技術筆記

筆記更新紀錄

|  |  |  |
| --- | --- | --- |
| 日期 yyyy/MM/dd | 更新人員 | 更新備註 |
| 2019/04/03 | Miles | 建立文件 |
| 2019/04/08 | Miles | 添加測試配置文件內的自定義屬性 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目錄

[1 Spring Cloud基本練習與理解 3](#_Toc33634656)

[2 Spring Cloud Config 9](#_Toc33634657)

[3 Spring Cloud Eureka 10](#_Toc33634658)

[4 Spring Cloud Zuul 10](#_Toc33634659)

[5 Spring Cloud Consul 11](#_Toc33634660)

[6 Spring Cloud 使用HTTPS 11](#_Toc33634661)

[99 IDE 環境安裝 12](#_Toc33634662)

## 1 Spring Cloud基本練習與理解

|  |
| --- |
| Spring Cloud |
| 簡介: |
| 首先要了解Spring Boot   1. 甚麼是SpringBoot?   Takes an opinionated view of building production-ready Spring applications. Spring Boot favors convention over configuration and is designed to get you up and running as quickly as possible.以上擷取官網  簡化了建構Spring的剛複雜與麻煩。Spring Boot傾向於約定優於配置，旨在讓您盡快啟動並運行。  並且它是建構在Spring Cloud之上的基礎模組。 |
| 練習: |
| 1. 使用技術: 專案管理(Gradle)、技術框架(Spring Cloud)… 2. 參考架構規範: Microservice 3. 建立Gradle專案，並且建立子專案(Sub Project)，父專案與子專案目錄結構大致如下:        1. 因為使用專案管理Gradle框架，所以每一個專案(不管是父專案還是子專案)都需要有一個build.gradle的配置檔，而settings.gradle 主要做注入專案的動作，以下是settings.gradle基本結構:     其他可以參考GitHub範例。   1. 父專案的build.gradle 尚未研究，但是可以使用一些語法，例如allProject、subProject進行專案管理。 2. 首先，Spring Cloud 有分為 Spring Cloud Config Client 與 Spring Cloud Config Server   兩端，我們可以建立基本的Server配置   1. 我們先在SpringCloud-Config專案建立build.gradle配置，並且倒入基本套件與注入依賴。以下是SpringCloud-Service01-Config Server端的基本配置:       以上build.gradle部分內容尚未釐清目的所以需要再花時間研究。其中，    第一個是說明依賴Spring Cloud Config Server端需要使用的套件  第二個是說明依賴Spring Cloud Eureka starter啟動需要使用的套件  以上可以說明就是專案(SpringCloud-Config)為一個配置專案，並且使用到了Eureka。  Eureka可以參考以下章節: Spring Cloud Eureka介紹。   1. 建立第一個Spring Cloud Config Server 配置專案入口     其中，@SpringBootApplication 是一個SpringBoot 的入口，@ComponentScan 是自動掃描，@EnableConfigServer是一個Spring Cloud 的配置路口，@EnableDiscoveryClient是代表這是一個Eureka的Clinet端口。  以上配置代表該專案的主程式是一個微服務入口。   1. 基本配置:我們使用多個配置方式，如果只想用一個配置可以將以下圖中bootstrap-xxx.yml內容複製至bootstrap.yml中覆蓋     其中yaml 與 properties 差異後續研究補上  其中bootstrap 與 application 差異請參考後續內容 |
| 其他筆記自定義屬性 |
| 1. 建立一個RestController如下:      1. 啟動Server後至瀏覽器輸入<http://localhost:8990/miya>，便會看到     讀取文件內容與法 @Value(“${属性名}”)   1. 在application.yml或者是bootstrap.yml中鍵入以下屬性 |
| 關於 bootstrap 與 application: |
| Spring Cloud 建構於 Spring Boot 之上，在 Spring Boot中有兩種Content，一種是bootstrap，另一種是application，bootstrap是應用程式的父親context，也就是說它會優先加載於application。Bootstrap主要用於額外的資源配置訊息 |
| 遠端取的配置內容 |
| 1. 可以將配置文件存放在遠端Git或者是其他資料夾中，方便統一管理與更新 2. 如果要遠端配置，那請在SpringCloud-Service01-Config 當中先將配置遠端的部分撰寫完畢，參考如下:     細節可能還需要再研究…   1. Config配置遠端存放位置之後Client如果要取得配置，則要在bootstrap.yml中撰寫取得規則:     主要是Spring.cloud.config中的內容  spring.cloud.config.name 配置檔名稱，這裡有一個問題，如果如以上規則spring.application.name名稱設為eureka而下方的spring.cloud.config.name設為service  那麼在取得配置的時候會先參考spring.cloud.config.name中的內容，但是實際上這個service的名稱是eureka。  spring.cloud.config.profile 配置的檔案後綴名稱。  spring.cloud.config.uri: 指向配置Service的URL。  spring.cloud.config.label: 存在Git的分支，預設master。  取得配置實體檔案的規則請參考以下:  Spring 取的配置文件有一定的規則，其中  {application}-{profile}.properties  {application}-{profile}.yml  master/{application}-{profile}.properties  master/{application}-{profile}.yml |
|  |
| . |

## 2 Spring Cloud Config

|  |
| --- |
| 簡介: |
|  |
|  |
|  |
|  |
|  |
|  |
| 參考文獻: |
|  |

## 3 Spring Cloud Eureka

|  |
| --- |
| Spring Cloud Eureka 是什麼? |
| 1. Eureka是微服務(microservice)架構中最為核心與基礎的模塊，它主要是用來實現各種微服務實例的自動化和發現。 2. Spring Cloud Eureka是Spring Cloud Netfilx微服務套件中的一部分，它是建立在Netfilx Eureka基礎上二次封裝，主要負責完成微服務架構中的服務治理功能。 3. Eureka 是一個註冊中心，經過註冊後Eureka可以發現被受註冊的Service。 4. Eureka 有分成 Server端 與Client端。Server端主要是建構這個Service為Eureka Server，   其他的Service可以是一併註冊為Client，那麼Eureka Server就會發現。 |
| 參考文獻: |
| URL: <https://zhuanlan.zhihu.com/p/34976125> |

## 4 Spring Cloud Zuul

|  |
| --- |
| Spring Cloud Zuul 是什麼? |
| 1. Zuul 閘道器 2. Zuul 透過自動發現機制(Eureka)，無須變更軟體組態，只要服務發現新增了服務，他就可以做一個內外部的代理功能，其實自己也是Service一種。 3. Zuul 是設備和網站到接收端(後台架構)應用程序所有請求的前門。 4. Zuul 具有 實現動態路由、監控、彈性、安全性 的特性。 |
| 參考文獻: |
| URL: <https://blog.csdn.net/j080624/article/details/81234465> |

## 5 Spring Cloud Consul

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

## 6 Spring Cloud 使用HTTPS

|  |
| --- |
| 使用HTTPS以前 |
| 關於HTTPS是什麼? |
| 參考GitHub >> Technical-documents >> 技術筆記 HTTP與HTTPS 的技術文件 |
| 產生一個SSL相關的憑證 |
| 1. 使用keytool生成keystore文件，並且將該文件複製到根目錄底下(?)         尚未達到瀏覽器認證。 |

## 7 Spring Cloud 使用外部Tomcat部屬與打包(WAR)

|  |
| --- |
| 1.將Spring 打包War |
| 我們要將Spring 專案中部分程式碼修改    必須繼承SpringBootServletInitializer 因為    此抽象類別有說明該類別主要是給打包War檔使用，如果不使用的話會產生出Jar檔案包。 |

|  |
| --- |
| 2.將Gradle 專案產生War |
| 我們要導入一個套件    鍵入apply plugin: ‘war’ |

|  |
| --- |
| 3.排除掉Spring Boot內部的Tomcat |
| 每一包Project都要鍵入以上語法或者是放在全域之中。 |

|  |
| --- |
| 3.排除掉Spring Boot內部的Tomcat |
| 每一包Project都要鍵入以上語法或者是放在全域之中。 |

|  |
| --- |
| 4.修改配置檔案 |
| 我們要將所有URL連結補上War包名稱，因為Tomcat預設連結會吃War包的命名。    我們可以看出差異性。 |

|  |
| --- |
| 5.打包專案(Eclipse) |
| 我們可以在根專案底下按下右鍵=>Run As =>Run Configurations=>  找到Gradle Project    我們可以在Gradle Project 按下右鍵=> Create Configuration，  右邊的視窗鍵入  Name : [name]  Gradle Tasks (tab)  Gradle Tasks: [build]  Working Directory:[${workspace\_loc:/SpringCloudRoot}]  並且按下Run 按鈕之後即可以完成。 |

|  |
| --- |
| 6.尋找打包後的War |
| War 會預設存放在各專案的build資料夾中的libs當中 |

|  |
| --- |
| 7.安裝Tomcat |
| 請自行去官方網站下載壓縮檔後解壓縮放置非中文的目錄底下。 |

|  |
| --- |
| 8.配置Service |
| 可以進入Tomcat =>conf=>server.xml  根據微服務有多少Service就複製多少個Service 標籤    我們可以修改port，並且將Engine 標籤 的名稱同步Service 標籤的name。  修改Host標籤當中的appBase=資料夾名稱  多個微服務的話就複製多個並且如上方式更改名稱。 |

|  |
| --- |
| 9.將War包複製到Tomcat 的webapp底下 |
| 因為我們剛剛在上一步動作中有修改appBase這個屬性，該屬性主要是從哪一個資料夾取得War檔案。  所以我們要將War複製到這些配置的路徑底下。 |

|  |
| --- |
| 10.啟動Server |
| 找到startup.bat  即可成功運行。 |

## 99 IDE 環境安裝

|  |
| --- |
| 環境安裝步驟(暫定) |
| 1. 安裝Gradle (6.0.1) |
| 請自行google |
| 1. 安裝JDK(1.8以上) |
| 請自行google |
| 1. 下載Eclipse |
| 請自行至Eclipse下載安裝 |
| 1. Eclipse 下載套件 |
| 請至Eclipse Marketplace ，並且安裝以下套件        管理Andriod 專案使用(可以不用安裝)    Spring Tools4 (可安裝4.5.0)    安裝後系統會提示重新啟動Eclipse |
| 1. 安裝MySQL DataBase |
| 請自行Google安裝 |
|  |
|  |