[Initializing the Gradle project](https://guides.gradle.org/building-spring-boot-2-projects-with-gradle/" \l "initializing_the_gradle_project)

First we need to initialize the Gradle project. For that we use Gradle’s init task which creates a template project with an empty build file. The generated project includes the Gradle wrapper out of the box such that you can easily share the project with users that do not have Gradle locally installed. It also adds the default source directories, test dependencies and JCenter as default dependency repository. Please have a look at its [documentation](https://docs.gradle.org/current/userguide/build_init_plugin.html) to read more about the init task.

First we need to create the sample project folder in our home directory and initialize the project:

$ mkdir ~/gradle-spring-boot-project

$ cd ~/gradle-spring-boot-project

$ gradle init --type java-application

The generated project has the following structure:

gradle-spring-boot-project

├── build.gradle

├── gradle

│ └── wrapper

│ ├── gradle-wrapper.jar

│ └── gradle-wrapper.properties

├── gradlew

├── gradlew.bat

├── settings.gradle

└── src

├── main

│ └── java

│ └── App.java

└── test

└── java

└── AppTest.java

C:\Users\Daniel\workspace\sharp>**gradle -q dependencies**

**------------------------------------------------------------**

**Root project**

**------------------------------------------------------------**

**annotationProcessor - Annotation processors and their dependencies for source set 'main'.**

**No dependencies**

**apiElements - API elements for main. (n)**

**No dependencies**

**archives - Configuration for archive artifacts.**

**No dependencies**

**bootArchives - Configuration for Spring Boot archive artifacts.**

**No dependencies**

**compile - Dependencies for source set 'main' (deprecated, use 'implementation' instead).**

**No dependencies**

**compileClasspath - Compile classpath for source set 'main'.**

**\--- com.google.guava:guava:27.0.1-jre**

**+--- com.google.guava:failureaccess:1.0.1**

**+--- com.google.guava:listenablefuture:9999.0-empty-to-avoid-conflict-with-guava**

**+--- com.google.code.findbugs:jsr305:3.0.2**

**+--- org.checkerframework:checker-qual:2.5.2**

**+--- com.google.errorprone:error\_prone\_annotations:2.2.0**

**+--- com.google.j2objc:j2objc-annotations:1.1**

**\--- org.codehaus.mojo:animal-sniffer-annotations:1.17**

**compileOnly - Compile only dependencies for source set 'main'.**

**No dependencies**

**default - Configuration for default artifacts.**

**\--- com.google.guava:guava:27.0.1-jre**

**+--- com.google.guava:failureaccess:1.0.1**

**+--- com.google.guava:listenablefuture:9999.0-empty-to-avoid-conflict-with-guava**

**+--- com.google.code.findbugs:jsr305:3.0.2**

**+--- org.checkerframework:checker-qual:2.5.2**

**+--- com.google.errorprone:error\_prone\_annotations:2.2.0**

**+--- com.google.j2objc:j2objc-annotations:1.1**

**\--- org.codehaus.mojo:animal-sniffer-annotations:1.17**

**implementation - Implementation only dependencies for source set 'main'. (n)**

**\--- com.google.guava:guava:27.0.1-jre (n)**

**runtime - Runtime dependencies for source set 'main' (deprecated, use 'runtimeOnly' instead).**

**No dependencies**

**runtimeClasspath - Runtime classpath of source set 'main'.**

**\--- com.google.guava:guava:27.0.1-jre**

**+--- com.google.guava:failureaccess:1.0.1**

**+--- com.google.guava:listenablefuture:9999.0-empty-to-avoid-conflict-with-guava**

**+--- com.google.code.findbugs:jsr305:3.0.2**

**+--- org.checkerframework:checker-qual:2.5.2**

**+--- com.google.errorprone:error\_prone\_annotations:2.2.0**

**+--- com.google.j2objc:j2objc-annotations:1.1**

**\--- org.codehaus.mojo:animal-sniffer-annotations:1.17**

**runtimeElements - Elements of runtime for main. (n)**

**No dependencies**

**runtimeOnly - Runtime only dependencies for source set 'main'. (n)**

**No dependencies**

**testAnnotationProcessor - Annotation processors and their dependencies for source set 'test'.**

**No dependencies**

**testCompile - Dependencies for source set 'test' (deprecated, use 'testImplementation' instead).**

**No dependencies**

**testCompileClasspath - Compile classpath for source set 'test'.**

**+--- com.google.guava:guava:27.0.1-jre**

**| +--- com.google.guava:failureaccess:1.0.1**

**| +--- com.google.guava:listenablefuture:9999.0-empty-to-avoid-conflict-with-guava**

**| +--- com.google.code.findbugs:jsr305:3.0.2**

**| +--- org.checkerframework:checker-qual:2.5.2**

**| +--- com.google.errorprone:error\_prone\_annotations:2.2.0**

**| +--- com.google.j2objc:j2objc-annotations:1.1**

**| \--- org.codehaus.mojo:animal-sniffer-annotations:1.17**

**\--- junit:junit:4.12**

**\--- org.hamcrest:hamcrest-core:1.3**

**testCompileOnly - Compile only dependencies for source set 'test'.**

**No dependencies**

**testImplementation - Implementation only dependencies for source set 'test'. (n)**

**\--- junit:junit:4.12 (n)**

**testRuntime - Runtime dependencies for source set 'test' (deprecated, use 'testRuntimeOnly' instead).**

**No dependencies**

**testRuntimeClasspath - Runtime classpath of source set 'test'.**

**+--- com.google.guava:guava:27.0.1-jre**

**| +--- com.google.guava:failureaccess:1.0.1**

**| +--- com.google.guava:listenablefuture:9999.0-empty-to-avoid-conflict-with-guava**

**| +--- com.google.code.findbugs:jsr305:3.0.2**

**| +--- org.checkerframework:checker-qual:2.5.2**

**| +--- com.google.errorprone:error\_prone\_annotations:2.2.0**

**| +--- com.google.j2objc:j2objc-annotations:1.1**

**| \--- org.codehaus.mojo:animal-sniffer-annotations:1.17**

**\--- junit:junit:4.12**

**\--- org.hamcrest:hamcrest-core:1.3**

**testRuntimeOnly - Runtime only dependencies for source set 'test'. (n)**

**No dependencies**

**(n) - Not resolved (configuration is not meant to be resolved)**

**================================================**

Just replace:

* compile with implementation (if you don't need transitivity) or api (if you need transitivity)
* testCompile with testImplementation
* debugCompile with debugImplementation
* androidTestCompile with androidTestImplementation
* compileOnly is still valid. It was added in 3.0 to replace provided and not compile. (providedintroduced when Gradle didn't have a configuration name for that use-case and named it after Maven's provided scope.)

It is one of the breaking changes coming with Gradle 3.0 that Google [announced at IO17](https://youtu.be/7ll-rkLCtyk?t=22m20s).

The compile configuration is [now deprecated](https://youtu.be/7ll-rkLCtyk?t=29m18s) and should be replaced by implementation or api

From the [Gradle documentation](https://docs.gradle.org/current/userguide/java_library_plugin.html" \l "sec:java_library_separation):

dependencies {

api 'commons-httpclient:commons-httpclient:3.1'

implementation 'org.apache.commons:commons-lang3:3.5'

}

Dependencies appearing in the api configurations will be transitively exposed to consumers of the library, and as such will appear on the compile classpath of consumers.

Dependencies found in the implementation configuration will, on the other hand, not be exposed to consumers, and therefore not leak into the consumers' compile classpath. This comes with several benefits:

* dependencies do not leak into the compile classpath of consumers anymore, so you will never accidentally depend on a transitive dependency
* faster compilation thanks to reduced classpath size
* less recompilations when implementation dependencies change: consumers would not need to be recompiled
* cleaner publishing: when used in conjunction with the new maven-publish plugin, Java libraries produce POM files that distinguish exactly between what is required to compile against the library and what is required to use the library at runtime (in other words, don't mix what is needed to compile the library itself and what is needed to compile against the library).

**The compile configuration still exists, but should not be used as it will not offer the guarantees that the api and implementation configurations provide.**

**Note:** if you are only using a library in your app module -the common case- you won't notice any difference.  
you will only see the difference if you have a complex project with modules depending on each other, or you are creating a library.