03. Gradle - Comprehensive Notes

---

1. Introduction to Gradle

Gradle is a **build automation tool** used for building, testing, and deploying applications. It is:
✅ **Faster** than Maven due to incremental builds.
✅ **Flexible** because it supports **Groovy** and **Kotlin DSL**.
✅ **Widely used** in **Java, Kotlin, Android, and Spring Boot** projects.

🔷 **Why use Gradle?**

- **Performance:** Uses incremental builds and parallel execution.
- **Flexibility:** Supports both **declarative and imperative** scripting.
- **Better Dependency Management:** Uses a powerful dependency resolution mechanism.
- **Integration with Build Tools:** Works with **Maven**, **Ivy**, and other repositories.

---

2. Gradle Architecture

🛠️ Key Components of Gradle

Component	Description
Build Script	Written in **Groovy (build.gradle)** or **Kotlin (build.gradle.kts)**
Project	Represents a single software component (app, library, service)

Component	Description
Task	A unit of work like compilation, testing, or packaging
Dependency Management	Manages external libraries from **Maven Central, JCenter**
Plugins	Extend Gradle's capabilities (e.g., Java, Spring Boot, Android)
Gradle Daemon	Keeps Gradle running in the background to speed up builds

---

3. Setting Up Gradle

🔷 Installing Gradle

1. **Using SDKMAN** (Recommended)

```
sdk install gradle
```

2. **Using Homebrew (Mac/Linux)**

```
brew install gradle
```

3. **Manually Downloading**
   - Download from [Gradle's official site](#).
   - Extract and set the `GRADLE_HOME` path.

🔷 Verify Installation

```
gradle -v
```

4. Gradle Build Script (Groovy vs Kotlin)

Gradle scripts are written in:

- **Groovy DSL** (`build.gradle`)
- **Kotlin DSL** (`build.gradle.kts`)

📌 Sample Gradle Build Script

Groovy DSL (`build.gradle`)

```
plugins {
    id 'java'
}

group 'com.example'
version '1.0.0'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'junit:junit:4.13.2'
}

tasks.register('hello') {
    doLast {
        println 'Hello, Gradle!'
```

```
    }
}
```

Kotlin DSL (`build.gradle.kts`)

```
plugins {
    java
}

group = "com.example"
version = "1.0.0"

repositories {
    mavenCentral()
}

dependencies {
    implementation("org.springframework.boot:spring-boot-starter-web")
    testImplementation("junit:junit:4.13.2")
}

tasks.register("hello") {
    doLast {
        println("Hello, Gradle!")
    }
}
```

---

5. Gradle Build Lifecycle

Gradle has **three phases**:

1️⃣ **Initialization Phase** – Identifies projects to be built.
2️⃣ **Configuration Phase** – Evaluates the `build.gradle` script.
3️⃣ **Execution Phase** – Runs the tasks.

🔷 Running a Build

```
gradle build
```

This executes the following tasks:

✅ `compileJava` → Compiles Java files
✅ `processResources` → Copies resource files
✅ `test` → Runs unit tests
✅ `jar` → Packages compiled files into a JAR
✅ `assemble` → Creates output artifacts

---

6. Tasks in Gradle

A **task** is a piece of work like **compiling code, running tests, or creating a JAR**.

🔷 Creating Custom Tasks

Groovy

```
task hello {
    doLast {
        println 'Hello, World!'
```

```
    }
}
```

Kotlin

```
tasks.register("hello") {
    doLast {
        println("Hello, World!")
    }
}
```

Run the task:

```
gradle hello
```

🔷 Listing Available Tasks

```
gradle tasks
```

---

7. Dependency Management in Gradle

Gradle manages dependencies using **repositories** like **Maven Central** or **JCenter**.

🔷 Declaring Dependencies

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
```

```
    testImplementation 'junit:junit:4.13.2'
}
```

🔷 Dependency Configurations

Configuration	Usage
`implementation`	Used for normal dependencies
`testImplementation`	Used only in test scope
`compileOnly`	Only for compile-time (not at runtime)
`runtimeOnly`	Only required at runtime

🔷 Viewing Dependencies

```
gradle dependencies
```

---

8. Gradle Plugins

Plugins **extend** Gradle's functionality.

🔷 Applying a Plugin

```
plugins {
    id 'java'
    id 'application'
}
```

🔷 Common Gradle Plugins

Plugin	Usage
`java`	Java project support
`application`	Creates a runnable application
`war`	Builds a WAR file
`spring-boot`	Used for Spring Boot projects

---

9. Building and Running Java Applications

🔷 Building a JAR File

```
gradle jar
```

🔷 Running a Java Application

Add this to `build.gradle`:

```
application {
    mainClass = 'com.example.MainApp'
}
```

Run the application:

```
gradle run
```

10. Advanced Gradle Features

🔷 Multi-Module Projects

Gradle supports **multi-module** builds.

Project Structure

```
/my-project
  ├── /app (submodule)
  ├── /library (submodule)
  ├── build.gradle
  ├── settings.gradle
```

settings.gradle

```
rootProject.name = 'my-project'
include 'app', 'library'
```

🔷 Caching & Incremental Builds

Gradle optimizes builds with caching and parallel execution.

```
gradle build --parallel --scan
```

11. Gradle vs Maven

Feature	Gradle	Maven
Performance	🚀 Faster (incremental builds)	Slower
DSL	Groovy/Kotlin	XML
Flexibility	High	Rigid
Dependency Management	Dynamic versioning	Static versioning

---

12. Gradle Interview Questions & Answers

✅ Q1: What is Gradle?

A: Gradle is a build automation tool used for Java, Kotlin, and Android projects.

✅ Q2: Why is Gradle faster than Maven?

A: Gradle supports **incremental builds**, **task caching**, and **parallel execution**.

✅ Q3: How do you define dependencies in Gradle?

A: Using `dependencies` block:

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter'
}
```

✅ Q4: What is the difference between `implementation` and `compileOnly`?

A: `implementation` is required at **compile and runtime**, while `compileOnly` is only needed at **compile-time**.

---

13. Conclusion

✅ **Gradle is fast, flexible, and widely used in modern Java projects**.
✅ **Supports Groovy & Kotlin DSL**.
✅ **Better than Maven in terms of performance and flexibility**.

Would you like **hands-on practice exercises** or a **Gradle-based project**? 🚀