Ant

Kurt Schmidt

Dept. of Computer Science, Drexel University

July 28, 2016

Originally from Bruce Char & Vera Zaychik

Intro

Ant

Ant is a tool for managing Java builds and distributions.

- A bit like `make`
- Platform agnostic
  - Run inside the JVM
  - Extended with Java classes
- Reads an XML configuration file
  - Called an *ant* or *build* file
  - `build.xml`, but default

`http://ant.apache.org`

Ant v. Make

- Ant runs inside the JVM
  - Cross platform
  - Uses XML, which is well-formed
  - Can only do what the JVM can do
  - Doesn't manage non-Java portions of a project
- Make commands are run in a local shell
  - More handy
  - Less portable
  - Syntax is a bit more arcane
    - Relies on whitespace

Anatomy

The Build File

- Contains a single *project*
- A project contains:
  - One more more *tasks*
  - Optionally, a *description*
  - Optionally, other named resources, like *path*
- A task:
  - Is a piece of code that can be executed
  - Can have multiple attributes
  - Can be assigned an `id` attribute
  - Can contain other resources

Tasks

Some Capabilities of Ant

- Ant has a core set of tasks. These allow you to:
  - Compile and run Java programs
  - Create directories
  - Move, copy, and delete files
  - Create Javadoc files from source
  - Send email
  - Fetch files from over a network
- Other tasks can be supplied
  - `junit` – Found in `junit.jar`
- Each task is defined by a Java class
  - Inherited from `org.apache.tools.ant.Task`

Common Tasks

`<javac srcdir=`*`dir`* `[includes=`*`fileList`*`]/>` Compiles Java source files

`<java classname=`*`class`*`/>` Runs a Java application

`<java fork='yes' jar=`*`jarFile`*`/>` Runs a Java application from a jarfile

`javadoc` Runs JavaDoc on source files

`mkdir` Creates a directory, including missing parent directories

`move`, `copy` Move/copy files

`delete` Removes files or directories

Build Files

Example – javac, delete & fileset

Compile all *.java files in the current directory, using javac

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project default="compile">
   <description>
      Compiles all files in current directory
      Added a "clean" target - uses a fileset to remove all *.class files
   </description>

   <target name="compile">
      <javac srcdir='./'/>
   </target>

  <target name="clean">
    <delete>
        <fileset dir="./">
           <include name='*.class'/>
        </fileset>
      </delete>
   </target>
</project>
```

Example – Dependencies, java, classpath


```
<project default="compile">

   <target name="compile" depends='foo,bar'/>

   <!-- Javac compiles dependencies for us -->
   <target name="foo" depends='bar'>
      <javac srcdir='.' includes='foo.java' />
   </target>

   <target name="bar">
      <javac srcdir='.' includes='bar.java'/>
   </target>

   <target name='run' depends='compile'>
      <java classname='foo'>
         <classpath>
            <pathelement path="${classpath}" />
            <pathelement location="." />
         </classpath>
      </java>
   </target>
</project>
```

Example – Providing Command Line Arguments


```
<target name="foo" depends='compile'>
   <java classname='foo'>
      <classpath>
         <pathelement path="${classpath}" />
         <pathelement location="." />
      </classpath>
      <arg line='2 args'/>
      <arg value='an arg'/>
      <arg path='../../index.html'/>
   </java>
</target>
```

Arguments are:

```
0: 2
1: args
2: an arg
3: /home/kschmidt/public_html/CS265/Labs/index.html
```

Example – `jar`, Creating a Jarfile

- A custom manifest file can be identified
- Include needed files

```
<target name="jar" depends='compile'>
   <jar destfile='./foo.jar' manifest='man.mf'>
      <fileset dir='./'>
         <include name='*.class'/>
         <include name='annoy.mp3'/>
      </fileset>
   </jar>
</target>
```

Example – Running a Jarfile

- Jarfile manifest needs a valid `Main-Class` entry
- Target may still contain arguments, classpath, etc.

```
<target name='run' depends='jar'>
   <java jar='foo.jar' fork='true'>
      <classpath>
         <pathelement path='${classpath}' />
         <pathelement location='.' />
      </classpath>
      <arg line='2 args'/>
      <arg value='an arg'/>
      <arg path='../../index.html'/>
   </java>
</target>
```

Example – Using Subdirectories, Properties


```
<project default="all">
   <property name="obj-dir" location="obj" />
   <property name="src-dir" location="src" />

   <target name="init">
      <mkdir dir="${obj-dir}" />
      <mkdir dir="${lib-dir}" />
   </target>

   <target name="compile" depends="init">
      <javac srcdir="${src-dir}" destdir="${obj-dir}" />
   </target>

   <target name="clean-compile">
      <delete>
         <fileset dir="${obj-dir}" includes="**/*.class" />
      </delete>
   </target>
</project>
```

Example – Compiling and Running Classes in a Package

Both classes are in the `cs265_example` package

```
<project default="all">
   <property name='obj-dir' location='.'/>
   <property name='src-dir' location='.'/>

   <target name="foo" depends='bar'>
      <javac srcdir='${src-dir}' destdir='${obj-dir}'
         includes='foo.java' />
   </target>
   <target name="bar">
      <javac srcdir='${src-dir}' destdir='${obj-dir}'
         includes='bar.java' />
   </target>
   <target name='run' depends='compile'>
      <java classname='cs265_example.foo'>
         <classpath>
            <pathelement path="${classpath}" />
            <pathelement location="." />
         </classpath>
      </java>
   </target>
</project>
```

Invocation

How to Run Ant

`ant [-f file] [target]*`

- *file* – build file
  - `build.xml` by default
- `target` – list of targets to be built
  - Not required, if the project lists a default target

```
$ ant   # run default target in build.xml
$ ant test  # run text target in build.xml
$ ant -f b3.xml # run default target in b3.xml
```

Defining Properties on Command Line


```
<target name="foo" depends='compile'>
   <java classname='foo'>
      ...
      <arg value='${user}'/>
      <arg path='${outfile}'/>
   </java>
</target>
```

```
$ ant -D"user=$USER" -D"outfile=../foo.log" -f prop.xml foo
```

```
[java] You're in foo's main! The command-line args are:
[java]
[java] 0: kschmidt
[java] 1: /home/kschmidt/public_html/CS265/Labs/Java/foo.log
```