****

**本科毕业论文（设计）**

**（2019届）**

|  |  |
| --- | --- |
| **题 目** | **网络大数据的同步分流Java程序设计** |
| **学 院** | **计算机学院** |
| **专 业** | **计算机科学与技术** |
| **班 级** | **15052315** |
| **学 号** | **15051638** |
| **学生姓名** | **王永浩** |
| **指导教师** | **任彧** |
| **递交日期** | **2019年6月** |

**诚 信 承 诺**

我谨在此承诺：本人所写的毕业论文《校园学工信息化服务平台》均系本人独立完成，没有抄袭行为，凡涉及其他作者的观点和材料，均作了注释，若有不实，后果由本人承担。

承诺人（签名）：

年 月 日

摘要

本论文主要讲述了基于Java语言以及Spring Boot、Thymeleaf、druid及Bootstrap技术实现的网络大数据同步Web程序。本系统主要面向企业用户，能够实现数据备份的功能以及企业间数据同步的功能。本系统核心在于网络上的大数据的传输，其基于多线程文件分块下载模型，将下载的效率大大提高。本系统具备多数据源配置功能，能够满足用户的不同数据库要求，同时具备可视化数据同步状态管理，简化用户操作、便于用户理解与熟悉系统功能。

本论文将从系统的需求分析、技术原理、设计实现三个方面来阐述。系统总体架构思路是基于MVC模型。本论文将首先阐述项目课题的国内外背景调研以及相应的市场分析，调研系统的基本功能并进行需求分析与可行性分析。接着将介绍系统所使用到的Spring Boot、Java多线程、定时任务等相关技术原理。最后从系统的设计、具体实现以及系统测试等方面阐述该课题的系统。

通过本课题的研究与实现后发现，基于MVC模型的系统架构设计对后期系统的拓展与维护有很好的作用。在使用Bootstrap中也感觉到前端框架对开发者快速开发Web系统的便捷程度。Java多线程处理任务的机制大大提高系统利用率与提高系统响应效率。

关键词：Java Web程序；数据同步；多线程下载器;

ABSTRACT

This paper mainly describes the big data on the network synchronization Web program, based on Java language and Spring Boot, Thymeleaf, druid and Bootstrap technology. This system is mainly for enterprise users, and can realize the function of data backup and the function of data synchronization between enterprises. The core of the system lies in the transmission of big data on the network. Based on the multi-thread file block download model, the downloading efficiency is greatly improved. The system has multiple data source configuration functions, which can meet different database requirements of users, and has visual data synchronization state management, simplifying user operations, and facilitating user understanding and familiarity with system functions.

This paper will elaborate on three aspects of system requirements analysis, technical principle and design implementation. The overall architectural idea of ​​the system is based on the MVC model. This paper will firstly explain the domestic and international background research of the project and the corresponding market analysis, investigate the basic functions of the system and conduct demand analysis and feasibility analysis. Next, we will introduce the related technical principles of Spring Boot, Java multithreading, and timing tasks used by the system. Finally, the system of the subject is elaborated from the aspects of system design, implementation and system testing.

Through the research and implementation of this system, it is found that the system architecture design based on MVC model has a good effect on the expansion and maintenance of the later system. In the use of Bootstrap, I also feel the convenience of the front-end framework for developers to quickly develop Web systems. The mechanism of Java multi-threaded processing tasks greatly improves system utilization and improves system response efficiency.

**Key words**: Java Web Program; Data Synchronization; Multi-threaded downloader

**目录**

[摘要 3](#_Toc6149213)

[ABSTRACT 4](#_Toc6149214)

[1 绪论 1](#_Toc6149215)

[1.1 课题背景及意义 1](#_Toc6149216)

[1.2 课题在国内外的发展现状 1](#_Toc6149217)

[1.3 课题研究的主要内容 2](#_Toc6149218)

[1.4 本论文的组织结构 2](#_Toc6149219)

[2 系统需求 4](#_Toc6149220)

[2.1 系统需求概述 4](#_Toc6149221)

[2.2 系统需求采集 4](#_Toc6149222)

[2.3 系统需求分析 5](#_Toc6149223)

[3 网络大数据同步系统关键技术介绍 7](#_Toc6149224)

[3.1 Spring Boot技术介绍 7](#_Toc6149225)

[3.1.1 Spring Boot技术概述 7](#_Toc6149226)

[3.1.2 Spring Boot处理请求的流程 8](#_Toc6149227)

[3.1.3 Spring Boot配置 10](#_Toc6149228)

[3.1.4 Spring Boot打包、部署 12](#_Toc6149229)

[3.1.5 Spring Boot监控 12](#_Toc6149230)

[3.2 Druid技术介绍 13](#_Toc6149231)

[3.2.1 Druid监控 13](#_Toc6149232)

[3.2.2 Filter-Chain模式的扩展API 13](#_Toc6149233)

[3.3 Java并发包 13](#_Toc6149234)

[3.3.1 Java并发简介 13](#_Toc6149235)

[3.3.2 Java内存模型 14](#_Toc6149236)

[4 网络大数据同步系统设计 16](#_Toc6149237)

[4.1 系统总体设计 16](#_Toc6149238)

[4.2 主要业务流程设计 18](#_Toc6149239)

[4.3 部分API设计 20](#_Toc6149240)

[4.3.1 用户登陆 21](#_Toc6149241)

[4.3.2 修改某张表的同步状态 21](#_Toc6149242)

[4.3.3 修改数据源 21](#_Toc6149243)

[4.3.4 获取所有表的同步状态 21](#_Toc6149244)

[4.3.5 获取某张表的结构信息 21](#_Toc6149245)

[4.3.6 获取服务端所有同步状态的表 21](#_Toc6149246)

[4.3.7 客户端请求同步某张表 21](#_Toc6149247)

[4.3.8 服务端审批同步表请求 21](#_Toc6149248)

[4.3.9 客户端获取待下载文件列表 21](#_Toc6149249)

[4.3.10 客户端下载文件块 22](#_Toc6149250)

[4.3.11 通知服务端文件下载完成与入库状态 22](#_Toc6149251)

[4.3.12 客户端用户获取该用户的所有同步表 22](#_Toc6149252)

[5 网络大数据同步系统实现 23](#_Toc6149253)

[5.1 实现综述 23](#_Toc6149254)

[5.2 开发环境 24](#_Toc6149255)

[5.2.1 系统平台与编辑器 24](#_Toc6149256)

[5.2.2 浏览器开发环境 24](#_Toc6149257)

[5.2.3 服务端开发环境 24](#_Toc6149258)

[5.3 系统实现 25](#_Toc6149259)

[5.3.1 客户端的实现 25](#_Toc6149260)

[5.3.2 多线程下载器的实现 27](#_Toc6149261)

[5.3.3 Chrome浏览器测试 29](#_Toc6149262)

[5.4 服务端实现 29](#_Toc6149263)

[5.4.1 服务端的实现 29](#_Toc6149264)

[5.4.2 MVC层的定义 30](#_Toc6149265)

[5.4.3 服务端的测试 32](#_Toc6149266)

[5.5 主要界面实现 32](#_Toc6149267)

[5.5.1 服务端界面 32](#_Toc6149268)

[5.5.2 客户端界面 35](#_Toc6149269)

[6 平台测试 39](#_Toc6149270)

[6.1 单元测试 39](#_Toc6149271)

[6.2 接口测试 40](#_Toc6149272)

[6.3 集成测试 41](#_Toc6149273)

[6.4 测试结果及分析 42](#_Toc6149274)

[7 总结与展望 44](#_Toc6149275)

[致谢 45](#_Toc6149276)

[参考文献 46](#_Toc6149277)

# 绪论

## 课题背景及意义

随着互联网的发展，我们逐步进入数据时代，数据在笔者看来已经是生产力的要素之一，我们在互联网上的任何行为都将产生数据并被持久化存储到存储介质中。在数据时代，数据可以对一切事物进行描述、记录、分析与重组，数据可以刻画一个现实中的人物形象，比如基于大数据的人物画像分析，某求职网站可以根据应聘者提供的不同维度的能力数据刻画出用户的能力雷达图，进而能够实现相关职位的匹配与推荐，其次企业收集的用户行为信息也能帮助企业进行决策的制定与相关产品的个性化推荐等等。当今社会的每一个个体都会和一大串数据挂钩，在互联网上我们通过这些数据就能很方便的得到我们感兴趣的东西或者资料。数据时代给社会带来了便捷与高效率的运转。

在数据时代，数据的备份、传输、共享与同步就显得尤为重要。数据同步服务不同于数据挖掘那样可以直接利用数据产生价值，但它就如同是连接在线系统和离线系统的快速通道，让数据的价值在互联网的每个角落都活着、都存在价值。好的同步工具可以很大程度上提升数据开发的效率、数据同步也能实现数据的二次生产价值。

## 课题在国内外的发展现状

根据市场研究公司Technavio相关报告显示，中国的政府投资将引领中国的数据中心市场在2020年左右达到将近15%的年均复合增长率（CAGR）。据相关数据报告显示，在2014年全球数据总量超过6ZB(万亿GB)，2015年全球数据总量将触及9ZB。当前全球产生的数据的增长速度大约为40%，预测在2020年全球的数据总量可能要达到40ZB。在2015年中国大数据市场产生的数据价值规模接近116亿元人民币，并且在2016年超过了160亿元人民币。据相关预测，预计到2021年中国的大数据市场价值规模将接近900亿元人民币。大数据市场的数据的稳健增长在带来市场规模的同时，也将带动数据同步服务等相关配套产业的升级与发展。

目前关于数据同步系统的市场处于竞争激烈的状态。放眼全球，主要市场集中在互联网巨头手中，国外提供数据同步服务的主要是云服务商，如云市场巨头亚马逊（AWS），还有微软云以及谷歌云；国内提供数据同步服务的除了同花顺数据开发有限公司、云服务商阿里云以外，还有新型创业公司，如北京的datapipeline等。可以看到，目前市场上对数据同步服务的潜在需求很大，市场上也有许多此类产品，竞争较为激烈。当前数据同步服务的产业前景也属于上升期，未来对数据同步的需求就会持续增长。

目前已经有许多比较成熟的面相企业的数据同步服务平台，不同平台采取了不同的模式，解决了企业协同不同的痛点，但传统数据融合方案也面临着不同的问题：

1. 依赖个体技术能力：

平台自身适用范围较小，实际生产环境需求对个人技术能力依赖性强。这里的个人也泛指企业之间，大量数据在不同企业之间实现同步，数据的提供商技术能力可能会很强，数据的同步方企业的技术能力可能较为薄弱，这样就会存在技术能力偏差，影响整体项目工程的快速实现与投入使用。此时就需要一个能提供端到端、服务端与客户端整体化的可实时接入的数据同步服务平台。

1. 海量数据支持弱，开发维护量大：

对海量数据的支持能力弱，行业信息系统众多，数据要求多样，现有数据集成架构缺少对海量数据的高并发支持能力，从平台开发到日常的管理运营，依然存在大量的脚本开发工作。针对复杂的企业数据，提供多样化可变数据源接入，多线程同步的高效率工作模式，以及断点重传的容错保障机制实现对海量数据的高并发支持能够解决当前的主要困境。

## 课题研究的主要内容

本课题实现的系统的主要内容是基于Java和Spring Boot等技术的网络大数据同步功能的设计与实现。内容主要包括以下四个部分：

研究并比较Spring Boot及其他Web后台开发框架的优劣势比较。

研究并使用Spring Boot开发Web程序，并且结合Thymeleaf模板引擎实现前端后台的联动响应。

研究并使用druid，使系统具备热切换数据源的功能，支持不同类型的数据库接入课题研究的系统，以便提高系统的兼容性，目前支持MySQL，SQL server以及Oracle三种市面上常用的数据类型。

研究Java多线程编码技术，了解其并发包相关机制，并结合课题系统的设计需求，设计并实现多线程下载器。

## 本论文的组织结构

第一部分：主要阐述该论文课题的背景以及其所具备的实在意义、国内外市场发展情况以及国内外相关企业在该领域发展现状、论文课题的主要研究内容以及论文的组织结构。

第二部分：根据调研进行课题系统的需求分析，调研系统需求并进行系统分析、可行性分析以及技术选型。

第三部分：课题系统采用的主要技术的介绍，如Spring Boot的原理介绍，Spring Boot是如何提高开发效率的，Spring Boot处理请求的流程等。

第四部分：阐述课题系统的设计，包括系统的架构设计、主要功能模型的设计以及相关接口的设计。

第五部分：阐述课题系统的具体实现，将讲述课题系统主要相关功能模块的实现以及相关页面设计及相关的业务数据流图。

第六部分：阐述课题系统的测试情况，包括课题系统的主要功能测试和接口测试及单元测试。

第七部分：将记录对本次毕业课题的总结，课题系统的完善计划以及个人成长的感悟总结。

# 系统需求

## 系统需求概述

在互联网时代，数据的爆发增长带来了数据开发的红利，数据在企业间的商务合作中也越来越频繁。根据数据同步市场上的产品功能特点，结合企业间商务合作抽象出便于企业间数据同步的业务流。系统需要具备基础的用户功能，数据出库与入库功能，数据的网络传输功能，数据表的同步配置功能，用户同步配置功能。此外，该课题系统拟添加数据样例展示功能，数据同步日志展示功能，数据源的可视化热切换功能，请求同步数据与审批功能以及token验证用户权限。

## 系统需求采集

该课题系统主要面向企业用户，数据流符合商业买卖规范，具备请求同步审批。课题系统的主要基础功能的采集来自市场现有的数据同步产品。在系统需求概述中描述课题系统的特性功能是调研在数据市场中企业用户最为关注的功能点。本课题系统分为数据源提供者的服务端以及数据同步方的客户端。具体将分开阐述两端的需求。

数据源提供者的服务端：

1. 登录与注册功能
2. 用户可以基于系统配置文件配置的用户名和密码登录到系统
3. 在启动后，超级管理员用户可以通过注册添加新的管理员用户
4. 用户登录系统后可以修改密码
5. 数据源配置功能
6. 具备基于配置文件的数据源配置功能
7. 管理员登录系统后能够看到当前系统所使用的数据源信息
8. 管理员可以更改数据源的信息，提交更改后能够热切换数据源
9. 同步表的配置
10. 系统运行后，能够提供界面化的同步表状态展示
11. 可以单独修改数据表是否允许同步以及同步的用户有哪些
12. 鼠标停留在同步表的同步用户上时，能够展示该用户的同步状态
13. 客户端同步请求审批
14. 展示所有同步请求
15. 管理员可以对单个用户的同步请求进行审批，即允许同步、禁止同步、未同步
16. 同步请求审批设置完后能够回显到当前页面
17. 系统状态展示主页
18. 用户登陆系统后，展示当前库下所有表数，同步表数，同步用户数，总计同步成功次数
19. 根据总计同步成功次数是否为零，展示最新的前5条同步日志。

数据同步方的客户端：

1. 登录与注册功能
2. 用户只能将账号和密码配置到系统配置文件中，系统启动后将会有个线程去服务端注册配置的账户，注册不成功将尝试登陆
3. 用户登录系统后可以修改密码
4. 数据源配置功能
5. 具备基于配置文件的数据源配置功能
6. 用户登录系统后能够看到当前系统所使用的数据源信息
7. 用户可以更改数据源的信息，提交更改后能够热切换数据源
8. 同步表
9. 系统运行后，能够界面化的展示服务端同步以及相关表的前五行样例数据
10. 用户可以对服务端的一张表进行同步请求操作，页面回显请求状态，等待服务端管理员审批同步请求
11. 系统状态展示主页
12. 用户登陆系统后，展示服务端的所有同步表数，当前用户的同步任务数，待审批同步请求数以及当前用户的成功同步次数
13. 根据同步成功次数是否为零，展示最新的前5条同步日志。

## 系统需求分析

根据上述的需求采集，本系统主要涉及到Web端的数据同步。可以分为服务端与客户端。

系统的主要用户群是企业用户，主要业务逻辑涉及到数据源修改，用户相关功能，数据表配置功能，请求同步与审批功能。针对上述采集的信息可以将课题系统分别以服务端与客户端抽象出各自的功能模块。

服务端：

1）登录模块：

服务端账号依据系统中配置的相关账户信息为准，超级管理员基此登录到系统。系统也提供注册页面，系统启动后用户也可以注册新的账户。

2）数据源配置模块：

两种方式，基于配置文件在系统启动后，数据源便已经建立；另外一种是可视化的界面配置，服务端的用户登陆到系统后，校验用户携带的token是否合法，如果合法将可以进行数据源的配置。

3）同步表配置模块：

管理员登陆后能够对当前数据源下的所有表进行同步配置，包括表状态。

4）请求审批模块：

管理员登录系统后可以查看用户对当前数据源的同步请求。并进行审批。

5) 主页展示模块

展示系统的状态。

客户端端：

1）登录模块：

客户端的账号信息是基于系统配置文件中配置的信息，系统启动后请求服务端，获取合法token以便于后期和服务端交流。

2）数据源配置模块：

两种方式，基于配置文件在系统启动后，数据源便已经建立；另外一种是可视化的界面配置，服务端的用户登陆到系统后，校验用户携带的token是否合法，如果合法将可以进行数据源的配置。

3）同步表查看模块：

用户登陆后能够查看所有的服务端同步表，以及相关表的前五条样例数据。

4) 主页展示模块

展示系统的状态。

1. 网络大数据同步系统关键技术介绍
   1. Spring Boot技术介绍

Spring Boot是由Pivotal团队基于开源项目Spring的另一个开源框架，其主要目的是用来简化在用Spring搭建新应用时的开发搭建过程。开源框架Spring Boot使用了特定的方式——约定大于配置，来进行管理配置项目，进而使开发人员不再需要自行定义各种各样的项目配置。在笔者的理解中，Spring Boot 其实不是新的框架，它采用默认的配置方式让用户快速的使用该框架进行生产，就如同开发中整合了Maven 来管理所有的Jar 包，在这方面解放开发者，让开发者更加专注于业务流的开发中。其次Spring Boot 整合了所有的流行开源框架，特别是对Spring开源框架家族中的其他框架的完美融合，以及在微服务领域的大面积应用，使得Spring Boot越来越流行。Spring Boot在2013年开始研发，2014年发出第一个版本，但是直到2018年才开始在中国流行。再此期间有很多开发者尝试研究并使用Spring Boot 并且在网上写下了大量相关文章，与此同时逐渐地有些公司在企业内部进行了尝试使用，并且也将其使用经验分享了出来。之后使用Spring Boot的开发者也就越来越多，可以从 Spring Boot 关键字在百度指数中可以看出。如图3.1-1所示：

图3.1-1 Spring Boot搜索指数

图为2013年到2019年的百度搜索指数。可以看出2018年是Spring Boot在中国最火的时候。

### Spring Boot技术概述

Spring Boot的流行得益于其敏捷的快速开发功能。随着Spring Boot的逐渐流行，开发者越来越多，其将会更加完善。目前Spring Boot具备如下特点：

1)通过Spring官方的引导页面可以在快速构建一个Web项目

2)提供多种对外服务的数据接口形式，如 REST API、WebSocket、Web、Streaming、Tasks

3)简单方便的安全策略，无缝集成Spring Security

4)不仅支持关系数据库也支持非关系数据库，如NoSql

5)其具有内嵌Java Web容器，如 Tomcat、Jetty，方便开发者开发调试

6)自动配置：针对很多Spring应用程序常见的应用功能，Spring Boot能自动提供相关配置

7)通过添加dev-tool依赖支持热启动

8)带有系统状态监控功能，只需要添加Actuator依赖即可

9)支持各种 IDE，如 IntelliJ IDEA 、Eclipse

其核心技术理念有两个，即AOP与IOC。

AOP即Java中的面向切面编程，其设计思想是将程序处理流程相同的与业务代码无关的代码逻辑抽离出来，利用动态代理或则静态代理技术将共用代码适当的织入系统中，这样系统的耦合度就会降低，便于后期对系统的维护以及升级；IOC即Java中的控制反转，其设计思想是解放开发者，让程序依靠Spring容器实现对象内存的申请释放与管理以及让Spring容器去管理项目中对象引用关系。这样极大的提高了对象内存的管理，面向接口编程便于团队间的协作与后期系统功能的拓展。

### Spring Boot处理请求的流程

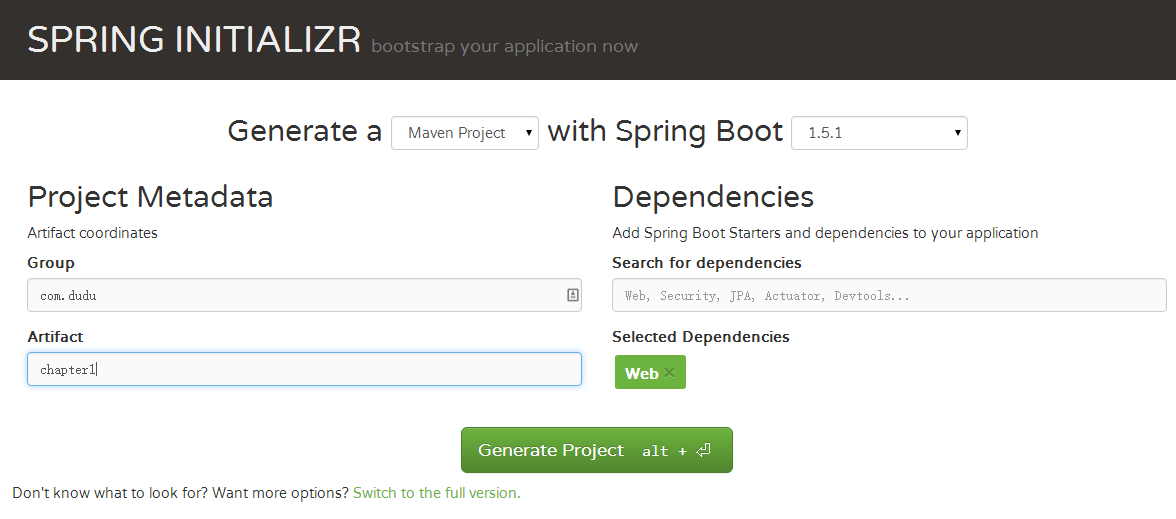
构建Spring Boot 项目只需要去Spring官方引导页面如图3.1.2-1所示：

图3.1.2-1 Spring Boot新建项目引导

生成一个Project，并下载下来解压导入到开发IDE中即可。

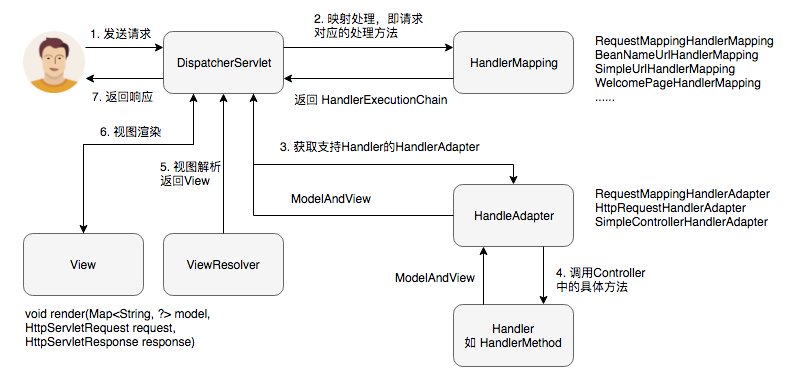
Spring Boot构建于Spring之上，其解析处理请求逻辑和Spring MVC中的逻辑是一样的。处理解析请求主要涉及两个核心点：其一是处理器映射，即系统选择哪个控制器来处理该请求。在控制器内将实现请求参数数据解析与返回参数数据解析；其二是视图解析器，即选择哪个视图解析器来渲染控制器传来的返回参数数据。下面是框架处理请求的主要逻辑图，如图3.1.2-2所示：

图3.1.2-2 Spring处理请求流程

其主要处理流程分为七个步骤：

1. 首先网络用户发送请求request，分发请求处理器DispatcherServlet作为前端负载均衡实现了Servlet接口，根据模板设计模式，整个请求处理流程为：HttpServlet.service -> FrameworkServlet.doGet -> FrameworkServlet.processRequest -> DispatcherServlet.doService -> DispatcherServlet.doDispatch。这其中的 doDispatch(HttpServletRequest request, HttpServletResponse response)方法即为整个请求处理的核心处理流程。
2. 获取请求的参数url，根据请求处理映射器选择和请求参数相对应的处理方法，即遍历handlerMappings列表，获取主要执行链对象HandlerExecutionChain(其包含一个处理器对象 handler 如HandlerMethod 对象、多个 HandlerInterceptor 拦截器对象)。这时的handlerMappings列表里面的所有handler为上下文中所有HandlerMapping接口的实现类，主要有如下四个（RequestMappingHandlerMapping、BeanNameUrlHandlerMapping、SimpleUrlHandlerMapping、WelcomePageHandlerMapping），遍历handlerMappings列表，根据请求参数url针对每个handlerMapping试图获取HandlerExecutionChain，如果获取成功（即不为null），则立即返回。
3. 获取请求对应的请求处理适配器 HandlerAdapter，HandlerAdapter 将会把2中的handler包装成一个适配器，进而能够达到支持多种类型的处理器，即这就是Java设计模式种的适配器设计模式在Spring中的应用之一，进而很能够很方便的支持很多种类型的处理器。DispatcherServlet中的HandlerAdapter列表如下3种（RequestMappingHandlerAdapter、HttpRequestHandlerAdapter、SimpleControllerHandlerAdapter），依次遍历，调用HanderAdapter.supports判断是否支持。
4. 数据流到Contrller层后，调用Controller的具体方法处理请求，并经过业务处理后返回一个模型视图对象 ModelAndView。HandlerAdapter会为每一个请求生成一个ServletInvocableHandlerMethod实例，该实例的核心方法是invokeAndHandle，其处理逻辑包含输入参数的模板处理和返回数据的模板解析。
5. 到视图解析层，系统会遍历DispatcherServlet的视图解析器ViewResolver的列表，获取与ModelAndView对应的View对象，最后通过将其传入入口方法DispatcherServlet.processDispatchResult中完成。
6. 渲染，调用5中获取的视图对象View的render方法，完成对Model数据的渲染。此处的 Model 实际是一个 Map 数据结构。
7. 最后DispatcherServlet 将第6步中渲染后的数据返回响应给用户，到此一个请求的流程就结束。最后浏览器将最终数据解析给用户。

### Spring Boot配置

Spring Boot的核心思想之一就是“约定大于配置（convention over configuration）”。所谓的约定大于配置也称作按约定编程，它是一种软件设计范式，旨在减少软件开发人员需做决定的数量，获得更简单的好处，而又不失其灵活性。也就是说，系统开发人员只需要在系统编码时规定框架应用中不符约定的部分或则系统需要深度定制的功能部分。比如，如果模型中有个名为 Person 的类，那么与之对应的数据库中对应的表就会默认命名为 person。当偏离这一约定时，也可理解为当前约定不符合设计时，将该类对应的表命名改为“person\_info”时，才需要写有关该约定的个性化配置。

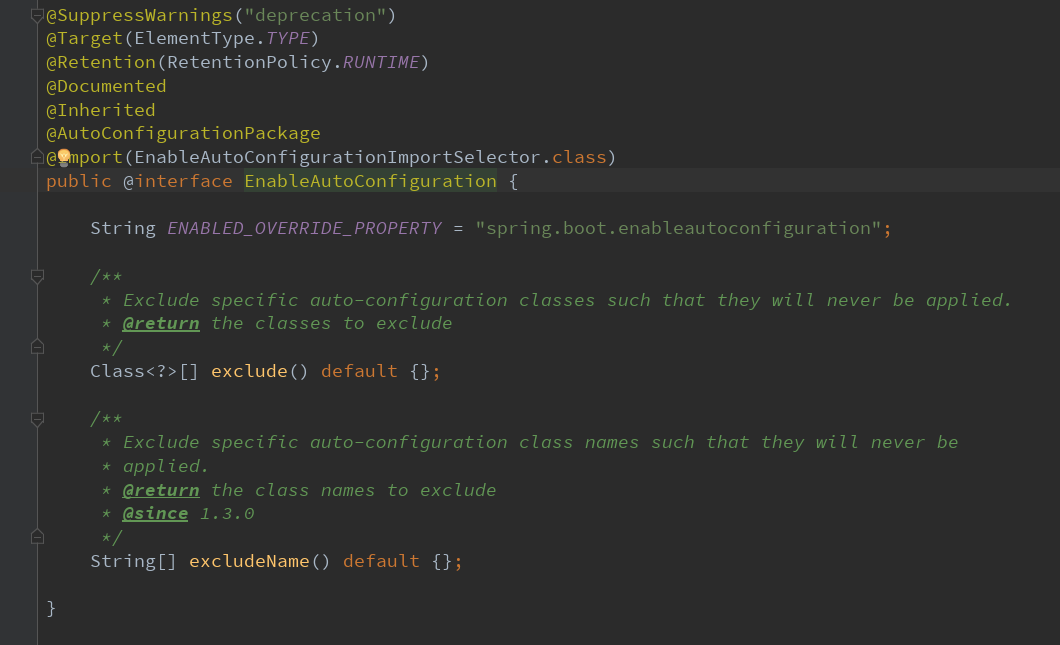
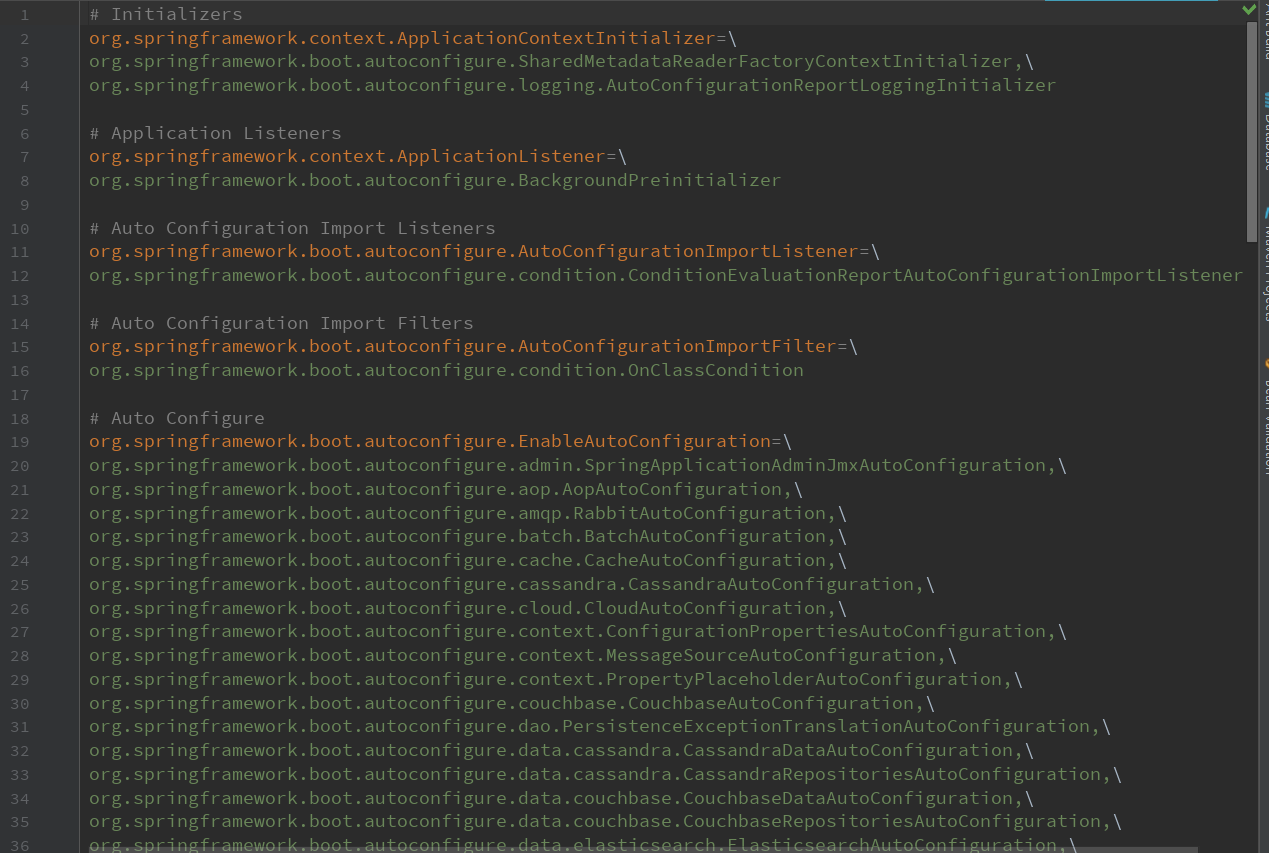
Spring Boot的自动配置时如何实现的呢？Spring Boot在启动时借助于注解@EnableAutoConfiguration。该注解的定义如下图3.1.3-1所示：

图3.1.3-1 注解EnableAutoConfiguration定义图

在这个注解定义里面，最为关键功能是@Import注解。其引入类EnableAutoConfigurationImportSelector，EnableAutoConfigurationImportSelector使用SpringFactoriesLoader.loadFactoryNames方法来扫描类路径下具有MEAT-INF/Spring.factories文件的jar包（1.5版本以前使用EnableAutoConfigurationImportSelector类，1.5以后这个类废弃了使用的是AutoConfigurationImportSelector类），通过IDE找到其中一个依赖jar，下面是Spring-Boot-autoconfigure-1.5.4.RELEASE.jar下的MEAT-INF中的Spring.factories文件的部分内容。如图3.1.3-2所示：

图3.1.3-2 Spring.factories默认配置项截图

基此，我们可以发行自己约定的自动配置项。

### Spring Boot打包、部署

Spring Boot让部署变得更简单，主要是源于其内嵌Web容器，这使得开发者在开发调试阶段就很轻松，开发者无需关心Web容器，只需要专注业务逻辑开发就好了，因而提供了无与伦比的便利。在持续构建领域使用的最为广泛的工具之一是Jenkins ，它是一个独立的开源自动化服务器，可以用于自动化构建，测试和部署软件的任务。Jenkins 结合Docker能够迅速部署项目，甚至可以运行在安装了Java Runtime Environment 的任何机器。而且Spring Boot + Docker + Jenkins 将会越来越受欢迎，也会将 Spring Boot 项目的部署做得更简单化、智能化。

其次就是基于maven的打包部署，maven能够将项目系统打包成脚本运行的jar包形式，当然这个jar包内嵌了Web容器，如Tomcat等。它也能将系统项目打包成war包的形式，放到Web服务器的Tomcat中实现Web服务。

### Spring Boot监控

Spring Boot 就是一个自带监控功能的开源框架，在Spring Boot项目设计之初就把监控功能考虑到位，并且为其专门提供了一款组件来完成这个任务，这个组件是 Spring Boot Actuator 。可以在pom文件中引入该组件依赖，其可以查看系统应用的详细配置信息，比如系统中容器创建并管理的 beans、系统自动化配置的相关信息以及一些环境属性等。

## Druid技术介绍

Druid是阿里巴巴开源的Java数据库连接池，它的功能不仅仅是数据库连接池，它还具备SQL监控功能，即在打开SQL监控功能后能够清楚地看到SQL的执行情况与执行效率。同时它还具有一个SQLParser 功能，能够解析SQL语法是否正确。

其设计模式基于责任链式，便于插拔模块，使用更灵活。阿里巴巴是一个重度使用关系数据库的公司，他们在生产环境中大量的使用Druid数据库连接池组件，通过长期在极高负载的生产环境中实际使用、修改和完善，让Druid逐步发展成最好的数据库连接池。Druid在监控、可扩展性、稳定性和性能方面都有明显的优势。

* + 1. Druid监控

Druid提供的监控特性式基于其提供的监控功能。通过这个功能，我们能够清楚的知道连接池状况以及SQL的工作情况。主要体现在如下几个方面：

1. Druid能监控到SQL的执行时间、ResultSet持有时间、返回行数、更新行数、错误次数、错误堆栈等信息。
2. 具备监控SQL执行所消耗的时间区间分布功能。所谓的执行区间是指某条SQL执行所消耗的时间区间占比分布。在耗时区间分布的功能下，开发者能够很清晰的知道SQL的执行所消耗的时间情况。
3. Druid会监控连接池的逻辑连接的申请和关闭次数、物理连接创建和销毁次数、非空等待次数、PSCache命中率等。

Druid的监控功能让开发者在时刻掌握数据库连接的状态，同时也能及时排查线上问题，为系统的顺利运行保驾护航。

### Filter-Chain模式的扩展API

Druid提供Filter-Chain模式的插件框架，开发者可以通过编写自定义的Filter并把它配置到Druid的DataSource中就可以工作。Druid实在JDBC底层进行的拦截，通过拦截JDBC的各种API，进而可以实现对API的个性化的扩展。同时Druid也提供了一系列内置Filter，不同的Filter链在一起，相互协作完成相应的功能。

* 1. Java并发包
     1. Java并发简介

在计算机发展史上，最早期的计算机程序就是一串指令，在随后大发展中引入了线程的概念，如果线程使用得当，能够提高系统的工作效率，同时也能降低程序的开发成本以及维护成本。到如今的多线程并发时代，开发者编写的多线程程序越来越多，也越来越高效。在Java中线程无处不在，在JVM启动时就会创建一个线程做JVM内部工作。

* + 1. Java内存模型

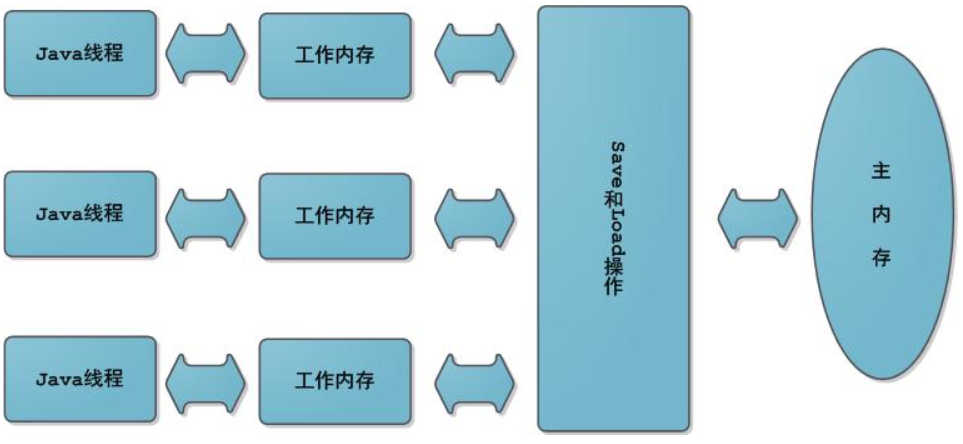
Java并发包，即java.util.concurrent.\*是Java体系中核心功能之一。Java并发技术涉及到Java内存模型，其设计思想如图3.3.2-1所示：

图3.3.2-1 Java内存模型

Java内存模型规定所有的变量内容都必须存储在主存中，每一个工作线程也都拥有自己的编程内存，所有的线程对变量的操作都必须在其工作内存中进行，不能够直接操作主内存。并且不同线程之间也是不能够相互访问到工作内存的内容，线程间的信息交流只能通过主内存进行。其工作关系就如上图所示。

因为课题系统中使用到了Java并法宝中的ScheduledThreadPoolExecutor类，所以这里只简单介绍其工作原理，该类主要用于执行定时任务，其能够在制定延迟事件之后执行任务，也能够执行周期性重复的任务。该类继承了ThreadPoolExecutor，

并且实现了接口ScheduledExecutorService。接口的主要方法如图3.3.2-2所示：

图3.3.2-2 接口定义

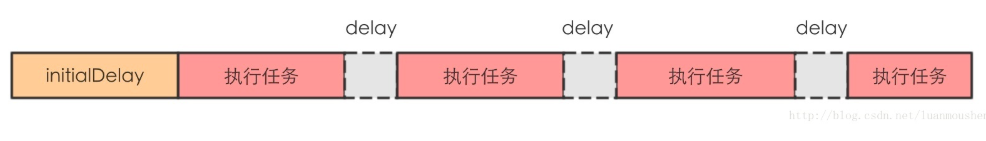
其中最常用到的是执行重复任务的方法scheduleWithFixedDelay，该方法的第一个参数是要周期执行的任务，第二个参数是在系统启动后在延迟initial时间后开始执行该周期任务，第三个参数是周期执行任务的周期时延，最后一个参数是时间单位。其工作模式是每延迟周期时间执行一次任务，工作模式如图3.3.2-3所示：

图3.3.2-3 工作模式图

# 网络大数据同步系统设计

## 系统总体设计

本课题系统采用的是B/S架构方式，数据源提供的服务端是一个B/S系统，数据的同步客户端也是一个B/S系统。下面两图分别展示了课题系统的服务端与客户端的整体架构。如图4.1-1，图4.1-2展示系统架构图。

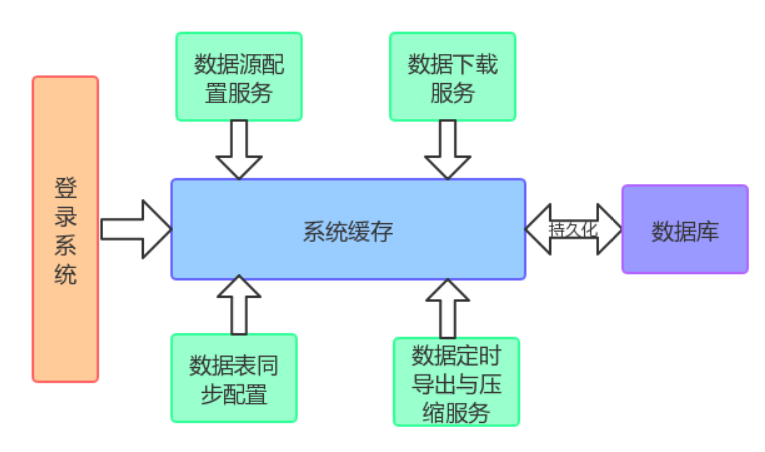


图4.1-1 服务端系统架构图

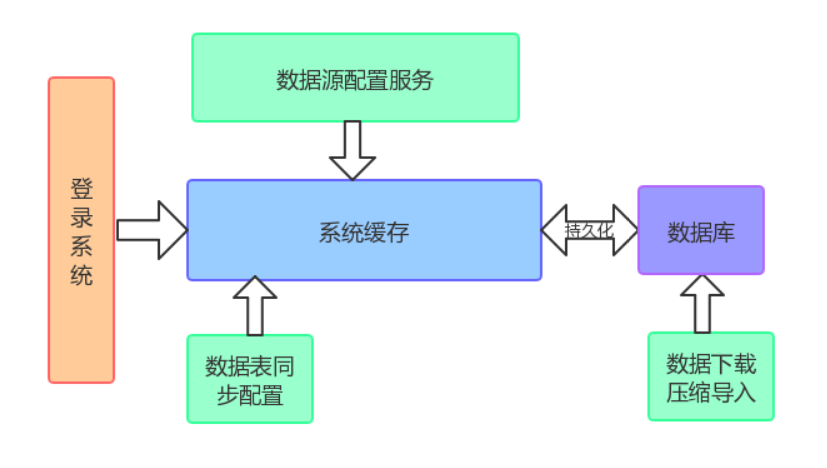


图4.1-2 客户端系统架构图

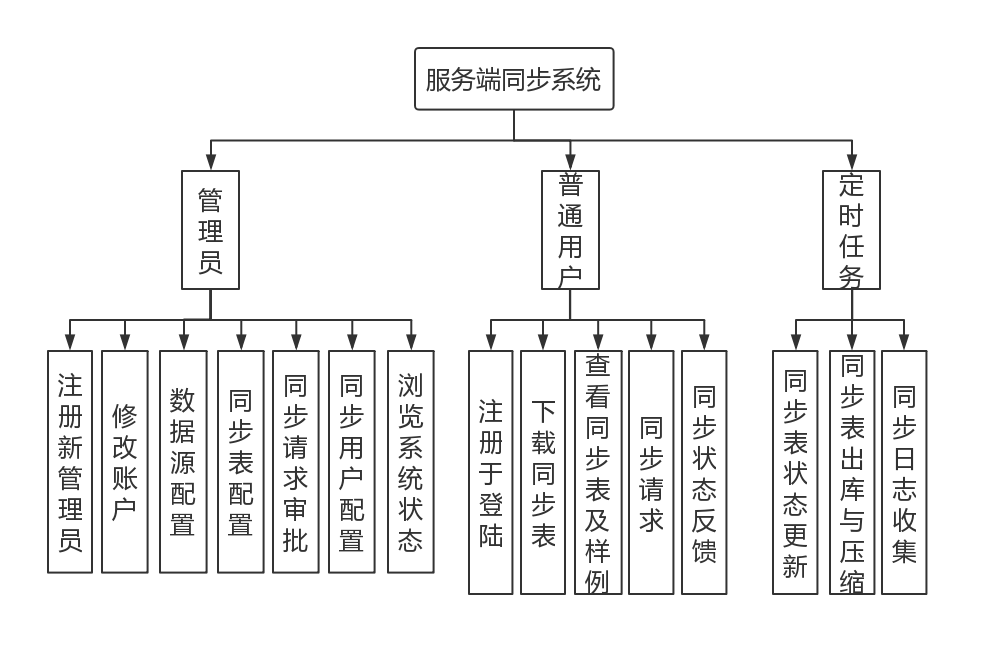
服务端的主要功能是数据源配置，同步表配置以及同步请求审批。其功能模块结构图如图4.1-3所示。

图4.1-3 服务端同步系统功能模块示意图

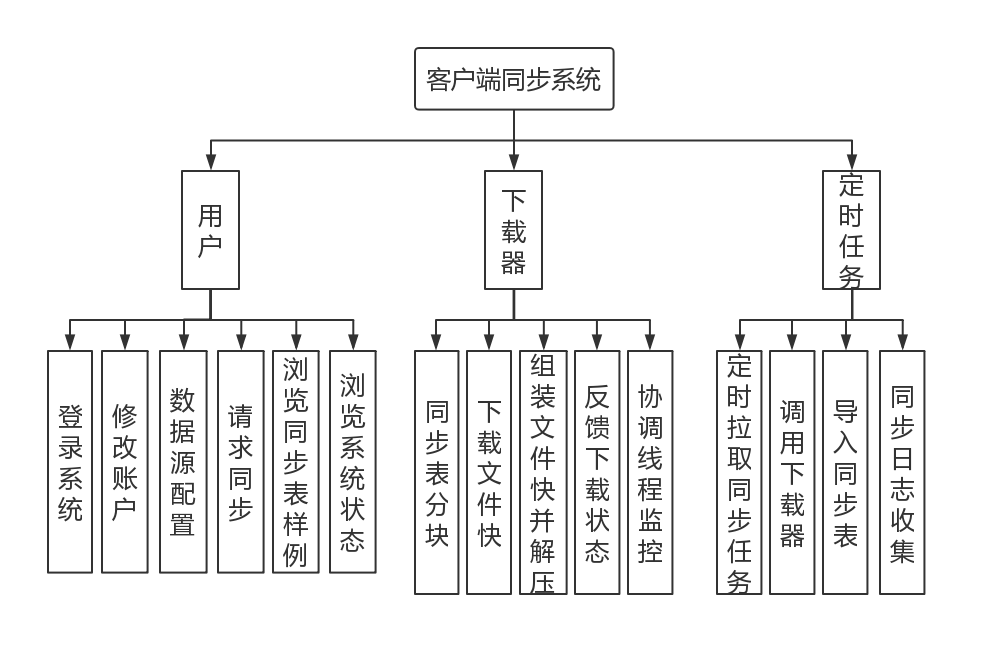
 客户端的主要功能是查看服务端可以同步的数据表，能够热切换数据源以及请求同步表的功能。其功能模块结构图如图4-4所示。

图4.1-4 客户端同步系统功能模块示意图

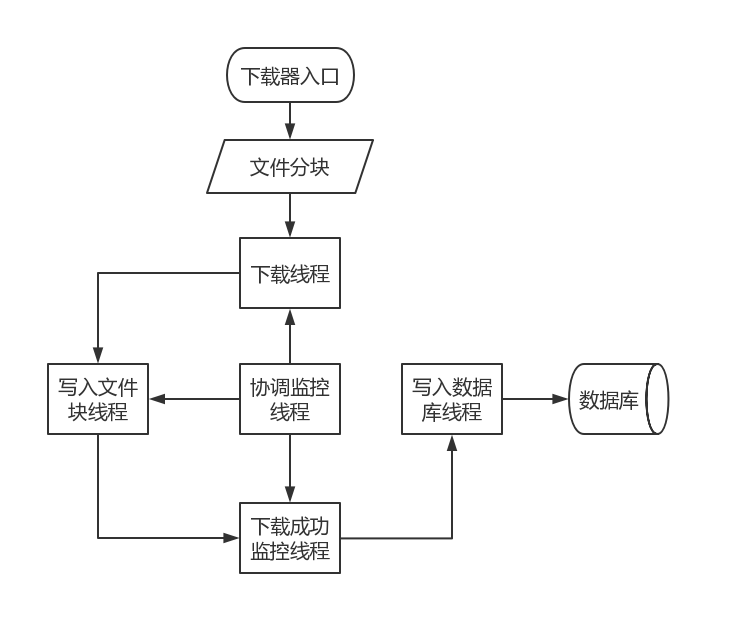
该课题系统的核心难点在于多线程下载器的设计与实现。该下载器具有多线程下载、断点续传、记忆下载以及下载熔断的保障机制。断点续传的设计思路是采用文件块的形式，将在服务端压缩的网络大数据在客户端分块进行多线程下载，并记录每一块文件块的下载状态，最后合并成最终文件；断点续传基于文件块下载状态，对下载成功的文件块记录保存，避免下载达到99%时下载失败丢弃所有已下载成功的文件块，节约网络资源。下载熔断也是基于文件快状态，当网络波动时会调度下载线程，使其处于挂起状态，保留一个线程不断尝试下载，如果尝试线程下载成功，将恢复所有下载线程的工作。其设计结构图如图4-5所示。

图4.1-5 多线程下载器设计图

## 主要业务流程设计

整个课题项目主要实现企业间数据的同步功能，项目启动时依赖系统配置文件，需要先修改配置文件。根据系统设计总结出整个流程的交互情况如图4.2-1所示，总共需要9个步骤：

1. 服务端管理员登陆系统。
2. 服务端管理员配置数据表是否位同步表。
3. 服务端管理员配置同步表的同步用户。
4. 客户端启动，向服务端注册账户。
5. 客户端登录系统后查看所有同步表以及数据样例，并可以申请同步。
6. 服务端管理员审批客户的同步请求。
7. 客户端感知到审批结果，通过将等待开始拉取数据。
8. 服务端后台每天凌晨准备好数据后等待客户端下载。
9. 客户端下载数据，并执行入库任务，并将结果反馈给服务端。

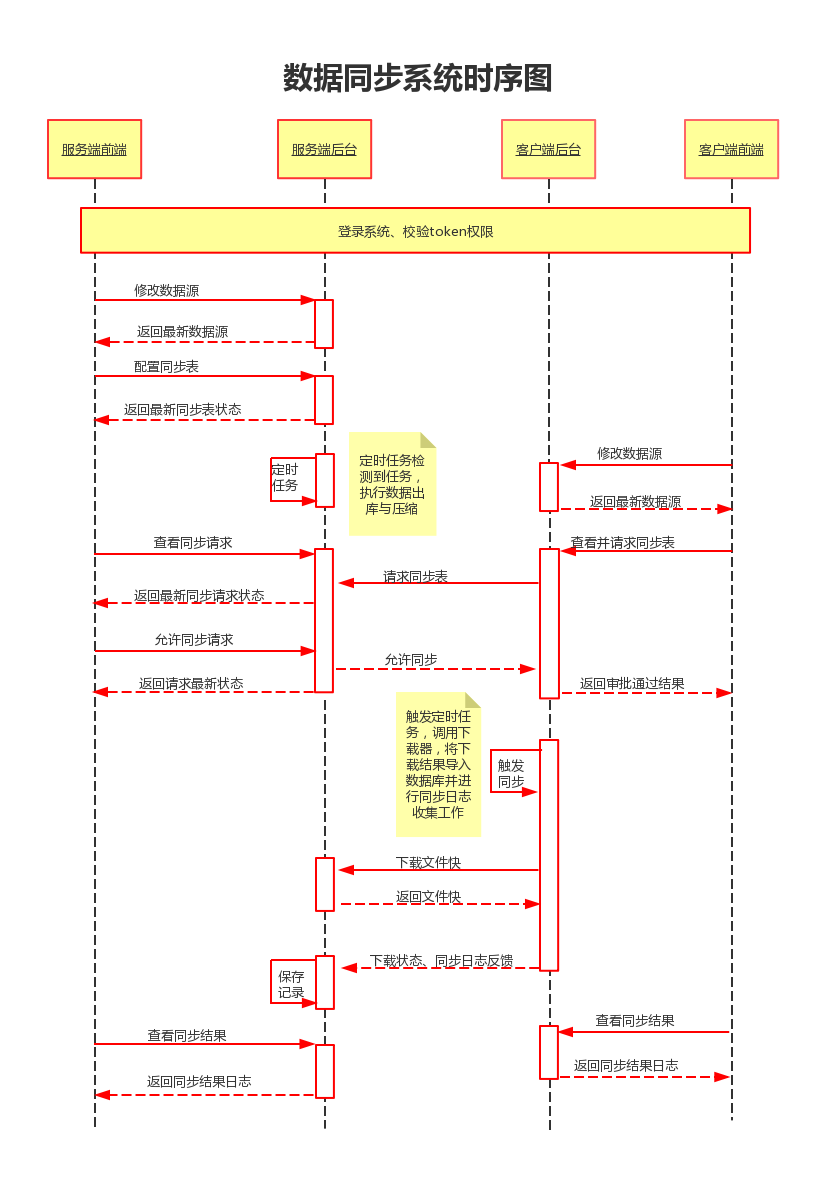


图 4.2-1 系统业务流程时序图

## 部分API设计

### 用户登陆

method: POST

url: /multiDownload/login

功能描述：提交包含用户名和密码的JSON数据，获取Token

### 修改某张表的同步状态

method: POST

url: /multiDownload/changeOneTable

功能描述：提交包含表名、同步状态与同步用户的JSON数据

### 修改数据源

method: POST

url: /multiDownload/changeDataSource

功能描述: 提交包含用户、密码、DB类型以及DB的url的JSON数据

### 获取所有表的同步状态

method: POST

url: /multiDownload/allTableStatus

功能描述：服务端用于查看数据便同步状态以及同步表配置

### 获取某张表的结构信息

method: POST

url: /multiDownload/tableStruct

功能描述：发送包含表名的JSON信息，用于客户端创建同步表的任务。

### 获取服务端所有同步状态的表

method: POST

url: /multiDownload/allSyncTables

功能描述：用于客户端查看所有同步表。

### 客户端请求同步某张表

method: POST

url: /multiDownload/requestSyncTable

功能描述：提交包含表名和用户名的JSON数据。

### 服务端审批同步表请求

method: POST

url: /multiDownload/confirmSyncTable

功能描述：提交包含表名、用户名以及审批码（只能为0，3，4）的JSON数据。

### 客户端获取待下载文件列表

method: GET

url: /multiDownload/fetchDataList

功能描述：用于客户端获取可同步表的列表。

### 客户端下载文件块

method: POST

url: /multiDownload/download

功能描述：提交文件块描述信息的数据，返回对应的序列化文件块。

### 通知服务端文件下载完成与入库状态

method: POST

url: /multiDownload/markDownloaded

功能描述：客户端下载完文件后告知服务端下载完成服务端对带下载文件进行状态管理

### 客户端用户获取该用户的所有同步表

method: POST

url: /multiDownload/userTable

功能描述：提交包含数据表名的JSON数据，查询客户端用户获取自己账户下的所有同步表。

# 网络大数据同步系统实现

## 实现综述

该课题系统采用Java语言编写。主要实现两个B/S系统，分别为服务端与客户端。在编码实现方面，采用了支持快速开发的Java Web后台框架Spring Boot，系统涉及到多数据源，因此采用阿里巴巴集团开源的支持多数据源的数据库连接池工具druid作为多数据源热切换的第三方组件，两个B/S系统都采用Thymeleaf为模板引擎，方便生成前端代码。在前端采用jQuery，样式上采用较为流行的前端框架Bootstrap，其提供简约的UI风格能让用户愉快的工作。使用Tomcat为Web容器对外提供Web服务，采用lombok小工具类提高开发效率，是编码风格更加简约化。jQuery前端使用框架和工具库如表5-1所示，Spring Boot后端服务依赖的框架和工具库如表5-2所示：

表5-1 jQuery前端使用框架和工具库

|  |  |  |
| --- | --- | --- |
| 库或框架 | 版本 | 作用 |
| jQuery | ^3.3.1 | 前端基础框架 |
| popper | ^1.14.3 | Tooltips提示小插件 |
| jquery.serializeJSON | ^2.9.0 | 前端数据序列化成json |
| chart | ^2.7.2 | 图标可视化工具 |
| Bootstrap | ^4.1.3 | 前端UI框架 |

表5-2 Spring Boot服务端使用框架和工具库

|  |  |  |
| --- | --- | --- |
| 库或框架 | 版本 | 作用 |
| Tomcat | ^2.0.0 | 后端Web容器 |
| Druid | ^1.1.4 | 数据库连接池 |
| Thymeleaf | ^1.9.22 | 前端模板引擎 |
| jjwt | ^0.9.0 | Token生成与校验 |
| Lombok | ^1.16.18 | 代码简介 |
| Fastjson | ^1.2.31 | Json解析与生成工具 |
| mysql | ^5.0.1 | MySQL数据库驱动 |
| ojdbc7 | ^12.1.0.1 | Oracle数据库连接驱动 |
| sqljdbc4 | ^4.2 | SQL server数据库连接驱动 |
| Maven | ^4.0.0 | 项目依赖包管理 |

## 开发环境

好的开发环境就能提高开发效率。当项目工程的工程量达到一定程度时，项目的依赖以及第三方工具就会变得越来越多，因此在开发大型项目的时候选择对的开发环境尤为重要。

### 系统平台与编辑器

该研究课题系统是在windows上开发，因为windows操作系统目前来说是电脑操作系统里的佼佼者，依托windows系统开发网络大数据同步系统，这样使得系统的市场会更大。其次，代码编辑采用IntelijiIDEA编辑器，其由Java编写而成，功能强大，是目前Java开发的主要编辑器之一，同时也是最受欢迎的编辑器。其主要特点是简单易用且功能强大。下面将介绍该编辑器的主要功能和插件：

1. 代码规约检查插件: 用于规范化团队编码风格，有利于后期系统维护。
2. GitHub: 借助git功能，具备git代码版本以及代码仓库备份与协作开发。
3. 多线程调试: 该IDE对多线程调试支持的较为完善，方便多线程编码与调试
4. translator 中英文实时翻译插件，适合阅读源码注释时，对陌生单词的理解。
5. docker 能够很好的集成docker到开发环境，也能很好的支持其部署。

### 浏览器开发环境

本次课题系统在前端使用Chrome浏览器进行系统编码测试开发，Chrome浏览器由Google公司开发，是开发者首选的浏览器之一，其对不同时期以及版本的Web标准兼容程度较高，并且其使用V8作为其引擎使得JavaScript具有较高的执行效率，因此Chrome浏览器也就成为开发者的强大的开发者工具。该课题系统主要使用Chrome浏览器的以下几个功能：

1. 审查元素：用于查看网页源代码，能够迅速定位到元素所在位置，方便查看模板引擎生成的html是否符合预期，便于调试项目。
2. 控制台：用于在浏览器中的console打印相关信息，同时该功能也能执行JavaScript代码段，能够验证JavaScript是否有问题，也便于调试。
3. 网络选项：可以查看页面请求的数据，能够即使查到接口是否正常，便于进行前后台联调，测试接口。

### 服务端开发环境

本系统采用的Java语言开发，对开发环境的要求就是使用JVM，在windows平台下，需要下载Java JDK。系统基于脚本运行时将采用Spring Boot提供的内嵌Tomcat容器，只需将系统工程借助maven打包成jar。系统如果不是基于脚本运行，需要借助maven打包成war包，放到Web服务器中，一般可采用nginx+Tomcat的组合方式，nginx处理静态页面，同时对静态页面做缓存，把动态页面交给Tomcat去处理，实现Web服务的动静分离，提高系统的工作效率。

## 系统实现

系统分为服务端与客户端两个B/S系统。下面将分别介绍两端的具体实现。

### 客户端的实现

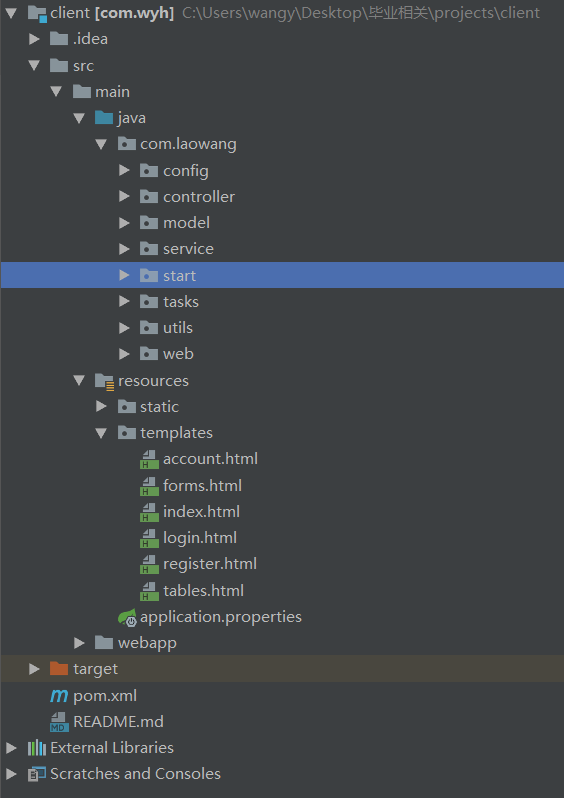
客户端基于Spring Boot做的Web系统，采用MVC架构方式，前端页面采用模板引擎Thymeleaf，界面UI使用的是BootStrap脚手架。整体项目的文件结构如图5.3.1-1所示

图5.3.1-1 客户端目录结构图

下面将分别介绍项目文件下的主要java文件。

Config包下面包含DruidDBCofig文件，其主要用于个性化定制符合系统项目的数据源，便于在系统启动时，更好的维护数据库连接。主要在该文件内配置数据库连接池的主要参数。其次，该包下还包含HttpMessageConverter文件，其主要用于转换HTTP请求中JSON数据。

Controller包下面包含Web项目的所有控制器，在Spring的处理逻辑中作为请求处理器处理前端的HTTP请求，其将会把处理结果封装到ModelAndView中，并在templates目录下的模板中渲染，并返回给代理浏览器。

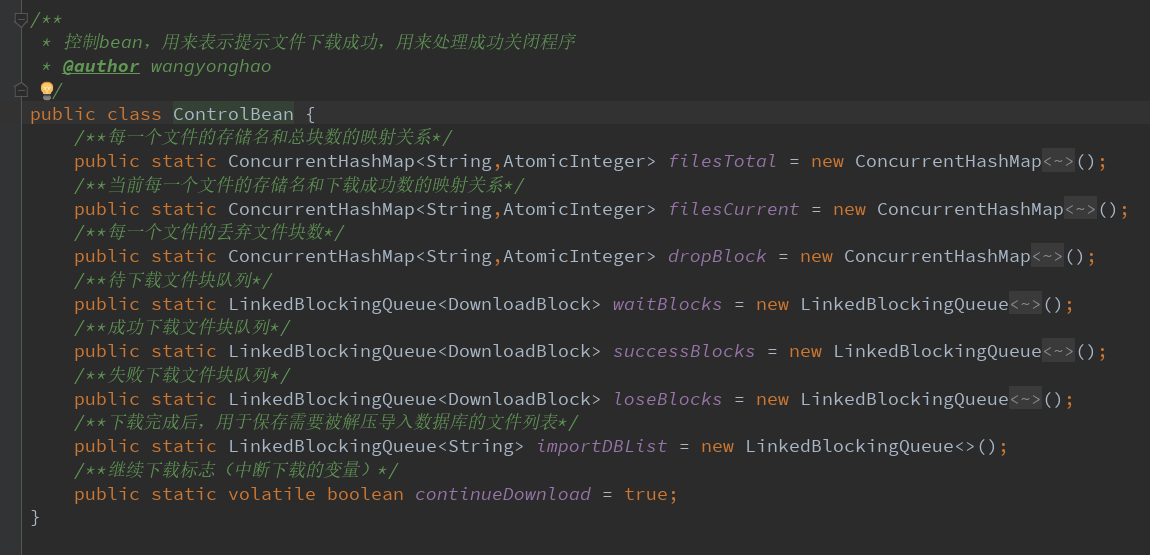
Model包中包含系统所需要的各种模型Bean的定义，包括现在控制bean，用户bean，日志bean以及带下载的文件快bean等等。其中将重点介绍和下载器相关的Bean类的定义，控制bean的定义如下图5.3.1-2所示：

图5.3.1-2 控制bean结构定义图

控制bean中定义的数据是下载器正常工作的核心定义。该多线程下载器的设计如图（上面的图）。该类中定义的属性是在多线程下工作的，需要保证线程安全，本系统在设计实现这块内容是采用Java中的并发包内的ConcurrentHashMap来做文件与文件块的映射，该HashMap类能够保证在多线程操作中数据的安全性，同时又不失其工作效率。文件块存放在阻塞队列中，本课题系统的在设计实现这个功能时采用Java语言提供的LinkedBlockingQueue阻塞队列。下载器的不同类型线程都会从对应的阻塞队列中获取不哦则操作文件块，这种设计使得线程工作单一，便于系统功效运行。

Service包中主要是处理系统的业务逻辑Java文件。

Start包中包含系统启动的启动类，以及系统启动后回调任务所做的初始化工作，如系统的同步状态以及用户同行权限token的获取等等。

Tasks包中是下载器的文件，稍后将会详细介绍下载器的具体实现。

Utils包中包括系统所需要的一些工具类，主要用来辅助service等的工作。

Web包中只包含一个restTemplate的配置Java类。

Templates目录下主要存放的是Thymeleaf模板文件，用于渲染模型的视图文件。

### 多线程下载器的实现

多线程下载器在设计之时，采用文件分块的方式主要原因如下：

1. HTTP没有规定传输内容的大小，但是随着传输内容的增大，网络链路距离的增加基于TCP传输协议的HTTP报文就会越不稳定。
2. 传输大文件会占用昂贵的网络带宽资源。
3. 基于spring和tomcat的web应用处理HTTP请求是采用链式处理的，一个过大的HTTP请求会占用服务器的内存资源，导致内存不够用等问题。
4. 如果网络波动，重传也是会浪费网络带宽等资源。
5. 原始传输不支持断点续传，大文件传到99%失败，要重新下载实则浪费资源等。

下载器的设计思路是：对同步表压缩文件以分块的方式请求与传输。每个HTTP请求只会请求一个文件块去下载。采用多个线程去下载文件块，能够减少因网络IO带来的负面效率影响。客户端维护待下载文件以及其对应的文件块和下载状态，以便记忆用户下载记录、文件下载完成判断、多文件同时下载的支持、服务端挂了，客户端梯度行的尝试下载与中断下载逻辑判断、随机文件读写的方式保存文件块到本地。

下载器具体的工作流程将以下面一段文字介绍的方式进行说明。

初始化工作，下载器downloader接收到传入的文件名（服务端文件标示）和大小之后，对文件进行分块，每一块文件包含文件的属性，文件块之间相互独立。把文件块全部丢进待下载队列waitBlocks中，同时初始化filesTotal（每个文件名与总块数的映射），filesCurrent（每个文件名与成功下载的文件块数的映射），dropBlock（每个文件名与丢弃的文件块数之间的映射）键值对变量。

初始化完成后即启动3个下载线程（downloadThread），一个写线程（writeThread），1个重任务线程（reTaskThread），1个善后线程（dropStoreThread），它们的工作就像上图描述的那样，职责单一的处理任务。详细如下：

下载线程：从待下载队列中取出文件块去下载，下载成功把文件块添加到下载成功队列中，下载失败则把文件块添加到下载失败队列中，失败包含两种：其一为返回的response状态吗不是ok，待下载的服务端文件可能被修改或则不存在了，这个时候启动计数器，失败5次则丢弃文件块（避免浪费服务端连接资源），不把文件块加入到下载失败队列中，同时dropBlock映射更新；另外一种是获取不到服务端的响应，可能服务端挂了，网络链路有问题，这个时候会暂停所有下载线程，同时把对应的文件块加入到下载失败队列中，reTaskThread会用一定的策略（梯度增加时间的休眠，唤醒后打开下载线程尝试连接下载）把下载失败的文件块放回到待下载队列中，这样能控制下载线程和服务器的交互，避免网络中断还不间断的请求服务器。

写线程就比较单一，只负责把下载成功队列中的文件块取出来，写入到磁盘，及时释放内存，同时更新filesCurrent状态。

善后线程工作比较重要，它有两种工作状态：其一，当filesTotal和filesCurrent中维护的相同文件的文件块数相等时，表示成功下载文件，这时需要把写线程写的临时文件更名，同时和服务器交互，文件已经下载完成，服务端把下载标志位置为有效，以后获取的下载任务将不包含这个文件，和服务器成功交互后，会删除filesTotal和filesCurrent维护的对应文件映射。另一个是，如果dropBlock所维护的文件块不为0，并且dropblock的数量加上filesCurrent中对应文件的文件块数量等于filesTotal总数量时，说明此文件下载失败，会把writeThread写入的临时文件删除，同时把三个映射列表更新（移除对应的文件及其映射），等待下次获取待下载文件时下载。

文件块定义的描述文件截图如图5.3.2-1所示

图5.3.2-1 文件快定义描述图

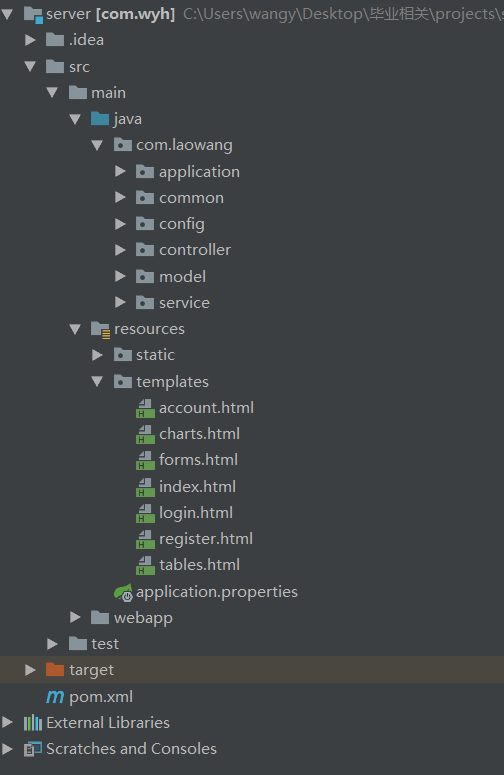
### Chrome浏览器测试

客户端的测试采用的是chrome浏览器，分别对用户相关接口以及下载相关接口进行测试，结果符合预期。

## 服务端实现

### 服务端的实现

服务端的整体架构与客户端的架构整体保持一致，也是基于Spring的MVC分层架构，其目录结构如图5.4.1-1所示：

图5.4.1-1 客户端项目目录结构

### MVC层的定义

在控制层，服务端设计了两种不同的cotroller，其中一种是控制页面跳转，另外一种是提供接口服务的，如登陆验证，以及下载文件块的接口等。这一层的主要接口在上文中已经有所提及，这里就不再赘述。

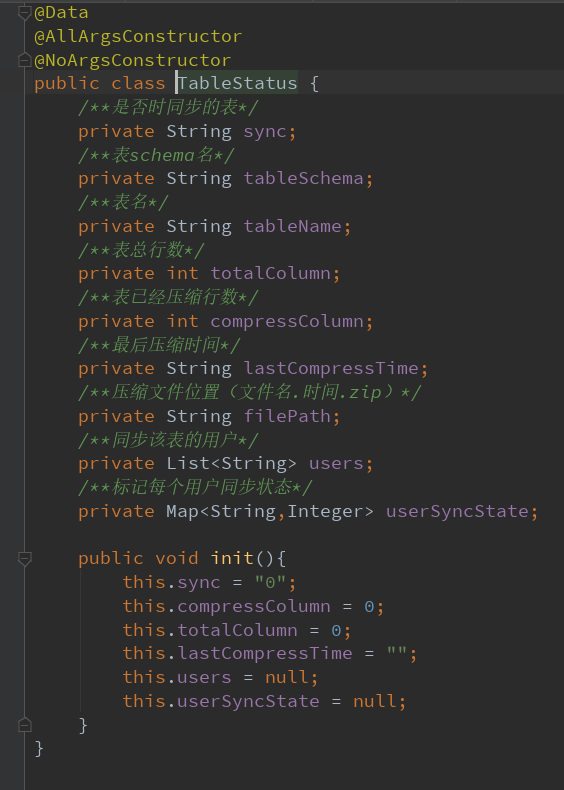
Model层，系统在设计中定义了数据源model，用户model，下载结果model，日志model以及同步表model。这其中最重要的是同步表model，其定义的对象缓存着系统的状态信息，其定义描述文件如下图5.4.2-1所示：

图5.4.2-1 数据表同步model

同步表model中sync属性表示数据表是否是同步状态；tableSchema属性表示该同步表在数据库的哪个schema下；tableName表示数据表表名；totalColumn表示当前数据表的所有记录数；compressColumn表示数据表已经压缩的行数；这里的totalColumn和compressColumn能够实现同步表的全量出库与增量出库，具体实现是，当系统初始化时，前者数据大于零，后者数据为零，将会进行全量导出数据库，在系统运行过程中，某次操作使得同步表的数据行数增加，进而导致totalColumn与compressColumn数据值不相等，触发增量数据导出数据库，具体将导出compressColumn+1~totalColumn的所有记录，从而实现了同步表的增量出库同步。

View层的视图文件模板存放在项目的templates文件夹下，其主要包含七个页面，分别是主页，数据源切换页面，数据表配置页面，账号修改页面，同步请求审批页面，登陆页面以及注册页面。控制页面跳转的控制器是PageCotroller。

### 服务端的测试

服务端和客户端一样，同样采用chrome浏览器进行测试，经过开发调试，最终所有测试均符合预期。。

## 主要界面实现

### 服务端界面

登陆系统



图5.5.1-1 服务端登陆界面

主页展示

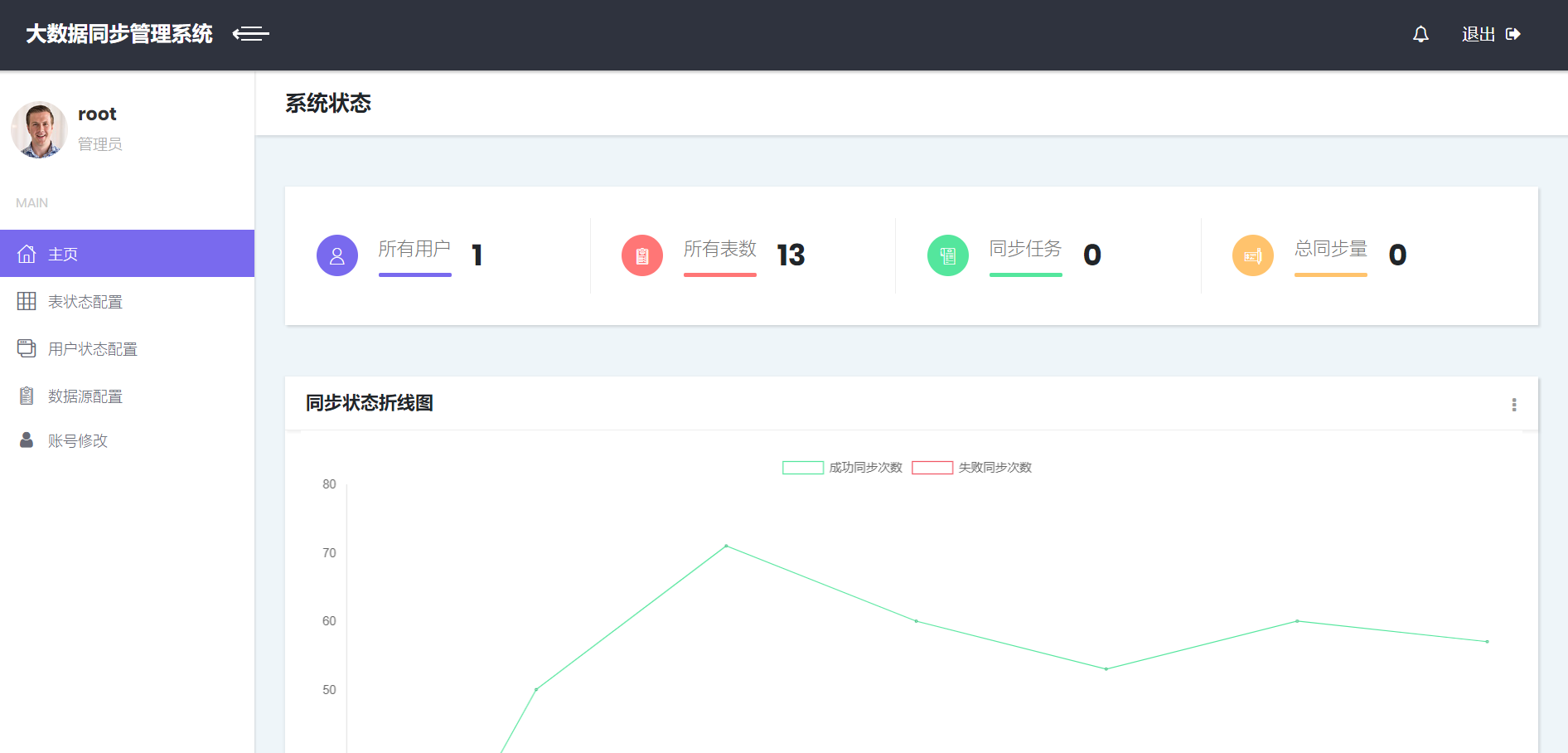


图5.5.1-2 服务端主页系统状态展示界面

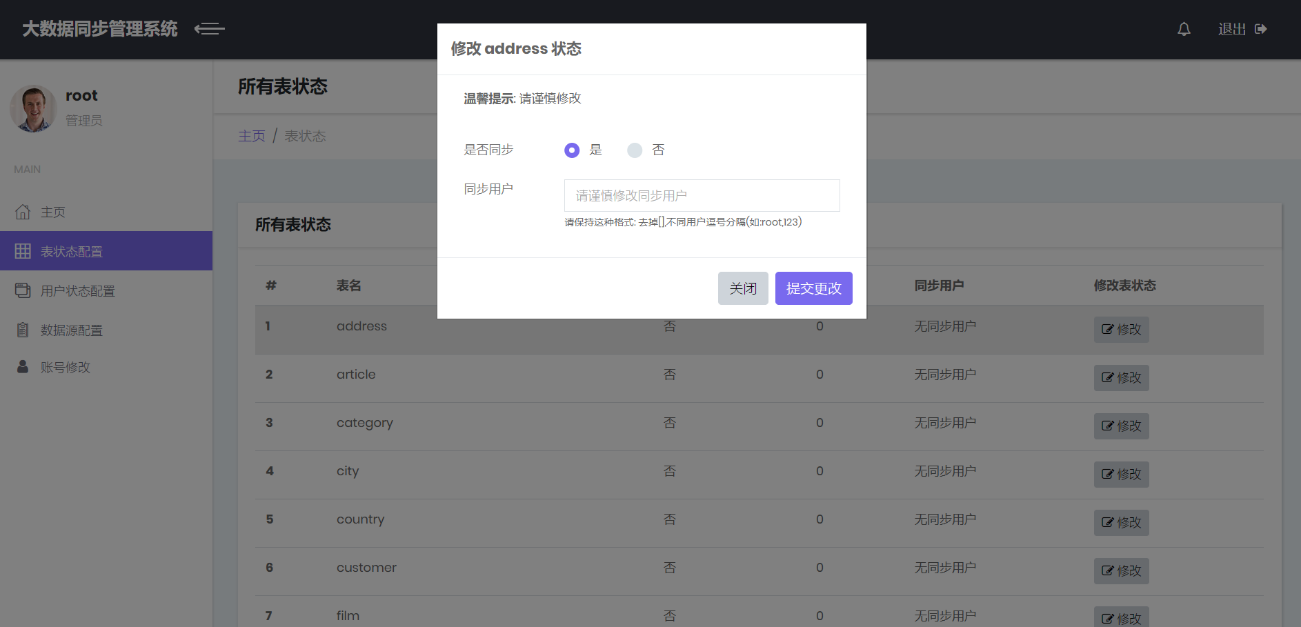
同步表状态配置页面

图5.5.1-3 服务端同步表状态配置界面

同步请求审批页面

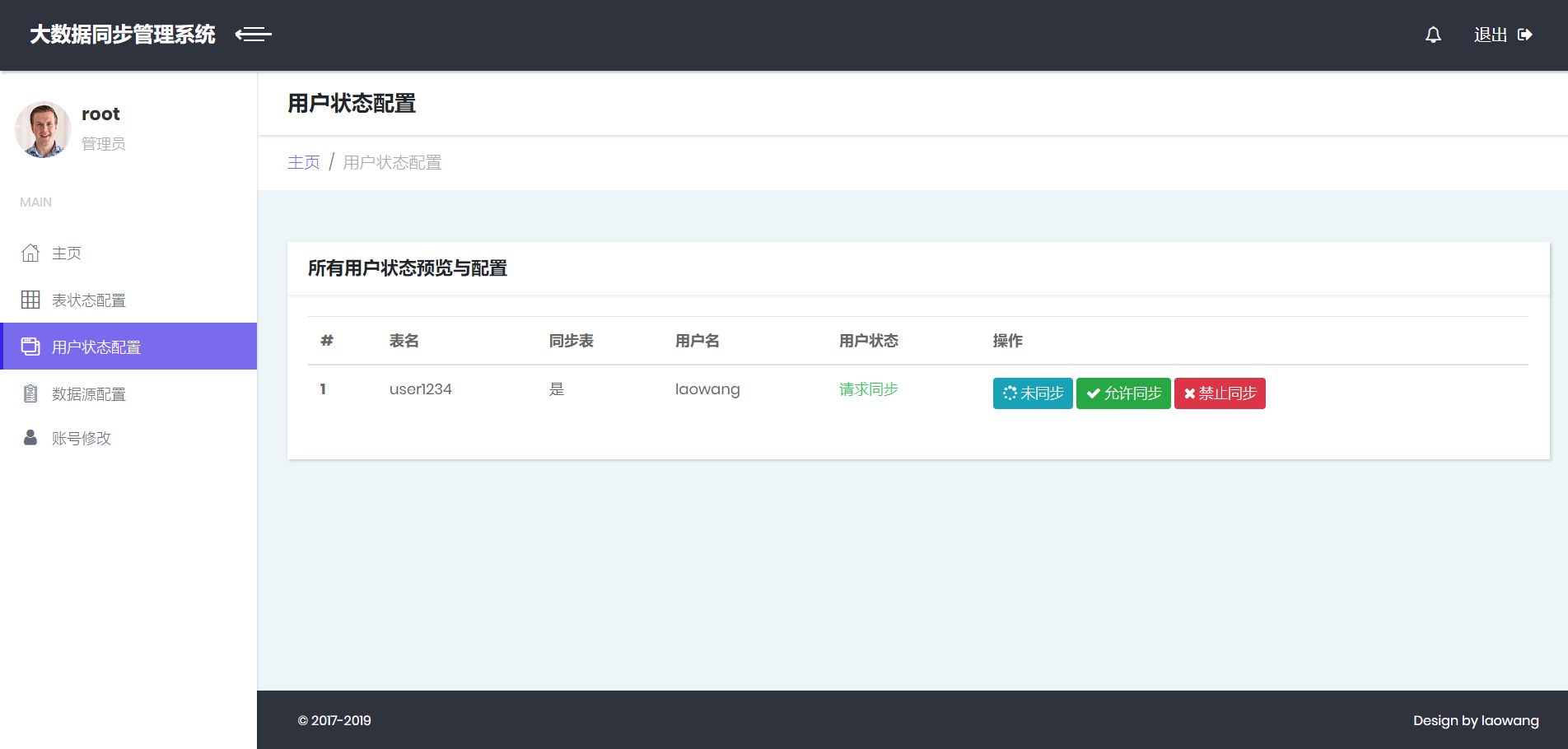


图5.5.1-4 服务端同步请求审批界面

数据源配置页面

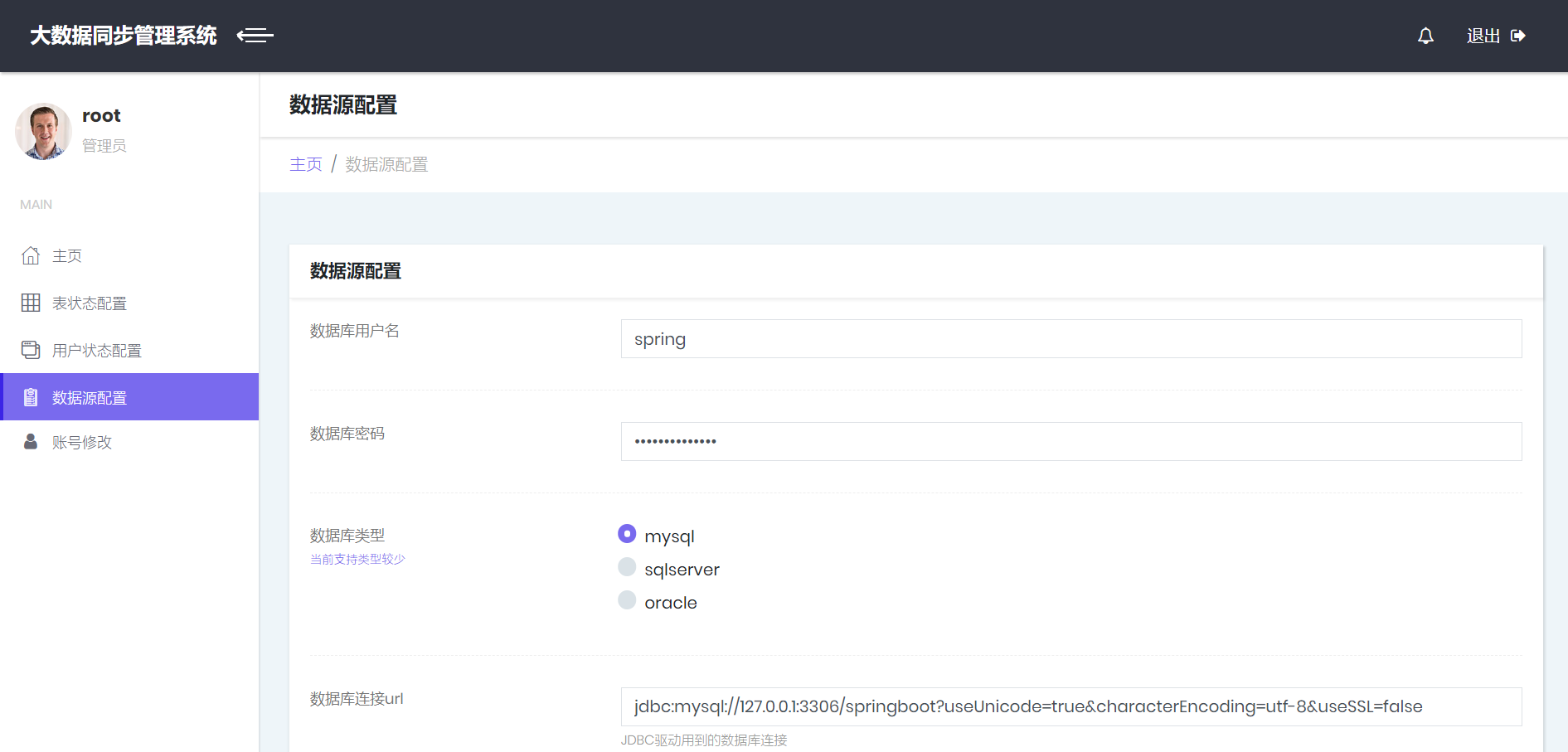


图5.5.1-5 服务端数据源配置界面

账号修改页面

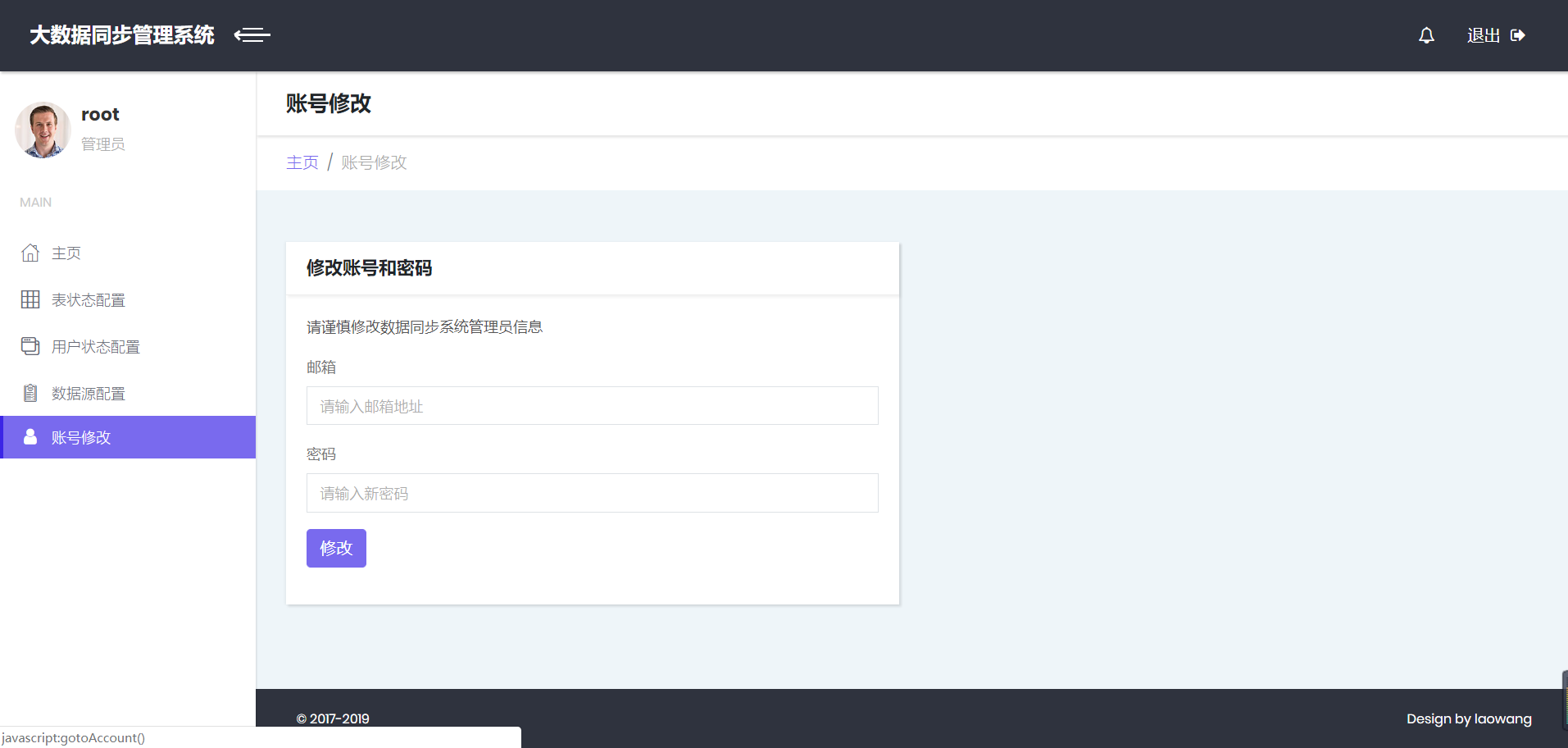


图5.5.1-6 服务端账号修改页面界面

同步日志状态展示

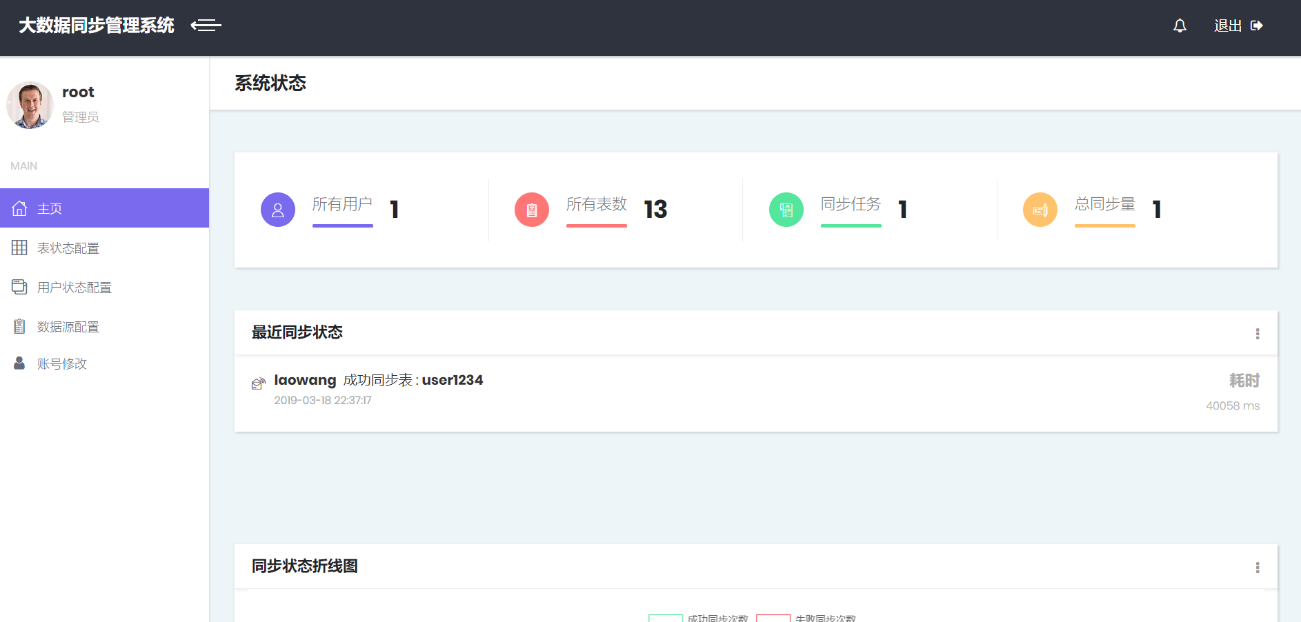


图5.5.1-7 服务端同步日志查看界面

### 客户端界面

系统展示界面

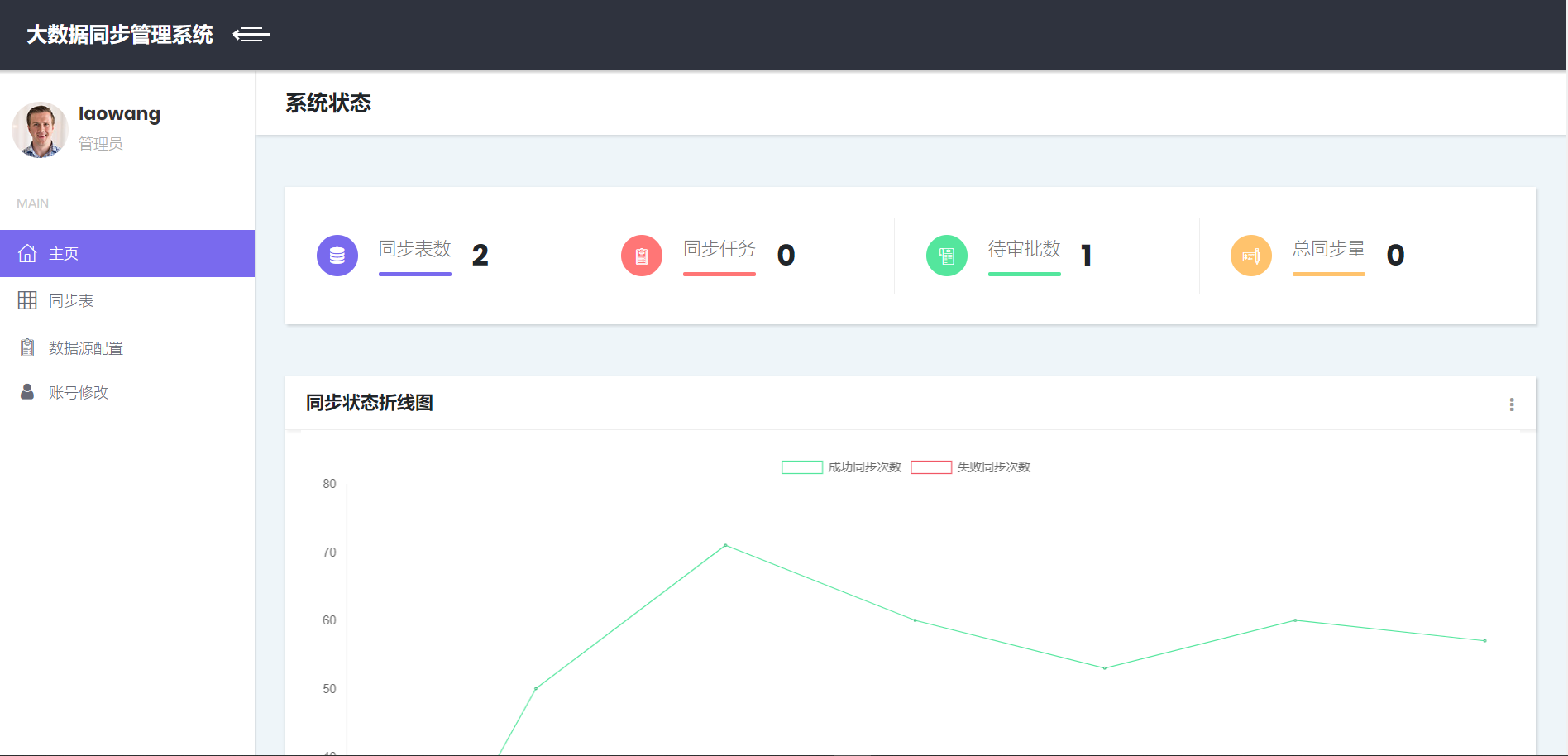


图5.5.2-1 客户端系统状态展示界面

查看同步表页面

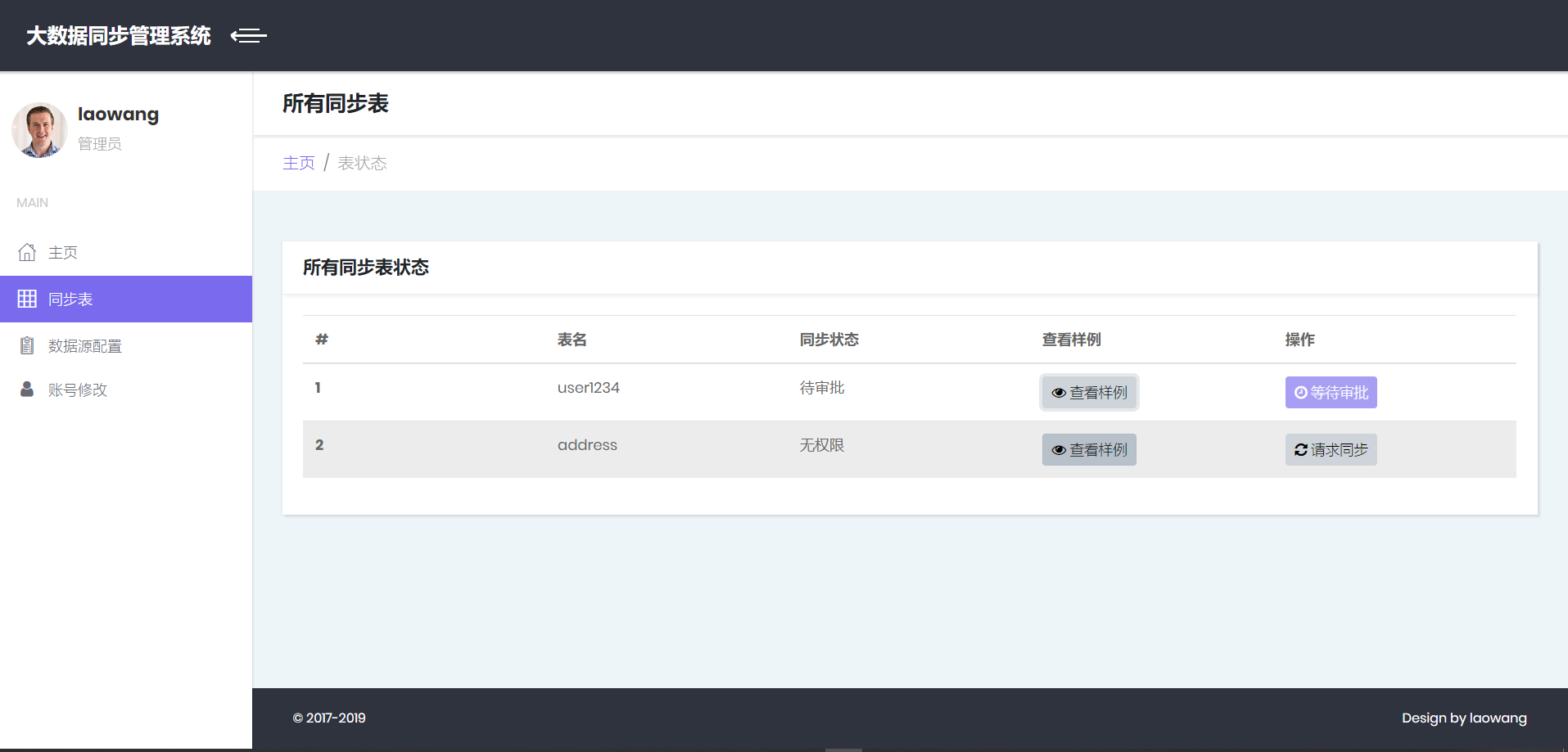


图5.5.2-2 客户端查看同步表界面

查看数据样例页面

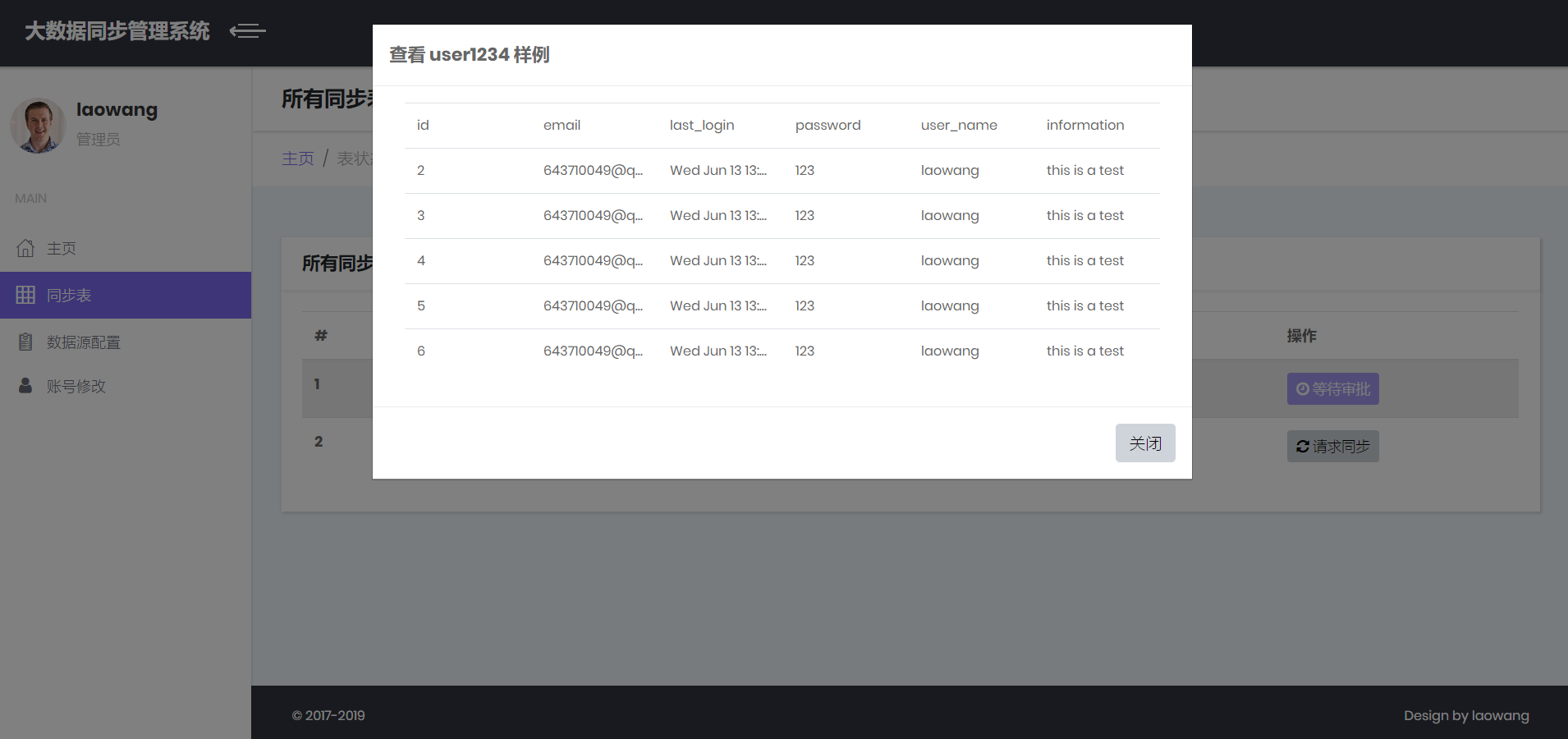


图5.5.2-3 客户端查看同步表数据样例界面

数据源配置页面

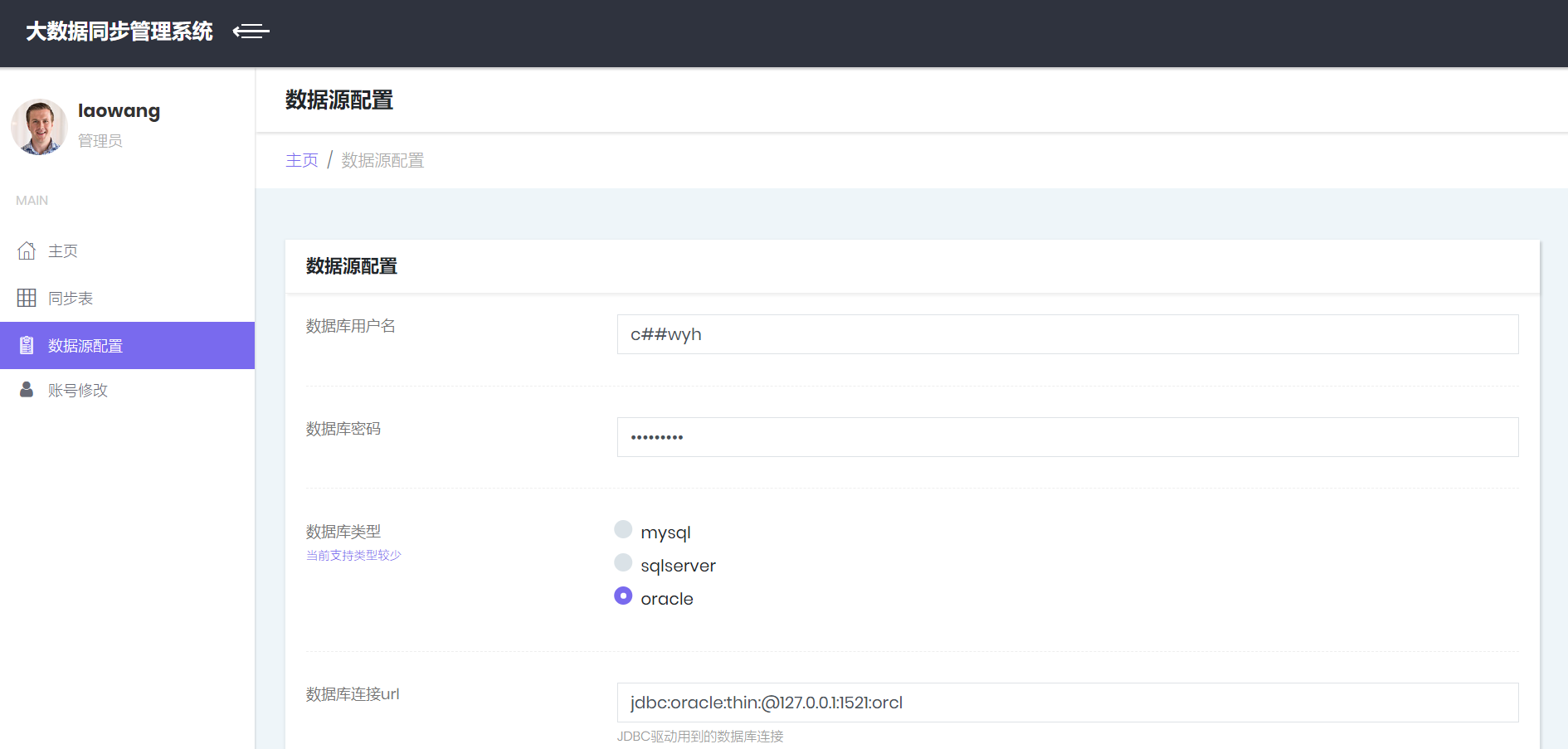


图5.5.2-3 客户端数据源配置界面

账号修改页面

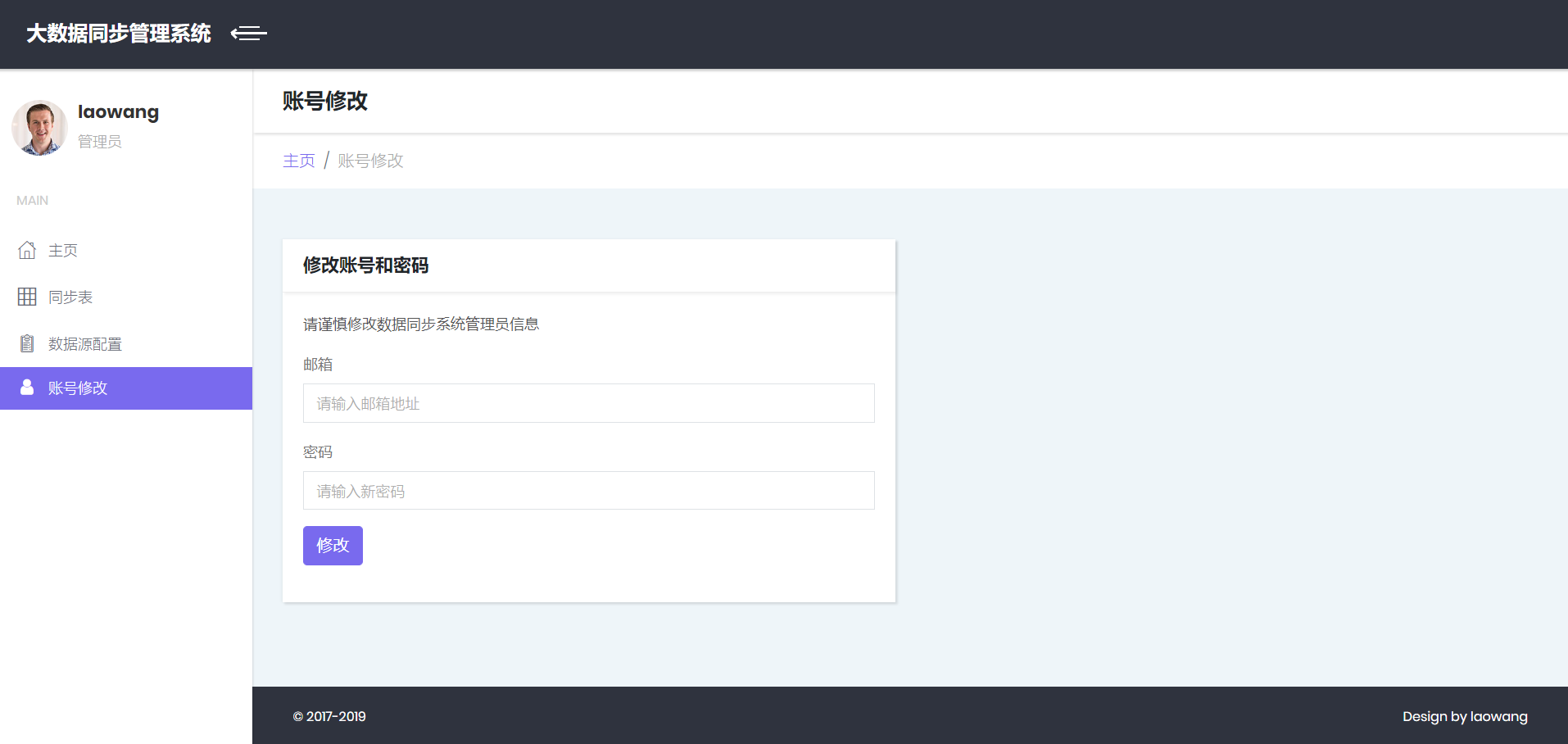


图5.5.2-4 客户端账号修改页面界面

查看同步日志页面

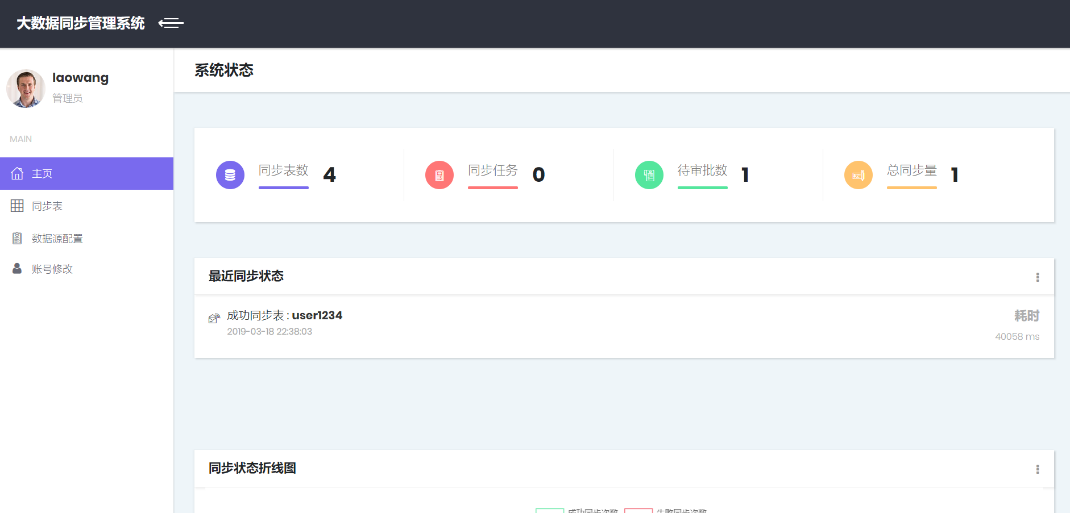


图5.5.2-5 客户端查看同步表日志界面

# 平台测试

测试是软件工程中重要环节之一，也是项目系统交付的最后环节。测试是必不可少的阶段，因为测试可以发现程序中的错误，并且能够调试错误执行的过程，便于对程序漏斗的早期发现。

本课题系统将搜先进行接口测试，检查接口是否正确工作，包括响应的数据是否正确等等。接着将会进行模块间测试，检查系统每个模块工作逻辑是否存在偏差，是否存在漏洞。最后将会继集成测试，服务端与客户端联动，检查数据同步是否能够符合预期。

## 单元测试

单元测试是集成测试的基本单位，其属于软件过程的一个阶段，不能被缺省。单元测试是最容易发现程序是否存在bug的一项工作。在编写完代码变异检查之后，就要进行接口测试。测试接口数据流是否正常进出，接口参数数目，次序是否和变元一致。同时检查系统缓存数据结构是否符合预期，有没有被接口异常修改。

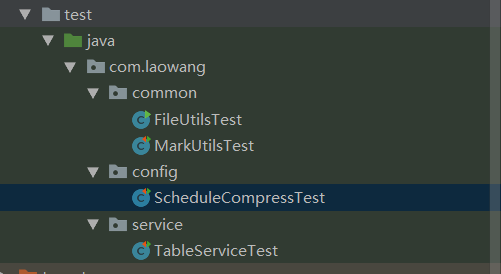
SpringBoot集成Junit4进行单元测试时最好用且最常用的方法。在IntelijiIDEA中可以很方便的快速生成测试用例，如图6.1-1所示是关键接口与功能的测试目录：

图 6.1-1单元测试目录结构

测试目录中的每个文件可以测试对应开发文件目录下的所有方法，我们可以针对每一个方法编写测试代码。Spring结合Junit4进行系统测试时，会启动整个项目，并在项目启动完成后自动依次执行全部测试方法。下面将展示其中一个测试文件中的测试代码，如图6.1-2所示：



图 6-2 测试文件内容截图

单元测试将会依赖系统启动后，检查测试用例是否符合预期，最终经过调试与修补结果全部符合预期。

## 接口测试

接口测试是系统开发中最常接触的测试，而且通常情况下会一边开发一边测试，本次课题系统的接口在开发时进行了初步调试测试，检查了接口返回的数据是否符合预期等，在系统完成后将会再次测试全部接口，所有接口都会带有token数据，用来鉴权，在下表中将不提及。部分接口测试结果如表6-1：

表6-1 API测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试接口 | 测试参数 | 返回结果 | 是否符合预期 |
| 用户登录接口 | 传递userName，password组成的JSON数据 | 返回结果码为0，  Data中包含token，登陆成功 | 符合 |
| 修改同步表状态 | 传递同步表tableName，sync组成的JSON数据 | 返回结果码为0，  同时返回罪行数据库所有表状态 | 符合 |
| 修改数据源 | 传递数据库连接用户与密码，数据库类型与连接url组成的JSON数据 | 返回结果码为0，  同时返回最新的数据源连接数据 | 符合 |
| 审批同步请求 | 传递包含用户名，表名与审批码组合的JSON数据 | 返回结果码为0，  同时返回同步表的最新状态信息 | 符合 |
| 发起同步请求 | 发送用户名以及密码组成的JSON数据 | 返回结果码为0 | 符合 |
| 下载文件块 | 发送文件路径、文件块的开始位置以及结束位置组成的JSON | 返回结果码为0，  同时包含文件快对象的序列化 | 符合 |
| 编辑服务端下载状态 | 传递文件路径，用户信息组成的JSON数据 | 返回结果码为0，同时msg信息为成功 | 符合 |

## 集成测试

集成测试是整个测试中最后的环节，将会进行服务端与客户端联调的方式进行。下面将分别服务端与客户端两个方面测试功能的正常行。服务端的功能测试见表6-2，客户端的功能测试见表6-3。

表6-2服务端部分测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试功能 | 测试过程 | 测试结果 | 是否符合预期 |
| 登录功能 | 打开相应的网页页面，填写用户名和密码提交 | 弹出登陆成功提示框，页面跳转到系统状态显示 | 符合 |
| 修改数据源 | 提交新数据源所需要的相应数据源信息 | 弹出修改数据源成功，数据源状态回显到当前页 | 符合 |
| 修改同步表 | 点击进入同步表配置页面，修改某张表同步状态，以及添加同步表的同步用户组 | 单独修改同步表同步状态时，提示修改成功，添加用户组时，页面回显相应用户 | 符合 |
| 审批同步请求 | 点击进入同步请求审批页面，针对每一个同步请求，都可进行允许同步请求与禁止同步请求操作以及未同步 | 允许同步后，状态回显，客户端检测到审批通过。禁止同步请求时，服务端回显，客户端感知 | 符合 |
| 修改账号信息 | 进入账号修改页面，提交新的用户密码以及邮箱信息 | 弹出修改账号成功信息 | 符合 |
| 查看系统状态 | 点击系统主页 | 展示当前系统下数据表数、同步表数、同步请求数以及总同步次数信息，还有同步日志信息 | 符合 |

表6-3客户端部分测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试内容 | 测试过程 | 预期结果 | 结果 |
| 查看同步表以及数据样例 | 点击同步表页面，点击查看数据样例 | 页面展示服务端所有同步表，弹出相应同步表的5条数据样例 | 符合 |
| 修改数据源 | 点击修改数据源界面，提交新数据源的数据信息 | 弹出修改数据源成功信息 | 符合 |
| 请求同步表 | 点击同步表页面，点击请求同步按钮 | 显示同步表，  相应同步表的按钮变成待审批状态 | 符合 |
| 查看系统状态 | 点击系统主页 | 返回系统相应状态数据 | 符合 |

## 测试结果及分析

对数据同步系统进行了模块测试、接口测试以及集成测试，测试用例共计112个，总的测试结果如表6-4所示。

表6-4 测试结果表

|  |  |  |
| --- | --- | --- |
| 功能模块 | 用例数 | Bug数 |
| 服务端页面 | 12 | 0 |
| 客户端页面 | 10 | 1 |
| 服务端接口 | 50 | 0 |
| 客户端接口 | 40 | 0 |

测试完毕后，检查bug是否影响系统功能，并解决相应bug，系统正常工作。

# 总结与展望

经过长达数月的努力，最终实现了毕设项目系统的初期构想。系统总体完成度达到预期。本课题系统在大数据时代下具有一定的实用价值，它服务于数据，她能够实现数据的迁移备份与玩咯数据的定时同步，同时该系统还支持多数据源，能偶提供不同数据库类型下的数据同步解决方案。此外，系统还实现了数据买卖双方的互动，对数据买方提供对同步表的同步请求，同时对数据卖方提供同步请求的审批。系统的友好性较为完善。

在实现毕设系统的道路上，我也面对了很多困难与挫折，我也曾废寝忘食的调试bug。在实现系统前，一切的努力都是值得的！在实现系统关键功能上，我不惧困难敢于尝试。如在设计多线程下载器时，我经过三种版本迭代，因为初期设计考虑不周到，在多线程下载器服务于数据同步系统时，并没有很好的融合性。所以不断迭代版本，并进一步完善多线程下载器，最终才能够很好地应用到数据同步系统中。在这个方面，我总结出在代码开发中，前期准备工作很重要，不然中期的一点改动或者功能不吻合都会拖拉系统开发时间；还有在热切换数据源功能的开发上，我调研过很多方案，每种方案都进行尝试，失败也是接踵而至。最终在两种方案中选择最好、最高效的切换方式——采用druid重新创建数据源。在这个功能实现上，我感悟出尝试是寻找最好方案的好途径，面对新事物要不怕失败，敢于尝试！

在完成毕设系统后，发现系统还可以更完善，还能为系统添砖加瓦。系统还能在安全方面做出改进，比如突然系统崩溃，系统缓存数据可能因此而丢失，导致不可弥补的灾难。你解决方案是，在系统内开一个后台线程，该线程在系统启动时会检查文件系统内是否存在系统缓存快照，如果存在将把同步系统恢复到当前状态。该线程在同步系统工作时会定期的对系统的缓存做快照，并持久化到文件系统中。其次，数据同步系统还应该支持更多的数据源接入，当前系统支持Oracle、MySQL，SQLServer三种数据库，支持的数据库种类较少。这些展望的功能在未来的时间内，我会安排时间把我的想法实现。

本毕设系统是对我大学生涯的技术检验，同时也是证明我个人能力的有效课程。在这次课程中，我真刀实战的为我的毕设付出心血，同时也收获了很多，我学会了开发语言、工具及框架，同时我更学会了遇到问题应该如何解决。我未来的职业道路应该吸取本次实战中的感悟，在以后的成长中，做一个更优秀的人！

致谢

本次课题系统的实现过程，不仅是对我大学生涯的成长检验，也是自我能力的表达。转眼间，大学已经接近尾声，在步入社会的最后一刻，学校安排导师指导我们做毕设，知道我们在大学生涯里的最后阶段的成长与提升。我很感谢校方对我们大学生的关爱与悉心培养！同时我也很感谢我的指导老师。在指导老师的安排与督促下，我慢慢地实现了系统，同时我也成长为一个能独挡一面你的小青年！

在大学生生涯里，我接触到了很多人，交到了很多好朋友，也见识到了社会的丰富多彩，同时也学到了很多知识与技能！我感谢从我人生中经过的每一个人，因为你们的出现，我的人生剧情才会波澜起伏，才会历久弥新的让我难以忘却！

我从僻远的农村到大城市里学习，初来乍到的我对身边的很多事物都很好奇，我见识到大城市的繁华，但当我步入校园时，仿佛来到了另一个世界，校园里的一切似乎是如此的安静，体育场上是莘莘学子活力的展示场地，我们放肆着青春在校园里四处跑荡。我们终将在四年后离开，但是我们会把热血的青春与活力洋洋洒洒的留在校园。我很感谢学校为我们提供的学习环境，在这片生机盎然的土地上，每一位学子都将迎来改变与成长。我很幸运的在大学中追求成长与完美，我收获到了很多荣誉，同时我也遇到过很多挫折，不过这些都是我人生里的片言风雨，我会继续成长，直到我的天空下也能呵护小树与小草！

再次感谢学校、老师、同学还有校园，谢谢你们陪伴我的成长。今日毕业的我定会回报社会，回报祖国。争取在社会上也能做一个有价值的人！

参考文献

[1]. T. Hey. The Fourth Paradigm: Data-intensive Scientific Discovery[C]//International Symposium on Information Management in a Changing World. Springer, Berlin, Heidelberg, 2012:1-l.

[2]. P. Teli, M. Thomas, K. Chandrasekaran. Big Data Migration between Data Centers in Online Cloud Environment[J]. Procedia Technology, 2016, 24: 1558 -1 565.

[3].王永强. 面向大数据的高性能网络传输关键技术研究[D].西北大学,2017.

[4].预计2021年中国大数据市场规模将达到898亿元[J].电脑知识与技术(经验技巧),2018(10):112-113.

[5].陆泽宁,刘晓洁,黄泽源.基于极值点自适应的快速文件同步方法[J].网络新媒体技术,2018,7(06):17-23.

[6]. Tridgell A. Efficient algorithms for sorting and synchronization[M]. Canberra: Australian National University, 1999.

[7]徐旦.低带宽环境下远程文件同步技术研究[D].北京邮电大学, 2011.

[8]梁丽木, 刘晓洁, 胡晓勤, 等.一种低带宽网络文件同步方法的设计与实现[J].四川大学学报 (自然科学版) , 2011, 48 (1) :55-60.

[9].钱峥, 胡亚旦, 黄旋旋.基于“消息中间件”技术的气象信息总线[J].气象科技, 2016, 44 (2) :217-222.

[10].鲍婷婷，陈鹏，李玉涛. 基于消息中间件技术的分布式气象数据同步系统设计和实现[J].气象科技,2018, 46(6) :1124-1129

[11]徐梓荐,叶盛,张孝.分布式异构数据库数据同步工具[J].软件学报,2019,30(03):684-699.

[12]孟杨. 基于文件分块授权的云存储系统的设计与实现[D].西安电子科技大学,2018.

[13]王敏.Java多线程同步机制下的网络售票系统[J].电子技术与软件工程,2019(04):224.