**Gradle**

[**Gradle**](https://en.wikipedia.org/wiki/Gradle) is an excellent open-source construction tool that is capable of the development of any kind of software.

This tool is capable of developing applications with industry standards and supports a variety of languages including [Groovy](https://www.geeksforgeeks.org/difference-between-groovy-and-java/), [C++](https://www.geeksforgeeks.org/c-plus-plus/), [Java](https://www.geeksforgeeks.org/java/), [Scala](https://www.geeksforgeeks.org/scala-programming-language/), and [C](https://www.geeksforgeeks.org/c/).

Gradle also is capable of controlling the development tasks with compilation and packaging to testing, deployment, and publishing.

**Working of Gradle**

**1. Gradle Projects:**

* The projects created by Gradle are a web application or a [JAR file](https://www.geeksforgeeks.org/jar-files-java/). These projects are a combination of one or more tasks.
* These projects are capable to be deployed on the various development life cycles.
* A Gradle project can be described as building a wall with bricks N in number which can be termed as tasks.

**2. Gradle Tasks:**

* The tasks are the functions which are responsible for a specific role.
* These tasks are responsible for the creating of classes, Javadoc, or publish archives into the repository which makes the whole development of the Gradle project.
* These tasks help Gradle decide what input is to be processed for a specific output.

Tasks can be categorized into two different ways:

* **Default Task**:
  + These are **predefined tasks** provided by Gradle.
  + They run automatically if no custom task is defined.
  + Examples: init, wrapper (used to set up a basic Gradle project).
* **Custom Task**:
  + These are **user-defined tasks** created by developers.
  + Used to perform **specific actions** required in a project.

You can define a custom task in the build.gradle file like this:

Build.gradle : task hello {

doLast {

println 'Hello from custom task!'

}

}

**Structure**

**1. Project Root Directory**

The root directory of a Gradle project contains all configuration files and sub-projects. It includes:

* settings.gradle or settings.gradle.kts
* build.gradle or build.gradle.kts
* gradle folder
* gradlew and gradlew.bat
* src directory

**2. build.gradle (Build Script File)**

* This is the **main configuration file** for Gradle.
* Contains **plugins**, **dependencies**, **repositories**, and **custom tasks**.
* Can be written using **Groovy DSL** (build.gradle) or **Kotlin DSL** (build.gradle.kts).

**Example:**

plugins {

id 'java'

}

repositories {

mavenCentral()

}

dependencies {

implementation 'org.springframework:spring-core:5.3.9'

}

**3. settings.gradle**

* Defines the **project name** and includes **sub-projects** in multi-module builds.
* Important in large applications with multiple modules.

**Example:**

rootProject.name = 'MyProject'

include 'app', 'library'

**4. gradle Directory**

* Contains **Gradle wrapper files**, like wrapper.properties.
* Ensures that everyone uses the **same Gradle version** regardless of their local setup.

**5. Gradle Wrapper Files**

* Files: gradlew, gradlew.bat, and the gradle folder.
* These allow building the project without installing Gradle globally.
* Ensures consistent build environment across systems.

**6. src Directory:** This is the **source folder** where all your code is written.

**Typical Structure:**

src/

├── main/

│ ├── java/ # Java source code

│ └── resources/ # Configuration files

└── test/

├── java/ # Unit test code

└── resources/ # Test config files

**7. Dependencies Block:** Declares the **external libraries** needed for the project.

* Dependency scopes:
  + implementation – Main code dependencies.
  + testImplementation – For testing libraries.
  + compileOnly, runtimeOnly – Other scopes.

**8. Plugins Block:** Plugins add extra functionality like support for Java, Spring Boot, Android, etc.

Syntax:

plugins {

id 'java'

id 'application'

}

**9. Tasks:** Gradle builds are based on **tasks**.

**Two Types:**

* **Default Tasks**: Provided by Gradle like build, clean, assemble.
* **Custom Tasks**: Defined by the user to perform specific actions.

**Example:**

task greet {

doLast {

println 'Hello from Gradle!'

}

}

**10. Build Lifecycle Phases**

Gradle build goes through multiple phases:

1. **Initialization** – Initializes settings and projects.
2. **Configuration** – Evaluates build scripts and tasks.
3. **Execution** – Executes the selected tasks.

**Conclusion**

Gradle build structure is simple yet highly flexible. It organizes the project using key files like build.gradle, settings.gradle, and source folders like src/main/java. Understanding this structure is crucial for managing builds, handling dependencies, and automating tasks efficiently.