CmpE 226 Apache Ant

Apache Ant has been a mainstay for building Java projects for many years. This lecture provides an introduction to Ant

- Agenda
  - Building projects with Ant
  - Demo
- Key points
  - Configure a project to use Apache Ant to build, deploy, and test your code

- Another Neat Tool
  - Java's make tool
    - Recall make and makefiles (imake, gmake)
      - Manages the build process and source dependencies
  - Written in Java
    - Supports JDK 1.1 and greater
  - Uses XML
- What ANT isn't
  - An editor, compiler, jar tool, version control, testing framework
- ANT is
  - Java application
  - A tool to manage and run compiling, jaring, version control, documentation, and testing

Ant basics

- Ant receives it instructions through an XML build file.
  - Ant by default will look for a file named build.xml
  - A build file contains one or more projects
    - Projects contain one or more targets
      - Targets contain one or more tasks
- Extend Ant through Java classes not OS dependent shell scripts
  - Platform independent

Advantages of Ant over .bat/.sh files

- Platform independent, where there's Java
  - Consistent behavior across operating systems
- Open Source (free)
- Uses XML
- Rich feature set
- Extensible
  - How? Write some Java classes
- Well known, widely accepted, easy to use

Common features

- Compile Java classes
- Create Jars, EARs, WARs,
- Interact with VS
  - Subversion
  - CVS
  - Perforce
- Run applications
- Run unit tests (JUnit)
  - Create reports
- Logging

- Conditionals
- Call other Ant builds
- Email
  - (e.g., results of a build)
- Execute system commands
- Create JavaDocs

More . . .

Installing Ant

- Apache Open Source Project
  - Free
  - Many other projects available, www.apache.org
- Download jar
  - http://ant.apache.org/
- Extract jar
- Configure environment
  - Define `ANT_HOME` to reference where Ant was installed
  - Add to PATH:
    - **`$ANT_HOME/bin`** or for Windows: **`%ANT_HOME%\bin`**
  - Define JAVA_HOME to your JDK
- Optional packages added to CLASSPATH or Ant's lib directory
- Copy JUnit's jar file into Ant's lib directory
- Do not install Ant into the JDK directory (JAVA_HOME)

The Ant command

- Command line driven

```
ant [options] [target]
```

- Help

```
ant –help
```

- Common options:

```
-file or –f
-verbose
-projecthelp or -p
```

A build file's root <project>

- Encapsulates the needs of a typical software project from building to testing to deploying
- Attributes
  - Name
    - The name of the project
  - Default
    - The default target to run
  - Basedir
    - The base directory of the project – **Ant's working directory**

```
1. <project name="hw4" default="build" basedir ="."> 
2.
3. </project>
```

<target>

- Key building block in Ant
- Attributes
  - Name
    - Target name
  - Depends
    - Requires the named target to have ran
  - If or unless
    - Supports limited conditional logic

```
1. <target name="build" depends="a_target">
2.
3. </target>
```

Build file example

```
1. <project name="hw4" default="build" basedir =".">
2.
3. <target name="init">
4. </target>
5.
6. <target name="build" depends="init"
7.         description="build project java files">
8.
9. </target>
10.
11.</project>
```

Properties

- Variable within your build.xml
- Notation: ${variable_name}
  - Case sensitive
- Built-in properties
  - `basedir` – abs. dir. The ant script is running in
  - `ant.file` – absolute path of the current build file
  - `ant.version` – version of Ant
  - `ant.project.name` – project name
  - `ant.java.version` – JVM version Ant found

Properties Cont.

- JDK System properties are access through their name
  - E.g.,

    Property `file.separator` is ${file.separator}

    Property `user.home` is ${user.home}

    (Customization, CVS username/password, etc.)

- Creating a timestamp
  - `</tstamp>`
  - Creates variables DSTAMP, TSTAMP, TODAY
  - Generally called from "init"

Ant Build.xml

- Most projects have a common set of targets
- Quasi-Standard target names
  - `init, prepare, fetch, compile, test, jar, war, docs, deploy, clean`
- A good practice is to group properties in your "init" target
- To view a project's targets

```
ant -projecthelp or ant -p
```

Ant & Compiling

- Target "javac"
- Recursively through source directory
- Supported arguments to javac can be set as attributes of the javac target
  - -g (debugging) => debug="true"
  - -deprecation => deprecation="true"
- Setting boolean attributes
  - Positive: on, yes, true
  - Negative: off, no, false

Ant & Jaring

- Target "jar"
- Use "copy" in conjunction with "jar" to stage files

```
1. <jar destfile="hw4.jar"
2.      basedir="${output}"
3.      manifest="name.mf"
4.      depends="another_target" />
5.
6. <copy file="name"
7.       todir="where"
8.       overwrite="true" />
```

Ant & Logging

- Target “record”
- Recording what did and did not build
- Levels: quite, verbose, debug, info (default)
- MailLogger: Send emails on success or failure
- Start

```
<record name="everything.log" loglevel="verbose"/>
```

- Stop

```
<record name="leverthing.log" action="stop"/>
```

Misc. Ant

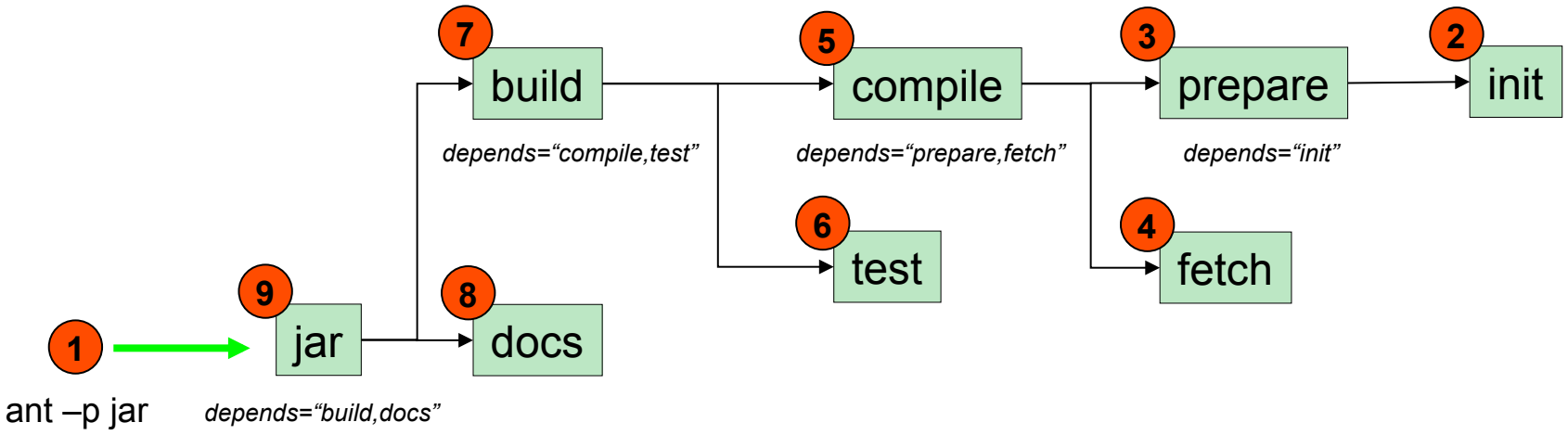
- Calling other build.xml files
  - Build subprojects

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

- Other commands (see documentation)
  - `ftp, telnet, javah, javacc, zip, unjar, unzip, touch, tar, sleep, rename, move, mail, gzip, gunip, chmod, echo, …`

Linking Targets

- Use a target's attribute `depends` to specify build's flow
  - Called before a target's commands (body)
  - Hierarchy of relationships (One or many dependencies)



The “init” target

- Set project properties
  - By name-value
  - Loading a properties file
    - CVS password, OS/host specific configurations
- Create classpath

```
1. <target name="init">
2.     <property name="jarname" value="cmpe271_hw3.jar" />
3.     <property name="build.dir" value="%{basedir}" />
4.     <property name="build.dir.lib" value= "${build.dir}/lib" />
5.     <loadproperties srcFile="hw3_build.props" />
6. </target>
```

The “prepare” target

- Build project structure
  - Make directories
  - Specify logging, output files

```
1. <target name="prepare" depends="init">
2.     <mkdir dir="${build.dir}" />
3.     <mkdir dir="${build.dir.lib}" />
4.     <record
5.         name="${ant.project.name}.log
6.         action="start" />
7. </target>
```

The “compile” target

- Compile Java classes
- A lot easier than batch files – right?

```
1. <target name="compile" depends="prepare">
2.     <javac srcdir="${src.dir}"
3.            destdir="${build.dir.classes}"
4.            classpath="${classpath}" />
5. </target>
```

The “test” target

- Run the project’s unit tests
- Generate reports
- Fork test targets
  - Isolate test side effects from the build process
    - E.g., System.exit()
  - Faster

```
1. <target name="test" depends="compile">
2.     <junit haltonfailure="no">
3.         <formatter type="xml" />
4.         <test name="com.gash.filter.FilterTest" />
5.     </junit>
6. </target>
```

Conditionals

- Set a property if a condition is satisfied
- Test can include:
  - <or>, <and>, <equals>, <os>, ...
  - See conditions.html for list

```
1. <target name="set-env"
2.     <condition property="install.os" value="mac">
3.         <os name="Mac OS" />
4.     </condition>
5. </target>
```

Logging

- Recorder
  - Writes the output of the build process to a file
  - Attributes
    - Name
    - Action (start, stop)
    - Appender (new file or append to)

```
1. <target name="record-on" depends="init">
2.     <record loglevel="info"
3.             name="${somewhere}/filename.log" />
4. </target>
```

Putting it all together, a build.xml

Using Ant to manage complex projects

- A complex builds can be organized into multiple files controlled by a master build

```
<ant antfile="${a-dir}/another-build.xml"
target="site">
```

- Targets can call other targets within the same ant file

```
<target name="more-targets">
  <antcall target="step-a"/>
  <antcall target="step-b"/>
</target>
```

- Also, ant files can import ant files to set common properties

```
<import file="${b-dir}/yet-another-build.xml"/>
```

References

- Ant
  - JavaDocs – Extensive examples
  - http://ant.apache.org/
  - Extreme Programming w/ Ant, Wiemeyer, et. Al.
  - Ant Developer's Handbook, Williamson, et. Al.
  - ANT The Java Build Tool in Practice, Matzke
  - Ant in Anger
    - Distributed with binaries

Q & A

- Questions
- Examples