harrychinese 刘忠武
面朝大海, 春暖花开
我的开源
http://anydbtest.codeplex.com

昵称：harrychinese
园龄：8年4个月
粉丝：62
关注：12
+加关注

< 2019年3月 >

日	一	二	三	四	五	六
24	25	26	27	28	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

最新随笔
1. [转载] 历史上前端领域的重要技术革命
2. 前后端要不要分离以及如何做
3. 微电子工艺基础知识讲解(集成电路历史/厂商/产业链)
4. 半导体封装工艺流程简介
5. 半导体知识：蚀刻（Etch）工艺讲解

Spring Security 之方法级的安全管控

默认情况下, Spring Security 并不启用方法级的安全管控. 启用方法级的管控后, 可以针对不同的方法通过注解设置不同的访问条件.
Spring Security 支持三种方法级注解, 分别是 JSR-205 注解/@Secured 注解/prePostEnabled注解. **这些注解不仅可以直接加 controller 方法上, 也可以注解 Service 或 DAO 类中的方法.**

启用方法级的管控代码是, 新建一个 WebSecurityConfigurerAdapter Configuration 类, 加上 @EnableGlobalMethodSecurity() 注解, 通过@EnableGlobalMethodSecurity 参数开启相应的方法级的管控.

==========
JSR-205 注解
==========
通过 @EnableGlobalMethodSecurity(jsr250Enabled=true), 开启 JSR-205 注解.

@DenyAll 注解, 拒绝所有的访问
@PermitAll 注解, 运行所有访问
@RolesAllowed({"USER","ADMIN"}), 该方法只允许有 ROLE_USER 或 ROLE_ADMIN 角色的用户访问.

==========
@Secured 注解
==========
通过 @EnableGlobalMethodSecurity(securedEnabled=true), 开启 @Secured 注解.
只有满足角色的用户才能访问被注解的方法, 否则将会抛出 AccessDenied 异常.
例子:
@Secured("ROLE_TELLER","ROLE_ADMIN"), 该方法只允许 ROLE_TELLER 或 ROLE_ADMIN 角色的用户访问.
@Secured("IS_AUTHENTICATED_ANONYMOUSLY"), 该方法允许匿名用户访问.

==========
@PreAuthorize 类型的注解(支持 Spring 表达式)
==========
@EnableGlobalMethodSecurity(prePostEnabled=true), 开启 prePostEnabled 相关的注解.
JSR-205 和 @Secured 注解功能较弱, 不支持 Spring EL 表达式. 推荐使用 @PreAuthorize 类型的注解.
具体有4个注解.
@PreAuthorize 注解, 在方法调用之前, 基于表达式结果来限制方法的使用.
@PostAuthorize 注解, 允许方法调用, 但是如果表达式结果为 false, 将抛出一个安全性异常.
@PostFilter 注解, 允许方法调用, 但必要按照表达式来过滤方法的结果.
@PreFilter 注解, 允许方法调用, 但必须在进入方法之前过来输入值.

例子:

```
@PreAuthorize("hasRole('ADMIN')") //必须有 ROLE_ADMIN 角色
public void addBook(Book book);

//必须同时具备 ROLE_ADMIN 和 ROLE_DBA 角色
@PreAuthorize("hasRole('ADMIN') AND hasRole('DBA')")
public void addBook(Book book);
```

6. 一文解析刻蚀机和光刻机的原理及区别

7. 一文看懂光刻机

8. 一文看懂国产芯片现状

9. 集成电路产业链全景图

10. 半导体知识讲解：IC基础知识及制造工艺流程

我的标签

工具(74)

Java(66)

SpringBoot(66)

Python(60)

数据仓库(53)

大数据(29)

半导体行业(15)

Docker(11)

ETL(11)

AnyDbTest(10)

更多

随笔档案

2019年3月 (2)

2019年2月 (18)

2019年1月 (15)

2018年12月 (13)

2018年11月 (18)

2018年10月 (20)

2018年9月 (10)

2018年8月 (3)

2018年7月 (6)

2018年6月 (6)

```
@PreAuthorize ("#book.owner == authentication.name")
public void deleteBook(Book book);


@PostAuthorize ("returnObject.owner == authentication.name")
public Book getBook();
```

==========
@PreAuthorize 表达式
==========

1. returnObject 保留名

对于 @PostAuthorize 和 @PostFilter 注解, 可以在表达式中使用 returnObject 保留名, returnObject 代表着被注解方法的返回值, 我们可以使用 returnObject 保留名对注解方法的结果进行验证.

比如:

```
@PostAuthorize ("returnObject.owner == authentication.name")
public Book getBook();
```

2. 表达式中的 # 号

在表达式中, 可以使用 #argument123 的形式来代表注解方法中的参数 argument123.

比如:

```
@PreAuthorize ("#book.owner == authentication.name")
public void deleteBook(Book book);
```

还有一种 #argument123 的写法, 即使用 Spring Security @P注解来为方法参数起别名, 然后在 @PreAuthorize 等注解表达式中使用该别名. 不推荐这种写法, 代码可读性较差.

```
@PreAuthorize("#c.name == authentication.name")
public void doSomething(@P("c") Contact contact);
```

3. 内置表达式有:

表达式	备注
hasRole([role])	如果有当前角色, 则返回 true(会自动加上 ROLE_ 前缀)
hasAnyRole([role1, role2])	如果有任一角色即可通过校验, 返回true, (会自动加上 ROLE_ 前缀)
hasAuthority([authority])	如果有指定权限, 则返回 true
hasAnyAuthority([authority1, authority2])	如果有任一指定权限, 则返回true
principal	获取当前用户的 principal 主体对象
authentication	获取当前用户的 authentication 对象,
permitAll	总是返回 true, 表示全部允许
denyAll	总是返回 false, 代表全部拒绝
isAnonymous()	如果是匿名访问, 返回true
isRememberMe()	如果是remember-me 自动认证, 则返回 true
isAuthenticated()	如果不是匿名访问, 则返回true
isFullAuthenticated()	如果不是匿名访问或remember-me认证登陆, 则返回true
hasPermission(Object target, Object	

2018年5月 (12)
2018年4月 (9)
2018年3月 (9)
2018年2月 (1)
2017年12月 (2)
2017年11月 (2)
2017年10月 (4)
2017年9月 (1)
2017年8月 (1)
2017年7月 (1)
2017年4月 (1)
2017年2月 (1)
2017年1月 (1)
2016年12月 (1)
2016年10月 (1)
2016年9月 (2)
2016年3月 (1)
2016年2月 (2)
2016年1月 (3)
2015年11月 (1)
2015年10月 (3)
2015年9月 (1)
2015年8月 (3)
2015年7月 (1)
2015年6月 (2)
2015年3月 (1)
2015年2月 (1)
2015年1月 (1)
2014年9月 (1)

permission)	
hasPermission(Object target, String targetType, Object permission)	

==========
示例代码
==========

---
pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-freemarker</artifactId>
</dependency>
```

SecurityConfig 配置类

SecurityConfig 配置类开启方法级管控(仅启用 prePostEnabled 类的注解), 并 hard coded 了一个内置的用户清单.

因为没有重载 configure(HttpSecurity http) 方法, 用的是Spring security 的 basic Authentication 和内置的login form.

```
@EnableWebSecurity
@Configuation
@EnableGlobalMethodSecurity(prePostEnabled=true)
public class SecurityConfig extends WebSecurityConfigurerAdapter{

    @Override
    public void configure(AuthenticationManagerBuilder builder)
            throws Exception {
        builder.inMemoryAuthentication()
                .withUser("123").password("123").roles("USER")
                .and()
                .withUser("ADMIN").password("ADMIN").roles("ADMIN")
                .and()
                .withUser("124").password("124").roles("USER2");
    }

    @SuppressWarnings("deprecation")
    @Bean
    public NoOpPasswordEncoder passwordEncoder() {
```

2014年8月 (1)
2014年7月 (1)
2014年6月 (1)
2014年5月 (3)
2014年4月 (2)
2014年3月 (3)
2014年2月 (4)
2014年1月 (2)
2013年11月 (3)
2013年10月 (3)
2013年7月 (5)
2013年6月 (3)
2013年3月 (1)
2012年12月 (2)
2012年11月 (2)
2012年10月 (5)
2012年9月 (1)
2012年8月 (2)
2012年4月 (4)
2012年2月 (2)
2012年1月 (5)
2011年12月 (1)
2011年11月 (6)
2011年10月 (4)
2011年9月 (2)
2011年8月 (17)
2011年7月 (2)
2011年6月 (4)
2011年5月 (5)

```
        return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();
    }
}
```

BookService 配置类

为BookService Service类的方法加上 @PreAuthorize 注解, 对权限进行控制.

```
@Service
class BookService{
    @PreAuthorize("hasRole('ADMIN')")
    public void addBook(Book book) {
        System.out.println("you have added a book successfully");
    }

    @PreAuthorize("hasAnyRole('ADMIN','USER')")
    public Book getBook() {
        Book book=new Book("A");
        return book ;
    }

    @PreAuthorize("hasRole('ADMIN')")
    public void deleteBook(Book book) {
        System.out.println("Book deleted");
    }
}
```

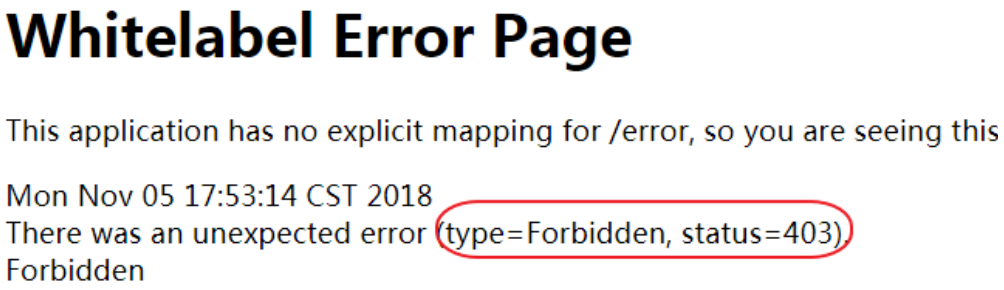
Entity 和 Controller 类

Entity 和 Controller 类没有特别之处, 这里不用太关注.

```
View Code
```

测试结果

使用 123 用户(角色是 USER)登陆, 可以访问 http://localhost:8080/get ; 使用 124 用户(角色是 USER2)登陆, 访问 http://localhost:8080/get 报下面的403权限问题. 符合预期



==========

参考

==========

https://www.concretepage.com/spring/spring-security/preauthorize-postauthorize-in-

2011年4月 (2)
2011年3月 (8)
2011年2月 (3)
2011年1月 (8)
2010年12月 (5)
2010年11月 (1)

personal

open source AnyDbTest on codeplex
PetterLiu blog
博客同步
我的技术书签

阅读排行榜

1. 修改python默认的编码方式(19353)
2. SqlAlchemy个人学习笔记完整汇总(19272)
3. 系统研究Airbnb开源项目airflow(10331)
4. window下使用virtualenv(8983)
5. MyBatis 学习笔记(7403)

spring-security
https://www.jianshu.com/p/bcb3c6445b2b
https://www.jianshu.com/p/41b7c3fb00e0
https://blog.csdn.net/l18767118724/article/details/72934564
https://spring.io/guides/topicals/spring-security-architecture/
https://www.logicbig.com/tutorials/spring-framework/spring-security/roles-allowed-annotation.html

标签: SpringBoot, Java, SpringSecurity

harrychinese
关注 - 12
粉丝 - 62
+加关注

« 上一篇: SpringBoot系列: 使用 Swagger 生成 API 文档
» 下一篇: Spring Security 之API 项目安全验证(基于basic-authentication)

posted @ 2018-11-05 15:00 harrychinese 阅读(120) 评论(0) 编辑 收藏

相关博文:
· Spring Security
· SpringSecurity
· Spring Security学习笔记
· spring security总结
· spring security控制权限的几种方法