

# Introduction to **Gradle**



N.L. Hsueh

Feng Chia University

# Gradle 介紹

- Gradle 是一個建構自動化工具
  - 協助自動化完成 編譯、測試、檢查程式碼、打包 等工作
- 使用 Groovy DSL(領域特定語言) 作為設定檔語言
  - 比起 XML, 縮減自 Groovy 語言的 Groovy DSL 有更高的可讀性, 並且可以執行更複雜的工作。
- 需要先安裝 Java
- 類似的工具, 如: ant、maven
  - 使用 XML 語言
- 許多開源軟體或企業使用中
  - ex: Netflix、Google 都將 Gradle 作為使用 Java 語言時的官方建置工具

領域特定語言:

「針對特定應用領域的程式語言」

可以全新設計, 也可以是現有程式語言"瘦身"之後的不同版本。

# Gradle 和 Maven 的差異

	設定檔檔名	設定檔語法	設定檔寫法	建構速度	靈活性
<b>gradle</b>	build.gradle	Groovy DSL	較精簡、易懂	較快	自定義的彈性 更大且便利
<b>maven</b>	pom.xml	XML	較繁瑣	較慢	自定義較複雜

# Groovy DSL 範例

```
build.gradle
1  plugins {
2      id 'java'
3      id 'checkstyle' version '2.12.1'
4  }
5
6  version = '1.0'
7
8  repositories {
9      mavenCentral()
10 }
11
12 dependencies {
13     testImplementation 'junit:junit:4.13.2'
14
15     implementation 'com.google.guava:guava:31.0.1-jre'
16 }
```

# XML 範例

```
pom.xml
6   <version>1.0</version>
7   <plugin>
8     <groupId>org.apache.maven.plugins</groupId>
9     <artifactId>maven-checkstyle-plugin</artifactId>
10    <version>2.12.1</version>
11    <executions>
12      <execution>
13        <configuration>
14          <configLocation>config/checkstyle/checkstyle.xml</configLocation>
15          <consoleOutput>true</consoleOutput>
16          <failsOnError>true</failsOnError>
17        </configuration>
18        <goals>
19          <goal>check</goal>
20        </goals>
21      </execution>
22    </executions>
23  </plugin>
24 </plugin>
25 <dependencies>
26   <dependency>
27     <groupId>junit</groupId>
28     <artifactId>junit</artifactId>
29     <version>4.13.2</version>
30   </dependency>
31   <dependency>
32     <groupId>com.google.guava</groupId>
33     <artifactId>guava</artifactId>
34     <version>31.1-jre</version>
35   </dependency>
36 </dependencies>
37 <build>
38   <plugins>
39     <plugin>
40       <groupId>org.apache.maven.plugins</groupId>
41       <artifactId>maven-compiler-plugin</artifactId>
42       <version>2.3.2</version>
43     </plugin>
44   </plugins>
45 </build>
```

# 安裝 Gradle

# 安裝 Gradle (1/4)

- 確認 Java 環境版本為 1.8 以上

```
$ java -version  
java version "1.8.0_121"
```

- 下載 Gradle (7.5.1 版本) 的二進位檔

- <https://gradle.org/releases/>

📁 v7.5.1

📅 Aug 05, 2022

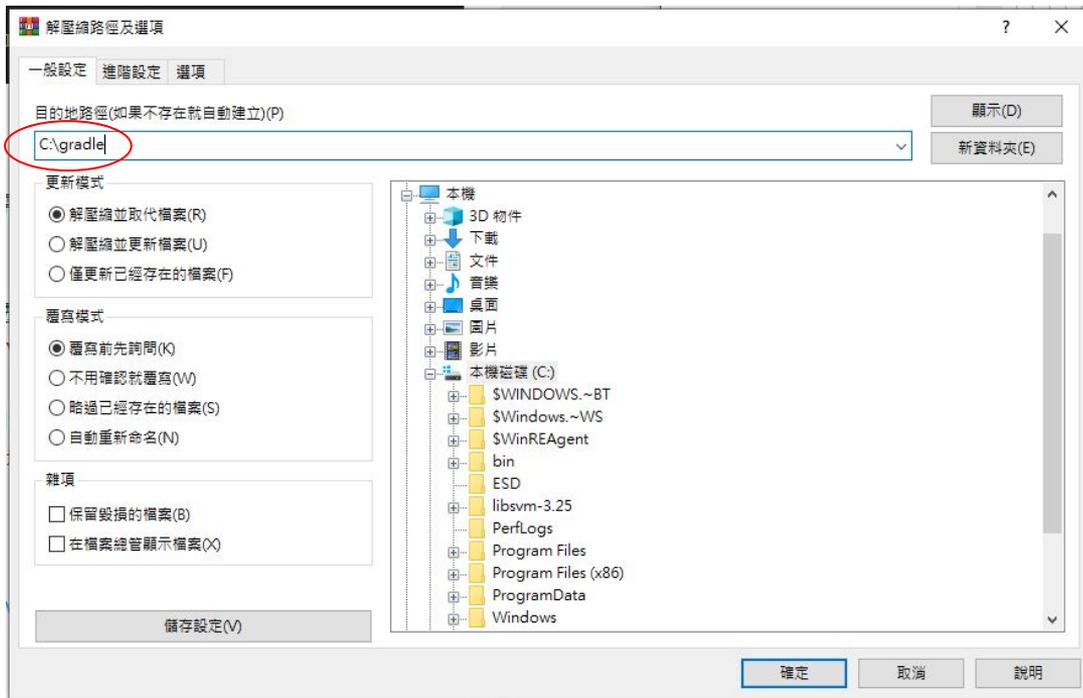
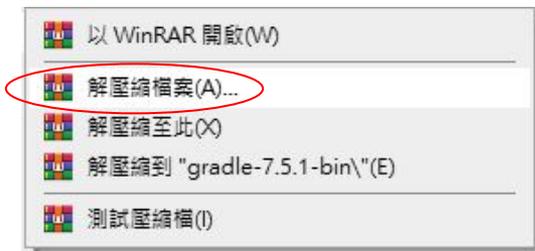
- Download: **binary-only or complete (checksums)**
- User Manual
- API Javadoc
- DSL Reference
- Release Notes

# 安裝 Gradle (2/4)

- 解壓縮 zip 檔案至 C:\gradle



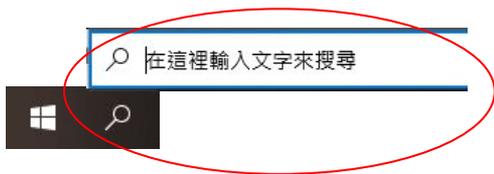
gradle-7.5.1-bin  
.zip



# 安裝 Gradle (3/4)

- 設定環境變數

- 開啟左下角的系統內建搜尋



- 搜尋

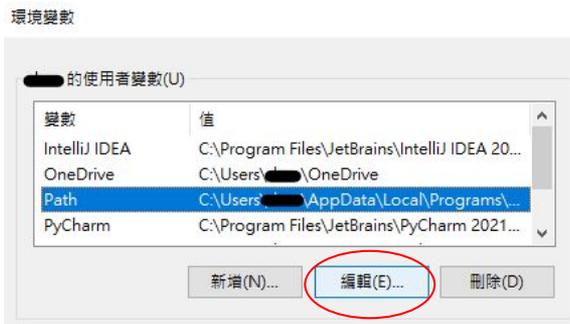


- 點選 **環境變數**，進行修改

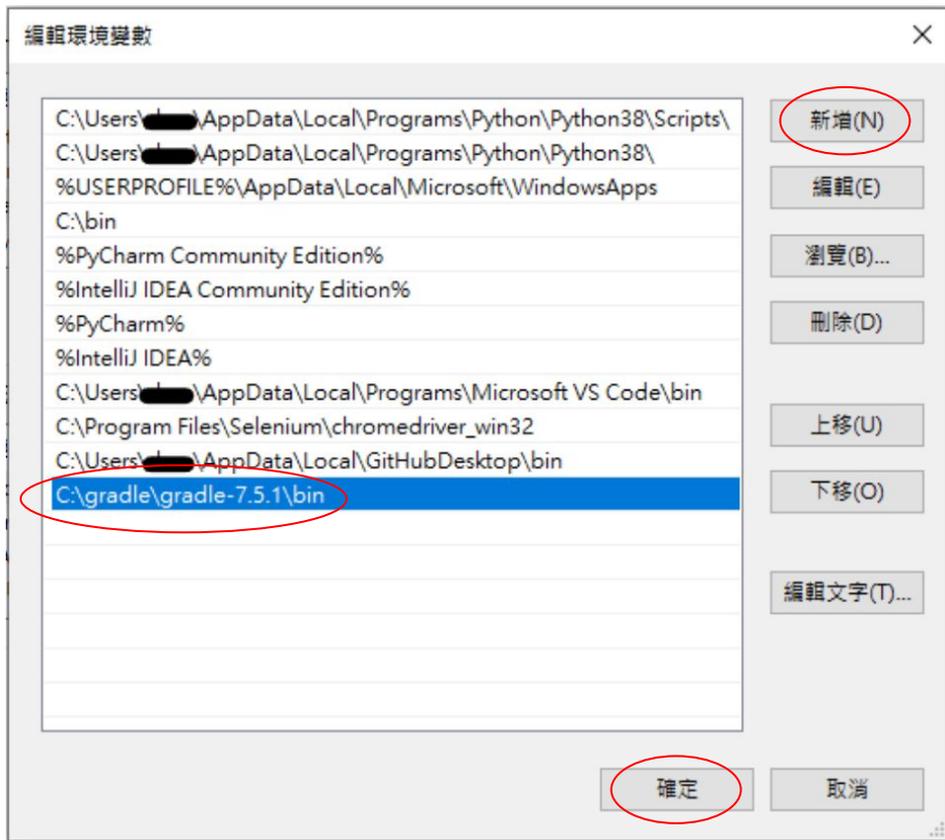


# 安裝 Gradle (4/4)

- 設定環境變數
  - 點選使用者變數的 **Path**, 按下**編輯**



- 點選**新增**  
輸入解壓縮後的資料夾路徑  
加上 \gradle-7.5.1\bin  
(C:\gradle\gradle-7.5.1\bin)



# 確認 gradle 安裝完成

- 開啟系統內建搜尋
- 搜尋並開啟



- 在命令提示字元中  
輸入 **gradle -v**  
查看 gradle 的版本

```
cmd 命令提示字元
Microsoft Windows [版本 10.0.19043.2130]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\>gradle -v

Welcome to Gradle 7.5.1!

Here are the highlights of this release:
- Support for Java 18
- Support for building with Groovy 4
- Much more responsive continuous builds
- Improved diagnostics for dependency resolution

For more details see https://docs.gradle.org/7.5.1/release-notes.html

-----
Gradle 7.5.1
-----

Build time:      2022-08-05 21:17:56 UTC
Revision:        d1daa0cbf1a0103000b71484e1dbfe096e095918

Kotlin:          1.6.21
Groovy:          3.0.10
Ant:             Apache Ant(TM) version 1.10.11 compiled on July 10 2021
JVM:             11.0.13 (Eclipse Adoptium 11.0.13+8)
OS:             Windows 10 10.0 amd64
```

# 設定檔意義

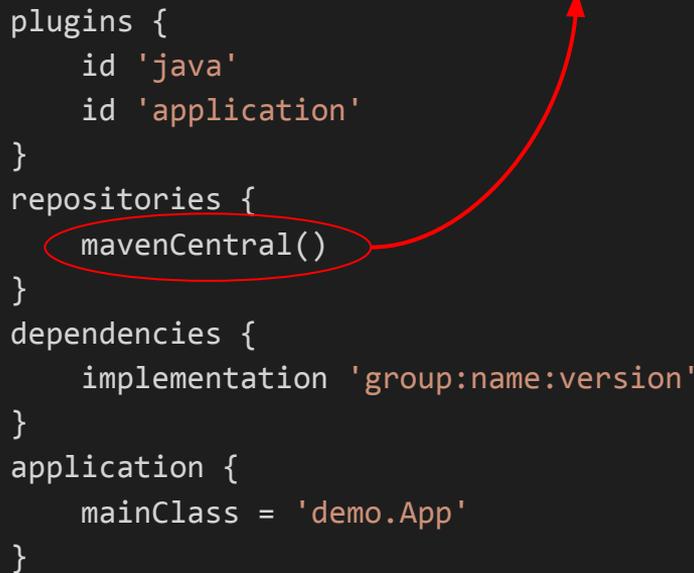
- **build.gradle** 位於專案根目錄, 用以調整建構所需的設定

ex: 定義專案所需的插件、libraries ...

- **plugins**
  - 插件可以增加 Gradle 的功能提供額外的指令控制第三方軟體的行為
  - ex: 可以加入 JaCoCo Plugins 將覆蓋率報告包含在建置流程中
- **repositories**
  - 程式 libraries 的遠端儲存庫
- **dependencies**
  - 設定 libraries 和版本
- **application**
  - 程式客製化設定  
ex: main (程式入口點)

官方定義的 [Maven 儲存庫](#)  
託管開源的 library 供 Java 專案使用

```
plugins {  
    id 'java'  
    id 'application'  
}  
repositories {  
    mavenCentral()  
}  
dependencies {  
    implementation 'group:name:version'  
}  
application {  
    mainClass = 'demo.App'  
}
```



# Gradle 中的任務

- task: 專案建構過程中的任務, 表示一項工作
- 開啟命令提示字元, 並輸入 **gradle tasks**, 查看所有預設的任務種類

```
C:\Users\shan\Desktop\demo>gradle tasks
> Task :tasks

-----
Tasks runnable from root project 'demo'
-----

Application tasks
-----
run - Runs this project as a JVM application
runShadow - Runs this project as a JVM application using the shadow jar
startShadowScripts - Creates OS specific scripts to run the project as a JVM application using the shadow jar

Build tasks
-----
assemble - Assembles the outputs of this project.
build - Assembles and tests this project.
buildDependents - Assembles and tests this project and all projects that depend on it.
buildNeeded - Assembles and tests this project and all projects it depends on.
classes - Assembles main classes.
clean - Deletes the build directory.
jar - Assembles a jar archive containing the main classes.
testClasses - Assembles test classes.
```

# 實際演練

初始化 Java 專案

# 使用 Gradle 初始化 Java 專案 (1/4)

- 新增一個 demo 資料夾
- 在命令提示字元中, 進入該 demo 資料夾 (**cd 資料夾路徑**)

```
C:\Users\1\Desktop>cd Desktop\demo  
C:\Users\1\Desktop\demo>
```

- 輸入 **gradle init**, 並選擇要產生的專案類型 (**2:application**)
  - basic: 空的專案 / application: 有簡單內容的應用程式

```
C:\Users\1\Desktop\demo>gradle init  
Starting a Gradle Daemon, 1 incompatible and 1 stopped Daemons could not be reused, use --status for details  
Select type of project to generate:  
1: basic  
2: application  
3: library  
4: Gradle plugin  
Enter selection (default: basic) [1..4] 2
```

# 使用 Gradle 初始化 Java 專案 (2/4)

- 選擇語 (**3:Java**)

- 能以不同語言建立 application

```
Select implementation language:
1: C++
2: Groovy
3: Java
4: Kotlin
5: Scala
6: Swift
Enter selection (default: Java) [1..6] 3
```

- 選擇 "不" 拆分多個子專案 (**1:no**)

- 2: 包含多個子專案 (資料夾) 的範例, 整體專案結構較複雜

```
Split functionality across multiple subprojects?:
1: no - only one application project
2: yes - application and library projects
Enter selection (default: no - only one application project) [1..2] 1
```

## 使用 Gradle 初始化 Java 專案 (3/4)

- 選擇 Groovy 作為腳本的語言 (1: Groovy)

```
Select build script DSL:  
1: Groovy  
2: Kotlin  
Enter selection (default: Groovy) [1..2] 1
```

- 不使用新版 API 產生建構 (保持預設選項, 直接按 Enter)

```
Generate build using new APIs and behavior (some features may change in the next minor release)? (default: no) [yes, no]
```

- 選擇單元測試框架 (4: JUnit Jupiter)

```
Select test framework:  
1: JUnit 4  
2: TestNG  
3: Spock  
4: JUnit Jupiter  
Enter selection (default: JUnit Jupiter) [1..4] 4
```

## 使用 Gradle 初始化 Java 專案 (4/4)

- 專案名稱和 package 使用預設名稱, 直接按 Enter

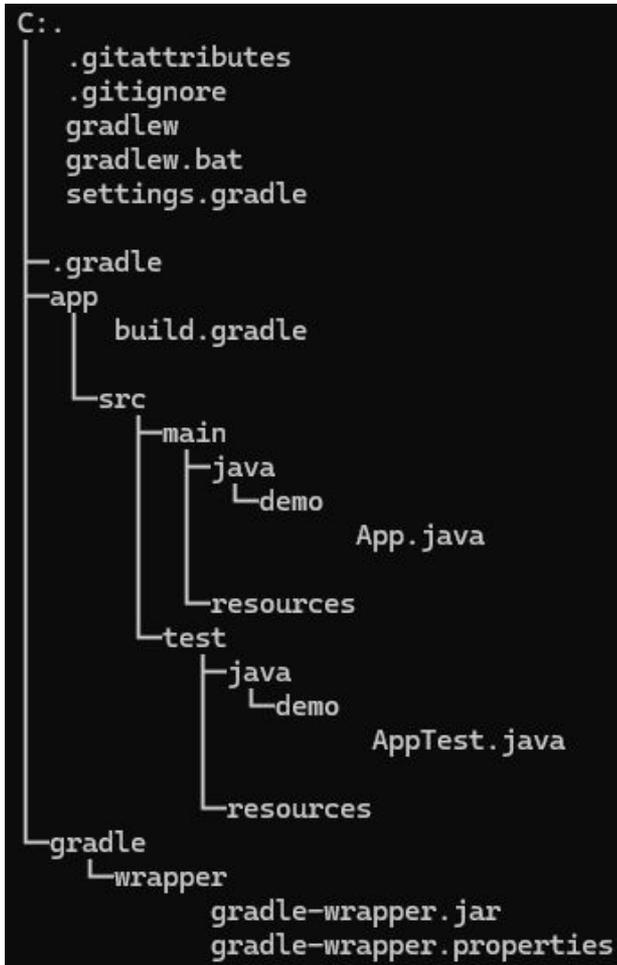
```
Project name (default: demo):
Source package (default: demo):

> Task :init
Get more help with your project: https://docs.gradle.org/7.5.1/samples/sample\_building\_java\_applications.html
BUILD SUCCESSFUL in 1m 41s
2 actionable tasks: 2 executed
```

- 看到 **BUILD SUCCESSFUL**, 表示已經成功初始化專案

# Gradle 專案結構

- 原始碼路徑: `src/main/java/*`
- 測試程式碼路徑: `src/test/java/*`
- 專案建置腳本: `build.gradle`
- 其他 gradle 設定:  
專案根目錄, 與 `src` 資料夾在同一層



# 設定檔說明 (build.gradle)

- demo\app\build.gradle
  - **plugins**
    - 套用 application 插件
  - **repositories**
    - 使用 maven 儲存庫
  - **dependencies**
    - 應用的 libraries  
寫法: 'group:name:version'
  - **application**
    - 定義程式進入點 (main)

```
plugins {
    id 'application'
}
repositories {
    mavenCentral()
}
dependencies {
    testImplementation 'junit:junit:4.13.2'
    implementation 'com.google.guava:guava:31.0.1-jre'
}
application {
    mainClass = 'demo.App'
}
```

# 主程式說明 (App.java)

- demo\app\src\main\java\demo\App.java

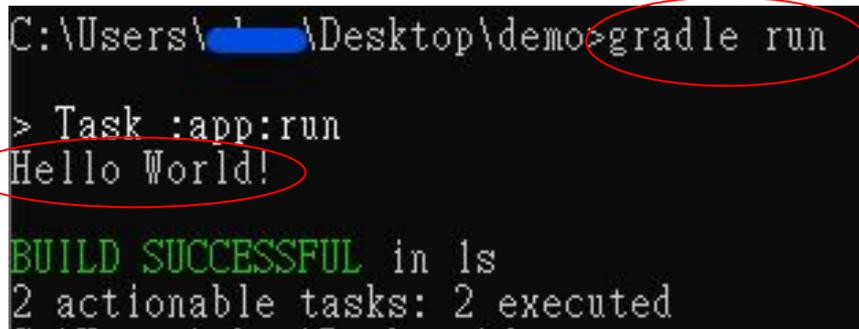
```
4 package demo;
5
6 public class App {
7     public String getGreeting() {
8         return "Hello World!";
9     }
10
11     public static void main(String[] args) {
12         System.out.println(new App().getGreeting());
13     }
14 }
15
```

回傳 Hello World 的函式

呼叫函式, 並印出 Hello World

# 執行專案

- 任務 **run**: 編譯 java 檔 並產生 class 檔, 接著 執行 class 檔
- 在命令提示字元中, 輸入 **gradle run**



```
C:\Users\...\Desktop\demo>gradle run
> Task :app:run
Hello World!
BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
```

- 成功印出 Hello World!  
且 **BUILD SUCCESSFUL**

# 建置並封裝程式

- 任務 **build**:

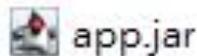
先編譯 java 檔，接著將編譯產生的 class 檔打包成 jar 檔案

- 輸入 **gradle build**

```
C:\Users\1\Desktop\demo>gradle build  
BUILD SUCCESSFUL in 2s  
7 actionable tasks: 6 executed, 1 up-to-date
```

build 不包含執行程式  
因此沒有結果輸出

- 成功建置後，查看 **demo\app\build\libs\** 資料夾
- Gradle 將產生 **app.jar** 檔案



## 執行 jar 檔

- 輸入 `java -cp app\build\libs\app.jar demo.App`  
以執行 jar 檔
- 印出 Hello World! 表示成功

```
C:\Users\... \Desktop\demo>java -cp app\build\libs\app.jar demo.App  
Hello World!
```

# 實際演練 II

新增 plugin, library

# 新增 Commons Lang library 判斷字串 (1/2)

- 引用在 Maven 儲存庫中的 [Apache Commons Lang](#) package

Home » org.apache.commons » commons-lang3



## Apache Commons Lang

Apache Commons Lang, a package of Java utility classes for the classes that are in java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang.

License	Apache 2.0
Categories	Core Utilities
Tags	apache commons
Ranking	#8 in MvnRepository (See Top Artifacts) #2 in Core Utilities
Used By	24,451 artifacts

Central (18)	Redhat GA (14)	Redhat EA (2)	RWJF (1)	ICM (5)
Version	Vulnerabilities	Repository	Usages	Date
3.12.x <b>3.12.0</b>		Central	5,465	Mar 02, 2021
3.11.x 3.11		Central	3,507	Jul 16, 2020
3.10.x 3.10		Central	2,212	Mar 27, 2020
3.9.x 3.9		Central	4,433	Apr 15, 2019

- 選擇最新版本: 3.12.0

# 新增 Commons Lang library 判斷字串 (2/2)

- 選擇 **Gradle (Short)**, 查看寫法

```
Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen
// https://mvnrepository.com/artifact/org.apache.commons/commons-lang3
implementation 'org.apache.commons:commons-lang3:3.12.0'
```

- 複製新增 lib 的寫法
- 加在 **build.gradle** 的 dependencies { }

```
dependencies {
    testImplementation 'junit:junit:4.13.2'

    implementation 'com.google.guava:guava:31.0.1-jre'
    implementation 'org.apache.commons:commons-lang3:3.12.0'
}
```



## Apache Commons Lang » 3.12.0

Apache Commons Lang, a package of Java utility classes for the classes that are in java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang.

License	Apache 2.0
Categories	Core Utilities
Tags	apache commons
HomePage	<a href="https://commons.apache.org/proper/commons-lang/">https://commons.apache.org/proper/commons-lang/</a>
Date	Mar 02, 2021
Files	jar (573 KB) View All
Repositories	Central Xceptance
Ranking	#8 in MvnRepository (See Top Artifacts) #2 in Core Utilities
Used By	24,451 artifacts

```
Maven Gradle Gradle (Short) Gradle (Kotlin) SBT Ivy Grape Leiningen Buildr
// https://mvnrepository.com/artifact/org.apache.commons/commons-lang3
implementation group: 'org.apache.commons', name: 'commons-lang3', version: '3.12.0'
```

# 設定 shadowJar 套件

預設的 build 方法只會打包編譯後的 class 檔，由於程式引入 libraries，因此需要 shadowJar 套件，將外部的 libraries 一起打包到 jar 檔裡面。

- 在 build.gradle 中的 plugins {} 中新增 shadowJar 套件

```
plugins {  
    id 'application'  
    id 'com.github.johnrengelman.shadow' version '5.1.0'  
}
```



- 在 build.gradle 的最後，新增下方程式碼

- 設定 shadowJar 任務
- 設定 mainClass 名稱

```
shadowJar {  
    mainClassName = 'demo.App'  
}
```

## 修改主程式 判斷字串是否為數字

- 在 demo\app\src\main\java\demo\App.java 中, import StringUtils 類別

```
import org.apache.commons.lang3.StringUtils;
```

- 在 main 區塊新增以下程式碼, 以判斷 text1、text2 字串是否為數字

```
public class App {  
    public static void main(String[] args) {  
        String text1 = "a12340";  
        String text2 = "1234";  
  
        boolean result1 = StringUtils.isNumeric(text1);  
        boolean result2 = StringUtils.isNumeric(text2);  
  
        System.out.println(text1 + " is a numeric? " + result1);  
        System.out.println(text2 + " is a numeric? " + result2);  
    }  
}
```

# 執行程式

修改程式和設定檔後，需要重新執行程式，確保正確

- 任務 **run**: 編譯 java 檔 並產生 class 檔，接著 執行 class 檔
- 在命令提示字元中，輸入 **gradle run**

```
C:\Users\...\Desktop\demo>gradle run
> Task :app:run
Hello World!
a12340 is a numeric? false
1234 is a numeric? true
BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
```

成功判斷字串

# 建置並封裝程式

- 任務 **build**:

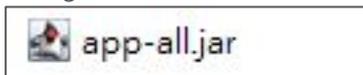
先編譯 java 檔，接著將編譯產生的 class 檔和 libraries 打包成 jar 檔案

- 在命令提示字元中，輸入 **gradle build**

```
C:\Users\...\Desktop\demo>gradle build  
BUILD SUCCESSFUL in 2s  
11 actionable tasks: 4 executed, 7 up-to-date
```

build 不包含執行程式  
因此沒有結果輸出

- 成功後，**demo\app\build\libs\** 下產生 **app-all.jar** 檔案



## 執行 jar 檔

- 輸入 `java -cp app\build\libs\app-all.jar demo.App` 以執行 jar 檔

```
C:\Users\...\Desktop\demo>java -cp app\build\libs\app-all.jar demo.App
Hello World!
a12340 is a numeric? false
1234 is a numeric? true
```

- 能夠成功執行，並印出結果