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 4](#_Toc7701060)

[2 Spring Cloud Config 10](#_Toc7701061)

[3 Spring Cloud Eureka 10](#_Toc7701062)

[4 Spring Cloud Zuul 11](#_Toc7701063)

[5 Spring Boot 使用不同 Server 11](#_Toc7701064)

[6 Spring Data JPA 12](#_Toc7701065)

[6 Spring Boot Log(默認日誌) 12](#_Toc7701066)

[7 Spring Boot Test 14](#_Toc7701067)

[8 Spring Session 14](#_Toc7701068)

[9 JSON Web Token(JWT) 15](#_Toc7701069)

[10 Spring Batch(不重要，暫時不需要看) 17](#_Toc7701070)

[11 Spring Security 16](#_Toc7701071)

[11 Spring Security 建置OAuth2 16](#_Toc7701072)

## 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

|  |
| --- |
| Spring Cloud Config |
| 簡介: |
|  |
| 自己的理解: |
|  |
| 基本練習: |
|  |
| 參考文獻: |
| URL: |

## 3 Spring Cloud Eureka

|  |
| --- |
| Spring Cloud Eureka |
| 簡介: |
| 1. Eureka:   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。   1. 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。 |
| 自己的理解: |
| 1. Eureka 是一個註冊中心，經過註冊後Eureka可以發現被受註冊的Service。 2. 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。 |
| 基本練習: |
|  |
| 參考文獻: |
| URL: <https://zhuanlan.zhihu.com/p/34976125> |

## 4 Spring Cloud Zuul

|  |
| --- |
| Spring Cloud Zuul |
| 簡介: |
| 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。 |
| 自己的理解: |
|  |
| 基本練習: |
|  |
| 參考文獻: |
| URL: <https://blog.csdn.net/j080624/article/details/81234465> |

## 5 Spring Boot 使用不同 Server

|  |
| --- |
| Spring Boot 使用不同 Server |
|  |
|  |
|  |
|  |

## 6 Spring Data JPA

|  |
| --- |
| Spring Data JPA |
| 簡介: |
|  |
|  |
|  |

## 7 Spring Boot Log(默認日誌)

|  |
| --- |
| Spring Boot Log(默認日誌) |
| 解析: |
| Spring Boot Logback是Spring Boot內建的Log日誌系統。使用Commons Logging，支援Java Util Logging、Log4J、Log4J2和Logback。每種Logger都可以通過配置使用控制台或者文件輸出內容。  Spring Boot 本身支援Tomcat 所以可以使用Log4j。  Spring Boot 預設Logback 所以可以使用。 |
| 基本使用 |
| 1.    2.    如果要使用Logback 也是使用一樣的語法。 |
| 手動設定Log細節 |
| 1. |
| 推薦使用: |
| 推薦使用Log4j2，因為整體效能速度是其他的Log日誌系統中最快的甚至速度高達10倍之多。 |

## 8 Spring Boot Test

|  |
| --- |
| Spring Boot Test |
| 簡介: |
|  |
| 基本使用: |
|  |

## 9 Spring Session

|  |
| --- |
| Spring Session |
| 簡介 |
| 1. session 存放在伺服端，主要是讓Web應用程式可以看似記得是瀏覽器發出的請求，連接數個請求間的關係。   記得:我覺得應該是說這個存放資訊的箱子，並且放在伺服端內。   1. 至於所謂的連接數個請求的關係這個部分由Web容器幫你執行。 2. Session 裡面會存放Session Id 3. 以下參考     瀏覽器發送請求會一併帶Session Id去接收端  如果沒有Session Id的時候Web 容器會產生一個Session物件與Session Id，並且將這個Session Id回傳給瀏覽器，而瀏覽器會將其Session Id存放在Cookie中 |
| 參考文獻: |
| 1. OpenHome: <https://openhome.cc/Gossip/ServletJSP/BehindHttpSession.html> |

## 10 JSON Web Token(JWT)

|  |
| --- |
| JSON Web Token |
| 簡介: |
| 首先要先了解Http得Session和Cookie等原理。  Token是什麼?  當使用者登入後，他會得到一串看起來毫無意義的亂數字串，但這個亂數字串實際上對應到資料庫中使用者的身分，簡單來說就是另類的帳號，這也被叫做Token。  但是使用Token有個問題?  因為所有的Token字串都存放在…資料庫當中，這代表著一堆使用者同時進入伺服端。那是不是伺服端的負擔就提高了。  所以我們要使用JWT !  JWT是使用一段字串例如  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ  以上是由官方網站提供的一個範例  標頭.內容.簽名  以上的規則就是這樣(請參考顏色對應)  最後告訴你，JWT不是技術、不是技術、不是技術，他是一個規範  所以JWT會在Client 與Server互相傳遞，代表該參數不要放入任何敏感字眼!(例如密碼) |
|  |
|  |
| 參考網站: |
| 1. 關於HttpSession 觀念: <https://openhome.cc/Gossip/ServletJSP/BehindHttpSession.html> 2. 關於JWT基本說明: <https://yami.io/jwt/> 3. 關於JWT基本範例: <http://blog.leapoahead.com/2015/09/06/understanding-jwt/> 4. JWT官方網站: <https://jwt.io/> |

## 11 Spring Security

|  |
| --- |
| Spring Security |
| 簡介: |
| 1.首先…請先看關於JSON Web Token(JWT)，再來看這篇 |
|  |
|  |

## 12 Spring Security 建置OAuth2

|  |
| --- |
| Spring Security |
| 簡介: |
| 1. 首先…請先看關於JSON Web Token(JWT) 在看Spring Security，再來看這篇 |
| 參考網站: |
| OAuth2基本解說: <https://blog.yorkxin.org/2013/09/30/oauth2-1-introduction> |

## 13 ~~Spring Batch~~

|  |
| --- |
| Spring Batch 暫時考慮不使用，改成研究Quarkz |
|  |
|  |
|  |
|  |