Project build with Ant

D. Conan

Internship course
February 2008

Outline

1 Motivations: Ant beyond make-like tools

- MAKE was a marked improvement over shell script
  - Abstraction of specific shell commands into a more general and consistent syntax
  - Determines automatically which pieces of a large program need to be rebuilt and issues the commands
- But...
  - MAKE mainly/historically targets the language C et C++
    - Implicit rules and pattern rules
      - ★ Execute `make -p` in a directory with no Makefile
      - ★ Default suffixes: `.out .a .ln .o .c .cc .C .cpp .p .f .F .r .y .l .s .S .mod .sym .def .h .info .dvi .tex .texinfo .texi .txinfo .w .ch .web .sh .elc .el`
  - Make mainly/historically targets Unix-like operating systems
    - Machine-specific aspects of Make $\Longrightarrow$ less portable than the code to build
- Thus...
  - ANT builds software across multiple platforms with plug-ins for Java

2 References and principles

- References
  - Web site: `http://ant.apache.org`
  - A short guide [Edlich, 2002]
  - Manual pages: `http://ant.apache.org/manual`
  - Tutorials:
    `http://ant.apache.org/manual/tutorial-HelloWorldWithAnt.html`
    `http://supportweb.cs.bham.ac.uk/docs/tutorials/docsystem/build/tutorials/ant/ant.html`
- Principles
  - Build files are "procedural"
  - XML-based build and configuration files
  - Pure Java $\Longrightarrow$ Extensible using Java classes
    - Target tree where various tasks get executed
    - Each task is run by an object that implements a Task interface
  - Organisation of source code and other artefacts: `src`, `build`, `dist`...

3 XML-based build file structure

- Project: Upper most XML element containing targets
  - `name`, `default` target, and `basedir` for path calculations
- Targets: Set of tasks you want to be executed
  - The `depends` attribute specifies the order in which targets should be executed
    - Automatic management of the chain of dependencies
    - Can be condition-based (see `if` and `unless` attributes)
- Task: A piece of code that can be executed
  - Can have multiple attributes (or arguments)
  - Common structure: `<name attribute1="value1" attribute2="value2" ... />`
  - Catalog of predefined tasks + you can build specific tasks
- Properties of the project
  - A name and a value
  - <property name="product.name" value="mediatheque"/>
  - `<echo>${product.name}=${buildir}</echo>`

4 Example build file

```
<project name="MyProject" default="dist" basedir="." description="Example">
  <property name="src" location="${basedir}/src"/>
  <property name="build" location="${basedir}/build"/>
  <property name="dist"  location="${basedir}/dist"/>
  <target name="init" description="create the build directory structure">
    <mkdir dir="${build}"/>
  </target>
  <target name="compile" depends="init" description="compile the source ">
    <javac srcdir="${src}" destdir="${build}"/>
  </target>
  <target name="dist" depends="compile" description="generate the distribution">
    <mkdir dir="${dist}/lib"/>
    <jar jarfile="${dist}/lib/MyProject.jar" basedir="${build}"/>
  </target>
  <target name="clean" description="clean up" >
    <delete dir="${build}"/><delete dir="${dist}"/>
  </target>
</project>
```

5 Running Ant

- No argument $\Longrightarrow$ build.xml file in the current directory
  - Use a build file other than `build.xml`: `-buildfile file`
- Assign properties on the command line: `-Dproperty=value`
- Execute a target different from the default one
  - First, `ant -p` or `ant help`
  - Next, `ant` *targetName*

6 Path-like structure

- `path id`, `path refid`, and `pathelement`

```
<path id="compilecp">
  <fileset dir="${externals}">
    <include name="**/*.jar"/>
  </fileset>
</path>
<path id="executecp">
  <path refid="compile" />
  <pathelement location="${build}"/>
</path>
<target name="compile" depends="init" description="compile the source ">
  <javac srcdir="${src}" destdir="${build}">
      <classpath refid="compilecp"/>
  </javac>
</target>
```

7 Some Ant tasks *

- Archive tasks: BUnzip2, BZip2, GUnzip, GZip, Tar, Untar, etc.
- Compile tasks: Depend, Javac, Rmic, etc.
- Deployment tasks: ServerDeploy...
- Documentation tasks: Javadoc/Javadoc2, etc.
- Execution tasks: Ant, AntCall, Exec, Java, Subant, etc.
- File tasks: Chmod, Chown, Concat, Copy, Delete, Filter, Get, Mkdir, Move, Patch, Touch, etc.
- Property tasks: Available, Basename, Condition, Dirname, Echoproperties, LoadProperties, Property, Uptodate, etc.
- Remote tasks: FTP, Scp, Sshexec, Telnet, etc.
- SCM tasks for CVS, Suversion, etc.
- Testing tasks: Junit, JunitReport

8 More on Ant *

- Plug-in mechanism for using third party external tasks
  - Writing Tasks `http://ant.apache.org/manual/tutorial-writing-tasks.html`
- `filesets` and `patternsets`: Bundling files by inclusion/exclusion
- Tasks using properties, filesets and paths
  `http://ant.apache.org/manual/tutorial-tasks-filesets-properties.html`

References

[Edlich, 2002] Edlich, S. (2002). *Ant: Pocket Guide*. O'Reilly.