FREE eBook

LEARNING

ant

Free unaffiliated eBook created from
Stack Overflow contributors.

#ant

Table of Contents

About 1

Chapter 1: Getting started with ant 2

Remarks 2

Minimum Java Versions 2

Versions 2

Examples 3

Hello World 4

Bootstrap Apache Ivy 4

Print environment information before build 5

Run JUnit 7

Create jar package 7

Installation or Setup 8

Chapter 2: Ant Properties 9

Introduction 9

Examples 9

How to declare and use property in Ant. 9

Chapter 3: Ant Tasks for Git 11

Examples 11

Get started: 11

Clone 11

Pull 11

Add / Commit / Push 11

Chapter 4: Basic File IO 13

Examples 13

Write a file using echo 13

Print the contents of a file 13

Credits 14

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ant

It is an unofficial and free ant ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ant.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ant

Remarks

This section provides an overview of what Ant is, and why a developer might want to use it.

It should also mention any large subjects within Ant, and link out to the related topics. Since the Documentation for Ant is new, you may need to create initial versions of those related topics.

Minimum Java Versions

Various versions of Ant require different versions of the Java runtime (the JRE) in order to run.

Ant Version	Minimum Java Version
1.1 up to 1.5.4	1.1
1.6.x releases	1.2
1.7.x releases	1.3
1.8.x releases	1.4
1.9.x releases	1.5
1.10.x releases	1.8

Versions

Version	Release Date
1.4.1	2001-10-11
1.5.0	2002-07-10
1.5.1	2002-10-03
1.5.2	2003-03-03
1.5.3	2003-04-09
1.5.4	2003-08-12
1.6.0	2003-12-18
1.6.1	2004-02-12

Version	Release Date
1.6.2	2004-07-16
1.6.3	2005-04-28
1.6.4	2005-05-19
1.6.5	2005-06-02
1.7.0	2006-12-13
1.7.1	2008-07-09
1.8.0	2010-02-02
1.8.1	2010-04-30
1.8.2	2010-12-20
1.8.3	2012-03-13
1.8.4	2012-05-23
1.9.0	2013-03-10
1.9.1	2013-05-22
1.9.2	2013-07-12
1.9.3	2013-12-29
1.9.4	2014-04-30
1.9.5	2015-06-03
1.9.6	2015-07-02
1.9.7	2016-04-12
1.9.8	2016-12-31
1.9.9	2017-02-06
1.10.0	2016-12-31
1.10.1	2017-02-06

Examples

Hello World

Add the following to a file named `build.xml` in your project directory:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="HelloWorld" default="main">
    <target name="main" description="this is target main">
        <echo message="Hello World" />
    </target>
</project>
```

From a command prompt on a computer running Windows, executing `ant main` will display similar to the following:

```
$ ant main
Buildfile: C:\Users\<me>\Projects\HelloWorld\build.xml

main:
     [echo] Hello World

BUILD SUCCESSFUL
```

Also, user can now run the command `ant` as `default` target name added to the project. When `ant` command is run, it looks for project's `default` target and execute it.

```
$ ant
Buildfile: C:\Users\<me>\Projects\HelloWorld\build.xml

main:
     [echo] Hello World

BUILD SUCCESSFUL
```

If the build script is written by some one else and the end user like to see what target he can run, run the command which will show all the targets which has descriptions.

```
$ ant -p
```

Bootstrap Apache Ivy

Add the following target in your `build.xml`

```
<!-- Bootstrap ivy -->
<target name="ivy.bootstrap" description="Download Apache Ivy">

    <!-- Define the version to use -->
    <property name="ivy.version">2.4.0</property>

    <!-- Create directory if not exists -->
    <mkdir dir="${user.home}/.ant/lib" quiet="true" />

    <!-- Download it -->
    <echo message="Downloading Apache Ivy..." />
```

```
    <get dest="${user.home}/.ant/lib/ivy.jar"
src="https://repo1.maven.org/maven2/org/apache/ivy/ivy/${ivy.version}/ivy-${ivy.version}.jar"
/>
</target>
```

After running the task `ant ivy.bootstrap`, you will now be able to resolve dependencies using apache ivy.

```
<target name="ivy.resolve" description="Resolve all artifacts.">

    <!-- Define lib driectory -->
    <property name="dir.lib">lib</property>

    <!-- Create directory if not exists -->
    <mkdir dir="${dir.lib}" />

    <!-- Configure -->
    <property name="ivy.dep.file" value="ivy.xml" />
    <ivy:settings file="ivysettings.xml" />

    <!-- Retrieve to a defined pattern -->
    <echo message="Resolving dependencies..." />
    <ivy:retrieve pattern="${dir.lib}/[artifact](-[classifier]).[ext]" />
</target>
```

Define your resources in `ivy.xml`

```
<ivy-module version="2.0">
        <info organisation="org.apache" module="java-build-tools"/>
        <dependencies>
                <dependency org="junit" name="junit" rev="4.11" />
                <dependency org="org.apache.commons" name="commons-compress" rev="1.9" />
        </dependencies>
</ivy-module>
```

And any custom repositories in `ivysettings.xml`

```
<ivysettings>
    <settings defaultResolver="chain"/>
    <resolvers>
        <chain name="chain">
            <ibiblio name="central" m2compatible="true"/>
            <ibiblio name="github" m2compatible="true" root="http://github.com/"/>
        </chain>
    </resolvers>
</ivysettings>
```

Download your dependencies by running `ant ivy.resolve`.

Print environment information before build

The following is handy to have in build logs that identifies the build machine, and some parameters; simply make you `main` task depend on this task to print it before every build.

```
<!-- Print Environment Info -->
<target name="environment">

        <!-- Get the current timestamp -->
        <tstamp>
                <format property="TODAY_UK" pattern="yyyy-MM-dd HH:mm:ss:sss zzz"
locale="cn,CN" />
        </tstamp>

        <!-- Get the hostname of the system -->
        <exec executable="hostname" outputproperty="os.hostname" />

        <!-- Print a bunch of information -->
        <echo message="" />
        <echo message="  Build Information" />
        <echo message="" />
        <echo message="  OS Information" />
        <echo message="" />
        <echo message="    User       : ${user.name}" />
        <echo message="    Hostname   : ${os.hostname}" />
        <echo message="" />
        <echo message="    Name       : ${os.name}" />
        <echo message="    Version    : ${os.version}" />
        <echo message="    Build      : ${os.arch}" />
        <echo message="" />
        <echo message="" />
        <echo message="  Java Information" />
        <echo message="" />
        <echo message="    Version    : ${ant.java.version} / ${java.version}" />
        <echo message="    Java Home  : ${java.home}" />
        <echo message="" />
        <echo message="" />
        <echo message="  Current Time : ${TODAY_UK}" />
        <echo message="" />
</target>
```

This will result in the following output,

```
environment:
     [echo]
     [echo]   Build Information
     [echo]
     [echo]   OS Information
     [echo]
     [echo]     User       : <User Name>
     [echo]     Hostname   : <Host Name>
     [echo]
     [echo]     Name       : Windows 8.1
     [echo]     Version    : 6.3
     [echo]     Build      : amd64
     [echo]
     [echo]
     [echo]   Java Information
     [echo]
     [echo]     Version    : 1.8 / 1.8.0_45
     [echo]     Java Home  : C:\Program Files\Java\jdk1.8.0_45\jre
     [echo]
     [echo]
     [echo]   Current Time : 2016-04-18 00:40:11:011 EDT
```

Run JUnit

The following will run JUnit on the tests matching `test/**/*Test.java`. This need the junit.jar to be in the `lib` folder.

```
<project name="Project" default="junit" basedir=".">
    <path id="classpath">
        <fileset dir="lib" includes="**/*.jar"/>
        <pathelement path="build"/>
    </path>

    <target name="compile">
        <javac srcdir="test" destdir="build" classpathref="classpath"/>
    </target>

    <target name="junit" depends="compile">
        <junit fork="true" logfailedtests="false">
            <classpath refid="classpath"/>
            <batchtest>
                <fileset dir="test" includes="**/*Test.java"/>
                <formatter type="plain" usefile="false"/>
            </batchtest>
        </junit>
    </target>
</project>
```

Create jar package

The following will create `dist/output.jar` from the source code in `src` and the libraries in `lib`, and will use `src/Main.java` as the main class.

```
<project name="Project" default="main" basedir=".">

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

    <path id="classpath">
        <fileset dir="lib" includes="**/*.jar"/>
        <pathelement path="${build.dir}"/>
    </path>

    <target name="clean">
      <delete dir="${build.dir}"/>
      <delete dir="${dist.dir}"/>
    </target>


    <target name="compile">
        <mkdir dir="${build.dir}"/>
        <javac srcdir="${src.dir}" destdir="${build.dir}" classpathref="classpath"/>
        <copy todir="${build.dir}">
            <fileset dir="${src.dir}" excludes="**/*.java"/>
        </copy>
    </target>

    <target name="jar" depends="compile">
```

```
        <mkdir dir="${dist.dir}"/>
        <jar destfile="${dist.dir}/${ant.project.name}.jar" basedir="${build.dir}">
            <fileset dir="${build.dir}"/>
            <restrict>
                <archives>
                    <zips>
                        <fileset dir="lib" includes="**/*.jar"/>
                    </zips>
                </archives>
            </restrict>
            <manifest>
                <attribute name="Main-Class" value="Main"/>
            </manifest>
        </jar>
    </target>

    <target name="main" depends="clean,jar"/>
</project>
```

Installation or Setup

Installing Ant is very simple. Follow the steps given below to install Ant on windows platform:

1. Download latest ant version from Apache website
2. Unzip the file on your machine.
3. Set ANT_HOME in environment variables
4. Add %ANT_HOME%\bin to your PATH environment variable.
5. Set CLASSPATH=%ANT_HOME%\lib;%CLASSPATH%
6. Now open command prompt and enter `ant` command. You should see below:

```
Buildfile: build.xml does not exist!
Build failed
```

Alternatively, using Homebrew on macOS or Linuxbrew on Linux you can simply run: `brew install ant`

When using brew it isn't necessary to set up the environment variables.

Several Linux distributions also support installing Ant from their respective package managers.

To test Ant is installed properly navigate to the command prompt and execute

```
ant -version
```

This command will print the Ant version and also shows that Ant was successfully installed.

Ant's own installation instructions page is available on the Apache Ant website.

Read Getting started with ant online: https://riptutorial.com/ant/topic/4223/getting-started-with-ant

Chapter 2: Ant Properties

Introduction

Properties are key-value-pairs where Apache Ant tries to expand ${key} to value at runtime.

Ant properties are very helpful if you have to do a lot to processing to create installables or do custom deployments etc.

For example, you can mark ${src.dir} as source code directory,${lib.dir} as library for project,${javadoc.dir} for javadocs etc.

Instead of writing complete path at every location you can refer them by this place holder.

Examples

How to declare and use property in Ant.

Ant provides some built-in properties

Property Name	Value
basedir	the absolute path of the project's basedir
ant.file	the absolute path of the buildfile.
ant.version	the version of Ant
ant.project.default-target	the name of the currently executing project's default target
ant.project.name	name of the project
ant.java.version	JVM version Ant detected

In this example, We will create custom ant properties and use them to create a temporary directory and copy a file in it.

1. **Properties declared within same file.**

```
<project name="Test Project for Ant" default="init">
    <property name="temp.dir" value="${basedir}/temp" />

    <target name="init" description="initialize">
        <mkdir dir="${temp.dir}" />
        <copy file="${basedir}/test.xml" todir="${temp.dir}/" />
    </target>
</project>
```

In Ant, ${basedir} will refer to the base location or the location where your ant file is present. Here I declared a property named as

temp.dir

which will refer to basedir/temp location.

So, we call target init it will replace the placeholder ${temp.dir} with its actual value and start executing our script. This target will create a directory named temp under base directory copy test.xml file to temp directory.

2. **Properties declared in different files.**

In this example, We will refer properties declared in different file. This is a sample file(app_version.xml) which contains application version.

```
<project name="Project Properties">
     <property name="app.version" value="1.0" />
</project>
```

To include this file we will add the import ant task to import this file while executing ant targets.

```
<import file="app_version.xml" />
```

Above code will look like

```
<project name="Test Project for Ant" default="init">
<import file="app_version.xml" />
<property name="temp.dir" value="${basedir}/temp" />

<target name="init" description="initialize">
    <mkdir dir="${temp.dir}" />
    <copy file="${basedir}/test.xml" todir="${temp.dir}/" />
    <echo message="App version is:${app.version}" />
</target>
```

Once the file is imported, it can be directly accessed via property name(app.version).

I used .xml file, same use case will also work for .properties files.

Read Ant Properties online: https://riptutorial.com/ant/topic/9181/ant-properties

Chapter 3: Ant Tasks for Git

Examples

Get started:

```
<macrodef name = "git">
    <attribute name = "command" />
    <attribute name = "dir" default = "" />
    <element name = "args" optional = "true" />
    <sequential>
        <echo message = "git @{command}" />
        <exec executable = "git" dir = "@{dir}">
            <arg value = "@{command}" />
            <args/>
        </exec>
    </sequential>
</macrodef>
<macrodef name = "git-clone-pull">
    <attribute name = "repository" />
    <attribute name = "dest" />
    <sequential>
        <git command = "clone">
            <args>
                <arg value = "@{repository}" />
                <arg value = "@{dest}" />
            </args>
        </git>
        <git command = "pull" dir = "@{dest}" />
    </sequential>
</macrodef>
```

Clone

```
<git command="clone">
    <args>
        <arg value = "-v" />
        <arg value = "git@YOURGITURL:GITUSER/GITREPO" />
        <arg value = "repo" />
    </args>
</git>
```

Pull

```
<git command = "pull" dir = "repository_path" />
```

Add / Commit / Push

```
<input message="Commit message" addproperty="commit-message" />
<git command="add">
    <args>
```

```
                    <arg value="." />
    </args>
</git>
<git command="commit">
    <args>
                    <arg value="-am ${commit-message}" />
    </args>
</git>
<git command="push" />
```

Read Ant Tasks for Git online: https://riptutorial.com/ant/topic/7893/ant-tasks-for-git

Chapter 4: Basic File IO

Examples

Write a file using echo

Simply specifying a *file* destination, `echo` will create, write, or append to a file.

```
<echo file=example.txt" append="false">
    hello world
</echo>
```

Print the contents of a file

To print the contents of a file, we can use `loadfile` to read a file into a local property, and then use `echo` to print the value of it.

```
<loadfile property="contents" srcFile="example.txt" />
<echo message="${contents}" />
```

Read Basic File IO online: https://riptutorial.com/ant/topic/5932/basic-file-io

Credits

S. No	Chapters	Contributors
1	Getting started with ant	Chad Nouis, Community, fgb, Josh Doug, Lucas Oliveira, Matt Clark, Maverick, Rao, Squidward
2	Ant Properties	Maverick
3	Ant Tasks for Git	Lucas Oliveira
4	Basic File IO	Matt Clark