PROP" value="@{attributo}" />
        </replacetokens>
      </filterchain>
    </concat>
```

Quindi si itera sulla lista di attributi della classe.

antContrib esempio ⁽⁷⁾

```
<for list="@{attributeList}" param="attributo">
  <sequential>
    <concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">
      <fileset dir="@{templatePath}" includes="dichiarazioni.tmplt" />
      <filterchain>
        <replacetokens>
          <token key="TIPO_PROP"
            value="\${@{packageName}}.{@{className}}.{@{attributo}}" />
          <token key="NOME_PROP" value="@{attributo}" />
        </replacetokens>
      </filterchain>
    </concat>
```

Questa porzione di codice crea il codice relativo alle dichiarazioni degli attributi della classe.

antContrib esempio (7)

```
<for list="@{attributeList}" param="attributo">
  <sequential>
    <concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">
      <fileset dir="@{templatePath}" includes="dichiarazioni.tmplt" />
      <filterchain>
        <replacetokens>
          <token key="TIPO_PROP"
            value="\${@{packageName}}.{@{className}}.{@{attributo}}" />
          <token key="NOME_PROP" value="@{attributo}" />
        </replacetokens>
      </filterchain>
    </concat>
```

File:
ES08 - antContrib/esempi/antContrib/dtoTempl/dichiarazioni.tmplt

```
private @TIPO_PROP@ @NOME_PROP@;
```

antContrib esempio (7)

```
<for list="@{attributeList}" param="attributo">
  <sequential>
    <concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">
      <fileset dir="@{templatePath}" includes="dichiarazioni.tmplt" />
      <filterchain>
        <replacetokens>
          <token key="TIPO_PROP"
            value="\${@{packageName}.\@{className}.\@{attributo}}" />
          <token key="NOME_PROP" value="@{attributo}" />
        </replacetokens>
      </filterchain>
    </concat>
```

File:
ES08 - antContrib/esempi/antContrib/dtoTempl/dichiarazioni.tmplt

```
private @TIPO_PROP@ @NOME_PROP@;
```

Se non specificati, il begintoken e l'endtoken, saranno @.

antContrib esempio (7)

```
<for list="@{attributeList}" param="attributo">
  <sequential>
    <concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">
      <fileset dir="@{templatePath}" includes="dichiarazioni.tmplt" />
      <filterchain>
        <replacetokens>
          <token key="TIPO_PROP"
            value="\${@{packageName}}.{@{className}}.{@{attributo}}" />
          <token key="NOME_PROP" value="@{attributo}" />
        </replacetokens>
      </filterchain>
    </concat>
```

File:
ES08 - antContrib/esempi/antContrib/dtoTempl/dichiarazioni.tmplt

```
private @TIPO_PROP@ @NOME_PROP@;
```

Se non specificati, il begintoken e l'endtoken, saranno @.

antContrib esempio (8)

```

<capitalize input="@{attributo}" property="attributeCap"/>
<concat destFile="@{srcPath}/{filePath}/{fileName}.tmp" append="true">
  <fileset dir="@{templatePath}" includes="body.tmpl" />
  <filterchain>
    <replacetokens>
      <token key="TIPO_PROP"
        value="\${@{packageName}}.{@{className}}.{@{attributo}}" />
      <token key="CNAME_PROP" value="\${attributeCap}" />
      <token key="NO" value="@{attributo}" />
    </replacetokens>
  </filterchain>
</concat>
</sequential>
</for>

```

File:
ES08 - antContrib/esempi/antContrib/dtoTempl/body.tmpl

```

public void set@CNAME_PROP@ (@TIPO_PROP@ @CNAME_PROP@) {
    this.@CNAME_PROP@=@CNAME_PROP@;
}

public @TIPO_PROP@ get@CNAME_PROP@() {
    return this.@CNAME_PROP@;
}

```

antContrib esempio (8)

```
<capitalize input="@{attributo}" property="attributeCap"/>
<concat destFile="@{srcPath}/${filePath}/${fileName}.tmp" append="true">
  <fileset dir="@{templatePath}" includes="body.tmplt" />
  <filterchain>
    <replacetokens>
      <token key="TIPO_PROP"
        value="\${@{packageName}.\@{className}.\@{attributo}}"/>
      <token key="CNOME_PROP" value="\${attributeCap}"/>
      <token key="NOME_PROP" value="@{attributo}"/>
    </replacetokens>
  </filterchain>
</concat>
</sequential>
</for>
```

Qui si chiude il ciclo per la creazione degli attributi e degli accessor methods.

antContrib esempio (9)

```
<concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">  
  <fileset dir="@{srcPath}" includes="{filePath}/{fileName}.tmp" />  
</concat>
```

```
<delete file="@{srcPath}/{filePath}/{fileName}.tmp" />  
<concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">  
  <fileset dir="@{templatePath}" includes="suffix.tmplt" />  
</concat>
```

```
</sequential>  
</macrodef>
```

Infine si assemblano i vari frammenti di codice prodotti,

antContrib esempio (9)

```
<concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">  
  <fileset dir="@{srcPath}" includes="{filePath}/{fileName}.tmp" />  
</concat>
```

```
<delete file="@{srcPath}/{filePath}/{fileName}.tmp" />
```

```
<concat destFile="@{srcPath}/{filePath}/{fileName}" append="true">  
  <fileset dir="@{templatePath}" includes="suffix.tmplt" />  
</concat>
```

```
</sequential>
```

```
</macrodef>
```

E si chiude il codice usando il template:
ES08 – antContrib/esempi/antContrib/dtoTemp/suffix.tmplt
che consiste unicamente della graffa di chiusura della classe.

antContrib esempio (10)

```
<macrodef name="getClassesByPackage">
  <attribute name="propertiesPath" />
  <attribute name="packageName" />
  <attribute name="property" />
  <sequential>
    <property file="@{propertiesPath}" />
    <var name="@{property}" unset="true" />
    <property name="@{property}" value="\${classes.@{packageName}.nomi}"
  />
  </sequential>
</macrodef>
```

la macroDef `getClassesByPackage` ha il compito, a partire dal nome di un package, di reperire tutti i nomi delle classi che ne fanno parte, e organizzarli in una lista.

antContrib esempio (11)

esempi/ES07 - antContrib/build.xml

```
<project name="inizializzaAmbiente" default="makeB" basedir=".">
```

```
<import file="bobby.macrodef" />
```

```
<property name="propertiesFile" value="classes.properties" />
```

```
<property name="sourcesPath" value="sources" />
```

```
<property name="buildPath" value="build" />
```

```
<property name="classesPath" value="classes" />
```

```
<property name="jarsPath" value="lib" />
```

```
<property name="projectName" value="store_dto" />
```

```
<taskdef resource="net/sf/antcontrib" />
```

```
<taskdef resource="it/bobpuley/antlib.xml" />
```

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

```
  <delete dir="{buildPath}/**" />
```

```
</target>
```

Il file di properties contenente le info sui DTO da creare.

antContrib esempio (11)

esempi/ES07 - antContrib/build.xml

```
<project name="inizializzaAmbiente" default="makeB" basedir=".">
```

```
  <import file="bobby.macrodef" />
```

```
  <property name="propertiesFile" value="classes.properties" />
```

```
  <property name="sourcesPath" value="sources" />
```

```
  <property name="buildPath" value="build" />
```

```
  <property name="classesPath" value="classes" />
```

```
  <property name="jarsPath" value="lib" />
```

```
  <property name="projectName" value="store_dto" />
```

```
  <taskdef resource="net/sf/antcontrib" class="org.apache.tools.ant.taskdefs.optional.AntContribTask" />
```

```
  <taskdef resource="it/bobpuley/antcontrib" class="org.apache.tools.ant.taskdefs.optional.AntContribTask" />
```

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

```
    <delete dir="{buildPath}/**" />
```

```
  </target>
```

La cartella nella quale verranno salvati i sorgenti generati.

antContrib esempio (11)

esempi/ES07 - antContrib/build.xml

```
<project name="inizializzaAmbiente" default="makeB" basedir=".">
```

```
<import file="bobby.macrodef" />
```

```
<property name="propertiesFile" value="classes.properties" />
```

```
<property name="sourcesPath" value="sources" />
```

```
<property name="buildPath" value="build" />
```

```
<property name="classesPath" value="classes" />
```

```
<property name="jarsPath" value="lib" />
```

```
<property name="projectName" value="store_dto" />
```

```
<taskdef resource="net/sf/antcontrib" />
```

```
<taskdef resource="it/bobpuley/antcontrib" />
```

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

```
  <delete dir="{buildPath}/**" />
```

```
</target>
```

La directory target base per i files compilati e le library create.

antContrib esempio (11)

esempi/ES07 - antContrib/build.xml

```
<project name="inizializzaAmbiente" default="makeB" basedir=".">
```

```
<import file="bobby.macrodef" />
```

```
<property name="propertiesFile" value="classes.properties" />
```

```
<property name="sourcesPath" value="sources" />
```

```
<property name="buildPath" value="build" />
```

```
<property name="classesPath" value="classes" />
```

```
<property name="jarsPath" value="lib" />
```

```
<property name="projectName" value="store_dto" />
```

```
<taskdef resource="net/sf/antcontrib" />
```

```
<taskdef resource="it/bobpuley/antlib.xml" />
```

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

```
  <delete dir="{buildPath}/**" />
```

```
</target>
```

La directory delle classi compilate.

antContrib esempio (11)

esempi/ES07 - antContrib/build.xml

```
<project name="inizializzaAmbiente" default="makeB" basedir=".">
```

```
  <import file="bobby.macrodef" />
```

```
  <property name="propertiesFile" value="classes.properties" />
```

```
  <property name="sourcesPath" value="sources" />
```

```
  <property name="buildPath" value="build" />
```

```
  <property name="classesPath" value="classes" />
```

```
  <property name="jarsPath" value="lib" />
```

```
  <property name="projectName" value="store_dto" />
```

```
  <taskdef resource="net/sf/antcontrib/antlib.xml" />
```

```
  <taskdef resource="it/bobpuley/antcontrib/test/antlib.xml" />
```

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

```
    <delete dir="{buildPath}/**" />
```

```
  </target>
```

La directory del java archive finale.

antContrib esempio (11)

esempi/ES07 - antContrib/build.xml

```
<project name="inizializzaAmbiente" default="makeB" basedir=".">
```

```
<import file="bobby.macrodef" />
```

```
<property name="propertiesFile" value="classes.properties" />
```

```
<property name="sourcesPath" value="sources" />
```

```
<property name="buildPath" value="build" />
```

```
<property name="classesPath" value="classes" />
```

```
<property name="jarsPath" value="lib" />
```

```
<property name="projectName" value="store_dto" />
```

```
<taskdef resource="net/sf/antcontrib/antlib.xml" />
```

```
<taskdef resource="it/bobpuley/antext/test/antlib.xml" />
```

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

```
  <delete dir="{buildPath}/**" />
```

```
</target>
```

Il nome del jar.

antContrib esempio (12)

```
<target name="makeB" depends="clean">
```

```
<makeBeans propertiesPath="${propertiesFile}"  
            srcPath="${sourcesPath}" templatePath="dtoTempl"/>
```

```
<mkdir dir="${buildPath}/${jarsPath}" />
```

```
<mkdir dir="${buildPath}/${classesPath}" />
```

```
<javac srcdir="${sourcesPath}" destdir="${buildPath}/${classesPath}"  
        debug="on" source="1.5" />
```

```
<jar destfile="${buildPath}/${jarsPath}/${projectName}.jar"  
      basedir="${buildPath}/${classesPath}" />
```

```
</target>
```

```
</project>
```

Chiamata la macrodef che si occupa di generare i sorgenti java a partire dal file properties.

antContrib esempio (12)

```
<target name="makeB" depends="clean">
```

```
  <makeBeans propertiesPath="{propertiesFile}"  
    srcPath="{sourcesPath}" templatePath="dtoTempl"/>
```

```
  <mkdir dir="{buildPath}/{jarsPath}" />
```

```
  <mkdir dir="{buildPath}/{classesPath}" />
```

```
  <javac srcdir="{sourcesPath}" destdir="{buildPath}/{classesPath}"  
    debug="on" source="1.5" />
```

```
  <jar destfile="{buildPath}/{jarsPath}/{projectName}.jar"  
    basedir="{buildPath}/{classesPath}" />
```

```
</target>
```

```
</project>
```

Compila i sorgenti.

antContrib esempio (12)

```
<target name="makeB" depends="clean">
```

```
  <makeBeans propertiesPath="${propertiesFile}"  
    srcPath="${sourcesPath}" templatePath="dtoTempl"/>
```

```
  <mkdir dir="${buildPath}/${jarsPath}" />
```

```
  <mkdir dir="${buildPath}/${classesPath}" />
```

```
  <javac srcdir="${sourcesPath}" destdir="${buildPath}/${classesPath}"  
    debug="on" source="1.5" />
```

```
  <jar destfile="${buildPath}/${jarsPath}/${projectName}.jar"  
    basedir="${buildPath}/${classesPath}" />
```

```
</target>
```

```
</project>
```



Archivia le classi in un jar file.



antContrib esempio (13)

esempi/ES07 - antContrib/classes.properties

```
packages.nomi=it.bobpuley,it.bobpuley.user,it.bobpuley.prod
```

```
classes.it.bobpuley.nomi=BaseDto,BaseProduct
```

```
extends.it.bobpuley.BaseDto=Object  
it.bobpuley.BaseDto.idApp=Integer
```

```
extends.it.bobpuley.BaseProduct=it.bobpuley.BaseDto  
it.bobpuley.BaseProduct.idProduct=Long  
it.bobpuley.BaseProduct.prezzo=Double
```

```
classes.it.bobpuley.user.nomi=Utente,Gruppo
```

```
extends.it.bobpuley.user.Utente=it.bobpuley.BaseDto  
it.bobpuley.user.Utente.nome=String  
it.bobpuley.user.Utente.cognome=String  
it.bobpuley.user.Utente.gruppo=Integer
```



antContrib esempio (14)

```
extends.it.bobpuley.user.Gruppo=it.bobpuley.BaseDto  
it.bobpuley.user.Gruppo.idGruppo=Integer  
it.bobpuley.user.Gruppo.descrizione=String
```

```
classes.it.bobpuley.prod.nomi=Cd,Libro,Dvd
```

```
extends.it.bobpuley.prod.Cd=it.bobpuley.BaseProduct  
it.bobpuley.prod.Cd.titolo=Integer  
it.bobpuley.prod.Cd.autore=String  
it.bobpuley.prod.Cd.CasaDiscografica=String
```

```
extends.it.bobpuley.prod.Libro=it.bobpuley.BaseProduct  
it.bobpuley.prod.Libro.titolo=Integer  
it.bobpuley.prod.Libro.autore=String  
it.bobpuley.prod.Libro.casaEditrice=String
```

```
extends.it.bobpuley.prod.Dvd=it.bobpuley.BaseProduct  
it.bobpuley.prod.Dvd.titolo=Integer  
it.bobpuley.prod.Dvd.regista=String  
it.bobpuley.prod.Dvd.attori=java.util.Collection
```



antContrib esempio (15)

esempi/ES07 - antContrib/dtoTempl/*.tmplt

prefix.tmplt:

```
package bobbuley.replace('CUSTOM_PACK');
```

```
import java.io.Serializable;
```

```
public class bobbuley.replace('NOME_CLASSE') extends  
bobbuley.replace('EXTENDS') implements Serializable{
```

dichiarazioni.tmplt:

```
private @TIPO_PROP@ @NOME_PROP@;
```



antContrib esempio (16)

body.tmplt:

```
public void set@CNAME_PROP@ (@TIPO_PROP@ @NOME_PROP@) {
    this.@NOME_PROP@=@NOME_PROP@;
}

public @TIPO_PROP@ get@CNAME_PROP@(){
    return this.@NOME_PROP@;
}
```

suffix.tmplt:

```
}
```



Ant Contrib

esempi\ES08 - antContrib