Gradle: Multi Project Builds

Root Project 경로
- Root Project는 이름으로 나타내지 않고 : 으로 나타낸다.

Sub Project 경로
- Root Project - Sub Project1 - Sub Project2
  ⏎ : Sub Project1 : Sub Project2

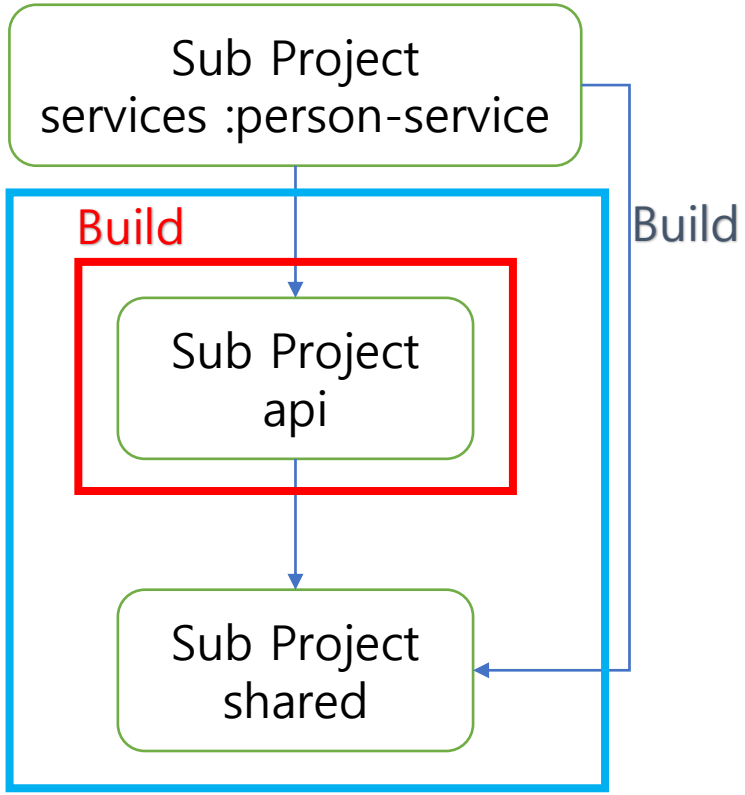
```
Root project 'multiproject'
+--- Project ':api'
+--- Project ':services'
|    +--- Project ':services:shared'
|    \--- Project ':services:webservice'
\--- Project ':shared'
```

Executing Tasks by name

Sub Project의 Gradle Task만 확인하고 싶은 경우
- gradle <project-path>:tasks
- Ex) gradle :api:tasks

- Root Dir에서 `gradle test` 실행
  → 모든 Sub Project의 test task가 실행된다.
- Services project dir에서 `gradle test` 실행
  → services:shared & services:webservice에서 test 실행
- `gradle :services:build` 실행
  → `services project` 에서만 `build` 실행

Gradle: Multi Project Builds




```
$ gradle :api:build

> Task :shared:compileJava
> Task :shared:processResources
> Task :shared:classes
> Task :shared:jar
> Task :api:compileJava
> Task :api:processResources
> Task :api:classes
> Task :api:jar
> Task :api:assemble
> Task :api:compileTestJava
> Task :api:processTestResources
> Task :api:testClasses
> Task :api:test
> Task :api:check
> Task :api:build

BUILD SUCCESSFUL in 0s
```

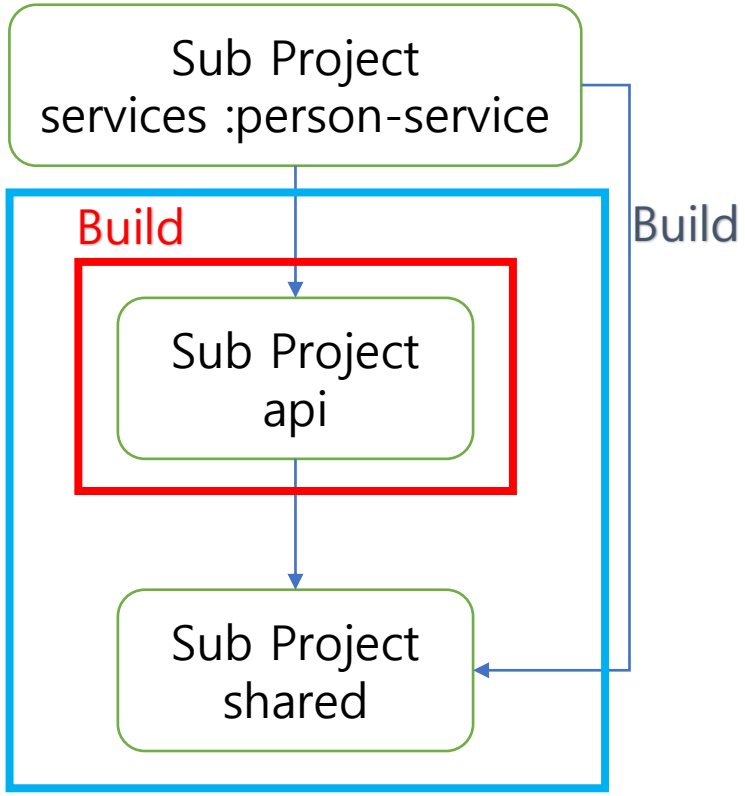
Build task

- Api project를 build task 실행
  → shared project 까지 build 실행

생각해보면, 당연하다.
api 프로젝트가 shared 프로젝트를 의존한다는 말의 의미는
api 프로젝트가 shared 프로젝트를 사용하고 있다는 말이다.

따라서, api 프로젝트를 build 하기 위해서는 먼저 shared 프로젝트가 build되어야 한다.

Gradle: Multi Project Builds

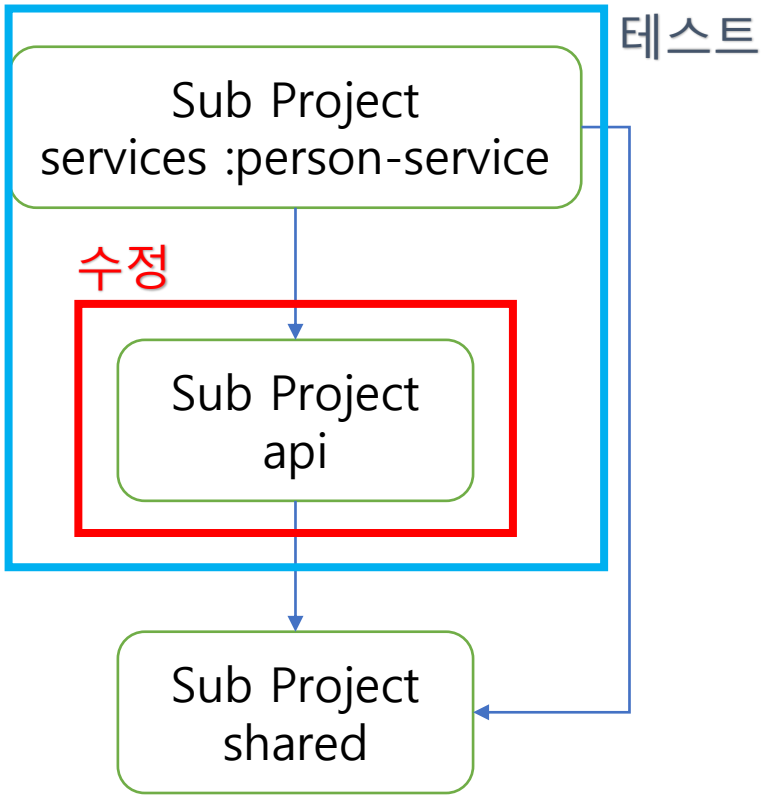



```
$ gradle :api:buildNeeded

> Task :shared:compileJava
> Task :shared:processResources
> Task :shared:classes
> Task :shared:jar
> Task :api:compileJava
> Task :api:processResources
> Task :api:classes
> Task :api:jar
> Task :api:assemble
> Task :api:compileTestJava
> Task :api:processTestResources
> Task :api:testClasses
> Task :api:test
> Task :api:check
> Task :api:build
> Task :shared:assemble
> Task :shared:compileTestJava
> Task :shared:processTestResources
> Task :shared:testClasses
> Task :shared:test
> Task :shared:check
> Task :shared:build
> Task :shared:buildNeeded
> Task :api:buildNeeded

BUILD SUCCESSFUL in 0s
```

The buildNeeded task

- builds AND tests all the projects
  from the project dependencies of the testRuntime configuration:

Gradle: Multi Project Builds



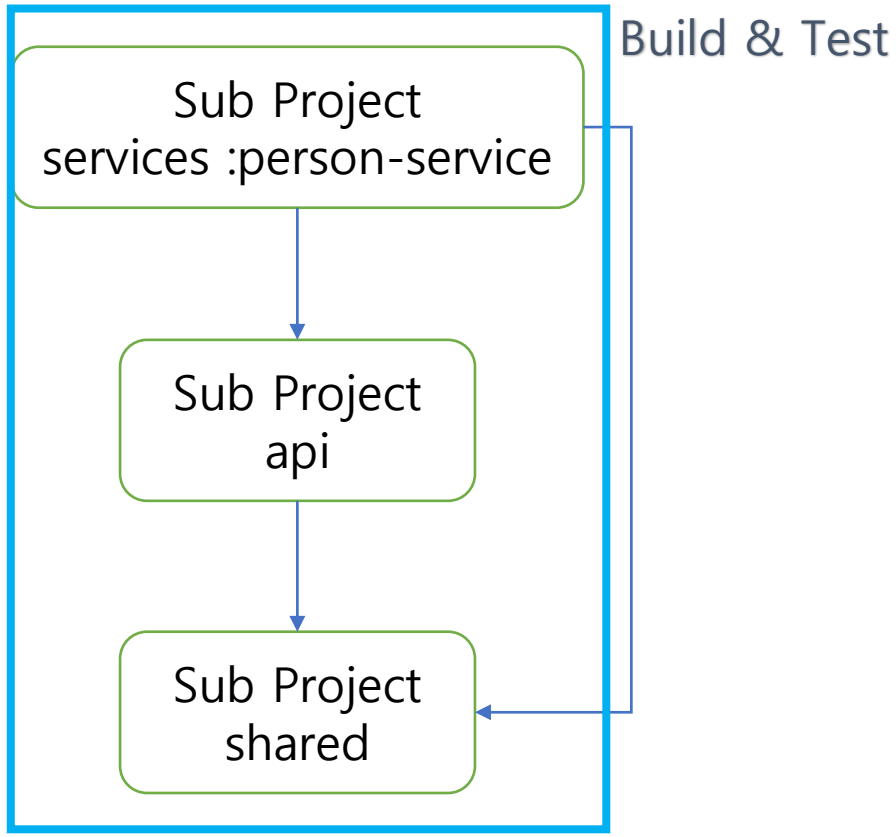

```
$ gradle :api:buildDependents

> Task :shared:compileJava
> Task :shared:processResources
> Task :shared:classes
> Task :shared:jar
> Task :api:compileJava
> Task :api:processResources
> Task :api:classes
> Task :api:jar
> Task :api:assemble
> Task :api:compileTestJava
> Task :api:processTestResources
> Task :api:testClasses
> Task :api:test
> Task :api:check
> Task :api:build
> Task :services:person-service:compileJava
> Task :services:person-service:processResources
> Task :services:person-service:classes
> Task :services:person-service:jar
> Task :services:person-service:assemble
> Task :services:person-service:compileTestJava
> Task :services:person-service:processTestResources
> Task :services:person-service:testClasses
> Task :services:person-service:test
> Task :services:person-service:check
> Task :services:person-service:build
> Task :services:person-service:buildDependents
> Task :api:buildDependents
```

The buildDependents task

- tests ALL the projects that have a project dependency (in the testRuntime configuration) on the specified project:

만약 api 프로젝트를 수정 후 Test를 한다면, 당연히 api 프로젝트를 의존하고 있는 person-service 프로젝트 까지 Test를 해주어야 한다.

Gradle: Multi Project Builds



`Root Project`에서 `gradle build` 을 수행

- build and test ALL projects.
- Gradle guarantees that these tasks will execute in order of their dependencies.