Lab7 report: Comparing build tools for Java

11510352 李子强

In this report, I will explore three Java build automation tools which includes Ant, Maven, and Gradle. For the release time, Ant came first, Maven went next, and Gradle was the latest. As for build tools, two main application is dependency management and build life-cycle management.

Apache Ant is short for Another Neat Tool which is a Java library used for automating build processes for Java applications. In the past, Make was the only build automation tool. However, a lot of notation from C programs didn't fit Java ecosystem, so Ant comes out as a better alternative. Ant can be used for non-Java applications. It was initially part of Apache Tomcat codebase and was released as a standalone project in 2000.

In many aspects, Ant is very similar to Make, and it's simple enough so anyone can start using it without any prerequisites. Ant build files are written in XML, and by convention, they're called build.xml. Different phases of a build process are called "targets".

The main benefit of Ant is its flexibility. Ant doesn't impost any coding conventions or project structures. Consequently, it means that Ant require developer to write all the command by themselves. If a project is relative large, XML build files are too complex to maintain.

At first, Ant had no build-in support for dependency management. However, as dependency management became a must in the later years/ Apache Ivy was developed as a sub-project of the Apache Ant project. It's integrated with Apache Ant, and it follows the same design principles.

However, the initial Ant limitations due to not having built-in support for dependency management and inconvenience when working with unmanageable XML build files led to the creation of Maven.

Maven is released in 2004 to give a better build solution than Ant. Maven still use XML files, but within more manageable way. Maven relies on conventions and provides predefined commands (goal). Besides, Maven support download dependency from Internet automatically, which revolutionized the way we develop software. However, Maven has its own problem. Dependency management cannot handle conflict well between different version of library files.

There are three built-in build life-cycles in Maven: default, clean and site. The default life-cycle handles your project deployment, the clean life-cycle handles project cleaning, while the site life-cycle handles the creation of your project's site documentation.

Maven became very popular since build files were now standardized and it took significantly less time to maintain build files, comparing to Ant. However, though more standardized than Ant files, Maven configuration files still tend to get big and verbose.

Maven's strict conventions come with a price of being a lot less flexible than Ant. Goal customization is very hard, so writing custom build scripts is a lot harder to do, compared with Ant.

Gradle combines the former two and made

many improvements based on that. It has the power and flexibility of Ant and Maven's life cycle management and easy to use. As a result, Gradle was published in 2012 and quickly gained wide attention. For example, Google uses Gradle as the default build tool for Android.

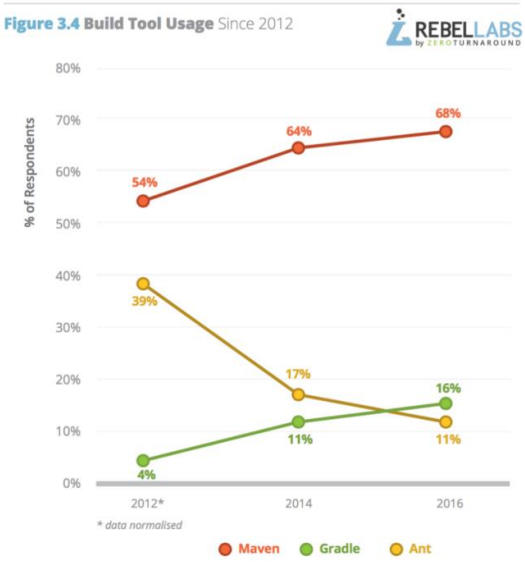
A Gradle build has three distinct phases. During the initialization phase, Gradle determines which projects are going to take part in the build and creates a Project instance for each of these projects. During the configuration phase. The build scripts of all projects which are part of the build are executed. During the Execution phase, Gradle determines the subset of the tasks, created and configured during the configuration phase, to be executed. The subset is determined by the task name arguments passed to the Gradle command and the current directory. Gradle then executes each of the selected tasks.

One of the first things we can note about Gradle is that it's not using XML files, unlike Ant or Maven.

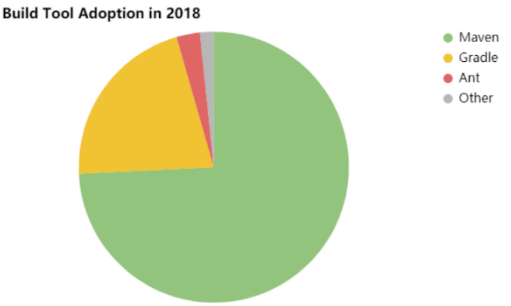
Over time, developers became more and more interested in having and working with a domain specific language – which, simply put, would allow them to solve problems in a specific domain using a language tailored for the domain.

This was adopted by Gradle, which is using a DSL based on Groovy. This led to smaller configuration files with less clutter since the language was specifically designed to solve specific domain problems. Gradle's configuration file is by convention called build.gradle.

For beginners, Ant is the clearest. If you read the XML configuration file, you can understand what it does, but the Ant file can easily become complicated. Maven and Gradle, especially Gradle, have a lot of out-of-the-box plugins. However, for skillful developer, Gradle's DSL is shorter and easier to understand than Ant and Maven. Not only that, only Gradle provides both conventions and custom commands.



As the statistics from Internet, Maven keeps more than half market share. However, if Gradle's adoption rate does increase, although Gradle seems to be taking more share from Ant than Maven. Gradle will continue to rise steady, particularly if the support from the Android community continues.



The statistics shows in 2018 Maven isn't going anywhere. The tool was sitting at a comfortable 75.7% last year, and it commands 74.2% of the market now. As for Gradle, it clawed a bit more of the market, mostly from Ant, and now has just shy of 1/5 of the market - 21.3%.