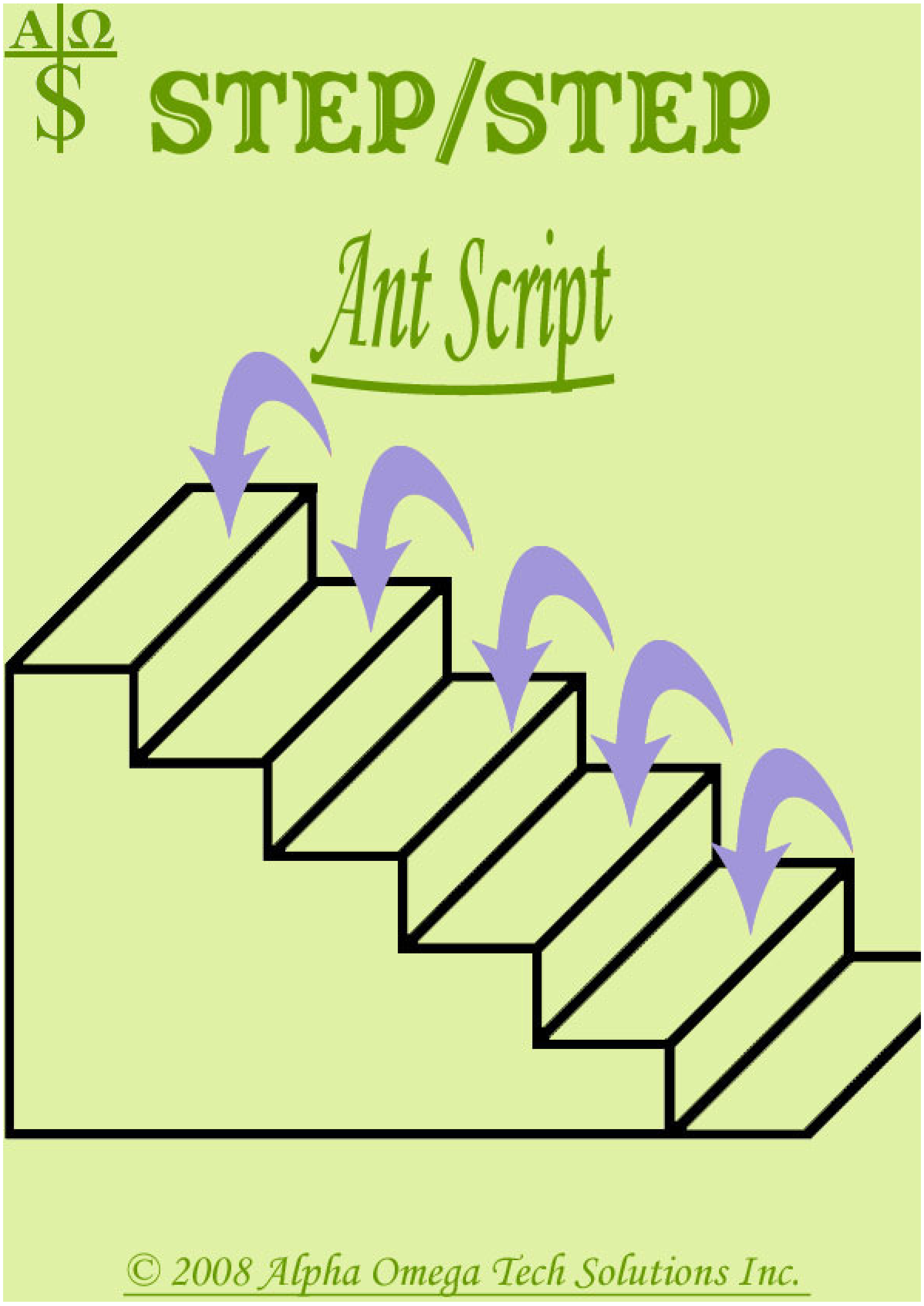

A|Ω
$
STEP/STEP
Ant Script
© 2008 Alpha Omega Tech Solutions Inc.

Table of Contents

About the Author

The author is sun certified expert in JAVA/J2EE technologies with over 8 years of experience in the industry. He has a systematic, methodical and practical approach in handling the projects he undertakes. Adding to his expertise list is his various projects on animations and Graphical designing. His new venture is the STEP/STEP documentation of various tips and tricks required for certain products and procedures.

1.0 Introduction

This document guides you Step/Step to learn and create Ant scripts using Jakarta Ant technology. Ant script is the robust and most frequent used build technology in the real world. This guide will give you Step/Step® process to learn Ant Scripts.

The Ant script needs an xml file (default file name is build.xml) to execute. The xml files has all the information with specific formats which can be understand by the ANT system

2.0 Download, Install and Configure ANT

Note!

The latest Ant jar and files can be downloaded from http://ant.apache.org/bindownload.cgi site. You can download the binary format in .zip etc or you can also download the source code. For the new users it is highly recommended to download the binary files.

Configuration

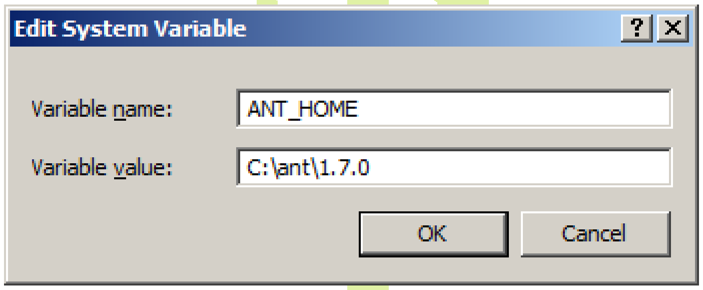
2.1 Setting up your machine with Ant

Follow the steps and setup the Ant in your machine.

1. Download and copy the files to your hard drive.
2. Unzip the file to anywhere in your hard drive.
3. Right click you're **My Computer** icon in the **Desktop** and select the properties.
4. In the properties popup select the **Advanced** tab and click **Environment Variables** button.
5. Click the **New** button in the **System Variable** section and enter the following

   Variable name - ANT_HOME

   Variable value – The ant path location in your system (for example C:\ant\1.7.0)



Edit System Variable

Variable name: ANT_HOME

Variable value: C:\ant\1.7.0

6. Click the **OK** button.
7. Set the **PATH** as below in your system (If you have a PATH variable exists in your system then append the ANT_HOME to it)

   set PATH=%PATH%;**%ANT_HOME%\bin**

3.0 Build.xml a glance

Note!

The build configuration file build.xml has the following main tags.

A. Project
    a. Target
        i. Tasks

3.1 Build xml a glance

As mentioned earlier the build configuration file (build.xml) has main tags. These tags have many extra features to have more control over your build process. For example start compiling only if there is a java file in the expected source folder or start the compressing or jar building target only if the class files are available etc.

Projects
Project tag has three attributes as
Name – name of the project
Default – Name of the default target to be used when no target is supplied
Basedir – the base directory
Description – description of the project

Targets
A target is a set of tasks which can be grouped together. We can control and set dependencies on these targets to prevent executing follow up tasks before completing the initial tasks.
Name – name of the target
Depends – name of the targets which has to be completed before this target. You can give more than one target names separated by comma (,)
If – the target will execute only if the condition passed.
Unless – the target will execute if the condition failed
Description – description of the target

Tasks
A task is a set of code or tags which can be executed in the build process. These tasks have different attributes to set. There are build in tasks, options tasks and also we can write our own tasks for better control.

Properties
A project can have set of properties which can be used for any task in that project. These properties shall be used to store the source folder, destination folder and als oany other properties.

Built-in Properties
Ant has access to all the System properties which has been mentioned in the following table. It has few build in properties which are mentioned in the following table.

System Properties Key	Description
java.version	Java Runtime Environment version
java.vendor	Java Runtime Environment vendor
java.vendor.url	Java vendor URL
java.home	Java installation directory
java.vm.specification.version	Java Virtual Machine specification version
java.vm.specification.vendor	Java Virtual Machine specification vendor
java.vm.specification.name	Java Virtual Machine specification name
java.vm.version	Java Virtual Machine implementation version
java.vm.vendor	Java Virtual Machine implementation vendor
java.vm.name	Java Virtual Machine implementation name
java.specification.version	Java Runtime Environment specification version
java.specification.vendor	Java Runtime Environment specification vendor
java.specification.name	Java Runtime Environment specification name
java.class.version	Java class format version number
java.class.path	Java class path
java.ext.dirs	Path of extension directory or directories
os.name	Operating system name
os.arch	Operating system architecture
os.version	Operating system version
file.separator	File separator ("/" on UNIX)
path.separator	Path separator (":" on UNIX)
line.separator	Line separator ("\n" on UNIX)
user.name	User's account name
user.home	User's home directory
user.dir	User's current working directory

Build in Properties Key	Description
basedir	the absolute path of the project's basedir (as set with the basedir attribute of <project>).
ant.file	the absolute path of the buildfile.
ant.version	the version of Ant
ant.project.name	the name of the project that is currently executing;
	it is set in the name attribute of <project>.
ant.java.version	the JVM version Ant detected; currently it can hold
	the values "1.2", "1.3", "1.4" and "1.5".
ant.home	home directory of Ant

Token Filters
This is a very good functionality provided in Ant. When copying files to a destination folder, this filter will replace the token in the files with a specified value. For example we can set the filter as

```
<filter token="version" value="5.0"/>
  <copy todir="${dest.dir}" filtering="true">
    <fileset dir="src"/>
  </copy>
```

The above code will copy all the files from the src.dir directory to the dest.dir directory replacing all the occurrences of the string @version@ with 5.0 in each file. Very cool feature hah.

Path-like Structures
The Ant has provision to set PATH and CLASSPATH type variables in the configuration file. We have different option like fileset, dirset, files, include and exclude options. We have to use : or ; to separate the arguments.

Command-line Arguments
We can pass variables from one task to another process or task. We have to use any one of the following attributes to pass command line arguments. We can give more than one command line arguments which have to be separated by ; or :

Attribute	Description
value	Single command line argument
file	To specify a file as argument
path	To specify a file as a normal string
pathref	This is the reference to the a pathref somewhere in the build file
line	space limited list of command line arguments.

The format is <arg value=”-ls -lat”/>

References
Any project element can be assigned an identifier using its id attribute. We can use reuse the element in any place by just referring the id. This will help to remove duplicate code in the configuration xml files. CLASSPATH or PATH is the best example for this

```
<project ... >
  <path id="project.class.path">
    <pathelement location="lib/"/>
    <pathelement path="/lib/tools.jar/"/>
  </path>

  <target ... >
    <compile ...>
      <classpath refid="project.class.path"/>
    </compile>
  </target>
</project>
```

4.0 Running Ant Scripts

Note!

If everything is set as per the installation process then running an Ant command is very simple. We have two options to run the ant script

1. Command line execution

ant <options> <target1 target 2 … target n>

Options	Descriptions
-help, -h	print this message
-projecthelp, -p	print project help information
-version	print the version information and exit
-diagnostics	Print information that might be helpful to diagnose or report problems.
-quiet, -q	be extra quiet
-verbose, -v	be extra verbose
-debug, -d	print debugging information
-emacs, -e	produce logging information without adornments
-lib <path>	specifies a path to search for jars and classes
-logfile <file>	use given file for log
-l <file>	use given file for log
-logger <classname>	the class which is to perform logging
-listener <classname>	add an instance of class as a project listener
-noinput	do not allow interactive input
-buildfile <file>	use given buildfile
-file <file>	use given buildfile
-f <file>	use given buildfile
-D<property>=<value>	use value for given property
-keep-going, -k	execute all targets that do not depend
	on failed target(s)
-propertyfile <name>	load all properties from file with -D
	properties taking precedence
-inputhandler <class>	the class which will handle input requests
-find <file>	search for buildfile towards the root of
-s <file>	the file system and use it
-noinput	A niceness value for the main thread: 1 (lowest) to 10 (highest); 5 is the default
-nouserlib	Run ant without using the jar files from /home/kev/.ant/lib
-noclasspath	Run ant without using CLASSPATH
-noclasspath	Java 1.5+ : use the OS proxies
-main <class>	override Ant's normal entry point

Few examples

ant → this will look for a build.xml in the current folder and execute it. If there is no file as build.xml then it will fail.

ant –buildfile config.xml → This will execute the ant scripts on config.xml

ant –buildfile config.xml deploy→ This will execute the ant scripts on config.xml and start with the target deploy

2. Java execution

   java -Dant.home=<ant home physical folder> org.apache.tools.ant.launch.Launcher [options] [target]

   For example

   java -Dant.home=c:\ant17 org.apache.tools.ant.launch.Launcher –buildfile config.xml deploy

5.0 Core Ant Tasks

Note!

This is the important section to know for a better configuration management. We have tags associated to different tasks to use in our build file. Each task has its own functionality. Here I have mentioned most of the important tasks. You can see the complete list of tasks inside your installable.

1. Ant: This task is used to call an ant script inside an ant script. We can invoke another build process from the current process. We can

   ```
   <ant antfile="subproject/subbuild.xml" target="compile" inheritrefs="false" inheritAll="true">
     <property name="file" value="data.properties"/>
   </ant>
   ```

2. AntCall: This task is used to call another target in the same build file. In the below example the targets compile and deploy are called form the main tags. Its not mandatory that these tags has to be called using antcall. It can be also controlled by depends attributes.

   ```
   <target name="main">
     <antcall target="compile">
       <param name="file" value="java"/>
     </antcall>

   <antcall target="deploy">
       <param name="file" value="jar"/>
     </antcall>
   </target>
   ```

3. AntStructure: This task is used to create a DTD for your build file.

   ```
   <antstructure output=”mydtd.dtd”>
   ```

4. Apply: This task is used to execute system commands. We can execute any UNIX commands with parameters etc. The following task will list all the files excluding .txt files from /com/aots/learnant folder.

   ```
   <apply executable="ls">
     <arg value="-lat"/>
     <fileset dir="/com/aots/learnant">
       <patternset>
          <exclude name="**/*.txt"/>
       </patternset>
     </fileset>
   </apply>
   ```

5. BuildNumber: This task is to track the build number. This task will read the number from the given file and increment and write it back. The default file name is build.number. We can specific any file name using the property file.

   ```
   <buildnumber/>
   <buildnumber file="aotsbuild.number"/>
   ```

6. GUnzip/BUnzip2: This task is used to unzip and expand the tar files with gz and bz2 extensions. These files are expanded in the current folder.

```
<gunzip src="com.aots.tar.gz" dest="test2.tar"/>

<bunzip src="com.aots.tar.gbz2" dest="subdir"/>

<gunzip dest=".">
  <url url="http://alphaomegatechsolutions.com/test.tar.gz"/>
</gunzip>
```

7. BZip2: This task is used to zip the source files using GZip or BZAip2 algorithms. The source(src) and the destination (destfile or zipfiles) attributes are mandatory

```
<gzip src="com.aots.test.tar" destfile="com.aots.test.tar.gz"/>

<bzip2 src="com.aots.test.tar" destfile="com.aots.test.tar.bz2"/>

<gzip destfile="com.aots.archive.tar.gz">
  <url url="ttp://alphaomegatechsolutions.com/test.tar"/>
</gzip>
```

8. Checksum: This task will generate checksum files using MD5 or SHA-1 algorithm. The same task can be used to verify the files checksum value.

```
<checksum file="test.java" forceOverwrite="yes"/>
<checksum file="test.java" verifyProperty="isMD5ok"/>
```

9. Chmod: This task is used to change the mode of any file.

```
<chmod file="build.sh" perm="755"/>
```

10. Condition: This task is used to setup properties based on conditions.

```
<condition property="isSunOSonSparc">
   <os name="SunOS" arch="sparc"/>
 </condition>
```

11. Copy: Copy a file to a new file or to a new folder.

```
<copy file="myfile.txt" tofile="mycopy.txt"/>
```

12. Copydir: Copy the whole directory tree from the source to the destination.

```
<copydir src="${src}/resources"
    dest="${dist}"
    includes="**/*.java"
    excludes="**/Test.java"
 />
```

13. Cvs: This tag is used to checkout files from CVS repository. We can get all the latest files in a specific folder before starting the build process.

```
 <cvs cvsRoot=":pserver:anoncvs@cvs.aots.com:/home/cvspublic"
     package="aots"
     dest="${dest.dir}"
 />
```

14. CvsChangeLog: This tag is used to get all the logs in the CVS repository in am xml format.

```
 <cvschangelog dir="dev/network"
            destfile="changelog.xml"
            daysinpast="10"
 />
```

```
<cvschangelog dir="dve/network"
        destfile="changelog.xml"
        start="01 Jan 2007"
        end="01 Jan 2008"
/>
```

15. CvsVersion: This task is used to get the CVS client and server version.
```
<cvsversion cvsRoot=":pserver:anoncvs@cvs.aots.com:/home/cvspublic"
    passfile="/home/myself/.cvspass"
    serverversionproperty="apachecvsversion"
    clientversionproperty="localcvsversion"
/>
```
16. CVSPass: This task will add a new password entry in the CVS password file.
```
<cvspass cvsRoot=":pserver:anoncvs@cvs.aots.com:/home/cvspublic"
    password="aotspass"
/>
```

17. Defaultexcludes: This task will update the default excludes for all the process below. We can both add and remove excludes
```
<defaultexcludes echo="true" add="*.java" remove="*.txt"/>
```

18. Delete: This task is used to delete a file, folder and its entire sub directories. We can use these properties includes, includesfile, excludes, excludesfile or defaultexcludes to add or remove files from deleting. But all these tasks are depreciated and it's recommended to use resource collections.
```
<delete>
  <fileset dir="." includes="**/*.bak"/>
</delete>
```

19. Diagnostics: This task used to run ant script in diagnostic mode
```
<target name="diagnostics" description="diagnostics">
  <diagnostics/>
</target>
```

20. Dirname: This task is used to identify the directory of a specified file and store in a property variable.
```
<dirname property="foo.dirname" file="foo.txt"/>
```

21. Ear: This task is used to create ear file for Enterprise Deployment.
```
<ear destfile="build/myapp.ear" appxml="src/metadata/application.xml">
  <fileset dir="build" includes="*.jar,*.war"/>
</ear>
```

22. Echo: This task is used to echo message during the build process. We can write the message in the console or to a specified file. We have options to setup the level of logging as error, warning, info, verbose and debug.
```
<echo file="aots.lgo" level="error">
build filed!!!!!!!!!!!!!!!!
</echo>

<echo message="Hello AOTS"/>
```

23. Exec: This task it used to execute a system command like exec or cmd etc
The following example set the PATH variable
```
<property environment="env"/>
<exec ... >
```

```
  <env key="PATH" path="/user/bin"/>
</exec>
```

The following example will open the index.html in IE

```
<property name="browser" location="C:/Program Files/Internet Explorer/iexplore.exe"/>
<property name="file" location="aots/main/index.html"/>
<exec executable="${browser}" spawn="true">
    <arg value="${file}"/>
</exec>
```

24. Fail: This task will exit the current build with an error message. We can also add in or unless properties in the fail task.
    `<fail message="Build filed"/>`

25. GenKey: This task is used to create a keystore for the encryption process.
    `<genkey alias="aots" storepass="AOTSPASS" dname="CN=AOTS, OU=Training, O=alphaomegatechsolutions.com, C=IN"/>`

26. Get: This task is used to get a file from a specified URL
    `<get src="http://www.alphaomegatechsolutions.com/" dest="temp/index.htm"/>`

27. Import: This task is used to import another build file into the current file. The optional will decide whether to stop the build if the file is not available.
    `<import file="../build-common.xml" optional=”true”/>`

28. Input: This task is used to create user interaction during the build process.

```
  <input
    message="Do you want to delete all class and recompile your Java files (Y/N)?"
    validargs="Y,N"
    addproperty="delete.classes"
  />
  <condition property="deleteclasses">
    <equals arg1="N" arg2="${delete.classes}"/>
  </condition>
  <delete if="deleteclasses" file="lib/class/*.class"/>
```

29. Jar: This task is used to create jar file from a specific folder. The includes or excludes properties can be used to add or prevent few files adding in the jar file

```
  <jar destfile="${dist}/lib/aots.jar"
      basedir="${build}/classes"
      includes="com/aots/**"
      excludes="**/Demo.class"
  />
```

30. Java: This task is used to execute a java class during the execution of the ant script. We can use all the properties used in the java command.

```
<java classname=”com.aots.test”>
<java jar=”aots.jar”>
```

31. Javac: This task is used to compile java files in the build process

```
  <javac srcdir="${src}"
        destdir="${build}"
        classpath="aots.jar"
       includes="com/aots/**"
       excludes="**/Demo.class"
        debug="on"
        source="1.4" />
```

32. Javadoc/Javadoc2: This task is used to create java documentation for the java classes.

33. Length: This task is used to find the length of a file or a string.
```
<length string="alphaomegatechsolutions" property="length" />
<length file="aots.txt" property="length" />
```

34. LoadFile: This task is used to load a file in the build process.
```
    <loadfile property="message" srcFile="message.txt"/>
```

35. LoadProperties: This task is used to load the contents of the properties file as ant properties. The properties should be as key=value format
```
    <loadproperties srcFile="file.properties"/>
```

36. LoadResource: This task will load a text resource to a single property.
```
<loadresource property="homepage">
  <url url="http://alphaomegatechsolutions.com/index.htm"/>
</loadresource>
```

37. MakeURL: This task is used to convert a file name to a URL.
```
    <makeurl file="*.java" property="java.url"/>
```

38. Mail: This task is used to send mail using SMTP.
```
<mail mailhost="smtp.alphaomegatechsolutions.com" mailport="1076" subject="Test Mail..">
  <from address="ajs@alphaomegatechsolutions.com"/>
  <replyto address="ajs@alphaomegatechsolutions.com"/>
  <to address="aaj@alphaomegatechsolutions.com"/>
  <message>The is the current build. Test the application once you get this mail.</message>
  <attachments>
    <fileset dir="dist">
      <include name="**/*.jar"/>
    </fileset>
  </attachments>
</mail>
```

39. Manifest: This task is used to create a manifest file.
```
  <manifest file="MANIFEST.MF">
    <attribute name="Built-By" value="kev"/>
    <section name="common">
      <attribute name="Specification-Title" value="Example"/>
      <attribute name="Specification-Version" value="1.1"/>
      <attribute name="Specification-Vendor" value="AOTS"/>
      <attribute name="Implementation-Title" value="common"/>
      <attribute name="Implementation-Version" value="1.1 January 01 2009"/>
      <attribute name="Implementation-Vendor" value="AOTS CORP"/>
    </section>
    <section name="com/aots/Demo.class">
      <attribute name="Sealed" value="false"/>
    </section>
  </manifest>
```

40. ManifestClassPath: This task will convert the path to a value for Manifest file.

41. Mkdir: This task is used to create a new folder
```
            <mkdir dir="build"/>
```

42. Move: This task is used to move a file to a new file or to a folder
<move file="a.txt" tofile="b.txt"/> - this can be also used for renaming the file.

```
<move file="a.txt" todir="new/dir"/>
```

43. Nice: This task is used to set the thread priority to the current process.
```
    <nice newpriority="10"/>
```

44. Parallel: This task is used to execute parallel process.

45. PreSetDef: This task is used to create preset definitions and can be used later by calling the name. This can be used like a function in java.
```
  <presetref name="createjar">
      <jar destfile="${dist}/lib/aots.jar"
         basedir="${build}/classes"
         includes="com/aots/**"
         excludes="**/Demo.class"
      />
  </presetdef>
```

Later it can use just by calling the name

```
</createjar>
```

46. Replace: This task is same as the filter task which replaces a token with a specified value.

47. ResourceCount: This task is used to find the number of files in a file list.
```
    <resourcecount property="counter">
      <filelist dir="." files="*.java" />
    </resourcecount>
```

48. Retry: This task is used to repeat a step few time if it fails.
```
<retry retrycount="3">
  <get src="aots.jar"
       dest="/com/build/aots.jar" />
</retry>
```

49. Rmic: This task is used to execute the rmic task in ant build. We can use includes or excludes properties. The property base is the destination folder for the class files.
```
    <rmic classname="com.xyz.FooBar" base="${build}/classes"/>
```

50. SignJar: This task is used to sign the jar file.
```
    <signjar jar="aots.jar" alias="aots" storepass="aotspass"/>
```

51. Sleep: This task is used to sleep the build process for a while. We can use milliseconds, seconds and hours also.
```
<sleep milliseconds="10"/>
```

52. Sql: This task is used to execute SQL commands using Java JDBC connectivity.
```
    <sql
       driver="jdbc driver name"
       url="jdbc url"
       userid="ui"
       password="pass"
       >
    insert
    insert into aots_users values (1, "aots1");
    </sql>
```

53. Sync: This task is used to synchronize two folders. This file will overwrite the data and also it will remove the files from the todir folder.
```
<sync todir="Server">
  <fileset dir="local"/>
</sync>
```

54. Tar: This task is used to create a tar file from the basedir folder
```
<tar destfile="${dist}/manual.tar" basedir="htdocs/manual"/>
```

55. Tempfile: This task is used to create a temp file in the build process.
```
<tempfile property="temp.file" suffix=".xml"/>
```

56. Touch: This task is used to update the last modified date of a file.
```
<touch file="test.txt" datetime="01/01/2008 12:00 am"/>
```

57. Truncate: This task is used to set the maximum length or size of a file. We can set from 0 byte, KB, MB, GB, TB(TerraBytes) and PB (Petabytes)
```
<truncate file="foo" length="1K" />
```

58. Unjar: This task is to un-jar the given tar file.
```
<unzip src="aots.zip" dest="com/aots/"/>
```

59. Untar: This task is to un-tar the given tar file.

60. Unwar: This task is to un-war the given tar file.

61. Unzip: This task is to unzip the given tar file.

62. War: This task is used to create a war file.
63. Zip: This task is used to create a zip file. A zipfileset property is used to create a zip with multiple zip files.

6.0 Optional tasks

1. Attrib: This task is used to set the properties of a file
```
<attrib file="run.bat" readonly="true" hidden="true"/>
```

2. Chgrp: This task is used to change a group of a specified file.
```
<chgrp file="start.sh" group="coders"/>
```

3. Chown: This task is used to change the owner of a file.
```
<chown file="start.sh" owner="coderajs"/>
```

4. ClearCase: These tasks are used to do all functionalities in the ClearCase server.
   a. CCCheckin – This task is used to check-in files.
   b. CCCheckout – This task is used to checkout files.
   c. CCUnCheckout – This task is used to undo the checkout files.
   d. CCUpdate – This task is used to perform the ClearCase update.
   e. CCMklabel – This task is used to create label for the view
   f. CCLock – This task is used to lock a file.
   g. CCUnlock – This task is used to unlock an locked files
   h. CCMkdir – This task is used to create a directory in the ClearCase

   Sample
```
<ccmcheckin file="c:/AOTS.java" comment="Success"/>
<ccmcheckout file="c:/AOTS.java" comment="Success"/>
```

5. ejbc: This task is used to do EJB compilation on the EJB files.
```
<ejbc descriptors="DEST"
        src="src"
        dest="DEST"
        manifest="MANIFEST.MF"
        classpath="buildclasspath">
  <include name="*.ser"/>
</ejbc>
```

6. FTP: This task is used to ftp files to a ftp server
```
<ftp server="ftp.apache.org"
    remotedir="incoming"
    userid="anonymous"
    password="me@myorg.com"
    depends="yes">
  <fileset dir="htdocs/manual"/>
</ftp>
```

7. JavaCC: This task is to call JavaCC (Java Compiler Compiler) to generate grammar files to process.

8. Javah: This task is used to generate JNI headers from java files.

9. JspC: This task is used to compile JSP files and convert to java source files.

10. JUnit: This task is used to execute test scripts from JUNIT.

11. Pvcs: This task is used to connect to PVCS version control server to do any process.

12. Scp: This task is used to copy a file to or from a remote server.
```
<scp file="myfile.txt" todir="userid:password@hostname:/home/data"/>
```

13. Sound: This task is used to create sound during the build process for specific tasks.
```
<sound>
  <success source="sound.wav"/>
</sound>
```

14. Splash: This task is used to create a splash image during the build process
```
<splash imageurl="aotslogo.gif" useproxy="true" showduration="5000"/>
```

15. Sshexec: This task is used to run a command in the remote server.
```
<sshexec host="hostname" username="uid" password="pass" command="ls -lat"/>
```

16. Telnet: This task is used to get a telnet session in the build process.
```
<telnet userid="uid" password="pass" server="aots">
  <read>/home/test</read>
  <write>ls</write>
  <read string="/home/test"/>
</telnet>
```

17. WebLogic JSP Compiler: This task is used to compile a JSP with WebLogic JSP compiler
```
<wljspc src="source" dest="dest" package="aots.jsp">
  <classpath>
    <pathelement location="${weblogic.classpath}"/>
    <pathelement path="${compile.dest}"/>
  </classpath>
</wljspc>
```

7.0 An example

The first step is to create the application code. For this sample process we can write the famous Hello World program.

In the following example we are going to do the following process using Ant script

a) Verify whether the java file is available to proceed.

b) Compile the java file and convert it to class file

c) Bundle it to a jar file.

7.1 Step 1: Create the java program as below

```
package com.aots.learn;

public class HelloWorld {

        public HelloWorld()
        {
                System.out.println("Hello World!!!!!!!!!");
        }

        public static void main(String args[])
        {
                new HelloWorld();
        }
}
```

Note: Make sure the java file is created in the right package.

7.2 Step 2: Create build.xml

We have to create the build configuration files as below. This file has to be places in the root folder. The following file has the necessary tasks to verify the java file, compile and bundle it to a jar file.

If you are not very clear about the tasks then refer to section 5 or 6.

```
<?xml version="1.0" ?>
<project default="main">
        <property name="src" location="src"/>
        <property name="build" location="build"/>
        <property name="dist"  location="dist"/>
  <!-- Main Task -->
  <target name="main" description="Main target">
    <echo>
      Building the .jar file.
    </echo>
  </target>
        <!-- File Verification Task -->
        <target name="file.check">
          <condition property="file.run">
            <and>
              <available file="src/com/aots/learn/HelloWorld.java"/>
            </and>
          </condition>
        </target>

        <!-- Java Compilation Task-->
  <target name="compile" description="Compilation target" depends="file.check" if="file.run">
        <echo>Java File is available. Start the process...</echo>
    <javac srcdir="src/com/aots/learn"/>
  </target>
        <!-- Jar Bundling Task-->
 <target name="compress" description="Compression target" depends="compile">
    <jar jarfile="Project.jar" basedir="src/com/aots/learn" includes="*.class" />
 </target>
</project>
```

7.3 Step 3: System Verification

Make sure that the following system properties are set properly as mentioned in the earlier steps.

a) JAVA_HOME

b) ANT_HOME

c) PATH

d) CLASSPATH

7.4 Step 4: Run the build.xml files

You can use any IDE like eclipse to run the Ant scripts, but it is not necessary. We can run the ant script using DOS window.

a) Open the DOS window and go to the root folder where you have the build.xml and base package folder.

1. Run the ant command to execute the Ant script. Since we have named the configuration files as build.xml the system will recognize the file.

8.0 Appendices A: References

1. Visit www.java.sun.com to get more information about JDK.

2. Visit http://ant.apache.org/ to get more information on Apache Ant scripts.