Ant

- Originally ANT = Another Neat Tool
  - Created by James Duncan Davidson
  - Now an Apache open-source project
- Ants are amazing insects
  - Can carry 50 times their own weight
  - Find the shortest distance around obstacles
  - Work in shifts around the clock

Ant

- Ant is a Java-based build tool designed to be
  - Cross-platform
  - Easy to use
  - Extensible
  - Scalable
- Other benefits
  - Fast
  - XML format
  - Tight integration with JUnit
  - Built-in support for J2EE development (EJB compilation and packaging)
  - Integrates with FTP, Telnet, application servers, SQL commands
  - de Facto standard for most open source Java projects, such as Apache Tomcat
  - Built-in support in Eclipse and other IDEs

A First Project with Ant

- Each **project** requires a build.xml file
- XML files contain a tree-like structure of elements (nodes)
- The top-level element is the project
  - Each build.xml file can contain only one project
- Each project can contain targets
  - Each **target** can contain tasks
    - Each **task** can contain **attributes** and **elements**

Ant vs. Make

- Ant and Make have many similarities
- Here are some of the differences
  - Ant is platform independent
    - Java based
    - File and path resources are generated at run-time
    - Make typically uses tools expected in the underlying OS
  - Ant does not require hidden tab characters
  - Ant leaves file dependencies to the tasks to work out, it only specifies target dependencies
  - Ant allows easy addition of source files without having to change the build script

A First Project with Ant

- Suppose you have some Java source code in the current directory or subdirectories

```
public class HelloWorld {
  public static void main(String [] args) {
    System.out.println("Hello Ant users");
  }
}
```

- Place the following in a file named build.xml in the same directory (see examples/FirstProject)

```
<?xml version="1.0"?>
<project name="firstbuild" default="compile" >
    <target name="compile">
       <javac srcdir="." />
       <echo>compilation complete!</echo>
    </target>
</project>
```

A First Project with Ant

- Run the Ant build script with either

  ant

  ant compile

- A successful build (see Examples/FirstProject)

```
Buildfile: build.xml


compile:
    [javac] Compiling 1 source file
     [echo] compilation complete!


BUILD SUCCESSFUL
Total time: 6 seconds
```

A First Project with Ant

- A failed build

```
public class HelloWorld {
  public static void main(String [] args) {
    System.out.println("Hello Ant users")
  }
}
```

```
Buildfile: C:\Ant\examples\Failure1\build.xml


compile:
    [javac] Compiling 1 source file
    [javac] C:\Ant\examples\Failure1\HelloWorld.java:4: ';' expected
    [javac]   }
    [javac]   ^
    [javac] 1 error
    [javac] BUILD FAILED: file:C:/Ant/examples/Failure1/build.xml:4: Compile failed; see the
   compiler error output for details.
Total time: 1 second
```

A First Project with Ant

- Another failure

```
<?xml version="1.0"?>
<project name="firstbuild" default="compile" >
   <target name="compile">
      <javac sourcedir="." />
      <echo>compilation complete!</echo>
   </target>
</project>
```

```
Buildfile: C:\Ant\examples\Failure2\build.xml


compile:
      [javac] BUILD FAILED: file:C:/Ant/examples/Failure2/build.xml:4: The <javac> task
     doesn't support the "sourcedir" attribute.
Total time: 370 milliseconds
```

A First Project with Ant

- Another failure

```
<?xml version="1.0"?>
<project name="firstbuild" default="compile" >
    <target name="compile">
       <jaavac srcdir="." />
       <echo>compilation complete!</echo>
    </target>
</project>
```

```
Buildfile: C:\Ant\examples\Failure3\build.xml
compile:
      [jaavac] BUILD FAILED: file:C:/Ant/examples/Failure3/build.xml:4: Could not create task
      or type of type: jaavac.
Ant could not find the task or a class this task relies upon.
This is common and has a number of causes; the usual
solutions are to read the manual pages then download and
install needed JAR files, or fix the build file:
 - You have misspelt 'jaavac'.
   Fix: check your spelling….
```

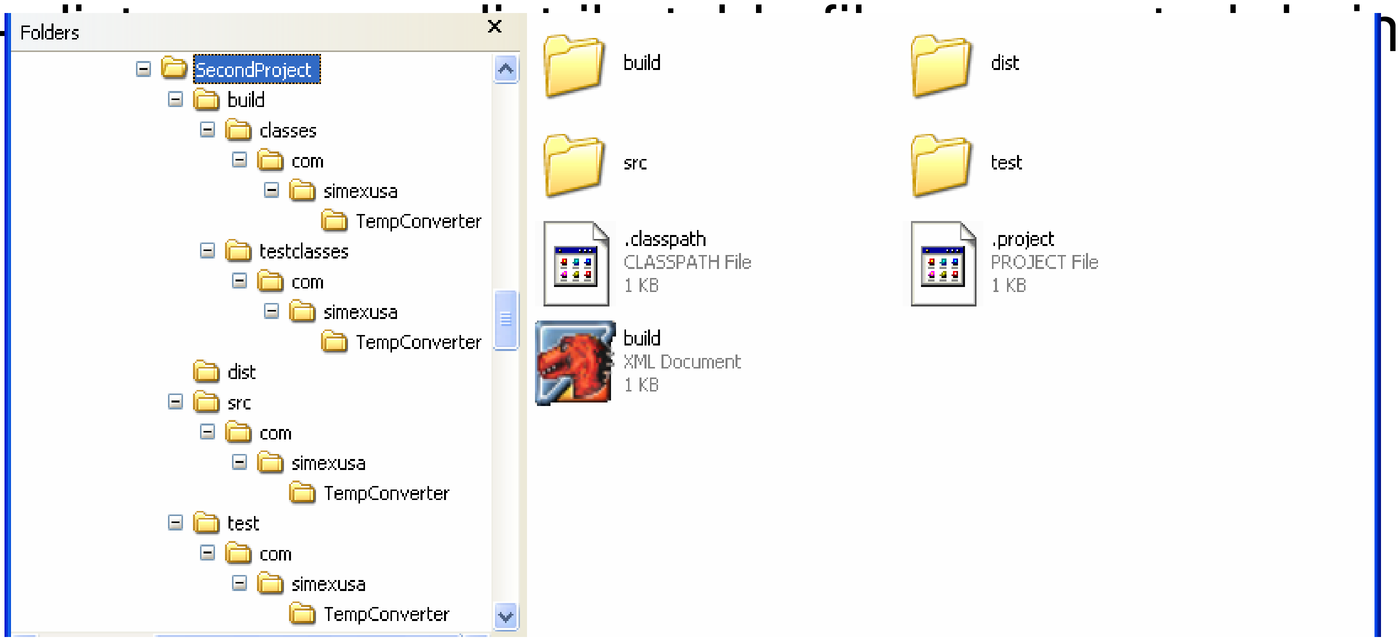
Improving the project structure and organization with Ant

- As our projects grow beyond toy examples, we will probably want to do the following
  - Place source code in packages
  - Separate source code from object code
  - Separate test code from source code
  - Create a distribution such as a JAR file
  - Provide a mechanism to delete all intermediate and object code
  - Execute our program from a guaranteed up-to-date compile

Sample project directory structure

- Project Name

– src	source files
– test	unit-test source files
– build	intermediate files generated during a build
• classes	.class files from .java files
• testclasses	.class files from unit test .java files
• html	.html files from .java or .xml files
–	ng a



Folders

```
SecondProject
  build
    classes
      com
        simexusa
          TempConverter
    testclasses
      com
        simexusa
          TempConverter
  dist
  src
    com
      simexusa
        TempConverter
  test
    com
      simexusa
        TempConverter
```

build
dist
src
test
.classpath CLASSPATH File 1 KB
.project PROJECT File 1 KB
build XML Document 1 KB

A Second Project with Ant

```
<?xml version="1.0" ?>
<project name="SecondProject" default="execute">
  <target name="init">
    <mkdir dir="build/classes" />
    <mkdir dir="build/testclasses" />
    <mkdir dir="dist" />
  </target>

  <target name="compile" depends="init">
    <javac srcdir="src"
          destdir="build/classes"
     />
  </target>

  <target name="test-compile" depends="compile">
    <javac srcdir="test"
          destdir="build\testclasses"
          classpath="C:\Program Files\JUnit\junit3.8.1\junit.jar;build\classes"
     />
  </target>
```

see examples/SecondProject

Automatically builds package structure

build.xml continued on next slide

A Second Project with Ant

build.xml continued from previous slide

```
<target name="archive“ depends="compile,test-compile" >
  <jar destfile="dist/SecondProject.jar"
      basedir="build/classes" />
 </target>

 <target name="clean" depends="init">
  <delete dir="build" />
  <delete dir="dist" />
 </target>

 <target name="execute" depends="compile" >
  <java
   classname="com.simexusa.TempConverter.TempConverter"
   classpath="build/classes"
   fork=“true”/>
 </target>
</project>
```

fileset

- A fileset is a set of files rooted from a single directory
- By default, a fileset specified with only a root directory will include all of the files in that directory and all subdirectories recursively

```
<fileset dir="src" />
```

```
<fileset dir="docs">
</fileset>
```

Patternsets in filesets

- A fileset can contain patternsets
- A patternset is a collection of file matching patterns and must be nested in a fileset
  - include (also includesfile)
  - exclude (also excludesfile)

```
<fileset dir=“src”>
  <include name=“*.java” />
</fileset>
```

Contains all .java files in src

```
<fileset dir=“src”>
  <include name=“**/*.java” />
</fileset>
```

Contains only .java files in src and subdirectories below src

```
<fileset dir=“src” includes=“**/*.java”>
```

Same as previous