ANT

Another Neat Tool

What is ANT?

What is Apache Ant?

Ant is a Java-based build tool. In theory, it is kind of like Make, without Make's wrinkles and with the full portability of pure Java code.

Why do you call it Ant?
According to Ant's original author, James Duncan Davidson, the name is an acronym for "Another Neat Tool".
Later explanations go along the lines of "ants do an extremely good job at building things", or "ants are very small and can carry a weight dozens of times their own" - describing what Ant is intended to be.

Installing ANT (Windows)

Downloading and Installing Ant
1) Make sure you have JDK1.3 or better installed on your machine.

2) Download ant from http://jakarta.apache.org/ant/ and unzip it in a directory.

3) Set the class path to the bin directory of the ant.
Let's assume that Ant is installed in c:\ant\.
The following code has to be put into autoexec.bat file:

```
set ANT_HOME=c:\ant
set JAVA_HOME=c:\jdk1.3
set PATH=%PATH%;%ANT_HOME%\bin
```

Testing ANT (Windows)

Go to command prompt (cmd.exe) and issue the command:

```
C:\anttest>Ant
```

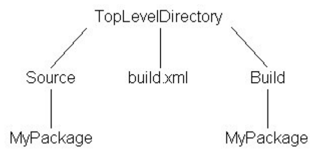
If everything is installed correctly Ant will give the message:

```
Buildfile: build.xml does not exist! Build failed
C:\anttest>
```

Directory structure

In order for Ant to compile only source files which have been changed, you must set up your directory structure in which your source files are in parallel to your build files.

Make a directory in which to build your project.
This directory should countain a 'source' directory for .java files and a 'build' directory for .class files.
It must also include your Ant build.xml file.
In your source directory, you layout your package hierarchy.
Example:



Our first build.xml

```
<?xml version="1.0"?>
<!-- Build file for our first application -->
<project name="Ant test project" default="build" basedir=".">
 <target name="build" >
   <javac srcdir="src"
        destdir="build/src"
        debug="true"
        includes="**/*.java"
    />
 </target>
</project>
```

Properties and multiple targets (with dependencies)

```
< project name="lab3" default="init" basedir=".">

    < property name="src" value="." />
    < property name="build" value="build" />

    < target name="init">
        <mkdir dir="${build}" />
    </target>

    < target name="compile" depends="init" >
        < !-- Compile the java code -->
        < javac srcdir="${src}" destdir="${build}" />
    </target>

</project>
```

Project

The first line is the Project tag, and gives your project its name.
`< project name="lab3" default="init" basedir=".">`

The name is the name of your project

The default attribute points to the target which by default will start the building process.

The basedir sets the directory from which all path calcutlations are done.

Both the name and basedir attributes are optional fields *but THE DEFAULT ATTRIBUTE IS REQUIRED*.

Target

A target is a set of tasks waiting to be executed.

```
< target name="compile" depends="init" >
    < javac srcdir="${src}" destdir="${build}" />
</target>
```

A target can be dependent upon another target, as is the "compile" target in the above example.

Task

Tasks are the snippets of code that get the job done.

```
< javac srcdir="${src}" destdir="${build}" />
```

In the above example, the task of the "compile" target runs javac in the source directory, and places the .class files in the directory "build" which was created in the "init" target (if the directory "build" already exists, then it will NOT be created anew).

Notice that "src" and "build" were defined above in tags known as Property tags

Property tags

- Property tags define name-value pairs (i.e. variables) that are used in task attributes

  < property name="src" value="." />

  Note that when we refer to a property later on we call it between "${" and "}"

Running an example

To run ant, copy the xml file listed above into its own file and save it as "build.xml".

Place it in a directory where you have a .java file ready for compiling and type "ant". You should get some status as Ant runs, and when it finishes, you should have a "build" directory with a bunch of class files!

Main advantages

- Automatization of complex processes

-Platform independence

- Process Optimization
    -Uses a single JVM for the whole process
    -Intercepts exceptions
    -Ricompiles only thhe needed files

A more complex example - 1

```
<?xml version="1.0"?>
<project name="Sample project" default="compile" basedir=".">
 <property name ="src" value= ="src"/>
 <property name =" build.dir" value= ="src"/>
 <property name =" build.classes" value="${build.dir}/classes"/>
 <property name =" build.lib" value="${build.dir}/lib"/>

 <target name="prepare" >
   <mkdir dir="${build.dir}"/>
   <mkdir dir="${build.classes}"/>
   <mkdir dir="${build.lib}"/>
 </target>

 <target name= "clean" description= "Removes all generated files">
   <delete dir="${build.dir}"/>
  </target>
  <!-- continua...-->
```

A more complex example - 2

```
<target name= "compile"  depends= "prepare">
  <javac srcdir="${src.dir}"  destdir="${build.classes}"/>
</target>

<target name="jar" depends= "compile">
  <jar jarfile="${build.dir}/distribution.jar"
       basedir="${build.classes}"
       excludes="**/*Test.class" />
 </target>

<target name= "all"  depends= "clean,jar">
</target>

</project>
```

Java related commands

```
<javac srcdir="..." destdir="..." />

<java classname="..." >
        <classpath>
                <pathelement path="..." />
        <classpath/>
<java>

 <javadoc>
        <package name="..." />
        <sourcepath location="..." />
        <classpath location ="..." />
 </javadoc>


 <rmic base="..." />
```

File system related Commands

```
<chmod perm="444" file="*" />
<copy file="pippo" todir="a" tofile="b" />
<move file="pippo" todir="a" tofile="b" />
<mkdir dir="mydir" />
<delete file="pippo" /> <delete dir="*" />
<touch file="pippo" />

<gzip> <gunzip>
<jar> <unjar>
<zip> <unzip>
<war> <unwar>
<ear>
<tar>
```

Other commands

```
<echo message="..."  />
<mail>
<get src="URL" dest="..." />

<exec executable="..." dir="..." vmlauncher="false" os="Windows 2000" >
  <arg line="..." />
</exec>
```

Various forms for conditional execution

<cvs> revision control system
<sql> executes sql queries via jdbc
<style> performs xslt transformations