**ANT**

**1. What is ANT?**

ANT full form is Another Needed Tool. Ant is a build tool that is java based. A build tool performs the following tasks:

Compiling java code into byte code.  
Placing this byte code in a package.  
Deployment to production systems.  
Document creation and release notes preparation.

**Capabilities of ANT:**

ANT tool is extended by using java classes. The configuration files are XML-based. Each task of building directory tree is executed by using the object that implements the Task interface.

ANT provides the cross-platform deployment that can run on any platform.

**2. Explain ANT functionality.**

Ant is an open source project available under the Apache license. Therefore, its source code can be downloaded and modified.

Additionally, Ant uses XML build files which make its development easy.  
Cross Platform.

Use of XML along with Java makes Ant makes it the perfect solution for developing programs designed to run or be built across a range of different operating systems.  
Extensible.

New tasks are used to extend the capabilities of the build process, while build listeners are used to help hook into the build process to add extra error tracking functionality.

As Ant is extensible and open, it can be integrated with any editor or development environment easily.

**3. What is a build tool?**

A built tool is software which is used to build project, directory structure, copy necessary files to that directory ,compile files ,create jars, set path and class-path ,Build the documentation ,Validate the source code, deploy, debug, and run, clear the workspace.

4. **Why Ant is a great build tool?**

 Ant is great build tool due to following reason:

1. Ant is a Java-based build tool designed to be cross-platform, easy to use, extensible, and scalable.

2. Ant can be used in a small personal project as well as ant can be used in a large, multi-team software project.

3. Ant syntax is very easy to learn.

4. Ant syntax used XML format .We need only specifies our task only on build.xml file.

5. Ant is easy to use .eliminating the full-time make file engineer common on large Make-based software projects.

**4. How you can explain ant property?**

A project can have a set of properties .A property has name and value .The name is case sensitive and Properties are immutable this mean once set property its will not change. Properties may be used in the value of task attributes.

**5. What is dependency? How it is used into ant? What is its use?**

Dependencies are do something when complete it. In ant we are using dependencies by using an attribute “depends” .In this attribute we have pass values for which the target depends .This mean we first need to execute the target which is passed into this attribute.

## 6. Tell me how to set classpath in ant?

PATH- and CLASSPATH-type references can be specified using both ":" and ";" as separator characters. Ant converts the separator to the correct character of the current operating system.  
<classpath>  
<pathelement path="${classpath}"/>  
<pathelement location="lib/helper.jar"/>  
</classpath>  
The location attribute specifies a single file or directory relative to the project's base directory (or an absolute filename)  
The path attribute accepts colon- or semicolon-separated lists of locations. The path attribute is intended to be used with predefined paths - in any other case; multiple elements with location attributes should be preferred.  
The <classpath> tag supports path and location attributes of its own:  
<classpath path="${classpath}"/>

## 7. Tell me how to start to use Ant and provide a "Hello World" ant script?

Before start using ant, we should be clear about the project name and the .java files and most importantly, the path where the .class files are to be placed.  
  
For example, we want the application HelloWorld to be used with ant. The Java source files are in a subdirectory called Dirhelloworld, and the .class files are to put into a sub directory called Helloworldclassfiles.  
  
1. The build file by name build.xml is to be written. The script is as follows  
  
<project name=”HelloWorld” default=”compiler” basedir=”.”>  
<target name=”compiler”>  
<mkdir dir = “Helloworldclassfiles”>  
<javac srcdir=”Dirhelloworld” destdir=”Helloworldclassfiles”>  
</target>  
</project>  
  
2. Now run the ant script to perform the compilation:  
  
C :> ant  
Buildfile: build.xml

## 8. How to use ant to run a Java application?

The following is an example to run a java application in using ant:  
  
public class HelloWorld {

public static void main(String[] args) {

// Prints "Hello, World" to the terminal window.

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

}

}

Create two **targets** : **compile** and **run** in your **build.xml** file :

<project name="Hello World" basedir=".">

<target name="compile">

<javac srcdir="." includeantruntime="true" includes="HelloWorld.java" destdir="."></javac>

</target>

<target name="run" depends="compile">

<java classname="HelloWorld" fork="true"></java>

</target>

</project>

**MAVEN**

**1. What is Maven?**

Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

Maven handles following activities of a developer

• Build  
• Documentation  
• Reporting  
• Dependencies  
• SCMs  
• Releases  
• Distribution  
• Mailing list

**2. What does it mean when you say Maven uses Convention over Configuration?**

Maven uses Convention over Configuration which means developers are not required to create build process themselves. Developers do not have to mention each and every configuration details.

**3. What is POM?**

POM stands for Project Object Model. It is fundamental Unit of Work in Maven. It is an XML file. It always resides in the base directory of the project as pom.xml. It contains information about the project and various configuration details used by Maven to build the project(s).

**4. Name the 3 build lifecycle of Maven.**

The three build lifecycles are −

* **clean:**cleans up artifacts created by prior builds.
* **default (or build):**This is used to build the application.
* **site:** generates site documentation for the project.

**5. What is a Maven Repository?**

A repository is a place i.e. directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

**6. What is dependency scope? Name all the dependency scope.**

Dependency scope includes dependencies as per the current stage of the build. Various Dependency Scopes are −

* **compile −** This scope indicates that dependency is available in classpath of project. It is default scope.
* **provided −** This scope indicates that dependency is to be provided by JDK or web-Server/Container at runtime.
* **runtime −** This scope indicates that dependency is not required for compilation, but is required during execution.
* **test −** This scope indicates that the dependency is only available for the test compilation and execution phases.
* **system −** This scope indicates that you have to provide the system path.
* **import −** This scope is only used when dependency is of type pom. This scope indicates that the specified POM should be replaced with the dependencies in that POM's <dependencyManagement> section.

**7. What is a system dependency?**

Dependency with scope system are always available and are not looked up in repository, they are usually used to tell Maven about dependencies which are provided by the JDK or the VM. Thus, system dependencies are especially useful for resolving dependencies on artifacts which are now provided by the JDK.

**8. What is the use of optional dependency?**

Any transitive dependency can be marked as optional using "optional" element. As example, A depends upon B and B depends upon C. Now B marked C as optional. Then A will not use C.

**9. List out the build, source and test source directory for POM in Maven?**

• Build = Target  
• Source = src/main/java  
• Test = src/main/test

**10. What is a Mojo?**

A. A mojo is a Maven plain Old Java Object . Mojo is associated with a Maven goal. A custom plugin is a set of related Mojos (or goals) in a single plugin artifact.

**11. List out what are the build phases in Maven?**

Build phases in Maven are

* **Validate**
* **Compile**
* **Test**
* **Package**
* **Integration-tests**
* **Verify**
* **Install**
* **Deploy**

**ANT VS MAVEN**

|  |  |
| --- | --- |
| **Ant** | **Maven** |
| Ant **doesn't has formal conventions**, so we need to provide information of the project structure in build.xml file. | Maven **has a convention** to place source code, compiled code etc. So we don't need to provide information about the project structure in pom.xml file. |
| Ant is **procedural**, you need to provide information about what to do and when to do through code. You need to provide order. | Maven is **declarative**, everything you define in the pom.xml file. |
| There is **no life cycle** in Ant. | There is **life cycle** in Maven. |
| It is **a tool** box. | It is **a framework**. |
| It is **mainly a build tool**. | It is **mainly a project management tool**. |
| The ant scripts are **not reusable**. | The maven plugins are **reusable**. |
| It is **less preferred** than Maven. | It is **more preferred** than Ant. |