

Creating buildfiles using ANT script:

Most of the times, we always create the build (jar file or war file) using the IDE on the developer's machine. But while moving the CODEBASE to staging server, we again need to build the application in the form of jar or war file from the source code and resource files on the server. Using ANT Script we can compile the Java source code to class files and then create the jar/war file of the application on the server.

In this article, we will create jar file using ANT script from the application's source code and we will set the classpath of the external libraries used in this application.

ANT uses the build.properties file as the default properties file. However, we can mention the properties file name on the script itself to be referred by ANT tool.

ANT attributes and tasks:

In ANT script, the project is divided in three parts namely *name*, *default* and *basedir*.

name	name of the Project
default	default target to use if no target is mentioned
basedir	the base directory from where all the paths mentioned in the scripts will be referred

ANT targets:

The target is defined to accomplish specific tasks. While writing ANT scripts, we divide the tasks in different targets and the targets can depend on other target. In other words we can define targetA which depends on targetB and targetA will start executing only after the targetB finishes. This is done by using *depends* attribute.

```
<target name="targetA" />
<target name="targetB" depends=" targetA" />
<target name="targetC" depends=" targetB" />
<target name="targetD" depends=" targetC, targetB, targetA" />
```

If we want to execute the targetD first, Below is the flow how the dependent targets will be getting executed using left to right approach.

- When we execute targetD, the control first goes to targetC. As the targetC depends on targetB so it will look for executing targetB and then to targetA.
- Finally, targets will be getting executed in order targetA, targetB, targetC.

Below is an example ANT script which is used to create the jar file of a project say, ANTTTest.

Step1: Create the bin directory.
Step2: Compile the source code.
Step3: Create the jar.

The build.properties file

```
# Project properties:
project.version=1.0.0
project.name=ANTTest

# path of the external jar files
lib.path=./lib
build.xml:
```

```

<project name="ANTTest" default="buildAll" basedir=".">
  <property file="build.properties" />
  <property name="anttest.src" location="${basedir}/src/com/anttest" />
  <property name="anttest.classes" location="bin/anttest" />

  <path id="lib.classpath.anttest">
    <fileset file="${lib.path}/JarFile1.jar" />
    <fileset file="${lib.path}/JarFile2.jar" />
    . . . . .
  </path>

  <target name="clean">
    <delete dir="${basedir}/bin" />
  </target>

  <target name="init">
    <mkdir dir="${anttest.classes}" />
  </target>

  <target name="compile.anttest" depends="init">
    <echo>Compiling project</echo>
    <javac srcdir="${anttest.src}" destdir="${anttest.classes}" debug="true"
      target="1.4" source="1.4">
      <classpath>
        <path refid="lib.classpath.anttest" />
      </classpath>
    </javac>
  </target>

  <target name="pack_jar.anttest" depends="compile.anttest" >
    <jar destfile="${basedir}/ANTTest.jar" basedir="${anttest.classes}"
    manifest="MANIFEST.MF"/>
    <fileset dir="${anttest.src}" includes="**" />
  </target>

  <target name="buildAll" depends="pack_jar.anttest">
  </target>
</project>

```