AntUtil

Preface

After doing things over and over I usually intend to automate that. In earlier time I wrote (windows) batch files. But since some months sometimes I do that with Ant.

Sometimes also things Ant was never designed for ...

Installation

On my machine I have a directory c:\bin which is listed in the %PATH% variable, so I can invoke shell command whereever I am.

The AntUtil contains several files: the buildfile, the batchfile and an optional propertyfile. Additionally some targets require template files.

antutil.bat

```
rem Simply invoke Ant and store the currend directory as Ant property.
@echo off
ant -buildfile c:\bin\antutil.xml %* -DcurDir="%CD%"
```

antutil.properties

```
 # used to save common settings
 # on my machine empty
```

antutil.xml

```
 <?xml version="1.0" encoding="ISO-8859-1"?>

 <!-- Here the work is done :-) -->

 <project name="antutil" default="usage">
     <!-- I use AntContrib on several places -->
     <taskdef resource="net/sf/antcontrib/antcontrib.properties"/>

     <!-- the directory where antutil was invoked -->
     <property name="curDir" value="."/>

     <!-- load the common properties -->
     <property file="antutil.properties"/>


     <target name="zip" description="Create a zip-file with files listed in files2zip.txt">
         <property name="zipname" value="zipname"/>
         <property name="files2zip" value="files2zip.txt"/>
         <!-- Ant waits for finishing notepad, which I want here. -->
         <echo>Start Notepad and wait for its end.</echo>
         <exec executable="notepad.exe"><arg value="${files2zip}"/></exec>
         <echo>Create ZIP-File ${zipname}.zip</echo>
         <zip destfile="${curDir}/${zipname}.zip" basedir="${curDir}" excludes="${zipname}.zip"
includesfile="${files2zip}" />
     </target>


     <!-- this is the 'user front end' for target mget -->
     <target name="download" description="Download files listed in a textfile">
         <property name="filelist" value="files2load.txt"/>
         <!-- Ant waits for finishing notepad, which I want here. -->
         <echo>Start Notepad and wait for its end.</echo>
         <echo file="${curDir}/${filelist}" message="uri=dest"/>
         <exec executable="notepad.exe"><arg value="${curDir}/${filelist}"/></exec>
```

```
        <echo>Load files</echo>
        <antcall target="mget"/>
        <delete file="${curDir}/${filelist}"/>
    </target>

    <!-- mget needs a file with uri=dest pairs to download -->
    <target name="mget" description="download multiple files" if="filelist">
        <!-- Network configuration -->
        <property name="proxy.host" value="MyProxy" />
        <property name="proxy.port" value="8080" />
        <setproxy proxyHost="${proxy.host}" proxyPort="${proxy.port}"/>
        <!-- Load the files -->
        <loadfile property="files" srcFile="${curDir}/${filelist}"/>
        <foreach target="sget" param="value" list="${files}" delimiter="${line.separator}" parallel="true"/>
    </target>

    <!-- load a single file -->
    <target name="sget" if="value">
        <!-- split a line http://server/dir/file.txt=c:/temp/newfile.txt into uri (http://...) and dest (c:
/...)
        <propertyregex property="uri"  input="${value}" regexp="(.*)=(.*)" select="\1"/>
        <propertyregex property="dest" input="${value}" regexp="(.*)=(.*)" select="\2"/>
        <!-- check whether a destination filename is there. Otherwise use the same filename in the dest-
directory -->
        <script language="javascript"> <![CDATA[
            importClass(java.io.File);
            dest = new File(project.getProperty("dest"));
            if (dest.isDirectory()) {
                destDir = dest;
                uri = project.getProperty("uri");
                uri = uri.substring(uri.lastIndexOf("/")+1);
                destFile = new File(destDir, uri);
            } else {
                destDir = dest.getParent();
                destFile = dest;
            }
            project.setProperty("destDir", destDir);
            project.setProperty("destFile", destFile);
        ]]></script>
        <!-- get needs an existing dest-dir -->
        <mkdir dir="${destDir}"/>
        <get src="${uri}" dest="${destFile}" />
    </target>


    <!-- create a small sample project file -->
    <target name="mkproject" description="Erzeugt eine kleine build.xml">
        <copy file="buildxml.xml-template-small.xml" tofile="${curDir}/build.xml"/>
    </target>
    <target name="mkp" depends="mkproject"/>



    <!-- create a big build environment -->


    <!-- read initial project data from file -->
    <target name="create-read" if="create.file">
        <property file="${curDir}/${create.file}"/>
        <echo message="Propertyfile: ${create.file}"/>
        <echo message="Project Name: ${prj.name}"/>
        <echo message="Ant-Common  : ${ant.common}"/>
    </target>

    <!-- ask the user for initial project data -->
    <target name="create-input" unless="create.file">
        <echo>Please enter some required data for generating the files:</echo>
        <input addproperty="prj.name" message="project name:"/>

        <property name="antcommon.default" value="c:/seu/ant-common"/>
        <input addproperty="antcommon.tmp"
```

```
                message="Please input [default: '${antcommon.default}']:"/>
            <condition property="ant.common" value="${antcommon.tmp}">
                <not><equals arg1="" arg2="${antcommon.tmp}"/></not>
            </condition>
            <property name="ant.common" value="${antcommon.default}"/>
            <!--<input addproperty="ant.common" message="directory containing ant-snippets:"/>-->
            <echo message="Project Name: ${prj.name}"/>
            <echo message="Ant-Common  : ${ant.common}"/>
        </target>

        <!-- do the work -->
        <target name="create" depends="create-read,create-input">
            <property name="template.buildxml" value="buildxml.xml-template.xml"/>
            <property name="template.buildproperties" value="buildxml.prop-template.properties"/>
            <property name="template.bat" value="buildxml.bat-template.txt"/>

            <!-- create build.xml -->
            <copy tofile="${curDir}/build.xml" file="${template.buildxml}">
                <filterchain id="replace.filter">
                    <replacetokens>
                        <token key="projectName"  value="${prj.name}"/>
                        <token key="AntCommonDir" value="${ant.common}"/>
                    </replacetokens>
                </filterchain>
            </copy>

            <!-- create build.properties -->
            <copy tofile="${curDir}/build.properties" file="${template.buildproperties}">
                <filterchain refid="replace.filter"/>
            </copy>

            <!-- create create.txt for creation of buildenvironment whithout typing in the data again -->
            <copy tofile="${curDir}/create.bat" file="${template.bat}">
                <filterchain refid="replace.filter"/>
            </copy>
        </target>


        <target name="help" depends="usage"/>
        <target name="usage">
            <echo>
 zip            Open Notepad and zip all pattern
                -Dzipname     name of the zipfile (without suffif) in curDir
                -Dfiles2zip   name of pattern file

 create         Create a  build.xml and build.properties

 mkp, mkproject Create a small build.xml for tests

 download       Open Notepad and load all specified files (URI=dest).
            </echo>
        </target>

 </project>


== buildxml.xml-template-small.xml ==

 <?xml version="1.0" encoding="ISO-8859-1"?>
 <project name="test" basedir="." default="main">
     <taskdef resource="net/sf/antcontrib/antcontrib.properties"/>

     <target name="main">
     </target>

     <target name="script">
         <script language="javascript"> <![CDATA[
             // import statements
             importPackage(java.text);
             importClass(java.io.File);
         ]]></script>
```

```
    </target>
</project>


== buildxml.bat-template.txt ==
@echo off
echo prj.name=<at:var at:name="projectName" />>>create.txt
echo ant.common=<at:var at:name="AntCommonDir" />>>>create.txt
call antutil -Dcreate.file=create.txt create
del create.txt


== buildxml.prop-template.properties ==
# Property file for <at:var at:name="projectName" />
#-------------------------------------------------------------------------------

# Version of the program
version=1.0


# Directory settings
#
# Suggestions of "Java Development with Ant" - Appendix D - p.551
# for directory structure (used by target 'seed').
# Additionals from Appendix D: p.558 (test specific)
#---------------------------------------------------------------

# Temporary staging area for classes and more
build.dir=build

# Directory containing java *.class files and resources, ready for packaging.
build.classes.dir=${build.dir}/classes

# Directory for junit test classes and resources, ready for runned by junit.
test.classes.dir=${build.dir}/test

# Distribution directory
dist.dir=dist

# Documentation files stored in their presentation formats
docs.dir=docs

# Sample files
etc.dir=etc

# Project dependencies, typically third-party .jar files
lib.dir=lib

# Root directory of all Sources
src.dir=src

# Directory for application: Java source code, package directory below
# whithout test file
#
# Adapted from Ant´s own buildfile
java.dir=${src.dir}/main

# Directory where the JUnit-Tests lie.
test.src.dir=${src.dir}/test/junit

# Directory for test data, e.g. input data and expected results
test.data.dir=${src.dir}/test/data

# Documentation in XML format, to be transformed into presentation
# format during the build process
xdocs.dir=${src.dir}/xdocs

# Metadata for the JAR file
metainf.dir=${src.dir}/META-INF

# Root directory of web content (.html, .jpg, .JSP)
```

```
web.dir=${src.dir}/web

# Web deployment information, such as web.xml
webinf.dir=${web.dir}/WEB-INF

# Directory containing license files of the application and 3rd party libraries
license.dir=licenses


# Build-specific properties
#-------------------------------------------------------------
# Settings for java compiler (see manual for <javac>)
javac.debug=off
javac.deprecation=off
javac.target=1.2
javac.optimize=off
javac.compiler=modern

# Name of the JAR file containing the application
application.jar=${ant.project.name}-${version}.jar
application.tests.jar=${ant.project.name}-${version}-tests.jar
application.manifest=${metainf.dir}/application.mf

# Name of the DIST-File (without suffix)
dist.filename=${ant.project.name}-${version}-${DSTAMP}-${TSTAMP}


# Who should create the JavaDoc?
#   JDK      - javadoc tool from current JDK (default)
#   XDoclet  - XDoclet´s javadoc capabilities
#javadocs.kind=JDK
javadocs.dir=${build.dir}/api
javadocs.titel=${ant.project.name}

#copyright.text=All rights reserved.


# Test-specific properties
#-------------------------------------------------------------

# test.classes.dir: set in "Directory settings"
# test.src.dir: set in "Directory settings"

# Should JUnit use another JVM
junit.fork=true

# If the value is set, the build will fail if one testcase failes.
#test.halt-if-failed=

# Where is JUnit?
# See library settings
#junit.jar=

# Where is the XML output of JUnit runs
junit.out.dir.xml=${build.dir}/junit-xml

# Where should the HTML report generated by <junitreport>
junit.out.dir.html=${build.dir}/junit-html



# Library settings
#-------------------------------------------------------------
junit.jar=${lib.dir}/junit.jar
log4j.jar=${lib.dir}/log4j.jar


== buildxml.xml-template.xml ==
 <?xml version="1.0" encoding="ISO-8859-1"?>

 <project name="<at:var at:name="projectName" />" basedir="." default="main">
```

```
    <description>
        Something about the project.
    </description>

    <import file="<at:var at:name="AntCommonDir" />/all.xml"/>



<!-- ====================================================================== -->
<!-- Properties                                                             -->
<!-- ====================================================================== -->



<!-- ====================================================================== -->
<!-- task declarations                                                      -->
<!-- ====================================================================== -->



<!-- ====================================================================== -->
<!-- Datatype declarations                                                  -->
<!-- ====================================================================== -->
<path id="compile.classpath">
    <!-- contains all runtime libraries -->
    <path location="${log4j.jar}"/>
</path>
<property name="compile.cp" refid="compile.classpath"/>


<path id="test.classpath">
    <path refid="compile.classpath"/>
    <path location="${junit.jar}"/>
    <path location="${build.classes.dir}"/>
    <path location="${test.classes.dir}"/>
</path>
<property name="test.cp" refid="test.classpath"/>


<!-- ====================================================================== -->
<!-- Default target                                                         -->
<!-- ====================================================================== -->
    <target name="main"
            depends="all"
            description="Default target, calls 'all'"
    />


    <target name="all"
            depends="dist,test"
            description="Build and test everything; create a distribution"
    />


<!-- ====================================================================== -->
<!-- Build / compile / test targets                                         -->
<!-- ====================================================================== -->
    <target name="compile"
            depends=""
            description="compiles the sources">
        <mkdir dir="${build.classes.dir}"/>
        <javac srcdir="${java.dir}"
               destdir="${build.classes.dir}"
               debug="${javac.debug}"
               deprecation="${javac.deprecation}"
               target="${javac.target}"
               optimize="${javac.optimize}"
               compiler="${javac.compiler}">
            <classpath refid="compile.classpath"/>
```

```
    </javac>
</target>


<target name="archive" depends="jar"/>


<target name="jar"
        depends="compile"
        description="Jars the classes">
    <mkdir dir="${dist.dir}"/>
    <jar destfile="${dist.dir}/${application.jar}"
         basedir="${build.classes.dir}"
         excludes="**/*Test.class **/*TestCase.class"
         manifest="${application.manifest}"
    />
</target>


<target name="compile-tests"
        depends=""
        description="compiles test sources">
    <mkdir dir="${test.classes.dir}"/>
    <javac srcdir="${test.src.dir}"
           destdir="${test.classes.dir}"
           debug="${javac.debug}"
           deprecation="${javac.deprecation}"
           target="${javac.target}"
           optimize="${javac.optimize}"
           compiler="${javac.compiler}">
        <classpath refid="test.classpath"/>
    </javac>
</target>


<target name="jar-tests"
        depends="compile-tests"
        description="Jars the testcases">
    <mkdir dir="${dist.dir}"/>
    <jar destfile="${dist.dir}/${application.tests.jar}"
         basedir="${test.classes.dir}"
         includes="**/*Test.class  **/*TestCase.class"
         manifest="${application.manifest}"
    />
</target>


<target name="junit"
        depends="compile-tests"
        description="run the 'testcase'-specified or all unit tests">
    <!-- see "Java Development with Ant" - Appendix D - pp.558 -->
    <copy todir="${test.classes.dir}">
        <fileset dir="${test.src.dir}" excludes="**/*.java" />
    </copy>
    <mkdir dir="${junit.out.dir.xml}"/>
    <junit printsummary="no"
           errorProperty="test.failed"
           failureProperty="test.failed"
           fork="${junit.fork}">
        <classpath refid="test.classpath" />
        <sysproperty key="docs.dir" value="${test.classes.dir}"/>
        <formatter type="xml"/>

        <test name="${testcase}" if="testcase" />
        <batchtest todir="${junit.out.dir.xml}" unless="testcase">
            <fileset dir="${test.classes.dir}" includes="**/*Test.class" />
        </batchtest>
    </junit>
</target>
```

```
    <!-- Implementation, if no XML-Include would chosen. -->
    <target name="junitreport"
            depends=""
            description="Format the JUnit-Result">
        <mkdir dir="${junit.out.dir.html}" />
        <junitreport todir="${junit.out.dir.html}">
            <fileset dir="${junit.out.dir.xml}">
                <include name="*.xml"/>
            </fileset>
            <report format="frames" todir="${junit.out.dir.html}"/>
        </junitreport>
    </target>


    <target name="test"
            depends="junit,junitreport"
            description="Run the unit tests and format the result">
        <condition property="test.fail">
            <and>
                <isset property="test.failed" />
                <isset property="test.halt-if-failed" />
            </and>
        </condition>
        <fail if="test.fail">
            Unit tests failed. Check log or reports for details.
        </fail>
    </target>


<!-- ================================================================== -->
<!-- Documentation targets                                              -->
<!-- ================================================================== -->
    <target name="docs"
            depends="javadocs,printerdocs"
            description="Generate all documentation">
    </target>


    <target name="javadocs"
            depends=""
            description="Generate the Javadoc pages">
        <mkdir dir="${javadocs.dir}" />
        <condition property="bottom.text1" value="Copyright &#169; ${YEAR} ${copyright.text}&lt;br&gt;">
            <and>
                <not><equals arg1="" arg2="${copyright.text}"/></not>
                <isset property="copyright.text"/>
            </and>
        </condition>
        <property name="bottom.text1" value=""/>
        <javadoc
          destdir="${javadocs.dir}"
          classpath="${compile.cp}"
          author="true"
          version="true"
          use="true"
          Splitindex="false"
          access="protected"
          windowtitle="${javadocs.titel} API">

            <fileset dir="${java.dir}" defaultexcludes="yes">
                <include name="**/*.java" />
            </fileset>

            <doctitle><![CDATA[<h1>${javadocs.titel}</h1>]]></doctitle>
            <bottom><![CDATA[<i>${bottom.text1}Generated: ${DATE} ${TIME}</i> ]]></bottom>
            <tag name="todo" scope="all" description="To do:" />
        </javadoc>
    </target>
```

```
    <target name="printerdocs"
            depends=""
            description="Generate printable documents">
        <echo message="Target 'printerdocs' not yet implemented."/>
    </target>


    <target name="todo"
            depends=""
            description="Creates a ToDo-List">
        <!-- see "Java Development with Ant" - Ch.11 XDoclet p.262 -->
        <property name="xdoclet.home" value="C:/ZipDrive/02/XDoclet/xdoclet-bin-1.2b2/lib"/>
        <path id="xdoclet.classpath">
            <fileset dir="${xdoclet.home}">
                <include name="log4j.jar"/>
                <include name="xdoclet-1.2b2.jar"/>
                <include name="*.jar"/>
                <!--
                <include name=""/>
                <include name=""/>
                <include name=""/>
                -->
            </fileset>
        </path>
        <taskdef name="document"
                 classname="xdoclet.doc.DocumentDocletTask"
                 classpathref="xdoclet.classpath"/>
        <document sourcepath="${src.dir}"
                  destdir="${build.dir}/todo"
                  classpathref="xdoclet.classpath">
            <fileset dir="${src.dir}">
                <include name="**/*.java"/>
            </fileset>
            <info header="To-do list"
                  projectname="${ant.project.name}"
                  tag="todo"
            />
        </document>
    </target>


<!-- ================================================================================ -->
<!-- Distribution / install targets                                                   -->
<!-- ================================================================================ -->
    <target name="deploy"
            depends=""
            description="Deploy the code"> <!-- usually to a remote server -->
        <echo message="Target 'deploy' not yet implemented."/>
    </target>


    <target name="dist"
            depends="jar,jar-tests,docs"
            description="Produce the distribution">
        <zip destfile="${dist.dir}/${dist.filename}.zip">
            <fileset dir="${dist.dir}">
                <include name="${application.jar}"/>
                <include name="${application.tests.jar}"/>
            </fileset>
            <zipfileset dir="${javadocs.dir}" prefix="api"/>
            <!-- @TODO: add the printerdocs and anything else you want -->
        </zip>
        <tar destfile="${dist.dir}/${dist.filename}.tar">
            <tarfileset dir="${dist.dir}">
                <include name="${application.jar}"/>
                <include name="${application.tests.jar}"/>
            </tarfileset>
            <tarfileset dir="${javadocs.dir}" prefix="api"/>
        </tar>
        <gzip src="${dist.dir}/${dist.filename}.tar"
              zipfile="${dist.dir}/${dist.filename}.tar.gz"/>
```

```
        <bzip2 src="${dist.dir}/${dist.filename}.tar"
               zipfile="${dist.dir}/${dist.filename}.tar.bz2"/>

        <checksum>
            <fileset dir="${dist.dir}"/>
        </checksum>
    </target>


    <target name="distclean"
            depends=""
            description="Clean up the distribution files only">
        <delete includeEmptyDirs="true">
            <fileset dir="${dist.dir}" />
        </delete>
    </target>


    <target name="install"
            depends=""
            description="Perform a local installation">
        <echo message="Target 'install' not yet implemented."/>
    </target>


    <target name="uninstall"
            depends=""
            description="Remove a local installation">
        <echo message="Target 'uninstall' not yet implemented."/>
    </target>


<!-- ================================================================== -->
<!-- Internal targets                                                   -->
<!-- ================================================================== -->
    <target name="clean"
            depends=""
            description="Delete all generated files and directories">
        <delete includeEmptyDirs="true" failonerror="false">
            <fileset dir="${build.dir}" />
            <fileset dir="${dist.dir}" />
            <fileset dir="${docs.dir}" />
        </delete>
    </target>


    <target name="seed"
            depends=""
            description="Creates the directory structure">
        <!-- see "Java Development with Ant" - Appendix D - p.551 -->
        <mkdir dir="${build.dir}"/>
        <mkdir dir="${javadocs.dir}"/>
        <mkdir dir="${junit.out.dir.xml}"/>
        <mkdir dir="${junit.out.dir.html}"/>
        <mkdir dir="${src.dir}"/>
        <mkdir dir="${java.dir}"/>
        <mkdir dir="${test.src.dir}"/>
        <mkdir dir="${docs.dir}"/>
        <mkdir dir="${etc.dir}"/>
        <mkdir dir="${lib.dir}"/>
        <mkdir dir="${xdocs.dir}"/>
        <mkdir dir="${metainf.dir}"/>
        <mkdir dir="${web.dir}"/>
        <mkdir dir="${webinf.dir}"/>
        <mkdir dir="${build.dir}"/>
        <mkdir dir="${dist.dir}"/>
        <mkdir dir="${build.classes.dir}"/>
        <mkdir dir="${test.classes.dir}"/>
        <mkdir dir="${license.dir}"/>
        <mkdir dir="${test.data.dir}"/>
```

```
        <manifest file="${application.manifest}">
            <section name="Build">
                <attribute name="Built-By" value="${user.name}"/>
                <attribute name="Built-date" value="${DSTAMP}-${TSTAMP}"/>
            </section>
            <section name="${ant.project.name}">
                <attribute name="Version" value="${version}"/>
            </section>
            <!-- Package version info (see java.sun.com/<manifest>-manual) -->
            <section name="common">
                <attribute name="Specification-Title"    value="spec-title"/>
                <attribute name="Specification-Version"  value="${version}"/>
                <attribute name="Specification-Vendor"   value="${user.name}"/>
                <attribute name="Implementation-Title"   value="common"/>
                <attribute name="Implementation-Version" value="${version} ${TODAY}"/>
                <attribute name="Implementation-Vendor"  value="${user.name}"/>
            </section>
        </manifest>

        <echo file="${license.dir}/license.${ant.project.name}.txt"
              message="License for ${ant.project.name}"/>

        <!-- Create JBuilder projectfile JPX -->
    </target>


    <target name="example"
            depends="seed"
            description="creates an example application">
        <!-- unzip the files from example.zip into the configured directories -->
    </target>


    <target name="usage">
        <echo message="Execute 'ant -projecthelp' for build file help."/>
        <echo message="Execute 'ant -help' for Ant help."/>
        <echo message=" " />
        <echo message="Additional target specific information:"/>
        <echo message="  test, junit:    -Dtestcase=&lt;classname&gt; forces to run only that testcase"/>
        <!--
        <echo message="Executing 'ant -help'" />
        <exec executable="ant.bat" taskname="help"><arg value="-help"/></exec>
        <echo message="Executing 'ant -projecthelp'" />
        <exec executable="ant.bat" taskname="proj-help"><arg value="-projecthelp"/></exec>
       -->
    </target>


    <target name="help" depends="usage"/>


    <target name="debug">
        <echoproperties/>
    </target>

</project>
```