



Ant Overview





Sang Shin
sang.shin@sun.com

Nick Zoda

Revision History

- 12/31/2002: Initial Draft
- 01/11/2003: Draft Version With Examples
- 01/23/2003: Added Speaker Notes
- 06/12/2005: Rewrote many slides (Sang)

Agenda

- What is 'Ant'?
- Ant tasks
 - Archive tasks
 - Compile tasks
 - File Tasks
 - Property Tasks
 - Execution tasks
 - Mail Tasks
- Ant Examples



What is Ant?



Ant Overview

- Apache Jakarta Project
- Open Source, Community Development
 - <http://jakarta.apache.org/ant/>
- Java Based Tool
 - Extensible Using Java Classes
- XML Based Configuration 'Build' File
 - Easy to Read and Modify
- Cross Platform
 - Can Use OS-Specific Commands If Necessary

The Ant Advantage

- Build IN Java, USING Java, and FOR Java
- Supports Java Tools (javac, javadoc, etc.)
- XML Build File is Easier to Build, Read, and Maintain than MAKE file
- Easier to Extend
- Supports Cross Platform Java Development

Ant Advantages (cont'd)

- Ant is much faster than using MAKE
 - MAKE is shell-based
 - Each command is a new process
 - Ant runs within the JVM
 - Each command is executed from within the JVM
 - Tools like javac are new threads – not new processes
 - Compiling large numbers of java source files is MUCH, MUCH faster with Ant
- Ant's Debug Options are very helpful
 - '-verbose' and '-debug' options

Still More Ant Advantages

- XML File Maintenance
 - XML much easier to read than MAKEFILE
 - User does not need to know all command line interface options to tools that can be called programmatically
 - Even OS-Specific commands can be setup in 'TaskDefs' and/or included from other sources
 - See <http://jakarta.apache.org/ant/manual/OptionalTasks> for a list of many optional tasks

Ant Basics

- Each 'Project' has a Build File
 - Default build file name is 'build.xml'
 - Can Specify with '-buildfile' command line option
- Each Build File is made up of Targets
 - Examples are: 'compile', 'test', 'install', 'clean', etc.
- Each Target is made up of Tasks
 - Invoke a command or another program

Ant Basics

- Targets can have Dependencies
 - Examples: 'install' depends on 'compile'
 - Can handle cascading dependencies
 - Dependency executed only if required



Ant Tasks



Ant Tasks

- Property Tasks
- Archive tasks
- Compile tasks
- File Tasks
- Execution tasks
- Mail Tasks
- Many more



Property Tasks



property

- Sets a property (by name and value pair), 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

property

sets the property foo.dist to the value "dist".

```
<property name="foo.dist" value="dist"/>
```

reads a set of properties from a file called "foo.properties".

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

reads a set of properties from the URL address

"http://www.mysite.com/bla/props/foo.properties".

```
<property url="http://www.mysite.com/bla/props/foo.properties"/>
```

read a set of properties from "\${user.home}/.ant-global.properties".

```
<property file="${user.home}/.ant-global.properties"/>
```

Example property file

```
build.compiler=jikes  
deploy.server=lucky  
deploy.port=8080  
deploy.url=http://${deploy.server}:${deploy.port}/
```

System Properties

java.version Java Runtime Environment version
java.vendor Java Runtime Environment vendor
java.vendor.url Java vendor URL
java.home Java installation directory
java.vm.specification.version Java Virtual Machine specification version
java.vm.specification.vendor Java Virtual Machine specification vendor
java.vm.specification.name Java Virtual Machine specification name
java.vm.version Java Virtual Machine implementation version
java.vm.vendor Java Virtual Machine implementation vendor
java.vm.name Java Virtual Machine implementation name
java.specification.version Java Runtime Environment specification version
java.specification.vendor Java Runtime Environment specification vendor
java.specification.name Java Runtime Environment specification name

System Properties

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



Archiving Tasks



jar

jars all files in the `${build}/classes` directory into a file called `app.jar` in the `${dist}/lib` directory

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

jars all files in the `${build}/classes` directory into a file called `app.jar` in the `${dist}/lib` directory. Files with the name `Test.class` are excluded

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

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
- 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

war

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

war

then the war file myapp.war created with

<fileset> specifies set of html or JSP files, which should go under root

<lib> specifies set of jar files, which should go under ./WEB-INF/lib

<classes> specifies set of java classes, which should go under ./WEB-INF/classes

<zipfileset> specifies set of resources, which should go under images

```
<war destfile="myapp.war" webxml="src/metadata/mymap.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>
```

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



Compiling Tasks



javac

Compiles all .java files under the `{src}` directory, and stores the .class files in the `{build}` directory. The classpath used includes xyz.jar, and compiling with debug information is on. The source level is 1.4, so you can use assert statements.

```
<javac srcdir="{src}"  
    destdir="{build}"  
    classpath="xyz.jar"  
    debug="on"  
    source="1.4"  
/>
```

javac

Compiles .java files under the `${src}` and `${src2}` directories, and stores the .class files in the `${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. You didn't specify a source or target level, so the actual values used will depend on which JDK you ran Ant with.

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



File Tasks



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

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">  
  <include name="**/*.java"/>  
  <exclude name="**/*Test*" />  
</fileset>
```

Groups the same files as the example above, but using the `<filename>` selector.

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

fileset

Groups the same files as the example in the previous slide, but also establishes a PatternSet that can be referenced in other <fileset> elements, rooted at a different directory.

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

Groups all files in directory \${client.src}, using the same patterns as the above example.

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

copy

- 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

copy

Copy a directory to another directory

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

Copy a set of files (files in src_dir directory excluding any java source files) to a directory

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

delete

deletes the file /lib/ant.jar.

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

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

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

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

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



Execution Tasks



java

- 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.

java

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

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

exec

- 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

exec

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

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

Starts the `${browser}` with the specified `${file}` and end the ant process.
The browser will let be open.

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

dependset

- 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.

dependset

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

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



Mail Tasks



mail

Sends an eMail from config@myisp.com to all@xyz.com with a subject of Test Build. Replies to this email will go to me@myisp.com. Any zip files from the dist directory are attached. 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.

```
<mail mailhost="smtp.myisp.com" mailport="1025" subject="Test build">  
  <from address="config@myisp.com"/>  
  <replyto address="me@myisp.com"/>  
  <to address="all@xyz.com"/>  
  <message>The ${buildname} nightly build has completed</message>  
  <fileset dir="dist">  
    <include name="**/*.zip"/>  
  </fileset>  
</mail>
```



Misc. Tasks



echo

- 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

echo

Some examples

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

```
<echo message="Embed a line break:${line.separator}"/>
```

A message which only appears in -debug mode.

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

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

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

tsttmp

- Sets the DSTAMP, TSTAMP, and TODAY properties in the current project, based on the current date and time
- 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"
- 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.

tsttmp

Sets the standard DSTAMP, TSTAMP, and TODAY properties according to the default formats.

```
<tstamp/>
```

Sets three properties with the standard formats, prefixed with "start.": start.DSTAMP, start.TSTAMP, and start.TODAY.

```
<tstamp prefix="start"/>
```



Example Build.xml



Example build.xml (page 1)

```
<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>
```

Example build.xml (page 2)

```
<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>
```

Example build.xml (page 3)

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

Example Build.xml: Hello2 from J2EE 1.4 Tutorial



Example – Hello2 build.xml from J2EE 1.4 Tutorial

```
<project name="hello2-example" default="build" basedir=". ">
  <target name="init">
    <tstamp/>
  </target>

  <!-- Configure the context path for this application -->
  <property name="example" value="hello2" />

  <!-- Configure properties -->
  <property file="../../common/build.properties"/>
  <property file="../common/build.properties"/>

  &targets;
  &webtargets;

  <target name="build" depends="copy"
    description="Compile app Java files" >
    <javac srcdir="src" destdir="${build}">
      <include name="**/*.java" />
      <classpath refid="classpath"/>
    </javac>
  </target>
</project>
```

<j2eetutorial14>/examples/web/common/targets.xml (page 1)

```
<target name="prepare" depends="init"
description="Create build directories.">
  <mkdir dir="${build}" />
</target>
```

```
<target name="copy" depends="prepare"
description="Copy HTML and JSP pages" >
  <copy todir="${build}">
    <fileset dir="web">
      <include name="**/*.html" />
      <include name="**/*.jsp" />
      <include name="**/*.jspf" />
      <include name="**/*.jspx" />
      <include name="**/*.gif" />
      <include name="**/*.xml" />
      <include name="**/*.tld" />
      <include name="**/*.tag" />
      <include name="**/*.jpg" />
      <include name="**/*.css" />
    </fileset>
  </copy>
</target>
```

<j2eetutorial14>/examples/web/common/targets.xml (page 2)

```
<target name="create-war" depends="build"
  description="Packages the WAR file">
  <echo message="Creating the WAR..." />
  <delete file="${assemble.war}/${war.file}" />
  <delete dir="${assemble.war}/WEB-INF" />
  <copy todir="${assemble.war}/WEB-INF">
    <fileset dir=".">
      <include name="*.xml" />
      <exclude name="build.xml" />
      <exclude name="web.xml" />
    </fileset>
  </copy>
  <copy todir="${assemble.war}/WEB-INF/classes/">
    <fileset dir="${build}">
      <include name="**/*.class" />
    </fileset>
  </copy>
  <copy todir="${assemble.war}/WEB-INF/tags">
    <fileset dir="${build}">
      <include name="*.tag" />
    </fileset>
  </copy>
```

<j2eetutorial14>/examples/web/common/targets.xml (page 2)

```
<copy todir="${assemble.war}/WEB-INF">
  <fileset dir="${build}">
    <include name="*.tld" />
  </fileset>
</copy>
<copy todir="${assemble.war}">
  <fileset dir="${build}">
    <include name="*.jsp" />
    <include name="*.gif" />
  </fileset>
</copy>
<war destfile="${assemble.war}/${war.file}"
  webxml="./web.xml" filesonly="true" >
  <fileset dir="${assemble.war}" includes="WEB-INF/**, *.jsp, *.gif" />
</war>
<copy file="${assemble.war}/${war.file}" todir="." />
</target>
```