Gradle Command-Line

Table of Contents

이번 문서에서는 Gradle command-line의 기본 사용법을 설명한다.

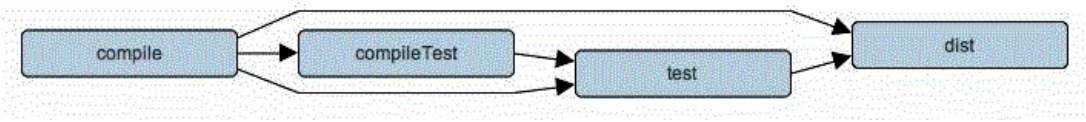
기본 명령어

Gradle Command Line 은 명령어 창에서 **gradle [option...] [task...]** 를 입력해서 실행하게 된다. 그러면 명령어에 의해 build.gradle이 실행되어 빌드를 수행하게 된다.
command line의 자세한 설명을 알고 싶으면 gradle --help 명령어를 이용하면 된다.

멀티 태스크(Multiple tasks) 실행

프로젝트를 빌드할 때 커맨드라인상에서 여러 태스크를 동시에 실행할 수 있다. 예를 들어 gradle compile test 명령어는compile 태스크를 실행한 뒤 test 태스크를 실행한다.

Gradle은 태스크를 수행할 때 의존성있는 태스크까지 같이 수행하게 된다.
예를 들어 아래에 4가지의 태스크가 있다. 그 중에서 test 태스크와 dist 태스크는 compile 태스크에 의존하고 있다.



코드로 보면 다음과 같다.

```
task compile << {
    println 'compiling source'
}

task compileTest(dependsOn: compile) << {
    println 'compiling unit tests'
}

task test(dependsOn: [compile, compileTest]) << {
    println 'running unit tests'
}

task dist(dependsOn: [compile, test]) << {
    println 'building the distribution'
}
```

gradle dist test 를 실행해보자.

```
:compile
compiling source
:compileTest
compiling unit tests
:test
running unit tests
:dist
building the distribution

BUILD SUCCESSFUL

Total time: 5.709 secs
```

결과를 보면 태스크들이 compile -> compileTest -> test -> dist 순으로 순서대로 실행되면서 중복으로 의존되고 있는 compile은 한 번만 실행된 것을 알 수 있다. 즉 빌드 중에 어떤 태스크가 여러 가지의 다른 태스크를 의존하고 있더라도 해당 태스크는 오직 **한 번만 실행**된다.

태스크 제외하기

-x 옵션과 태스크 이름을 포함해서 실행하면 해당 태스크는 빌드시 제외된다.
gradle dist -x test 로 test 태스크를 제외해본다.

```
:compile
compiling source
:dist
building the distribution
```

태스크명 약어(Abbreviated task name) 사용하기

태스크를 사용할 때 반드시 풀네임을 사용할 필요없다. 풀네임중에서 고유한 철자만 사용한다면 태스크는 실행된다.

예를 들어 gradle d를 수행해보자.

```
FAILURE: Build failed with an exception.

* What went wrong:
Task 'd' is ambiguous in root project 'commandLine'. Candidates are: 'dependencies', 'dist'.

* Try:
Run gradle tasks to get a list of available tasks. Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output.

BUILD FAILED

Total time: 5.668 secs
```

Task 'd' is ambiguous in root project 'commandLine'. Candidates are: 'dependencies', 'dist'. 에러 내용에 주목한다.
이 에러는 d는 불명확하기 때문에 dependencies인지 dist인지 알 수 없어서 발생한것이다.
대신에 gradle di를 실행해본다.

```
:compile
compiling source
```

```
:compileTest
compiling unit tests
:test
running unit tests
:dist
building the distribution

BUILD SUCCESSFUL

Total time: 4.838 secs
```

dist 태스크가 실행되는 것을 확인할 수 있다.

혹은 카멜케이스 표기법의 앞글자만 따서 실행할 수 있다. 예를 들면 compileTest는 cT로 약어로 쓴다. gradle cT 를 실행해보자.

```
:compile
compiling source
:compileTest
compiling unit tests

BUILD SUCCESSFUL

Total time: 3.782 secs
```

-x 옵션을 사용하여 태스크를 제외할 때도 약어가 가능하다.

다른 build 파일 선택해서 빌드하기

gradle 명령어를 실행하면 현재 디렉토리에서 build.gradle 파일을 찾게 된다. -b 옵션을 주면 다른 빌드 파일을 선택할 수 있다.

먼저 subdir/myproject.gradle 파일을 만들고 다음과 같이 수정한다.

```
task hello << {
    println "using build file '$buildFile.name' in '$buildFile.parentFile.name'."
}
```

그리고 아래처럼 -b옵션과 함께 빌드파일을 선택해서 실행한다.

```
gradle -q -b subdir/myproject.gradle hello
```

```
using build file 'myproject.gradle' in 'subdir'.
```

myproject.gradle 파일을 읽어서 hello 태스크가 동작하는 것을 확인할 수 있다.

프로젝트 디렉토리만 따로 선택하는 방법이 있다. 이럴 땐 -p 옵션을 사용한다.
방금 만들었던 myproject.gradle 파일을 build.gradle로 변경하고 다음 명령어를 실행한다.

```
gradle -q -p subdir hello
```

```
using build file 'build.gradle' in 'subdir'.
```

빌드 중에 에러가 발생하는 경우

기본적으로 빌드 중에 어떤 태스크에서 오류가 발생한다면 빌드는 즉시 중단된다. 이것은 빌드를 빨리 완료시킴으로서 발생할 수 있는 에러를 미연에 방지한다.
그대신 —continue 옵션을 사용하면 빌드 중에 오류가 나더라도 의존성없는 태스크들은 모두 수행하게 된다.

다음과 같은 태스크 4개가 있다고 하자. comple3에서는 에러가 나도록 한다.

```
task compile1 << {
    println 'start compile1'
}
task compile2 << {
    println 'start compile2'
}
task compile3 << {
    throw new Exception("fail")
}
task compile4 << {
    println 'start compile4'
}
```

gradle -q compile1 compile2 compile3 compile4 명령어를 실행해보면 다음과 같이 compile2에서 수행이 중단된다.

start compile1
start compile2

FAILURE: Build failed with an exception.

대신 gradle -q --continue compile1 compile2 compile3 compile4 를 실행해보면 compile3에서 실패가 나더라도 compile4를 실행하게 된다.

start compile1
start compile2
start compile4

FAILURE: Build failed with an exception.

빌드 정보 얻기

Gradle은 빌드의 자세한 정보를 보여주기 위해 여러가지의 내장(built-in)된 태스크를 제공한다. 이것은 빌드의 구조나 의존성 정보를 파악하거나 에러를 디버깅하는데 유용하다.

예제 테스트를 위해 github에 튜터리얼 소스를 올려놓았습니다. 다운로드 받으려면 아래처럼 사용하세요.
git clone https://github.com/sjune/gradle-tutorial.git
cd gradle-tutorial/projectReports/

1.프로젝트 목록

gradle projects 명령어는 해당 프로젝트의 서브 프로젝트의 목록을 계층형으로 보여준다..

```
i. ------------------------------
Root project
i. ------------------------------
Root project 'projectReports'
+--- Project ':api' - The shared API for the application
₩--- Project ':webapp' - The Web application implementation

To see a list of the tasks of a project, run gradle :tasks
For example, try running gradle :api:tasks
```

projectReports 프로젝트 아래 api 와 webapp 라는 서브 프로젝트가 등록된 것을 확인할 수 있다. 만약 서브 프로젝트가 없으면 출력되지 않는다.

프로젝트의 세부 정보를 기술하려면 description 를 사용한다.

```
1 description = 'The shared API for the application'
```

2.태스크 목록

빌드에 사용 가능한 태스크의 목록을 조회하려면 gradle -q tasks 명령어를 사용한다.

```
i. ------------------------
All tasks runnable from root project
i. ------------------------
Default tasks: dists

Build tasks
i. -------
clean - Deletes the build directory (build)dists - Builds the distribution
libs - Builds the JAR

Build Setup tasks
i. -------------
init - Initializes a new Gradle build. [incubating]wrapper - Generates Gradle wrapper files. [incubating]

Help tasks
i. ------
dependencies - Displays all dependencies declared in root project 'projectReports'.dependencyInsight - Displays the insight into a specific dependency in root project 'projectReports'.
help - Displays a help message
projects - Displays the sub-projects of root project 'projectReports'.
properties - Displays the properties of root project 'projectReports'.
tasks - Displays the tasks runnable from root project 'projectReports' (some of the displayed tasks may belong to subprojects).

To see all tasks and more detail, run with --all.
```

기본적으로 이 리포트는 태스크 그룹에 할당되어있는 태스크들을 보여준다. dists 태스크는 build에 속해 있는데 아래처럼 group 속성을 이용하면 가능하다.

```
1 dists {
2     description = 'Builds the distribution'
3     group = 'build'
4 }
```

태스크에 대해 더 많은 정보를 얻으려면 gradle tasks -all 을 사용한다.

3.태스크의 상세 설명 조회

gradle help —task [태스크] 는 해당 태스크의 자세한 설명을 조회해준다. 다음 예는 libs 태스크의 full 태스크 경로, 태스크 타입, description 을 읽어서 보여준다.

gradle -q help --task libs 를 실행해본다.

Detailed task information for libs

Paths
:api:libs
:webapp:libs

Type
Task (org.gradle.api.Task)

Description
Builds the JAR

4.프로젝트의 의존성 정보 목록

dependencies 옵션을 통해서 의존성 정보를 리포트할 수 있다.
gradle -q dependencies api:dependencies webapp:dependencies 를 이용하면 root, api, webapp 각각의 프로젝트의 의존성 정보를 모두 출력해준다.

전체 정보에서 특정 설정 정보만 필터링해서 보고 싶다면 —configuration 옵션을 추가해서 사용하면 된다.
gradle -q api:dependencies --configuration testCompile 를 실행해본다.

i. ----------

Project :api - The shared API for the application

i. ----------

testCompile
₩--- junit:junit:4.11
₩--- org.hamcrest:hamcrest-core:1.3

5.특정 의존성 분석 정보 얻기

gradle dependencyInsight 을 실행하면 특정 의존성의 분석 정보를 얻을 수 있다.
gradle -q webapp:dependencyInsight --dependency groovy --configuration compile 를 실행해본다.

org.codehaus.groovy:groovy-all:2.2.0
₩--- projectReports:api:1.0-SNAPSHOT
₩--- compile

webapp 프로젝트의 compile 설정이 api 프로젝트의에 의해 groovy 라이브러리가 종속되고 있다는 것을 계층적으로 보여주고 있다.
이처럼 dependencyInsight 태스크는 의존성의 의존관계가 어떻게 해결되었는지 분석할 때 매우 편리하다. 의존관계가 어디서 와서 왜 해당 버전이 선택되어 있는지 확인할 수 있기 때문이다.

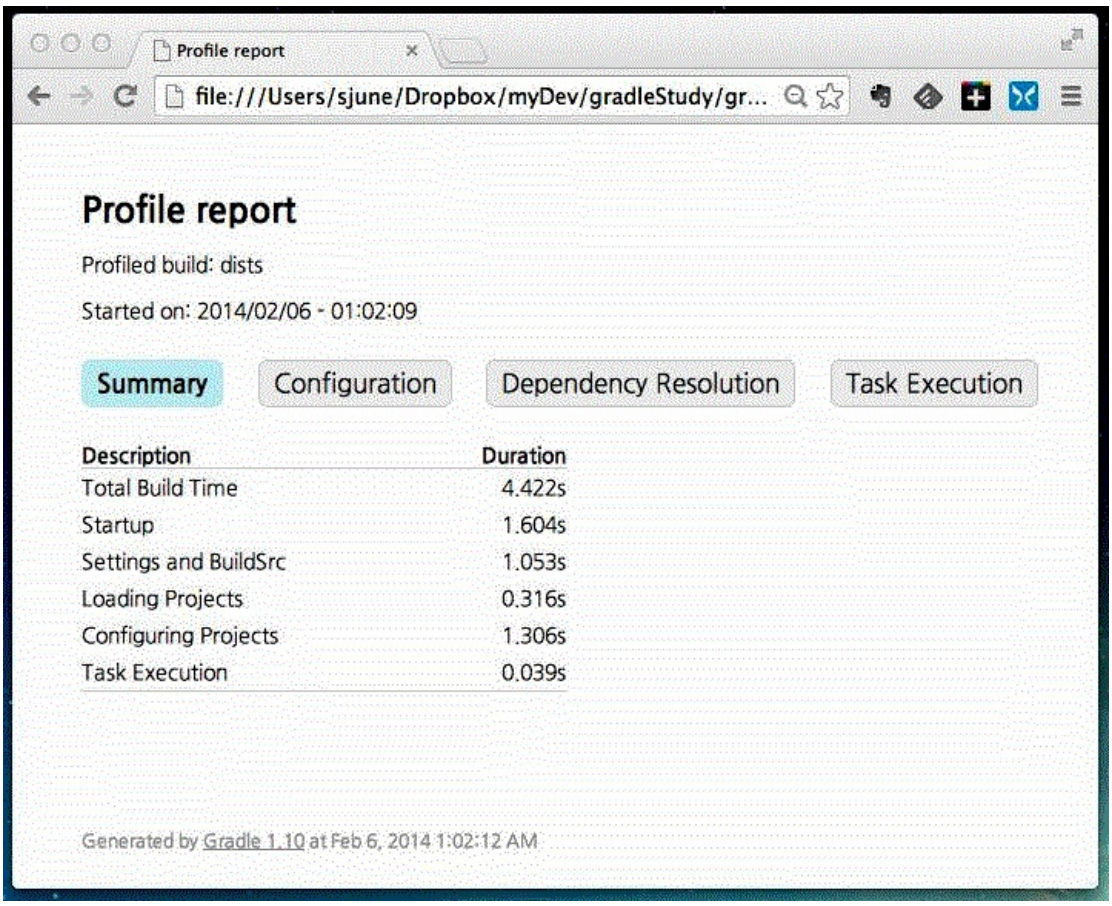
6. 프로젝트의 properties 목록

gradle properties 는 선택된 프로젝트의 프로퍼티 목록을 보여준다.
gradle -q api:properties 를 실행해본다.

```
                i. ----------------------
Project :api - The shared API for the application
                i. ----------------------
allprojects: [project ':api']
ant: org.gradle.api.internal.project.DefaultAntBuilder@12345
antBuilderFactory: org.gradle.api.internal.project.DefaultAntBuilderFactory@12345
artifacts: org.gradle.api.internal.artifacts.dsl.DefaultArtifactHandler@12345
...
```

7. 빌드 프로파일링

gradle —profile 를 수행하면 빌드하는 데 유용한 몇가지 정보를 build/reports/profile 디렉토리에 출력한다.
이 리포트에는 빌드 시간에 대한 개요 외에도 설정 단계 및 태스크 실행 시간 등이 표시된다.



참고

http://www.gradle.org/docs/current/userguide/tutorial_gradle_command_line.html

---

링크 목록

- http://www.gradle.org/docs/current/userguide/tutorial_gradle_command_line.html - http://www.gradle.org/docs/current/userguide/tutorial_gradle_command_line.html

관련 키워드

Gradle
