Gradle：建置自動化工具

Gradle 是一個強大的建置自動化工具，常用於 Java、Kotlin 和 Android 專案，但也足夠靈活，適用於其他語言。以下是一個初學者友好的指南，幫助你開始使用 Gradle 並有效地使用它：

---

1. 安裝 Gradle

在使用 Gradle 之前，你需要先安裝它：- **Mac/Linux（使用 Homebrew）**：- 在終端中運行 `brew install gradle`。- **Windows（或手動安裝）**：- 從 gradle.org 下載 Gradle 二進制文件。- 解壓縮到一個資料夾（例如 `C:\Gradle` 或 `/opt/gradle`）。- 將 Gradle `bin` 目錄（例如 `C:\Gradle\gradle-8.x\bin`）添加到系統的 PATH 環境變量。- **驗證安裝**：- 打開終端並運行 `gradle -v`。你應該會看到 Gradle 版本和詳細信息。

如果你在使用帶有 **Gradle Wrapper** 的專案（在共享專案中很常見），你不需要全局安裝 Gradle——只需在專案目錄中使用 `./gradlew`（Linux/Mac）或 `gradlew.bat`（Windows）。

---

2. 了解 Gradle 基礎

Gradle 使用 **Groovy** 或 **Kotlin** 腳本（通常是 `build.gradle` 或 `build.gradle.kts`）來定義專案的建置方式。以下是你需要了解的內容：- **專案**：Gradle 建置可以包含一個或多個專案（例如單一應用程式或多模組設置）。- **任務**：這些是 Gradle 執行的動作，例如 `compileJava`、`test` 或 `build`。- **依賴**：Gradle 管理專案所需的庫（例如來自 Maven Central 或 Google 的存儲庫）。

---

3. 創建一個簡單的 Gradle 專案

讓我們設置一個基本的 Java 專案來看看 Gradle 的運作：1. **創建專案資料夾**：- 創建一個目錄（例如 `my-gradle-project`）並在終端中導航到它。2. **初始化 Gradle**：- 運行 `gradle init`。- 按照提示：選擇“application”、“Java”和“Groovy”（或 Kotlin）作為建置腳本。- 這將創建一個基本結構，包含 `build.gradle` 文件和範例代碼。3. **探索生成的** `build.gradle`：“‘groovy plugins { id ‘java’ id ‘application’ }

repositories { mavenCentral() }

dependencies { implementation ‘org.slf4j:slf4j-api:1.7.36’ }

application { mainClass = ‘com.example.App’ // 根據你的包名進行調整 }“-plugins：添加對 `Java` 和運行應用程式的支持。 -repositories：`Gradle` 查找依賴的位置（例如 `Maven Central`）。 -dependencies：專案使用的庫。 -application ‘：指定要運行的主類。

4. **運行任務**：
    - 建置專案：`gradle build`。
    - 運行應用程式：`gradle run`。
    - 列出可用任務：`gradle tasks`。

---

4. 常用 Gradle 命令

以下是你經常使用的命令：- `gradle build`：編譯和打包你的專案。- `gradle clean`：刪除 `build` 目錄以重新開始。- `gradle test`：運行專案的測試。- `gradle dependencies`：顯示依賴樹。- `./gradlew <task>`：使用專案的 Gradle Wrapper 而不是全局安裝。

---

5. 添加依賴

要使用外部庫，編輯 `build.gradle`。例如，添加 Gson（一個 JSON 解析庫）：

```
dependencies {
    implementation 'com.google.code.gson:gson:2.10.1'
}
```

然後運行 `gradle build` 來下載它。

---

6. 自定義你的建置

你可以在 `build.gradle` 中添加自定義任務或調整設置。例如，自定義任務：

```
task hello {
    doLast {
        println 'Hello, Gradle!'
    }
}
```

使用 `gradle hello` 運行它。

---

7. 使用 Gradle Wrapper（對團隊推薦）

Gradle Wrapper 確保每個人都使用相同的 Gradle 版本：- 生成它：`gradle wrapper`。- 使用它：`./gradlew build`（或在 Windows 上使用 `gradlew.bat build`）。

---

8. 成功秘訣

- **文檔**：查看 Gradle 的官方文檔 以深入了解。
- **快取**：Gradle 快取依賴和任務輸出，因此隨著時間的推移，建置會變得更快。
- **插件**：探索插件（例如用於 Spring Boot、Android 或代碼質量）以擴展功能。

範例：建置和運行 Java 應用程式

如果你的 `src/main/java/com/example/App.java` 看起來像這樣：

```
package com.example;
public class App {
    public static void main(String[] args) {
        System.out.println("Hello from Gradle!");
    }
}
```

運行 `gradle run`，你會看到輸出。

如果你需要幫助完成特定的 Gradle 任務、專案設置或故障排除，請告訴我！你正在進行什麼樣的專案？