

http://www.nit.net.cn/__local/C/EE/E5/AEF522EFB83938A8BD165923DEC_001BEC5E_9976.png

**毕业论文（设计）开题报告**

（含文献综述、外文翻译）



题 目 基于SpringSecurity安全框架的权限管理系统

姓 名 黄广进

学 号 3180421071

专业班级 计算机184

指导教师 宋光慧

学 院 计算机与数据工程院

日 期 2022/2/9

目录

一、 文献综述 1

1 背景介绍 1

2 国内外研究现状 1

2.1 研究方向及进展 1

2.2 存在问题 1

3 发展趋势和研究展望 1

4 参考文献 2

二、 开题报告 3

1 选题背景和意义 3

1.1 选题背景 3

1.2 选题意义 3

2 项目内容及可行性分析 4

3 研究方法 4

4 拟解决的关键问题 4

5 预期结果 4

6 实施进度计划 5

7 参考文献 5

三、 外文翻译 6

(二) 架构与实现 6

9 技术概述 6

四、 外文原文 14

# 一、 文献综述

## 背景介绍

在基于 B/S 结构的 J2EE 企业级应用系统中， 客户登陆 认证、 访问授权控制是重要组成部分，Spring Security 提供了 一个较成熟的安全框架。 Spring Security 是由 Acegi Security 发展而来的，是 Spring 框架的组成部分，Spring Security 框架主 要 提供“认 证”和“授 权访问”，“认证”为用户建立一个他所需要声明的主题，即用 户或在你系统中执行的其他设备或系统。 另一个是“授权”， 授权是指已经提前建立好了某个用户被授予的权利，能否执 行 某 个 操 作， 在 执 行 之 前 就 已 经 进 行 了 授 权 判 断 。 Spring Security 也支持大量的第三方验证模型。 Spring Security 通过使用 Servlet 过滤器对 Web 安全性进 行支持，过滤器过 滤 Servlet 的访 问 请 求，将 这 些 请求 转 发 给 认证授权管理决策处理器进行安全处理，从而曾强了系统的 访问安全性。 每一个请求发出后，至少要经过以下 4 个过滤 器。

## 国内外研究现状

### 2.1 研究方向及进展

软件体系结构的传统单片方法要求整个应用程序堆栈为每个部署捆绑在一起。这个概念给应用程序带来了许多缺点，尤其对分布式团队来说，不灵活的可伸缩性、高昂的资源成本和重构工作带来了许多困扰。

所以对于微服务架构（MSA）应该通过以下方式解决这些问题：将应用程序分解为独立的服务；每项服务负责一项业务能力，并独立部署和执行。应用程序通过网络通信协议和Internet相互通信，因此这种架构风格在很大程度上依赖于应用程序编程接口（API）。鉴于此，微服务应用程序中的API需要得到适当的保护，以保护应用程序和应用程序，使它的资源可以抵御处理API调用的威胁。

研究通过开发一个安全性证明来减少MSA和API安全性方面的知识差距。使用Spring框架、SpringSecurity和OAuth2的MSA应用程序的概念（POC），然后执行。通过POC使用单元测试和手动测试技术进行安全测试。

Spring Security 是一个高度自定义的安全框架。在保护API的同时，利用 Spring IoC/DI和 AOP 功能，为系统提供了声明式安全访问控制功能，减少了为系统安全而编写大量重复代码的工作。

### 2.2 存在问题

相比于别的安全框架，例如Shiro，Spring Security无法独立运行，需要依赖于Spring容器。

## 发展趋势和研究展望

微服务架构是 SOA 架构思想的一种扩展，更加强调服务个体的独立性、拆分粒度更小。很多公司在业务高速发展的时候，服务组件也会相应的不断增加。微服务就很好的解决了服务与服务之间的解耦。

在而在MSA应用程序（微服务架构）中，微服务通过服务API端点相互通信。在里面换句话说，API端点是服务可以访问并获得所需资源的位置，履行他们的职责。API端点在保证应用程序的正确运行方面起着至关重要的作用。与之交互的服务和系统，是相互之间交换数据的接口服务。

因此，API端点的安全性是MSA应用程序中的关键安全因素之一。这个对现有文献和相关工作的回顾发现，目前还没有实验结果来证实这一结论当使用这些框架和安全标准来保护MSA体系结构中的API端点。因此，本研究实施了库存的POC管理系统使用MSA作为实验系统来检查技术。

## 参考文献

[1]Alex B, Taylor L, Winch R, et al. Spring Security Reference[J]. URL https://docs. spring. io/springsecurity/site/docs/current/reference/htmlsingle/.[utoljára megtekintve: 2017. 04. 21.], 2004, 12.

[2]Nguyen Q, Baker O F. Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API[J]. J. Softw., 2019, 14(6): 257-264.

[3]Kritikos, K., & Massonet, P. (2016). An integrated meta-model for cloud application security modelling. Procedia Computer Science, 84-93.

[4]Nguyen Q, Baker O F. Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API[J]. J. Softw., 2019, 14(6): 257-264.

[5]陈启航, 崔鑫. 基于 J2EE/Spring Security 的用户权限管理与实现[J].福建电脑, 2010 (12): 137-138.

[6]李琼. 基于 Spring Security 的企业级应用安全架构的研究与实现[D].北京交通大学, 2012.

[7]梁锋.基于EGR-RBAC和SpringSecurity框架的访问控制模型研究[D].合肥工业大学,2013.

[8]郑义平.基于RBAC的通用权限管理设计与实现[J].金融科技时代,2017 (4): 43-47.

[9]赖万钦. 基于spring security 3框架的用户RBAC权限设计应用[J].福建电脑, 2012, 28(10): 141-142.

# 二、 开题报告

## 1 选题背景和意义

### 选题背景

在实际开发的应用系统中，为了安全起见，一般都必备用户认证（登录）和权限控制的功能，以识别用户是否合法，以及根据权限来控制用户是否能够执行某项操作。

### 选题意义

SpringSecurity是基于Spring AOP和Servlet过滤器的安全框架。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，Dependency Injection）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。它是一个轻量级的安全框架，它确保基于Spring的应用程序提供身份验证和授权支持。它与Spring MVC有很好地集成，并配备了流行的安全算法实现捆绑在一起。安全主要包括两个操作“认证”与“验证”(有时候也会叫做权限控制)。“认证”是为用户建立一个其声明的角色的过程，这个角色可以一个用户、一个设备或者一个系统。“验证”指的是一个用户在你的应用中能够执行某个操作。在到达授权判断之前，角色已经在身份认证过程中建立了。

目前常用的安全框架主要是Spring Security和Apache Shiro，而相比于Shiro，Spring Security基于Spring开发，项目中如果使用Spring作为基础，配合Spring Security做权限更加方便。而Shiro需要和Spring进行整合；功能比Shiro更加丰富，例如安全防护方面；社区资源相对比Shiro更加丰富；如果使用的是Spring Boot，Spring Cloud的话，三者可以无缝集成。

## 2 项目内容及可行性分析

项目内容:

1. 主要包括将Spring Security整合到Spring boot框架当中，实现按钮级别的用户权限管理。
2. 前端使用AntDesgin Pro框架，实现基于用户，部门，分组，角色以及权限的管理系统。
3. 基于用户权限实现菜单、按钮粒度的权限控制，并用一个业务模块进行演示。

可行性分析：

设计以Spring boot项目为基础，将Spring Security整合到其中，主要实现认证和授权两个功能。

1. 首先需要搭建一个前后端分离的简单SpringBoot的项目。
2. 构建合理的数据库表，包括user，menu，permission等关键表。
3. 配置开启Spring Security。
4. 前端使用AntDesgin Pro框架做一个简单的页面展示。

## 3 研究方法

研究方法主要使用文献法，主要通过阅读一些spring security的官方文档以及一些优秀的个人文章，掌握spring security的相关知识以及正确配置使用，从而达到研究目的。

## 4 拟解决的关键问题

需要在不影响原先应用正常功能的前提下，将Spring Security整合到其中去，并开启授权和认证功能。在安全的前提下有效实现权限管理的功能。

## 预期结果

能够有效的阻止未授权的用户登录访问未开放的页面，同时基于用户权限实现菜单、按钮粒度的权限控制。

## 实施进度计划

* 1. 2021/1/7-2022/1/17 确定毕业设计的内容以及实现方法
  2. 2022/1/18-2022/2/1 文献阅读掌握了解spring security
  3. 2022/2/2-2022/2/7 整合spring security 到spring boot项目中去

## 参考文献

[1]蒋丛萃,史卓.Spring security在电子商务中的设计应用[J].电脑知识与技术,2016,12(14):247-248.DOI:10.14004/j.cnki.ckt.2016.1773.

[2]周常志,甘恒.基于SpringBoot的智慧就业服务平台的设计与实现[J].电脑知识与技术,2021,17(28):182-184+187.DOI:10.14004/j.cnki.ckt.2021.3040.

[3]梁锋.基于EGR-RBAC和SpringSecurity框架的访问控制模型研究[D].合肥工业大学,2013.

[4]董一璠,李宝林,何先波.基于Spring Security安全框架的联通资源管理系统安全分析[J].电子设计工程,2013,21(08):61-64.DOI:10.14022/j.cnki.dzsjgc.2013.08.046.

[5]孙恩斯.Spring Security安全框架应用研究[J].信息系统工程,2019(03):72.

[6]周文红,晏素芬,蒋玉芳,邓朝晖.Spring Security安全框架应用[J].计算机与现代化,2013(11):88-90.

[7]胡安明.Spring安全框架在信息系统中的应用研究[J].广东技术师范学院学报,2012,33(12):15-17.

[8]赛序朋,侯瑞春,丁香乾.Spring Security的Web资源保护功能研究与扩展[J].微型机与应用,2015,34(15):5-7.DOI:10.19358/j.issn.1674-7720.2015.15.002.

[9]廖先琴.使用Spring Security开发安全的Java程序[J].河南科技,2014(09):1.

[10]赖万钦.基于spring security 3框架的用户RBAC权限设计应用[J].福建电脑,2012,28(10):141-142+114.

# 三、 外文翻译

Spring Security 参考手册

Ben Alex , Luke Taylor , Rob Winch , Gunnar Hillert

摘要：Spring Security 为基于javaEE的企业应用程序提供一个全面的解决方案。正如你将从这个参考指南发现的，我们试图为你提供一个有用的并且高度可配置的安全系统。

## 架构与实现

### 技术概述

#### 9.1 运行环境

Spring Security 3.0需要Java 5.0的运行环境或者更高的版本. 由于 Spring Security 是以独立的方式运作, 就不需要什么特殊的配置文件到你的Java运行环境。特别是, 不需要配置专门的Java认证和授权服务(JAAS)策略文件或将Spring Security的位置放到普通路径中。

同样，如果你使用的是EJB容器或者Servlet容器也没有必要把任何特殊的配置文件放到任何地方，也不包括Spring Security的服务器类加载器。所有必须的文件都将包含在你的应用程序中。

这种设计给部署时间提供了最大的灵活性，你可以简单的复制你的目标文件(可以是JAR, WAR或者EAR)从一个系统到另一个系统，它会立即开始工作。

#### 9.3 验证

Spring Security可以在很多不同的认证环境下使用。虽然我们推荐人们使用Spring Security，不与已存在的容器管理认证系统结合，但它也是支持的-使用你自己的属性验证系统进行整合。

什么是Spring Security验证?让我们考虑一个大家都很熟悉的标准的验证场景。

1. 提示用户输入用户名和密码进行登录。
2. 该系统 (成功) 验证该用户名的密码正确。
3. 获取该用户的环境信息 (他们的角色列表等).
4. 为用户建立安全的环境。
5. 用户进行，可能执行一些操作，这是潜在的保护的访问控制机制，检查所需权限，对当前的安全的环境信息的操作。

前三个项目构成的验证过程，所以我们将看看这些是如何发生在Spring Security中的。

1. 用户名和密码进行组合成一个实例UsernamePasswordAuthenticationToken (一个Authentication接口的实例, 我们之前看到的).
2. 令牌传递到AuthenticationManager实例进行验证。
3. 该AuthenticationManager完全填充Authentication实例返回成功验证。
4. 安全环境是通过调用 SecurityContextHolder.getContext().setAuthentication(…​), 传递到返回的验证对象建立的。

从这一点上来看，用户被认为是被验证的。让我们看看一些代码作为一个例子: 如图9.1所示



图9.1

在这里我们已经写了一个小程序，要求用户输入一个用户名和密码并执行上述序列。这个AuthenticationManager我们这里将验证用户的用户名和密码将其设置成一样的，它给每一个用户分配一个单一的角色。从上面输出的将是类似的东西:

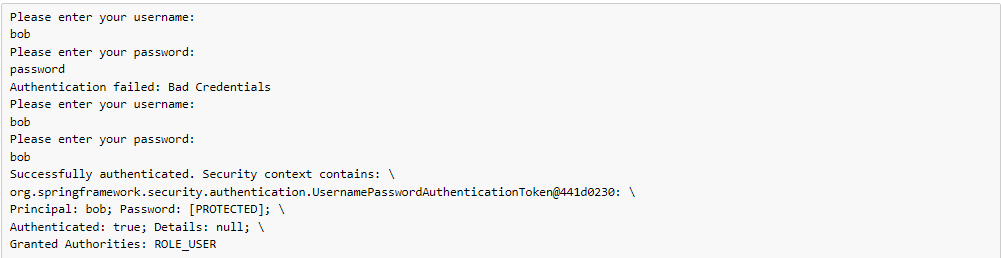


图9.2

请注意，你通常不需要写任何这样的代码。这个过程通常会发生在内部，以一个web认证过滤器为例，我们刚刚在这里的代码显示，在Spring Security中究竟是什么构成了验证的问题，有一个相对简单的答案。用户验证时，SecurityContextHolder包含一个完全填充的Authentication对象的用户进行身份验证。

直接设置SecurityContextHolder的内容

事实上，Spring Security不介意你如何把Authentication对象包含在SecurityContextHolder内。唯一的关键要求是SecurityContextHolder包含Authentication在AbstractSecurityInterceptor之前(我们会看到更多的版本)需要用户授权操作。

你可以(很多用户都这样做)写一个自己的过滤器或MVC控制器来提供验证系统的交互，这些都不是基于Spring Security的。比如，你也许使用容器管理认证，从ThreadLocal或JNDI里获得当前用户信息。或者，你的公司可能有一个遗留系统，它是一个企业标准，你不能控制它。这种情况下，很容易让Spring Security工作，也能提供验证能力。你所需要的就是写一个过滤器(或等价物)从指定位置读取第三方用户信息，把它放到SecurityContextHolder里。在这种情况下，你还需要考虑的事情通常是由内置的认证基础设施自动照顾。例如，缓存请求的情况下你可能需要事先创建一个HTTP会话，在编写客户端响应之前 。

如果你想知道AuthenticationManager是如何以现实世界的例子来实现，我们可以来看看 核心服务一章.

#### 9.5 Spring Security的访问控制(授权)

负责Spring Security访问控制决策的主要接口是AccessDecisionManager。它有一个decide方法，它需要一个Authentication对象请求访问,一个"secure object"(见下文)和安全元数据属性的列表适用的对象(如一个列表哪些角色需要被访问授权)。

1. 安全和AOP建议

如果你熟悉AOP的话，就会知道有几种不同的拦截方式：之前，之后，抛异常和环绕。 其中环绕是非常有用的，因为advisor可以决定是否执行这个方法，是否修改返回的结果，是否抛出异常。 Spring Security为方法调用提供了一个环绕advice，就像web请求一样。 我们使用Spring的标准AOP支持制作了一个处理方法调用的环绕advice，我们使用标准Filter建立了对web请求的环绕advice。

对那些不熟悉AOP的人，需要理解的关键问题是Spring Security可以帮助你保护方法的调用，就像保护web请求一样。大多数人对保护服务层里的安全方法非常按兴趣。这是因为在目前这一代J2EE程序里，服务器放了更多业务相关的逻辑。如果你只是需要保护服务层的方法调用，Spring标准AOP平台就够了。如果你想直接保护领域对象，你会发现AspectJ非常值得考虑。

可以选择使用AspectJ还是SpringAOP处理方法验证，或者你可以选择使用filters处理web请求验证。 你可以不选，选择其中一个，选择两个，或者三个都选。主流的应用是处理一些web请求验证，再结合一些在服务层里的Spring AOP方法调用验证。

1. 安全对象和AbstractSecurityInterceptor

那么什么是一个"安全对象"呢?Spring Security使用术语是指可以有安全性的任何对象(如授权决策)应用于它。最常见的例子就是方法调用和web请求。

Spring Security支持的每个安全对象类型都有它自己的类型,他们都是AbstractSecurityInterceptor的子类。很重要的是,如果主体是已经通过了验证,在AbstractSecurityInterceptor被调用的时候,SecurityContextHolder将会包含一个有效的Authentication。

AbstractSecurityInterceptor提供了一套一致的工作流程，来处理对安全对象的请求，通常是:

查找当前请求里分配的"配置属性"。

把安全对象，当前的Authentication和配置属性,提交给AccessDecisionManager来进行以此认证决定。

有可能在调用的过程中,对Authentication进行修改。

允许安全对象进行处理（假设访问被允许了）。

在调用返回的时候执行配置的AfterInvocationManager。如果调用引发异常,AfterInvocationManager将不会被调用。

1. 配置属性是什么?

一个"配置属性"可以看做是一个字符串,它对于AbstractSecurityInterceptor使用的类是有特殊含义的。它们由框架内接口ConfigAttribute表示。它们可能是简单的角色名称或拥有更复杂的含义,这就与AccessDecisionManager实现的先进程度有关了。AbstractSecurityInterceptor和配置在一起的 SecurityMetadataSource 用来为一个安全对象搜索属性。通常这个属性对用户是不可见的。配置属性将以注解的方式设置在受保护方法上，或者作为受保护URLs的访问属性。例如,当我们看到像<intercept-url pattern='/secure/\*\*' access='ROLE\_A,ROLE\_B'/>命名空间中的介绍,这是说配置属性ROLE\_A和ROLE\_B适用于匹配Web请求的特定模式。在实践中,使用默认的AccessDecisionManager配置, 这意味着,任何人谁拥有GrantedAuthority只要符合这两个属性将被允许访问。严格来说,它们只是依赖于AccessDecisionManager实施的属性和解释。使用前缀ROLE\_是一个标记,以表明这些属性是角色,应该由Spring Security的RoleVoter前缀被消耗掉。这只是使用AccessDecisionManager的选择基础。我们将在授权章看到AccessDecisionManager是如何实现的。

1. RunAsManager

假设AccessDecisionManager决定允许执行这个请求,AbstractSecurityInterceptor会正常执行这个请求。话虽如此，罕见情况下，用户可能需要把SecurityContext的Authentication换成另一个Authentication, 这是由AccessDecisionManager 调用RunAsManager。这也许在,有原因,不常见的情况下有用,比如服务层方法需要调用远程系统表现不同的身份。 因为Spring Security自动传播安全身份，从一个服务器到另一个（假设你使用了配置好的RMI或者HttpInvoker远程调用协议客户端），就可以用到它了。

1. AfterInvocationManager

按照下面安全对象执行和返回的方式-可能意味着完全的方法调用或过滤器链的执行-在AbstractSecurityInterceptor得到一个最后的机会来处理调用。这种状态下AbstractSecurityInterceptor对有可能修改返回对象感兴趣。你可能想让它发生，因为验证决定不能“关于如何在”一个安全对象调用。高可插拔性,AbstractSecurityInterceptor通过控制AfterInvocationManager在实际需要的时候修改对象。这里类实际上可能替换对象，或者抛出异常，或者什么也不做。如果调用成功后，检查调用才会执行。如果出现异常，额外的检查将被跳过。

AbstractSecurityInterceptor 和它的相关对象如图9.3所示

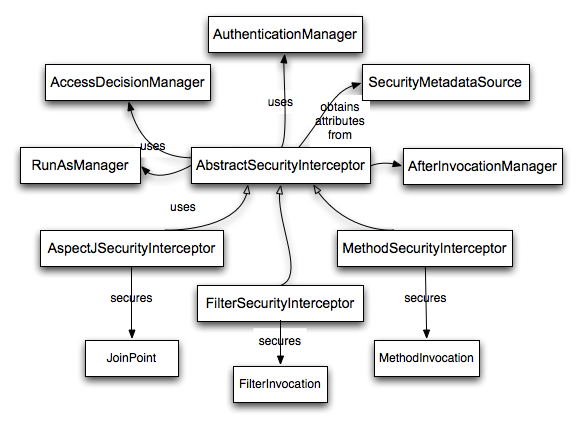


图 9.3 安全拦截器和“安全对象”模型

只有当开发人员考虑一个全新的拦截方法和授权请求时才需要直接使用安全对象。例如，为了确保对消息系统的调用，它有可能建立建立一个新的安全对象。任何东西都需要安全，并且还提供了一种方法去调用（如建议语义的AOP）能够被做成一个安全对象。不得不说的是，大多数Spring应用程序将只使用三种目前完全支持的安全对象类型(AOP Alliance MethodInvocation, AspectJ JoinPoint和web请求FilterInvocation)。

#### 9.6 Localization

Spring Security支持终端用户看到异常消息的本地化。如果你的应用程序是专为讲英语的用户设计的，你不需要做任何事情，因为默认所有的安全信息都是英文的，如果你需要支持其他地方，你需要知道的一切都被包含在这部分。

所有异常消息都可以本地化，包括有关验证失败和访问被拒绝（授权失败）的消息。这主要集中在开发者和系统发布（包括不正确的属性，接口违反合同，使用不正确的构造器，开始验证，日志调试等级）异常和日志消息没有本地化，而是使用硬编码的Spring Security的英文代码。

在spring-security-core-xx.jar的运输中你会发现一个org.springframework.security 包含了 messages.properties文件,以及一些常用版本的本地化语言。这应该是你的ApplicationContext,因为Spring Security实现了Spring的MessageSourceAware界面，希望这些消息是依赖于应用程序上下文启动的时候注入。通常你需要做的是创建你的应用程序上下文参考消息里面的bean。一个例子如下所示：

<bean id="messageSource"

class="org.springframework.context.support.ReloadableResourceBundleMessageSource">

<property name="basename" value="classpath:org/springframework/security/messages"/>

</bean>

该messages.properties是按照标准的资源束命名方式，为Spring Security的消息所支持的默认语言。这个默认的文件是英文的。

如果您希望自定义messages.properties文件，或支持其他语言，您应该复制该文件，相应地重命名它，并在上面的bean定义中注释它。在这个文件中没有大量的消息密钥，因此本地化不应该被认为是一个重大举措。如果你对这个文件执行定为操作，请考虑与社区分享你的工作通过记录JIRA任务和附加被你恰当命名的messages.properties本地化版本。

Spring Security依赖于Spring"s的本地化支持，以实际查找适当的消息。为了这项工作，你必须确保从传入请求的区域存储在Spring’s`org.springframework.context.i18n.LocaleContextHolder`。Spring MVC的` DispatcherServlet 会自动为你的程序做，但因为Spring Security的过滤器在那之前被调用，`localecontextholder需在过滤器被呼叫之前建立在包含正确的Locale里。你也可以在你自己的过滤器里面做这个（必须做完这项在Spring Security的web.xml过滤之前）或者你可以使用Spring的RequestContextFilter。请参阅Spring Framework文档，以进一步详细说明使用Spring定位。

“联系人”示例应用程序设置为使用本地化消息。

# 四、 外文原文

