Spring Security

第一章 了解 spring security

spring security 是基于 spring 的安全框架。它提供全面的安全性解决方案，同时在 Web 请求级和方法调用级处理身份确认和授权。在 Spring Framework 基础上，spring security 充分利用了依赖注入（DI）和面向切面编程（AOP）功能，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作。是一个轻量级的安全框架。它与 Spring MVC 有很好地集成.

1.1 spring security 核心功能

（1）认证（你是谁，用户/设备/系统）

（2）验证（你能干什么，也叫权限控制/授权，允许执行的操作）

1.2 spring security 原理

基于 Filter , Servlet, AOP 实现身份认证和权限验证

第二章 实例驱动学习

使用的框架和技术

spring boot 2.0.6 版本

spring security 5.0.9 版本

maven 3 以上

jdk8 以上

idea 2019

第一个例子：初探

1.创建 maven 项目

2.加入依赖：spring boot 依赖， spring security 依赖

```
<!--加入 spring boot -->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-parent</artifactId>
    <version>2.0.6.RELEASE</version>
</parent>
<!--web 开发相关依赖-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!--spring security-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

3.创建应用启动类

```
@SpringBootApplication
public class FirstApplication {
    public static void main(String[] args) {
        SpringApplication.run(FirstApplication.class,args);
    }
```

```
}
```

4.创建 Controller，接收请求

```
@RestController
@RequestMapping("/hello")
public class HelloSecurityController {
    @RequestMapping("/world")
     public String sayHello(){
          return "Hello Spring Secuirty  安全管理框架";
     }
}
```

5.框架生成的用户

用户名： user

密码： 在启动项目时，生成的临时密码。uuid

日志中生成的密码：

generated security password: 9717464c-fafd-47b3-9995-2c18b24f7336

6.自定义用户名和密码

需要在 springboot 配置文件中设置登录的用户名和密码

在 resource 目录下面创建 spring boot 配置文件

application.yml(application.properties)

```
spring:
  security:
```

```
    user:
        name: wkcto
        password: wkcto
```

name:自定义用户名称

password：自定义密码

7.关闭验证

//排除 Secuirty 的配置，让他不启用

```
@SpringBootApplication(exclude = {SecurityAutoConfiguration.class})
public class FirstApplication { }
```

第二个例子：使用内存中的用户信息

1）使用：WebSecurityConfigurerAdapter 控制安全管理的内容。

需要做的使用：继承 WebSecurityConfigurerAdapter，重写方法。实现自定义的认证信息。重写下面的方法。

protected void configure(AuthenticationManagerBuilder auth)

2）spring security 5 版本要求密码比较加密，否则报错

java.lang.IllegalArgumentException: There is no PasswordEncoder mapped for the id "null"

实现密码加密：

1）创建用来加密的实现类（选择一种加密的算法）

```
@Bean
public PasswordEncoder passwordEncoder(){
    //创建 PasawordEncoder 的实现类， 实现类是加密算法
    return new BCryptPasswordEncoder();
}
```

2）给每个密码加密

```
PasswordEncoder pe = passwordEncoder();
pe.encode("123456")
```

注解：

1. @Configuration ：表示当前类是一个配置类(相当于是 spring 的 xml 配置文件），在这个类方法的返回值是 java 对象，这些对象放入到 spring 容器中。

2. @EnableWebSecurity：表示启用 spring security 安全框架的功能

3. @Bean：把方法返回值的对象，放入到 spring 容器中。

第三个例子：基于角色 Role 的身份认证， 同一个用户可以有不同的角色。同时可以开启对方法级别的认证。

基于角色的实现步骤：

1.设置用户的角色

继承 WebSecurityConfigurerAdapter

重写 configure 方法。指定用户的 roles

```
auth.inMemoryAuthentication()
        .withUser("admin")
        .password(pe.encode("admin"))
        .roles("admin","normal");
```

2.在类的上面加入启用方法级别的注解

```
@EnableGlobalMethodSecurity(prePostEnabled = true)
```

3.在处理器方法的上面加入角色的信息，

指定方法可以访问的角色列表

```
//指定 normal 和 admin 角色都可以访问的方法
@RequestMapping("/helloUser")
@PreAuthorize(value = "hasAnyRole('admin','normal')")
public String helloCommonUser(){
    return "Hello 拥有 normal, admin 角色的用户";
}
```

使用@PreAuthorize 指定在方法之前进行角色的认证。

hasAnyRole('角色名称 1','角色名称 N')

第四个例子，基于 jdbc 的用户认证。

从数据库 mysql 中获取用户的身份信息（用户名称，密码，角色）

1）在 spring security 框架对象用户信息的表示类是 UserDetails.

UserDetails 是一个接口，高度抽象的用户信息类（相当于项目中的 User 类）

User 类：是 UserDetails 接口的实现类， 构造方法有三个参数：

username，password, authorities

需要向 spring security 提供 User 对象， 这个对象的数据来自数据库的查询。

2）实现 UserDetailsService 接口，

重写方法 UserDetails loadUserByUsername(String var1)

在方法中获取数据库中的用户信息， 也就是执行数据库的查询，条件是用户名称。