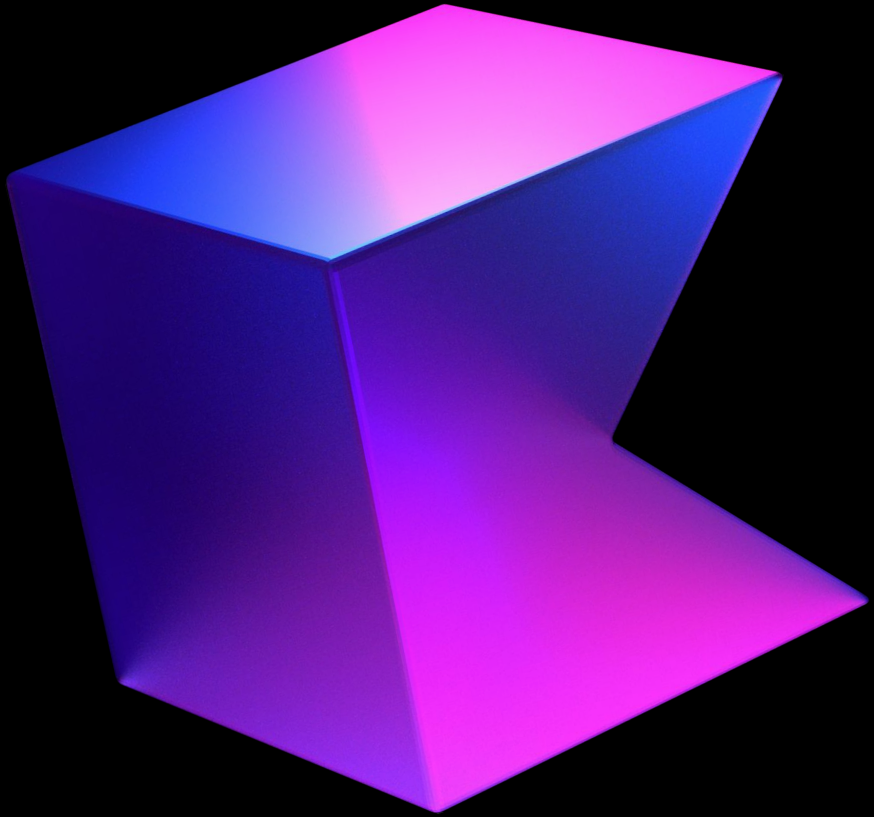
JET BRAINS Kotlin

Build Systems



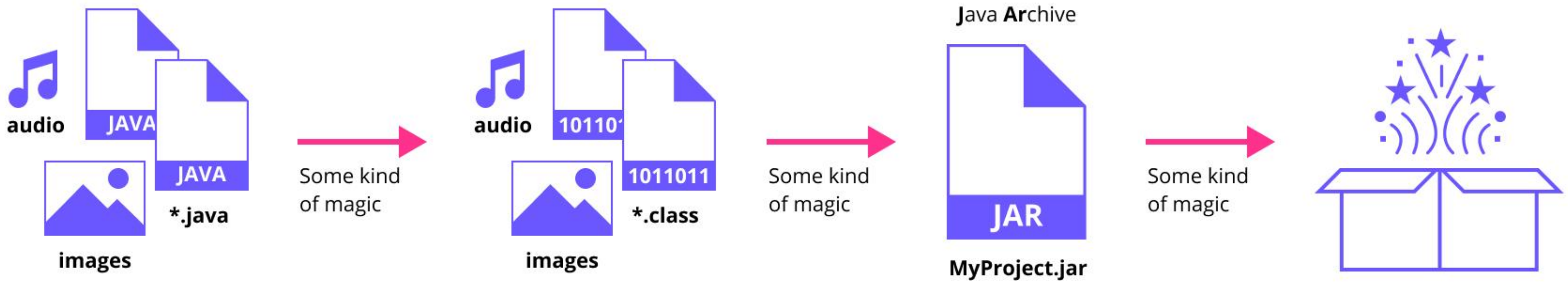
@kotlin | Developed by JetBrains

What? Why?

Build system – Software that automates the process of getting some kind of an artifact (executable, library) from the source code. Build systems can be used for:

- Configuring your build once and using it forever ~~(copy-paste into new projects)~~
- Unifying builds and reusing logic in various projects
- Dependencies management*
- Testing and verification
- Incremental builds*

How?


audio
JAVA
JAVA
*.java
images
Some kind of magic
audio
101101
1011011
*.class
images
Some kind of magic
Java Archive
JAR
MyProject.jar
Some kind of magic

Maven

pom.xml

Project **O**bject **M**odel

Declarative: You define the configuration without specifying how to achieve it.

Convention: You describe what you need with specific rules.

Lifecycle: It can support everything from compilation to tests and so on.

Plugins allow you to do the unconventional heavy-lifting.

Coordinates are located in pom.xml: *groupId*, *artifactId*, *version*.

Repositories: You can load (and cache) the dependencies on demand.

Learn more: *search.maven.org* (Maven Central)

pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>com.mycompany.app</groupId>
        <artifactId>my-app</artifactId>
        <version>1.0-SNAPSHOT</version>

        <properties>
                <maven.compiler.source>1.7</maven.compiler.source>
                <maven.compiler.target>1.7</maven.compiler.target>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.12</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
</project>
```

Gradle

build.gradle

settings.gradle

DSL: It uses Kotlin or Groovy instead of XML.

Tasks: You can define actions which might depend on each other and be quite complex.

Plugins provide unconventional predefined tasks to do the heavy-lifting.

Modules have independent compilation units. Each unit is built into a separate JAR (or some other kind of artifact).

Repositories: You can reuse Maven repositories.

Dependency Management: You can easily declare and resolve dependencies.

Language Agnostic: Gradle can be used for Kotlin, Java, Scala, C++, JS, and COBOL.

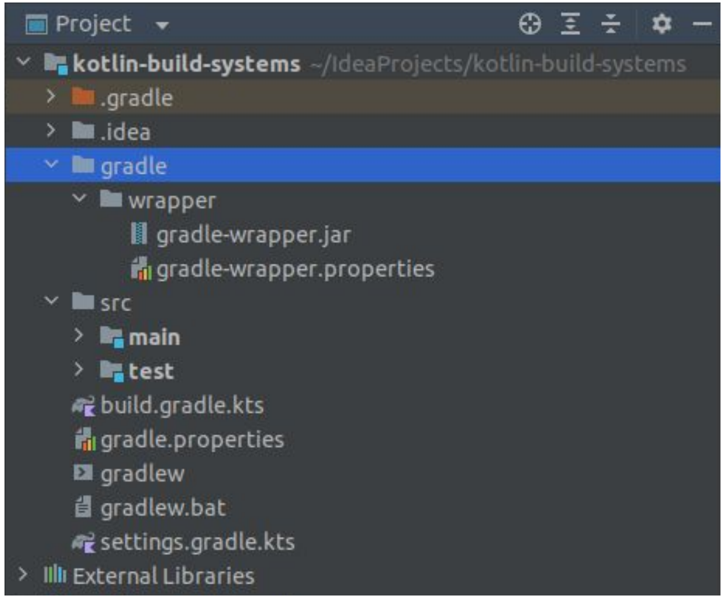
Learn more: *docs.gradle.org*

Gradle project structure

```
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── src
├── build.gradle.kts / build.gradle
├── gradle.properties
├── gradlew
├── gradlew.bat
└── settings.gradle.kts / settings.gradle
```

- Gradle root project == IntelliJ IDEA project
- Gradle project != IntelliJ IDEA project
- Gradle module != IntelliJ IDEA module
- Gradle project ~ IntelliJ IDEA module
- Gradle root project might have subprojects that have subprojects and so on
- Tasks may be defined in any project

Don't push to GitHub



```
Project
kotlin-build-systems ~/IdeaProjects/kotlin-build-systems
  .gradle
  .idea
  gradle
    wrapper
      gradle-wrapper.jar
      gradle-wrapper.properties
  src
    main
    test
  build.gradle.kts
  gradle.properties
  gradlew
  gradlew.bat
  settings.gradle.kts
External Libraries
```

Gradle DSL

Fill out the **build.gradle** or **build.gradle.kts** file to set up the project.

```
plugins {
    kotlin("jvm") version "1.7.10"
}

repositories {
     mavenCentral()
}

dependencies {
    implementation(kotlin("stdlib"))
}

tasks {
     withType<JavaCompile> {
          targetCompatibility = "11"
     }
}
```

Gradle repositories

Specify where to find the libraries needed by the project. The search is carried out from top to bottom

```
repositories {
      mavenCentral()
      google()
      maven {
            url = uri("https://your.company.com/maven")
            credentials {
                  username = "admin"
                  password = "12345"
            }
      }
      flatDir {
            dirs("libraries")
      }
}
```

Don’t push the credentials to GitHub, please!
Use secrets, environmental variables, etc.

Gradle dependencies

- `compilationOnly` – Used only during compilation
- `runtimeOnly` – Used only during runtime
- `implementation` – Used in both
- `api` – Dependency “leaks”, meaning you can access its dependencies

- `testCompilationOnly`
- `testRuntimeOnly`
- `testImplementation`
- `testApi`

Gradle dependencies

```kotlin
val ktorVersion: String = "6.6.6"

dependencies {
        // string notation, e.g. group:name:version
        implementation("commons-lang:commons-lang:2.6")
        implementation("io.ktor:ktor-serialization-jackson:$ktorVersion")
        // map notation:
        implementation("org.jetbrains.kotlinx", "kotlinx-datetime", "7.7.7")
        // dependency on another project
        implementation(project(":neighborProject"))
        // putting all jars from 'libs' onto the compile classpath
        implementation(fileTree("libs"))
        // api dependency - internals are accessible
        api("io.ktor:ktor-server-content-negotiation:$ktorVersion")
        // test dependencies
        testImplementation("org.jetbrains.kotlin:kotlin-test-junit")
        testImplementation(kotlin("test"))
}
```

Gradle dependencies

```
dependencies {
      implementation("org.hibernate:hibernate") {
         version {
             // If there is a version conflict, strictly select version "3.1" of hibernate
             strictly("3.1")
         }
         exclude(module = "cglib") // by artifact name
         exclude(group = "org.jmock") // by group
         exclude(group = "org.unwanted", module = "buggyModule") // by both
         // disabling all transitive dependencies of this dependency
         isTransitive = false
      }
}
```

BOM

There are direct and transitive dependencies, which may lead to version conflicts.

```
myProject → thing:1.0 → anotherThing:1.1
myProject → thirdThing:1.0 → anotherThing:1.2
```

Maven’s Bill Of Materials (BOM) offers a solution.

```
val ktorVersion: String = "2.0.0"

dependencies {
    implementation(enforcedPlatform("io.ktor:ktor-bom:$ktorVersion"))
    implementation(enforcedPlatform("io.ktor:ktor-server-core"))
    implementation(enforcedPlatform("io.ktor:ktor-server-netty"))
}
```

Gradle wrapper

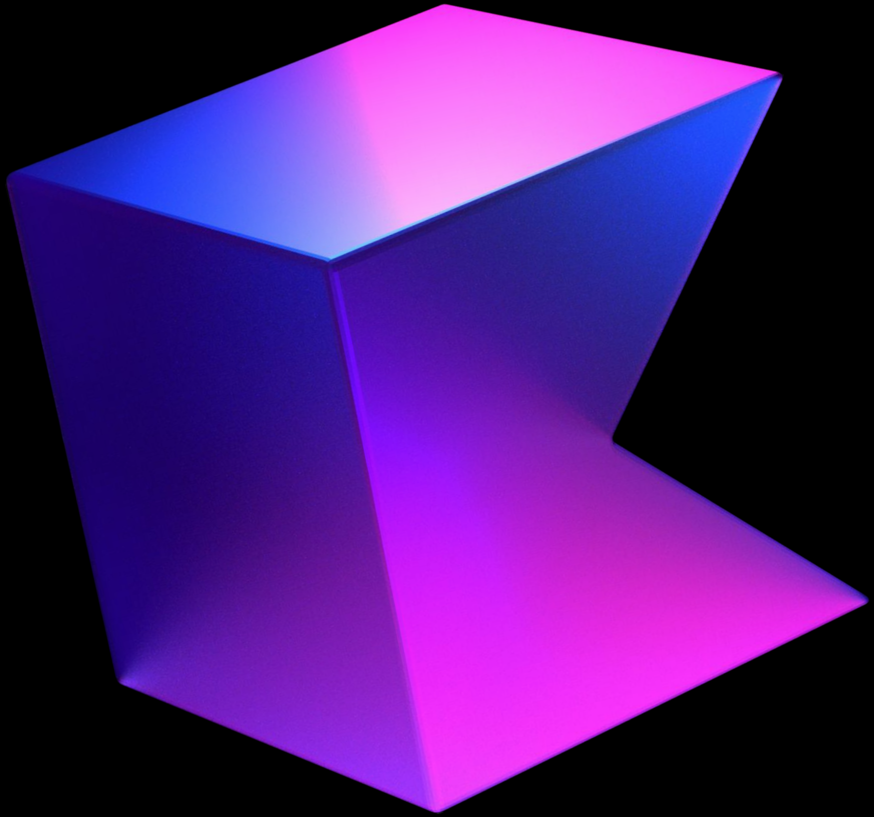
A Gradle wrapper (gradlew) is a shell script that downloads and caches the required version of Gradle.

- `gradlew` – used in *nix
- `gradlew.bat` – used in Windows

The version is specified in `projectRoot/gradle/wrapper/gradle-wrapper.properties`:

```
distributionUrl=https\://services.gradle.org/distributions/gradle-7.5.1-bin.zip
```

Thanks!



@kotlin | Developed by JetBrains