**Ant**

Apache Ant (Another Neat Tool) is an open source project started by Apache Software Foundation. Ant is a Java library and a software tool used for automate software build processes such as compile, run, test and assemble Java application. It is designed and developed by Apache Software Foundation and initially released on 19 July 2000.

It is a better alternate of Make build tool of Unix. Ant is written in Java and require JVM to build the Java projects.

Ant uses XML to describe build code and by default it's XML file name is build.xml.

Apache Ant Features

* Open Source
* Flexible
* Easy To Use
* Cross Platform
* Extensible
* Scalable
* XML

**Open Source**

Apache Ant is an open source library. It allows user to access source code and reproduce it. Ant has open source software license. It can be used to study, redistribute

**Flexible**

Ant is very flexible in nature, it can be used with variety of programming languages without much effort. Apart from Java, other useful programming languages like C, C++ can also use Ant to automate their tasks.

**Easy To Use**

Ant uses very simple syntax which is easy to learn and use. If a user already know about XML, it is more easy to work because Ant uses XML. It's build file is created using XML tags.

**Cross Platform**

Apache Ant is cross-platform and handles Java class paths and file directory structures in aportable manner. It can execute on various platforms.

**Extensible**

Apache Ant is extensible in nature, it is easy to extend. It can extensible using Java and other programming languages also.

**Scalable**

Apache Ant has built-in support for J2EE development, such as EJB compilation and packaging. It can be used in a small personal project and can also be used in a large software project.

**XML Markup**

Ant uses XML files called build files to describe how to build, test and deploy an application. Using XML, it enables developers to edit files directly in any XML editor. It facilitates parsing the build file at run time. Using XML as the format also allows enables developers to create templates easily and to generate build files dynamically.

**How to install Apache Ant:**

1. Make sure you have a Java environment installed.
2. Download Ant.

The latest stable version of Ant is available from the Ant web page <https://ant.apache.org/>

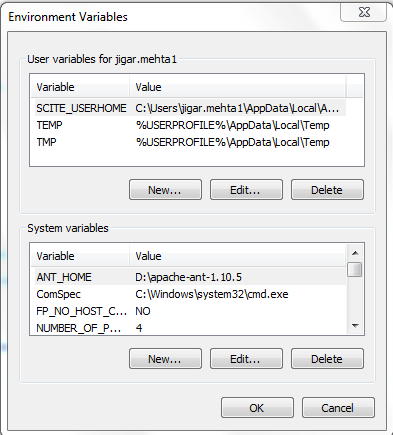
The binary distribution of Ant is available as 3 different archives

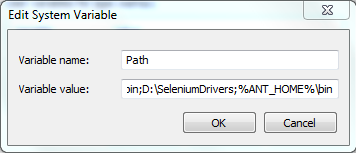
1. .zip—Recommended compression format for Windows, can also be used on other platforms. Supported by many programs and some operating systems natively.
2. .tar.gz—Using the tar program to gather files together, and gzip to compress and uncompress.
3. .tar.bz2—Using the tar program to gather files together, and bzip2 to compress and uncompress.

Choose the format that is best supported for your platform.

1. Uncompress the downloaded file into a directory.

Set environmental variables: JAVA\_HOME to your Java environment, ANT\_HOME to the directory you uncompressed Ant to, and %ANT\_HOME%/bin (Windows) to your PATH.





The binary distribution of Ant consists of the following directory layout:

ant

+--- README, LICENSE, fetch.xml, other text files. //basic information

+--- bin // contains launcher scripts

+--- lib // contains Ant JARs plus necessary dependencies

+--- manual // Ant documentation (a must read ;-)

+--- etc // contains xsl goodies to:

// - create an enhanced report from xml output of various tasks.

// - migrate your build files and get rid of 'deprecated' warning

// - ... and more ;-)

Only the bin and lib directories are required to run Ant.

To install Ant, choose a directory and copy the distribution files there. This directory will be known as ANT\_HOME.

**Check Installation**

You can check the basic installation with opening a new shell and typing ant. You should get a message like this

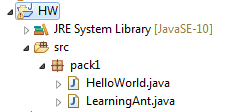
Buildfile: build.xml does not exist!

Build failed

So Ant works. This message is there because you need to write a buildfile for your project. With a ant -version you should get an output like

Apache Ant(TM) version 1.9.2 compiled on July 8 2013

Create a Project and create below java files



**package** pack1;

**public** **class** HelloWorld {

**public** **static** **void** main(String[] args) {

System.***out***.println("Hello World");

}

}

**package** pack1;

**public** **class** LearningAnt {

**public** **static** **void** main(String[] args) {

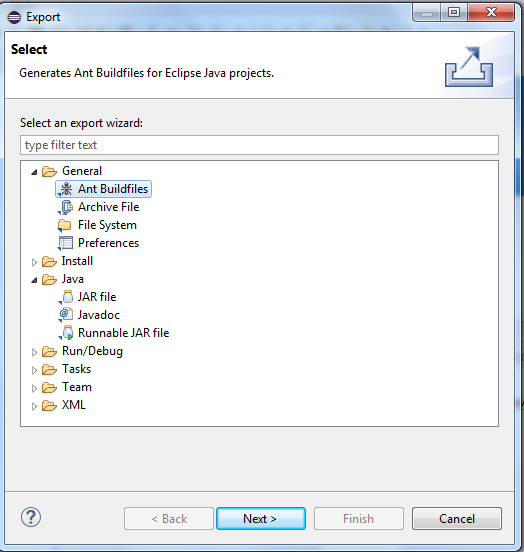
System.***out***.println("Learning Ant");

}

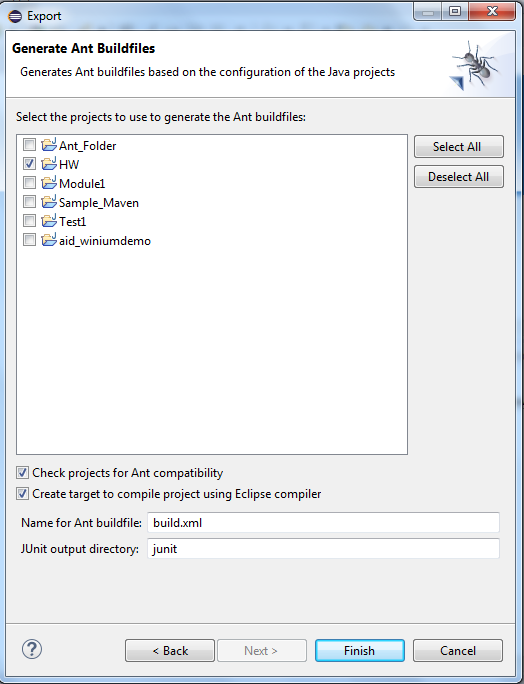
}

How to create build.xml in Eclipse?

1. Right click on Project HW and select option ‘Export’. Below screen will appear. Select Ant Buildfiles and select ‘Next’.



1. Select Finish



Below build.xml file is created

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!-- WARNING: Eclipse auto-generated file.

Any modifications will be overwritten.

To include a user specific buildfile here, simply create one in the same

directory with the processing instruction <?eclipse.ant.import?>

as the first entry and export the buildfile again. -->

<project basedir="." default="build" name="HW">

<property environment="env"/>

<property name="debuglevel" value="source,lines,vars"/>

<property name="target" value="10"/>

<property name="source" value="10"/>

<path id="HW.classpath">

<pathelement location="bin"/>

</path>

<path id="run.HelloWorld.classpath">

<path refid="HW.classpath"/>

</path>

<target name="init">

<mkdir dir="bin"/>

<copy includeemptydirs="false" todir="bin">

<fileset dir="src">

<exclude name="\*\*/\*.launch"/>

<exclude name="\*\*/\*.java"/>

</fileset>

</copy>

</target>

<target name="clean">

<delete dir="bin"/>

</target>

<target depends="clean" name="cleanall"/>

<target depends="build-subprojects,build-project" name="build"/>

<target name="build-subprojects"/>

<target depends="init" name="build-project">

<echo message="${ant.project.name}: ${ant.file}"/>

<javac debug="true" debuglevel="${debuglevel}" destdir="bin" includeantruntime="false" source="${source}" target="${target}">

<src path="src"/>

<classpath refid="HW.classpath"/>

</javac>

</target>

<target description="Build all projects which reference this project. Useful to propagate changes." name="build-refprojects"/>

<target description="copy Eclipse compiler jars to ant lib directory" name="init-eclipse-compiler">

<copy todir="${ant.library.dir}">

<fileset dir="${ECLIPSE\_HOME}/plugins" includes="org.eclipse.jdt.core\_\*.jar"/>

</copy>

<unzip dest="${ant.library.dir}">

<patternset includes="jdtCompilerAdapter.jar"/>

<fileset dir="${ECLIPSE\_HOME}/plugins" includes="org.eclipse.jdt.core\_\*.jar"/>

</unzip>

</target>

<target description="compile project with Eclipse compiler" name="build-eclipse-compiler">

<property name="build.compiler" value="org.eclipse.jdt.core.JDTCompilerAdapter"/>

<antcall target="build"/>

</target>

<target name="HelloWorld">

<java classname="pack1.HelloWorld" failonerror="true" fork="yes">

<classpath refid="run.HelloWorld.classpath"/>

</java>

</target>

</project>

How to run buid.xml?

Right click on build.xml and select Run as-> Ant Build

Below output is displayed:

Buildfile: D:\Eclipse SimRel WorkSpace\HW\build.xml

build-subprojects:

init:

build-project:

[echo] HW: D:\Eclipse SimRel WorkSpace\HW\build.xml

build:

BUILD SUCCESSFUL

Total time: 994 milliseconds

Observation:

Using command prompt run HelloWorld and Learning Ant and see the output. It will display the

Hello World

Learning Ant

Make changes to these java files, run the build.xml from Eclipse and run the class files from CMD. The changes made to these class files will be displayed.

How to run build.xml through cmd?

1. Ant (and pres enter)
2. Ant –f build2.xml