Ant

Build Tool for Java

Overview

- ANT - Another Neat Tool
- A product of the Apache Software Foundation
- https://ant.apache.org/
- A Java application that can call the Java compiler
    - Requires JDK, not just JRE

Why not use make?

- Makefiles are based on shell commands
  - Shell commands are mostly platform dependent
- It is easy to introduce hidden errors into a Makefile

```
hello: main.o factorial.o hello.o
    $(CC) main.o factorial.o hello.o -o hello
```

  - Ex. A space before a tab before a command $(CC) invalidates the Makefile

Why Use Ant?

- Platform independence
    - Lots of predefined tasks implemented in Java
    - Hides platform-specific properties, ex path separators
- XML build scripts can be validated
- Ant can be extended (more platform independent tasks)

Preparing for Ant

- Prerequisites
  - JDK installed
  - JAVA_HOME environment variable set to JDK install folder (the folder that contains bin/, include/, lib/)

Downloading Ant

- Download from https://ant.apache.org/bindownload.cgi
  - Two code bases: 1.9.x and 1.10.x
    - 1.9.x requires JDK 5
    - 1.10.x requires JDK 8

Installing Ant

- Unzip (untar) the downloaded archive to a folder of your choice (/tools in this tutorial).
- Create an environment variable called ANT_HOME
  - export ANT_HOME=/tools/apache-ant-1.10.1
- Add ant bin/ folder to your PATH environment variable
  - export PATH=$PATH:$ANT_HOME/bin
- Confirm installation:
  - $ ant -version

Example

build.xml

```
<?xml version=“1.0”?>
<project name=“Hello” default=“info”>
    <target name=“info”>
        <echo>Hello, World</echo>
    </target>
</project>
```

```
$ ant
info:
    [echo] Hello, World

BUILD SUCCESSFUL
Total time: 0 seconds
$
```

Command line options

- Ant looks for build.xml in the current directory
  - $ ant [-f path to build script] [target]
  - Other
    - $ ant -help ant -h
    - $ ant -projecthelp ant -p
    - $ ant -version

build.xml (begin)

```
<?xml version="1.0"?>
<!--
     build.xml
     Ant script for CS364, Program #1

     David Apostal
-->
<project name="Hi-Low Guesser" default="compile" basedir=".">
    <!-- define properties to be used later -->
    <property name="src.dir" location="src" />
    <property name="build.dir" location="build" />
```

build.xml (middle)

```
<!-- delete existing directories (and contents) -->
<target name="clean" description="delete build artifacts">
    <delete dir="${build.dir}" />
</target>

<!-- create output directory -->
<target name="init">
    <mkdir dir="${build.dir}" />
</target>

<target name="compile"
      description="compile project source code"
      depends="clean, init">
    <javac srcdir="${src.dir}" destdir="${build.dir}" />
</target>
```

build.xml (end)

```
    <target name="run-server" description="run the server" depends="compile">
        <java
            fork="true"
            failonerror="yes"
            classpath="${build.dir}"
            classname="hw1.GuessServer">
            <arg line="4444 10"/>
        </java>
    </target>

    <target name="run-client" description="run the client" depends="compile">
        <java
            fork="true"
            failonerror="yes"
            classpath="${build.dir}"
            classname="hw1.GuessClient">
            <arg line="localhost 4444"/>
        </java>
    </target>
</project>
```