**信息与软件工程学院**



学 号：

姓 名：

专业方向： 软件工程

企业名称：思爱普（北京）软件系统有限公司成

都分公司

实习岗位名称： Software Design Engineer

企业指导教师：

院内代管教师：

**企业实习总结报告**

摘要

在众多企业级系统中，认证与授权模块的开发是必不可少的。但是现如今的普适性认证与授权模块开发框架与设计方法已经无法满足日益增长的认证授权模块的性能需求，另外，企业架构的不断复杂化也对现有企业级系统中认证与授权模块的开发带来了巨大的挑战，因此，该模块的开发方法以及框架设计的优化工作就显得尤为重要。此次实习参与的“基于RBAC的企业级系统认证与授权模块开发探究”项目和“基于列存储数据库的系统数据交互优化”项目，希望从系统后端实现，数据库设计以及数据库存储模型等方面探究企业级系统相应模块的性能优化方法。

关键词：RBAC，数据库设计，认证与授权，列存储

**ABSTRACT**

In many enterprise-class systems, the development of authentication and authorization module is essential. In addition, the increasing complexity of enterprise architecture also poses a great challenge to the development of authentication and authorization module in existing enterprise systems, so it is important to optimize the development method and framework design of this module. In this internship, we participated in the project of "Exploring the Development of RBAC-based Authentication and Authorization Module for Enterprise-level System" and the project of "Optimization of Data Interaction for Column-based Database System", in which we hope to explore the development of RBAC-based authentication and authorization module from system back-end implementation, database design and database storage model. It is to investigate the performance optimization method of the corresponding module in the enterprise system.

Keywords: RBAC , Database design, Authentication and authorization, Column storage

目 录

[1.实习概况 3](#_Toc49433484)

[1.1. 实习单位与实习岗位概况，对企业所属行业的认识 3](#_Toc49433485)

[1.1.1. 实习单位概况 3](#_Toc49433486)

[1.1.2. 实习岗位概况 4](#_Toc49433487)

[1.1.3. 所属行业认识 4](#_Toc49433488)

[1.2. 实习项目课题背景、价值、意义以及国内外研究现状 5](#_Toc49433489)

[1.2.1. 课题背景、价值、意义 5](#_Toc49433490)

[1.2.2. 国内外研究现状 6](#_Toc49433491)

[1.3. 实习项目整体执行完成情况概述 6](#_Toc49433492)

[2.复杂工程问题归纳与实施方案可行性研究 7](#_Toc49433493)

[2.1. 实习项目的初步分析与实习课题的目标与任务 7](#_Toc49433494)

[2.1.1. 用户需求初步分析 7](#_Toc49433495)

[2.1.2. 实习课题目标与任务 8](#_Toc49433496)

[2.1.3. 相关复杂工程问题总结 8](#_Toc49433497)

[2.2. 解决方案及可行性研究 8](#_Toc49433498)

[2.2.1. 基于Redis的缓存机制优化 8](#_Toc49433499)

[2.2.2. 数据库设计优化 9](#_Toc49433500)

[2.2.3. 用户认证实现方式研究 9](#_Toc49433501)

[2.2.4. 资源授权框架研究 9](#_Toc49433502)

[2.3. 实习课题任务相关的详细需求分析 9](#_Toc49433503)

[2.3.1. 需求分析 10](#_Toc49433504)

[3.针对复杂工程问题的方案设计与实现 11](#_Toc49433505)

[3.1. 针对复杂软件工程问题而开展设计 11](#_Toc49433506)

[3.1.1. 应用场景设计 11](#_Toc49433507)

[3.1.2. 资源访问控制模型设计 11](#_Toc49433508)

[3.1.3. 数据库设计 12](#_Toc49433509)

[3.2. 针对复杂工程问题的具体实现、测试以及结果分析 13](#_Toc49433510)

[3.2.1. 项目数据库初始化以及生成测试数据 13](#_Toc49433511)

[3.2.2. 框架搭建 16](#_Toc49433512)

[3.2.3. 接口实现 18](#_Toc49433513)

[3.2.4. 基于Redis的缓存机制优化 21](#_Toc49433514)

[3.2.5. 项目测试以及结论 22](#_Toc49433515)

[4.知识技能学习情况 26](#_Toc49433516)

[4.1. Spring框架和Spring MVC框架 26](#_Toc49433517)

[4.2. 数据库操作方法：JdbcTemplate 27](#_Toc49433518)

[4.3. 基于Spring的安全管理框架Spring Security 27](#_Toc49433519)

[4.4. RBAC权限管理模型 29](#_Toc49433520)

[5.工程协作交流情况 30](#_Toc49433521)

[6.工程计划管控与执行情况 31](#_Toc49433522)

[6.1. 针对复杂问题的执行计划安排 31](#_Toc49433523)

[6.2. 存在的主要问题及应对项目风险与挑战的能力培养情况 31](#_Toc49433524)

[7. 职业素养与工程伦理的学习与培养 33](#_Toc49433525)

[8. 对软件工程实践以及软件工程领域发展的认识 35](#_Toc49433526)

[9.结束语 38](#_Toc49433527)

[9.1. 课题完成情况、有待进一步解决的问题及方向 38](#_Toc49433528)

[9.2. 本人对于企业实习的收获及体会 38](#_Toc49433529)

[致谢 40](#_Toc49433530)

1.实习概况

1.1. 实习单位与实习岗位概况，对企业所属行业的认识

1.1.1. 实习单位概况

SAP创立于1972年的德国，是全球商业软件市场的领导厂商，提供优质的应用程序和服务，帮助超过25个行业内各种规模的企业实现卓越运营。根据市值排名为全球第三大独立软件制造商。在全球120多个国家拥有109,000个企业客户，其中包括财富500强80%以上的企业，并在包括欧洲、美洲、中东及亚太地区的50个国家雇用52,921名员工。公司总部位于德国沃尔多夫。SAP将企业软件定义为企业资源计划、商务智能和相关应用程序的集合。SAP的核心业务是销售其研发的商业软件解决方案及其服务的用户许可证。SAP解决方案包括标准商业软件及技术以及行业特定应用，主要用途是帮助企业建立或改进其业务流程，使之更为高效灵活，并不断为该企业产生新的价值。

另外，SAP还提供与其软件相关的咨询、维护和培训服务。SAP历来坚持与客户及独立业务合作伙伴建立紧密合作，共同进行软件开发。

SAP在中国拥有众多的合作伙伴，包括浙江中之杰[4-5]  、亿达集团-亿达信息技术、中国石化、IBM、HP、Sun、美盈森、埃森哲、毕博、凯捷中国、德勤、源讯、高维信诚、广州数通、神州数码、太极计算机、东软软件、汉普、新波信息科技、北京龙象信益、清华紫光，方正科技、华软新元、广东新盛通、明基逐鹿等。SAP在众多的项目中与这些伙伴密切合作，将先进的管理理念和方法转变为切实帮助中国企业成功的现实。

SAP中国研究院正式成立于2003年11月20日，其前身为1997年创立的 SAP中国研发中心。作为SAP全球十四大研究院之一，中国研究院多年来持续创新，在企业应用、商业分析、移动商务、数据库和技术平台以及云计算五大市场 上持续突破，并基于业界领先的SAP HANA平台，不断提升各领域的研发工作，致力于打造面向不同行业，适合各类企业规模的全系列产品及解决方案。SAP中国研究院是增长最快的SAP海外研发机构之一，现已成为除德国总部外 SAP第三大研究院。

1.1.2. 实习岗位概况

我在SAP担任SDE(Software Design Engineer – 即软件设计工程师)的工作，我的日常工作是参与完成数据库设计，独立完成API接口编写以及测试等工作，配合同项目组的其他成员完成系统项目的整体开发工作，以满足客户需求为目标，做出有竞争力的产品，同时编写出完善的开发文档以及软件操作文档，使自己在完成项目开发的同时提高自己项目描述以及过程表达能力。

我的岗位职责有如下四点：

1.根据软件设计需求设计数据库，并编写SQL语句。

2.根据项目需求编写API接口文档以及编码实现API接口。

3.编写测试用例测试API接口，并尝试不断优化接口性能。

4.对接口性能进行测试，尝试探索其他方式优化现有实现方式。

我的具体工作流程如下：

项目准备阶段：

1. 由企业导师分配任务并每周跟进，通过会议形式讨论任务具体需求。
2. 配合导师准备Jenkins从而对项目进行集成管理
3. 创建项目所需数据库，并且保证数据库连接正常

项目开发阶段：

1. 根据需求设计接口以及数据库
2. 编码实现符合功能要求的API接口
3. 编写测试用例，并通过Junit工具进行本地测试
4. 通过Git将项目代码推到远端代码仓库，同时Jenkins对代码进行集成管理。

项目收尾阶段：

1. 测试优化API接口
2. 编写项目开发文档

1.1.3. 所属行业认识

SAP属于软件公司，是目前世界第三大软件公司，并且是目前世界上最大的企业应用软件供应商。作为全球领先的企业管理软件解决方案提供商，SAP帮助各行业不同规模的企业实现卓越运营。从企业后台到公司决策层、从工厂仓库到商铺店面、从电脑桌面到移动终端—SAP助力用户和企业高效协作，获取商业洞见，并从竞争中脱颖而出。SAP的软件和服务能够帮助客户实现盈利性的运营，不断提升应变能力，实现可持续的增长。全球188个国家的232,000家客户正在从SAP解决方案中获益, 其中包括财富500强80%的企业及85%最有价值的品牌。

SAP最重要的就是它开发的ERP系统。[ERP系统](http://baike.baidu.com/view/86915.htm" \t "_blank)是[企业资源](http://baike.baidu.com/view/2062419.htm)计划(Enterprise Resource Planning )的简称，是指建立在信息技术基础上，集信息技术与先进管理思想于一身，以系统化的[管理思想](http://baike.baidu.com/view/141611.htm" \t "_blank)，为企业员工及决策层提供决策手段的管理平台。它是从[MRP](http://baike.baidu.com/view/112988.htm)（物料需求计划）发展而来的新一代集成化[管理信息系统](http://baike.baidu.com/view/2670.htm" \t "_blank)，它扩展了MRP的功能，其核心思想是[供应链管理](http://baike.baidu.com/view/10365.htm" \t "_blank)。它跳出了传统企业边界，从[供应链](http://baike.baidu.com/view/3235.htm)

范围去优化企业的资源，优化了[现代企业](http://baike.baidu.com/view/1089325.htm" \t "_blank)的运行模式，反映了市场对企业合理调配资源的要求。它对于改善[企业](http://baike.baidu.com/view/38340.htm)业务流程、提高企业核心竞争力具有显著作用。[ERP](http://baike.baidu.com/subview/3609/6859738.htm)行业人才稀缺成为[SAP](http://baike.baidu.com/subview/8784/8599843.htm" \t "_blank)发展的制约因素之一，鉴于此，国内的ERP培训行业逐渐开始发展。

1.2. 实习项目课题背景、价值、意义以及国内外研究现状

1.2.1. 课题背景、价值、意义

随着软件行业的迅猛发展，企业级软件产品的质量以及软件性能逐渐成为企业生存与发展的核心。为了使商用系统的性能在技术更新换代的潮流中也能够保持其领先性，大多数企业都会尝试采用最先进的开发方式以及开发工具在该公司所交付系统的模块开发上进行优化，从而为客户提供行业领先的、具有良好性能的企业级软件，使得本企业能够在市场中立足。

就我实习的公司 -- SAP而言，一个完整的[ERP系统](http://baike.baidu.com/view/86915.htm)是[企业资源](http://baike.baidu.com/view/2062419.htm)计划，即指建立在信息技术基础上，集信息技术与先进管理思想于一身，以系统化的[管理思想](http://baike.baidu.com/view/141611.htm)，为企业员工及决策层提供决策手段的管理平台。由此可以看出，一个ERP系统是非常的庞大并且错综复杂的。一个好的ERP系统，不光要保证其好的逻辑性还要有较好的流畅性和用户体验。而我的工作就是聚焦于ERP系统中常见的用户认证与授权模块的优化，从而使得新一代ERP系统中的用户认证与资源授权模块的适用性以及性能在同一类型产品中取得领先。

本课题可移植性强，发展前景较好。因为国内软件业因对软件性能优化的重要作用认识较晚，并且大多数企业受制与开发成本管控以及交付时间而无法投入时间和成本去开发出一套实用性强、性能优良的用户认证与资源授权模块的开发框架。

1.2.2. 国内外研究现状

1.国外发展现状：

国际上，软件优化以及框架研发工作是一件非常重要的工程工作。在IBM、Microsoft等开发大型系统软件公司，其研发人员占比较国内企业有着显著的差别。如果一个公司能够研发出一套高效的开发框架，那么将会极大程度上提高该公司软件开发的效率，同时开发成本也会得到显著的降低。

2.国内发展现状：

中国软件产业缺乏完整的产业生态链，呈现出来两大特点：硬件方面，整机产品强，核心元器件弱；软件方面，应用软件强，基础软件弱。

虽然中国有 QQ、微信、支付宝等一大批应用软件，但在操作系统、数据库、中间件等基础软件上，市场份额却微乎其微，这些基础软件基本被微软、谷歌、甲骨文、IBM 等国外公司垄断，看似红火的 ICT 产业和互联网产业，其实是建立在美国的基础软硬件之上的，国内设计公司只是把国外公司提供的模块做集成工作，国外上游企业只要断了技术支持和技术授权，国内的研发就会停摆。

就开源软件而言，大多数开源系统和工具都是由国外程序员贡献的，例如：Linux、MySQL 等，虽然中国拥有数以百万计的庞大程序员队伍，但在对代码的贡献量方面却极其有限。

1.3. 实习项目整体执行完成情况概述

实习期间，公司分配给我的任务均已完成。其中包括：

1. 完成企业级系统中用户认证与资源授权模块框架设计、开发与性能测试工作。

2. 完成用户认证与资源授权模块底层行式数据库与列式数据块性能分析与研究工作。

3. 其他还包括一些参与项目中的小任务，例如选取一种密文尽可能短的对称加密算法用于保证项目中数据库数据的安全，该项目属于亚洲基础设施投资银行项目。

2.复杂工程问题归纳与实施方案可行性研究

2.1. 实习项目的初步分析与实习课题的目标与任务

在为期半年的实习期间，我共参与了3个项目:

1.基于可搜索加密算法的恶意行为检测中间件（实验室项目）

2.基于RBAC的用户认证与授权模块开发

3.基于数据库列存储模型的后端数据交互优化

其中项目1为疫情原因下的实验室项目，因为并非公司所作项目，故在此报告中可能简单提及但不过多赘述，详细内容在初期报告中可见。项目2，3为公司实习期间独立完成的项目，这两个项目偏研究探索性质，与传统工程项目有所不同。

实习时间轴如下图所示

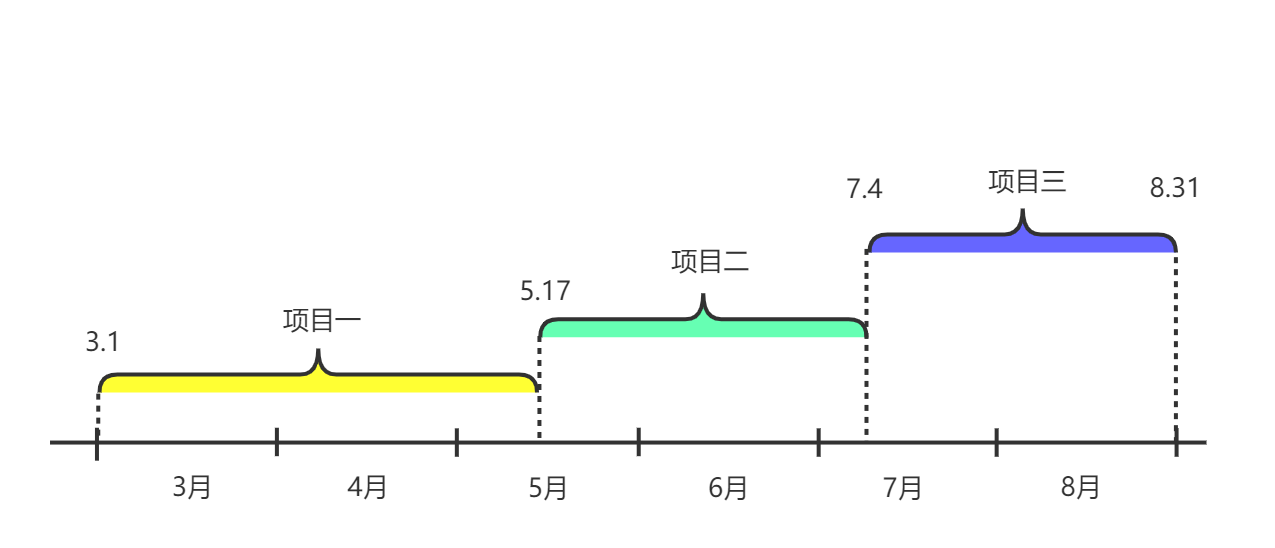


图 2-1 实习工作任务时间轴

2.1.1. 用户需求初步分析

基于RBAC的用户认证与授权模块的需求主要分为两类：功能性需求和非功能性需求。而我们此次的项目主要用于设计一个基于RBAC的用户认证与授权模块的开发框架，因此会更聚焦与该项目的非功能性需求。在功能性需求方面，基本包括用户的登陆，以及用户访问资源时根据用户所在组或是其所属角色进行资源数据的访问控制等功能。而在我们需要重点考虑的非功能性需求中就包括系统的性能要求以及可靠性要求。

2.1.2. 实习课题目标与任务

实习课题目标：完成用户认证与资源授权模块的调研、设计、开发以及性能测试及优化工作，并且在此基础上尝试开发出一套具有高适用性的模块开发框架。

此次实习课题，我的主要任务是：

1. 完成调研现阶段国内外企业级软件系统在用户认证与授权模块中开发模式以及使用方法。

2. 完成用户认证与授权模块的设计与开发工作。

3. 通过测试不同授权模型下该模块性能，得出较优方案。

4. 尝试通过数据库底层物理结构优化该模块运行性能。

2.1.3. 相关复杂工程问题总结

在企业级系统的实际开发过程中，系统的响应时间是一个衡量该系统品质的一个重要指标。而企业级系统与一般系统有所不同，首先，其数据量规模较一般系统更大，可能达到百万级别，甚至更高。其次，企业级系统的数据可能是分布式存储或者云存储，因此，网络I/O也会较一般系统流量更大。另外，企业级系统还需要考虑高并发对系统响应时间的影响。合格的企业级商用系统是能够保证高并发的情况下，依旧能够带给用户较为舒适的响应环境。

而大多数商用系统都有一个不可缺少的模块，那就是系统的认证与权限管理模块。认证即系统用户的身份认证，权限管理即判断某个用户是否有对某类资源的某种操作权限。这一模块所需要的两个功能都离不开系统与数据库之间的数据传递。因此我们项目所需要解决的主要问题就是针对企业级系统的认证与权限管理模块，应该如何在与数据库交互层面进行响应时间的优化。

2.2. 解决方案及可行性研究

2.2.1. 基于Redis的缓存机制优化

通过对Redis充分了解后，我们发现Redis对于大量的数据，其缓存他机制可以缓解接口（数据库）的压力，因此我们考虑通过基于Redis的缓存机制来提高在用户认证与授权过程中对查询的响应时间。

我们最初考虑的缓存思路为执行查询，若缓存中存在数据，则查询缓存，如果缓存中不存在数据，那么再查询实时接口。我们通过模拟缓存机制的方法，测试五个并发查询的情况。当五个并发同时运行的时候，第一个查询实时服务，而其余的则查询缓存，结果显示，反应速度较无缓存有所提高。但是，我们发现当我们的实时接口查询速度很慢的时候，就会导致，五个并发查询都会去直接查询接口服务，而不会仅一个查询接口，其余并发访问查询缓存。这样一来，缓存的作用就非常小了。而如果对于相同的请求，能够控制其等待一段时间后，再去执行，即得到第一次查询后存入缓存数据库，这中方式下的查询效率就会相比之前有极大的提升。但是，这种方法会导致如果缓存失效，所有其他并发查询的等待就会降低系统的响应时间。因此我们又产生了两种解决方法，第一种是在缓存存在的时间里进行异步查询以更新缓存，而第二种方法则是使用二级缓存，在一级缓存失效的时候读取二级缓存，二级缓存异步更新。经过初步验证发现，这两种方法能够解决之前存在的问题，但仍需进一步的讨论优化，从而解决上文提出的主要问题：在企业级系统的认证与权限管理模块优化系统响应时间。

2.2.2. 数据库设计优化

大部分数据库也存在缓存机制，从实验中可以发现，相同的查询，在连续两次的查询中所消耗的时间是不一样的。而在不同数据库中所表现的差异也是不一样的。然后我们的项目是提供一个可扩展，可移植的基于RBAC的用户认证与授权框架，因此我们暂不考虑数据库级别的查询优化。

2.2.3. 用户认证实现方式研究

在实习项目中，我们使用SpringSecurity自主实现分布式系统的用户验证授权及权限验证功能，其中功能点包括：基于REST请求的登录、用户名密码验证以及验证成功后给用户授权、HTTP请求的权限配置和验证等。

2.2.4. 资源授权框架研究

在实习项目中，我们基于SpringBoot以及Mysql数据库搭建了一个满足资源授权模块基本需求的后端框架。基于对不同授权模型对该后端框架进行研究分析，最后选取出性能较优，可扩展性较强的基于RBAC的资源授权模块后端开发框架。

2.3. 实习课题任务相关的详细需求分析

2.3.1. 需求分析

在企业级系统的实际开发过程中，系统的响应时间是一个衡量该系统品质的一个重要指标。而企业级系统与一般系统有所不同，首先，其数据量规模较一般系统更大，可能达到百万级别，甚至更高。其次，企业级系统的数据可能是分布式存储或者云存储，因此，网络I/O也会较一般系统流量更大。另外，企业级系统还需要考虑高并发对系统响应时间的影响。合格的企业级商用系统是能够保证高并发的情况下，依旧能够带给用户较为舒适的响应环境。

而大多数商用系统都有一个不可缺少的模块，那就是系统的认证与权限管理模块。认证即系统用户的身份认证，权限管理即判断某个用户是否有对某类资源的某种操作权限。这一模块所需要的两个功能都离不开系统与数据库之间的数据传递。因此我们项目所需要解决的主要问题就是针对企业级系统的认证与权限管理模块，应该如何在与数据库交互层面进行响应时间的优化。

3.针对复杂工程问题的方案设计与实现

3.1. 针对复杂软件工程问题而开展设计

3.1.1. 应用场景设计

在用户认证授权模块中，我们从基于RBAC的资源授权访问控制开始。在SAP实习期间，我接触了SAP一个独特的协助服务方式。在SAP内部有一个专为SAP员工解决一系列问题的部门，而该部门通过员工提出的服务请求工单（service tickets），或者称之为服务票据来辅助员工与部门人员的沟通以及对服务过程的程序化处理。因此在我们的资源授权访问控制系统中，我们以Service Tickets为资源对象进行后续模块的设计、开发与测试工作。

3.1.2. 资源访问控制模型设计

基于SpringBoot和Restful API设计理论，我们实现了四种主要的服务票据的API，包括 “getAllTicket()”、 “getOwnTicket()”、 “getTicket()”、 “addTicket()”、“updateTicket()”和 “delTicket()”。如表3-1所示。

表3-1 API设计细节

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **API** | **请求参数** | **返回值** | **请求方法** | **说明** |
| getAllTicket() | 无 | Ticket list | GET | 获取系统中所有票据 |
| getOwnTicket() | 无 | Ticket list | GET | 获取登录用户票据 |
| getTicket() | Ticket ID | Ticket | GET | 获取单个票据 |
| addTicket() | Ticket’s Info | Ticket ID | POST | 新增票据 |
| updateTicket() | Ticket’s Info | Ticket ID | PATCH | 修改票据信息 |
| delTicket() | Ticket ID | 无 | DELETE | 删除单个票据 |

同时，为了探寻实现资源访问控制的最佳方法，我们还设计了三种用户角色以及多个用户权限。

其中三种角色为经理、处理人员以及客户：

1. 票据经理：经理是这个服务票据管理系统的管理者，他能够进行票据的所有操作。

2. 票据处理人员：客户服务的处理员 ，他只是能够读取票据的内容和更新票据的状态。

3. 客户：服务票据系统的客户，在服务过程中能够创建服务票据，并查询票据信息。

因为项目所针对的资源只有一种，所以权限是基于票据的操作。该项目中还会有其他资源加入，以及权限还包括对用户信息数据的操作权限。

因此，关于票据的操作如下有五个权限（主要为其中四大权限）：

1. 服务票据创建：在系统中创建一个新的服务票据，通常情况下，服务票据是由客户创建的，所以客户和经理都有这个权限。

2. 服务票据读取：获取服务票据的信息，如创建时间，创建该票据的用户和服务票据的内容。几乎所有角色都有这个权限。

3. 票据更新：更新服务票据的信息，如创建时间、创建该票据的用户、服务票据的内容等。更新服务票的信息，如该服务票的状态。几乎所有角色都有此权限。

4.票据删除：删除系统中的服务票。从系统中删除服务票据。只有管理人员有此权限。

5. 票据清空：删除系统中的所有票据，这对于这个系统来说是一个非常危险的操作，所以我们在开发和测试中不使用这个权限来执行这个操作。所以我们不使用这个权限，在开发和测试中执行这个操作，但它的存在是有原因的，只有管理者拥有这一权限。

3.1.3. 数据库设计

在数据库设计方面，我们设计了六张数据库表，分别是Users、Roles、privileges、privilege\_role、role\_user以及service\_tickets。

部分表的字段详细信息见下表：

表3-2 数据库设计细节（1.users）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| User\_id | Int(11) | Unique ID |
| Username | Varchar(50) | The username of user |
| Password | Varchar(100) | The password of user |

表3-2 数据库设计细节（2.roles）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| Role\_id | Int(11) | Unique ID |
| Name | Varchar(50) | The name of role |

表3-2 数据库设计细节（3.privileges）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| Privilege\_id | Int(11) | Unique ID |
| Name | Varchar(50) | The name of privilege |

表3-2 数据库设计细节（4.service ticket）

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| Id | Int(11) | Unique ID |
| Updated\_time | Timestamp | Create time or updated time |
| User\_id | Int(11) | Id of creator |
| Content | Varchar(255) | Content of ticket |

除上述四张表外，另外两张为多对多关联表，因此不在此列出其详细内容。

3.2. 针对复杂工程问题的具体实现、测试以及结果分析

3.2.1. 项目数据库初始化以及生成测试数据

在项目开始前，我们准备了两个相同的数据库，数据库一“java\_reference”用于项目开发使用，二数据库一结构完全相同的数据库二“java\_reference\_test”则用于项目测试。两个数据库在数据库结构/表结构等方面完全相同，仅表中数据量以及数据内容差异较大。如下则是开发数据库创建脚本代码（测试数据库类似）：

代码 3-1 MYSQL脚本代码

|  |
| --- |
| CREATE TABLE `privileges` (  `privilege\_id` int(11) NOT NULL,  `privilege\_code` varchar(50) NOT NULL,  `description` varchar(50) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  CREATE TABLE `privilege\_role` (  `id` int(11) NOT NULL,  `role\_id` int(11) NOT NULL,  `privilege\_id` int(11) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  CREATE TABLE `roles` (  `role\_id` int(11) NOT NULL,  `name` varchar(50) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  CREATE TABLE `role\_user` (  `id` int(11) NOT NULL,  `role\_id` int(11) NOT NULL,  `user\_id` int(11) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  CREATE TABLE `service\_ticket` (  `id` int(11) NOT NULL,  `update\_time` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp(),  `user\_id` int(11) NOT NULL,  `content` varchar(255) DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  CREATE TABLE `users` (  `user\_id` int(11) NOT NULL,  `username` varchar(50) NOT NULL,  `password` varchar(100) NOT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8;  ALTER TABLE `privileges`  ADD PRIMARY KEY (`privilege\_id`);  ALTER TABLE `privilege\_role`  ADD PRIMARY KEY (`id`);  ALTER TABLE `roles`  ADD PRIMARY KEY (`role\_id`);  ALTER TABLE `role\_user`  ADD PRIMARY KEY (`id`);  ALTER TABLE `service\_ticket`  ADD PRIMARY KEY (`id`);  ALTER TABLE `users`  ADD PRIMARY KEY (`user\_id`); |

Service ticket测试数据插入脚本如下：

代码 3-2 数据插入SQL脚本

|  |
| --- |
| DROP TABLE IF EXISTS `service\_ticket\_memory`;  CREATE TABLE `service\_ticket\_memory`(      `id` int(11) AUTO\_INCREMENT,    `update\_time` timestamp NOT NULL DEFAULT current\_timestamp() ON UPDATE current\_timestamp(),    `user\_id` int(11) NOT NULL,    `content` varchar(255) DEFAULT NULL,    PRIMARY KEY(`id`)  ) ENGINE = MEMORY AUTO\_INCREMENT = 1 DEFAULT CHARSET = utf8;    CREATE  PROCEDURE add\_st\_memory(IN n INTEGER)  BEGIN    DECLARE i INT;    set i = 1;      WHILE (i <= n ) DO        INSERT into service\_ticket\_memory(id, update\_time , user\_id, content ) VALUES (NULL,now(),cast( floor(rand()) as int)+1, CONCAT('testdata', id) );        set i=i+1;      END WHILE;  END;  call add\_st\_memory(100);  INSERT INTO service\_ticket SELECT \* FROM service\_ticket\_memory; |

3.2.2. 框架搭建

框架搭建部分，主要包括后端接口逻辑的实现，以及Spring Security以及Spring Boot 的配置工作等等。

项目框架采用Spring boot + Spring security搭建。Spring boot框架下编写完成dao层，domin层，service层以及controller层。其中domin层包含四个主要实体类，Privilege类，Role类，ServiceTicket类以及User类。用于与数据库交互的Dao层包括PrivilegeDao，ServiceTicketDao以及UserDao，分别负责对应数据库表的增删改查操作。二Service层则主要为ServiceTicketService 以及用于实现操作权限管理的PrivilegeCheckService。二Controller层则是Restful API接口的前台实现，这一部分浇在下一节中详细说明。而借助Spring security所完成的工作则是用户认证功能，详细实现见如下WebSecurityConfig类:

代码 3-3 WebSecurityConfig类

|  |  |
| --- | --- |
| |  | | --- | | package corp.sap.internal.exp.config;  import ……  @EnableGlobalAuthentication  @Configuration  public class WebSecurityConfig extends WebSecurityConfigurerAdapter {      @Autowired      AuthenticationEntryPointImpl authenticationEntryPoint;      @Autowired      AuthenticationSuccessHandlerImpl authenticationSuccessHandler;      @Autowired      AuthenticationFailureHandlerImpl authenticationFailureHandler;      @Autowired      LogoutSuccessHandlerImpl logoutSuccessHandler;      @Bean      public UserDetailsService userDetailsService(){          return new UserDetailServiceImpl();      }      @Override      protected void configure(AuthenticationManagerBuilder auth) throws Exception{          //configure the way of auth          auth.userDetailsService(userDetailsService()).passwordEncoder(passwordEncoder());      }      @Override      protected void configure(HttpSecurity httpSecurity) throws Exception{          //HTTP configuration, includes login and logout, exception and session management and soon          httpSecurity                  .httpBasic()                  .and()                  .authorizeRequests()                      .antMatchers("/","/login").permitAll()                      .antMatchers("/api/v3/ticket/\*\*").authenticated()                      .and()                  .logout()                      .permitAll()                      .logoutSuccessHandler(logoutSuccessHandler)                      .and()                  .sessionManagement()                      .maximumSessions(1);          httpSecurity.csrf().disable();      }      @Bean      public PasswordEncoder passwordEncoder(){          return new BCryptPasswordEncoder();      }  } | |

3.2.3. 接口实现

操作接口初步设计为四个接口，而这四个接口采用的是四种不同的请求方法。依据Restful API规范编写代码如下：

代码 3-4 ServiceTIcketController类

|  |  |
| --- | --- |
| |  | | --- | | package corp.sap.internal.exp.controllers.v3;  import ….  @RestController @RequestMapping(path = "/api/v3/ticket") public class ServiceTicketController {  @Autowired  private ServiceTicketService serviceTicketService;   @GetMapping("/getAllTicket")  public Object getAllTicket() {  return ResponseWrapper.success(serviceTicketService.getAllTicket());  }   @GetMapping("/getOwnTicket")  public Object getOwnTicket(Authentication auth) {  Integer userId = ((User) auth.getPrincipal()).getId();   return ResponseWrapper.success(serviceTicketService.getTicketByUserID(userId));  }   @GetMapping("/{id}")  public Object getTicket(Authentication auth, @PathVariable(value = "id") Integer id) {  Integer userId = ((User) auth.getPrincipal()).getId();  List<ServiceTicket> rt = serviceTicketService.getTicketByTicketId(userId, id);  if(rt == null) return ResponseWrapper.fail(ProcessingStatusCode.NO\_PERMISSION);  return ResponseWrapper.success(rt);  }   @PostMapping("/")  public Object addTicket(Authentication auth, @RequestBody ServiceTicket serviceTicket) {  Integer userId = ((User) auth.getPrincipal()).getId();  List<ServiceTicket> rt = serviceTicketService.addTicket(userId, serviceTicket.getContent());  if (rt == null) return ResponseWrapper.fail(ProcessingStatusCode.NO\_PERMISSION);  return ResponseWrapper.success(rt);  }   @PatchMapping("/{id}")  public Object updateTicket(Authentication auth, @PathVariable(value = "id") Integer id, @RequestBody ServiceTicket serviceTicket) {  Integer userId = ((User) auth.getPrincipal()).getId();  Object rt = serviceTicketService.updateTicket(id, userId, serviceTicket.getContent());  if(rt == null) return ResponseWrapper.fail(ProcessingStatusCode.NO\_PERMISSION);  if (((List<ServiceTicket>)rt).isEmpty()) {  return ResponseWrapper.fail(ProcessingStatusCode.PARAM\_NOT\_VALID);  }  return ResponseWrapper.success(rt);   }   @DeleteMapping("/{id}")  public Object delTicket(Authentication auth, @PathVariable(value = "id") Integer id) {  Integer userId = ((User) auth.getPrincipal()).getId();  Object rt = serviceTicketService.delTicket(id, userId);  if (rt == null) return ResponseWrapper.fail(ProcessingStatusCode.NO\_PERMISSION);  if ((Integer) rt == 0) {  return ResponseWrapper.fail(ProcessingStatusCode.PARAM\_NOT\_VALID);  }   return ResponseWrapper.success(rt);  } } | |

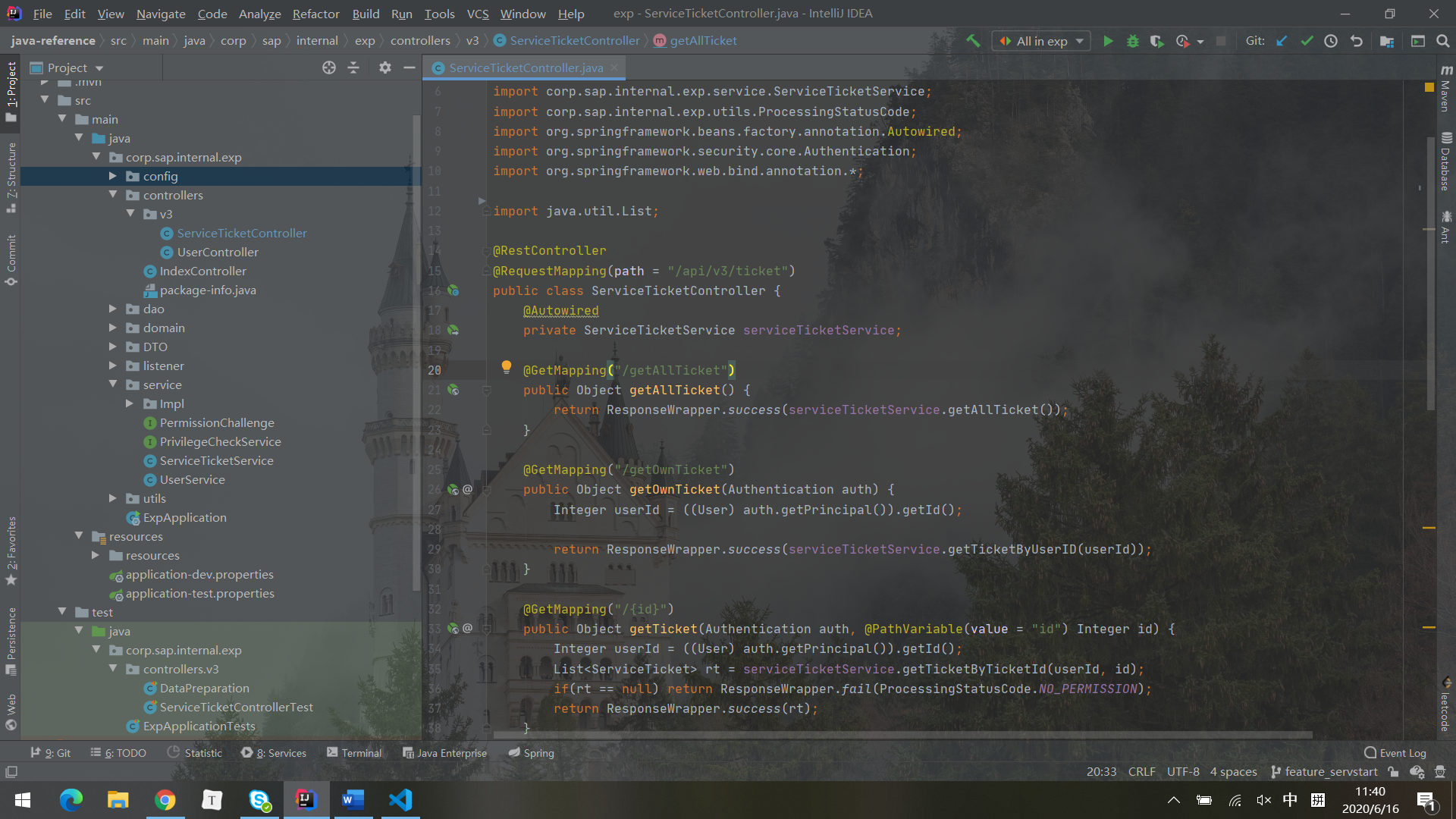


图 3-1 项目截图

3.2.4. 基于Redis的缓存机制优化

通过对Redis充分了解后，我们发现Redis对于大量的数据，其缓存他机制可以缓解接口（数据库）的压力，因此我们考虑通过基于Redis的缓存机制来提高在用户认证与授权过程中对查询的响应时间。

我们最初考虑的缓存思路为执行查询，若缓存中存在数据，则查询缓存，如果缓存中不存在数据，那么再查询实时接口。我们通过模拟缓存机制的方法，测试五个并发查询的情况。当五个并发同时运行的时候，第一个查询实时服务，而其余的则查询缓存，结果显示，反应速度较无缓存有所提高。但是，我们发现当我们的实时接口查询速度很慢的时候，就会导致，五个并发查询都会去直接查询接口服务，而不会仅一个查询接口，其余并发访问查询缓存。这样一来，缓存的作用就非常小了。而如果对于相同的请求，能够控制其等待一段时间后，再去执行，即得到第一次查询后存入缓存数据库，这种方式下的查询效率就会相比之前有极大的提升。但是，这种方法会导致如果缓存失效，所有其他并发查询的等待就会降低系统的响应时间。因此我们又产生了两种解决方法，第一种是在缓存存在的时间里进行异步查询以更新缓存，而第二种方法则是使用二级缓存，在一级缓存失效的时候读取二级缓存，二级缓存异步更新。经过初步验证发现，这两种方法能够解决之前存在的问题，但仍需进一步的讨论优化，从而解决上文提出的主要问题：在企业级系统的认证与权限管理模块优化系统响应时间。

3.2.5. 项目测试以及结论

项目测试采用的是基于Junit以及MockMvc等工具进行测试工作，为了验证四种方式对用户资源授权访问过程执行速度的影响。这四种方式分别是POJO Query（通过java逻辑进行Join）、JOIN Query（通过数据库SQL语句进行Join）、POJO Query with Cache（调用Redis作为缓存的POJO Query）以及JOIN Query（调用Redis作为缓存的JOIN Query）。

为了测试删除票据接口以及避免ID错误以及ID不存在等问题，我们设计提前插入100条数据，同时使得ticket id 等于创造票据的user id。

在进行权限检查测试之前，我们通过创建和初始化测试数据库的SQL脚本，准备好权限、角色以及权限和角色之间的关系的数据。

为了测试方法的效率，我们需要准备大量的用户，同时我们也需要轻松控制用户的数量。因此，我们通过Java类方法将用户数据插入到用户表中。

以一百个用户为例，每个用户都有一个用户ID，用户名和密码，还有一个暂时的角色（因为可能有些用户将来会有多个角色）。

我们这样给用户分配了三种不同的角色，具体如下。

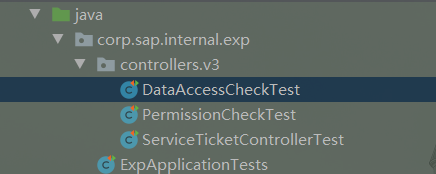
1.将所有用户按照用户id分成三组，从一到一百。

2.如果`mod(userId，3)==0`，该用户将被分配为客户，如果`mod(userId，3)==1`，该用户将是经理。而其他的将是处理者。

3. 根据最后一步，将用户和角色之间的关系插入到相应的表中。

此次项目的测试包含四个测试类，测试类包截图如下图所示为：

图3-2 测试类结构



其中ServiceTicketControllerTest类主要负责测试对票据的CRUD操作：

代码 3-5 ServiceTicketControllerTest类测试方法

|  |
| --- |
| @Test public void testCRUDServiceTicket() throws Exception {   String contentTest = UUID.randomUUID().toString();  String contentUpdated = UUID.randomUUID().toString();   // CREATE  MvcResult mvcResult = mockMvc  .perform(post("/api/v3/ticket/")  .content(mapper.writeValueAsString(new ServiceTicket(contentTest)))  .contentType(MediaType.APPLICATION\_JSON)  .with(httpBasic("admin", "123456")))  .andExpect(status().isOk())  .andExpect(jsonPath("data.content").value(contentTest))  .andReturn();   Integer ticketId = JsonPath.parse(mvcResult.getResponse().getContentAsString()).read("data.id", Integer.class);   // READ  mockMvc.perform(get("/api/v3/ticket/{id}", ticketId).with(httpBasic("admin", "123456")))  .andExpect(status().isOk()).andExpect(jsonPath("data.content").value(contentTest));   // UPDATE  mockMvc.perform(patch("/api/v3/ticket/{id}", ticketId).content(mapper.writeValueAsString(new ServiceTicket(contentUpdated))).contentType(MediaType.APPLICATION\_JSON).with(httpBasic("admin", "123456")))  .andExpect(jsonPath("data.content").value(contentUpdated));   // READ again  mockMvc.perform(get("/api/v3/ticket/{id}", ticketId).with(httpBasic("admin", "123456")))  .andExpect(status().isOk()).andExpect(jsonPath("data.content").value(contentUpdated));    // DELETE  mockMvc.perform(delete("/api/v3/ticket/{id}", ticketId).with(httpBasic("admin", "123456")))  .andExpect(status().isOk());   // READ again  // But in fact server should return 404  mockMvc.perform(get("/api/v3/ticket/{id}", ticketId).with(httpBasic("admin", "123456")))  .andExpect(status().isOk()).andExpect(jsonPath("data").isEmpty());  } |

除此之外的其他测试代码就不一一列举。

如下是Jenkins中项目的构建过程。

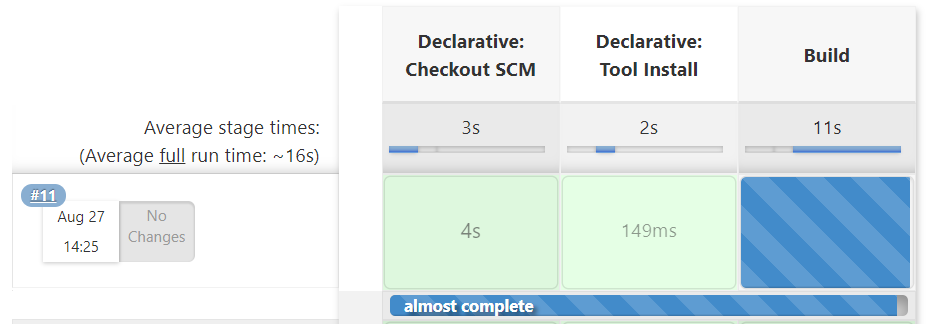


图3-3 Jenkins运行截图

在测试代码完成编写后，我们通过采用不同的授权方法测试不同的数据量发现，在数据库与服务器分离以及网络带宽足够时，POJO Query with Cache的方式下性能是最优的，因此这一项目也为日后基于RBAC辅以Redis的用户认证与授权模块的发展思路提供了一些帮助。

4.知识技能学习情况

4.1. Spring框架和Spring MVC框架

Spring框架以及Spring MVC是Java语言中五大传统框架中的核心业务层框架以及控制层框架。Spring框架的核心思想则是IOC（控制反转）、DI（依赖注入）以及AOP（面向切面编程）。而在实习项目中，使用最多的Spring思想是IOC（控制反转）和DI（依赖注入），但是在相关知识学习的过程中，为了保证后续项目的顺利，也大致了解了AOP这一核心思想即在某一行为执行前后加入相应操作，例如记录日志，配置事务等。

IOC即控制反转，与之相对的则是正向控制。正向控制即当需要用到某个类对象时，手动实例化。而控制反转则是设计编写好的类不是在其他类中显式实例化的，而是把这个工作交给了Spring框架的IOC容器。IOC会判断一个对象的实例化是否需要依赖另一个对象，而这一依赖关系则需要在配置文件中手动配置。IOC容器会把某个类和该类所依赖对象类都实例化之后再注入，而这也就是Spring的另一个核心思想，即依赖注入。依赖注入所提倡的接口编程使得在项目启动的时候，Spring会根据配置文件或是注释扫描并创建所需要的类放入IOC容器，而编程人员只需要通过Spring规定的方式获取就好了。Spring思想是将应用程序的主动性转变为被动性。以前的设计模式中，我们需要的类需要手动实例化，这一方式也导致类之间的耦合度会越来越高，每一个类都变得不可分割，从而导致不便于测试。然后Spring出现后，凭借其DI以及IOC的核心思想，这一困境得以解决。

正因为Spring的特性思想，我们可以将Spring 和Spring MVC这类控制层框架，或者再和Mybatis这类持久层框架整合，将Controller层，Mapper接口层，Serviec逻辑层，Dao层等都交予Spring统一管理。服务启动后，前台发送请求。控制层收到请求，根据URL在IOC容器中找到相应的Controller类和方法，Controller层因为注入了Service层，所以能够调用Service层的类。而后同样因为依赖注入，Service层调用Dao层。最后由Dao层对数据库操作，得到数据后返回到前台，至此前后端的交互就通过URL接口的方式实现了。在此期间的一切Controller层，Service层，Dao层中的类都是由Spring进行实例化，而不需要我们进行显式实例化。这一过程大大降低了程序的耦合程度。

4.2. 数据库操作方法：JdbcTemplate

在工程项目中，无法避免的就是和数据库进行交互。在以前的项目工程中，大多数时间使用的是数据层持久化框架，例如之前工作室开发的工程项目使用的便是Mybatis 或者Jpa这类持久化框架。而在本次的企业工程项目中，根据项目需要，使用的是和Spring有很好融合性的JDBC以及相关JdbcTemplate接口及方法。JdbcTemplate将我们使用的JDBC的流程封装起来，包括了异常的捕捉、SQL的执行、查询结果的转换等。Spring大量使用Template Method模式来封装固定流程的动作，XXXTemplate等类别都是基于这种方式的实现。

除了大量使用Template Method来封装一些底层的操作细节，Spring也大量使用callback方式类回调相关类别的方法以提供JDBC相关类别的功能，使传统的JDBC的使用者也能清楚了解Spring所提供的相关封装类别方法的使用。

在此次项目中，需要比较采用两种JdbcTempla操作方式 的操作性能，其中第一种就是简单的例如Jdbctemplate.query() 或是Jdbctemplate.update()等一次Sql操作的运行。而第二种则采用批处理的方式运行SQL脚本。这两种方法在时间上，一般是第二种会优于第一种，以为批处理方式下的数据库连接只操作一次。而单条SQL语句的执行模式下，每一次的执行都需要去连接数据库或是启动一次事务，从而导致不必要的网络I/O消耗。

4.3. 基于Spring的安全管理框架Spring Security

简单来说，Spring security所解决的问题就是用户登录认证以及用户操作授权即权限验证工作。而在我们的项目中，我们必须用到的是Spring Security提供的用户认证，使除根路径外一切API都需要用户认证而禁止一切匿名用户访问。而另一验证功能只在初期模型搭建时有所使用。鉴于Spring Security体系强大且繁杂，所学内容无法完全展现出来，因此就仅描述所学Spring Security框架中的部分核心组件。而恰恰是所有核心组件，它们之间的依赖构建起了整个Spring Security框架。

首先学习到的组件是SecurityContextHolder，该组件用于存储安全上下文（security context）的信息。当前操作的用户是谁，该用户是否已经被认证，他拥有哪些角色权限…这些都被保存在SecurityContextHolder中。SecurityContextHolder默认使用ThreadLocal 策略来存储认证信息。看到ThreadLocal 也就意味着，这是一种与线程绑定的策略。Spring Security在用户登录时自动绑定认证信息到当前线程，在用户退出时，自动清除当前线程的认证信息。但这一切的前提，是你在web场景下使用Spring Security，而如果是Swing界面，Spring也提供了支持，SecurityContextHolder的策略则需要被替换。

有时编程人员需要在程序运行时，得到用户认证信息以及用户身份信息等，因此另一Spring Security的核心组件Authentication就显得尤为重要了。该组件接口的源码如下所示：

1. 代码 1-1 Authentication接口源码

|  |  |
| --- | --- |
| |  | | --- | | package org.springframework.security.core;// <1>  public interface Authentication extends Principal, Serializable { // <1>  Collection<? extends GrantedAuthority> getAuthorities(); // <2>  Object getCredentials();// <2>  Object getDetails();// <2>  Object getPrincipal();// <2>  boolean isAuthenticated();// <2>  void setAuthenticated(boolean var1) throws IllegalArgumentException; } | |

Authentication是spring security包中的接口，直接继承自Principal类，而Principal是位于java.security包中的。可以见得，Authentication在spring security中是最高级别的身份/认证的抽象。由这个顶级接口，我们可以得到用户拥有的权限信息列表，密码，用户细节信息，用户身份信息，认证信息。authentication.getPrincipal()可以返回一个Object，然后又将Principal强转成了Spring Security中最常用的UserDetails，这在Spring Security中非常常见，接口返回Object，使用instanceof判断类型，强转成对应的具体实现类。接口详细解读如下： getAuthorities()，权限信息列表，默认是GrantedAuthority接口的一些实现类，通常是代表权限信息的一系列字符串。getCredentials()，密码信息，用户输入的密码字符串，在认证过后通常会被移除，用于保障安全。getDetails()，细节信息，web应用中的实现接口通常为 WebAuthenticationDetails，它记录了访问者的ip地址和sessionId的值。getPrincipal()，最重要的身份信息，大部分情况下返回的是UserDetails接口的实现类，也是框架中的常用接口之一。

4.4. RBAC权限管理模型

RBAC基于角色的权限访问控制是商业系统中最常见的权限管理技术之一。除此之外，权限模型还包括ACL（访问控制列表），ABAC（基于属性的权限控制），PBAC（基于策略的权限控制）。RBAC是一种思想，任何编程语言都可以实现。在RBAC中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限，这就极大地简化了权限的管理。在一个组织中，角色是为了完成各种工作而创造，用户则依据它的责任和资格来被指派相应的角色，用户可以很容易地从一个角色被指派到另一个角色。角色可依新的需求和系统的合并而赋予新的权限，而权限也可根据需要而从某角色中回收。

在计算机系统安全中，基于角色的访问控制(RBAC)或基于角色的安全是一种限制授权用户访问系统的方法。它被大多数拥有500名以上员工的企业所采用，可以实现强制访问控制（MAC）或自由裁量访问控制（DAC）。

基于角色的访问控制(RBAC)是一种政策中立的访问控制机制，围绕角色和权限进行定义。RBAC的组成部分，如角色-权限、用户-角色和角色-角色关系，使得执行用户分配变得简单。NIST的一项研究表明，RBAC可以满足商业和政府组织的许多需求。RBAC可以用来方便管理拥有数百个用户和数千个权限的大型组织的安全。虽然RBAC与MAC和DAC访问控制框架不同，但它可以在不复杂的情况下执行这些策略。

5.工程协作交流情况

在实习项目开发过程中，为在给定计划时间内完成任务目标，不仅项目组内部成员时常进行沟通交流，项目组与其他研发团队，测试组成员，整个项目的负责人也时常进行沟通。

在做测试工作时，我体会到了需求的重要性，因为测试是基于需求来的，因此需求文档一定写的规范详细，并且在做测试之前一定要仔细与写需求的人沟通以掌握测试重点，分辨bug的优先级。在做回归性测试的时候，挑选的测试用例中有些是别人写的，因此我懂得了测试用例书写规范的重要性，只有写作规范，才能有高继承性，要做到详尽但不冗长，最好的标准是达到让从没有接触过这方面测试的人无障碍的阅读使用。当遇到不能理解的测试用例时，一定要积极的与原作者沟通，有歧义的地方讨论改正，共同完成该用例的测试，这样能大大提升测试效率，并且共同改正后的测试用例更易让以后测试的人理解。

发现BUG要及时与开发人员沟通，确定该问题是否为BUG，有时某些测试环境因素导致的小问题可能并非BUG。确定问题为BUG后，积极帮助开发人员重现BUG，追踪BUG修改状态，以便更好更快的修正该错误，开展工作。

参加到Share Session这个项目中后，我在该项目中担任UI的角色，因此充分体会到了跟需求人员沟通的重要性。如果沟通不善，导致UI设计不符合需求或者不符合客户预期，就要返工重做，平均一个界面返工3次。所以后期的一些界面我都是找需求人员再三确定再开工的。磨刀不误砍柴功，虽然之前沟通花了点时间，但是返工的次数变少大大降低了时间成本。

小组内部每天都会开Scrum Meeting，促进组员间沟通并同步状态，这是一种沟通效率很高的模式，并且成功打破隔阂，使我受益匪浅。

6.工程计划管控与执行情况

6.1. 针对复杂问题的执行计划安排

经过这次实习，我明白了预先制定计划的重要性。在每进行一个项目之前，负责人都会告诉我们项目的时间表，有时精确到周计划，有时精确到日计划，已经规定了时间的任务坚决不能拖延，大家也都严格遵守时间线。通过提前制定精确的计划，可以使每天的工作都很有目的性，效率也就大大的提高了。

以下为QA测试步骤，我测试时就按照以下步骤进行，既能明确工作内容同时还大大提高了效率，并且能够更好地与他人合作：

Sprints阶段：

开发人员通过SVN number在VersionOne上更新BIs/defects的状态

QA通过Jenkins检测QA系统释放的版本号

QA测试新功能并在VersionOne上更新测试状态

QA在VersionOne上记录下defects的具体内容并通知开发人员测试结果

在本地书写测试文档并交由SFSF module team审核

在TestLink上书写测试用例

回归测试阶段：

1.把所有还未解决的在VersionOne中的defects转移到JIRA中

2.准备回归测试计划，挑选测试用例，开始回归测试

3.找到的defects上报JIRA

4.上传测试文档到JIRA ticket上。

6.2. 存在的主要问题及应对项目风险与挑战的能力培养情况

在项目主要的存在的问题第一就是自身的问题。因为在此之前没有任何大型的项目经历，所以在许多情况下不能在设计上做出比较优化的解决方案，所以在设计方案时，很多情况下需要向资深的developer或者arch进行讨论确定，在一定程度上会造成时间上的浪费。与此同时，遇到技术性难题的时候，因为没有经验，在很多情况下，没有确切的解决方案。在网上进行解决方案进行搜索的时候，很多情况下并不符合我遇到问题的情况。其次就是对于敏捷开发模式刚开始的不适应，以及与同事之间的交流合作一开始也有一些障碍，因为有时候我不能很好的理解他们的意思。经过一段时间的学习与培养，当我遇到问题的时候，先尝试着自己去解决，并在网上进行检索，学会找到自己需要的内容。如果实在没有解决方案，或者合适的方案，我则会去和arch进行讨论，然后进行确认，先把自己的想法告诉他，他给予相应的建议，我从中找出自己的不足，不断进步。与此同时，因为实习期间很多情况下时间比较紧急，有很多的技术原来没有接触过，但是因为项目的需要，所以需要再很短的情况下就要学会使用，在这种情况下，大大提高了我的学习能力。这些对于作为实习生应对项目风险都有极大的作用。同时，因为自己的不断努力，提高自己，能适应项目不断变化的需求挑战，然后制定更优化的解决方案。

7. 职业素养与工程伦理的学习与培养

首先谈一下我对软件工程系统的质量的学习与认识。我认为质量是一个企业的代名词，质量达不到要求，客户的体验不好，自然会质疑公司的能力。而且前期不做好质量，做到版本的后期会越做越痛苦，因为发现原来做的系统因为性能问题，或者扩展性问题而越做越痛苦，我自己就在这地方挖坑，因为前期结构设计的不合理，导致在继承测试阶段性能没有达标而进行重构。并且如果在开发阶段，引入一个和其他地方关联性比较强的问题，一直没被发现，然后几个版本之后发现，那么可能很多代码都是基于这个错误的逻辑继续开发的，到时候修改起来，很可能牵一发而动全省。所以软件质量十分重要。通过学习，我认识到要想软件工程系统质量完成的比较好，需要做到以下几点。

第一就是质量意识的培养。即使个人拥有质量意识，但是整个团队都拥有质量意识是比较困难的。所以大家应该培养共有的质量价值观，从企业文化中将质量意识植入人心，有了共同的价值才能更好的执行规则。第二就是应该有清晰的质量目标和向导，就是量化好什么样的代码是质量好的，比如每千行代码少于多少个bug，bug要根据影响划分出不同级别。质量一定要作为一个评价研发人员工作或绩效的重要因素。此外，重要的地方请重要的人把关。假设你现在有一个10个普通级别的人的开发团队，我建议换成一个1个大牛级别+5个普通级别的团队，哪怕花5倍的工资请一个大牛，然后要他做核心的框架、系统设计、重点问题公关和保证其他人开发的质量把控。有些地方如果做的不好修改代价可能会比较小，比如某个程序写的不好，或某个独立的功能有问题，但是有的地方代价会很大，比如架构设计不合理，或系统设计有问题。而且如前言所说，软件开发越是靠前的步骤出问题，带来的代价就会越大，而且会大的明显。还有一点很重要，就是要有编码规范。大型软件的编写不是一个人的事，定位、修改问题，也不是只改自己的部分，我们甚至定位的大部分问题都是老版本和别人的问题，这个时候，如果大家没有一个统一格式的编程规范，就会很难读懂别人的逻辑。而且好的编程规范确实能让我们的程序可读性更好，更少的犯错误。因为我刚开始工作的时候，就是代码相对不是很规范，被架构教育之后去调整的。还有就是要构建自动化测试环境。我们QA同事每天都会跑自动化测试脚本，这样就很好的能发现问题，并定位，提高软件的质量。

关于软件工程系统的环境即是构筑一个新软件时所依赖的条件工具和基础设施等，包括软环境和硬环境。软件工程系统的环境具有多维性，表现在不仅要集成与软件开发技术相关的工具，还要集成与支持技术、管理技术相关的工具，并将它们有机地结合在一起。在这一点上，我们公司做的就很不错，在支持技术上架构提供更好的方案。团队内部布置有SM、PO、PM、QA、developer人员，管理与技术很好的结合在一起。

关于软件工程的职业健康、安全和服务意识，对于我自己的理解是在项目中的感受。在我们的开发项目中，我们设置了DPP日志审查。所有对用户的操作都会记录下来，生成日志，并将相应的操作存储到数据库里面。做为平台管理员可以可以在数据隐私保护中查看相应的操作。与此同时，我们的PM也是积极与客户方面进行交流，有什么需求变更，我们PO就会及时的写成story，发起change request请求，然后进行开发。并且在UAT阶段之前，我们也是给客户进行培训整个系统的使用与注意事项。

我对职业道德和规范的理解是爱岗敬业，诚实守信，素质修养，忠于职守，乐于奉献。尊职敬业，是从业人员应该具备的一种崇高精神，是做到求真务实、优质服务、勤奋奉献的前提和基础。敬业奉献是从业人员的职业道德的内在要求。实事求是，一票否决。实事求是，不光是思想路线和认识路线的问题，也是一个道德问题，而且是统计职业道德的核心。依法行事，严守秘密。把职业道德渗透到工作的各个环节，融于工作的全过程，增强人们以德的意识，从根本上消除腐败现象。公正透明，服务社会。优质服务是职业道德所追求的最终目标，优质服务是职业生命力的延伸。

在实习期间，我爱岗敬业，遵规守纪，无早退迟到行为。并乐于奉献，积极学习，全身心投入到工作之间。遇到困难不退缩，不畏惧，多向前辈请教，在克服困难中不断进步。实事求是，坚持自己的想法，遇到和别人意见不同时，善于进行沟通交流。

8. 对软件工程实践以及软件工程领域发展的认识

随着计算机的广泛应用，信息产业悄然兴起，越来越成为经济发展中的先导产业。目前，信息化已从整体上引导着世界经济和社会发展，信息处理技术已成为现代社会生产力、竞争力的关键，具有十分重要的战略位置。这一期间，计算机软件以其独特的知识形态，广泛渗透于国名经济建设的各行各业中，对信息采集、信息处理、信息交换以及生产过程、社会生活的变革产生了深远的影响。计算机软件，尤其是软件工程作为软件产业的依托学科已经成为信息技术的核心之一。

参考其他成熟工作领域中的做法，软件工程也研究管理学理论在软件工程中的应用。软件项目管理方法抛弃了以前个人的作坊式开发方式，根据管理科学的理论，结合软件产品开发的实际，保证工程化系统开发方法的顺利实施。为了使软件项目的能够按照预定的成本、进度、质量顺利完成，软件管理方法对成本、人员、进度、质量、风险、文档等进行分析、管理和控制。进行软件项目管理有利于将开发人员的个人开发能力转化成企业的开发能力，企业的开发能力越高，表明这个企业的软件生产越趋向于成熟，其开发风险也越小。软件项目管理已经是公认的软件开发企业的核心竞争力之一。

软件工程发展趋势：

1.模块化：

随着全球化的发展趋势和全球化市场竞争压力的增加，一方面企业需要更多的业务灵活性和创新能力；另一方面企业面临的IT环境复杂度的增加和历史遗留系统的增加，对企业的IT提出了新的挑战。模块化的思想恰恰能够帮助企业从根本上解决了这一问题，它一方面通过抽象、封装、分解、层次化等基本的科学方法，对各种软件构件和软件应用进行打包，提高对企业现资产的重用水平和能力；另一方面，基于模块化思想，业界提出了SOA技术，它提供一组基于标准的方法和技术，通过有效整合和重用现有应用系统和各种资源，对各种服务进行服务组件化，并基于服务组件实现各种新的业务应用的快速组装，帮助企业很好应对业务的灵活性要求。它通过有效平衡业务的灵活性和IT的灵活性、平衡业务的灵活

性和IT的复杂度，为IT人提供了一个业务视角，让IT人学会用业务的眼睛看世界，有效的拉近了IT和业务的距离。

2.开放计算：

“开放计算”是能够确定行业基础框架，指导行业发展和技术融合的，它融合了“开放标准”、“开放架构”和“开源软件”三个方面。通过坚持“开放标准”，不同企业开发和使用的软件可以互连互通，不同的软件工程工具能够更好地集成，不同国界和不同文化能够更好地协作交流，用户的投资能够得到很好的保证。“开放架构”通过一组开放的架构标准和技术，有效地解决了商业模式的创新对IT灵活性要求的增加和现有IT环境的复杂度之间的矛盾。而“开源软件”有效地促进了开放标准的发展，同时有效利用社区驱动的开发与协作创新，优化软件设计中的网络效应，开源软件越来越被中小企业和个人用户所认可。

3.合理地开发治理：

软件开发治理的研究对象是软件开发团队，其目的是在软件开发生命周期中，通过定义整个开发组织中的各种角色、职责和技能要求，不断改进软件团队的生产效率和软件产品质量。软件开发流程和软件工程工具是软件开发治理的两个重要组成部分。软件开发过程中，不同的团队、不同的文化、不同的规模、不同的软件类型、不同的质量要求，都会影响开发团队对软件开发方法和过程的选用。

RMC就是合理的开发治理的优秀成果，基于RMC，企业可以首先将各种管理活动的最佳实践总结成流程构件，然后基于各种最佳实践流程构件建立统一的方法库。通过重用方法库中的各种流程构件，面向不同类型的、不同规模的IT组织或软件团队，企业能够快速地量身定做出适用的IT治理流程或软件开发流程，保证了流程的灵活配置能力。

4. 全球化软件协作交付：

全球化的世界必然带来全球化的软件交付模式。

企业的合并和收购趋势不断产生众多新的分布式开发团队，企业为了提供全球化的24 x 7 支持和开发能力，也在不断加强全球化软件协作交付能力。

全球化软件协作交付的另一个重要的驱动力来自于软件外包行业的发展。纵观当今外包行业的发展，不难看到外包和全球化交付正在成为软件交付发展的标准模式。

我个人的理解，软件工程就是按照工程学的管理方式，有组织、有计划的在成本限额以内按时完成开发且能在实际机器上可靠有效地运行的软件，并不断对软件进行优化、维护及管理的工作过程。

软件业作为一个服务业，要想得到发展，首先必须形成一个对软件服务有迫切需要的市场。其次，这个市场中的消费者必须具备足够的购买力。软件的消费群体可以分为个体消费和企业消费。中国的企业群体，数量庞大，但综合质量不高，规模较大的企业极少，因此，中国的企业对大型软件的消费数量是有限的。至于个人消费，目前在中国还难以成大器。

另外，对于软件的质量很难作出绝对的评判。只有相互之间的对比，才能比较出谁优谁劣。

日新月异的科技社会，软件工程的发展是没有止境的。软件工程的发展历史本身就是一个不断提出问题又不断解决问题的过程。这一过程必然导致大量新技术新方法的不断涌现，从而推动软件工程的长足进步。

大量事实表明，为了解决软件危机，既要有技术措施，又要有必要的组织管理措施。软件工程需要从管理和技术两方面研究如何科学、正确、合理地开发和维护软件，软件开发必须遵循软件工程的思想不断精细化。

9.结束语

9.1. 课题完成情况、有待进一步解决的问题及方向

从2020年五月中旬开始在SAP进行线下实习，我在企业指导老师的帮助下独立完成了用户认证与授权模块项目的设计、开发以及测试工作。在资源授权模块，我们探索了多种可行方法，最后发现借用Redis进行缓存处理的基于RBAC模式的具有独特数据库以及接口逻辑设计的方案效果最优。而在用户认证方面除了借用Spring security提供的帮助外，我们后续还需要解决类似与单点登录等用户授权模块功能的开发。

以及在数据库底层存储结构方面，探索一种进一步优化用户认证与授权模块的方式。

9.2. 本人对于企业实习的收获及体会

在本次实习过程中收获了很多。

第一，学习了很多的知识。对于SpringBoot、SpringSecurity、JPA、MySql数据库、Jenkins、redis等技术都有了比较系统的了解与掌握。

第二，学会了更好的与别人交流合作。因为是敏捷开发，每日都会有daily meeting，同时没个sprint都会有plan meeting，在meeting room中都会进行讨论，这时候，自己的都沟通表达能力得到了显著的提高

第三，学会了合理分配并计划时间。因为项目的task完成的日期都是固定的，所以怎么合理的分配自己的时间去完成各个task十分重要。同时锻炼了自己的心理素质以及在压力状态下更高效的完成任务。

第四，学会了写出更好的文档。因为在自己的每个task都要增加comment去叙述自己完成task的解决方案与步骤，这样大大提高了我的写作能力。

参考文献

1. 埃克尔、陈昊鹏. Java编程思想[M]. 北京：机械工业出版社，2007
2. 史蒂夫·迈克康奈尔、金戈. 代码大全[M]. 北京：电子工业出版社，2006
3. 付磊、张益军. Redis开发与运维[M]. 北京：机械工业出版社，2017
4. 阎宏. Java与模式[M]. 北京：电子工业出版社，2002
5. 罗伯特·马丁、韩磊. Clean Code[M]. 北京：电子工业出版社，2012

致谢

首先我要感谢电子科技大学，感谢信息与软件工程学院以及卓越工程师计划。根据学院软件工程本科生培养方案的要求，我获得了这次难得的思爱普软件系统有限公司成都分公司的实习机会。学院和老师为我提供了一个很好的平台，使我在学习的道路上受益匪浅。

另外，我还要感谢院内实习指导老师，他的每一个意见与建议都让我在这次实习上以及未来学习道路上更进一步，并且，在我撰写报告以及其他方面给予了我极大的鼓励与帮助。

感谢我的企业指导老师，作为我实习所在小组的技术大牛，他在项目研究以及工程开发上给我提供了诸多帮助，同时也将他在企业上的开发经验以及实际开发过程中所需要注意的通用规范教授给我。

最后感谢我的父母，同学以及公司同事在我实习过程中给予我的帮助和鼓励，有了他们我的实习才能取得圆满的成功。

在此向他们表示由衷的感谢 。