

**本科生毕业论文**

**办公自动化管理系统开发**

院 系 自动化学院

专业班级 自动化1302班

姓 名 沈顺天

学 号 U201314574

指导教师 邓忠华

2017年5月15日

**学位论文原创性声明**

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

**学位论文版权使用授权书**

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

# 摘 要

本课题使用Java SSH框架开发后台应用，提供RESTful的API，使用Vue作为前端框架开发SPA（single page application）实现了于新的框架的办公自动化系统(OA系统)。达到了较传统办公自动化系统更好的用户体验，且开发方式更加灵活快捷，系统更易于扩展。

近年来,计算机科学领域出现了许多新兴开发技术, 本课题的目标是使用其中一些新技术来开发办公自动化系统,达到更好的用户体验，更好的开发体验。

在目前众多的前后端开发技术中，经过尝试和对比，笔者最终选择了Vue作为系统的前端框架，使用Java Web的SSH框架作为系统的后端框架。前端使用Vue是因为其渲染效率高，能显著提升用户体验，并且能轻松构建大型的前端单页面应用。后端使用SSH的Java web框架。在互联网技术的快速发展过程中，Java Web的后端技术被证明是坚实可靠的一套开发体系。

本课题最终实现的办公自动化系统体验明显好于传统应用，且开发代码前后端分离，工程划分明确清晰，便于管理和扩展。本课题的实现思路可以作为办公自动化系统开发或其他传统软件系统开发的一种新的实践。

**关键词：**办公自动化系统；Vue前端；SSH后端；单页面应用；RESTful API

# Abstract

The target of this project is to develop the OA system with new techniques generated in the internet tide wave. Trying to achieve better user experience, better developing experience. Also, to provide a new approach of developing traditional system.

Among the various frontend techniques, Vue is choosen as my project's frontend framework. The backend framework choosen is Java Web's SSH framework. The reason I choose Vue is that Vue performs very efficiently in the browser which can help promote user experience sharply. And Vue is really easy to use for developing complicated frontend system. The reason to use Java Web's SSH as backend framework is that it has been proved to be reliable in the developing of the backend techniques. And it will save much effort if the backend technique remains the sane when a company tries to transfer to our technique stack.

The final product of this project achieves much better user experience comparing with the traditional OA system application. And the code is separated between backend and frontend which makes the whole project has a clear division of work, and it is easier to maintain and extend. This project can be an example of using the new approach to develop traditional system application.

**Key Words：**office automation system ; Vue frontend; SSH backend; single page application; RESTful API

**目 录**

[摘 要 I](#_Toc481762544)

[Abstract II](#_Toc481762545)

[1 绪论 1](#_Toc481762546)

[1.1 研究背景和意义 1](#_Toc481762547)

[1.2 互联网技术发展及其影响 2](#_Toc481762548)

[1.3 论文主要工作和安排 4](#_Toc481762549)

[2 技术方案论证 8](#_Toc481762550)

[2.1 基于Java SSH框架的后台程序开发 8](#_Toc481762551)

[2.2 基于Vue的前端SPA开发 19](#_Toc481762552)

[2.3 本章小结 24](#_Toc481762553)

[3 办公自动化系统主要模块的实现 26](#_Toc481762554)

[3.1全局模块描述 26](#_Toc481762555)

[3.2 用户自定义首页的实现 28](#_Toc481762556)

[3.3 审批管理模块实现 30](#_Toc481762557)

[3.4 公文处理模块实现 32](#_Toc481762558)

[3.5 常用工具模块实现 34](#_Toc481762559)

[3.6 本章小结 36](#_Toc481762560)

[4 总结 37](#_Toc481762561)

[致谢 37](#_Toc481762562)

[参考文献 38](#_Toc481762563)

# 1 绪论

## 1.1 研究背景和意义

OA系统（Office Automation System）是公司在日常的运行过程中，公司员工使用次数最多的辅助工具。办公自动化系统的功能相对比较复杂，而且使用该系统的人员种类较为繁多。这些都使其用传统的开发技术开发难度较大，且用户体验较差，最终完成的系统也不易维护和扩展。

OA系统属于较为成熟的软件领域，产品迭代周期较长，应用的技术较为成熟，市面上的产品差异化较小。较为常见的自动化办公系统开发模式为纯后端开发。常见的后台开发技术有.net和Java Web。

传统的开发模式之所以称之为纯后端开发，主要是因为在这种开发模式下，用户看到的网页是使用后端技术渲染后得到的。比如在Java web中常见的JSP技术，就是通过渲染JSP文件来生成html页面和相应的JavaScript脚本以及CSS样式表，最后通过网络传输将这些数据传输给用户。这种开发方式的好处是简单直接，不需要前后端定义复杂的数据接口，大部分工作都由后端人员开发完成。缺点是后端人员工作量大，前端人员发挥的作用较少，前端人员自由度较低。前后端人员工作耦合度过大，不便于分开同时进行开发。

随着市场对于OA系统交互复杂度要求的提升，用户对系统响应速度的要求的提升，传统的技术栈已经慢慢难以满足市场的需求。可能是出于原始系统难以更换框架的原因，也可能是对于互联网技术不稳定性的考虑，这些新的开发模式和技术都还远远没有普及。

本课题以办公自动化系统的开发为切入点，但着重点并不是在于对于该系统的具体实现，只是因为办公自动化系统是传统软件行业中非常具有代表性的一个软件系统。且市场中的产品目前都还是使用的固有的重后端开发模式，市面上的办公自动化系统往往体验都比较差，项目架构极其繁重难以扩展和维护。所以本课题引入新的软件开发架构和技术，以办公自动化系统作为尝试的入口，通过实现一个用户体验良好且项目结构可扩招可维护的办公自动化系统，为该行业的从业者提供一个新的思路和参考。

## 1.2 互联网技术发展及其影响

互联网技术的发展日新月异，许许多多的技术都最先应用于改善互联网产品的体验，也有很多技术就是为了解决互联网产品发展过程中遇见的问题而产生的。OA系统的技术栈较为成熟，较为常见的自动化办公系统开发模式为纯后端开发，其中常见的后台技术为.net和Java的SSH后台框。这些成熟的框架，在传统软件领域的多年验证下，被证明为是合理且稳定的。但是技术在不断在进步，随着PHP， python等脚本语言开始抢占后端的份额，JavaScript的node.js也在后端开始发挥作用，并且凭借着其和前端JavaScript无缝的衔接，慢慢开始引起人们的注意。在快速迭代、快速开发的潮流下，传统开发模式安全稳定的优势慢慢变得不再明显，反而是其迭代缓慢，难以扩展维护的缺点暴露无遗。前后端分离的开发模式更是已经占据了互联网产品开发的主流，后端的任务被大大减轻，而前端的框架层出不穷，应用这些新技术做出的产品往往有着传统开发模式难以比拟的用户体验。

传统的OA系统开发模式固然是被验证过的，稳定且合理的，如果为了风险性考虑，企业确实应该选择较为稳妥的开发模式。所以本课题站在改进传统OA系统开发模式的角度，尝试使用新的前后端框架和技术，希望能够达到相对传统OA系统更好的用户体验，可以为需要转型技术栈的开发人员提供一些新的思路。

互联网的发展，究其目的其实还是为了解决生活中不便或者效率低下的事务。比如支付宝推出的二维码支付；共享单车利用了互联网进行单车共享实现资源最大化利用；外卖行业也是通过手机app和网页来进行外卖的销售。而这一切的产品都是基于良好的用户体验。体验不佳而且无法紧跟潮流的产品, 往往很快便会淡出了人们的生活。人们对于软件应用用户体验的标准是很高。软件界面除了美观整洁外，不能有过多的干扰信息；UI的响应时间不能超过300ms。否则用户就会认为界面卡顿或是自己操作失败。要达到这样的响应速度，传统的点击网页再刷新跳转的方式显然是不可能做到。

目前市面上办公自动化系统的体验并不好，操作起来反馈不及时，经常要等待浏览器的刷新，且无法满足移动办公的需求。而使用传统的开发模式没有办法改进用户体验。想要提升这些用户体验就必须改变现有的开发模式，而这正是本课题希望达到的目的。

在手机端，Android和iOS系统的原生应用都已经能够满足用户的体验要求。以前网页端由于性能的限制，还达不到原生应用般流畅的体验。并且前端的模块化开发一直没有很好的解决方案，导致较少有很复杂的前端产品出现。但是随着现在网页前端技术的快速发展，随着Angular，React，Vue等框架的出现，前端的模块化开发和性能都有了长足的进步。性能和前端项目建构的模块化开发不再是问题。Google的Gmail横空出世更是为人们展现出前端开发产品的无限可能。现在这些前端技术已经慢慢普及。比如我们现在使用的各种手机端的软件。仔细观察，会发现有时候打开这些app会出现类似网页加载的页面效果，其实就是因为这些app已经加入了很多前端的页面。淘宝，京东，美团这类对实时更新业务需求极大的应用， 甚至大部分页面都是前端页面，但是因为其体验越来越好，人们已经很难分辨出哪些页面是前端网页，哪些页面是原生应用。细心的用户可能会发现有些手机app提醒更新的频率越来越低，但是功能确在变化，这正是因为许多更新都直接通过前端网页热更新的形式就已经实现了。随着前端技术越来越成熟，手机端浏览器内核性能越来越强，原生应用和前端web应用的差别会越来越小甚至被抹平。在web页面之前，大多数的软件系统都是使用的客户端加服务器的模式，每当客户端代码需要更新的时候， 就要重新下载安装。而后来人民慢慢转变为浏览器加服务器的模式，就是因为前端性能慢慢增强，慢慢可以满足用户交互的性能要求。而现在随着前端单页面应用的崛起，单一的web浏览器页面能够兼具客户端性能和浏览器热更新的优点，已经在互联网应用中得到了广泛的实践。

## 1.3 论文主要工作和安排

### 1.3.1 预计达到的目标

综合以上的因素，本课题选择使用Java SSH框架开发后台应用并提供RESTful API，使用Vue作为前端框架消费API并实现前端的单页面应用。应用前后端分离开发的模式，通过定义API文档的方式，前后端分离进行开发。

目标是尝试使用新的技术栈来开发传统的软件系统，实现有较好用户体验的办公自动化系统。 实现的系统项目结构清晰， 模块划分合理，易于维护和扩展。 并最终希望通过这个项目给出一个案例，为想要转型新技术栈的传统软件开发者提供一个参考。本课题的前后端代码都公布在github上， 代码链接在文末参考处给出。

### 1.3.2 论文主要工作

论文的主要工作按照技术层面划分包含两部分。后端部分包括服务器的搭建，java后台的开发，RESTful API的设计和实现。前端部分包括Vue框架的学习， SPA应用的实现，对API的消费并实现办公自动化系统各个模块功能。

前后端技术栈中的关键技术会在后面的论文技术方案论证中详细说明，下图展示了前后端的技术栈框架。

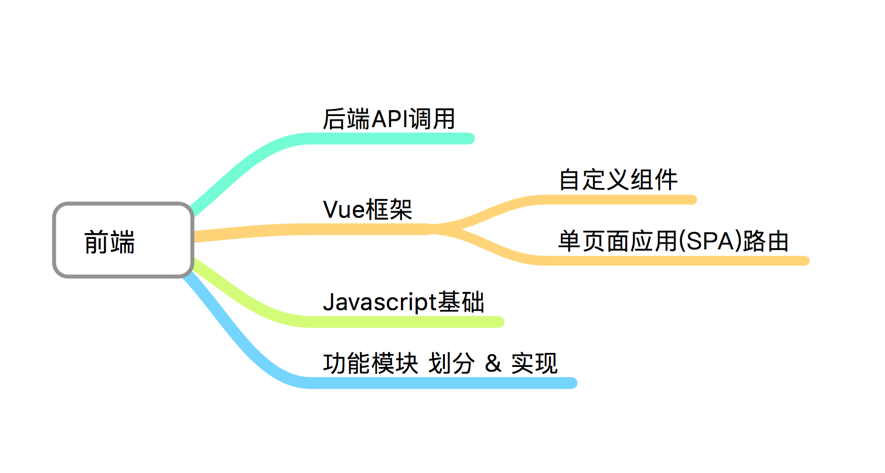


图 1.1 前端框架

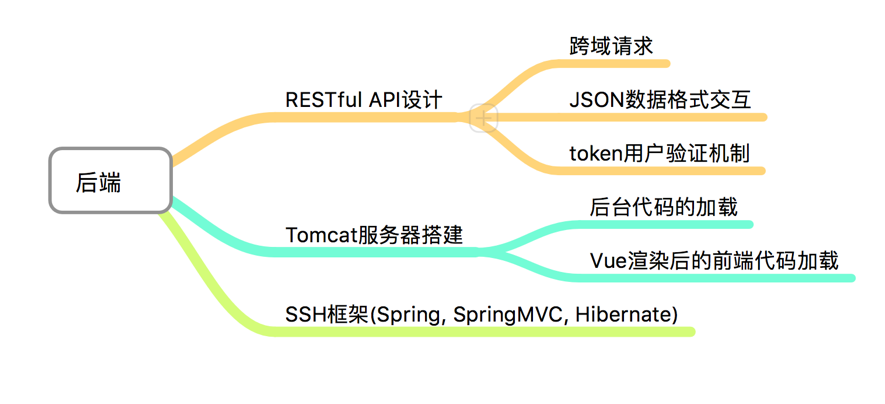


图1.2 后端框架

本课题的系统功能模块划分主要有：自定义首页，审批管理模块，公文处理模块，常用工具模块。实现的功能基本包括了办公自动化系统的常见功能。

其中自定义首页是为了满足不同使用群体对首页的个性化定制需求，多维度展现个人，部门，项目组所需要的工作空间。通过每个用户不同的配置，在服务器端存储用户的个人配置数据。在用户登录时加载这些配置信息，并用来初始化系统的首页，以此实现自定义首页的效果，从而提升个人效率。

审批管理模块通过高效的无纸化办公，提高审批的处理速度，提高员工的生产力，节省不必要的人力物力开销。该模块分为用户提交的审批和用户需要审核的审批两部分，包括了审批申请者和管理者的功能，便于上级和下级的沟通使用。每一个审批定义了不同的状态值。审批者和提交审批者通过修改状态值，来推动审批流程的进行。在审批状态变化时，相应的人员会收到通知，以保证审批流程快速有序的执行下去。

公文处理模块与审批管理模块类似，但其流程的管理更加复杂，通过系统的修改、撤销、提交、通过、终止、移交、督办 等流程管理工具，简化员工对于繁琐的公文管理条例的研究时间。通过公文实时状态监控，短信邮件提醒等工具大大提高公文处理效率。传统的公文处理需要人力将公文传达到各个部门进行审批，而且审批的进度只能通过人工的询问，效率非常低下。办公自动化系统中的公文处理模块实现了完全的线上公文处理，通过电子印章，数字签名等技术，繁杂的公文处理流程只需通过简单的数字传递即可流畅顺利的执行，大大提升了公文处理的效率。且处理好的公文也是电子形式保存的，不会丢失， 也不会损坏，即使公文处理中的某一步出错也可以轻松的回溯出错的缘由。

常用工具模块包括的功能有：会议管理工具，调查问卷工具，投票管理工具。问卷工具包括问卷制作，问卷发布，问卷填写，问卷结果报表生成等功能，基本满足调查问卷所需的常见需求。投票管理工具包括发起投票，公布投票，收集投票数据，生成投票结果报表功能。会议管理包括会议室管理，会议室申请，会议人员管理，会议提醒等功能，基本满足公司会议常见的需求。包括了企业办公常见的工具，与OA系统集成在一起更方便了与公司员工相关联，进一步提升生产效率。且常用工具还可以进一步开发扩展。



图 1.3 功能模块

### 1.3.3 论文结构安排

论文共分四章，本章介绍的是研究的背景，以及技术发展的历史和本课题希望达到的目的和效果。在后续的章节中，第二章将从技术层面对论文所做的工作进行阐述，第三章将从业务层面进行阐述。具体结构安排如下：

第二章介绍技术方案的论证。后端部分包括Java SSH框架的使用，Tomcat服务器的搭建，RESTful API设计等内容，将详细阐述如何一步步搭建办公自动化系统的后台服务。在论证的过程中将以一个简单的调查问卷功能作为例子进行详细阐述。前端部分包括Vue框架的使用，为什么选取该框架，前端模块化开发的实践等内容，也将以调查问卷功能作为实例进行详细讲解。

第三章将着重站在业务前提下。对项目的每一个业务功能模块，如调查问模块，对其实现细节进行详细的阐述。将主要的几个功能模块的实现细节和设计思路进行详细的解释。

第四章总结全文，概括毕业设计中所做的主要的工作，谈谈该项目对于目前传统软件行业的借鉴意义，还有哪些遗漏需要补足以及后续项目的维护计划。

# 2 技术方案论证

传统的办公自动化系统大多是基于Java SSH的后端工程。业务逻辑是：后端接受客户端的请求，根据客户端的请求，进行相应的数据操作，再将数据填充到准备好的JSP模板中。再将JSP渲染成.HTML文件，完成一次交互。这种交互方式缺点很明显：一是每次用户请求后刷新页面，长时间的加载和空白的浏览器页面让软件的用户体验极差；二是前端功能太弱，页面很难做出优秀的用户体验，并且每次刷新都是加载一整个页面，浪费网络带宽。

我们的课题选择使用Java SSH技术作为后台开发的框架，但并不是用来渲染页面，而是用来提供JSON数据交互RESTful api。用户界面的交互逻辑主要在Vue前端实现。将前后端的工作划分清晰，便于开发和维护。基于后台的JSON数据，客户端可以使用网页前端，android原生应用， iOS原生应用等多种方式进行开发，灵活度大大增加，且便于实现更好的用户体验。 每次客户端和服务器的交互仅仅是json数据，客户端页面在第一次加载好后不再重复刷新达到类似原生应用的体验，也大大减少了带宽的占用。

这种新型的前后端开发模式在互联网行业已经非常普遍，许多公司甚至将自己的API公开出来，让很多开发者可以开发出体验极佳的第三方客户端。而在传统软件领域，这一开发模式还尚未被广泛采用。

## 2.1 基于Java SSH框架的后台程序开发

### 2.1.1 后端SSH框架的介绍

首先对Java 的SSH开发框架进行介绍。SSH分别指的是Spring，Struts2 和 Hibernate。而Struts2因为过于臃肿现在慢慢被SpringMVC取代，本项目中也使用SpringMVC代替Struts2来进行http请求的分发管理。

Spring是三者中最著名也是最重要的，Spring提供了一种非常松散的框架支持，在不侵入用户代码的情况下提供了依赖注入和控制反转的功能。Spring还可以很好的组织SpringMVC和Hibernate进行整合开发。Spring让开发者不用去自己初始化一个对象，不用使用单例，不用担心内存泄漏等等的问题。可以专注于业务逻辑的开发。



图 2.1 spring架构

SpringMVC是作为Struts2的替代者。它实现的功能是对后台代码进行MVC形式的分层。MVC指的是Model-View-Controller(模型层-视图层-控制层)。这是一种在后台开发中被广泛采用的开发模式，有利于将不同模块的代码分隔开，便于开发和维护。SpringMVC相较Struts2，结构更简单，功能更强大，且性能更优。

如下图所示，Controller是网络请求的主要管理者，按照传统的开发模式， Controller处理完请求后会返回一个View-Model然后再后端渲染成html页面和相应的js脚本。而在我们的项目架构中，后端只需要系统JSON数据即可，所以我们将Controller的返回值直接改为responsebody，示例代码如下：

1. @CrossOrigin
2. @PostMapping(value = "/detail"， produces = { ControllerCons.PRODUCES\_UTF\_8 })
3. @ResponseBody
4. **public** String saveQuestionnaire(@RequestBody QuestionnaireEntity questionnaireEntity， HttpServletRequest request) {
5. **if** (!TokenManager.verifyToken(request)) {
6. **return** ResultHelper.generateTokenInvalidResult();
7. }
8. **if** (questionnaireEntity == **null** || !questionnaireEntity.isValid()) {
9. **return** ResultHelper.generateSimpleResult(**false**， "请求数据不完整");
10. }
11. getService().saveOrUpdateQuestionnaire(questionnaireEntity);
12. **return** ResultHelper.generateSimpleResult(**true**， "保存成功");
13. }



图 2.2 SpringMVC 架构

Hibernate的作用非常明确，就是简化应用对于数据库的管理。使用Hibernate只需要在xml文件中配置好数据库和java对象的映射，或者直接在Java代码中使用annotation进行ORM映射，即可方便的使用Hibernate的API对数据库的数据进行增删改查和更为复杂的操作。

Hibernate让开发者可以不用书写SQL语句，而是可以选择书写更为面向对象化的HQL或者直接使用Hibernate的链式调用API进行数据库的操作，大大提高了开发效率。

### 2.1.2 SSH框架在本课题中的应用

本项目横向将代码按照业务逻辑划分为四个大的模块：自定义首页模块，审批管理模块，公文处理模块，常用工具模块。按照代码的层次横向分为三层：Dao层，负责基础的数据库增删改查的封装；Service层，使用Dao层来实现业务逻辑，完成客户端的响应逻辑；Controller层，获取并分发用户的请求，调用Service层来完成用户响应，并返回用户响应结果。项目架构如下。

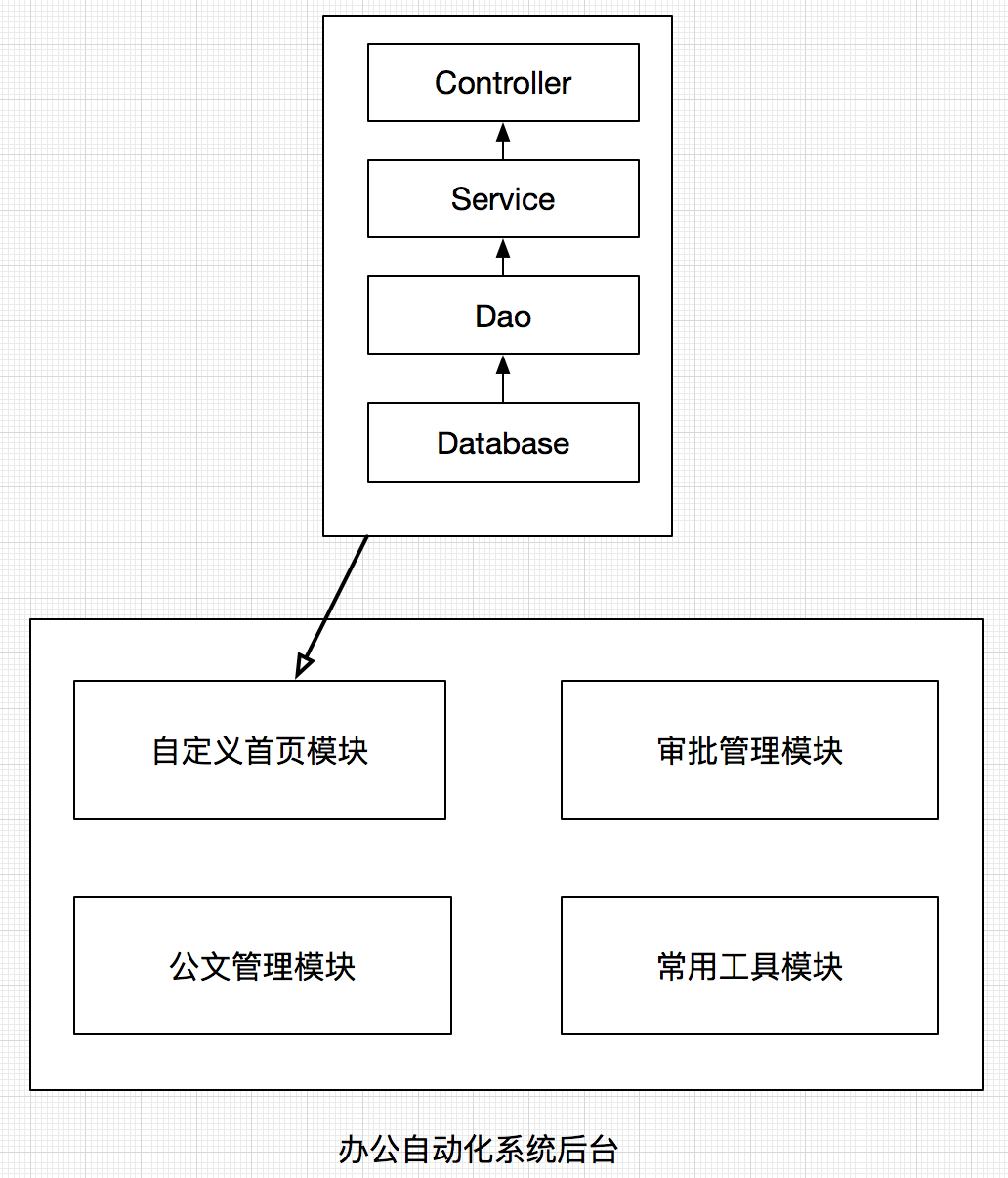


图 2.3 后端模块架构

通过配置spring.xml这个配置文件，Spring可以作为容器在web应用的生命周期内管理。在本项目中，Dao和Service层都是放在Spring的容器中进行管理的。配置文件的例子如下，通过bean标签声明bean，通过class指定bean的类型，spring会在初始化的时候通过找到对应的class并通过默认的构造方法创建一个bean，bean标签中可以声明子标签property，表示类中的一个field属性， 该属性需要有set方法以使得spring可以对其赋值。每一个bean还可以有一个唯一的id属性，可以供其他的bean标签对其进行引用。通过下面这么几行简单的配置文件，spring在初始化的时候就能够生成两个bean并将其纳入自己的容器中对其生命周期进行管理。

1. **<bean** id="questionnaireDao"
2. class="com.ssthouse.officeautomation.dao.impl.QuestionnaireDaoImpl"**>**
3. **<property** name="sessionFactory" ref="sessionFactory"**></property>**
4. **</bean>**
5. **<bean** id="questionnaireSerice"
6. class="com.ssthouse.officeautomation.service.impl.QuestionnaireServiceImpl"**>**
7. **<property** name="dao" ref="questionnaireDao"**></property>**
8. **</bean>**

当需要使用容器中的bean时，可以通过下面简单的几行代码进行获取，其中getServiceClass()是一个自定义的方法，只需传入想要获取的bean的class即可。用户同样可以传入bean的id来获取bean，只需在context.getBean()方法中传入一个代表id的字符串即可。

1. ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("spring.xml");
2. ServiceClass mService =  context.getBean(getServiceClass());

Spring管理bean的这一特性被称为控制反转。利用这一特性我们可以将需要只需要有限个数实例的类在spring中进行初始化，并在整个web程序的任何地方任何时间都可以轻易的获取。比如对数据库特定表进行操作的Dao类，处理特定业务逻辑的Service类，还有一些全局的工具类。

SpringMVC将客户端的URL请求进行了划分，如”baseURL/questionnaire/detail”和”baseURL/questionnaire/answer”将会被同一个Controller进行管理， 而Controller中又有不同的方法来处理具体的客户端请求，因此不同URL的处理逻辑分发将非常清晰。下面给出一个控制调查问卷的Controller的例子，出于篇幅考虑，其中方法的具体实现被去除。可以看到Controller中的方法有两个相同的标签：@CrossOrigin含义为支持跨域请求， @ResponseBody标签的含义是方法返回值直接作为http的response的body。其中支持跨域请求是因为本课题的前后端代码分离，所以可能会放在不同的服务器上，在尝试获取后台数据的时候是跨域的，如果不显示的声明跨域请求的支持会导致前端调用API时出错。只需在Controller类上声明url的基本匹配，凡是url被requestMapping的正则表达式匹配的客户端请求都会被该Controller中的方法进行处理。

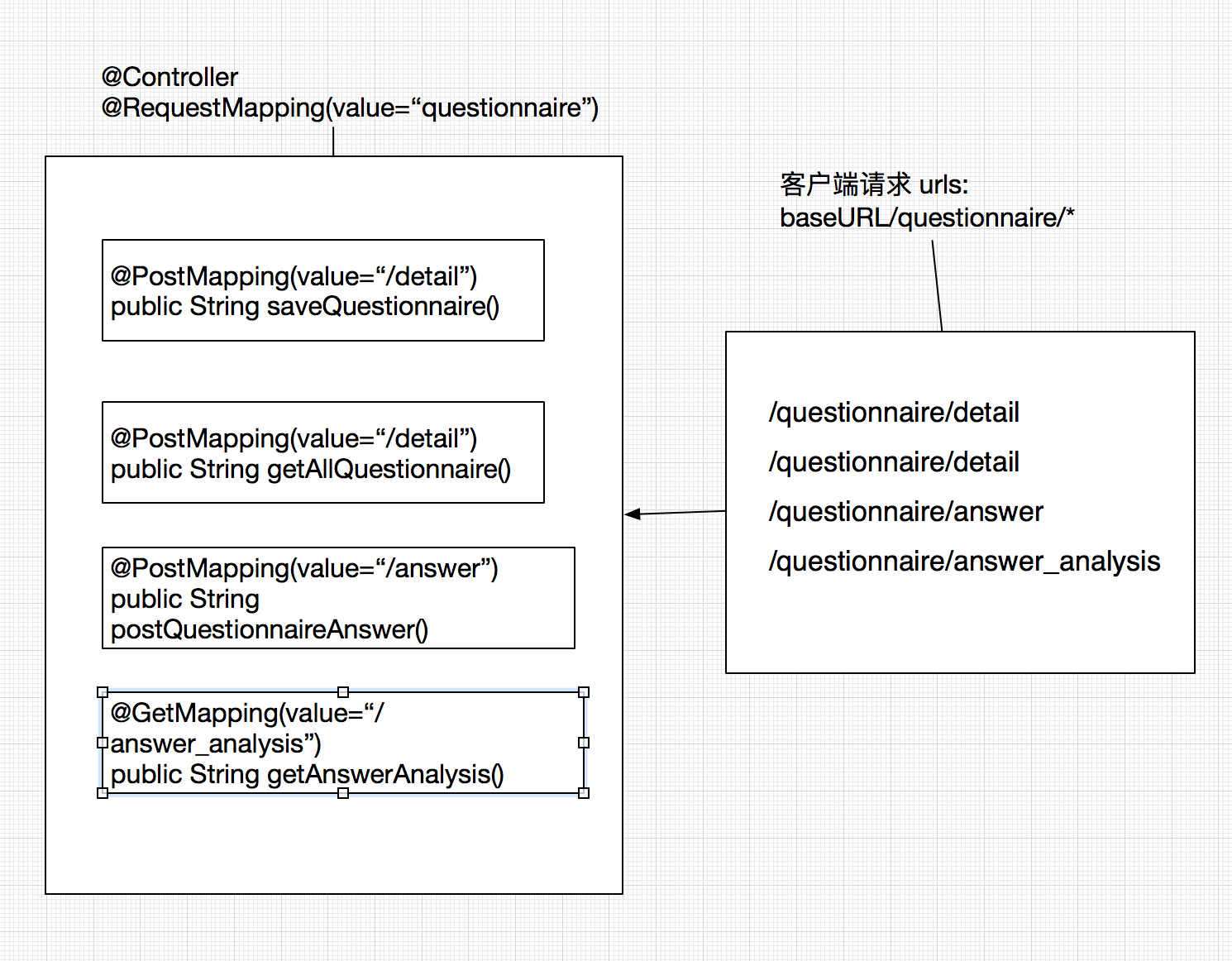


图 2.4 后端Controller示例

Hibernate的作用是简化后台应用对于数据库的管理。使用的方法有两种：一是在xml文件中配置好数据库和java对象的映射；二是直接在Java代码中使用annotation进行ORM映射，即可方便的使用Hibernate的API对数据库的数据进行增删改查和更为复杂的操作。

在以前版本的Hibernate中，数据库的ORM映射只能使用xml文件进行定义， 比较繁琐，如果bean较多的话，会产生很多xml文件。本课题选用的Hibernate5.0的版本，可以直接在Java数据类中进行数据库表的映射，通过简单的几个标注便可以将java的字段和数据库表中的字段对应起来。下面以问卷调查中的questionnaire表和Java中的QuestionnaireEntity进行映射为例，进行详细的说明。

在本课题的数据库中，问卷调查表questionnaire有这样几个字段：id，title，deadline，creater\_id，published。分别代表的是主键，标题，截止日期，创建者id，问卷是否公布。可以看到下图中左边的表格标识了这些字段在数据库中的数据类型，包括int表示数字，varchar表示字符串，tinyint表示布尔类型。通过Hibernate我们可以不用去管理这些复杂的数据类型转换问题，而只用在Java类型使用annotation声明式的将java数据类型和数据库数据类型匹配起来，然后通过Hibernate的API进行数据库操作即可。

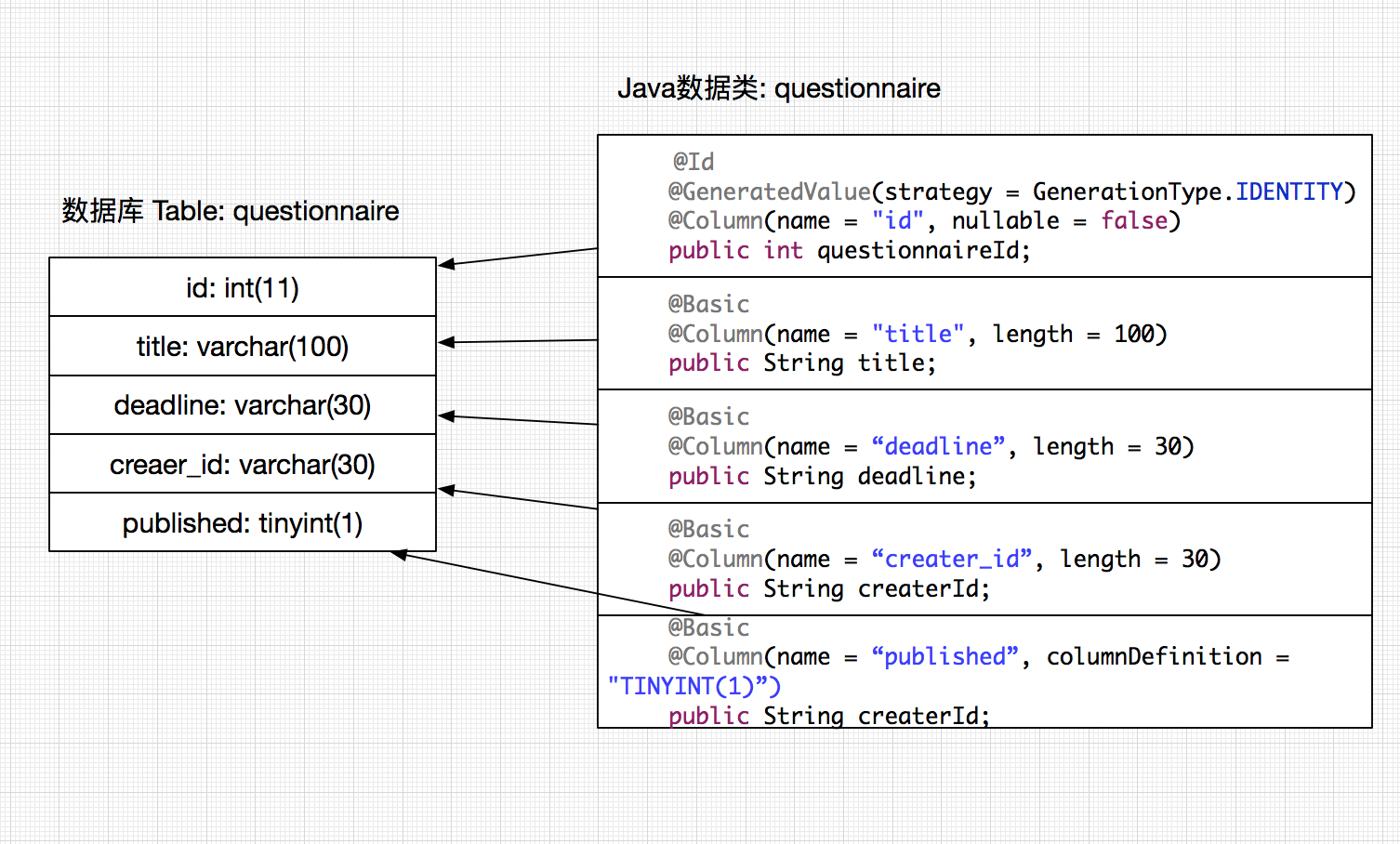


图 2.5 数据库映射

### 2.1.3 RESTful API的设计

关于RESTful API，全称为：Resource Representational State Transfer，意为资源表现层状态转移。其意义是提供了一个后台API设计规范，用简洁的话说就是URL定位资源，用HTTP动词（GET，POST，DELETE，UPDATE）描述操作。使用RESTful创建的API可以被各种不同的客户端所消费，且URL功能划分清晰明了，并解放了后端的开发任务，只需要提供功能明确的API而无需纠结于繁杂的界面编程。



图 2.6 REST API图解

同样RESTful对于前端的开发也有很大的帮助。以浏览器前端web页面为例，通过RESTful 的API，前端可以轻易的获取后台数据，操作后台数据，就像操作本地数据库一样方便。

下面列出本项目定义的一些URL，基本遵循了RESTful API设计的规范:

GET /questionnaire/detail 获取问卷详情

POST /questionnaire/detail 新建问卷数据

POST /questionnaire/answer 填写问卷数据

GET /questionnaire/answer\_analysis 获取问卷统计数据

解决了URL接口定义的问题，接下来就是前后端具体交互数据的定义。前后端常用的数据交互格式有JSON，XML。其中XML因为数据量较大且解析复杂已经慢慢背弃用，而JSON以简单的语法完成了数据描述的功能，现已成为前后端数据交互的主流格式，本课题也是选取的JSON作为数据传递的格式。

JSON的书写语法非常简单，几句话即可表述清楚：数据在名称、值对中，数据由逗号分隔，花括号保存对象，方括号保存数组。

下面是一个本课题中定义的JSON交互格式，其中“ok”字段表示该请求是否成功，当ok为true时表示后续的user数据不为空。“user”字段是一个用户数据对象，其中又包括“username”，“password”，“gender”等描述用户详细信息的字段,通过简单的解析这个JSON数据便可以处理用户信息的数据。

{

"ok": true,

"user": {

"id": 1,

"username": "ssthouse",

"password": "ssthouse",

"name": "沈顺天",

"gender": "male",

"phoneNumber": "18771037960",

"description": "android developer",

"department": "wap ShangHai",

"isAdmin": false

}

}

在Java中，我们使用了Controller进行客户端请求的管理，又使用了@ResponseBody使方法返回的字符串直接作为返回报文的body。本课题使用了第三方的JSON解释和生成库：Gson，这是Google公司出品的一个开源的Java端JSON处理库，运行效率较高。通过在Java端定义和定义好的JSON格式相同的Java数据对象， 再通过new Gson().toJson(new User())即可得到上面的JSON字符串数据。

在前端处理得到的JSON数据则更为方便，因为JavaScript中自带就有JSON数据的解析工具，而且JavaScript本身的对象也就和JSON数据格式极为类似，可以非常方便的处理得到的JSON数据。

### 2.1.4 Tomcat服务器和应用架构

本课题选用Tomcat作为Java后台应用的服务器，通过监听计算机的8080端口响应客户端的请求。在Java后台应用编写完成后进行打包，将打包好的jar放置在Tomcat服务器的根目录，然后启动Tomcat服务器，服务器会自动解压jar包并启动后台应用。

比较特殊的地方在于，本课题的前后端代码是完全分离的，而用户想要获取前端的页面也需要服务器进行支持。因为前端的Vue代码经过渲染后得到的就是普通的html，CSS和JavaScript脚本文件，所以我们的解决方案是将Vue前端编译成的代码直接放置在Tomcat服务器的根目录，服务器直接将其作为静态文件提供即可。客户端，前端页面和后台应用三者的关系如下图所示:

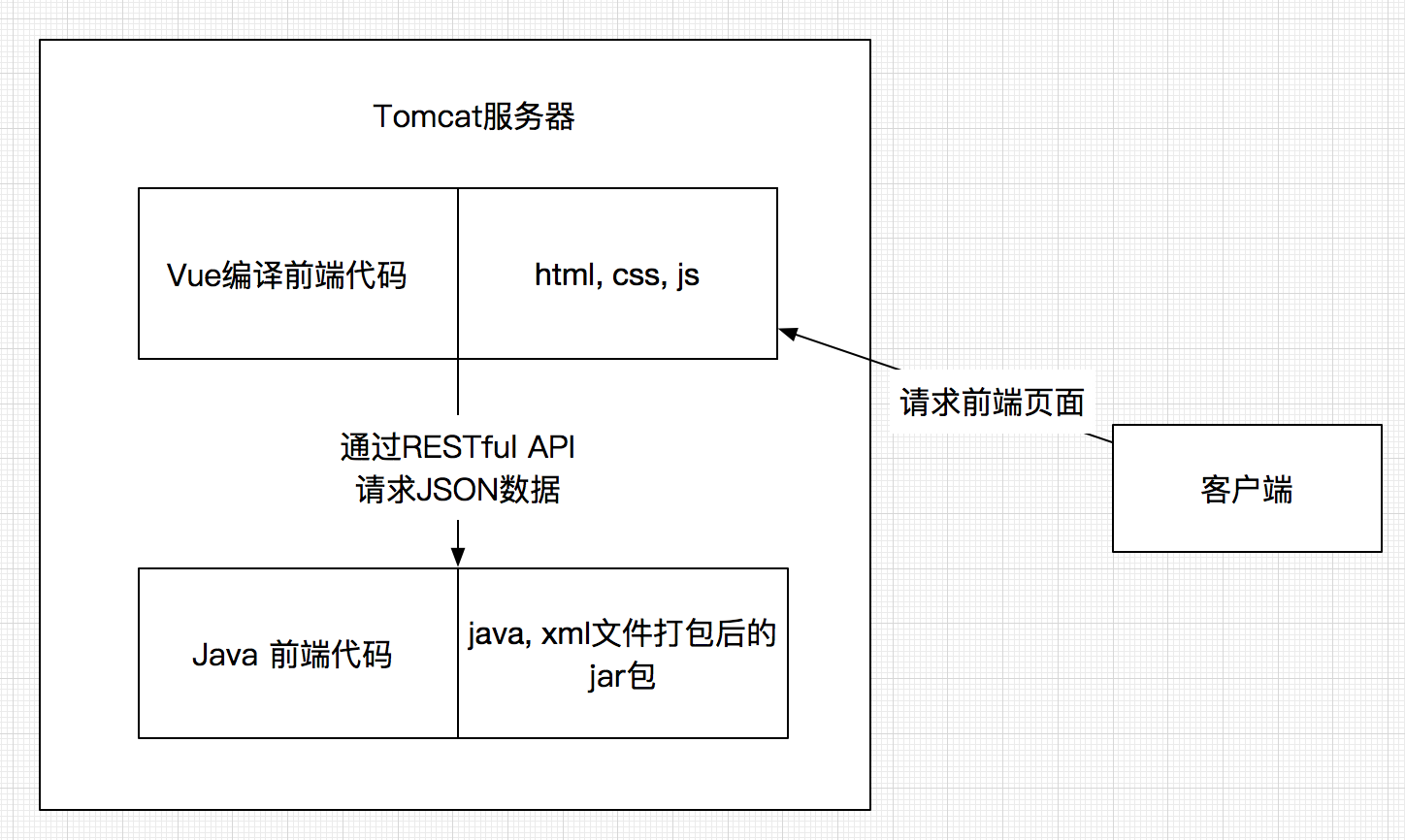


图 2.7 Tomcat架构

Vue前端渲染好的代码和Java后端打包好的代码都放置在Tomcat上，但是两者并不是直接在服务器端进行联系，而是通过网络进行联系。用户请求服务器拿到前端的页面，前端页面在浏览器中运行脚本，向Java后端进行数据请求。这样的架构好处在于前后端分工明确，可以完全分离进行同步开发，只需要定义好一套API结构和沟通的数据格式即可。而前端更是解放了思路，可以构建出用户交互体验更好的产品。

Java后端应用的架构完全遵循面向对象的开发理念。封装实现，面向接口编程，均按照规范进行编码。程序主要分为Dao，Service，Controller三层，每层都分为interface包和implement包，而每一层之间的调用都只对interface进行调用，完全隐藏了implement包中的实现细节。应用的整体架构如下:

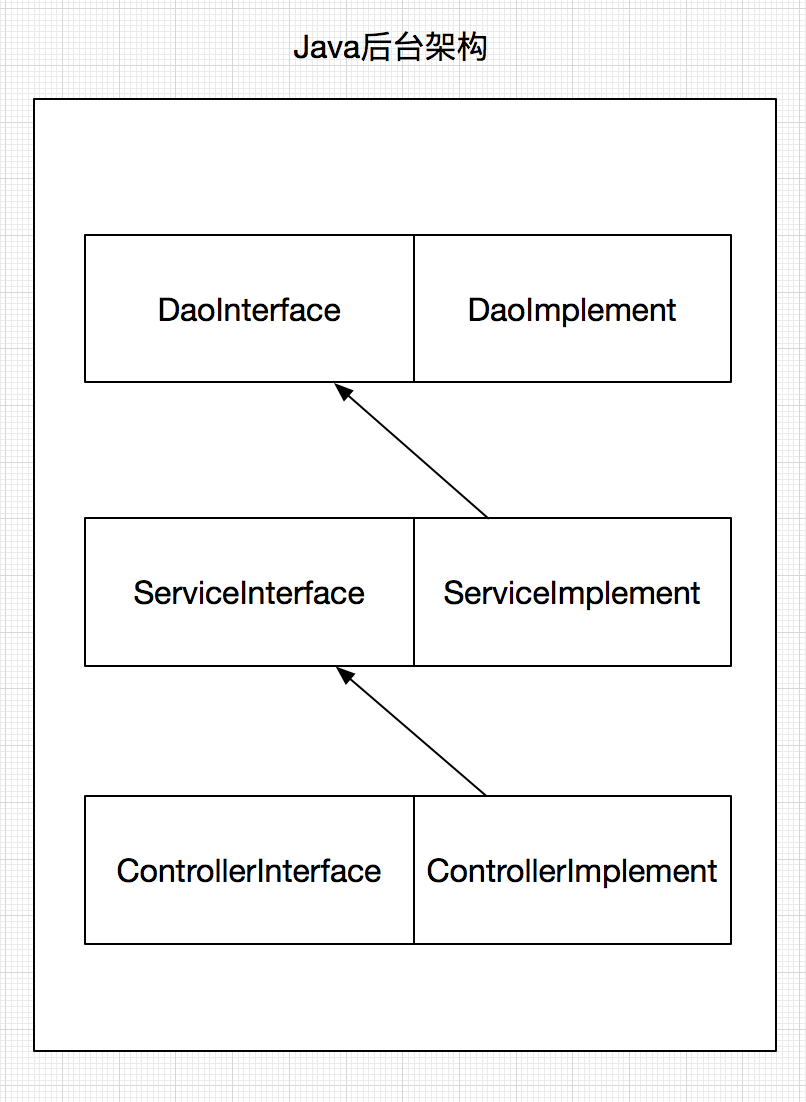


图 2.8 后端架构分层

而能够实现这样的架构离不开Spring容器的帮助。通过在Service层加入一个DaoInterface的字段，再通过Spring容器中拿到一个DaoImplement的实例。因为最终拿到的只是interface，所以能够直接通过interface的定义知道能够使用Dao层的哪些哪些服务而不会接触到这些服务的具体实现。

这种进行编码的好处在于，每一层之间是相互独立的。正如前后端直接通过定义API接口来相互独立进行开发一样，后端的每一层次之间也是通过定义interface来进行独立开发。在应用的API格式和interface接口定义好后，内部的实现发生什么变化并不会对层次间的调用造成影响。比如Dao层当前使用的是MySQL数据库进行的数据存储，当系统遇到需求变更需要修改为SqlServer时，Dao层的变化并不会影响到上层的代码，从而实现了系统灵活的扩展性。

## 2.2 基于Vue的前端SPA开发

### 2.2.1 Vue和传统前端开发的区别

传统的前端开发其实是和后端开发紧紧联系在一起的，比如使用Java Web后台开发，常用的模式便是使用JSP进行页面开发。前端程序员往往也需要了解JSP中一些Java语言的语法才能使用Java后台传递过来的数据。且因为书写的是Java和html，JavaScript混合在一起的代码，所以也不利于开发出交互更好的页面。更为很重要的是，前端的代码和后端联系在一起，导致前后端开发联系过于紧密，每当后端或前端代码需要修改时，往往需要前后端的程序员一起工作，极大的限制了前后端程序员的工作效率。

Vue作为一个前端框架，解决的主要问题是前端开发中的前端代码的模块化开发。通过编写.vue文件，前端人员可以定义出可重用的组件，在Vue中称为Component，Component通过注册到Vue，可以在整个项目中全局使用。而每个Component的编写都是一个独立的Vue文件，而不会互相干扰，最终的.vue代码经过Vue编译器的编译会生成普通的html，js，CSS文件。通过这种编写.vue文件再编译得到最终产品的策略，Vue实现了前端的模块化管理。

除了Vue，专注于解决前端模块化开发的框架还有React，Angular。其中React甚至衍生出了React Native这种更贴近原生开发的框架。而本课题之所以选择Vue作为前端框架，主要原因是因为Vue更加的简洁易用。Vue的作者表示Vue 的定位是用于构建用户界面的渐进式框架。通过响应式和组件化的开发方式，Vue大大降低了大型项目的开发上手难度。

除了快速上手，学习曲线平缓外，Vue的性能也强于React和Angular，并且相对于React和Angular那种较重的开发方式。一旦使用React或Angular就意味着必须使用其配套的许多框架，这样会使项目变的越来越重。而使用Vue，开发者可以自由的选择需要的组件和使用其他的第三方组件进行快速开发，而当项目需要转换框架时也可以相对自由的转换而不用受困于项目整体的框架。

### 2.2.2 办公自动化系统单页面应用实践

Single page application称作单页面应用，意思就是单一Web页面的应用。实现的方式是加载单个HTML 页面并在用户与应用程序交互时动态更新该页面的Web应用程序。所有的html，CSS和JavaScript脚本文件，都会在第一痴用户请求的时候就全部被加载。用户接下来的所有的操作都在这个web页面上完成，都由JavaScript来控制。不会出现跳转到新的页面或刷新当前页面的情况。因为所有的页面交互都在一个html中存在，单页应用来说模块化的开发和设计显得尤其重要。其实单页面应用也正是因为目前前端模块化架构渐渐成熟才慢慢被人们重视起来的。其实与SPA类似的一个概念是PWA技术。PWA技术的提出也就是在SPA的基础上，通过Google对于Android的掌控能力，加入了一些原生应用的API提出的一种前端应用。Progressive Web App是一个利用现代浏览器的能力加上Android的部分原生API来达到类似APP的用户体验的技术。让Web App和Native App之间的差距更小。

可以看出SPA的目的是为了让用户获得类似原生应用的体验，而又同时能够拥有前端页面无需安装随时热更新的优点。在一个Web浏览器窗口进行操作可以使用户注意力更集中，而不会被页面的跳转打断思绪，也不会因为页面的加载刷新而急躁。这也正是最初使用客户端开发技术的原因。

SPA的实现遇到两个主要的问题：一个是需要在单页面中和服务器进行交互而不能依靠浏览器的URL跳转；另一个问题是单页面会涉及到很多的用户交互UI，如何将这些UI元素整合到一个html文件中，也就是如何将单页面的项目模块化工程化。

对一个第一个问题，解决办法是前端和后台通过AJAX数据交互。ajax的全称是Asynchronous JavaScript + XML。之所以叫做异步，是因为用户服务器发送请求的时候，不必要停在那等待结果，而是可以同时做其他的事情。等到有了结果我们再来处理这个事。xml只是一种数据格式，并不是强制性的要求。现在大部分应用都使用JSON进行数据交互，JSON的好处在本文后台技术部分已有介绍，不再赘述。

在Vue中，对于AJAX的实现是vue-resource库，通过vue-resource可以方便的对后端数据进行交互。其实vue-resource只是对不同浏览器对于AJAX请求进行了封装，封装出了非常易用的一个API。如使用vue-resource对后端服务器问卷列表进行GET请求，调用方式为: this.$http.get(“baseURL/officeautomation/questionnaire/list”)，该方法会返回一个Promise。Promise是ES6中加入的一个非常实用的特性，通过调用Promise的then（）方法，可以传入两个回调方法，分别对应请求成功和请求失败的处理逻辑。在请求成功的回调方法中会传入一个参数response，代表的就是服务器成功返回的http包，response中的body参数也就是服务器返回的数据了。这样一个逻辑清晰的客户端服务器的交互就完成了。拿到数据后，便可以更新界面或执行其他的用户操作了。

解决了浏览器跳转，数据获取的部分。现在需要解决单页面交互逻辑的部分。在传统的URL跳转逻辑中，不同的URL对应了不同的界面，也就对应了不同的逻辑，那么在SPA中如何在一个固定的URL下体现出界面的逻辑变化呢？这一点其实很好解决，并且解决方案借鉴了原来在URL跳转中的逻辑，使得原有的项目修改为单页面应用也能减少页面逻辑的修改。通过在baseURL: http://localhost:8081/#/oa\_system/workflow后面加#号再加上当前页面的逻辑URL即可，仅仅在原有URL跳转的逻辑上加上了一个小小的改变，便在此基础上实现了SPA应用的页面逻辑跳转。至于在页面不刷新的情况下修改URL，Vue给出了一个完整的解决方案，这个解决方案通过Vue的封装为vue-router，下面以办公自动化系统中一个简单的逻辑跳转进行演示。

在初始化进入办公自动化首页时，默认的URL为: 域名/#/，通过vue-router的默认链接跳转设置可以使URL自动跳转至: 域名/#/oa\_system/homepage。

### 2.2.3用户界面的框架选择和实践

本课题使用的前端框架是Vue，Vue的特点在于组件式和响应式开发。为了充分利用这两个特点，提高代码的复用率，我们选用的UI组件框架也是基于Vue的，在对基于Vue的组件框架进行筛选尝试后我们选择了Element和Vue-material两个UI框架进行混合使用。

选择Element框架的原因是：该框架UI组件较全，常用的控件都有并且风格比较统一，基于Vue开发，方便数据绑定。且非常适合于较为复杂的桌面端的web应用开发，正如其官网所说的那样:

简化流程；

清晰明确；

帮助用户识别；

选择Vue-material的原因是其使用了material design。Material design称为材料设计，原本是Android5.0之后Google推荐的安卓应用开发的设计语言。但经过Google的进一步推广后慢慢成为了在网页端也非常流行一种设计语言， 其核心思想是强化用户和UI的交互反馈，通过阴影表示层次和距离，通过大量卡片布局表示控件，通过流畅的模拟现实的动画响应用户的操作。并且Material-design对于跨平台web应用有着非常好的支持。遵循这一开发准则可以使开发者开发的web应用轻松的适配移动端和桌面端的显示。本课题主要面向的是桌面web客户端，所以没有对移动端页面做专门的适配，但我认为这也是值得去做的一点， 因为现在很多公司对于移动办公的需求也是越来越高，所以对移动端的适配还是很有必要的，不过暂时出于工作量的考虑尚未进行适配工作。

作为单页面应用，如何进行划分决定了用户体验的好坏。本课题使用的Element中提供有NavMenu组件进行侧栏导航。可将导航栏固定在左侧，提高导航可见性，方便页面之间切换；顶部可放置常用工具，如搜索条、帮助按钮、通知按钮等。适用于中后台的管理型、工具型网站。本课题正是管理型的网站，所以选用侧栏导航作为应用的主模块导航。

通过点击左侧的菜单，动态的切换右边界面主题的Vue组件。 这当中使用到了Vue的<keep-alive>标签和<router-view>标签。<keep-alive>标签的作用是在不同的component进行切换的时候，并且保存component的状态，不至于丢失用户的操作到一半的数据。使用<router-view>标签的作用是为了结合vue-router， 在点击menu时操作当前URL的逻辑路由跳转。可以实现原始URL跳转的前进后退功能而不刷新浏览器页面。同时用户点击浏览器的前进后退标签也会触发vue-router的路由。

在每一个主要的模块的操作界面中，右侧的主界面有一个类似浏览器标签页的地方，如下图所示。 这是Element组件中的Tab标签，其作用是分隔内容上有关联但属于不同类别的数据集合。例如下面的公文处理模块，在首页tab中有许多操作按钮，如点击查看某一个公文的详细内容，将会跳转到一个新的tab页面。通过这样的方式将众多的用户操作组件组织起来，划分成逻辑清晰的tab模块。类似于许多邮箱web应用，大家会发现其实也是一个单页面web应用，通过动态的创建关闭tab标签， 实现在单一web界面操作各种业务逻辑，且不会丢失每部分操作逻辑的状态，方便用户回到这部分UI继续工作。其中第一个tab标签是主标签，包含了跳转到其他标签的操作按钮，是这个模块所有功能的集合页面，并且是不可关闭的。例如下图中公文处理模块的主tab，包含了用户发起的公文操作列表，用户待处理的公文操作列表，右下角有发起新公文流程的按钮。这个按钮在Material Design中被称为floating action button，简称问fab。其作用是给用户提供一个常用功能的快捷方式，一般为增加新元素的功能。

## 2.3 本章小结

本章详细介绍了该课题前后端实现的基本原理，项目架构的主要实现是前后端分离，进而前后端都可以选择最适合的框架进行独立开发。在确定了前后端分离并通过RESTful API进行沟通后，后端选择了使用SSH框架提供Json数据服务。通过面向接口编程，将后端架构划分为Dao，Service，Controller三层，互相之间松依赖，每层内部紧内聚。前端使用了Vue进行模块化架构，通过响应式和组件式的编程方式，前端代码架构逻辑非常清晰。加上Element和Vue-material 的UI组件，使用Vue官方提供的命令行脚手架vue-cli，很快就搭建起了前端的系统框架。且通过vue-resource，vue-router简化了前后端的网络请求交互，使得交互代码的编写更加优雅简洁。

在开发阶段，vue通过hot load技术使得浏览器页面实时渲染刷新，极大提高了前端开发效率，相比于传统的写好一段代码后，需要手动重启后台服务器，手动刷新浏览器页面，再查看效果的开发方式节省了大量时间和精力。在发布应用时，使用Vue提供的编译命令npm run build将编写的.vue文件打包成普通的html， CSS， html文件。放置到Tomcat服务器的根目录即可轻松的访问到前端页面，再通过浏览器加载JavaScript脚本实现和后端的数据交互。整个架构前后端完全分离，只存在json的数据交互。前后端各自的架构选择和代码变更都不会对系统整体架构产生影响，完全解放了前后端人员的生产力。

# 3 办公自动化系统数据表的设计

## 3.1个人首页模块

### 3.1.1用户个人信息表

用户个人信息表包含了使用该系统的所有用户的个人信息，包括常见的用户名密码等字段和一些更加信息的用户信息。

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| username | 用户登录名 |
| password | 用户登录密码，尚未做加密处理 |
| name | 用户姓名 |
| gender | 使用male和female表示男女 |
| phone\_number | 电话号码 |
| description | 用户个人简介 |
| department | 部门名称 |
| is\_admin | 时候具有管理员权限 |

### 3.1.2用户主页配置表

用户主页配置表用于存储用户自定义首页的数据，通过将自定义首页的卡片布局名称存储在home\_page\_config字段，通过逗号分隔来实现自定义首页功能。

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| username | 标识该配置是哪位用户的 |
| home\_page\_config | 用户的首页配置 |

### 3.1.3公告数据表

公告用于展示一些企业信息，一般有另外的链接进行公告展示，故只包含标题和链接字段。

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| title | 公告的标题 |
| link | 点击公告跳转的链接 |
| creater | 该公告的创建者username |

## 3.2审批管理模块

### 3.2.1事假数据表

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| username | 该请假流程的发起者用户名 |
| leave\_type | 请假类型，具体类型常见代码 |
| begin\_date | 请假起始日期 |
| end\_date | 请假结束流程 |
| day\_num | 请假天数 |
| description | 对于该请假的具体描述 |
| approver\_username | 审核人用户名 |
| state | 当前请假状态 |

### 3.2.2加班数据表

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| is\_holiday | 该加班是否为法定节假日假期 |
| beginDate | 加班起始日期 |
| endDate | 加班结束日期 |
| hourNum | 加班小时数目 |
| description | 加班详情描述 |
| username | 加班申请者用户名 |
| approverUsername | 加班审核者用户名 |
| state | 加班审核状态 |

### 3.2.3出差数据表

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| destination | 出差目的地 |
| beginDate | 出差起始日期 |
| endDate | 出差结束日期 |
| dayNum | 出差天数 |
| description | 出差详情描述 |
| username | 出差申请者用户名 |
| approver\_username | 出差审核者用户名 |
| state | 出差审核状态 |

## 3.3公文处理模块

### 3.3.1公文数据表

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| id | 主键 |
| owner | 公文发起者用户名 |
| state | 公文状态 |
| begin\_date | 公文起始日期 |
| deadline | 公文截止日期 |
| type | 公文类型，具体类型见代码 |
| title | 公文标题 |
| target\_organization | 主送单位 |
| copy\_to\_organization | 抄送单位 |
| content | 公文具体内容 |
| check\_comment | 核稿意见 |
| countersign | 会签意见 |
| sign | 签发意见 |
| executors | 公文下一步的执行人 |

## 3.4常用工具模块

### 3.4.1投票数据表

|  |  |
| --- | --- |
| 字段名称 | 字段含义 |
| questionnaire\_id | 主键 |
| title | 问卷标题 |
| deadline | 问卷截止日期 |
| creater\_id | 问卷创建者用户名 |
| published | 问卷是否已公布 |

### 3.4.1调查问卷数据表

### 3.4.1会议管理数据表

# 3 办公自动化系统主要模块的实现

该课题的办公自动化系统主要由四个模块组成，四个模块分别为：自定义主页模块，审批管理模块，公文处理模块，常用工具模块。还有部分全局使用的模块下面也会进行详细的介绍。下面的篇幅会对项目实现过程中遇到的一些问题进行详细的描述和讲解，有兴趣的读者可以查阅代码中的详细实现方法。

## 3.1全局模块描述

### 3.1.1 用户模块token验证

一个web应用最基础的功能便是用户管理，包括用户注册，登录，用户权限验证等等。本课题的前后端完全分离，发布时前后端代码甚至可能都不在一台服务器上，相比于传统的web页面直接由后台渲染生成多了一个用户验证的问题。因为不在同一台服务器，出于安全考虑需要一个验证机制，常见的验证机制有token验证和oauth验证。其中oauth验证较为复杂，本课题选择了使用token验证机制。

Token验证机制原理非常简单，客户端提交用户名和密码到服务器，服务器验证通过后会为用户生成一个具有时效性的token字符串，用户拿到该字符串后在之后每次和服务器交互时只要带上token，服务器便可验证用户是否有效。要实现token验证主要有两个问题：一是如何在服务器端生成一个具有验证身份作用的token字符串，二是如何在前端每次发出请求时都带上token字符串。

对于第一个问题，本项目采用了HS256的加密方式，加上系统当前时间作为salt实现的token生成机制。并且在token中编码进了用户名，因为用户名是每次请求基本都需要添加的数据字段，如果在token中带入的话可以节省后续请求中很多的用户名字段添加工作。每次前端向服务器发出请求时都会验证token的有效性， 验证token是否过期，如果token有效则可从token中解码出用户名信息。

对于第二个问题， 一种比较麻烦的解决方案是定义一个全局的js模块，在其中增加一个token字段用于保存token数据，然后在后续的每次网络中都获取该token并放入请求包的header中。但这样做会导致代码极其繁琐， 好在vue-resource除了封装了网络请求外还提供了许多实用的api，其中就包括全局定义的请求头字段。通过简单的vue-resource配置: Vue.http.headers.common['token'] = responseBody.token，vue-resource便会在每次向服务器端发出请求时自动带上token到请求包header中。Java后端可在HttpServletRequest参数中获取header的token字段即可获取到token。

### 3.1.1 vuex保存全局数据

单页面应用中有许多的数据是需要被全局使用的，比如用户个人信息。用户个人信息在初始登录的时候便获取到了，如果在后续每次需要使用的时候再进行数据请求不仅浪费了网络带宽，更是浪费了数据请求的时间容易造成用户的等待。

对于这个问题，vue给出了一个官方的并且非常实用的解决方案： vuex。

可以看出vuex是和vue紧密联系在一起的，通过在应用初始化时进行简单的一步加载，vuex即可在Vue的所有组件中随时被使用。下面将介绍vuex的加载和配置。

Vuex的加载非常方便，首先创建Vuex实例：new Vuex.Store()，在创建实例时需要传入vuex需要管理的状态变量和变量的getter，actions等方法，具体的api读者可以查阅vuex的官网阅读。在创建好vuex实例后，在整个Vue页面的初始化入口App.vue中，在初始化Vue变量时加入store字段，传入vuex实例。接下来便可以在全局的Vue组件中通过this.$store获取vuex实例。

Vuex的主要功能是管理数据，这些数据其实往往也代表着用户界面的一些状态，如下图所示，vuex将数据的操作和状态的变更结合起来，分为了三个部分：用户界面View通过获取vuex中的数据也就是状态显示界面，用户通过操作界面View发出更改vuex中数据的action，action再真正的变更vuex中数据进而触发用户界面的变更。这便是vuex管理数据和用户交互的基本逻辑。



图 3.1 Vuex架构图示

## 3.2 用户自定义首页的实现

### 3.2.1 动态显示首页的前提

首页定制是为了让不同岗位，不同等级的工作人员都能够定制出最适合自己的首页，为了让首页定义简便美观，我们在界面中使用了大量的material-design中的卡片布局。这些卡片布局可以由用户定义显示与否和显示顺序。卡片布局的样式如下所示:



图3.2卡片布局图示

这里使用的卡片布局使用到了前端中常用的栅格布局。栅格布局的原理很好理解， 就是将其父容器的横向空间分为24栏， 子组件可以设定自己的栏数。 从而当父容器大小发生变化时，子组件的大小也会相应的适配。也就是说当界面的宽度变化时，卡片的大小也会相应改变从而适应界面布局。卡片布局用到的标签有<el-row>和<el-col>标签。

而除了布局的自动适配外， 更重要的自定义布局的前提是可以让布局代码动态的生成。这里用到了Vue中提供的一个非常实用的标签属性v-for， 加入了该属性的标签将会按照v-for中的变量就行迭代生成。得益于vue中组件式的开发模式，所有组件都可以通过Vue.component(componentName， component)进行全局注册，并可以通过is属性设置component的名称来动态转化组件，这里恰好使用到了通过is属性动态改变标签的种类实现动态布局，其大体思路如下:

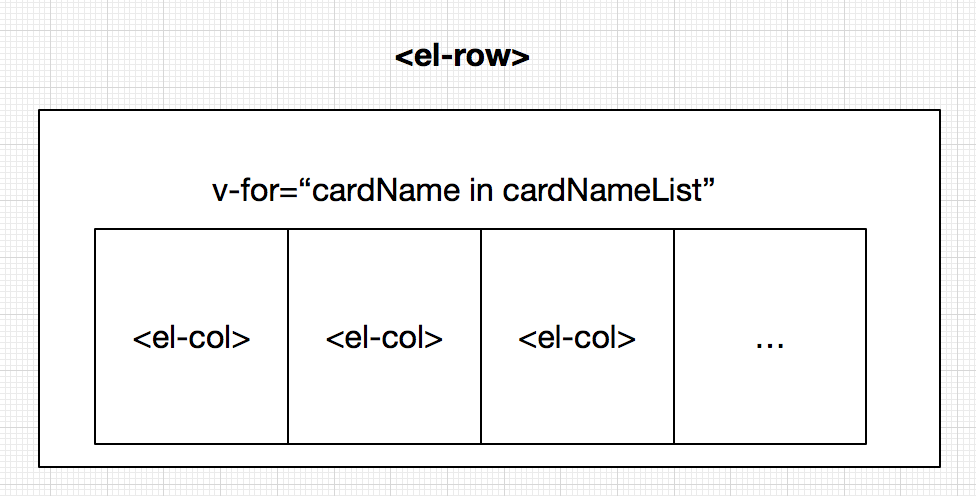


图 3.3自定义首页原理图示

### 3.2.2 动态首页配置的数据存储

既然要为每位用户存储个人首页的定制信息，必然要为每位用户在后台数据库中保存一份首页配置数据。这里我们在后台创建了一张user\_config表，在表中只有三个字段，一个是主键id，一个是username用于表示该配置文件属于哪位用户， homepage\_config字段存储用户的首页配置信息。

Homepage\_config字段数据类型就是普通的varchar，它那么是如何存储用户首页配置信息的呢？我们的做法是在该字段中存储一个字符串，字符串由逗号分隔为一个个小的字符串，每个小的字符串表示一个卡片布局的组件名称，而字符串之间的排列顺序也就是卡片布局。这样的解决方案确实比较简单，但是对于卡片布局的自定义也足够使用了。

在确定了数据结构后，接下来便是在前端实现用户的自定义配置功能。通过点击首页主tab上的配置按钮，会弹出一个自定义首页的配置dialog，界面如下图所示。 通过勾选需要的卡片，前端会将选择的卡片布局的组件名用字符逗号进行拼接， 使用cardNameList.join(‘，’)。将新的配置文件发送到后台进行存储并更新当前首页UI。



图 3.4 首页定制UI

## 3.3 审批管理模块实现

该模块分为两个卡片布局，一个显示用户发起的审批的列表，对于该列表可以进行的操作有详情，编辑和删除。点击查看详情将跳转到新的tab页面，展示审批请求的报表详情，点击编辑按钮可以编辑尚未发布，尚处于草稿阶段的审批，点击删除可以删除掉尚未发布的审批。另一个显示用户需要审批的列表，对于该列表用户可以进行的操作仅有查看详情，在详情页面，用户可以选择同意和不同意，用户审批的结果会及时同步到发出该审批的用户界面上。

****

图 3.5 审批详情示例

****

图 3.6发起审批图示



图 3.7审核审批页面

通过点击审批管理模块主tab上的红色floating action button，即可添加新的审批流程，在审批流程的详情填写页面可以选择请假类型，请假的日期区间，请假的事由，流程的审批人等信息。这里填写的审批人在该审批发布时即可在管理者卡片中看到该审批流程。

## 3.4 公文处理模块实现

公文处理模块教审批管理模块有许多相似的地方，包括都有审批者和发起者，都有业务状态的变迁，都需要多名用户之间的交互。但是不同之处在于公文处理模块逻辑更加复杂，包含的人员更多，转移的状态更多。在本课题中对于公文处理模块只进行了简化的实现，因为本课题的目标是做出一个实例而不是图大图全的实现办公自动化的所有功能。在审批管理模块中已经证明该架构和开发栈足以开发出交互，功能复杂的模块。

公文处理包括发文单和收文单两种，两种公文的填写格式如下图所示，主要区别在于公文处理中几个步骤不同，发文单特有的流程步骤为: 拟办，批办，办理；收文单特有的流程步骤有：核稿，会签，签发。

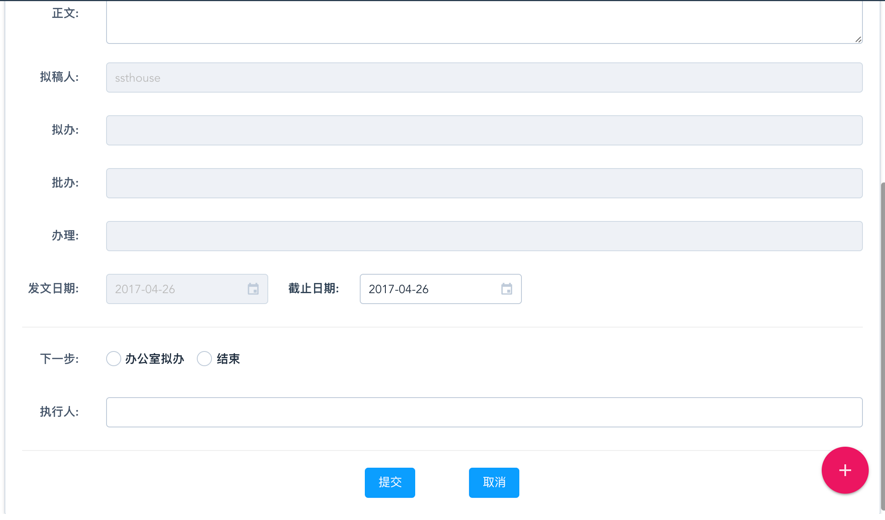


图 3.8发起发文单页面

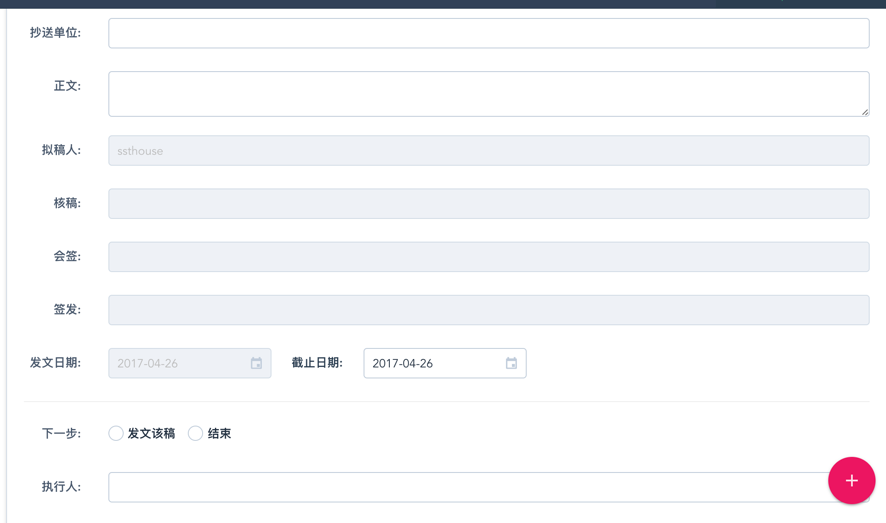


图 3.9发起收文单页面

用户对于用户自己发起的公文审批流程， 可以进行的操作有：查看，催办，作废和转办。查看的功能是查看可以打印的公文报表；催办是给当前公文流程的执行人发去短信或邮件进行提醒；作为是将用户自己发起的公文流程就此终止，不再继续执行，该公文涉及到的其他人员的公文列表中也将被废除；转办的功能是将当前状态的执行人变更，以加快公文流程的执行。具体操作界面如下。 其中短信提醒功能并没有实现，因为需要使用第三方的短信发送平台，需要较为复杂的申请流程才能使用，故暂时只可通过邮件提醒。



图 3.10公文处理主页面

## 3.5 常用工具模块实现

本课题的常用工具模块选取的是企业工作中常用几个工具。包括调查问卷工具，投票管理工具，会议管理工具。这些模块可能很多第三方的工具也能提供支持，但是将其集成在办公自动化系统中的好处在于：这些工具都可以轻易的和公司的人员信息关联起来，而这些第三方的工具很难获取到公司的人员信息。且集成在办公自动化系统中发布的问卷和投票以及会议都可以及时的提醒到相关的员工，提高了工具的使用效率。

除了基本的问卷和投票功能，该系统工具的一个特点在于提供了投票，问卷统计结果。这里使用的图表展示工具为chart.js开发库。这个开发库并不是基于Vue组件开发的库。使用这个库的主要原因在于Vue的生态系统中暂时没有特别完备的图表UI库。而这里使用图表只是进行简单的展示工作，并不需要动态的数据变化和交互，所以不需要使用到Vue的响应式的数据变更特性，所以这里使用一个纯js的UI图表库进行统计数据的展示。展示效果如下图所示。

三个工具中，调查问卷工具的实现较为复杂，因为涉及到用户动态设计问卷的界面，所以UI交互比较复杂，花费了较多的时间。最终实现出来的效果还是不错的。有希望做调查问卷功能模块的读者可以查阅这部分的实现代码。其主要思路也是使用v-for对所有用户添加的问卷题目卡片进行动态生成，同时使用is标签属性达到动态改变题目类型的效果。

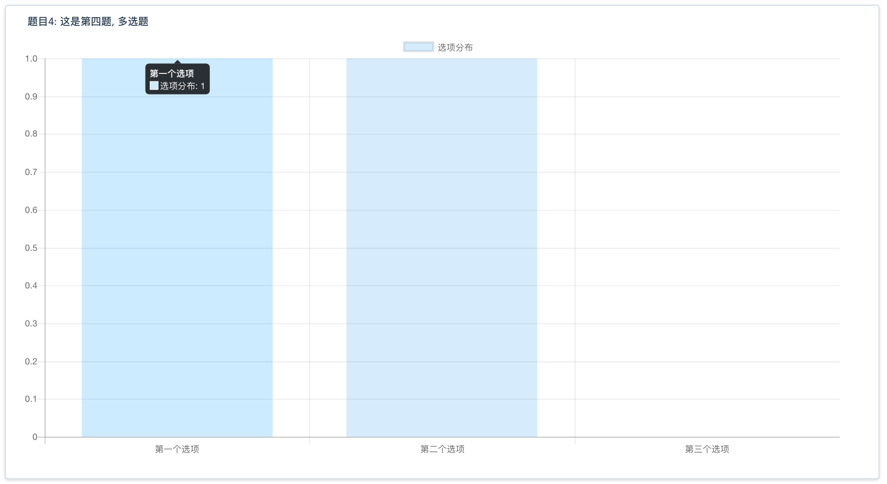


图 3.11问卷报表柱状图

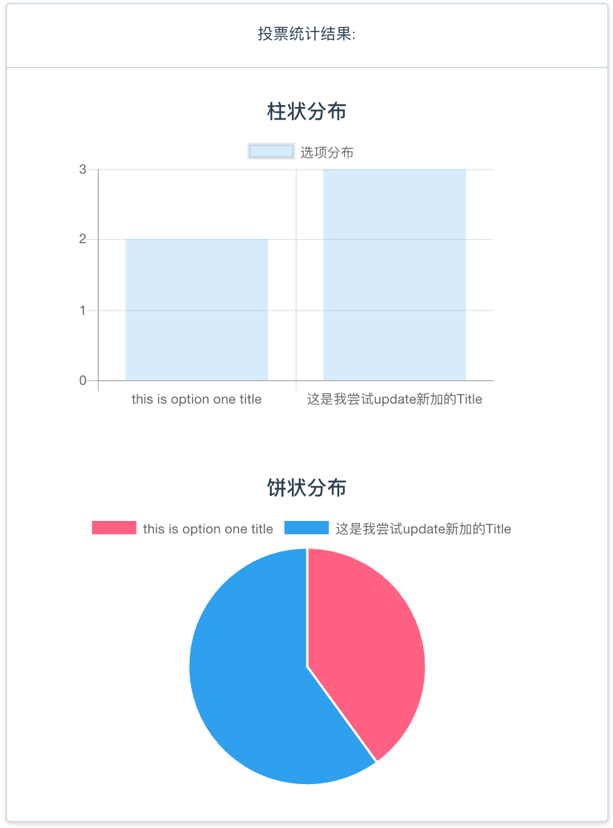


图 3.12投票统计报表

## 3.6 本章小结

本章主要站在开发办公自动化系统的业务逻辑的角度上，按照主要的模块，讲解了在开发过程中较为重要的技术点。因为没有贴上代码，只是用文字描述可能读者还没有一个直观的认识，所以结合示意图和最终的UI效果进行了讲解。希望了解更多具体实现细节的同学可以去github下载本项目的前后端代码运行查看，会有更加直观的了解。本项目的代码使用git管理，git branch逻辑清晰，每一次commit都尽可能的描述清楚做了哪些工作，读者不妨回溯到最初的版本一步步查看项目的构建过程。

# 4 总结

在系统功能和体验上，本课题大致完成了办公自动化系统常见的几个功能模块， 且用户体验良好，甚至接近于将浏览器页面全屏后接近于原生桌面应用的体验。 常用工具模块花费的时间较多，实用性也最强，是完成效果最好的模块。公文处理模块业务逻辑进行了简化，只能起到示例作用，实用性还不够。总的来说基于前后端分离，后端Java SSH后台加上前端Vue框架的架构实现的办公自动化系统体验远好于传统开发栈开发出的web应用。

在技术上，前后端完全分离，基于定义的RESTful API接口独立开发的模式对于前后端的开发效率都有了极大的提升。前后端可以自由的选择框架，选择最利于开发推进的工具而不用顾忌对另一端造成的影响，开发体验非常顺畅。特别是前端使用Vue进行架构后，基于响应式和组件化的开发模式让前端的模块划分非常清晰，有利于前端项目的维护和扩展。

当然，因为时间原因，而且开发者只有我一个人，不可能开发出完整的办公自动化系统。所以本课题只是选取了最重要，最常用的一些模块进行开发。并且对部分模块的业务逻辑也进行了简化，并没有达到实际使用的标准。但是本课题的目标本就是为了提供一个新的开发思路，通过使用新的技术栈，新的开发理念，证明传统的软件开发也可以通过使用这些技术获得更好的用户体验和开发体验。而课题最终的结果也证明了，这种开发模式获得的web应用体验和项目的开发体验都远好于传统的开发模式。

该课题的源代码链接如下，感兴趣的读者可以查看源代码更直观的了解系统的实现细节：

https://github.com/ssthouse/office\_automation\_backend\_eclipse

https://github.com/ssthouse/office\_automation\_frontend

本课题在Eclipse V4.6.2和WebStorm V2017.1开发,使用到的框架如表4-1所示:

表4-1 课题框架简述

|  |  |  |
| --- | --- | --- |
| 框架名称 | 版本号 | 框架描述 |
| Spring | 4.3.6 | 提供了面向切面和控制反转的容器框架 |
| SpringMVC | 4.3.6 | 提供了构建Web运用的MVC框架 |
| Hibernate | 5.2.3 | 提供了数据库处理的框架 |
| Vue.js | 2.1.10 | 前端的渐进式开发框架 |
| Element | 1.1.6 | 适用于桌面端的，基于Vue的UI框架 |
| Vue Material | 0.7.1 | 基于Vue，遵循Material Design的UI框架 |
| MySQL | 5.7.14 | 提供数据库支持 |

# 致谢

本课题在邓忠华老师和彭刚老师的指导下完成。邓忠华老师在选题之初就为我们给出了课题详细的介绍，包括框架的选择，办公自动化系统功能的要求和划分， 以及最终需要达到的效果。所以课题在开始的之初就目标非常明确。而在开发技术的选取方面，老师给予了我们非常大的自由度，所以根据我个人的开发经验选取了Java SSH后台和Vue前端这条最适合自己的开发栈。非常感谢老师给予的标准和自由度。同时非常感谢老师为我提供的几篇关于办公自动化系统的论文，通过阅读这些论文，我对办公自动化系统的业务逻辑和技术架构的发展有了更加细致的理解，也正是由此发现办公自动化系统的传统架构有待更新， 从而找到本课题的切入点。

同时感谢同在邓忠华老师和彭刚老师门下一同进行毕业设计的同学们，在每次会议时大家都积极的交流自己的学习心得和记录，互相帮助解决遇到的技术问题， 并且讨论交流更好的实现思路。这个互相学习讨论的过程给了我很多技术上实现的灵感。同学们互相分享自己找到的优秀的学习资料，源代码也大大加快了我们的课题进度。

这里还要特别感谢我的室友杨涛峰。他是一名前端开发者，在探讨本课题的前后端架构时，他提供给了我很多有用的建议，让我少走了很多弯路，也使得课题可以顺利的完成。正是在和他探讨前端模块化开发框架的过程中让我选定了Vue作为前端开发框架，最终也结果证明这一框架的实现效果非常好。

感谢学校为我们提供了自由检索万方，知网这些学术平台的条件，让我们能够轻松的获取到前人总结的经验，帮助我们更好的完成自己的课题。

最后，再次向我的指导老师邓忠华老师和对我提出过很多指导意见的彭刚老师、一起努力奋斗的同学们表示感谢，感谢参与毕业设计答辩和评审的各位老师。

# 参考文献

[1] 杨文清.办公自动化管理系统的研究与设计 网络安全技术与应用， 2014.

[2] 莫琳.办公自动化中计算机技术的应用探讨. 通讯世界， 2015.

[3] 王卫平.基于B/S的办公自动化系统的设计与实现[D]. 大连理工大学硕士学位论文 2009

[4] 车璐.办公自动化到移动终端适配方案的设计与实现. 北京电子工业出版社，2006.

[5] 贾童舒.办公自动化管理系统的研究与开发[D]. 天津大学硕士学位论文. 2012.

[6] Bao rong Zhong; Hong Du; Hua yun Yu. 2008. The Research and Implementation of Decision Support System for Oil Production Based on SOA. Second International Conference on Genetic and Evolutionary Computing.

[7] Yang Yu; Qiaoyan Wen. 2010. The design of the cloud mobile OA system. 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems.

[8] Cheng Wang; Zhixiao Lin; Fangsheng Guotang; Lixiao Zheng; Zhigang Zhu.2014. Design and Implementation of WPF-based Mini OA System. Ninth International Conference on P2P， Parallel， Grid, Cloud and Internet Computing

[9] Shen Zihao; Wang hui. 2010. Research on the application of workflow technology in OA system. International Conference on Computer Science and Information Technology.

[10] Wu Wanping.2009. Research and Implementation of Mobile OA System Based on 3G. International Forum on Computer Science-Technology and Applications