技術筆記

筆記更新紀錄

|  |  |  |
| --- | --- | --- |
| 日期 yyyy/MM/dd | 更新人員 | 更新備註 |
| 2020/02/15 | Miles | 建立文件 |
| 2020/02/18 | Miles | 添加在不使用任何配置的情況之下預設的Security |
| 2020/02/19 | Miles | 添加不需要放置@EnableWebSecurity的原因 |
|  |  |  |
|  |  |  |
|  |  |  |

目錄

[1關於Spring Security 4](#_Toc33027266)

[2剖析Spring Security 概念 4](#_Toc33027267)

[3 不需要使用EnableWebSecurity 的原因 4](#_Toc33027268)

[3使用Spring Security(預設情況) 6](#_Toc33027269)

[3使用Spring Security(UserDetails 與 User) 8](#_Toc33027270)

[4使用Spring Security(身分驗證管理器) 10](#_Toc33027271)

[3使用Spring Security(身分驗證管理器-自定義使用者權限) 14](#_Toc33027272)

[99其他語法與效果 15](#_Toc33027273)

## 1關於Spring Security

|  |
| --- |
| Spring Security 是什麼? |
| Spring Security是一個基於Java / Java EE的框架，為企業應用程式提供身份驗證，授權和其他安全功能。為Spring的其中一個模組。 |

## 2剖析Spring Security 概念

|  |
| --- |
| 1 安全認證主要分成兩個部分(Spring Security 核心主要功能) |
| 1 認證方式(Authentication)，驗證身分的合法性。 |
| 2 授權書(Authorization)，我們也可以稱為訪問控制。授權代表你可以做甚麼。 |

|  |
| --- |
| 2 使用的Lib 介紹(一般情況) |
| 1 spring-security-core: 包含核心的驗證與訪問控制類別和介面。任何Spring Security都需要這個核心模塊Jar。 |
| 2 spring-security-web: 網頁驗證HTTP或者是Service需要這個模塊(jar)。 |
| 3 spring-security-config: |
| 4 spring-security-test: Spring Security 的測試。 |

|  |
| --- |
| 2 使用的Lib介紹(使用Spring Cloud) |
| 1 spring-cloud-starter-security: 此Lib會自動幫我們導入spring-security-core、spring-security-web、sprint-security-cofig等Lib |

|  |
| --- |
| 3 使用的Lib介紹(使用Spring Boot) |
|  |

|  |
| --- |
| 4 其實就是一連串的Filter |
| 實現了各種Filter 其原理使用AOP的改念。  可以說是建立多個Filter鏈(Filter Chan) |

## 3不綁定@EnableWebSecurity 的原因(參考用)

|  |
| --- |
| 1 Spring Boot 幫我啟動了Spring Security (核心概念) |
| 當我們使用了Spring Boot 或者是 Spring Cloud 的starter-config時，Spring 的factories會自動的呼叫EnableAutoConfiguration 並且指向SecurityAUtoConfiguration 類別。    該類別會Import三個類別    1 SpringBootWebSecurityConfiguration.class  2 WebSecurityEnablerConfiguration.class  3 SecurityDataConfiguration.class  如果有導入Spring Security 相關jar檔(core) 那麼該類別(SecurityAutoConfiguration將會自動啟用。[其中WebSecurityEnablerConfiguration.class當中有一個 @EnableWebSecurity](mailto:其中WebSecurityEnablerConfiguration.class當中有一個@EnableWebSecurity)會幫我們啟動Spring Security 功能。 |
| 結論就是 當我們使用Spring Boot 或者是Spring Cloud 的時候，並不需要取啟用@EnableWebSecurity 這個Annotation。 |
| 如果無法類似以上的方式找尋Auto Configurar 找到該Annotation，那就安全起見，還是要加上@EnableWebSecurity這個Annotation。 |

## 4關於@EnableWebSecurity 的運作

|  |
| --- |
| 進入原始碼看說明 |
| 1添加這個Annotation在@Configuration 的類別上就能擁有Spring Security功能。 |
| 2大致上就是啟動了該Annotation之後會幫我們呼叫WebSecurityConfiguration這個配置類別 |
| 3 該類別會呼叫WebSecurityConfigurerAdapter這個抽象類別(也就是之後我們要繼承的類別)，並且呼叫這個類別的配置方法(我們已經@Overrid)，包含產生各種Filter。 |
| 4 還有後續…尚未研究。 |

## 5使用Spring Security(預設情況的簡單範例)

|  |
| --- |
| 1 導入Lib |
| 因為使用微服務，所以可以直接在Gradle導入    範例暫時使用2.1.3.RELEASE 版本。  如果非使用微服務，可導入以下Lib |

|  |
| --- |
| 2 寫一個測試的RestController |
| 我們寫了一個入口 |

|  |
| --- |
| 3 啟動這個專案 |
| 因為我們沒有做任何Security的配置，所以這時候Spring Security的容器會幫我們產生一個預設的密碼    以上Using generated security password 就是預設的密碼。帳號則是user  也代表說在預設情況之下，如果都不進行任何配置，Spring Security會對所有URL進行表單驗證模式，並且帳號密碼也是使用預設的規則。 |

|  |
| --- |
| 4 瀏覽器測試結果 |
| 瀏覽器輸入localhost:XXXX/test 會出現以下畫面:    特別注意，該畫面是Spring提供的httpBasicLogin畫面，做為測試用。  輸入帳號user 與密碼:#######    如果我們把URL輸入改成localhost:XXXX/logout    以上詢問是否登出的畫面也是Spring提供的測試用畫面 |

## 6剖析WebSecurityConfigurerAdapter 概念

|  |
| --- |
| WebSecurityConfigurerAdapter是什麼? |
| 1 WebSecurityConfigurerAdapter 是預設情況下Spring Security的攔截URL或Http配置安全性相關類別。 |
| 2 WebSecurityConfigurerAdapter是提供一個方便開發人員配置WebSecurity的基本類別。 |
| 3 WebSecurityConfigurerAdapter 是Spring Security Config 內置提供的一個WebSecurityConfigurer抽象類別。其本身就是一個WebSecurityConfigurer(因為實作了該介面)。 |

|  |
| --- |
| 配置方法1 HttpSecurity |
| 此方法用於請求控制的配置 |

|  |
| --- |
| 配置方法2 WebSecurity(較少使用) |
| 此方法用整個FilterChainProxy的配置 |

|  |
| --- |
| 配置方法3 |
| 此方法用配置認證管理器 |

## 7剖析WebSecurityConfigurer 概念

|  |
| --- |
| WebSecurityConfigurer是什麼? |
| 1 WebSecurityConfigurer 是一個提供產生WebSecurity的介面。 |

## 8剖析SecurityConfigurer 概念

|  |
| --- |
| WebSecurityConfigurer是什麼? |
| 1 WebSecurityConfigurer 是一個提供產生WebSecurity的介面。 |

## 9剖析HttpSecurity配置器 概念

|  |
| --- |
| HttpSecurity是什麼? |
| HttpSecurity是一個安全建構器SecurityBuilder，目的是建構一個對Http請求進行安全控的DefaultSecurityFilterChain。(更白話一點的說法就是建立一個攔截Http請求的Filter，但是因為有很多實際情況發生，所以並不是只單純建立一種一種Filter，而是多種不同的Filter，故稱之為SecurityFilterChain)。 |

|  |  |
| --- | --- |
| HttpSecurity 可以建立那些Filter? | |
| 以下參考(尚未使用到的不會寫在簡介裡，方便未來學習後補上、名稱下方的使用方法名稱) | |
| 名稱(方法名稱/對應產生類型) | **簡介** |
| Anonymous()  /  AnonymousConfigurer |  |
| requiresChannel ()  /  ChannelSecurityConfigurer |  |
| cors ()  /  CorsConfigurer |  |
| csrf ()  /  CsrfConfigurer | **添加csrf的支援，當使用WebSecurityConfigurerAdapter的時候，預設會啟用。** |
| DefaultLoginPageConfigurer | **在HttpSecurity Soruce Code沒有找到** |
| exceptionHandling ()  /  ExceptionHandlingConfigurer |  |
| authorizeRequests ()  /  ExpressionUrlAuthorizationConfigurer | **配置FilterSecurityInterceptor** |
| formLogin ()  /  FormLoginConfigurer | **指定為表單驗證，如果沒有指定FormLoginConfigurer#loginPage(String) 會跳到Spring 提供的預設登入頁面。** |
| HeadersConfigurer |  |
| HttpBasicConfigurer |  |
| JeeConfigurer |  |
| LogoutConfigurer |  |
| PortMapperConfigurer |  |
| RememberMeConfigurer |  |
| RequestCacheConfigurer |  |
| SecurityContextConfigurer |  |
| ServletApiConfigurer |  |
| SessionManagementConfigurer |  |
| X509Configurer |  |
| OAuth2ClientConfigurer |  |
| OAuth2LoginConfigurer |  |
| OAuth2ResourceServerConfigurer |  |
| OpenIDLoginConfigurer |  |

## 10剖析WebSecurity配置器 概念

|  |
| --- |
| WebSecurity是什麼? |
| 基本上跟HttpSecurity的用意差不多，但是差別在於Web Security的建構目標是整個Spring Security 安全過濾器FilterChainProxy。而HttpSecurity的建構目標只是FilterChainProxy中的一個SecurityFilterChain。 |
| 因為較少使用(應該說一般人不會去使用)所以暫時忽略。 |

## 11剖析AuthenticationManagerBuilder 配置器 概念

|  |
| --- |
| AuthenticationManagerBuilder 是什麼? |
| 認證管理器的配置當中。 |
| 而Spring Security 的驗證主要是交由AuthenticationProvider實作的介面來處理(如下表格) |

|  |  |
| --- | --- |
| AuthenticationManagerBuilder 常見的幾種SecurityConfigurer | |
| 方法(方法名稱/產生類型) | **簡介** |
| inMemoryAuthention()  /  InMemoryUserDetailsManagerConfigurer | **直接將使用者資訊存入記憶體當中。** |
| jdbcAuthention()  /  JdbcUserDetailsManagerConfigurer |  |
| ldapAuthention()  /  LdapAuthenticationProviderConfigurer |  |
| userDetailsService(T)  /  UserDetailsService | **用來載入使用者資訊的一種組件。** |
| authenticationProvider(T)  /  AuthenticationProvider | **直接撰寫一個實作AuthenticationProvider介面的驗證方式。當中可能就包含UserDetailsSerivce方法** |
|  |  |

|  |
| --- |
| AuthenticationManagerBuilder 使用inMemoryAuthention() 簡單範例 |
| 使用inMemoryAuthention可以直接寫入使用者相關權限 |
|  |
| 也可以自行定義UserDetails ，使用User進行產生UserDetails的實體，例如: |
|  |

|  |
| --- |
| AuthenticationManagerBuilder 使用authenticationProvider() 簡單範例 |
| 使用authenticationProvider 需要一個實作AuthenticationProvider介面物件，如果不使用任何認證方式，Spring Security 會使用DaoAuthenticationProvider作為預設。 |
|  |
| 直接在DaoAuthenticationProvider傳入UserDetailsService |
|  |

## 12使用Spring Security(UserDetails 與 User)

|  |
| --- |
| 1 UserDetails 是什麼? |
| 這是一個提供提取使用者資訊的介面。 |

|  |
| --- |
| 2 User 是什麼? |
| 這是一個存放使用者的資訊。該類別實作了UserDetails介面，所以該物件具備提取使用者資訊的方法。 |

|  |
| --- |
| 2.1 User 有哪些參數 |
| 我們進入原始碼可以看到    1 password : 使用者密碼  2 username : 使用者名稱  3 authorities :　使用權限  4 accountNonExpired : 帳戶是否過期  5 accountNonLocked : 帳戶是否凍結  6 credentialsNonExpired : 使用者密碼是否過期  7 enabled : 帳號是否可使用 |

|  |
| --- |
| 3 建立一個UserDetails ，他需要UserDetailsService 幫忙產生! |
| 因為SpringSecurity 會自動呼叫UserDetailsService介面(前提是你要啟用Spring Security)  所以當我們要使用時，可以直接Overrid 這個介面，讓Spring Security呼叫我們也得UserDetailsService    該介面只有一個方法需要實作    時候做後可以使用    該Service便會產生UserDetails的物件 |

|  |
| --- |
| 4 建立User |
| 以上流程會幫我們產生UserDetails 進行使用者的比對，這時候我們要產生User物件進行存放。 |

|  |
| --- |
| 5 整個建立的流程大致上這樣… |
| 當我們進入檢 |

## 13使用Spring Security(身分驗證管理器)

|  |
| --- |
| 1 導入Lib |
| 因為使用微服務，所以可以直接在Gradle導入    範例暫時使用2.1.3.RELEASE 版本。  如果非使用微服務，可導入以下Lib |

|  |
| --- |
| 2. 建立SpringSecurity Config類別 |
| 作為一個設定類別，要加上@Configuration  啟動SpringSecurity類別要加上@EnableWebSecurity  繼承WebSecutiryConfigurerAdapter 橋接器 |

|  |
| --- |
| 2.1 建立SpringSecurity Config類別 => 配置身分驗證管理器 |
| 當我們繼承了了WebSecutiryConfigurerAdapter 可以Overrid    配置我們要如何管理身分的驗證。  我們要配置存放這些身分的資訊在記憶體當中(可以存放在不同地方，之後會再研究)    配置編碼方式，Spring Secutiry5之後強制要求密碼需要進行加密，所以最基本的使用方式就是使用PasswrodEncoder 並且使用Spring 官方推薦的BCryptPasswordEncoder 進行加密。  /\*\* 配置密碼的編碼方式 spring secutiry5 之後強制要求密碼需要進行編碼 我們使用Spring 官方推薦的加密演算法方法BCryptPasswordEncoder \*/  PasswordEncoder pwdEncoder = new BCryptPasswordEncoder();  inMemoryUserDetailsManagerConfigurer.passwordEncoder(pwdEncoder);  我們叫建立使用者身分，該範例暫時把身份寫在程式裡。後續會根據不同情況取得身分。 |

|  |
| --- |
| 2.2 整個身分驗證配置大致上是這樣的… |
|  |

|  |
| --- |
| 2.3 配置Http攔截處理(基本做法後續會在研究) |
| 暫時先用以上的程式碼，該部分後續會再討論 |

|  |
| --- |
| 2.4 測試看看吧 |
| 當我們輸入了網址URL(再次聲明，我們使用SpringBoot並且建立了一個RESTController)  會出現Spring 官方提供的登入畫面。    登入後進會成功進入我們的RESTController的RequsetMapping!!! |

## 14使用Spring Security(身分驗證管理器-自定義使用者權限)

|  |
| --- |
| 1 建立SpringSecurity Config類別 => 配置身分驗證管理器(自定義使用者權限) |
| 當我們繼承了了WebSecutiryConfigurerAdapter 可以Override    配置我們要如何管理身分的驗證。  我們要配置存放這些身分的資訊在UserDetail當中，所以可以使用UserDetailsService()，並且該Service會回傳UserDetails這個類別。    當我們建立UserDetails的時候要傳入使用者的名稱(username)並且傳入密碼與權限範圍。  而權限範圍可以是一個Array。  我們也可以自定義使用者權限 |

## 99其他語法與效果

|  |
| --- |
| 1 預設情況之下不啟動Spring Security (暫時沒用，之後研究) |
| 語法為:security.basic.enabled: false |

## 00文獻參考

|  |
| --- |
| URL |
| Spring Security 參考手冊 : <https://www.springcloud.cc/spring-security-zhcn.html> |
| WebSecurityConfigurer 原理說明: <https://blog.csdn.net/andy_zhang2007/article/details/90573895> |
| SecurityConfigurer 原理說明: <https://blog.csdn.net/andy_zhang2007/article/details/89888347> |
| HttpSecurity 原理說明: <https://blog.csdn.net/andy_zhang2007/article/details/90183080> |
| HttpSecurity 產生Filter的方法: <https://blog.csdn.net/Michael_HM/article/details/78964813> |
| AuthenticationManagerBuilder 原理說明: <https://blog.csdn.net/andy_zhang2007/article/details/90665519> |
| 一個完整範例參考: <https://www.cnblogs.com/cjsblog/p/9152455.html> |