文章编号: 1000-5641(2018)03-0109-12

研究生信息平台中权限管理的设计与实现

顾 航[1], 夏 帆[1], 宋树彬[2], 肖李敏[2],
董启文[1], 徐林昊[3], 周傲英[1]

(1. 华东师范大学 数据科学与工程学院, 上海 200062;
2. 华东师范大学 研究生院, 上海 200062;
3. 印孚瑟斯技术中国有限公司, 上海 200135)

摘要: 认证与授权是保障软件系统中数据与服务安全的重要机制. 在研发下一代华东师范大学研究生院信息管理系统的过程中, 针对新的业务需求 (如学籍异动管理和预毕结业审核等流程审批管理), 设计了一种基于访问域模型的权限管理模块, 结合 Spring Security 组件实现了一种多层级、可配置、高性能的权限拦截器, 有效地解决了用户认证与授权问题. 该权限管理模块在实际使用中不会给 Web 服务请求带来响应延迟问题, 能够防御常见的网络攻击 (如会话攻击或跨域请求伪造), 而且可以灵活地满足华东师范大学各个院系和职能部门在使用研究生院信息平台过程中经常发生的用户授权变更需求.

关键词: 授权; 权限管理; 访问域模型
中图分类号: TP315 **文献标志码**: A **DOI:** 10.3969/j.issn.1000-5641.2018.03.012

Design and implementation of an authorization system for a graduate school information

GU Hang[1], XIA Fan[1], SONG Shu-bin[2], XIAO Li-min[2], DONG Qi-wen[1],
XU Lin-hao[3], ZHOU Ao-ying[1]

(1. *School of Data Science and Engineering, East China Normal University, Shanghai* 200062, *China*;
2. *Graduate School, East China Normal University, Shanghai* 200062, *China*;
3. *Infosys Technologies China Ltd, Shanghai* 200135, *China*)

Abstract: Authentication and authorization are critical to ensuring the security of data and services in software systems. To satisfy the need for authorization management during the development of the next generation information platform for East China Normal University's Graduate School, this paper proposes an access domain-based authorization module and uses Spring Security components to implement a hierarchical, configurable,

收稿日期: 2017-09-19
基金项目: 国家重点研发计划 (2016YFB1000905); 国家自然科学基金广东省联合重点项目(U1401256); 国家自然科学基金(61672234, 61402177); 华东师范大学信息化软科学研究课题 (41600-10201-562940/008)
第一作者: 顾 航, 男, 硕士研究生, 研究方向为数据科学技术应用. E-mail: 553778439@qq.com.
通信作者: 夏 帆, 男, 博士后, 研究方向为社交媒体分析. E-mail: xiafan68@qq.com.

high-performance privilege interceptor. The approach can effectively defend against popular network attacks, such as session attacks and CSRF, guarantee low latency for web service access, and provide a flexible way to meet the frequently changing authorization requirements of faculty from different schools and departments.

Keywords: authorization; authority management; access domain model

0 引 言

华东师范大学(简称华东师大)现有的研究生院信息系统包括招生、学籍、培养和学位这4个子系统, 从2005年开始至2011年结束经历了3个研发周期. 当系统最终交付时, 很多现行的研究生管理制度已经发生了较大的变化, 如学籍异动管理规范和预毕(结)业审核等. 从业务角度来看, 现有的研究生院信息系统已经无法有效地支持从教育部到华东师大自身研究生管理的新的业务需求, 造成了大量的信息缺失与不一致, 使得研究生院教职工不得不花费大量的时间与精力以人工辅助的方式完成研究生的信息维护, 尤其是与流程审批相关的业务管理.

另一方面, 从技术角度来看, 与现代基于移动互联的系统设计思维相比, 原有研究生院信息系统所采用的框架与软件产品已经很落后了, 这在现实使用中带来了诸如用户体验差、系统响应速度缓慢, 以及浏览器兼容性差等问题. 从系统维护角度来看, 由于当时采用的很多软件产品与技术已经不再流行, 很难找到一家软件公司对原有系统的源代码进行改造与升级, 这使得研究生院的很多功能需求变更都无法实现.

基于上述原因, 无论从业务角度还是技术角度, 研究生院必须以新业务需求为导向, 通过基于Web 3.0的软件设计理念, 实现现有信息系统的升级换代. 为此, 研究生院经过充分调研, 依托与印孚瑟斯技术中国有限公司成立的联合实验室项目, 于2016年8月组建了华东师大研发团队, 启动了自主研发模式的下一代研究生院信息平台项目. 在新信息平台的第一期项目研发过程中, 研发团队以学籍业务为切入点, 旨在利用成熟的主流开源软件框架打造信息平台的基础设施, 为后续业务子系统的迁移做充分技术准备.

在设计新一代研究生院信息平台的过程中, 华东师大各个院系、职能部门以及研究生院对服务的用户授权和数据安全提出了4个方面的要求: 第一, 能够防御常见的网络攻击, 如会话攻击或跨域请求伪造; 第二, 确保各个院系教职工和职能部门仅能管理与其教学或教务业务对应研究生, 而不能查看其职责范围以外的研究生信息. 换言之, 信息平台必须基于教学单位的组织结构, 提供一种多层次用户授权机制. 然而, 华东师大的教学单位存在实体与虚体的概念, 不同的教职工可能隶属于多个不同的教学与职能部门, 甚至一些院系需要多个院系秘书管理研究生且每个院系秘书负责不同年级研究生. 第三, 用户授权机制要有效地支持流程审批等业务, 提供灵活可配置的授权机制[1-2]. 例如, 学籍异动包括20多个类别, 每个类别都会有不同的审批流程, 每个审批流程中的步骤都需要为院系秘书、分管领导和研究生院学籍管理员分配相应的权限. 在实际使用中, 甚至还会发生因人事调动或岗位调整等因素影响到审批权限的动态变更; 第四, 用户权限管理与验证机制不能影响系统服务的响应速度. 终上所述, 华东师大在研究生管理方面存在着权限管理业务复杂多变的形态, 极需新的信息平台能够提供一种多层级、可配置、高性能的解决方案.

图1展示了一个Web请求从发送到接收到结果所经历的5个步骤, 其中“验证请求权限”是确保系统中数据与服务安全的关键环节. 一旦有非法Web访问突破了“验证请求权

限”这一层, 那么全校研究生的数据就有被泄露的风险. 因此, 要确保研究生院信息平台上数据与服务的访问安全性, 为具有不同权限的教职工提供便捷的服务授权配置机制, 以及能够防御目前主流的跨域请求伪造和会话攻击, 就必须设计并实现一种可靠性非常高的权限管理模块. 基于上述原因, 本文采用了 Spring Security 组件, 通过实现用户权限拦截器栈以及验证用户请求地址和参数等方法, 有效地解决了数据与服务安全问题.

图 1　用户请求流图

Fig. 1　Process flow diagram of typical user request

本文主要贡献如下.

设计了一种基于访问域模型的权限管理功能模块, 结合 Spring 安全组件实现了一个多层级、可配置、高性能的权限拦截器, 有效地解决了构建华东师大研究生院信息平台过程中研究生院、各院系和职能部门提出的用户认证与授权问题, 以及防御常见的网络攻击 (如会话攻击或跨域请求伪造) 所带来的数据安全问题.

基于访问域的权限访问接口, 为研究生院信息平台提供了一个可配置用户权限的 Web 授权管理服务, 通过为角色分配可管理的院系以及可访问的 RESTful 服务接口, 灵活地实现了为不同用户提供对数据和服务的双重授权能力.

实验结果表明, 结合 Spring Security 组件和基于实体图 (EntityGraph) 的数据访问模式来实现权限管理模块, 不会造成系统服务的响应延迟问题.

本文内容组织如下: 第 1 节讨论相关工作; 第 2 节给出需求分析; 第 3 节给出研究生院信息平台中用户权限管理的具体设计与实现; 第 4 节给出性能优化方案; 第 5 节展示性能测试结果; 第六节总结全文.

1　相关工作

目前, 主流的 Java Web 安全框架主要分为 Java JAAS、Spring Security 和 Apache Shiro 等 3 种. JAAS 是 Java JDK 提供的用户认证与授权解决方案, 提供一种可插拔的认证开发模式. 与 Spring Security 相比, JAAS 不支持单点登录, 依赖 Servlet 或 EJB 且配置文件复杂; 与 Apache Shiro 相比, JAAS 不易于理解且开发门槛高[3-6]. Apache Shiro 提供了一种便于开发且安全性高的 Java 安全框架, 在认证、授权、加密和会话管理 4 个方面具备良好的特性. Shiro 易于开发人员的理解, 便于与其他 Web 框架和应用进行整合. 然而, Shiro 不支持单点登录, 无法像 Spring Security 一样为每个方法提供相应的用户权限验证机制[7]. Spring Security 专门为 Spring Web 框架提供了认证与授权机制, 优势在于: ① 支持单点登录; ② 提供了多层拦截器栈模式, 通过配置 URL 路径的方式为不同的 REST 服务指定相应的拦截方法; ③ 拦截方法可以方便地获取从 Web 前端传过来的值, 开发人员能够按照实际需求, 依据请求中的参数去设计拦截方法; ④ 提供了细粒度的权限管理编程接口, 通过安全注释对 REST 服务进行个性化配置, 以及对结果进行二次过滤. 综上所述, Spring Security 在权限管理上不仅支持 URL 粒度上的授权, 而且能够控制到每个 REST 服务的参数以及返回值, 大大提升了系统中数据与服务的安全性[7-9].

基于华东师大研究生院信息平台在用户授权方面的实际需求, 结合对主流安全框架的特性理解 (见表 1), 项目实施采用了 Spring Security 组件, 主要是由于: 第一, 信息平台采用了 Spring Boot 框架; 第二, 业务需求对权限管理的灵活性和粒度提出了更高的要求; 第三, 信息平台需要支持华东师大统一身份认证的单点登录模式.

表 1 3 种主流安全框架的特性对比

Tab. 1 Comparison of three mainstream security frameworks

特性	Spring Security	Apache Shiro	Java JAAS
权限粒度	很细	较细	较细
安全性	高	较高	较高
灵活性	低	高	低
兼容性	低	高	低
单点登录	支持	不支持	不支持

2 需求分析

从组织架构的角度来看, 需要考虑如何为研究生院、教学单位和职能部门进行用户授权管理. 换言之, 就是如何按照研究生院管理规范条例所规定的权限范围, 分别为隶属于研究生院、教学单位和职能部门的教职工进行用户授权管理: 第一, 信息平台仅允许职能部门的教职工浏览与其业务职责相关的全校研究生的信息. 例如, 财务处的教职工只能浏览研究生的缴费信息, 而图书馆的教职工只能浏览并审批研究生的毕业离校申请. 第二, 对于研究生院的教职工, 信息平台则按照其所属部门赋予对应的业务管理员的权限. 例如, 学籍办的老师具备操作所有与学籍管理相关的业务权限, 层级和性质而培养办的老师则只能够操作与培养管理相关的业务权限. 第三, 对于教学单位, 华东师大按照层级和性质这两个维度进行区分. 按照层级, 教学单位分为部—院—系—所; 按照性质, 教学单位则分为实体和虚体. 这里, 虚体教学单位是指在现实中并不具备实际办公场所的教学单位, 但是却有隶属于其的导师和研究生. 因此, 对于隶属于教学单位的教职工, 信息平台需要根据其所隶属的教学单位的层级和性质为其进行相应的授权. 例如, 外语学院为一级教学单位, 包含两个二级教学单位, 日语系和继续教育部; 对于直属于外语学院的教师, 信息平台允许他们访问所有外语学院的研究生信息, 而隶属于日语系的教职工则无法访问继续教育部的研究生信息. 第四, 每一位研究生属于且仅属于一个教学单位, 但一名教职工则可以隶属于多个教学单位或职能部门.

从业务角度来看, 信息平台业务主要分为两类: 单一功能和流程审批. 对于单一功能的业务, 信息平台只要考虑为可以使用该业务的用户提供授权即可; 对于流程审批而言, 信息平台则需要考虑如何为流程中的每一个步骤进行用户授权管理, 确保不同类型的用户只能看到与其审批权限相关的信息, 并执行相应的审批操作; 当某个审批流程发生变化时, 研究生院管理员可以方便地对新的审批流程进行用户授权的更新操作.

基于上述分析, 我们对用户授权的具体需求总结如下.

(1) 分层管理. 由于隶属于不同部门的教职工是按照教学单位进行研究生的业务管理, 因此需要以教学单位的层级来进行用户授权. 具体而言, 信息平台将用户权限等级划分成校级、教学一级、教学二级、学生和导师: 校级管理全校研究生; 教学一级和教学二级分别管理隶属于教学一级单位和二级单位的研究生 (外语学院 (教学一级) 的秘书能管理外语学院的所有研究生, 但不能管理法学院 (教学一级) 的研究生; 英语系 (教学二级) 的秘书能管理英语系的所有研究生, 但不能管理日语系 (教学二级) 的研究生); 每一位学生只能看到自己的信息; 每一位导师可以浏览并管理自己的研究生.

(2) 业务管理. 具有相同层级权限的角色能具备不同的业务操作权限. 例如, 财务处的教职

工虽然具备校级权限, 但仅能够浏览全校研究生的缴费信息, 并不能访问全校研究的其他信息; 学籍管理员也具备校级权限, 但却能访问所有与学籍相关的业务功能.

(3) 流程管理. 具备为业务审批流程的每一个步骤进行用户授权管理, 并且在流程发生变化后进行用户授权的更新操作.

(4) 配置管理. 具备创建不同的角色, 并且为不同的角色进行分配相应的层级权限、业务管理和流程管理的功能.

(5) 响应速度. 用户在访问信息平台提供的服务时会进行相应的用户权限验证, 但对用户授权信息的访问不应当造成服务响应速度的延迟.

图 2 展示了一位具备 3 种角色的用户, 即 A (校级)、B (外国语学院, 教学一级)、C (机械学院, 教学一级), 且假设这 3 种角色都具有流程审批功能. 当该用户以校级权限的身份登录系统后, 该用户则可以审批所有研究生提交的申请; 当该用户以教学一级的身份登录系统后, 则该用户仅能访问审批外国语学院和机械学院的研究生的申请.

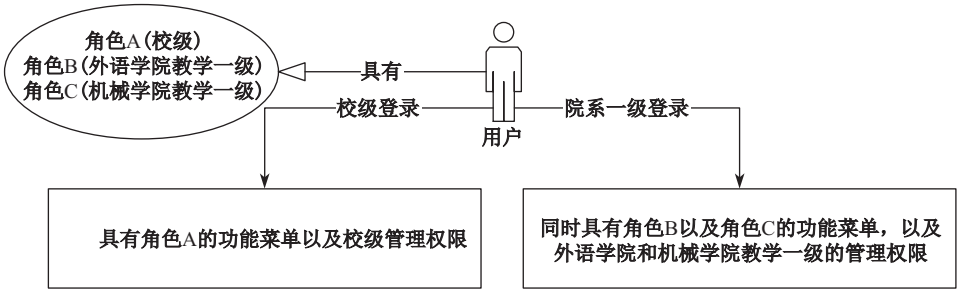

图 2　用户登录授权图

Fig. 2　User login and authorization chart

3　设计与实现

研究生院信息平台上的权限管理模块包括登录授权子模块、权限验证子模块、角色管理子模块、账号管理子模块等 4 个子模块, 其中, 登录授权子模块和权限验证子模块对所有用户开放, 而角色管理子模块和账号管理子模块只能被超级管理员访问.

(1) 登录授权子模块使用单点登录技术, 通过华东师大公共数据库的统一门户认证接口进行身份验证, 验证成功后系统自动跳转到研究生院信息平台的首页; 首页会列举该用户所具备的所有角色, 由用户选择角色并完成登录.

(2) 权限验证子模块根据系统中设定的账号信息和角色信息, 通过基于 Spring Security 的拦截器去验证用户的每个访问请求 (即进行权限判断). 该模块作为应用服务器的一部分在后台运行.

(3) 角色管理子模块用于管理信息平台中的所有角色信息, 包括角色名称、角色等级 (即该角色可以访问的哪个院系的研究生) 以及每个角色所能访问的功能页面.

(4) 账号管理子模块用于管理信息平台中所有的用户账号与角色之间的对应关系, 包括为账号分配角色以及设定权限范围.

3.1　访问域模型

访问域模型记录了与权限管理相关的所有信息, 图 3 给出了访问域模型的实体关系图: ① 一个账号 (Account) 可以是一名教职工或者一名研究生; ② 每个账号可以具备多种角色 (Role), 并且每个账号与所拥有的角色都对应一个访问权限 (AccountPrivilege); ③ 每个功能页面 (Page) 对应了多个 RESTful 服务接口 (RestAPI); ④ 每个角色能够访问多个页面, 并且有

权访问这些页面对应的RESTful服务接口.

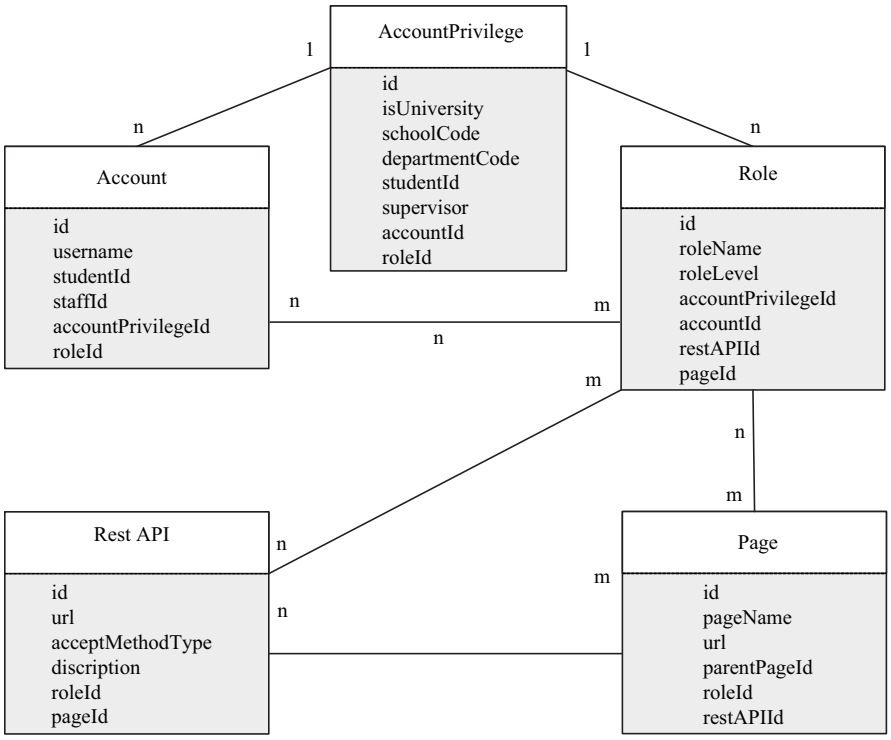

图3 权限模块ER图

Fig. 3 Entity relation diagram for authority module

基于访问域模型, 系统管理员可以很方便地定义角色, 为每一名用户分配角色, 并通过Spring Security的权限拦截器实现访问请求的权限验证. 例如系统管理员定义一个新角色NewRole, 并通过配置使得该角色有权访问“学籍管理”模块下所有页面; 然后系统管理员将该角色分配给账号TestAccount, 并将其权限范围设置为“机械学院”. 那么TestAccount在登录后选择NewRole进入系统, 就能够访问“学籍管理”模块中所有页面, 并有权操作所有机械学院的学生.

3.2 登录与授权

对于用户登录, 登录授权子模块通过调用华东师大统一门户认证的单点登录接口库来进行用户身份认证. 由于单点登录技术已十分成熟且由于篇幅有限, 这里不作过多阐述.

用户授权涉及角色管理子模块和账号管理子模块. 一方面, 角色管理子模块通过管理访问域模型中的角色(Role)、服务接口(RestAPI)以及功能页面(Page)来完成(参见图3). 具体而言, 当为一个角色配置可访问的服务接口时, 可以通过给角色配置可访问页面的方式来配置可访问的服务接口. 之所以采用这种方式, 是因为对于超级管理员来说, 页面比接口更易于理解. 图4展示了“学生”角色的配置界面. 另一方面, 账号管理子模块则通过访问域模型中的账号(Account)、权限(Account Privilege)以及角色(Role)来完成(参见图3). 对于权限等级不是“校级”或“学生”的角色(如图4所示), 系统管理员需要为该账号分配权限, 主要是因为: ①当为某个用户分配“教学一级”或“教学二级”的角色时, 需要为该用户指定其所能管理的院系(可以是多个); ②当某个用户分配“导师”的角色时, 需要为该导师指定所带学生的名字. 图5展示了

权限配置界面.

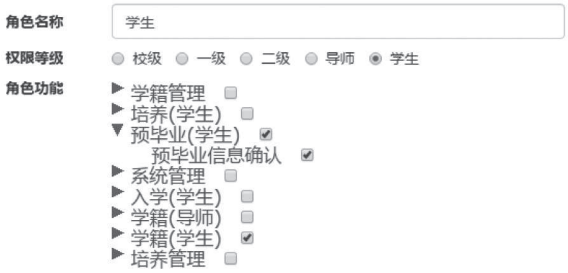

图 4　角色配置界面

Fig. 4　Role configuration interface

图 5　权限配置界面

Fig. 5　Authority configuration interface

3.3　权限验证

当系统管理员为用户分配好角色和权限之后, 权限验证子模块则会对所有访问系统服务的请求进行权限验证, 以确保对数据与服务的访问安全性. 对任意一个访问请求, 权限子系统通过两个步骤来实现权限验证: ① 使用 Spring Security 的拦截器对用户请求进行拦截, 验证该用户是否为合法用户, 是否正确登录, 以及该请求是否为恶意攻击等; ② 使用正则表达式对访问请求的 URL 进行匹配, 判断该请求是否包含敏感信息(如学号、教职工号、院系等), 对于包含了敏感信息的访问请求, 系统需要执行特殊的权限判断, 否则只判断当前用户是否具有访问该 URL 的权限. 图 6 展示了权限拦截的具体流程.

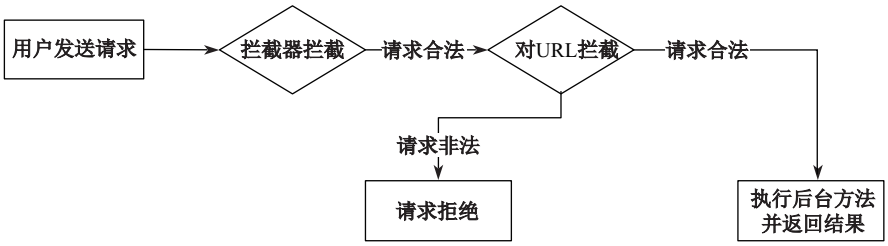

图 6　权限拦截流程图

Fig. 6　Process of authority interception

需要注意的是, 权限验证的第一步不涉及业务逻辑相关的验证 (即不依赖访问域模型), 而第二步则使用访问域模型来执行权限验证.

3.3.1　拦截器栈

基于 Java Servlet 过滤器, Spring Security 实现了一个拦截器栈, 该拦截器栈中的每一个拦截器负责一个具体的权限验证任务. 对于拦截器栈的设计, 需要特别注意的是, 拦截器栈中的拦截器的等级要由弱变强, 否则会造成拦截器栈的内部错误.

当一个用户请求进入到拦截器栈之后, Spring Security 会通过 8 个拦截器依次去验证该请求的合法性, 而这些验证不涉及访问域模型. 表 2 给出了拦截器栈中每个拦截器的具体功能.

表 2 拦截器栈中各拦截器功能

Tab. 2 Functions of interceptors in the interceptor stack

拦截器名称	功能
HttpSessionContextIntegrationFilter	处理用户的会话中的会话上下文对象, 如果当前的会话中没有这个对象, 则生成一个上下文对象并放入该用户的会话中
LogoutFilter	处理注销请求
AuthenticationProcessingFilter	判断用户是否登录
SecurityContextHolderAwareRequestFilter	将用户发来的请求进行包装, 时候后续拦截器能够基于包装后的对象进行处理
AnonymousAuthenticationFilter	当用户发送匿名请求时, 为当前用户赋予一个匿名权限
ExceptionTranslationFilter	捕捉并处理 Spring Security 中产生的异常
SessionFixationProtectionFilter	防御会话固定攻击
FilterSecurityInterceptor	保护HTTP资源的安全, 并在请求拒绝的时候抛出异常

对于表 2 中前 7 个拦截器, 信息平台采用 Spring Security 的默认实现, 主要是因为: 这几个拦截器的功能不涉及具体业务逻辑上的权限验证, 使用默认的实现就能够很好地满足本项目的需求; 而最后一个拦截器则需要根据信息平台的权限管理需求来实现. 具体而言, FilterSecurityInterceptor 主要依赖 AuthenticationManager 的接口实现与 AccessDecisionManager 的接口实现. 前者是为了给用户进行授权, 后者是对用户发送的请求进行访问权限判断. 需要注意的是, 对 URL 的拦截是在 AccessDicisionManager 中配置的.

3.3.2 URL 拦截

Spring Security 提供了 PreAuthorize 注解, 用于在执行用户请求的方法之前进行业务逻辑上的权限判断. 由于在每一个服务接口上添加该注解增加了系统开发的复杂度且不利于后期维护, 为此我们通过规范化 URL 路径的方式 (即在 URL 中定义变量参数以及命名规则), 并通过正则表达式的匹配模式来实现基于 URL 拦截的权限验证.

具体而言, 当用户执行查找或修改操作时, 会向应用服务器发送一个 URL 请求, 并且该请求会携带某些参数用于完成具体业务. 如果请求中包括了敏感信息 (如学号、职工号和综合查询条件), 系统则执行由根据也无需求制定的权限验证 (图 7 展示了对携带学号的 URL 请求的权限验证过程); 否则该请求则被视为普通请求, 仅执行一些共性的访问性验证.

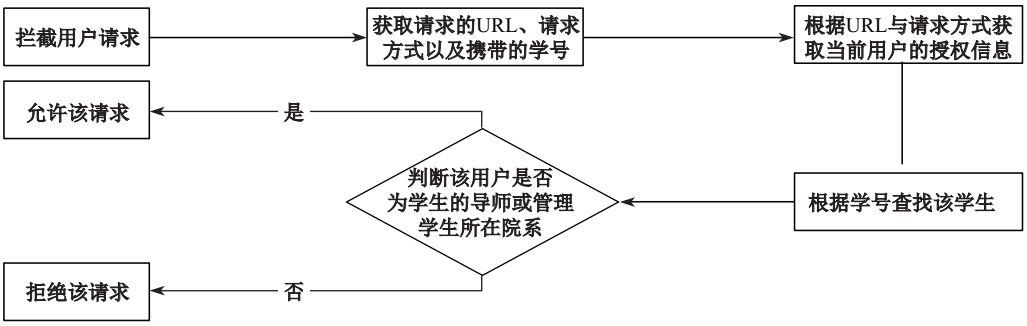

图 7 对携带学号的请求权限验证流程图

Fig. 7 Verification process for requests with a student number

例如携带学号请求的URL需要满足以下两种规则之一: ①请求路径中包含学

生学号, 例如接口/api/xj/students/{sno}/profile用于查询学生的信息简介, 当查询学号为10000的学生时, 将{sno}替换为10000; ②请求参数中包含参数sno, 用于指定学号, 例如接口/api/xj/students?sno=10000用于获取学号为10000的学生的学籍信息. 系统能够从URL中提取出学号, 并按照图7所示的流程进行权限验证.

对于携带职工号的请求权限验证与图7类似, 不同点在于最后只需要判断当前用户是否有权限管理该职工所在部门. 对于包含综合查询条件的请求, 首先要判断条件中是否包含学号、职工号等信息, 如果包含, 则先进行学号与职工号的权限验证, 然后再进行部门验证, 图8展示了具体验证流程. 如果是不包含敏感信息的普通请求, 就判断当前用户的权限能否访问该URL即可.

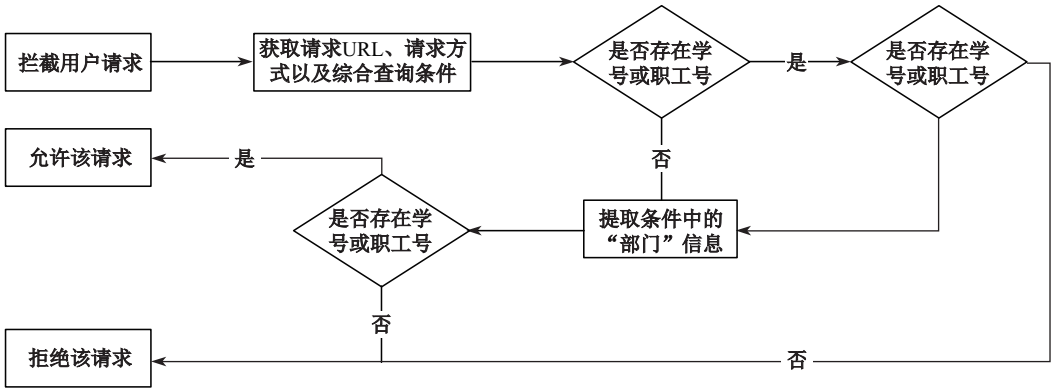

图8　对携带综合查询条件的请求权限验证流程图

Fig. 8　Verification process for requests with a query condition

4　性能优化

在系统实现中, REST服务接口是通过JPA (Java Persistence API)来访问权限表中的数据. 初始实现直接使用JPA提供的查询接口来执行多对多或者一对多等主外键连接查询, 然而在调用权限服务时出现了响应速度慢的问题, 即“$n+1$查询”问题. 这里, “$n+1$查询”问题是指当使用JPA查询一个包含n个外键的实体时 (每个实体对应数据库中的一个表), 该实体中声明的每个外键就会产生一次额外的查询请求; 当访问一个包含n个外键的实体时, 系统就不得不执行$n+1$个查询; 如果所查询的实体对象有多个时 (即数据库中的多条记录), 每个实体对象都会执行额外的n次查询, 因此会造成查询速度大大降低. 由于权限管理的服务接口执行频率很高 (即所有用户请求都需要首先执行权限验证), 因此对权限管理的代码性能优化对整个系统而言具有十分重要的意义.

为了解决“$n+1$查询”问题, 在权限服务的接口实现中采用JPA2.1的一个新特性, 实体图注解 (或EntityGraph注解). 该注解能够定义哪些外键属性是需要查询的, 且可以包含多层嵌套查询. 在查询执行时, 与未采用实体图的方法相比, JPA会根据实体图的定义生成一条包含所需查询外键属性的SQL语句, 而不是$n+1$个查询. 由于实体图是通过注解方式配置在实体类上, 并在实体类对应的Repository查询接口中完成对实体图的调用, 而不需要重写查询接口, 因此能够在不改变接口实现的情况下解决“$n+1$查询”问题, 达到提升系统响应速度的目的.

图9展示了RestAPI实体类的实体图定义, 其中RestAPI实体类与Role实体类和Page实体类之间具有复杂的外键关联关系 (见图3), 如访问RestAPI实体类中某个roles属性的pages属性. RestAPI的实体图包含了2个命名实体图(NamedEntityGraph): restAPI.all和restAPI.roles (行31至行38), 其中, restAPI.roles实体图用于查找出RestAPI实体类中的roles属性, 而restAPI.all实体图则用于查找RestAPI实体类中所要访问的roles属性和pages属

性(行33至行34), 以及通过定义pages的子图来查找每个roles属性所关联的pages属性(行33至行36). 由上可见, 通过定义子图以及子图之间嵌套关系(包括多级嵌套查询), 实体图可以很方便地声明如何访问多个实体类.

```
30 @NamedEntityGraphs({
31     @NamedEntityGraph(name = "restAPI.all",
32         attributeNodes = {
33             @NamedAttributeNode(value = "roles",subgraph = "pages"),
34             @NamedAttributeNode(value = "pages") },
35         subgraphs = @NamedSubgraph(name = "pages",
36             attributeNodes = @NamedAttributeNode("pages"))),
37     @NamedEntityGraph(name = "restAPI.roles",
38         attributeNodes = { @NamedAttributeNode(value = "roles") }) })
39 public class RestAPI extends EntityId {
```

图9 RestAPI实体类的实体图定义

Fig. 9 Definition of entity graph in RestAPI

图10给出了RestAPI实体类对应的Repository查询接口的实现, 即在findRestAPI方法上, 通过实体图注解调用图9中RestAPI实体类定义的restAPI.roles实体图, 其中行17指定了该查询接口使用的实体图名称, 而行18则指定了数据加载模式.

```
17 @EntityGraph(value = "restAPI.roles",
18         type = EntityGraph.EntityGraphType.FETCH)
19 RestAPI findRestAPI(String api, String httpMethod);
```

图10 基于RestAPI实体图定义的权限服务接口实现

Fig. 10 Implementation of authority service interface based on RestAPI entity definition

结合图9和图10, 我们可以发现, 实体图可以很方便地在权限模型上定义复杂的数据拓扑结构, 通过在对应的Repository方法上去引用权限模型上已定义的拓扑关系, 由Spring框架在调用权限服务接口时生成相应的SQL连接查询; 当查询结果返回时, Spring框架会依据服务接口定义的数据拓扑结构, 将结果封装成对应的权限模型对象, 从而达到减少开发工作量和提升服务接口的响应速度的双重目标.

5 实 验

为了验证获取用户权限的两种方法在实际使用中的性能差异, 首先安装了1台Centos7测试服务器, 配备了8个主频为2.40GHz的Intel Xeon CPU E5-2630芯片, 内存为16 GB, 磁盘容量为4 TB且转速为7200 r/s. 接下来, 我们在该服务器上部署了对应的2套研究生院信息系统, 并导入了40 000名研究生的学籍信息和华东师大600多名教职工的账号.

第一组实验测试了5种用于获取用户权限的访问请求: 用户登录; 激活角色; 获取授权页面; 获取用户角色与获取角色列表. 选择这5种访问请求的主要原因在于: 第一, 这些请求是权限管理模块的核心功能; 第二, 与这些请求对应的获取用户权限的SQL查询涉及了所有的权限域表, 并且需要执行复杂的连接操作. 在实验中, 每种访问请求被随机执行了10次, 实验结果为10次访问请求的平均响应时间.

图11展示了优化前后基本权限请求耗时对比. 从图中可以看到, 与优化前的访问请求的响应速度相比, 优化后的访问请求的响应速度提高了4到10倍. 主要原因是在系统实现中, 获取授权页面的执行频率很高(几乎所有业务访问请求都需要调用获取授权页面的服务), 因此优化后

对其他页面访问的响应速度有了较大的提升.

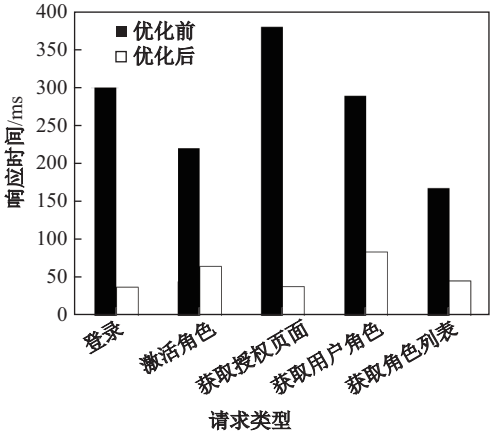

图 11　权限请求的响应时间

Fig. 11　Response time of authority request

第二组实验测试了两种不同业务访问请求, 即 REST 查询中是否有用户账号信息 (在实现中, 所有的 Web 访问请求都会调用对应的 REST 服务, 而有些 REST 服务的 URL 中是需要带参数的). 具体而言, 我们测试了两个最常用的学籍服务: 学生分页查询与单个学生查询. 前者用于对比优化前后“带综合查询条件的访问请求的权限验证”的响应速度, 后者则用于对比优化前后“带账号的访问请求的权限验证”的响应速度. 为了去除“带账号的访问请求的权限验证”的影响, 综合查询条件中不包含学号.

图 12 展示了优化前后业务访问请求耗时对比. 从结果中可以发现, 与优化前的方法相比, 优化后的“综合查询条件验证”的响应速度提升了近 25%, 而优化后的“带账号的访问请求的权限验证”的响应速度则提升了近 30 倍. 主要原因是: 对账号的权限验证需要执行多次多表连接查询 (角色表与账号表中的外键都会导致一次额外的查询), 因此效率很低; 而优化后仅需要执行一条 SQL 查询, 因此效率得到了极大的提升. 对于实际用户体验而言, 优化前的页面响应速度很慢, 大部分在 1 s 以上 (重新加载“授权页面”需要 3 s 左右的响应时间, 而“单个学生信息维护”页面甚至需要 5 s 左右的响应时间), 用户体验非常差; 优化后访问页面的响应速度都在 1 s 以内, 极大地改善了用户体验.

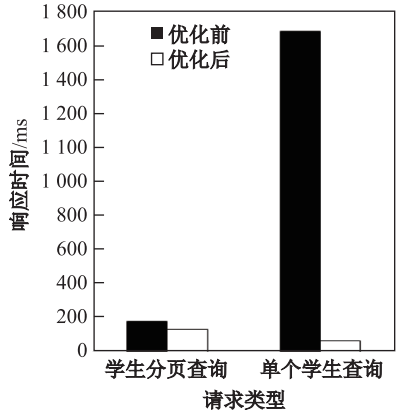

图 12　业务请求的响应时间

Fig. 12　Response time of business request

6 结 论

在实际 Web 应用中, 一个功能完备的权限管理模块是保障软件系统中数据与服务安全的基础, 并且其他上层业务需求也需要基于该模块来设计实现. 本文根据华东师范大学研究生院管理中的实际需求, 并结合目前比较流行的 Spring Security 框架, 设计实现了一个针对该校研究生院管理平台的权限管理模块. 本文详细阐述了权限模块的访问域模型, 并基于该模型实现了一种多层级、可配置、高性能的权限拦截器. 最后还针对在实际使用过程中出现的效率问题进行了优化. 经过实际测试证明, 本文设计并实现的权限管理模块具有较高的验证效率, 能够满足研究生院信息平台的实际使用需求.

[参 考 文 献]

[1] 吴波, 王晶. 基于基本 RBAC 模型的权限管理框架的设计与实现 [J]. 计算机系统应用, 2011(4): 50-54.

[2] 贾青梅, 杨正球. 统一权限管理模块的设计与实现 [C]//2009 通信理论与技术新发展——第十四届全国青年通信学术会议论文集 [C]. 中国通信学会青年工作委员会, 2009: 233-237.

[3] ZHAO F, WANG L, TIAN X. Design and implementation of authorization management system based on RBAC [J]. Computer & Digital Engineering, 2012, 532/533(43): 586-590.

[4] 桂艳峰, 林作铨. 一个基于角色的 Web 安全访问控制系统 [J]. 计算机研究与发展, 2003, 8: 1186-1194.

[5] 顾春华, 肖宝亮. RBAC 模型层次关系中的角色权限 [J]. 华东理工大学学报 (自然科学版), 2007(1): 96-99.

[6] 杨柳, 危韧勇, 陈传波. 一种扩展型基于角色权限管理模型 (E-RBAC) 的研究 [J]. 计算机工程与科学, 2006, 9: 126-128.

[7] 桂艳峰, 林作铨. 一个基于角色的 Web 安全访问控制系统 [J]. 计算机研究与发展, 2003, 8: 1186-1194.

[8] NI P, LIAO J, WANG C, et al. Web information recommendation based on user behaviors [C]//Computer Science and Information Engineering, 2009 WRI World Congress on. IEEE Xplore, 2009: 426-430.

[9] ZHANG Y, JOSHI J B D. Role Based Access Control [M]. New York: Springer, 2009.

(责任编辑: 李 艺)

(上接第 54 页)

[12] 席博彦, 包图雅. 关于r-平均凸函数的一些性质 [J]. 数学的实践与认识, 2008, 38(12): 113-119.

[13] GILL P M, PEARCE C E M, PECARIC J. Hadamard's inequality for r-convex functions [J]. Math Aanl Appl, 1997, 215(2): 461-470.

[14] 张孔生, 刘敏. P 方凸函数及其 Jensen 型和 Rado 型不等式 [J]. 阜阳师范学院学报(自然科学版), 2005, 22(2): 18-20.

[15] 吴善和. 对数凸函数与琴生型不等式 [J]. 高等数学研究, 2004, 7(5): 61-64.

[16] 陈少元. AH-凸函数及其应用 [J]. 湖北职业技术学院学报, 2013, 16(2): 106-109.

[17] 宋振云. AR-数及其 Jensen 型不等式 [J]. 安徽师范大学学报(自然科学版), 2015, 38(4): 331-336.

[18] 宋振云. AM-凸函数及其 Jensen 型不等式 [J]. 淮北师范大学学报(自然科学版), 2015, 36(1): 1-7.

[19] XI B Y, QI F. Some integral inequalities of Hermite-Hadamard type for s-logarithmically convex functions [J]. Acta Mathematica Scientia, 2015, 35(3): 515-524.

[20] 匡继昌. 常用不等式 [M]. 4 版. 济南: 山东科学技术出版社, 2010: 53-63.

(责任编辑: 林 磊)