技術筆記

筆記更新紀錄

|  |  |  |
| --- | --- | --- |
| 日期 yyyy/MM/dd | 更新人員 | 更新備註 |
| 2020/02/10 | Miles | 建立文件 |
| 2020/02/10 | Miles | 建立 對另一個專案依賴 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目錄

[1 關於微服務 3](#_Toc30437851)

[11 4](#_Toc30437852)

[12 5](#_Toc30437853)

## 1建構工具 Gradle

|  |
| --- |
| Gradle |
| 簡介 |
| 1. Gradle是一個參考Apache Ant與Apache Maven概念的專案自動化建構工具。 2. Gradle使用Groovy語言來聲明設定，而不是使用XML。 3. 支援Java、Groovy、Scala等。 4. Gradle與Ant有很緊密的結合，甚至在建構時可以把Ant建構指令碼直接匯入。 |
| 特點 |
| 1. 自動處理套件相依關係 – 取得Maven Repos的概念。 2. 自動處理部屬問題 – 取自Ant的概念。 3. 條件判斷寫法直覺 –使用Groovy語言。 4. 容易辨識與直觀。 |
|  |
|  |
| 建構基本的Gradle Java Project(範例) |
| 1. 建立一個資料夾(Project)      1. 建立 build.gradle 檔案，內容為      1. 建立一個java檔案(測試用) \src\main\java\com\example\Main.java   4   1. 進行編譯 語法為 gradle build      1. 測試 |
| 建構基本的Gradle Java Project(說明) |
| 1. 基本上Gradle Project目錄結構與Maven相似，所以可以參考Maven目錄結構 2. 步驟4解釋當中，建立流程如同Ant的task流程，每一個都是一個任務。 |
| Gradle 基本指令操作 |
| 1. 如果不知道要下哪一個task，可以直接下tasks。     執行後可以看到許多的 task，它們以特定的群組為單位集合在一起(以下僅有部分) |
| 其他語法參考 |
| 如果有在學到或者碰到會再補上   |  |  | | --- | --- | | 語法 | 說明 | | gradle | 主要是help與歡迎Gradle。 | | gradle tasks | 提示可以使用那些task。 | | gradle build | 編譯gradle專案，並且產生一個jar檔案(預設)，jar檔位置在根目錄/build/…。 | | gradle build run | 編譯gradle專案，並且run該專案的main的入口。 | | gradle eclipse | 產生eclipse專案。 | | gradle clean | 清理專案。 | |  |  | |  |  | |
| build.gradle基本結構(範例) |
| 使用gradle結構  // 引用java plugin 獲得編譯 java 專案相關的 task  apply plugin: 'java'  //引用 application plugin 獲得執行 java 專案相關的 task  apply plugin:'application'  //執行 application plugin 用到的參數  mainClassName = "com.example.Main"  //設定 repository server  repositories {  mavenCentral()  }  //宣告專案的相依函式庫  dependencies {  compile group: 'commons-logging', name: 'commons-logging', version: '1.1.1'  compile group: 'log4j', name: 'log4j', version: '1.2.16'  } |
| build.gradle基本結構(說明) |
| 1. apply plugin : 引用插件。   Gradle Plugins 插件 Gradle 在它的核心中有意地提供了一些小但有用的功能，用于在真實世界中的自動化。 所有有用的功能，例如以能夠編譯Java 代碼为例，都是通過插件進行添加的。   1. mainClassName: 入口程式的名稱(需要加上package)。 2. repositories{}: lib來源server，可以當作你要從哪裡抓lib，上面範例是從maven官方函式庫抓lib。 3. dependencies{}: 專案需要使用的函示來倒入。 |
| Closure(閉包) |
| Gradle 支援Closure，而Closure(封閉、閉包)是一個擁有閒置變數(Free Variable)的運算式。  閒置變數真正扮演的角色依當時詞彙環境而定。支援閉包的程式語言通常具有一級函式（First-class function）。建立函式不等於建立閉包。如果函式的閒置變數與當時語彙環境綁定，該函式才稱為閉包。  function doSome(){  var x = 10;  function f(y){  return x+y;  }  return f;  }  var foo = doSome(); //建立一個函式物件  foo(20); //結果為30  foo(40); //結果為50  從上面例子來看，f建立了一個閉包，f當中的x參數看似好像沒定義，其實x是從外部函式doSome()捕捉而來。閉包是個捕捉了外部函式變數(或使之繼續存活)的函式。以上範例，函式f建立了閉包，因為它將變數x關入了自己的函式當中。如果形式閉包的函式物件持續存活，被關閉的變數x也會繼續存活。就像是延續了變數x的生命週期。  看起來閉包應該是{}的x+y兩者產生的一種形式。 |
| Gradle 支援Closure |
|  |
| Build Script 與 Project 物件 |
| 我們寫的build.gradle檔案，它是Gradle Build Script的預設檔名。  Build Script檔案被Gradle載入後轉換成BuildScript物件，本質上它是一個Groovy Script。  Groovy Script可以設定Base Class，它的效果就如同替Closure指定delegate(代表)一般，任何你在Build Script內使用的方法、屬性都會交給Base Class處理，對Gradle來說它將這個Base Class封裝成Project物件。  根據Gradle官網提示: There is a one-to-one relationship between a Project and a “build.gradle" file. During build initialisation, Gradle assembles a Project object for each project which is to participate in the build.  (Project 與 build.gradle文件之間存在一對一的關係。在建構初始化期間，Gradle為每個參與建構的項目組裝一個Project物件)。 |
| Build Script 與 Project 物件嘗試解讀 |
| apply plugin: 'java'  apply plugin: 'application'    mainClassName = "tw.com.codedata.HelloWorld"    repositories {      mavenCentral()  }    dependencies {      compile group: 'commons-logging', name: 'commons-logging', version: '1.1.1'      compile group: 'log4j', name: 'log4j', version: '1.2.16'  }  以上build.gradle本質上就是一個java object。所以它將會有對應的method能使用。  1. apply plugin: 'java'  Javadoc URL: <https://docs.gradle.org/current/javadoc/org/gradle/api/plugins/PluginAware.html#apply-groovy.lang.Closure->  可以找到三種方法      (2)  (3)  在文件中可以查到三個apply方法，根據以上範例，我們使用的是Map那組，也可以知道有三種Key值，from、plugin、to。 |
| Gradle DSL文件導讀 |
| 可以參考URL: <https://docs.gradle.org/current/dsl/>     1. Build script => build.gradle   對應一個project   1. Settings script => settings.gradle   看起來比較偏向多個project的導入運作。     1. Init script => 尚未研究 |
| 多個模組的建立方式 |
| 1. 多個模組有分為 分層式與水平式兩種 |
| 參考網站 |
| 1. 基本說明：   <http://www.codedata.com.tw/java/understanding-gradle-3-getting-started/>   1. 插件說明：   <https://www.kancloud.cn/kancloud/gradle2-user-guide/52725>   1. Grable Project API: <https://docs.gradle.org/current/javadoc/org/gradle/api/Project.html> |

## 2對另一個專案依賴

|  |
| --- |
| 對於另一個專案的依賴方式 |
| 我要如何取得另一個專案的方法? |
| 直接在dependencies當中鍵入compile project(':專案名稱') 即可 |
| 如果我遇到要依賴的專案在另一個專案之中(另一個專案的子專案)改怎麼做? |
| 鍵入compile project(':根專案名稱:子專案名稱')即可 |

## 3

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |