**内部**

封面页边距要求：上、下为3.8cm,左右为3.2cm,页眉页脚为3.0cm

密级：仿宋14磅，距顶边4cm

统一印刷，校名题字36号，居中，距顶边5.2cm

中国科学技术大学

统一印刷，黑48号，居中，距顶边8.5cm

工程硕士学位论文

统一印刷



黑体26磅加粗居中，单倍行距，距顶边15.8cm

**基于Spring和微服务架构的寄件快递单处理系统的设计与实现**

此列宋体16磅居中上下对齐，单倍行距，距上边界20.5cm

此列居右，黑体16磅，距左边界6cm，统一印刷

|  |  |
| --- | --- |
| 作者姓名： | 张松 |
| 工程领域： | SA14225164 |
| 校内导师： | 薛美盛 |
| 企业导师： | 郑耸 |
| 完成时间： | 二○一六年八月十日  距下边界5cm |

University of Science and Technology of China

**Limited**

Arial 20磅居中，行距30，距顶边5cm

密级：Times New Roman 14磅，距顶边4cm

A dissertation for master’s degree

of engineering

Arial 26磅居中，行距30，距顶边6cm

校标尺寸(cm)

4.2×4.2，居中，距顶边8.2cm



Arial 26磅加粗居中，行距30磅，距顶边16cm，根据字数需要可以适当上下调整文本框位置

**Design and Implementation of Processing System for Sending Order**

**Based on Spring and**

**Microservice Architecture**

|  |  |
| --- | --- |
| Author’s Name： | Song Zhang |
| Speciality： | Software Engineering |
| Supervisor：  Times New Roman 16磅，行距30磅，此列居右，距左边界6cm | Prof. Meisheng Xue  Times New Roman 16磅，居中上下对齐，行距30磅 |
| Advisor： | Dr. Song Zheng |
| Finished time: | August 10th, 2016  距下边界5cm |

书脊

书脊页边距要求：上、下为3.8cm,左右为3.2cm,页眉页脚为3.0cm

|  |
| --- |
| 3cm左右  **基于Spring和**  黑体12磅，行距14磅  **微服务**  **架构**  **的**  **寄件**  **快递单**  **处理**  **系统**  **的**  **设计**  **与**  **实现**  **张松**  **中国**  3cm左右  **科学**  **技术 大学** |

页边距要求：上、下为2.54cm,左右为3.17cm

此页可下载

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关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，可以将学位论文编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开 □保密（\_\_\_\_年）

作者签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**摘 要**

宋体12磅，行距20磅，段前段后0

二字间空一个汉字符位，黑体16磅加粗居中，单倍行距

宋体10.5磅居中

**随着电子商务的发展，快递业务迅速成长。目前末端快递员的主要业务是派送业务，寄件业务被忽视。目前的寄件订单服务主要将寄件单按照普通的快递单来处理，揽件时间由快递员自行安排，仅提供参考价格，存在快递员操作不规范，用户体验差，服务确定性得不到保证，系统拓展性不高等问题。**

**针对末端寄件服务存在的上述问题，本课题在充分调研末端寄件需求，分析业务细节的基础上，基于Spring框架，采用微服务架构思想，使用现有的中间件平台，设计整体系统架构，构建一套完整的寄件快递单处理系统。在系统上实现消费者和快递员线上线下的业务协作，产生和执行寄件时效任务，显示跟踪物流信息，接入移动应用，配置相关的运营活动等功能。将系统部署到实际生产环境，设计和完成对系统质量的评估。使系统达到稳定处理一定规模访问量，提供时间确定和价格确定的可靠服务，灵活拓展末端新业务的目的。**

**关键词：**寄件快递单 处理系统 微服务架构

宋体12磅，行距20磅， “关键词”三字加粗

**ABSTRACT**

Time New Roman 10.5磅

Times New Roman 12磅，行距20磅，段前段后0

Arial 16磅加粗居中，单倍行距

With the development of electronic commerce, the courier business is growing rapidly. At present the main business of the end courier is the delivery service while mailing service is ignored. The current order service for sending is primarily processed according to the ordinary delivery orders, and the time pieces are arranged by courier at random. And the service only provides the reference price. The existing problem is that the courier operations are not standardized, user experience is poor, no deterministic service quality is ensure and the system scalability is not high.

To solve these problems of the end service for sending, this subject conducts the full investigation of the needs for sending and the detailed analysis of the business, and adopts the Spring framework, the microservice architecture and the existing middleware platform to design the overall system architecture and builds a complete processing system for sending order. This processing system of this subject implements the online and off-line business collaboration between couriers and consumers, production and performance of logistics tasks of sending, display of the tracking logistics information, connection of the mobile application, configuration of the operational activities and other functions. This subject deploys the system in the production environment, designs and completes the assessment of the system's quality. And this subject makes the system read the target of providing the stable treatment of visits which reaches a certain size, the reliable service of deterministic time and price and the flexible scalability of end business.

**Key Words:** sending order, processing system, microservice architecture

Times New Roman 12磅，行距20磅 “Key Words”加粗

**目 录**

一级节标题宋体12磅，单倍行距，段前6磅，段后0磅，两端对齐，页码右对齐，左缩进一个汉字符位

标题：二字间空一个汉字符位，黑体16磅加粗居中，单倍行距段前24磅，段后18磅

章标题： 宋体14磅,单倍行距, 段前6磅，段后0磅，两端对齐,页码右对齐

第1章 绪论 1

二级节标题宋体10.5磅，单倍行距，段前6磅，段后0磅，两端对齐，页码右对齐，左缩进2个汉字符位

* 1. 论文研究背景 2
  2. 国内外技术研究应用现状和发展趋势 3
  3. 国内外研究现状 4
     1. 寄件快递单处理系统 5
     2. Spring框架和微服务架构 6
  4. 章节安排 7

第2章 关键技术介绍

* 1. Spring框架 8
  2. 微服务架构 9
  3. 中间件平台和无线组件 10
  4. 本章小结 11

第3章 寄件快递单处理系统的需求分析

* 1. 角色分析 8
     1. 寄件人 10
     2. 快递员 11
     3. 运营人员 12
  2. 功能性需求分析 13
  3. 非功能性需求分析 14
  4. 本章小结 15

第4章 寄件快递单处理系统的设计

* 1. 系统总体设计 15
  2. 寄件订单子系统的设计 16
  3. 寄件派单子系统的设计 17
  4. 寄件运营后台子系统的设计 18
  5. 数据库设计 19
  6. 本章小结 20

第5章 寄件快递单处理系统的具体实现

* 1. Spring框架 8
  2. 微服务架构 9
  3. 本章小结 22

第6章 寄件快递单处理系统的测试和分析

* 1. 功能测试 8
  2. 性能测试 9
  3. 本章小结 22

第7章 总结和展望

* 1. 本系统的特点 8
  2. 成果和展望 9

参考文献 71

附录 88

致谢 90

**第1章 绪论**

一级节标题：黑体14磅顶左，单倍行距，段前24磅，段后6磅，序号与题名间空一个汉字符位

章标题：黑体16磅加粗居中，单倍行距，段前24磅，段后18磅，章序号与章名间空一个汉字符位

* 1. 论文研究背景

近年来，我国电子商务发展迅猛。2014年我国电子商务交易额达到13.37万亿元，同比增长28.6%；网络零售交易额达到2.79万亿元，同比增长49.7%。2015年预计达到20.8万亿元，网络零售额平均增长超过50%，达到4万亿元，位居世界第一[[1]](#中国网络购物市场研究报告)。

截止2015年12月，我国手机网民规模达6.20亿，占网民总数90.1%，移动互联网应用已经应用到基础的商务交易、信息查询等领域，塑造了全新的社会生活形态，潜移默化地改变着网民的日常生活，移动互联网应用在未来还将更加贴近网民生活[[2]](#第37次中国互联网络发展状况统计报告)。

随着国内电子商务的繁荣发展，快递行业发展迅速，2010-2014年中国快递发展指数年均增速29.6%。2014年我国快递行业以140亿的年业务量，超过美国，成为世界第一[[3]](#中国快递发展指数首次发布)[[4]](#中国快递业务量达140亿件跃居世界第一)。而且快递行业继续高速发展，2015年全国快递服务企业业务量累计完成206.7亿件，同比增长48%[[5]](#邮政行业运行状况2015)。从服务范围上，快递基本覆盖了全国所有区域，也到达了农村。比如，相关数据显示，淘宝上由15家民营快递公司组成的民营快递网络，能够将包裹送达超过50%的乡镇，菜鸟的配送网络实现全国95%的区县覆盖，快递服务网点目前基本实现了在县一级的全覆盖[[6]](#菜鸟网络农村淘宝全覆盖)。

如图1-1和图1-2，是中国快递业务数据统计[[5]](#邮政行业运行状况2015)[[7]](#邮政行业运行状况2014)[[8]](#邮政行业运行状况2013)[[9]](#邮政行业运行状况2012)[[10]](#邮政行业运行状况2011)[[11]](#邮政行业运行状况2010)：

图1-1 2008-2015年中国快递业务收入

图1-2 2008-2015年中国快递业务量及增长情况

电商将实体经济从线下带到线上，快递行业又让商品实物从线上回归线下，互联网电商的影响，也在不断深化，不仅了建立完善的电子商务平台，更是对商品的供应、仓储、物流配送进行全渠道整合，在提高物流、服务等环节展开竞争，全面提高自身的软实力。

以快递员为主体的快递末端，不仅提供了主要的派送服务，也发展了上门寄件、同城寄件等业务。尤其随着电商无理由退换货等服务的提供，同城、二手交易的发展，更多寄件需求产生。

* 1. 论文研究的目的和意义

近几年，移动互联网迅猛发展，自从06年谷歌开发出智能移动操作系统Android，开放源码并授权手机厂商大量生产后，安卓手机持有量迅速上涨，苹果公司的iOS系统和手机硬件也吸引了大量用户，目前在智能移动手机市场在平分秋色。

对接安卓和iOS系统，能迅速地将软件延伸到最广泛的用户，直接面向消费者拓展寄件业务。同时减少了智能终端推广代价，初期没有终端硬件投入成本，只需要软件APP的营销推广，后期也只需维护应用层软件系统即可，这种方式也是移动互联网领域广泛采用的方式。

此外，移动端市场竞争激烈，需求持续变化，每月、每周都有新的活动和业务变化，新的业务的种类也围绕核心业务在不断形成。相应的软件应用形式也在持续地更新，在保证基本用户习惯，核心流程稳定的前提下，应用可能持续向水平方向，以满足不断拓展的业务需求。伴随着软件推广和运营活动开展，订单的数量也会持续上涨，如果软件具有很好的伸缩性，则能提供良好的响应和用户体验。

目前快递员主要在网点收取寄件或者电话联系消费者，如果结合寄件订单处理，同时建立了快递员和消费者的实时消息联系，实现线上线下互操作，不仅方便了快递员，减少了沟通成本，也能够更加提供给消费者更加透明、更加确定的寄件服务。

综上，建立一套标准、规范、能直接对接移动APP、能快速拓展的寄件快递单处理系统，具有广阔前景和实际社会价值。

* 1. 国内外技术研究应用现状和发展趋势

**（**1**）寄件快递单处理系统**

国外的研究者，基于成熟的末端智能网络，在寄件快递单处理上，有具体的研究和应用实现。

比如Michael A. Rivalto在《System for Automated Package Pick-up and Delivery》[[28]](#SystemforAutomatedPackagePickUp)中，围绕智能收寄设备，研究应用了一套自动化的快递件收寄处理系统，使用中心数据库存储收取、寄送的订单信息。其中寄件订单的处理包括投寄、信息录入(目的地、快递公司、支付信息)、取件等环节，由中心服务器和数据库进行信息的处理，系统包括了向快递员发出通知的功能。

Ole-Petter Skaaksrud和Cameron Dee Dryden等人在《Internet-based Shipping, Tracking, and Delivery Network Supporting A Plurality of Mobile Digital Image Capture and Processing (MICAP) Systems》[[29]](#InternetBasedShippingTracking)中，借助图像处理，在手机端自动采集、生成快递单信息，在信息中心集中处理。

Ole-Petter Skaaksrud等人考虑了快递包裹的提货、投送、运输等环节，以及运单信息的更新、包裹处理节点的消息通知等问题，在手机端web网络环境下，对处理系统做了具体实现。

国内的研究者，比如赵园园在《电子商务环境下社区智能快递系统助力快递末端配送效率提升》[[30]](#赵30园园电子商务环境下社区智能快递系统助力快递末端配送效率提升)中，针对快递员末端，围绕智能快递柜，建立智能快递系统，处理配送的寄件快递单信息，同时将系统拆分为快件存放子系统、短信支撑子系统、收件监控子系统，描述了取寄件流程的具体细节。

许校境、郑召文在《基于RFID的快递系统的研究与应用》[[31]](#许31校境郑召文基于RFID的快递系统的研究与应用)中，基于RFID技术实现了一套快递单处理系统，涵盖了寄件功能，并对包裹取走、短信发送、物流追踪等环节，以及单据的详细属性信息都做了分析。

武防震、姚国祥在《基于UML的快递系统建模》[[32]](#武防震32姚国祥基于UML的快递系统建模)中，使用标准建模语言UML，根据快递软件系统处理的寄件和派件业务，对快递单在物流链路变化过程中的状态和对应的软件模型做了整体设计。

以上应用和研究，主要围绕末端智能设备构建末端网络和快递单处理系统，针对寄件业务中快递员和消费者之间，基于位置、短信和寄件订单的流程处理，但在安卓、ios设备已经普及的前提下，缺乏实用性。在寄件快递单的处理上，对后端架构的设计缺少可扩展性，容错能力，没有提供时间和价格确定的服务保证。

随着智能手机的广泛普及，用户对APP应用的使用频率增加，直接面对智能手机APP，能够对接主流快递公司的寄件订单处理系统，具有更大的适应性和实际价值。

（2）Spring框架和微服务架构

Spring框架是Rod Johnson、Juergen Hoeller等开发的，用于支持JavaBean构件运行的容器[[33]](#李刚33轻量级JavaEE企业应用实战)。该框架提供了依赖注入方式的构件组装机制和基于AOP技术的事务和日志管理等功能。Spring框架的出现，解决了传统J2EE架构中, EJB构件开发难度大、数据访问效率低、难以进行单元测试等问题。Spring框架也有效地弥补了普通轻量级架构存在的构件之间耦合度高，缺乏统一的事务和日志管理服务等不足。Spring框架在业界已得到了广泛的推崇和应用，也使得J2EE的应用开发变得越来越简单、高效而且规范。

微服务架构已经被证明是一种可行的新型架构，Villamizar和M. Garces等人指出，微服务架构能够实现高效部署可扩展的应用，在云计算技术的支持下，还能为企业级应用提供动态调整计算资源的能力。近年来，已经被国外大型互联网公司，比如Amazon, Netflix和LinkedIn等公司，在部署大型应用时所采用[[34]](#E34valuatingTheMonolithicAndTheMicros)。国内的银行和电信行业，逐渐将微服务架构应用投入市场，例如，中国建设银行升级微信公众平台[[35]](#邢帆35微服务)，使应用由原有的单一推送功能，转变为具有多种业务功能的综合性服务平台，使用的就是微服务架构。

微服务架构将应用拆分为多个子服务，实现独立测试、部署、扩展、维护和升级，降低应用的复杂度，更加灵活高效地在云平台上扩展。微服务架构，在未来的企业业务细分，转型和升级，以及云计算平台技术迅速发展的大背景下，具有广泛的应用前景和实际的工业生产价值。

* 1. 论文的组织安排

本课题是在快递业务高速发展，快递末端寄件业务不断拓展，快递员和消费者之间存在信息缺位，针对寄件服务的不足而提出的。其中涉及到多个方面的问题：服务保证、快递员和消费者的业务互操作、业务拓展、系统稳定等。课题拟基于微服务的思想，采用Spring框架，并结合某公司的相关技术平台，实现一个寄件快递单处理系统。课题主要包含以下3点内容：1.寄件快递单处理系统总体结构的设计；2.寄件订单、派单子系统的设计和实现；3.运营后台子系统的设计和实现。

本文各章节安排如下：

第1章，绪论，阐述本文研究内容的背景和意义，以及相关领域的国内外研究现状和发展趋势，介绍本次论文的主要内容和目标。

第2章，关键技术介绍，介绍寄件快递单处理系统使用的核心技术和方法，阐述Spring框架、微服务架构和所使用的平台技术和组件，说明了该系统采用的关键技术和及实现的技术方案。

第3章，寄件快递单处理系统的需求分析，介绍寄件快递单处理系统的具体需求，分析系统依赖的环境和要达到的目标。

第4章，寄件快递单处理系统的的设计，对寄件快递单处理系统的系统架构进行了设计，对子系统和子模块，做了具体细分和详细描述。

第5章，寄件快递单处理系统的具体实现，给出了主要子系统的实现方法，Spring框架和微服务思想在实例中的具体应用，最后简单分析了系统的应用效果，以及和同类系统的应用效果的比较。

第6章，总结，对寄件快递单处理系统的技术思路和实现成果做了总结，分析了系统的创新点和论文的主要工作，指出了系统的不足之处，指出了需要改进的要点和进一步工作。

**第2章 关键技术介绍**

一级节标题：黑体14磅顶左，单倍行距，段前24磅，段后6磅，序号与题名间空一个汉字符位

* 1. 寄件快递单处理系统概述

段落文字：宋体12磅（英文用Times New Roman体12磅），两端对齐书写，段落首行左缩进2个汉字符。行距20磅（段落中有数学表达式时，可根据需要设置该段的行距），段前0磅，段后0磅

* 1. 微服务架构
  2. 软件扩展模型

如图1-3，根据Martin L. Abbott的理论[[12]](#TheArtofAcalability)，软件拓展模型可以分为三个维度：

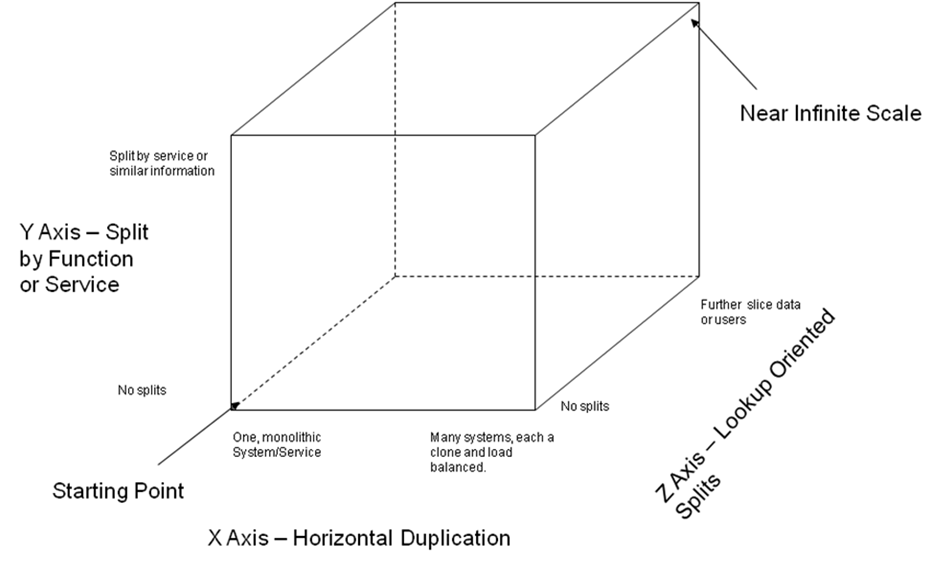


图1-3 软件架构拓展的立方体模型

（1）X轴向拓展：

基于负载均衡等方法，通过重复的方式，在多台机器上部署应用，运行并处理请求的方式，叫做X轴向拓展。

（2）Y轴向拓展：

将应用分成多个不同的子服务，每个子服务负责一个或多个相关的功能，各个自服务负责的功能不同，这种拓展方式叫做Y轴向拓展。

有多种方法将应用分解为子服务，一种通过基于动词的语义进行分解，比如满足单一用例的结账服务。还有一种方法是通过名词将应用拆分，分解并建立不同的子服务，分别负责某一个名词对应实体的相关的所有操作。应用一般使用上述两种方式拆分为对应的子服务。

（3）Z轴向拓展：

与X轴向拓展类似，每个服务器运行同一份代码，但是不同点在于，每个服务器只负责整个数据的一个子集，系统中某个组件专门负责将每个请求路由到合适的服务器上。比如，一种路由策略将请求的某个属性作为访问如数据的主键，将不同的请求映射到对应的服务器上。或者将请求源进行区分，如普通用户和付费用户，将请求路由到不同的服务器上。这种拓展应用的方法，叫做Z轴向拓展。

* 1. 三种软件拓展模型优缺点

（1）X轴向拓展的优缺点:

X轴向拓展的优点：如果存在N份代码拷贝，运行在独立的服务机器上，则每一台服务器只承受1/N的负载，拓展方式简单，因而广泛使用。

X轴向拓展的缺点：一个缺点在于每一份代码拷贝都可能访问到全部数据，机器缓存需要更大的存储空间，保证读取的效率。另一个缺点是解决代码规模增加引起的开发量的增加和应用复杂度的增加。

（2）Y轴向拓展的优缺点:

Y轴向拓展的优点：基于功能将应用拆分，配合开发资源，降低每次迭代的开发量，减少了应用复杂度[[13]](#AnIndustrialCaseStudyofPerformance)。

Y轴向拓展的缺点：对应用实现的功能要详细分析，分割功能存在部分取舍的代价。

（3）Z轴向拓展的优缺点:

Z轴向拓展的优点： 每个服务器只处理数据的一个子集。提高了缓存的使用效率，降低内存占用和I/O流量。由于正常情况下一个请求分发到多个服务器上，从而提高了事务的拓展性。由于能保证部分数据访问失败，从而提高了故障的隔离性。

Z轴向拓展的缺点：1.应用复杂度的提高。2.分配数据的分配策略相对复杂，如果数据需要再次分配，则复杂度更高。3.开发量增加和应用复杂度增加的问题仍然没有解决。

* 1. 微服务架构的优点

微服务思想，是Y轴向拓展的，将整个应用拆分为多个“微服务”，具有以下优势[[14]](#PatternMicroservicesArchitecture)[[15]](#MicroservicesfromTheorytoPractice)。

1.每个微服务相对较小，在需求和设计上都更容易被开发人员理解；每个功能对应的微服务可采用不同的技术方案、技术工具，各子团队的开发人员使用对应的集成开发环境，提高开发效率[[16]](#LightweightServiceDescriptionsForRes)；web容器的负担更小，启动速度更快，使的开发效率提高，应用部署时间缩短[[17]](#MicroserviceBasedArchitectureforthe)。

2.每个服务独立于其他的服务，部署上也可以独立进行，方便更加频繁地部署新版本的服务[[18]](#图书馆学研究)。

3.扩展开发团队更加简单，方便将开发人员组织在多个团队上。每个团队负责单个服务，能独立于其他的团队开发、部署和拓展自己的服务。

4.更好的故障隔离。比如，如果某个服务存在内存泄漏，只有这个服务受到影响，其他的服务能够继续处理请求。比较而言，集成在一起的整个系统架构，如果有一个组件不正常，整个系统都会受到影响。

* 1. Spring框架
  2. 主流web服务框架对比

本课题考虑的应用对接物流数据平台，寄件快递单处理流程部分类似于电商订单的处理，而且考虑了直接对接公司电商平台和中间件平台，便于业务开展和技术复用，考虑使用Java语言。同时，Java语言作为一种流行的开发语言，具有跨平台性，以及面向对象的语言特性。对于接入主流WEB框架，运行在Tomcat等开源服务器上，具有毫无疑问的优势。同时Java自带垃圾回收机制，舍弃了C、C++对内存的管理要求，简化了开发复杂度，提高程序的可靠性和开发速度，对于web程序而言是一个很好的选择。目前JavaEE等规范建立了WEB开发标准，对于企业级的WEB应用，应对复杂的平台环境，达到企业级集成，高效的生产能力，企业级的应用组件，未来可保证的技术支持等方面，具有难以替代的优势[[19]](#于J2EE的订单跟踪微架构研究与应用)[[20]](#基于Spring框架的供应链订单管理系统)，本课题主要考虑基于Java（尤其是符合JavaEE平台标准）的WEB服务框架。

（1）Struts框架

Struts是一个Java的基于请求的WEB框架。Struts 2遵循MVC模式，其中C-控制器被称为动作控制器(Action Controller)。Struts 2通过映射文件的配置将请求导向对应的动作(Action)，而且通过动作将数据展示给用户。Struts 2支持基于注解的配置方式，并且在WEB应用中使用Action类作为Model。Struts 2提供了强大的API支持对于过滤器的配置[[21]](#基于Struts框架的JavaWeb应用开发研究)[[22]](#Struts2TheModern)。Struts在很多应用中和Spring搭配，应用在表现层，持久化用Hibernate实现，称为SSH(Struts+Spring+Hibernate)框架组合[[23]](#基于Spring、Struts和Hibernate)。

（2）Play框架

Play框架是一个全栈的Java Web应用框架。Play遵循MVC架构模式，主要特点是提高了开发效率，代码的热重载(hot code reloading)和程序错误信息在浏览器中直接显示。

Play框架使用了SBT(Simple Build Tool)提高构建速度，并且提供热重载的功能，使代码的改变的效果能在”编码+刷新”的流程中快速体现，并配合Maven进行代码仓库的管理，相比于其他框架代码改变能迅速获得反馈。

Play框架提供了良好的对单元测试的支持，配合使用Sonar等测试工具快速实现单元测试。此外，Play框架还能直接在浏览器中查看错误信息和发生错误的代码片段。

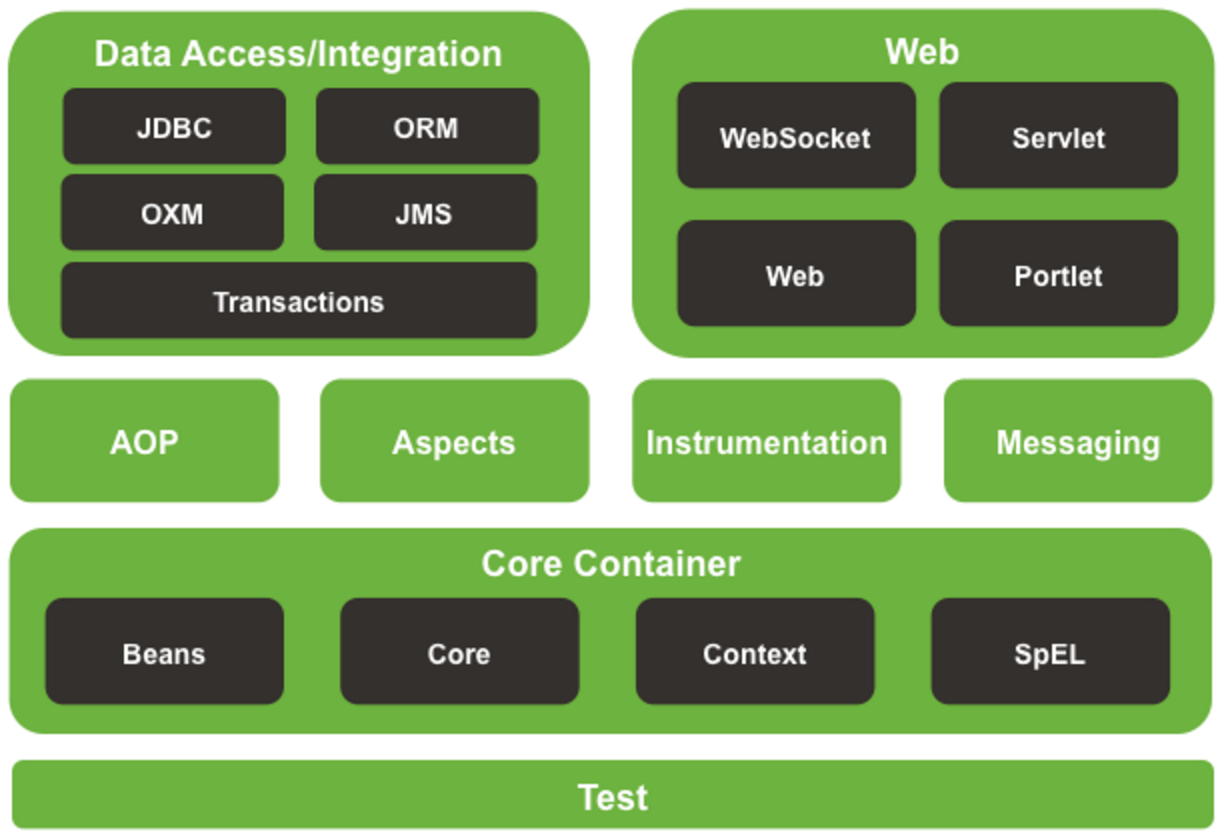
Play框架使用了JBoss Netty作为web服务器，它实现了对请求的异步处理，实现了非阻塞的服务，区别于一些不支持异步处理的JavaEE框架。同时支持了socket和流处理(streaming)。

（3）Spring框架

Spring框架是一个建立Java应用的轻量级框架，支持包括Java的WEB应用。它也是一个基于Java的基于请求的框架，核心是IoC容器，通过依赖注入绑定对象组件。同时Spring还提供了事务管理和对其他各种技术框架的无缝集成[[24]](#基于Spring框架面向多业务Web平台开发框架研究)。

* 1. Spring框架的组成和优势
     1. Spring框架的组成

目前的Spring框架依据功能特性，分成了20个独立的模块，它们共同组成了Sping框架的核心容器、数据访问/集成、Web、AOP(Aspect Oriented Programming, 面向切面编程)、Instrumentation、消息和测试，如图所示：

图 Spring的组成模块

Spring框架的核心：核心容器模块，包含了spring-core，spring-context，spring-beans，spring-context-support，spring-expression五个子模块。其中，spring-core和spring-beans模块组成了整个Spring框架的核心组件，提供了对IoC和依赖注入的支持，并通过用工厂模式实现的BeanFactory，将应用配置和依赖描述于程序的逻辑代码分离。

基于spring-core和spring-beans模块建立的spring-context模块，负责管理Spring中对象的生命周期，并提供了对事件的传递、资源文件的加载和上下文创建的支持。spring-context模块还提供了对Java EE特性的支持。

Spring-aop模块提供了对AOP（面向切面编程）的支持，让程序能够通过对方法拦截器和pointcut（切入点）的定义，实现应用代码和共用的功能逻辑代码解耦的目的。

* + 1. Spring框架的优势

1.IoC(Inversion of Control控制反转)。IoC技术是一种将对依赖的组件的创建和管理转移到外部的技术。比如类B依赖于类A的实例，在传统编程方式中，要么实现在类B中实现类A的实例，要么通过工厂类获得类A的实例。采用IoC之后，类A的实例，是在运行时环境中，通过框架注入到类B当中的，这个操作称为依赖注入。而IoC是实现依赖注入的一种高效的方式，通过这种方式，Spring，作为一个容器，提供了应用类的所有依赖。通过显式配置或者自动绑定的方式，Spring允许很少的开发量就能完成IoC，实现了非侵入式和插件化的编程模型[[25]](#论spring的零配置与XML配置)。

2.AOP(Aspect Oriented Programming, 面向切面编程)。AOP是一种通过在不需要对核心业务代码做修改的情况下，实现横向的关注点(如事务管理、日志等)的技术。它可以通过预编译和运行时期动态代理的方式来实现。AOP提供了在编程需要的时候，就能获取到像拦截器这样模块中的横切代码的能力，从而实现了横切关注点的功能。AOP使的开发人员只需关注系统中主要业务逻辑部分，对于系统级别的日志处理、事务管理，提供了方便的切面式的实现方式[[26]](#SpringAOP技术在电子商务中的应用)。Spring提供了基于代理的AOP能力。

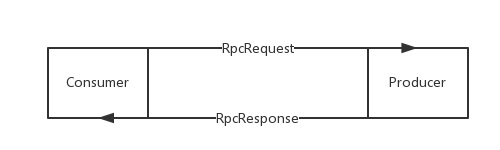
3.Spring还提供了对于Junit4的支持，方便单元测试，并且方便地集成各种开源框架。同时Spring还提供了数据访问\集成、web编程的支持：如JDBC、ORM、WebSocket、Servlet等。

相比于Struts框架，Spring提供了更好的对于后端的支持，其中IoC和AOP功能，在事务、组件充分解耦等方面，提供了对于我们的寄件快递单处理系统，这种企业级的关注于业务逻辑的系统的完整技术支持。

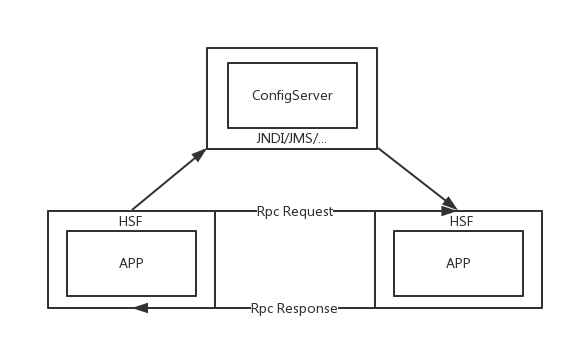
相比于Play框架，Spring优势在于：(1)发展时间久，更成熟，目前存在很多基于Spring的web应用，采用Spring能够实现无缝对接。(2) 大量的技术文档，成熟的技术团队，开发者社区的技术支持。(3)Spring提供了更加丰富的组件和API接口支持，能够集成各种开源框架，方便企业级拓展[[27]](#SpringFrameworkACompaniontoJavaEE)。

* 1. 中间件平台和无线组件
  2. HSF介绍

HSF(High-Speed Service Framework)，是基于RPC(Remote Procedure Call)的高性能分布式服务框架。HSF的基础是RPC，RPC是提供了远程调用的功能，也就是，A机器调用B机器上的一个对象的方法，B机器将方法的返回值发送给B机器，如下图所示：

RPC调用方式

RPC的实现方式可以是RMI,Hessian等。

HSF通过配置中心(ConfigServer)的设计，让服务发布者(Producer)能够发布服务，服务的调用者(Consumer)能够查找和调用服务，如图：

HSF的配置中心设计

服务发布者负责发布服务，它将服务注册到配置中心上，配置中心采用数据库存储或分布式文件系统存储，存储的内容包括服务的名称、发布服务的Producer 的机器的IP地址、超时时间、序列化方式等信息，并且通过索引和缓存提高查找速度。

在服务的发布者(Producer)和配置中心(ConfigServer) 之间，是典型的CS模式，可以通过JNDI、JMS、WebService等多种实现方式，调用配置中心提供的管理接口。JDNI的接口返回所查找的服务的信息，以Java对象的形式组织，并序列化成字节流，通过网络传输返回给服务调用者。

服务调用者得到服务发布者的具体信息，包括IP地址、端口号、服务名称、超时时间等，然后利用上文描述的RPC实现，进行服务调用(具体实现上，HSF实现了异步调用和对调用超时异常的处理逻辑)。

* 1. 中间件平台

（1）Taobao Notify消息中间件

Taobao Notify中间件是一个Message Oriented Middleware (MOM)，具有通用的MOM所具有的松散耦合和异步处理的特点，并且具有高可靠性、高可用、高效率、水平扩展、安全性，同时保证了消息100%投递。

中间件将消息使用本地数据库存储，并使用事务操作消息的传递，能够可靠的管理消息的提交、存储、投递和订阅关系。

（2）定时任务队列中间件

所使用的定时任务队列中间件，是基于Redis实现的消息队列和对消费者的延迟实现的，能够在分布式集群上快速处理定时任务。

（3）Taobao Tair分布式缓存中间件

Taobao Tair分布式缓存中间件，是一个分布式的Key/Value存储引擎，基于hash算法，将所有Key对应的Value分配到不同的数据节点中，最终提供负载均衡。并通过多备份存储，在节点故障时进行数据迁移，保证了故障下的弱一致性，提供保证安全策略。

（4）Taobao Diamond分布式配置管理中间件

Taobao Diamond分布式配置管理中间件，以Mysql为中心的，在分布式集群的服务器上，将数据以缓存和本地磁盘文件的形式存储，通过长轮询的方式推送数据的实时变化，实现配置的动态推送。

* 1. 无线组件

（1）Agoo无线消息通道

Agoo是一个移动消息推送服务组件。它具有向移动APP推送消息的功能，能够实现高效、精确、实时的消息推送，在客户端以通知的方式呈现消息。

（2）ACCS无线数据通道

ACCS(Alibaba Cloud Channel Service)是阿里的无线通道服务，能够在无线客户端实现稳定的网络接入链路，并具有弱网下的优化、抗抖动、加密传输等特征，实现高效的数据传输。

* 1. 其他技术
     1. iBatis

iBatis框架是一个轻量级的数据映射框架，它提供了持久化API，能够方便实现已有的数据库模式到应用程序的数据库持久层的转化。iBatis框架通过XML编码的SQL映射文件，实现了SQL模版的静态化和执行结果到Java对象的转化。使用iBatis的架构层次如图：

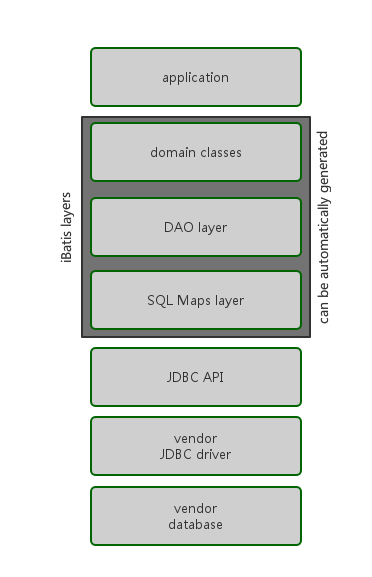


图 iBatis应用的架构层次

除了iBatis，在Java EE领域，应用的比较多的还有Hibernate。Hibernate是严格以对象为中心的持久化方式，而iBatis则是围绕传统数据库模式，进行设计和实现。

在数据库的数据模式完整、确定的 场景，且没有很高吞吐量的要求的情况下，Hibernate能较好地组织和工作在应用架构中。Hibernate提供了很好的通过Java对象操作数据库的机制，但是也给程序带来复杂度的提高，尤其是面对传统数据库模型，使用Hibernate编写的代码往往包含了复杂的SQL查询代码。

在传统数据库模型的应用场景，且查询使用占比很大的情况下，iBatis更加简洁，给程序结构增加的复杂度也更少，从而更具优势。

* + 1. Mockito

Mockito框架是一个mocking（一种支持单元测试的方法）框架，它通过代理、方法拦截等方式，将被测试类和它的依赖分离开来，提供单元测试支持。而且Mockito还提供了更简洁的API支持，测试结果可读性高，对程序错误的验证结果也很简洁。

* 1. 本章小结

本章在分析了微服务架构的优势，并且将Spring框架和其他主流服务框架进行比较后，选择了Spring作为本课题的开发框架。同时选用了Redis实现定时任务队列，还是用了现有的中间件和其他组件，后者主要是以接口的方式提供功能。iBatis在数据库方面提供了优秀的持久化支持，而Mockito在单元测试方面提供了良好的支持。

微服务架构因其扩展性高、灵活性强、适用于分布式云平台，能适应变化的多开发团队合作，近年逐渐被广泛应用于企业级的web服务架构搭建，具有很大的优势和前景。Spring框架、其他中间件和组件的结合使用，使得整个寄件快递单处理系统能够成熟、稳定地得到实现。

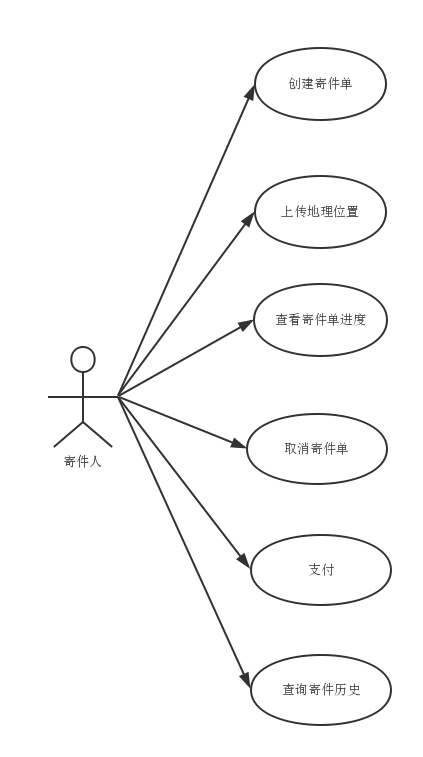
**第3章 寄件快递单处理系统的需求分析**

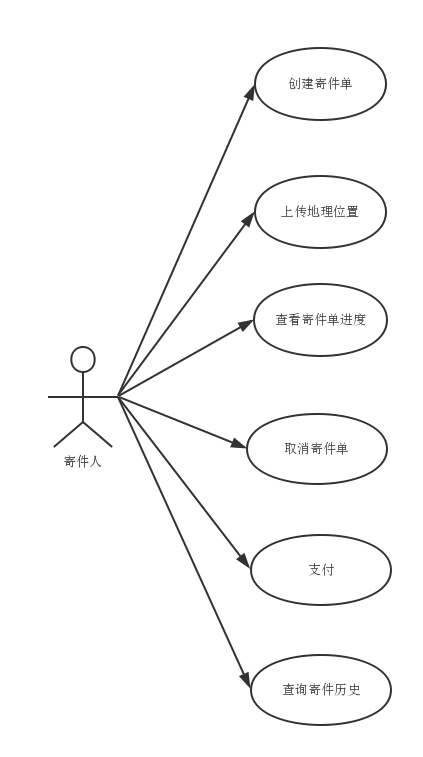
* 1. 用户角色分析

寄件订单处理系统包括寄件人、快递员和后台运营三种用户，基于三个角色，针对系统设计的用例图如图3-1、图3-2和图3-3。

* 1. 寄件人

系统最主要的用户是寄件用户，寄件人的在手机应用上登录后并选择进入寄件模块，在允许应用获得地理定位的前提下，进行创建寄件单的操作，包括填入必要的寄件人、收件人、物品大小重量等信息。下单后，系统会将寄件订单推送给附件小件员或者上门寄件服务提供者(统一为快递员角色)，快递员在规定的时间内上门取件，寄件人进行支付，快递员打包带回快递包裹并填写运单，回传运单号到系统，寄出包裹。寄件人能在快递包裹模块，继续跟进寄出包裹的物流信息，也可以回到寄件模块查看历史寄件记录。





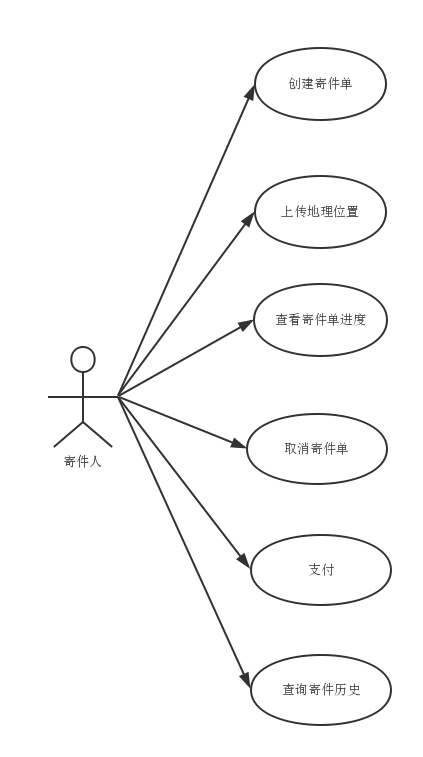


图3-1 寄件人用例图

* 1. 快递员

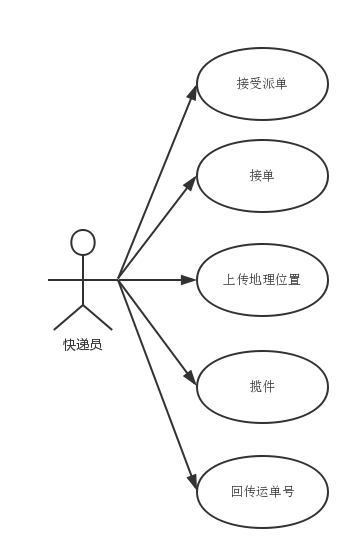
每个寄件订单系统都进行计时，在规定的时间内，附件快递员没有接单的情况下，会转成预约寄件单，并将记录转给外部的结算系统进行赔偿。兜底的服务商直接通过系统派单，继续提供寄件服务。在等待时间内，寄件人能够查看寄件时间、是否被接单、快递员信息等。

图3-2 快递员用例图

* 1. 运营人员

本系统直接依托于公司开发的互联网移动端产品，要能提供给后台运营人员，相关的运营活动、线上配置能力，包括向用户进行寄件服务消息、运营活动消息推送，比如红包优惠等，以及在开发协助下，对应用的补丁包的上传和对首页界面的配置等功能。

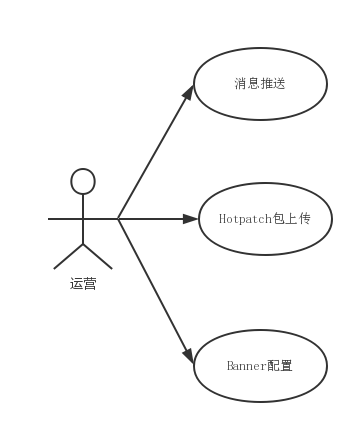
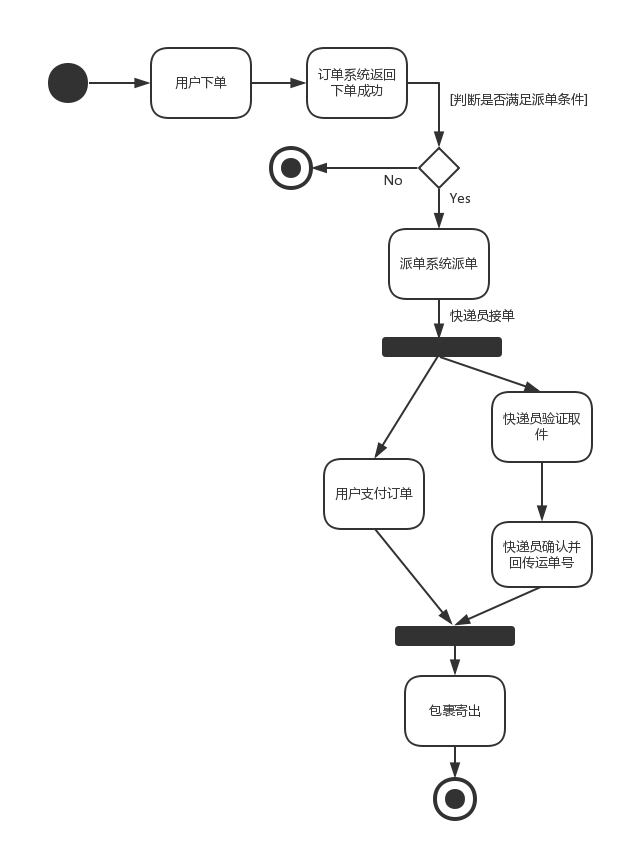


图3-3 运营用例图

* 1. 功能性需求分析

本课题拟实现的寄件快递单处理系统，具有完整的寄件订单管理功能，还能对接小件员系统，将订单配送给小件员进行后续的运单处理。此外系统还提供了对于业务异常或系统异常的处理。

核心业务流程用下面的活动图表示：



本课题设计的寄件快递单处理系统，拟实现的功能有：

（1）寄件人通过移动应用提交寄件信息，系统提供校验、记录、生成订单的功能。

（2）基于地理位置，获取附近小件员信息和其他服务提供者信息的功能。

（3）根据配置的优先级以及快递员评价模型，将订单推送给附近快递员的功能。

（4）快递员抢单，揽件和回传运单号的功能。

（5）寄件人查看寄件订单的服务计时和状态变化的功能，查看历史寄件单的功能，通过公司物流平台服务跟踪物流信息的功能。

（6）使用结算平台服务，寄件人进行支付，快递员查看支付状态、结算结果的功能。

（7）运营人员使用运营后台，发布寄件服务相关消息，优惠活动消息的功能。

* 1. 非功能性需求分析

本章在分析了微服务架构的优势，并且将Spring框架和其他主流服务框架进行比较后，选择了Spring作为本课题的开发框架。同时选用了Redis实现定时任务队列，还是用了现有的中间件和其他组件，后者主要是以接口的方式提供功能。

* 1. 本章小结

本章在分析了微服务架构的优势，并且将Spring框架和其他主流服务框架进行比较后，选择了Spring作为本课题的开发框架。同时选用了Redis实现定时任务队列，还是用了现有的中间件和其他组件，后者主要是以接口的方式提供功能。

**第4章 寄件快递单处理系统的设计**

* 1. 系统总体设计
  2. 子系统划分

寄件快递单处理系统划分为：寄件订单子系统，寄件派单子系统，寄件运营后台子系统，结算子系统，反作弊子系统等，本课题主要研究实现寄件订单子系统，寄件派单子系统，寄件运营后台子系统。

* 1. 系统总体架构

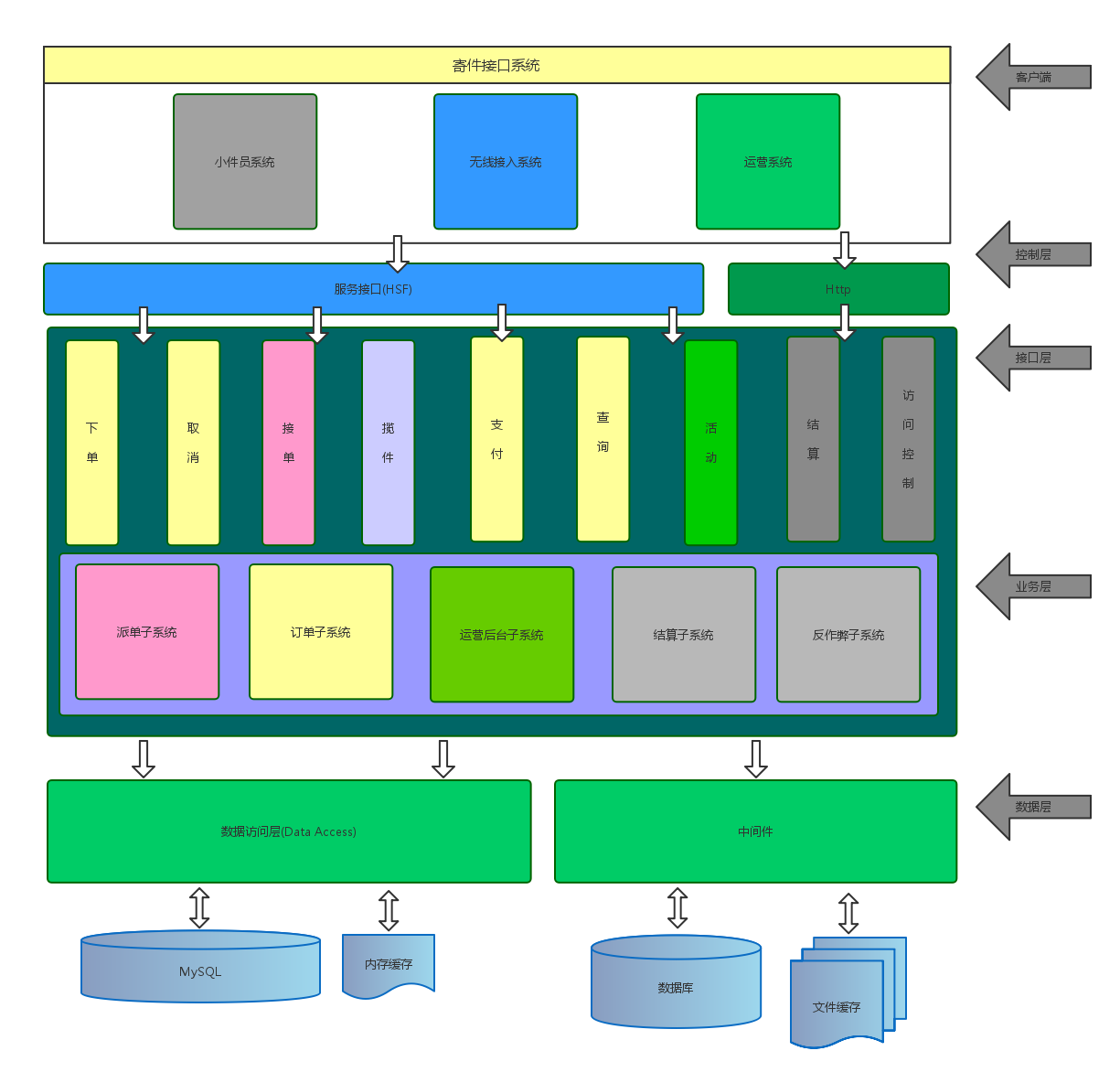
整体系统的架构图如图3-4所示，寄件快递单处理系统整体划分为：

图4-1 整体系统架构图

（1）寄件接口层：主要负责客户端应用的接入；

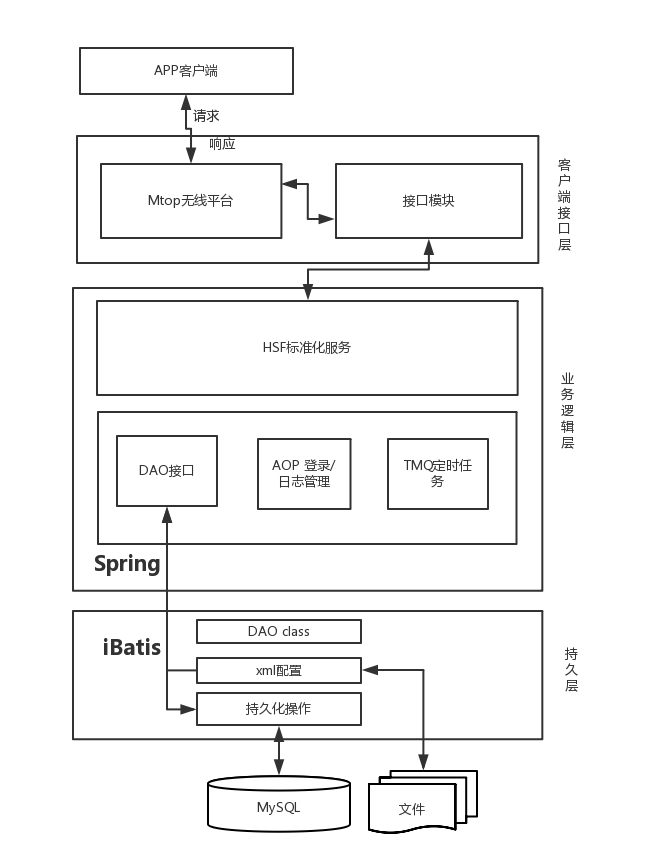
（2）控制层：主要利用公司HSF协议和中间件平台进行服务管理和访问控制；

（3）功能接口层：主要以微服务的方式独立实现和提供各种寄件流程必须的服务；

（4）业务逻辑层：通过各个子系统分别实现对应的功能模块，支撑上层的功能服务;

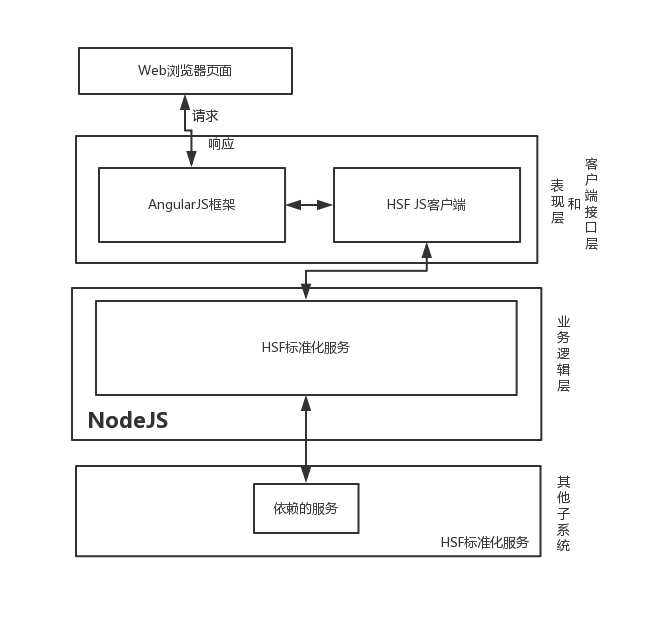
（5）数据访问层：提供对于非持久化缓存、持久化存储和静态配置等数据访问的功能。

整个寄件快递单处理系统按照微服务的思想，拆分为多个独立提供微服务的子系统，各子系统之间通过公司开发的基于http协议的HSF通信方式进行通信，并且通过消息中间件Notify等方式实现子系统之间的弱依赖。

订单子系统和派单子系统的技术开发框架如下：

订单子系统和派单子系统的技术框架

订单和派单两个子系统的技术框架都是在Spring基础上，通过Spring AOP管理登录验证和登录状态信息，根据xml文件对表的描述，通过iBatis和对应的DAO类实现数据到MySQL数据库。同时，MTOP平台负责和客户端APP连接，客户端接口模块将请求转到订单和派单系统的服务。根据整体的微服务设计，基于HSF方式，将服务标准化，各子系统之间可以灵活、方便地调用。



运营后台子系统的开发框架如图：

运营后台子系统的技术框架

运营后台子系统也是通过HSF的方式对外提供标准化微服务，并且依赖其他子系统提供的微服务，表现层使用AngularJS框架实现Web界面的展现，数据控制等，客户端接口层使用HSF的JS客户端访问后端的HSF标准化服务，这些服务是在NodeJS上提供的，整个技术栈不同于订单子系统和派单子系统的Java技术栈，是基于JavaScript语言的框架结构。

运营后台子系统依赖其他的子系统平台实现持久化存储，本身是一个相对简单的子系统，是整个快递单处理系统的一个拓展部分。

* 1. 微服务的管理方案

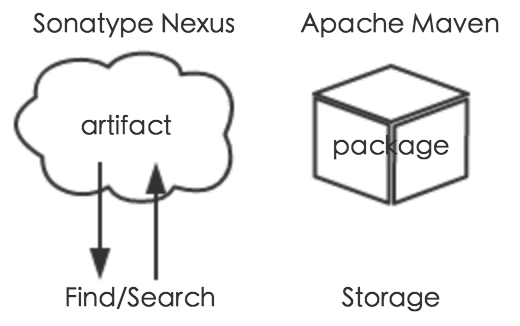
基于HSF，每个APP Server(子系统的服务器)都对外提供自己的微服务，并通过HSF，查询和调用其他服务器对外发布的微服务，整个系统要对这些微服务进行标准化定义和生命周期的管理。

（1）标准化定义

至少包括服务的名称、实现类、版本号，还包括重试次数、超时时间（都有默认参数）等其他配置，针对不同的开发框架，做具体的定义。

（2）服务发现和共享

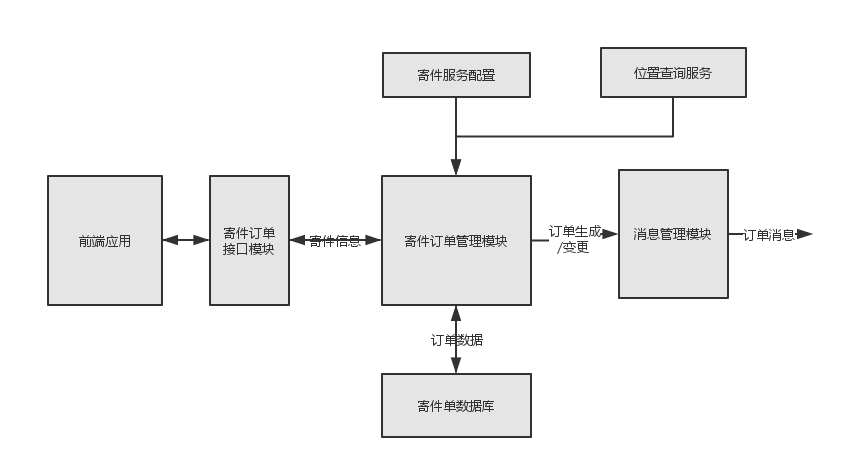
采用开源的Sonatype Nexus和 Apache Maven管理工具。通过Sonatype Nexus对这些服务进行仓库存储、查阅、下载构建包，方便开发。通过Apache Maven能对服务进行依赖管理，构建，编译，部署等。

两者使用上，开发人员通过Nexus检索和发现需要使用的程序的Artifact（工件）信息，同时每个子系统将自身程序打包后存储在Maven Repository（Maven仓库）中，这种使用方式的图像化表示如下：

Nexus和Maven工具的使用

* 1. 寄件订单子系统的设计

本系统的寄件订单子系统的模块划分如图3-5所示。

图3-5 寄件订单子系统

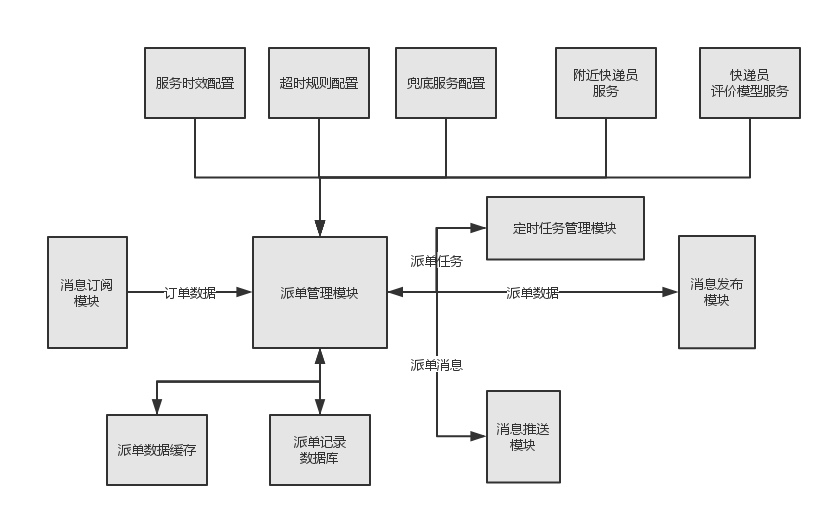
寄件订单系统，是直接接入应用的子系统。具体应用包括：小件员SDK，消费者(寄件人)APP，运营后台。前端应用将寄件人的地理位置信息、选择的服务、填写的地址信息等，通过http请求传入。寄件订单接口模块，基于公司的无线开放平台MTOP，接入无线应用。

普通寄件用户在APP上下单，后端的寄件订单管理模块获取服务配置、地理位置服务提供的具体信息，组装订单对象，并进行持久化存储。小件员通过接入SDK，获取到派单子系统推送的派单信息之后，向订单系统发送请求，订单系统处理抢单逻辑，并返回抢单结果。

此外，根据微服务的思想，让整个系统拆分出来的子系统，充分解耦，实现独立迭代开发。寄件订单系统，与业务流程的下游子系统：派单系统之间，通过消息中间件Notify连。寄件订单系统，在订单创建、变更时，以消息生产者的身份，向订阅这个消息的派单系统，发出对应的消息。

* 1. 寄件派单子系统的设计

寄件派单子系统的模块划分如图3-6所示。

图3-6 寄件派单子系统

寄件派单子系统，主要订阅订单子系统的订单消息，并进行派单相关操作。

依赖的主要配置有：寄件时效配置、超时规则、兜底服务配置。其中，兜底服务是指为了保证服务的可靠性，保证在承诺的时间内有快递员接单，对接业务上合作的物流服务商，如果附件快递员没有抢单，导致订单超时，服务商提供上门服务，保证服务承诺的时效。

派单系统对接外部的子系统的微服务，包括附件快递员服务和快递员评价模型服务。通过上传的地理经纬度，获取附近小件员列表，具体寻找的范围可配置。通过快递员评价模型服务，和配置的业务规则，进行优先级排序，将订单信息推送给小件员和附近提供上门寄件的服务商。

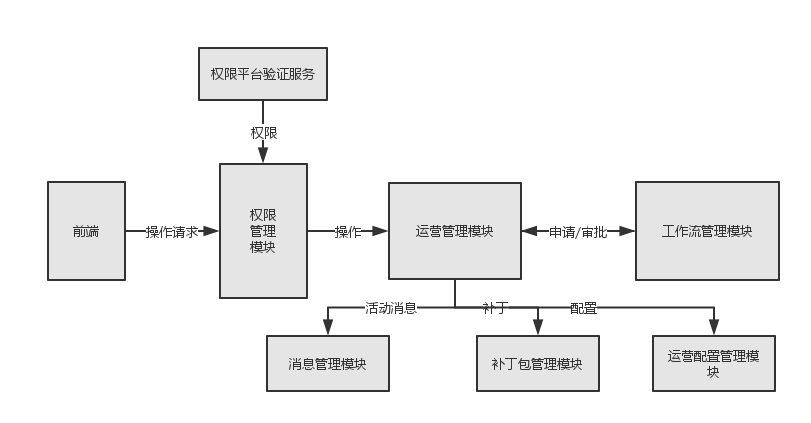
由于派单系统在订单高峰期，尤其是寄件优惠活动开展的时间段，要承受较大的派单处理压力，因此使用了定时消息中间件。定时消息中间件，本质上是一个分布式消息队列，实现对于寄件任务的排队处理，缓解系统压力，保证系统稳定。此外，还使用了分布式缓存系统Tair进行数据的预读取，减少系统数据库访问压力。

对于派单处理成功，业务或系统异常等结果，派单系统都会通过Notify消息中间件，发布处理结果的消息，供订阅的子系统消费和处理。对于派送出去的订单，将通过无线消息通道，推送给小件员应用端。

* 1. 寄件运营后台子系统的设计

基于微服务思想，整个寄件快递单可以方便地拓展，满足前后端需求，运营子系统正是拓展出的子系统，要实现寄件服务变更消息，运营活动消息推送，补丁包发布，运营配置管理等功能。

运营后台子系统的模块划分如图3-7所示。

图3-7 运营后台子系统

运营后台面向公司内部的运营人员使用，包括改变服务端线上配置，向全部移动应用推送消息等敏感操作，考虑到运营人员非专业开发技术人员，可能存在对系统的误操作，因此本系统增加了权限管理和工作流审批模块，保证整个系统的安全性。

相应的权限管理模块和工作流管理模块，起到代理的作用，通过配置和调用相关服务，实现接入公司的权限管理平台和工作流(审核)平台，达到减少系统开发、复用系统功能的目的。

对于运营后台的操作，经过权限验证，工作流审核后，实现了面向移动端的消息推送，运营管理配置(包括移动端运营相关文案)等功能。

* 1. 数据库设计

在寄件业务场景中，首先需要存储寄件订单信息，寄件处理的后期，形成运单，还要考虑订单和运单之间的映射关系，此外还要存储派单记录，对派单行为进行持久化存储，存储运营操作记录，对运营操作信息进行持久化存储等，数据库实现对这些信息的持久化存储，构成整个系统架构中的数据访问层的主要组成部分。此外，还要方便与公司其他业务的数据库进行操作，比如用户数据库、评价模型数据库等。

本系统主要的订单派单场景属于典型关系型数据库设计场景，考虑到目前普遍使用的MySQL数据库系统，具有开源、跨平台、对各种开源存储引擎的支持等优点。而且，公司基于MySQL数据库开发了一套管理平台，实现方便的数据查询、订正、迁移等操作，外部主要业务系统也适用MySQL作为数据库系统，方便进行表的联合查询等。因此本系统拟采用MySQL作为数据库系统。

数据库主要分为寄件订单表、寄件订单和快递单映射关系表、快递单表、派单记录表、运营后台操作记录表：

（1）寄件订单表。负责存储寄件人的登录信息、地图经纬度、寄件收件地址、选择的服务类型等。注：其中的sync\_version是为数据同步的乐观锁而设计的。表结构见表3-1。

表3-1寄件订单表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列 | **列名** | **数据类型** | **长度** | **允许**  **为空** | **是否**  **自增** | **备注** |
| 1 | id | bigint unsigned |  |  | Y | 数据库记录主键 |
| 2 | gmt\_create | datetime |  |  |  | 记录创建时间 |
| 3 | gmt\_modified | datetime |  |  |  | 记录修改时间 |
| 4 | user\_id | bigint unsigned |  |  |  | 寄件用户id |
| 5 | order\_id | bigint unsigned |  |  |  | 寄件订单id |
| 6 | order\_code | varchar | 128 |  |  | 寄件订单号 |
| 7 | order\_status | bigint unsigned |  |  |  | 寄件订单状态 |
| 8 | order\_source | bigint unsigned |  | Y |  | 寄件订单来源 |
| 9 | out\_express\_id | bigint unsigned |  | Y |  | 外部快递单id |
| 10 | out\_express\_source | bigint unsigned |  | Y |  | 外部快递单来源 |
| 11 | pay\_status | bigint unsigned |  |  |  | 支付状态 |
| 12 | order\_price | decimal |  |  |  | 寄件订单价格 |
| 13 | express\_man\_user\_id | bigint unsigned |  | Y |  | 快递员用户id |
| 14 | receiver\_user\_id | bigint unsigned |  | Y |  | 收件人用户id |
| 15 | expect\_pick\_date | datetime |  |  |  | 预计揽件时间 |
| 16 | take\_date | datetime |  | Y |  | 抢单时间 |
| 17 | pick\_date | datetime |  | Y |  | 揽件时间 |
| 18 | pay\_date | datetime |  | Y |  | 支付时间 |
| 19 | express\_date | datetime |  | Y |  | 外部快递派送时间 |
| 20 | sender\_name | varchar | 128 |  |  | 寄件人姓名 |
| 21 | sender\_phone | varchar | 128 |  |  | 寄件人手机号 |
| 22 | sender\_address | varchar | 128 |  |  | 寄件人地址 |
| 23 | sender\_area\_id | bigint unsigned |  |  |  | 寄件人区域id |
| 24 | receiver\_name | varchar | 128 |  |  | 收件人姓名 |
| 25 | receiver\_phone | varchar | 128 |  |  | 收件人手机号 |
| 26 | receiver\_ address | varchar | 128 |  |  | 收件人地址 |
| 27 | receiver\_area\_id | bigint unsigned |  |  |  | 收件人区域id |
| 28 | express\_service\_id | bigint unsigned |  |  |  | 寄件服务id |
| 29 | express\_service\_price | decimal |  |  |  | 寄件服务价格 |
| 30 | longitude | decimal |  |  |  | 寄件人位置经度 |
| 31 | latitude | decimal |  |  |  | 寄件人位置纬度 |
| 32 | sync\_version | bigint unsigned |  |  |  | 记录同步版本 |
| 33 | validate\_status | bigint unsigned |  | Y |  | 寄件订单有效性状态 |

（2）寄件订单和外部快递单映射表。存储订单和外部快递单的映射关系见表3-2。

表3-2 寄件订单快递单映射表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列 | **列名** | **数据类型** | **长度** | **允许**  **为空** | **是否**  **自增** | **备注** |
| 1 | id | bigint unsigned |  |  | Y | 数据库记录主键 |
| 2 | gmt\_create | datetime |  |  |  | 记录创建时间 |
| 3 | gmt\_modified | datetime |  |  |  | 记录修改时间 |
| 4 | user\_id | bigint unsigned |  |  |  | 寄件用户id |
| 5 | order\_id | bigint unsigned |  |  |  | 寄件订单id |
| 6 | express\_id | bigint unsigned |  |  |  | 快递单id |
| 7 | order\_source | bigint unsigned |  | Y |  | 寄件订单来源 |
| 8 | express\_source | bigint unsigned |  | Y |  | 快递单来源 |
| 9 | status | bigint unsigned |  |  |  | 映射状态 |

（3）快递单表。存储揽件并投送后产生的物流快递单数据见表3-3。

表3-3快递单表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列 | **列名** | **数据类型** | **长度** | **允许**  **为空** | **是否**  **自增** | **备注** |
| 1 | id | bigint unsigned |  |  | Y | 数据库记录主键 |
| 2 | gmt\_create | datetime |  |  |  | 记录创建时间 |
| 3 | gmt\_modified | datetime |  |  |  | 记录修改时间 |
| 4 | express\_id | bigint unsigned |  |  |  | 快递单id |
| 5 | express\_delivery\_date | datetime |  | Y |  | 快递派送时间 |
| 6 | mail\_no | varchar |  |  |  | 运单号 |
| 7 | cp\_code | varchar |  |  |  | 快递公司编码 |
| 8 | receiver\_user\_id | bigint unsigned |  |  |  | 收件人用户id |
| 9 | receiver\_name | varchar |  |  |  | 收件人姓名 |
| 10 | receiver\_address | varchar |  |  |  | 收件人地址 |
| 11 | receiver\_phone | varchar |  |  |  | 收件人手机号 |
| 12 | status | bigint unsigned |  |  |  | 快递单状态 |
| 13 | notification\_count | bigint unsigned |  | Y |  | 派送通知次数 |
| 14 | longitude | decimal |  | Y |  | 收件人位置经度 |
| 15 | latitude | decimal |  | Y |  | 收件人位置纬度 |

（4）派单记录表。存储派单记录见表3-4。

表3-4派单记录表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列 | **列名** | **数据类型** | **长度** | **允许**  **为空** | **是否**  **自增** | **备注** |
| 1 | id | bigint unsigned |  |  | Y | 数据库记录主键 |
| 2 | gmt\_create | datetime |  |  |  | 记录创建时间 |
| 3 | gmt\_modified | datetime |  |  |  | 记录修改时间 |
| 4 | sender\_id | bigint unsigned |  |  |  | 寄件用户id |
| 5 | order\_id | bigint unsigned |  |  |  | 寄件订单id |
| 6 | dispatch\_id | bigint unsigned |  |  |  | 派单记录id |
| 7 | dispatch\_date | datetime |  |  |  | 派单时间 |
| 8 | notify\_user\_ids | varchar | 1024 | Y |  | 通知的快递员用户id集合 |
| 9 | notify\_count | bigint unsigned |  |  |  | 通知次数 |
| 10 | is\_accepted | tinyint |  |  |  | 是否被接单 |

（5）运营后台操作记录表。存储运营后台操作记录，其中，工单详细信息、审批人信息等存储在外部的工作流平台。注：表中action\_feature字符串主要存储操作的系统参数信息。表结构见表3-5。

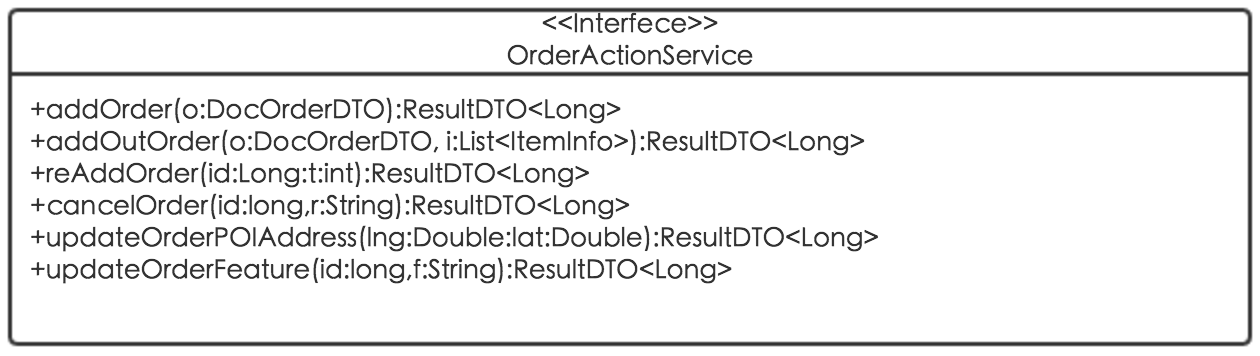
表3-5运营后台操作记录表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 列 | **列名** | **数据类型** | **长度** | **允许**  **为空** | **是否**  **自增** | **备注** |
| 1 | id | bigint unsigned |  |  | Y | 数据库记录主键 |
| 2 | gmt\_create | datetime |  |  |  | 记录创建时间 |
| 3 | gmt\_modified | datetime |  |  |  | 记录修改时间 |
| 4 | work\_id | bigint unsigned |  |  |  | 操作人工号 |
| 5 | work\_name | varchar | 128 |  |  | 操作人姓名 |
| 6 | work\_flow\_id | bigint unsigned |  |  |  | 工单id |
| 7 | action\_type | bigint unsigned |  |  |  | 操作类型 |
| 8 | action\_feature | varchar | 1024 | Y |  | 操作特征 |
| 9 | action\_status | bigint unsigned |  |  |  | 操作状态 |
| 10 | action\_result | varchar | 128 | Y |  | 操作结果 |

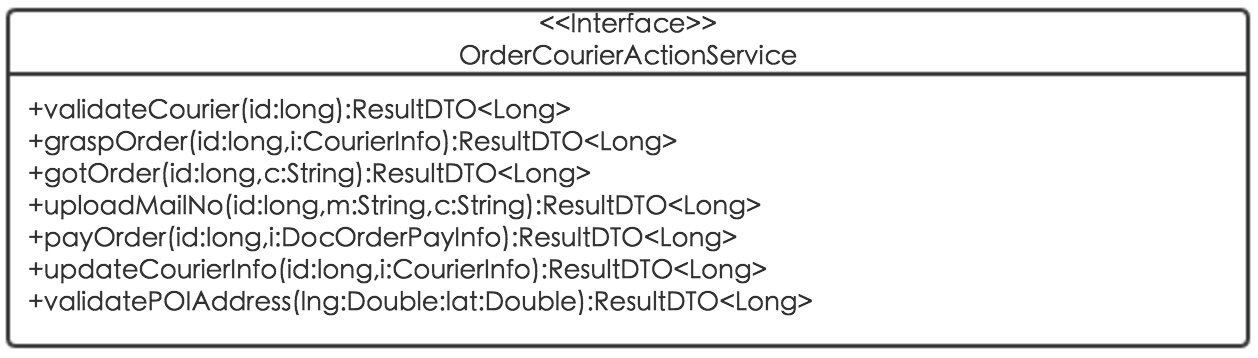
* 1. 本章小结

本章对寄件快递单处理系统划分为的寄件订单子系统、寄件派单子系统、寄件运营后台子系统，进行了内部功能模块的划分和实际，并且进行了数据库设计，对寄件订单表、寄件订单和外部快递单映射表、快递单表、派单记录表、运营后台操作记录表，进行了具体字段的定义。

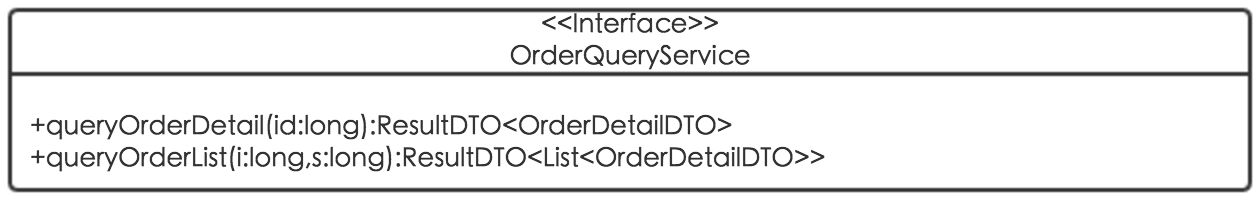
**第5章 寄件快递单处理系统的具体实现**

* 1.  寄件订单子系统

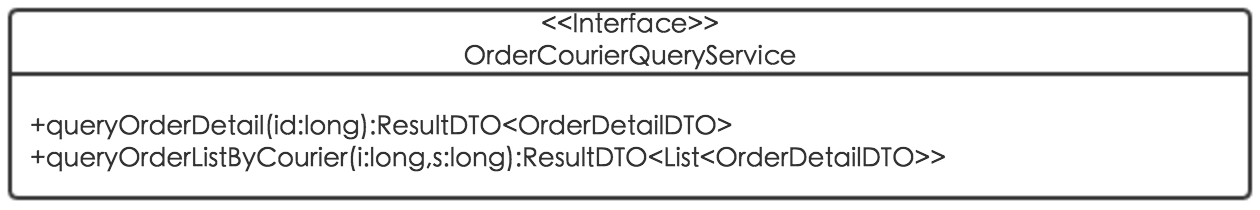
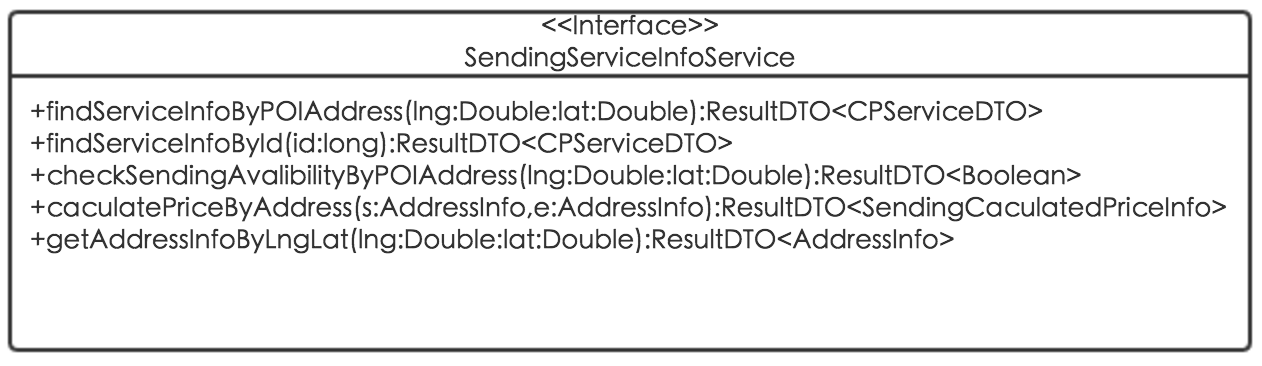
定义的服务：

OrderActionService

OrderCourierActionService

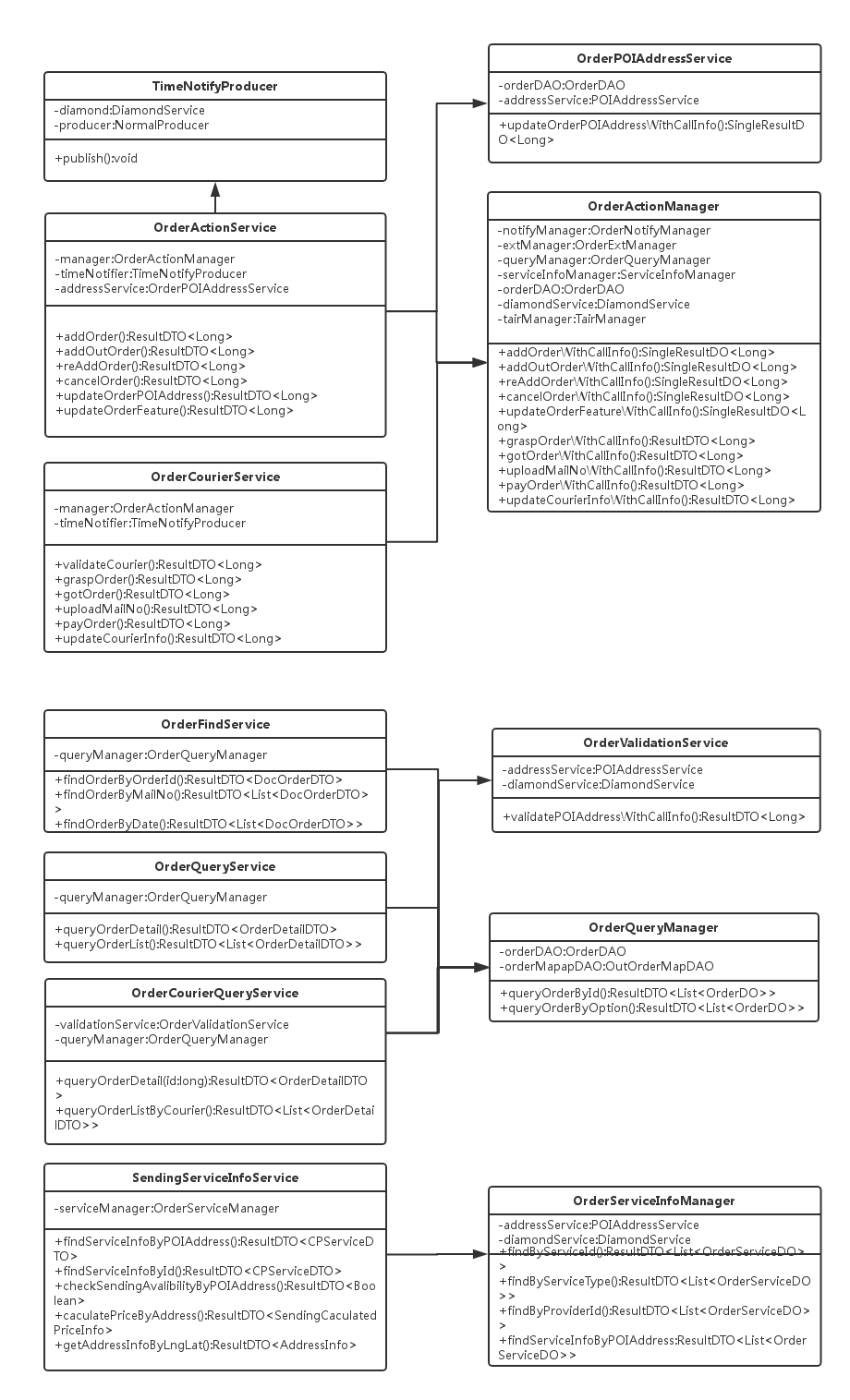
OrderFindService

OrderQueryService

OrderCourierQueryService

SendingServiceInfoService

这些服务类依赖对应的管理器，或其他的服务类进行工作，部分类之间的依赖关系如下(类方法的参数从略)：



订单子系统关键类的类图

创建订单部分，关键代码：

@Override

public ResultDTO<Long> addOrder(DocOrderCreateDTO order) {

ResultDTO<Long> result = new ResultDTO<Long>();

if (!DocValidation.checkCreateDocOrderDO(result, order)) {

return result;

}

RpcCallInfo call = getCallInfo("NBOrderService#addOrder");

if (logger.isInfoEnabled()) {

logger.info("IN@NBOrderService.addOrder,order={}, callInfo={}", new Object[]{order, call});

}

try {

return orderManager.addOrder(order, call);

} catch (LogisticsException e) {

logger.error("error@NBOrderService.addOrder,order={}", new Object[]{order, e});

throw e;

} catch (Exception e) {

logger.error("error@NBOrderService.addOrder,order={}", new Object[]{order, e});

DocValidation.setFaildResult(result, ServiceErrorCodeEnum.S02);

}

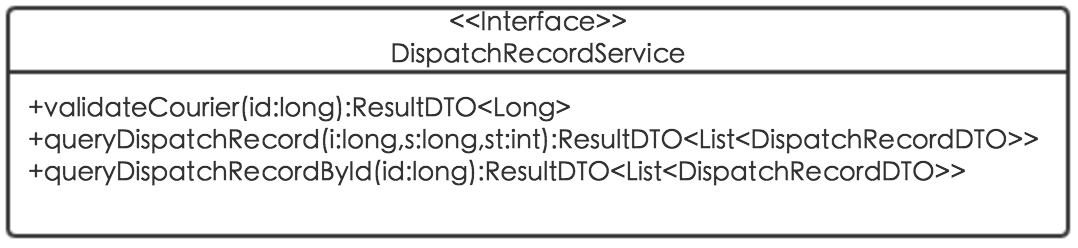
if (logger.isInfoEnabled()) {

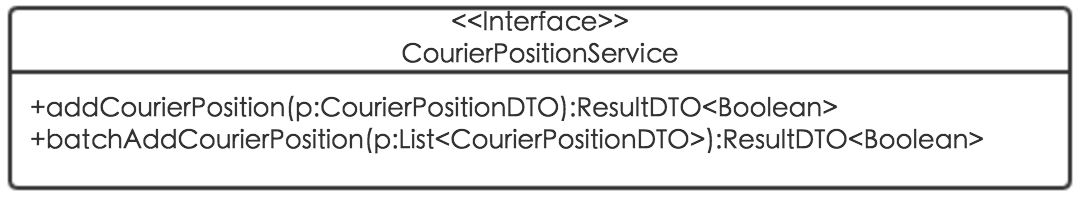
logger.info("success@NBOrderService.addOrder, result={}", new Object[]{result});

}

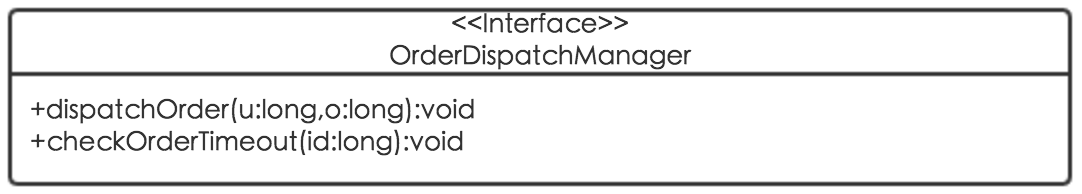
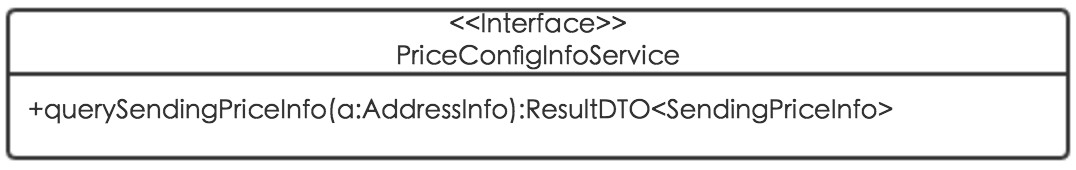
return result;

}

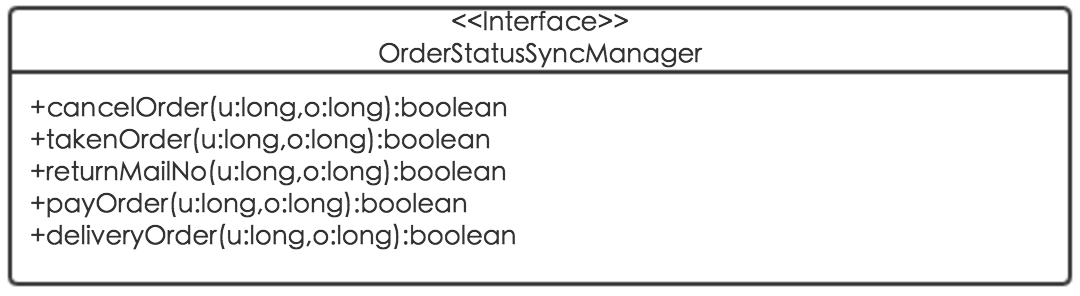
* 1.  寄件派单子系统

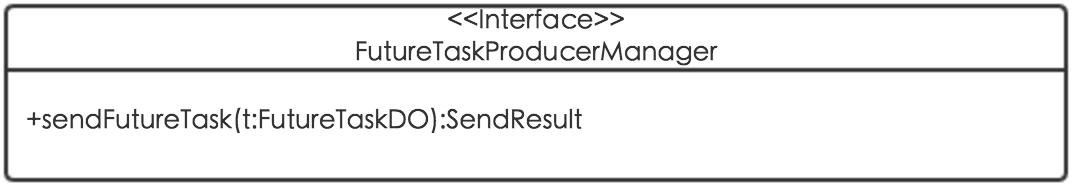
DispatchRecordService

CourierPositionService

PriceConfigInfoService

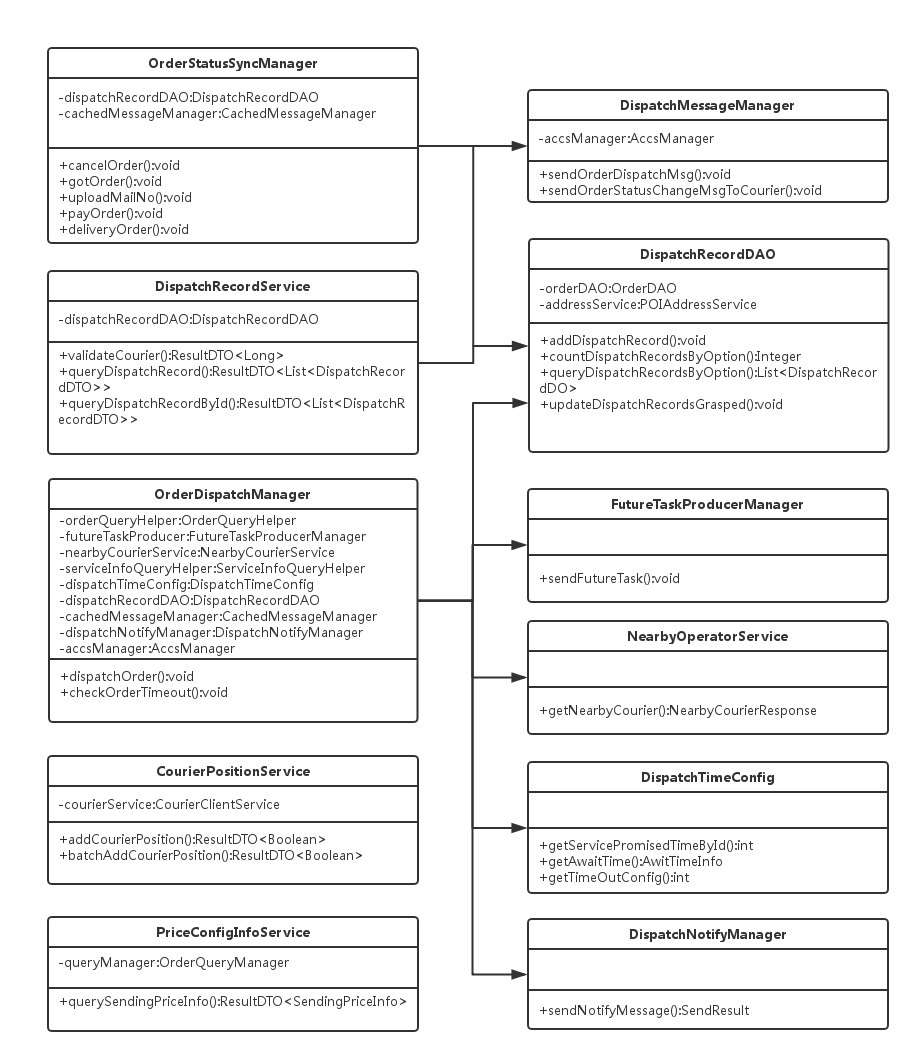
OrderDispatchManager

OrderStatusSyncManager



FutureTaskProducerManager

这些服务类依赖对应的持久化和管理类，或其他的服务类进行工作，部分类之间的依赖关系如下(类方法的参数从略)：

派单子系统关键类的类图

派单逻辑的关键代码如下：

@Override

public void dispatchOrder(long userId, long orderId) {

// 查询订单的信息

OrderDO orderDO = nbOrderQueryHelper.queryOrderDOByUserOrderId(userId, orderId);

// 判断订单是否可派送

boolean isDispatch = checkOrderCanDispatch(orderDO);

if (!isDispatch) {

logger.error("OrderDispatchManagerImpl.dispatchOrder checkOrderCanDispatch:" + isDispatch + " userId:" + userId + " orderId:" + orderId);

return;

}

DispatchDO dispatchDO = dispatchDOHelper.convertOrderDOToDispatchDO(orderDO);

//延迟5秒下发

createTimingDispatchTask(orderDO, DateUtils.addSeconds(new Date(),5));

return;

// 根据发件人地址查询附近的小件员

NearbyCourierResponse result = nearbyCourierService.getNearbyCouriers(dispatchDO.getSendDetailAddress(), dispatchDO.getPoi(), dispatchDO.getSenderLongitudeD(), dispatchDO.getSenderLatitudeD());

if (!result.isSuccess()) {

logger.error("nearbyCourierService fail!, sendAddress:" + dispatchDO.getSendDetailAddress()

+ " poi:" + dispatchDO.getPoi() + " LongitudeD: " + dispatchDO.getSenderLongitudeD()

+ " LatitudeD:" + dispatchDO.getSenderLatitudeD() + " errorMsg:" + result.getErrorMsg()

+ " orderId: " + orderId + " userId:" + userId);

}

// 派送列表

List<Long> dispatchList = new ArrayList<Long>();

// 消息

List<JSONObject> messages = new ArrayList<JSONObject>();

String message = sendAccsMsgToUser(orderId, userId, facilitatorName, MsgContentConstants.SERVER\_TYPE\_FACILITATOR, 0);

messages.add(JSON.parseObject(message));

dispatchUsers.add(facilitatorName);

dispatchList.add(facilitatorsDTO.getId());

if(messages != null && !messages.isEmpty()){

JSONArray array = new JSONArray(messages.size());

for(int i=0,size=messages.size();i<size;i++){

array.add(messages.get(i));

}

cachedMessageManager.putMessageByOrderId(

orderId,MsgTypeConsta nts.C\_ORDER\_DIAPATCHER,array.toJSONString()

);

}

if (dispatchList.size() > 0) {

List<Long> levelOne = new ArrayList<Long>(); // 第一批派送

List<Long> levelTwo = new ArrayList<Long>();// 第二批派送

// 进行分级别派送

for (Long dispatchId : dispatchList) {

// 判断是否要分批派单

if (awaitTimeConfig.isAwait()) {

createDispatchOrderTask(userId, dispatchId, dispatchDO);

} else {

dispatchOrderToUser(userId, dispatchId, dispatchDO);

}

}

// 启动定时检查任务,检查是否没人抢单

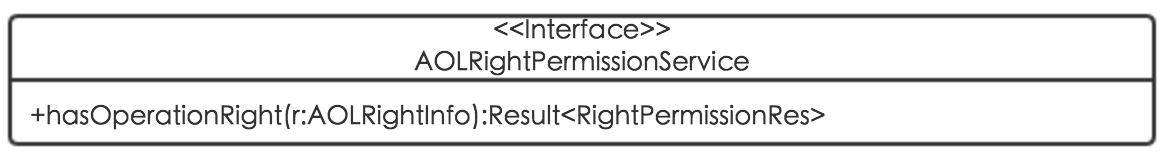
createCheckOrderDeliveryTask(userId, dispatchDO);

// 启动超时判断任务，检查订单是否超时，超时通知SDK

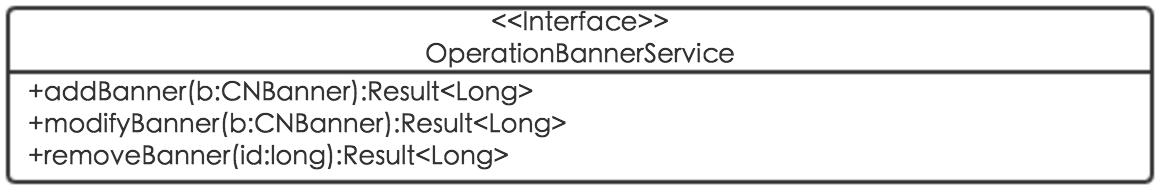
createCheckOrderTimeoutTask(orderId, userId);

}

}

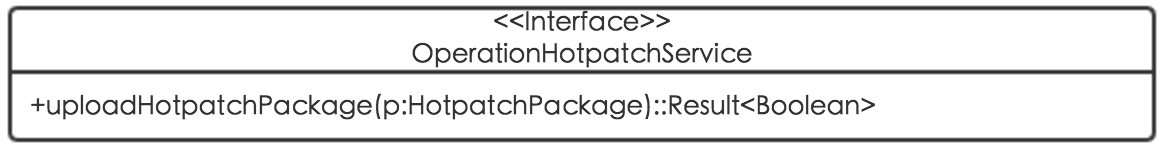
* 1.  寄件运营后台子系统

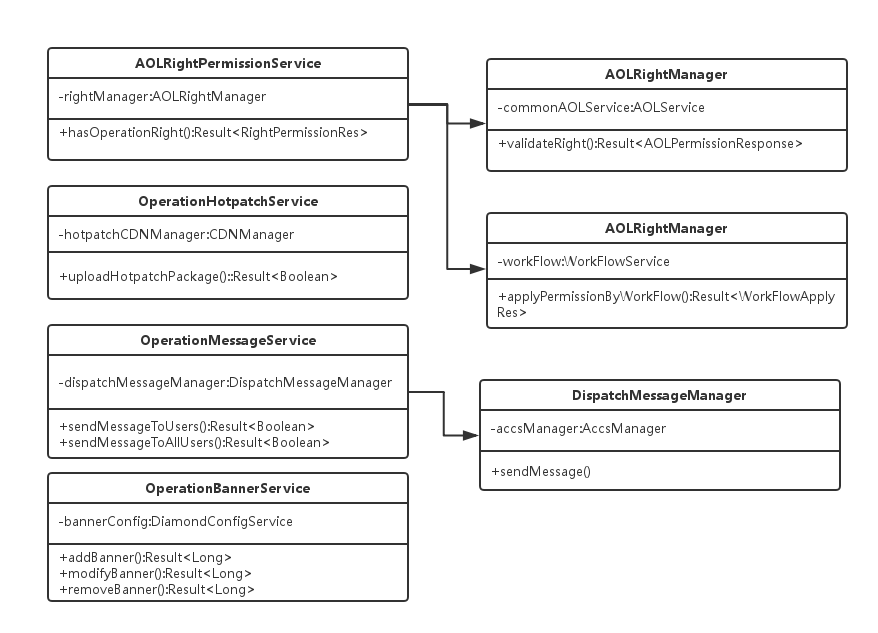
AOLRightPermissionService



OperationBannerService

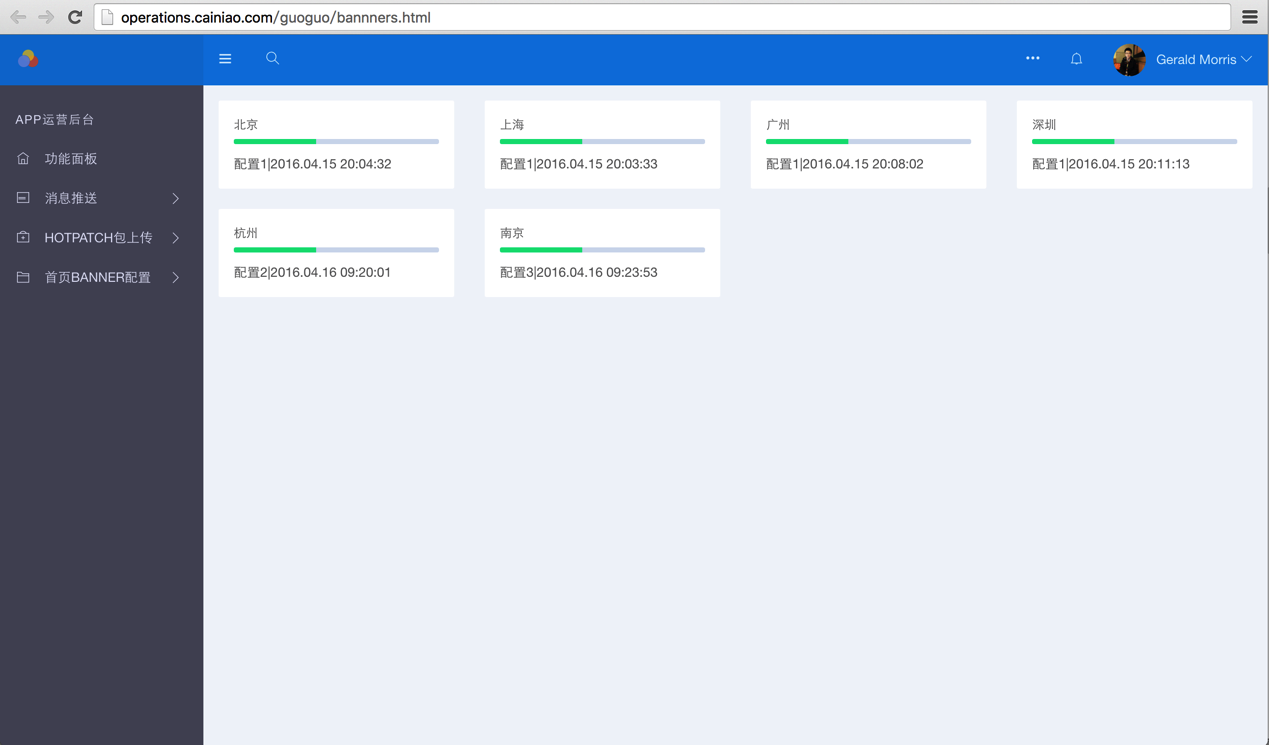
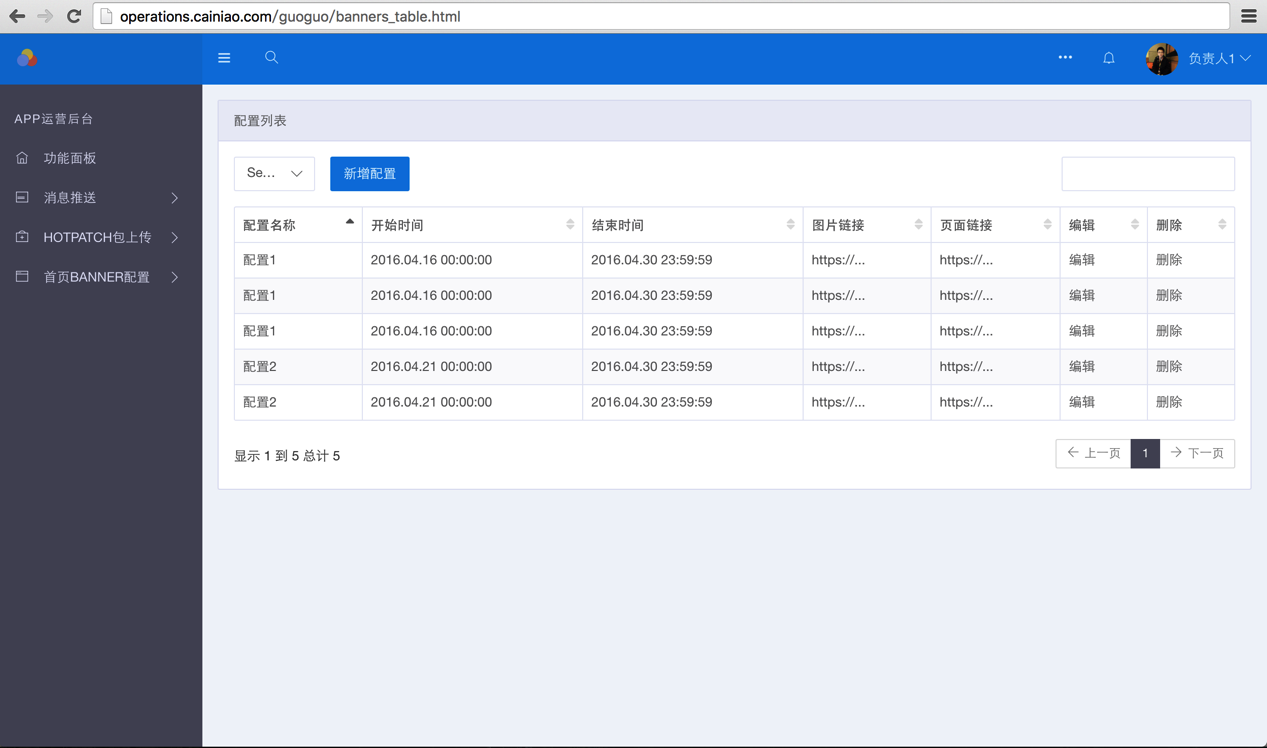
OperationMessageService

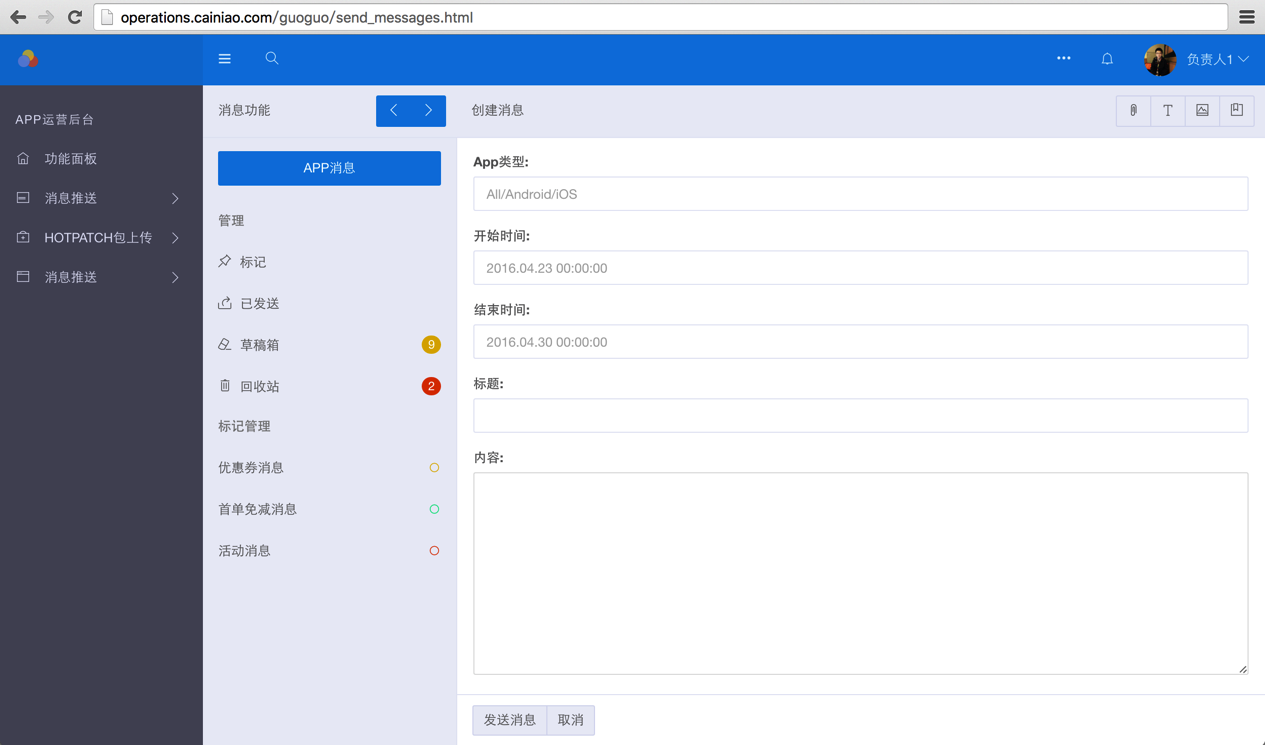
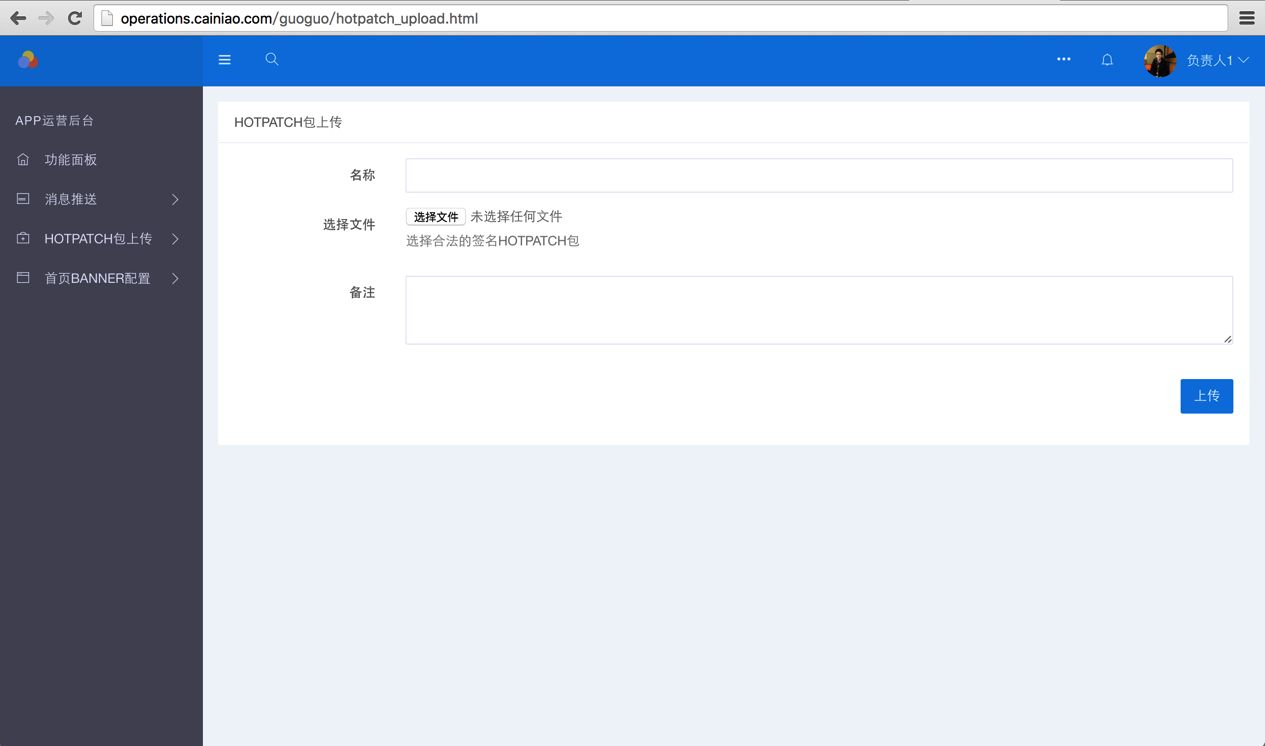
OperationHotpatchService

这些服务类依赖对应的管理器，或其他的服务类进行工作，部分类之间的依赖关系如下(类方法的参数从略)：

运营后台子系统关键类的类图

实现的效果如图：





* 1. 微服务管理的实现

（一）服务标准化定义实例

在每个技术平台（框架）上，对服务的定义形式不同，其中，在Spring中，对服务以Bean的形式进行定义，具体定义形式如下：

服务发布方（Producer）：

<bean id="senderServiceImpl" class="com.company.service.core.SenderServiceImpl

" />

<bean id="senderServiceProvider" class="com.taobao.hsf.app.spring.util.HSFSpring

ProviderBean" init-method="init">

<property name="serviceInterface">

<value>com.company.service.core.SenderService</value>

</property>

<property name="target">

<ref bean="com.company.service.core.SenderServiceImpl" />

</property>

<property name="serviceName">

<value>com.company.service.core.SenderService</value>

</property>

<property name="serviceVersion">

<value>1.0.0</value>

</property>

<property name="serviceGroup">

<value>HSF</value>

</property>

</bean>

服务调用方（Consumer）：

（二）Maven配置和Nexus信息实例

**第6章 寄件快递单处理系统的测试和分析**

* 1. 寄件快递单处理系统的测试方案
  2. 寄件快递单处理系统的测试用例
  3. 寄件订单系统功能测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 寄件订单子系统功能测试用例表 | | | | | |
| 功能点 | 用例说明 | 前置条件 | 输入 | 预期结果 | 测试结果 |
| 寄件服务信息查询 | 校验位置 | 用户已经登录 | 经度，纬度 | 位置校验结果 | 与预期一致 |
| 根据距离计算价格 | 用户已登录，通过区域位置验证 | 寄件人区域id，收件人区域id | 返回对应的寄件费用 | 与预期一致 |
| 服务信息 | 用户已登录，通过区域位置验证 | 经度，纬度 | 返回服务信息列表 | 与预期一致 |
| 获取具体位置信息 | 用户已经登录 | 经度，纬度 | 返回具体位置信息 | 与预期一致 |
| 寄件人订单操作 | 创建订单 | 用户已登录，通过区域位置验证 | 服务id，包裹类型，加价，寄件人和收件人位置和用户信息 | 返回订单id | 与预期一致 |
| 取消订单 | 用户已登录，订单处于已创建状态 | 订单id，取消原因 | 返回订单id | 与预期一致 |
| 重新创建订单 | 用户已登录，订单处于取消状态 | 订单id，重建类型 | 返回订单id | 与预期一致 |
| 计算订单支付金额 | 用户已登录，订单处于已接单状态 | 订单id，标准价格，寄件服务价格，优惠券id | 返回实际支付价格，优惠券实际使用价格 | 与预期一致 |
| 支付订单 | 用户已登录，订单处于已接单状态 | 订单id，支付类型，标准价格，寄件服务价格，优惠券id | 返回实际支付价格，订单编号，支付状态，支付时间 | 与预期一致 |
| 查询订单记录 | 用户已经登录 | 开始时间，结束时间 | 返回订单信息列表 | 与预期一致 |
| 查询订单详细信息 | 用户已经登录 | 订单id | 返回订单详情 | 与预期一致 |
| 更新订单的寄件人位置 | 用户已登录，订单处于已创建状态 | 经度，纬度 | 更新成功：订单id；失败：-1 | 与预期一致 |
| 快递员订单操作 | 验证快递员身份 | 用户录入快递员，已登录 | 登录id | 验证结果:成功 | 与预期一致 |
| 接单 | 用户已登录，通过快递员身份验证，订单处于已创建状态 | 订单id | 接单成功：订单id；接单失败：-1 | 与预期一致 |
| 揽收包裹 | 用户已登录，通过快递员身份验证，订单处于已接单状态 | 订单id，揽收码 | 成功：订单id；失败：-1 | 与预期一致 |
| 回传运单号 | 用户已登录，通过快递员身份验证，订单处于已揽收状态 | 订单id，运单编号 | 成功：订单id；失败：-1 | 与预期一致 |
| 验证到达寄件人位置 | 用户已登录，通过快递员身份验证，订单处于已接单位置 | 经度，纬度，订单id | 是否在区域内 | 与预期一致 |

* 1. 寄件派单系统功能测试用例

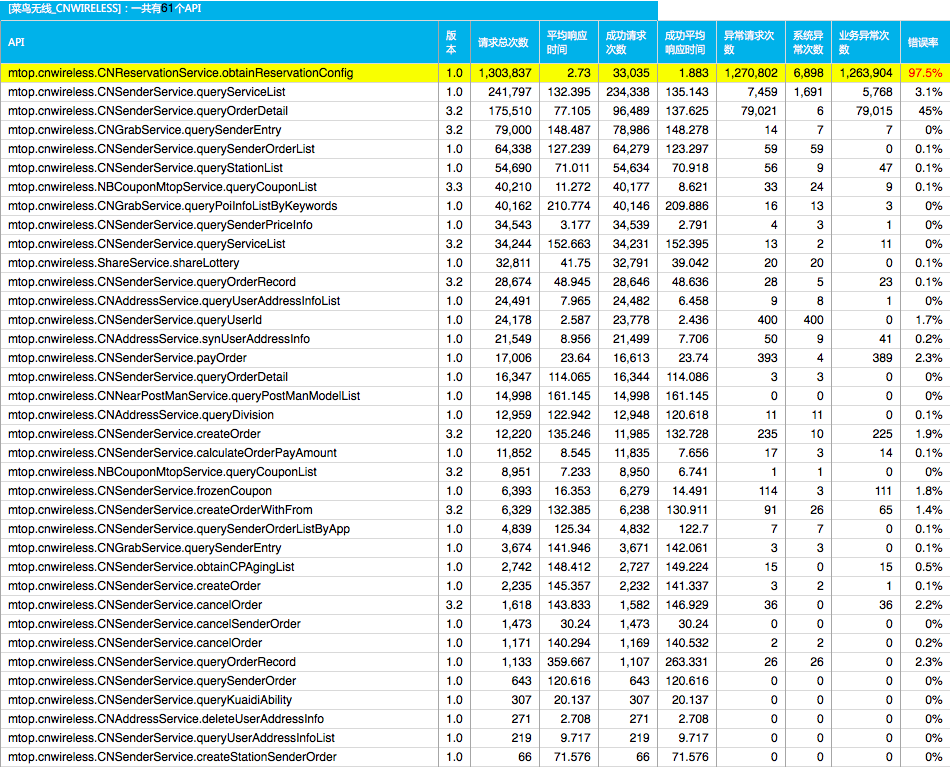
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 寄件派单子系统功能测试用例表 | | | | | |
| 功能点 | 用例说明 | 前置条件 | 输入 | 预期结果 | 测试结果 |
| 派单信息查询 | 验证快递员身份 | 用户录入快递员，已登录 | 登录id | 验证结果:成功 | 与预期一致 |
| 派单记录信息 | 用户已登录，通过快递员身份验证 | 页码，页大小，派单状态 | 派单记录列表 | 与预期一致 |
| 派单记录详情 | 用户已登录，通过快递员身份验证 | 派单记录id | 单条派单记录详情 | 与预期一致 |
| 位置服务 | 上传位置信息 | 用户已登录，且为注册快递员 | 登录id，经度，纬度，apk信息 | 上传成功/失败 | 与预期一致 |
| 寄件信息查询 | 寄件费用 | 用户已登录，且为注册快递员 | 区域id | 首重价格，续重价格 | 与预期一致 |
| 短信服务 | 发送短信 | 用户已登录，且为注册快递员 | 派单id，消息内容 | 发送结果:成功 | 与预期一致 |
| 获取派单短信消息记录 | 用户已登录，且为注册快递员 | 派单id | 获得短信消息列表 | 与预期一致 |

* 1. 寄件运营后台功能测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 寄件运营后台子系统功能测试用例表 | | | | | |
| 功能点 | 用例说明 | 前置条件 | 输入 | 预期结果 | 测试结果 |
| 权限验证 | 操作权限 | 操作用户已登录 | 页面路径，用户id | 正常访问页面，或被禁止访问 | 与预期一致 |
| 消息推送 | 向特定用户推送 | 用户录入快递员，通过权限验证 | 目的用户id，消息标题，消息内容 | 目标用户app接收到消息推送 | 与预期一致 |
| 向所有用户推送 | 用户录入快递员，通过权限验证 | 消息标题，消息内容 | 目标用户app接收到消息推送 | 与预期一致 |
| Hotpatch包上传 | 上传补丁包 | 用户录入快递员，通过权限验证 | 上传任务名称，包文件，备注 | 上传成功/失败 | 与预期一致 |
| 首页Banner配置 | 增加配置 | 用户录入快递员，通过权限验证 | 配置名称，开始、结束时间，图片链接 | 成功：配置id，失败：-1 | 与预期一致 |
| 编辑配置 | 用户录入快递员，通过权限验证 | 配置id，配置名称，开始、结束时间，图片链接 | 成功：配置id，失败：-1 | 与预期一致 |
| 删除配置 | 用户录入快递员，通过权限验证 | 配置id | 成功：配置id，失败：-1 | 与预期一致 |

* 1. 线上API数据统计

目前上线后的寄件快递单处理系统，对所有的服务进行管理，并统计了线上的服务的api调用数据，其中某一天的具体的统计数据记录如图：



**第7章 总结和展望**

* 1. 用户角色分析

**参考文献**

注释正文：宋体10.5磅(英文用Times New Roman体10.5磅)，行距20磅，段前段后0磅，续行缩进两个字符左对齐

标题：黑体16磅加粗居中，行距20磅，段前24磅，段后18磅

[1] 中国互联网络信息中心CNNIC. 2014年中国网络购物市场研究报告[R].

Beijing: CNNIC, 2015.

[2] 中国互联网络信息中心CNNIC. 第37次中国互联网络发展状况统计报告[R]. Beijing: CNNIC, 2016.

[3] 国家邮政局. 中国快递发展指数首次发布[N/OL]. [2015-03-30]. http://www.mot.gov.cn/jiaotongyaowen/201510/t20151014\_1897390.html.

[4] 张意轩. 2014年中国快递业务量达140亿件 跃居世界第一[N]. 人民日报海外版, 2015-01-07.

[5] 国家邮政局. 国家邮政局公布2015年邮政行业运行情况[N/OL]. [2016-01-14]. http://www.spb.gov.cn/dtxx\_15079/201601/t20160114\_710673.html.

[6] 郭雪红. 菜鸟网络打通县村物流 农村淘宝实现100%覆盖[N/OL]. [2015-07-09]. http://hzdaily.hangzhou.com.cn/mrsb/html/2015-07/09/content\_2010946.htm.

[7] 国家邮政局. 国家邮政局公布2014年邮政行业运行情况[N/OL]. [2015-02-24].

http://www.spb.gov.cn/yzshjd/ywgl/201503/t20150324\_437767.html.

[8] 国家邮政局. 国家邮政局公布2013年邮政行业运行情况[N/OL]. [2014-01-15]. http://www.spb.gov.cn/dtxx\_15079/201401/t20140115\_274540.html.

[9] 国家邮政局. 国家邮政局公布2012年邮政行业运行情况[N/OL]. [2013-01-16]. http://www.spb.gov.cn/dtxx\_15079/201301/t20130116\_158179.html.

[10] 国家邮政局. 国家邮政局公布2011年邮政行业运行情况[N/OL]. [2012-01-20]. http://www.spb.gov.cn/dtxx\_15079/201201/t20120120\_156963.html.

[11] 国家邮政局. 国家邮政局公布2010年邮政行业运行情况[N/OL]. [2011-01-25]. http://www.spb.gov.cn/xytj/tjxx/201101/t20110125\_159862.html.

[12] Martin L. Abbott, Michael T. Fisher. The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise[M]. 2nd Edition. Boston: Addison-Wesley Professional, 2015.

[13] Thijmen de Gooijer, Anton Jansen, Heiko Koziolek, et al. An Industrial Case Study of Performance and Cost Design Space Exploration[C]// Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. New York: ACM, 2012: 205-216.

[14] Chris Richardson. Pattern: Microservices Architecture[N/OL]. [2014-05-28]. http://microservices.io/patterns/microservices.html.

[15] Shahir Daya, Nguyen Van Duy, Kameswara Eati, et al. Microservices from Theory to Practice[N/OL]. [2015-08-03]. http://ibm.com/redbooks.

[16] Jose Ignacio Fern, Carlos A. Iglesias, Mercedes Garijo. Microservices: Lightweight Service Descriptions For Rest Architectural Style[C]// International Conference on Agents and Artificial Intelligence (ICAART). Valencia: INSTICC Press, 2010: 513-516.

[17] Vinh D. Le, Melanie M. Neff, Royal V. Stewart, et al. Microservice-based Architecture for the NRDC[C]// Industrial Informatics (INDIN). Cambridge: 2015 IEEE 13th International Conference on, 2015: 1655-1664.

[18] 马海收，吴振新. 微服务在数字资源长期保存系统中的发展研究[C]// 2011图书馆信息技术的应用、服务和创新学术研讨会暨第3届数字图书馆与开放源代码软件(DLIB&OSS2011)学术研讨会. 北京：图书馆学研究，2011：237-244.

[19] 杨霄彩. 基于J2EE的订单跟踪微架构研究与应用[J]. 计算机与数字工程，2010，38(4)：164-166.

[20] 张冀峰. 基于Spring框架的供应链订单管理系统的设计与实现[D]. 天津：南开大学，2014.

[21] 周会强. 基于Struts框架的Java Web应用开发研究[J]. 科技通报, 2012, 28(6):36-37.

[22] Nielet D‘mello, Larkins Carvalho. Struts 2 - The modern web application framework[J]. International Journal of Modern Engineering Research, 2013, 3(3):1854-1856.

[23] 顾海振. 基于Spring、Struts和Hibernate整合应用的货物管理系统[D]. 山东：山东大学， 2014.

[24] 张文宇, 许明健, 薛昱. 论spring的零配置与XML配置[J]. 计算机系统应用, 2015, 24(2):270-275.

[25] 范为. 基于Spring框架面向多业务Web平台开发框架研究[J]. 现代计算机, 2012, 12(23):131-133.

[26] 何思平，方美琪. Spring AOP技术在电子商务中的应用[C]// 2008年IT服务促进企业信息研讨会论文集. 北京：信息化纵横，2008：32-34.

[27] Ankur Bawiskar, Prashant Sawant, Vinayak Kankate, B.B. Meshram. Spring Framework: A Companion to JavaEE[J]. International Journal of Computational Engineering & Management, 2012, 15(3):41-49.

[28] Michael A. Rivalto. System for automated package pick-up and delivery: US, 6,690,997[P]. 2001-09-13.

[29] Ole-Petter Skaaksrud, Cameron Dee Dryden, Jeffrey Robert Smith, et al. Internet-based shipping, tracking, and delivery network supporting a plurality of mobile digital image capture and processing (MICAP) systems: US, 7,870,999[P].

[30] 赵园园. 电子商务环境下社区智能快递系统助力快递末端配送效率提升[J]. 物流技术，2015, 34(1):158-160.

[31] 许校境，郑召文. 基于RFID的快递系统的研究与应用[J]. 无线通信技术，2013，14(4):57-60.

[32] 武防震，姚国祥. 基于UML的快递系统建模[J]. 微机发展，2003.13(3):123-125.

[33] 李刚. 轻量级Java EE企业应用实战[M]. 北京：电子工业出版社. 2008.

[34] Villamizar M., Garces O.; Castro H., el al. Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud: proceedings of the 10th Computing Colombian Conference, Bogota, Colombia, Sept 21-25, 2015[C]. Los Alamitos: IEEE Computer Society, 2015.

[35] 邢帆. 微服务 On Line[J]. 中国信息化，2015.12(6):76-77.

**附录1 ×××××**

黑体16磅加粗居中，单倍行距，段前24磅，段后18磅，附录二字与题名间空一个汉字符位

……

正文部分：宋体12磅（英文用Times New Roman体12磅），两端对齐书写，段落首行左缩进2个汉字符。行距20磅（段落中有数学表达式时，可根据需要设置该段的行距），段前0磅，段后0磅。

**致 谢**

宋体12磅，行距20磅，段前段后0

二字间空一个汉字符，黑体16磅加粗居中，单倍行距，段前24磅，段后18磅

在论文完成、即将毕业之际，感触很深。中国科学技术大学软件学院，在我求学期间，给我提供了优秀的学习实践环境，老师们的殷勤教诲，也给我提供了很大帮助。在企业实习期间，有幸接触到互联网实践项目，也给我的实践能力，带来了很大提升。薛美盛老师，作为我的校内导师，给我的论文撰写提供了很大帮助，极富耐心和很有针对性地指出了我的论文多方面的问题。对于在论文撰写期间，薛美盛导师和其他帮助过我的学院老师们，再次表示衷心的感谢。

感谢学校同学，无论在课堂还是实验室，和他们的相处都倍感融洽和轻松，在和他们的讨论中，我的知识面得到了拓展，对学问的认识也深化了不少，和他们在一起，我的学习能力得到了很大提升。

最后再感谢我的父母和亲人们，是他们在我求学期间，在我背后，默默支持和鼓励着我前进，给我很大的精神动力，谢谢你们。

2015年6月