

## CS520 Web Programming

Introduction to Ant

Chengyu Sun  
California State University, Los Angeles

## Build

- ◆ Compilation
- ◆ Maintenance
- ◆ Distribution
- ◆ Deployment

## Background

- ◆ A build tool for Java
- ◆ Developed by James Duncan Davidson, who also wrote Tomcat
- ◆ Ant stands for
  - Another Neat Tool (ANT), or
  - Ant, the small creature who can *build* big things

## Why Ant

- ◆ vs. `javac *.java`
  - Re-compile the modified files
  - Re-compile the files affected by the modified files (Dependency)
- ◆ vs. scripts
  - Easier to use
  - Platform independent

## Why Not Make

- ◆ Compiler is invoked once for every source file
  - Fine for C/C++ compilers
  - Bad for Java compiler – JVM startup overhead
- ◆ Platform dependent
  - `make`, `gmake`, `nmake` ...
  - relies on shell command for most tasks
- ◆ Syntax problem
  - Tab

## Ant installation

- ◆ Download from <http://ant.apache.org>
- ◆ Unzip to a local directory
- ◆ Environment variables
  - `JAVA_HOME`: JDK directory
  - `ANT_HOME`: Ant directory
  - Add `$ANT_HOME/bin` (`%ANT_HOME%/bin`) to `PATH`

## Ant Basics

- ◆ Build file
- ◆ Property
- ◆ Target
- ◆ Task
- ◆ Type

## Build File

- ◆ build.xml
- ◆ An XML file
- ◆ `<project>`
  - *name* – name of the project
  - *basedir* – the directory where the project is located

## Sample Build File

```
<project name="demo" default="compile" basedir=".">
  <property name="tomcat.home" value="d:\tomcat"/>

  <path id="project.classpath">
    <pathelement path="{javac.classpath}"/>
    <fileset dir="{tomcat.home}">
      <include name="common/lib/**/*.jar"/>
    </fileset>
  </path>

  <target name="compile" depends="init">
    <javac srcdir="src" destdir="classes">
      <classpath refid="project.classpath"/>
    </javac>
  </target>

  <target name="init">
    <mkdir dir="classes"/>
  </target>
</project>
```

## Properties

- ◆ `<property>`
    - name
    - value
- ```
<property name="tomcat.home" value="d:\tomcat"/>

<fileset dir="{tomcat.home}">
  <include name="common/lib/**/*.jar"/>
  <include name="shared/lib/**/*.jar"/>
  <include name="server/lib/**/*.jar"/>
</fileset>
```

## Targets and Tasks

- ◆ Targets
  - Things we want to do
    - ♦ compile, install, clean ...
- ◆ Tasks
  - "Commands" that can be used to accomplish a *target*
  - `<javac>`, `<copy>`, `<move>`, `<depend>` ....
  - More portable than actual shell commands

## Multiple Targets

- ◆ More targets
  - separate source and class directories
  - remove compiled classes
- ◆ Default target
- ◆ Target dependency

## Example 1

- ◆ `<target name="compile">`
- ◆ `<target name="init">`
- ◆ `<target name="clean">`

## Types

- ◆ Mostly related to describing files and directories
  - `<fileset>`
  - `<path>`

## `<fileset>`

- ◆ A set of files
- ◆ `<fileset>`
  - `dir` – root directory of this file set
  - `<include>`
  - `<exclude>`

## `<fileset>` Examples

```
<fileset dir="d:/tomcat/common/lib"/>
<fileset dir="d:/tomcat/common/lib">
  <include name="*.jar"/>
</fileset>

<fileset dir="d:/tomcat/common/lib">
  <include name="**/*.jar"/>
</fileset>

<fileset dir="d:/tomcat/common/lib">
  <include name="**/*.jar"/>
  <exclude name="**/naming*.jar"/>
</fileset>
```

## `<path>`

- ◆ A list of files and directories
- ◆ `<path>`
  - files: `<fileset>`, `<filelist>`
  - directories: `<dirset>`
  - other paths: `<pathelement>`, `<path>`
- ◆ `<classpath>`

## Example 2

- ◆ Specify a classpath that include
  - Java class libraries
  - Tomcat class libraries
    - ◆ `common/lib/*.jar`
    - ◆ `shared/lib/*.jar`
    - ◆ `server/lib/*.jar`
  - Additional class libraries
    - ◆ `lib/*.jar`

## The Scripting Language Analogy

◆ build.xml	→	◆ Program
◆ Property	→	◆ Variable
◆ Target	→	◆ Function
◆ Task	→	◆ Command
◆ Type		◆ Data type

## Using Ant for Your Projects

- ◆ You don't have to be an Ant expert
- ◆ Start with a sample build file
  - Understand it
  - Adapt it
- ◆ Ant Manual
  - <http://ant.apache.org/manual/index.html>