Using Java Ant

Andy Verkeyn
Department of Information Technology
Ghent University

April 2003
Version 1.0

Contents

- **Introduction**
  - What is ant?
  - How to use ant?
    - **Project skeleton, Running ant**
  - Basics
    - **Targets, Properties, Built-in tasks**
  - Example
- **Advanced topics**
  - Specifying directories and files
  - Sample tasks
  - Using own tasks

Contents

- **Introduction**
  - What is ant?
  - How to use ant?
    - **Project skeleton, Running ant**
  - Basics
    - **Targets, Properties, Built-in tasks**
  - Example
- **Advanced topics**
  - Specifying directories and files
  - Sample tasks
  - Using own tasks

What is ant ?

- **Ant**
  - A free, java based build tool (cfr. make)
  - Does not rely on shell commands
  - Executes java classes
    - **Platform independent**
  - Uses XML configuration files
  - ANT = Another Neat Tool (or ants = good at building things)
  - More info: http://ant.apache.org
- **History**
  - Created by James Duncan Davidson as part of the Tomcat (webserver) project
  - First official stand alone release in Jul. 2000 (version 1.1)

What is XML ?

- **XML = Extensible Markup Language**
  - Standardized by W3C (World Wide Web Consortium)
  - Looks a lot like HTML... but better structured!
- **Syntax rules**
  - Elements are formed by a start tag and a matching end tag

    `<element> Text </element>`

    Empty element : `<element/>`
  - There must be a single root element
  - All elements are properly nested in the root (tree-structure)
  - A start tag can contain attributes with double quoted values

    `<element att1="val1" att2="val2"> Text </element>`
  - Case sensitive

Ant project XML skeleton

- **Skeleton**

```
<project name="myproj" default="mytarget" basedir=".">
  <description>A simple example</description>
  <target name="mytarget" description="Sample">
    <taskname id="taskid" attrib1="value1">
    </taskname>
  </target>
</project>
```

- **Remarks**
  - Only the `project-default` attribute is required.
  - The `project-description` element is optional.

Running ant

- **Running ant**
  - `ant [options] [target]`
  - Default: searches for build.xml in the current directory
  - If no target is specified, the default project target is used.
- **Options**
  - `-buildfile` *`filename`*: uses *filename* instead of build.xml
  - `-f` *`filename`*
  - `-find` *`filename`*: searches for *filename* from the current directory towards the root until it is found
  - `-Dproperty=value`: sets or overwrites a property

Basics: Target element

- **Syntax**

```
<target name="C" depends="B,A" description="Sample">
```

- **Remarks**
  - Only the `name` attribute is required.
  - The `depends` targets are executed from left to right.
- **Optional attributes**
  - `if="property"`
  - `unless="property"`
  - The target is only executed "if" or "unless" the property is set. The value of the property is not important, only that is has been set.

Basics: Using properties

- **Property element**

```
<property name="version" value="1.0" />
<property name="build" location="build" />
<property name="class" location="${build}/classes" />
<property file="foo.properties" />
```

- **Remark**
  - The attribute `location` always resolves to an absolute path.
  - Ant properties are immutable!
- **Built-in properties**
  - Java system properties (E.g.: `os.name`, `user.home`,...)
  - `basedir`: see project tag
  - `ant.file`: absolute path of the build file
  - `ant.version`, `ant.project.name`, `ant.java.version`

Basics: Built-in tasks

- **Compile tasks**
  - javac
  - rmic
- **Archive tasks**
  - jar, zip, unzip
- **Documentation tasks**
  - javadoc
- **Execution tasks**
  - ant, antcall
  - java
- **File tasks**
  - copy, delete, move
  - mkdir
- **Property tasks**
  - basename, dirname
  - condition
  - property
- **Miscellaneous tasks**
  - echo
  - input
- **It is also possible to write your own tasks in java**

Example

```
<project name="MyProject" default="dist" basedir=".">
  <description>Simple example build file </description>
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist" location="dist"/>
  <target name="compile" description="compile source">
    <mkdir dir="${build}"/>
    <javac srcdir="${src}" destdir="${build}"/>
  </target>
  <target name="dist" depends="compile">
    <mkdir dir="${dist}/lib"/>
    <jar destfile="${dist}/lib/MyProject.jar"
         basedir="${build}"/>
  </target>
</project>
```

Contents

- Introduction
  - What is ant?
  - How to use ant?
    - **Project skeleton, Running ant**
  - Basics
    - **Targets, Properties, Built-in tasks**
  - Example
- **Advanced topics**
  - Specifying directories and files
  - Sample tasks
  - Using own tasks

Specifying directories and files

- **Specifying files**

```
<fileset dir="basedir"
         includes="*.class,**/*.gif">
         excludes="*.java">
</fileset>
```

- **Remark**
  - Only the `dir` attribute is required.
- **Wildcards**
  - * and ? : as usual
  - ** : matches zero or more subdirectories
  - shorthand: `"subdir/"` is expanded to `"subdir/**"`.

Specifying directories and files

- **Using elements**

```
<fileset dir="basedir">
  <include name="*.class" />
  <include name="**/*.gif" />
  <exclude name="*.java" />
</fileset>
```

- **Remarks**
  - Only one pattern can be specified with these elements.
  - Additionally, they accept the `if` and `unless` attributes.
- **Specifying directories**
  - Same syntax with `dirset` element instead of `fileset`.

Specifying directories and files

- **Specifying path structures**

```
<path id="base.path">
  <pathelement path=".;myApp" />
  <pathelement location="lib/helper.jar" />
</path>
<path id="test.path">
  <pathelement refid="base.path" />
  <pathelement location="testclasses" />
</path>
```

- **Attributes**
  - `path`: a list of ":" or ";" separated paths.
  - `location`: single directory or jar file.
  - `refid`: includes a previously defined path structure

Specifying directories and files

- **Examples of path structure elements supporting the same syntax**
  - `classpath`, `sourcepath`, ...
- **Notational shortcut**
  - The `path` and `classpath` elements also accept the attributes `path` and `location` directly
  - `<path id="base.path" path="${classpath}"/>`
- **Combined elements**
  - Dirsets and filesets can be specified as nested elements

Specifying directories and files

- **Example**

```
<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>
</classpath>
```

Sample tasks: javac

- **Syntax**

```
<javac srcdir="sourcedir" includes="..." excludes="..."
       destdir="build"  classpath=".:my.jar"
       deprecation="on" listfiles="on">
</javac>
```

- **Remarks**
  - It is also allowed to specify only `srcdir` as attribute and use the `include` and `exclude` fileset elements.
  - `srcdir` and `classpath` are path-like structures and can be defined using nested elements `src` and `classpath`.
  - Only java files without a class file or class files that are older than their java file will be compiled.

Sample tasks: rmic

- **Syntax**

```
<rmic base="classes" includes="**/Remote*.class"
      excludes="..." classpath=".:my.jar" />
<rmic base="classes" classname="package.MyClass"
      classpath=".:my.jar" />
```

- **Remarks**
  - `base`: the directory to put the compiled files (required)
  - `classname`: name of a single class to compile.
  - Only `base` is required.
  - It is also allowed to specify only `base` as attribute and use the `include` and `exclude` fileset elements.
  - `classpath` is a path-like structure and can be defined using the nested element `classpath`.

Sample tasks: jar

- **Syntax**

```
<jar destfile="test.jar" basedir="class"
     includes="*.class,*.gif" excludes="*.java,*.txt"
     manifest="test.mf">
</jar>
```

- **Specifying source files**
  - One or more fileset elements can be used instead of the `basedir`, `includes` and `excludes` attributes.
  - It is also allowed to specify only `basedir` as attribute and use the `include` and `exclude` elements.

Sample tasks: jar

- **The manifest file can also be included inline**

```
<jar destfile="test.jar">
  <fileset dir="class">
    <include name="*.class />
    <include name=".gif" />
    <exclude name="*.java />
    <exclude name="*.txt" />
  </fileset>
  <manifest>
    <attribute name="Manifest-Version" value="1.0" />
    <attribute name="Built-By" value="${user.name}" />
  </manifest>
</jar>
```

Sample tasks: javadoc

- **Syntax**

```
<javadoc sourcepath="." sourcefiles="f1.java,f2/*.java"
         destdir="docs"
         windowtitle="..." doctitle="..."
         overview="true" version="true" author="true"
         nodeprecated="true" />
```

- **Remarks**
  - At least one of the `sourcepath` and `sourcefiles` attributes or a nested `fileset` element is required.
  - Several attributes are also allowed as nested elements, e.g. doctitle.
  - `sourcepath` is a path-like structure and can be defined using the nested element `sourcepath`.

Sample tasks: javadoc

- **Example**

```
<javadoc destdir="docs/api" windowtitle="Test API"
         version="true" author="true“>
  <fileset dir="src">
    <include name="com/dummy/test/**" />
    <exclude name="com/dummy/test/doc-files/**"/>
  </fileset>
  <doctitle>Test</doctitle>
</javadoc>
```

Sample tasks: ant

- **Runs ant on a supplied (subproject) file.**
- **Syntax**

```
<ant antfile="subproject/sub.xml" dir="." target="tgt">
  <property name="passProperty" value="propVal"/>
</ant>
```

- **Remarks**
  - No attributes are required, defaults
    - **antfile = "build.xml" (relative to dir directory)**
    - **dir = project's base directory**
    - **target = default target of the buildfile.**
  - By default all properties will be inherited by the subproject
    - **inheritAll="false"**

Sample tasks: antcall

- **Calls a target within the same buildfile**
- **Syntax**

```
<antcall target="otherTarget">
  <param name="passParam" value="paramVal"/>
</antcall>
```

- **Remarks**
  - The `target` attribute is required.
  - Properties can be passed using the `param` element.
  - After executing the child target, control returns to the caller.

Sample tasks: java

- **Runs a java class**
- **Syntax**

```
<java classname="ClassFile" jar="JarFile" fork="true"
      classpath=".:myjar.jar">
  <arg name="argValue" />
</java>
```

- **Remarks**
  - One of the attributes `classname` or `jar` is required. If a jarfile is specified, fork must be set true.
  - `fork`: launches the Java class in a new JVM.
  - `classpath` is a path-like structure and can be defined using the nested element `classpath`.

Sample tasks: copy

- **Copies one or more files**
- **Syntax**

```
<copy file="myfile.java" tofile="yourfile.java" />
<copy todir="targetdir" overwrite="true">
  <fileset dir="sourcedir" />
</copy>
```

- **Remarks**
  - If the `file` attribute is used, either `tofile` or `todir` must be set.
  - If multiple files are specified using one ore more fileset elements, only the `todir` attribute is allowed (and required).
  - Default for `overwrite` attribute is false.

Sample tasks: move

- **Move one or more files (same syntax as copy)**
- **Syntax**

```
<move file="myfile.java" tofile="yourfile.java" />
<move todir="targetdir" overwrite="true">
  <fileset dir="sourcedir" />
</move>
```

- **Remarks**
  - If the `file` attribute is used, either `tofile` or `todir` must be set.
  - If multiple files are specified using one ore more fileset elements, only the `todir` attribute is allowed (and required).
  - Default for `overwrite` attribute is false.

Sample tasks: delete

- **Deletes files and/or directories**
- **Syntax**

```
<delete file="myfile.java" />
<delete dir="toremove" />
<delete>
  <fileset dir="toremove" />
</delete>
```

- **Remark**
  - Either the `file` attribute, the `dir` attribute or one or more fileset elements must be provided.

Sample tasks: mkdir

- **Creates a directory.**
- **Syntax**

```
<mkdir dir="bin/classes" />
```

- **Remarks**
  - Parent directories are also created if necessary.

Sample tasks: basename/dirname

- **Sets the property to the base file name**

```
<basename property="jar.file" file="lib/my.jar"/>
```

  - Result: jar.file="my.jar"
- **Sets the property to the absolute directory name**

```
<dirname property="jar.dir" file="lib/my.jar"/>
```

  - Result: jar.dir="base_project_directory/lib"

Sample tasks: condition

- **Sets a property if a condition is true**
- **Syntax**

```
<condition property="propName" value="trueValue">
  <nested-conditions />
</condition>
```

- **Nested conditions**
  - `<not> <and> <or>`
  - `<istrue/isfalse value="${p}"> <isset property="p">`
  - `<available>`: true if a class, file or resource exists
  - `<uptodate>`: true if a target file is more up to date than a source file
  - `<equals>`: true if two given argument strings are equal

Sample tasks: condition

- **If-then-else simulation**

```
<target name="if-part">
  <condition property="if-cond">
    <available file="name.jar" />
  </condition>
  <antcall target="then-part" />
  <antcall target="else-part" />
</target>
```

IF

```
<target name="then-part" if="if-cond">
</target>
```

THEN

```
<target name="else-part" unless="if-cond">
</target>
```

ELSE

Sample tasks: echo/input

- **Show messages and ask input**

```
<echo message="Text to show" />

<input message="Prompt:" validargs="y,n"
       addproperty="answer" />
```

- **Remarks**
  - `validargs`: list of acceptable input (case sensitive)
  - `addproperty`: property containing the given input

Using own tasks

- **Writing own tasks in Java**
  - Derive a java class from `org.apache.tools.ant.Task`.
  - Write a setter method for each attribute, e.g. message:

    ```
    public void setMessage(String);
    ```

    - **Ant does simple type conversions (e.g. primitives or String)**
  - To support character data in your element:

    ```
    public void addText(String);
    ```

  - To support nested elements, e.g. inner:

    ```
    public InnerImplementation createInner();
    ```

  - Implement the task:

    ```
    public void execute() throws
      org.apache.tools.ant.BuildException;
    ```

Using own tasks

- **Example**

```
import org.apache.tools.ant.BuildException;
import org.apache.tools.ant.Task;
public class MyOwnTask extends Task {
  private String msg;
  public void execute() throws BuildException {
    System.out.println(msg);
  }
  public void setMessage(String msg) {
    this.msg = msg;
  }
}
```

Using own tasks

- **Deploying**
  - Include your class in the classpath before invoking ant
  - Add a `taskdef` element to the project
  - Use the task as other ant tasks
- **Example**

```
<project name="OwnTaskExample" default="main"
         basedir=".">
  <taskdef name="mytask" classname="MyOwnTask"/>
  <target name="main">
    <mytask message="Hello World!"/>
  </target>
</project>
```