Apache Ant 1.5.3 Manual

Version 0.1

Andrew Peter Marlow

June 29, 2003

This is the manual for version 1.5.3 of Apache Ant. If your version of Ant (as verified with `ant -version`) is older or newer than this version then this is not the correct manual set. Please use the documentation appropriate to your current version. Also, if you are using a version older than the most recent release, we recommend an upgrade to fix bugs as well as provide new functionality

This document should be distributed as part of the Apache source distribution.

Contents

Chapter 1

Introduction

Apache Ant is a Java-based build tool. In theory, it is kind of like make, without make's wrinkles.

Why?

Why another build tool when there is already make, gnumake, nmake, jam, and others? Because all those tools have limitations that Ant's original author couldn't live with when developing software across multiple platforms. Make-like tools are inherently shell-based: they evaluate a set of dependencies, then execute commands not unlike what you would issue on a shell. This means that you can easily extend these tools by using or writing any program for the OS that you are working on; however, this also means that you limit yourself to the OS, or at least the OS type, such as Unix, that you are working on.

Makefiles are inherently evil as well. Anybody who has worked on them for any time has run into the dreaded tab problem. "Is my command not executing because I have a space in front of my tab?!!" said the original author of Ant way too many times. Tools like Jam took care of this to a great degree, but still have yet another format to use and remember.

Ant is different. Instead of a model where it is extended with shell-based commands, Ant is extended using Java classes. Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by an object that implements a particular Task interface.

Granted, this removes some of the expressive power that is inherent in being able to construct a shell command such as 'find . -name foo -exec rm ', but it gives you the ability to be cross-platform - to work anywhere and everywhere. And hey, if you really need to execute a shell command, Ant has an `<exec>` task that allows different commands to be executed based on the OS it is executing on.

Chapter 2

Getting, Installing and Building Ant

2.1 Getting Ant

2.1.1 Binary Edition

The latest stable version of Ant is available from the Ant web page `http://ant.apache.org/`. If you like living on the edge, you can download the latest version from `http://cvs.apache.org/builds/ant/nightly/`.

2.1.2 Source Edition

If you prefer the source edition, you can download the source for the latest Ant release from `http://ant.apache.org/srcdownload.cgi`. Again, if you prefer the edge, you can access the code as it is being developed via CVS. The Jakarta website has details on accessing CVS. Please checkout the ant module. See the section Building Ant on how to build Ant from the source code. You can also access the Ant CVS repository on-line.

2.2 System Requirements

Ant has been used successfully on many platforms, including Linux, commercial flavours of Unix such as Solaris and HP-UX, Windows 9x and NT, Novell Netware 6 and MacOS X. To build and use Ant, you must have a JAXP-compliant XML parser installed and available on your classpath.

The binary distribution of Ant includes the latest version of the Apache Xerces2 XML parser. Please see `http://java.sun.com/xml/` for more information about JAXP. If you wish to use a different JAXP-compliant parser, you should remove `xercesImpl.jar` and `xml-apis.jar` from Ant's lib directory. You can then either put the jars from your preferred parser into Ant's lib directory or put the jars on the system classpath.

For the current version of Ant, you will also need a JDK installed on your system, version 1.1 or later. Some tasks work better on post-1.1 systems; some tasks only work on Java 1.2 and successors. A future version of Ant -Ant 2.0- will require JDK 1.2 or later, though Ant 1.x strives to retain 1.1 compatibility.

Note: The Microsoft JVM/JDK is not adequate on its own, although the MS compiler is supported.

Note #2: If a JDK is not present, only the JRE runtime, then many tasks will not work.

2.3 Installing Ant

The binary distribution of Ant consists of the following directory layout:

```
ant
 +--- bin  // contains launcher scripts
 |
 +--- lib  // contains Ant jars plus necessary dependencies
 |
 +--- docs // contains documentation
 |      +--- ant2    // a brief description of ant2 requirements
 |      |
 |      +--- images  // various logos for html documentation
 |      |
 |      +--- manual  // Ant documentation (a must read ;-)
 |
 +--- etc // contains xsl goodies to:
          //   - create an enhanced report from xml output of various tasks.
          //   - migrate your build files and get rid of 'deprecated' warning
          //   - ... and more ;-)
```

Only the bin and lib directories are required to run Ant. To install Ant, choose a directory and copy the distribution file there. This directory will be known as `ANT_HOME`.

Windows 95, Windows 98 and Windows ME Note:

On these systems, the script used to launch Ant will have problems if `ANT_HOME` *is a long filename (i.e. a filename which is not of the format known as "8.3"). This is due to limitations in the OS's handling of the "for" batch-file statement. It is recommended, therefore, that Ant be installed in a short, 8.3 path, such as* `C:\Ant`.

On these systems you will also need to configure more environment space to cater for the environment variables used in the Ant launch script. To do this, you will need to add or update the following line in the `config.sys` file

```
shell=c:\command.com c:\ /p /e:32768
```

2.3.1 Setup

Before you can run ant there is some additional set up you will need to do:

- Add the bin directory to your path.
- Set the `ANT_HOME` environment variable to the directory where you installed Ant. On some operating systems the ant wrapper scripts can guess `ANT_HOME` (Unix dialects and Windows NT/2000) - but it is better to not rely on this behavior.
- Optionally, set the `JAVA_HOME` environment variable (see the Advanced section below). This should be set to the directory where your JDK is installed.

Note: Do not install Ant's ant.jar file into the lib/ext directory of the JDK/JRE. Ant is an application, whilst the extension directory is intended for JDK extensions. In particular there are security restrictions on the classes which may be loaded by an extension.

2.3.2 Optional Tasks

Ant supports a number of optional tasks. An optional task is a task which typically requires an external library to function. The optional tasks are packaged together with the core Ant tasks.

The external libraries required by each of the optional tasks is detailed in the Library Dependencies section. These external libraries may either be placed in Ant's lib directory, where they will be picked up automatically, or made available on the system CLASSPATH environment variable.

2.3.3 MS-Windows

Assume Ant is installed in `c:\ant`. The following sets up the environment:

```
set ANT_HOME=c:\ant
set JAVA_HOME=c:\jdk1.2.2
set PATH=%PATH%;%{\tt ANT\_HOME}%\bin
```

2.3.4 Unix (bash)

Assume Ant is installed in `/usr/local/ant`. The following sets up the environment:

```
export ANT_HOME=/usr/local/ant
export JAVA_HOME=/usr/local/jdk-1.2.2
export PATH=${PATH}:${ANT_HOME}/bin
```

2.3.5 Unix (csh)

```
setenv ANT_HOME /usr/local/ant
setenv JAVA_HOME /usr/local/jdk-1.2.2
set path=( $path ANT_HOME/bin )
```

2.3.6 Advanced

There are lots of variants that can be used to run Ant. What you need is at least the following:

- The classpath for Ant must contain ant.jar and any jars/classes needed for your chosen JAXP-compliant XML parser.
- When you need JDK functionality (such as for the javac task or the rmic task), then for JDK 1.1, the classes.zip file of the JDK must be added to the classpath; for JDK 1.2 or JDK 1.3, tools.jar must be added. The scripts supplied with Ant, in the bin directory, will add the required JDK classes automatically, if the `JAVA_HOME` environment variable is set.
- When you are executing platform-specific applications, such as the exec task or the cvs task, the property `ant.home` must be set to the directory containing where you installed Ant. Again this is set by the Ant scripts to the value of the `ANT_HOME` environment variable.

The supplied ant shell scripts all support an `ANT_OPTS` environment variable which can be used to supply extra options to ant. Some of the scripts also read in an extra script stored in the users home directory, which can be used to set such options. Look at the source for your platform's invocation script for details.

2.4 Building Ant

To build Ant from source, you can either install the Ant source distribution or checkout the ant module from CVS.

Once you have installed the source, change into the installation directory.

Set the `JAVA_HOME` environment variable to the directory where the JDK is installed. See Installing Ant for examples on how to do this for your operating system.

Make sure you have downloaded any auxiliary jars required to build tasks you are interested in. These should either be available on the `CLASSPATH` or added to the lib directory. See Library Dependencies (section 2.5 for a list of jar requirements for various features. Note that this will make the auxiliary jars available for the building of Ant only. For running Ant you will still need to make the jars available as described under Installing Ant.

Your are now ready to build Ant:

```
build -Ddist.dir=<directory_to_contain_Ant_distribution> dist    (Windows)

build.sh -Ddist.dir=<directory_to_contain_Ant_distribution> dist    (Unix)
```

This will create a binary distribution of Ant in the directory you specified.

The above action does the following:

- If necessary it will bootstrap the Ant code. Bootstrapping involves the manual compilation of enough Ant code to be able to run Ant. The bootstrapped Ant is used for the remainder of the build steps.

- Invokes the bootstrapped Ant with the parameters passed to the build script. In this case, these parameters define an Ant property value and specify the "dist" target in Ant's own build.xml file.

On most occasions you will not need to explicitly bootstrap Ant since the build scripts do that for you. If however, the build file you are using makes use of features not yet compiled into the bootstrapped Ant, you will need to manually bootstrap. Run bootstrap.bat (Windows) or bootstrap.sh (UNIX) to build a new bootstrap version of Ant.

If you wish to install the build into the current `ANT_HOME` directory, you can use:
`build install` (Windows)
`build.sh install` (Unix)
You can avoid the lengthy Javadoc step, if desired, with:
`build install-lite` (Windows)
`build.sh install-lite` (Unix)
This will only install the bin and lib directories. Both the install and install-lite targets will overwrite the current Ant version in `ANT_HOME`.

2.5 Library Dependencies

The following libraries are needed in your `CLASSPATH` or in the install directory's lib directory if you are using the indicated feature. Note that only one of the regexp libraries is needed for use with the mappers. You will also need to install the Ant optional jar containing the task definitions to make these tasks available. Please refer to the Installing Ant / Optional Tasks section above.

Jar Name	**Needed for**	**Available at**
An XSL transformer like Xalan or XSL:P	style task	`http://xml.apache.org/xalan-j/index.html` for Xalan. XSL:P used to live at `http://www.clc-marketing.com/xslp/`, but the link doesn't work any longer and we are not aware of a replacement site.
jakarta-regexp-1.2.jar	regexp type with mappers	`jakarta.apache.org/regexp/`
junit.jar	junit tasks	`www.junit.org`
xalan.jar	junitreport.task	`xml.apache.org`
stylebook.jar	stylebook task	CVS repository of `xml.apache.org`
testlet.jar	**deprecated** test task	Build from the gzip compress tar archive in `http://avalon.apache.org/historiccvs/testlet/`
antlr.jar	antlr task	`www.antlr.org`
bsf.jar	script task	`oss.software.ibm.com/developerworks/projects/bsf`
netrexx.jar	netrexx task	`www2.hursley.ibm.com/netrexx`
js.jar	javascript with script task	`www.mozilla.org/rhino`
jpython.jar	python with script task	`www.jpython.org`
jacl.jar and tcljava.jar	TCL with script task	`www.scriptics.com/java`
BeanShell JAR(s)	BeanShell with script task	`www.beanshell.org`
netcomponents.jar	ftp and telnet tasks	`www.savarese.org/oro/downloads`
bcel.jar	classfileset data type, JavaClassHelper used by the ClassConstants filter reader and optionally used by ejbjar for dependency determination	`jakarta.apache.org/bcel/`
mail.jar	Mail task with Mime encoding, and the MimeMail task	`http://java.sun.com/products/javamail/`
activation.jar	Mail task with Mime encoding, and the MimeMail task	`http://java.sun.com/products/javabeans/glasgow/jaf.html`
jdepend.jar	jdepend task	`http://www.clarkware.com/software/JDepend.html`

2.6 Platform Specific Issues

2.6.1 Unix

You should use a GNU version of tar to untar the ant source tree, if you have downloaded this as a tar file.

- Ant does not preserve file permissions when a file is copied, moved or archived. Use `<chmod>` to set permissions, and when creating a tar archive, use the mode attribute of `<tarfileset>` to set the permissions in the tar file.

- Ant is not symbolic link aware in moves, deletes and when recursing down a tree of directories to build up a list of files. Unexpected things can happen.

2.6.2 Microsoft Windows

Windows 9x (win95, win98, win98SE and winME) has a batch file system which does not work fully with long file names, so we recommend that ant and the JDK are installed into directories without spaces, and with 8.3 filenames. The Perl and Python launcher scripts do not suffer from this limitation.

All versions of windows are usually case insensitive, although mounted file systems (Unix drives, Clearcase views) can be case sensitive underneath, confusing patternsets.

Ant can often not delete a directory which is open in an Explorer window. There is nothing we can do about this short of spawning a program to kill the shell before deleting directories.

2.6.3 Apple MacOS X

MacOS X is the first of the Apple platforms that Ant supports completely; it is treated like any other Unix.

2.6.4 Novell Netware

To give the same level of sophisticated control as Ant's startup scripts on other platforms, it was decided to make the main ant startup on NetWare be via a Perl Script, "runant.pl". This is found in the bin directory (for instance - bootstrap\bin or dist\bin).

One important item of note is that you need to set up the following to run ant:

- CLASSPATH - put ant.jar, xercesImpl.jar, xml-apis.jar and any other needed jars on the system classpath.

- `ANT_OPTS` - On NetWare, `ANT_OPTS` needs to include a parameter of the form, "-envCWD=ANT_HOME", with `ANT_HOME` being the fully expanded location of Ant, not an environment variable. This is due to the fact that the NetWare System Console has no notion of a current working directory.

It is suggested that you create up an ant.ncf that sets up these parameters, and calls perl `ANT_HOME/dist/bin/runant.pl`

The following is an example of such an NCF file(assuming ant is installed in 'sys:/apache-ant/'):

```
envset CLASSPATH=SYS:/apache-ant/bootstrap/lib/ant.jar
envset CLASSPATH=$CLASSPATH;SYS:/apache-ant/lib/xercesImpl.jar
envset CLASSPATH=$CLASSPATH;SYS:/apache-ant/lib/xml-apis.jar
envset CLASSPATH=$CLASSPATH;SYS:/apache-ant/lib/optional/junit.jar
envset CLASSPATH=$CLASSPATH;SYS:/apache-ant/bootstrap/lib/optional.jar

setenv ANT_OPTS=-envCWD=sys:/apache-ant
envset ANT_OPTS=-envCWD=sys:/apache-ant
setenv ANT_HOME=sys:/apache-ant/dist/lib
envset ANT_HOME=sys:/apache-ant/dist/lib

perl sys:/apache-ant/dist/bin/runant.pl
```

Ant works on JVM version 1.3 or higher. You may have some luck running it on JVM 1.2, but serious problems have been found running Ant on JVM 1.1.7B. These problems are caused by JVM bugs that will not be fixed.

JVM 1.3 is supported on Novell NetWare versions 5.1 and higher.

2.6.5 Other platforms

Support for other platforms is not guaranteed to be complete, as certain techniques to hide platform details from build files need to be written and tested on every particular platform. Contributions in this area are welcome.

Chapter 3

Using Ant

3.1 Writing a Simple Buildfile

Ant's buildfiles are written in XML. Each buildfile contains one project and at least one (default) target. Targets contain task elements. Each task element of the buildfile can have an id attribute and can later be referred to by the value supplied to this. The value has to be unique. (For additional information, see the Tasks section below.)

3.2 Projects

A project has three attributes:

Attribute	**Description**	**Required**
name	the name of the project.	No
default	the default target to use when no target is supplied.	Yes.
basedir	the base directory from which all path calculations are done. This attribute might be overridden by setting the "basedir" property beforehand. When this is done, it must be omitted in the project tag. If neither the attribute nor the property have been set, the parent directory of the buildfile will be used.	No

Optionally, a description for the project can be provided as a top-level `<description>` element (see the description type).

Each project defines one or more targets. A target is a set of tasks you want to be executed. When starting Ant, you can select which target(s) you want to have executed. When no target is given, the project's default is used.

3.3 Targets

A target can depend on other targets. You might have a target for compiling, for example, and a target for creating a distributable. You can only build a distributable when you have compiled first, so the distribute target depends on the compile target. Ant resolves these dependencies.

It should be noted, however, that Ant's depends attribute only specifies the order in which targets should be executed - it does not affect whether the target that specifies the dependency(s) gets executed if the dependent target(s) did not (need to) run.

Ant tries to execute the targets in the depends attribute in the order they appear (from left to right). Keep in mind that it is possible that a target can get executed earlier when an earlier target depends on it:

```
<target name="A"/>
<target name="B" depends="A"/>
<target name="C" depends="B"/>
<target name="D" depends="C,B,A"/>
```

Suppose we want to execute target D. From its depends attribute, you might think that first target C, then B and then A is executed. Wrong! C depends on B, and B depends on A, so first A is executed, then B, then C, and finally D.

A target gets executed only once, even when more than one target depends on it (see the previous example).

A target also has the ability to perform its execution if (or unless) a property has been set. This allows, for example, better control on the building process depending on the state of the system (java version, OS, command-line property defines, etc.). To make a target sense this property, you should add the if (or unless) attribute with the name of the property that the target should react to. Note: Ant will only check whether the property has been set, the value doesn't matter. A property set to the empty string is still an existing property. For example:

```
<target name="build-module-A" if="module-A-present"/>
<target name="build-own-fake-module-A" unless="module-A-present"/>
```

In the first example, if the module-A-present property is set (to any value), the target will be run. In the second example, if the module-A-present property is set (again, to any value), the target will not be run.

If no if and no unless attribute is present, the target will always be executed.

The optional description attribute can be used to provide a one-line description of this target, which is printed by the -projecthelp command-line option. Targets without such a description are deemed internal and will not be listed, unless either the -verbose or -debug option is used.

It is a good practice to place your tstamp tasks in a so-called initialization target, on which all other targets depend. Make sure that target is always the first one in the depends list of the other targets. In this manual, most initialization targets have the name "init".

A target has the following attributes:

Attribute	Description	Required
name	the name of the target.	Yes
depends	a comma-separated list of names of targets on which this target depends.	No
if	the name of the property that must be set in order for this target to execute.	No
unless	the name of the property that must not be set in order for this target to execute.	No
description	a short description of this target's function.	No

A target name can be any alphanumeric string valid in the encoding of the XML file. The empty string ”” is in this set, as is comma ”,” and space ” ”. Please avoid using these, as they will not be supported in future Ant versions because of all the confusion they cause. IDE support of unusual target names, or any target name containing spaces, varies with the IDE.

Targets beginning with a hyphen such as ”-restart” are valid, and can be used to name targets that should not be called directly from the command line.

3.4 Tasks

A task is a piece of code that can be executed.

A task can have multiple attributes (or arguments, if you prefer). The value of an attribute might contain references to a property. These references will be resolved before the task is executed.

Tasks have a common structure:

```
<name attribute1="value1" attribute2="value2" ... />
```

where name is the name of the task, attributeN is the attribute name, and valueN is the value for this attribute.

There is a set of built-in tasks, along with a number of optional tasks, but it is also very easy to write your own.

All tasks share a task name attribute. The value of this attribute will be used in the logging messages generated by Ant.

Tasks can be assigned an id attribute:

```
<taskname id="taskID" ... />
```

where taskname is the name of the task, and taskID is a unique identifier for this task. You can refer to the corresponding task object in scripts or other tasks via this name. For example, in scripts you could do:

```
<script ... >
  task1.setFoo("bar");
</script>
```

to set the foo attribute of this particular task instance. In another task (written in Java), you can access the instance via project.getReference(”task1”). Note1: If ”task1” has not been run yet, then it has not been configured (ie., no attributes have been set), and if it is going to be configured later, anything you've done to the instance may be overwritten.

Note2: Future versions of Ant will most likely not be backward-compatible with this behaviour, since there will likely be no task instances at all, only proxies.

3.5 Properties

A project can have a set of properties. These might be set in the buildfile by the property task, or might be set outside Ant. A property has a name and a value; the name is case-sensitive. Properties may be used in the value of task attributes. This is done by placing the property name between ”${” and ”}” in the attribute value. For example, if there is a ”builddir” property with the value ”build”, then this could be used in an attribute like this: ${builddir}/classes. This is resolved at run-time as build/classes.

3.6 Built-in Properties

Ant provides access to all system properties as if they had been defined using a `<property> task`. For example, $os.name expands to the name of the operating system.

For a list of system properties see the Javadoc of `System.getProperties`.

In addition, Ant has some built-in properties:

`basedir`	`the absolute path of the project's basedir (as set with the basedir attribute of <project>).`
`ant.file`	`the absolute path of the buildfile.`
`ant.version`	`the version of Ant`
`ant.project.name`	`the name of the project that is currently executing; it is set in the name attribute of <project>.`
`ant.java.version`	`the JVM version Ant detected; currently it can hold the values "1.1", "1.2", "1.3" and "1.4".`

3.7 Example Buildfile

```
<project name="MyProject" default="dist" basedir=".">
    <description>
        simple example build file
    </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist"  location="dist"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
  </target>

  <target name="compile" depends="init"
        description="compile the source " >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <target name="dist" depends="compile"
        description="generate the distribution" >
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar file -->
    <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar" basedir="${build}"/>
  </target>

  <target name="clean"
        description="clean up" >
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

Notice that we are declaring properties outside any target. The `<property>`, `<typedef>` and `<taskdef>` tasks are special in that they can be declared outside any target. When you do this they are evaluated before any targets are executed. No other tasks can be declared outside targets. We have given some targets descriptions; this causes the projecthelp invocation option to list them as public targets with the descriptions; the other target is internal and not listed.

Finally, for this target to work the source in the src subdirectory should be stored in a directory tree which matches the package names. Check the `<javac>` task for details.

3.8 Token Filters

A project can have a set of tokens that might be automatically expanded if found when a file is copied, when the filtering-copy behavior is selected in the tasks that support this. These might be set in the buildfile by the filter task.

Since this can potentially be a very harmful behavior, the tokens in the files must be of the form @token@, where token is the token name that is set in the `<filter>` task. This token syntax matches the syntax of other build systems that perform such filtering and remains sufficiently orthogonal to most programming and scripting languages, as well as with documentation systems.

Note: If a token with the format @token@ is found in a file, but no filter is associated with that token, no changes take place; therefore, no escaping method is available - but as long as you choose appropriate names for your tokens, this should not cause problems.

Warning: If you copy binary files with filtering turned on, you can corrupt the files. This feature should be used with text files only.

3.9 Path-like Structures

You can specify PATH- and CLASSPATH-type references using both ":" and ";" as separator characters. Ant will convert the separator to the correct character of the current operating system.

Wherever path-like values need to be specified, a nested element can be used. This takes the general form of:

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

The location attribute specifies a single file or directory relative to the project's base directory (or an absolute filename), while the path attribute accepts colon- or semicolon-separated lists of locations. The path attribute is intended to be used with predefined paths - in any other case, multiple elements with location attributes should be preferred.

As a shortcut, the `<classpath>` tag supports path and location attributes of its own, so:

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

can be abbreviated to:

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

In addition, DirSets, FileSets, and FileLists can be specified via nested `<dirset>`, `<fileset>`, and `<filelist>` elements, respectively. Note: The order in which the files building up a FileSet are added to the path-like structure is not defined.

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

This builds a path that holds the value of ${classpath}, followed by all jar files in the lib directory, the classes directory, all directories named classes under the apps subdirectory of ${build.dir}, except those that have the text Test in their name, and the files specified in the referenced FileList.

If you want to use the same path-like structure for several tasks, you can define them with a `<path>` element at the same level as targets, and reference them via their id attribute - see References for an example.

A path-like structure can include a reference to another path-like structure via nested `<path>` elements:

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

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

The shortcuts previously mentioned for `<classpath>` are also valid for `<path>`. For example:

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

can be written as:

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

3.10 Command-line Arguments

Several tasks take arguments that will be passed to another process on the command line. To make it easier to specify arguments that contain space characters, nested arg elements can be used.

Attribute Description Required value a single command-line argument; can contain space characters. Exactly one of these. file The name of a file as a single command-line argument; will be replaced with the absolute filename of the file. path A string that will be treated as a path-like string as a single command-line argument; you can use ; or : as path separators and Ant will convert it to the platform's local conventions. line a space-delimited list of command-line arguments.

It is highly recommended to avoid the line version when possible. Ant will try to split the command line in a way similar to what a (Unix) shell would do, but may create something that is very different from what you expect under some circumstances.

Examples

```
<arg value="-l -a"/>
```

is a single command-line argument containing a space character.

```
<arg line="-l -a"/>
```

represents two separate command-line arguments.

```
<arg path="/dir;/dir2:\dir3"/>
```

is a single command-line argument with the value \dir;\dir2;\dir3 on DOS-based systems and /dir:/dir2:/dir3 on Unix-like systems.

3.11 References

The id attribute of the buildfile's elements can be used to refer to them. This can be useful if you are going to replicate the same snippet of XML over and over again - using a `<classpath>` structure more than once, for example.

The following example:

```
<project ... >
  <target ... >
    <rmic ...>
      <classpath>
        <pathelement location="lib/"/>
        <pathelement path="${java.class.path}/"/>
        <pathelement path="${additional.path}"/>
      </classpath>
    </rmic>
  </target>

  <target ... >
    <javac ...>
      <classpath>
        <pathelement location="lib/"/>
        <pathelement path="${java.class.path}/"/>
        <pathelement path="${additional.path}"/>
      </classpath>
    </javac>
  </target>
</project>
```

could be rewritten as:

```
<project ... >
  <path id="project.class.path">
    <pathelement location="lib/"/>
    <pathelement path="${java.class.path}/"/>
    <pathelement path="${additional.path}"/>
  </path>

  <target ... >
    <rmic ...>
      <classpath refid="project.class.path"/>
    </rmic>
  </target>

  <target ... >
    <javac ...>
      <classpath refid="project.class.path"/>
    </javac>
  </target>
</project>
```

All tasks that use nested elements for PatternSets, FileSets or path-like structures accept references to these structures as well.

Chapter 4

Running Ant

4.1 Command Line

If you've installed Ant as described in the Installing Ant section, running Ant from the command-line is simple: just type ant.

When no arguments are specified, Ant looks for a build.xml file in the current directory and, if found, uses that file as the build file and runs the target specified in the default attribute of the `<project>` tag. To make Ant use a build file other than build.xml, use the command-line option -buildfile file, where file is the name of the build file you want to use.

If you use the -find [file] option, Ant will search for a build file first in the current directory, then in the parent directory, and so on, until either a build file is found or the root of the filesystem has been reached. By default, it will look for a build file called build.xml. To have it search for a build file other than build.xml, specify a file argument. Note: If you include any other flags or arguments on the command line after the -find flag, you must include the file argument for the -find flag, even if the name of the build file you want to find is build.xml. You can also set properties on the command line. This can be done with the -Dproperty=value option, where property is the name of the property, and value is the value for that property. If you specify a property that is also set in the build file (see the property task), the value specified on the command line will override the value specified in the build file. Defining properties on the command line can also be used to pass in the value of environment variables - just pass -DMYVAR=%MYVAR% (Windows) or `-DMYVAR=$MYVAR` (Unix) to Ant. You can then access these variables inside your build file as `${MYVAR}`. You can also access environment variables using the property task's environment attribute.

Options that affect the amount of logging output by Ant are: -quiet, which instructs Ant to print less information to the console; -verbose, which causes Ant to print additional information to the console; and -debug, which causes Ant to print considerably more additional information.

It is also possible to specify one or more targets that should be executed. When omitted, the target that is specified in the default attribute of the project tag is used.

The -projecthelp option prints out a list of the build file's targets. Targets that include a description attribute are listed as "Main targets", those without a description are listed as "Subtargets", then the "Default" target is listed.

4.2 Command-line Options Summary

```
ant [options] [target [target2 [target3] ...]]
Options:
  -help                  print this message
  -projecthelp           print project help information
  -version               print the version information and exit
  -diagnostics           print information that might be helpful to
                         diagnose or report problems.
  -quiet, -q             be extra quiet
  -verbose, -v           be extra verbose
  -debug                 print debugging information
  -emacs                 produce logging information without adornments
  -logfile <file>        use given file for log
    -l     <file>                ''
  -logger <classname>    the class which is to perform logging
  -listener <classname>  add an instance of class as a project listener
  -buildfile <file>      use given buildfile
    -file    <file>              ''
    -f       <file>              ''
  -D<property>=<value>   use value for given property
  -propertyfile <name>   load all properties from file with -D
                         properties taking precedence
  -inputhandler <class>  the class which will handle input requests
  -find <file>           search for buildfile towards the root of the
                         filesystem and use it
```

For more information about -logger and -listener see Loggers and Listeners.

For more information about -inputhandler see InputHandler.

4.2.1 Examples

ant runs Ant using the build.xml file in the current directory, on the default target.

ant -buildfile test.xml runs Ant using the test.xml file in the current directory, on the default target.

ant -buildfile test.xml dist runs Ant using the test.xml file in the current directory, on the target called dist.

ant -buildfile test.xml -Dbuild=build/classes dist runs Ant using the test.xml file in the current directory, on the target called dist, setting the build property to the value build/classes.

4.3 Files

The Ant wrapper script for Unix will source (read and evaluate) the file /.antrc before it does anything. On Windows, the Ant wrapper batch-file invokes `%HOME%\antrc_pre.bat` at the start and %HOME%\antrc_post.bat at the end. You can use these files, for example, to set/unset environment variables that should only be visible during the execution of Ant. See the next section for examples.

4.4 Environment Variables

The wrapper scripts use the following environment variables (if set):

- JAVACMD - full path of the Java executable. Use this to invoke a different JVM than `JAVA_HOME`/bin/java(.exe).
- `ANT_OPTS` - command-line arguments that should be passed to the JVM. For example, you can define system properties or set the maximum Java heap size here.
- `ANT_ARGS` - Ant command-line arguments. For example, set `ANT_ARGS` to point to a different logger, include a listener, and to include the -find flag. Note: If you include -find in `ANT_ARGS`, you should include the name of the build file to find, even if the file is called build.xml.

4.5 Cygwin Users

The Unix launch script that come with Ant works correctly with Cygwin. You should not have any problems launching Ant form the Cygwin shell. It is important to note however, that once Ant is runing it is part of the JDK which operates as a native Windows application. The JDK is not a Cygwin executable, and it therefore has no knowledge of the Cygwin paths, etc. In particular when using the `<exec>` task, executable names such as "/bin/sh" will not work, even though these work from the Cygwin shell from which Ant was launched. You can use an executable name such as "sh" and rely on that command being available in the Windows path.

4.6 Running Ant via Java

If you have installed Ant in the do-it-yourself way, Ant can be started with:

```
java -Dant.home=c:\ant org.apache.tools.ant.Main [options] [target]
```

These instructions actually do exactly the same as the ant command. The options and target are the same as when running Ant with the ant command. This example assumes you have set your classpath to include:

- ant.jar
- jars/classes for your XML parser
- the JDK's required jar/zip files

Chapter 5

Ant Tasks

Given the large number of tasks available with Ant, it may be difficult to get an overall view of what each task can do. The following tables provide a short description of each task and a link to the complete documentation.

5.1 Overview of Ant Tasks

5.1.1 Archive Tasks

Task Name	Description
BUnzip2	Expands a file packed using GZip or BZip2.
BZip2	Packs a file using the GZip or BZip2 algorithm. This task does not do any dependency checking; the output file is always generated
Cab	Creates Microsoft CAB archive files. It is invoked similar to the Jar or Zip tasks. This task will work on Windows using the external cabarc tool (provided by Microsoft), which must be located in your executable path.
Ear	An extension of the Jar task with special treatment for files that should end up in an Enterprise Application archive.
GUnzip	Expands a GZip file.
GZip	GZips a set of files.
Jar	Jars a set of files.
Jlink	*Deprecated.* Use the zipfileset and zipgroupfileset attributes of the Jar or Zip tasks instead.
Manifest	Creates a manifest file.
Rpm	Invokes the rpm executable to build a Linux installation file. This task currently only works on Linux or other Unix platforms with RPM support.

Task Name	Description
SignJar	Signs a jar or zip file with the javasign command-line tool.
Tar	Creates a tar archive.
Unjar	Unzips a jarfile.
Untar	Untars a tarfile.
Unwar	Unzips a warfile.
Unzip	Unzips a zipfile.
War	An extension of the Jar task with special treatment for files that should end up in the WEB-INF/lib, WEB-INF/classes, or WEB-INF directories of the Web Application Archive.
Zip	Creates a zipfile.

5.1.2 Audit/Coverage Tasks

Task Name	Description
JDepend	Invokes the JDepend parser. This parser ”traverses a set of Java source-file directories and generates design-quality metrics for each Java package”.
JProbe	These tasks run the tools from the JProbe suite. This task was written using JProbe Suite Server Side 3.0.
MMetrics	Computes the metrics of a set of Java source files, using the Metamata Metrics/WebGain Quality Analyzer source-code analyzer, and writes the results to an XML file.
Maudit	Performs static analysis on a set of Java source-code and byte-code files, using the Metamata Metrics/WebGain Quality Analyzer source-code analyzer.

5.1.3 Compile Tasks

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

5.1.4 Deployment Tasks

Task Name	Description
ServerDeploy	Task to run a ”hot” deployment tool for vendor-specific J2EE server.

5.1.5 Documentation Tasks

Task Name	Description
Javadoc/Javadoc2	Generates code documentation using the javadoc tool. The Javadoc2 task is deprecated; use the Javadoc task instead.
Stylebook	Executes the Apache Stylebook documentation generator. Unlike the command-line version of this tool, all three arguments are required to run the Stylebook task.

5.1.6 EJB Tasks

Task Name	Description
EJB Tasks	(See the documentation describing the EJB tasks.)

5.1.7 Execution Tasks

Task Name	Description
Ant	Runs Ant on a supplied buildfile, optionally passing properties (with possibly new values). This task can be used to build sub-projects.
AntCall	Runs another target within the same buildfile, optionally passing properties (with possibly new values).
Apply/ExecOn	Executes a system command. When the os attribute is specified, the command is only executed when Ant is run on one of the specified operating systems.
Dependset	This task compares a set of source files with a set of target files. If any of the source files is newer than any of the target files, all the target files are removed.
Exec	Executes a system command. When the os attribute is specified, the command is only executed when Ant is run on one of the specified operating systems.
Java	Executes a Java class within the running (Ant) VM, or in another VM if the fork attribute is specified.
Parallel	A container task that can contain other Ant tasks. Each nested task specified within the `<parallel>` tag will be executed in its own thread.
Sequential	A container task that can contain other Ant tasks. The nested tasks are simply executed in sequence. Its primary use is to support the sequential execution of a subset of tasks within the `<parallel>` tag.
Sleep	A task for suspending execution for a specified period of time. Useful when a build or deployment process requires an interval between tasks.
Waitfor	Blocks execution until a set of specified conditions become true. This task is intended to be used with the Parallel task to synchronize a set of processes.

5.1.8 File Tasks

Task Name	Description
Checksum	Generates a checksum for a file or set of files. This task can also be used to perform checksum verifications.
Chmod	Changes the permissions of a file or all files inside the specified directories. Currently, it has effect only under Unix. The permissions are also UNIX style, like the arguments for the chmod command.
Concat	Concatenates multiple files into a single one or to Ant's logging system.
Copy	Copies a file or Fileset to a new file or directory.
Copydir	*Deprecated.* Use the Copy task instead.
Copyfile	*Deprecated.* Use the Copy task instead.
Delete	Deletes either a single file, all files and sub-directories in a specified directory, or a set of files specified by one or more FileSets.

Task Name	Description
Deltree	*Deprecated.* Use the Delete task instead.
Filter	Sets a token filter for this project, or reads multiple token filters from a specified file and sets these as filters. Token filters are used by all tasks that perform file-copying operations.
FixCRLF	Modifies a file to add or remove tabs, carriage returns, linefeeds, and EOF characters.
Get	Gets a file from a URL.
Mkdir	Creates a directory. Non-existent parent directories are created, when necessary.
Move	Moves a file to a new file or directory, or a set(s) of file(s) to a new directory.
Patch	Applies a "diff" file to originals.
Rename	*Deprecated.* Use the Move task instead.
RenameExtensions	*Deprecated.* Use the Move task with a glob mapper instead.
Replace	Replace is a directory-based task for replacing the occurrence of a given string with another string in selected file.
ReplaceRegExp	Directory-based task for replacing the occurrence of a given regular expression with a substitution pattern in a file or set of files.
Tempfile	Generates a name for a new temporary file and sets the specified property to that name.
Touch	Changes the modification time of a file and possibly creates it at the same time.

5.1.9 Java2 Extensions Tasks

Task Name	Description
Jarlib-available	Check whether an extension is present in a FileSet or an ExtensionSet. If the extension is present, the specified property is set.
Jarlib-display	Display the "Optional Package" and "Package Specification" information contained within the specified jars.
Jarlib-manifest	Task to generate a manifest that declares all the dependencies in manifest. The dependencies are determined by looking in the specified path and searching for Extension/"Optional Package" specifications in the manifests of the jars.
Jarlib-resolve	Try to locate a jar to satisfy an extension, and place the location of the jar into the specified property.

5.1.10 Logging Tasks

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

5.1.11 Mail Tasks

Task Name	Description
Mail	A task to send SMTP email.
MimeMail	*Deprecated.* Use the Mail task instead.

5.1.12 Miscellaneous Tasks

Task Name	Description
Echo	Echoes text to System.out or to a file.
Fail	Exits the current build by throwing a BuildException, optionally printing additional information.
GenKey	Generates a key in keystore.
Input	Allows user interaction during the build process by displaying a message and reading a line of input from the console.
Script	Executes a script in a BSF-supported language.
Sound	Plays a sound file at the end of the build, according to whether the build failed or succeeded.
Splash	Displays a splash screen.
Sql	Executes a series of SQL statements via JDBC to a database. Statements can either be read in from a text file using the src attribute, or from between the enclosing SQL tags.
Taskdef	Adds a task definition to the current project, such that this new task can be used in the current project.
TStamp	Sets the DSTAMP, TSTAMP, and TODAY properties in the current project, based on the current date and time.
Typedef	Adds a data-type definition to the current project, such that this new type can be used in the current project.
XmlValidate	Checks that XML files are valid (or only well-formed). This task uses the XML parser that is currently used by Ant by default, but any SAX1/2 parser can be specified, if needed.

5.1.13 .NET Tasks

Task Name	Description
.NET Tasks	(See the documentation describing the .NET tasks.)

5.1.14 Pre-process Tasks

Task Name	Description
ANTLR	Invokes the ANTLR Translator generator on a grammar file.
AntStructure	Generates a DTD for Ant buildfiles that contains information about all tasks currently known to Ant.
IContract	Instruments Java classes using the iContract DBC preprocessor. This task can generate a properties file for iControl, a graphical user interface that lets you turn on/off assertions.

Task Name	Description
JavaCC	Invokes the JavaCC compiler-compiler on a grammar file.
Javah	Generates JNI headers from a Java class.
JJTree	Invokes the JJTree preprocessor for the JavaCC compiler-compiler. It inserts parse-tree building actions at various places in the JavaCC source that it generates. The output of JJTree is run through JavaCC to create the parser. This task only invokes JJTree if the grammar file is newer than the generated JavaCC file.
MParse	Invokes the Metamata MParse compiler-compiler on a grammar file.
Native2Ascii	Converts files from native encodings to ASCII with escaped Unicode. A common usage is to convert source files maintained in a native operating system encoding to ASCII, prior to compilation.
Translate	Identifies keys in files, delimited by special tokens, and translates them with values read from resource bundles.
Xslt/Style	Processes a set of documents via XSLT.

5.1.15 Property Tasks

Task Name	Description
Available	Sets a property if a specified file, directory, class in the classpath, or JVM system resource is available at runtime.
Basename	Sets a property to the last element of a specified path.
BuildNumber	Task that can be used to track build numbers.
Condition	Sets a property if a certain condition holds true - this is a generalization of Available and Uptodate.
Dirname	Sets a property to the value of the specified file up to, but not including, the last path element.
Echoproperties	Lists the current properties.
LoadFile	Loads a file into a property.
LoadProperties	Load a file's contents as Ant properties. This task is equivalent to using `<property file="..."/>` except that it supports nested `<filterchain>` elements, and it cannot be specified outside a target.
PathConvert	Converts a nested path, path reference, filelist reference, or fileset reference to the form usable on a specified platform and/or to a list of items separated by the specified separator and stores the result in the specified property.
Property	Sets a property (by name and value), or set of properties (from a file or resource) in the project.
PropertyFile	Creates or modifies property files. Useful when wanting to make unattended modifications to configuration files for application servers and applications. Typically used for things such as automatically generating a build number and saving it to a build properties file, or doing date manipulation.
Uptodate	Sets a property if a given target file is newer than a set of source files.
XmlProperty	Loads property values from a valid XML file.

5.1.16 Remote Tasks

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

5.1.17 SCM Tasks

Task Name	Description
Cvs	Handles packages/modules retrieved from a CVS repository.
CvsChangeLog	Generates an XML report of the changes recorded in a CVS repository.
CVSPass	Adds entries to a .cvspass file. Adding entries to this file has the same affect as a cvs login command.
CvsTagDiff	Generates an XML-formatted report file of the changes between two tags or dates recorded in a CVS repository.
ClearCase	Tasks to perform the ClearCase cccheckin, cccheckout, ccuncheckout, and ccupdate commands.
Continuus/Synergy	Tasks to perform the Continuus ccmcheckin, ccmcheckout, ccmcheckintask, ccmreconfigure, and ccmcreateTask commands.
Microsoft Visual SourceSafe	Tasks to perform the Visual SourceSafe vssget, vsslabel, vsshistory, vsscheckin, vsscheckout, vssadd, vsscp, and vsscreate commands.
Perforce	Tasks to perform the Perforce p4sync, p4change, p4edit, p4submit, p4have, p4label, p4counter, p4reopen, p4revert, and p4add commands.
Pvcs	Allows the user extract the latest edition of the source code from a PVCS repository.
SourceOffSite	Tasks to perform the SourceOffSite sosget, soslabel, soscheckin, and soscheckout commands.
StarTeam	Tasks to perform the StarTeam stcheckout, stcheckin, stlabel, and stlist commands. The Starteam task is deprecated; use STCheckout instead.

5.1.18 Testing Tasks

Task Name	Description
Junit	Runs tests from the Junit testing framework. This task has been tested with JUnit 3.0 up to JUnit 3.7; it won't work with versions prior to JUnit 3.0.
hline JunitReport	Merges the individual XML files generated by the Junit task and applies a stylesheet on the resulting merged document to provide a browsable report of the testcases results.
Test	Executes a unit test in the org.apache.testlet framework.

5.1.19 Visual Age for Java Tasks

Task Name	Description
Visual Age for Java Tasks	(See the documentation describing the Visual Age for Java tasks.)

5.2 Core Tasks

5.2.1 Ant

Description

Runs Ant on a supplied buildfile. This can be used to build subprojects.

When the antfile attribute is omitted, the file ”build.xml” in the supplied directory (dir attribute) is used.

If no target attribute is supplied, the default target of the new project is used.

By default, all of the properties of the current project will be available in the new project. Alternatively, you can set the inheritAll attribute to false and only ”user” properties (i.e., those passed on the command-line) will be passed to the new project. In either case, the set of properties passed to the new project will override the properties that are set in the new project (See also the property task).

You can also set properties in the new project from the old project by using nested property tags. These properties are always passed regardless of the setting of inheritAll. This allows you to parameterize your subprojects.

References to data types can also be passed to the new project, but by default they are not. If you set the inheritrefs attribute to true, all references will be copied, but they will not override references defined in the new project.

Nested `<reference>` elements can also be used to copy references from the calling project to the new project, optionally under a different id. References taken from nested elements will override existing references in the new project.

Inherited references are not available to top level tasks of the child project.

Parameters

Attribute	Description	Required
antfile	the buildfile to use. Defaults to "build.xml". This file is expected to be a filename relative to the dir attribute given.	No
dir	the directory to use as a basedir for the new Ant project. Defaults to the current project's basedir, unless inheritall has been set to false, in which case it doesn't have a default value. This will override the basedir setting of the called project.	No
target	the target of the new Ant project that should be executed. Defaults to the new project's default target.	No
output	Filename to write the ant output to. This is relative to the value of the dir attribute if it has been set or to the base directory of the current project otherwise.	No
inheritAll	If true, pass all properties to the new Ant project. Defaults to true.	No
inheritRefs	If true, pass all references to the new Ant project. Defaults to false.	No

Parameters specified as nested elements

property

See the description of the property task. Note that the refid attribute points to a reference in the calling project, not in the new one.

reference

Used to chose references that shall be copied into the new project, optionally changing their id.

Attribute	Description	Required
refid	The id of the reference in the calling project.	Yes
torefid	The id of the reference in the new project.	No, defaults to the value of refid.

Basedir of the new project

The basedir value of the new project is affected by the two attributes dir and inheritall, see the following table for details:

dir attribute	inheritAll attribute	new project's basedir
value provided	true	value of dir attribute
value provided	false	value of dir attribute
omitted	true	basedir of calling project (the one whose build file contains the `<ant>` task).
omitted	false	basedir attribute of the ¡project¿ element of the new project

Examples

```
<ant antfile="subproject/subbuild.xml" dir="subproject" target="compile"/>

<ant dir="subproject"/>

<ant antfile="subproject/property_based_subbuild.xml">
  <property name="param1" value="version 1.x"/>
  <property file="config/subproject/default.properties"/>
</ant>

<ant inheritAll="false" antfile="subproject/subbuild.xml">
  <property name="output.type" value="html"/>
</ant>
```

The build file of the calling project defines some `<path>` elements like this:

```
<path id="path1">
  ...
</path>
<path id="path2">
  ...
</path>
```

and the called build file (subbuild.xml) also defines a `<path>` with the id path1, but path2 is not defined:

```
<ant antfile="subbuild.xml" inheritrefs="true"/>
```

will not override subbuild's definition of path1, but make the parent's definition of path2 available in the subbuild.

```
<ant antfile="subbuild.xml"/>
```

as well as

```
<ant antfile="subbuild.xml" inheritrefs="false"/>
```

will neither override path1 nor copy path2.

```
<ant antfile="subbuild.xml" inheritrefs="false">
  <reference refid="path1"/>
</ant>
```

will override subbuild's definition of path1.

```
<ant antfile="subbuild.xml" inheritrefs="false">
  <reference refid="path1" torefid="path2"/>
</ant>
```

will copy the parent's definition of path1 into the new project using the id path2

5.2.2 AntCall

Description

Call another target within the same build-file optionally specifying some properties (param's in this context)

By default, all of the properties of the current project will be available in the new project. Alternatively, you can set the inheritAll attribute to false and only "user" properties (i.e., those passed on the command-line) will be passed to the new project. In either case, the set of properties passed to the new project will override the properties that are set in the new project (See also the property task).

You can also set properties in the new project from the old project by using nested param tags. These properties are always passed regardless of the setting of inheritAll. This allows you to parameterize your subprojects.

Nested `<reference>` elements can be used to copy references from the calling project to the new project, optionally under a different id. References taken from nested elements will override existing references in the new project.

When a target is invoked by antcall, all of its dependent targets will also be called within the context of any new parameters. For example. if the target "doSomethingElse" depended on the target "init", then the antcall of "doSomethingElse" will call "init" during the call. Of course, any properties defined in the antcall task or inherited from the calling target will be fixed and not overridable in the init task -or indeed in the "doSomethingElse" task.

Parameters

Attribute	Description	Required
target	The target to execute.	Yes
inheritAll	If true, pass all properties to the new Ant project. Defaults to true.	No
inheritRefs	If true, pass all references to the new Ant project. Defaults to false.	No

Note on inheritRefs

`<antcall>` will not override existing references, even if you set inheritRefs to true. As the called build files is the same build file as the calling one, this means it will not override any reference set via an id attribute at all. The only references that can be inherited by the child project are those defined by nested

`<reference>` elements or references defined by tasks directly (not using the id attribute).

Inherited references are not available to top level tasks of the child project.

Parameters specified as nested elements

param

Specifies the properties to set before running the specified target. See property for usage guidelines.

reference

Used to chose references that shall be copied into the new project, optionally changing their id.

Attribute	Description	Required
refid	The id of the reference in the calling project.	Yes
torefid	The id of the reference in the new project.	No, defaults to the value of refid.

Examples

```
<target name="default">
  <antcall target="doSomethingElse">
    <param name="param1" value="value"/>
  </antcall>
</target>

<target name="doSomethingElse">
  <echo message="param1=${param1}"/>
</target>
```

Will run the target 'doSomethingElse' and echo 'param1=value'.

```
<antcall ... >
  <reference refid="path1" torefid="path2"/>
</antcall>
```

will copy the parent's definition of path1 into the new project using the id path2.

5.2.3 AntStructure

Description

Generates a DTD for Ant buildfiles which contains information about all tasks currently known to Ant.

Note that the DTD generated by this task is incomplete, you can always add XML entities using `<taskdef>` or `<typedef>`. See here for a way to get around this problem.

This task doesn't know about required attributes, all will be listed as #IMPLIED.

Parameters

Attribute	Description	Required
output	file to write the DTD to	Yes

Examples

```
<antstructure output="project.dtd"/>
```

5.2.4 Apply/*ExecOn*

The name execon is deprecated and only kept for backwards compatibility.

Description

Executes a system command. When the os attribute is specified, then the command is only executed when Ant is run on one of the specified operating systems.

The files and/or directories of a number of FileSets are passed as arguments to the system command.

If you specify a nested mapper and the dest attribute, the timestamp of each source file is compared to the timestamp of a target file which is defined by the nested mapper element and searched for in the given dest.

At least one fileset is required, and you must not specify more than one mapper.

Parameters

Task Name	Description	
Attribute	Description	Required
executable	the command to execute without any command line arguments.	Yes
dest	the directory where the ¡apply¿ expects the target files will be placed by the command, when it is executed.	Yes, if you specify a nested mapper
dir	the directory in which the command should be executed.	No
relative	whether the filenames should be passed on the command line as absolute or relative pathnames (relative to the base directory of the corresponding fileset for source files or the dest attribute for target files).	No, default is false
os	list of Operating Systems on which the command may be executed.	No
output	the file to which the output of the command should be redirected.	No
append	whether output should be appended to or overwrite an existing file. Defaults to false. If you set parallel to false, you will probably want to set this one to true.	No
outputproperty	the name of a property in which the output of the command should be stored.	No
resultproperty	the name of a property in which the return code of the command should be stored. Only of interest if failonerror=false. If you set parallel to false, only the result of the first execution will be stored.	No
timeout	Stop the command if it doesn't finish within the specified time (given in milliseconds).	No
failonerror	Stop the buildprocess if the command exits with a returncode other than 0.	No
failifexecutionfails	Stop the build if we can't start the program. Defaults to true.	No
skipemptyfilesets	Don't run the command, if no source files have been found or are newer than their corresponding target files.	No, default is false
parallel	Run the command only once, appending all files as arguments. If false, command will be executed once for every file. Defaults to false.	No

Attribute	Description	Required
type	One of file, dir or both. If set to file, only the names of plain files will be sent to the command. If set to dir, only the names of directories are considered.	No, default is file
newenvironment	Do not propagate old environment when new environment variables are specified.	No, default is false
vmlauncher	Run command using the Java VM's execution facilities where available. If set to false the underlying OS's shell, either directly or through the antRun scripts, will be used. Under some operating systems, this gives access to facilities not normally available through the VM including, under Windows, being able to execute scripts, rather than their associated interpreter. If you want to specify the name of the executable as a relative path to the directory given by the dir attribute, it may become necessary to set vmlauncher to false as well.	No, default is true

Parameters specified as nested elements

fileset

You can use any number of nested `<fileset>` elements to define the files for this task and refer to `<fileset>`s defined elsewhere.

arg

Command line arguments should be specified as nested `<arg>` elements. See Command line arguments.

srcfile

By default the file names of the source files will be added to the end of the command line. If you need to place it somewhere different, use a nested `<srcfile>` element between your ¡arg¿ elements to mark the insertion point.

targetfile

`<targetfile>` is similar to `<srcfile>` and marks the position of the target filename on the command line. If omitted, the target filenames will not be added to the command line at all. This element can only be specified, if you also define a nested mapper and the dest attribute.

env

It is possible to specify environment variables to pass to the system command via nested `<env>` elements. See the description in the section about exec

Examples

```
<apply executable="ls">
  <arg value="-l"/>
  <fileset dir="/tmp">
    <patternset>
      <exclude name="**/*.txt"/>
    </patternset>
  </fileset>
  <fileset refid="other.files"/>
</apply>
```

invokes ls -l, adding the absolute filenames of all files below /tmp not ending in .txt and all files of the FileSet with id other.files to the command line.

```
<apply executable="somecommand" parallel="false">
  <arg value="arg1"/>
  <srcfile/>
  <arg value="arg2"/>
  <fileset dir="/tmp"/>
</apply>
```

invokes somecommand arg1 SOURCEFILENAME arg2 for each file in /tmp replacing SOURCEFILENAME with the absolute filename of each file in turn. If parallel had been set to true, SOURCEFILENAME would be replaced with the absolute filenames of all files separated by spaces.

```
<apply executable="cc" dest="src/C" parallel="false">
  <arg value="-c"/>
  <arg value="-o"/>
  <targetfile/>
  <srcfile/>
  <fileset dir="src/C" includes="*.c"/>
  <mapper type="glob" from="*.c" to="*.o"/>
</apply>
```

invokes `cc -c -o TARGETFILE SOURCEFILE` for each .c file that is newer than the corresponding .o, replacing TARGETFILE with the absolute filename of the .o and SOURCEFILE with the absolute name of the `.c` file.

5.2.5 Available

Description

Sets a property if a resource is available at runtime. This resource can be a file, a directory, a class in the classpath, or a JVM system resource.

If the resource is present, the property value is set to true by default; otherwise, the property is not set. You can set the value to something other than the default by specifying the value attribute.

Normally, this task is used to set properties that are useful to avoid target execution depending on system parameters.

Parameters

Attribute	Description	Required
property	The name of the property to set.	Yes
value	The value to set the property to. Defaults to "true".	No
classname	The class to look for in the classpath.	Yes
file	The file to look for.	Yes
resource	The resource to look for in the JVM.	Yes
classpath	The classpath to use when looking up classname or resource.	No
filepath	The path to use when looking up file.	No
classpathref	The classpath to use, given as a reference to a path defined elsewhere.	No
type	The type of file to look for, either a directory (type="dir") or a file (type="file"). If not set, the property will be set if the name specified in the file attribute exists as either a file or a directory.	No
ignoresystemclasses	Ignore Ant's runtime classes, using only the specified classpath. Only affects the "classname" attribute. Defaults to "false"	No

Parameters specified as nested elements

classpath

Available's classpath attribute is a path-like structure and can also be set via a nested `<classpath>` element.

filepath

Available's filepath attribute is a path-like structure and can also be set via a nested `<filepath>` element.

Examples

```
<available classname="org.whatever.Myclass" property="Myclass.present"/>
```

sets the Myclass.present property to the value "true" if the class org.whatever.Myclass is found in Ant's classpath.

```
<property name="jaxp.jar" value="./lib/jaxp11/jaxp.jar"/>
<available file="${jaxp.jar}" property="jaxp.jar.present"/>
```

sets the jaxp.jar.present property to the value ”true” if the file ./lib/jaxp11/jaxp.jar is found.

```
<available file="/usr/local/lib" type="dir" property="local.lib.present"/>
```

sets the local.lib.present property to the value ”true” if the directory /usr/local/lib is found.

```
...in project ...
<property name="jaxp.jar" value="./lib/jaxp11/jaxp.jar"/>
<path id="jaxp" location="${jaxp.jar}"/>
...in target ...
<available classname="javax.xml.transform.Transformer"
  classpathref="jaxp" property="jaxp11.present"/>
```

sets the jaxp11.present property to the value ”true” if the class javax.xml.transform.Transformer is found in the classpath referenced by jaxp (in this case, ./lib/jaxp11/jaxp.jar).

```
<available property="have.extras" resource="extratasks.properties">
  <classpath>
    <pathelement location="/usr/local/ant/extra.jar/>
  </classpath>
</available>
```

sets the have.extras property to the value ”true” if the resource-file extratasks.properties is found.

5.2.6 Basename

Description

Task to determine the basename of a specified file, optionally minus a specified suffix.

When this task executes, it will set the specified property to the value of the last path element of the specified file. If file is a directory, the basename will be the last directory element. If file is a full-path, relative-path, or simple filename, the basename will be the simple file name, without any directory elements.

Parameters

Attribute	Description	Required
file	The path to take the basename of.	Yes
property	The name of the property to set.	Yes
suffix	The suffix to remove from the resulting basename (specified either with or without the ”.”).	No

Examples

```
<basename property="jar.filename" file="${lib.jarfile}"/>
```

will set jar.filename to myjar.jar, if lib.jarfile is defined as either a full-path filename (eg., /usr/local/lib/myjar.jar), a relative-path filename (eg., lib/myjar.jar), or a simple filename (eg., myjar.jar).

```
<basename property="cmdname" file="D:/usr/local/foo.exe" suffix=".exe"/>
```

will set cmdname to foo.

```
<property environment="env"/>
<basename property="temp.dirname" file="${env.TEMP}"/>
```

will set temp.dirname to the last directory element of the path defined for the TEMP environment variable.

5.2.7 BuildNumber

Description

This is a basic task that can be used to track build numbers.

It will first attempt to read a build number from a file (by default, build.number in the current directory), then set the property build.number to the value that was read in (or to 0, if no such value). It will then increment the number by one and write it back out to the file. (See the PropertyFile task if you need finer control over things such as the property name or the number format.)

Parameters

Attribute	Description	Required
file	The file to read and write the build number from/to.	No; defaults to ”build.number”

Examples

```
<buildnumber/>
```

Read, increment, and write a build number to the default file, build.number.

```
<buildnumber file="mybuild.number"/>
```

Read, increment, and write a build number to the file mybuild.number.

5.2.8 BUnzip2

Description

Expands a file packed using GZip or BZip2.

If dest is a directory the name of the destination file is the same as src (with the ".gz" or ".bz2" extension removed if present). If dest is omitted, the parent dir of src is taken. The file is only expanded if the source file is newer than the destination file, or when the destination file does not exist.

Parameters

Attribute	Description	Required
src	the file to expand.	Yes
dest	the destination file or directory.	No

Examples

```
<gunzip src="test.tar.gz"/>
```

expands test.tar.gz to test.tar

```
<bunzip2 src="test.tar.bz2"/>
```

expands test.tar.bz2 to test.tar

```
<gunzip src="test.tar.gz" dest="test2.tar"/>
```

expands test.tar.gz to test2.tar

```
<gunzip src="test.tar.gz" dest="subdir"/>
```

expands test.tar.gz to subdir/test.tar (assuming subdir is a directory).

5.2.9 BZip2

Description

Packs a file using the GZip or BZip2 algorithm. The output file is only generated if it doesn't exist or the source file is newer.

Parameters

Attribute	Description	Required
src	the file to gzip/bzip.	Yes
zipfile	the destination file.	Yes

Examples

```
<gzip src="test.tar" zipfile="test.tar.gz"/>
```

```
<bzip2 src="test.tar" zipfile="test.tar.bz2"/>
```

5.2.10 Checksum

Description

Generates checksum for files. This task can also be used to perform checksum verifications.

Parameters

Parameters

Attribute	Description	Required
file	The file to generate checksum for.	One of either file or at least one nested fileset element.
algorithm	Specifies the algorithm to be used to compute the checksum. Defaults to "MD5". Other popular algorithms like "SHA" may be used as well.	No
provider	Specifies the provider of the algorithm.	No
fileext	The generated checksum file's name will be the original filename with "." and fileext added to it. Defaults to the algorithm name being used.	No
property	This attribute can mean two different things, it depends on the presence of the verifyproperty attribute. **If you don't set the verifyproperty attribute**, property specifies the name of the property to be set with the generated checksum value. **If you set the verifyproperty attribute**, property specifies the checksum you expect to be generated (the checksum itself, not a name of a property containing the checksum). This cannot be specified when fileext is being used or when the number of files for which checksums is to be generated is greater than 1.	No

Attribute	Description	Required
forceoverwrite	Overwrite existing files even if the destination files are newer. Defaults to "no".	No
verifyproperty	Specifies the name of the property to be set with "true" or "false" depending upon whether the generated checksum matches the existing checksum. When this is set, the generated checksum is not written to a file or property, but rather, the content of the file or property is used to check against the generated checksum.	No
readbuffersize	The size of the buffer)in bytes) to use when reading a file. Defaults to "8192" - you may get a better performance on big files if you increase this value.	No

Parameters specified as nested elements

fileset

FileSets are used to select files to generate checksums for.

Examples

Example 1

```
<checksum file="foo.bar"/>
```

Generates a MD5 checksum for foo.bar and stores the checksum in the destination file foo.bar.MD5. foo.bar.MD5 is overwritten only if foo.bar is newer than itself.

Example 2

```
<checksum file="foo.bar" forceOverwrite="yes"/>
```

Generates a MD5 checksum for foo.bar and stores the checksum in foo.bar.MD5. If foo.bar.MD5 already exists, it is overwritten.

Example 3

```
<checksum file="foo.bar" property="foobarMD5"/>
```

Generates a MD5 checksum for foo.bar and stores it in the Project Property foobarMD5.

Example 4

```
<checksum file="foo.bar" verifyProperty="isMD5ok"/>
```

Generates a MD5 checksum for foo.bar, compares it against foo.bar.MD5 and sets isMD5ok to either true or false, depending upon the result.

Example 5

```
<checksum file="foo.bar" algorithm="SHA" fileext="asc"/>
```

Generates a SHA checksum for foo.bar and stores the checksum in the destination file foo.bar.asc. foo.bar.asc is overwritten only if foo.bar is newer than itself.

Example 6

```
<checksum file="foo.bar" property="${md5}" verifyProperty="isEqual"/>
```

Generates a MD5 checksum for foo.bar, compares it against the value of the property md5, and sets isEqual to either true or false, depending upon the result.

Example 7

```
<checksum>
  <fileset dir=".">
    <include name="foo*"/>
  </fileset>
</checksum>
```

Works just like Example 1, but generates a .MD5 file for every file that begins with the name foo.

Example 8

```
<condition property="isChecksumEqual">
  <checksum>
    <fileset dir=".">
      <include name="foo.bar"/>
    </fileset>
  </checksum>
</condition>
```

Works like Example 4, but only sets isChecksumEqual to true, if the checksum matches - it will never be set to false. This example demonstrates use with the Condition task.

Note: When working with more than one file, if condition and/or verifyproperty is used, the result will be true only if the checksums matched correctly for all files being considered.

5.2.11 Chmod

Description

Changes the permissions of a file or all files inside specified directories. Right now it has effect only under Unix. The permissions are also UNIX style, like the argument for the chmod command.

See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task holds an implicit FileSet and supports all of FileSet's attributes and nested elements directly. More FileSets can be specified using nested ¡fileset¿ elements.

Parameters

Attribute	Description	Required
file	the file or single directory of which the permissions must be changed.	exactly one of the two or nested `<fileset>` elements.
dir	the directory which holds the files whose permissions must be changed.	
perm	the new permissions.	Yes
includes	comma- or space-separated list of patterns of files that must be included.	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No
parallel	process all specified files using a single chmod command. Defaults to true.	No
type	One of file, dir or both. If set to file, only the permissions of plain files are going to be changed. If set to dir, only the directories are considered.	No, default is file

Examples

```
<chmod file="${dist}/start.sh" perm="ugo+rx"/>
```

makes the ”start.sh” file readable and executable for anyone on a UNIX system.

```
    <chmod dir="${dist}/bin" perm="ugo+rx" includes="**/*.sh"/>
```

makes all ".sh" files below ${dist}/bin readable and executable for anyone on a UNIX system.

```
<chmod perm="g+w">
  <fileset dir="shared/sources1">
    <exclude name="**/trial/**"/>
  </fileset>
  <fileset refid="other.shared.sources"/>
</chmod>
```

makes all files below shared/sources1 (except those below any directory named trial) writable for members of the same group on a UNIX system. In addition all files belonging to a FileSet with id other.shared.sources get the same permissions.

5.2.12 Concat

Description

Concatenates a file, or a series of files, to a single file or the console. The destination file will be created if it does not exist, though the the append attribute may be used to alter this behavior.

FileSets and/or FileLists are used to select which files are to be concatenated. There is no singular 'file' attribute to specify a single file to cat – a fileset or filelist must also be used in these cases.

Parameters

Attribute	Description	Required
destfile	The destination file for the concatenated stream. If not specified the console will be used instead.	No
append	Specifies whether or not the file specified by 'destfile' should be overwritten. Defaults to "no".	No
encoding	Specifies the encoding for the input files. Please see http://java.sun.com/products/jdk/1.2/docs/guide/ internat/encoding.doc.html for a list of possible values. Defaults to the platform's default character encoding.	No

Parameters specified as nested elements

fileset

FileSets are used to select files to be concatenated. Note that the order in which the files selected from a fileset are concatenated is not guaranteed. If this is an issue, use multiple filesets or consider using filelists.

filelist

FileLists are used to select files to be concatenated. The file ordering in the files attribute will be the same order in which the files are concatenated.

Examples

Concatenate a string to a file:

```
<concat destfile="README">Hello, World!</concat>
```

Concatenate a series of files to the console:

```
<concat>
  <fileset dir="messages" includes="*important*"/>
</concat>
```

Concatenate a single file, appending if the destination file exists:

```
<concat destfile="NOTES" append="true">
  <filelist dir="notes" files="note.txt"/>
</concat>
```

Concatenate a series of files, overwriting if the destination file exists:

```
<concat destfile="${docbook.dir}/all-sections.xml">
  <filelist dir="${docbook.dir}/sections"
       files="introduction.xml,overview.xml"/>
  <fileset dir="${docbook.dir}"
       includes="sections/*.xml"
       excludes="introduction.xml,overview.xml"/>
</concat>
```

5.2.13 Condition

Description

Sets a property if a certain condition holds true - this is a generalization of Available and Uptodate.

If the condition holds true, the property value is set to true by default; otherwise, the property is not set. You can set the value to something other than the default by specifying the value attribute.

Conditions are specified as nested elements, you must specify exactly one condition.

Parameters

Attribute	Description	Required
property	The name of the property to set.	Yes
value	The value to set the property to. Defaults to ”true”.	No

Parameters specified as nested elements

All conditions to test are specified as nested elements, for a complete list see here.

Examples

```
<condition property="javamail.complete">
  <and>
    <available classname="javax.activation.DataHandler"/>
    <available classname="javax.mail.Transport"/>
  </and>
</condition>
```

sets the property javamail.complete if both the JavaBeans Activation Framework and JavaMail are available in the classpath.

```
<condition property="isMacOsButNotMacOsX">
  <and>
    <os family="mac"/>

    <not>
      <os family="unix"/>

    </not>
  </and>
</condition>
```

sets the property isMacOsButNotMacOsX if the current operating system is MacOS, but not MacOS X - which Ant considers to be in the Unix family as well.

```
<condition property="isSunOSonSparc">
  <os name="SunOS" arch="sparc"/>

</condition>
```

sets the property isSunOSonSparc if the current operating system is SunOS and if it is running on a sparc architecture.

5.2.14 Supported conditions

These are the nested elements that can be used as conditions in the ¡condition¿ and ¡waitfor¿ tasks.

not

The `<not>` element expects exactly one other condition to be nested into this element, negating the result of the condition. It doesn't have any attributes and accepts all nested elements of the condition task as nested elements as well.

and

The `<and>` element doesn't have any attributes and accepts an arbitrary number of conditions as nested elements - all nested elements of the condition task are supported. This condition is true if all of its contained conditions are, conditions will be evaluated in the order they have been specified in the build file.

The `<and>` condition has the same shortcut semantics as the Java && operator, as soon as one of the nested conditions is false, no other condition will be evaluated.

or

The `<or>` element doesn't have any attributes and accepts an arbitrary number of conditions as nested elements - all nested elements of the condition task are supported. This condition is true if at least one of its contained conditions is, conditions will be evaluated in the order they have been specified in the build file.

The `<or>` condition has the same shortcut semantics as the Java —— operator, as soon as one of the nested conditions is true, no other condition will be evaluated.

available

This condition is identical to the Available task, all attributes and nested elements of that task are supported, the property and value attributes are redundant and will be ignored.

uptodate

This condition is identical to the Uptodate task, all attributes and nested elements of that task are supported, the property and value attributes are redundant and will be ignored.

os

Test whether the current operating system is of a given type. Each defined attribute is tested and the result is true only if all the tests succeed.

Attribute	Description	Required
family	The name of the operating system family to expect.	No
name	The name of the operating system to expect.	No
arch	The architecture of the operating system to expect.	No
version	The version of the operating system to expect.	No

Supported values for the family attribute are:

- windows (for all versions of Microsoft Windows)
- dos (for all Microsoft DOS based operating systems including Microsoft Windows and OS/2)

- mac (for all Apple Macintosh systems)
- unix (for all Unix and Unix-like operating systems)
- netware (for Novell NetWare)
- os/2 (for OS/2)
- win9x for Microsoft Windows 95 and 98
- z/os for z/OS and OS/390

equals

Tests whether the two given Strings are identical

Attribute	Description	Required
arg1	First string to test.	Yes
arg2	Second string to test.	Yes
casesensitive	Perform a case sensitive comparision. Default is true.	No
trim	Trim whitespace from arguments before comparing them. Default is false.	No

isset

Test whether a given property has been set in this project.

Attribute	Description	Required
property	The name of the property to test.	Yes

checksum

This condition is identical to the Checksum task, all attributes and nested elements of that task are supported, the property and overwrite attributes are redundant and will be ignored.

http

The http condition checks for a valid response from a web server of the specified url. By default, HTTP responses errors of 400 or greater are viewed as invalid.

Attribute	Description	Required
url	The full URL of the page to request. The web server must return a status code below the value of errorsBeginAt	Yes.
errorsBeginAt	The lowest HTTP response code that signals an error; by default '400'; server errors, not-authorized, not-found and the like are detected	No

socket

The socket condition checks for the existence of a TCP/IP listener at the specified host and port.

Attribute	Description	Required
server	The DNS name or IP address of the server.	Yes.
port	The port number to connect to.	Yes.

filesmatch

Test two files for matching. Nonexistence of either file results in "false". This test does a byte for byte comparision, so test time scales with byte size. NB: if the files are different sizes , one of them is missing or the filenames match the answer is so obvious the detailed test is omitted.

Attribute	Description	Required
file1	First file to test	Yes.
file2	Second file to test	Yes.

contains

Tests whether a string contains another one.

Attribute	Description	Required
string	The string to search in.	Yes
substring	The string to search for.	Yes
casesensitive	Perform a case sensitive comparision. Default is true.	No

istrue

Tests whether a string equals any of the ant definitions of true, that is "true","yes", or "on"

Attribute	Description	Required
Attribute	Description	Required
value	value to test	Yes

```
<istrue value="${someproperty}"/>
<istrue value="false"/>
```

isfalse

Tests whether a string is not true, the negation of ¡istrue¿

Attribute	Description	Required
Attribute	Description	Required
value	value to test	Yes

```
<isfalse value="${someproperty}"/>
<isfalse value="false"/>
```

5.2.15 Copy

Description

Copies a file or FileSet to a new file or directory. By default, files are only copied if the source file is newer than the destination file, or when the destination file does not exist. However, you can explicitly overwrite files with the overwrite attribute.

FileSets are used to select a set of files to copy. To use a `<fileset>`, the todir attribute must be set.

Note: If you employ filters in your copy operation, you should limit the copy to text files. Binary files will be corrupted by the copy operation. This applies whether the filters are implicitly defined by the filter task or explicitly provided to the copy operation as filtersets

Parameters

Attribute	Description	Required
file	The file to copy.	Yes, unless a nested `<fileset>` element is used.
preservelastmodified	Give the copied files the same last modified time as the original source files. (Note: Ignored on Java 1.1)	No; defaults to false.
tofile	The file to copy to.	With the `file` attribute, either `tofile` or `todir` can be used. With nested `<fileset>` elements, if the set of files is greater than 1, or if only the dir attribute is specified in the `<fileset>`, or if the file attribute is also specified, then only todir is allowed.
todir	The directory to copy to.	
overwrite	Overwrite existing files even if the destination files are newer.	No; defaults to false.
filtering	Indicates whether token filtering using the global build-file filters should take place during the copy. Note: Nested `<filterset>` elements will always be used, even if this attribute is not specified, or its value is false (no, or off). No; defaults to false. flatten Ignore the directory structure of the source files, and copy all files into the directory specified by the todir attribute. Note that you can achieve the same effect by using a flatten mapper.	No; defaults to false.
includeEmptyDirs	Copy any empty directories included in the FileSet(s).	No; defaults to true.
failonerror	Log a warning message, but do not stop the build, when the file to copy does not exist. Only meaningful when copying a single file.	No; defaults to true.
verbose	Log the files that are being copied.	No; defaults to false.
encoding	The encoding to assume when filter-copying the files. since Ant 1.5.	No - defaults to default JVM encoding

Parameters specified as nested elements

fileset

FileSets are used to select sets of files to copy. To use a fileset, the todir attribute must be set.

mapper

You can define filename transformations by using a nested mapper element. The default mapper used by `<copy>` is the identity mapper.

filterset

FilterSets are used to replace tokens in files that are copied. To use a FilterSet, use the nested `<filterset>` element.

filterchain

The Copy task supports nested FilterChains.

If `<filterset>` and `<filterchain>` elements are used inside the same `<copy>` task, all `<filterchain>` elements are processed first followed by `<filterset>` elements.

Examples

Copy a single file

```
<copy file="myfile.txt" tofile="mycopy.txt"/>
```

Copy a single file to a directory

```
<copy file="myfile.txt" todir="../some/other/dir"/>
```

Copy a directory to another directory

```
<copy todir="../new/dir">
  <fileset dir="src_dir"/>
</copy>
```

Copy a set of files to a directory

```
<copy todir="../dest/dir">
  <fileset dir="src_dir">
    <exclude name="**/*.java"/>
  </fileset>
</copy>

<copy todir="../dest/dir">
  <fileset dir="src_dir" excludes="**/*.java"/>
</copy>
```

Copy a set of files to a directory, appending .bak to the file name on the fly

```
<copy todir="../backup/dir">
  <fileset dir="src_dir"/>
  <mapper type="glob" from="*" to="*.bak"/>
</copy>
```

Copy a set of files to a directory, replacing @TITLE@ with Foo Bar in all files.

```
<copy todir="../backup/dir">
  <fileset dir="src_dir"/>
  <filterset>
    <filter token="TITLE" value="Foo Bar"/>
  </filterset>
</copy>
```

Unix Note: File permissions are not retained when files are copied; they end up with the default UMASK permissions instead. This is caused by the lack of any means to query or set file permissions in the current Java runtimes. If you need a permission-preserving copy function, use `<exec executable="cp" ... >` instead.

Windows Note: If you copy a file to a directory where that file already exists, but with different casing, the copied file takes on the case of the original. The workaround is to delete the file in the destination directory before you copy it.

5.2.16 Copydir

Deprecated

This task has been deprecated. Use the Copy task instead.

Description

Copies a directory tree from the source to the destination.

It is possible to refine the set of files that are being copied. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes src) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

Parameters

Attribute	Description	Required
Attribute	Description	Required
src	the directory to copy.	Yes
dest	the directory to copy to.	Yes
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No
filtering	indicates whether token filtering should take place during the copy	No
flatten	ignore directory structure of source directory, copy all files into a single directory, specified by the dest attribute (default is false).	No
forceoverwrite	overwrite existing files even if the destination files are newer (default is false).	No

Examples

```
<copydir src="${src}/resources"
         dest="${dist}"
/>
```

copies the directory *src/resourcesto*dist.

```
<copydir src="${src}/resources"
         dest="${dist}"
         includes="**/*.java"
         excludes="**/Test.java"
/>
```

copies the directory *src/resourcesto*dist recursively. All java files are copied, except for files with the name Test.java.

```
<copydir src="${src}/resources"
         dest="${dist}"
         includes="**/*.java"
         excludes="mypackage/test/**"/>
```

copies the directory *src/resourcesto*dist recursively. All java files are copied, except for the files under the mypackage/test directory.

5.2.17 Copyfile

Deprecated

This task has been deprecated. Use the Copy task instead.

Description

Copies a file from the source to the destination. The file is only copied if the source file is newer than the destination file, or when the destination file does not exist.

Parameters

Attribute	Description	Required
Attribute	Description	Required
src	the filename of the file to copy.	Yes
dest	the filename of the file where to copy to.	Yes
filtering	indicates whether token filtering should take place during the copy	No
forceoverwrite	overwrite existing files even if the destination files are newer (default is false).	No

Examples

```
<copyfile src="test.java" dest="subdir/test.java"/>

<copyfile src="${src}/index.html" dest="${dist}/help/index.html"/>
```

5.2.18 Cvs

Description

Handles packages/modules retrieved from a CVS repository.

When doing automated builds, the get task should be preferred over the checkout command, because of speed.

Parameters

Attribute	Description	Required
command	the CVS command to execute. No, default ”	checkout”.
compression	true or false - if set to true, this is the same as compressionlevel=”3” No. Defaults to	false.
compressionlevel	A number between 1 and 9 (corresponding to possible values for CVS' -z#argument). Any other value is treated as compression=”false”	No. Defaults to no compression.
cvsRoot	the CVSROOT variable.	No
cvsRsh	the CVS_RSH variable.	No
dest	the directory where the checked out files should be placed.	No, default is project's basedir.
package	the package/module to check out.	No
tag	the tag of the package/module to check out.	No
date	Use the most recent revision no later than the given date	No
quiet	suppress informational messages.	No, default ”false”
noexec	report only, don't change any files.	No, default to ”false”
output	the file to direct standard output from the command.	No, default output to ANT Log as MSG_INFO.
error	the file to direct standard error from the command.	No, default error to ANT Log as MSG_WARN.
append	whether to append output/error when redirecting to a file.	No, default to ”false”.
port	Port used by CVS to communicate with the server.	No, default port 2401.
passfile	Password file to read passwords from.	No, default file /.cvspass.
failonerror	Stop the build process if the command exits with a return code other than 0. Defaults to false	No

Examples

```
<cvs cvsRoot=":pserver:anoncvs@cvs.apache.org:/home/cvspublic"
     package="ant"
     dest="${ws.dir}"
/>
```

checks out the package/module ”ant” from the CVS repository pointed to by the cvsRoot attribute, and stores the files in ”${ws.dir}”.

```
<cvs dest="${ws.dir}" command="update"/>
```

updates the package/module that has previously been checked out into ”${ws.dir}”.

```
<cvs command="-q diff -u -N" output="patch.txt"/>
```

silently (-q) creates a file called patch.txt which contains a unified (-u) diff which includes new files added via ”cvs add” (-N) and can be used as input to patch. The equivalent, using `<commandline>` elements, is:

```
<cvs output="patch">
    <commandline>
        <argument value="-q"/>
        <argument value="diff"/>
        <argument value="-u"/>
        <argument value="-N"/>
    </commandline>
</cvs>
```

or:

```
<cvs output="patch">
    <commandline>
        <argument line="-q diff -u -N"/>
    </commandline>
</cvs>
```

You may include as many `<commandline>` elements as you like. Each will inherit the failonerror, compression, and other ”global” parameters from the `<cvs>` element.

```
<cvs command="update -A -d"/>
```

Updates from the head of repository ignoring sticky bits (-A) and creating any new directories as necessary (-d).

Note: the text of the command is passed to cvs ”as-is” so any cvs options should appear before the command, and any command options should appear after the command as in the diff example above. See the cvs manual for details, specifically the Guide to CVS commands

5.2.19 CvsChangeLog

Description

Generates an XML-formatted report file of the change logs recorded in a CVS repository.

Parameters

Attribute	Description	Required
dir	The directory from which to run the CVS log command.	No; defaults to ${basedir}.
destfile	The file in which to write the change log report.	Yes
usersfile	Property file that contains name-value pairs mapping user IDs and names that should be used in the report in place of the user ID.	No
daysinpast	Sets the number of days into the past for which the change log information should be retrieved.	No
start	The earliest date from which change logs are to be included in the report.	No
end	The latest date to which change logs are to be included in the report.	No

Parameters specified as nested elements

user

The nested `<user>` element allows you to specify a mapping between a user ID as it appears on the CVS server and a name to include in the formatted report. Anytime the specified user ID has made a change in the repository, the `<author>` tag in the report file will include the name specified in displayname rather than the user ID.

Attribute	Description	Required
displayname	The name to be used in the CVS change log report.	Yes
userid	The userid of the person as it exists on the CVS server.	Yes

Examples

```
<cvschangelog dir="dve/network"
              destfile="changelog.xml"
/>
```

Generates a change log report for all the changes that have been made under the dve/network directory. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
              destfile="changelog.xml"
              daysinpast="10"
/>
```

Generates a change log report for any changes that were made under the dve/network directory in the past 10 days. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
              destfile="changelog.xml"
              start="20 Feb 2002"
              end="20 Mar 2002"
/>
```

Generates a change log report for any changes that were made between February 20, 2002 and March 20, 2002 under the dve/network directory. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
              destfile="changelog.xml"
              start="20 Feb 2002"
/>
```

Generates a change log report for any changes that were made after February 20, 2002 under the dve/network directory. It writes these changes into the file changelog.xml.

```
<cvschangelog dir="dve/network"
              destfile="changelog.xml"/>
     <user displayname="Peter Donald" userid="donaldp"/>
</cvschangelog>
```

Generates a change log report for all the changes that were made under the dve/network directory, substituting the name "Peter Donald" in the `<author>` tags anytime it encounters a change made by the user ID "donaldp". It writes these changes into the file changelog.xml.

Generate Report

Ant includes a basic XSLT stylesheet that you can use to generate a HTML report based on the xml output. The following example illustrates how to generate a HTML report from the XML report.

```
<style in="changelog.xml"
       out="changelog.html"
       style="${ant.home}/etc/changelog.xsl">
  <param name="title" expression="Ant ChangeLog"/>
  <param name="module" expression="ant"/>
  <param name="cvsweb" expression="http://cvs.apache.org/viewcvs/"/>
</style>
```

Sample Output

```
<changelog>
  <entry>
    <date>2002-03-06</date>
    <time>12:00</time>
    <author>Peter Donald</author>
    <file>
      <name>org/apache/myrmidon/build/AntlibDescriptorTask.java</name>
      <revision>1.3</revision>
      <prevrevision>1.2</prevrevision>
    </file>
    <msg><![CDATA[Use URLs directly rather than go via a FIle.
```

This allows temp[lates to be stored inside jar]]¿¡/msg¿

```
  </entry>
</changelog>
```

5.2.20 CVSPass

Description

Adds entries to a .cvspass file. Adding entries to this file has the same affect as a cvs login command.

Parameters

Attribute	Description	Required
cvsroot	the CVS repository to add an entry for.	Yes
password	Password to be added to the password file.	Yes
passfile	Password file to add the entry to.	No, default is /.cvspass.

Examples

```
  <cvspass cvsroot=":pserver:anoncvs@cvs.apache.org:/home/cvspublic"
       password="anoncvs"
  />
```

Adds an entry into the /.cvspass password file.

5.2.21 CvsTagDiff

Description

Generates an XML-formatted report file of the changes between two tags or dates recorded in a CVS repository.

Parameters

Attribute	Description	Required
startTag	The earliest tag from which diffs are to be included in the report.	exactly one of the two.
startDate	The earliest date from which diffs are to be included in the	report.
endTag	The latest tag from which diffs are to be included in the report.	exactly one of the two.
endDate	The latest date from which diffs are to be included in the	report.
destfile	The file in which to write the diff report.	Yes
rootdir	Root directory for the package, if different from the package name.	No

Parameters inherited from the cvs task

Attribute	Description	Required
compression	true, false, or the number 1–9 (corresponding to possible values for CVS `-z#` argument). Any other value is treated as false	No. Defaults to no compression. if passed true, level 3 compression is assumed.
cvsRoot	the CVSROOT variable.	No
cvsRsh	the CVS_RSH variable.	No
package	the package/module to analyze.	Yes
quiet	suppress informational messages.	No, default ”false”
port	Port used by CVS to communicate with the server.	No, default port 2401.
passfile	Password file to read passwords from.	No, default file /.cvspass.
failonerror	Stop the buildprocess if the command exits with a returncode other than 0. Defaults to false	No

Examples

```
<cvstagdiff cvsRoot=":pserver:anoncvs@cvs.apache.org:/home/cvspublic"
              destfile="tagdiff.xml"
              package="ant"
              startTag="ANT_14"
              endTag="ANT_141"
/>
```

Generates a tagdiff report for all the changes that have been made in the ant module between the tags ANT_14 and ANT_141. It writes these changes into the file tagdiff.xml.

```
<cvstagdiff
```

```
              destfile="tagdiff.xml"
              package="ant"
              startDate="2002-01-01"
              endDate="2002-31-01"
/>
```

Generates a tagdiff report for all the changes that have been made in the ant module in january 2002. In this example cvsRoot has not been set. The current cvsRoot will be used (assuming the build is started from a folder stored in cvs. It writes these changes into the file tagdiff.xml.

```
<cvstagdiff
              destfile="tagdiff.xml"
              package="ant"
              rootdir="apache/ant"
              startDate="2002-01-01"
              endDate="2002-31-01"
/>
```

Generates a tagdiff report for all the changes that have been made in the ant module in january 2002, with rootdir indicating that the actual location of the ant module in cvs is apache/ant rather than ant. In this example cvsRoot has not been set. The current cvsRoot will be used (assuming the build is started from a folder stored in cvs. It writes these changes into the file tagdiff.xml.

Generate Report

Ant includes a basic XSLT stylesheet that you can use to generate a HTML report based on the xml output. The following example illustrates how to generate a HTML report from the XML report.

```
        <style in="tagdiff.xml"
               out="tagdiff.html"
               style="${ant.home}/etc/tagdiff.xsl">
          <param name="title" expression="Ant Diff"/>
          <param name="module" expression="ant"/>
          <param name="cvsweb" expression="http://cvs.apache.org/viewcvs/"/>
        </style>
```

Sample Output

```
<?xml version="1.0" encoding="UTF-8"?>
<tagdiff startTag="ANT_14" endTag="ANT_141">
  <entry>
    <file>
      <name>src/main/org/apache/tools/ant/DirectoryScanner.java</name>
      <revision>1.15.2.1</revision>
      <prevrevision>1.15</prevrevision>
    </file>
  </entry>
</tagdiff>
```

5.2.22 Delete

Description

Deletes a single file, a specified directory and all its files and subdirectories, or a set of files specified by one or more FileSets. When specifying a set of files, empty directories are not removed by default. To remove empty directories, use the includeEmptyDirs attribute.

If you use this task to delete temporary files created by editors and it doesn't seem to work, read up on the default exclusion set in Directory-based Tasks, and see the defaultexcludes attribute below.

Parameters

Attribute	Description	Required
Attribute	Description	Required
file	The file to delete, specified as either the simple filename (if the file exists in the current base directory), a relative-path filename, or a full-path filename.	At least one of the two, unless a `<fileset>` is specified.
dir	The directory to delete, including all its files and subdirectories. Note: dir is not used to specify a directory name for file; file and dir are independent of each other. WARNING: Do not set dir to ".", "${basedir}", or the full-pathname equivalent unless you truly intend to recursively remove the entire contents of the current base directory (and the base directory itself, if different from the current working directory).	

Attribute	Description	Required
verbose	Show the name of each deleted file ("true"/"false"). Default is "false" when omitted.	No
quiet	If the specified file or directory does not exist, do not display a diagnostic message (unless Ant has been invoked with the ?verbose or ?debug switches) or modify the exit status to reflect an error. When set to "true", if a file or directory cannot be deleted, no error is reported. This setting emulates the -f option to the Unix rm command. Default is "false". Setting this to "true" implies setting failonerror to "false".	No
failonerror	Controls whether an error (such as a failure to delete a file) stops the build or is merely reported to the screen. Only relevant if quiet is "false". Default is "true".	No
includeEmptyDirs	Set to "true" to delete empty directories when using filesets. Default is "false".	No
includes	Deprecated. Use ¡fileset¿. Comma- or space-separated list of patterns of files that must be deleted. All files are relative to the directory specified in dir.	No
includesfile	Deprecated. Use ¡fileset¿. The name of a file. Each line of this file is taken to be an include pattern	No
excludes	Deprecated. Use ¡fileset¿. Comma- or space-separated list of patterns of files that must be excluded from the deletion list. All files are relative to the directory specified in dir. No files (except default excludes) are excluded when omitted.	No

Attribute	Description	Required
excludesfile	Deprecated. Use ¡fileset¿. The name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	Indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No

Examples

```
<delete file="/lib/ant.jar"/>
```

deletes the file /lib/ant.jar.

```
<delete dir="lib"/>
```

deletes the lib directory, including all files and subdirectories of lib.

```
<delete>
  <fileset dir="." includes="**/*.bak"/>
</delete>
```

deletes all files with the extension .bak from the current directory and any subdirectories.

```
<delete includeEmptyDirs="true">
  <fileset dir="build"/>
</delete>
```

deletes all files and subdirectories of build, including build itself.

5.2.23 Deltree

Deprecated

This task has been deprecated. Use the Delete task instead.

Description

Deletes a directory with all its files and subdirectories.

Parameters

Attribute	Description	Required
dir	the directory to delete.	Yes

Examples

```
<deltree dir="dist"/>
```

deletes the directory dist, including its files and subdirectories.

```
<deltree dir="${dist}"/>
```

deletes the directory ${dist}, including its files and subdirectories.

5.2.24 Dependset

A task to manage arbitrary dependencies between files.

Description

The dependset task compares a set of source files with a set of target files. If any of the source files is more recent than any of the target files, all of the target files are removed.

Source files and target files are specified via nested FileSets and/or nested FileLists. Arbitrarily many source and target filesets/filelists may be specified, but at least one filelist/fileset is required for both sources and targets.

Use a FileSet when you want to use wildcard include or exclude patterns and don't care about missing files. Use a FileList when you want to consider the non-existence of a file as if it were out of date. If there are any non-existing files in any source or target FileList, all target files will be removed.

DependSet is useful to capture dependencies that are not or cannot be determined algorithmically. For example, the ¡style¿ task only compares the source XML file and XSLT stylesheet against the target file to determined whether to restyle the source. Using dependset you can extend this dependency checking to include a DTD or XSD file as well as other stylesheets imported by the main stylesheet.

Parameters

(none)

Parameters Specified as Nested Elements

srcfileset

The nested srcfileset element specifies a FileSet. All files included in this fileset will be compared against all files included in all of the targetfileset filesets and targetfilelist filelists. Multiple srcfileset filesets may be specified.

srcfilelist

The nested srcfilelist element specifies a FileList. All files included in this filelist will be compared against all files included in all of the targetfileset filesets and targetfilelist filelists. Multiple srcfilelist filelists may be specified.

targetfileset

The nested targetfileset element specifies a FileSet. All files included in this fileset will be compared against all files included in all of the srcfileset filesets and sourcefilelist filelists, and if any are older, they are all deleted.

targetfilelist

The nested targetfilelist element specifies a FileList. All files included in this filelist will be compared against all files included in all of the srcfileset filesets and sourcefilelist filelists, and if any are older, they are all deleted.

Examples

```
<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>
```

In this example derived HTML files in the ${output.dir} directory will be removed if any are out-of-date with respect to:

1. the DTD of their source XML files
2. a common DTD (imported by the main DTD)
3. a subordinate XSLT stylesheet (imported by the main stylesheet), or
4. the buildfile

If any of the source files in the above example does not exist, all target files will also be removed. To ignore missing source files instead, use filesets instead of filelists for the source files.

5.2.25 Dirname

Description

Task to determine the directory path of a specified file.

When this task executes, it will set the specified property to the value of the specified file up to, but not including, the last path element. If the specified file is a path that ends in a filename, the filename will be dropped. If the specified file is just a filename, the directory will be the current directory.

Parameters

Attribute	Description	Required
file	The path to take the dirname of.	Yes
property	The name of the property to set.	Yes

Examples

```
<dirname property="antfile.dir" file="${ant.file}"/>
```

will set antfile.dir to the directory path for ${ant.file}.

```
<dirname property="foo.dirname" file="foo.txt"/>
```

will set foo.dirname to the project's basedir.

5.2.26 Ear

Description

An extension of the Jar task with special treatment for files that should end up in an Enterprise Application archive.

(The Ear task is a shortcut for specifying the particular layout of a EAR file. The same thing can be accomplished by using the prefix and fullpath attributes of zipfilesets in a Zip or Jar task.)

The extended zipfileset element from the zip task (with attributes prefix, fullpath, and src) is available in the Ear task.

Parameters

Attribute	Description	Required
destfile	the EAR file to create.	Yes
appxml	The deployment descriptor to use (META-INF/application.xml).	Yes, unless update is set to true
basedir	the directory from which to jar the files.	No
compress	Not only store data but also compress them, defaults to true	No
encoding	The character encoding to use for filenames inside the archive. Defaults to UTF8. It is not recommended to change this value as the created archive will most likely be unreadable for Java otherwise.	No
filesonly	Store only file entries, defaults to false	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
manifest	the manifest file to use.	No
update	indicates whether to update or overwrite the destination file if it already exists. Default is "false".	No
duplicate	behavior when a duplicate file is found. Valid values are "add", "preserve", and "fail". The default value is "add".	No

Nested elements

metainf

The nested metainf element specifies a FileSet. All files included in this fileset will end up in the META-INF directory of the ear file. If this fileset includes a file named MANIFEST.MF, the file is ignored and you will get a warning.

Example

```
<ear destfile="${build.dir}/myapp.ear"
         appxml="${src.dir}/metadata/application.xml">
  <fileset dir="${build.dir}" includes="*.jar,*.war"/>
</ear>
```

5.2.27 Echo

Description

Echoes a message to the current loggers and listeners which means System.out unless overridden. A level can be specified, which controls at what logging level the message is filtered at.

The task can also echo to a file, in which case the option to append rather than overwrite the file is available, and the level option is ignored

Parameters

Attribute	Description	Required
message	the message to echo.	Yes, unless data is included in a character section within this element.
file	the file to write the message to.	No
append	Append to an existing file?	No - default is false.
level	Control the level at which this message is reported. One of "error", "warning", "info", "verbose", "debug"	No - default is "warning".

Examples

```
<echo message="Hello, world"/>
```

```
<echo>This is a longer message stretching over
two lines.
</echo>
```

```
<echo>
This is a longer message stretching over
three lines; the first line is a blank
</echo>
```

As XML parsers are wont to do, the first newline in the text element has been included in the text.

```
<echo message="Deleting drive C:" level="debug"/>
```

A message which only appears in -debug mode.

```
<echo level="error">
Imminent failure in the antimatter containment facility.
Please withdraw to safe location at least 50km away.
</echo>
```

A message which appears even in -quiet mode.

```
<echo file="runner.csh" append="false">#\!/bin/tcsh
java-1.3.1 -mx1024m ${project.entrypoint} $$*
```

Generate a shell script by echoing to a file. Note the use of a double $ symbol to stop Ant filtering out the single $ during variable expansion

5.2.28 Exec

Description

Executes a system command. When the os attribute is specified, then the command is only executed when Ant is run on one of the specified operating systems.

Cygwin Users

In general the `<exec>` task will not understand paths such as /bin/sh for the executable parameter. This is because the Java VM in which Ant is running is a Windows executable and is not aware of Cygwin conventions.

Parameters

Attribute	Description	Required
command	the command to execute with all command line arguments. deprecated, use executable and nested ¡arg¿ elements instead.	Exactly one of the two.
executable	the command to execute without any command line	arguments.
dir	the directory in which the command should be executed.	No
os	list of Operating Systems on which the command may be executed. If the current OS's name is contained in this list, the command will be executed. The OS's name is determined by the Java Virtual machine and is set in the ”os.name” system property.	No
output	the file to which the output of the command should be redirected.	No
append	whether output should be appended to or overwrite an existing file. Defaults to false.	No
outputproperty	the name of a property in which the output of the command should be stored.	No
resultproperty	the name of a property in which the return code of the command should be stored. Only of interest if failonerror=false	No
timeout	Stop the command if it doesn't finish within the specified time (given in milliseconds).	No
failonerror	Stop the buildprocess if the command exits with a returncode other than 0. Defaults to false	No
failifexecutionfails	Stop the build if we can't start the program. Defaults to true.	No
newenvironment	Do not propagate old environment when new environment variables are specified.	No, default is false
vmlauncher	Run command using the Java VM's execution facilities where available. If set to false the underlying OS's shell, either directly or through the antRun scripts, will be used. Under some operating systems, this gives access to facilities not normally available through the VM including, under Windows, being able to execute scripts, rather than their associated interpreter. If you want to specify the name of the executable as a relative path to the directory given by the dir attribute, it may become necessary to set vmlauncher to false as well.	No, default is true

Examples

```
<exec dir="${src}" executable="cmd.exe" os="Windows 2000" output="dir.txt">
  <arg line="/c dir"/>
</exec>
```

Parameters specified as nested elements

arg

Command line arguments should be specified as nested ¡arg¿ elements. See Command line arguments.

env

It is possible to specify environment variables to pass to the system command via nested `<env>` elements.

Attribute	Description	Required
key	The name of the environment variable.	Yes
value	The literal value for the environment variable.	Exactly one of these.
path	The value for a PATH like environment variable. You can use ; or : as separators and Ant will convert it to the platform's local conventions.	
hline file	The value for the environment variable. Will be replaced by the absolute filename of the file by Ant.	

Errors and return codes

By default the return code of a `<exec>` is ignored; when you set failonerror="true" then any non zero response is treated as an error. Alternatively, you can set resultproperty to the name of a property and have it assigned to the result code (barring immutability, of course).

If the attempt to start the program fails with an OS dependent error code, then `<exec>` halts the build unless failifexecutionfails is set. You can use that to run a program if it exists, but otherwise do nothing.

What do those error codes mean? Well, they are OS dependent. On Windows boxes you have to look in include
error.h in your windows compiler or wine files; error code 2 means 'no such program', which usually means it is not on the path. Any time you see such an error from any ant task, it is usually not an ant bug, but some configuration problem on your machine.

Examples

```
<exec executable="emacs">
```

```
  <env key="DISPLAY" value=":1.0"/>
</exec>
```

starts emacs on display 1 of the X Window System.

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

adds ${basedir}/bin to the PATH of the system command.

Note: Although it may work for you to specify arguments using a simple arg-element and separate them by spaces it may fail if you switch to a newer version of the JDK. JDK ¡ 1.2 will pass these as separate arguments to the program you are calling, JDK ¿= 1.2 will pass them as a single argument and cause most calls to fail.

Note2: If you are using Ant on Windows and a new DOS-Window pops up for every command which is executed this may be a problem of the JDK you are using. This problem may occur with all JDK's ¡ 1.2.

Timeouts: If a timeout is specified, when it is reached the sub process is killed and a message printed to the log. The return value of the execution will be "-1", which will halt the build if failonerror=true, but be ignored otherwise.

5.2.29 Fail

Description

Exits the current build (just throwing a BuildException), optionally printing additional information.

The message of the Exception can be set via the message attribute or character data nested into the element.

Parameters

Attribute	Description	Required
Attribute	Description	Required
message	A message giving further information on why the build exited	No
if	Only fail if a property of the given name exists in the current project	No
unless	Only fail if a property of the given name doesn't exist in the current project	No

Examples

```
  <fail/>
```

will exit the current build with no further information given.

```
BUILD FAILED

build.xml:4: No message
```

```
  <fail message="Something wrong here."/>
```

will exit the current build and print something like the following to wherever your output goes:

```
BUILD FAILED

build.xml:4: Something wrong here.
```

```
  <fail>Something wrong here.</fail>
```

will give the same result as above.

5.2.30 Filter

Description

Sets a token filter for this project or read multiple token filter from an input file and sets these as filters. Token filters are used by all tasks that perform file copying operations through the Project commodity methods.

Note 1: the token string must not contain the separators chars (@).

Note 2: Either token and value attributes must be provided, or only the filtersfile attribute.

Parameters

Attribute	Description	Required
Attribute	Description	Required
token	the token string without @	Yes*
value	the string that should be put to replace the token when the file is copied	Yes*
filtersfile	The file from which the filters must be read. This file must be a formatted as a property file.	Yes*

* see notes 1 and 2 above parameters table.

Examples

```
  <filter token="year" value="2000"/>
  <copy todir="${dest.dir}" filtering="true">
    <fileset dir="${src.dir}"/>
  </copy>
```

will copy recursively all the files from the src.dir directory into the dest.dir directory replacing all the occurrences of the string @year@ with 2000.

```
<filter filtersfile="deploy_env.properties"/>
```

will read all property entries from the deploy_env.properties file and set these as filters.

5.2.31 FixCRLF

Description

Adjusts a text file to local conventions.

The set of files to be adjusted can be refined with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. Patterns provided through the includes or includesfile attributes specify files to be included. Patterns provided through the exclude or excludesfile attribute specify files to be excluded. Additionally, default exclusions can be specified with the defaultexcludes attribute. See the section on directory based tasks, for details of file inclusion/exclusion patterns and their usage.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes srcdir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

The output file is only written if it is a new file, or if it differs from the existing file. This prevents spurious rebuilds based on unchanged files which have been regenerated by this task.

Parameters

Attribute	Description	Required
Attribute	Description	Required
srcDir	Where to find the files to be fixed up.	Yes
destDir	Where to place the corrected files. Defaults to srcDir (replacing the original file)	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No

Attribute	Description	Required
eol	Specifies how end-of-line (EOL) characters are to be handled. The EOL characters are CR, LF and the pair CRLF. Valid values for this property are: • asis: leave EOL characters alone • cr: convert all EOLs to a single CR • lf: convert all EOLs to a single LF • crlf: convert all EOLs to the pair CRLF Default is based on the platform on which you are running this task. For Unix platforms, the default is ”lf”. For DOS based systems (including Windows), the default is ”crlf”. For Mac OS, the default is ”cr”. This is the preferred method for specifying EOL. The ”cr” attribute (see below) is now deprecated. N.B.: One special case is recognized. The three characters CR-CR-LF are regarded as a single EOL. Unless this property is specified as ”asis”, this sequence will be converted into the specified EOL type.	No
cr	Deprecated. Specifies how CR characters are to be handled at end-of-line (EOL). Valid values for this property are: • asis: leave EOL characters alone. • add: add a CR before any single LF characters. The intent is to convert all EOLs to the pair CRLF. • remove: remove all CRs from the file. The intent is to convert all EOLs to a single LF. Default is based on the platform on which you are running this task. For Unix platforms, the default is ”remove”. For DOS based systems (including Windows), the default is ”add”. N.B.: One special case is recognized. The three characters CR-CR-LF are regarded as a single EOL. Unless this property is specified as ”asis”, this sequence will be converted into the specified EOL type.	

Attribute	Description	Required
javafiles	Used only in association with the ”tab” attribute (see below), this boolean attribute indicates whether the fileset is a set of java source files (”yes”/”no”). Defaults to ”no”. See notes in section on ”tab”.	No
tab	Specifies how tab characters are to be handled. Valid values for this property are: • add: convert sequences of spaces which span a tab stop to tabs • asis: leave tab and space characters alone • remove: convert tabs to spaces Default for this parameter is ”asis”. N.B.: When the attribute ”javafiles” (see above) is ”true”, literal TAB characters occurring within Java string or character constants are never modified. This functionality also requires the recognition of Java-style comments. N.B.: There is an incompatibility between this and the previous version in the handling of white space at the end of lines. This version does not remove trailing whitespace on lines.	No
tablength	TAB character interval. Valid values are between 2 and 80 inclusive. The default for this parameter is 8.	No
eof	Specifies how DOS end of file (control-Z) characters are to be handled. Valid values for this property are: • add: ensure that there is an EOF character at the end of the file • asis: leave EOF characters alone • remove: remove any EOF character found at the end Default is based on the platform on which you are running this task. For Unix platforms, the default is remove. For DOS based systems (including Windows), the default is asis.	No

Attribute	Description	Required
encoding	The encoding of the files	No - defaults to default JVM encoding

Examples

```
<fixcrlf srcdir="${src}"
     eol="lf"
     eof="remove"
     includes="**/*.sh"
/>
```

Replaces EOLs with LF characters and removes eof characters from the shell scripts. Tabs and spaces are left as is.

```
<fixcrlf srcdir="${src}"
     eol="crlf"
     includes="**/*.bat"
/>
```

Replaces all EOLs with cr-lf pairs in the batch files. Tabs and spaces are left as is. EOF characters are left alone if run on DOS systems, and are removed if run on Unix systems.

```
<fixcrlf srcdir="${src}"
     tab="add"
     includes="**/Makefile"
/>
```

Sets EOLs according to local OS conventions, and converts sequences of spaces and tabs to the minimal set of spaces and tabs which will maintain spacing within the line. Tabs are set at 8 character intervals. EOF characters are left alone if run on DOS systems, and are removed if run on Unix systems. Many versions of make require tabs prior to commands.

```
<fixcrlf srcdir="${src}"
     tab="remove"
     tablength="3"
     eol="lf"
     javafiles="yes"
     includes="**/*.java"
/>
```

Converts all EOLs in the included java source files to a single LF. Replace all TAB characters except those in string or character constants with spaces, assuming a tab width of 3. If run on a unix system, any CTRL-Z EOF characters at the end of the file are removed. On DOS/Windows, any such EOF characters will be left untouched.

```
<fixcrlf srcdir="${src}"
     tab="remove"
     includes="**/README*"
/>
```

Sets EOLs according to local OS conventions, and converts all tabs to spaces, assuming a tab width of 8. EOF characters are left alone if run on DOS systems, and are removed if run on Unix systems. You never know what editor a user will use to browse README's.

5.2.32 GenKey

Description

Generates a key in keystore. This task needs Java1.2 or later

Parameters

Attribute	Description	Required
alias	the alias to add under	Yes.
storepass	password for keystore integrity. Must be at least 6 characters long	Yes.
keystore	keystore location	No
storetype	keystore type	No
keypass	password for private key (if different)	No
sigalg	the algorithm to use in signing	No
keyalg	the method to use when generating name-value pair	No
verbose	(true — false) verbose output when signing	No
dname	The distinguished name for entity Yes if dname element	unspecified
validity	(integer) indicates how many days certificate is valid	No
keysize	(integer) indicates the size of key generated	No

Alternatively you can specify the distinguished name by creating a subelement named dname and populating it with param elements that have a name and a value. When using the subelement it is automatically encoded properly and commas (",") are replaced with " ".

The following two examples are identical:

Examples

```
<genkey alias="apache-group" storepass="secret"
     dname="CN=Ant Group, OU=Jakarta Division, O=Apache.org, C=US"/>

<genkey alias="apache-group" storepass="secret" >
  <dname>
    <param name="CN" value="Ant Group"/>
    <param name="OU" value="Jakarta Division"/>
    <param name="O"  value="Apache.Org"/>
    <param name="C"  value="US"/>
  </dname>
</genkey>
```

5.2.33 Get

Description

Gets a file from a URL. When the verbose option is ”on”, this task displays a ’.’ for every 100 Kb retrieved. Any URL schema supported by the runtime is valid here, including http:, ftp: and jar:; https: is only valid if the appropriate support is added to the pre-1.4 Java runtimes.

This task should be preferred above the CVS task when fetching remote content. CVS is significantly slower than loading a compressed archive compared to http/ftp.

The usetimestamp option enables you to control downloads so that the remote file is only fetched if newer than the local copy. If there is no local copy, the download always takes place. When a file is downloaded, the timestamp of the downloaded file is set to the remote timestamp, if the JVM is Java1.2 or later. NB: This timestamp facility only works on downloads using the HTTP protocol.

A username and password can be specified, in which case basic ’slightly encoded plain text’ authentication is used. This is only a secure authentication mechanism over an HTTPS link.

If you need to go through a firewall, use ¡setproxy¿ to set up the proxy first.

Parameters

Attribute	Description	Required
src	the URL from which to retrieve a file.	Yes
dest	the file where to store the retrieved file.	Yes
verbose	show verbose progress information (”on”/”off”).	No; default ”false”
ignoreerrors	Log errors but don’t treat as fatal.	No; default ”false”
usetimestamp	conditionally download a file based on the timestamp of the local copy. HTTP only	No; default ”false”
username	username for ’BASIC’ http authentication	if password is set
password	password:	required if username is set

Examples

```
<get src="http://ant.apache.org/" dest="help/index.html"/>
```

Gets the index page of http://ant.apache.org/, and stores it in the file help/index.html.

```
<get src="http://jakarta.apache.org/builds/tomcat/nightly/ant.zip"
  dest="optional.jar"
  verbose="true"
  usetimestamp="true"/>
```

Gets the nightly ant build from the tomcat distribution, if the local copy is missing or out of date. Uses the verbose option for progress information.

```
<get src="https://insecure-bank.org/statement/user=1214"
  dest="statement.html"
  username="1214";
  password="secret"/>
```

Fetches some file from a server with access control. Because https is being used the fact that basic auth sends passwords in plaintext is moot.

5.2.34 GUnzip/BUnzip2

Description

Expands a file packed using GZip or BZip2.

If dest is a directory the name of the destination file is the same as src (with the ".gz" or ".bz2" extension removed if present). If dest is omitted, the parent dir of src is taken. The file is only expanded if the source file is newer than the destination file, or when the destination file does not exist.

Parameters

Attribute	Description	Required
src	the file to expand.	Yes
dest	the destination file or directory.	No

Examples

```
<gunzip src="test.tar.gz"/>
```

expands test.tar.gz to test.tar

```
<bunzip2 src="test.tar.bz2"/>
```

expands test.tar.bz2 to test.tar

```
<gunzip src="test.tar.gz" dest="test2.tar"/>
```

expands test.tar.gz to test2.tar

```
<gunzip src="test.tar.gz" dest="subdir"/>
```

expands test.tar.gz to subdir/test.tar (assuming subdir is a directory).

5.2.35 GZip/BZip2

Description

Packs a file using the GZip or BZip2 algorithm. The output file is only generated if it doesn't exist or the source file is newer.

Parameters

Attribute	Description	Required
src	the file to gzip/bzip.	Yes
zipfile	the destination file.	Yes

Examples

```
<gzip src="test.tar" zipfile="test.tar.gz"/>

<bzip2 src="test.tar" zipfile="test.tar.bz2"/>
```

5.2.36 Input

Description

Allows user interaction during the build process by prompting for input. To do so, it uses the configured InputHandler.

The prompt can be set via the message attribute or as character data nested into the element.

Optionally a set of valid input arguments can be defined via the validargs attribute. Input task will no accept value that don't match one of the predefined.

Optionally a property can be created from the value entered by the user. This property can then be used during the following build run. Input behaves according to property task which means that existing properties cannot be overriden.

Parameters

Attribute	Description	Required
message	the Message which gets displayed to the user during the build run.	No
validargs	comma separated String containing valid input arguments. If set, input task will reject any input not defined here. Validargs are compared case sensitive. If you want 'a' and 'A' to be accepted you will need to define both arguments within validargs.	No
addproperty	the name of a property to be created from input. Behaviour is equal to property task which means that existing properties cannot be overriden.	No

Examples

```
<input/>
```

Will pause the build run until return key is pressed when using the default InputHandler, the concrete behavior is defined by the InputHandler implementation you use.

```
<input>Press Return key to continue...</input>
```

Will display the message "Press Return key to continue..." and pause the build run until return key is pressed (again, the concrete behavior is implementation dependent).

```
<input
  message="Press Return key to continue..."
/>
```

Will display the message "Press Return key to continue..." and pause the build run until return key is pressed (see above).

```
<input
  message="All data is going to be deleted from DB continue (y/n)?"
  validargs="y,n"
  addproperty="do.delete"
/>
<condition property="do.abort">
  <equals arg1="n" arg2="${do.delete}"/>
</condition>
<fail if="do.abort">Build aborted by user.</fail>
```

Will display the message "All data is going to be deleted from DB continue (y/n)?" and require 'y' to continue build or 'n' to exit build with following message "Build aborted by user.".

```
<input
  message="Please enter db-username:"
  addproperty="db.user"
/>
```

Will display the message "Please enter db-username:" and set the property db.user to the value entered by the user.

5.2.37 Jar

Description

Jars a set of files.

The basedir attribute is the reference directory from where to jar.

Note that file permissions will not be stored in the resulting jarfile.

It is possible to refine the set of files that are being jarred. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes basedir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

You can also use nested file sets for more flexibility, and specify multiple ones to merge together different trees of files into one JAR. The extended fileset and groupfileset attributes from the zip task are also available in the jar task. See the Zip task for more details and examples.

If the manifest is omitted, a simple one will be supplied by Ant.

The update parameter controls what happens if the JAR file already exists. When set to yes, the JAR file is updated with the files specified. When set to no (the default) the JAR file is overwritten. An example use of this is provided in the Zip task documentation. Please note that ZIP files store file modification times with a granularity of two seconds. If a file is less than two seconds newer than the entry in the archive, Ant will not consider it newer.

(The Jar task is a shortcut for specifying the manifest file of a JAR file. The same thing can be accomplished by using the fullpath attribute of a zipfileset in a Zip task. The one difference is that if the manifest attribute is not specified, the Jar task will include an empty one for you.)

Manifests are processed by the Jar task according to the Jar file specification. Note in particular that this may result in manifest lines greater than 72 bytes being wrapped and continued on the next line.

Parameters

Attribute	Description	Required
Attribute	Description	Required
destfile	the JAR file to create.	Yes
basedir	the directory from which to jar the files.	No
compress	Not only store data but also compress them, defaults to true	No
encoding	The character encoding to use for filenames inside the archive. Defaults to UTF8. It is not recommended to change this value as the created archive will most likely be unreadable for Java otherwise.	No

Attribute	Description	Required
filesonly	Store only file entries, defaults to false	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No
manifest	the manifest file to use. This can be either the location of a manifest, or the name of a jar added through a fileset. If its the name of an added jar, the task expects the manifest to be in the jar at META-INF/MANIFEST.MF	No
update	indicates whether to update or overwrite the destination file if it already exists. Default is ”false”.	No
whenempty	behavior when no files match. Valid values are ”fail”, ”skip”, and ”create”. Default is ”skip”.	No
duplicate	behavior when a duplicate file is found. Valid values are ”add”, ”preserve”, and ”fail”. The default value is ”add”.	No
index	whether to create an index list to speed up classloading. This is a JDK 1.3+ specific feature. Defaults to false.	No
manifestencoding	The encoding used to read the JAR manifest, when a manifest file is specified.	No, defaults to the platform encoding.

Nested elements

metainf

The nested metainf element specifies a FileSet. All files included in this fileset will end up in the META-INF directory of the jar file. If this fileset includes a file named MANIFEST.MF, the file is ignored and you will get a warning.

manifest

The manifest nested element allows the manifest for the Jar file to be provided inline in the build file rather than in an external file. This element is identical to the manifest task, but the file and mode attributes must be omitted.

If both an inline manifest and an external file are both specified, the manifests are merged.

When using inline manifests, the Jar task will check whether the build file is more recent that the Jar file when deciding whether to rebuild the Jar. This will not take into account property file changes which may affect the resulting Jar.

Examples

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

jars all files in the *build/classesdirectoryintoafilecalledapp.jarinthe*dist/lib directory.

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

jars all files in the *build/classesdirectoryintoafilecalledapp.jarinthe*dist/lib directory. Files with the name Test.class are excluded.

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

jars all files in the *build/classesdirectoryintoafilecalledapp.jarinthe*dist/lib directory. Only files under the directory mypackage/test are used, and files with the name Test.class are excluded.

```
<jar destfile="${dist}/lib/app.jar">
  <fileset dir="${build}/classes"
           excludes="**/Test.class"
  />
  <fileset dir="${src}/resources"/>
</jar>
```

jars all files in the *build/classesdirectoryandalsointhe*\${*src*}*/resourcesdirectorytogetherintoafilecalledapp.jari*
and ${src}/resources/mypackage/image.gif, they will appear in the same directory in the JAR (and thus be considered in the same package by Java).

```
<jar destfile="test.jar" basedir=".">
  <include name="build"/>
  <manifest>
    <attribute name="Built-By" value="${user.name}"/>
    <section name="common/class1.class">
      <attribute name="Sealed" value="false"/>
    </section>
  </manifest>
</jar>
```

This is an example of an inline manifest specification. Note that the Built-By attribute will take the value of the Ant property ${user.name}. The manifest produced by the above would look like this:

```
Manifest-Version: 1.0
Built-By: conor
Created-By: Apache Ant 1.5alpha

Name: common/class1.class
Sealed: false
```

5.2.38 Java

Description

Executes a Java class within the running (Ant) VM or forks another VM if specified.

If odd things go wrong when you run this task, set fork="true" to use a new JVM.

Parameters

Attribute	Description	Required
Attribute	Description	Required
classname	the Java class to execute.	Either jar or classname
jar	the location of the jar file to execute (must have a Main-Class entry in the manifest). Fork must be set to true if this option is selected.	Either jar or classname
args	the arguments for the class that is executed. deprecated, use nested ¡arg¿ elements instead.	No
classpath	the classpath to use.	No
classpathref	the classpath to use, given as reference to a PATH defined elsewhere.	No
fork	if enabled triggers the class execution in another VM (disabled by default)	No
jvm	the command used to invoke the Java Virtual Machine, default is 'java'. The command is resolved by java.lang.Runtime.exec(). Ignored if fork is disabled.	No
jvmargs	the arguments to pass to the forked VM (ignored if fork is disabled). deprecated, use nested ¡jvmarg¿ elements instead.	No

Attribute	Description	Required
maxmemory	Max amount of memory to allocate to the forked VM (ignored if fork is disabled)	No
failonerror	Stop the buildprocess if the command exits with a returncode other than 0. Default is ”false”	No
dir	The directory to invoke the VM in. (ignored if fork is disabled)	No
output	Name of a file to write the output to.	No
append	whether output should be appended to or overwrite an existing file. Defaults to false.	No
newenvironment	Do not propagate old environment when new environment variables are specified. Default is ”false” (ignored if fork is disabled).	No
timeout	Stop the command if it doesn't finish within the specified time (given in milliseconds). It is highly recommended to use this feature only if fork is enabled.	No

Parameters specified as nested elements

arg and jvmarg

Use nested `<arg>` and `<jvmarg>` elements to specify arguments for the Java class and the forked VM respectively. See Command line arguments.

sysproperty

Use nested `<sysproperty>` elements to specify system properties required by the class. These properties will be made available to the VM during the execution of the class (either ANT's VM or the forked VM). The attributes for this element are the same as for environment variables.

classpath

Java's classpath attribute is a PATH like structure and can also be set via a nested classpath element.

env

It is possible to specify environment variables to pass to the forked VM via nested env elements. See the description in the section about exec

Settings will be ignored if fork is disabled.

Examples

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

Run a class in this JVM with a new jar on the classpath

```
   <java jar="dist/test.jar"
      fork="true"
      failonerror="true"
      maxmemory="128m"
      >
    <arg value="-h"/>
    <classpath>
      <pathelement location="dist/test.jar"/>
      <pathelement path="${java.class.path}"/>
    </classpath>
  </java>
```

Run the jar using the manifest supplied entry point, forking (as required), and with a maximum memory of 128MB. Any non zero return code breaks the build.

```
<java classname="test.Main"/>

<java classname="test.Main"
      fork="yes" >
  <sysproperty key="DEBUG" value="true"/>
  <arg value="-h"/>
  <jvmarg value="-Xrunhprof:cpu=samples,file=log.txt,depth=3"/>
</java>
```

Note: you can not specify the (highly deprecated) MSJVM, ”jview.exe” as the JVM, as it takes different parameters for other JVMs, That JVM can be started from `<exec>` if required.

5.2.39 Javac

Description

Compiles a Java source tree.

The source and destination directory will be recursively scanned for Java source files to compile. Only Java files that have no corresponding .class file or where the class file is older than the .java file will be compiled.

Note: Ant uses only the names of the source and class files to find the classes that need a rebuild. It will not scan the source and therefore will have no knowledge about nested classes, classes that are named different from the source file, and so on. See the ¡depend¿ task for dependency checking based on other than just existence/modification times.

When the source files are part of a package, the directory structure of the source tree should follow the package hierarchy.

It is possible to refine the set of files that are being compiled. This can be done with the includes, includesfile, excludes, and excludesfile attributes. With the includes or includesfile attribute, you specify the files you want to have included. The exclude or excludesfile attribute is used to specify the files you want to have excluded. In both cases, the list of files can be specified by either the filename, relative to the directory(s) specified in the srcdir attribute or nested ¡src¿ element(s), or by using wildcard patterns. See the section on directory-based tasks, for information on how the inclusion/exclusion of files works, and how to write wildcard patterns.

It is possible to use different compilers. This can be specified by either setting the global build.compiler property, which will affect all `<javac>` tasks throughout the build, or by setting the compiler attribute, specific to the current `<javac>` task. Valid values for either the build.compiler property or the compiler attribute are:

- classic (the standard compiler of JDK 1.1/1.2) ? javac1.1 and javac1.2 can be used as aliases.
- modern (the standard compiler of JDK 1.3/1.4) ? javac1.3 and javac1.4 can be used as aliases.
- jikes (the Jikes compiler).
- jvc (the Command-Line Compiler from Microsoft's SDK for Java / Visual J++) ? microsoft can be used as an alias.
- kjc (the kopi compiler).
- gcj (the gcj compiler from gcc).
- sj (Symantec java compiler) ? symantec can be used as an alias.
- extJavac (run either modern or classic in a JVM of its own).

The default is javac1.x with x depending on the JDK version you use while you are running Ant. If you wish to use a different compiler interface than those supplied, you can write a class that implements the CompilerAdapter interface (package org.apache.tools.ant.taskdefs.compilers). Supply the full classname in the build.compiler property or the compiler attribute.

The fork attribute overrides the build.compiler property or compiler attribute setting and expects a JDK1.1 or higher to be set in JAVA_HOME.

You can also use the compiler attribute to tell Ant which JDK version it shall assume when it puts together the command line switches - even if you set fork="true". This is useful if you want to run the compiler of JDK 1.1 while you current JDK is 1.2+. If you use compiler="javac1.1" and (for example) depend="true" Ant will use the command line switch -depend instead of -Xdepend.

This task will drop all entries that point to non-existent files/directories from the classpath it passes to the compiler.

Windows Note:When the modern compiler is used in unforked mode on Windows, it locks up the files present in the classpath of the ¡javac¿ task, and does not release them. The side effect of this is that you will not be able to delete or move those files later on in the build. The workaround is to fork when invoking the compiler.

Parameters

Attribute	Description	Required
Attribute	Description	Required
srcdir	Location of the java files. (See the note below.)	Yes, unless nested `<src>` elements are present.
destdir	Location to store the class files.	No
includes	Comma- or space-separated list of files (may be specified using wildcard patterns) that must be included; all .java files are included when omitted.	No
includesfile	The name of a file that contains a list of files to include (may be specified using wildcard patterns).	No
excludes	Comma- or space-separated list of files (may be specified using wildcard patterns) that must be excluded; no files (except default excludes) are excluded when omitted.	No
excludesfile	The name of a file that contains a list of files to exclude (may be specified using wildcard patterns).	No
classpath	The classpath to use.	No
sourcepath	The sourcepath to use; defaults to the value of the srcdir attribute (or nested ¡src¿ elements). To suppress the sourcepath switch, use sourcepath="".	No
bootclasspath	Location of bootstrap class files.	No
classpathref	The classpath to use, given as a reference to a path defined elsewhere.	No
sourcepathref	The sourcepath to use, given as a reference to a path defined elsewhere.	No
bootclasspathref	Location of bootstrap class files, given as a reference to a path defined elsewhere.	No
extdirs	Location of installed extensions.	No
encoding	Encoding of source files. (Note: gcj doesn't support this option yet.)	No
nowarn	Indicates whether the -nowarn switch should be passed to the compiler; defaults to off.	No

Attribute	Description	Required
debug	Indicates whether source should be compiled with debug information; defaults to off. If set to off, -g:none will be passed on the command line for compilers that support it (for other compilers, no command line argument will be used). If set to true, the value of the debuglevel attribute determines the command line argument.	No
debuglevel	Keyword list to be appended to the -g command-line switch. This will be ignored by all implementations except modern and classic(ver ¿= 1.2). Legal values are none or a comma-separated list of the following keywords: lines, vars, and source. If debuglevel is not specified, by default, nothing will be appended to -g. If debug is not turned on, this attribute will be ignored.	No
optimize	Indicates whether source should be compiled with optimization; defaults to off.	No
deprecation	Indicates whether source should be compiled with deprecation information; defaults to off.	No
target	Generate class files for specific VM version (e.g., 1.1 or 1.2). Note that the default value depends on the JVM that is running Ant. In particular, if you use JDK 1.4 the generated classes will not be usable for a 1.1 Java VM unless you explicitly set this attribute to the value 1.1 (which is the default value for JDK 1.1 to 1.3).	No
verbose	Asks the compiler for verbose output.	No
depend	Enables dependency-tracking for compilers that support this (jikes and classic).	No
includeAntRuntime	Whether to include the Ant run-time libraries in the classpath; defaults to yes.	No
includeJavaRuntime	Whether to include the default run-time libraries from the executing VM in the classpath; defaults to no.	No
fork	Whether to execute javac using the JDK compiler externally; defaults to no.	No
executable	Complete path to the javac executable to use in case of fork=”yes”. Defaults to the compiler of the Java version that is currently running Ant. Ignored if fork=”no”	No

Attribute	Description	Required
memoryInitialSize	The initial size of the memory for the underlying VM, if javac is run externally; ignored otherwise. Defaults to the standard VM memory setting. (Examples: 83886080, 81920k, or 80m)	No
memoryMaximumSize	The maximum size of the memory for the underlying VM, if javac is run externally; ignored otherwise. Defaults to the standard VM memory setting. (Examples: 83886080, 81920k, or 80m)	No
failonerror	Indicates whether the build will continue even if there are compilation errors; defaults to true.	No
source	Value of the -source command-line switch; will be ignored by all implementations except javac1.4 (or modern when Ant is not running in a 1.3 VM) and jikes. If you use this attribute together with jikes, you must make sure that your version of jikes supports the -source switch. Legal values are 1.3 and 1.4 – by default, no -source argument will be used at all.	No
compiler	The compiler implementation to use. If this attribute is not set, the value of the build.compiler property, if set, will be used. Otherwise, the default compiler for the current VM will be used. (See the above list of valid compilers.)	No
listfiles	Indicates whether the source files to be compiled will be listed; defaults to no.	No

Parameters specified as nested elements

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes srcdir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements. src, classpath, sourcepath, bootclasspath and extdirs

`<javac>`'s srcdir, classpath, sourcepath, bootclasspath, and extdirs attributes are path-like structures and can also be set via nested `<src>`, `<classpath>`, `<sourcepath>`, `<bootclasspath>` and `<extdirs>` elements, respectively.

compilerarg

You can specify additional command line arguments for the compiler with nested `<compilerarg>` elements. These elements are specified like Command-line Arguments but have an additional attribute that can be used to enable arguments only if a given compiler implementation will be used.

Attribute	Description	Required
value line file path	See Command-line Arguments.	Exactly one of these.
compiler	Only pass the specified argument if the chosen compiler implementation matches the value of this attribute. Legal values are the same as those in the above list of valid compilers.)	No

Examples

```
<javac srcdir="${src}"
       destdir="${build}"
       classpath="xyz.jar"
       debug="on"
/>
```

compiles all .java files under the *srcdirectory, andstoresthe.classfilesinthe*build directory. The classpath used includes xyz.jar, and compiling with debug information is on.

```
<javac srcdir="${src}"
       destdir="${build}"
       fork="true"
/>
```

compiles all .java files under the *srcdirectory, andstoresthe.classfilesinthe*build directory. This will fork off the javac compiler using the default javac executable.

```
<javac srcdir="${src}"
       destdir="${build}"
       fork="java$$javac.exe"
/>
```

compiles all .java files under the *srcdirectory, andstoresthe.classfilesinthe*build directory. This will fork off the javac compiler, using the executable named java*javac.exe.Notethatthe* sign needs to be escaped by a second one.

```
<javac srcdir="${src}"
       destdir="${build}"
       includes="mypackage/p1/**,mypackage/p2/**"
       excludes="mypackage/p1/testpackage/**"
       classpath="xyz.jar"
       debug="on"
/>
```

compiles .java files under the *srcdirectory, andstoresthe.classfilesinthe*build directory. The classpath used includes xyz.jar, and debug information is on.

Only files under mypackage/p1 and mypackage/p2 are used. All files in and below the mypackage/p1/testpackage directory are excluded from compilation.

```
<javac srcdir="${src}:${src2}"
       destdir="${build}"
       includes="mypackage/p1/**,mypackage/p2/**"
       excludes="mypackage/p1/testpackage/**"
       classpath="xyz.jar"
       debug="on"
/>
```

is the same as the previous example, with the addition of a second source path, defined by the property src2. This can also be represented using nested ¡src¿ elements as follows:

```
<javac destdir="${build}"
       classpath="xyz.jar"
       debug="on">
  <src path="${src}"/>
  <src path="${src2}"/>
  <include name="mypackage/p1/**"/>
  <include name="mypackage/p2/**"/>
  <exclude name="mypackage/p1/testpackage/**"/>
</javac>
```

If you want to run the javac compiler of a different JDK, you should tell Ant, where to find the compiler and which version of JDK you will be using so it can choose the correct command line switches. The following example executes a JDK 1.1 javac in a new process and uses the correct command line switches even when Ant is running in a Java VM of a different version:

```
<javac srcdir="${src}"
       destdir="${build}"
       fork="yes"
       executable="/opt/java/jdk1.1/bin/javac"
       compiler="javac1.1"
/>
```

Note: If you wish to compile only source files located in certain packages below a common root, use the include/exclude attributes or `<include>`/`<exclude>` nested elements to filter for these packages. Do not include part of your package structure in the srcdir attribute (or nested `<src>` elements), or Ant will recompile your source files every time you run your compile target. See the Ant FAQ for additional information.

Note: If you are using Ant on Windows and a new DOS window pops up for every use of an external compiler, this may be a problem of the JDK you are using. This problem may occur with all JDKs < 1.2.

Jikes Notes

Jikes supports some extra options, which can be set be defining the properties shown below prior to invoking the task. The setting for each property will be in affect for all ¡javac¿ tasks throughout the build. The Ant developers are aware that this is ugly and inflexible ? expect a better solution in the future. All the options are boolean, and must be set to true or yes to be interpreted as anything other than false. By default, build.compiler.warnings is true, while all others are false.

Property	Description	Default
build.compiler.emacs	Enable emacs-compatible error messages.	false
build.compiler.fulldepend	Enable full dependency checking; see the +F switch in the Jikes manual.	false
build.compiler.pedantic	Enable pedantic warnings.	false
build.compiler.warnings Deprecated. Use `<javac>`'s nowarn attribute instead.	Don't disable warning messages.	true

5.2.40 Javadoc/Javadoc2

Description

Generates code documentation using the javadoc tool.

The source directory will be recursively scanned for Java source files to process but only those matching the inclusion rules, and not matching the exclusions rules will be passed to the javadoc tool. This allows wildcards to be used to choose between package names, reducing verbosity and management costs over time. This task, however, has no notion of "changed" files, unlike the javac task. This means all packages will be processed each time this task is run. In general, however, this task is used much less frequently.

This task works seamlessly between different javadoc versions (1.1, 1.2 and 1.4), with the obvious restriction that the 1.2 attributes will be ignored if run in a 1.1 VM.

NOTE: since javadoc calls System.exit(), javadoc cannot be run inside the same VM as ant without breaking functionality. For this reason, this task always forks the VM. This overhead is not significant since javadoc is normally a heavy application and will be called infrequently.

NOTE: the packagelist attribute allows you to specify the list of packages to document outside of the Ant file. It's a much better practice to include everything inside the build.xml file. This option was added in order to make it easier to migrate from regular makefiles, where you would use this option of javadoc. The packages listed in packagelist are not checked, so the task performs even if some packages are missing or broken. Use this option if you wish to convert from an existing makefile. Once things are running you should then switch to the regular notation.

DEPRECATION: the javadoc2 task simply points to the javadoc task and it's there for back compatibility reasons. Since this task will be removed in future versions, you are strongly encouraged to use javadoc instead.

In the table below, 1.1 means available if your current Java VM is a 1.1 VM, 1.2 for either 1.2 or 1.3 and 1.4 for a 1.4 Java VM. 1.2+ means any VM of at least version 1.2.

Parameters

Attribute	Description	Required	
Attribute	Description	Availability	Required
sourcepath sourcepathref hline sourcefiles	Specify where to find source files Specify where to find source files by reference to a PATH defined elsewhere. Comma separated list of source files	all all	At least one of the three or nested `<sourcepath>`, `<fileset>` or `<packageset>`
destdir	Destination directory for output files	all	Yes, unless a doclet has been specified.
maxmemory	Max amount of memory to allocate to the javadoc VM	all	No
packagenames	Comma separated list of package files (with terminating wildcard)	all	No
packageList	The name of a file containing the packages to process	1.2+	No
classpath	Specify where to find user class files	all	No
Bootclasspath	Override location of class files loaded by the bootstrap class loader	1.2+	No
classpathref	Specify where to find user class files by reference to a PATH defined elsewhere.	all	No
bootclasspathref	Override location of class files loaded by the bootstrap class loader by reference to a PATH defined elsewhere.	1.2+	No

Attribute	Description	Required	
Extdirs	Override location of installed extensions	1.2+	No
Overview	Read overview documentation from HTML file	1.2+	No
access	Access mode: one of public, protected, package, or private	all	No (default protected)
Public	Show only public classes and members	all	No
Protected	Show protected/public classes and members (default)	all	No
Package	Show package/protected/public classes and members	all	No
Private	Show all classes and members	all	No
Old	Generate output using JDK 1.1 emulating doclet	1.2	No
Verbose	Output messages about what Javadoc is doing	1.2+	No
Locale	Locale to be used, e.g. en_US or en_US_WIN	1.2+	No
Encoding	Source file encoding name	all	No
Version	Include @version paragraphs	all	No
Use	Create class and package usage pages	1.2+	No
Author	Include @author paragraphs	all	No
Splitindex	Split index into one file per letter	1.2+	No
Windowtitle	Browser window title for the documentation (text)	1.2+	No
Doctitle	Include title for the package index(first) page (html-code)	1.2+	No
Header	Include header text for each page (html-code)	1.2+	No
Footer	Include footer text for each page (html-code)	1.2+	No

Attribute	Description	Required	
bottom	Include bottom text for each page (html-code)	1.2+	No
link	Create links to javadoc output at the given URL	1.2+	No
linkoffline	Link to docs at ¡url¿ using package list at ¡url2¿ - separate the URLs by using a space character.	1.2+	No
group	Group specified packages together in overview page. The format is as described below.	1.2+	No
nodeprecated	Do not include @deprecated information	all	No
nodeprecatedlist	Do not generate deprecated list	1.2+	No
notree	Do not generate class hierarchy	all	No
noindex	Do not generate index	all	No
nohelp	Do not generate help link	1.2+	No
nonavbar	Do not generate navigation bar	1.2+	No
serialwarn	Generate warning about @serial tag	1.2+	No
helpfile	Specifies the HTML help file to use	1.2+	No
stylesheetfile	Specifies the CSS stylesheet to use	1.2+	No
charset	Charset for cross-platform viewing of generated documentation	1.2+	No
docencoding	Output file encoding name	all	No
doclet	Specifies the class file that starts the doclet used in generating the documentation.	1.2+	No
docletpath	Specifies the path to the doclet class file that is specified with the -doclet option.	1.2+	No

Attribute	Description	Required	
docletpathref	Specifies the path to the doclet class file that is specified with the -doclet option by reference to a PATH defined elsewhere.	1.2+	No
additionalparam	Lets you add additional parameters to the javadoc command line. Useful for doclets. Parameters containing spaces need to be quoted using &quot;.	all	No
failonerror	Stop the buildprocess if the command exits with a returncode other than 0.	all	No
excludepackagenames	comma separated list of packages you don't want docs for.	all	No
defaultexcludes	indicates whether default excludes should be used (yes — no); default excludes are used when omitted.	all	No
useexternalfile	indicates whether the sourcefile name specified in srcfiles or as nested source elements should be written to a temporary file to make the command line shorter. Also applies to the package names specified via the packagenames attribute or nested package elements. (yes — no). Default is no.	1.2+	No

Attribute	Description	Required	
source	Necessary to enable javadoc to handle assertions present in J2SE v 1.4 source code. Set this to "1.4" to documents code that compiles using "javac -source 1.4".	1.4	No

Format of the group attribute

The arguments are comma-delimited. Each single argument is 2 space-delimited strings, where the first one is the group's title and the second one a colon delimited list of packages.

If you need to specify more than one group, or a group whose title contains a comma or a space character, using nested group elements is highly recommended.

E.g.,

```
group="XSLT_Packages org.apache.xalan.xslt*,
   XPath_Packages org.apache.xalan.xpath*"
```

Parameters specified as nested elements

packageset

A DirSet. All matched directories that contain Java source files will be passed to javadoc as package names. Package names are created from the directory names by translating the directory separator into dots. Ant assumes the base directory of the packageset points to the root of a package hierarchy.

The packagenames, excludepackagenames and defaultexcludes attributes of the task have no effect on the nested `<packageset>` elements.

fileset

A FileSet. All matched files will be passed to javadoc as source files. Ant will automatically add the include pattern **/*.java to these filesets.

Nested filesets can be used to document sources that are in the default package or if you want to exclude certain files from documentation. If you want to document all source files and don't use the default package, packagesets should be used instead as this increases javadocs performance.

The packagenames, excludepackagenames and defaultexcludes attributes of the task have no effect on the nested `<fileset>` elements.

package

Same as one entry in the list given by packagenames.

Parameters

Attribute	Description	Required
name	The package name (may be a wildcard)	Yes

excludepackage

Same as one entry in the list given by excludepackagenames.

Parameters

Same as for package.

source

Same as one entry in the list given by sourcefiles.

Parameters

Attribute	Description	Required
file	The source file to document	Yes

bf doctitle

Same as the doctitle attribute, but you can nest text inside the element this way.

header

Similar to `<doctitle>`.

footer

Similar to `<doctitle>`.

bottom

Similar to `<doctitle>`.

link

Create link to javadoc output at the given URL. This performs the same role as the link and linkoffline attributes. You can use either syntax (or both at once), but with the nested elements you can easily specify multiple occurrences of the arguments.

Parameters

Attribute	Description	Required
href	The URL for the external documentation you wish to link to	Yes
offline	True if this link is not available online at the time of generating the documentation	No
packagelistLoc	The location to the directory containing the package-list file for the external documentation	Only if the offline attribute is true

group

Separates packages on the overview page into whatever groups you specify, one group per table. This performs the same role as the group attribute. You can use either syntax (or both at once), but with the nested elements you can easily specify multiple occurrences of the arguments.

Parameters

Attribute	Description	Required
title	Title of the group	Yes, unless nested `<title>` given
packages	List of packages to include in that group. Multiple packages are separated with ':'.	Yes, unless nested `<package>`s given

The title may be specified as a nested `<title>` element with text contents, and the packages may be listed with nested `<package>` elements as for the main task.

doclet

The doclet nested element is used to specify the doclet that javadoc will use to process the input source files. A number of the standard javadoc arguments are actually arguments of the standard doclet. If these are specified in the javadoc task's attributes, they will be passed to the doclet specified in the `<doclet>` nested element. Such attributes should only be specified, therefore, if they can be interpreted by the doclet in use.

If the doclet requires additional parameters, these can be specified with `<param>` elements within the `<doclet>` element. These paramaters are restricted to simple strings. An example usage of the doclet element is shown below:

```
<javadoc ... >
   <doclet name="theDoclet"
           path="path/to/theDoclet">
      <param name="-foo" value="foovalue"/>
      <param name="-bar" value="barvalue"/>
   </doclet>
</javadoc>
```

tag

The tag nested element is used to specify custom tags. This option is only available with Java 1.4.

Parameters

Attribute	Description	Required
name	Name of the tag (e.g. todo)	Yes
description	Description for tag (e.g. To do:)	Yes
enabled	Whether or not the tag is enabled (defaults to true)	No
scope	Scope for the tag - the elements in which it can be used. This is a comma separated list of some of the elements: overview, packages, types, constructors, methods, fields or the default, all.	No

taglet

The taglet nested element is used to specify custom taglets. This option is only available with Java 1.4.

Parameters

Attribute	Description	Required
name	The name of the taglet class (e.g. com.sun.tools.doclets.ToDoTaglet)	Yes
path	A path specifying the search path for the taglet class (e.g. /home/taglets). The path may also be specified by a nested `<path>` element	No

sourcepath, classpath and bootclasspath

Javadoc's sourcepath, classpath and bootclasspath attributes are PATH like structure and can also be set via nested sourcepath, classpath and bootclasspath elements respectively.

Example

```
<javadoc packagenames="com.dummy.test.*"
         sourcepath="src"
         excludepackagenames="com.dummy.test.doc-files.*"
         defaultexcludes="yes"
         destdir="docs/api"
         author="true"
         version="true"
         use="true"
         windowtitle="Test API">
  <doctitle><![CDATA[<h1>Test</h1>]]></doctitle>
  <bottom><![CDATA[<i>Copyright &#169;
           2000 Dummy Corp. All Rights Reserved.</i>]]></bottom>
  <tag name="todo" scope="all" description="To do:" />
  <group title="Group 1 Packages" packages="com.dummy.test.a*"/>
  <group title="Group 2 Packages"
   packages="com.dummy.test.b*:com.dummy.test.c*"/>
  <link offline="true"
    href="http://java.sun.com/products/jdk/1.2/docs/api/"
    packagelistLoc="C:\tmp"/>
  <link href="http://developer.java.sun.com/developer/products/xml/docs/api/"/>
</javadoc>
```

is the same as

```
<javadoc
        destdir="docs/api"
        author="true"
        version="true"
        use="true"
        windowtitle="Test API">

  <packageset dir="src" defaultexcludes="yes">
    <include name="com/dummy/test/**" />
    <exclude name="com/dummy/test/doc-files/**"/>
  </packageset>

  <doctitle><![CDATA[<h1>Test</h1>]]></doctitle>
  <bottom><![CDATA[<i>Copyright &#169;
           2000 Dummy Corp. All Rights Reserved.</i>]]></bottom>
  <tag name="todo" scope="all" description="To do:" />
  <group title="Group 1 Packages" packages="com.dummy.test.a*"/>
  <group title="Group 2 Packages"
   packages="com.dummy.test.b*:com.dummy.test.c*"/>
  <link offline="true"
   href="http://java.sun.com/products/jdk/1.2/docs/api/"
   packagelistLoc="C:\tmp"/>
  <link href="http://developer.java.sun.com/developer/products/xml/docs/api/"/>
</javadoc>
```

or

```
<javadoc
        destdir="docs/api"
        author="true"
        version="true"
        use="true"
        windowtitle="Test API">
```

```
    <fileset dir="src" defaultexcludes="yes">
      <include name="com/dummy/test/**" />
      <exclude name="com/dummy/test/doc-files/**"/>
    </fileset>

    <doctitle><![CDATA[<h1>Test</h1>]]></doctitle>
    <bottom><![CDATA[<i>Copyright &#169;
               2000 Dummy Corp. All Rights Reserved.</i>]]></bottom>
    <tag name="todo" scope="all" description="To do:" />
    <group title="Group 1 Packages" packages="com.dummy.test.a*"/>
    <group title="Group 2 Packages" packages="com.dummy.test.b*:com.dummy.test.c*"/>
    <link offline="true"
     href="http://java.sun.com/products/jdk/1.2/docs/api/" packagelistLoc="C:\tmp"/>
    <link href="http://developer.java.sun.com/developer/products/xml/docs/api/"/>
  </javadoc>
```

5.2.41 LoadFile

Description

Load a text file into a single property. Unless an encoding is specified, the encoding of the current locale is used.

Parameters

Attribute	Description	Required
srcFile	source file	Yes
property	property to save to	Yes
encoding	encoding to use when loading the file	No
failonerror	Whether to halt the build on failure	No, default ”true”

The LoadFile task supports nested FilterChains.

Examples

```
    <loadfile property="message"
      srcFile="message.txt"/>
```

Load file message.txt into property ”message”; an ¡echo¿ can print this.

```
    <loadfile property="encoded-file"
      srcFile="loadfile.xml"
      encoding="ISO-8859-1"/>
```

Load a file using the latin-1 encoding

```
    <loadfile
      property="optional.value"
```

```
  srcFile="optional.txt"
  failonerror="false"/>
```

Load a file, don't fail if it is missing (a message is printed, though)

```
<loadfile
  property="mail.recipients"
  srcFile="recipientlist.txt">
  <filterchain>
    <striplinebreaks/>
  </filterchain>
</loadfile>
```

Load a property which can be used as a parameter for another task (in this case mail), merging lines to ensure this happens.

```
<loadfile
  property="system.configuration.xml"
  srcFile="configuration.xml">
    <expandproperties/>
</loadfile>
```

Load an XML file into a property, expanding all properties declared in the file in the process.

5.2.42 LoadProperties

Description

Load a file's contents as Ant properties. This is equivalent to `<property file="..."/>` except that it supports nested `<filterchain>` elements and it cannot be specified outside a target.

If you want to simulate property's prefix attribute, please use prefixlines filter.

Parameters

Attribute	Description	Required
srcFile	source file	Yes

The LoadProperties task supports nested FilterChains.

Examples

```
<loadproperties srcFile="file.properties"/>
```

Load contents of file.properties as Ant properties.

```
<loadproperties srcFile="file.properties">
  <filterchain>
```

```
    <linecontains>
      <contains value="import."/>
    </linecontains>
  </filterchain>
</loadproperties>
```

Read the lines that contain the string ”import.” from the file ”file.properties” and load them as Ant properties.

5.2.43 Mail

Description

A task to send SMTP email. This task can send mail using either plain text, UU encoding, or MIME format mail, depending on what is available. Attachments may be sent using nested fileset elements.

Note: This task may depend on external libraries that are not included in the Ant distribution. See Library Dependencies for more information.

Parameters

Attribute	Description	Required
Attribute	Description	Required
from	Email address of sender.	Either a from attribute, or a `<from>` element.
tolist cclist bcclist	Comma-separated list of recipients. Comma-separated list of recipients to carbon copy Comma-separated list of recipients to carbon copy	At least one of these, or the equivalent elements.
message messagefile	Message to send in the body of the email. File to send as the body of the email. Property values in the file will be expanded.	One of these or a `<message>` element.
messagemimetype	The content type of the message. The default is text/plain.	No
files	Files to send as attachments to the email. Separate multiple file names using a comma or space. You can also use ¡fileset¿ elements to specify files.	No
failonerror	flag to indicate whether to halt the build on any error. The default value is true.	No.

Attribute	Description	Required
includefilenames	Include filename(s) before file contents. Valid only when the plain encoding is used. The default value is false.	No
mailhost	Host name of the SMTP server. The default value is localhost.	No
mailport	TCP port of the SMTP server. The default value is 25.	No
encoding	Specifies the encoding to use for the content of the email. Values are mime, uu, plain, or auto. The default value is auto.	No
subject	Email subject line.	No

Parameters specified as nested elements

`to / cc / bcc / from`

Adds an email address element. It takes the following attributes:

Attribute	Description	Required
Attribute	Description	Required
name	The display name for the address.	No
address	The email address.	Yes

message

Specifies the message to include in the email body. It takes the following attributes:

Attribute	Description	Required
Attribute	Description	Required
src	The file to use as the message.	No
mimetype	The content type to use for the message.	No

If the src attribute is not specified, then text can be added inside the `<message>` element. Property expansion will occur in the message, whether it is specified as an external file or as text within the `<message>` element.

Examples

```
<mail from="me"
      tolist="you"
      subject="Results of nightly build"
      files="build.log"/>
```

Sends an email from me to you with a subject of Results of nightly build and includes the contents of the file build.log in the body of the message.

```
<mail mailhost="smtp.myisp.com" mailport="1025" subject="Test build">
```

```
  <from address="me@myisp.com"/>
  <to address="all@xyz.com"/>
  <message>The ${buildname} nightly build has completed</message>
  <fileset dir="dist">
    <includes name="**/*.zip"/>
  </fileset>
</mail>
```

Sends an eMail from me@myisp.com to all@xyz.com with a subject of Test Build and attaches any zip files from the dist directory. The task will attempt to use JavaMail and fall back to UU encoding or no encoding in that order depending on what support classes are available. ${buildname} will be replaced with the buildname property's value.

5.2.44 Manifest

Description

Creates a manifest file.

This task can be used to write a Manifest file, optionally replacing or updating an existing file.

Manifests are processed according to the Jar file specification.. Specifically, a manifest element consists of a set of attributes and sections. These sections in turn may contain attributes. Note in particular that this may result in manifest lines greater than 72 bytes being wrapped and continued on the next line.

Parameters

Attribute	Description	Required
Attribute	Description	Required
file	the manifest-file to create/update.	Yes
mode	One of "update" or "replace", default is "replace".	No
encoding	The encoding used to read the existing manifest when updating.	No, defaults to UTF-8 encoding.

Nested elements

attribute

One attribute for the manifest file. Those attributes that are not nested into a section will be added to the "Main" section.

Attribute	Description	Required
name	the name of the attribute.	Yes
value	the value of the attribute.	Yes

section

A manifest section - you can nest attribute elements into sections.

Attribute	Description	Required
name	the name of the section.	No, if omitted it will be assumed to be the main section.

Examples

```
<manifest file="MANIFEST.MF">
  <attribute name="Built-By" value="${user.name}"/>
  <section name="common">
    <attribute name="Specification-Title" value="Example"/>
    <attribute name="Specification-Version" value="${version}"/>
    <attribute name="Specification-Vendor" value="Example Organization"/>
    <attribute name="Implementation-Title" value="common"/>
    <attribute name="Implementation-Version" value="${version} ${TODAY}"/>
    <attribute name="Implementation-Vendor" value="Example Corp."/>
  </section>
  <section name="common/class1.class">
    <attribute name="Sealed" value="false"/>
  </section>
</manifest>
```

Creates or replaces the file MANIFEST.MF. Note that the Built-By attribute will take the value of the Ant property ${user.name}. The same is true for the ${version} and ${TODAY} properties. This example produces a MANIFEST.MF that contains package version identification for the package common.

The manifest produced by the above would look like this:

```
Manifest-Version: 1.0
Built-By: bodewig
Created-By: Apache Ant 1.5alpha

Name: common
Specification-Title: Example
Specification-Vendor: Example Organization
Implementation-Vendor: Example Corp.
Specification-Version: 1.1
Implementation-Version: 1.1 February 19 2002
Implementation-Title: common

Name: common/class1.class
Sealed: false
```

5.2.45 Mkdir

Description

Creates a directory. Also non-existent parent directories are created, when necessary.

Parameters

Attribute	Description	Required
dir	the directory to create.	Yes

Examples

```
<mkdir dir="${dist}"/>
```

creates a directory ${dist}.

```
<mkdir dir="${dist}/lib"/>
```

creates a directory ${dist}/lib.

5.2.46 Move

Description

Moves a file to a new file or directory, or sets of files to a new directory. By default, the destination file is overwritten if it already exists. When overwrite is turned off, then files are only moved if the source file is newer than the destination file, or when the destination file does not exist.

FileSets are used to select sets of files to move to the todir directory.

Parameters

Attribute	Description	Required
file	the file to move	One of file or at least one nested fileset element
preservelastmodified	Give the moved files the same last modified time as the original source files. (Note: Ignored on Java 1.1)	No; defaults to false.
tofile todir	the file to move to todir the directory to move to	With the file attribute, either tofile or can be used. With nested filesets, if the fileset size is greater than 1 or if the only entry in the fileset is a directory or if the file attribute is already specified, only todir is allowed

Attribute	Description	Required
overwrite	overwrite existing files even if the destination files are newer (default is "true")	No
filtering	indicates whether token filtering should take place during the move. See the filter task for a description of how filters work.	No
flatten	ignore directory structure of source directory, copy all files into a single directory, specified by the todir attribute (default is "false").Note that you can achieve the same effect by using a flatten mapper	No
includeEmptyDirs	Copy empty directories included with the nested FileSet(s). Defaults to "yes".	No
failonerror	Log a warning message, but do not stop the build, when the file to move does not exist. Only meaningful when moving a single file.	No; defaults to true.
verbose	Log the files that are being moved.	No; defaults to false.
encoding	The encoding to assume when filter-moving the files. since Ant 1.5.	No - defaults to default JVM encoding

Parameters specified as nested elements

mapper

You can define file name transformations by using a nested mapper element. The default mapper used by `<copy>` is the identity.

filterchain

The Move task supports nested FilterChains.

If `<filterset>` and `<filterchain>` elements are used inside the same `<move>` task, all `<filterchain>` elements are processed first followed by `<filterset>` elements.

Examples

Move a single file (rename a file)

```
<move file="file.orig" tofile="file.moved"/>
```

Move a single file to a directory

```
<move file="file.orig" todir="dir/to/move/to"/>
```

Move a directory to a new directory

```
<move todir="new/dir/to/move/to">
  <fileset dir="src/dir"/>
</move>
```

Move a set of files to a new directory

```
<move todir="some/new/dir">
  <fileset dir="my/src/dir">
    <include name="**/*.jar"/>
    <exclude name="**/ant.jar"/>
  </fileset>
</move>
```

Append ".bak" to the names of all files in a directory.

```
<move todir="my/src/dir">
  <fileset dir="my/src/dir">
    <exclude name="**/*.bak"/>
  </fileset>
  <mapper type="glob" from="*" to="*.bak"/>
</move>
```

5.2.47 Parallel

Description

Parallel is a container task - it can contain other Ant tasks. Each nested task within the parallel task will be executed in its own thread.

Parallel tasks have a number of uses in an Ant build file including:

- Taking advantage of available processing resources to reduce build time
- Testing servers, where the server can be run in one thread and the test harness is run in another thread.

Care must be taken when using multithreading to ensure the tasks within the threads do not interact. For example, two javac compile tasks which write classes into the same destination directory may interact where one tries to read a class for dependency information while the other task is writing the class file. Be sure to avoid these types of interactions within a `<parallel>` task

The parallel task has no attributes and does not support any nested elements apart from Ant tasks. Any valid Ant task may be embedded within a parallel task, including other parallel tasks.

Note that while the tasks within the parallel task are being run, the main thread will be blocked waiting for all the child threads to complete.

If any of the tasks within the `<parallel>` task fails, the remaining tasks in other threads will continue to run until all threads have completed. In this situation, the parallel task will also fail.

The parallel task may be combined with the sequential task to define sequences of tasks to be executed on each thread within the parallel block

Examples

```
<parallel>
  <wlrun ... >
  <sequential>
    <sleep seconds="30"/>
    <junit ... >
    <wlstop/>
  </sequential>
</parallel>
```

This example represents a typical pattern for testing a server application. In one thread the server is started (the wlrun task). The other thread consists of a three tasks which are performed in sequence. The sleep task is used to give the server time to come up. Another task which is capable of validating that the server is available could be used in place of the sleep task. The test harness is then run. Once the tests are complete, the server is stopped (using wlstop in this example), allowing both threads to complete. The parallel task will also complete at this time and the build will then continue.

```
<parallel>
  <javac ...> <!-- compiler servlet code -->
  <wljspc ...> <!-- precompile JSPs -->
</parallel>
```

This example shows two independent tasks being run to achieve better resource utilization during the build. In this instance, some servlets are being compiled in one thead and a set of JSPs is being precompiled in another. As noted above, you need to be careful that the two tasks are independent, both in terms of their dependencies and in terms of their potential interactions in Ant's external environment.

5.2.48 Patch

Description

Applies a diff file to originals. ; requires "patch" to be on the execution path.

Parameters

Attribute	Description	Required
patchfile	the file that includes the diff output	Yes
originalfile	the file to patch	No, tries to guess it from the diff file
backups	Keep backups of the unpatched files	No
quiet	Work silently unless an error occurs	No
reverse	Assume patch was created with old and new files swapped.	No

Attribute	Description	Required
ignorewhitespace	Ignore whitespace differences.	No
strip	Strip the smallest prefix containing num leading slashes from filenames.	No
dir	The directory in which to run the patch command.	No, default is the project's basedir.

Examples

```
<patch patchfile="module.1.0-1.1.patch"/>
```

applies the diff included in module.1.0-1.1.patch to the files in base directory guessing the filename(s) from the diff output.

```
<patch patchfile="module.1.0-1.1.patch" strip="1"/>
```

like above but one leading directory part will be removed. i.e. if the diff output looked like

```
--- a/mod1.0/A  Mon Jun  5 17:28:41 2000
+++ a/mod1.1/A  Mon Jun  5 17:28:49 200
```

the leading a/ will be stripped.

5.2.49 PathConvert

Description

Converts a nested `<path>` or reference to a Path, FileSet, DirSet, or FileList into a path form for a particular platform, and stores the result in a given property. It can also be used when you need to convert a Path, FileSet, or DirSet into a list, separated by a given character, such as a comma or space, or, conversely, to convert a list of files in a FileList into a path.

Nested `<map>` elements can be specified to map Windows drive letters to Unix paths, and vice-versa.

Parameters

Attribute	Description	Required
targetos	The target architecture. Must be one of 'unix', 'windows', 'netware' or 'os/2'. This is a shorthand mechanism for specifying both pathsep and dirsep according to the specified target architecture.	Yes, unless pathsep and/or dirsep are specified.

Attribute	Description	Required
dirsep	The character(s) to use as the directory separator in the generated paths.	No, defaults to current JVM File.separator
pathsep	The character(s) to use as the path-element separator in the generated paths.	No, defaults to current JVM File.pathSeparator
property	The name of the property in which to place the converted path.	Yes
refid	What to convert, given as a reference to a `<path>`, `<fileset>`, `<dirset>`, or `<filelist>` defined elsewhere	No; if omitted, a nested `<path>` element must be supplied.
setonempty	Should the property be set, even if the result is the empty string?	No; default is ”true”.

Parameters specified as nested elements

map

Specifies the mapping of path prefixes between Unix and Windows.

Attribute	Description	Required
from	The prefix to match. Note that this value is case-insensitive when the build is running on a Windows platform and case-sensitive when running on a Unix platform.	Yes
to	The replacement text to use when from is matched.	Yes

Each map element specifies a single replacement map to be applied to the elements of the path being processed. If no map entries are specified, then no path prefix mapping is performed.

Note: The map elements are applied in the order specified, and only the first matching map element is applied. So, the ordering of your map elements can be important, if any from values are prefixes of other from values.

path

If the refid attribute is not specified, then a nested ¡path¿ element must be supplied. See Path-like Structures for details.

Examples

In the examples below, assume that the ${wl.home} property has the value d: weblogic, and ${wl.home.unix} has the value /weblogic.

Example 1

```
<path id="wl.path">
  <pathelement location="${wl.home}/lib/weblogicaux.jar"/>
```

```
    <pathelement location="${wl.home}/classes"/>
    <pathelement location="${wl.home}/mssqlserver4/classes"/>
    <pathelement location="c:\winnt\System32"/>
  </path>

  <pathconvert targetos="unix" property="wl.path.unix" refid="wl.path">
    <map from="${wl.home}" to="${wl.home.unix}"/>
    <map from="c:" to=""/>
  </pathconvert>
```

will generate the path shown below and store it in the property named wl.path.unix.

/weblogic/lib/weblogicaux.jar:/weblogic/classes:/weblogic/mssqlserver4/classes:/WINNT/SYSTEM32

Example 2

Given a FileList defined as:

```
<filelist id="custom_tasks.jars"
      dir="${env.HOME}/ant/lib"
      files="njavac.jar,xproperty.jar"/>
```

then:

```
  <pathconvert targetos="unix"
   property="custom_tasks.jars" refid="custom_tasks.jars">
    <map from="${env.HOME}" to="/usr/local"/>
  </pathconvert>
```

will convert the list of files to the following Unix path:

/usr/local/ant/lib/njavac.jar:/usr/local/ant/lib/xproperty.jar

Example 3

```
  <fileset dir="${src.dir}" id="src.files">
    <include name="**/*.java"/>
  </fileset>

  <pathconvert pathsep="," property="javafiles" refid="src.files"/>
```

This example takes the set of files determined by the fileset (all files ending in .java), joins them together separated by commas, and places the resulting list into the property javafiles. The directory separator is not specified, so it defaults to the appropriate character for the current platform. Such a list could then be used in another task, like javadoc, that requires a comma separated list of files.

5.2.50 Property

Description

Sets a property (by name and value), or set of properties (from file or resource) in the project. Properties are case sensitive.

Properties are immutable: whoever sets a property first freezes it for the rest of the build; they are most definately not variable.

There are five ways to set properties:

1. By supplying both the name and value attribute.
2. By supplying both the name and refid attribute.
3. By setting the file attribute with the filename of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
4. By setting the resource attribute with the resource name of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
5. By setting the environment attribute with a prefix to use. Properties will be defined for every environment variable by prefixing the supplied name and a period to the name of the variable.

Although combinations of these ways are possible, only one should be used at a time. Problems might occur with the order in which properties are set, for instance.

The value part of the properties being set, might contain references to other properties. These references are resolved at the time these properties are set. This also holds for properties loaded from a property file.

A list of predefined properties can be found here.

Parameters

Attribute	Description	Required
name	the name of the property to set.	No
value	the value of the property.	One of these, when using the name attribute
location	Sets the property to the absolute filename of the given file. If the value of this attribute is an absolute path, it is left unchanged (with / and characters converted to the current platforms conventions). Otherwise it is taken as a path relative to the project's basedir and expanded.	
refid	Reference to an object defined elsewhere. Only yields reasonable results for references to PATH like structures or properties.	

Attribute	Description	Required
resource	the resource name of the property file.	One of these, when not using the name attribute
file	the filename of the property file.	
environment	the prefix to use when retrieving environment variables. Thus if you specify environment=”myenv” you will be able to access OS-specific environment variables via property names ”myenv.PATH” or ”myenv.TERM”. Note that if you supply a property name with a final ”.” it will not be doubled. i.e. environment=”myenv.” will still allow access of environment variables through ”myenv.PATH” and ”myenv.TERM”. This functionality is currently only implemented on select platforms. Feel free to send patches to increase the number of platforms this functionality is supported on ;). Note also that properties are case sensitive, even if the environment variables on your operating system are not, e.g. it will be *env.Pathnot*env.PATH on Windows 2000.	
classpath	the classpath to use when looking up a resource.	No
classpathref	the classpath to use when looking up a resource, given as reference to a ¡path¿ defined elsewhere..	No
prefix	Prefix to apply to properties loaded using file or resource. A ”.” is appended to the prefix if not specified.	No

Parameters specified as nested elements

classpath

Property’s classpath attribute is a PATH like structure and can also be set via a nested classpath element.

Examples

```
<property name="foo.dist" value="dist"/>
```

sets the property foo.dist to the value ”dist”.

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

reads a set of properties from a file called ”foo.properties”.

```
<property resource="foo.properties"/>
```

reads a set of properties from a resource called ”foo.properties”.

Note that you can reference a global properties file for all of your Ant builds using the following:

```
<property file="${user.home}/.ant-global.properties"/>
```

since the ”user.home” property is defined by the Java virtual machine to be your home directory. This technique is more appropriate for Unix than Windows since the notion of a home directory doesn’t exist on Windows. On the JVM that I tested, the home directory on Windows is ”C:∵ Different JVM implementations may use other values for the home directory on Windows.

```
<property environment="env"/>
<echo message="Number of Processors = ${env.NUMBER_OF_PROCESSORS}"/>
<echo message="ANT_HOME is set to = ${env.ANT_HOME}"/>
```

reads the system environment variables and stores them in properties, prefixed with ”env”. Note that this only works on select operating systems. Two of the values are shown being echoed.

5.2.51 Record

Description

A recorder is a listener to the current build process that records the output to a file.

Several recorders can exist at the same time. Each recorder is associated with a file. The filename is used as a unique identifier for the recorders. The first call to the recorder task with an unused filename will create a recorder (using the parameters provided) and add it to the listeners of the build. All subsequent calls to the recorder task using this filename will modify that recorders state (recording or not) or other properties (like logging level).

Some technical issues: the file’s print stream is flushed for ”finished” events (buildFinished, targetFinished and taskFinished), and is closed on a buildFinished event.

Parameters

Attribute	Description	Required
name	The name of the file this logger is associated with.	yes
action	This tells the logger what to do: should it start recording or stop? The first time that the recorder task is called for this logfile, and if this attribute is not provided, then the default for this attribute is ”start”. If this attribute is not provided on subsequent calls, then the state remains as previous. [Values = start—stop, Default = no state change]	no
append	Should the recorder append to a file, or create a new one? This is only applicable the first time this task is called for this file. [Values = yes—no, Default=yes]	no
emacsmode	Removes [task] banners like Ant’s -emacs command line switch if set to true.	no, default is false
loglevel	At what logging level should this recorder instance record to? This is not a once only parameter (like append is) – you can increase or decrease the logging level as the build process continues. [Values= error—warn—info—verbose—debug, Default = no change]	no

Examples

The following build.xml snippet is an example of how to use the recorder to record just the `<javac>` task:

```
...
<compile >
    <record name="log.txt" action="start"/>
    <javac ...
    <record name="log.txt" action="stop"/>
<compile/>
...
```

The following two calls to ¡record¿ set up two recorders: one to file ”records-simple.log” at logging level info (the default) and one to file ”ISO.log” using logging level of verbose.

```
...
<record name="records-simple.log"/>
<record name="ISO.log" loglevel="verbose"/>
...
```

Notes

There is some functionality that I would like to be able to add in the future. They include things like the following:

Attribute	Description	Required
listener	A classname of a build listener to use from this point on instead of the default listener.	no
includetarget	A comma-separated list of targets to automatically record. If this value is "all", then all targets are recorded. [Default = all]	no
excludetarget	A comma-separated list of targets to automatically record. If this value is "all", then all targets are recorded. [Default = all]	no
includetask excludetask	A comma-separated list of task to automatically record or not. This could be difficult as it could conflict with the includetarget/excludetarget. (e.g.: includetarget="compile" excludetask="javac", what should happen?)	no
action	add greater flexibility to the action attribute. Things like close to close the print stream.	no

5.2.52 Rename

Deprecated

This task has been deprecated. Use the Move task instead.

Description

Renames a given file.

Parameters

Attribute	Description	Required
src	file to rename.	Yes
dest	new name of the file.	Yes
replace	Enable replacing of existing file (default: on).	No

Examples

```
<rename src="foo.jar" dest="${name}-${version}.jar"/>
```

Renames the file foo.jar to $name-$version.jar (assuming name and version being predefined properties). If a file named $name-$version.jar already exists, it will be removed prior to renaming foo.jar.

5.2.53 Replace

Description

Replace is a directory based task for replacing the occurrence of a given string with another string in selected file.

If you want to replace a text that crosses line boundaries, you must use a nested `<replacetoken>` element.

Parameters

Attribute	Description	Required
file	file for which the token should be replaced.	Exactly one of the two.
dir	The base directory to use when replacing a token in multiple files.	
encoding	The encoding of the files upon which replace operates.	No - defaults to default JVM encoding
token	the token which must be replaced.	Yes, unless a nested replacetoken element or the replacefilterfile attribute is used.
value	the new value for the token. When omitted, an empty string ("") is used.	No
summary	Indicates whether a summary of the replace operation should be produced, detailing how many token occurrences and files were processed	No, by default no summary is produced
propertyFile	valid property file from which properties specified using nested `<replacefilter>` elements are drawn.	Yes only if property attribute of `<replacefilter>` is used.
replacefilterfile	valid property file. Each property will be treated as a replacefilter where token is the name of the property and value is the properties value.	No.
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No

Attribute	Description	Required
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No

Examples

```
<replace file="${src}/index.html" token="@@@" value="wombat"/>
```

replaces occurrences of the string ”@@@” with the string ”wombat”, in the file ${src}/index.html.

Parameters specified as nested elements

This task forms an implicit FileSet and supports all attributes of `<fileset>` as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

If either the text you want to replace or the replacement text cross line boundaries, you can use nested elements to specify them.

Examples

```
<replace dir="${src}" value="wombat">
  <include name="**/*.html"/>
  <replacetoken><![CDATA[multi line
token]]></replacetoken>
</replace>
```

replaces occurrences of the string ”multi line
ntoken” with the string ”wombat”, in all HTML files in the directory ${src}.Where n is the platform specific line separator.

```
<replace file="${src}/index.html">
  <replacetoken><![CDATA[two line
token]]></replacetoken>
  <replacevalue><![CDATA[two line
token]]></replacevalue>
</replace>
```

replacefilter

In addition to allowing for multiple replacements, optional nested `<replacefilter>` elements allow replacement values to be extracted from a property file. The name of this file is specified using the `<replace>` attribute propertyFile.

Attribute	Description	Required
token	The string to search for.	Yes
value	The replacement string.	Either may be specified, but not both. Both can be omitted, if desired.
property	Name of the property whose value is to serve as the replacement value.	

If neither value nor property is used, the value provided using the `<replace>` attribute value and/or the `<replacevalue>` element is used. If no value was specified using either of these options, the token is replaced with an empty string.

Examples

```
<replace
    file="configure.sh"
    value="defaultvalue"
    propertyFile="source/name.properties">
  <replacefilter
    token="@token1@"/>
  <replacefilter
    token="@token2@"
    value="value2"/>
  <replacefilter
    token="@token3@"
    property="property.key"/>
</replace>
```

In file configure.sh, replace all instances of "@token1@" with "defaultvalue", all instances of "@token2@" with "value2", and all instances of "@token3@" with the value of the property "property.key", as it appears in property file src/name.properties.

Note: It is possible to use either the `token/<replacetoken>` and `value/<replacevalue>` attributes/elements, the nested replacefilter elements, or both in the same operation.

5.2.54 Rmic

Description

Runs the rmic compiler for a certain class.

Rmic can be run on a single class (as specified with the classname attribute) or a number of classes at once (all classes below base that are neither _Stub nor _Skel classes). If you want to rmic a single class and this class is a class nested into another class, you have to specify the classname in the form Outer$$Inner instead of Outer.Inner.

It is possible to refine the set of files that are being rmiced. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes base) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

It is possible to use different compilers. This can be selected with the "build.rmic" property or the compiler attribute. There are three choices:

1. sun (the standard compiler of the JDK)

2. kaffe (the standard compiler of Kaffe)

3. weblogic

The miniRMI project contains a compiler implementation for this task as well, please consult miniRMI's documentation to learn how to use it.

Parameters

Attribute	Description	Required
base	the location to store the compiled files.	Yes
classname	the class for which to run rmic.	No
filtering	indicates whether token filtering should take place	No
sourcebase	Pass the "-keepgenerated" flag to rmic and move the generated source file to the given sourcebase directory.	No
stubversion	Specify the JDK version for the generated stub code. Specify "1.1" to pass the "-v1.1" option to rmic.	No
classpath	The classpath to use during compilation	No
classpathref	The classpath to use during compilation, given as reference to a PATH defined elsewhere	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
verify	check that classes implement Remote before handing them to rmic (default is false)	No
iiop	indicates that portable (RMI/IIOP) stubs should be generated	No
iiopopts	additional arguments for IIOP class generation	No
idl	indicates that IDL output files should be generated	No
idlopts	additional arguments for IDL file generation	No
debug	generate debug info (passes -g to rmic). Defaults to false.	No
includeAntRuntime	whether to include the Ant run-time libraries; defaults to yes.	No
includeJavaRuntime	whether to include the default run-time libraries from the executing VM; defaults to no.	No

Attribute	Description	Required
extdirs	location of installed extensions.	No
compiler	The compiler implementation to use. If this attribute is not set, the value of the build.rmic property, if set, will be used. Otherwise, the default compiler for the current VM will be used. (See the above list of valid compilers.)	No

Parameters specified as nested elements

classpath and extdirs

Rmic's classpath and extdirs attributes are PATH like structure and can also be set via a nested classpath and extdirs elements.

compilerarg

You can specify additional command line arguments for the compiler with nested `<compilerarg>` elements. These elements are specified like Command-line Arguments but have an additional attribute that can be used to enable arguments only if a given compiler implementation will be used.

Attribute	Description	Required
value line file path	See Command-line Arguments.	Exactly one of these.
compiler	Only pass the specified argument if the chosen compiler implementation matches the value of this attribute. Legal values are the same as those in the above list of valid compilers.)	No

Examples

```
<rmic classname="com.xyz.FooBar" base="${build}/classes"/>
```

runs the rmic compiler for the class com.xyz.FooBar. The compiled files will be stored in the directory ${build}/classes.

```
<rmic base="${build}/classes" includes="**/Remote*.class"/>
```

runs the rmic compiler for all classes with .class files below ${build}/classes whose classname starts with Remote. The compiled files will be stored in the directory ${build}/classes.

5.2.55 Sequential

Description

Sequential is a container task - it can contain other Ant tasks. The nested tasks are simply executed in sequence. Sequential's primary use is to support the sequential execution of a subset of tasks within the parallel task

The sequential task has no attributes and does not support any nested elements apart from Ant tasks. Any valid Ant task may be embedded within the sequential task.

Example

```
<parallel>
  <wlrun ... >
  <sequential>
    <sleep seconds="30"/>
    <junit ... >
    <wlstop/>
  </sequential>
</parallel>
```

This example shows how the sequential task is used to execute three tasks in sequence, while another task is being executed in a separate thread.

5.2.56 SignJar

Description

Signs jar or zip files with the javasign command line tool. The tool detailed dependency checking: files are only signed if they are not signed. The signjar attribute can point to the file to generate; if this file exists then its modification date is used as a cue as to whether to resign any JAR file. Note: Requires Java 1.2 or later.

Parameters

Attribute	Description	Required
jar	the jar file to sign	Yes, unless nested filesets have been used.
alias	the alias to sign under	Yes.
storepass	password for keystore integrity.	Yes.
keystore	keystore location	No
storetype	keystore type	No
keypass	password for private key (if different)	No
sigfile	name of .SF/.DSA file	No
signedjar	name of signed JAR file	No
verbose	(true — false) verbose output when signing	No; default false
internalsf	(true — false) include the .SF file inside the signature block	No; default false

Attribute	Description	Required
sectionsonly	(true — false) don't compute hash of entire manifest	No; default false
lazy	flag to control whether the presence of a signature file means a JAR is signed	No; default false
maxmemory	Specifies the maximum memory the jarsigner VM will use. Specified in the style of standard java memory specs (e.g. 128m = 128 MBytes)	No

Parameters as nested elements

Attribute	Description	Required
fileset	fileset of JAR files to sign. Will be ignored if the jar attribute of the task has been set.	No

Examples

```
<signjar jar="${dist}/lib/ant.jar" alias="apache-group" storepass="secret"/>
```

signs the ant.jar with alias "apache-group" accessing the keystore and private key via "secret" password.

5.2.57 Sleep

Description

A task for sleeping a short period of time, useful when a build or deployment process requires an interval between tasks.

Parameters

Attribute	Description	Required
hours	hours to to add to the sleep time	No
minutes	minutes to add to the sleep time	No
seconds	seconds to add to the sleep time	No
milliseconds	milliseconds to add to the sleep time	No
failonerror	flag controlling whether to break the build on an error.	No

The sleep time is the sum of specified values, hours, minutes seconds and milliseconds. A negative value can be supplied to any of them provided the total sleep time is positive

Note that sleep times are always hints to be interpred by the OS how it feels - small times may either be ignored or rounded up to a minimum timeslice. Note also that the system clocks often have a fairly low granularity too, which complicates measuring how long a sleep actually took.

Examples

```
<sleep milliseconds="10"/>
```

Sleep for about 10 mS.

```
<sleep seconds="2"/>
```

Sleep for about 2 seconds.

```
<sleep hours="1" minutes="-59" seconds="-58"/>
```

Sleep for one hour less 59:58, or two seconds again

```
<sleep/>
```

Sleep for no time at all. This may yield the CPU time to another thread or process.

5.2.58 Sql

Description

Executes a series of SQL statements via JDBC to a database. Statements can either be read in from a text file using the src attribute or from between the enclosing SQL tags.

Multiple statements can be provided, separated by semicolons (or the defined delimiter). Individual lines within the statements can be commented using either –, // or REM at the start of the line.

The autocommit attribute specifies whether auto-commit should be turned on or off whilst executing the statements. If auto-commit is turned on each statement will be executed and committed. If it is turned off the statements will all be executed as one transaction.

The onerror attribute specifies how to proceed when an error occurs during the execution of one of the statements. The possible values are: continue execution, only show the error; stop execution and commit transaction; and abort execution and transaction and fail task.

Parameters

Attribute	Description	Required
driver	Class name of the jdbc driver	Yes
url	Database connection url	Yes
userid	Database user name	Yes
password	Database password	Yes
src	File containing SQL statements	Yes, unless statements enclosed within tags

Attribute	Description	Required
encoding	The encoding of the files containing SQL statements	No - defaults to default JVM encoding
delimiter	String that separates SQL statements	No, default ";"
autocommit	Auto commit flag for database connection (default false)	No, default "false"
print	Print result sets from the statements (default false)	No, default "false"
showheaders	Print headers for result sets from the statements (default true)	No, default "true"
output	Output file for result sets (defaults to System.out)	No (print to System.out by default)
append	whether output should be appended to or overwrite an existing file. Defaults to false.	No
classpath	Classpath used to load driver	No (use system classpath)
classpathref	The classpath to use, given as a reference to a path defined elsewhere.	No (use system classpath)
onerror	Action to perform when statement fails: continue, stop, abort	No, default "abort"
rdbms	Execute task only if this rdbms	No (no restriction)
version	Execute task only if rdbms version match	No (no restriction)
caching	Should the task cache loaders and the driver?	No (default=true)

Parameters specified as nested elements

transaction

Use nested `<transaction>` elements to specify multiple blocks of commands to the executed executed in the same connection but different transactions. This is particularly useful when there are multiple files to execute on the same schema.

Attribute	Description	Required
src	File containing SQL statements	Yes, unless statements enclosed within tags

fileset

You can specify multiple source files via nested fileset elements. Each file of the fileset will be run in a transaction of its own, the order by which the files of a single fileset will be executed is not defined.

classpath

Sql's classpath attribute is a PATH like structure and can also be set via a nested classpath element. It is used to load the JDBC classes.

Examples

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass"
    src="data.sql"
/>
```

Connects to the database given in url as the sa user using the org.database.jdbcDriver and executes the SQL statements contained within the file data.sql

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass"
    >
insert
into table some_table
values(1,2,3,4);

truncate table some_other_table;
</sql>
```

Connects to the database given in url as the sa user using the org.database.jdbcDriver and executes the two SQL statements inserting data into some_table and truncating some_other_table

Note that you may want to enclose your statements in `<![CDATA[ ... ]]>` sections so you don't need to escape `<`, `>` & or other special characters. For example:

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass"
    ><![CDATA[

update some_table set column1 = column1 + 1 where column2 < 42;

]]></sql>
```

The following connects to the database given in url as the sa user using the org.database.jdbcDriver and executes the SQL statements contained within the files data1.sql, data2.sql and data3.sql and then executes the truncate operation on some_other_table.

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass" >
  <transaction  src="data1.sql"/>
  <transaction  src="data2.sql"/>
  <transaction  src="data3.sql"/>
  <transaction>
    truncate table some_other_table;
  </transaction>
</sql>
```

The following example does the same as (and may execute additional SQL files if there are more files matching the pattern data*.sql) but doesn't guarantee that data1.sql will be run before data2.sql.

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass">
  <fileset dir=".">
    <include name="data*.sql"/>
  </fileset>
  <transaction>
    truncate table some_other_table;
  </transaction>
</sql>
```

The following connects to the database given in url as the sa user using the org.database.jdbcDriver and executes the SQL statements contained within the file data.sql, with output piped to outputfile.txt, searching /some/jdbc.jar as well as the system classpath for the driver class.

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass"
    src="data.sql"
    print="yes"
    output="outputfile.txt"
    >
<classpath>
        <pathelement location="/some/jdbc.jar"/>
</classpath>
</sql>
```

The following will only execute if the RDBMS is ”oracle” and the version starts with ”8.1.”

```
<sql
    driver="org.database.jdbcDriver"
    url="jdbc:database-url"
    userid="sa"
    password="pass"
    src="data.sql"
    rdbms="oracle"
    version="8.1."
    >
insert
into table some_table
values(1,2,3,4);

truncate table some_other_table;
</sql>
```

5.2.59 Xslt/Style

Description

Process a set of documents via XSLT.

This is useful for building views of XML based documentation, or for generating code.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

It is possible to refine the set of files that are being processed. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes basedir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

This task supports the use of a nested `<param>` element which is used to pass values to an ¡xsl:param¿ declaration.

This task supports the use of a nested xmlcatalog element which is used to perform Entity and URI resolution

`<style>` and `<xslt>` refer to the same Ant task and can be used interchangeably.

If you want to use Xalan-J 1 or XSL:P, you also need Ant's optional.jar

Parameters

Attribute	Description	Required
basedir	where to find the source XML file, default is the project's basedir.	No
destdir	directory in which to store the results.	Yes, unless in and out have been specified.
extension	desired file extension to be used for the targets. If not specified, the default is ".html".	No
style	name of the stylesheet to use - given either relative to the project's basedir or as an absolute path DEPRECATED - can be specified as a path relative to the basedir attribute of this task as well.	Yes
classpath	the classpath to use when looking up the XSLT processor.	No
classpathref	the classpath to use, given as reference to a path defined elsewhere.	No
force	Recreate target files, even if they are newer than their corresponding source files or the stylesheet.	No; default is false
processor	name of the XSLT processor to use. Permissible values are "trax" for a TraX compliant processor (ie JAXP interface implementation such as Xalan 2 or Saxon), "xslp" for the XSL:P processor, "xalan" for the Apache XML Xalan (version 1) processor the name of an arbitrary XSLTLiaison class. Defaults to trax, followed by xalan and then xslp (in that order). The first one found in your class path is the one that is used. DEPRECATED - XSL:P and xalan are deprecated and no more supported..	No

Attribute	Description	Required
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
in	specifies a single XML document to be styled. Should be used with the out attribute.	No
out	specifies the output name for the styled result from the in attribute.	No
scanincludeddirectories	If any directories are matched by the includes/excludes patterns, try to transform all files in these directories. Default is true	No
reloadstylesheet	Control whether the stylesheet transformer is created anew for every transform opertaion. If you set this to true, performance may suffer, but you may work around a bug in certain Xalan-J versions. Default is false. Since Ant 1.5.2.	No

Parameters specified as nested elements

classpath

The classpath to load the processor from can be specified via a nested ¡classpath¿, as well - that is, a path-like structure.

xmlcatalog

The xmlcatalog element is used to perform Entity and URI resolution.

param

Param is used to pass a parameter to the XSL stylesheet.

Parameters

Attribute	Description	Required
name	Name of the XSL parameter	Yes
expression	XSL expression to be placed into the param. To pass a text value into the style sheet it needs to be escaped using single quotes.	Yes

outputproperty ('trax' processors only)

Used to specify how you wish the result tree to be output as specified in the XSLT specifications.

Parameters

Attribute	Description	Required
name	Name of the property	Yes
value	value of the property.	Yes

Examples

```
<style basedir="doc" destdir="build/doc"
       extension=".html" style="style/apache.xsl"/>
```

Using an xmlcatalog

```
<xslt basedir="doc" destdir="build/doc"
      extension=".html" style="style/apache.xsl">
  <xmlcatalog refid="mycatalog"/>
</xslt>

<xslt basedir="doc" destdir="build/doc"
   extension=".html" style="style/apache.xsl">
   <xmlcatalog>
       <dtd
         publicId="-//ArielPartners//DTD XML Article V1.0//EN"
         location="com/arielpartners/knowledgebase/dtd/article.dtd"/>
   </xmlcatalog>
</xslt>
```

Using XSL parameters

```
<xslt basedir="doc" destdir="build/doc"
      extension=".html" style="style/apache.xsl">
  <param name="date" expression="07-01-2000"/>
</xslt>
```

Then if you declare a global parameter "date" with the top-level element ¡xsl:param name="date"/¿, the variable $date will subsequently have the value 07-01-2000.

Using output properties

```
<xslt in="doc.xml" out="build/doc/output.xml"
      style="style/apache.xsl">
  <outputproperty name="method" value="xml";/>
  <outputproperty name="standalone" value="yes"/>
  <outputproperty name="encoding" value="iso8859_1"/>
  <outputproperty name="indent" value="yes"/>
</xslt>
```

5.2.60 Tar

Description

Creates a tar archive.

The basedir attribute is the reference directory from where to tar.

This task is a directory based task and, as such, forms an implicit Fileset. This defines which files, relative to the basedir, will be included in the archive. The tar task supports all the attributes of Fileset to refine the set of files to be included in the implicit fileset.

In addition to the implicit fileset, the tar task supports nested filesets. These filesets are extended to allow control over the access mode, username and groupname to be applied to the tar entries. This is useful, for example, when preparing archives for Unix systems where some files need to have execute permission.

Early versions of tar did not support path lengths greater than 100 characters. Modern versions of tar do so, but in incompatible ways. The behaviour of the tar task when it encounters such paths is controlled by the longfile attribute. If the longfile attribute is set to fail, any long paths will cause the tar task to fail. If the longfile attribute is set to truncate, any long paths will be truncated to the 100 character maximum length prior to adding to the archive. If the value of the longfile attribute is set to omit then files containing long paths will be omitted from the archive. Either option ensures that the archive can be untarred by any compliant version of tar. If the loss of path or file information is not acceptable, and it rarely is, longfile may be set to the value gnu. The tar task will then produce a GNU tar file which can have arbitrary length paths. Note however, that the resulting archive will only be able to be untarred with GNU tar. The default for the longfile attribute is warn which behaves just like the gnu option except that it produces a warning for each file path encountered that does not match the limit.

This task can perform compression by setting the compression attribute to ”gzip” or ”bzip2”.

Parameters

Attribute	Description	Required
destfile	the tar-file to create.	Yes
basedir	the directory from which to tar the files.	No
longfile	Determines how long files (¿100 chars) are to be handled. Allowable values are "truncate", "fail", "warn", "omit" and "gnu". Default is "warn".	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
compression	compression method. Allowable values are "none", "gzip" and "bzip2". Default is "none".	No

Nested Elements

The tar task supports nested tarfileset elements. These are extended Filesets which, in addition to the standard fileset elements, support three additional attributes

Attribute	Description	Required
mode	A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion	No
username	The username for the tar entry. This is not the same as the UID, which is not currently set by the tar task.	No
group	The groupname for the tar entry. This is not the same as the GID, which is not currently set by the tar task.	No
prefix	If the prefix attribute is set, all files in the fileset are prefixed with that path in the archive.	No
fullpath	If the fullpath attribute is set, the file in the fileset is written with that path in the archive. The prefix attribute, if specified, is ignored. It is an error to have more than one file specified in such a fileset.	No
preserveLeadingSlashes	Indicates whether leading '/'s should be preserved in the file names. Default is false.	No

Examples

```
  <tar tarfile="${dist}/manual.tar" basedir="htdocs/manual"/>
  <gzip zipfile="${dist}/manual.tar.gz" src="${dist}/manual.tar"/>
```

tars all files in the htdocs/manual directory into a file called manual.tar in the ${dist} directory, then applies the gzip task to compress it.

```
  <tar destfile="${dist}/manual.tar"
       basedir="htdocs/manual"
       excludes="mydocs/**, **/todo.html"
  />
```

tars all files in the htdocs/manual directory into a file called manual.tar in the ${dist} directory. Files in the directory mydocs, or files with the name todo.html are excluded.

```
<tar destfile="${basedir}/docs.tar">
  <tarfileset dir="${dir.src}/docs"
              fullpath="/usr/doc/ant/README"
              preserveLeadingSlashes="true">
    <include name="readme.txt"/>
  </tarfileset>
  <tarfileset dir="${dir.src}/docs"
              prefix="/usr/doc/ant"
              preserveLeadingSlashes="true">
    <include name="*.html"/>
  </tarfileset>
</tar>
```

Writes the file docs/readme.txt as /usr/doc/ant/README into the archive. All *.html files in the docs directory are prefixed by /usr/doc/ant, so for example docs/index.html is written as /usr/doc/ant/index.html to the archive.

```
<tar longfile="gnu"
     destfile="${dist.base}/${dist.name}-src.tar" >
  <tarfileset dir="${dist.name}/.." mode="755" username="ant" group="ant">
    <include name="${dist.name}/bootstrap.sh"/>
    <include name="${dist.name}/build.sh"/>
  </tarfileset>
  <tarfileset dir="${dist.name}/.." username="ant" group="ant">
    <include name="${dist.name}/**"/>
    <exclude name="${dist.name}/bootstrap.sh"/>
    <exclude name="${dist.name}/build.sh"/>
  </tarfileset>
</tar>
```

This example shows building a tar which uses the GNU extensions for long paths and where some files need to be marked as executable (mode 755) and the rest are use the default mode (read-write by owner). The first fileset selects just the executable files. The second fileset must exclude the executable files and include all others.

Note: The tar task does not ensure that a file is only selected by one fileset. If the same file is selected by more than one fileset, it will be included in the tar file twice, with the same path.

Note: The patterns in the include and exclude elements are considered to be relative to the corresponding dir attribute as with all other filesets. In the example above, ${dist.name} is not an absolute path, but a simple name of a directory, so ${dist.name} is a valid path relative to ${dist.name}/...

5.2.61 Taskdef

Description

Adds a task definition to the current project, such that this new task can be used in the current project. Two attributes are needed, the name that identifies this task uniquely, and the full name of the class (including the packages) that implements this task.

You can also define a group of tasks at once using the file or resource attributes. These attributes point to files in the format of Java property files. Each line defines a single task in the format:

```
taskname=fully.qualified.java.classname
```

Taskdef should be used to add your own tasks to the system. See also ”Writing your own task”.

Parameters

Attribute	Description	Required
name	the name of the task	Yes, unless file or resource have been specified.
classname	the full class name implementing the task	Yes, unless file or resource have been specified.
file	Name of the property file to load taskname/classname pairs from.	No
resource	Name of the property resource to load taskname/classname pairs from.	No
classpath	the classpath to use when looking up classname or resource.	No
classpathref	Reference to a classpath to use when looking up classname or resource.	No
loaderRef	the name of the loader that is used to load the class, constructed from the specified classpath. Use this to allow multiple tasks/types to be loaded with the same loader, so they can call each other. (introduced in ant1.5)	No

Parameters specified as nested elements

classpath

Taskdef's classpath attribute is a PATH like structure and can also be set via a nested classpath element.

Examples

```
<taskdef name="myjavadoc" classname="com.mydomain.JavadocTask"/>
```

makes a task called myjavadoc available to Ant. The class com.mydomain.JavadocTask implements the task.

5.2.62 Tempfile

Description

This task sets a property to the name of a temporary file. Unlike the Java1.2 method to create a temporary file, this task does work with Java1.1. It does not actually create the temporary file, but it does guarantee that the file did not exist when the task was executed.

Parameters

Attribute	Description	Required
destdir	The directory the temporary file should be located in. If not set, the current directory is used.	No
prefix	A prefix for the temporary file name.	No
property	The name of the property to set with the value of the temporary file name.	Yes
suffix	A suffix for the temporary file name.	No

Examples

```
<tempfile property="temp.file"/>
```

will set temp.file to the name of a new temporary file.

```
<tempfile property="temp.file" suffix=".xml"/>
```

will set temp.file to the name of a new temporary file with a suffix of .xml.

```
<tempfile property="temp.file" destdir="build"/>
```

will set temp.file to the name of a new temporary file located in the build subdirectory.

5.2.63 Touch

Description

Changes the modification time of a file and possibly creates it at the same time. In addition to working with a single file, this Task can also work a Fileset (which also includes directories).

For JDK 1.1 only the creation of new files with a modification time of now works, all other cases will emit a warning.

Parameters

Attribute	Description	Required
file	the name of the file	unless a nested fileset element has been specified.
millis	specifies the new modification time of the file in milliseconds since midnight Jan 1 1970	No
datetime	specifies the new modification time of the file in the format MM/DD/YYYY HH:MM AM_or_PM.	No

Attribute	Description	Required

If both millis and datetime are omitted the current time is assumed.

Examples

```
<touch file="myfile"/>
```

creates myfile if it doesn't exist and changes the modification time to the current time.

```
<touch file="myfile" datetime="06/28/2000 2:02 pm"/>
```

creates myfile if it doesn't exist and changes the modification time to Jun, 28 2000 2:02 pm (14:02 for those used to 24 hour times).

```
<touch datetime="09/10/1974 4:30 pm">
  <fileset dir="src_dir"/>
</touch>
```

changes the modification time to Oct, 09 1974 4:30 pm of all files and directories found in src_dir.

5.2.64 TStamp

Description

Sets the DSTAMP, TSTAMP, and TODAY properties in the current project. By default, the DSTAMP property is in the format ”yyyyMMdd”, TSTAMP is in the format ”hhmm”, and TODAY is in the format ”MMMM dd yyyy”. Use the nested ¡format¿ element to specify a different format.

These properties can be used in the build-file, for instance, to create time-stamped filenames, or used to replace placeholder tags inside documents to indicate, for example, the release date. The best place for this task is probably in an initialization target.

Parameters

Attribute	Description	Required
prefix	Prefix used for all properties set. The default is no prefix.	No

Nested Elements

The Tstamp task supports a ¡format¿ nested element that allows a property to be set to the current date and time in a given format. The date/time patterns are as defined in the Java SimpleDateFormat class. The format element also allows offsets to be applied to the time to generate different time values.

Attribute	Description	Required
property	The property to receive the date/time string in the given pattern.	Yes
pattern	The date/time pattern to be used. The values are as defined by the Java SimpleDateFormat class.	Yes
timezone	The timezone to use for displaying time. The values are as defined by the Java TimeZone class.	No
offset	The numeric offset to the current time	No
unit	The unit of the offset to be applied to the current time. Valid Values are • millisecond • second • minute • hour • day • week • month • year	No
locale	The locale used to create date/time string. The general form is ”language, country, variant” but either variant or variant and country may be omitted. For more information please refer to documentation for the Locale class.	No

Examples

```
<tstamp/>
```

sets the standard DSTAMP, TSTAMP, and TODAY properties according to the default formats.

```
<tstamp>
  <format property="TODAY_UK" pattern="d-MMMM-yyyy" locale="en"/>
</tstamp>
```

sets the standard properties as well as the property `TODAY_UK` with the date/time pattern ”d-MMMM-yyyy” using English locale (eg. 21-May-2001).

```
<tstamp>
```

```
    <format property="touch.time" pattern="MM/dd/yyyy hh:mm aa"
            offset="-5" unit="hour"/>
</tstamp>
```

Creates a timestamp, in the property touch.time, 5 hours before the current time. The format in this example is suitable for use with the ¡touch¿ task. The standard properties are set also.

```
<tstamp prefix="start"/>
```

Sets three properties with the standard formats, prefixed with ”start.”: start.DSTAMP, start.TSTAMP, and start.TODAY.

5.2.65 Typedef

Description

Adds a data type definition to the current project, such that this new type can be used in the current project. Two attributes are needed, the name that identifies this data type uniquely, and the full name of the class (including the packages) that implements this type.

You can also define a group of data types at once using the file or resource attributes. These attributes point to files in the format of Java property files. Each line defines a single data type in the format:

```
typename=fully.qualified.java.classname
```

Typedef should be used to add your own types to the system. Data types are things like paths or filesets that can be defined at the project level and referenced via their ID attribute.

Custom data types usually need custom tasks to put them to good use.

Parameters

Attribute	Description	Required
name	the name of the data type	Yes, unless file or resource have been specified.
classname	the full class name implementing the data type	Yes, unless file or resource have been specified.
file	Name of the property file to load typename/classname pairs from.	No
resource	Name of the property resource to load typename/classname pairs from.	No
classpath	the classpath to use when looking up classname.	No

Attribute	Description	Required
loaderRef	the name of the loader that is used to load the class, constructed from the specified classpath. Use this to allow multiple tasks/types to be loaded with the same loader, so they can call each other. (introduced in ant1.5)	No

Parameters specified as nested elements

classpath

Typedef's classpath attribute is a PATH like structure and can also be set via a nested classpath element.

Examples

```
<typedef name="urlset" classname="com.mydomain.URLSet"/>
```

makes a data type called urlset available to Ant. The class com.mydomain.URLSet implements this type.

5.2.66 Unjar/Untar/Unwar/Unzip

Description

Unzips a zip-, war-, tar- or jarfile.

For JDK 1.1 "last modified time" field is set to current time instead of being carried from the archive file.

PatternSets are used to select files to extract from the archive. If no patternset is used, all files are extracted.

FileSets may be used used to select archived files to perform unarchival upon.

File permissions will not be restored on extracted files.

The untar task recognizes the long pathname entries used by GNU tar.

Parameters

Attribute	Description	Required
src	archive file to expand.	Yes, if filesets are not used.
dest	directory where to store the expanded files.	Yes
overwrite	Overwrite files, even if they are newer than the corresponding entries in the archive (true or false, default is true).	No
compression	compression method for untar. Allowable values are "none", "gzip" and "bzip2". Default is "none".	No

Attribute	Description	Required

Examples

```
<unzip src="${tomcat_src}/tools-src.zip" dest="${tools.home}"/>

<gunzip src="tools.tar.gz"/>
<untar src="tools.tar" dest="${tools.home}"/>

<unzip src="${tomcat_src}/tools-src.zip"
       dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
</unzip>

<unzip dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
    <fileset dir=".">
        <include name="**/*.zip"/>
        <exclude name="**/tmp*.zip"/>
    </fileset>
</unzip>
```

5.2.67 Untar

Description

Unzips a zip-, war-, tar- or jarfile.

For JDK 1.1 ”last modified time” field is set to current time instead of being carried from the archive file.

PatternSets are used to select files to extract from the archive. If no patternset is used, all files are extracted.

FileSets may be used used to select archived files to perform unarchival upon.

File permissions will not be restored on extracted files.

The untar task recognizes the long pathname entries used by GNU tar.

Parameters

Attribute	Description	Required
src	archive file to expand.	Yes, if filesets are not used.

Attribute	Description	Required
dest	directory where to store the expanded files.	Yes
overwrite	Overwrite files, even if they are newer than the corresponding entries in the archive (true or false, default is true).	No
compression	compression method for untar. Allowable values are ”none”, ”gzip” and ”bzip2”. Default is ”none”.	No

Examples

```
<unzip src="${tomcat_src}/tools-src.zip" dest="${tools.home}"/>

<gunzip src="tools.tar.gz"/>
<untar src="tools.tar" dest="${tools.home}"/>

<unzip src="${tomcat_src}/tools-src.zip"
       dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
</unzip>

<unzip dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
    <fileset dir=".">
        <include name="**/*.zip"/>
        <exclude name="**/tmp*.zip"/>
    </fileset>
</unzip>
```

5.2.68 Unwar

Description

Unzips a zip-, war-, tar- or jarfile.

For JDK 1.1 ”last modified time” field is set to current time instead of being carried from the archive file.

PatternSets are used to select files to extract from the archive. If no patternset is used, all files are extracted.

FileSets may be used used to select archived files to perform unarchival upon.

File permissions will not be restored on extracted files.

The untar task recognizes the long pathname entries used by GNU tar.

Parameters

Attribute	Description	Required
src	archive file to expand.	Yes, if filesets are not used.
dest	directory where to store the expanded files.	Yes
overwrite	Overwrite files, even if they are newer than the corresponding entries in the archive (true or false, default is true).	No
compression	compression method for untar. Allowable values are ”none”, ”gzip” and ”bzip2”. Default is ”none”.	No

Examples

```
<unzip src="${tomcat_src}/tools-src.zip" dest="${tools.home}"/>

<gunzip src="tools.tar.gz"/>
<untar src="tools.tar" dest="${tools.home}"/>

<unzip src="${tomcat_src}/tools-src.zip"
       dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
</unzip>

<unzip dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
    <fileset dir=".">
        <include name="**/*.zip"/>
        <exclude name="**/tmp*.zip"/>
    </fileset>
</unzip>
```

5.2.69 Unzip

Unzips a zip-, war-, tar- or jarfile.

For JDK 1.1 ”last modified time” field is set to current time instead of being carried from the archive file.

PatternSets are used to select files to extract from the archive. If no patternset is used, all files are extracted.

FileSets may be used used to select archived files to perform unarchival upon.

File permissions will not be restored on extracted files.

The untar task recognizes the long pathname entries used by GNU tar.

Parameters

Attribute	Description	Required
src	archive file to expand.	Yes, if filesets are not used.
dest	directory where to store the expanded files.	Yes
overwrite	Overwrite files, even if they are newer than the corresponding entries in the archive (true or false, default is true).	No
compression	compression method for untar. Allowable values are ”none”, ”gzip” and ”bzip2”. Default is ”none”.	No

Examples

```
<unzip src="${tomcat_src}/tools-src.zip" dest="${tools.home}"/>

<gunzip src="tools.tar.gz"/>
<untar src="tools.tar" dest="${tools.home}"/>

<unzip src="${tomcat_src}/tools-src.zip"
       dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
</unzip>

<unzip dest="${tools.home}">
    <patternset>
        <include name="**/*.java"/>
        <exclude name="**/Test*.java"/>
    </patternset>
    <fileset dir=".">
        <include name="**/*.zip"/>
        <exclude name="**/tmp*.zip"/>
    </fileset>
</unzip>
```

5.2.70 Uptodate

Description

Sets a property if a target file or set of target files is more up-to-date than a source file or set of source files. A single source file is specified using the srcfile attribute. A set of source files is specified using the nested ¡srcfiles¿ elements. These are FileSets, whereas multiple target files are specified using a nested ¡mapper¿ element.

By default, the value of the property is set to true if the timestamp of the target file(s) is more recent than the timestamp of the corresponding source file(s). You can set the value to something other than the default by specifying the value attribute.

If a ¡srcfiles¿ element is used, without also specifying a ¡mapper¿ element, the default behavior is to use a merge mapper, with the to attribute set to the value of the targetfile attribute.

Normally, this task is used to set properties that are useful to avoid target execution depending on the relative age of the specified files.

Parameters

Attribute	Description	Required
property	The name of the property to set.	Yes
value	The value to set the property to.	No; defaults to true.
srcfile	The file to check against the target file(s).	Yes, unless a nested ¡srcfiles¿ element is present.
targetfile	The file for which we want to determine the status.	Yes, unless a nested ¡mapper¿ element is present.

Parameters specified as nested elements

srcfiles

The nested `<srcfiles>` element allows you to specify a set of files to check against the target file(s).

Note: You can specify either the srcfile attribute or nested `<srcfiles>` elements, but not both.

mapper

The nested `<mapper>` element allows you to specify a set of target files to check for being up-to-date with respect to a set of source files.

Examples

```
<uptodate property="xmlBuild.notRequired"
 targetfile="${deploy}\xmlClasses.jar" >
  <srcfiles dir= "${src}/xml" includes="**/*.dtd"/>
</uptodate>
```

sets the property xmlBuild.notRequired to true if the ${deploy}/xmlClasses.jar file is more up-to-date than any of the DTD files in the ${src}/xml directory.

This can be written as:

```
<uptodate property="xmlBuild.notRequired">
  <srcfiles dir= "${src}/xml" includes="**/*.dtd"/>
  <mapper type="merge" to="${deploy}\xmlClasses.jar"/>
</uptodate>
```

as well. The xmlBuild.notRequired property can then be used in a `<target>` tag's unless attribute to conditionally run that target. For example, running the following target:

```
<target name="xmlBuild" depends="chkXmlBuild" unless="xmlBuild.notRequired">
  ...
</target>
```

will first run the chkXmlBuild target, which contains the ¡uptodate¿ task that determines whether xmlBuild.notRequired gets set. The property named in the unless attribute is then checked for being set/not set. If it did get set (ie., the jar file is up-to-date), then the xmlBuild target won't be run.

The following example shows a single source file being checked against a single target file:

```
<uptodate property="isUpToDate"
          srcfile="/usr/local/bin/testit"
          targetfile="${build}/.flagfile"/>
```

sets the property isUpToDate to true if /usr/local/bin/testit is newer than ${build}/.flagfile.

5.2.71 Waitfor

Description

Blocks execution until a set of specified conditions become true. This is intended to be used with the parallel task to synchronize a set of processes.

The conditions to wait for are defined in nested elements, if multiple conditions are specified, then the task will wait until all conditions are true..

If both maxwait and maxwaitunit are not specified, the maxwait is 3 minutes (180000 milliseconds).

If the timeoutproperty attribute has been set, a property of that name will be created if the condition didn't come true within the specified time.

Parameters

Attribute	Description	Required
maxwait	The maximum amount of time to wait for all the required conditions to become true before failing the task. Defaults to 180000 maxwaitunits.	No
maxwaitunit	The unit of time that must be used to interpret the value of the maxwait attribute. Defaults to millisecond. Valid Values are • millisecond • second • minute • hour • day • week	No
checkevery	The amount of time to wait between each test of the conditions. Defaults to 500 checkeveryunits.	No
checkeveryunit	The unit of time that must be used to interpret the value of the checkevery attribute. Defaults to millisecond. Valid Values are • second • minute • hour • day • week	No

Attribute	Description	Required
timeoutproperty	the name of the property to set if maxwait has been exceeded.	No

Nested Elements

The available conditions that satisfy the `<waitfor>` task are the same as those for the `<condition>` task. See here for the full list.

Examples

```
<waitfor maxwait="30" maxwaitunit="second">
    <available file="errors.log"/>
</waitfor>
```

waits up to 30 seconds for a file called errors.log to appear.

```
<waitfor maxwait="3" maxwaitunit="minute" checkevery="500">
    <http url="http://localhost/myapp/index.html"/>
</waitfor>
```

waits up to 3 minutes (and checks every 500 milliseconds) for a web server on localhost to serve up the specified URL.

```
<waitfor maxwait="10" maxwait="second">
    <and>
        <socket server="dbserver" port="1521"/>
        <http url="http://webserver/mypage.html"/>
    </and>
</waitfor>
```

waits up to 10 seconds for a server on the dbserver machine to begin listening on port 1521 and for the http://webserver/mypage.html web page to become available.

5.2.72 War

Description

An extension of the Jar task with special treatment for files that should end up in the WEB-INF/lib, WEB-INF/classes or WEB-INF directories of the Web Application Archive.

(The War task is a shortcut for specifying the particular layout of a WAR file. The same thing can be accomplished by using the prefix and fullpath attributes of zipfilesets in a Zip or Jar task.)

The extended zipfileset element from the zip task (with attributes prefix, fullpath, and src) is available in the War task.

Parameters

Attribute	Description	Required
destfile	the WAR file to create.	Yes
warfile	Deprecated name of the file to create -use destfile instead.	No
webxml	The deployment descriptor to use (WEB-INF/web.xml).	Yes, unless update is set to true
basedir	the directory from which to jar the files.	No
compress	Not only store data but also compress them, defaults to true	No
encoding	The character encoding to use for filenames inside the archive. Defaults to UTF8. It is not recommended to change this value as the created archive will most likely be unreadable for Java otherwise.	No
filesonly	Store only file entries, defaults to false	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No
manifest	the manifest file to use.	No
update	indicates whether to update or overwrite the destination file if it already exists. Default is ”false”.	No
duplicate	behavior when a duplicate file is found. Valid values are ”add”, ”preserve”, and ”fail”. The default value is ”add”.	No

Nested elements

lib

The nested lib element specifies a FileSet. All files included in this fileset will end up in the WEB-INF/lib directory of the war file.

classes

The nested classes element specifies a FileSet. All files included in this fileset will end up in the WEB-INF/classes directory of the war file.

webinf

The nested webinf element specifies a FileSet. All files included in this fileset will end up in the WEB-INF directory of the war file. If this fileset includes a file named web.xml, the file is ignored and you will get a warning.

metainf

The nested metainf element specifies a FileSet. All files included in this fileset will end up in the META-INF directory of the war file. If this fileset includes a file named MANIFEST.MF, the file is ignored and you will get a warning.

Examples

Assume the following structure in the project's base directory:

```
thirdparty/libs/jdbc1.jar
thirdparty/libs/jdbc2.jar
build/main/com/myco/myapp/Servlet.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
```

then the war file myapp.war created with

```
<war destfile="myapp.war" webxml="src/metadata/myapp.xml">
  <fileset dir="src/html/myapp"/>
  <fileset dir="src/jsp/myapp"/>
  <lib dir="thirdparty/libs">
    <exclude name="jdbc1.jar"/>
  </lib>
  <classes dir="build/main"/>
  <zipfileset dir="src/graphics/images/gifs"
              prefix="images"/>
</war>
```

will consist of

```
WEB-INF/web.xml
WEB-INF/lib/jdbc2.jar
WEB-INF/classes/com/myco/myapp/Servlet.class
META-INF/MANIFEST.MF
index.html
front.jsp
images/small/logo.gif
images/large/logo.gif
```

using Ant's default manifest file. The content of WEB-INF/web.xml is identical to src/metadata/myapp.xml.

5.2.73 XmlProperty

Description

Loads property values from a valid xml file.

Parameters

Attribute	Description	Required
file	The XML file to parse.	Yes
prefix	The prefix to prepend to each property	No
keepRoot	If false, it doesn't include the xml root tag as a first value in the property name.	No, default is true.
validate	If true, it enables validation.	No, default is false.
collapseAttributes	If true, it treats attributes as nested elements.	No, default is false.

Examples

```
<xmlproperty file="somefile.xml" />
```

Load contents of somefile.xml as Ant properties, generating the property names from the file's element and attribute names.

```
<root-tag myattr="true">
 <inner-tag someattr="val">Text</inner-tag>
 <a2><a3><a4>false</a4></a3></a2>
</root-tag>
```

This is an example xml file.

```
root-tag(myattr)=true
root-tag.inner-tag=Text
root-tag.inner-tag(someattr)=val
root-tag.a2.a3.a4=false
```

These are the properties loaded by this task from the previous example file.

```
<xmlproperty file="somefile.xml" collapseAttributes="true"/>
```

Load contents of somefile.xml as Ant properties collapsing attributes as nodes.

```
root-tag.myattr=true
root-tag.inner-tag=Text
root-tag.inner-tag.someatt=val
root-tag.a2.a3.a4=false
```

These are the properties loaded by this task from the previous example file, with attribute collapsing true.

5.2.74 Xslt

Description

Process a set of documents via XSLT.

This is useful for building views of XML based documentation, or for generating code.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

It is possible to refine the set of files that are being processed. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes basedir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

This task supports the use of a nested `<param>` element which is used to pass values to an `<xsl:param>` declaration.

This task supports the use of a nested xmlcatalog element which is used to perform Entity and URI resolution

`<style>` and `<xslt>` refer to the same Ant task and can be used interchangeably.

If you want to use Xalan-J 1 or XSL:P, you also need Ant's optional.jar

Parameters

Attribute	Description	Required
basedir	where to find the source XML file, default is the project's basedir.	No
destdir	directory in which to store the results.	Yes, unless in and out have been specified.
extension	desired file extension to be used for the targets. If not specified, the default is ".html".	No
style	name of the stylesheet to use - given either relative to the project's basedir or as an absolute path DEPRECATED - can be specified as a path relative to the basedir attribute of this task as well.	Yes

Attribute	Description	Required
classpath	the classpath to use when looking up the XSLT processor.	No
classpathref	the classpath to use, given as reference to a path defined elsewhere.	No
force	Recreate target files, even if they are newer than their corresponding source files or the stylesheet.	No; default is false
processor	name of the XSLT processor to use. Permissible values are "trax" for a TraX compliant processor (ie JAXP interface implementation such as Xalan 2 or Saxon), "xslp" for the XSL:P processor, "xalan" for the Apache XML Xalan (version 1) processor the name of an arbitrary XSLTLiaison class. Defaults to trax, followed by xalan and then xslp (in that order). The first one found in your class path is the one that is used. DEPRECATED - XSL:P and xalan are deprecated and no more supported..	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
in	specifies a single XML document to be styled. Should be used with the out attribute.	No
out	specifies the output name for the styled result from the in attribute.	No
scanincludeddirectories	If any directories are matched by the includes/excludes patterns, try to transform all files in these directories. Default is true	No

Attribute	Description	Required
reloadstylesheet	Control whether the stylesheet transformer is created anew for every transform opertaion. If you set this to true, performance may suffer, but you may work around a bug in certain Xalan-J versions. Default is false. Since Ant 1.5.2.	No

Parameters specified as nested elements

classpath

The classpath to load the processor from can be specified via a nested ¡classpath¿, as well - that is, a path-like structure.

xmlcatalog

The xmlcatalog element is used to perform Entity and URI resolution.

param

Param is used to pass a parameter to the XSL stylesheet.

Parameters

Attribute	Description	Required
name	Name of the XSL parameter	Yes
expression	XSL expression to be placed into the param. To pass a text value into the style sheet it needs to be escaped using single quotes.	Yes

outputproperty ('trax' processors only)

Used to specify how you wish the result tree to be output as specified in the XSLT specifications.

Parameters

Attribute	Description	Required
name	Name of the property	Yes
value	value of the property.	Yes

Examples

```
<style basedir="doc" destdir="build/doc"
       extension=".html" style="style/apache.xsl"/>
```

Using an xmlcatalog

```
<xslt basedir="doc" destdir="build/doc"
      extension=".html" style="style/apache.xsl">
```

```
  <xmlcatalog refid="mycatalog"/>
</xslt>

<xslt basedir="doc" destdir="build/doc"
   extension=".html" style="style/apache.xsl">
   <xmlcatalog>
       <dtd
         publicId="-//ArielPartners//DTD XML Article V1.0//EN"
         location="com/arielpartners/knowledgebase/dtd/article.dtd"/>
   </xmlcatalog>
</xslt>
```

Using XSL parameters

```
<xslt basedir="doc" destdir="build/doc"
      extension=".html" style="style/apache.xsl">
  <param name="date" expression="07-01-2000"/>
</xslt>
```

Then if you declare a global parameter "date" with the top-level element ¡xsl:param name="date"/¿, the variable $date will subsequently have the value 07-01-2000. Using output properties

```
<xslt in="doc.xml" out="build/doc/output.xml"
      style="style/apache.xsl">
  <outputproperty name="method" value="xml";/>
  <outputproperty name="standalone" value="yes"/>
  <outputproperty name="encoding" value="iso8859_1"/>
  <outputproperty name="indent" value="yes"/>
</xslt>
```

5.2.75 Zip

Description

Creates a zipfile.

The basedir attribute is the reference directory from where to zip.

Note that file permissions will not be stored in the resulting zipfile.

It is possible to refine the set of files that are being zipped. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes basedir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

Or, you may place within it nested file sets, or references to file sets. In this case basedir is optional; the implicit file set is only used if basedir is set. You may use any mixture of the implicit file set (with basedir set, and optional attributes like includes and optional subelements like `<include>`); explicit nested `<fileset>` elements so long as at least one fileset total is specified. The ZIP file will only reflect the relative paths of files within each fileset. The Zip task and its derivatives know a special form of a fileset named zipfileset that has additional attributes (described below).

The Zip task also supports the merging of multiple zip files into the zip file. This is possible through either the src attribute of any nested filesets or by using the special nested fileset zipgroupfileset.

The update parameter controls what happens if the ZIP file already exists. When set to yes, the ZIP file is updated with the files specified. (New files are added; old files are replaced with the new versions.) When set to no (the default) the ZIP file is overwritten. Please note that ZIP files store file modification times with a granularity of two seconds. If a file is less than two seconds newer than the entry in the archive, Ant will not consider it newer.

The whenempty parameter controls what happens when no files match. If skip (the default), the ZIP is not created and a warning is issued. If fail, the ZIP is not created and the build is halted with an error. If create, an empty ZIP file (explicitly zero entries) is created, which should be recognized as such by compliant ZIP manipulation tools.

This task will now use the platform's default character encoding for filenames - this is consistent with the command line ZIP tools, but causes problems if you try to open them from within Java and your filenames contain non US-ASCII characters. Use the encoding attribute and set it to UTF8 to create zip files that can safely be read by Java.

Starting with Ant 1.5.2, `<zip>` can store Unix permissions inside the archive (see description of the filemode and dirmode attributes for `<zipfileset>`). Unfortunately there is no portable way to store these permissions. Ant uses the algorithm used by Info-Zip's implementation of the zip and unzip commands - these are the default versions of zip and unzip for many Unix and Unix-like systems.

Parameters

Attribute	Description	Required
destfile	the zip-file to create.	Yes
zipfile	the deprecated old name of destfile.	Yes
basedir	the directory from which to zip the files.	No
compress	Not only store data but also compress them, defaults to true	No
encoding	The character encoding to use for filenames inside the zip file. For a list of possible values see http://java.sun.com/products/jdk/1.2/docs/guide /internat/encoding.doc.html. Defaults to the platform's default character encoding.	No
filesonly	Store only file entries, defaults to false	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	No
update	indicates whether to update or overwrite the destination file if it already exists. Default is ”false”.	No
whenempty	behavior when no files match. Valid values are ”fail”, ”skip”, and ”create”. Default is ”skip”.	No
duplicate	behavior when a duplicate file is found. Valid values are ”add”, ”preserve”, and ”fail”. The default value is ”add”.	No

Parameters specified as nested elements

fileset

The zip task supports any number of nested `<fileset>` elements to specify the files to be included in the archive.

zipfileset

A `<zipfileset>` is a special form of a `<fileset>` that adds some extra functionality. It supports all attributes of `<fileset>` in addition to those listed below.

Parameters

Attribute	Description	Required
prefix	all files in the fileset are prefixed with that path in the archive.	No
fullpath	the file described by the fileset is placed at that exact location in the archive.	No
src	may be used in place of the dir attribute to specify a zip file whose contents will be extracted and included in the archive.	No
filemode	A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion. Only applies to plain files. Default is 644. since Ant 1.5.2.	No
dirmode	A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion. Only applies to directories. Default is 755. since Ant 1.5.2.	No

The fullpath attribute can only be set for filesets that represent a single file. The prefix and fullpath attributes cannot both be set on the same fileset.

When using the src attribute, include and exclude patterns may be used to specify a subset of the zip file for inclusion in the archive as with the dir attribute.

zipgroupfileset

A `<zipgroupfileset>` allows for multiple zip files to be merged into the archive. Each file found in this fileset is added to the archive the same way that zipfileset src files are added.

Examples

```
<zip destfile="${dist}/manual.zip"
     basedir="htdocs/manual"
/>
```

zips all files in the htdocs/manual directory into a file called manual.zip in the ${dist} directory.

```
<zip destfile="${dist}/manual.zip"
     basedir="htdocs/manual"
     update="true"
/>
```

zips all files in the htdocs/manual directory into a file called manual.zip in the ${dist} directory. If manual.zip doesn't exist, it is created; otherwise it is updated with the new/changed files.

```
<zip destfile="${dist}/manual.zip"
     basedir="htdocs/manual"
```

```
       excludes="mydocs/**, **/todo.html"
  />
```

zips all files in the htdocs/manual directory. Files in the directory mydocs, or files with the name todo.html are excluded.

```
  <zip destfile="${dist}/manual.zip"
       basedir="htdocs/manual"
       includes="api/**/*.html"
       excludes="**/todo.html"
  />
```

zips all files in the htdocs/manual directory. Only html files under the directory api are zipped, and files with the name todo.html are excluded.

```
  <zip destfile="${dist}/manual.zip">
    <fileset dir="htdocs/manual"/>
    <fileset dir="." includes="ChangeLog.txt"/>
  </zip>
```

zips all files in the htdocs/manual directory, and also adds the file ChangeLog.txt in the current directory. ChangeLog.txt will be added to the top of the ZIP file, just as if it had been located at htdocs/manual/ChangeLog.txt.

```
  <zip destfile="${dist}/manual.zip">
    <zipfileset dir="htdocs/manual" prefix="docs/user-guide"/>
    <zipfileset dir="." includes="ChangeLog27.txt" fullpath="docs/ChangeLog.txt"/>
    <zipfileset src="examples.zip" includes="**/*.html" prefix="docs/examples"/>
  </zip>
```

zips all files in the htdocs/manual directory into the docs/user-guide directory in the archive, adds the file ChangeLog27.txt in the current directory as docs/ChangeLog.txt, and includes all the html files in examples.zip under docs/examples. The archive might end up containing the files:

```
    docs/user-guide/html/index.html
    docs/ChangeLog.txt
    docs/examples/index.html
```

The code

```
  <zip destfile="${dist}/manual.zip">
    <zipfileset dir="htdocs/manual" prefix="docs/user-guide"/>
    <zipgroupfileset dir="." includes="examples*.zip"/>
  </zip>
```

zips all files in the htdocs/manual directory into the docs/user-guide directory in the archive and includes all the files in any file that maches examples*.zip, such as all files within examples1.zip or examples_for_brian.zip.

5.3 Optional Tasks

5.3.1 NET Tasks

`<CSC>`

This task compiles CSharp source into executables or modules. This task compiles CSharp source into executables or modules. The task will only work on win2K/XP or other platforms with csc.exe or an equivalent. CSC must be on the execute path.

All parameters are optional: `<csc/>` should suffice to produce a debug build of all *.cs files. References to external files do require explicit enumeration, so are one of the first attributes to consider adding.

The task is a directory based task, so attributes like includes=”**/*.cs” and excludes=”broken.cs” can be used to control the files pulled in. By default, all *.cs files from the project folder down are included in the command. When this happens the destFile -if not specified- is taken as the first file in the list, which may be somewhat hard to control. Specifying the output file with ’destfile’ seems prudent.

Also, dependency checking only works if destfile is set.

Attribute	Description	Required
Attribute	Description	Example Values
additionalModules	Semicolon separated list of modules to refer to	
defaultexcludes	indicates whether default excludes should be used or not	”true”(default) or ”false”
definitions	defined constants	”RELEASE;BETA1”
debug	include debug information	”true”(default)
destFile	name of exe/library to create	”example.exe”
docFile	name of file for documentation	”doc.xml”
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	
extraOptions	Any extra options which aren’t explicitly supported by the CSharp task	”/warnaserror+ /baseaddress:0x12840000”
failOnError	Should a failed compile halt the build?	”true”(default) or ”false”

Attribute	Description	Required
fileAlign	set the file alignment. Valid values are 0,512, 1024, 2048, 4096, 8192, and 16384 0 means 'leave to the compiler'	512
fullpaths	print the full path of files on on errors	
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	
includeDefaultReferences	Flag which when true automatically includes the common assemblies in dotnet, and tells the compiler to link in mscore.dll	"true"(default) or "false"
includesfile	the name of a file. Each line of this file is taken to be an include pattern	
incremental	Incremental build flag. Avoid till it works	"true" or "false"(default)
mainClass	name of main class for executables	"com.example.project.entrypoint"
noConfig	a flag which tells the compiler not to read in the compiler settings files 'csc.rsp' in its bin directory and then the local directory	"true" or "false"(default)
optimize	optimisation flag	"true" or "false"(default)
references	Semicolon separated list of dlls to refer to	"mylib.dll;nunit.dll"
referenceFiles	Ant Path descriptioon of references to include. Wildcards should work.	
srcDir	source directory (default = project directory)	"."
targetType	Type of target	"exe", "module", "winexe" or "library"
unsafe	enable the unsafe keyword	"true" or "false"(default)
utf8output	require all compiler output to be in utf-8 format	"true" or "false"(default)
warnLevel	level of warning currently between 1 and 4 with 4 being the strictest.	"1"-"4"

Attribute	Description	Required
win32Icon	filename of icon to include	”res/myicon.ico”
win32res	filename of a win32 resource (.RES)file to include This is not a .NET resource, but it what windows is used to.	”res/myapp.res”

Example

```
<csc
        optimize="true"
        debug="false"
        docFile="documentation.xml"
        warnLevel="4"
        unsafe="false"
        targetType="exe"
        incremental="false"
        definitions="RELEASE"
        excludes="src/unicode_class.cs"
        mainClass = "MainApp"
        destFile="NetApp.exe"
        />
```

`<ilasm>`

Task to assemble .net ’Intermediate Language’ files. The task will only work on windows until other platforms support csc.exe or an equivalent. ilasm.exe must be on the execute path too.

All parameters are optional: `<il/>` should suffice to produce a debug build of all *.il files. The option set is roughly compatible with the CSharp class; even though the command line options are only vaguely equivalent. [The low level commands take things like /OUT=file, csc wants /out:file ... /verbose is used some places; /quiet here in ildasm... etc.] It would be nice if someone made all the command line tools consistent (and not as brittle as the java cmdline tools)

The task is a directory based task, so attributes like includes=”*.il” and excludes=”broken.il” can be used to control the files pulled in. Each file is built on its own, producing an appropriately named output file unless manually specified with outfile

Attribute	Description	Required
Attribute	Description	Example
defaultexcludes	indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted.	

Attribute	Description	Required
debug	include debug information	true (default)
excludes	comma separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	
extraOptions	Any extra options which aren't explicitly supported by the ilasm task, primarily because they arent really documented: use ilasm /? to see them	
failOnError	Should a failed compile halt the build?	"true"(default)
fullpaths	Should error text provide the full path to files	"true"(default)
includes	comma separated list of patterns of files that must be included. All files are included when omitted.	
includesfile	the name of a file. Each line of this file is taken to be an include pattern	
keyfile	the name of a file containing a private key, with which the assembly output is checksumed and then MD5 signed to have a strong name	
listing	Produce a listing (off by default). Listings go to the current output stream	"on", "off" (default)
outputFile	filename of output	"example.exe"
resourceFile	name of resource file to include	"resources.res"
srcDir	source directory (default = project directory)	
targetType	Type of target. library means DLL is output.	"exe"(default), "library"
verbose	output progress messages	"on", "off" (default)

Attribute	Description	Required

Example

```
<ilasm
        outputFile="app.exe"
        verbose="on"
        listing="on"
        owner="secret"
        />
```

`<WsdlToDotnet>`

Why add a wrapper to the MS WSDL tool? So that you can verify that your web services, be they written with Axis or anyone else's SOAP toolkit, work with .NET clients.

This task is dependency aware when using a file as a source and destination; so if you `<get>` the file (with usetimestamp="true") then you only rebuild stuff when the WSDL file is changed. Of course, if the server generates a new timestamp every time you ask for the WSDL, this is not enough...use the `<filesmatch>` `<condition>` to to byte for byte comparison against a cached WSDL file then make the target conditional on that test failing.

Attribute	Description	Required
Attribute	Description	Example
destFile	name of file to generate. Required	ApacheNet.cs
srcFile	name of WSDL file to use. Required if url is not set	service.wsdl
url	url to retrive WSDL from. required if srcFile is unset	http://localhost/service?wsdl
server	generate server stubs, not client proxy code. optional; default false	"false"(default)
namespace	namespace to place the source in. optional; default ""	Apache.Net
language	language; one of "CS", "JS", or "VB" optional;	"CS" (default)
failOnError	Should failure halt the build?	"true"(default)
extraOptions	Any extra options which aren't explicitly supported by the task, like all the proxy server config stuff	

Attribute	Description	Required

Change Log

```
Version 0.5
This revision goes along with NET 1.0 (SP1)

   1. CSC: added filealign
   2. CSC: added reference to office.dll
   3. CSC: dependency checking! only if destFile is set!
   4. WsdlToDotnet written

Version 0.4
This is the beta-2 revision of the tasks.

   1. ILASM: pulled the owner attribute, added keyfile for giving
      binaries a strong name (MD5 hash of the checksum)
   2. CSC: added win32res , noConfig, utf8output, fullpaths
   3. CSC:

Version 0.3

The changes here reflect Beta-1 of the dotnet SDK and experience of
use in more complex projects. This build does not work with the older
SDK, primarily because the automatic reference feature references
libraries only found in the new SDK version.

External changes

    * Recursive inclusion of .cs and .il files
    * Documentation enhanced, includes examples and details of all parameters
    * The csc task automatically includes the common dotnet assemblies,
      so there is no need to remember to refer to 'System.dll',
      'System.Web.Services', etc. This feature can be disabled by
      setting the 'includeDefaultReferences' flag to false.
    * References can also be referred to using the ReferenceFiles
      parameter, which is an ant path specification. The old 'references'
      string is still retained.
    * An 'extraoptions' attribute enables the build file to include any
      CSC options which are not explicitly supported in the CSC task.

Internal changes

    * Some minor refactoring (move common code a method)
    * Application of Jedits JavaStyle task resulted in a major reshaping of the codebase and
    * Removed throws clause from methods which can't throw exceptions
```

```
The test harness has been expanded to include unicode source file (the
build works but the rest of the system has 'issues' with high unicode
package and method names)

Version 0.2
First public edition, added to the ant cvs tree.
Tested on the PDC build of the dotnet SDK only, and still immature.
The command execution code was refactored out into a 'NetCommand'
class for re-use. The Ilasm task was added at this time.
Version 0.1
Initial proof of concept; very rudimentary support for CSC only.
```

5.3.2 ANTLR

Description

Invokes the ANTLR Translator generator on a grammar file.

To use the ANTLR task, set the target attribute to the name of the grammar file to process. Optionally, you can also set the outputdirectory to write the generated file to a specific directory. Otherwise ANTLR writes the generated files to the directory containing the grammar file.

This task only invokes ANTLR if the grammar file is newer than the generated files.

Antlr 2.7.1 Note: To successfully run ANTLR, your best option is probably to build the whole jar with the provided script mkalljar and drop the resulting jar (about 300KB) into ${ant.home}/lib. Dropping the default jar (70KB) is probably not enough for most needs and your only option will be to add ANTLR home directory to your classpath as described in ANTLR install.html document.

Antlr 2.7.2 Note: Instead of the above, you will need antlrall.jar that can be created by the antlr-all.jar target of the Makefile provided with the download.

Parameters

Attribute	Description	Required
target	The grammar file to process.	Yes
outputdirectory	The directory to write the generated files to. If not set, the files are written to the directory containing the grammar file.	No
glib	An optional super grammar file that the target grammar overrides. This feature is only needed for advanced vocabularies.	No

Attribute	Description	Required
debug	When set to ”yes”, this flag adds code to the generated parser that will launch the ParseView debugger upon invocation. The default is ”	no”.
Note	: ParseView is a separate component that needs to be installed or your grammar will have compilation errors.	No
html	Emit an html version of the grammar with hyperlinked actions.	No
diagnostic	Generates a text file with debugging infomation based on the target grammar.	No
trace	Forces all rules to call traceIn/traceOut if set to ”yes”. The default is ”no”.	No
traceParser	Only forces parser rules to call traceIn/traceOut if set to ”yes”. The default is ”no”.	No
traceLexer	Only forces lexer rules to call traceIn/traceOut if set to ”yes”. The default is ”no”.	No
traceTreeWalker	Only forces tree walker rules to call traceIn/traceOut if set to ”yes”. The default is ”no”.	No
dir	The directory to invoke the VM in.	No

Nested Elements

ANTLR supports a nested `<classpath>` element, that represents a PATH like structure. It is given as a convenience if you have to specify the original ANTLR directory. In most cases, dropping the appropriate ANTLR jar in the normal Ant lib repository will be enough.

jvmarg

Additional parameters may be passed to the new VM via nested `<jvmarg>` attributes, for example:

```
<antlr target="...">
  <jvmarg value="-Djava.compiler=NONE"/>
  ...
</antlr>
```

would run ANTLR in a VM without JIT.

`<jvmarg>` allows all attributes described in Command line arguments.

Example

```
<antlr
```

```
    target="etc/java.g"
    outputdirectory="build/src"
/>
```

This invokes ANTLR on grammar file etc/java.g, writing the generated files to build/src.

5.3.3 Cab

Description

The cab task creates Microsoft cab archive files. It is invoked similar to the jar or zip tasks. This task will work on Windows using the external cabarc tool (provided by Microsoft) which must be located in your executable path.

To use this task on other platforms you need to download and compile libcabinet from http://trill.cis.fordham.edu/ barbacha/cabinet_library/.

See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes basedir) as well as the nested `<include>`, `<exclude>` and `<patternset>` elements.

Parameters

Attribute	Description	Required
cabfile	the name of the cab file to create.	Yes
basedir	the directory to start archiving files from.	Yes
verbose	set to "yes" if you want to see the output from the cabarc tool. defaults to "no".	No
compress	set to "no" to store files without compressing. defaults to "yes".	No
options	use to set additional command-line options for the cabarc tool. should not normally be necessary.	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No

Attribute	Description	Required
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No

Parameters specified as nested elements

fileset

The cab task supports any number of nested `<fileset>` elements to specify the files to be included in the archive.

Examples

```
<cab cabfile="${dist}/manual.cab"
     basedir="htdocs/manual"
  />
```

cabs all files in the htdocs/manual directory into a file called manual.cab in the ${dist} directory.

```
<cab cabfile="${dist}/manual.cab"
     basedir="htdocs/manual"
     excludes="mydocs/**, **/todo.html"
  />
```

cabs all files in the htdocs/manual directory into a file called manual.cab in the ${dist} directory. Files in the directory mydocs, or files with the name todo.html are excluded.

```
<cab cabfile="${dist}/manual.cab"
     basedir="htdocs/manual"
     includes="api/**/*.html"
     excludes="**/todo.html"
     verbose="yes"
  />
```

Cab all files in the htdocs/manual directory into a file called manual.cab in the ${dist} directory. Only html files under the directory api are archived, and files with the name todo.html are excluded. Output from the cabarc tool is displayed in the build output.

5.3.4 Clearcase Tasks

CCCheckin

Description

Task to perform a Checkin command to ClearCase.

Parameters

Attribute	Description	Required
viewpath	Path to the ClearCase view file or directory that the command will operate on	No
comment	Specify a comment. Only one of comment or commentfile may be used.	No
commentfile	Specify a file containing a comment. Only one of comment or commentfile may be used.	No
nowarn	Suppress warning messages	No
preservetime	Preserve the modification time	No
keepcopy	Keeps a copy of the file with a .keep extension	No
identical	Allows the file to be checked in even if it is identical to the original	No

Examples

```
<cccheckin viewpath="c:/views/viewdir/afile"
        commentfile="acomment.txt"
        nowarn="true"
        identical="true"/>
```

Does a ClearCase checkin on the file c:/views/viewdir/afile. Comment text from the file acomment.txt is added to ClearCase as a comment. All warning messages are suppressed. The file is checked in even if it is identical to the original.

CCCheckout

Description

Task to perform a Checkout command to ClearCase.

Parameters

Attribute	Values	Required
viewpath	Path to the ClearCase view file or directory that the command will operate on	No

Attribute	Values	Required
reserved	Specifies whether to check out the file as reserved or not	Yes
out	Creates a writable file under a different filename	No
nodata	Checks out the file but does not create an editable file containing its data	No
branch	Specify a branch to check out the file to	No
version	Allows checkout of a version other than main latest	No
nowarn	Suppress warning messages	No
comment	Specify a comment. Only one of comment or commentfile may be used.	No
commentfile	Specify a file containing a comment. Only one of comment or commentfile may be used.	No

Examples

```
<cccheckout viewpath="c:/views/viewdir/afile"
        reserved="true"
        branch="abranch"
        nowarn="true"
        comment="Some comment text"/>
```

Does a ClearCase checkout on the file c:/views/viewdir/afile. It is checked out as reserved on branch called abranch. All warning messages are suppressed. A Some comment text is added to ClearCase as a comment.

CCUnCheckout

Description

Task to perform a UnCheckout command to ClearCase.

Parameters

Attribute	Values	Required
viewpath	Path to the ClearCase view file or directory that the command will operate on	No
keepcopy	Specifies whether to keep a copy of the file with a .keep extension or not	No

Examples

```
<ccuncheckout viewpath="c:/views/viewdir/afile"
        keepcopy="true"/>
```

Does a ClearCase uncheckout on the file c:/views/viewdir/afile. A copy of the file called c:/views/viewdir/afile.keep is kept.

CCUpdate

Description

Task to perform an Update command to ClearCase.

Parameters

Attribute	Values	Required
viewpath	Path to the ClearCase view file or directory that the command will operate on	No
graphical	Displays a graphical dialog during the update	No
log	Specifies a log file for ClearCase to write to	No
overwrite	Specifies whether to overwrite hijacked files or not	No
rename	Specifies that hijacked files should be renamed with a .keep extension	No
currenttime	Specifies that modification time should be written as the current time. Either currenttime or preservetime can be specified.	No
preservetime	Specifies that modification time should preserved from the VOB time. Either currenttime or preservetime can be specified.	No

Examples

```
<ccupdate viewpath="c:/views/viewdir"
        graphical="false"
        log="log.log"
        overwrite="true"
        currenttime="true"
        rename="false"/>
```

Does a ClearCase update on the directory c:/views/viewdir. A graphical dialog will be displayed. The output will be logged to log.log and it will overwrite any hijacked files. The modified time will be set to the current time.

5.3.5 Continuus/Synergy Tasks

These ant tasks are wrappers around Continuus Source Manager. They have been tested with version 5.1 on Windows 2000, but should work on other platforms with ccm installed.

CCMCheckin

Description

Task to checkin a file

Parameters

Attribute	Values	Required
file	Path to the file that the command will operate on	Yes
comment	Specify a comment. Default is "Checkin" plus the date	No
task	Specify the task number used to check in the file (may use 'default')	No
ccmdir	path to the ccm executable file, required if it is not on the PATH	No

Examples

```
<ccmcheckin file="c:/wa/com/foo/MyFile.java"
        comment="mycomment"/>
```

Checks in the file c:/wa/com/foo/MyFile.java. Comment attribute mycomment is added as a task comment. The task used is the one set as the default.

CCMCheckout

Description

Task to perform a Checkout command to Continuus

Parameters

Attribute	Values	Required
file	Path to the file that the command will operate on	Yes
comment	Specify a comment.	No
task	Specify the task number used to checkin the file (may use 'default')	No
ccmdir	path to the ccm executable file, required if it is not on the PATH	No

Attribute	Values	Required

Examples

```
<ccmcheckout file="c:/wa/com/foo/MyFile.java"
        comment="mycomment"/>
```

Check out the file c:/wa/com/foo/MyFile.java. Comment attribute mycomment is added as a task comment The used task is the one set as the default.

CCMCheckinTask

Description

Task to perform a check in default task command to Continuus

Parameters

Attribute	Values	Required
comment	Specify a comment.	No
task	Specify the task number used to check in the file (may use 'default')	No
ccmdir	path to the ccm executable file, required if it is not on the PATH	No

Examples

```
<ccmcheckintask comment="blahblah/>
```

Does a Checkin default task on all the checked out files in the current task.

CCMReconfigure

Description

Task to perform an reconfigure command to Continuus.

Parameters

Attribute	Values	Required
recurse	recurse on subproject (default false)	No
verbose	do a verbose reconfigure operation (default false)	No
ccmproject	Specifies the ccm project on which the operation is applied.	Yes
ccmdir	path to the ccm executable file, required if it is not on the PATH	No

Examples

```
<ccmreconfigure ccmproject="ANTCCM_TEST#BMO_1"
         verbose="true" />
```

Does a Continuus reconfigure on the project `ANTCCM_TEST#BMO_1`.

CCMCreateTask

Description

Create a Continuus task.

Parameters

Attribute	Values	Required
comment	Specify a comment.	No
platform	Specify the target platform	No
ccmdir	path to the ccm executable file, required if it is not on the PATH	No
resolver	Specify the resolver	No
release	Specify the CCM release	No
subsystem	Specify the subsystem	No
task	Specify the task number used to checkin the file (may use 'default')	No

Examples

```
<ccmcreatetask resolver="${user.name}"
           release="ANTCCM_TEST" comment="blahblah" />
```

Creates a task for the release ANTCCM_TEST with the current user as the resolver for this task.

5.3.6 Depend

A task to manage Java class file dependencies.

Description

The depend task works by determining which classes are out of date with respect to their source and then removing the class files of any other classes which depend on the out-of-date classes.

To determine the class dependencies, the depend task analyses the class files of all class files passed to it. Depend does not parse your source code in any way but relies upon the class references encoded into the class files by the compiler. This is generally faster than parsing the Java source.

To learn more about how this information is obtained from the class files, please refer to the Java Virtual Machine Specification

Since a class' dependencies only change when the class itself changes, the depend task is able to cache dependency information. Only those class files which have changed will have their dependency information re-analysed. Note that if you change a class' dependencies by changing the source, it will be recompiled anyway. You can examine the dependency files created to understand the dependencies of your classes. Please do not rely, however, on the format of the information, as it may change in a later release.

Once depend discovers all of the class dependencies, it "inverts" this relation to determine, for each class, which other classes are dependent upon it. This "affects" list is used to discover which classes are invalidated by the out of date class. The class files of the invalidated classes are removed, triggering the compilation of the affected classes.

The depend task supports an attribute, "closure" which controls whether depend will only consider direct class-class relationships or whether it will also consider transitive, indirect relationships. For example, say there are three classes, A, which depends on B, which in-turn depend on C. Now say that class C is out of date. Without closure, only class B would be removed by depend. With closure set, class A would also be removed. Normally direct relationships are sufficient - it is unusual for a class to depend on another without having a direct relationship. With closure set, you will notice that depend typically removes far more class files.

The classpath attribute for `<depend>` is optional. If it is present, depend will check class dependencies against classes and jars on this classpath. Any classes which depend on an element from this classpath and which are older than that element will be deleted. A typical example where you would use this facility would be where you are building a utility jar and want to make sure classes which are out of date with respect to this jar are rebuilt. You should not include jars in this classpath which you do not expect to change, such as the JDK runtime jar or third party jars, since doing so will just slow down the dependency check. This means that if you do use a classpath for the depend task it may be different from the classpath necessary to actually compile your code.

Performance

The performance of the depend task is dependent on a number of factors such as class relationship complexity and how many class files are out of date. The decision about whether it is cheaper to just recompile all classes or to use the depend task will depend on the size of your project and how interrelated your classes are.

Limitations

There are some source dependencies which depend will not detect.

- If the Java compiler optimizes away a class relationship, there can be a source dependency without a class dependency.

- Non public classes cause two problems. Firstly depend cannot relate the class file to a source file. In the future this may be addressed using the source file attribute in the classfile. Secondly, neither depend nor the compiler tasks can detect when a non public class is missing. Inner classes are handled by the depend task.

The most obvious example of these limitations is that the task can't tell which classes to recompile when a constant primitive data type exported by other classes is changed. For example, a change in the definition of something like

```
public final class Constants {
  public final static boolean DEBUG=false;
}
```

will not be picked up by other classes.

Parameters

Attribute	Values	Required
srcDir	This is the directory where the source exists. depend will examine this to determine which classes are out of date. If you use multiple source directories you can pass this attribute a path of source directories.	Yes
destDir	This is the root directory of the class files which will be analysed. If this is not present, the srcdir is used.	No
cache	This is a directory in which depend can store and retrieve dependency information. If this is not present, depend will not use a cache	No
closure	This attribute controls whether depend only removes classes which directly depend on out of date classes. If this is set to true, depend will traverse the class dependency graph deleting all affected classes. Defaults to false	No
dump	If true the dependency information will be written to the debug level log	No
classpath	The classpath containg jars and classes for which `<depend>` should also check dependencies	No

Attribute	Values	Required

Parameters specified as nested elements

The depend task's classpath attribute is a PATH-like structure and can also be set via a nested `<classpath>` element.

Additionally, this task forms an implicit FileSet and supports all attributes of `<fileset>` (dir becomes srcdir), as well as the nested `<include>`, `<exclude>`, and `<patternset>` elements.

Examples

```
<depend srcdir="${java.dir}"
        destdir="${build.classes}"
        cache="depcache"
        closure="yes"/>
```

removes any classes in the ${build.classes} directory that depend on out-of-date classes. Classes are considered out-of-date with respect to the source in the ${java.dir} directory, using the same mechanism as the `<javac>` task. In this example, the `<depend>` task caches its dependency information in the depcache directory.

```
<depend srcdir="${java.dir}" destdir="${build.classes}"
        cache="depcache" closure="yes">
  <include name="**/*.java"/>
  <excludesfile name="${java.dir}/build_excludes"/>
</depend>
```

does the same as the previous example, but explicitly includes all .java files, except those that match the list given in ${java.dir}/build_excludes.

5.3.7 EJB Tasks

by

```
    * Paul Austin (p_d_austin@yahoo.com)
    * Holger Engels (hengels@innovidata.com)
    * Tim Fennell (tfenne@rcn.com)
    * Martin Gee (martin.gee@icsynergy.com)
    * Conor MacNeill
    * Cyrille Morvan (cmorvan@ingenosya.com)
    * Greg Nelson (gn@sun.com)

Version @VERSION@
$Id: ejb.html,v 1.23.2.17 2003/03/19 13:43:32 conor Exp $
```

Ant provides a number of optional tasks for developing Enterprise Java Beans (EJBs). In general these tasks are specific to the particular vendor's EJB Server.

At present the tasks support:

- Borland Application Server 4.5
- iPlanet Application Server 6.0
- JBoss 2.1 and above EJB servers
- Weblogic 4.5.1 through to 7.0 EJB servers
- JOnAS 2.4.x and 2.5 Open Source EJB server
- IBM WebSphere 4.0

Over time we expect further optional tasks to support additional EJB Servers.

EJB Tasks

<table>
<tr><td colspan="2">Task</td><td>Application Servers</td></tr>
<tr><td colspan="2">blgenclient</td><td>Borland Application Server 4.5</td></tr>
<tr><td colspan="2">ddcreator</td><td>Weblogic 4.5.1</td></tr>
<tr><td colspan="2">ejbc</td><td>Weblogic 4.5.1</td></tr>
<tr><td colspan="2">iplanet-ejbc</td><td>iPlanet Application Server 6.0</td></tr>
<tr><td rowspan="7">ejbjar</td><td colspan="2">Nested Elements</td></tr>
<tr><td>borland</td><td>Borland Application Server 4.5</td></tr>
<tr><td>iPlanet</td><td>iPlanet Application Server 6.0</td></tr>
<tr><td>jboss</td><td>JBoss</td></tr>
<tr><td>jonas</td><td>JOnAS 2.4.x and 2.5</td></tr>
<tr><td>weblogic</td><td>Weblogic 5.1 to 7.0</td></tr>
<tr><td>websphere</td><td>IBM WebSphere 4.0</td></tr>
<tr><td colspan="2">wlrun</td><td>Weblogic 4.5.1 to 7.0</td></tr>
<tr><td colspan="2">wlstop</td><td>Weblogic 4.5.1 to 7.0</td></tr>
</table>

ddcreator

Description:

ddcreator will compile a set of Weblogic text-based deployment descriptors into a serialized EJB deployment descriptor. The selection of which of the text-based descriptors are to be compiled is based on the standard Ant include and exclude selection mechanisms.

Parameters

Attribute	Values	Required
descriptors	This is the base directory from which descriptors are selected.	Yes
dest	The directory where the serialized deployment descriptors will be written	Yes
classpath	This is the classpath to use to run the underlying weblogic ddcreator tool. This must include the weblogic.ejb.utils.DDCreator class	No

Attribute	Description	Required

Examples

```
<ddcreator descriptors="${dd.dir}"
           dest="${gen.classes}"
           classpath="${descriptorbuild.classpath}">
  <include name="*.txt"/>
</ddcreator>
```

ejbc

Description:

The ejbc task will run Weblogic's ejbc tool. This tool will take a serialized deployment descriptor, examine the various EJB interfaces and bean classes and then generate the required support classes necessary to deploy the bean in a Weblogic EJB container. This will include the RMI stubs and skeletons as well as the classes which implement the bean's home and remote interfaces.

The ant task which runs this tool is able to compile several beans in a single operation. The beans to be compiled are selected by including their serialized deployment descriptors. The standard ant include and exclude constructs can be used to select the deployment descriptors to be included.

Each descriptor is examined to determine whether the generated classes are out of date and need to be regenerated. The deployment descriptor is de-serialized to discover the home, remote and implementation classes. The corresponding source files are determined and checked to see their modification times. These times and the modification time of the serialized descriptor itself are compared with the modification time of the generated classes. If the generated classes are not present or are out of date, the ejbc tool is run to generate new versions.

Parameters

Attribute	Description	Required
descriptors	This is the base directory from which the serialized deployment descriptors are selected.	Yes
dest	The base directory where the generated classes, RIM stubs and RMI skeletons are written	Yes
manifest	The name of a manifest file to be written. This manifest will contain an entry for each EJB processed	Yes

Attribute	Description	Required
src	The base directory of the source tree containing the source files of the home interface, remote interface and bean implementation classes.	Yes
classpath	This classpath must include both the weblogic.ejbc class and the class files of the bean, home interface, remote interface, etc of the bean being processed.	No
keepgenerated	Controls whether ejbc will keep the intermediate Java files used to build the class files. This can be useful when debugging.	No, defaults to false.

Examples

```
<ejbc descriptors="${gen.classes}"
           src="${src.dir}"
           dest="${gen.classes}"
           manifest="${build.manifest}"
           classpath="${descriptorbuild.classpath}">
  <include name="*.ser"/>
</ejbc>
```

iplanet-ejbc

Description:

Task to compile EJB stubs and skeletons for the iPlanet Application Server 6.0. Given a standard EJB 1.1 XML descriptor as well as an iAS-specific EJB descriptor, this task will generate the stubs and skeletons required to deploy the EJB to iAS. Since the XML descriptors can include multiple EJBs, this is a convenient way of specifying many EJBs in a single Ant task.

For each EJB specified, the task will locate the three classes that comprise the EJB in the destination directory. If these class files cannot be located in the destination directory, the task will fail. The task will also attempt to locate the EJB stubs and skeletons in this directory. If found, the timestamps on the stubs and skeletons will be checked to ensure they are up to date. Only if these files cannot be found or if they are out of date will the iAS ejbc utility be called to generate new stubs and skeletons.

Parameters

Attribute	Description	Required
ejbdescriptor	Standard EJB 1.1 XML descriptor (typically titled ”ejb-jar.xml”).	Yes

Attribute	Description	Required
iasdescriptor	iAS-specific EJB XML descriptor (typically titled "ias-ejb-jar.xml").	Yes
dest	The is the base directory where the RMI stubs and skeletons are written. In addition, the class files for each bean (home interface, remote interface, and EJB implementation) must be found in this directory.	Yes
classpath	The classpath used when generating EJB stubs and skeletons. If omitted, the classpath specified when Ant was started will be used. Nested "classpath" elements may also be used.	No
keepgenerated	Indicates whether or not the Java source files which are generated by ejbc will be saved or automatically deleted. If "yes", the source files will be retained. If omitted, it defaults to "no".	No
debug	Indicates whether or not the ejbc utility should log additional debugging statements to the standard output. If "yes", the additional debugging statements will be generated. If omitted, it defaults to "no".	No
iashome	May be used to specify the "home" directory for this iAS installation. This is used to find the ejbc utility if it isn't included in the user's system path. If specified, it should refer to the "[install-location]/iplanet/ias6/ias" directory. If omitted, the ejbc utility must be on the user's system path.	No

Examples

```
<iplanet-ejbc ejbdescriptor="ejb-jar.xml"
              iasdescriptor="ias-ejb-jar.xml"
              dest="${build.classesdir}"
              classpath="${ias.ejbc.cpath}"/>

<iplanet-ejbc ejbdescriptor="ejb-jar.xml"
              iasdescriptor="ias-ejb-jar.xml"
              dest="${build.classesdir}"
              keepgenerated="yes"
              debug="yes"
              iashome="${ias.home}">
              <classpath>
                  <pathelement path="."/>
                  <pathelement path="${build.classpath}"/>
              </classpath>
</iplanet-ejbc>
```

wlrun

Description:

The wlrun task is used to start a weblogic server. The task runs a weblogic instance in a separate Java Virtual Machine. A number of parameters are used to control the operation of the weblogic instance. Note that the task, and hence ant, will not complete until the weblogic instance is stopped.

Parameters

Attribute	Description	Required	
BEA Home	The location of the BEA Home where the server's config is defined. If this attribute is present, wlrun assumes that the server will be running under Weblogic 6.0	N/A	Yes
home	The location of the weblogic home that is to be used. This is the location where weblogic is installed.	Yes	Yes. Note this is the absolute location, not relative to BEA home.
Domain	The domain to which the server belongs.	N/A	Yes
classpath	The classpath to be used with the Java Virtual Machine that runs the Weblogic Server. Prior to Weblogic 6.0, this is typically set to the Weblogic boot classpath. Under Weblogic 6.0 this should include all the weblogic jars	Yes	Yes
wlclasspath	The weblogic classpath used by the Weblogic Server.	No	N/A
properties	The name of the server's properties file within the weblogic home directory used to control the weblogic instance.	Yes	N/A
name	The name of the weblogic server within the weblogic home which is to be run. This defaults to "myserver"	No	No

Attribute	Description	Required for 4.5.1 and 5.1	Required for 6.0
policy	The name of the security policy file within the weblogic home directory that is to be used. If not specified, the default policy file weblogic.policy is used.	No	No
username	The management username used to manage the server	N/A	No
password	The server's management password	N/A	Yes
pkPassword	The private key password so the server can decrypt the SSL private key file	N/A	No
jvmargs	Additional argument string passed to the Java Virtual Machine used to run the Weblogic instance.	No	No
weblogicMainClass	name of the main class for weblogic	No	No

Nested Elements

The wlrun task supports nested `<classpath>` and `<wlclasspath>` elements to set the respective classpaths.

Examples

This example shows the use of wlrun to run a server under Weblogic 5.1

```
<wlrun taskname="myserver"
       classpath="${weblogic.boot.classpath}"
       wlclasspath="${weblogic.classes}:${code.jars}"
       name="myserver"
       home="${weblogic.home}"
       properties="myserver/myserver.properties"/>
```

This example shows wlrun being used to run the petstore server under Weblogic 6.0

```
<wlrun taskname="petstore"
       classpath="${weblogic.classes}"
       name="petstoreServer"
       domain="petstore"
       home="${weblogic.home}"
       password="petstorePassword"
       beahome="${bea.home}"/>
```

wlstop

Description:

The wlstop task is used to stop a weblogic instance which is currently running. To shut down an instance you must supply both a username and a password. These will be stored in the clear in the build script used to stop the instance. For security reasons, this task is therefore only appropriate in a development environment.

This task works for most version of Weblogic, including 6.0. You need to specify the BEA Home to have this task work correctly under 6.0

Parameters

Attribute	Description	Required
BEAHome	This attribute selects Weblogic 6.0 shutdown.	No
classpath	The classpath to be used with the Java Virtual Machine that runs the Weblogic Shutdown command.	Yes
user	The username of the account which will be used to shutdown the server	Yes
password	The password for the account specified in the user parameter.	Yes
url	The URL which describes the port to which the server is listening for T3 connections. For example, t3://localhost:7001	Yes
delay	The delay in seconds after which the server will stop. This defaults to an immediate shutdown.	No

Nested Element

The classpath of the wlstop task can be set by a `<classpath>` nested element.

Examples

This example show the shutdown for a Weblogic 6.0 server

```
<wlstop classpath="${weblogic.classes}"
        user="system"
        url="t3://localhost:7001"
        password="foobar"
        beahome="${bea.home}"/>
```

ejbjar

Description:

This task is designed to support building of EJB jar files (EJB 1.1 & 2.0). Support is currently provided for 'vanilla' EJB jar files - i.e. those containing only the user generated class files and the standard deployment descriptor. Nested elements provide support for vendor specific deployment tools. These currently include:

- Borland Application Server 4.5
- iPlanet Application Server 6.0

- JBoss 2.1 and above
- Weblogic 5.1/6.0 session/entity beans using the weblogic.ejbc tool
- IBM WebSphere 4.0
- TOPLink for WebLogic 2.5.1-enabled entity beans
- JOnAS 2.4.x and 2.5 Open Source EJB server

The task works as a directory scanning task, and performs an action for each deployment descriptor found. As such the includes and excludes should be set to ensure that all desired EJB descriptors are found, but no application server descriptors are found. For each descriptor found, ejbjar will parse the deployment descriptor to determine the necessary class files which implement the bean. These files are assembled along with the deployment descriptors into a well formed EJB jar file. Any support files which need to be included in the generated jar can be added with the `<support>` nested element. For each class included in the jar, ejbjar will scan for any super classes or super interfaces. These will be added to the generated jar.

If no nested vendor-specific deployment elements are present, the task will simply generate a generic EJB jar. Such jars are typically used as the input to vendor-specific deployment tools. For each nested deployment element, a vendor specific deployment tool is run to generate a jar file ready for deployment in that vendor's EJB container.

The jar files are only built if they are out of date. Each deployment tool element will examine its target jar file and determine if it is out of date with respect to the class files and deployment descriptors that make up the bean. If any of these files are newer than the jar file the jar will be rebuilt otherwise a message is logged that the jar file is up to date.

The task uses the jakarta-BCEL framework to extract all dependent classes. This means that, in addition to the classes that are mentioned in the deployment descriptor, any classes that these depend on are also automatically included in the jar file.

Naming Convention

Ejbjar handles the processing of multiple beans, and it uses a set of naming conventions to determine the name of the generated EJB jars. The naming convention that is used is controlled by the "naming" attribute. It supports the following values

- descriptor

  This is the default naming scheme. The name of the generated bean is derived from the name of the deployment descriptor. For an Account bean, for example, the deployment descriptor would be named Account-ejb-jar.xml. Vendor specific descriptors are located using the same naming convention. The weblogic bean, for example, would be named Account-weblogic-ejb-jar.xml. Under this arrangement, the deployment descriptors can be separated from the code implementing the beans, which can be useful when the same bean code is deployed in separate beans.

This scheme is useful when you are using one bean per EJB jar and where you may be deploying the same bean classes in different beans, with different deployment characteristics.

- ejb-name

  This naming scheme uses the `<ejb-name>` element from the deployment descriptor to determine the bean name. In this situation, the descriptors normally use the generic descriptor names, such as ejb-jar.xml along with any associated vendor specific descriptor names. For example, If the value of the `<ejb-name>` were to be given in the deployment descriptor as follows:

  ```
  <ejb-jar>
      <enterprise-beans>
          <entity>
              <ejb-name>Sample</ejb-name>
              <home>org.apache.ant.ejbsample.SampleHome</home>
  ```

  then the name of the generated bean would be Sample.jar

  This scheme is useful where you want to use the standard deployment descriptor names, which may be more compatible with other EJB tools. This scheme must have one bean per jar.

- directory

  In this mode, the name of the generated bean jar is derived from the directory containing the deployment descriptors. Again the deployment descriptors typically use the standard filenames. For example, if the path to the deployment descriptor is /home/user/dev/appserver/dd/sample, then the generated bean will be named sample.jar

  This scheme is also useful when you want to use standard style descriptor names. It is often most useful when the descriptors are located in the same directory as the bean source code, although that is not mandatory. This scheme can handle multiple beans per jar.

- basejarname

  The final scheme supported by the `<ejbjar>` task is used when you want to specify the generated bean jar name directly. In this case the name of the generated jar is specified by the "basejarname" attribute. Since all generated beans will have the same name, this task should be only used when each descriptor is in its own directory.

This scheme is most appropriate when you are using multiple beans per jar and only process a single deployment descriptor. You typically want to specify the name of the jar and not derive it from the beans in the jar.

Dependencies

In addition to the bean classes, ejbjar is able to ad additional classes to the generated ejbjar. These classes are typically the support classes which are used by the bean's classes or as parameters to the bean's methods.

In versions of Ant prior to 1.5, ejbjar used reflection and attempted to add the super classes and super interfaces of the bean classes. For this technique to work the bean classes had to be loaded into Ant's JVM. This was not always possible due to class dependencies.

The ejbjar task in Ant releases 1.5 and later uses the jakarta-BCEL library to analyze the bean's class files directly, rather than loading them into the JVM. This also allows ejbjar to add all of the required support classes for a bean and not just super classes.

In Ant 1.5, a new attribute, dependency has been introduced to allow the buildfile to control what additional classes are added to the generated jar. It takes three possible values

- none - only the bean classes and interfaces described in the bean's descriptor are added to the jar.

- super - this is the default value and replicates the original ejbjar behaviour where super classes and super interfaces are added to the jar

- full - In this mode all classes used by the bean's classes and interfaces are added to the jar

The super and full values require the jakarta-BCEL library to be available. If it is not, ejbjar will drop back to the behaviour corresponding to the value none.

Parameters

Attribute	Description	Required
descriptordir	The base directory under which to scan for EJB deployment descriptors. If this attribute is not specified, then the deployment descriptors must be located in the directory specified by the ’srcdir’ attribute.	No
srcdir	The base directory containing the .class files that make up the bean. Included are the home- remote- pk- and implementation- classes and all classes, that these depend on. Note that this can be the same as the descriptordir if all files are in the same directory tree.	Yes
destdir	The base directory into which generated jar files are deposited. Jar files are deposited in directories corresponding to their location within the descriptordir namespace. Note that this attribute is only used if the task is generating generic jars (i.e. no vendor-specific deployment elements have been specified).	Yes
naming	Controls the naming convention used to name generated EJB jars. Please refer to the description above.	No
basejarname	The base name that is used for the generated jar files. If this attribute is specified, the generic jar file name will use this value as the prefix (followed by the value specified in the ’genericjarsuffix’ attribute) and the resultant ejb jar file (followed by any suffix specified in the nested element).	No
basenameterminator	String value used to substring out a string from the name of each deployment descriptor found, which is then used to locate related deployment descriptors (e.g. the WebLogic descriptors). For example, a basename of ’.’ and a deployment descriptor called ’FooBean.ejb-jar.xml’ would result in a basename of ’FooBean’ which would then be used to find FooBean.weblogic-ejb-jar.xml and FooBean.weblogic-cmp-rdbms-jar.xml, as well as to create the filenames of the jar files as FooBean-generic.jar and FooBean-wl.jar. This attribute is not used if the ’basejarname’ attribute is specified. No, defaults to ’-	’.
genericjarsuffix	String value appended to the basename of the deployment descriptor to create the filename of the generic EJB jar file. No, defaults to ’-generic.	jar’.
classpath	This classpath is used when resolving classes which are to be added to the jar. Typically nested deployment tool elements will also support a classpath	No.

Nested Elements

In addition to the vendor specific nested elements, the ejbjar task provides three nested elements.

Classpath

The `<classpath>` nested element allows the classpath to be set. It is useful when setting the classpath from a reference path. In all other respects the behaviour is the same as the classpath attribute.

dtd

The `<dtd>` element is used to specify the local location of DTDs to be used when parsing the EJB deployment descriptor. Using a local DTD is much faster than loading the DTD across the net. If you are running ejbjar behind a firewall you may not even be able to access the remote DTD. The supported vendor-specific nested elements know the location of the required DTDs within the vendor class hierarchy and, in general, this means `<dtd>` elements are not required. It does mean, however, that the vendor's class hierarchy must be available in the classpath when Ant is started. If your want to run Ant without requiring the vendor classes in the classpath, you would need to use a `<dtd>` element.

Attribute	Description	Required
publicId	The public Id of the DTD for which the location is being provided	Yes
location	The location of the local copy of the DTD. This can either be a file or a resource loadable from the classpath.	Yes

support

The `<support>` nested element is used to supply additional classes (files) to be included in the generated jars. The `<support>` element is a FileSet, so it can either reference a fileset declared elsewhere or it can be defined in-place with the appropriate `<include>` and `<exclude>` nested elements. The files in the support fileset are added into the generated EJB jar in the same relative location as their location within the support fileset. Note that when ejbjar generates more than one jar file, the support files are added to each one.

Vendor-specific deployment elements

Each vendor-specific nested element controls the generation of a deployable jar specific to that vendor's EJB container. The parameters for each supported deployment element are detailed here.

Jboss element

The jboss element searches for the JBoss specific deployment descriptors and adds them to the final ejb jar file. JBoss has two deployment descriptors jboss.xml and jaws.xml (for container manager persistence only). The JBoss server uses hot deployment and does not require compilation of additional stubs and skeletons.

Attribute	Description	Required
destdir	The base directory into which the generated weblogic ready jar files are deposited. Jar files are deposited in directories corresponding to their location within the descriptordir namespace.	Yes
genericjarsuffix	A generic jar is generated as an intermediate step in build the weblogic deployment jar. The suffix used to generate the generic jar file is not particularly important unless it is desired to keep the generic jar file. It should not, however, be the same as the suffix setting.	No, defaults to '-generic.jar'.
suffix	String value appended to the basename of the deployment descriptor to create the filename of the JBoss EJB jar file.	No, defaults to '.jar'.
keepgeneric	This controls whether the generic file used as input to ejbc is retained.	No, defaults to false

Weblogic element

The weblogic element is used to control the weblogic.ejbc compiler for generating weblogic EJB jars. Prior to Ant 1.3, the method of locating CMP descriptors was to use the ejbjar naming convention. So if your ejb-jar was called, Customer-ejb-jar.xml, your weblogic descriptor was called Customer-weblogic-ejb-jar.xml and your CMP descriptor had to be Customer-weblogic-cmp- rdbms-jar.xml. In addition, the `<type-storage>` element in the weblogic descriptor had to be set to the standard name META-INF/weblogic-cmp-rdbms-jar.xml, as that is where the CMP descriptor was mapped to in the generated jar.

There are a few problems with this scheme. It does not allow for more than one CMP descriptor to be defined in a jar and it is not compatible with the deployment descriptors generated by some tools.

In Ant 1.3, ejbjar parses the weblogic deployment descriptor to discover the CMP descriptors, which are then included automatically. This behaviour is controlled by the newCMP attribute. Note that if you move to the new method of determining CMP descriptors, you will need to update your weblogic deployment descriptor's `<type-storage>` element. In the above example, you would define this as META-INF/Customer-weblogic-cmp-rdbms-jar.xml.

Attribute	Description	Required
destdir	The base directory into which the generated weblogic ready jar files are deposited. Jar files are deposited in directories corresponding to their location within the descriptordir namespace.	Yes
genericjarsuffix	A generic jar is generated as an intermediate step in build the weblogic deployment jar. The suffix used to generate the generic jar file is not particularly important unless it is desired to keep the generic jar file. It should not, however, be the same as the suffix setting.	No, defaults to ’-generic.jar’.
suffix	String value appended to the basename of the deployment descriptor to create the filename of the WebLogic EJB jar file.	No, defaults to ’.jar’.
classpath	The classpath to be used when running the weblogic ejbc tool. Note that this tool typically requires the classes that make up the bean to be available on the classpath. Currently, however, this will cause the ejbc tool to be run in a separate VM	No
wlclasspath	Weblogic 6.0 will give a warning if the home and remote interfaces of a bean are on the system classpath used to run weblogic.ejbc. In that case, the standard weblogic classes should be set with this attribute (or equivalent nested element) and the home and remote interfaces located with the standard classpath attribute	No
keepgeneric	This controls whether the generic file used as input to ejbc is retained.	No, defaults to false
compiler	This allows for the selection of a different compiler to be used for the compilation of the generated Java files. This could be set, for example, to Jikes to compile with the Jikes compiler. If this is not set and the build.compiler property is set to jikes, the Jikes compiler will be used. If this is not desired, the value ”default” may be given to use the default compiler	No
rebuild	This flag controls whether weblogic.ejbc is always invoked to build the jar file. In certain circumstances, such as when only a bean class has been changed, the jar can be generated by merely replacing the changed classes and not rerunning ejbc. Setting this to false will reduce the time to run ejbjar.	No, defaults to true.
keepgenerated	Controls whether weblogic will keep the generated Java files used to build the class files added to the jar. This can be useful when debugging	No, defaults to false.
args	Any additional arguments to be passed to the weblogic.ejbc tool.	No.
weblogicdtd	Deprecated. Defines the location of the ejb-jar DTD in the weblogic class hierarchy. This should not be necessary if you have weblogic in your classpath. If you do not, you should use a nested `<dtd>` element, described above. If you do choose to use an attribute, you should use a nested `<dtd>` element.	No.
wldtd	Deprecated. Defines the location of the weblogic-ejb-jar DTD which covers the Weblogic specific deployment descriptors. This should not be necessary if you have weblogic in your classpath. If you do not, you should use a nested `<dtd>` element, described above.	No.

Attribute	Description	Required
ejbdtd	Deprecated. Defines the location of the ejb-jar DTD in the weblogic class hierarchy. This should not be necessary if you have weblogic in your classpath. If you do not, you should use a nested `<dtd>` element, described above.	No.
newCMP	If this is set to true, the new method for locating CMP descriptors will be used.	No. Defaults to false
oldCMP	Deprecated This is an antonym for newCMP which should be used instead.	No.
noEJBC	If this attribute is set to true, Weblogic's ejbc will not be run on the EJB jar. Use this if you prefer to run ejbc at deployment time.	No.
ejbcclass	Specifies the classname of the ejbc compiler. Normally ejbjar determines the appropriate class based on the DTD used for the EJB. The EJB 2.0 compiler featured in weblogic 6 has, however, been deprecated in version 7. When using with version 7 this attribute should be set to "weblogic.ejbc" to avoid the deprecation warning.	No.
jvmargs	Any additional arguments to be passed to the Virtual Machine running weblogic.ejbc tool. For example to set the memory size, this could be jvmargs="-Xmx128m"	No.
jvmdebuglevel	Sets the weblogic.StdoutSeverityLevel to use when running the Virtual Machine that executes ejbc. Set to 16 to avoid the warnings about EJB Home and Remotes being in the classpath	No.
outputdir	If set ejbc will be given this directory as the output destination rather than a jar file. This allows for the generation of "exploded" jars.	No.

The weblogic nested element supports three nested elements. The first two, `<classpath>` and `<wlclasspath>`, are used to set the respective classpaths. These nested elements are useful when setting up class paths using reference Ids. The last, `<sysproperty>`, allows Java system properties to be set during the compiler run. This turns out to be necessary for supporting CMP EJB compilation in all environments.

TOPLink for Weblogic element

Deprecated

The toplink element is no longer required. Toplink beans can now be built with the standard weblogic element, as long as the newCMP attribute is set to "true"

The TopLink element is used to handle beans which use Toplink for the CMP operations. It is derived from the standard weblogic element so it supports the same set of attributes plus these additional attributes

Attribute	Description	Required
toplinkdescriptor	This specifies the name of the TOPLink deployment descriptor file contained in the 'descriptordir' directory.	Yes
toplinkdtd	This specifies the location of the TOPLink DTD file. This can be a file path or a file URL. This attribute is not required, but using a local DTD is recommended.	No, defaults to dtd file at www.objectpe

Examples

This example shows ejbjar being used to generate deployment jars using a Weblogic EJB container. This example requires the naming standard to be used for the deployment descriptors. Using this format will create a ejb jar file for each variation of '*-ejb-jar.xml' that is found in the deployment descriptor directory.

```
<ejbjar srcdir="${build.classes}"
        descriptordir="${descriptor.dir}">
  <weblogic destdir="${deploymentjars.dir}"
            classpath="${descriptorbuild.classpath}"/>
  <include name="**/*-ejb-jar.xml"/>
  <exclude name="**/*weblogic*.xml"/>
</ejbjar>
```

If weblogic is not in the Ant classpath, the following example shows how to specify the location of the weblogic DTDs. This example also show the use of a nested classpath element.

```
<ejbjar descriptordir="${src.dir}" srcdir="${build.classes}">
   <weblogic destdir="${deployment.webshop.dir}"
            keepgeneric="true"
            args="-g -keepgenerated ${ejbc.compiler}"
            suffix=".jar"
            oldCMP="false">
     <classpath>
       <pathelement path="${descriptorbuild.classpath}"/>
     </classpath>
   </weblogic>
   <include name="**/*-ejb-jar.xml"/>
   <exclude name="**/*-weblogic-ejb-jar.xml"/>
   <dtd
    publicId="-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
    location="${weblogic.home}/classes/weblogic/ejb/deployment/xml/ejb-jar.dtd"
   <dtd
    publicId="-//BEA Systems, Inc.//DTD WebLogic 5.1.0 EJB//EN"
    location="${weblogic.home}/classes/weblogic/ejb/deployment/xml/weblogic-ejb
</ejbjar>
```

This example shows ejbjar being used to generate a single deployment jar using a Weblogic EJB container. This example does not require the deployment descriptors to use the naming standard. This will create only one ejb jar file - 'TheEJBJar.jar'.

```
<ejbjar srcdir="${build.classes}"
        descriptordir="${descriptor.dir}"
        basejarname="TheEJBJar">
  <weblogic destdir="${deploymentjars.dir}"
            classpath="${descriptorbuild.classpath}"/>
  <include name="**/ejb-jar.xml"/>
  <exclude name="**/weblogic*.xml"/>
</ejbjar>
```

This example shows ejbjar being used to generate deployment jars for a TOPLink-enabled entity bean using a Weblogic EJB container. This example does not require the deployment descriptors to use the naming standard. This will create only one TOPLink-enabled ejb jar file – 'Address.jar'.

```
<ejbjar srcdir="${build.dir}"
        destdir="${solant.ejb.dir}"
        descriptordir="${descriptor.dir}"
        basejarname="Address">
        <weblogictoplink destdir="${solant.ejb.dir}"
                classpath="${java.class.path}"
                keepgeneric="false"
                toplinkdescriptor="Address.xml"
                toplinkdtd="file:///dtdfiles/toplink-cmp_2_5_1.dtd"
                suffix=".jar"/>
        <include name="**/ejb-jar.xml"/>
        <exclude name="**/weblogic-ejb-jar.xml"/>
</ejbjar>
```

This final example shows how you would set-up ejbjar under Weblogic 6.0. It also shows the use of the `<support>` element to add support files

```
<ejbjar descriptordir="${dd.dir}" srcdir="${build.classes.server}">
   <include name="**/*-ejb-jar.xml"/>
   <exclude name="**/*-weblogic-ejb-jar.xml"/>
   <support dir="${build.classes.server}">
        <include name="**/*.class"/>
   </support>
   <weblogic destdir="${deployment.dir}"
             keepgeneric="true"
             suffix=".jar"
             rebuild="false">
     <classpath>
        <pathelement path="${build.classes.server}"/>
```

```
         </classpath>
         <wlclasspath>
            <pathelement path="${weblogic.classes}"/>
         </wlclasspath>
      </weblogic>
   </ejbjar>
```

WebSphere element

The websphere element searches for the websphere specific deployment descriptors and adds them to the final ejb jar file. Websphere has two specific descriptors for session beans:

1. ibm-ejb-jar-bnd.xmi
2. ibm-ejb-jar-ext.xmi

and another two for container managed entity beans:

1. Map.mapxmi
2. Schema.dbxmi

In terms of WebSphere, the generation of container code and stubs is called deployment. This step can be performed by the websphere element as part of the jar generation process. If the switch ejbdeploy is on, the ejbdeploy tool from the websphere toolset is called for every ejb-jar. Unfortunately, this step only works, if you use the ibm jdk. Otherwise, the rmic (called by ejbdeploy) throws a ClassFormatError. Be sure to switch ejbdeploy off, if run ant with sun jdk.

For the websphere element to work, you have to provide a complete classpath, that contains all classes, that are required to reflect the bean classes. For ejbdeploy to work, you must also provide the classpath of the ejbdeploy tool and set the websphere.home property (look at the examples below).

Attribute	Description	Required
destdir	The base directory into which the generated weblogic ready jar files are deposited. Jar files are deposited in directories corresponding to their location within the descriptordir namespace.	Yes
ejbdeploy	Decides wether ejbdeploy is called. When you set this to true, be sure, to run ant with the ibm jdk.	No, defaults to true
suffix	String value appended to the basename of the deployment descriptor to create the filename of the WebLogic EJB jar file.	No, defaults to '.jar'.
keepgeneric	This controls whether the generic file used as input to ejbdeploy is retained.	No, defaults to false
rebuild	This controls whether ejbdeploy is called although no changes have occurred.	No, defaults to false
tempdir	A directory, where ejbdeploy will write temporary files	No, defaults to '_ejbdeploy_temp'.
dbName dbSchema	These options are passed to ejbdeploy.	No
dbVendor	This option is passed to ejbdeploy. Valid options are for example: • SQL92 • SQL99 • DB2UDBWIN_V71 • DB2UDBOS390_V6 • DB2UDBAS400_V4R5 • ORACLE_V8 • INFORMIX_V92 • SYBASE_V1192 • MYSQL_V323 • MSSQLSERVER_V7 This is also used to determine the name of the Map.mapxmi and Schema.dbxmi files, for example Account-DB2UDBWIN_V71-Map.mapxmi and Account-DB2UDBWIN_V71-Schema.dbxmi.	No

Attribute	Description	Required
codegen quiet novalidate noinform trace use35MappingRules	These options are all passed to ejbdeploy. All options except 'quiet' default to false.	No
rmicOptions	This option is passed to ejbdeploy and will be passed on to rmic.	No

This example shows ejbjar being used to generate deployment jars for all deployment descriptors in the descriptor dir:

```
     <property name="websphere.home" value="${was4.home}"/>
     <ejbjar srcdir="${build.class}" descriptordir="etc/ejb">
      <include name="*-ejb-jar.xml"/>
      <websphere dbvendor="DB2UDBOS390_V6"
                 ejbdeploy="true"
                 oldCMP="false"
                 tempdir="/tmp"
                 destdir="${dist.server}">
        <wasclasspath>
          <pathelement
location="${was4.home}/deploytool/itp/plugins/org.eclipse.core.boot/boot.jar"/>
          <pathelement
location="${was4.home}/deploytool/itp/plugins/com.ibm.etools.ejbdeploy/runtime/batc
          <pathelement location="${was4.home}/lib/xerces.jar"/>
          <pathelement location="${was4.home}/lib/ivjejb35.jar"/>
          <pathelement location="${was4.home}/lib/j2ee.jar"/>
          <pathelement location="${was4.home}/lib/vaprt.jar"/>
        </wasclasspath>
      <classpath>
        <path refid="build.classpath"/>
      </classpath>
      </websphere>
      <dtd
       publicId="-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
       location="${lib}/dtd/ejb-jar_1_1.dtd"/>
    </ejbjar>
```

iPlanet Application Server (iAS) element

The `<iplanet>` nested element is used to build iAS-specific stubs and skeletons and construct a JAR file which may be deployed to the iPlanet Application Server 6.0. The build process will always determine if the EJB stubs/skeletons and the EJB-JAR file are up to date, and it will do the minimum amount of work required.

Like the WebLogic element, a naming convention for the EJB descriptors is most commonly used to specify the name for the completed JAR file. For example, if the EJB descriptor ejb/Account-ejb-jar.xml is found in the descriptor directory, the iplanet element will search for an iAS-specific EJB descriptor file named ejb/Account-ias-ejb-jar.xml (if it isn't found, the task will fail) and a JAR file named ejb/Account.jar will be written in the destination directory. Note that when the EJB descriptors are added to the JAR file, they are automatically renamed META-INF/ejb-jar.xml and META-INF/ias-ejb-jar.xml.

Of course, this naming behaviour can be modified by specifying attributes in the ejbjar task (for example, basejarname, basenameterminator, and flatdestdir) as well as the iplanet element (for example, suffix). Refer to the appropriate documentation for more details.

Parameters:

destdir	The base directory into which the generated JAR files will be written. Each JAR file is written in directories which correspond to their location within the "descriptordir" namespace.	Yes
classpath	The classpath used when generating EJB stubs and skeletons. If omitted, the classpath specified in the "ejbjar" parent task will be used. If specified, the classpath elements will be prepended to the classpath specified in the parent "ejbjar" task. Note that nested "classpath" elements may also be used.	No
keepgenerated	Indicates whether or not the Java source files which are generated by ejbc will be saved or automatically deleted. If "yes", the source files will be retained. If omitted, it defaults to "no".	No
debug	Indicates whether or not the ejbc utility should log additional debugging statements to the standard output. If "yes", the additional debugging statements will be generated. If omitted, it defaults to "no".	No
iashome	May be used to specify the "home" directory for this iAS installation. This is used to find the ejbc utility if it isn't included in the user's system path. If specified, it should refer to the [install-location]/iplanet/ias6/ias directory. If omitted, the ejbc utility must be on the user's system path.	No
suffix	String value appended to the JAR filename when creating each JAR. If omitted, it defaults to ".jar".	No

As noted above, the iplanet element supports additional `<classpath>` nested elements.

Examples

This example demonstrates the typical use of the `<iplanet>` nested element. It will name each EJB-JAR using the ”basename” prepended to each standard EJB descriptor. For example, if the descriptor named ”Account-ejb-jar.xml” is processed, the EJB-JAR will be named ”Account.jar”

```
<ejbjar srcdir="${build.classesdir}"
        descriptordir="${src}">

        <iplanet destdir="${assemble.ejbjar}"
                 classpath="${ias.ejbc.cpath}"/>
        <include name="**/*-ejb-jar.xml"/>
        <exclude name="**/*ias-*.xml"/>
</ejbjar>
```

This example demonstrates the use of a nested classpath element as well as some of the other optional attributes.

```
<ejbjar srcdir="${build.classesdir}"
        descriptordir="${src}">

        <iplanet destdir="${assemble.ejbjar}"
                 iashome="${ias.home}"
                 debug="yes"
                 keepgenerated="yes">
                 <classpath>
                     <pathelement path="."/>
                     <pathelement path="${build.classpath}"/>
                 </classpath>
        </iplanet>
        <include name="**/*-ejb-jar.xml"/>
        <exclude name="**/*ias-*.xml"/>
</ejbjar>
```

This example demonstrates the use of basejarname attribute. In this case, the completed EJB-JAR will be named ”HelloWorld.jar” If multiple EJB descriptors might be found, care must be taken to ensure that the completed JAR files don't overwrite each other.

```
<ejbjar srcdir="${build.classesdir}"
        descriptordir="${src}"
        basejarname="HelloWorld">

        <iplanet destdir="${assemble.ejbjar}"
                 classpath="${ias.ejbc.cpath}"/>
```

```
            <include name="**/*-ejb-jar.xml"/>
            <exclude name="**/*ias-*.xml"/>
    </ejbjar>
```

This example demonstrates the use of the dtd nested element. If the local copies of the DTDs are included in the classpath, they will be automatically referenced without the nested elements. In iAS 6.0 SP2, these local DTDs are found in the [iAS-install-directory]/APPS directory. In iAS 6.0 SP3, these local DTDs are found in the [iAS-install-directory]/dtd directory.

```
    <ejbjar srcdir="${build.classesdir}"
            descriptordir="${src}">
            <iplanet destdir="${assemble.ejbjar}">
                     classpath="${ias.ejbc.cpath}"/>
            <include name="**/*-ejb-jar.xml"/>
            <exclude name="**/*ias-*.xml"/>

            <dtd publicId="-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 1.1//EN"
                 location="${ias.home}/APPS/ejb-jar_1_1.dtd"/>
            <dtd publicId="-//Sun Microsystems, Inc.//DTD iAS Enterprise JavaBeans 1.0//EN"
                 location="${ias.home}/APPS/IASEjb_jar_1_0.dtd"/>
    </ejbjar>
```

JOnAS (Java Open Application Server) element

The `<jonas>` nested element is used to build JOnAS-specific stubs and skeletons thanks to the GenIC specific tool, and construct a JAR file which may be deployed to the JOnAS Application Server. The build process will always determine if the EJB stubs/skeletons and the EJB-JAR file are up to date, and it will do the minimum amount of work required.

Like the WebLogic element, a naming convention for the EJB descriptors is most commonly used to specify the name for the completed JAR file. For example, if the EJB descriptor ejb/Account-ejb-jar.xml is found in the descriptor directory, the `<jonas>` element will search for a JOnAS-specific EJB descriptor file named ejb/Account-jonas-ejb-jar.xml and a JAR file named ejb/Account.jar will be written in the destination directory. But the `<jonas>` element can also use the JOnAS naming convention. With the same example as below, the EJB descriptor can also be named ejb/Account.xml (no base name terminator here) in the descriptor directory. Then the `<jonas>` element will search for a JOnAS-specific EJB descriptor file called ejb/jonas-Account.xml. This convention do not follow strictly the ejb-jar naming convention recommendation but is supported for backward compatibility with previous version of JOnAS.

Note that when the EJB descriptors are added to the JAR file, they are automatically renamed META-INF/ejb-jar.xml and META-INF/jonas-ejb-jar.xml.

Of course, this naming behavior can be modified by specifying attributes in the ejbjar task (for example, basejarname, basenameterminator, and flatdestdir) as well as the iplanet element (for example, suffix). Refer to the appropriate documentation for more details.

Parameters:

Attribute	Description	Required
destdir	The base directory into which the generated JAR files will be written. Each JAR file is written in directories which correspond to their location within the "descriptordir" namespace.	Yes
jonasroot	The root directory for JOnAS.	Yes
classpath	The classpath used when generating EJB stubs and skeletons. If omitted, the classpath specified in the "ejbjar" parent task will be used. If specified, the classpath elements will be prepended to the classpath specified in the parent "ejbjar" task (see also the ORB attribute documentation below). Note that nested "classpath" elements may also be used.	No
keepgenerated	true if the intermediate Java source files generated by GenIC must be deleted or not. If omitted, it defaults to false.	No
nocompil	true if the generated source files must not be compiled via the java and rmi compilers. If omitted, it defaults to false.	No
novalidation	true if the XML deployment descriptors must be parsed without validation. If omitted, it defaults to false.	No
javac	Java compiler to use. If omitted, it defaults to the value of build.compiler property.	No

Attribute	Description	Required
javacopts	Options to pass to the java compiler.	No
rmicopts	Options to pass to the rmi compiler.	No
secpropag	true if the RMI Skel. and Stub. must be modified to implement the implicit propagation of the security context (the transactional context is always provided). If omitted, it defaults to false.	No
verbose	Indicates whether or not to use -verbose switch. If omitted, it defaults to false.	No
additionalargs	Add additional args to GenIC.	No
keepgeneric	true if the generic JAR file used as input to GenIC must be retained. If omitted, it defaults to false.	No
suffix	String value appended to the JAR filename when creating each JAR. If omitted, it defaults to ".jar".	No
orb	Choose your ORB : RMI, JEREMIE, DAVID. If omitted, it defaults to the one present in classpath. If specified, the corresponding JOnAS JAR is automatically added to the classpath.	No
nogenic	If this attribute is set to true, JOnAS's GenIC will not be run on the EJB JAR. Use this if you prefer to run GenIC at deployment time. If omitted, it defaults to false.	No

As noted above, the jonas element supports additional `<classpath>` nested elements.

Examples

This example shows ejbjar being used to generate deployment jars using a JOnAS EJB container. This example requires the naming standard to be used for the deployment descriptors. Using this format will create a EJB JAR file for each variation of '*-jar.xml' that is found in the deployment descriptor directory.

```
<ejbjar srcdir="${build.classes}"
        descriptordir="${descriptor.dir}">
  <jonas destdir="${deploymentjars.dir}"
       jonasroot="${jonas.root}"
       orb="RMI"/>
  <include name="**/*.xml"/>
  <exclude name="**/jonas-*.xml"/>
  <support dir="${build.classes}">
       <include name="**/*.class"/>
  </support>
</ejbjar>
```

This example shows ejbjar being used to generate a single deployment jar using a JOnAS EJB container. This example does require the deployment descriptors to use the naming standard. This will create only one ejb jar file - 'TheEJBJar.jar'.

```
<ejbjar srcdir="${build.classes}"
        descriptordir="${descriptor.dir}"
        basejarname="TheEJBJar">
  <jonas destdir="${deploymentjars.dir}"
            jonasroot="${jonas.root}"
            suffix=".jar"
            classpath="${descriptorbuild.classpath}"/>
  <include name="**/ejb-jar.xml"/>
  <exclude name="**/jonas-ejb-jar.xml"/>
</ejbjar>
```

5.3.8 Echoproperties

Description

Displays all the current properties in the project. The output can be sent to a file if desired. You can also specify a subset of properties to save by naming a prefix: only properties starting with this prefix will be saved. This task can be used as a somewhat contrived means of returning data from an `<ant>` invocation, but is really for debugging build files.

Parameters

Attribute	Description	Required
destfile	If specified, the value indicates the name of the file to send the output of the statement to. The generated output file is compatible for loading by any Java application as a property file. If not specified, then the output will go to the Ant log.	No
prefix	a prefix which is used to filter the properties only those properties starting with this prefix will be echoed.	No
failonerror	By default, the "failonerror" attribute is enabled. If an error occurs while writing the properties to a file, and this attribute is enabled, then a BuildException will be thrown, causing the build to fail. If disabled, then IO errors will be reported as a log statement, and the build will continue without failure from this task.	No

Examples

```
<echoproperties/>
```

Report the current properties to the log.

```
<echoproperties destfile="my.properties"/>
```

Report the current properties to the file "my.properties", and will fail the build if the file could not be created or written to.

```
<echoproperties destfile="my.properties" failonerror="false" />
```

Report the current properties to the file "my.properties", and will log a message if the file could not be created or written to, but will still allow the build to continue.

```
<echoproperties prefix="java."/>
```

List all properties beginning with "java."

5.3.9 FTP

Description

The ftp task implements a basic FTP client that can send, receive, list, delete files, and create directories. See below for descriptions and examples of how to perform each task.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

The ftp task makes no attempt to determine what file system syntax is required by the remote server, and defaults to Unix standards. remotedir must be specified in the exact syntax required by the ftp server. If the usual Unix conventions are not supported by the server, separator can be used to set the file separator that should be used instead.

See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task does not currently use the proxy information set by the `<setproxy>` task, and cannot go through a firewall via socks.

Warning: for the get and delete actions to work properly with a Windows 2000 ftp server, it needs to be configured to generate Unix style listings, and not the default MS-DOS listing. Or someone needs to write the code to parse MS-DOS listings -any takers?

Parameters

Attribute	Description	Required
server	the address of the remote ftp server.	Yes
port	the port number of the remote ftp server. Defaults to port 21.	No
userid	the login id to use on the ftp server.	Yes
password	the login password to use on the ftp server.	Yes
remotedir	the directory to which to upload files on the ftp server.	No
action	the ftp action to perform, defaulting to "send". Currently supports "put", "get", "del", "list", "chmod" and "mkdir".	No
binary	selects binary-mode ("yes") or text-mode ("no") transfers. Defaults to "yes"	No
passive	selects passive-mode ("yes") transfers. Defaults to "no"	No
verbose	displays information on each file transferred if set to "yes". Defaults to "no".	No
depends	transfers only new or changed files if set to "yes". Defaults to "no".	No
newer	a synonym for depends.	No
separator	sets the file separator used on the ftp server. Defaults to "/".	No
umask	sets the default file permissions for new files, unix only.	No
chmod	sets or changes file permissions for new or existing files, unix only. If used with a put action, chmod will be issued for each file.	No
listing	the file to write results of the "list" action. Required for the "list" action, ignored otherwise.	No
ignoreNoncriticalErrors	flag which permits the task to ignore some non-fatal error codes sent by some servers during directory creation: wu-ftp in particular. Default: false	No
skipFailedTransfers	flag which enables unsuccessful file put, delete and get operations to be skipped with a warning and the remainder of the files still transferred. Default: false	No

Sending Files

The easiest way to describe how to send files is with a couple of examples:

```
<ftp server="ftp.apache.org"
```

```
       userid="anonymous"
       password="me@myorg.com">
    <fileset dir="htdocs/manual"/>
  </ftp>
```

Logs in to ftp.apache.org as anonymous and uploads all files in the htdocs/manual directory to the default directory for that user.

```
  <ftp server="ftp.apache.org"
       remotedir="incoming"
       userid="anonymous"
       password="me@myorg.com"
       depends="yes"
  >
    <fileset dir="htdocs/manual"/>
  </ftp>
```

Logs in to ftp.apache.org as anonymous and uploads all new or changed files in the htdocs/manual directory to the incoming directory relative to the default directory for anonymous.

```
  <ftp server="ftp.apache.org"
       port="2121"
       remotedir="/pub/incoming"
       userid="coder"
       password="java1"
       depends="yes"
       binary="no"
  >
    <fileset dir="htdocs/manual">
      <include name="**/*.html"/>
    </fileset>
  </ftp>
```

Logs in to ftp.apache.org at port 2121 as coder with password java1 and uploads all new or changed HTML files in the htdocs/manual directory to the /pub/incoming directory. The files are transferred in text mode. Passive mode has been switched on to send files from behind a firewall.

```
  <ftp server="ftp.nt.org"
       remotedir="c:\uploads"
       userid="coder"
       password="java1"
       separator="\"
       verbose="yes"

  >
    <fileset dir="htdocs/manual">
```

```
      <include name="**/*.html"/>
    </fileset>
  </ftp>
```

Logs in to the Windows-based ftp.nt.org as coder with password java1 and uploads all HTML files in the htdocs/manual directory to the c:
uploads directory. Progress messages are displayed as each file is uploaded.

Getting Files

Getting files from an FTP server works pretty much the same way as sending them does. The only difference is that the nested filesets use the remotedir attribute as the base directory for the files on the FTP server, and the dir attribute as the local directory to put the files into. The file structure from the FTP site is preserved on the local machine.

```
  <ftp action="get"
       server="ftp.apache.org"
       userid="anonymous"
       password="me@myorg.com">
    <fileset dir="htdocs/manual">
      <include name="**/*.html"/>
    </fileset>
  </ftp>
```

Logs in to ftp.apache.org as anonymous and recursively downloads all .html files from default directory for that user into the htdocs/manual directory on the local machine.

. Deleting Files

As you've probably guessed by now, you use nested fileset elements to select the files to delete from the remote FTP server. Again, the filesets are relative to the remote directory, not a local directory. In fact, the dir attribute of the fileset is ignored completely.

```
  <ftp action="del"
       server="ftp.apache.org"
       userid="anonymous"
       password="me@myorg.com">
    <fileset>
      <include name="**/*.tmp"/>
    </fileset>
  </ftp>
```

Logs in to ftp.apache.org as anonymous and tries to delete all *.tmp files from the default directory for that user. If you don't have permission to delete a file, a BuildException is thrown. Listing Files

```
  <ftp action="list"
       server="ftp.apache.org"
       userid="anonymous"
       password="me@myorg.com"
       listing="data/ftp.listing">
    <fileset>
      <include name="**"/>
    </fileset>
  </ftp>
```

This provides a file listing in data/ftp.listing of all the files on the FTP server relative to the default directory of the anonymous user. The listing is in whatever format the FTP server normally lists files.

Creating Directories

Note that with the mkdir action, the directory to create is specified using the remotedir attribute.

```
  <ftp action="mkdir"
       server="ftp.apache.org"
       userid="anonymous"
       password="me@myorg.com"
       remotedir="some/remote/dir"/>
```

This creates the directory some/remote/dir beneath the default root directory. As with all other actions, the directory separator character must be correct according to the desires of the FTP server.

5.3.10 IContract

Description

Instruments Java classes with iContract DBC preprocessor. The task can generate a properties file for iControl, a graphical user interface that lets you turn on/off assertions. iControl generates a control file that you can refer to from this task using the controlfile attribute.

Parameters

Attribute	Description	Required
srcdir	Location of the java files.	Yes
instrumentdir	Indicates where the instrumented source files should go.	Yes
repositorydir	Indicates where the repository source files should go.	Yes
builddir	Indicates where the compiled instrumented classes should go. Defaults to the value of instrumentdir. NOTE: Don't use the same directory for compiled instrumented classes and uninstrumented classes. It will break the dependency checking. (Classes will not be reinstrumented if you change them).	No
repbuilddir	Indicates where the compiled repository classes should go. Defaults to the value of repositorydir.	No
pre	Indicates whether or not to instrument for preconditions. Defaults to true unless controlfile is specified, in which case it defaults to false.	No
post	Indicates whether or not to instrument for postconditions. Defaults to true unless controlfile is specified, in which case it defaults to false.	No
invariant	Indicates whether or not to instrument for invariants. Defaults to true unless controlfile is specified, in which case it defaults to false.	No
failthrowable	The full name of the Throwable (Exception) that should be thrown when an assertion is violated. Defaults to java.lang.Error	No
verbosity	Indicates the verbosity level of iContract. Any combination of error*,warning*,note*,info*,progress*,debug* (comma separated) can be used. Defaults to error*	No
quiet	Indicates if iContract should be quiet. Turn it off if many your classes extend uninstrumented classes and you don't want warnings about this. Defaults to false	No
updateicontrol	If set to true, it indicates that the properties file for iControl in the current directory should be updated (or created if it doesn't exist). Defaults to false.	No
controlfile	The name of the control file to pass to iContract. Consider using iControl to generate the file. Default is not to pass a file.	Only if updateicontrol=true
classdir	Indicates where compiled (unistrumented) classes are located. This is required in order to properly update the icontrol.properties file, not for instrumentation.	Only if updateicontrol=true
targets	Name of the file that will be generated by this task, which lists all the classes that iContract will instrument. If specified, the file will not be deleted after execution. If not specified, a file will still be created, but it will be deleted after execution.	No

Note: iContract will use the java compiler indicated by the project's build.compiler property. See documentation of the Javac task for more information. Nested includes and excludes are also supported.

Example

Note: iContract will use the java compiler indicated by the project's build.compiler property. See documentation of the Javac task for more information.

Nested includes and excludes can be done very much the same way as any subclass of MatchingTask.

Example

```
<icontract
   srcdir="${build.src}"
   instrumentdir="${build.instrument}"
   repositorydir="${build.repository}"
   builddir="${build.instrclasses}"
   updateicontrol="true"
   classdir="${build.classes}"
   controlfile="control"
   targets="targets"
   verbosity="error*,warning*"
   quiet="true"
>
   <classpath refid="compile-classpath"/>
</icontract>
```

5.3.11 Jarlib-available

Description

Check whether an extension is present in a fileset or an extensionSet. If the extension is present then a property is set.

Note that this task works with extensions as defined by the "Optional Package" specification. For more information about optional packages, see the document Optional Package Versioning in the documentation bundle for your Java2 Standard Edition package, in file guide/extensions/versioning.html or online at http://java.sun.com/j2se/1.3/docs/guide/extensions/versioning.html.

See the Extension and ExtensionSet documentation for further details

Parameters

Attribute	Description	Required
property	The name of property to set if extensions is available.	Yes
file	The file to check for extension	No, one of file, nested ExtensionSet or nested fileset must be present.

Parameters specified as nested elements

extension

Extension the extension to search for.

fileset

FileSets are used to select sets of files to check for extension.

extensionSet

ExtensionSets is the set of extensions to search for extension in.

Examples

Search for extension in single file

```
<jarlib-available property="myext.present" file="myfile.jar">
  <extension
    extensionName="org.apache.tools.ant"
    specificationVersion="1.4.9"
    specificationVendor="Apache Software Foundation"/>
</jarlib-available>
```

Search for extension in single file refencing external Extension

```
<extension id="myext"
  extensionName="org.apache.tools.ant"
  specificationVersion="1.4.9"
  specificationVendor="Apache Software Foundation"/>

<jarlib-available property="myext.present" file="myfile.jar">
  <extension refid="myext"/>
</jarlib-available>
```

Search for extension in fileset

```
<extension id="myext"
  extensionName="org.apache.tools.ant"
  specificationVersion="1.4.9"
```

```
    specificationVendor="Apache Software Foundation"/>

  <jarlib-available property="myext.present">
    <extension refid="myext"/>
    <fileset dir="lib">
      <include name="*.jar"/>
    </fileset>
  </jarlib-available>
```

Search for extension in extensionSet

```
  <extension id="myext"
    extensionName="org.apache.tools.ant"
    specificationVersion="1.4.9"
    specificationVendor="Apache Software Foundation"/>

  <jarlib-available property="myext.present">
    <extension refid="myext"/>
    <extensionSet id="exts3">
      <libfileset
        includeUrl="false"
        includeImpl="true"
        dir="lib">
        <include name="*.jar"/>
      </libfileset>
    </extensionSet>
  </jarlib-available>
```

5.3.12 Jarlib-display

Description

Display the "Optional Package" and "Package Specification" information contained within the specified jars.

Note that this task works with extensions as defined by the "Optional Package" specification. For more information about optional packages, see the document Optional Package Versioning in the documentation bundle for your Java2 Standard Edition package, in file guide/extensions/versioning.html or online at http://java.sun.com/j2se/1.3/docs/guide/extensions/versioning.html.

See the Extension and ExtensionSet documentation for further details

Parameters

Attribute	Description	Required
file	The file to display extension information about.	No, but one of file or fileset must be present.

Parameters specified as nested elements

fileset

FileSets contain list of files to display Extension information about.

Examples

Display Extension info for a single file

```
<jarlib-display file="myfile.jar">
```

Display Extension info for a fileset

```
<jarlib-display>
  <fileset dir="lib">
    <include name="*.jar"/>
  </fileset>
</jarlib-display>
```

5.3.13 Jarlib-manifest

Description

Task to generate a manifest that declares all the dependencies in manifest. The dependencies are determined by looking in the specified path and searching for Extension / ”Optional Package” specifications in the manifests of the jars.

Note that this task works with extensions as defined by the ”Optional Package” specification. For more information about optional packages, see the document Optional Package Versioning in the documentation bundle for your Java2 Standard Edition package, in file guide/extensions/versioning.html or online at http://java.sun.com/j2se/1.3/docs/guide/extensions/versioning.html.

See the Extension and ExtensionSet documentation for further details

Parameters

Attribute	Description	Required
destfile	The file to generate Manifest into	Yes.

Parameters specified as nested elements

extension

Extension the extension that this library implements.

depends

ExtensionSets containing all dependencies for jar.

options

ExtensionSets containing all optional dependencies for jar. (Optional dependencies will be used if present else they wilkl be ignored)

Examples

Basic Manifest generated for single Extension

```
<extension id="e1"
    extensionName="MyExtensions"
    specificationVersion="1.0"
    specificationVendor="Peter Donald"
    implementationVendorID="vv"
    implementationVendor="Apache"
    implementationVersion="2.0"
    implementationURL="http://somewhere.com"/>

<jarlib-manifest destfile="myManifest.txt">
    <extension refid="e1"/>
</jarlib-manifest>
```

Search for extension in fileset
A large example with required and optional dependencies

```
<extension id="e1"
    extensionName="MyExtensions"
    specificationVersion="1.0"
    specificationVendor="Peter Donald"
    implementationVendorID="vv"
    implementationVendor="Apache"
    implementationVersion="2.0"
    implementationURL="http://somewhere.com"/>

<extensionSet id="option.ext">
    <libfileset dir="lib/option">
        <include name="**/*.jar"/>
    </libfileset>
</extensionSet>

<extensionSet id="depends.ext">
    <libfileset dir="lib/required">
        <include name="*.jar"/>
    </libfileset>
</extensionSet>

<jarlib-manifest destfile="myManifest.txt">
    <extension refid="e1"/>
    <depends refid="depends.ext"/>
    <options refid="option.ext"/>
</jarlib-manifest>
```

5.3.14 Jarlib-resolve

Description

Try to locate a jar to satisfy an extension and place location of jar into property. The task allows you to add a number of resolvers that are capable of locating a library for a specifc extension. Each resolver will be attempted in specified order until library is found or no resolvers are left. If no resolvers are left and failOnError is true then a BuildException will be thrown.

Note that this task works with extensions as defined by the "Optional Package" specification. For more information about optional packages, see the document Optional Package Versioning in the documentation bundle for your Java2 Standard Edition package, in file guide/extensions/versioning.html or online at http://java.sun.com/j2se/1.3/docs/guide/extensions/versioning.html.

See the Extension and ExtensionSet documentation for further details

Parameters

Attribute	Description	Required
property	The name of property to set to library location.	Yes
failOnError	True if failure to locate library should result in build exception.	No, defaults to true.
checkExtension	True if librarys returned by nested resolvers should be checked to see if they supply extension.	No, defaults to true.

Parameters specified as nested elements

extension

Extension the extension to resolve. Must be present

location

The location sub element allows you to look for a library in a location relative to project directory.

Attribute	Description	Required
location	The pathname of library.	Yes

url

The url resolver allows you to download a library from a URL to a local file.

Attribute	Description	Required
url	The URL to download.	Yes
destfile	The file to download URL into.	No, But one of destfile or destdir must be present
destdir	The directory in which to place downloaded file.	No, But one of destfile or destdir must be present

ant

The ant resolver allows you to run a ant build file to generate a library.

Attribute	Description	Required
antfile	The build file.	Yes
destfile	The file that the ant build creates.	Yes
target	The target to run in build file.	No

Examples

Resolve Extension to file. If file does not exist or file does not implement extension then throw an exception.

```
<extension id="dve.ext"
  extensionName="org.realityforge.dve"
  specificationVersion="1.2"
  specificationVendor="Peter Donald"/>

<jarlib-resolve property="dve.library">
  <extension refid="dve.ext"/>
  <location location="/opt/jars/dve.jar"/>
</jarlib-resolve>
```

Resolve Extension to url. If url does not exist or can not write to destfile or files does not implement extension then throw an exception.

```
<extension id="dve.ext"
  extensionName="org.realityforge.dve"
  specificationVersion="1.2"
  specificationVendor="Peter Donald"/>

<jarlib-resolve property="dve.library">
  <extension refid="dve.ext"/>
  <url url="http://www.realityforge.net/jars/dve.jar" destfile="lib/dve.jar"/>
</jarlib-resolve>
```

Resolve Extension to file produce by ant build. If file does not get produced or ant file is missing or build fails then throw an exception (Note does not check that library implements extension).

```
<extension id="dve.ext"
  extensionName="org.realityforge.dve"
  specificationVersion="1.2"
  specificationVendor="Peter Donald"/>

<jarlib-resolve property="dve.library" checkExtension="false">
  <extension refid="dve.ext"/>
  <ant antfile="../dve/build.xml" target="main" destfile="lib/dve.jar"/>
</jarlib-resolve>
```

Resolve Extension via multiple methods. First check local file to see if it implements extension. If it does not then try to build it from source in parralel directory. If that fails then finally try to download it from a website. If all steps fail then throw a build exception.

```
<extension id="dve.ext"
  extensionName="org.realityforge.dve"
  specificationVersion="1.2"
  specificationVendor="Peter Donald"/>

<jarlib-resolve property="dve.library">
  <extension refid="dve.ext"/>
  <location location="/opt/jars/dve.jar"/>
  <ant antfile="../dve/build.xml" target="main" destfile="lib/dve.jar"/>
  <url url="http://www.realityforge.net/jars/dve.jar" destfile="lib/dve.jar"/>
</jarlib-resolve>
```

5.3.15 JavaCC

Description

Invokes the JavaCC compiler compiler on a grammar file.

To use the javacc task, set the target attribute to the name of the grammar file to process. You also need to specify the directory containing the JavaCC installation using the javacchome attribute, so that ant can find the JavaCC classes. Optionally, you can also set the outputdirectory to write the generated file to a specific directory. Otherwise javacc writes the generated files to the directory containing the grammar file.

This task only invokes JavaCC if the grammar file is newer than the generated Java files. javacc assumes that the Java class name of the generated parser is the same as the name of the grammar file, ignoring the .jj. If this is not the case, the javacc task will still work, but it will always generate the output files.

Parameters

Attribute	Description	Required
target	The grammar file to process.	Yes
javacchome	The directory containing the JavaCC distribution.	Yes
outputdirectory	The directory to write the generated files to. If not set, the files are written to the directory containing the grammar file.	No
buildparser	Sets the BUILD_PARSER grammar option. This is a boolean option.	No
buildtokenmanager	Sets the BUILD_TOKEN_MANAGER grammar option. This is a boolean option.	No
cachetokens	Sets the CACHE_TOKENS grammar option. This is a boolean option.	No
choiceambiguitycheck	Sets the CHOICE_AMBIGUITY_CHECK grammar option. This is an integer option.	No
commontokenaction	Sets the COMMON_TOKEN_ACTION grammar option. This is a boolean option.	No
debuglookahead	Sets the DEBUG_LOOKAHEAD grammar option. This is a boolean option.	No

Attribute	Description	Required
debugparser	Sets the DEBUG_PARSER grammar option. This is a boolean option.	No
debugtokenmanager	Sets the DEBUG_TOKEN_MANAGER grammar option. This is a boolean option.	No
errorreporting	Sets the ERROR_REPORTING grammar option. This is a boolean option.	No
forcelacheck	Sets the FORCE_LA_CHECK grammar option. This is a boolean option.	No
ignorecase	Sets the IGNORE_CASE grammar option. This is a boolean option.	No
javaunicodeescape	Sets the JAVA_UNICODE_ESCAPE grammar option. This is a boolean option.	No
lookahead	Sets the LOOKAHEAD grammar option. This is an integer option.	No
optimizetokenmanager	Sets the OPTIMIZE_TOKEN_MANAGER grammar option. This is a boolean option.	No
otherambiguitycheck	Sets the OTHER_AMBIGUITY_CHECK grammar option. This is an integer option.	No
sanitycheck	Sets the SANITY_CHECK grammar option. This is a boolean option.	No
static	Sets the STATIC grammar option. This is a boolean option.	No
unicodeinput	Sets the UNICODE_INPUT grammar option. This is a boolean option.	No
usercharstream	Sets the USER_CHAR_STREAM grammar option. This is a boolean option.	No
usertokenmanager	Sets the USER_TOKEN_MANAGER grammar option. This is a boolean option.	No

Example

```
<javacc
    target="src/Parser.jj"
    outputdirectory="build/src"
    javacchome="c:/program files/JavaCC"
    static="true"
/>
```

This invokes JavaCC on grammar file src/Parser.jj, writing the generated files to build/src. The grammar option STATIC is set to true when invoking JavaCC.

5.3.16 Javah

Description

Generates JNI headers from a Java class.

When this task executes, it will generate the C header and source files that are needed to implement native methods. JNI operates differently depending on whether JDK1.2 (or later) or pre-JDK1.2 systems are used.

Parameters

Attribute	Description	Required
class	the fully-qualified name of the class (or classes, separated by commas)	Yes
outputFile	concatenates the resulting header or source files for all the classes listed into this file	Yes
destdir	sets the directory where javah saves the header files or the stub	files.
force	specifies that output files should always be written (JDK1.2 only)	No
old	specifies that old JDK1.0-style header files should be generated (otherwise output file contain JNI-style native method function prototypes) (JDK1.2 only)	No
stubs	generate C declarations from the Java object file (used with old)	No
verbose	causes Javah to print a message concerning the status of the generated files	No
classpath	the classpath to use.	No
bootclasspath	location of bootstrap class files.	No
extdirs	location of installed extensions.	No

Either outputFile or destdir must be supplied, but not both. Examples

```
<javah destdir="c" class="org.foo.bar.Wibble"/>
```

makes a JNI header of the named class, using the JDK1.2 JNI model. Assuming the directory 'c' already exists, the file org_foo_bar_Wibble.h is created there. If this file already exists, it is left unchanged.

```
<javah outputFile="wibble.h">
  <class name="org.foo.bar.Wibble,org.foo.bar.Bobble"/>
</javah>
```

is similar to the previous example, except the output is written to a file called wibble.h in the current directory.

```
<javah destdir="c" force="yes">
  <class name="org.foo.bar.Wibble"/>
  <class name="org.foo.bar.Bobble"/>
  <class name="org.foo.bar.Tribble"/>
</javah>
```

writes three header files, one for each of the classes named. Because the force option is set, these header files are always written when the Javah task is invoked, even if they already exist.

```
<javah destdir="c" verbose="yes" old="yes" force="yes">
  <class name="org.foo.bar.Wibble"/>
  <class name="org.foo.bar.Bobble"/>
  <class name="org.foo.bar.Tribble"/>
</javah>
<javah destdir="c" verbose="yes" stubs="yes" old="yes" force="yes">
  <class name="org.foo.bar.Wibble"/>
  <class name="org.foo.bar.Bobble"/>
  <class name="org.foo.bar.Tribble"/>
</javah>
```

writes the headers for the three classes using the 'old' JNI format, then writes the corresponding .c stubs. The verbose option will cause Javah to describe its progress.

5.3.17 JspC

Description

Ant task to run the JSP compiler and turn JSP pages into Java source.

It can be used to precompile JSP pages for fast initial invocation of JSP pages, deployment on a server without the full JDK installed, or simply to syntax check the pages without deploying them. In most cases, a javac task is usually the next stage in the build process. The task does basic dependency checking to prevent unnecessary recompilation -this checking compares source and destination timestamps, and does not factor in class or taglib dependencies, or `<jsp:include>` references.

By default the task uses the Jasper JSP compiler. This means the task needs jasper.jar and jasper-runtime.jar, which come with builds of Tomcat 4/Catalina from the Jakarta Tomcat project. We recommend Tomcat version 4.1.x for the most robust version of jasper.

There are many limitations with this task which partially stem from the many versions of Jasper, others from implementation 'issues' in the task (i.e. nobody's willingness to radically change large bits of it to work around jasper). Because of this and the fact that JSP pages do not have to be portable across implementations -or versions of implementations- this task is better used for validating JSP pages before deployment, rather than precompiling them. For that, just deploy and run your httpunit junit tests after deployment to compile and test your pages, all in one go.

Parameters

The Task has the following attributes:

Attribute	Description	Required
destdir	Where to place the generated files. They are located under here according to the given package name.	Yes
srcdir	Where to look for source jsp files.	Yes
verbose	The verbosity integer to pass to the compiler. Default="0"	No
package	Name of the destination package for generated java classes.	No
compiler	class name of a JSP compiler adapter, such as "jasper" or "jasper41"	No -defaults to "jasper"
ieplugin	Java Plugin classid for Internet Explorer.	No
mapped	(boolean) Generate separate write() calls for each HTML line in the JSP.	No
classpath	The classpath to use to run the jsp compiler. This can also be specified by the nested element classpath Path).	No, but it seems to work better when used
classpathref	A Reference. As per classpath	No
failonerror	flag to control action on compile failures: default=yes	No
uribase	The uri context of relative URI references in the JSP pages. If it does not exist then it is derived from the location of the file relative to the declared or derived value of uriroot.	No
uriroot	The root directory that uri files should be resolved against.	No
compiler	Class name of jsp compiler adapter to use. Defaults to the standard adapter for Jasper.	No
compilerclasspath	The classpath used to find the compiler adapter specified by the compiler attribute.	No
webinc	Output file name for the fraction of web.xml that lists servlets.	No
webxml	File name for web.xml to be generated	No

The mapped option will, if set to true, split the JSP text content into a one line per call format. There are comments above and below the mapped write calls to localize where in the JSP file each line of text comes from. This can lead to a minor performance degradation (but it is bound by a linear complexity). Without this options all adjacent writes are concatenated into a single write.

The ieplugin option is used by the `<jsp:plugin>` tags. If the Java Plug-in COM Class-ID you want to use changes then it can be specified here. This should not need to be altered.

uriroot specifies the root of the web application. This is where all absolute uris will be resolved from. If it is not specified then the first JSP page will be used to derive it. To derive it each parent directory of the first JSP page is searched for a WEB-INF directory, and the directory closest to the JSP page that has one will be used. If none can be found then the directory Jasperc was called from will be used. This only affects pages translated from an explicitly declared JSP file -including references to taglibs

uribase is used to establish the uri context of relative URI references in the JSP pages. If it does not exist then it is derived from the location of the file relative to the declared or derived value of uriroot. This only affects pages translated from an explicitly declared JSP file.

Parameters specified as nested elements

This task is a directory based task, like javac, so the jsp files to be compiled are located as java files are by javac. That is, elements such as includes and excludes can be used directly inside the task declaration.

Elements specific to the jspc task are:-

classpath

The classpath used to compile the JSP pages, specified as for any other classpath.

classpathref

a reference to an existing classpath

webapp

Instructions to jasper to build an entire web application. The base directory must have a WEB-INF subdirectory beneath it. When used, the task hands off all dependency checking to the compiler.

Attribute	Description	Required
basedir	the base directory of the web application	Yes

Example

```
<jspc srcdir="${basedir}/src/war"
      destdir="${basedir}/gensrc"
      package="com.i3sp.jsp"
      compiler="jasper41"
      verbose="9">
  <include name="**/*.jsp" />
</jspc>
```

Build all jsp pages under src/war into the destination /gensrc, in a package heirarchy beginning with com.i3sp.jsp.

```
<jspc
      destdir="interim"
      verbose="1"
      srcdir="src"
      compiler="jasper41"
      package="com.i3sp.jsp">
  <include name="**/*.jsp" />
</jspc>
<depend
         srcdir="interim"
         destdir="build"
         cache="build/dependencies"
         classpath="lib/taglibs.jar"/>
<javac
         srcdir="interim"
         destdir="build"
         classpath="lib/taglibs.jar"
         debug="on"/>
```

Generate jsp pages then javac them down to bytecodes. Include lib/taglib jar in the java compilation. Dependency checking is used to scrub the java files if class dependencies indicate it is needed.

Notes

Using the package attribute it is possible to identify the resulting java files and thus do full dependency checking - this task should only rebuild java files if their jsp file has been modified. However, this only works with some versions of jasper. By default the checking supports tomcat 4.0.x with the ”jasper” compiler, set the compiler to ”jasper41” for the tomcat4.1.x dependency checking. Even when it does work, changes in .TLD imports or in compile time includes do not get picked up.

Jasper generates JSP pages against the JSP1.2 specification -an implementation of version 2.3 of the servlet specification is needed to compile or run the java code.

5.3.18 JDepend

Description

Invokes the JDepend parser.

This parser ”traverses a set of Java source file directories and generates design quality metrics for each Java package”. It allows to ”automatically measure the quality of a design in terms of its extensibility, reusability, and maintainability to effectively manage and control package dependencies.”

Source file directories are defined by nested `<sourcespath>`, see nested elements.

Optionally, you can also set the outputfile name where the output is stored. By default the task writes its report to the standard output.

The task requires at least the JDepend 1.2 version.

Note: whereas the JDepend tool can be customized to exclude some packages, the current jdepend And Task does not have parameters to allow these exclusions. Read JDepend specific documentation for that purpose.

Parameters

Attribute	Description	Required
outputfile	The output file name. If not set, the output is printed on the standard output.	No
format	The format to write the output in. The default is "text", the alternative is "xml"	No
fork	Run the tests in a separate VM.	No, default is "off"
haltonerror	Stop the build process if an error occurs during the jdepend analysis.	No, default is "off"
timeout	Cancel the operation if it doesn't finish in the given time (measured in milliseconds). (Ignored if fork is disabled.)	No
jvm	The command used to invoke the Java Virtual Machine, default is 'java'. The command is resolved by java.lang.Runtime.exec(). (Ignored if fork is disabled.)	No, default "java"
dir	The directory to invoke the VM in. (Ignored if fork is disabled)	No
classpathref	the classpath to use, given as reference to a PATH defined elsewhere.	No

Nested Elements

jdepend supports two nested elements `<classpath>` and `<sourcespath>`, that represent PATH like structures.

`<sourcespath>` is used to define the paths of the source code to analyze.

Examples

```
<jdepend classpathref="base.path">
    <sourcespath>
        <pathelement location="src"/>
    </sourcespath>
</jdepend>
```

This invokes JDepend on the src directory, writing the output on the standard output. The classpath is defined using a classpath reference.

```
<jdepend outputfile="docs/jdepend.xml" fork="yes" format="xml">
    <sourcespath>
        <pathelement location="src"/>
    </sourcespath>
    <classpath>
        <pathelement location="classes"/>
        <pathelement location="lib/jdepend.jar"/>
    </classpath>
</jdepend>
```

This invokes JDepend in a separate VM on the src and testsrc directories, writing the output to the ¡docs/jdepend.xml¿ file in xml format. The classpath is defined using nested elements.

5.3.19 JJTree

Description

Invokes the JJTree preprocessor for the JavaCC compiler compiler. It inserts parse tree building actions at various places in the JavaCC source that it generates. The output of JJTree is run through JavaCC to create the parser.

To use the jjtree task, set the target attribute to the name of the jjtree grammar file to process. You also need to specify the directory containing the JavaCC installation using the javacchome attribute, so that ant can find the JavaCC classes. Optionally, you can also set the outputdirectory to write the generated file to a specific directory. Otherwise jjtree writes the generated JavaCC grammar file to the directory containing the JJTree grammar file.

This task only invokes JJTree if the grammar file is newer than the generated JavaCC file.

Parameters

Attribute	Description	Required
target	The jjtree grammar file to process.	Yes
javacchome	The directory containing the JavaCC distribution.	Yes
outputdirectory	The directory to write the generated file to. If not set, the files are written to the directory containing the grammar file.	No
buildnodefiles	Sets the BUILD_NODE_FILES grammar option. This is a boolean option.	No
multi	Sets the MULTI grammar option. This is a boolean option.	No
nodedefaultvoid	Sets the NODE_DEFAULT_VOID grammar option. This is a boolean option.	No
nodefactory	Sets the NODE_FACTORY grammar option. This is boolean option.	No
nodescopehook	Sets the NODE_SCOPE_HOOK grammar option. This is a boolean option.	No
nodeusesparser	Sets the NODE_USES_PARSER grammar option. This is a boolean option.	No
static	Sets the STATIC grammar option. This is a boolean option.	No
visitor	Sets the VISITOR grammar option. This is a boolean option.	No
nodepackage	Sets the NODE_PACKAGE grammar option. This is a string option.	No
visitorexception	Sets the VISITOR_EXCEPTION grammar option. This is a string option.	No
nodeprefix	Sets the NODE_PREFIX grammar option. This is a string option.	No

Example

```
<jjtree
    target="src/Parser.jjt"
    outputdirectory="build/src"
    javacchome="c:/program files/JavaCC"
    nodeusesparser="true"
/>
```

This invokes JJTree on grammar file src/Parser.jjt, writing the generated grammar file, Parser.jj, file to build/src. The grammar option NODE_USES_PARSER is set to true when invoking JJTree.

5.3.20 Jlink

Deprecated

This task has been deprecated. Use the zipfileset and zipgroupfileset attributes of the Jar task or Zip task instead.

Description

Links entries from sub-builds and libraries.

The jlink task can be used to build jar and zip files, similar to the jar task. However, jlink provides options for controlling the way entries from input files are added to the output file. Specifically, capabilities for merging entries from multiple zip or jar files is available.

If a mergefile is specified directly (eg. at the top level of a mergefiles pathelement) and the mergefile ends in ".zip" or ".jar", entries in the mergefile will be merged into the outfile. A file with any other extension will be added to the output file, even if it is specified in the mergefiles element. Directories specified in either the mergefiles or addfiles element are added to the output file as you would expect: all files in subdirectories are recursively added to the output file with appropriate prefixes in the output file (without merging).

In the case where duplicate entries and/or files are found among the files to be merged or added, jlink merges or adds the first entry and ignores all subsequent entries.

jlink ignores META-INF directories in mergefiles. Users should supply their own manifest information for the output file.

It is possible to refine the set of files that are being jlinked. This can be done with the includes, includesfile, excludes, excludesfile, and defaultexcludes attributes on the addfiles and mergefiles nested elements. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns. The patterns are relative to the base directory.

Parameters

Attribute	Description	Required
outfile	the path of the output file.	Yes
compress	whether or not the output should be compressed. true, yes, or on result in compressed output. If omitted, output will be uncompressed (inflated).	No
mergefiles	files to be merged into the output, if possible.	At least one of mergefiles
addfiles	files to be added to the output.	At least one of addfiles

Examples

The following will merge the entries in mergefoo.jar and mergebar.jar into out.jar. mac.jar and pc.jar will be added as single entries to out.jar.

```
<jlink compress="false" outfile="out.jar">
   <mergefiles>
     <pathelement path="${build.dir}/mergefoo.jar"/>
     <pathelement path="${build.dir}/mergebar.jar"/>
   </mergefiles>
   <addfiles>
     <pathelement path="${build.dir}/mac.jar"/>
     <pathelement path="${build.dir}/pc.zip"/>
   </addfiles>
</jlink>
```

Non-deprecated alternative to the above:

```
<jar compress="false" destfile="out.jar">
  <zipgroupfileset dir="${build.dir}">
    <include name="mergefoo.jar"/>
    <include name="mergebar.jar"/>
  </zipgroupfileset>
  <fileset dir="${build.dir}">
    <include name="mac.jar"/>
    <include name="pc.jar"/>
  </fileset>
</jar>
```

Suppose the file foo.jar contains two entries: bar.class and barnone/myClass.zip. Suppose the path for file foo.jar is build/tempbuild/foo.jar. The following example will provide the entry tempbuild/foo.jar in the out.jar.

```
<jlink compress="false" outfile="out.jar">
   <mergefiles>
```

```
    <pathelement path="build/tempbuild"/>
  </mergefiles>
</jlink>
```

However, the next example would result in two top-level entries in out.jar, namely bar.class and barnone/myClass.zip

```
<jlink compress="false" outfile="out.jar">
  <mergefiles>
    <pathelement path="build/tempbuild/foo.jar"/>
  </mergefiles>
</jlink>
```

5.3.21 JProbe Coverage

```
JProbe

by

    * Stephane Bailliez (sbailliez@imediation.com)
```

This task runs the tools from the JProbe suite. For more information, visit http://www.sitraka.com. An evaluation version is available for download if you already don't own it.

This task has been written using JProbe Suite Server Side 3.0.

It is highly recommended to read the JProbe documentation to understand the values of the command line arguments described below. This document is less complete than the manual, it only gives the basic information and is not intended as a replacement to the manual.

Tasks

JPCoverage	Measure coverage of Java code.
JPCovMerge	Merge different snapshots into one.
JPCovReport	Create a report from a snapshot

JPCoverage

Perform code covering functions by comparing source code line execution to the programs source code as a whole.

Parameters

Attribute	Description	Required
home	The directory where JProbe is intalled.	Yes
vm	Indicates which virtual machine to run. Must be one of ”jdk117”, ”jdk118” or ”java2”.If ”java2” is specified, the user is also required to specify a path via javaexe, otherwise it will check if the current executing VM is 1.2+ and use its java.home property to determine its location.	No, default to embedded VM if 1.2+
javaexe	The path to the java executable.	No, use only for java2 vm.
applet	Run an applet. The default is false, unless the file under analysis ends with htm or html.	No, default is ”false”.
seedname	Seed name for the temporary snapshot files (files will be named seed.jpc, seed1.jpc, seed2.jpc, ...)	No, default to ”snapshot”
exitprompt	Toggles display of the console prompt: ”Press Enter to close this window.” ”always”: Always displays the prompt. ”never”: Never displays the prompt. ”error”: Only displays prompt after an error.	No, default is ”never”
finalsnapshot	Type of snapshot to send at program termination. Must be one of ”none”,”coverage”,”all”	No, default to ”coverage”
recordfromstart	Must be one of ”coverage”, ”all”, ”none”. If you want Coverage to start analyzing as soon as the program begins to run, use ”all”. If not, select ”none”.	No, default to ”coverage”
warnlevel	Set warning level (0-3, where 0 is the least amount of warnings).	No, default to 0
snapshotdir	The path to the directory where snapshot files are stored. Choose a directory that is reachable by both the remote and local computers, and enter the same path on the command line and in the viewer.	No, default to current directory
workingdir	The physical path to the working directory for the VM.	No, default is current directory.
tracknatives	Test native methods. Note that testing native methods with Java 2 disables the JIT	No, default to ”false”.
classname	the name of the class to analyze.	Yes

Nested Elements

classpath

jpcoverage supports a nested `<classpath>` element, that represents a PATH like structure.

jvmarg

Additional parameters may be passed to the VM via nested `<jvmarg>` attributes, for example:

```
<jpcoverage home="c:\jprobe" classname="MyClass">
  <jvmarg value="-classic"/>
  <classpath path="."/>
</jpcoverage>
```

would run the coverage on "MyClass" in classic mode VM.

`<jvmarg>` allows all attributes described in Command line arguments.

arg

Parameters may be passed to the executed class via nested `<arg>` attributes, as described in Command line arguments.

socket

Define a host and port to connect to if you want to do remote viewing.

Attribute	Description	Required
host	the host name/ip of the machine on which the Viewer is running	No, default to localhost
port	The port number on which you will connect to the Viewer	No, default to 4444

filters

Defines class/method filters based on pattern matching. The syntax is filters is similar to a fileset.

Attribute	Description	Required
defaultexclude	As a default, the coverage excludes all classes and methods. Default filters are equivalent to `<filters>` `<exclude class="*" method="*"/>` `</filters>`	No, default to "true"

As seen above, nested elements are include and exclude with a name attribute.

Attribute	Description	Required
class	The class mask as a simple regular expression	No, defaults to "*"
method	The method mask as a simple regular expression	No, defaults to "*"
enabled	is the filter enabled?	No, defaults to true

Example of filters

```
<filters>
    <include class="com.mycompany.*" method="*"/>
    <exclude class="com.mycompany.MyClass" method="test*"/>
</filters>
```

reports the coverage on all packages, classes and methods from com.mycompany except all methods starting by test on the class MyClass in the package com.mycompany

triggers

Define a number of events to use for interacting with the collection of data performed during coverage. For example you may run a whole application but only decide to collect data once it reaches a certain method and once it exits another one.

The only type of nested element is the method element (triggers are performed on method) and it has the following attributes:

Attribute	Description	Required
name	The name of the method(s) as a regular expression. The name is the fully qualified name on the form package.classname.method	Yes
event	the event on the method that will trigger the action. Must be ”enter” or ”exit”.	Yes
action	the action to execute. Must be one of ”clear”, ”pause”, ”resume”, ”snapshot”, ”suspend”, or ”exit”. They respectively clear recording, pause recording, resume recording, take a snapshot, suspend the recording and exit the program.	Yes

Example of triggers

```
<triggers>
  <method name="ClassName.*()" event="enter" action="snapshot"/>
  <method name="ClassName.MethodName()" event="exit" action="exit"/>
</triggers>
```

Will take a snapshot when it enters any method of the class ClassName and will exit the program once it exits the method MethodName of the same class.

JPCovMerge

Description

Perform the merge of several snapshots into a single one.

Parameters

Attribute	Description	Required
home	The directory where JProbe is installed.	Yes
tofile	the output filename that will be the result of the name.	Yes
verbose	Perform the merge in verbose mode giving details about the snapshot processing.	No. Default to false

jpcovmerge collects snapshots using the nested `<FileSet>` element. Example of merge

```
<jpcovmerge home="c:\jprobe" tofile="merge.jpc" verbose="true">
  <fileset dir="./snapshots">
    <include name="snap*.jpc"/>
  </fileset>
</jpcovmerge>
```

would run the merge in verbose mode on all snapshot files starting by snap in the directory snapshots. The resulting file will be named merge.jpc.

JPCovReport

Description

Generate a readable/printable report of a snapshot. Note that you will need Jakarta Oro in Ant classpath, to run the reference feature.

Parameters

Attribute	Description	Required
home	The directory where JProbe is intalled.	Yes
format	The format of the generated report. Must be "xml", "html" or "text"	No, default to "html"
type	The type of report to be generated. Must be "executive", "summary", "detailed" or "verydetailed"	No. Default to "detailed"
percent	A numeric value for the threshold for printing methods. Must be between 0 and 100.	No, default to 100
snapshot	The name of the snapshot file that is the source to the report.	Yes
tofile	The name of the generated output file	Yes
includesource	Include text of the source code lines. Only applies to format="xml" and type="verydetailed"	No. Defaults to "yes"

sourcepath

Path to source files can be set via nested sourcepath elements that are PATH like structures. reference (only applies to format="xml")

A reference is a set of classes whose coverage information will be checked against. Since Coverage is only able to give you information about loaded classes, it will only report classes that were at least used in some points in your tests, therefore you will not be able to know what classes are not exercised at all during your tests. The reference is an additional feature that will analyze the bytecode of all classes in a given classpath that match some filters and modify the XML report accordingly. In short, it will:

- remove the classes that do not exists in the reference classpath. (For example you might have in your report some helper test classes that you do not want to appear in the report, but are unable to filter without adding hundred of filters for all your classes).
- add classes that exists in the reference set and match the filters but are not reported.
- remove abstract methods that are incorrectly reported in JProbe 3.0 (should be fixed in a later SP)
- remove classes/methods that do not match the filters.

classpath

Path to the reference set of files can be set via nested classpath elements that are PATH like structures.

filters

Nested elements are include and exclude with a class and method attribute.

Attribute	Description	Required
class	The class mask as a simple regular expression	No, default to *
method	The method mask as a simple regular expression	No, default to *

Example of report

```
<jpcovreport home="c:\jprobe" snapshot="merge.jpc"
  format="xml" tofile="result.xml">
  <sourcepath path="./src"/>
  <reference>
    <classpath path="./bin/classes"/>
    <filters>
      <include class="com.mycompany.*"/>
      <exclude class="com.mycompany.MyClass" method="test*"/>
    </filters>
  </reference>
</jpcovreport>
```

would generate the report of the file merge.jpc and write it to result.xml using the source path src. As well, it will modify the result.xml by analyzing all classes in the ./bin/classes that are port of the package com.mycompany except the method that start by test from the class MyClass.

Recommendation

If you generate your main code and your testcases in a separate directory, say bin/classes and test/classes. You should mostly end up with a reference such as:

```
<reference>
    <classpath path="./bin/classes"/>
</reference>
```

With such a reference, your XML report will be cleaned up against parasite classes from your testcases (that as a common practice, generally match the exact package structure of the class you exercise).

HTML reports

You will find in Ant etc directory a stylesheet called coverage-frames.xsl. This file can be used to generate a framed report a la javadoc similar to the one for JUnit. It needs either Xalan 1.2.2 or Xalan 2.x.

Xalan 1.2.2 (you must have xalan.jar and bsf.jar in your classpath)

```
<style processor="xalan" in="./reports/xml/results.xml" out="./reports/html/dummy.file"
 style="${ant.home}/etc/coverage-frames.xsl">
    <param name="output.dir" expression="'${basedir}/reports/html'"/>
</style>
```

Xalan 2.x (note the parameter without single quote)

```
<style processor="trax" in="./reports/xml/results.xml" out="./reports/html/dummy.file"
 style="${ant.home}/etc/coverage-frames.xsl">
  <param name="output.dir" expression="${basedir}/reports/html"/>
</style>
```

5.3.22 JUnit

5.3.23 Description

This task runs tests from the JUnit testing framework. The latest version of the framework can be found at http://www.junit.org. This task has been tested with JUnit 3.0 up to JUnit 3.8.1; it won't work with versions prior to JUnit 3.0.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

Note: You must have junit.jar and the class files for the `<junit>` task in the same classpath. You can do one of:

1. Put both junit.jar and the optional tasks jar file in ANT_HOME/lib.
2. Do not put either in ANT_HOME/lib, and instead include their locations in your CLASSPATH environment variable.

3. Do neither of the above, and instead, specify their locations using a `<classpath>` element in the build file. See the FAQ for details.

Tests are defined by nested test or batchtest tags (see nested elements).

Parameters

Attribute	Description	Required
printsummary	Print one-line statistics for each testcase. Can take the values on, off, and withOutAndErr. withOutAndErr is the same as on but also includes the output of the test as written to System.out and System.err.	No; default is off.
fork	Run the tests in a separate VM.	No; default is off.
haltonerror	Stop the build process if an error occurs during the test run.	No; default is off.
errorproperty	The name of a property to set in the event of an error.	No
haltonfailure	Stop the build process if a test fails (errors are considered failures as well).	No; default is off.
failureproperty	The name of a property to set in the event of a failure (errors are considered failures as well).	No.
filtertrace	Filter out Junit and Ant stack frames from error and failure stack traces.	No; default is on.
timeout	Cancel the individual tests if they don't finish in the given time (measured in milliseconds). Ignored if fork is disabled.	No
maxmemory	Maximum amount of memory to allocate to the forked VM. Ignored if fork is disabled.	No
jvm	The command used to invoke the Java Virtual Machine, default is 'java'. The command is resolved by java.lang.Runtime.exec(). Ignored if fork is disabled.	No; default is java.
dir	The directory in which to invoke the VM. Ignored if fork is disabled.	No
newenvironment	Do not propagate the old environment when new environment variables are specified. Ignored if fork is disabled.	No; default is false.
includeantruntime	Implicitly add the Ant classes required to run the tests and JUnit to the classpath in forked mode.	No; default is true.
showoutput	Send any output generated by tests to Ant's logging system as well as to the formatters. By default only the formatters receive the output.	No

By using the errorproperty and failureproperty attributes, it is possible to perform setup work (such as starting an external server), execute the test, clean up, and still fail the build in the event of a failure.

The filtertrace attribute condenses error and failure stack traces before reporting them. It works with both the plain and XML formatters. It filters out any lines that begin with the following string patterns:

```
"junit.framework.TestCase"
"junit.framework.TestResult"
"junit.framework.TestSuite"
"junit.framework.Assert."
"junit.swingui.TestRunner"
"junit.awtui.TestRunner"
"junit.textui.TestRunner"
"java.lang.reflect.Method.invoke("
"org.apache.tools.ant."
```

Nested Elements

The `<junit>` task supports a nested `<classpath>` element that represents a PATH like structure.

jvmarg

If fork is enabled, additional parameters may be passed to the new VM via nested `<jvmarg>` elements. For example:

```
<junit fork="yes">
  <jvmarg value="-Djava.compiler=NONE"/>
  ...
</junit>
```

would run the test in a VM without JIT.

`<jvmarg>` allows all attributes described in Command-line Arguments.

sysproperty

Use nested `<sysproperty>` elements to specify system properties required by the class. These properties will be made available to the VM during the execution of the test (either ANT's VM or the forked VM, if fork is enabled). The attributes for this element are the same as for environment variables.

```
<junit fork="no">
  <sysproperty key="basedir" value="${basedir}"/>
  ...
</junit>
```

would run the test in ANT's VM and make the basedir property available to the test.

env

It is possible to specify environment variables to pass to the forked VM via nested `<env>` elements. For a description of the `<env>` element's attributes, see the description in the exec task.

Settings will be ignored if fork is disabled.

formatter

The results of the tests can be printed in different formats. Output will always be sent to a file, unless you set the usefile attribute to false. The name of the file is determined by the name of the test and can be set by the outfile attribute of `<test>`.

There are three predefined formatters - one prints the test results in XML format, the other emits plain text. The formatter named brief will only print detailed information for testcases that failed, while plain gives a little statistics line for all test cases. Custom formatters that need to implement org.apache.tools.ant.taskdefs.optional.junit.JUnitResultFormatter can be specified.

If you use the XML formatter, it may not include the same output that your tests have written as some characters are illegal in XML documents and will be dropped.

Attribute	Description	Required
type	Use a predefined formatter (either xml, plain, or brief).	Exactly one of these.
classname	Name of a custom formatter class.	
extension	Extension to append to the output filename.	Yes, if classname has been used.
usefile	Boolean that determines whether output should be sent to a file.	No; default is true.

test

Defines a single test class.

Attribute	Description	Required
name	Name of the test class.	Yes
fork	Run the tests in a separate VM. Overrides value set in `<junit>`.	No
haltonerror	Stop the build process if an error occurs during the test run. Overrides value set in `<junit>`.	No
errorproperty	The name of a property to set in the event of an error. Overrides value set in `<junit>`.	No
haltonfailure	Stop the build process if a test fails (errors are considered failures as well). Overrides value set in `<junit>`.	No
failureproperty	The name of a property to set in the event of a failure (errors are considered failures as well). Overrides value set in `<junit>`.	No
filtertrace	Filter out Junit and Ant stack frames from error and failure stack traces. Overrides value set in `<junit>`.	No; default is on.
todir	Directory to write the reports to.	No; default is the current directory.
outfile	Base name of the test result. The full filename is determined by this attribute and the extension of formatter.	No; default is TEST-name, where name is the name of the test specified in the name attribute.
if	Only run test if the named property is set.	No
unless	Only run test if the named property is not set.	No

Tests can define their own formatters via nested `<formatter>` elements.

batchtest

Define a number of tests based on pattern matching.

batchtest collects the included files from any number of nested `<fileset>`s. It then generates a test class name for each file that ends in .java or .class.

Attribute	Description	Required
fork	Run the tests in a separate VM. Overrides value set in `<junit>`.	No
haltonerror	Stop the build process if an error occurs during the test run. Overrides value set in `<junit>`.	No
errorproperty	The name of a property to set in the event of an error. Overrides value set in `<junit>`.	No
haltonfailure	Stop the build process if a test fails (errors are considered failures as well). Overrides value set in `<junit>`.	No
failureproperty	The name of a property to set in the event of a failure (errors are considered failures as well). Overrides value set in `<junit>`	No
filtertrace	Filter out Junit and Ant stack frames from error and failure stack traces. Overrides value set in `<junit>`.	No; default is on.
todir	Directory to write the reports to.	No; default is the current directory.
if	Only run tests if the named property is set.	No
unless	Only run tests if the named property is not set.	No

Batchtests can define their own formatters via nested `<formatter>` elements.
bf Examples

```
<junit>
  <test name="my.test.TestCase"/>
</junit>
```

Runs the test defined in my.test.TestCase in the same VM. No output will be generated unless the test fails.

```
<junit printsummary="yes" fork="yes" haltonfailure="yes">
  <formatter type="plain"/>
  <test name="my.test.TestCase"/>
</junit>
```

Runs the test defined in my.test.TestCase in a separate VM. At the end of the test, a one-line summary will be printed. A detailed report of the test can be found in TEST-my.test.TestCase.txt. The build process will be stopped if the test fails.

```
<junit printsummary="yes" haltonfailure="yes">
  <classpath>
    <pathelement location="${build.tests}"/>
    <pathelement path="${java.class.path}"/>
  </classpath>
  <formatter type="plain"/>

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

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

Runs my.test.TestCase in the same VM, ignoring the given CLASSPATH; only a warning is printed if this test fails. In addition to the plain text test results, for this test a XML result will be output to result.xml. Then, for each matching file in the directory defined for ${src.tests} a test is run in a separate VM. If a test fails, the build process is aborted. Results are collected in files named TEST-name.txt and written to ${reports.tests}.

5.3.24 JUnitReport

Merge the individual XML files generated by the JUnit task and eventually apply a stylesheet on the resulting merged document to provide a browsable report of the testcases results.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

Requirements

The task needs Xalan 2.x; although Xalan 1.2.2 does work, but as Xalan1 is not supported, we do not recommend this.

If you do you use Xalan 1.2.2 you will need a compatible (older) version of Xerces. as well as BSF(bsf.jar). Again, using Xalan 2 is simpler and supported.

Parameters

Attribute	Description	Required
tofile	The name of the XML file that will aggregate all individual XML testsuite previously generated by the JUnit task.	No. Default to TESTS-TestSuites.xml
todir	The directory where should be written the file resulting from the individual XML testsuite aggregation.	No. Default to current directory

Nested Elements

fileset

junitreport collects individual xml files generated by the JUnit task using the nested `<FileSet>` element.

report

Generate a browsable report based on the document created by the merge.

Parameters

Attribute	Description	Required
format	The format of the generated report. Must be "noframes" or "frames".	No, default to "frames"
styledir	The directory where the stylesheets are defined. They must be conforming to the following conventions: • frames format: the stylesheet must be named junit-frames.xsl. • noframes format: the stylesheet must be named junit-noframes.xsl.	No. Default to embedded stylesheets.
todir	The directory where the files resulting from the transformation should be written to.	No. Default to current directory

Example of report

```
<junitreport todir="./reports">
  <fileset dir="./reports">
    <include name="TEST-*.xml"/>
  </fileset>
  <report format="frames" todir="./report/html"/>
</junitreport>
```

would generate a TESTS-TestSuites.xml file in the directory reports and generate the default framed report in the directory report/html.

5.3.25 Metamata Metrics

`MMetrics`

```
    * Stephane Bailliez (sbailliez@imediation.com)
```

Requirements

This task requires Metamata Development environment 2.0/Webgain Quality Analyzer 2.0. An evaluation version is available at Webgain. (Though you will not be able to use Metrics from the command line if you do not have a registered version). You also need a TRaX compliant processor(such as Xalan 2.x) via JAXP 1.1

Description

Invokes the Metamata Metrics / WebGain Quality Analyzer source code analyzer on a set of Java files.

mmetrics will compute the metrics of a set of Java files and write the results to an XML file. As a convenience, a stylesheet is given in etc directory, so that an HTML report can be generated from the XML file.

Parameters

Attribute	Description	Required
metamatahome	The home directory containing the Metamata distribution.	Yes
tofile	The XML were the resulting metrics will be written to.	Yes
granularity	Metrics granularity of the source files. Must be either files (compilation-units), types (types and compilation-units) or methods (methods, types and compilation-units).	Yes
maxmemory	Set the maximum memory for the JVM. this is a convenient way to set the -mx or -Xmx argument.	No

Nested elements

For specifying the source code to analyze, you can either use a path or fileset elements (though a single path element is preferred, see note below).

jvmarg

Additional parameters may be passed to the VM via nested `<jvmarg>` attributes. `<jvmarg>` allows all attributes described in Command line arguments.

classpath

Sets class path (also source path unless one explicitly set). Overrides METAPATH/CLASSPATH environment variables. The classpath element represents a PATH like structure.

sourcepath

Sets source path. Overrides the SOURCEPATH environment variable. The sourcepath element represents a PATH like structure.

path

Sets the list of directories to analyze the code for metrics.;It represents a PATH structure.

fileset

Sets a set of files to analyze for metrics.source It represents a FILESET structure.

Note: For the sake of readability, it is highly recommended to analyze for a single unique directory instead than using filesets or several directories. Otherwise there will be multiple metrics outputs without any way to know what metrics refers to what source. Chance are also that the XML handler that does some heuristic will be confused by the different outputs.

Example

```
<mmetrics tofile="mmetrics.xml"
        metamatahome="c:/metamata"
        granularity="methods">
  <classpath>
    <pathelement location="c:/metamata/examples/metricsexamples"/>
  </classpath>
  <sourcepath>
    <pathelement location="c:/metamata/examples/metricsexamples"/>
  </sourcepath>
  <path>
    <pathelement location="c:/metamata/examples/metricsexamples"/>
  </path>
</mmetrics>
```

This invokes Metamata Metrics installed in c:/metamata on the metrics example. (Note that here, classpath and sourcepath are not normally not needed) Generating a report As a convenience, there is an XSL file(mmetrics-frames.xsl) that allows you to generate a full framed HTML report of the metrics. You can find it in the etc directory of Ant. As it uses the Xalan redirect extensions, you will need Xalan and Xerces to run it. The stylesheet takes an output.dir parameter (otherwise it will be generated in the current directory), it can be run in Ant as follows:

```
<style in=java "${metrics.xml}" style="mmetrics-frames.xsl" out="null.tmp">
    <param name="output.dir" expression="${report.dir}"/>
</style>
```

5.3.26 Metamata Audit

`MAudit`

```
    * Stephane Bailliez (sbailliez@imediation.com)
```

Requirements

This task requires Metamata Development environment 2.0/Webgain Quality Analyzer 2.0. An evaluation version is available at Webgain. , Jakarta Oro and a XML parser (via JAXP).

Description

Invokes the Metamata Audit/ Webgain Quality Analyzer on a set of Java files.

maudit performs static analysis of the Java source code and byte code files to find and report errors of style and potential problems related to performance, maintenance and robustness. . As a convenience, a stylesheet is given in etc directory, so that an HTML report can be generated from the XML file.

Parameters

Attribute	Description	Required
tofile	The XML file to which the Audit result should be written to.	Yes
metamatahome	The home directory containing the Metamata distribution.	Yes
fix	Automatically fix certain errors (those marked as fixable in the manual).	No.Default to false.
list	Creates listing file for each audited file. A .maudit file will be generated in the same location as the source file.	No. Default to false.
unused	Finds declarations unused in search paths. It will look for unused global declarations in the source code within a use domain specified by the searchpath element.	No. Default to false.
maxmemory	Set the maximum memory for the JVM. this is a convenient way to set the -mx or -Xmx argument.	No

Nested elements

jvmarg

Additional parameters may be passed to the VM via nested `<jvmarg>` attributes. `<jvmarg>` allows all attributes described in Command line arguments.

You can avoid using the `<jvmarg>` by adding these empty entries to metamata.properties located at ${metamata.home}/bin

```
metamata.classpath=
metamata.sourcepath=
metamata.baseclasspath=
```

classpath

Sets class path (also source path unless one explicitly set). Overrides METAPATH/CLASSPATH environment variables. The classpath element represents a PATH like structure.

sourcepath

Sets source path. Overrides the SOURCEPATH environment variable. The sourcepath element represents a PATH like structure.

sourcepath

Sets the search path to use as the use domain when looking for unused global declarations. The searchpath element represents a PATH like structure.

fileset

Sets the Java files to audit via a FILESET structure. Whatever the filter is, only the files that ends with .java will be included for processing. Note that the base directory used for the fileset MUST be the root of the source files otherwise package names deduced from the file path will be incorrect.

Example

```
<maudit tofile="c:/metamata/examples/auditexamples/audit.xml"
        metamatahome="c:/metamata" fix="yes">
  <classpath>
    <pathelement location="c:/metamata/examples/auditexamples"/>
  </classpath>
  <sourcepath>
    <pathelement location="c:/metamata/examples/auditexamples"/>
  </sourcepath>
  <fileset dir="c:/metamata/examples/auditexamples">
      <include name="*.java"/>
  </fileset>
</maudit>
```

This invokes Metamata Audit installed in c:/metamata on the audit examples and fix automatically the fixable errors.

Generating a report

As a convenience, there is an XSL file(mmetrics-frames.xsl) that allows you to generate a full framed HTML report of the metrics. You can find it in the etc directory of Ant. As it uses the Xalan redirect extensions, you will need Xalan and Xerces to run it. The stylesheet takes an output.dir parameter (otherwise it will be generated in the current directory), it can be run in Ant as follows:

```
<style in=java "${audit.xml}" style="maudit-frames.xsl" out="null.tmp">
```

```
    <param name="output.dir" expression="${report.dir}"/>
</style>
```

5.3.27 MimeMail

Deprecated

This task has been deprecated. Use the mail task instead.

Description

Sends SMTP mail with MIME attachments. JavaMail and Java Activation Framework are required for this task.

Multiple files can be attached using FileSets.

Parameters

Attribute	Description	Required
message	The message body	No, but only one of of ’message’ or ’messageFile’ may be specified. If not specified, a fileset must be provided.
messageFile	A filename to read and used as the message body	
messageMimeType	MIME type to use for ’message’ or ’messageFile’ when attached.	No, defaults to ”text/plain”
tolist	Comma-separated list of To: recipients	Yes, at least one of ’tolist’, ’cclist’, or ’bcclist’ must be specified.
cclist	Comma-separated list of CC: recipients	
bcclist	Comma-separated list of BCC: recipients	
mailhost	Host name of the mail server.	No, default to ”localhost”
subject	Email subject line.	No
from	Email address of sender.	Yes
failonerror	Stop the build process if an error occurs sending the e-mail.	No, default to ”true”

Examples

Send a single HTML file as the body of a message

```
    <mimemail messageMimeType="text/html" messageFile="overview-summary.html"
        tolist="you" subject="JUnit Test Results: ${TODAY}" from="me"/>
```

Sends all files in a directory as attachments

```
<mimemail message="See attached files"
 tolist="you" subject="Attachments" from="me">
   <fileset dir=".">
       <include name="dist/*.*"/>
   </fileset>
</mimemail>
```

5.3.28 MParse

`MParse`

`by`

```
* Stephane Bailliez (sbailliez@imediation.com)
```

Requirements

This task requires Metamata Development environment 2.0 freely available at Metamata.

Description

Invokes the Metamata MParse compiler compiler on a grammar file.

To use the mparse task, set the target attribute to the name of the grammar file to process. You also need to specify the directory containing the Metamata installation using the metamatahome attribute, so that Ant can find the MParse classes.

This task only invokes MParse if the grammar file is newer than the generated Java files. MParse assumes that the Java class name of the generated parser is the same as the name of the grammar file, less the .jj extension.

For additional information about MParse, please consult the online manual available here (PDF)

Parameters

Attribute	Description	Required
target	The .jj grammar file to process. It will only be processed if the grammar is newer than the corresponding .java file.	Yes
metamatahome	The home directory containing the Metamata distribution.	Yes
verbose	Enable all messages	No
debugparser	Enables parser debugging	No
debugscanner	Enables scanner debugging	No
cleanup	Remove the intermediate Sun JavaCC file created during the transformation of the grammar file.	No. Default to false
maxmemory	Set the maximum memory for the JVM. this is a convenient way to set the -mx or -Xmx argument.	No

Nested elements

jvmarg

Additional parameters may be passed to the VM via nested `<jvmarg>` attributes. `<jvmarg>` allows all attributes described in Command line arguments.

classpath

Sets class path (also source path unless one explicitly set). Overrides METAPATH/CLASSPATH environment variables. The classpath element represents a PATH like structure.

sourcepath

Sets source path. Overrides the SOURCEPATH environment variable. The sourcepath element represents a PATH like structure.

Example

```
<mparse target="c:/metamata/examples/parseexamples/javagrammars/singlefile/JavaParser.jj"
        metamatahome="c:/metamata" cleanup="true">
  <classpath>
    <pathelement location="c:/metamata/examples/"/>
  </classpath>
</mparse>
```

This invokes Metamata MParse installed in c:/metamata on one of the grammar file example (JavaParser.jj) and cleans up the intermediate Sun JavaCC file.

5.3.29 Native2Ascii

Description

Converts files from native encodings to ASCII with escaped Unicode. A common usage is to convert source files maintained in a native operating system encoding, to ASCII prior to compilation.

Files in the directory src are converted from a native encoding to ASCII. By default, all files in the directory are converted. However, conversion may be limited to selected files using includes and excludes attributes. For more information on file matching patterns, see the section on directory based tasks. If no encoding is specified, the default encoding for the JVM is used. If ext is specified, then output files are renamed to use it as a new extension. More sophisticated file name translations can be achieved using a nested `<mapper>` element. By default an identity mapper will be used. If dest and src point to the same directory, the ext attribute or a nested `<mapper>` is required.

This task forms an implicit File Set, and supports all attributes of `<fileset>` (dir becomes src) as well as nested `<include>`, `<exclude>`, and `<patternset>` elements.

Attribute	Description	Required
reverse	Reverse the sense of the conversion, i.e. convert from ASCII to native	No
encoding	The native encoding the files are in (default is the default encoding for the JVM)	No
src	The directory to find files in (default is basedir)	No
dest	The directory to output file to	Yes
ext	File extension to use in renaming output files	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No

Examples

```
<native2ascii encoding="EUCJIS" src="srcdir" dest="srcdir"
   includes="**/*.eucjis" ext=".java"/>
```

Converts all files in the directory srcdir ending in .eucjis from the EUCJIS encoding to ASCII and renames them to end in .java.

```
<native2ascii encoding="EUCJIS" src="native/japanese" dest="src"
   includes="**/*.java"/>
```

Converts all the files ending in .java in the directory native/japanese to ASCII, placing the results in the directory src. The names of the files remain the same.

5.3.30 NetRexxC

Description

Compiles a NetRexx source tree within the running (Ant) VM.

The source and destination directory will be recursively scanned for NetRexx source files to compile. Only NetRexx files that have no corresponding class file or where the class file is older than the java file will be compiled.

Files in the source tree are copied to the destination directory, allowing support files to be located properly in the classpath. The source files are copied because the NetRexx compiler cannot produce class files in a specific directory via parameters

The directory structure of the source tree should follow the package hierarchy.

It is possible to refine the set of files that are being compiled/copied. This can be done with the includes, includesfile, excludes, excludesfile and defaultexcludes attributes. With the includes or includesfile attribute you specify the files you want to have included by using patterns. The exclude or excludesfile attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the defaultexcludes attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of ¡fileset¿ (dir becomes srcdir) as well as the nested ¡include¿, ¡exclude¿ and ¡patternset¿ elements.

All properties except classpath, srcdir and destDir are also available as properties in the form ant.netrexxc.attributename, eg.

```
<property name="ant.netrexxc.verbose" value="noverbose"/>
```

or from the command line as ant -Dant.netrexxc.verbose=noverbose ...

Parameters

Attribute	Description	Required
binary	Whether literals are treated as the java binary type rather than the NetRexx types	No
classpath	The classpath to use during compilation	No
comments	Whether comments are passed through to the generated java source	No
compact	Whether error messages come out in compact or verbose format. Default is the compact format.	No
compile	Whether the NetRexx compiler should compile the generated java code	No
console	Whether or not messages should be displayed on the 'console'	No
crossref	Whether variable cross references are generated	No
decimal	Whether decimal arithmetic should be used for the NetRexx code	No
defaultexcludes	indicates whether default excludes should be used or not ("yes"/"no"). Default excludes are used when omitted.	No
destDir	the destination directory into which the NetRexx source files should be copied and then compiled	Yes
diag	Whether diagnostic information about the compile is generated	No
excludes	comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file. Each line of this file is taken to be an exclude pattern	No
explicit	Whether variables must be declared explicitly before use	No
format	Whether the generated java code is formatted nicely or left to match NetRexx line numbers for call stack debugging	No
includes	comma- or space-separated list of patterns of files that must be included. All files are included when omitted.	No
includesfile	the name of a file. Each line of this file is taken to be an include pattern	No
java	Whether the generated java code is produced	No
keep	Sets whether the generated java source file should be kept after compilation. The generated files will have an extension of .java.keep, not .java	No
logo	Whether the compiler text logo is displayed when compiling	No
replace	Whether the generated .java file should be replaced when compiling	No
savelog	Whether the compiler messages will be written to NetRexxC.log as well as to the console	No

Attribute	Description	Required
sourcedir	Tells the NetRexx compiler to store the class files in the same directory as the source files. The alternative is the working directory	No
srcDir	Set the source dir to find the source NetRexx files	Yes
strictargs	Tells the NetRexx compiler that method calls always need parentheses, even if no arguments are needed, e.g. aStringVar.getBytes vs. aStringVar.getBytes()	No
strictassign	Tells the NetRexx compile that assignments must match exactly on type	No
strictcase	Specifies whether the NetRexx compiler should be case sensitive or not	No
strictimport	Whether classes need to be imported explicitly using an import statement. By default the NetRexx compiler will import certain packages automatically	No
strictprops	Whether local properties need to be qualified explicitly using this	No
strictsignal	Whether the compiler should force catching of exceptions by explicitly named types	No
symbols	Whether debug symbols should be generated into the class file	No
time	Asks the NetRexx compiler to print compilation times to the console	No
trace	Turns on or off tracing and directs the resultant trace output	No
utf8	Tells the NetRexx compiler that the source is in UTF8	No
verbose	Whether lots of warnings and error messages should be generated	No
suppressMethodArgumentNotUsed	Tells whether we should filter out the &Method argument not used& messages in strictargs mode.	no
suppressPrivatePropertyNotUsed	Tells whether we should filter out the &Private Property defined, but not used& messages in strictargs mode.	no
suppressVariableNotUsed	Tells whether we should filter out the &Variable set but not used& messages in strictargs mode. Please be careful with this one, as you can hide errors behind it!	no

Attribute	Description	Required
suppressExceptionNotSignalled	Tells whether we should filter out the &Exception is declared, but not signalled within the method& messages in strictsignal mode.	no
suppressDeprecation	Tells wether we should filter out any deprecation-messages of the compiler out.	no

Examples

```
<netrexxc srcDir="/source/project" includes="vnr/util/*"
 destDir="/source/project/build" classpath="/source/project2/proj.jar"
 comments="true" crossref="false" replace="true" keep="true"/>
```

5.3.31 Perforce Tasks

```
Perforce Tasks User Manual

by

    * Les Hughes (leslie.hughes@rubus.com)
    * Kirk Wylie (kirk@radik.com)

Version 1.1 - 2001/01/09
```

These tasks provide an interface to the Perforce SCM.

The org.apache.tools.ant.taskdefs.optional.perforce package consists of a simple framework to support p4 functionality as well as some Ant tasks encapsulating frequently used (by me :-) p4 commands. However, the addition of new p4 commands is a pretty simple task (see the source). Although it is possible to use these commands on the desktop, they were primarily intended to be used by automated build systems.

Note: These tasks require the oro 2.0.XXX regular expression package. Simply download this package and copy the jakarta-oro-2.0.XXX.jar file into Ant's lib directory. You will also need the Perforce client executable (p4 or p4.exe but not p4win.exe) in your path.

The Tasks

- P4Sync Synchronise a workspace to a depot
- P4Change Request a new changelist from the Perforce server
- P4Edit Open files for edit (checkout)
- P4Submit Submit a changelist to the Perforce server (checkin)
- P4Have List current files in client view, useful for reporting
- P4Label Create a label reflecting files in the current workspace

- P4Counter Obtain or set the value of a counter
- P4Reopen Move files between changelists
- P4Revert Revert files
- P4Add Add files

General P4 Properties

Each p4 task requires a number of settings, either through build-wide properties, individual attributes or environment variables. These are

Property	Attribute	Env Var	Description	Default
p4.port	port	P4PORT	The p4d server and port to connect to	perforce:1666
p4.client	client	P4CLIENT	The p4 client spec to use	The logged in username
p4.user	user	P4USER	The p4 username	The logged in username
–	view	–	The client, branch or label view to operate upon. See the p4 user guide for more info.	`//...`

Your local installation of Perforce may require other settings (e.g. P4PASSWD, P4CONFIG). At the moment, these can only be set outside of Ant, as environment variables.

Additionally, you may also specify the following attributes:

Attribute	Description	Required
failonerror	Specifies whether to stop the build (true—yes—on) or keep going (false—no—off) if an error is returned from the p4 command.	No; defaults to true.

Examples

Setting in the environment:- (Unix csh)

```
setenv P4PORT myperforcebox:1666
```

(Unix sh et al)

```
P4USER=myp4userid; export P4USER
```

Using build properties:-

```
<property name="p4.client" value="nightlybuild"/>
```

Using task attributes:-

```
<p4Whatever
        port="myserver:1666"
        client="smoketest"
        user="smoketestdude"
        .
        .
        .
/>
```

For more information regarding the underlying 'p4' commands you are referred to the Perforce Command Reference available from the Perforce website.

Taskdefs

Standard taskdefs (for you to copy'n'paste) – normally this is done automatically if you install this optional task.

```
<taskdef name="p4sync" classname="org.apache.tools.ant.taskdefs.optional.perfor
<taskdef name="p4change" classname="org.apache.tools.ant.taskdefs.optional.perf
<taskdef name="p4edit" classname="org.apache.tools.ant.taskdefs.optional.perfor
<taskdef name="p4submit" classname="org.apache.tools.ant.taskdefs.optional.perf
<taskdef name="p4have" classname="org.apache.tools.ant.taskdefs.optional.perfor
<taskdef name="p4label" classname="org.apache.tools.ant.taskdefs.optional.perfo
<taskdef name="p4counter" classname="org.apache.tools.ant.taskdefs.optional.per
<taskdef name="p4reopen" classname="org.apache.tools.ant.taskdefs.optional.perf
<taskdef name="p4revert" classname="org.apache.tools.ant.taskdefs.optional.perf
<taskdef name="p4add" classname="org.apache.tools.ant.taskdefs.optional.perforc
```

Task Descriptions

P4Sync

Description

Synchronize the current workspace with the depot.

Parameters

Attribute	Description	Required
force	force a refresh of files, if this attribute has been set.	no - if ommitted, it will be off, otherwise a refresh will be forced.
label	sync client to label	no

Examples

```
<p4sync label="nightlybuild-0.0123" force="foo"/>
<p4sync view="//depot/projects/projectfoo/main/src/..."/>
```

P4Change

Description

Request a new changelist from the Perforce server. This task sets the ${p4.change} property which can then be passed to P4Submit, P4Edit, or P4Add.

Parameters

Attribute	Description	Required
description	Description for ChangeList. If none specified, it will default to "AutoSubmit By Ant"	No.

Examples

```
<p4change description="Change Build Number in Script">
```

P4Edit

Description:

Open file(s) for edit. P4Change should be used to obtain a new changelist for P4Edit as, although P4Edit can open files to the default change, P4Submit cannot yet submit it.

Parameters

Attribute	Description	Required
view	The filespec to request to edit	Yes
change	An existing changelist number to assign files to.	No, but see above.

Examples

```
<p4edit
    view="//depot/projects/projectfoo/main/src/Blah.java..."
    change="${p4.change}"/>
```

P4Submit

Description:

Submit a changelist, usually obtained from P4Change.

Parameters

Attribute	Description	Required
change	The changelist number to submit	Yes

Examples

```
<p4submit change="${p4.change}"/>
```

P4Have

Description:

List handy file info reflecting the current client contents.

Parameters

Attribute	Description	Required
	None	– –

Examples

```
<p4have/>
```

P4Label

Description:

Create a new label and set contents to reflect current client file revisions.

Parameters Attribute Description Required name The name of the label Yes view client view to use for label No desc Label Description No lock Lock the label once created. No Examples

```
<p4label
        name="NightlyBuild:${DSTAMP}:${TSTAMP}"
        desc="Auto Nightly Build"
        lock="locked"
/>
```

P4Counter

Description:

Obtain or set the value of a counter. When used in its base form (where only the counter name is provided), the counter value will be printed to the output stream. When the value is provided, the counter will be set to the value provided. When a property name is provided, the property will be filled with the value of the counter. You may not specify to both get and set the value of the counter in the same Task.

The user performing this task must have Perforce ”review” permissions as defined by Perforce protections in order for this task to succeed.

Parameters Attribute Description Required name The name of the counter Yes value The new value for the counter No property The property to be set with the value of the counter No

Examples

Print the value of the counter named ”last-clean-build” to the output stream:

```
<p4counter name="last-clean-build"/>
```

Set the value of the counter based on the value of the ”TSTAMP” property:

```
<p4counter name="last-clean-build" value="${TSTAMP}"/>
```

Set the value of the ”p4.last.clean.build” property to the current value of the ”last-clean-build” counter:

```
<p4counter name="last-clean-build" property="${p4.last.clean.build}"/>
```

P4Reopen Description:

Move (or reopen in Perforce speak) checkout files between changelists. Parameters Attribute Description Required tochange The changelist to move files to. Yes

Examples

Move all open files to the default changelist

```
<p4reopen view="//..." tochange="default"/>
```

Create a new changelist then reopen into it, any files from the view //projects/foo/main/...

```
<p4change description="Move files out of the way"/>
<p4reopen view="//projects/foo/main/..." tochange="${p4.change}"/>
```

P4Revert Description:

Reverts files. Parameters Attribute Description Required change The changelist to revert. No revertOnlyUnchanged Revert only unchanged files (p4 revert -a) No Examples Revert everything!

```
<p4revert view="//..."/>
```

Revert any unchanged files in the default change

```
<p4revert change="default" revertonlyunchanged="true"/>
```

P4Add Description:

Adds files specified in nested fileset children. Parameters Attribute Description Required commandlength A positive integer specifying the maximum length of the commandline when calling Perforce to add the files. Defaults to 450, higher values mean faster execution, but also possible failures. No changelist If specified the open files are associated with the specified pending changelist number; otherwise the open files are associated with the default changelist. No Examples Require a changelist, add all java files starting from a directory, and submit

```
<p4change/>
<p4add commandlength="20000" changelist="${p4.change}">
    <fileset dir="../dir/src/" includes="**/*.java"/>
<p4add>
<p4submit change="${p4.change}"/>
```

Change History Sept 2000 – Internal Release within Rubus Nov 2000 V1.0 Initial Release donated to ASF :-) Jan 2001 V1.1 fixed cross platform (NT/Unix) bug refactored p4 output handling code refactored exec'ing code

5.3.32 PropertyFile

PropertyFile

by

* Thomas Christen (chr@active.ch) * Jeremy Mawson (jem@loftinspace.com/au)

Table of Contents

* Introduction * PropertyFile Task * Entry Task

Introduction

Ant provides an optional task for editing property files. This is very useful when wanting to make unattended modifications to configuration files for application servers and applications. Currently, the task maintains a working property file with the ability to add properties or make changes to existing ones. However, any comments are lost.

PropertyFile Task Parameters Attribute Description Required file Location of the property file to be edited Yes comment Header for the file itself no Parameters specified as nested elements Entry

Use nested ¡entry¿ elements to specify actual modifications to the property file itself. Attribute Description Required key Name of the property name/value pair Yes value Value to set (=), to add (+) or subtract (-) At least one must be specified default Initial value to set for a property if it is not already defined in the property file. For type date, an additional keyword is allowed: "now" type Regard the value as : int, date or string (default) No operation "+" or "=" (default) for all datatypes "-" (for date and int only). No pattern For int and date type only. If present, Values will be parsed and formatted accordingly. No unit The unit of the value to be applied to date +/- operations. Valid Values are:

* millisecond * second * minute * hour * day (default) * week * month * year

This only applies to date types using a +/- operation. No

The rules used when setting a property value are shown below. The operation occurs after these rules are considered.

* If only value is specified, the property is set to it regardless of its previous value. * If only default is specified and the property previously existed in the property file, it is unchanged. * If only default is specified and the property did not exist in the property file, the property is set to default. * If value and default are both specified and the property previously existed in the property

file, the property is set to value. * If value and default are both specified and the property did not exist in the property file, the property is set to default.

Examples

The following changes the my.properties file. Assume my.properties look like:

```
# A comment
akey=novalue
```

After running, the file would now look like

```
#Thu Nov 02 23:41:47 EST 2000
akey=avalue
adate=2000/11/02 23\:41
anint=1
formated.int=0014
formated.date=028 17\:34
```

The slashes conform to the expectations of the Properties class. The file will be stored in a manner so that each character is examined and escaped if necessary. Note that the original comment is now lost. Please keep this in mind when running this task against heavily commented properties files. It may be best to have a commented version in the source tree, copy it to a deployment area, and then run the modifications on the copy. Future versions of PropertyFile will hopefully eliminate this shortcoming.

```
<propertyfile
    file="my.properties"
    comment"My properties">
  <entry  key="akey" value="avalue"/>
  <entry  key="adate" type="date" value="now"/>
  <entry  key="anint" type="int" operation="+"/>
  <entry  key="formated.int" type="int" default="0013" operation="+" pattern="0000"/>
  <entry  key="formated.date" type="date" value="now" pattern="DDD HH:mm"/>
</propertyfile>
```

To produce dates relative from today :

```
<propertyfile
    file="my.properties"
    comment="My properties">
  <entry  key="formated.date-1"
      type="date" default="now" pattern="DDD"
      operation="-" value="1"/>
  <entry  key="formated.tomorrow"
      type="date" default="now" pattern="DDD"
      operation="+" value="1"/>
</propertyfile>
```

Concatenation of strings :

```
<propertyfile
    file="my.properties"
    comment="My properties">
  <entry  key="progress" default="" operation="+" value="."/>
</propertyfile>
```

Each time called, a ”.” will be appended to ”progress”

5.3.33 Pvcs

Note: Before using this task, the user running ant must have access to the commands of PVCS (get and pcli) and must have access to the repository. Note that the way to specify the repository is platform dependent so use property to specify location of repository. by

* Thomas Christensen (tchristensen@nordija.com) * Don Jeffery (donj@apogeenet.com)

Version 1.1 - 2001/06/27

Problems with UNC pathnames and the use of () in paths are fixed and an updateonly argument introduced. Version 1.0 - 2001/01/31

Initial release. Table of Contents

* Introduction * Pvcs Task

Introduction The pvcs task allows the user of ant to extract the latest edition of the source code from a PVCS repository. PVCS is a version control system developed by Merant. This version has been tested agains PVCS version 6.5 and 6.6 under Windows and Solaris. Pvcs Task Description The pvcs task is set to point at a PVCS repository and optionally a project within that repository, and can from that specification get the latest version of the files contained by the repository. Parameters Attribute Description Required repository The location of the repository (see your PVCS manuals) Yes pvcsproject The project within the PVCS repository to extract files from (”/” is root project and that is default if this attribute isn’t specified) No label Only files marked with this label are extracted. No promotiongroup Only files within this promotion group are extracted. Using both the label and the promotiongroup tag will cause the files in the promotion group and with that label to be extracted. No force If set to yes all files that exists and are writable are overwritten. Default no causes the files that are writable to be ignored. This stops the PVCS command get to stop asking questions! No workspace By specifying a workspace, the files are extracted to that location. A PVCS workspace is a name for a location of the workfiles and isn’t as such the location itself. You define the location for a workspace using the PVCS GUI clients. If this isn’t specified the default workspace for the current user is used. No pvcsbin On some systems the PVCS executables pcli and get are not found in the PATH. In such cases this attribute should be set to the bin directory of the PVCS installation containing the executables mentioned before. If this attribute isn’t specified the tag expects the executables to be found using the PATH environment variable. No ignorereturncode If set to true the return value from executing the pvcs commands are ignored. No updateonly If set to true files are gotten only if newer than existing local files. No filenameformat The format of your folder names in a format suitable for java.text.MessageFormat. Defaults to 0-arc(1). Repositories where the archive extension is not -arc should set this. No linestart Used to parse the output of the pcli command. It defaults to "P:. The parser already knows about / and , this property is useful in cases where the repository is accessed on a Windows platform via a drive letter mapping. No Nested Elements pvcsproject element

pvcs supports a nested ¡pvcsproject¿ element, that represents a project within the PVCS repository to extract files from. By nesting multiple ¡pvcsproject¿ elements under the ¡pvcs¿ task, multiple projects can be specified. Parameters Attribute Description Required name The name of the pvcs project Yes Examples The following set-up extracts the latest version of the files in the pvcs repository.

```
<!-- ================================================================= -->
<!-- Get the latest version                                            -->
<!-- ================================================================= -->
<target name="getlatest">
  <pvcs repository="/mnt/pvcs" pvcsproject="/myprj"/>
</target>
```

Now run: ant getlatest

This will cause the following output to appear:

```
getlatest:
[pvcs] PVCS Version Manager (VMGUI) v6.6.10 (Build 870) for Windows NT/80x86
[pvcs] Copyright 1985-2000 MERANT. All rights reserved.
[pvcs] PVCS Version Manager (get) v6.6.10 (Build 870) for Windows NT/80x86
[pvcs] Copyright 1985-2000 MERANT. All rights reserved.
[pvcs] c:\myws\myprj\main.java <- C:\mypvcs\archives\myprj\main.java-arc
[pvcs] rev 1.1
[pvcs] c:\myws\myprj\apache\tool.java <- C:\mypvcs\archives\myprj\apache\tools.java-arc
[pvcs] rev 1.5

BUILD SUCCESSFUL

Total time: 19 seconds
```

This next example extracts the latest version of the files in the pvcs repository from two projects using nested ¡pvcsproject¿ elements.

```
<!-- ================================================================-->
<!-- Get latest from myprj and myprj2                                -->
<!-- ================================================================-->
<target name="getlatest2">
  <pvcs repository="/mnt/pvcs">
    <pvcsproject name="/myprj"/>
    <pvcsproject name="/myprj2"/>
  </pvcs>
</target>
```

Now run: ant getlatest2

This will cause the following output to appear:

```
getlatest2:
[pvcs] PVCS Version Manager (VMGUI) v6.6.10 (Build 870) for Windows NT/80x86
[pvcs] Copyright 1985-2000 MERANT.  All rights reserved.
[pvcs] PVCS Version Manager (get) v6.6.10 (Build 870) for Windows NT/80x86
[pvcs] Copyright 1985-2000 MERANT.  All rights reserved.
[pvcs] c:\myws\myprj\main.java <- C:\mypvcs\archives\myprj\main.java-arc
[pvcs] rev 1.1
[pvcs] c:\myws\myprj\apache\tool.java <- C:\mypvcs\archives\myprj\apache\tool.jav
[pvcs] rev 1.5
[pvcs] c:\myws\myprj2\apache\tool2.java <- C:\mypvcs\archives\myprj2\apache\tool2
[pvcs] rev 1.2

BUILD SUCCESSFUL

Total time: 22 seconds
```

5.3.34 RenameExtensions

Deprecated

This task has been deprecated. Use the move task with a glob mapper instead. Description

Renames files in the srcDir directory ending with the fromExtension string so that they end with the toExtension string. Files are only replaced if replace is true

See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns. This task forms an implicit FileSet and supports all attributes of ¡fileset¿ (dir becomes srcDir) as well as the nested ¡include¿, ¡exclude¿ and ¡patternset¿ elements. Parameters Attribute Description Required defaultexcludes indicates whether default excludes should be used or not (”yes”/”no”). Default excludes are used when omitted. No excludes comma- or space-separated list of patterns of files that must be excluded. No files (except default excludes) are excluded when omitted. No excludesfile the name of a file. Each line of this file is taken to be an exclude pattern No fromExtention The string that files must end in to be renamed Yes includes comma- or space-separated list of patterns of files that must be included. All files are included when omitted. No includesfile the name of a file. Each line of this file is taken to be an include pattern No replace Whether the file being renamed to should be replaced if it already exists No srcDir The starting directory for files to search in Yes toExtension The string that renamed files will end with on completion Yes

Examples

```
<renameext srcDir="/source/project1" includes="**" excludes="**/samples/*" fromExtension="
```

5.3.35 ReplaceRegExp

Description

ReplaceRegExp is a directory based task for replacing the occurrence of a given regular expression with a substitution pattern in a selected file or set of files.

Similar to regexp type mappers this task needs a supporting regular expression library and an implementation of org.apache.tools.ant.util.regexp.Regexp. Ant comes with implementations for the java.util.regex package of JDK 1.4, jakarta-regexp and jakarta-ORO, but you will still need the library itself.

There are cross-platform issues for matches related to line terminator. For example if you use $ to anchor your regular expression on the end of a line the results might be very different depending on both your platform and the regular expression library you use. It is 'highly recommended' that you test your pattern on both Unix and Windows platforms before you rely on it.

- Jakarta Oro defines a line terminator as '
  n' and is consistent with Perl.
- Jakarta RegExp uses a system-dependant line terminator.
- JDK 1.4 uses '
  n', '
  r
  n', '
  u0085', '
  u2028', '
  u2029' as a default but is configured in the wrapper to use only '
  n' (UNIX_LINE)

We strongly recommend that you use Jakarta Oro. Parameters Attribute Description Required file file for which the regular expression should be replaced. Yes if no nested ¡fileset¿ is used match The regular expression pattern to match in the file(s) Yes, if no nested ¡regexp¿ is used replace The substitution pattern to place in the file(s) in place of the regular expression. Yes, if no nested ¡substitution¿ is used flags The flags to use when matching the regular expression. For more information, consult the Perl5 syntax g : Global replacement. Replace all occurences found i : Case Insensitive. Do not consider case in the match m : Multiline. Treat the string as multiple lines of input, using "`caret`" and "$" as the start or end of any line, respectively, rather than start or end of string. s : Singleline. Treat the string as a single line of input, using "." to match any character, including a newline, which normally, it would not match. No byline Process the file(s) one line at a time, executing the replacement on one line at a time (true/false). This is useful if you want to only replace the first occurence of a regular expression on each line, which is not easy to do when processing the file as a whole. Defaults to false. No

Examples

```
<replaceregexp file="${src}/build.properties"
                         match="OldProperty=(.*)"
                         replace="NewProperty=\1"
                         byline="true"/>
```

replaces occurrences of the property name "OldProperty" with "NewProperty" in a properties file, preserving the existing value, in the file ${src}/build.properties

Parameters specified as nested elements

This task supports a nested FileSet element.

This task supports a nested Regexp element to specify the regular expression. You can use this element to refer to a previously defined regular expression datatype instance.

```
    <regexp id="id" pattern="expression"/>
    <regexp refid="id"/>
```

This task supports a nested Substitution element to specify the substitution pattern. You can use this element to refer to a previously defined substitution pattern datatype instance.

```
    <substitution id="id" pattern="expression"/>
    <substitution refid="id"/>
```

Examples

```
<replaceregexp byline="true">
  <regexp pattern="OldProperty=(.*)"/>
  <substitution expression="NewProperty=\1"/>
  <fileset dir=".">
   <includes="*.properties"/>
  </fileset>
 </replaceregexp>
```

replaces occurrences of the property name "OldProperty" with "NewProperty" in a properties file, preserving the existing value, in all files ending in .properties in the current directory

```
<replaceregexp match="\s+" replace=" " flags="g" byline="true">
    <fileset dir="${html.dir}" includes="**/*.html" />
</replaceregexp>
```

replaces all whitespaces (blanks, tabs, etc) by one blank remaining the line separator. So with input

```
<html>    <body>
<<TAB>><h1>    T E S T   </h1>  <<TAB>>
<<TAB>> </body></html>
```

would converted to

```
<html> <body>
 <h1> T E S T </h1> </body></html>
```

5.3.36 Rpm

Description

A basic task for invoking the rpm executable to build a Linux installation file. The task currently only works on Linux or other Unix platforms with rpm support.

Parameters

Attribute Description Required

specFile The name of the spec File to be used. Yes topDir this is the directory which will have the expected subdirectories, SPECS, SOURCES, BUILD, SRPMS. If this isn't specified, the baseDir value is used No cleanBuildDir This will remove the generated files in the BUILD directory. No removeSpec this will remove the spec file from SPECS No removeSource Flag (optional, default=false) to remove the sources after the build. See the the –rmsource option of rpmbuild. No command very similar idea to the cvs task. the default is "-bb" No output/error where standard output and error go No

5.3.37 ServerDeploy

ANT ServerDeploy User Manual

by

* Christopher A. Longo (cal@cloud9.net) * Cyrille Morvan (cmorvan@ingenosya.com)

At present the tasks support:

* Weblogic servers * JOnAS 2.4 Open Source EJB server

Over time we expect further optional tasks to support additional J2EE Servers.

Task Application Servers serverdeploy Nested Elements generic Generic task jonas JOnAS 2.4 weblogic Weblogic ServerDeploy element Description:

The serverdeploy task is used to run a "hot" deployment tool for vendor-specific J2EE server. The task requires nested elements which define the attributes of the vendor-specific deployment tool being executed. Vendor-specific deployment tools elements may enforce rules for which attributes are required, depending on the tool.

Parameters:

Attribute Description Required

action This is the action to be performed. For most cases this will be "deploy". Some tools support additional actions, such as "delete", "list", "undeploy", "update"... Yes source A fully qualified path/filename of the component to be deployed. This may be an .ear, .jar, .war, or any other type that is supported by the server. Tool dependant Nested Elements

The serverdeploy task supports a nested classpath element to set the classpath. Vendor-specific nested elements

Parameters used for all tools:

Attribute Description Required classpath The classpath to be passed to the JVM running the tool. The classpath may also be supplied as a nested element. Tool dependant server The address or URL for the server where the component will be deployed. Tool dependant username The user with privileges to deploy applications to the server. Tool dependant password The password of the user with privileges to deploy applications to the server. Tool dependant

Also supported are nested vendor-specific elements. Generic element

This element is provided for generic Java-based deployment tools. The generic task accepts (but does not require) nested arg and jvmarg elements. A JVM will be spawned with the provided attributes. It is recommended that a vendor-specific element be used over the generic one if at all possible.

The following attributes are supported by the generic element.

Attribute Description Required

classname This is the fully qualified classname of the Java based deployment tool to execute. Yes

Nested Elements

The generic element supports nested ¡arg¿ and ¡jvmarg¿ elements.

Example

This example shows the use of generic deploy element to deploy a component using a Java based deploy tool:

```
<serverdeploy action="deploy" source="${lib.dir}/ejb_myApp.ear">
    <generic classname="com.yamato.j2ee.tools.deploy.DeployTool"
        classpath="${classpath}"
        username="${user.name}"
        password="${user.password}">
        <arg value="-component=WildStar"/>
        <arg value="-force"/>
        <jvmarg value="-ms64m"/>
        <jvmarg value="-mx128m"/>
    </generic>
</serverdeploy>
```

WebLogic element

The WebLogic element contains additional attributes to run the weblogic.deploy deployment tool.

Valid actions for the tool are deploy, undeploy, list, update, and delete.

If the action is deploy or update, the application and source attributes must be set. If the action is undeploy or delete, the application attribute must be set. If the username attribute is omitted, it defaults to ”system”. The password attribute is required for all actions.

Attribute Description Required application This is the name of the application being deployed Yes component This is the component string for deployment targets. It is in the form ¡component¿:¡target1¿,¡target2¿... Where component is the archive name (minus the .jar, .ear, .war extension). Targets are the servers where the components will be deployed no debug If set to true, additional information will be printed during the deployment process. No

Examples

This example shows the use of serverdeploy to deploy a component to a WebLogic server:

```
<serverdeploy action="deploy" source="${lib.dir}/ejb_myApp.ear">
    <weblogic application="myapp"
        server="t3://myserver:7001"
        classpath="${weblogic.home}/lib/weblogic.jar"
        username="${user.name}"
        password="${user.password}"
        component="ejb_foobar:myserver,productionserver"
        debug="true"/>
</serverdeploy>
```

This example shows serverdeploy being used to delete a component from a WebLogic server:

```
<serverdeploy action="delete" source="${lib.dir}/ejb_myApp.jar"/>
    <weblogic application="myapp"
        server="t3://myserver:7001"
        classpath="${weblogic.home}/lib/weblogic.jar"
        username="${user.name}"
        password="${user.password}"/>
</serverdeploy>
```

JOnAS (Java Open Applicaton Server) element

The JOnAS element contains additional attributes to run the JonasAdmin deployment tool.

Valid actions for the tool are deploy, undeploy, list and update.

You can't use user and password property with this task.

Attribute Description Required

jonasroot The root directory for JOnAS. Yes orb Choose your ORB : RMI, JEREMIE, DAVID, ... If omitted, it defaults to the one present in classpath. The corresponding JOnAS JAR is automatically added to the classpath. If your orb is DAVID (RMI/IIOP) you must specify davidhost and davidport properties. No davidhost The value for the system property : david.CosNaming.default_host . No davidport The value for the system property : david.CosNaming.default_port . No classname This is the fully qualified classname of the Java based deployment tool to execute. Default to org.objectweb.jonas.adm.JonasAdmin No Nested Elements

The jonas element supports nested ¡arg¿ and ¡jvmarg¿ elements.

Examples

This example shows the use of serverdeploy to deploy a component to a JOnAS server:

```
<serverdeploy action="deploy" source="${lib.dir}/ejb_myApp.jar">
    <jonas server="MyJOnAS" jonasroot="${jonas.root}">

        <classpath>
            <pathelement path="${jonas.root}/lib/RMI_jonas.jar"/>
            <pathelement path="${jonas.root}/config/"/>
        </classpath>
    </jonas>
</serverdeploy>
```

This example shows serverdeploy being used to list the components from a JOnAS server and a WebLogic server:

```
<serverdeploy action="list"/>
    <jonas jonasroot="${jonas.root}" orb="JEREMIE"/>
    <weblogic application="myapp"
        server="t3://myserver:7001"
        classpath="${weblogic.home}/lib/weblogic.jar"
        username="${user.name}"
        password="${user.password}"/>
</serverdeploy>
```

5.3.38 Setproxy

Setproxy Task

Sets Java's web proxy properties, so that tasks and code run in the same JVM can have through-the-firewall access to remote web sites, and remote ftp sites. Apache Ant

Description

Sets Java's web proxy properties, so that tasks and code run in the same JVM can have through-the-firewall access to remote web sites, and remote ftp sites. You can nominate an http and ftp proxy, or a socks server, reset the server settings, or do nothing at all.

Examples

¡setproxy/¿

do nothing

¡setproxy proxyhost="firewall"/¿

set the proxy to firewall:80

¡setproxy proxyhost="firewall" proxyport="81"/¿

set the proxy to firewall:81

¡setproxy proxyhost=""/¿

stop using the http proxy; don't change the socks settings

¡setproxy socksproxyhost="socksy"/¿

use socks via socksy:1080

¡setproxy socksproxyhost=""/¿

stop using the socks server

Parameters

Attribute Description Type nonProxyHosts A list of hosts to bypass the proxy on. These should be separated with the vertical bar character '—'. Only in Java 1.4 does ftp use this list. e.g. fozbot.corp.sun.com—*.eng.sun.com. String proxyHost the HTTP/ftp proxy host. Set this to "" for the http proxy option to be disabled String proxyPort the HTTP/ftp proxy port number; default is 80 int socksProxyHost The name of a Socks server. Set to ”” to turn socks proxying off. String socksProxyPort Set the ProxyPort for socks connections. The default value is 1080 int

Parameters as nested elements

5.3.39 Script

Description

Execute a script in a BSF supported language.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

All items (tasks, targets, etc) of the running project are accessible from the script, using either their name or id attributes (as long as their names are considered valid Java identifiers, that is). The name ”project” is a pre-defined reference to the Project, which can be used instead of the project name.

BeanShell users: This task now natively supports the BeanShell scripting language, using language=”beanshell”. The BeanShell engine is still required.

Scripts can do almost anything a task written in Java could do. Parameters Attribute Description Required language The programming language the script is written in. Must be a supported BSF language Yes src The location of the script as a file, if not inline No Examples

```
¡project name=”squares” default=”main” basedir=”.”¿
¡target name=”setup”¿
¡script language=”javascript”¿ ¡![CDATA[
for (i=1; i¡=10; i++) echo = squares.createTask(”echo”); main.addTask(echo);
echo.setMessage(i*i);
]]¿ ¡/script¿
¡/target¿
¡target name=”main” depends=”setup”/¿
¡/project¿
```

generates

```
setup:
main: 1 4 9 16 25 36 49 64 81 100
BUILD SUCCESSFUL
```

Another example, using references by id and two different scripting languages:

```
¡project name=”testscript” default=”main”¿ ¡target name=”sub”¿ ¡echo id=”theEcho”/¿
¡/target¿
¡target name=”sub1”¿ ¡script language=”netrexx”¿¡![CDATA[ theEcho.setMessage(”In
sub1”) sub.execute ]]¿¡/script¿ ¡/target¿
¡target name=”sub2”¿ ¡script language=”javascript”¿¡![CDATA[ theEcho.setMessage(”In
sub2”); sub.execute(); ]]¿¡/script¿ ¡/target¿
¡target name=”main” depends=”sub1,sub2”/¿ ¡/project¿
```

generates

```
sub1: In sub1
```

sub2: In sub2
main:
BUILD SUCCESSFUL

5.3.40 Sound

Description

Plays a sound-file at the end of the build, according to whether the build failed or succeeded. You can specify either a specific sound-file to play, or, if a directory is specified, the ¡sound¿ task will randomly select a file to play. Note: At this point, the random selection is based on all the files in the directory, not just those ending in appropriate suffixes for sound-files, so be sure you only have sound-files in the directory you specify.

Unless you are running on Java 1.3 or later, you need the Java Media Framework on the classpath (javax.sound). Nested Elements success

Specifies the sound to be played if the build succeeded. fail

Specifies the sound to be played if the build failed. Nested Element Parameters

The following attributes may be used on the ¡success¿ and ¡fail¿ elements: Attribute Description Required source the path to a sound-file directory, or the name of a specific sound-file, to be played. Yes loops the number of extra times to play the sound-file; default is 0. No duration the amount of time (in milliseconds) to play the sound-file. No Examples

¡target name=”fun” if=”fun” unless=”fun.done”¿ ¡sound¿ ¡success source=”*user.home/sounds/b*
failsource = ”user.home/sounds/ohno.wav” loops=”2”/¿ ¡/sound¿ ¡property name=”fun.done” value=”true”/¿ ¡/target¿

plays the bell.wav sound-file if the build succeeded, or the ohno.wav sound-file if the build failed, three times, if the fun property is set to true. If the target is a dependency of an ”initialization” target that other targets depend on, the fun.done property prevents the target from being executed more than once.

¡target name=”fun” if=”fun” unless=”fun.done”¿ ¡sound¿ ¡success source=”//intranet/sounds/su
¡fail source=”//intranet/sounds/failure”/¿ ¡/sound¿ ¡property name=”fun.done” value=”true”/¿ ¡/target¿

randomly selects a sound-file to play when the build succeeds or fails.

5.3.41 SourceOffSite

by

* Jesse Stockall

Version 1.1 2002/01/23

Contents

* Introduction * The Tasks

Introduction

These tasks provide an interface to the Microsoft Visual SourceSafe SCM via SourceGear’s SourceOffSite product. SourceOffSite is an add-on to Microsoft’s VSS, that allows remote development teams and tele-commuters that need fast and secure read/write access to a centralized SourceSafe database via any TCP/IP connection. SOS provides Linux ,Solaris & Windows clients. The org.apache.tools.ant.taskdefs.optional.sos package consists of a simple framework to support SOS functionality as well as some Ant tasks encapsulating frequently used SOS commands. Although it is possible to use these commands on the desktop, they were primarily intended to be used by automated build systems. These tasks have been tested with SourceOffSite version 3.5.1 connecting to VisualSourceSafe 6.0. The tasks have been tested with Linux, Solaris & Windows2000.

The Tasks sosget Retrieves a read-only copy of the specified project or file. soslabel Assigns a label to the specified project. soscheckin Updates VSS with changes made to a checked out file or project, and unlocks the VSS master copy. soscheckout Retrieves a read-write copy of the specified project or file, locking the VSS master copy

Task Descriptions SOSGet Description Task to perform GET commands with SOS Parameters Attribute Values Required soscmd Directory which contains soscmd(.exe) soscmd(.exe) must be in the path if this is not specified No vssserverpath path to the srcsafe.ini - eg.

server
vss
srcsafe.ini Yes sosserverpath address & port of the SOS server - eg. 192.168.0.1:8888 Yes projectpath SourceSafe project path without the "$" Yes file Filename to act upon If no file is specified then act upon the project No username SourceSafe username Yes password SourceSafe password No localpath Override the working directory and get to the specified path No soshome The path to the SourceOffSite home directory No nocompress true or false - disable compression No recursive true or false - Only works with the GetProject command No version a version number to get - Only works with the GetFile command No label a label version to get - Only works with the GetProject command No nocache true or false - Only needed if SOSHOME is set as an environment variable No verbose true or false - Status messages are displayed No

Example

```
<sosget verbose="true"
        recursive="true"
        username="build"
        password="build"
        localpath="tmp"
        projectpath="/SourceRoot/project1"
        sosserverpath="192.168.10.6:8888"
        vssserverpath="d:\vss\srcsafe.ini"/>
```

Connects to a SourceOffsite server on 192.168.10.6:8888 with build,build as the username & password. The SourceSafe database resides on the same box as the SOS server & the VSS database is at "d:
vss

srcsafe.ini" Does a recursive GetProject on $/SourceRoot/project1, using tmp as the working directory.

SOSLabel Description

Task to perform Label commands with SOS Parameters Attribute Values Required soscmd Directory which contains soscmd(.exe) soscmd(.exe) must be in the path if this is not specified No vssserverpath path to the srcsafe.ini - eg.

server
vss
srcsafe.ini Yes sosserverpath address and port of the SOS server - eg. 192.168.0.1:8888 Yes projectpath SourceSafe project path without the "$" Yes username SourceSafe username Yes password SourceSafe password No label The label to apply to a project Yes comment A comment to be applied to all files being labeled No verbose true or false - Status messages are displayed No Example

```
<soslabel username="build"
          password="build"
          label="test label"
          projectpath="/SourceRoot/project1
          sosserverpath="192.168.10.6:8888"
          vssserverpath="d:\vss\srcsafe.ini"/>
```

Connects to a SourceOffsite server on 192.168.10.6:8888 with build,build as the username & password. The SourceSafe database resides on the same box as the SOS server & the VSS database is at "d:
vss
srcsafe.ini". Labels the $/SourceRoot/project1 project with "test label".

SOSCheckIn Description

Task to perform CheckIn commands with SOS

Parameters

Attribute Values Required soscmd Directory which contains soscmd(.exe) soscmd(.exe) must be in the path if this is not specified No vssserverpath path to the srcsafe.ini - eg.

server
vss
srcsafe.ini Yes sosserverpath address and port of the SOS server - eg. 192.168.0.1:8888 Yes projectpath SourceSafe project path without the "$" Yes file Filename to act upon If no file is specified then act upon the project No username SourceSafe username Yes password SourceSafe password No localpath Override the working directory and get to the specified path No soshome The path to the SourceOffSite home directory No nocompress true or false - disable compression No recursive true or false - Only works with the CheckOutProject command No nocache true or false - Only needed if SOSHOME is set as an environment variable No verbose true or false - Status messages are displayed No comment A comment to be applied to all files being checked in No

Example

```
<soscheckin username="build"
            password="build"
            file="foobar.txt"
            verbose="true"
            comment="comment abc"
            projectpath="/SourceRoot/project1"
            sosserverpath="server1:8888"
            vssserverpath="\\server2\vss\srcsafe.ini"/>
```

Connects to a SourceOffsite server on server1:8888 with build,build as the username & password. The SourceSafe database resides on a different box (server2) & the VSS database is on a share called "vss". Checks-in only the "foobar.txt" file adding a comment of "comment abc". Extra status messages will be displayed on screen.

SOSCheckOut Description Task to perform CheckOut commands with SOS Parameters Attribute Values Required soscmd Directory which contains soscmd(.exe) soscmd(.exe) must be in the path if this is not specified No vssserverpath path to the srcsafe.ini - eg.

server
vss
srcsafe.ini Yes sosserverpath address and port of the SOS server - eg. 192.168.0.1:8888 Yes projectpath SourceSafe project path without the "$" Yes file Filename to act upon If no file is specified then act upon the project No username SourceSafe username Yes password SourceSafe password No localpath Override the working directory and get to the specified path No soshome The path to the SourceOffSite home directory No nocompress true or false - disable compression No recursive true or false - Only works with the CheckOutProject command No nocache true or false - Only needed if SOSHOME is set as an environment variable No verbose true or false - Status messages are displayed No

Example

```
<soscheckout soscmd="/usr/local/bin"
             verbose="true"
             username="build"
             password="build"
             projectpath="/SourceRoot/project1"
             sosserverpath="192.168.10.6:8888"
             vssserverpath="\\server2\vss\srcsafe.ini"/>
```

Connects to a SourceOffsite server on server1:8888 with build,build as the username & password. The SourceSafe database resides on a different box (server2) & the VSS database is on a share called "vss". Checks-out "project1", Only the "project1" directory will be locked as the recursive option was not set. Extra status messages will be displayed on screen. The soscmd(.exe) file to be used resides in /usr/local/bin.

5.3.42 Splash

by Les Hughes (leslie.hughes@rubus.com) Description

This task creates a splash screen. The splash screen is displayed for the duration of the build and includes a handy progress bar as well. Use in conjunction with the sound task to provide interest whilst waiting for your builds to complete... Parameters Attribute Description Required Default imageurl A URL pointing to an image to display. No antlogo.gif from the classpath showduration Initial period to pause the build to show the splash in milliseconds. No 5000 ms Deprecated properties The following properties can be used to configure the proxy settings to retrieve an image from behind a firewall. However, the settings apply not just to this task, but to all following tasks. Therefore they are now mostly deprecated in preference to the ¡setproxy¿ task, that makes it clear to readers of the build exactly what is going on. We say mostly as this task's support includes proxy authentication, so you may still need to use its proxy attributes. useproxy Use a proxy to access imgurl. Note: Only tested on JDK 1.2.2 and above No None proxy IP or hostname of the proxy server No None port Proxy portnumber No None user User to authenticate to the proxy as. No None password Proxy password No None

Examples

```
<splash/>
```

Splash images/ant_logo_large.gif from the classpath.

```
<splash imageurl="http://jakarta.apache.org/images/jakarta-logo.gif"
          useproxy="true"
          showduration="5000"/>
```

Splashes the jakarta logo, for an initial period of 5 seconds.

5.3.43 Starteam Tasks

* STCheckout * STCheckin * STLabel * STList * StarTeam (deprecated)

These tasks make use of functions from the StarTeam API. As a result they are only available to licensed users of StarTeam. You must have starteam-sdk.jar in your classpath to run these tasks. For more information about the StarTeam API and how to license it, see the StarBase web site. Common Parameters for All Starteam Tasks

The following parameters, having to do with making the connection to a StarTeam project, are common to all the following tasks except the deprecated StarTeam task. Attribute Description Required username The username of the account used to log in to the StarTeam server. yes password The password of the account used to log in to the StarTeam server. yes URL A string of the form servername:portnum/project/view which enables user to set all of these elements in one string. Either this ... servername The name of the StarTeam server. ... or all four of these must be defined. serverport The port number of the StarTeam server. projectname The name of the StarTeam project on which to operate. viewname The name of the view in the StarTeam project on which to operate. STCheckout Description

Checks out files from a StarTeam project.

The includes and excludes attributes function differently from other tasks in Ant. Inclusion/exclusion by folder is NOT supported.

Parameters See also the required common StarTeam parameters.

Attribute Description Required rootstarteamfolder The root of the subtree in the StarTeam repository from which to check out files. Defaults to the root folder of the view ('/'). no rootlocalfolder The local folder which will be the root of the tree to which files are checked out. If this is not supplied, then the StarTeam "default folder" associated with rootstarteamfolder is used. no createworkingdirs creates local folders even when the corresponding StarTeam folder is empty. Defaults to "true". no deleteuncontrolled if true, any files NOT in StarTeam will be deleted. Defaults to "true". no includes Only check out files that match at least one of the patterns in this list. Patterns must be separated by commas. Patterns in excludes take precedence over patterns in includes. no excludes Do not check out files that match at least one of the patterns in this list. Patterns must be separated by commas. Patterns in excludes take precedence over patterns in includes. no label Check out files as of this label. The label must exist in starteam or an exception will be thrown. If not specified, the most recent version of each file will be checked out. no recursive Indicates if subfolders should be searched for files to check out. Defaults to "true". no forced If true, checkouts will occur regardless of the status that StarTeam is maintaining for the file. If rootlocalfolder is set then this should be set "true" as otherwise the checkout will be based on statuses which do not relate to the target folder. Defaults to "false". no locked If true, file will be locked against changes by other users. If false (default) has no effect. Either or neither, but not both, may be true. unlocked If true, file will be unlocked so that other users may change it. This is a way to reverse changes that have not yet been checked in. If false (default) has no effect.

Examples

```
<stcheckout servername="STARTEAM"
            serverport="49201"
            projectname="AProject"
            viewname="AView"
            username="auser"
            password="secret"
            rootlocalfolder="C:\dev\buildtest\co"
            force="true"
/>
```

The minimum necessary to check out files out from a StarTeam server. This will check out all files in the AView view of the AProject project to C:
dev
buildtest

co. Empty folders in StarTeam will have local folders created for them and any non-StarTeam files found in the tree will be deleted.

```
<stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootlocalfolder="C:\dev\buildtest\co"
            forced="true"
/>
```

And this is a simpler way of accomplishing the same thing as the previous example, using the URL attribute.

```
<stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootlocalfolder="C:\dev\buildtest\co"
            rootstarteamfolder="\Dev"
            excludes="*.bak *.old"
            label="v2.6.001"
            forced="true"
/>
```

This will check out all files from the Dev folder and below that do not end in .bak or .old with the label v2.6.001.

```
<stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootlocalfolder="C:\dev\buildtest\co"
            includes="*.htm,*.html"
            excludes="index.*"
            forced="true"
/>
```

This is an example of overlapping includes and excludes attributes. Because excludes takes precedence over includes, files named index.html will not be checked out by this command.

```
<stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootlocalfolder="C:\dev\buildtest\co"
            includes="*.htm,*.html"
            excludes="index.*"
            forced="true"
            recursive="false"
/>
```

This example is like the previous one, but will only check out files in C:
dev
buildtest
co, because of the turning off of the recursive attribute.

```
<stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootstarteamfolder="src/java"
            rootlocalfolder="C:\dev\buildtest\co"
            forced="true"
/>
```

```
<stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootstarteamfolder="src/java"
/>
```

```
 <stcheckout URL="STARTEAM:49201/Aproject/AView"
            username="auser"
            password="secret"
            rootstarteamfolder="src/java"
            rootlocalfolder="C:\dev\buildtest\co\src\java"
            forced="true"
/>
```

In the preceding three examples, assuming that the AProject project has a default folder of "C:
work
AProject", the first example will check out the tree of files rooted in the src/java folder of the AView view of the AProject in the StarTeam repository to a local tree rooted at C:
dev
buildtest
co, the second to a tree rooted at C:
work
AProject
src
java (since no rootlocalfolder is specified) and the third to a tree rooted at C:
dev
buildtest
co

src
java. Note also, that since the second example does not set "forced" true, only those files which the repository considers out-of-date will be checked out.

STCheckin Description

Checks files into a StarTeam project. Optionally adds files and in the local tree that are not managed by the repository to its control.

The includes and excludes attributes function differently from other tasks in Ant. Inclusion/exclusion by folder is NOT supported.

Parameters See also the required common StarTeam parameters.

Attribute Description Required rootstarteamfolder The root of the subtree in the StarTeam repository into which to files will be checked. Defaults to the root folder of the view ('/'). no rootlocalfolder The local folder which will be the root of the tree to which files are checked out. If this is not supplied, then the StarTeam "default folder" associated with rootstarteamfolder is used. no comment Checkin comment to be saved with the file. no adduncontrolled if true, any files or folders NOT in StarTeam will be added to the repository. Defaults to "false". no includes Only check in files that match at least one of the patterns in this list. Patterns must be separated by commas. Patterns in excludes take precedence over patterns in includes. no excludes Do not check in files that match at least one of the patterns in this list. Patterns must be separated by commas. Patterns in excludes take precedence over patterns in includes. no recursive Indicates if subfolders should be searched for files to check in. Defaults to "false". no forced If true, checkins will occur regardless of the status that StarTeam is maintaining for the file. If rootlocalfolder is set then this should be set "true" as otherwise the checkin will be based on statuses which do not relate to the target folder. Defaults to "false". no unlocked If true, file will be unlocked so that other users may change it. If false (default) lock status will not change. no

Examples

```
<stcheckin servername="STARTEAM"
           serverport="49201"
           projectname="AProject"
           viewname="AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           forced="true"
/>
```

The minimum necessary to check files into a StarTeam server. This will check all files on the local tree rooted at C:
dev
buildtest
co into the AView view of the AProject project in the repository. For files and folders in the local tree but not in starteam, nothing will be done. Since the forced attribute is set, the files which are checked in will be checked in without

regard to what the StarTeam repository considers their status to be. This is a reasonable choice of attributes since StarTeam's status for a file is calculated based on the local file in the StarTeam default directory, not on the directory we are actually working with.

```
<stcheckin URL="STARTEAM:49201/Aproject/AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           forced="true"
/>
```

And this is a simpler way of giving the same commands as the command above using the URL shortcut.

```
<stcheckin URL="STARTEAM:49201/Aproject/AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           rootstarteamfolder="\Dev"
           excludes="*.bak *.old"
           forced="true"
/>
```

This will check all files in to the Dev folder and below that do not end in .bak or .old from the tree rooted at"C:
dev
buildtest
co" .

```
<stcheckin URL="STARTEAM:49201/Aproject/AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           includes="*.htm,*.html"
           excludes="index.*"
           forced="true"
/>
```

This is an example of overlapping includes and excludes attributes. Because excludes takes precedence over includes, files named index.html will not be checked in by this command.

```
<stcheckin URL="STARTEAM:49201/Aproject/AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           rootstarteamfolder="src/java"
```

```
              includes="*.htm,*.html"
              excludes="index.*"
              forced="true"
              recursive="false"
/>
```

This example is like the previous one, but will only check in files from C:
dev
buildtest
co, because of the turning off of the recursive attribute.

```
<stcheckin URL="STARTEAM:49201/Aproject/AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           rootstarteamfolder="src/java"
           includes="version.txt"
           forced="true"
           recursive="false"
/>
```

This example is like the previous one, but will only check only in one file, C:
dev
buildtest
co
version.txt to the StarTeam folder src/java.

```
<stcheckin URL="STARTEAM:49201/Aproject/AView"
           username="auser"
           password="secret"
           rootlocalfolder="C:\dev\buildtest\co"
           rootstarteamfolder="src/java"
           includes="version.java"
           forced="true"
           recursive="false"
           addUncontrolled="true"
           comment="Fix Bug #667"
/>
```

This example is like the previous one, but will only check only in one file, C:
dev
buildtest
co
version.java to the StarTeam folder src/java. Because the addUncontrolled attribute has been set, if StarTeam does not already control this file in this location, it will be added to the repository. Also, it will write a comment to the repository for this version of the file.

```
  <stcheckin URL="STARTEAM:49201/Aproject/AView"
             username="auser"
             password="secret"
             rootstarteamfolder="src/java"
             rootlocalfolder="C:\dev\buildtest\co"
             forced="true"
  />
```

```
  <stcheckin URL="STARTEAM:49201/Aproject/AView"
             username="auser"
             password="secret"
             rootstarteamfolder="src/java"
  />
```

```
 <stcheckin URL="STARTEAM:49201/Aproject/AView"
             username="auser"
             password="secret"
             rootstarteamfolder="src/java"
             rootlocalfolder="C:\dev\buildtest\co\src\java"
             forced="true"
  />
```

In the preceding three examples, assuming that the AProject project has a default folder of C:
work
buildtest
co
AProject, the first example will check in files from a tree rooted at C:
dev
buildtest
co, the second from a tree rooted at C:
work
buildtest
co
AProject
src
java, and the third from a tree rooted at C:
dev
buildtest
co
src
java all to a tree rooted at src/java

STLabel Description

Creates a view label in StarTeam at the specified view. The label will be classified by StarTeam as a "build label". This task will fail if there already exexists in viewname a label with the same name as the label parameter.

Parameters See also the required common StarTeam parameters.

Attribute Description Required label The name to be given to the label yes lastbuild The timestamp of the build that will be stored with the label. Must be formatted yyyyMMddHHmmss yes description A description of the label to be stored in the StarTeam project. no

Examples

This example shows the use of this tag. It will create a label named Version 6.2 with "Thorough description" as its description.

```
<tstamp>
  <format property="nowstamp" pattern="yyyyMMddHHmmss" locale="en"/>
</tstamp>
<stlabel URL="STARTEAM:49201/Aproject/AView"
         username="auser"
         password="secret"
         label="Version 6.2"
         lastbuild="${nowstamp}"
         description="Thorough description"
/>
```

STList Description

Produces a listing of the contents of the StarTeam repository at the specified view and StarTeamFolder. The listing will contain the name of the user, if any, who has the file locked, the size of the file, its lastModifiedDate in the repository, and the name of the file. Unless the rootLocalFolder is specified, listing will also show the status of the local file in the default local directory relative to the repository.

Parameters See also the required common StarTeam parameters.

Attribute Description Required rootstarteamfolder The root of the subtree in the StarTeam repository to be listed. Defaults to the root folder of the view ('/'). no rootlocalfolder The local folder which will be the root of the tree to which files are compared. If this is not supplied, then the StarTeam "default folder" associated with rootstarteamfolder is used and a status field will appear in the listing. Otherwise, the status field will not appear. no includes Only list files that match at least one of the patterns in this list. Patterns must be separated by commas. Patterns in excludes take precedence over patterns in includes. no excludes Do not list files that match at least one of the patterns in this list. Patterns must be separated by commas. Patterns in excludes take precedence over patterns in includes. no label List files, dates, and statuses as of this label. The label must exist in starteam or an exception will be thrown. If not specified, the most recent version of each file will be listed. no recursive Indicates if subfolders should be searched for files to list. Defaults to "true". no Examples

¡stlist url="WASHINGTON:49201/build" username="auser" password="secret" /¿

The above command might produce the following listing:

[stlist] Folder: Build (Default folder: C:/work/build) [stlist] Folder: dev (Default folder: C:/work/build/dev) [stlist] Out of date Sue Developer 1/1/02 7:25:47 PM CST 4368 build.xml [stlist] Missing George Hacker 1/1/02 7:25:49 PM CST 36 Test01.properties [stlist] Current 1/1/02 7:25:49 PM CST 4368 build2.xml [stlist] Folder: test (Default folder C:/work/build/dev/test) [stlist] Missing 1/1/02 7:25:50 PM CST 4368 build2.xml

while adding a rootlocalfolder and an excludes param ...

¡stlist url="WASHINGTON:49201/build" username="auser" password="secret" rootlocalfolder="srcdir2" excludes="*.properties" /¿

might produce this listing. The status is missing because we are not going against the default folder.

[stlist] overriding local folder to srcdir2 [stlist] Folder: Build (Local folder: srcdir2) [stlist] Folder: dev (Local folder: srcdir2/dev) [stlist] Sue Developer 1/1/02 7:25:47 PM CST 4368 build.xml [stlist] 1/1/02 7:25:49 PM CST 4368 build2.xml [stlist] Folder: test (Local folder: srcdir2/dev/test) [stlist] 1/1/02 7:25:50 PM CST 4368 build2.xml

Starteam Deprecated

This task has been deprecated. Use the STCheckout task instead. Description Checks out files from a StarTeam project.

The includes and excludes attributes function differently from other tasks in Ant. Multiple patterns must be separated by spaces, not commas. See the examples for more information. Parameters Attribute Description Required username The username of the account used to log in to the StarTeam server. yes password The password of the account used to log in to the StarTeam server. yes servername The name of the StarTeam server. yes serverport The port number of the StarTeam server. yes projectname The name of the StarTeam project. yes viewname The name of the view in the StarTeam project. yes targetfolder The folder to which files are checked out. What this precisely means is determined by the targetFolderAbsolute param. yes targetFolderAbsolute Determines how targetfolder is interpreted, that is, whether the StarTeam "default folder" for the project is factored in (false) or whether targetFolder is a complete mapping to foldername (true). If "true", the target tree will be rooted at targetfolder+"default folder". If false, the target tree will be rooted at targetfolder. Defaults to "false". no foldername The subfolder in the project from which to check out files. no force Overwrite existing folders if this is set to "true". Defaults to "false". no recursion Indicates if subfolders should be searched for files to check out. Defaults to "true". no verbose Provides progress information. Defaults to "false". no includes Only check out files that match at least one of the patterns in this list. Patterns must be separated by spaces. Patterns in excludes take precedence over patterns in includes. no excludes Do not check out files that match at least one of the patterns in this list. Patterns must be separated by spaces. Patterns in excludes take precedence over patterns in includes. no

Examples

```
<starteam servername="STARTEAM"
          serverport="49201"
          projectname="AProject"
          viewname="AView"
          username="auser"
          password="secret"
          targetfolder="C:\dev\buildtest\co"
/>
```

The minimum necessary to check out files out from a StarTeam server. This will check out all files in the AView view of the AProject project to C:
dev
buildtest
co.

```
<starteam servername="STARTEAM"
          serverport="49201"
          projectname="AProject"
          viewname="AView"
          username="auser"
          password="secret"
          targetfolder="C:\dev\buildtest\co"
          foldername="\Dev"
          excludes="*.bak *.old"
          force="true"
/>
```

This will checkout all files from the Dev folder and below that do not end in .bak or .old. The force flag will cause any existing files to be overwritten by the version in StarTeam.

```
<starteam servername="STARTEAM"
          serverport="49201"
          projectname="AProject"
          viewname="AView"
          username="auser"
          password="secret"
          targetfolder="C:\dev\buildtest\co"
          includes="*.htm *.html"
          excludes="index.*"
/>
```

This is an example of overlapping includes and excludes attributes. Because excludes takes precedence over includes, files named index.html will not be checked out by this command.

```
<starteam servername="STARTEAM"
          serverport="49201"
          projectname="AProject"
          foldername="src/java"
          viewname="AView"
          username="auser"
          password="secret"
          targetfolder="C:\dev\buildtest\co"
          targetfolderabsolute="true"
 />
```

```
<starteam servername="STARTEAM"
          serverport="49201"
          projectname="AProject"
          foldername="src/java"
          viewname="AView"
          username="auser"
          password="secret"
          targetfolder="C:\dev\buildtest\co"
          targetfolderabsolute ="false"
 />
```

```
<starteam servername="STARTEAM"
          serverport="49201"
          projectname="AProject"
          foldername="src/java"
          viewname="AView"
          username="auser"
          password="secret"
          targetfolder="C:\dev\buildtest\co\src\java"
          targetfolderabsolute="true"
 />
```

In the preceding three examples, assuming that the AProject project has a default folder of "AProject", the first example will check the files located in starteam under src/java out to a tree rooted at C:
dev
buildtest
co, the second to a tree rooted at C:
dev
buildtest
co

AProject
src
java and the third to a tree rooted at C:
dev
buildtest
co
src
java.

5.3.44 Stylebook

Description

This executes the apache Stylebook documentation generator. Unlike the commandline version of this tool, all three arguments are required to run stylebook.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

Being extended from ¡Java¿, all the parent's attributes and options are available. Do not set any apart from the classpath as they are not guaranteed to be there in future.

Parameters Attribute Description Required

book the book xml file that the documentation generation starts from Yes skindirectory the directory that contains the stylebook skin Yes targetdirectory the destination directory where the documentation is generated Yes

The user can also specify the nested ¡classpath¿ element which defines classpath in which the task is executed.

Examples

```
<stylebook targetdirectory="build/docs"
           book="src/xdocs/book.xml"
           skindirectory="src/skins/myskin"/>
```

The above will generate documentation in build/docs starting from the book src/xdocs/book.xml and using the skin located in directory src/skins/myskin.

5.3.45 Telnet

Description

Task to automate a remote telnet session. The task uses nested ¡read¿ to indicate strings to wait for, and ¡write¿ tags to specify text to send.

If you do specify a userid and password, the system will assume a common unix prompt to wait on. This behavior can be easily over-ridden.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information.

Parameters Attribute Values Required userid the login id to use on the telnet server. Only if password is specified password the login password to use on the telnet server. Only if userid is specified server the address of the remote telnet server. Yes port the port number of the remote telnet server. Defaults to port 23. No initialCR send a cr after connecting ("yes"). Defaults to "no". No timeout set a default timeout to wait for a response. Specified in seconds. Default is no timeout. No Nested Elements The commands to send to the server, and responses to wait for, are described as nested elements. read

declare (as a text child of this element) a string to wait for. The element supports the timeout attribute, which overrides any timeout specified for the task as a whole. It also has a string attribute, which is an alternative to specifying the string as a text element. Always declare an opening and closing ¡read¿ element to ensure that statements are not sent before the connection is ready, and that the connection is not broken before the final command has completed. write

describes the text to send to the server. The echo boolean attribute controls whether the string is echoed to the local log; this is "true" by default Examples A simple example of connecting to a server and running a command. This assumes a prompt of "ogin:" for the userid, and a prompt of "assword:" for the password.

```
<telnet userid="bob" password="badpass" server="localhost">
   <read>/home/bob</read>
   <write>ls</write>
   <read string="/home/bob"/>
</telnet>
```

This task can be rewritten as:

```
<telnet server="localhost">
   <read>ogin:</read>
   <write>bob</write>
   <read>assword:</read>
   <write>badpass</write>
   <read>/home/bob</read>
   <write>ls</write>
   <read>/home/bob</read>
</telnet>
```

A timeout can be specified at the ¡telnet¿ level or at the ¡read¿ level. This will connect, issue a sleep command that is suppressed from displaying and wait 10 seconds before quitting.

```
<telnet userid="bob" password="badpass" server="localhost" timeout="20">
   <read>/home/bob</read>
   <write echo="false">sleep 15</write>
   <read timeout="10">/home/bob</read>
</telnet>
```

The task can be used with other ports as well:

```
<telnet port="80" server="localhost" timeout="20">
   <read/>
   <write>GET / http/0.9</write>
   <write/>
   <read timeout="10">&lt;/HTML&gt;</read>
</telnet>
```

To use this task against the WinNT telnet service, you need to configure the service to use classic authentication rather than NTLM negotiated authentication. This can be done in the Telnet Server Admin app: select ”display/change registry settings”, then ”NTLM”, then set the value of NTLM to 1.

5.3.46 Test

Description

This is a primitive task to execute a unit test in the org.apache.testlet framework.

This task is deprectated as the Testlet framework has been abandoned in favor of JUnit by the Avalon community.

Note: This task depends on external libraries not included in the Ant distribution. See Library Dependencies for more information. Parameters Attribute Description Required showSuccess a boolean value indicating whether tests should display a message on success No showBanner a boolean value indicating whether a banner should be displayed when starting testlet engine No forceShowTrace a boolean indicating that a stack trace is displayed on any failure No showTrace a boolean indicating that a stack trace is displayed on error (but not normal failure) No

The user can also specify the nested ¡classpath¿ element which defines classpath in which the task is executed. The user also specifies a subelement per testlet executed which has content that specifies tasklet classname. Examples

¡test showSuccess=”false” showBanner=”false” showTrace=”true” forceShowTrace=”true”¿ ¡classpath refid=”test.classpath”/¿ ¡testlet¿org.foo.MyTestlet¡/testlet¿ ¡testlet¿org.foo.MyOtherTestlet¡/testlet¿ ¡/test¿

The above will run the testlets org.foo.MyTestlet and org.foo.MyOtherTestlet

5.3.47 Translate

Description

Identifies keys in files delimited by special tokens and translates them with values read from resource bundles.

A resource bundle contains locale-specific key-value pairs. A resource bundle is a hierarchical set of property files. A bundle name makes up its base family name. Each file that makes up this bundle has this name plus its locale. For example, if the resource bundle name is MyResources, the file that contains German text will take the name MyResources_de. In addition to language, country and variant are also used to form the files in the bundle.

The resource bundle lookup searches for resource files with various suffixes on the basis of (1) the desired locale and (2) the default locale (basebundlename), in the following order from lower-level (more specific) to parent-level (less specific):

```
basebundlename + "_" + language1 + "_" + country1 + "_" + variant1
basebundlename + "_" + language1 + "_" + country1
basebundlename + "_" + language1
basebundlename
basebundlename + "_" + language2 + "_" + country2 + "_" + variant2
basebundlename + "_" + language2 + "_" + country2
basebundlename + "_" + language2
```

The file names generated thus are appended with the string ".properties" to make up the file names that are to be used.

File encoding is supported. The encoding scheme of the source files, destination files and the bundle files can be specified. Destination files can be exlicitly overwritten using the forceoverwrite attribute. If forceoverwrite is false, the destination file is overwritten only if either the source file or any of the files that make up the bundle have been modified after the destination file was last modified.

FileSets are used to select files to translate.

Parameters

Attribute Description Required todir Destination directory where destination files are to be created. Yes starttoken The starting token to identify keys. Yes endtoken The ending token to identify keys. Yes bundle Family name of resource bundle. Yes bundlelanguage Locale specific language of resource bundle. Defaults to default locale's language. No bundlecountry Locale specific country of resource bundle. Defaults to default locale's country. No bundlevariant Locale specific variant of resource bundle. Defaults to the default variant of the country and language being used. No srcencoding Source file encoding scheme. Defaults to system default file encoding. No destencoding Destination file encoding scheme. Defaults to source file encoding. No bundleencoding Resource Bundle file encoding scheme. Defaults to source file encoding. No forceoverwrite Overwrite existing files even if the destination files are newer. Defaults to "no". No

Parameters specified as nested elements

fileset

FileSets are used to select files that contain keys for which value translated files are to be generated.

Examples

Translate source file encoded in english into its japanese equivalent using a resource bundle encoded in japanese.

```
  <translate toDir="$(dest.dir}/ja"
        starttoken="#"
        endtoken="#"
        bundle="resource/BaseResource"
        bundlelanguage="ja"
        forceoverwrite="yes"
        srcencoding="ISO8859_1"
        destencoding="SJIS"
        bundleencoding="SJIS">
        <fileset dir="${src.dir}">
            <include name="**/*.jsp"/>
        </fileset>
  </translate>
```

5.3.48 Visual Age for Java Tasks

At the moment there are three tasks which help integrating the VAJ repository contents into an external build process:

VAJLoad loads specified versions into the workspace VAJExport exports specified packages into the file system VAJImport imports specified files into the workspace

These tasks are described in detail below.

VAJLoad Description:

Loads a specified VAJ project version into the workspace. Parameters Attribute Description Required remote

name and port of a remote tool server. (format: ¡servername¿:¡port no¿). If this attribute is set, the tasks will be executed on the specified tool server. no Parameters specified as nested elements vajproject Attribute Description Required name name of the VAJ project to load into the workspace yes version name of the requested version yes

Example

```
<vajload remote="localhost:32767">
    <vajproject name="My Testcases" version="1.7beta"/>
    <vajproject name="JUnit" version="3.2"/>
</vajload>
```

VAJExport Description:

Exports Java source files, class files and/or resources from the workspace to the file system. Exports can be specified by giving the VAJ project name and package name(s). This works very similar to FileSets.

Parameters Attribute Description Required destdir location to store the exported files yes exportSources export source files (default: ”yes”) no exportResources export resource files (default: ”yes”) no exportClasses export class files (default: ”no”) no exportDebugInfo include debug info in exported class files (default: ”no”) no defaultexcludes

use default excludes when exporting (default: ”yes”). Default excludes are: IBM*/**, Java class libraries/**, Sun class libraries*/**, JSP Page Compile Generated Code/**, VisualAge*/** no overwrite overwrite existing files (default: ”yes”) no remote name and port of a remote tool server. (format: ¡servername¿:¡port no¿). If this attribute is set, the tasks will be executed on the specified tool server. no Parameters specified as nested elements include specifies the packages to include into the export Attribute Description Required name name of the VAJ project and package to export. The first element of the name must be the project name, then the package name elements separated by '/'. yes exclude specifies the packages to exclude from the export Attribute Description Required name name of the VAJ project/package not to export yes

Example

```
<vajexport destdir="${src.dir}" exportResources="no">
    <include name="MyProject/**"/>
    <exclude name="MyProject/test/**"/>
</vajexport>
```

This example exports all packages in the VAJ project 'MyProject', except packages starting with 'test'.

Default Excludes The default excludes are:

IBM*/** Java class libraries/** Sun class libraries*/** JSP Page Compile Generated Code/** VisualAge*/**

VAJImport Description:

Imports Java source files, class files and/or resources from the file system into VAJ. These imports can be specified with a fileset. Parameters Attribute Description Required vajProject imported files are added to this VAJ project yes importSources import source files (default: "yes") no importResources import resource files (default: "yes") no importClasses import class files (default: "no") no remote

name and port of a remote tool server. (format: ¡servername¿:¡port no¿). If this attribute is set, the tasks will be executed on the specified tool server. no Parameters specified as nested elements fileset A FileSet specifies the files to import. Example

```
<vajimport project="Test" importClasses="true">
    <fileset dir="${import.dir}">
        <include name="com/sample/**/*.class"/>
        <exclude name="com/sample/test/**"/>
    </fileset>
</vajimport>
```

This example imports all class files in the directory ${import.dir}/com/sample excluding those in the subdirectory test

The Plugin The tasks are usable within VAJ by running the org.apache.tools.ant.Main class, but this is quite inconvenient. Therefore a small GUI is provided which allows selecting a build file and executing its targets. This Plugin is accessible from the VAJ Tools menu (see Usage). Installation

At the moment the installation has it's rough edges. If something described below doesn't work for You, it's probably not Your fault but incomplete/wrong instructions. In this case, please contact one of the authors.

We assume C:
IBMVJava as VAJ install directory. If You have installed it elsewhere, adapt the paths below.

Plugin

* install the Visual Age IDE Tools (via File-¿Quick Start-¿ Add feature-¿'IBM IDE Utility class libraries' * import an appropriate XML parser to VAJ (we use Xerces 1.2.0 and are happy with it). Unfortunately the XML parser delivered with VAJ (in the project 'IBM XML Parser for Java') doesn't work with Ant. You have to remove that project (temporarily) from the workspace before importing another XML implementation. * import the Ant sources and resources into VAJ. * Create the directory C:
IBMVJava
ide
tools
org-apache-tools-ant. * export the Ant and XML parser class and resource files into this directory. Be sure to select class files and resources. Sources don't have to be exported. Some optional tasks have errors and can't be exported when You don't have the necessary packages in Your workspace (e.g. junit task, ejbc task). If You need this tasks either import these packages into VAJ, too, or copy the .class files directly from the binary distribution. * copy default.ini (in ant
src:.
taskdefs
optional
ide) to C:
IBMVJava
ide
tools
org-apache-tools-ant
default.ini. * if you want to access this help from the Workbench, create the directory C:
IBMVJava
ide
tools
org-apache-tools-ant
doc and copy the files VAJAntTool.html, toolmenu.gif and anttool1.gif to it. * VAJ has to be restarted to recognize the new tool. * Now if You open the context menu of a project, You should see the entry 'Ant Build' in the Tools submenu (see Usage). * Make sure the tool works as expected. Now You can remove Ant and the imported XML parser from Your workspace (and optionally add the IBM parser again).

Servlets for Remote Tool Access

* For a good introduction into the VAJ Remote Tool Access see the great introduction from Glenn McAllister at http://www7.software.ibm.com/vad.nsf/Data/Document4366. It is highly recommended to read this article before doing the installation (to understand what you do :-)). * insert the following lines into C:

IBMVJava
ide
tools
com-ibm-ivj-toolserver
servlets
servlet.properties. Typically this file is empty. If not, be careful not to delete the other lines.

servlet.vajload.code=org.apache.tools.ant.taskdefs.optional.ide.VAJLoadServlet
servlet.vajexport.code=org.apache.tools.ant.taskdefs.optional.ide.VAJExportServlet
servlet.vajimport.code=org.apache.tools.ant.taskdefs.optional.ide.VAJImportServletName

* export the following classes from the package org.apache.tools.ant to C:
IBMVJava
ide
tools
com-ibm-ivj-toolserver
servlets
: BuildException DirectoryScanner FileScanner Location * export the following classes from the package org.apache.tools.ant.taksdefs.optional.ide to C:
IBMVJava
ide
tools
com-ibm-ivj-toolserver
servlets
: VAJUtil VAJExportServlet VAJImportServlet VAJLoadServlet VAJLocalUtil VAJProjectDescription VAJToolsServlet VAJWorkspaceScanner * configure the Remote Access (via Window-¿Options..., then choose 'Remote Access To Tool API') as shown in the following picture:

Now you should be able to execute VAJ Tasks from the command line. Usage Plugin

When the tool is installed correctly and your Ant build file is configured, it is really easy to use. Go to your Workbench, select the project you want to deploy and open its context menu. In the submenu Tools you should find the new entry Ant Build. Klick it to start the tool! After a short time this frame should pop up: This frame contains the following elements:

* A menubar with some options described later * The name of your selected VAJ project * An entry field for the Ant XML buildfile with a browse [...] button. The full qualified filename, including the directory is needed here. * A list with tasks specified in the buildfile. Until your first save of the build info (described later), this list will be empty. When loading a build file by the (Re)Load button, this list is filled with all tasks which have a description attribute. The task you select in this list will be executed when pressing the Execute button. * A pulldown box for specifying the log level. * Four buttons. Two of them I have already described. The other are the Stop button to cancel a running build and the third one is just the Close button to exit our small tool! * Note that the build is canceled on the next console output after pressing the Stop button, not directly after pressing it.

After you have set up your buildprocess you might find it useful to save the data you've just entered, so we implemented an option to save it to the repository into your selected project. Make sure that you have an open edition of your project before selecting Save BuildInfo To Repository from the File menu. Now your information is saved to this edition of your project and will be loaded automatically the next time you start Ant Build. If you have closed the log window accidentally, it can be reopened with the Log item in the File menu, and if you want to know who developed this, just select About in the Help menu. Servlets for Remote Tool Access

With the servlets installed and the remote access running you can use Ant from the command line without any restrictions. Just make sure the remote attribute in your build file is set correctly. Frequently Asked Questions

Q: If I try to load a build file, I get the error "Can't load default task list". Why? A: Ant not only contains class files, but also resource files. This message appears if the file .../org/apache/tools/ant/taskdefs/defaults.properties is missing. Make sure that you import/export not only java/class files, but also all resource files when importing/exporting Ant.

Q: I want to load, export and build more then one Visual Age project to one jar! How to? A: The VA tasks are able to load and export several Projects all at once. You can choose whatever project you like for storing the tool information, it doesn't really matter

Q: When I load my build file, the list of targets is empty. Why? A: You need to add the optional "description" parameter to the targets you want to come up in the list. Then reload the build file in the "ant build" tool. We chose to display only targets with description to allow the build file developer to distinguish between targets for end users and helper targets.

Q: Is there a sample build file available? A: Now you can find an example in this manual

Q: Why does it export my entire workspace when I've already implicitly selected a project when starting the Tool? A: This selection does not carry into the buildfile you are using. Set the Project name at the beginning of the "includes" parameter.

Q: When I import Ant into my Workspace, I get Problems reported. Can I ignore them? A: It depends on the problems reported, and what you want to do with Ant. Problems you can't ignore:

* Classes from javax.xml.parser missing - install a compatible parser (see installation) * Classes from com.ibm.ivj.util missing - install the Visual Age IDE Utility feature (see installation). * Errors in optional tasks you use within your build file

Q: I want to use the same buildfile both within Visual Age and from the command line using my regular Ant environment. What do I need to be aware of? A: You have to specifie a remote server via the 'remote' attribute. Otherwise the three Visual Age tasks won't work when executing Ant from the command line.

Q: I can export packages from project 'ABC', but not from project 'XYZ'! Why? A: Common reasons are:

* The project is excluded by the default excludes (see attribute 'defaultexcludes' of VAJExport) * When looking at the project in the workspace, it is often difficult to distinguish between project name and version name (e.g. as in 'My GUI Components Java 2 3.5'). Check if you have the right project name by switching off the version name display temporarilly.

Q: How do I control the import/export of sourcefiles, compiled files and project resources explicity? A: Via the Boolean values exportClasses (default false) exportSources (default true) and exportResources (default true). In some situations, Resources are not exported correctly without this being explicity set. VAJ doesn't export resources correctly if a package contains only resources (see below). Known Problems

* Exporting a package containing just resources doesn't work. This is a VAJ Tool API bug. Workaround: create a dummy class and set 'exportSources' to false.

VisualAge for Java Versions This tool integration has been tested with versions 3.02 and 3.5 of VisualAge for Java. It should run with the 2.x Versions, too, but we didn't try. The graphical user interface is built with AWT so it is JDK independent by now.

History

```
1.0 2000/09/11 Initial Version
1.1 2001/02/14 Added Task documentation and more FAQs (thanks to Richard Bourke for the FAQ ac
1.2 2001/07/02

Added documentation of new remote feature.
Minor corrections.
```

5.3.49 Microsoft Visual SourceSafe Tasks

vssget Retrieves a copy of the specified VSS file(s). vsslabel Assigns a label to the specified version or current version of a file or project. vsshistory Shows the history of a file or project in VSS. vsscheckin Updates VSS with changes made to a checked out file, and unlocks the VSS master copy. vsscheckout Copies a file from the current project to the current folder, for the purpose of editing. vssadd Adds a new file into the VSS Archive vsscp Change the current project being used in VSS vsscreate Creates a project in VSS. Task Descriptions VssGet Description Task to perform GET commands to Microsoft Visual SourceSafe.

If you specify two or more attributes from version, date and label only one will be used in the order version, date, label. Parameters Attribute Values Required vsspath SourceSafe path which specifies the project/file(s) you wish to perform the action on. You should not specify the leading dollar-sign - it is prepended by Ant automatically. Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No localpath Override the working directory and get to the specified path No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where ss.ini resides. No writable true or false¡; default false/td¿ No recursive true or false; default false. Note however that in the SourceSafe UI , there is a setting accessed via Tools/Options/GeneralTab called "Act on projects recursively". If this setting is checked, then the recursive attribute is effectively ignored, and the get will always be done recursively No version a version number to get No, only one of these allowed date a date stamp to get at label a label to get for quiet suppress output (off by default) No autoresponse What to respond with (sets the -I option). By default, -I- is used; values of Y or N will be appended to this. No

Note that only one of version, date or label should be specified

Examples

```
<vssget localPath="C:\mysrc\myproject"
        recursive="true"
        label="Release1"
        login="me,mypassword"
        vsspath="/source/aProject"
        writable="true"/>
```

Does a get on the VSS-Project $/source/myproject using the username me and the password mypassword. It will recursively get the files which are labeled Release1 and write them to the local directory C:
mysrc
myproject. The local files will be writable.

VssLabel Description Task to perform LABEL commands to Microsoft Visual SourceSafe.

Assigns a label to the specified version or current version of a file or project.

Parameters Attribute Values Required vsspath SourceSafe path which specifies the project/file(s) you wish to perform the action on. You should not specify the leading dollar-sign - it is prepended by Ant automatically. Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where srssafe.ini resides. No label A label to apply to the hierarchy Yes version An existing file or project version to label. By default the current version is labelled. No comment The comment to use for this label. Empty or '-' for no comment. No

autoresponse What to respond with (sets the -I option). By default, -I- is used; values of Y or N will be appended to this. No

Examples

```
<vsslabel vsspath="/source/aProject"
          login="me,mypassword"
          label="Release1"/>
```

Labels the current version of the VSS project $/source/aProject with the label Release1 using the username me and the password mypassword.

```
<vsslabel vsspath="/source/aProject/myfile.txt"
          version="4"
          label="1.03.004"/>
```

Labels version 4 of the VSS file $/source/aProject/myfile.txt with the label 1.03.004. If this version already has a label, the operation (and the build) will fail.

VssHistory Description Task to perform HISTORY commands to Microsoft Visual SourceSafe. Parameters Attribute Values Required vsspath SourceSafe path which specifies the project/file(s) you wish to perform the action on. You should not specify the leading dollar-sign - it is prepended by Ant automatically. Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where srssafe.ini resides. No fromDate Start date for comparison See below toDate End date for comparison See below dateFormat Format of dates in fromDate and toDate. Used when calculating dates with the numdays attribute. This string uses the formatting rules of SimpleDateFormat. Defaults to DateFormat.SHORT. No fromLabel Start label for comparison No toLabel Start label for comparison No numdays The number of days for comparison. See below output File to write the diff. No recursive true or false No style brief, codediff, default or nofile. The default is default. No user Name the user whose changes we would like to see No Specifying the time-frame

There are different ways to specify what time-frame you wish to evaluate:

* Changes between two dates: Specify both fromDate and toDate * Changes before a date: Specify toDate * Changes after a date: Specify fromDate * Changes X Days before a date: Specify toDate and (negative!) numDays * Changes X Days after a date: Specify fromDate and numDays

Examples

```
<vsshistory vsspath="/myProject" recursive="true"
            fromLabel="Release1"
            toLabel="Release2"/>
```

Shows all changes between "Release1" and "Release2".

```
<vsshistory vsspath="/myProject" recursive="true"
            fromDate="01.01.2001"
            toDate="31.03.2001"/>
```

Shows all changes between January 1st 2001 and March 31st 2001 (in Germany, date must be specified according to your locale).

```
<tstamp>
  <format property="to.tstamp" pattern="M-d-yy;h:mma" />
</tstamp>

<vsshistory vsspath="/myProject" recursive="true"
            numDays="-14"
            dateFormat="M-d-yy;h:mma"
            toDate="${to.tstamp}"/>
```

Shows all changes in the 14 days before today. VssCheckin Description Task to perform CHECKIN commands to Microsoft Visual SourceSafe. Parameters Attribute Values Required vsspath SourceSafe path which specifies the project/file(s) you wish to perform the action on. You should not specify the leading dollar-sign - it is prepended by Ant automatically. Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No localpath Override the working directory and get to the specified path No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where srssafe.ini resides. No writable true or false No recursive true or false No comment Comment to use for the files that where checked in. No autoresponse 'Y', 'N' or empty. Specify how to reply to questions from VSS. No

Examples

```
<vsscheckin vsspath="/test/test*"
            localpath="D:\build\"
            comment="Modified by automatic build"/>
```

Checks in the file(s) named test* in the project test using the local directory D: build.

VssCheckout Description Task to perform CHECKOUT commands to Microsoft Visual SourceSafe.

If you specify two or more attributes from version, date and label only one will be used in the order version, date, label.

Parameters Attribute Values Required vsspath SourceSafe path which specifies the project/file(s) you wish to perform the action on. You should not specify the leading dollar-sign - it is prepended by Ant automatically. Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No localpath Override the working directory and get to the specified path No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where srssafe.ini resides. No writable true or false No recursive true or false No version a version number to get No, only one of these allowed date a date stamp to get at label a label to get for

Examples

```
<vsscheckout vsspath="/test"
             localpath="D:\build"
             recursive="true"
             login="me,mypass"/>
```

Does a recursive checkout of the project test to the directory D:
build. VssAdd Description Task to perform ADD commands to Microsoft Visual SourceSafe. Parameters Attribute Values Required localpath Specify the local file(s) to add to VSS Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where srssafe.ini resides. No writable true or false No recursive true or false No comment Comment to use for the files that where checked in. No autoresponse 'Y', 'N' or empty. Specify how to reply to questions from VSS. No

Examples

```
<vssadd localpath="D:\build\build.00012.zip"
            comment="Added by automatic build"/>
```

Add the file named build.00012.zip into the project current working directory (see vsscp).

VssCp Description

Task to perform CP (Change Project) commands to Microsoft Visual SourceSafe.

This task is typically used before a VssAdd in order to set the target project Parameters Attribute Values Required vsspath SourceSafe path which specifies the project you wish to make the current project. You should not specify the leading dollar-sign - it is prepended by Ant automatically. Yes login username[,password] - The username and password needed to get access to VSS. Note that you may need to specify both (if you have a password) - Ant/VSS will hang if you leave the password out and VSS does not accept login without a password. No ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No serverPath directory where srssafe.ini resides. No

Examples

```
<vsscp vsspath="/Projects/ant"/>
```

Sets the current VSS project to $/Projects/ant. VssCreate Description Task to perform CREATE commands to Microsoft Visual Source Safe.

Creates a new project in VSS. Parameters Attribute Values Required login username,password No vsspath SourceSafe path of project to be created Yes ssdir directory where ss.exe resides. By default the task expects it to be in the PATH. No quiet suppress output (off by default) No failOnError fail if there is an error creating the project (true by default) No autoresponse What to respond with (sets the -I option). By default, -I- is used; values of Y or N will be appended to this. No comment The comment to use for this label. Empty or '-' for no comment. No

Examples

```
<vsscreate vsspath="/existingProject/newProject"/>
```

Creates the VSS-Project $/existingProject/newProject.

5.3.50 Weblogic JSP Compiler

Description

Class to precompile JSP's using weblogic's jsp compiler (weblogic.jspc) Tested only on Weblogic 4.5.1 - NT4.0 and Solaris 5.7,5.8 Parameters Attribute Values Required src root of source tree for JSP, ie, the document root for your weblogic server Yes dest root of destination directory, what you have set as WorkingDir in the weblogic properties Yes package start package name under which your JSP's would be compiled Yes classpath Class path to use when compiling jsp's Yes

A classpath should be set which contains the weblogic classes as well as all application classes referenced by the JSP. The system classpath is also appended when the jspc is called, so you may choose to put everything in the classpath while calling Ant. However, since presumably the JSP's will reference classes being build by Ant, it would be better to explicitly add the classpath in the task

The task checks timestamps on the JSP's and the generated classes, and compiles only those files that have changed.

It follows the weblogic naming convention of putting classes in _dirName/_fileName.class for dirname/fileName.jsp

Example

```
<target name="jspcompile" depends="compile">
  <wljspc src="c:\\weblogic\\myserver\\public_html"
    dest="c:\\weblogic\\myserver\\serverclasses" package="myapp.jsp">
    <classpath>
      <pathelement location="${weblogic.classpath}"/>
      <pathelement path="${compile.dest}"/>
    </classpath>
  </wljspc>
</target>
```

Limitations

* This works only on weblogic 4.5.1 * It compiles the files thru the Classic compiler only. * Since it is my experience that weblogic jspc throws out of memory error on being given too many files at one go, it is called multiple times with one jsp file each.

5.3.51 XmlValidate

Description

This task checks xml files are valid (or only well formed). The task uses the SAX2 parser implementation provided by JAXP by default (probably the one that is used by Ant itself), but one can specify any SAX1/2 parser if needed.

This task supports the use of nested xmlcatalog elements and/or nested ¡dtd¿ elements which are used to resolve DTDs and entities. Parameters Attribute Description Required file the file(s) you want to check. (optionally can use an embedded fileset) No lenient if true, only check the xml document is well formed (ignored if the specified parser is as SAX1 parser) No classname the parser to use. No classpathref where to find the parser class. Optionally can use an embedded classpath element. No failonerror fails on a error if set to true (defaults to true). No warn log parser warn events. No Nested Elements dtd ¡dtd¿ is used to specify different locations for DTD resolution. Attribute Description Required publicId Public ID of the DTD to resolve Yes location Location of the DTD to use, which can be a file, a resource, or a URL Yes xmlcatalog

The xmlcatalog element is used to perform Entity resolution.

Examples

```
<xmlvalidate file="toto.xml"/>

<xmlvalidate failonerror="no" lenient="yes" warn="yes"
             classname="org.apache.xerces.parsers.SAXParser">
             classpath="lib/xerces.jar">
  <fileset dir="src" includes="style/*.xsl"/>
</xmlvalidate>

<xmlvalidate file="struts-config.xml" warn="false">
  <dtd publicId="-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
       location="struts-config_1_0.dtd"/>
</xmlvalidate>

<xmlvalidate failonerror="no">
  <fileset dir="${project.dir}" includes="**/*.xml"/>
  <xmlcatalog refid="mycatalog"/>
</xmlvalidate>

<xmlvalidate failonerror="no">
  <fileset dir="${project.dir}" includes="**/*.xml"/>
  <xmlcatalog>
       <dtd
         publicId="-//ArielPartners//DTD XML Article V1.0//EN"
         location="com/arielpartners/knowledgebase/dtd/article.dtd"/>
  </xmlcatalog>
</xmlvalidate>
```

Chapter 6

Concepts and Types

6.1 Concepts

6.1.1 build.sysclasspath

The value of the build.sysclasspath property control how the system classpath, ie. the classpath in effect when Ant is run, affects the behaviour of classpaths in Ant. The default behavior varies from Ant to Ant task.

The values and their meanings are:

only	Only the system classpath is used and classpaths specified in build files, etc are ignored. This situation could be considered as the person running the build file knows more about the environment than the person writing the build file
ignore	The system classpath is ignored. This situation is the reverse of the above. The person running the build trusts the build file writer to get the build file right
last	The classpath is concatenated to any specified classpaths at the end. This is a compromise, where the build file writer has priority.
first	Any specified classpaths are concatenated to the system classpath. This is the other form of compromise where the build runner has priority.

6.1.2 Common Attributes of all Tasks

All tasks share the following attributes:

Attribute	Description	Required
id	Unique identifier for this task instance, can be used to reference this task in scripts.	No
taskname	A different name for this task instance - will show up in the logging output.	No
description	Room for your comments	No

6.2 Core Types

6.2.1 Description

Description

Allows for a description of the project to be specified that will be included in the output of the ant -projecthelp command.

Parameters

(none)

Examples

```
<description>
This buildfile is used to build the Foo subproject within
the large, complex Bar project.
</description>
```

6.2.2 Directory-based Tasks

Some tasks use directory trees for the actions they perform. For example, the javac task, which compiles a directory tree with .java files into `.class` files, is one of these directory-based tasks. Because some of these tasks do so much work with a directory tree, the task itself can act as an implicit FileSet.

Whether the fileset is implicit or not, it can often be very useful to work on a subset of the directory tree. This section describes how you can select a subset of such a directory tree when using one of these directory-based tasks.

Ant gives you two ways to create a subset of files in a fileset, both of which can be used at the same time:

1. Only include files and directories that match any include patterns and do not match any exclude patterns in a given PatternSet.

2. Select files based on selection criteria defined by a collection of selector nested elements.

Patternset

We said that Directory-based tasks can sometimes act as an implicit `<fileset>`, but in addition to that, a FileSet acts as an implicit `<patternset>`.

The inclusion and exclusion elements of the implicit PatternSet can be specified inside the directory-based task (or explicit fileset) via either:

- the attributes includes and excludes.
- nested elements `<include>` and `<exclude>`.
- external files specified with the attributes includesfile and excludesfile.
- external files specified with the nested elements `<includesfile>` and `<excludesfile>`.

When dealing with an external file, each line of the file is taken as a pattern that is added to the list of include or exclude patterns.

When both inclusion and exclusion are used, only files/directories that match at least one of the include patterns and don’t match any of the exclude patterns are used. If no include pattern is given, all files are assumed to match the include pattern (with the possible exception of the default excludes).

Patterns

As described earlier, patterns are used for the inclusion and exclusion of files. These patterns look very much like the patterns used in DOS and UNIX:

’*’ matches zero or more characters, ’?’ matches one character.

Examples:

*.java matches .java, x.java and FooBar.java, but not FooBar.xml (does not end with .java).

?.java matches x.java, A.java, but not .java or xyz.java (both don’t have one character before .java).

Combinations of *’s and ?’s are allowed.

Matching is done per-directory. This means that first the first directory in the pattern is matched against the first directory in the path to match. Then the second directory is matched, and so on. For example, when we have the pattern /?abc/*/*.java and the path /xabc/foobar/test.java, the first ?abc is matched with xabc, then * is matched with foobar, and finally *.java is matched with test.java. They all match, so the path matches the pattern.

To make things a bit more flexible, we add one extra feature, which makes it possible to match multiple directory levels. This can be used to match a complete directory tree, or a file anywhere in the directory tree. To do this, ** must be used as the name of a directory. When ** is used as the name of a directory in the pattern, it matches zero or more directories. For example: /test/** matches all files/directories under /test/, such as /test/x.java, or /test/foo/bar/xyz.html, but not /xyz.xml.

There is one ”shorthand” - if a pattern ends with / or then ** is appended. For example, mypackage/test/ is interpreted as if it were mypackage/test/**.

Example patterns

**/CVS/*	Matches all files in CVS directories that can be located anywhere in the directory tree. Matches: CVS/Repository org/apache/CVS/Entries org/apache/jakarta/tools/ant/CVS/Entries But not: org/apache/CVS/foo/bar/Entries (foo/bar/ part does not match)
org/apache/jakarta/**	Matches all files in the org/apache/jakarta directory tree. Matches: org/apache/jakarta/tools/ant/docs/index.html org/apache/jakarta/test.xml But not: org/apache/xyz.java (jakarta/ part is missing).
org/apache/**/CVS/*	Matches all files in CVS directories that are located anywhere in the directory tree under org/apache. Matches: org/apache/CVS/Entries org/apache/jakarta/tools/ant/CVS/Entries But not: org/apache/CVS/foo/bar/Entries (foo/bar/ part does not match)
hline **/test/**	Matches all files that have a test element in their path, including test as a filename.

When these patterns are used in inclusion and exclusion, you have a powerful way to select just the files you want.

Selectors

The `<fileset>`, whether implicit or explicit in the directory-based task, also acts as an `<and>` selector container. This can be used to create arbitrarily complicated selection criteria for the files the task should work with. See the

Selector documentation for more information.

Standard Tasks/Filesets

Many of the standard tasks in ant take one or more filesets which follow the rules given here. This list, a subset of those, is a list of standard ant tasks that can act as an implicit fileset:

```
<checksum>
<copydir> (deprecated)
<delete>
<dependset>
<fixcrlf>
<javac>
<replace>
<rmic>
<style> (aka <xslt>)
<tar>
<zip>
<ddcreator>
<ejbjar>
<ejbc>
<cab>
<icontract>
<native2ascii>
<netrexxc>
<renameextensions>
<depend>
<ilasm>
<csc>
<vbc>
<translate>
<vajexport>
<image>
<jlink> (deprecated)
<jspc>
<wljspc>
```

Examples

```
<copy todir="${dist}">
  <fileset dir="${src}"
           includes="**/images/*"
           excludes="**/*.gif"
  />
</copy>
```

This copies all files in directories called images that are located in the directory tree defined by ${src} to the destination directory defined by ${dist}, but excludes all *.gif files from the copy.

```
<copy todir="${dist}">
  <fileset dir="${src}">
    <include name="**/images/*"/>
    <exclude name="**/*.gif"/>
  </fileset>
</copy>
```

The same as the example above, but expressed using nested elements.

```
<delete dir="${dist}">
    <include name="**/images/*"/>
    <exclude name="**/*.gif"/>
</delete>
```

Deleting the original set of files, the delete task can act as an implicit fileset.

Default Excludes There are a set of definitions that are excluded by default from all directory-based tasks. They are:

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

If you do not want these default excludes applied, you may disable them with the defaultexcludes=”no” attribute.

6.2.3 DirSet

DirSets are groups of directories. These directories can be found in a directory tree starting in a base directory and are matched by patterns taken from a number of PatternSets. DirSets can appear inside tasks that support this feature or at the same level as target (i.e., as children of `<project>`).

PatternSets can be specified as nested `<patternset>` elements. In addition, DirSet holds an implicit PatternSet and supports the nested `<include>`,

`<includesfile>`, `<exclude>` and `<excludesfile>` elements of `<patternset>` directly, as well as `<patternset>`'s attributes.

Attribute	Description	Required
dir	The root of the directory tree of this DirSet.	Yes
includes	A comma- or space-separated list of patterns of directories that must be included; all directories are included when omitted.	No
includesfile	The name of a file; each line of this file is taken to be an include pattern.	No
excludes	A comma- or space-separated list of patterns of directories that must be excluded; no directories are excluded when omitted.	No
excludesfile	The name of a file; each line of this file is taken to be an exclude pattern.	No
casesensitive	Specifies whether case-sensitivty should be applied (true—yes—on or false—no—off).	No; defaults to true.
followsymlinks	Shall symbolic links be followed? Defaults to true. See fileset's documentation.	No

Examples

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

Groups all directories named classes found under the apps subdirectory of `${build.dir}`, except those that have the text Test in their name.

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

Groups the same directories as the above example, but also establishes a PatternSet that can be referenced in other `<dirset>` elements, rooted at a different directory.

```
<dirset dir="${debug_build.dir}">
  <patternset refid="non.test.classes"/>
</dirset>
```

Groups all directories in directory `${debug_build.dir}`, using the same patterns as the above example.

6.2.4 FileList

FileLists are explicitly named lists of files. Whereas FileSets act as filters, returning only those files that exist in the file system and match specified patterns, FileLists are useful for specifying files that may or may not exist. Multiple files are specified as a list of files, relative to the specified directory, with no support for wildcard expansion (filenames with wildcards will be included in the list unchanged). FileLists can appear inside tasks that support this feature or at the same level as `<target>` (i.e., as children of `<project>`).

Attribute	Description	Required
dir	The base directory of this FileList.	Yes
files	The list of file names.	Yes

Examples

```
<filelist
    id="docfiles"
    dir="${doc.src}"
    files="foo.xml,bar.xml"/>
```

The files *doc.src*/*foo.xmland*doc.src/bar.xml. Note that these files may not (yet) actually exist.

```
<filelist
    id="docfiles"
    dir="${doc.src}"
    files="foo.xml
           bar.xml"/>
```

Same files as the example above.

```
<filelist refid="docfiles"/>
```

Same files as the example above.

6.2.5 FileSet

FileSets are groups of files. These files can be found in a directory tree starting in a base directory and are matched by patterns taken from a number of PatternSets and Selectors. FileSets can appear inside tasks that support this feature or at the same level as target - i.e., as children of project.

PatternSets can be specified as nested `<patternset>` elements. In addition, FileSet holds an implicit PatternSet and supports the nested `<include>`, `<includesfile>`, `<exclude>` and `<excludesfile>` elements of PatternSet directly, as well as PatternSet's attributes.

Selectors are available as nested elements.within the FileSet. If any of the selectors within the FileSet do not select the file, the file is not considered part of the FileSet. This makes FileSets euqivalent to an `<and>` selector container.

Attribute	Description	Required
dir	the root of the directory tree of this FileSet.	Yes
defaultexcludes	indicates whether default excludes should be used or not (yes — no); default excludes are used when omitted.	No
includes	comma- or space-separated list of patterns of files that must be included; all files are included when omitted.	No
includesfile	the name of a file; each line of this file is taken to be an include pattern.	No
excludes	comma- or space-separated list of patterns of files that must be excluded; no files (except default excludes) are excluded when omitted.	No
excludesfile	the name of a file; each line of this file is taken to be an exclude pattern.	No
casesensitive	Must the file system be treated in a case sensitive way? Defaults to true.	No
followsymlinks	Shall symbolic links be followed? Defaults to true. See the note below.	No

Note: All files/directories for which the canonical path is different from its path are considered symbolic links. On Unix systems this usually means the file really is a symbolic links but it may lead to false results on other platforms.

Examples

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

Groups all files in directory `${server.src}` that are Java source files and don't have the text Test in their name.

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

Groups the same files as the above example, but also establishes a PatternSet that can be referenced in other `<fileset>` elements, rooted at a different directory.

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

Groups all files in directory `${client.src}`, using the same patterns as the above example.

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

Groups the same files as the top example, but using the `<filename>` selector.

```
<fileset dir="${server.src}" casesensitive="yes">
  <filename name="**/*.java"/>
  <not>
    <filename name="**/*Test*"/>
  </not>
</fileset>
```

Groups the same files as the previous example using a combination of the `<filename>` selector and the ¡not¿ selector container.

6.2.6 File Mappers

Some tasks take source files and create target files. Depending on the task, it may be quite obvious which name a target file will have (using javac, you know there will be .class files for your .java files) - in other cases you may want to specify the target files, either to help Ant or to get an extra bit of functionality.

While source files are usually specified as filesets, you don't specify target files directly - instead, you tell Ant how to find the target file(s) for one source file. An instance of org.apache.tools.ant.util.FileNameMapper is responsible for this. It constructs target file names based on rules that can be parameterized with from and to attributes - the exact meaning of which is implementation-dependent.

These instances are defined in `<mapper>` elements with the following attributes:

Attribute	Description	Required
type	specifies one of the built-in implementations.	Exactly one of both
classname	specifies the implementation by class name.	Exactly one of both
classpath	the classpath to use when looking up classname.	No
classpathref	the classpath to use, given as reference to a path defined elsewhere.	No
from	the from attribute for the given implementation.	Depends on implementation.
to	the to attribute for the given implementation.	Depends on implementation.

Note that Ant will not automatically convert / or \ characters in the to and from attributes to the correct directory separator of your current platform. If you need to specify this separator, use `${file.separator}` instead.

Parameters specified as nested elements

The classpath can be specified via a nested `<classpath>`, as well - that is, a path-like structure.

The built-in mapper types are:

identity The target file name is identical to the source file name. Both to and from will be ignored.

Examples:

```
<mapper type="identity"/>
```

Source file name	Target file name
A.java	A.java
foo/bar/B.java	foo/bar/B.java
C.properties	C.properties
Classes/dir/dir2/A.properties	Classes/dir/dir2/A.properties

flatten The target file name is identical to the source file name, with all leading directory information stripped off. Both to and from will be ignored.

Examples:

```
<mapper type="flatten"/>
```

Source file name	Target file name
A.java	A.java
foo/bar/B.java	B.java
C.properties	C.properties
Classes/dir/dir2/A.properties	A.properties

merge The target file name will always be the same, as defined by to - from will be ignored.

Examples:

```
<mapper type="merge" to="archive.tar"/>
```

Source file name	Target file name
A.java	archive.tar
foo/bar/B.java	archive.tar
C.properties	archive.tar
Classes/dir/dir2/A.properties	archive.tar

glob Both to and from define patterns that may contain at most one *. For each source file that matches the from pattern, a target file name will be constructed from the to pattern by substituting the * in the to pattern with the text that matches the * in the from pattern. Source file names that don't match the from pattern will be ignored.

Examples:

```
<mapper type="glob" from="*.java" to="*.java.bak"/>
```

Source file name	Target file name
A.java	A.java.bak
foo/bar/B.java	foo/bar/B.java.bak
C.properties	ignored
Classes/dir/dir2/A.properties	ignored

```
<mapper type="glob" from="C*ies" to="Q*y"/>
```

Source file name	Target file name
A.java	ignored
foo/bar/B.java	ignored
C.properties	Q.property
Classes/dir/dir2/A.properties	Qlasses/dir/dir2/A.property

regexp Both to and from define regular expressions. If the source file name matches the from pattern, the target file name will be constructed from the to pattern, using \0 to \9 as back-references for the full match (\0) or the matches of the subexpressions in parentheses. Source files not matching the from pattern will be ignored.

Note that you need to escape a dollar-sign ($) with another dollar-sign in Ant.

The regexp mapper needs a supporting library and an implementation of org.apache.tools.ant.util.regexp.RegexpMatcher that hides the specifics of the library. Ant comes with implementations for the java.util.regex package of JDK 1.4, jakarta-regexp and jakarta-ORO. If you compile from sources and plan to use one of them, make sure the libraries are in your CLASSPATH. For information about using gnu.regexp or gnu.rex with Ant, see this article.

This means, you need optional.jar from the Ant release you are using and one of the supported regular expression libraries. Make sure, both will be loaded from the same classpath, that is either put them into your CLASSPATH, `ANT_HOME`/lib directory or a nested `<classpath>` element of the mapper - you cannot have optional.jar in `ANT_HOME`/lib and the library in a nested ¡classpath¿.

Ant will choose the regular-expression library based on the following algorithm:

If the system property `ant.regexp.matcherimpl` has been set, it is taken as the name of the class implementing org.apache.tools.ant.util.regexp.RegexpMatcher that should be used.

If it has not been set, first try the JDK 1.4 classes, then jakarta-ORO and finally try jakarta-regexp.

Examples:

```
<mapper type="regexp" from="^(.*)\.java$$" to="\1.java.bak"/>
```

Source file name	Target file name
A.java A.	java.bak
foo/bar/B.java	foo/bar/B.java.bak
C.properties	ignored
Classes/dir/dir2/A.properties	ignored

```
<mapper type="regexp" from="^(.*)/([^/]+)/([^/]*)$$" to="\1/\2/\2-\3"/>
```

Source file name	Target file name
A.java	ignored
foo/bar/B.java	foo/bar/bar-B.java
C.properties	ignored
Classes/dir/dir2/A.properties	Classes/dir/dir2/dir2-A.properties

```
<mapper type="regexp" from="^(.*)\.(.*)$$" to="\2.\1"/>
```

Source file name	Target file name
A.java	java.A
foo/bar/B.java	java.foo/bar/B
C.properties	properties.C
Classes/dir/dir2/A.properties	properties.Classes/dir/dir2/A

package

Sharing the same syntax as the glob mapper, the package mapper replaces directory separators found in the matched source pattern with dots in the target pattern placeholder. This mapper is particularly useful in combination with `<uptodate>` and `<junit>` output.

Example:

```
<mapper type="package"
        from="*Test.java" to="TEST-*Test.xml"/>
```

Source file name Target file name
org/apache/tools/ant/util/PackageMapperTest.java TEST-org.apache.tools.ant.util.PackageMapperTest.xml
org/apache/tools/ant/util/Helper.java ignored

6.2.7 Filter Chains and Filter Readers

Look at Unix pipes - they offer you so much flexibility - say you wanted to copy just those lines that contained the string `blee` from the first 10 lines of a file 'foo' to a file 'bar' - you would do something like

```
cat foo|head -n10|grep blee > bar
```

Ant was not flexible enough. There was no way for the `<copy>` task to do something similar. If you wanted the `<copy>` task to get the first 10 lines, you would have had to create special attributes:

```
<copy file="foo" tofile="bar" head="10" contains="blee"/>
```

The obvious problem thus surfaced: Ant tasks would not be able to accomodate such data transformation attributes as they would be endless. The task would also not know in which order these attributes were to be interpreted. That is, must the task execute the contains attribute first and then the head attribute or vice-versa? What Ant tasks needed was a mechanism to allow pluggable filter (data tranformer) chains. Ant would provide a few filters for which there have been repeated requests. Users with special filtering needs would be able to easily write their own and plug them in.

The solution was to refactor data transformation oriented tasks to support FilterChains. A FilterChain is a group of ordered FilterReaders. Users can define their own FilterReaders by just extending the java.io.FilterReader class. Such custom FilterReaders can be easily plugged in as nested elements of `<filterchain>` by using `<filterreader>` elements.

Example:

```
<copy file="${src.file}" tofile="${dest.file}">
  <filterchain>
    <filterreader classname="your.extension.of.java.io.FilterReader">
      <param name="foo" value="bar"/>
    </filterreader>
    <filterreader classname="another.extension.of.java.io.FilterReader">
      <classpath>
        <pathelement path="${classpath}"/>
      </classpath>
      <param name="blah" value="blee"/>
      <param type="abra" value="cadabra"/>
    </filterreader>
  </filterchain>
</copy>
```

Ant provides some built-in filter readers. These filter readers can also be declared using a syntax similar to the above syntax. However, they can be declared using some simpler syntax also.

Example:

```
<loadfile srcfile="${src.file}" property="${src.file.head}">
  <filterchain>
    <headfilter lines="15"/>
  </filterchain>
</loadfile>
```

is equivalent to:

```
<loadfile srcfile="${src.file}" property="${src.file.head}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.HeadFilter">
      <param name="lines" value="15"/>
    </filterreader>
  </filterchain>
</loadfile>
```

The following built-in tasks support nested `<filterchain>` elements.

```
Copy,
LoadFile,
LoadProperties,
Move
```

A FilterChain is formed by defining zero or more of the following nested elements. FilterReader ClassConstants ExpandProperties HeadFilter LineContains LineContainsRegExp PrefixLines ReplaceTokens StripJavaComments StripLineBreaks StripLineComments TabsToSpaces TailFilter

FilterReader The filterreader element is the generic way to define a filter. User defined filter elements are defined in the build file using this. Please note that built in filter readers can also be defined using this syntax. A FilterReader element must be supplied with a class name as an attribute value. The class resolved by this name must extend java.io.FilterReader. If the custom filter reader needs to be parameterized, it must implement org.apache.tools.type.Parameterizable. Attribute Description Required classname The class name of the filter reader. Yes

Nested Elements: `<filterreader>` supports `<classpath>` and `<param>` as nested elements. Each `<param>` element may take in the following attributes - name, type and value.

The following FilterReaders are supplied with the default distribution.

ClassConstants This filters basic constants defined in a Java Class, and outputs them in lines composed of the format name=value

Example: This loads the basic constants defined in a Java class as Ant properties.

```
<loadproperties srcfile="foo.class">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.ClassConstants"/>
  </filterchain>
</loadproperties>
```

Convenience method:

```
<loadproperties srcfile="foo.class">
  <filterchain>
    <classconstants/>
  </filterchain>
</loadproperties>
```

ExpandProperties If the data contains data that represents Ant properties (of the form ${...}), that is substituted with the property's actual value.

Example: This results in the property modifiedmessage holding the value "All these moments will be lost in time, like teardrops in the rain"

```
<echo
  message="All these moments will be lost in time, like teardrops in the ${weather}
  file="loadfile1.tmp"
  />
<property name="weather" value="rain" />
<loadfile property="modifiedmessage" srcFile="loadfile1.tmp">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.ExpandProperties"/>
  </filterchain>
</loadfile>
```

Convenience method:

```
<echo
  message="All these moments will be lost in time, like teardrops in the ${weather}
  file="loadfile1.tmp"
  />
<property name="weather" value="rain" />
<loadfile property="modifiedmessage" srcFile="loadfile1.tmp">
  <filterchain>
    <expandproperties/>
  </filterchain>
</loadfile>
```

HeadFilter This filter reads the first few lines from the data supplied to it. Parameter Name Parameter Value Required lines Number of lines to be read. Defaults to "10" No

Example:

This stores the first 15 lines of the supplied data in the property `${src.file.head}`

```
<loadfile srcfile="${src.file}" property="${src.file.head}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.HeadFilter">
      <param name="lines" value="15"/>
    </filterreader>
  </filterchain>
</loadfile>
```

Convenience method:

```
<loadfile srcfile="${src.file}" property="${src.file.head}">
  <filterchain>
    <headfilter lines="15"/>
  </filterchain>
</loadfile>
```

LineContains This filter includes only those lines that contain all the user-specified strings. Parameter Type Parameter Value Required contains Substring to be searched for. Yes

Example: This will include only those lines that contain foo and bar.

```
<filterreader classname="org.apache.tools.ant.filters.LineContains">
  <param type="contains" value="foo"/>
  <param type="contains" value="bar"/>
</filterreader>
```

Convenience method:

```
<linecontains>
  <contains value="foo">
  <contains value="bar">
</linecontains>
```

LineContainsRegExp Filter which includes only those lines that contain the user-specified regular expression matching strings. Parameter Type Parameter Value Required regexp Pattern of the substring to be searched for. Yes

Example: This will fetch all those lines that contain the pattern foo

```
<filterreader classname="org.apache.tools.ant.filters.LineContainsRegExp">
  <param type="regexp" value="foo*"/>
</filterreader>
```

Convenience method:

```
<linecontainsregexp>
  <regexp pattern="foo*">
</linecontainsregexp>
```

PrefixLines Attaches a prefix to every line. Parameter Name Parameter Value Required prefix Prefix to be attached to lines. Yes

Example: This will attach the prefix Foo to all lines.

```
<filterreader classname="org.apache.tools.ant.filters.PrefixLines">
  <param name="prefix" value="Foo"/>
</filterreader>
```

Convenience method:

```
<prefixlines prefix="Foo"/>
```

ReplaceTokens This filter reader replaces all strings that are sandwiched between begintoken and endtoken with user defined values. Parameter Type Parameter Name Parameter Value Required tokenchar begintoken Character marking the beginning of a token. Defaults to @ No tokenchar endtoken Character marking the end of a token. Defaults to @ No token User defined String. User defined search String Yes

Example: This replaces occurences of the string @DATE@ in the data with today's date and stores it in the property ${src.file.replaced}

```
<tstamp/>
<loadfile srcfile="${src.file}" property="${src.file.replaced}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.ReplaceTokens">
      <param type="token" name="DATE" value="${TODAY}"/>
    </filterreader>
  </filterchain>
</loadfile>
```

Convenience method:

```
<tstamp/>
<loadfile srcfile="${src.file}" property="${src.file.replaced}">
  <filterchain>
    <replacetokens>
      <token key="DATE" value="${TODAY}"/>
    </replacetokens>
  </filterchain>
</loadfile>
```

StripJavaComments This filter reader strips away comments from the data, using Java syntax guidelines. This filter does not take in any parameters.

Example:

```
<loadfile srcfile="${java.src.file}" property="${java.src.file.nocomments}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.StripJavaComments"/>
  </filterchain>
</loadfile>
```

Convenience method:

```
<loadfile srcfile="${java.src.file}" property="${java.src.file.nocomments}">
  <filterchain>
    <stripjavacomments/>
  </filterchain>
</loadfile>
```

StripLineBreaks This filter reader strips away specific characters from the data supplied to it. Parameter Name Parameter Value Required linebreaks Characters that are to be stripped out. Defaults to "\r\n" No

Examples: This strips the '\r' and '\n' characters.

```
<loadfile srcfile="${src.file}" property="${src.file.contents}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.StripLineBreaks"/>
  </filterchain>
</loadfile>
```

Convenience method:

```
<loadfile srcfile="${src.file}" property="${src.file.contents}">
  <filterchain>
    <striplinebreaks/>
  </filterchain>
</loadfile>
```

This treats the '(' and ')' characters as line break characters and strips them.

```
<loadfile srcfile="${src.file}" property="${src.file.contents}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.StripLineBreaks">
      <param name="linebreaks" value="()"/>
    </filterreader>
  </filterchain>
</loadfile>
```

StripLineComments This filter removes all those lines that begin with strings that represent comments as specified by the user. Parameter Type Parameter Value Required comment Strings that identify a line as a comment when they appear at the start of the line. Yes

Examples: This removes all lines that begin with #, –, REM, rem and //

```
<filterreader classname="org.apache.tools.ant.filters.StripLineComments">
  <param type="comment" value="#"/>
  <param type="comment" value="--"/>
  <param type="comment" value="REM "/>
  <param type="comment" value="rem "/>
  <param type="comment" value="//"/>
</filterreader>
```

Convenience method:

```
<striplinecomments>
  <comment value="#"/>
  <comment value="--"/>
  <comment value="REM "/>
  <comment value="rem "/>
  <comment value="//"/>
</striplinecomments>
```

TabsToSpaces This filter replaces tabs with spaces Parameter Name Parameter Value Required lines tablength Defaults to ”8” No

Examples: This replaces tabs in `${src.file}` with spaces.

```
<loadfile srcfile="${src.file}" property="${src.file.notab}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.TabsToSpaces"/>
  </filterchain>
</loadfile>
```

Convenience method:

```
<loadfile srcfile="${src.file}" property="${src.file.notab}">
  <filterchain>
    <tabstospaces/>
  </filterchain>
</loadfile>
```

TailFilter This filter reads the last few lines from the data supplied to it. Parameter Name Parameter Value Required lines Number of lines to be read. Defaults to ”10” No

Examples: This stores the last 15 lines of the supplied data in the property ${src.file.tail}

```
<loadfile srcfile="${src.file}" property="${src.file.tail}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.TailFilter">
      <param name="lines" value="15"/>
    </filterreader>
  </filterchain>
</loadfile>
```

Convenience method:

```
<loadfile srcfile="${src.file}" property="${src.file.tail}">
  <filterchain>
    <tailfilter lines="15"/>
  </filterchain>
</loadfile>
```

This stores the last 5 lines of the first 15 lines of the supplied data in the property ${src.file.mid}

```
<loadfile srcfile="${src.file}" property="${src.file.mid}">
  <filterchain>
    <filterreader classname="org.apache.tools.ant.filters.HeadFilter">
      <param name="lines" value="15"/>
    </filterreader>
    <filterreader classname="org.apache.tools.ant.filters.TailFilter">
```

```
      <param name="lines" value="5"/>
    </filterreader>
  </filterchain>
</loadfile>
```

Convenience method:

```
<loadfile srcfile="${src.file}" property="${src.file.mid}">
  <filterchain>
    <headfilter lines="15"/>
    <tailfilter lines="5"/>
  </filterchain>
</loadfile>
```

6.2.8 FilterSet

FilterSets are groups of filters. Filters can be defined as token-value pairs or be read in from a file. FilterSets can appear inside tasks that support this feature or at the same level as `<target>` - i.e., as children of `<project>`.

FilterSets support the id and refid attributes. You can define a FilterSet with an id attribute and then refer to that definition from another FilterSet with a refid attribute. It is also possible to nest filtersets into filtersets to get a set union of the contained filters.

In addition, FilterSets can specify begintoken and/or endtoken attributes to define what to match.

Filtersets are used for doing replacements in tasks such as `<copy>`, etc.

Filterset Attribute Description Default Required begintoken The string marking the beginning of a token (eg., @DATE@). @ No endtoken The string marking the end of a token (eg., @DATE@). @ No

Filter Attribute Description Required token The token to replace (eg., @DATE@) Yes value The value to replace it with (eg., Thursday, April 26, 2001). Yes

Filtersfile Attribute Description Required file A properties file of name-value pairs from which to load the tokens. Yes

Examples You are copying the version.txt file to the dist directory from the build directory but wish to replace the token @DATE@ with today's date.

```
<copy file="${build.dir}/version.txt" toFile="${dist.dir}/version.txt">
  <filterset>
    <filter token="DATE" value="${TODAY}"/>
  </filterset>
</copy>
```

You are copying the version.txt file to the dist directory from the build directory but wish to replace the token

```
<copy file="${build.dir}/version.txt" toFile="${dist.dir}/version.txt">
  <filterset begintoken="%" endtoken="*">
    <filter token="DATE" value="${TODAY}"/>
  </filterset>
</copy>
```

Copy all the docs but change all dates and appropriate notices as stored in a file.

```
<copy toDir="${dist.dir}/docs">
  <fileset dir="${build.dir}/docs">
    <include name="**/*.html">
  </fileset>
  <filterset begintoken="%" endtoken="*">
    <filtersfile file="${user.dir}/dist.properties"/>
  </filterset>
</copy>
```

Define a FilterSet and reference it later.

```
<filterset id="myFilterSet" begintoken="%" endtoken="*">
  <filter token="DATE" value="${TODAY}"/>
</filterset>

<copy file="${build.dir}/version.txt" toFile="${dist.dir}/version.txt">
  <filterset refid="myFilterSet"/>
</copy>
```

6.2.9 PatternSet

Patterns can be grouped to sets and later be referenced by their id attribute. They are defined via a patternset element, which can appear nested into a FileSet or a directory-based task that constitutes an implicit FileSet. In addition, patternsets can be defined as a stand alone element at the same level as target i.e., as children of project as well as as children of target.

Patterns can be specified by nested `<include>`, or `<exclude>` elements or the following attributes.

Attribute Description includes comma- or space-separated list of patterns of files that must be included. All files are included when omitted. includesfile the name of a file; each line of this file is taken to be an include pattern. You can specify more than one include file by using a nested includesfile elements. excludes comma- or space-separated list of patterns of files that must be excluded; no files (except default excludes) are excluded when omitted. excludesfile the name of a file; each line of this file is taken to be an exclude pattern. You can specify more than one exclude file by using a nested excludesfile elements.

Parameters specified as nested elements include and exclude Each such element defines a single pattern for files to include or exclude.

Attribute Description Required name the pattern to in/exclude. Yes if Only use this pattern if the named property is set. No unless Only use this pattern if the named property is not set. No includesfile and excludesfile If you want to list the files to include or exclude external to your build file, you should use the includesfile/excludesfile attributes or elements. Using the attribute, you can only specify a single file of each type, while the nested elements can be specified more than once - the nested elements also support if/unless attributes you can use to test the existance of a property.

Attribute Description Required name the name of the file holding the patterns to in/exclude. Yes if Only read this file if the named property is set. No unless Only read this file if the named property is not set. No

patternset Patternsets may be nested within one another, adding the nested patterns to the parent patternset.

Examples

```
<patternset id="non.test.sources">
  <include name="**/*.java"/>
  <exclude name="**/*Test*"/>
</patternset>
```

Builds a set of patterns that matches all .java files that do not contain the text Test in their name. This set can be referred to via `<patternset refid="non.test.sources"/>`, by tasks that support this feature, or by FileSets.

Note that while the includes and excludes attributes accept multiple elements separated by commas or spaces, the nested `<include>` and `<exclude>` elements expect their name attribute to hold a single pattern.

The nested elements allow you to use if and unless arguments to specify that the element should only be used if a property is set, or that it should be used only if a property is not set.

For example

```
<patternset id="sources">
  <include name="std/**/*.java"/>
  <include name="prof/**/*.java" if="professional"/>
  <exclude name="**/*Test*"/>
</patternset>
```

will only include the files in the sub-directory prof if the property professional is set to some value.

The two sets

```
<patternset includesfile="some-file"/>
```

and

```
<patternset>
  <includesfile name="some-file"/>
<patternset/>
```

are identical. The include patterns will be read from the file some-file, one pattern per line.

```
<patternset>
  <includesfile name="some-file"/>
  <includesfile name="${some-other-file}"
                if="some-other-file"
  />
<patternset/>
```

will also read include patterns from the file the property some-other-file points to, if a property of that name has been defined.

6.2.10 Path-like Structures

You can specify PATH- and CLASSPATH-type references using both ":" and ";" as separator characters. Ant will convert the separator to the correct character of the current operating system.

Wherever path-like values need to be specified, a nested element can be used. This takes the general form of:

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

The location attribute specifies a single file or directory relative to the project's base directory (or an absolute filename), while the path attribute accepts colon- or semicolon-separated lists of locations. The path attribute is intended to be used with predefined paths - in any other case, multiple elements with location attributes should be preferred.

As a shortcut, the `<classpath>` tag supports path and location attributes of its own, so:

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

can be abbreviated to:

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

In addition, DirSets, FileSets, and FileLists can be specified via nested `<dirset>`, `<fileset>`, and `<filelist>` elements, respectively.

Note: The order in which the files building up a FileSet are added to the path-like structure is not defined.

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

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

This builds a path that holds the value of ${classpath}, followed by all jar files in the lib directory, the classes directory, all directories named classes under the apps subdirectory of ${build.dir}, except those that have the text Test in their name, and the files specified in the referenced FileList.

If you want to use the same path-like structure for several tasks, you can define them with a `<path>` element at the same level as targets, and reference them via their id attribute - see References for an example.

A path-like structure can include a reference to another path-like structure via nested `<path>` elements:

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

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

The shortcuts previously mentioned for ¡classpath¿ are also valid for `<path>`. For example:

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

can be written as:

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

Command-line Arguments Several tasks take arguments that will be passed to another process on the command line. To make it easier to specify arguments that contain space characters, nested arg elements can be used.

Attribute Description Required value a single command-line argument; can contain space characters. Exactly one of these. file The name of a file as a single command-line argument; will be replaced with the absolute filename of the file.

path A string that will be treated as a path-like string as a single command-line argument; you can use ; or : as path separators and Ant will convert it to the platform's local conventions. line a space-delimited list of command-line arguments.

It is highly recommended to avoid the line version when possible. Ant will try to split the command line in a way similar to what a (Unix) shell would do, but may create something that is very different from what you expect under some circumstances.

Examples

```
<arg value="-l -a"/>
```

is a single command-line argument containing a space character.

```
<arg line="-l -a"/>
```

represents two separate command-line arguments.

```
<arg path="/dir;/dir2:\dir3"/>
```

is a single command-line argument with the value \dir;\dir2;\dir3 on DOS-based systems and /dir:/dir2:/dir3 on Unix-like systems.

References The id attribute of the buildfile's elements can be used to refer to them. This can be useful if you are going to replicate the same snippet of XML over and over again - using a `<classpath>` structure more than once, for example.

The following example:

```
<project ... >
  <target ... >
    <rmic ...>
      <classpath>
        <pathelement location="lib/"/>
        <pathelement path="${java.class.path}/"/>
        <pathelement path="${additional.path}"/>
      </classpath>
    </rmic>
  </target>

  <target ... >
    <javac ...>
      <classpath>
        <pathelement location="lib/"/>
        <pathelement path="${java.class.path}/"/>
        <pathelement path="${additional.path}"/>
      </classpath>
    </javac>
  </target>
</project>
```

could be rewritten as:

```
<project ... >
  <path id="project.class.path">
    <pathelement location="lib/"/>
    <pathelement path="${java.class.path}/"/>
    <pathelement path="${additional.path}"/>
  </path>

  <target ... >
    <rmic ...>
      <classpath refid="project.class.path"/>
    </rmic>
  </target>

  <target ... >
    <javac ...>
      <classpath refid="project.class.path"/>
    </javac>
  </target>
</project>
```

All tasks that use nested elements for PatternSets, FileSets or path-like structures accept references to these structures as well.

6.2.11 Selectors

Selectors are a mechanism whereby the files that make up a fileset can be selected based on criteria other than filename as provided by the `<include>` and `<exclude>` tags.

How to use a Selector A selector is an element of FileSet, and appears within it. It can also be defined outside of any target by using the `<selector>` tag and then using it as a reference.

Different selectors have different attributes. Some selectors can contain other selectors, and these are called Selector Containers. There is also a category of selectors that allow user-defined extensions, called Custom Selectors. The ones built in to Ant are called Core Selectors.

Core Selectors Core selectors are the ones that come standard with Ant. They can be used within a fileset and can be contained within Selector Containers.

The core selectors are:

`<contains>`	Select files that contain a particular text string
`<date>`	Select files that have been modified either before or after a particular date and time
`<depend>`	Select files that have been modified more recently than equivalent files elsewhere
`<depth>`	Select files that appear so many directories down in a directory tree
`<filename>`	Select files whose name matches a particular pattern. Equivalent to the include and exclude elements of a patternset.
`<present>`	Select files that either do or do not exist in some other location
`<size>`	Select files that are larger or smaller than a particular number of bytes.

Contains Selector

The `<contains>` tag in a FileSet limits the files defined by that fileset to only those which contain the string specified by the text attribute.

Attribute Description Required text Specifies the text that every file must contain Yes casesensitive Whether to pay attention to case when looking for the string in the text attribute. Default is true. No

Here is an example of how to use the Contains Selector:

```
<fileset dir="${doc.path}" includes="**/*.html">
    <contains text="script" casesensitive="no"/>
</fileset>
```

Selects all the HTML files that contain the string script.

Date Selector The `<date>` tag in a FileSet will put a limit on the files specified by the include tag, so that tags whose last modified date does not meet the date limits specified by the selector will not end up being selected.

Attribute Description Required datetime Specifies the date and time to test for using a string of the format MM/DD/YYYY HH:MM AM_or_PM. At least one of the two. millis The number of milliseconds since 1970 that should be tested for. It is usually much easier to use the datetime attribute. granularity The number of milliseconds leeway to give before deciding whether a files modification time matches a date. This is needed because not every file system supports tracking the last modified time to the millisecond level. The file will be selected provided the condition could be true were the granularity added or subtracted from the actual time. Default is 0 milliseconds except on Windows systems, where it is 2000 milliseconds (2 seconds). No when Indicates how to interpret the date, whether the files to be selected are those whose last modified times should be before, after, or equal to the specified value. Acceptable values for this attribute are: before - select files whose last modified date is before the indicated date after - select files whose last modified date is after the indicated date equal - select files whose last modified date is this exact date The default is equal. No

Here is an example of how to use the Date Selector:

```
<fileset dir="${jar.path}" includes="**/*.jar">
    <date datetime="01/01/2001 12:00 AM" when="before"/>
</fileset>
```

Selects all JAR files which were last modified before midnight January 1, 2001.

Depend Selector The `<depend>` tag selects files whose last modified date is later than another, equivalent file in another location.

The `<depend>` tag supports the use of a contained `<mapper>` element to define the location of the file to be compared against. If no `<mapper>` element is specified, the identity type mapper is used.

Attribute Description Required

targetdir The base directory to look for the files to compare against. The precise location depends on a combination of this attribute and the `<mapper>` element, if any. Yes

granularity The number of milliseconds leeway to give before deciding a file is out of date. This is needed because not every file system supports tracking the last modified time to the millisecond level. Default is 0 milliseconds except on Windows systems, where it is 2000 milliseconds (2 seconds). No

Here is an example of how to use the Depend Selector:

```
<fileset dir="${ant.1.5}/src/main" includes="**/*.java">
    <depend targetdir="${ant.1.4.1}/src/main"/>
</fileset>
```

Selects all the Java source files which were modified in the 1.5 release.

Depth Selector

The `<depth>` tag selects files based on how many directy levels deep they are in relation to the base directory of the fileset.

Attribute Description Required min The minimum number of directory levels below the base directory that a file must be in order to be selected. Default is no limit. At least one of the two. max The maximum number of directory levels below the base directory that a file can be and still be selected. Default is no limit.

Here is an example of how to use the Depth Selector:

```
<fileset dir="${doc.path}" includes="**/*">
    <depth max="1"/>
</fileset>
```

Selects all files in the base directory and one directory below that.

Filename Selector

The `<filename>` tag acts like the `<include>` and `<exclude>` tags within a fileset. By using a selector instead, however, one can combine it with all the other selectors using whatever selector container is desired.

Attribute Description Required name The name of files to select. The name parameter can contain the standard Ant wildcard characters. Yes casesensitive Whether to pay attention to case when looking at file names. Default is "true".

No negate Whether to reverse the effects of this filename selection, therefore emulating an exclude rather than include tag. Default is "false". No

Here is an example of how to use the Filename Selector:

```
<fileset dir="${doc.path}" includes="**/*">
    <filename name="**/*.css"/>
</fileset>
```

Selects all the cascading style sheet files.

Present Selector

The `<present>` tag selects files that have an equivalent file in another directory tree.

The `<present>` tag supports the use of a contained `<mapper>` element to define the location of the file to be tested against. If no `<mapper>` element is specified, the identity type mapper is used.

Attribute Description Required

targetdir The base directory to look for the files to compare against. The precise location depends on a combination of this attribute and the `<mapper>` element, if any. Yes

present Whether we are requiring that a file is present in the src directory tree only, or in both the src and the target directory tree. Valid values are: srconly - select files only if they are in the src directory tree but not in the target directory tree both - select files only if they are present both in the src and target directory trees Default is both. Setting this attribute to "srconly" is equivalent to wrapping the selector in the `<not>` selector container. No

Here is an example of how to use the Present Selector:

```
<fileset dir="${ant.1.5}/src/main" includes="**/*.java">
    <present present="srconly" targetdir="${ant.1.4.1}/src/main"/>
</fileset>
```

Selects all the Java source files which are new in the 1.5 release.

Size Selector

The `<size>` tag in a FileSet will put a limit on the files specified by the include tag, so that tags which do not meet the size limits specified by the selector will not end up being selected.

Attribute Description Required value The size of the file which should be tested for. Yes units The units that the value attribute is expressed in. When using the standard single letter SI designations, such as "k","M", or "G", multiples of 1000 are used. If you want to use power of 2 units, use the IEC standard: "Ki" for 1024, "Mi" for 1048576, and so on. The default is no units, which means the value attribute expresses the exact number of bytes. No when Indicates how to interpret the size, whether the files to be selected should be larger, smaller, or equal to that value. Acceptable values for this attribute are: less - select files less than the indicated size more - select files greater than the indicated size equal - select files this exact size The default is equal. No

Here is an example of how to use the Size Selector:

```
<fileset dir="${jar.path}">
  <patternset>
    <include name="**/*.jar"/>
  </patternset>
  <size value="4" units="Ki" when="more"/>
</fileset>
```

Selects all JAR files that are larger than 4096 bytes.

Selector Containers To create more complex selections, a variety of selectors that contain other selectors are available for your use. They combine the selections of their child selectors in various ways.

The selector containers are:

`<and>`	select a file only if all the contained selectors select it.
`<majority>`	select a file if a majority of its selectors select it.
`<none>`	select a file only if none of the contained selectors select it.
`<not>`	can contain only one selector, and reverses what it selects and doesn't select.
`<or>`	selects a file if any one of the contained selectors selects it.
`<selector>`	contains only one selector and forwards all requests to it without alteration. This is the selector to use if you want to define a reference. It is usable as an element of `<project>`.

All selector containers can contain any other selector, including other containers, as an element. Using containers, the selector tags can be arbitrarily deep. Here is a complete list of allowable selector elements within a container:

```
<and>
<contains>
<custom>
<date>
<depend>
<depth>
<filename>
<majority>
<none>
<not>
<or>
<present>
<selector>
<size>
```

And Selector The `<and>` tag selects files that are selected by all of the elements it contains. It returns as soon as it finds a selector that does not select the file, so it is not guaranteed to check every selector.

Here is an example of how to use the And Selector:

```
<fileset dir="${dist}" includes="**/*.jar">
    <and>
        <size value="4" units="Ki" when="more"/>
        <date datetime="01/01/2001 12:00 AM" when="before"/>
    </and>
</fileset>
```

Selects all the JAR file larger than 4096 bytes which haven't been update since the last millenium.

Majority Selector The `<majority>` tag selects files provided that a majority of the contained elements also select it. Ties are dealt with as specified by the allowtie attribute.

Attribute Description Required allowtie Whether files should be selected if there are an even number of selectors selecting them as are not selecting them. Default is true. No

Here is an example of how to use the Majority Selector:

```
<fileset dir="${docs}" includes="**/*.html">
    <majority>
        <contains text="project" casesensitive="false"/>
        <contains text="taskdef" casesensitive="false"/>
        <contains text="IntrospectionHelper" casesensitive="true"/>
    </majority>
</fileset>
```

Selects all the HTML files which contain at least two of the three phrases "project", "taskdef", and "IntrospectionHelper" (this last phrase must match case exactly).

None Selector

The `<none>` tag selects files that are not selected by any of the elements it contains. It returns as soon as it finds a selector that selects the file, so it is not guaranteed to check every selector.

Here is an example of how to use the None Selector:

```
<fileset dir="${src}" includes="**/*.java">
    <none>
        <present targetdir="${dest}"/>
        <present targetdir="${dest}">
            <mapper type="glob" from="*.java" to="*.class"/>
        </present>
    </none>
</fileset>
```

Selects only Java files which do not have equivalent java or class files in the dest directory.

Not Selector The `<not>` tag reverses the meaning of the single selector it contains.

Here is an example of how to use the Not Selector:

```
<fileset dir="${src}" includes="**/*.java">
    <not>
        <contains text="test"/>
    </not>
</fileset>
```

Selects all the files in the src directory that do not contain the string ”test”.

Or Selector

The `<or>` tag selects files that are selected by any one of the elements it contains. It returns as soon as it finds a selector that selects the file, so it is not guaranteed to check every selector.

Here is an example of how to use the Or Selector:

```
<fileset dir="${basedir}">
    <or>
        <depth max="0"/>
        <filename name="*.png"/>
        <filename name="*.gif"/>
        <filename name="*.jpg"/>
    </or>
</fileset>
```

Selects all the files in the top directory along with all the image files below it.

Selector Reference

The `<selector>` tag is used to create selectors that can be reused through references. It is the only selector which can be used outside of any target, as an element of the `<project>` tag. It can contain only one other selector, but of course that selector can be a container.

Here is an example of how to use the Selector Reference:

```
<project default="all" basedir="./ant">

    <selector id="completed">
        <none>
            <depend targetdir="build/classes">
                <mapper type="glob" from="*.java" to="*.class"/>
            </depend>
            <depend targetdir="docs/manual/api">
                <mapper type="glob" from="*.java" to="*.html"/>
            </depend>
        </none>
    </selector>
    <target>
        <zip>
            <fileset dir="src/main" includes="**/*.java">
                <selector refid="completed"/>
            </fileset>
        </zip>
    </target>

</project>
```

Zips up all the Java files which have an up-to-date equivalent class file and javadoc file associated with them.

Custom Selectors

You can write your own selectors and use them within the selector containers by specifying them within the ¡custom¿ tag.

First, you have to write your selector class in Java. The only requirement it must meet in order to be a selector is that it implements the `org.apache.tools.ant.types.selectors.FileSelector` interface, which contains a single method. See Programming Selectors in Ant for more information.

Once that is written, you include it in your build file by using the `<custom>` tag.

Attribute Description Required classname The name of your class that implements org.apache.tools.ant.types.selectors.FileSelector. Yes classpath The classpath to use in order to load the custom selector class. If neither this classpath nor the classpathref are specified, the class will be loaded from the classpath that Ant uses. No classpathref A reference to a classpath previously defined. If neither this reference nor the classpath above are specified, the class will be loaded from the classpath that Ant uses. No

Here is how you use `<custom>` to use your class as a selector:

```
<fileset dir="${mydir}" includes="**/*">
    <custom classname="com.mydomain.MySelector">
        <param name="myattribute" value="myvalue"/>
    </custom>
</fileset>
```

A number of core selectors can also be used as custom selectors by specifying their attributes using `<param>` elements. These are

Contains Selector with classname org.apache.tools.ant.types.selectors.ContainsSelector
Date Selector with classname org.apache.tools.ant.types.selectors.DateSelector
Depth Selector with classname org.apache.tools.ant.types.selectors.DepthSelector
Filename Selector with classname org.apache.tools.ant.types.selectors.FilenameSelector
Size Selector with classname org.apache.tools.ant.types.selectors.SizeSelector

Here is the example from the Depth Selector section rewritten to use the selector through `<custom>`.

```
<fileset dir="${doc.path}" includes="**/*">
```

```
    <custom classname="org.apache.tools.ant.types.selectors.DepthSelector">
        <param name="max" value="1"/>
    </custom>
</fileset>
```

Selects all files in the base directory and one directory below that.

For more details concerning writing your own selectors, consult Programming Selectors in Ant.

6.2.12 XMLCatalog

An XMLCatalog is a catalog of public resources such as DTDs or entities that are referenced in an XML document. Catalogs are typically used to make web references to resources point to a locally cached copy of the resource.

This allows the XML Parser, XSLT Processor or other consumer of XML documents to efficiently allow a local substitution for a resource available on the web.

This data type provides a catalog of resource locations based on the OASIS ”Open Catalog” standard. The catalog entries are used both for Entity resolution and URI resolution, in accordance with the org.xml.sax.EntityResolver and javax.xml.transform.URIResolver interfaces as defined in the Java API for XML Processing (JAXP) Specification.

For example, in a web.xml file, the DTD is referenced as:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

The XML processor, without XMLCatalog support, would need to retrieve the DTD from the URL specified whenever validation of the document was required.

This can be very time consuming during the build process, especially where network throughput is limited. Alternatively, you can do the following:

Copy web-app_2_2.dtd onto your local disk somewhere (either in the filesystem or even embedded inside a jar or zip file on the classpath).

Create an `<xmlcatalog>` with a `<dtd>` element whose location attribute points to the file.

Success! The XML processor will now use the local copy instead of calling out to the internet.

XMLCatalogs can appear inside tasks that support this feature or at the same level as target - i.e., as children of project for reuse across different tasks, e.g. XML Validation and XSLT Transformation. The XML Validate task uses XMLCatalogs for entity resolution. The XSLT Transformation task uses XMLCatalogs for both entity and URI resolution.

XMLCatalogs are specified as either a reference to another XMLCatalog, defined previously in a build file, or as a list of dtd or entity locations. A separate classpath for entity resolution may be specified inline via nested classpath elements; otherwise the system classpath is used for this as well.

XMLCatalogs can also be nested inside other XMLCatalogs. For example, a "superset" XMLCatalog could be made by including several nested XMLCatalogs that referred to other, previously defined XMLCatalogs.

Currently, only `<dtd>` and `<entity>` elements may be specified inline; these roughly correspond to OASIS catalog entry types PUBLIC and URI respectively.

Entity/DTD/URI Resolution Algorithm When an entity, DTD, or URI is looked up by the XML processor, the XMLCatalog searches its list of entries to see if any match. That is, it attempts to match the publicId attribute of each entry with the PublicID or URI of the entity to be resolved. Assuming a matching entry is found, XMLCatalog then executes the following steps: 1. Filesystem lookup The location is first looked up in the filesystem. If the location is a relative path, the ant project basedir attribute is used as the base directory. If the location specifies an absolute path, it is used as is. Once we have an absolute path in hand, we check to see if a valid and readable file exists at that path. If so, we are done. If not, we proceed to the next step.

2. Classpath lookup The location is next looked up in the classpath. Recall that jar files are merely fancy zip files. For classpath lookup, the location is used as is (no base is prepended). We use a Classloader to attempt to load the resource from the classpath. For example, if hello.jar is in the classpath and it contains foo/bar/blat.dtd it will resolve an entity whose location is foo/bar/blat.dtd. Of course, it will not resolve an entity whose location is blat.dtd.

3. URL-space lookup Finally, we attempt to make a URL out of the location. At first this may seem like this would defeat the purpose of XMLCatalogs – why go back out to the internet? But in fact, this can be used to (in a sense) implement HTTP redirects, substituting one URL for another. The mapped-to URL might also be served by a local web server. If the URL resolves to a valid and readable resource, we are done. Otherwise, we give up. In this case, the XML processor will perform its normal resolution algorithm. Depending on the processor configuration, further resolution failures may or may not result in fatal (i.e. build-ending) errors.

XMLCatalog attributes Attribute Description Required id a unique name for an XMLCatalog, used for referencing the XMLCatalog's contents from another XMLCatalog No refid the id of another XMLCatalog whose contents you would like to be used for this XMLCatalog No

XMLCatalog nested elements dtd/entity The dtd and entity elements used to specify XMLCatalogs are identical in their structure

Attribute Description Required publicId The public identifier used when defining a dtd or entity, e.g. "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN" Yes location The location of the local replacement to be used for the public identifier specified. This may be specified as a file name, resource name found on the classpath, or a URL. Relative paths will be resolved according to the base, which by default is the Ant project basedir. Yes

classpath

The classpath to use for entity resolution. The nested `<classpath>` is a path-like structure.

Examples

Set up an XMLCatalog with a single dtd referenced locally in a user's home directory:

```
<xmlcatalog>
    <dtd
        publicId="-//OASIS//DTD DocBook XML V4.1.2//EN"
        location="/home/dion/downloads/docbook/docbookx.dtd"/>
</xmlcatalog>
```

Set up an XMLCatalog with a multiple dtds to be found either in the filesystem (relative to the Ant project basedir) or in the classpath:

```
<xmlcatalog id="commonDTDs">
    <dtd
        publicId="-//OASIS//DTD DocBook XML V4.1.2//EN"
        location="docbook/docbookx.dtd"/>
    <dtd
        publicId="-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
        location="web-app_2_2.dtd"/>
</xmlcatalog>
```

Set up an XMLCatalog with a combination of DTDs and entities as well as a nested XMLCatalog:

```
<xmlcatalog id="allcatalogs">
    <dtd
        publicId="-//ArielPartners//DTD XML Article V1.0//EN"
        location="com/arielpartners/knowledgebase/dtd/article.dtd"/>
    <entity
        publicId="LargeLogo"
        location="com/arielpartners/images/ariel-logo-large.gif"/>
    <xmlcatalog refid="commonDTDs"/>
</xmlcatalog>
```

To reference the above XMLCatalog in an xslt task:

```
<xslt basedir="${source.doc}"
      destdir="${dest.xdocs}"
      extension=".xml"
      style="${source.xsl.converter.docbook}"
      includes="**/*.xml"
      force="true">
    <xmlcatalog refid="allcatalogs"/>
</xslt>
```

6.3 Optional Types

6.3.1 ClassFileSet

A classfileset is a specialised type of fileset which, given a set of ”root” classes, will include all of the class files upon which the root classes depend. This is typically used to create a jar with all of the required classes for a particular application.

classfilesets are typically used by reference. They are declared with an ”id” value and this is then used as a reference where a normal fileset is expected.

This type requires the jakarta-BCEL library.

Attributes The class fileset support the following attributes in addition to those supported by the standard fileset:

Attribute Description Required rootclass A single root class name No

Nested Elements Root

When more than one root class is required, multiple nested `<root>` elements may be used

Attribute Description Required classname The fully qualified name of the root class Yes

RootFileSet

A root fileset is used to add a set of root classes from a fileset. In this case the entries in the fileset are expected to be Java class files. The name of the Java class is determined by the relative location of the classfile in the fileset. So, the file org/apache/tools/ant/Project.class corresponds to the Java class org.apache.tools.ant.Project.

Examples

```
<classfileset id="reqdClasses" dir="${classes.dir}">
  <root classname="org.apache.tools.ant.Project" />
</classfileset>
```

This example creates a fileset containing all the class files upon which the org.apache.tools.ant.Project class depends. This fileset could then be used to create a jar.

```
<jar destfile="minimal.jar">
  <fileset refid="reqdClasses"/>
</jar>

<classfileset id="reqdClasses" dir="${classes.dir}">
  <rootfileset dir="${classes.dir}" includes="org/apache/tools/ant/Project*.class"/
</classfileset>
```

This example constructs the classfileset using all the class with names starting with Project in the org.apache.tools.ant package

6.3.2 Extension Package

Utility type that represents either an available "Optional Package" (formerly known as "Standard Extension") as described in the manifest of a JAR file, or the requirement for such an optional package.

Note that this type works with extensions as defined by the "Optional Package" specification. For more information about optional packages, see the document Optional Package Versioning in the documentation bundle for your Java2 Standard Edition package, in file guide/extensions/versioning.html or online at http://java.sun.com/j2se/1.3/docs/guide/extensions/versioning.html.

Attributes The extension type supports the following attributes:

Attribute Description Required extensionName The name of extension yes specificationVersion The version of extension specification (Must be in dewey decimal aka dotted decimal notation. 3.2.4) no specificationVendor The specification vendor no implementationVersion The version of extension implementation (Must be in dewey decimal aka dotted decimal notation. 3.2.4) no implementationVendor The implementation vendor no implementationVendorId The implementation vendor ID no implementationURL The url from which to retrieve extension. no

Examples

```
<extension id="e1"
    extensionName="MyExtensions"
    specificationVersion="1.0"
    specificationVendor="Peter Donald"
    implementationVendorID="vv"
    implementationVendor="Apache"
    implementationVersion="2.0"
    implementationURL="http://somewhere.com/myExt.jar"/>
```

Fully specifiec extension object.

```
<extension id="e1"
    extensionName="MyExtensions"
    specificationVersion="1.0"
    specificationVendor="Peter Donald"/>
```

Extension object that just species the specification details.

6.3.3 Set of Extension Packages

Utility type that represents a set of Extensions.

Note that this type works with extensions as defined by the "Optional Package" specification. For more information about optional packages, see the document Optional Package Versioning in the documentation bundle for your Java2 Standard Edition package, in file guide/extensions/versioning.html or online at http://java.sun.com/j2se/1.3/docs/guide/extensions/versioning.html.

Nested Elements extension Extension object to add to set.

fileset FileSets all files contained contained within set that are jars and implement an extension are added to extension set.

LibFileSet All files contained contained within set that are jars and implement an extension are added to extension set. However the extension information may be modified by attributes of libfileset

Examples

```
<extension id="e1"
    extensionName="MyExtensions"
    specificationVersion="1.0"
    specificationVendor="Peter Donald"
    implementationVendorID="vv"
    implementationVendor="Apache"
    implementationVersion="2.0"
    implementationURL="http://somewhere.com/myExt.jar"/>

<libfileset id="lfs"
               includeUrl="true"
               includeImpl="false"
               dir="tools/lib">
  <include name="*.jar"/>
</libfileset>

<extensionSet id="exts">
  <libfileset dir="lib">
    <include name="*.jar"/>
  </libfileset>
  <libfileset refid="lfs"/>
  <extension refid="e1"/>
</extensionSet>
```

Chapter 7

Loggers and Listeners

7.1 Overview

Ant has two related features to allow the build process to be monitored: listeners and loggers.

Listeners A listener is alerted of the following events:

build started build finished target started target finished task started task finished message logged Loggers Loggers extend the capabilities of listeners and add the following features:

Receives a handle to the standard output and error print streams and therefore can log information to the console or the -logfile specified file. Logging level (-quiet, -verbose, -debug) aware Emacs-mode aware Built-in Listeners/Loggers Classname Description Type org.apache.tools.ant.DefaultLogger The logger used implicitly unless overridden with the -logger command-line switch. BuildLogger org.apache.tools.ant.NoBannerLogger This logger omits output of empty target output. BuildLogger org.apache.tools.ant.listener.MailLogger Extends DefaultLogger such that output is still generated the same, and when the build is finished an e-mail can be sent. BuildLogger org.apache.tools.ant.listener.AnsiColorLogger Colorifies the build output. BuildLogger org.apache.tools.ant.listener.Log4jListener Passes events to Log4j for highly customizable logging. BuildListener org.apache.tools.ant.XmlLogger Writes the build information to an XML file. BuildLogger

DefaultLogger Simply run Ant normally, or:

ant -logger org.apache.tools.ant.DefaultLogger

NoBannerLogger Removes output of empty target output.

ant -logger org.apache.tools.ant.NoBannerLogger

MailLogger The MailLogger captures all output logged through DefaultLogger (standard Ant output) and will send success and failure messages to unique e-mail lists, with control for turning off success or failure messages individually.

Properties controlling the operation of MailLogger:

Property Description Required MailLogger.mailhost Mail server to use No, default ”localhost” MailLogger.from Mail ”from” address Yes, if mail needs to be sent MailLogger.failure.notify Send build failure e-mails? No, default "true" MailLogger.success.notify Send build success e-mails? No, default "true" MailLogger.failure.to Address(es) to send failure messages to, comma-separated Yes, if failure mail is to be sent MailLogger.success.to Address(es) to send success messages to, comma-separated Yes, if success mail is to be sent MailLogger.failure.subject Subject of failed build No, default "Build Failure" MailLogger.success.subject Subject of successful build No, default "Build Success" MailLogger.properties.file Filename of properties file that will override other values. No

ant -logger org.apache.tools.ant.listener.MailLogger

AnsiColorLogger The AnsiColorLogger adds color to the standard Ant output by prefixing and suffixing ANSI color code escape sequences to it. It is just an extension of DefaultLogger and hence provides all features that DefaultLogger does.

AnsiColorLogger differentiates the output by assigning different colors depending upon the type of the message.

If used with the -logfile option, the output file will contain all the necessary escape codes to display the text in colorized mode when displayed in the console using applications like cat, more, etc.

This is designed to work on terminals that support ANSI color codes. It works on XTerm, ETerm, Win9x Console (with ANSI.SYS loaded.), etc.

NOTE: It doesn't work on WinNT even when a COMMAND.COM console loaded with ANSI.SYS is used.

If the user wishes to override the default colors with custom ones, a file containing zero or more of the custom color key-value pairs must be created. The recognized keys and their default values are shown below:

```
AnsiColorLogger.ERROR_COLOR=2;31
AnsiColorLogger.WARNING_COLOR=2;35
AnsiColorLogger.INFO_COLOR=2;36
AnsiColorLogger.VERBOSE_COLOR=2;32
AnsiColorLogger.DEBUG_COLOR=2;34
```

Each key takes as value a color combination defined as Attribute;Foreground;Background. In the above example, background value has not been used.

This file must be specfied as the value of a system variable named ant.logger.defaults and passed as an argument using the -D option to the java command that invokes the Ant application. An easy way to achieve this is to add -Dant.logger.defaults= /path/to/your/file to the `ANT_OPTS` environment variable. Ant's launching script recognizes this flag and will pass it to the java command appropriately.

Format:

```
AnsiColorLogger.*=Attribute;Foreground;Background
```

Attribute is one of the following:

```
0 -> Reset All Attributes (return to normal mode)
1 -> Bright (Usually turns on BOLD)
```

```
2 -> Dim
3 -> Underline
5 -> link
7 -> Reverse
8 -> Hidden
```

Foreground is one of the following:

```
30 -> Black
31 -> Red
32 -> Green
33 -> Yellow
34 -> Blue
35 -> Magenta
36 -> Cyan
37 -> White
```

Background is one of the following:

```
40 -> Black
41 -> Red
42 -> Green
43 -> Yellow
44 -> Blue
45 -> Magenta
46 -> Cyan
47 -> White
ant -logger org.apache.tools.ant.listener.AnsiColorLogger
```

Log4jListener Passes build events to Log4j, using the full classname's of the generator of each build event as the category:

build started / build finished - org.apache.tools.ant.Project target started / target finished - org.apache.tools.ant.Target task started / task finished - the fully qualified classname of the task message logged - the classname of one of the above, so if a task logs a message, its classname is the category used, and so on. All start events are logged as INFO. Finish events are either logged as INFO or ERROR depending on whether the build failed during that stage. Message events are logged according to their Ant logging level, mapping directly to a corresponding Log4j level.

ant -listener org.apache.tools.ant.listener.Log4jListener

XmlLogger Writes all build information out to an XML file named log.xml, or the value of the XmlLogger.file property if present, when used as a listener. When used as a logger, it writes all output to either the console or to the value of -logfile. Whether used as a listener or logger, the output is not generated until the build is complete, as it buffers the information in order to provide timing information for task, targets, and the project.

By default the XML file creates a reference to an XSLT file "log.xsl" in the current directory; look in `ANT_HOME`/etc for one of these. You can set the property ant.XmlLogger.stylesheet.uri to provide a uri to a style sheet. this can be a relative or absolute file path, or an http URL. If you set the property to the empty string, "", no XSLT transform is declared at all.

```
ant -listener org.apache.tools.ant.XmlLogger ant -logger org.apache.tools.ant.XmlLogger
-verbose -logfile build_log.xml
```

Writing your own See the Build Events section for developers.

Notes:

A listener or logger should not write to standard output or error - Ant captures these internally and may cause an infinite loop.

Chapter 8

Editor/IDE Integration

All the modern Java IDEs support Ant almost out the box, with the notable exception of JBuilder Personal.

8.1 Antidote

Version 0.1 (2001/02/13)
Authors: Simeon H.K. Fitch

8.1.1 Overview

Antidote is the Ant subproject for developing a graphical user interface to facilitate the efficient use of Ant. In general, its purpose is to allow the quick generation, modification, and use of Ant build files, helping the user define a build process and track down build problems. It is not meant to be an IDE, but an enabler for the powerful features available in Ant, particularly for novice users, or users who want a rapid way of controlling their build process.

Status Antidote is still in the early stages of development, but does have a set of usable features, including:

Reading Ant build files. Selecting targets and executing them. Context highlighted build status console. Modification of (some) build file components. Saving modified build file. Current development tasks include:

A more complete set of target and task editing capabilities. A wizard for creating basic build files, including importing existing code bases. Better build progress monitoring. The Antidote source distribution comes with requirements and design documentation that better cover the details of application architecture, how to develop against it, and what the long term goals are. Furthermore, there is a TODO file listing the detailed, near-term tasks that need accomplishing.

Getting Involved The source code for Antidote is located in a separate Module (ant-antidote) in CVS. All the existing documentation can be found there where new contributors should read:

Design Overview Feature List Idea Refinement New Module HOWTO Static Class Diagrams Online discussions about Antidote occur on the Ant developer mailing list. The application infrastructure is fairly complete, but there are almost unlimited opportunities for feature contributions.

Aspiring contributors new to the project should (carefully) read the following for details on the contribution process:

Get Involved Project Guidelines Source Repositories (how to contribute patches)

8.2 AntRunner for JBuilder

Documentation may be found at `http://antrunner.sourceforge.net/`

8.3 AntWork Plugin for the Jext - Java Text Editor

by

```
Klaus Hartlage (KHartlage@t-online.de)
Version $Revision: 1.3.2.1 $ - $Date: 2002/09/03 15:27:58 $
```

You can download the plugin at: ftp://jext.sourceforge.net/pub/jext/plugins/AntWork.zip

Installation instructions from the Readme.txt: You have to enable the Jext Console to see the Ant output (menu: Edit-¿Options... - General Panel), because the Ant messages are redirected to the Jext console.

You can configure the Ant call in the Jext menu: Edit-¿Options... - Plugin Options - Antwork Plugin Panel; here you can set the ant home directory and the path to your build file.

You can start AntWork in the menu: Plugins-¿Ant-¿Work Now! In the appearing dialog box you can enter the target which you want to compile.

If a javac error occurs in the ant run an error-list opens within Jext. With a double-click on the error-message you jump to the error in the specified java text file.

8.4 Emacs

Emacs JDE has built-in text ANT integration: selection of target through text field, execution, hyperlink to compilation errors. Installation: built-in JDE 2.2.8 or later. Configuration: through customize menu "Jde Build Function"

8.5 IDEA

IDEA has built-in GUI ANT integration: GUI selection of targets, execution, hyperlink to compilation errors

NetBeans NetBeans 3.4 has very good Ant integration indeed.

jEdit jEdit is an open source java IDE with some great plugins for Java dev, a good XML editor and the Antfarm plugin to execute targets in a build file.

Eclipse Eclipse is IBM's counterpoint to NetBeans; an open source IDE with Java and ant support.

VisualAge for Java

WebSphere Studio Application Developer

Chapter 9

Developing with Ant

9.1 Writing Your Own Task

It is very easy to write your own task:

Create a Java class that extends org.apache.tools.ant.Task or another class that was desgined to be extended. For each attribute, write a setter method. The setter method must be a public void method that takes a single argument. The name of the method must begin with set, followed by the attribute name, with the first character of the name in uppercase, and the rest in lowercase. That is, to support an attribute named file you create a method setFile. Depending on the type of the argument, Ant will perform some conversions for you, see below. If your task shall contain other tasks as nested elements (like parallel), your class must implement the interface org.apache.tools.ant.TaskContainer. If you do so, your task can not support any other nested elements. See below. If the task should support character data (text nested between the start end end tags), write a public void addText(String) method. Note that Ant does not expand properties on the text it passes to the task. For each nested element, write a create, add or addConfigured method. A create method must be a public method that takes no arguments and returns an Object type. The name of the create method must begin with create, followed by the element name. An add (or addConfigured) method must be a public void method that takes a single argument of an Object type with a no-argument constructor. The name of the add (addConfigured) method must begin with add (addConfigured), followed by the element name. For a more complete discussion see below. Write a public void execute method, with no arguments, that throws a BuildException. This method implements the task itself. The Life-cycle of a Task The task gets instantiated using a no-argument constructor, at parser time. This means even tasks that are never executed get instantiated. The task gets references to its project and location inside the buildfile via its inherited project and location variables. If the user specified an id attribute to this task, the project registers a reference to this newly created task, at parser time. The task gets a reference to the target it belongs to via its inherited target variable. init() is called at parser time. All child elements of the XML element corresponding to this task are created via this task's createXXX() methods or instantiated and added to this task via its addXXX() methods, at parser time. All attributes of this task get set via their corresponding setXXX methods, at runtime. The content character data sections inside the XML element corresponding to this task is added to the task via its addText method, at runtime. All attributes of all child elements get set via their corresponding setXXX methods, at runtime. execute() is called at runtime. While the above initialization steps only occur once, the execute() method may be called more than once, if the task is invoked more than once. For example, if target1 and target2 both depend on target3, then running 'ant target1 target2' will run all tasks in target3 twice. Conversions Ant will perform for attributes Ant will always expand properties before it passes the value of an attribute to the corresponding setter method.

The most common way to write an attribute setter is to use a java.lang.String argument. In this case Ant will pass the literal value (after property expansion) to your task. But there is more! If the argument of you setter method is

boolean, your method will be passed the value true if the value specified in the build file is one of true, yes, or on and false otherwise. char or java.lang.Character, your method will be passed the first character of the value specified in the build file. any other primitive type (int, short and so on), Ant will convert the value of the attribute into this type, thus making sure that you'll never receive input that is not a number for that attribute. java.io.File, Ant will first determine whether the value given in the build file represents an absolute path name. If not, Ant will interpret the value as a path name relative to the project's basedir. org.apache.tools.ant.types.Path, Ant will tokenize the value specified in the build file, accepting : and ; as path separators. Relative path names will be interpreted as relative to the project's basedir. java.lang.Class, Ant will interpret the value given in the build file as a Java class name and load the named class from the system class loader. any other type that has a constructor with a single String argument, Ant will use this constructor to create a new instance from the value given in the build file. A subclass of org.apache.tools.ant.types.EnumeratedAttribute, Ant will invoke this classes setValue method. Use this if your task should support enumerated attributes (attributes with values that must be part of a predefined set of values). See org/apache/tools/ant/taskdefs/FixCRLF.java and the inner AddAsisRemove class used in setCr for an example. What happens if more than one setter method is present for a given attribute? A method taking a String argument will always lose against the more specific methods. If there are still more setters Ant could chose from, only one of them will be called, but we don't know which, this depends on the implementation of your Java virtual machine.

Supporting nested elements Let's assume your task shall support nested elements with the name inner. First of all, you need a class that represents this nested element. Often you simply want to use one of Ant's classes like org.apache.tools.ant.types.FileSet to support nested fileset elements.

Attributes of the nested elements or nested child elements of them will be handled using the same mechanism used for tasks (i.e. setter methods for attributes, addText for nested text and create/add/addConfigured methods for child elements).

Now you have a class NestedElement that is supposed to be used for your nested `<inner>` elements, you have three options:

public NestedElement createInner() public void addInner(NestedElement anInner) public void addConfiguredInner(NestedElement anInner) What is the difference?

Option 1 makes the task create the instance of NestedElement, there are no restrictions on the type. For the options 2 and 3, Ant has to create an instance of NestedInner before it can pass it to the task, this means, NestedInner must have a public no-arg constructor. This is the only difference between options 1 and 2.

The difference between 2 and 3 is what Ant has done to the object before it passes it to the method. addInner will receive an object directly after the constructor has been called, while addConfiguredInner gets the object after the attributes and nested children for this new object have been handled.

What happens if you use more than one of the options? Only one of the methods will be called, but we don't know which, this depends on the implementation of your Java virtual machine.

TaskContainer The TaskContainer consists of a single method, addTask that basically is the same as an add method for nested elements. The task instances will be configured (their attributes and nested elements have been handled) when your task's execute method gets invoked, but not before that.

When we said execute would be called, we lied ;-). In fact, Ant will call the perform method in org.apache.tools.ant.Task, which in turn calls execute. This method makes sure that Build Events will be triggered. If you execute the task instances nested into your task, you should also invoke perform on these instances instead of execute.

Example

Let's write our own task, which prints a message on the System.out stream. The task has one attribute, called message.

```
package com.mydomain;

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

public class MyVeryOwnTask extends Task {
    private String msg;

    // The method executing the task
    public void execute() throws BuildException {
        System.out.println(msg);
    }
    // The setter for the "message" attribute
    public void setMessage(String msg) {
        this.msg = msg;
    }
}
```

It's really this simple ;-)

Adding your task to the system is rather simple too:

Make sure the class that implements your task is in the classpath when starting Ant.

Add a `<taskdef>` element to your project. This actually adds your task to the system.

Use your task in the rest of the buildfile. Example

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

<project name="OwnTaskExample" default="main" basedir=".">
  <taskdef name="mytask" classname="com.mydomain.MyVeryOwnTask"/>

  <target name="main">
    <mytask message="Hello World! MyVeryOwnTask works!"/>
  </target>
</project>
```

Example 2

To use a task directly from the buildfile which created it, place the `<taskdef>` declaration inside a target after the compilation. Use the classpath attribute of `<taskdef>` to point to where the code has just been compiled.

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

<project name="OwnTaskExample2" default="main" basedir=".">

  <target name="build" >
    <mkdir dir="build"/>
    <javac srcdir="source" destdir="build"/>
  </target>

  <target name="declare" depends="build">
    <taskdef name="mytask"
        classname="com.mydomain.MyVeryOwnTask"
        classpath="build"/>
  </target>

  <target name="main" depends="declare">
    <mytask message="Hello World! MyVeryOwnTask works!"/>
  </target>
</project>
```

Another way to add a task (more permanently), is to add the task name and implementing class name to the default.properties file in the org.apache.tools.ant.taskdefs package. Then you can use it as if it were a built-in task.

Build Events

Ant is capable of generating build events as it performs the tasks necessary to build a project. Listeners can be attached to Ant to receive these events. This capability could be used, for example, to connect Ant to a GUI or to integrate Ant with an IDE.

To use build events you need to create an ant Project object. You can then call the addBuildListener method to add your listener to the project. Your listener must implement the org.apache.tools.antBuildListener interface. The listener will receive BuildEvents for the following events

Build started Build finished Target started Target finished Task started Task finished Message logged If you wish to attach a listener from the command line you may use the -listener option. For example:

ant -listener org.apache.tools.ant.XmlLogger will run Ant with a listener that generates an XML representation of the build progress. This listener is included with Ant, as is the default listener, which generates the logging to standard output.

Note: A listener must not access System.out and System.err directly since ouput on these streams is redirected by Ant's core to the build event system. Accessing these streams can cause an infinite loop in Ant. Depending on the version of Ant, this will either cause the build to terminate or the Java VM to run out of Stack space. A logger, also, may not access System.out and System.err directly. It must use the streams with which it has been configured.

Source code integration The other way to extend Ant through Java is to make changes to existing tasks, which is positively encouraged. Both changes to the existing source and new tasks can be incorporated back into the Ant codebase, which benefits all users and spreads the maintenance load around. Please consult the Getting Involved pages on the Jakarta web site for details on how to fetch the latest source and how to submit changes for reincorporation into the source tree.

Ant also has some task guidelines which provides some advice to people developing and testing tasks. Even if you intend to keep your tasks to yourself, you should still read this as it should be informative.

9.2 Tasks Desgined for Extension

These classes are desgined to be extended. Always start here when writting your own task. Class Description Task Base class for all tasks. AbstractCvsTask Another task can extend this with some customized output processing JDBCTask Handles JDBC configuration needed by SQL type tasks. MatchingTask This is an abstract task that should be used by all those tasks that require to include or exclude files based on pattern matching. Pack Abstract Base class for pack tasks. Unpack Abstract Base class for unpack tasks.

9.3 Build Events

Ant is capable of generating build events as it performs the tasks necessary to build a project. Listeners can be attached to Ant to receive these events. This capability could be used, for example, to connect Ant to a GUI or to integrate Ant with an IDE.

To use build events you need to create an ant Project object. You can then call the addBuildListener method to add your listener to the project. Your listener must implement the org.apache.tools.antBuildListener interface. The listener will receive BuildEvents for the following events

Build started Build finished Target started Target finished Task started Task finished Message logged If you wish to attach a listener from the command line you may use the -listener option. For example:

ant -listener org.apache.tools.ant.XmlLogger will run Ant with a listener that generates an XML representation of the build progress. This listener is included with Ant, as is the default listener, which generates the logging to standard output.

Note: A listener must not access System.out and System.err directly since ouput on these streams is redirected by Ant's core to the build event system. Accessing these streams can cause an infinite loop in Ant. Depending on the version of Ant, this will either cause the build to terminate or the Java VM to run out of Stack space. A logger, also, may not access System.out and System.err directly. It must use the streams with which it has been configured.

9.4 Source code integration

The other way to extend Ant through Java is to make changes to existing tasks, which is positively encouraged. Both changes to the existing source and new tasks can be incorporated back into the Ant codebase, which benefits all users and spreads the maintenance load around. Please consult the Getting Involved pages on the Jakarta web site for details on how to fetch the latest source and how to submit changes for reincorporation into the source tree.

Ant also has some task guidelines which provides some advice to people developing and testing tasks. Even if you intend to keep your tasks to yourself, you should still read this as it should be informative.

9.5 InputHandler

9.5.1 Overview

When a task wants to prompt a user for input, it doesn't simply read the input from the console as this would make it impossible to embed Ant in an IDE. Instead it asks an implementation of the org.apache.tools.ant.input.InputHandler interface to prompt the user and hand the user input back to the task.

To do this, the task creates an InputRequest object and passes it to the InputHandler Such an InputRequest may know whether a given user input is valid and the InputHandler is supposed to reject all invalid input.

Exactly one InputHandler instance is associated with every Ant process, users can specify the implementation using the -inputhandler command line switch.

InputHandler The InputHandler interface contains exactly one method

void handleInput(InputRequest request) throws org.apache.tools.ant.BuildException;

with some pre- and postconditions. The main postcondition is that this method must not return unless the request considers the user input valid, it is allowed to throw an exception in this situation.

Ant comes with two built-in implementations of this interface:

DefaultInputHandler This is the implementation you get, when you don't use the -inputhandler command line switch at all. This implementation will print the prompt encapsulated in the request object to Ant's logging system and re-prompt for input until the user enters something that is considered valid input by the request object. Input will be read from the console and the user will need to press the Return key.

PropertyFileInputHandler This implementation is useful if you want to run unattended build processes. It reads all input from a properties file and makes the build fail if it cannot find valid input in this file. The name of the properties file must be specified in the Java system property ant.input.properties.

The prompt encapsulated in a request will be used as the key when looking up the input inside the properties file. If no input can be found, the input is considered invalid and an exception will be thrown.

Note that ant.input.properties must be a Java system property, not an Ant property. I.e. you cannot define it as a simple parameter to ant, but you can define it inside the `ANT_OPTS` environment variable.

InputRequest Instances of org.apache.tools.ant.input.InputRequest encapsulate the information necessary to ask a user for input and validate this input.

The instances of InputRequest itself will accept any input, but subclasses may use stricter validations. org.apache.tools.ant.input.MultipleChoiceInputRequest should be used if the user input must be part of a predefined set of choices.

9.6 Using Ant Tasks Outside of Ant

9.6.1 Rationale

Ant provides a rich set of tasks for buildfile creators and administrators. But what about programmers? Can the functionality provided by Ant tasks be used in java programs?

Yes, and its quite easy. Before getting into the details, however, we should mention the pros and cons of this approach:

Pros Robust Ant tasks are very robust. They have been banged on by many people. Ant tasks have been used in many different contexts, and have therefore been instrumented to take care of a great many boundary conditions and potentially obscure errors. Cross Platform Ant tasks are cross platform. They have been tested on all of the volume platforms, and several rather unusual ones (Netware and OS/390, to name a few). Community Support Using Ant tasks means you have less of your own code to support. Ant code is supported by the entire Apache Ant community.

Cons Dependency on Ant Libraries Obviously, if you use an Ant task in your code, you will have to add "ant.jar" to your path. Of course, you could use a code optimizer to remove the unnecessary classes, but you will still probably require a chunk of the Ant core. Loss of Flexibility At some point, if you find yourself having to modify the Ant code, it probably makes more sense to "roll your own." Of course, you can still steal some code snippets and good ideas. This is the beauty of open source!

Example Let's say you want to unzip a zip file programmatically from java into a certain directory. Of course you could write your own routine to do this, but why not use the Ant task that has already been written?

In my example, I wanted to be able to unzip a file from within an XSLT Transformation. XSLT Transformers can be extended by plugging in static methods in java. I therefore need a function something like this:

/** * Unzip a zip file into a given directory. * * @param zipFilepath A pathname representing a local zip file * @param destinationDir where to unzip the archive to */ static public void unzip(String zipFilepath, String destinationDir)

The Ant task to perform this function is org.apache.tools.ant.taskdefs.Expand. All we have to do is create a dummy Ant Project and Target, set the Task parameters that would normally be set in a buildfile, and call execute().

First, let's make sure we have the proper includes:

import org.apache.tools.ant.Project; import org.apache.tools.ant.Target; import org.apache.tools.ant.taskdefs.Expand; import java.io.File;

The function call is actually quite simple:

static public void unzip(String zipFilepath, String destinationDir)

final class Expander extends Expand public Expander() project = new Project(); project.init(); taskType = "unzip"; taskName = "unzip"; target = new Target(); Expander expander = new Expander(); expander.setSrc(new File(zipfile)); expander.setDest(new File(destdir)); expander.execute();

In actual practice, you will probably want to add your own error handling code and you may not want to use a local inner class. However, the point of the example is to show how an Ant task can be called programmatically in relatively few lines of code.

The question you are probably asking yourself at this point is: How would I know which classes and methods have to be called in order to set up a dummy Project and Target? The answer is: you don't. Ultimately, you have to be willing to get your feet wet and read the source code. The above example is merely designed to whet your appetite and get you started. Go for it!

Chapter 10

Ant API

[tbd]

Chapter 11

License

```
/*
 * ============================================================================
 *                   The Apache Software License, Version 1.1
 * ============================================================================
 *
 *    Copyright (C) 2000-2002 The Apache Software Foundation. All
 *    rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without modifica-
 * tion, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of  source code must  retain the above copyright  notice,
 *    this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright notice,
 *    this list of conditions and the following disclaimer in the documentation
 *    and/or other materials provided with the distribution.
 *
 * 3. The end-user documentation included with the redistribution, if any, must
 *    include  the following  acknowledgment:  "This product includes  software
 *    developed  by the  Apache Software Foundation  (http://www.apache.org/)."
 *    Alternately, this  acknowledgment may  appear in the software itself,  if
 *    and wherever such third-party acknowledgments normally appear.
 *
 * 4. The names "Ant" and  "Apache Software Foundation"  must not be used to
 *    endorse  or promote  products derived  from this  software without  prior
 *    written permission. For written permission, please contact
 *    apache@apache.org.
 *
 * 5. Products  derived from this software may not  be called "Apache", nor may
 *    "Apache" appear  in their name,  without prior written permission  of the
 *    Apache Software Foundation.
 *
 * THIS SOFTWARE IS PROVIDED ‘‘AS IS’’ AND ANY EXPRESSED OR IMPLIED WARRANTIES,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
 * FITNESS  FOR A PARTICULAR  PURPOSE ARE  DISCLAIMED.  IN NO  EVENT SHALL  THE
 * APACHE SOFTWARE  FOUNDATION  OR ITS CONTRIBUTORS  BE LIABLE FOR  ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL,  EXEMPLARY, OR CONSEQUENTIAL  DAMAGES (INCLU-
 * DING, BUT NOT LIMITED TO, PROCUREMENT  OF SUBSTITUTE GOODS OR SERVICES; LOSS
 * OF USE, DATA, OR  PROFITS; OR BUSINESS  INTERRUPTION)  HOWEVER CAUSED AND ON
 * ANY  THEORY OF LIABILITY,  WHETHER  IN CONTRACT,  STRICT LIABILITY,  OR TORT
 * (INCLUDING  NEGLIGENCE OR  OTHERWISE) ARISING IN  ANY WAY OUT OF THE  USE OF
 * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * This software  consists of voluntary contributions made  by many individuals
 * on behalf of the  Apache Software Foundation.  For more  information  on the
 * Apache Software Foundation, please see <http://www.apache.org/>.
 *
 */
```

Chapter 12

Feedback and Troubleshooting

If things do not work, especially simple things like ant -version, then something is wrong with your configuration. Before filing bug reports and emailing all the ant mailing lists

Check your environment variables. Are `ANT_HOME` and `JAVA_HOME` correct? If they have quotes or trailing slashes, remove them. Unset CLASSPATH; if that is wrong things go horribly wrong. Ant does not need the CLASSPATH variable defined to anything to work. Make sure there are no versions of crimson.jar or other XML parsers in JRE/ext Is your path correct? is Ant on it? What about JDK/bin? have you tested this? If you are using Jikes, is it on the path? A createProcess error (especially with ID=2 on windows) usually means executable not found on the path. Which version of ant are you running? Other applications distribute a copy -it may be being picked up by accident. If a task is failing to run is optional.jar in `ANT_HOME`/lib? Are there any libraries which it depends on missing? If a task doesn't do what you expect, run ant -verbose or ant -debug to see what is happening If you can't fix your problem, start with the Ant User Mailing List . These are other ant users who will help you learn to use ant. If they cannot fix it then someone may suggest filing a bug report, which will escalate the issue. Remember of course, that support, like all open source development tasks, is voluntary. If you haven't invested time in helping yourself by following the steps above, it is unlikely that anyone will invest the time in helping you.

Also, if you don't understand something, the Ant User Mailing List is the place to ask questions. Not the developer list, nor the individuals whose names appears in the source and documentation. If they answered all such emails, nobody would have any time to improve ant.

To provide feedback on this software, please subscribe to the Ant User Mailing List

If you want to contribute to Ant or stay current with the latest development, join the Ant Development Mailing List

Archives of both lists can be found at http://archives.apache.org/eyebrowse/ViewLists. A searchable archive can be found at http://marc.theaimsgroup.com. If you know of any additional archive sites, please report them to the lists.

Chapter 13

Authors

Stephane Bailliez (sbailliez@imediation.com)
Nicola Ken Barozzi (nicolaken@apache.org)
Jacques Bergeron (jacques.bergeron@dogico.com)
Stefan Bodewig (stefan.bodewig@freenet.de)
Patrick Chanezon (chanezon@netscape.com)
James Duncan Davidson (duncan@x180.com)
Tom Dimock (tad1@cornell.edu)
Peter Donald (donaldp@apache.org)
dIon Gillard (dion@apache.org)
Erik Hatcher (ehatcher@apache.org)
Diane Holt (holtdl@yahoo.com)
Bill Kelly (bill.kelly@softwired-inc.com)
Arnout J. Kuiper (ajkuiper@wxs.nl)
Conor MacNeill
Stefano Mazzocchi (stefano@apache.org)
Erik Meade (emeade@geekfarm.org)
Sam Ruby (rubys@us.ibm.com)
Nico Seessle (nico@seessle.de)
Jon S. Stevens (jon@latchkey.com)
Magesh Umasankar
Roger Vaughn (rvaughn@seaconinc.com)
Dave Walend (dwalend@cs.tufts.edu)
Phillip Wells (philwells@rocketmail.com)
Craeg Strong (cstrong@arielpartners.com)
Version: 1.5.3
$Id: credits.html,v 1.15.2.8 2003/04/08 14:17:04 conor Exp $