技術筆記

筆記更新紀錄

|  |  |  |
| --- | --- | --- |
| 日期 yyyy/MM/dd | 更新人員 | 更新備註 |
| 2019/03/21 | Miles | 建立文件，添加技術目錄Apache Ant、Apache Maven、Gradle、Simple Factory…等等 |
| 2019/03/22 | Miles | 更新Gradle內容 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目錄

[1建構工具 Apache Ant 4](#_Toc4168119)

[2建構工具 Apache Maven 4](#_Toc4168120)

[3建構工具 Gradle 6](#_Toc4168121)

[4設計模式- Simple Factory 14](#_Toc4168122)

[5設計模式- 15](#_Toc4168123)

[5技術框架- Spring 16](#_Toc4168124)

[6技術框架- Hibernate 17](#_Toc4168125)

[7技術框架- JPA 18](#_Toc4168126)

[8技術框架- NodeJS 19](#_Toc4168127)

[9自動化打包工具- Jenkins 20](#_Toc4168128)

[10專案管理- Git 21](#_Toc4168129)

[11 Server- Oracle WebLogic Server 22](#_Toc4168130)

[12 Server- WildFly JBoss 23](#_Toc4168131)

[13 Server- Apache Tomcat 24](#_Toc4168132)

[14 HTTP概念 25](#_Toc4168133)

## 1建構工具 Apache Ant

|  |
| --- |
| Apache Ant |
| 簡介 |
| 早期使用XML撰寫的建構技術 |
| 優缺點 |
|  |
| 基本結構 |
|  |

## 2建構工具 Apache Maven

|  |
| --- |
| Apache Maven |
| 簡介 |
| 自動化編譯工具、Java專案的標準化、Java專案函示庫的統一 |
| 優缺點 |
|  |
| pom.xml基本模板 |
| <?xml version="1.0" encoding="UTF-8"?>  <!-- pom的模板文件，包含內容節點的定義 -->  <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  <!-- 模板版本 -->  <modelVersion>4.0.0</modelVersion>  ...  ...  ...  </project> |
| 註解 |
| 所有的pom文件需要<project>元素與三個子元素:groupId、artifactId、version。 |
| 子元素訊息 |
| 1. <groupId></groupId>:組織標識。 2. <artifactId></artifactId>:項目名稱(專案名稱) 。 3. <version></version>:版本號碼。 4. <packaging></packaging>:打包的格式，可以為:pom、jar、maven-plugin、ejb、war、ear、rar、par。 5. <name></name>:項目名稱，maven產生文件檔使用。 6. <url></url>:項目主頁的URL，maven產生文件檔使用。 7. <description></description>: |
| 基本目錄結構 |
| MavenProject/  ├── pom.xml  └── src      ├── main      │   └── java      │       └── tw      │           └── com      │               └── codedata      │                   └── App.java      └── test          └── java              └── tw                  └── com                      └── codedata                          └── AppTest.java    11 directories, 3 files |
| 註解 |
| 1.平行的目錄結構 src/main/java 與 src/test/java 下的 package 即為 groupdId。  2.以上使用maven-archetype-quickstart樣板產生 |
| 其他 |
| 1. super pom:父pom |
| 建構生命週期 |
| 什麼是建構生命週期，建構的階段，其中每一個階段都是生命周期的一部分。  以下舉例基本的序列:   |  |  |  | | --- | --- | --- | | 階段 | 處理 | 描述 | | prepare-resource | 拷貝資源 | 本階段可以定義需要拷貝的資源。 | | compile | 編譯 | 本階段完成程式碼編譯。 | | pagkage | 打包 | 本階段根據pom.xml所描述的配置創建jar/war檔。 | | install | 安裝 | 本階段在本地/遠程位置中安裝打包檔案。 | |

## 3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 |
| 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> |

## 4設計模式- Simple Factory

|  |
| --- |
| Simple Factory模式(工廠模式) |
| 簡介 |
| 1. 又稱 Static Factory。 2. 對客戶端隱藏產品產生的細節，物件如何生成，生成前是否與其他物件建立依賴關係，客戶端皆不用理會，用以將物件生成方式之變化 與客戶端程式碼隔離。 |
| 參考網站 |
| https://openhome.cc/Gossip/DesignPattern/SimpleFactory.htm |

## 6技術框架- Hibernate

|  |
| --- |
| Hibernate |
| 簡介 |
|  |
| 優缺點 |
|  |

## 8技術框架- NodeJS

|  |
| --- |
| NodeJS |
| 簡介 |
|  |
| 優缺點 |
|  |

## 9自動化打包工具- Jenkins

|  |
| --- |
|  |
| 簡介 |
|  |
| 優缺點 |
|  |

## 11 Server- Oracle WebLogic Server

|  |
| --- |
|  |
| 簡介 |
|  |
| 優缺點 |
|  |

## 12 Server- WildFly JBoss

|  |
| --- |
| WildFly JBoss |
| 簡介 |
|  |
| 優缺點 |
|  |

## 13 Server- Apache Tomcat

|  |
| --- |
| Apache Tomcat |
| 簡介 |
|  |
| 優缺點 |
|  |

## 14 HTTP概念

|  |
| --- |
| HTTP |
| 簡介 |
|  |
| 優缺點 |
|  |