Build Automation

Build automation involves scripting or automating the process of compiling computer source code into binary code. A comprehensive list of build automation systems can be found on wikipedia (https://en.wikipedia.org/wiki/List_of_build_automation_software).

From Building and Testing With Gradle, "*Gradle is a flexible yet model-driven JVM-based build tool. Gradle acknowledges and improves on the very best ideas from Make, Ant, Ivy, Maven, Rake, Gant, Scons, SBT, Leinengen, and Buildr. The best-of-breed features previously scattered among a set of tools are now made available via a unified Groovy DSL for scripting and Java API for tooling*."

XML was a popular choice as a build script format as it seemed to be human-readable, and it was very easy to write code to parse it. Experience has shown that large and complex XML files are only easy for machines to read, not for humans. More importantly, XML's strictly hierarchical structure limits its expressiveness. XML is designed to express hierarchical relationships (test is an ordered hierarchy of content objects) not program flow and data access (which is what is needed in a complex build file).

Gradle uses a domain specific language, the Gradle API accessed via Groovy. Every Gradle build file is an executable Groovy program. Unlike the XML formats of Ant and Maven, Gradle's Groovy-based build files allow you to do general-purpose programming tasks in your build file. This relieves much of the frustration developers have faced in lacking control flow in Ant or being forced into plug-in development in Maven to accomplish nonstandard tasks.

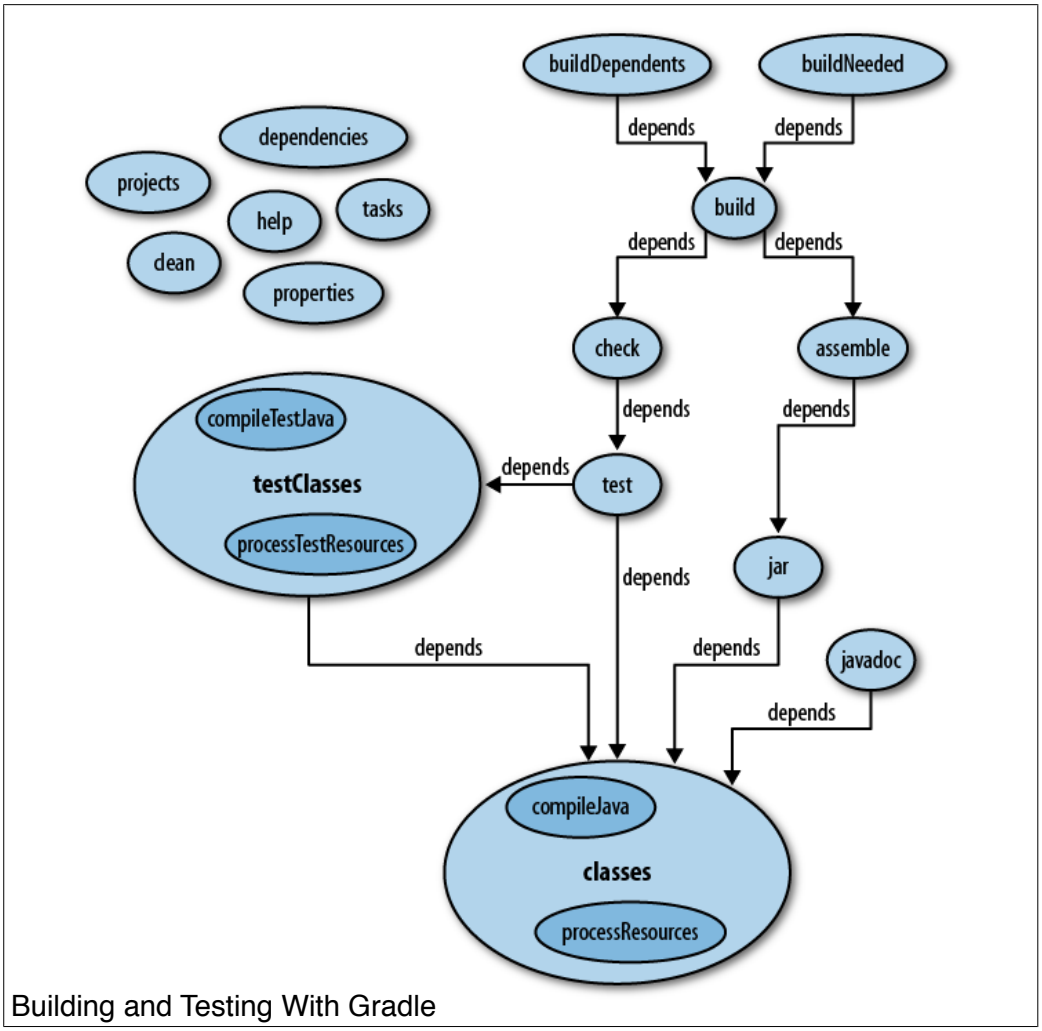
The Gradle Command Line

- `--help` or `-h` Prints out the help messages describing all command-line options.
- `-Dproperty=value` Defines a system property. This is a useful mechanism for passing parameters into a build from the command line.
- `--info` or `-i` Sets the Gradle log level to INFO, which causes a few more informative messages to be emitted over the default setting.
- `--debug` or `-d` Turns out debug logging for the build, including stack traces. This generates a lot of output, but can be quite useful for troubleshooting build problems.
- `--dry-run` or `-m` Evaluates and runs the build file, but does not execute any task actions.
- `--quiet` or `-q` Suppresses most output, showing error messages only.
- `--gui` Launches the Gradle GUI.
- `--stacktrace` or `-s` Emits an abbreviated stack trace when an exception is thrown by the build. Normally, stack trace logging is suppressed, so this is a helpful switch

when debugging a broken build. The stack trace is abbreviated by removing frames related to purely internal Groovy method calls.

- `--full-stacktrace` or `-S` Emits a longer version of the `--stacktrace` output, including all internal Groovy method calls. These are usually not of interest to the build developer.
- `properties` Emits all the properties of the build's Project object. The Project object is an object representing the structure and state of the current build.
- `tasks` Emits a list of all tasks available in the current build file. Note that plug-ins may introduce tasks of their own, so this list may be longer than the tasks you have defined.

Grade Tasks as a DAG



Building and Testing With Gradle

Working With Gradle

In this practical we will be working with a Gradle build template for Coursera's Algorithms, Part I course. This course is based on the book “Algorithms”, (4th Edition) by Robert Sedgewick and Kevin Wayne. The course consists of a number of homework assignments where solutions to assignments are placed into weekN/src directories. Manual test and JUnit tests are placed into weekN/test directories.

This project covers the entire build cycle of a Gradle project, verify, resolve dependencies, compile, test, assemble and deploy/execute. The project consists of several subproject, it uses various plugins and has external dependencies.

Resources

- The Algorithms repository https://github.com/kevin-wayne/algs4
- The Coursera Build Template https://github.com/pniederw/algs4partI

The Build Environment

The lab machines access the internet via a proxy. We may have to configure grade to recognise the HTTP/HTTPS proxy. Configuring an HTTP or HTTPS proxy is done via standard JVM system properties. These properties can be set directly in the build script; for example, setting the HTTP proxy host would be done with

System.setProperty('http.proxyHost', 'www.somehost.org').

Alternatively, the properties can be specified in a gradle.properties file, either in the build's root directory or in the Gradle home directory.

Configuring an HTTPS proxy in gradle.properties

systemProp.https.proxyHost=www.somehost.org
systemProp.https.proxyPort=8080