ANT - Training

Mithun Reddy Lacchannagari
(mithunreddytechnologies@gmail.com)

- Agenda

- ✓ Introduction
- ✓ Installation
- ✓ Content – build.xml
- ✓ Examples

- Introduction

Type	JAVA based Open source Build Tool
Vendor	Apache
Is Open Source?	Yes
Version	1.10.1
Operating system	Cross Platform
Creator	James Duncan Davidson, who is also the original author of Tomcat.
Software Download URL	http://ant.apache.org/bindownload.cgi
Is executable software?	No, download as zip, extract and use it.
Reference Websites	1)

- Introduction

Apache Ant is a Java based build technology from Apache Software Foundation.
Apache Ant's build files are written in XML.
The acronym for ANT is "Another Neat Tool".
Ant is an Apache project. It is open source software, and is released under the Apache Software License.

Pre requisites

Java
XML

- Introduction

http://en.wikipedia.org/wiki/James_Duncan_Davidson

James Duncan Davidson, while he was software engineer at Sun Microsystems (1997–2001), created Tomcat Java-based web server, still widely use in most of the Java web projects, and also Ant build tool, which uses XML to describe the build process and its dependencies, which is still the de facto standard for building Java-based Web applications.

List of Popular Build tools

SNo	Product	Vendor	Is Open Source?
1	Ant	Apache	Yes
2	Maven	Apache	Yes
3	Gradle	Apache	Yes

- Introduction

Why we need to choose ANT

- ANT is Java based build tool
- It is cross platform
- ANT scripts are written in XML, if we already familiar with XML, we can learn ANT easily.
- Ant can be easily invoked from the command line and it can integrate with free and commercial IDEs.
- It can create Java Doc
- It can checkout the code from Version Control Systems like SVN, Gut, CVS…
- It can execute the test scripts and test suites.
- It can Packaging the binaries (jar, war or ear)

- Installation

1) Download ANT software it is a ZIP file
http://ant.apache.org/bindownload.cgi

2) Copy *apache-ant-1.10.1-bin.zip* it into C:\Ant drive and extract into this(C:\Ant) directory, we will see four folders namely

C:\Ant\apache-ant-1.10.1

1) \bin ----> It contains the batch files, like ant.bat….
2) \etc ----> It contains the schemas and some extra jars.
3) \lib ----> It contains the jar files, like ant.jar
4) \manual ----> It contains the documentation about ANT.

- Sample - build.xml

```
<?xml version="1.0" ?>
<project name="Hello World" default="run" basedir=".">

        <target name="compile">
                <javac srcdir="." />
                <echo> Compilation Completed! </echo>
        </target>

         <target name="run" depends="compile">
         <echo> Running Java program! </echo>
    <java classname="HelloWorld"/>
        </target>

</project>
```

- Contents of build.xml

<?xml version="1.0"?>
Since Ant build files are XML files the document begins with an XML declaration which specifies which version of XML is in use.

The ***<project>*** element is the root element in Ant build file.

The project tag has only three attributes:
The *name* attribute indicates the project name.
The *default* attribute indicates the target name which called by default.
The *basedir* attribute indicates the the base directory having all path, it contain absolute

e.g. <project name="HelloWorld" default="run" basedir=".">

- Contents of build.xml

The ***<target>*** element represents a single stage in the build process. A build process can have multiple targets. Here we have two targets.

The target element have six attributes called
name : The name of the target
depends : a comma-separated list of names of targets on which this target depends.
description : A short description of this target's function.
id : Unique identifier for this task instance, can be used to reference this task in scripts.
if : the name of the property that must be set in order for this target to execute.
unless : the name of the property that must not be set in order for this target to execute.

Note: Each project element can contain multiple <*target*> elements.

- Contents of build.xml

- The default attribute of project element indicates the default target to be executed. Here the default target is *run*.
- When you see the *run* target, it in turn depends on the *compile* target, that is indicated by the *depends* attribute. So the *compile* target will be executed first.
- The compile target has two task elements <*javac*> and <*echo*>. The *javac* task is used to compile the java files. The attribute *srcdir="."* indicates all the java files in the current directory. *echo* task is used to display message on the console.
- The *run* target also performs two tasks, first the <echo> element displays the message in console and<*java*> element is used to run the Java program.

- Contents of build.xml

In above build.xml we used <java> and <jar> tags.

The <*java*> tag has following important attributes:
classname : Sets the Java class to be execute.
dir : The working directory of the process.
fork : If the fork attribute value is *true* , executes in a new VM.
jar : The location of the JSR file to execute.\

The <*jar*> tag following important attributes:
basedir : Directory from which to archive files.
destfile : The file to create.
update : If this attribute is true, I will update, otherwise it will overwrite existing file. The default value is false.
include : This attribute is to add specific files to jar.

The <*copy*> tag contains the following important attributes.
1)overwrite : Overwrite any existing destination file(s). By default value is fale.
2)todir : Sets the destination directory.

- Contents of build.xml

- Run the ANT script (build.xml) as follows.

ant

- We can run specific task as follows.

#ant compile
ant run

- If you don't pass any options along with ant command by default it will run build.xml file.
- Suppose if we have more than one build files with different names, we need to use –f or -buildfile option to run particular build file as follows.

#ant –f anotherbuildlfile.xml
#ant –buildfile anotherbuildfile.xml

- Properties

Properties are key – value pairs.
Ant provides 2 types of properties.

1) Built-in Properties
2) User defined properties.

- Built-in Properties - Examples

Ant provides the following pre-defined properties that can be used in the build file.

Properties	Description
ant.file	The full location of the build file.
ant.version	The version of the Apache Ant installation.
basedir	The basedir of the build, as specified in the basedir attribute of the project element.
ant.java.version	The version of the JDK that is used by Ant.
ant.project.name	The name of the project, as specified in the nameatrribute of the project element.
ant.project.default-target	The default target of the current project.
ant.project.invoked-targets	Comma separated list of the targets that were invoked in the current project.
ant.core.lib	The full location of the Ant jar file.
ant.home	The home directory of Ant installation.
ant.library.dir	The home directory for Ant library files - typically ANT_HOME/lib folder.

- User-defined - Properties

In addition to the built-in-properties Ant supports User defined properties.
Ant uses the **property element which allows you to specify user-defined-properties as follows.**
<property name="technology" value="Apache ANT"/>

- Property File

Defining the user-defined properties in build file, we will keep in separate file called properties file.
The file name usually we will give as build.properties . There is no hard and fast rule on file name. you can give any name with **“.properties**” extension.

Use the below tag to inclue the properties file in build file.

<property file="build.properties"/>

- Build – Script Examples

Questions ?

Thank you