文本复制检测报告单(全文标明引文)

№:ADBD2018R_2018052518054543665276718 2 检测时间:2018-05-25 18:05:45

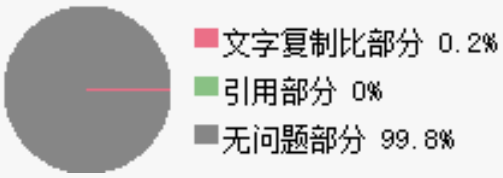
检测文献: 智能物流监控系统的数据平台技术研究
作者: 郑世同
检测范围: 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
大学生论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
CNKI大成编客-原创作品库
个人比对库
时间范围: 1900-01-01至2018-05-25
指导教师 邹阳
审阅意见 通过

可能已提前检测，检测时间：2018/5/24 1:04:59，检测结果：4.6%

检测结果

总文字复制比：0.2% 跨语言检测结果：0%
去除引用文献复制比：0.2% 去除本人已发表文献复制比：0.2%
单篇最大文字复制比：0.2%（CRH5型动车组列车网络控制数据分析管理系统的设计与实现）

重复字数： [60] 总段落数： [9]
总字数： [34551] 疑似段落数： [1]
单篇最大重复字数： [60] 前部重合字数： [0]
疑似段落最大重合字数：[60] 后部重合字数： [60]
疑似段落最小重合字数：[60]



指 标：疑似剽窃观点 疑似剽窃文字表述 疑似自我剽窃 疑似整体剽窃 过度引用

表格：0 公式：0 疑似文字的图片：0 脚注与尾注：0

0%（0）	智能物流监控系统的数据平台技术研究_第1部分（总3262字）
0%（0）	智能物流监控系统的数据平台技术研究_第2部分（总490字）
0%（0）	智能物流监控系统的数据平台技术研究_第3部分（总1642字）
2.2%（60）	智能物流监控系统的数据平台技术研究_第4部分（总2747字）
0%（0）	智能物流监控系统的数据平台技术研究_第5部分（总5782字）
0%（0）	智能物流监控系统的数据平台技术研究_第6部分（总10171字）
0%（0）	智能物流监控系统的数据平台技术研究_第7部分（总1806字）
0%（0）	智能物流监控系统的数据平台技术研究_第8部分（总6861字）
0%（0）	智能物流监控系统的数据平台技术研究_第9部分（总1790字）

（注释：无问题部分 文字复制比部分 引用部分）

1. 智能物流监控系统的数据平台技术研究_第1部分 总字数：3262

学号 142006010119

年级大四

本科毕业设计

智能物流监控系统的数据平台技术研究

专业计算机科学与技术

姓名郑世同

指导教师邹阳

评阅人

2018年5月

中国南京

BACHELOR'S DEGREE THESIS

OF HOHAI UNIVERSITY

Research on Data Platform Technology of Intelligent Logistics Monitoring System

College ：HOHAI University

Subject ：Computer Science

Name ： Zheng Shitong

Directed by ： Zou Yang Professor

NANJING CHINA

学术声明：

郑重声明

本人呈交的毕业设计，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计（论文）的研究成果不包含他人享有著作权的内容。对本设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计（论文）的知识产权归属于培养单位。

本人签名： 日期：

摘要

在万物互联的这个大背景下，将物流行业纳入物联网生态中是一件非常有必要的事情。本系统希冀为完善物联网生态中的物流行业而贡献自己的力量，系统设计的目的是构建一套适合自身物流监控需求的系统，让物流运输相对于所有物流参与者都能成为一个透明的过程。

系统的设计整体分为数据平台对接、数据服务、用户与订单服务部分。其中数据平台对接完成数据的定时更新，保证数据的实时性。数据服务完成了对于数据平台的一些基础数据读取的服务，后期的系统设计将围绕着这些数据完成。用户与订单服务是与用户应用对接的部分，完成前端应用的API提供

智能物流监控系统是一个前后端分离的项目，前端构建web端应用与Android端应用，后端使用springboot构建服务器端应用，开发API端口与前端交互。服务端使用redis技术缓存数据，可以为分布式系统提供session登录服务，使用了spring-data-jpa简化了数据库操作，提供API依照RESTful规范，且使用webSocket协议实现服务端与客户端的双向通信。针对需求完成了包括系统数据查看、订单管理、车辆管理等在内的设计与实现，其中包括每个模块的每个细节，完成了符合系统需求的数据库的设计，并最终完成了用户应用的实现。

关键词：物联网；物流监控；springboot；前后端分离；RESTful；webSocket；OneNET；数据平台

ABSTRACT

Under the background of the Internet of Everything, it is a very necessary thing to integrate the logistics industry into the ecosystem of the Internet of Things. This system hopes to contribute its own power to improve the logistics industry in the Internet of Things. The purpose of system design is to construct a system that is suitable for its own logistics monitoring needs, so that logistics and transportation can become a transparent process for all logistics participants.

The overall design of the system is divided into data platform docking, data services, user and order service parts. The data platform docking completes the regular update of data to ensure the real-time data. The data service has completed some basic data reading services for the data platform, and the later system design will be completed around these data. The user and order service is the part that interfaces with the user application and completes the API provision of the front-end application.

The intelligent logistics monitoring system is a front-end and back-end separated project. The front-end builds web-side applications and Android-side applications. The back-end uses springboot to build server-side applications and develops API

ports to interact with front-ends. The server uses the redis technology to cache data. It can provide session login services for distributed systems. It uses spring-data-jpa to simplify database operations, provides APIs in accordance with the RESTful specification, and uses the webSocket protocol to implement two-way communication between the server and the client. Completed the design and implementation including system data review, order management, and vehicle management for the needs, including each detail of each module, completed the design of the database that meets the system requirements, and finally completed the implementation of the user application.

Key words:Internet of things; logistics monitoring; springboot; front-end separation; RESTful; webSocket; OneNET; data platform

目录

3. 智能物流监控系统的数据平台技术研究_第3部分

总字数：1642

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第1章引言

1.1 项目背景

物联网技术是利用传感器嵌入各种末端设备，通过无线或有线网络接入互联网以实现人与物、物与物实时互联的新兴技术[1]。

智能物流监控系统是在物流量越来越大、物流价值越来越贵重的背景下为了保证运输过程中的安全性以及记录运输中的状态，借助互联网时代的大背景下产生的物联网典型应用之一。

为了适应当前时代的物流需求，我们需要一套可靠的贴合自身需求的物流监控系统，以便让物流的运输成为透明、可控的过程。

依托于物联网技术的智能物流监控系统，需要解决一系列问题：

·实时获取设备的各项数据

·实时数据的实时保存

·数据的结构与相互关联

·不同角色的权限控制

·数据的实时显示

市面上现有的物联网数据平台大部分已经具备支撑某一部分物联网系统的基础需求，且在某些方面有各自的特色，互联网企业例如百度、阿里、京东等目标着眼于智能，包括智能家居、智慧城市等，他们都有一套成熟的部署与服务方式，有自己的app，对于第三方的扩展应用与开发友好度不够。而传统硬件厂商则更多地着眼于硬件的操作与数据平台提供，对于第三方的开发具备更大的灵活性，因此第三方开发者将利用数据平台做出更加贴合自身需求的应用或者系统。

借助于现如今良好的物联网平台的生态环境支撑，本系统将作为物联网的第三方应用，吸收各大物联网厂商平台的资源，完成更加贴合自身在物流上的需求的系统设计。

本系统将专注于物流数据管理软件方面的设计与实现，对于实时获取设备的各项数据与实时数据的实时保存等于硬件交互的方面，已经有比较多的相对成熟的解决方案与产品。我们将选定某一产品作为系统的硬件支撑，系统设计的主要工作是在于设计数据结构与关联、控制不同角色的权限、实时显示物流数据。另外的，作为本系统的数据平台研究，主要任务将集中在数据结构与关联和控制角色权限，对于前端显示将交付于Android端于web端开发。

1.2 国内物联网平台的产业生态

针对自身在物流物联方面的需要，我们需要收集并对比国内各大物联网平台的优势与劣势，选择更贴合自身需求的物联网平台，借助平台对于第三方开发者的支持完成自己的物流监控平台。

目前的物联网产业市场基本上都已形成各自的一套系统解决方案，通常涵盖了物联网的基础功能，有些会根据各自平台的优势致力于特色功能。根据物联网平台解决不同层次的问题以及在不同层次对下层提供服务可以对物联网平台分为三种：基础设施类、开发工具类、运营服务类[2]。

第一类基础设施类，是以AzureIoT、阿里IoT、百度云等借助自身的云计算服务优势搭建的物联网平台[2]。这一类平台提供最基层的环境与服务，支撑硬件数据提取与保存至云存储平台。而实际上这点基层服务无法运行起应用至用户端，所以，基本上这些平台也都有各自的针对物联网的PAAS工具，对第三方开发者提供数据接口，所以与用户对接的应用会有极大的灵活性。有些平台也会有各自的官方应用，针对自家的物联网平台需求，为了完善整个物联网系统生态。

第二类开发工具类，是以QQ物联、远景能源EnOS、GE Predix等为主的物联网平台[2]。这些物联网平台要解决的核心问题是完善整个PaaS层，为了连接下层的IaaS与上层的SaaS，为物联网生态的末端设备与应用的交互提供开发工具。一方面面向硬件提供友好的环境与云计算平台，帮助硬件接入数据至云计算资源。另一方面向应用开发者提供可扩展且巧妙的存储结构以及友好的api，给开发者制定统一的开发标准，帮助物联网生态的最后一步交接至用户的使用上。而实际上，大部分这些物联网平台都不会放弃用户市场，为了抢占用户市场，这些物联网平台一般也都会研发各自的用户应用app作为官方应用，只是缺少了一些开发者自定义的功能，可以满足一些小成本物联网产品的需求。

第三类服务运营类,现有的大部分物联网平台都处于在服务运营类探索的阶段[2]。平台众多，大家都想在物联网发展初期定义一套物联网时代的生活方式，这些生活方式正体现在用户与物联网的交互上面，所以正需要一整套物联网应用，完成用户生活的各方面渗透。只是现在正处于各界小厂商或个人第三方开发者依据PaaS层开发贴合自身需求的应用阶段，夹杂一些物联网平台已经完成的某个领域较完善的SaaS层的服务，譬如小米开放平台正在逐步构建智能家居领域的蓝图、飞凤平台对于智慧城市领域的一系列探索。正缺少对各个领域的整合，想要完善统一整个物联网应用生态需要一个实力足够的组织完成方方面面的应用开发以及应用与自家物联网平台的连接，或者完善一个类似应用商店，对接至自家物联网平台，统一各方应用以供用户使用，但是由于涉及到生活的方方面面，各种复杂场景正待解决，物联网平台仍然处于是一个急需发展以满足需求、构建万物互联的重要阶段。

我们本次设计的物流监控数据平台正是依据第二类开发工具类物联网平台，借助PaaS层连接IaaS与SaaS,完成自身需求的开发，也就是在第三类介绍中处于个人第三方开发者依据PaaS层开发应用。物流监控数据平台是物联网的其中一个典型应用，为了完成整个世界与物联网万物互联的生态，这还只是冰山一角。

我们需要找到一个可以更好的帮助我们构建应用，解决硬件方与软件方交互的PaaS平台。在对比了多家PaaS平台之后，权衡了各家平台生态与我们需求的契合度之后选定了中移动的OneNet平台作为本次系统的PaaS层支撑[3]。

1.3 论文的主要工作和组织结构

本文主要阐述了本次系统设计的具体设计过程，设计中包括需要解决问题的方案分析、需求统一过程、设计与实现的具体实施。本文的组织结构从讨论的问题及其深度的区别方面分为六大章：

第一章：引言部分。引言即是本章，内容介绍了本系统被提出的背景、对当前物联网相关产业的分析以及相关行业生态的对比。

第二章：系统数据概述。先介绍了本系统设计与开发过程中依赖的平台以及框架技术，随后阐述了选定一个物联网平台作为本次系统数据支持的理由，以及对系统开发过程中使用的各种技术进行了介绍，主要描述了技术原理及其使用方法。

第三章：需求分析与概要设计。先对于系统整体解决方案进行了详尽的描述，介绍了系统部署与使用流程，此外详细分析了系统的需求，将系统要完成的任务细化为了一系列功能需求与非功能需求，最后将系统按照功能分为了几个模块，完成了系统的概要设计与数据库设计。

第四章：详细设计与实现。介绍了系统每个部分的详细实现过程，将概要设计方面的成果体现在了实现代码上。

第五章：总结与展望。总结了整个系统设计是否完成了既定的目标、系统设计有哪些不够合理的部分，以及在完成过程中有哪些不足之处，也展望了系统未来的发展前景。

5. 智能物流监控系统的数据平台技术研究_第5部分

总字数：5782

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第2章系统支撑平台与技术概述

2.1 OneNET平台介绍

本系统以OneNET为支撑平台，主要使用OneNET帮助我们解决软硬件数据交互问题，减少硬件工程师与软件工程师的沟通成本。

2.1.1 OneNET平台概述

OneNET平台是中国移动物联网有限公司为了解决物联网生态中硬件与软件连接、原始数据处理、简单的应用等通用且

消耗沟通成本的问题而自主研发的开放云平台[3]。

OneNET面向社会公共服务，着眼于开源的开发者环境，以开放态度以及互利共赢的理念，为各领域跨平台的物联网用户应用以及含物联网需求的各个行业的解决方案提供了更方便的连接与更稳定且灵活的云计算存储资源[3]。帮助物联网开发者（包含企业组织与个人）降低运维和沟通成本、更专注于自身系统与应用的开发，依托于OneNET平台，共同构建统一核心的物联网生态环境。

OneNET平台主要提供了PaaS层一套完整且成熟的技术支持服务，为第三方开发者提供开发工具与帮助解决末端设备连接问题。实际上最重要的，是解决了互联网软件工程师与硬件工程师之间的适配问题。原本要开发一套物联网系统，需要双方约定好协议与数据适配，且需要服务端完成后硬件才能投入测试，而此时双方只需要都按照OneNET约定的标准进行开发即可，可同步开发流程。硬件工程师的开发进度并不影响软件方的进度，从最开始互联网软件工程师就可以按部就班的借助OneNET的数据平台搭建自己的物联网能力服务器（当然也是可以没有自己的物联网能力服务器的，原始的OneNET数据平台与API支持可以满足基础功能开发）。

2.1.2 OneNET平台能力与架构

OneNET平台专注于PaaS层，为IaaS层与SaaS层提供连接的桥梁，分别向上层需求与下层需求提供中间层的整个系统核心能力。体现在架构示意图如下：

图2.1 OneNET架构示意图

2.1.3 OneNET平台产品案例

OneNET平台自身并不参与SaaS层的各个行业的细化应用，将对于SaaS层的开发支持交付于第三方开发者（组织或个人），构建了友好的应用构建环境，这将成为各行业应用百花齐放的沃土。

同时OneNET平台也提供了轻应用快速生成的服务，可以将一些通用的应用模块随意组合，更简化了产品开发的流程，可以满足一些小应用的通用需求。

借助OneNET平台对于SaaS层应用开发的友好服务支撑，产生了各个领域对于物联网系统的比较成熟的用户服务应用，完成了用户与生活很多领域的对接工作。其中包括智慧停车解决方案、共享经济解决方案、城市消防监测解决方案、畜牧物联网解决方案、车联网管理平台解决方案等等领域的成果，已经渗透至用户生活的很多方面。

以车联网管理平台解决方案为例，可以很好地展现物联网企业借助OneNET平台完成定制某一特定领域的工作与设计流程，以及OneNET平台在此类产品中占据的地位与整个系统中发挥的作用。

车联网管理平台使用DTU获取车辆不同部件传感器数据与GPS信息，联网将获取到的数据上传至OneNET平台，OneNET平台以合理的结构将数据存储在云资源，车联网管理系统Web端以及app端会通过使用OnenET平台提供的API接口获取或者操作数据，并最终进行通知、控制、展示等，完成车辆各方面状态以及位置的远程监管的功能点[4]。

已有的相对成熟的产品案例的系统架构与工作流程将对本系统有一个很好地指引作用。后续的物流监控系统将吸取前人成功的经验，参考既有的使用OneNET平台的系统架构，加以改进并融合自身需求，完成物流监控领域的解决方案。

2.2 Spring框架

Spring框架是一个JAVA框架，它是2003年兴起的Java EE轻量级、开源框架，是Java平台为了简化Java EE项目开发复杂度而开发的[5]。

Spring框架对于解决企业应用开发复杂度具有很多方面的优势，其中之一的优势是它的分层架构，分层架构可以指示性的帮助开发者在系统开发的不同阶段为解决不同方面或模块的问题而使用不同的组件，同时整合了现有的一些优秀框架与常用技术，使现有的框架更加实用，为J2EE应用程序开发提供了集成之后的框架[5]。Spring的目的并不是要取代那些现有的框架，而是要与它们无缝地整合[6]。

Spring被应用这么广泛的原因，是它具有紧密联系并巧妙配合的大约二十多个模块，这些模块被分组成Core Container，AOP (Aspect Oriented Programming), Instrumentation, Messaging, Test，Data Access/Integration, Web[5]。这些模块之间的关系即其架构图如下：

图2.2 Spring模块架构图

核心容器是实现框架的关键，其也体现了框架的设计思想。Beans和Core作为框架的基础部分，提供了Spring的核心功能：依赖注入和控制反转 [7]。BeanFactory是工厂模式的一个优秀且复杂的实现，由于它的存在，对于可以编程的单例不再是必须的选项，并且帮助开发者将一些特定的配置（包括一些与其他服务的连接）与依赖从业务逻辑中脱离出来，真正实现解耦[5]。

依赖注入和控制反转是Spring的一个重要特性，它的基本概念是：将创建对象的控制权反转，代码中一些重复出现的创建对象的任务不再出现，而由框架中的配置来自动完成 [7]。框架所做的是帮助减少重复性且可不必借助手动完成的任务工作量，只需要在编码时正确使用注解，便可以省去手动创建的工作。其余工作借助IoC容器，IoC容器可以读取配置信息（事先按照开发者需求完成配置），按照配置中的约定将对象创建完毕后可以自动注入调用者，过程中已经完成了将业务逻辑与服务连接的任务，这些都是耗时且可自动化完成的任务，不应当耗费人力去完成，让开发人员专注于业务逻辑的实现，而不是浪费时间在重复的服务连接上面。

AOP 即面向切面编程（Aspect Oriented Programming），是一种编程技术。在面向切面的思想中，开发者应当对于某一特定问题的解决进行模块化，这样当另外一个地方需要解决同样的问题时，可以引入这个模块，而不是重复书写同样的代码

，而且当对于这个问题的解决方案有改进时可以修改这个模块，只对于这个单一问题的解决有影响而不影响其他任何方面，调用者的使用只受模块的开放接口的结果影响[7]。

2.3 JPA规范与在Spring中的应用

JPA规范是一种关于对象关系映射框架的规范，通常称为ORM框架规范。在面向对象编程的思想下，开发者发现对于数据库的操作应当是在框架层面就被解决的问题，而不是一定需要在代码中嵌入sql语句，而刚好数据库是可以被类比成对象库的，可以编写框架完成代码中的对象与数据库中实体的映射任务，ORM就是解决这样问题的技术。由于开发者更偏向于面向对象的思考，所以将数据库领域映射至面向对象更贴合开发者的习惯[8]。因此，可以用“one to many”描述表间关系，而不是用“row”、“column”、“foreign key”[8]。

JPA规范是一种为统一所有遵守ORM模式的框架而为各框架编写者制定的行为规范[8]。它是一种ORM规范，而不是ORM框架，用于指导ORM框架的实现标准[8]。它制定规范，提供一些API接口，而不是去实现一些API接口，具体的实现则由各个包含ORM框架的服务厂商来完成（譬如JBoss应用服务器底层的JPA实现就是由hibernate完成的）[8]。

这样一来，JPA规范定义并公布了所有需要有的接口，而且这些接口会被所有人（包括使用者与框架开发者）所熟知。对于使用者来说只需要根据文档组织自己的代码、调用自己需要的某些接口，这样很符合面向接口编程的思想，也呼应了对于分层分模块解耦的倡导。而对于框架开发者来说，不需要再自己重新设计一套框架的使用流程，直接按部就班一步步实现JPA规范规定好的一些接口（甚至接口名都不需要自己定义）。它帮助统一了ORM框架的开发与使用方式，如果没有它，对于市面上不同的ORM框架，比如hibernate与TopLink，在一个项目中使用hibernate时需要开发者学习使用hibernate框架的JPA实现，当另一个项目使用TopLink又要重新学习TopLink框架的JPA实现。这样的关系可以用如下图来表示：

图2.3 JPA与ORM框架关系图

Spring中使用的Spring Data JPA 是为了在JPA的基础下，简化对于数据持久存储的访问层（包含查询与修改CURD）而努力。也就是说它在JPA提供的规范之下，提供了Repository层的实现，使得Domain classes的持久性方面的开发变得轻松了很多[8]。

2.4 RESTful API

REST是一种实现应用软件架构时应当遵守的约定和规范，它很好了契合了网络与软件之间的交互，帮助定义一套优秀的符合规范的软件架构。REST是面向资源的规范，它认为一般的应用软件除去一些特定需求与自身逻辑代码，剩下的是对于资源的交流过程，应当解决的是如何方面、快速、且易理解的将资源合理组织布局，它合理利用传输过程中的一些看似多余的数据，且对于URL正确地以资源为主角定义，可以帮助解决一些软件通信方面不必要的麻烦。而RESTful API就是符合REST思想，且遵守一系列REST规范的API服务。

RESTful API遵循的一些设计原则与规范如下：

1、 资源与URI。资源这个概念包含范围比较广泛，我们可以称所有出现在Web应用中可供用户查看与操作的“事物”为资源。资源可以体现在经过持久化的操作之后数据库某个数据表中的某一条数据或者被保存在磁盘的某个文件，也可以是Web服务器端应用接收到某一请求之后经过一系列代码运算操作之后得出的运算结果[9]。资源可以被看做具体的物理方面的实体对象，也可以被看做抽象的流程。

一个资源需要有标记用于标识它，这个标记可以是唯一的也可以是多个。对于Web Service的API，我们很自然的选择URI作为资源的标识[9]。我们认定，一个好的作为资源标识的URI应该具有良好的“可读性”[9]。因为具有良好可读性的URI对于使用者可以对于要访问的内容一目了然。譬如以下这类URI：

http://www.***.com/order/{order_id}（编号为order_id的订单）

http://www.***.com/user/{user_name}（名字是user_name的用户）

http://www.***.com/order/{order_id}/device/{device_id}（在编号为order_id的订单下，设备编号为device_id的设备）

2、 使用统一的接口与标准的HTTP方法。在面向资源的基础概念下，一个Web API是为了实现对某一特定资源的操作的服务。针对某一资源的操作，无非就是CURD操作，而这些CURD操作并不会因资源不同而存在不同。比较合理的体现应该是定义一致的接口来统一这些操作。而对于这些一致的CURD操作，我们在Web服务中又很容易可以把它们与HTTP请求建立联系（或者甚至可以说一一对应）。我们可以直接使用HTTP方法名（GET、POST、PUT、DELETE、PATCH等），用于组合Web API。这样可以使用URI与HTTP方法的配合，结果是URI尽情描述资源而不用管是何操作，由HTTP方法来描述当前API要完成的动作。

3、 无状态性。RESTful API应该像HTTP请求那样，作为无状态请求，它应该只关心资源会发生什么，而不需要关心请求的发出者的状态（即客户端状态）。每一次的请求都应该被视为一个新来的用户发出的，只需要针对本次请求做出相应的操作，不需要识别本次请求是否是第一次请求或者考虑用户后续访问的情况（本次不考虑Session技术，只针对Web API的设计）[9]。

RESTful API是一个优秀的设计规范，它指导API设计者应该怎么设计出更规范合理的API。不是说在做项目时一定要依赖这个规范，只是如果我们引入并严格遵守了这个规范之后，可以帮助我们解决一些不必要的麻烦。至少在本次系统设计中，这个规范可以帮助我们前后端配合开发时更好的理解某一API接口在服务器做了哪些事。所以，本系统的设计将以RESTful API的规范为标准，设计出一套符合规范的API。

2.5 WebSocket协议

WebSocket 协议是在2008年诞生，在2011年才正式成为国家标准，如今所有厂商的浏览器都已经支持了WebScoket协议[10]。

当使用HTTP协议，我们需要客户端知道服务器端的某些实时状态时，以前的做法是在客户端编码实现轮询，十分耗费资源。这就暴露出了HTTP协议的一个缺陷，HTTP协议只能由客户端发送请求再接收响应，通信过程无法由客户端主动发起[10]。因此WebSocket的作用就体现出来了，它也正是在这样的需求下被发明出来的。

WebSocket可以实现客户端与服务端的双向通信，实现消息推送的功能，它在客户端的实现也很方便，比较常用的API如下：

·WebSocket构造函数：用于新建一个WebSocket实例。

·webSocket.readyState：实例对象的当前状态。

·webSocket.onopen：设置回调函数，作为连接成功之后的后续动作。

·webSocket.onclose：设置回调函数，作为连接关闭之后的后续动作。

·webSocket.onmessage：设置回调函数，作为接收到服务器数据之后的后续动作。

·webSocket.send()：设置函数，用于向服务器端主动发送数据。

·webSocket.bufferedAmount：表示当前尚未被发送出去的数据的二进制数据的字节数。

·webSocket.onerror：设置回调函数，作为报错之后的后续动作。

6. 智能物流监控系统的数据平台技术研究_第6部分

总字数：10171

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第3章智能物流监控系统需求分析与概要设计

3.1 智能物流监控系统需求分析

3.1.1 智能物流监控系统需求概述

智能物流监控系统的整体需求[12]是构建系统帮助完成对于特定物品的物流运输流程的实时监控。

针对本系统的设计特点，考虑的需求与分析主要是几个方面问题的解决。

1、对于用户查看的数据方面，需要是合理组织且保证实时性的。

2、用户添加的信息（包括订单、车辆等），需要是可以配合发挥作用的。

3、需要对用户访问的权限进行控制。

以下的各种细化需求分析，都将是为了完成这些既定问题的解决方案而设计的。

3.1.2 智能物流监控系统的功能需求

系统的使用者有三种：箱体公司、货运公司、运输公司。

表3.1 系统角色

角色（参与者） 角色描述

箱体公司数据查看与用户管理

货运公司订单与数据查看

运输公司车辆管理与数据查看

箱体公司相当于系统的管理员角色，它应该有使用系统的最高权限。箱体公司应该具有账号和数据的相关功能，其功能需求的用例图如下：

图3.3 箱体公司用例图

货运公司属于系统的普通用户，只能在自己被允许的范围内使用系统，它需要完成订单的生成与完善、查看权限内的设备数据。货运公司的功能需求的用例图如下：

图3.4 货运公司用例图

运输公司属于系统的普通用户，只能在自己被允许的范围内使用系统，它需要完成运输车辆的添加、查看权限内的设备数据。货运公司的功能需求的用例图如下：

图3.5 运输公司用例图

针对三种用户以上的各个用例，功能点的用例描述如下：

表3.2 账号注册用例描述

用例名称账号注册

行为角色箱体公司、货运公司、运输公司

简要说明箱体公司注册的是管理员账号，货运公司与运输公司注册的是普通账号

前置条件用户已经进入相应的注册页面，且箱体公司账号注册是在拥有APIKey的情况下

后置条件返回结果是“注册成功”并附带添加的数据

表3.3 账号登录用例描述

用例名称账号登录

行为角色箱体公司、货运公司、运输公司

简要说明箱体公司登录管理员账号，货运公司与运输公司登录普通账号

前置条件用户已经进入相应的登录页面，且账号已经过上一步的注册

后置条件返回结果是“登录成功”并附带登录的账号信息

表3.4 账号获取信息用例描述

用例名称账号获取信息

行为角色箱体公司、货运公司、运输公司

简要说明箱体公司获取的是管理员账号信息，且有权限查看普通账号信息。货运公司与运输公司获取的是普通账号信息，无权限查看管理员账号信息

前置条件用户已经登录

后置条件返回结果是相应信息

表3.5 添加普通账号用例描述

用例名称添加普通账号

行为角色箱体公司

简要说明箱体公司拥有后台添加普通账号的权限

前置条件管理员账号已登录

后置条件返回结果是“添加成功”并附带添加的账号信息

表3.6 删除普通账号用例描述

用例名称删除普通账号

行为角色箱体公司

简要说明箱体公司拥有后台删除普通账号的权限

前置条件管理员账号已登录

后置条件返回结果是“删除成功”

表3.7 账查看所有设备用例描述

用例名称查看所有设备

行为角色箱体公司

简要说明箱体公司可以查看所有的设备信息

前置条件箱体公司已登录

后置条件返回结果是所有设备列表

表3.8 查看设备具体信息用例描述

用例名称查看设备具体信息

行为角色箱体公司、货运公司、运输公司

简要说明箱体公司可以查看所有设备的具体信息，货运公司只能查看自己订单下的设备信息，运输公司只能查看自己车辆关联的设备信息

前置条件用户已经登录，且提供设备id

后置条件返回结果是设备信息

表3.9 查看数据流用例描述

用例名称查看数据流

行为角色箱体公司、货运公司、运输公司

简要说明箱体公司可以查看所有设备的数据流，货运公司只能查看自己订单下的数据流，运输公司只能查看自己车辆关联的数据流

前置条件用户已经登录，且提供设备id

后置条件返回结果是数据流信息

表3.10 查看数据点用例描述

用例名称查看数据点

行为角色箱体公司、货运公司、运输公司

简要说明箱体公司可以查看所有设备的数据点，货运公司只能查看自己订单下的数据点，运输公司只能查看自己车辆关联的数据点

前置条件用户已经登录，且提供设备id、时间段与数据流id（可选）

后置条件返回结果是“注册成功”

表3.11 查看车辆信息用例描述

用例名称查看车辆信息
行为角色箱体公司、货运公司、运输公司
简要说明箱体公司可以查看所有车辆的信息，货运公司可以查看自己订单关联的车辆信息，运输公司可以查看自己的所有车辆信息
前置条件用户已经登录，且提供车辆id
后置条件返回结果是相应车辆信息
表3.12 查看所有车辆列表用例描述
用例名称查看所有车辆列表
行为角色箱体公司
简要说明箱体公司可以查看所有车辆的列表
前置条件箱体公司用户已经登录
后置条件返回结果是所有车辆列表
表3.13 查看属于某用户车辆列表用例描述
用例名称查看属于某用户的车辆列表
行为角色箱体公司、运输公司
简要说明箱体公司可以查看属于所有用户下的车辆列表，运输公司只能看到属于自己的车辆列表
前置条件用户已经登录，且提供车辆id
后置条件返回结果是“注册成功”
表3.14 添加车辆用例描述
用例名称添加车辆
行为角色运输公司
简要说明运输公司添加的车辆默认是只属于自己的
前置条件运输公司已经登录，且提供车辆信息
后置条件返回结果是“添加成功”并附带添加的车辆信息
表3.15 查看车辆关联设备列表用例描述
用例名称查看某车辆相关联的设备列表
行为角色箱体公司、运输公司
简要说明箱体公司可以查看所有车辆相关联的设备列表，运输公司只能查看自己的车辆相关联的设备列表
前置条件用户已经登录，且提供车辆id
后置条件返回结果是设备列表数据
表3.16 下订单用例描述
用例名称下订单
行为角色货运公司
简要说明无
前置条件货物公司已经登录，且提供货物信息
后置条件返回结果是“成功”并附带添加的订单信息
表3.17 录入订单数据用例描述
用例名称录入订单数据
行为角色货运公司
简要说明货运公司将订单与提供的箱体和车辆录入，且只能录入自己的订单
前置条件用户已经登录，订单id、设备id、车辆id
后置条件返回结果是录入的订单数据
表3.18 查看订单信息用例描述
用例名称查看订单信息
行为角色箱体公司、货运公司
简要说明箱体公司可以查看订单信息，货运公司只能查看自己的订单信息
前置条件用户已经登录，且提供车辆订单id
后置条件返回结果是订单信息
表3.19 查看订单条目用例描述
用例名称查看订单条目
行为角色箱体公司、货运公司
简要说明箱体公司可以查看所有订单条目，货运公司只能查看自己的订单下的订单条目
前置条件用户已经登录，且提供车辆id

后置条件返回结果是订单条目列表

3.1.3 智能物流监控系统的非功能需求

除了针对用户的每个操作的功能需求，数据平台的系统设计还需要考虑一些非功能需求，保障功能需求方面数据正常的同时，还要保证系统的数据完整性、系统的可扩充性和可维护性、技术适应性与应用适应性。

·系统的数据完整性

1、 需要保证数据平台数据每时每刻都是最新数据。

具体实现为轮询读取OneNET平台的最新数据，更新至自身的数据平台，轮询的间隔应该与硬件上传一次时间点数据的间隔一致。包括设备数据、每个设备的数据流、每个设备的数据点的实时更新。

2、 每次请求都应该有用户识别与权限识别

系统实现为使用session技术，每次用户登录后系统记录session，信息包含用户信息与用户权限。后续此对话的一些操作前，先对权限进行判定。

3、 帮助web端应用解决跨域问题

系统框架添加过滤器，对于每一个请求的响应头都加上相应的属性。

·系统的可扩充性和可维护性

1、 针对需求的变更，系统应当付出最小的成本

编码阶段应当严格将各功能模块化，且依照SpringBoot推荐的各package履行各自职责。

2、 当系统出现bug，应当有最快定位问题的手段

系统配置拦截器，对于每一次的请求都有详尽的记录，且配合系统日志配置，使系统运行阶段的每一个事件都可以被重现，最快定位问题。

3、 OneNET平台数据改变，导致系统正常运行阶段出错的解决。

这个是极个别情况，只是一个防患于未然的考虑。应当采取手段，在读取并更新OneNET数据抛出异常时，及时通知系统开发者，完成系统适配性的更新。

3.2 智能物流监控系统整体方案

智能物流监控系统的目的是为了解决在物流运输的工作中，被运输物体的实时状况无法被实时查看、运输过程无法重现的问题。它最终是为了服务于高端精密设备或物品的物流运输工作中，帮助保障运输物品的安全。

针对第一节的需求分析，为了完成发配物流任务（便于监管物流的整个流程）和物流运输中对于物品的实时监控，即需求分析成果中的各个功能需求与非功能性需求，我们对于智能物流监控系统的设计分为了两大部分：对于获取的系统基础数据平台的搭建，和监管物流流程用于接入物品监控的管理系统的设计。

3.2.1 系统基础数据平台架构设计

对于智能物流监控系统来说，需要考虑很多方面的问题，最首要的问题就是如何建立数据平台与硬件之间的联系[11]。平台需要可以接收到硬件传输的数据，并以合理的形式存储起来，硬件也需要根据约定传输正确格式的数据，所以这就需要硬件方面与平台结构设计事先制定好一套传输规范与传输协议，以保证整个系统正常且高效率运行。这是一个通用的问题，对于每一个涉及软硬件交流的系统设计都将会第一个面临这个问题。所以对于这个问题与其每个系统设计都来完成一套自己的解决方案，不如由某些有足够权威、被开发者普遍认可的个人或者组织制定一套完整且优秀的解决方案，以协议或者服务的形式开放在开源世界。

由于本次系统设计将不涉及与硬件方面传输数据的问题，所以系统将沿用公认的最佳实践“不重复造轮子”，使用开源世界中优秀的解决方案，在OneNET平台的技术支持下完成智能物流监控系统的数据平台架构设计。我们借助OneNET平台，帮助我们解决了硬件与软件交流与数据对接的问题。

OneNET平台开放了PaaS服务，完成了数据格式的设计与良好存储且处于黑盒状态，本系统只与OneNET开放的API接口进行交互。既然使用OneNET平台，就需要遵守OneNET规定的一些设计方法以及数据和操作流程（包括硬件方面的接入流程）的约定。

OneNET平台的数据是分层的结构，对于硬件方的接入也是分层进行的。它包含几个概念：产品、设备、数据流、数据点、产品APIKey、设备APIKey、触发器。

一个用户账号会产生多个产品，在每个产品下含有设备、APIKey、触发器与应用，设备与触发器可以有多个、在设备下可以含有多个数据流。关系图如下：

图3.1 OneNET概念关系图

为了完成OneNET与本系统数据平台的正常对接与后续流程的正常运营，我们需要遵守以下几点规约：

1、 约定统一使用HTTP协议传输数据。OneNET平台方为了兼容各个平台与各厂商硬件，对于硬件接入平台使用的传输协议提供了很多种，建立了一个物联网产业的生态环境。它支持的协议包括：LWM2M（NB-IoT）、EDP、MQTT、HTTP、MODBUS、JT/T808、TCP透传、RGMP等协议。我们需要根据自身系统需求选定合适的协议，用于后续的硬件接入与数据平台获取数据所统一使用的协议。选用HTTP协议的原因是它更简洁，且Web开发者更熟悉HTTP协议，开发可以直接使用，减少了很多不必要的麻烦，降低了学习成本。

2、 数据平台方需要提供APIKey给硬件方。OneNET平台为了识别用户权限，当用户创建产品后会为产品动态创建一个

APIKey作为身份标识，这个APIKey应该由创建者妥善保管，只有有权限使用的人可以拿到APIKey。后续的所有操作的接口请求都需要携带APIKey，用于证明请求者具有查看或者操作权限，OneNET只有判定请求者具有权限之后才会继续进一步动作。所以数据平台开发者拿到APIKey之后需要提供给硬件方使用，这样硬件发出的接入请求才会被认定是本产品的设备。这是一种数据安全的机制，也帮助OneNET平台区分不同产品下的数据。对于数据平台的部署来说，APIKey是必不可少的。

3、 每一台设备拥有的数据流是既定且不需要扩展的。由于硬件的制造成本与回收改造的成本都比较高，对于同一数据平台，我们应该在设计之初就应该约定好我们考虑的设备应该有几条数据流，以及有哪几条数据流。而且后续流程将不会再有任何数据流方面的变更，一旦有需要变更数据流的需求，我们应当考虑重新设计另外一套系统，以降低损失。

针对基于OneNET的系统设计，我们制定了一套包含了最初的硬件部署、数据存储、数据同步、API提供以及运行中数据更新、权限分发等流程在内的整体数据平台的架构，用于完善以及规范整个系统在管理者手动操作与系统自动完成任务之间的配合，指导出一套符合标准的开发流程。

·硬件方面

拥有OneNET平台账号的人，我们可以称之为管理员。在系统建立之初，管理员需要在平台完成系统需要的一些数据的部署，这些数据将是我们后期系统功能点的设计所围绕的核心。而这些数据的部署完全按照OneNET平台给定的流程，我们省去了自己设计这些部署操作和数据格式的工作。

首先管理员需要建立一个产品，这个产品将作为我们整个系统的代表，每部署一个这样的系统就应当对应一个产品（当然产品可以是很多个且多种多样的，只是由于我们只考虑自己需要的这一个系统，所以我们并不需要额外的产品，除非需求变更，需要另外一个系统，我们将建立另一个产品，对应一个重新设计的系统）。创建产品后，OneNET会为整个产品分配一些数据，我们后期开发需要的数据包括产品APIKey和正式环境注册码。产品作为一个容器的角色，被建立完成后就可以开始后续的往容器内添加数据自动操作了。以下的动作都将是被写入硬件的程序自动完成的。

硬件在投入使用之前需要先为自己在OneNET平台自动注册设备，这个操作需要使用上一步创建产品之后得到的APIKey和正式环境注册码。注册设备完成后，设备便在OneNET平台被记录且被分配了一个唯一的device_id（当然也会分配设备APIkey，是OneNET用于区分不同设备与阻断不同设备间访问的，对于某一设备的操作不应当影响其他设备。只是本数据平台只使用总的APIKey，关于权限有自己的一系列实现，所以并不需要这个设备APIKey），后续关于这个设备数据的操作都将在这个device_id下进行。

随后是添加数据流，硬件在开始传输某些数据前，需要在OneNET平台为这些数据提前添加数据流。设备会带着前面拿到的APIKey和device_id请求添加数据流接口，本数据平台约定只会有固定的五条数据流：温度（temperature）、湿度（humidity）、压力（pressure）、倾角（obliquity）和位置（position）。所以设备会连续添加五条数据流，且从系统被部署的那一刻开始便不会再改变。

最终便可以由设备自动运行，将设备投入系统使用。设备会每隔一段时间（可以是几秒），获取当前数据并上报至OneNET平台，实现对运输物品的实时监控。

·平台数据方面

对于数据平台来说，平台需要保证数据每时每刻都是当前的最新数据，所以这方面的开发主要是借助OneNET平台提供的API获取实时数据，并同步到平台自己的数据库中，这样可以使数据平台自身向前端提供的API可以获取到最新的数据。

在数据平台将会保存来自OneNET平台的设备、数据流、数据点的数据，所以就像硬件上传数据的流程一样，此部分设计需要一步步完成从设备到数据流再到数据点的更新。

此部分将要完成：

1、 通过产品APIKey获取到所有下属的设备device_id

2、 根据device_id获取对应的设备信息，并实时更新本地数据库

3、 根据device_id获取设备对应的数据流（应当被舍弃，因为前面已经提到数据流不会改变）

4、 根据device_id与datastream_id获取对应的各个时间的数据点，记录下（或者是更新）数据点。

这样确保前端显示以及后续系统使用的数据的都会是最新的，不用考虑数据延时性的问题。

·应用开发API方面

系统数据平台最终是服务于用户应用的，它将提供API接口给前端应用开发使用。而对于这些API，需要考虑以下几点：

1、 用户权限。

需要区分管理员（即箱体公司）与普通用户（包含货物公司与运输公司）。管理员需要具有最高权限，可以查看所有数据。普通用户只能查看相应订单或者相应车辆相关联的设备数据。

2、 数据操作

需要提供包括查看设备、查看数据流、查看数据点在内的API。

3、 触发器操作

OneNET平台提供触发器功能，可以在设备达到某个预警值时访问一个地址发送请求，数据平台需要完成自动告警的功能。

总的来说，系统中的硬件操作与OneNET交互产生数据，系统数据平台从OneNET获取数据后向外网提供API服务，以支撑起用户应用。以下是系统架构图：

图3.2 系统架构图

3.2.2 应用于物流的管理系统设计

为了接管整个运输流程，便于运输过程中的监控，我们以数据平台的实时设备数据为基础设计了一套物流流程管理以及与数据平台对接的解决方案。

解决方案需要完成下订单、订单录入、角色分管、分订单查询等功能。具体方案将包含为以下几点：

1、 货运公司会根据要运送的货物数量下订单

2、 运输公司有添加车辆信息的功能，并在有新订单生成时派遣车辆

3、 箱体公司会根据订单的货物数量提供箱体

4、 货运公司在箱体与车辆到达时将箱体信息与车辆信息录入订单记录

5、 货运公司可以查看到自己下的订单下的箱体信息

6、 运输公司可以查看到自己的车辆运输的箱体信息

7、 箱体公司可以查看到以上所有箱体信息

其中隐含了一些用户权限分配的功能，后续的一些物流管理系统方面的需求将从这些既定的方案中扩展功能及细节。数据平台的数据库将按照这些方案在数据表的设计上建立关联，这样就完成了运输流程与数据平台的系统对接。

3.3 智能物流监控系统概要设计

3.3.1整体系统框架结构

系统整体上各功能框架分为三个部分：OneNET平台对接、数据服务、用户与订单服务。

其中OneNET平台对接的设计任务是整个系统的难点和重点，是所有后续功能设计的基础。基于此，平台具有向外界提供数据的能力，完成数据提供设计是为了后续用户使用功能做铺垫。最终的用户与订单服务是整个系统的最终目的，这个阶段的设计可以完成用户账号与权限设计，订单服务完成对物流运输流程的监控。其结构图如下：

图3.6 系统框架结构图

3.3.2系统模块设计

系统根据分管功能、操作的数据、面向的用户等方面，对系统整体功能框架进行了模块划分。整个系统的设计将包括六大模块：平台数据更新、基础数据访问、用户账号与权限控制、订单管理、车辆管理、告警。

系统设计将围绕着这六个大模块展开的，最终扩展到每一个功能点上。

系统各模块介绍如下：

·平台数据更新

模块功能描述：数据平台需要根据自身需求与OneNET平台的数据格式设计自己的数据库，而建立好数据库之后，是没有数据的，需要从OneNET平台动态获取并时刻更新。

接口设计：

输入产品APIKey

输出从OneNET读取的产品下所有设备

输入设备信息

输出数据库设备被更新

输入设备id

输出从OneNET读取的设备下所有数据流

输入数据流信息

输出数据库数据流被更新

输入设备id与数据流id、时间段

输出从OneNET读取的设备数据点信息

输入数据点信息

输出数据库数据点被更新

从获取数据可以更新设备数据库和获取数据流信息，后可以更新数据流数据库和获取数据点信息，连带可以更新数据点数据库，其功能流程图如下：

图3.7 平台数据更新流程图

·基础数据访问

模块功能描述：系统最终被用户使用的就是这些展示硬件数据的功能，上一步模块以及保证数据平台的数据都是实时的，本模块负责将这些用户需要的数据开放给用户端。

接口设计：

输入设备id

输出产品下所有设备信息

输入设备id

输出设备数据流信息

输入设备id、数据流id、时间段
输出设备下时间段内的相应数据点
功能流程图：
图3.8 基础数据访问功能流程图
·用户账号与权限控制
模块功能描述：使用系统的用户分为三种，箱体公司、货运公司与运输公司。不同的用户会有自己不同的使用方式与职责，所以用户使用前需要有自己的账号，过程中会根据不同的用户分配不同的权限，不同用户之间的功能不互通。
接口设计：
输入箱体公司账号用户名、密码、产品APIKey
输出数据库新增一条箱体公司账号数据
输入货运公司或者运输公司账号用户名、密码、type
输出数据库新增一条普通账号数据
输入箱体公司账号用户名、密码
输出系统session添加账号登录记录
输入货运公司或者运输公司账号用户名、密码
输出系统session添加账号登录记录
功能流程图：
图3.9 用户账号与权限控制功能流程图
·订单管理
模块功能描述：最终投入物流监控使用，需要在物流开始时将流程就纳入系统管控范围内。物流开始时需要先下订单，根据订单分配箱体，这一部分是由货运公司完成的，运输中可以根据订单查看信息。
接口设计：
输入货物信息、用户信息
输出数据库新增一条订单
输入订单id
输出订单信息
输入订单id、设备id、车辆id
输出数据库新增一条订单条目
输入订单id
输出订单所有条目信息
功能流程图：
图3.10 订单管理功能流程图
·车辆管理
模块功能描述：运输使用的车辆也需要纳入系统管控。物流的流程中需要录入车辆的信息，这一部分是由运输公司完成的，运输过程中也可以根据车辆查看设备信息。
接口设计：
输入车牌号、驾驶员、手机号
输出数据库新增一条车辆数据
输入用户id
输出整个用户的所有车辆列表
输入车辆id
输出车辆信息
输入车辆id
输出对应的运输的设备信息
功能流程图：
图3.11 车辆管理功能流程图
·告警
模块功能描述：在某一数据达到某个预警值时，系统应该可以对用户发出警告信息，此部分使用OneNET平台提供的功能，系统只接收信号并分发至用户。
接口设计：
输入设备id、数据流id、数据点
输出是否应该告警
输入设备id、数据流

输出向相关的箱体公司账号、货运公司账号与运输公司账号发出警告

功能流程图：

图3.12 告警功能流程图

3.3.3 数据库设计

数据库实体关系图：

图3.13 数据库实体关系图

智能物流监控系统设计的数据库主要包括以下九张表：product表、device表、dataStream表、dataStreamPoint表、order表、orderItem表、car表、user表、manager表。

1、 product表实际上只是为迎合OneNET平台的概念预留的一张表，保存了本系统在OneNET平台的APIKey，在系统运行中不起实际作用。

2、 device表存储了设备信息，包含描述、唯一标识、协议与是否在线等信息。

3、 dataStream表存储了所有设备拥有的数据流信息，外键指向device的id字段，包含单位、当前值、标签等信息。

4、 dataSreamPoint表存储了所有设备的数据点，字段包括设备的六个数据流，以时间和设备id作为主键，每一条数据可以记录当前时间点某一设备的所有数据。

5、 manager表记录管理员账号信息，即箱体公司使用的账号，用于用户权限模块的设计与实现。

6、 user表记录普通用户账号信息，即货运公司与运输公司使用的账号，用户用户权限模块的设计与实现。

7、 car表记录车辆信息，包含司机姓名、司机手机号、车牌号等信息，用于后续记录订单条目时关联使用数据。

8、 order表记录订单信息，包含货物信息、下单者、是否完成等字段，下单者外键关联至用户id，用于订单系统源数据。

9、 orderItem表记录订单条目，包含货物编号、设备编号、车辆编号、订单编号等，车辆编号外键关联至车辆id，订单编号外键关联至订单id。

7. 智能物流监控系统的数据平台技术研究_第7部分

总字数：1806

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第4章智能物流监控系统的详细设计

4.1 平台数据更新模块

系统需要完成定时任务的配置，定时读取OneNET平台的数据，并更新至自身数据库。

模块第一步需要访问OneNET平台API，其中包括先获取设备信息，然后根据设备id获取数据流信息，再根据设备id与数据流id获取数据点信息。需要平台数据分析与转换，即拿到数据之后将数据转换成系统需要的数据格式以供系统使用。本地数据库访问与更新本地数据即对于repository与service的接口调用。定时循环执行任务要完成框架定时任务的配置，使用@schedule注解完成任务定时执行。

过程涉及数据库操作，需要实际操作类与底层service配合。UpdateDB作为实体操作类，主要涉及与Device、DataStream、DataStreamPoint三个实体类及其操作类的交互。用类设计图[13] 表示如下：

图4.1 平台数据更新类图

4.2 基础数据访问模块

这一部分完成基础数据的访问，过程涉及数据库的读取操作，需要与底层service配合。

模块需要本地数据库访问，即对于repository与service的接口调用，包括设备、数据流与数据点的数据获取，获取获取方式有所有数据以及条件获取，然后以合理的RESTful风格API的形式开放给应用。

DeviceController、DataStreamController、DataStreamPointController作为对外接口提供，主要涉及与Device、DataStream、DataStreamPoint三个实体类的操作类的交互。其类设计图如下：

图4.2 基础数据访问类图

4.3 用户账号与权限控制模块

这一部分贴合用户使用应用方面，需要对用户分配账号、识别账号并设定权限。

需要依赖数据库的用户数据表，数据表记录用户数据，在提供的接口中包含CURD的操作。将包含三种用户的分别，其中一种是箱体公司用户具有管理员权限，用manager表来标识，另外两种货运公司与运输公司是普通用户，用user表标识，type字段用于标识用户类型。

过程涉及数据库操作，UserController与ManagerController作为对外接口提供，主要涉及与User、Manager两个实体类的操作类的交互。

类设计图如下：

图4.3 用户账号与权限控制类图

4.4 订单管理模块

这一部分完成用户应用中的订单管理，需要提供给用户下订单、查订单、订单录入等的功能。

模块第一步货运公司用户登录后可以使用下订单功能，下订单会在订单表新增一条订单数据，这条数据可以被货运公司用户与箱体公司用户查看到。后续针对这些订单数据，箱体公司会派遣箱体，运输公司会派遣车辆，货运公司需要将货物与箱体和车辆录入系统，数据库会新增订单条目。

过程涉及数据库操作，主要是OrderController提供对外接口，与Order实体类的操作类的交互。

类设计图如下：

图4.4 订单管理类图

4.5 车辆管理模块

这一部分完成用户应用中的车辆管理，需要提供给用户添加车辆、查看车辆信息等的功能。

模块先要有运输公司用户添加车辆的功能，车辆数据库会新增一条车辆数据，后续这些数据可以由运输公司用户与箱体公司用户查看，可以是查看所有或者是条件查询，同样需要repository与service的数据读取的接口。

过程涉及数据库操作，主要是CarController提供对外接口，与Car实体类的操作类的交互。

类设计图如下：

图4.5 车辆管理类图

4.6 告警模块

本模块主要是接收告警信息，并以webSocket技术将告警信息推送给相应的人。

模块需要完成webSocket的基础配置工作，使得前端代码可以通过某个地址与服务端建立连接。系统需要开放一个API用于接收告警的通知，接收到通知后，要根据信息中的设备id在数据库中取到相关应该推送的用户，再调用向客户端发送数据的webSocket接口，完成向客户端推送消息的任务。

过程涉及到数据库操作，getWarnningController用于接收告警信号，webSocket用于推送数据的接口设计。主要包括orderItem、order、car三个表的数据库操作。其类设计图如下：

图4.6 告警类图

8. 智能物流监控系统的数据平台技术研究_第8部分

总字数：6861

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第5章智能物流监控系统的实现

系统使用springboot推荐代码组织方式，不同的包内代码完成各自的职责。Domain包用以jpa映射对象与实体，repository包作为数据库操作类继承自JpaRepository，service提供对数据一些操作接口，controller接收请求并对外提供接口。除此之外，项目包含config包存放框架配置文件与各个依赖库的配置，bean包含一些项目需要的特殊数据格式（而不是数据库实体的数据格式），schedule为项目自动运行任务的代码，filter与interceptor用于配置过滤器与拦截器，完成跨域解决、记录每个请求的log等功能。代码组织如下图：

图5.1 代码组织结构图

5.1 平台数据更新模块

此部分模块涉及三个数据库表的操作，对象关系映射在代码中映射为Device类、DataStream类、DataStreamPoint类。

实体类的设计如下：

Device类数据表设计：

表5.1 device表

字段字段类型字段说明

Id Long 设备id

tags String 标签（附带信息）

online Boolean 在线（附带信息）

protocal String 协议（附带信息）

title String 标识（附带信息）

description String 描述（附带信息）

authInfo String 设备唯一标识

createTime String 创建时间

lastmodifiedTime String 修改时间

product_id Long 外键关联至相应产品

DataStream类数据表设计：

表5.2 dataStream表

字段字段类型字段说明

Id Long 数据流id

tags String 标签（附带信息）

unit String 单位（附带信息）

unit_symbol String 单位符号（附带信息）

current_value String 当前值

uuid String 设备唯一标识

createTime String 创建时间

lastmodifiedTime String 修改时间

device_id Long 外键关联至相应设备

DataStreamPoint类数据表设计：

表5.3 dataStreamPoint表

字段字段类型字段说明

Device_id Long 外键至设备id

at String 时间

obliquity String 倾角数据

temperature String 温度数据

humidity String 湿度数据

location String 位置数据

position String 位置数据

pressure String 压力数据

根据OneNET平台提供的API，需要先访问一次性获取所有设备的接口，然后根据OneNET的API文档解析返回数据并传输给更新的方法。代码如下：

@Scheduled(fixedRate = 5000)//每隔五秒执行一次public void updateService(){HashMap<String, String> map = new HashMap<>();RestTemplate restTemplate = new RestTemplate();HttpHeaders requestHeaders = new HttpHeaders();requestHeaders.add("api-key", productService.getAPIKey);HttpEntity<String> requestEntity = new HttpEntity<String>(null, requestHeaders);ResponseEntity<Object> responseEntity = restTemplate.exchange("http://api.heclouds.com/devices",HttpMethod.GET,requestEntity, Object.class, map);HashMap<String, Object> onenetResponse = (HashMap<String, Object>) responseEntity.getBody();HashMap<String, Object> data = (HashMap<String, Object>) onenetResponse.get("data");ArrayList<HashMap<String, Object>> devices = (ArrayList<HashMap<String, Object>>) data.get("devices");for (HashMap<String, Object> _device:devices){updateDevice(_device);}}

图5.2 访问API解析数据的代码

由于有可能数据库内并没有某些设备信息，所以有可能是添加或者更新，Repository包里的数据库操作类继承JpaRepository使用JPA提供的接口saveAndFlush()完成这个功能。

在获取并解析数据完成之后，只需要调用saveAndFlush。在更新每个设备的同时，拿到device_id调用更新DataStream，以此类推更新DataStream时，拿到dataStream_id调用更新DataStreamPoint。执行完成后就完成了这一轮的更新数据，暂定硬件每五秒上传一次数据，所以程序设定每五秒执行一次。

5.2 基础数据访问模块

数据库即使用上一个模块设计的数据库。

代码需要完成的任务比较单一，只需要读取到请求的数据，获取相应的数据并返回即可。

根据id获取设备信息的接口作为例子，代码如下：

ResponseBody responseBody = new ResponseBody();Device device = deviceService.getById(id);responseBody.setSuccess(true);responseBody.setData(device);return responseBody;

图5.3 获取设备信息的代码

5.3 用户账号与权限控制模块

本模块主要涉及两个数据库表的操作，对象关系映射在代码中映射为Manager类、User类。

实体类数据表设计如下：

Manager类数据表设计：

表5.4 manager表

字段字段类型字段说明

Id Long 用户id

name String 用户名
password String 用户密码
createTime String 创建时间
lastmodifiedTime String 修改时间
product_id Long 外键关联至相应产品
User类数据表设计：
表5.5 user表
字段字段类型字段说明
Id Long 用户id
name String 用户名
password String 用户密码
createTime String 创建时间
lastmodifiedTime String 修改时间
manager_id Long 外键关联至被添加的manager（可为空）
type String 用户类型
首先，需要完成用户注册登录的实现，这部分涉及用户的识别，服务端对用户登录状态的保存等。以登录为例，登录操作使用session技术用于服务端保存登录状态与标识当前会话。
验证用户与保存session的代码如下：
ResponseBody responseBody = new ResponseBody();if (request.getSession().getAttribute("managerName") != null) {//服务端存在登录信息...//跳过登录环节，可省略}...//一些参数判断操作，可省略if (managerService.authenticateManager(managerName,password)) {session.setAttribute("managerId", manager.getId());session.setAttribute("managerName", manager.getName());responseBody.setData(manager);responseBody.setSuccess(true);responseBody.setMessage("登录成功");return responseBody;} else {...//错误情况考虑，可省略}
图5.4 登录过程代码
普通用户的登录过程可类比上部分代码。
另外，权限模块需要提供验证接口，用以判断当前请求是否已登录，以及当前用户的权限等级。
实现代码如下：
public int currentUserAuthority(HttpServletRequest request) {HttpSession session = request.getSession(false);if(session.getAttribute("managerId") != null){return 1;}else if(session.getAttribute("userId") != null){return (int) session.getAttribute("userType");}else {return 0;}}
图5.5 验证是否登录的代码
后续的系统开发中，当需要对某个接口判定是否可以访问时，就可以调用这个接口。
5.4 订单管理模块
本模块主要涉及一个数据库表的操作，对象关系映射在代码中映射为Order类。
实体类数据表设计如下：
Order类数据表设计：
表5.6 order表
字段字段类型字段说明
Id Long 订单id
goodsName String 货物名
goodsType String 货物类型
goodsNumber String 货物数量
complete String 是否已完成
user_id Long 外键关联至相应用户
OrderItem类数据表设计：
表5.7 orderItem表
字段字段类型字段说明
goodsNumbering String 货物编号
deviceAuthInfo String 设备标识
carId Long 车辆id
orderId Long 外键关联至相应订单
在controller内主要完成的任务将是对于订单的操作，包括下订单、完善订单条目、查看订单等。以添加订单为例，添加

时需要知道这条订单是哪个用户添加的，最可靠的方式是借用用户权限控制，利用session读取当前已登录的userId。

实现代码如下：

String goodsName = addOrder.getGoodsName();String goodsType = addOrder.getGoodsType();int goodsNumber = addOrder.getGoodsNumber();Long userId = null;try {userId = (Long) session.getAttribute("userId");}catch (Exception e){log.error("从session获取userId时出错" + e.toString());}if(userId == null){responseBody.setMessage("请先登录（货运公司）");return responseBody;}

图5.6 从session读取当前用户的代码

5.5 车辆管理模块

本模块主要涉及一个数据库表的操作，对象关系映射在代码中映射为Car类。

实体类数据表设计如下：

表5.8 car表

字段字段类型字段说明

Id Long 车辆id

driverName String 司机姓名

phone String 司机手机号

carNumber String 车牌号

user_id Long 外键关联至相应用户

在controller内主要完成的任务将是对于车辆的操作，包括添加车辆、修改车辆、查看车辆信息等。以添加车辆为例，添加时需要知道这条订单是哪个用户添加的，最可靠的方式是借用用户权限控制，利用session读取当前已登录的userId。

具体实现代码可以参照订单管理模块。

5.6 告警模块

本模块主要涉及告警信息接收、对指定用户推送的判断、webScocket的配置、推送消息。getWarnningController接收OneNET平台的告警请求，接收到请求后解析数据，从数据库操作类取到应该推送的用户，依据webSocket的配置，推送数据至用户。

从数据库操作类取到应该推送的用户时，分为三种用户的判定。对于箱体用户来说，因为是管理员的角色，所以所有的箱体用户都应该推送。对于货运公司来说，应该推送给包含这个设备的订单的下单者。对于运输公司来说，应该推送给运输着这个设备的车辆的拥有者。

代码如下：

WarnningUser warnningUser;OrderItem orderItem = OrderItemService.getByDeviceId(device_id);Long order_id = orderItem.getOrder_id();Order order = OrderService.getByOrderId(order_id);Long user_id = order.getUserId();User user = UserService.getOne(user_id);warnningUser.setHuoUser(user);//设置接收推送的货运公司用户String carNumber = orderItem.getCarNumber();Car car = CarService.getOne(carNumber);Long user_id1 = car.getUserId();User user1 = UserService.getOne(user_id1);warnningUser.setYun(user1);//设置接收推送的运输公司用户

图5.7 设置推送用户的代码

随后需要将包含要推送用户的数据推送给用户，使用webSocket的配置代码如下：

public class WebSocket { private Session session; private static CopyOnWriteArraySet<WebSocket> webSockets = new CopyOnWriteArraySet<>(); private MessageVO messageVO = new MessageVO(); @OnOpen public void onOpen(Session session) { //新连接时的动作 } @OnClose public void onClose() { //连接关闭时的动作 } @OnMessage public void onMessage(String message) { //客户端向服务端发送数据时的动作 } public void sendMessage(Message message) { for (WebSocket webSocket : webSockets) { try { webSocket.session.getBasicRemote().sendText(message); } catch (Exception e) { e.printStackTrace(); } } }}

图5.8 webSocket配置代码

随后客户端接收告警信息时，将告警信息正确的提示给用户即可。

9. 智能物流监控系统的数据平台技术研究_第9部分

总字数：1790

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第6章总结与展望

6.1 总结

本次系统设计项目组完成了智能物流监控系统的从数据对接到数据提供，最终至用户使用终端应用的所有工作。服务端数据平台方面完成了与OneNET数据平台的对接、基础数据提供、用户应用API支持等工作，web端与Android端应用完成了系

统提供给用户实际使用的界面设计以及对服务端服务的操作支持。

系统的设计与实现、前后端对接完成后，对于系统服务于高端精密设备或物品的物流运输工作中，帮助保障运输物品的安全这个整体目标已经基本满足。除此之外，在具体的需求方面，系统根据需求文档完成了功能需求的各个功能点的实现，同时在编码过程中针对非功能需求分别给出解决方案，保证了系统稳定且高效运行。

我在项目中完成了服务端数据平台的整体设计与编码实现。在整个项目设计与实现阶段，我主要完成的工作包括以下几点：

1、数据支持平台研究。对于市面上的多个数据支持平台进行了研究与对比，选出了最符合本系统需求的平台作为本系统的数据支持。同时仔细研究了这个平台的数据服务方式，为本系统使用平台奠定基础。

2、系统整体解决方案设计。一方面需要依据OneNET平台使用规约，另一方面需要决定系统设计的整体架构，我们约定了一套流程，用于规范整个系统从编码到部署正常投入使用的流程。

3、需求分析。根据系统需要完成的目标任务，将系统设计与实现阶段待解决的问题转化为各个功能点以及非功能的要求，为后续开发过程提供参考。

4、概要设计。根据系统既定的需求，设计数据库存储必要的数据。设计代码组织情况、各个包的调用关系、各个类完成的任务以及类与类之间的调用或继承关系等。

5、详细设计。具体编码实现，代码合理组织，在保证代码健壮性的同时完成既定的功能要求。

6.2 展望

智能物流监控系统是物联网技术的一个典型应用，它在物联网行业还未形成一个成熟的产业生态。同时物联网技术还在日新月异的发展着，虽然距离万物互联还有很长一段路要走，目前的物联网世界还需要各行各业的共同发展努力。

目前完成的本次智能物流监控系统是当前技术下比较普遍的一次简单实践，它面对的群体是某一个物流流程中的参与者，并不能完成一整个行业的解决方案。后期这个系统能改进的地方是应该能够成为一个这个物流监控行业通用的解决方案，可以支撑所有物流行业的用户，而不再是只有一家箱体公司和围绕着它的货运公司与运输公司，应该可以有不同的箱体提供商可以入驻系统，为所有围绕这些箱体提供商的物流服务商提供物流监控服务。这将是这个行业的一个通用的物联网解决方案，为物联网万物互联生态的构建增添一份自己的力量。

关于我完成的服务端数据平台，还有很多地方值得改进。

在数据平台定时更新模块，需要保证每时每刻数据库存储的都是最新数据，所以使用定时任务来轮询OneNET的API接口。而定时任务是集成在了项目内，也就是说项目是跟定时任务共存的，定时任务又是一个很耗费资源的过程，而且当定时任务部分代码更新时要连整个项目一起重新发包更新，这是一个不太合理的设定。后续的改进可以是将定时任务作为分布式任务，部署在不同的资源下，不影响当前服务的运行，也不受其影响。

告警功能做的还不够合理，当前是把需要通知的用户包含在了广播消息内，然后由前端判断是否应该让用户判断，应该是可以只让需要推送的用户接收到消息。

关于用户应用API的提供，还有很多关乎用户体验的应用扩展，比如分页功能、手机号验证、邮件通知等功能。

系统也保留了所有运输过程中设备状态的改变过程，存储在数据库中在以后的数据研究中可以用于大数据分析，帮助研究物流行业的其他方面的解决方案。

参考文献

[1] 胡永利, 孙艳丰, 尹宝才， 物联网信息感知与交互技术， 计算机学报, 2012, 35(6): 1147-1163.
[2] 物联网时代的巨头在哪里：3类平台，5种路径，
http://www.mixlink.org/article/7212.html.
[3] 中移物联网开放平台，https://open.iot.10086.cn/.
[4] OneNET车联网管理平台解决方案，
https://open.iot.10086.cn/case/detail72.html.
[5] 叶加青，Spring框架技术的应用，计算机时代，2009.
[6] Spring学习笔记--Spring简介，
https://www.cnblogs.com/ysw-go/p/5986695.html.
[7] 任振宇，汪成曦. 基于SSH框架的企业信息管理系统设计研究，科技资讯，2012.
[8] SpringData Jpa、Hibernate、Jpa 三者之间的关系，
https://blog.csdn.net/u014421556/article/details/52635000
[9] 我所理解的RESTful Web API，
https://www.cnblogs.com/artech/p/3506553.html
[10] WebSocket 教程，
http://www.ruanyifeng.com/blog/2017/05/websocket.html
[11] 陈海明, 崔丽, 谢开斌. 物联网体系结构与实现方法的比较研究, 计算机学报, 2013, 36(1): 168-187.
[12] 丁二玉，骆斌，需求工程——软件建模与分析，高等教育出版社，2008
[13] 王博文，盛枫，窦亮，杨宗源，UML类图的形式规约与精化研究，计算机应用与软件，2017

致谢

感谢邹阳导师的悉心指导，邹老师每周定期开会，了解项目的进展，给出指导性的意见，并为我们制定下一阶段的计划。从最初的需求场景设定、技术选型、到最后项目验收都给予了我们很大的帮助。在邹老师的指导下，我们少走了很多弯路，最终项目与论文得以顺利完成。邹老师深厚的知识水平，严谨的治学理念和切实的理论指导给了我很大的帮助。

还要感谢项目组的同学们，大家在这段时间里全身心投入到了项目设计与开发工作中，互帮互助，一起细致讨论分析系统的各个功能点的设计与实现。在大家的努力下，项目最终顺利开发完成。

最后还要感谢我的家人和朋友，在他们的鼓励和陪伴下，项目与论文得以顺利完成。