杭州电子科技大学

硕 士 学 位 论 文

题　　目：基于 Java 的电商软件开发与大数据研究

研 究 生　欧盛彪

专　　业　电子与通信工程

指导教师　李金新　副教授

完成日期　2018 年 03 月

杭州电子科技大学硕士学位论文

基于 java 的电商软件开发与大数据研究

研 究 生： 欧盛彪

指导教师： 李金新　副教授

2018 年 03 月

Dissertation Submitted to Hangzhou Dianzi University

for the Degree of Master

Research of the E-commerce Software Development and Big Data Based on Java

Candidate: Ou Shengbiao

Supervisor: Associate Prof. Li Jinxin

March，2018

杭州电子科技大学

学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明： 所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：欧益彪 日期：2018年 3月 18日

学位论文使用授权说明

本人完全了解杭州电子科技大学关于保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属杭州电子科技大学。本人保证毕业离校后，发表论文或使用论文工作成果时署名单位仍然为杭州电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。（保密论文在解密后遵守此规定）

论文作者签名：欧益彪 日期：2018年 3月 18日

指导教师签名：李金新 日期：2018年 3月 18日

摘　要

随着移动互联网技术的成熟和智能手机等移动终端的普及，人们逐渐适应了移动应用的互联网接入方式，其中的电商 app 在最近 10 年方兴未艾，例如淘宝、网易考拉海购、网易严选、唯品会等，所有的这些平台也让人们养成了网上购物的习惯，甚至还出现双 11 这样的购物狂欢节等，其中散发出的商业气息之巨大，从各大公司的电商网站就可见一斑，正因为以电商为代表的互联网的迅速发展，其产生的数据量更是不可估计，所以对其数据进行有效的利用和分析就显得尤为重要，在此大量数据基础上，又出现了一个巨大的市场空间，几乎所有的大型互联网公司都看到了这块巨大的蛋糕，许多公司都已经开始布局大数据。

本文以模拟“严选”电商为例，在深入理解电商后台、服务器接口和 android客户端的基础上，借助于大数据分析和统计，完成了以下4个方面的工作：

1.优惠券与用户标签配置后台的软件开发：配置后台可以实现电商优惠券和用户标签的插入、删除、修改、查询与接口的测试等；

2.基于优惠券和用户标签特定需求的软件接口：优惠券和用户标签相关的接口开发；

3.优惠券android客户端的APP开发：android客户端的交互（注册、登录、领券、展示我的优惠券）；

4.基于大量用户数据，完成了大数据清洗：并对用户兑换优惠券时产生的日志数据进行领券张数的统计。

创新点如下：

1. 通过MR分析日志数据，不断优化优惠券的配置，提高发券效率。

2. 系统化后台、接口、安卓、大数据，方便用户使用。

关键词：java，android，电商，软件开发，大数据

ABSTRACT

With the maturity of mobile Internet technology and the popularity of mobile terminals, such as smart phones, people have gradually adapted to the Internet access mode of mobile applications, Launched mobile APP corresponding to the mobile Internet market to occupy a certain space, in which electricity supplier app in the last 10 years is the rapid development, such as Taobao, NetEase koala sea purchase selected, NetEase yanxuan,Vipshop and so on, all of these platforms also allow people to develop online shopping habits, and even such a double 11 shopping carnival, which sends out the commercial atmosphere of the great, is remarkable from the company's website, because of the rapid development of the Internet as the representative of the electricity supplier, the amount of data generated is unpredictable, so it is very important use and Analysis on the effective data, this data base, and the emergence of a huge market space, almost all of the large internet company see this huge cake, many companies have started to lay out big data.

In this paper, based on the in-depth understanding of the business background, server interface and Android client, based on the analysis of the big data analysis and statistics, we have completed the following 4 aspects:

1. the software development of the coupons and user tags configuration background: the configuration background can realize the insertion, deletion, modification, query and interface test of e-commerce coupons and user tags;

2. software interfaces based on coupons and user tags specific requirements: interface development related to coupons and user tags;

3. coupon Android client APP development: Android client interaction (registration, login, voucher, show my coupon);

4. based on a large number of user data, large data cleaning is completed: the number of vouchers is counted for the user's log data generated when the coupon is exchanged.

The innovation points are as follows:

1. analyze the log data through MR, optimize the configuration of coupons, and improve the efficiency of issuing coupons.

2. systematized background, interface, Android, large data, convenient for users to use.

Keywords: java, android, e-commerce, software development, large data

目　　录

第一章 绪论

1.1 课题研究背景和意义

“互联网”已经成为现代智慧社会的一个十分流行的热词，而电商又是互联网行业中的代表词汇。在中国主要门户网站都开始转向移动终端，并且相应的移动应用程序都像雨后春笋般的隐现出来，目的就是在移动网络市场这块巨大蛋糕中分一杯羹[1]。官方数据显示：2015年我国的进出口交易总规模稳居世界第一，达到了30.7万亿元人民币。而其中不得不重视的是，跨境电商在这一年中总的交易规模达5.4万亿元之多，比去年同期增长28%以上,电商跨境进口交易规模和跨境出口交易规模分别达到了9072万亿和4.49万亿元人民币[2]。“电商”就是把“互联网”的概念引入人们的社会生活中，尤其是引入人们的日常购物生活中，移动互联网中的电商让人们随时随地享受购物的乐趣，不但可以节省人们大量的时间，也把所有商家暴露在一个共同的环境下，对消费者来说 ，这也是一个性价比很高的购物空间，也正因为如此，电商正成为人们越来越喜欢的购物方式。

在全面实现小康社会的进程中，现代社会发展迅速，人们的生活水平也逐年提高，我们对物质生活和精神生活的要求也在跟着提高，对人们的服务要求已经从最早时期的解决温饱的基本生活需求变成健康、购物、基础设施、精神享受等全方位人性化的服务需求，这个需求不但要求更广泛，而且每个方面的质量要求也越来越高，对电商行业来说，像淘宝这种国内电商已经无法满足一些人的需求，所以，近年来，一些国际电商随着消费升级也迅速的兴起，比如天猫国际和网易考拉海购等。正因为互联网在近年来是如此的火爆，而电商又是互联网行业中非常重要的组成部分，所以对电商进行分析与研究，其实就是对整个互联网行业进行研究与梳理，而经过这十几年来的发展，以电商为代表的互联网行业已是一个庞然大物，因此，对它进行研究就显得尤为重要，也是极有意义的一项工作。

一方面，消费升级还在进行时，跨境电商的发展也还在路上，随着经济全球化的发展，跨境电商颠覆了传统的进出口模式，在一定程度上对我国的经济有促进作用，所以，政府一定会解决跨境电商中存在的问题，实现其快速发展[3]。而且互联网对各个行业的融合也还在路上，比如对传统银行业，在马宇华看来，他是中国招商银行行长，今天的世界是一个互联网时代正在到来的世界，在互联网时代，当全社会都有意识去创新的时候，创新在哪个时候就没有了多大的价值。因此，对金融业而言，金融业的互联网必须越早越好。只有这样，我们才可能带领我们整个行业甚至是整个社会进行互联网变革。所有的这些影响、企业的变革，都需要大量的软件开发人才来做支撑。

另一方面，目前,对大数据的定义还没有一个确定的指标,对单一数据集而言，数据规模范围从几十TB到数PB不等,虽然不同的研究者对大数据的定义指标各不相同,但他们都认为

相对以前数据量将急剧增长，也就是说要具有规模性[4]。IDC 作为中国知名互联网内容提供商，其中一份研究报告这样写道，“在未来的十年里，各家企业管理的数据信息之和将是现在的五十倍，同时全球的服务器数量也是现在的十倍，对于企业数据中心来说需要处理的文件数保守估计也是现在的七十五倍以上，而全球具备 IT 专业技术知识的人员数量只能增加现在人员数量的一点五倍左右。因此，未来十年里，进行大数据数据处理的人才需求和具备相应专业知识的人才供应之间将会出现十分不平衡的现象[5]。目前，由于国内外开设大数据相关知识课程的时间还比较短，特别对国内来说，近几年大数据才慢慢火爆起来，在技术上精通大数据处理技术的人少之又少，因此，对于企业而言，非常渴望具备该领域的技术人员得到足够的供给，甚至其对应岗位的薪资已经是普通程序员的 3 到 5 倍。在未来，面对这样的一个缺口，国内的互联网企业想做大做强，我相信大数据人才的争夺将是其中的一个重要环节[6]。

1.2 国内外当前研究现状和发展趋势

1.2.1 国内电商与大数据的现状与发展

我国的城市化经历三十多年的发展，互联网经历二十几年的发展，在许多领域中，移动互联网都有应用，但是将互联网应用于电商方面，虽然说已经也发展了很多年，但是也可以说才刚开始起步，因为人们的购物需求从刚开始到现在已经发生了翻天覆地的变化，例如从国内走到国外，从日常的生活用品到大宗生活家电等，都在不断的变化，而其他领域的信息化管理服务经验特别是社交软件的发展也给电商提供了很好的借鉴意义。现在的社会是一个信息时代，在二十一世纪，以信息为代表的服务业在三大产业中所占的比重正越来越大，而电商在这个时代正有着指数式的生长和发展。而配合着网络飞速发展的现状，人类生活已经进入了“电商时代”，尤其是在近年来“大数据时代”的背景下，电商产业的发展迎来了一个新的高峰。就像现在的社会一样，社会在消费升级，电商也是一样在消费升级，现在的基于大数据出现的精准营销，更是在电商行业受到了极大的欢迎和追捧，大大的提高了电商公司的营销效率，一方面公司在追求软件开发的技术创新之外，另一方面也利用大数据工具在对已有的公司数据进行分析和提取，来为公司赚取更大的商业利润的同时，也对用户体验极差的商品说不。

在这个“电商时代”，手机购物更是越来越普遍，数据显示 2013 年移动端占比还只有 10%，一年后移动端占比就达到了 40%，增长了整整 3 倍，2015 年当时预测能占比在 60%-80% 左右[7]。另一方面电商所呈现出来的巨大商业利润也在逐年的提高，双 11 电商节的成交额也是逐年刷新记录，仅天猫双 11 一家电商平台在 2016 年双 11 购物狂欢节全天交易额就突破千亿元，达到 1207 亿元，可以预计的是，这个数字在今后的几年每年被刷新的可能性也是极大的，这个蛋糕会越来越大，同时这个行业的竞争也会越来越大。近年来，跨境电子商务在我国也发展十分迅速，其也受到了各个行业和社会各界的高度关注。我国为了跨境电子商务更好地发展，中国政府也制定出了相关的政策和法令，不仅对跨境电子商务的跨境发展重视

非凡，而且也积极营造良好的发展环境，努力引导各行各业都积极参与。跨境电子商务的迅速成长，慢慢改变了以前传统的国际贸易形式，毋庸置疑，对我国国际贸易的向前发展也有一定程度的促进作用。

1.2.2 国外电商与大数据的现状与发展

在欧美等发达国家，移动互联网技术应用到城市化进程方面领先我国，无论是在社会法令税收还是在社会实践上，这些都比我国要成熟，美国已经有了全球性成熟的电商系统和大数据技术，比如亚马逊，它已经成为了全球知名电商，其在当前移动互联网技术蓬勃发展的社会环境下，亚马逊又出现了许多创新性的研究和应用，因此在这些方面，亚马逊的电商发展之路有很大的借鉴意义，利用互联网技术，将社会的很多资源连接到一个平台上面来，并利用大数据工具监测、分析和整合各种数据，进而智能化地响应每一个市民的个性化需求，这就体现了聪明的零售商将更加注重相关的数据收集工作，让用户变得更加透明，而且在一定程度上降低了公司的运行成本，特别是智能仓库这一块，是我们国家的互联网值得借鉴的，这在未来的发展也是一片大好，必然相对应的人才需求也会增长迅速[8]。

1.3 本文的章节安排

整篇论文我们以模拟“严选”电商为例，可分为四部分：基于 java 的电商后台开发（以优惠券和用户标签为例）、接口开发（后台接口和安卓客户端接口）、Android 客户端开发（以优惠券为例）、简单理解基于 java 的大数据执行流程（以优惠券次数统计为例），具体章节安排如下：

(1) 第一章，绪论。首先介绍了基于 java 的电商软件开发与大数据研究的意义与背景，对比了国内外现状与发展，并介绍了研究课题的章节安排。

(2) 第二章，系统的需求分析及总体架构设计。根据现在互联网中电商的实际情况，提出了设计和开发需求，并用图和字结合的方式从总体上进行了理解，最后得到系统的总体设计框架和设计思路。

(3) 第三章，电商后台设计。介绍了电商后台的设计方案与数据库的设计和选择，后台框架原理与 web 服务器的选择，及服务器与数据库和浏览器客户端与服务器数据交互设计原理等。

(4) 第四章，电商后台的具体实现，即服务器及 Web 页面开发。结合数据库开发电商公司运营人员使用的优惠券和用户标签后台，及后台各模块进行了详细的解析，可以接受并且处理浏览器客户端发送的请求，并且把处理结果信息响应给浏览器客户端。

(5) 第五章，基于 java 的安卓客户端与大数据的研究。本章大致阐述了基于 java 的安卓客户端的相关知识（以电商优惠券为例）和大数据之计算、存储、数据库。并以优惠券的次数统计为例对大数据的应用进行了简单描述。

(6) 第六章，调试结果和系统整体工作流程。整套软件系统开发完成后，对所有功能进

行了整体的测试。通过对系统的测试能发现可能存在的内存泄漏和功能上的不足，通过手工和自动化测试，找到 bug 后进行修改，为使用该平台的用户提供更好的体验。对服务器的测试也是软件开发中重要的一环，得到相应测试数据后，有效对数据进行分析和评估，最后用图片的形式把各个功能展示出来。

(7) 第七章，总结与展望。对本系统设计进行了总结，并结合本文的内容和写该论文的写作过程中，总结了系统设计中存在的不足和可以进行改进的方向，并对今后的软件开发可以在哪些方面可以展望进行了罗列。

1.4 本章小结

本章简单的介绍了论文研究课题，最先我介绍了课题研究背景和意义，紧接着，分别从国内和国外两个方面，对电商与大数据的现状和发展进行了简要介绍，得出基于 java 的电商软件开发与大数据研究在现代社会中的意义，接着简要说明了我的研究课题的主要工作，大致阐述了全文的各个章节和其讲述的大致内容。

第二章 系统需求分析及总体设计

2.1 系统需求分析

本文第一章中阐述了现阶段国内外互联网电商的发展情况，在跨境电商这一块还有很大的需求，在消费升级的大背景下，以前的普通电商已经没法满足人们对品质的不断追求，即品质电商这块做的还不够好。本论文研究的是基于 java 的电商软件开发与大数据研究，其目的就是:不但可以让普通公民方便、快捷地享受购物的同时，也能利用大数据工具，提高电商平台的品质。因此，在设计本系统的时候我们不但要促进人们的购物欲，更要在对人们购物产生的数据进行分析处理，研究人们的心理和行为需求，从而了解人们最需要的是什么，只有这样才能使电商的服务能够真正方便广大人们的购物；而且能对评分很差的水货商品说不，更加的提升人们对电商的信任感和自信心。另外，本章对性能也做了简要需求分析，这样来保证整个系统能正常运行。

2.1.1 系统功能需求分析

有品质的电子商务，无非就是为广大网民消费者提供一个舒适、快捷、有品质的服务，使他们能以高的性价比买到他们自己想要的商品，这是广大网民消费者所关注的最基础的问题和需求。而对于电商开发者来说，需要及时得到广大网民消费者的真实需求，而对于公司运营人员来说，就要快而准的提供相应服务，所以整个电商系统开发就显得尤为重要，下面就是不同的人员属性需要做的具体事情：

对于软件开发人员，需要为广大网络用户和公司运营人员提供技术支持，进而使电商平台能正常的运行，比如安卓客户端、服务器接口和后台的开发等

对于数据开发人员，对于平时的日志数据等进行清洗、提取、分析等工作，为电商平台的未来发展提供依据。

对于运营人员，为用户提供运营需求，比如优惠券和用户标签的配置，进行一些促销活动等。

根据以上几点，本文提出以下几点技术和功能，系统总体流程图，如图 2.1 所示。

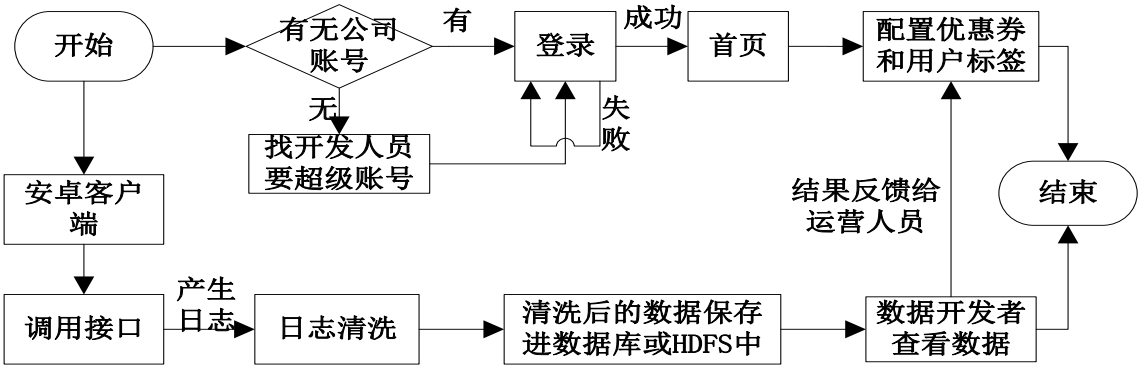


图 2.1 系统总体流程图

(1)浏览器 http 通信：利用计算机和网络的发展，包括局域网网速的显著提高，因此，在电商公司内部，使用浏览器 http 通信不但可以不安装客户端软件，而且还能够节省很多时间成本。

(2)框架开发：利用现有框架，可以大大提高开发效率，降低开发成本。

(3)数据开发：通过对数据进行分析，运营人员不但可以营销，而且还可以进行精准营销。

(4)配置功能：通过 Web 页面，运营人员可以配置不同的优惠券和用户标签。

(5)反馈功能：通过数据清洗和提取，可以得到有价值的反馈数据，进而指导电商往更好的方向发展，更加迎合广大消费者的喜爱。

为了实现以上的各个具体功能，按人员属性来分，可以将该系统分成三个部分，各部分的具体实现功能和指标如下：

(1)软件开发人员开发的各功能及相应的实现图，如图 2.2 所示。

● 登录功能；

●电商优惠券的增删改查（优惠券编码、激活码、生效日期、失效日期等，作用就是搞各种活动时来进行促销的）；

●电商用户标签的增删改查（给用户打上相对应的标签属性，主要作用是区分不同属性的用户）；

●开发有关优惠券和用户标签接口和安卓客户端，用来判断用户是否能领券和拥有哪些标签等。

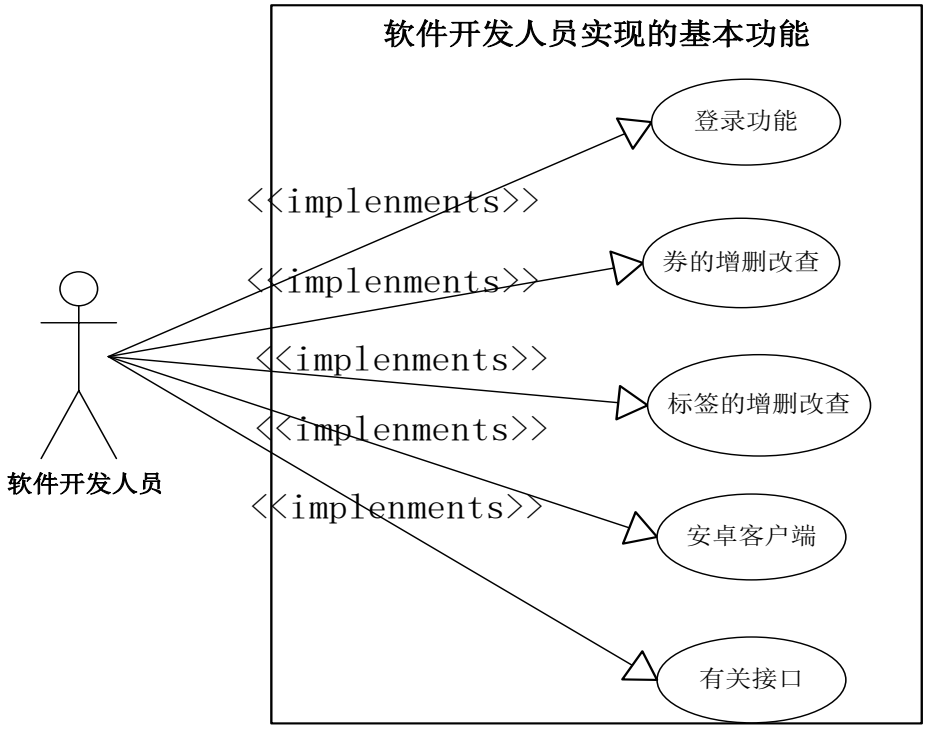


图 2.2 软件开发人员功能实现图

(2) 数据开发人员开发的各功能及相应的实现图，如图 2.3 所示。

●对于数据开发者，对于平时的日志数据，利用 Hadoop 中的 MapReduce 和 HDFS 等大数据工具对数据进行清洗，提取工作；

●清洗之后的数据要进行维护，存放在数据中心，例如 hive 等数据库，也可以直接存放在分布式文件系统 HDFS 中，一部分重要数据还可以反馈给运营人员，例如 kpi。

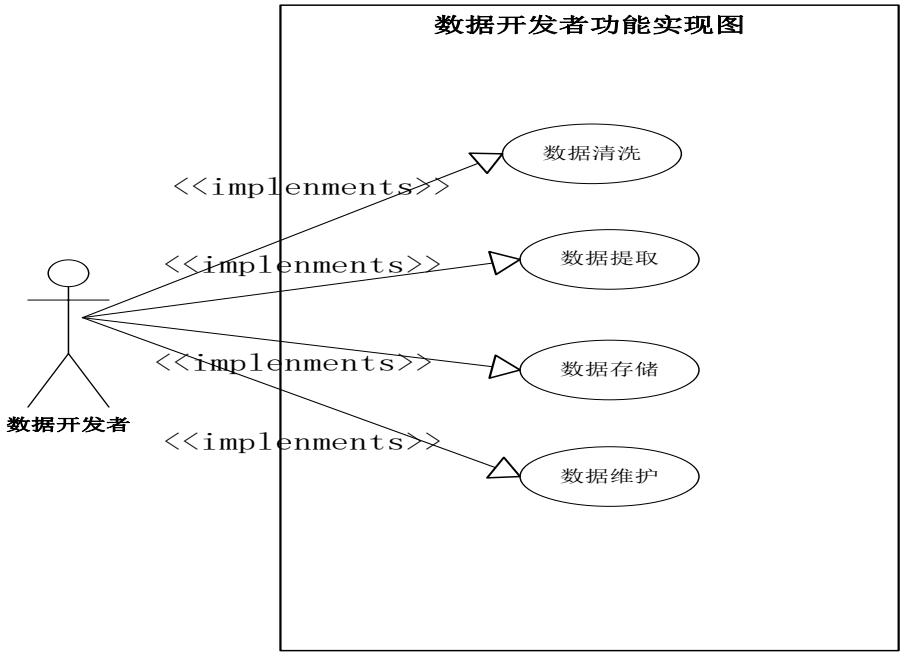


图 2.3 数据开发人员功能实现图

(3)运营人员使用的各功能及用例图，如图 2.4 所示。

●优惠券后台的使用；

●用户标签后台的使用；

●对有关接口进行测试；

●和数据开发者沟通得知核心 kpi 等，确定下一步运营计划。

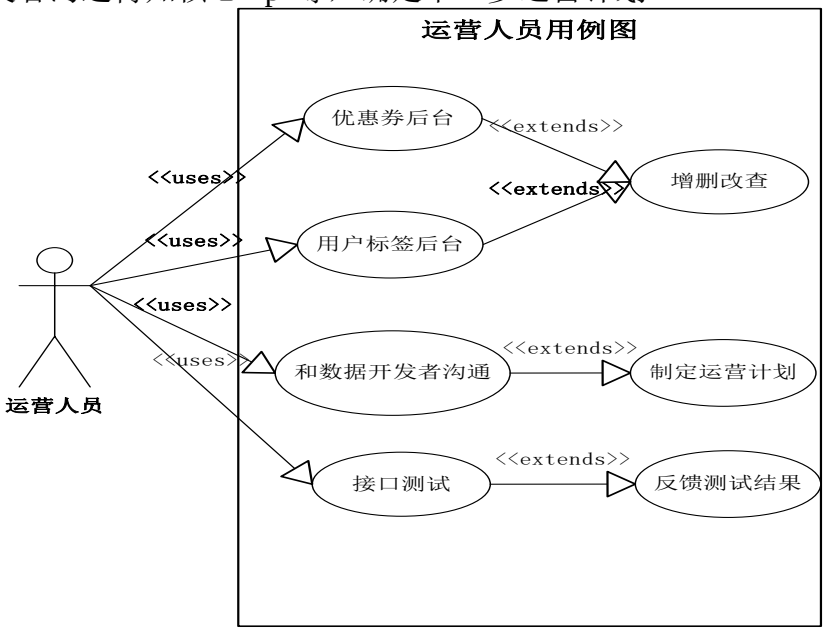


图 2.4 运营人员用例图

2.1.2 系统性能需求分析

由于该系统不但应用于公司内部，还应用于公司外部，而公司的使用环境多变，用户量可多可少，每个公司的网络情况又不一样等特点，公司外部更是如此，这就要求系统在运行时具有较高的性能。一个安全可靠的电商后台、安卓客户端和大数据清洗整个系统，除了能够满足上述的基本功能需求以外，在系统的性能上也需要达到下面几个要求：

(1)可靠性。系统各个功能模块应确保在限制的时间内和规定的使用条件下能完成所设计的功能，对用户操作的响应速度快，系统运行时错误率低。

(2)并发性。由于考虑到公司大小和个人网络环境复杂等情况，使用用户也可能比较多，数据量也可能比较大，因此必须要考虑多个用户同时向后端发出请求时系统的承载能力，系统应该能够实时响应用户请求和尽快跑出数据结果，避免出现系统访问错误、访问延时或者访问断开和数据处理时间长、处理结果错误等现象。

(3)一致性。信息在各个端传递的过程中保证一致，不出现信息丢失或者更改。

(4)可扩展性。系统本身应该具有良好的可扩展性。因为公司的实际情况，可能会对系统的需求做相应的更改或者增加，系统功能模块需要根据公司的具体需求来扩展。

(5)适应性。系统应该具备良好的交互界面，功能模块以及按键清晰，能够让用户使用方便，一目了然。

其中，前四个需求会在本文的性能测试中体现并且解决，适应性体现在后台页面和客户端界面设计成简单易懂。

2.2 系统总体设计思想

根据上文所述，对整个系统的需求分析，整个电商软件与大数据研究应该包括后台 Web 页面、后端服务器、安卓客户端和日志数据清洗四大部分。其中电商后台系统采用 B/S（Browser/Server）即浏览器/服务器架构，安卓客户端交互系统采用 C/S(Client/Server)即客户端与服务端架构，而日志数据的清洗工作采用开源的 MapReduce 和 HDFS。

2.2.1 系统网络总体结构图

电商软件开发与大数据研究包括配置页面（Web 页面）、Web 服务器、数据库、安卓客户端、HTTP 通信协议、数据清洗和清洗结果保存等组成部分。整个系统网络总体结构图如图 2.5 所示。

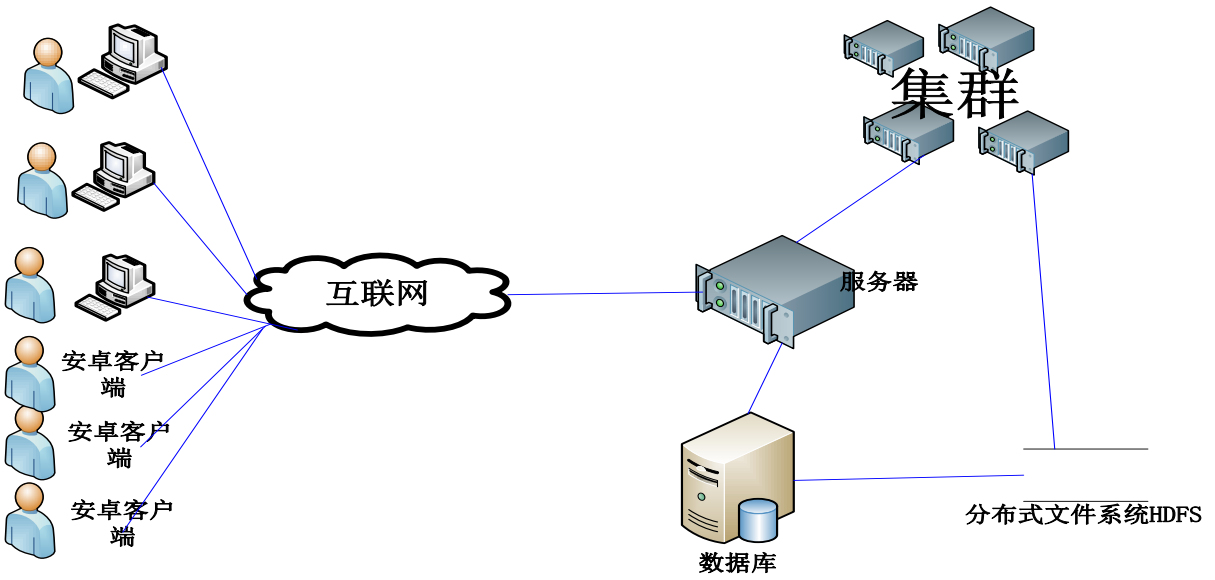


图 2.5 系统网络总体结构图

后台是整个电商研究的核心部分，部署了系统的数据库，服务器和 Web 前端页面。其数据清洗也是为了更好的为后台进行服务的，当然数据的清洗工作也不止于这点作用，比如还能分析用户行为和用户习惯等。

2.2.2 系统分层架构和数据清洗架构思想

在传统的软件系统开发中，为了直观，将操作数据库代码、业务逻辑处理以及前端界面等代码混写在一起，这样导致代码的可读性较差、耦合度高，也给系统中各功能模块的后期维护或者扩展带来极大的不便。为了方便系统后期的扩展和维护，在设计本系统的时候采用了分层架构的设计模式。分层架构的设计思想，有很多成功的例子。如在网络设计中，就把网络应用分成了功能各异的七个层次。实际网络中使用的 TCP/IP 参考模型，也遵循 OSI 七层网络模型，只是把 OSI 的应用层、表示层和会话层全部归并为应用层而已。

分层式架构在软件系统架构设计中是最常见、最重要的一种结构。分层式结构一般分为四层，从上之下分别为：表现层(UI)、控制层(Action)、业务逻辑层(Service)、数据访问层(DAO)。系统后台分层架构设计如图 2.6 所示和数据清洗 MapReduce 架构设计如图 2.7 所示。

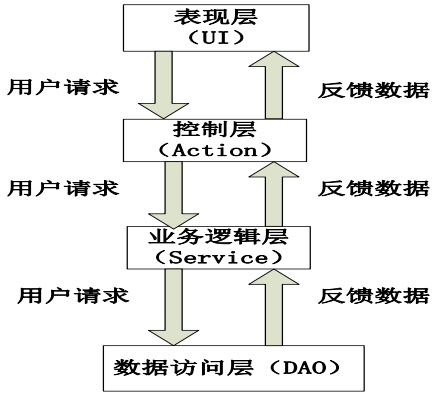


图 2.6 系统后台分层架构设计图

各层的功能如下：

(1)表现层(UI)：一般是指我们所能看到的前端。主要功能是能让我们在此上传信息和显示反馈给我们的信息，位于最外层（最上层），离用户最近。表现层主要表示成页面方式，或者表示成 WINFORM 方式，在控制层功能足够强大完善的情况下，表示层可以随意编写或改动，并得到控制层相应的功能支持。

(2)控制层(Action)：连接 UI 层和 Service 层，控制业务入口，封装前端数据，控制反馈结果等，在系统分层架构中，控制层非常重要，因为它位于表现层和业务逻辑层的中间，在数据交换传递程中起到了中间节点的作用。由于分层结构耦合度比较低，虽然上层结构依赖下层结构，但是上层无法知道底层的逻辑，也不能改变底层的逻辑，因此上层即使改变设计也不会对它调用的底层产生任何影响。

(3)业务逻辑层(Service)：连接 Action 层和 DAO 层，实现业务逻辑，具体包含：验证、计算、业务规则、事务控制、权限检查、日志打印和异常处理等，在系统分层架构中，业务逻辑层相当重要，因为它是位于数据访问层和控制层的中间，在业务逻辑和异常情况的处理以及权限控制等，都具有非常重要的作用。

(4)数据访问层(DAO)：主要工作是访问数据库，简言之就是实现对数据表的查询数据（Select），插入数据（Insert），修改数据（Update），删除数据（Delete）等操作。DAO 主要特点是直接处理数据并和数据库建立连接，而不是直接处理数据库，具体是为业务逻辑层或表示层提供数据服务。

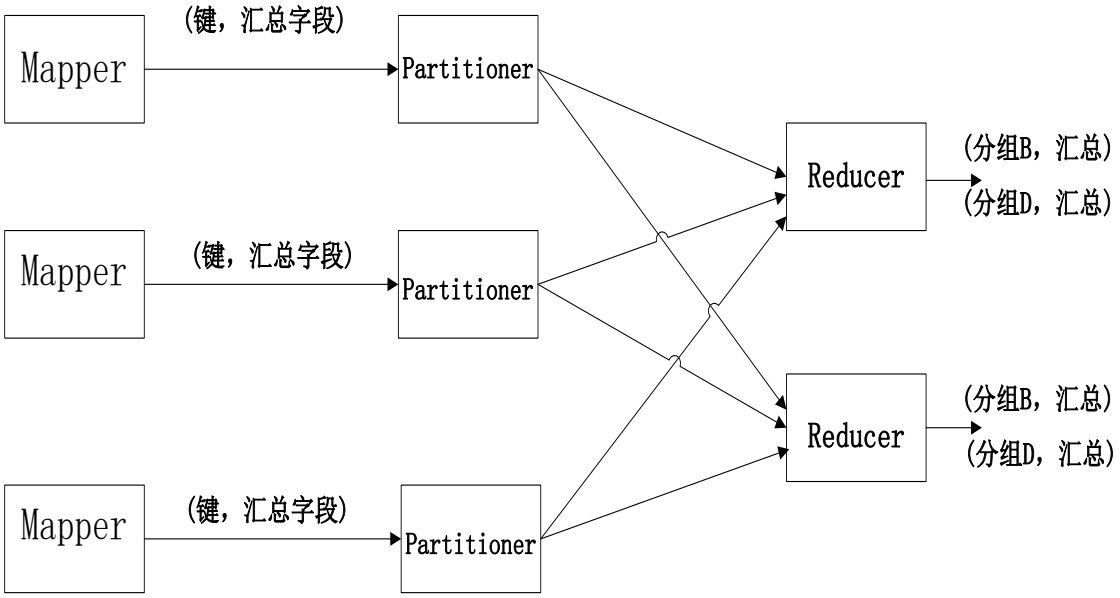


图 2.7 数据清洗 MapReduce 架构

数据清洗各模块的功能如下：

MapReduce 是面向大数据并行处理的计算模型、框架，其中有以下三层含义：

(1) MapReduce 是一个高性能并行计算平台（Cluster Infrastructure），以集群的方式进行

计算，他可以用普通的廉价计算机通过数十上千个节点来构成一个计算集群。

(2) MapReduce 是其实是一个软件框架（Software Framework）。它提供了一个大型但设计良好的并行计算软件框架。自动并行执行计算任务，计算数据的任务划分也是自动的，最重要的是，在集群上的哪些节点上执行任务也是自动分配的，这大大降低开发的难度，最后收集计算结果即可，由此可见 MapReduce 软件框架处理了软件开发人员十分烦恼的很多底层的程序细节[9-10]。

(3) MapReduce 是一个并行程序设计模式(Programming Model）。他利用函数式程序设计语言 Lisp 的设计思想，提供了一个简单的并行设计模式，用 Map 和 Reduce 两个函数编程，其有对应的高度抽象化的编程接口，利用这两个函数就能基本实现并行计算任务，可以简单方便地完成大量数据的编程和计算[10]。

2.3 本章小结

本章首先结合电商的实际情况，从两个方面进行系统设计的分析，分别是功能需求和性能需求，然后得到该系统需要完成的功能。再根据这些功能，确定了电商后台系统采用 B/S（Browser/Server）即浏览器/服务器架构，安卓客户端与服务器端采用 C/S 架构，而日志数据的清洗工作采用开源的 MapReduce 和 HDFS，由此得出了系统总体设计方案，最后还介绍了整个系统后台的分层架构和各层的功能以及数据清洗架构。

第三章 电商后台的设计

优惠券和用户标签配置后台是整个电商系统研究中的“核心”，因为不管是接口的开发，还是数据工作者对数据进行清洗加工等都是为电商公司做更好的营销服务，为安卓客户端用户提供更愉快的购物体验，这个后台需要接收并且存储客户端发送的服务请求和请求的具体内容，公司运营人员在平台上分配相应的优惠券和用户标签。配置后台的设计涉及到三大部分：后端接口、数据库和 Web 页面。本章首先介绍后台的架构，接着从方便开发者的角度进行技术方案选择。

3.1 电商后台的设计方案与 B/S 架构选择

3.1.1 电商后台方案

本设计的配置后台其实就是为公司运营人员设计的配置系统，可以实现对优惠券和用户标签的配置。在进行配置之前，运营人员需要对数据开发工作者的数据清洗处理结果进行查看和分析，得到最佳的配置方案后，运营人员可以通过页面上的按钮来进行配置。因此，在设计时对配置系统进行了分层次，分模块的划分，系统的分为 Web 页面，服务器后端接口和数据库。其功能架构图如图 3.1 所示。

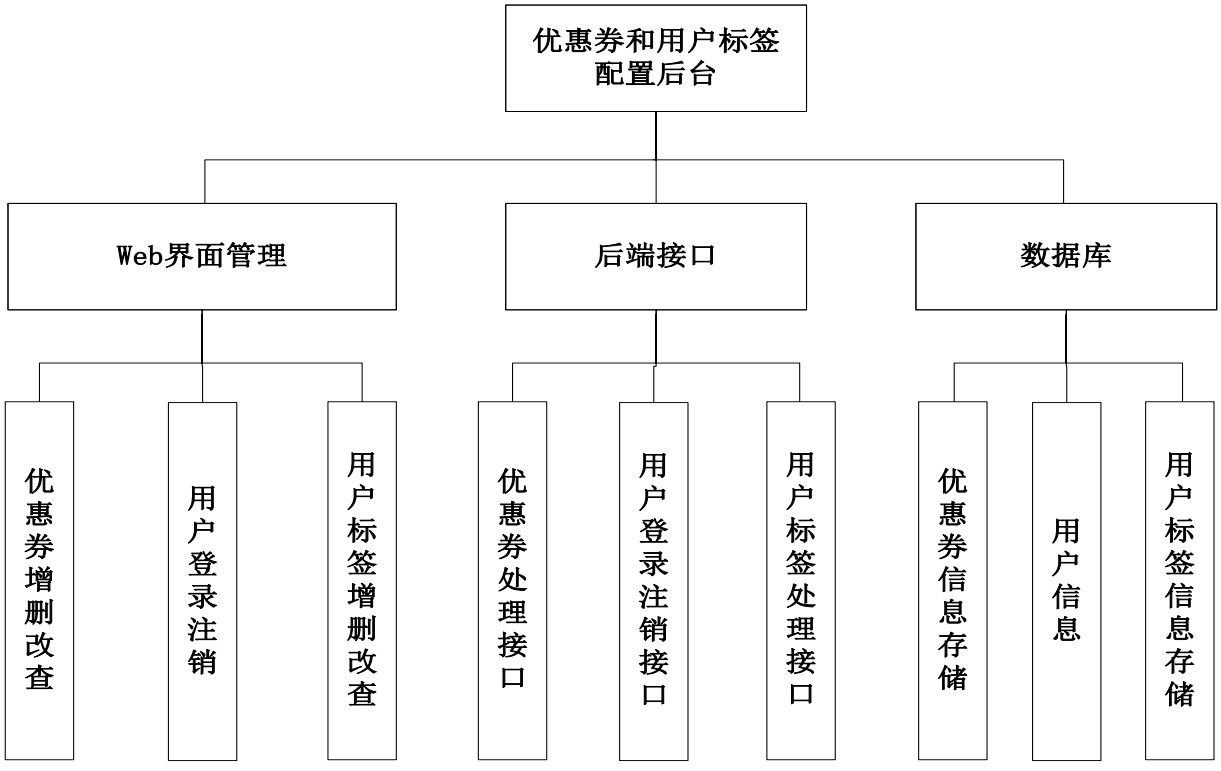


图 3.1 后台功能架构图

系统从功能上分成以下几个部分：

(1) Web 界面管理：配置后台服务的管理采用 B/S 架构，通过 Web 的方式对配置服务的进行管理，其管理的主要内容包括：优惠券和用户标签的增删改查，包括其生效时间、失效时间、对应的状态、对应的编码等各种参数。

(2) API 接口：后端系统要为客户端提供接口，客户端调用这些 API 接口就可以提交自己的优惠券和用户标签等信息，客户端在作登录、注销等操作时，也需要调用后端系统提供的接口。

(3) 数据库：存储优惠券和用户标签信息。

3.1.2 B/S 架构

B/S 架构即浏览器/服务器架构，目前 Web 应用技术日益成熟，B/S 架构超越 C/S 架构，变成目前开发人员中使用最广泛的架构模式。B 代表浏览器（Browser），几乎每个用户端都会有浏览器，S 代表服务器（Server），可以接收用户通过客户端浏览器上传的请求，经过相应处理后反馈数据信息。B/S 架构可分为三层，他们分别是：浏览器层、服务器层和数据库层。数据库建立在服务器上，服务器将客户端上传的数据请求经过处理，再通过指定接口转交给数据库。存储在数据库中的数据对象经过开发人员的操作处理，可以通过服务器反馈给用户浏览器。B/S 架构模型如下图 3.2 所示。

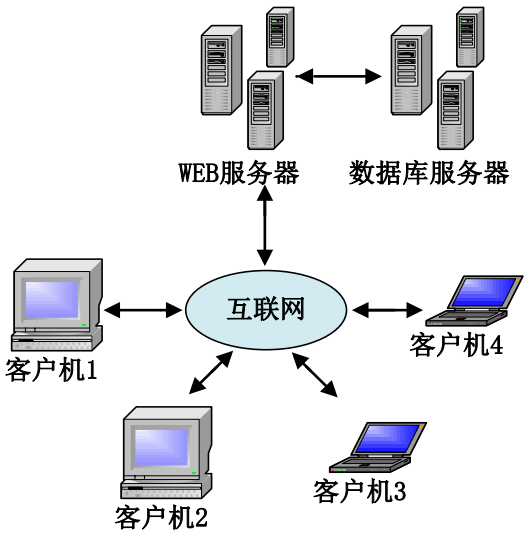


图 3.2 B/S 架构模型

如果使用传统的 C/S 架构，则会将大部分逻辑处理放在客户端实现。B/S 架构的优势是将大部分逻辑处理放在服务器层处理，这样可以减少浏览器端的逻辑处理，从而减少在客户端安装软件，从而解放客户端。对于 B/S 架构而言，其主要功能都是在服务器端完成的，而且方便了整个系统的开发和维护，用户想要完成所需的功能只需要一个浏览器即可。此外，B/S 架构在开发中也有很多优势：比如其采用的标准都是开放、开源的；B/S 架构的各层相互独立，对其中一层做功能逻辑的改变不会对其它层造成影响[11]。

但 B/S 架构也存在缺点，由于 B/S 架构如果将大量的逻辑处理放在服务器端（Server），使得服务器的工作量剧增，因此服务器的处理速度会降低，导致网页刷新时间变长，使用时会造成影响。此外，由于 B/S 框架太灵活，导致 B/S 架构在安全性和跨浏览器上的表现也不尽如人意。且利用浏览器代替客户端软件，也会存在交互界面比不上软件界面的缺点。但权

衡系统总体，B/S 架构比 C/S 架构更能适应现在的 Web 开发潮流。本配置系统选择 B/S 架构进行开发，就是充分利用了 B/S 架构的优势。配置系统的设计采用了 MVC 框架，即 Model 模型、View 视图、Controller 控制器框架。配置系统的实现模型如图 3.3 所示。

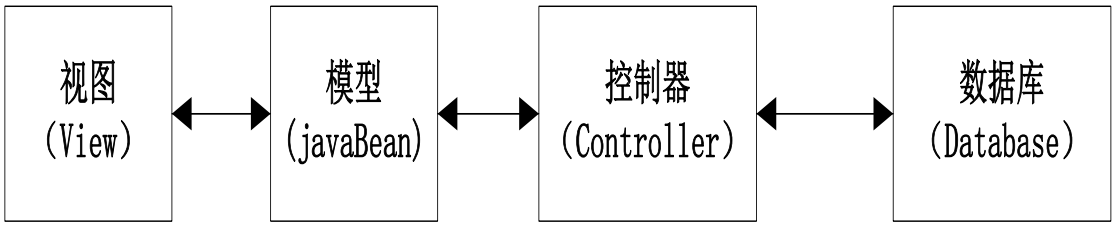


图 3.3 后台配置系统实现模型

上图是整个后台配置系统。视图（View）是应用程序中处理数据显示的部分，在本系统中即为后台配置平台的 Web 界面；控制器（Controller）是系统接收并处理用户请求的一层，控制器的工作一般为从前端视图读取用户提交的数据，并可以对用户数据进行必要的校验后再将数据信息传递给模型[11]。

3.2 Tomcat 与 Jetty 服务器

Web 服务器是一种专用计算机，他是在网络环境中给请求客户端提供响应的。Web 服务器是一种请求响应模式程序，也就是说当客户端发起一次请求的时候，其 web 服务器才会做出相应的响应，在完成后会断开服务器与客户机的连接。在万维网中，服务器就是这个网络中的一台计算机中的程序，该计算机有着唯一的 IP。我们能通常见到的 web 服务器有：Nginx 服务器、Tomcat 和 Jetty 服务器[12-13]。

不管具体是何种服务器，但是 Web 服务器的工作原理都是差不多的即：

第一步：客户端与服务器端经过三次握手建立起 http 连接；

第二步：客户端向服务器端发起请求；

第三步：服务器端响应客户端的请求；

第四步：服务器主动断开连接。

Tomcat 和 Jetty 都是两种开放源代码的 web 服务器，而且都是能为 java 程序服务的容器，并且支持 java web，它们都是开源的自由软件。

在我们的业务背景中，都是互联网程序的一般功能，在 web 服务器的选择上，如果使用 Tomcat，可以说这并不是一个很好的选择，因为 Tomcat 除了有 java servlet 规范，为了满足企业的其他特定需求，Tomcat 还大量增加了 J2EE 相关高级特性，而且在配置上面，Tomcat 比 Jetty 也要复杂许多，这样就不难知道 Tomcat 是一个重量级的服务器。对于分布式环境来说，正因为 Tomcat 的重量级，势必会消耗更多的内存，即使一台机器多消耗一点点内存，但是很多台机器加起来的话，这样的消耗也是不可估量的，此外，轻量级的 jetty 利用 nio 可以更加轻松的适用高并发的请求环境[14]。因此，我们在此选择 Jetty 来作为我们的 web 服务器。

3.3 ssm 框架与 ssh 框架

3.3.1 ssm 基本概念和整合步骤

先谈谈 ssm，即 springmvc、spring 和 mybatis，先了解基本概念：

1. Spring 是一个在 03 年迅速兴起的轻量级 java 开源框架，其中最显著的特点就是控制反转和面向切面编程，并且简单易学、耦度度低，所以任何的 java 应用程序一旦使用了 Spring 框架，就能大大的简化开发。然而在 03 年以前，企业软件应用的开发是很繁琐的[15,16]。

2. Spring MVC 是一个 web 层框架，也是在春天的基础上开发出来的框架产品。Spring MVC 之所以可以满足不同开发者的不同需求，是因为他对各个部分（控制器、模型、视图解析器等）都进行单独封装，也就是说具有低的耦合度[11,15]。其原理如图 3.4 所示：

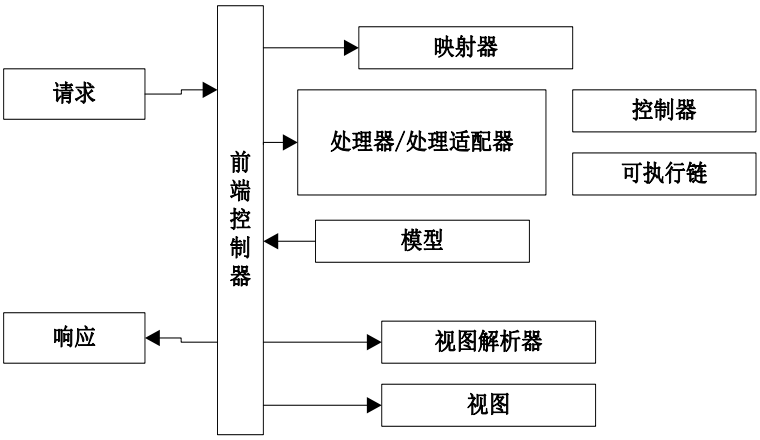


图 3.4 spring MVC 原理图

3. MyBatis 的前身是 iBatis，是一个持久层框架，前期是 apache 开发维护的，后来这个工程交给了谷歌公司才改名成现在的 Mybatis。持久层框架包含两个方面，一个是 SQL 映射，另一个是数据访问对象，有了 MyBatis 后，我们只需要写 Mybatis 代码，并不用写繁琐的 jdbc，并且参数也自动动态的进行了设置，还能产生动态 sql 等。MyBatis 所做的事情其实就是使用简单的 XML 通过接口将普通的 javabean 与 DataBase 表中的记录一一进行对应起来[17-18]。

ssm 整合步骤：

第一步：引入需要的 JAR 包，例如 spring 相关、sprin-webmvc、mybatis 和相应的整合包等。

第二步：Spring 与 MyBatis 的整合，例如建立 JDBC 属性文件、建立 spring-mybatis.xml 配置文件、Log4j 的配置、然后 JUnit 测试看是否整合成功。

第三步：上面已经完成了 Spring 与 Mybatis 的整合，接下来肯定就是整合 SpringMVC 框架了，在这个整合的过程中，最重要的就是这个框架自己的配置文件。

第四步：编写 web.xml 配置，把 spring-mybatis.xml 和春天 mvc 框架的 DispatcherServlet 配置进该配置文件，实际上就是整合这三个框架[19]。

3.3.2 ssh 基本概念和整合步骤

ssh，即 Struts、spring 和 hibernate，由于上面对 spring 已经做过介绍，所以这里不再赘述，Struts 和 Hibernate 两个框架的基本概念：

1. Struts1 是 Web 层框架，按照 MVC 模式设计而成。事实上，整个框架最核心的类其实就是一个 ActionServlet。我们可以将所有符合某个特征的请求全部交给他进行处理，他可以将不同的请求交给具体不同的接口去进行处理。而 Struts2 是在 1 版本的基础上开发的框架，在 1 中，有很多问题都没有得到很好的解决，并且在很多方面 2 都明显优于 1，但 Struts2 因为出在 Struts1 的后面，所以还隐藏着不少的风险，当软件开发者在开发项目的时候遇到问题时可能会很难驾驭，所以 1 和 2 的选择都各有各的优缺点。

2. Hibernate 实际上是和 Mybatis 一样，都是常见的 ORM 框架。使用该框架，也隐藏了底层访问数据库的代码，不管是增删改查的任何一个操作，我们只需要调用自己的 API 即可和数据库进行交互。而且对于不经常变动的数据，该框架还有二级缓存的功能[20]。

ssh 整合步骤：

1. 引入相关 jar 包，例如 Struts，spring，hibernate 和他们之间的整合包

2. Spring 与 hibernate 的整合，例如建立 JDBC 属性文件、建立 applicationcontext.xml 配置文件、Log4j 的配置、然后 JUnit 测试看是否整合成功。

3. 整合 Struts，上面已经完成了 2 大框架的整合，Struts 的配置文件单独放。

4. 编写 web.xml 配置，把应用上下文配置文件 applicationcontext.xml 和对应的 Struts 框架的核心类编写进 web.xml，实际上就是整合这三个框架。

在这里，经过综合考虑，一方面后台系统较为简单，另一方面我们对速度也有一点要求，因此，我们的后台选择 strut2 和 spring 框架做开发，这里我们没必要用 hibernate 这样的重量级框架，所以 DAO 层我们选择用 JdbcTemplate 和 DBCP 连接池。

3.4 数据库

3.4.1 数据库概述与数据库编码技术

1:概述：数据库(Database)在早期的定义就是数据的管理和数据的存储的一个仓库。但是随着时间的推移和市场的发展，也随着移动互联网和软件的发展，尤其是最近的二三十年，数据库的作用不再仅局限于存储和管理数据，而是面向更多的用户，根据用户实际需求来对各类数据对象进行相应的操作[21]。数据库发展至今类别也越来越多，我们平时应用中接触到最多的是关系型数据库，比如 mysql、oracle、DB2、SQLserver 等，还有非关系型数据库，比如 redis、MongoDB、HBASE 等。数据库的发展，也为我们在科学研究中提供了更好的方法去管理和利用更多的数据资源。数据库技术已经应用到各个领域的各类信息统计存储中。

2：编码技术：这里以 mysql 数据库为例进行讲解，在开发程序的时候，我们使用 mysql 数据库开发的时候，有时会碰到自己明明输入的是中文，为什么数据库中存储的就是？？？。

其解决办法是：

(1)、在配置 Connection URL 时，加上?useUnicode=true&characterEncoding=utf-8

(2)、编辑/etc/my.cnf

在[mysqld]下添加

default-character-set=utf8

在[client]下添加

default-character-set=utf8

由此我们看出，理解数据库的编码技术，重点是理解 utf-8 编码，UTF-8 是一种变长字节编码方式，也是互联网上使用最广泛的 Unicode 的一种实现方式，其大部分的操作系统都支持该编码，linux 系统默认支持的就是 utf-8 编码。UTF-8 编码是 Unicode 的一种实现方式，其中一个字符可以表示成 1-6 个字节，最高位字节 1 的个数是字节数，后面的所有字节都是 10 开头，由此可以计算出在该编码中，除去控制位之后用来编码的位数最多是 31 位,而且其和 Unicode 的位高低顺序也一模一样[22]。

3.4.2 数据库选择

Oracle 和 MySQL 是当今世界两大主流关系型数据库，DB-Engines 发布了 2016 年 8 月份的数据库系统排名榜单中排名第一二位，而且这两种数据库都是关系型数据库。两种数据库各自有优缺点：

(1)优点：MySQL 的优点是软件体积小，操作方便，系统处理速度快，由于免费使用可以减少成本。此外，一些大公司特别是互联网公司也在使用 MySQL 数据库，表明可靠性高。并且支持 C、C++、Java 等主流编程语言，可以在不同平台上工作。Oracle 的优点是具有强大的功能和性能，安全性更高，完善的公司技术支持和人性化的设计。

(2)缺点：MySQL 的缺点就系统定位就是中小型的数据库，因此在功能上也会相对不够完善，大企业用的不多；Oracle 的缺点是对运行平台的要求较高，操作、管理维护等相对复杂，最重要的是 Oracle 数据库不开源，收费高。

综合以上优缺点分析，本设计由于需要存储的信息比较简单，需要的功能也不多，所以对于后台这一部分这里选择使用 MySQL 数据库。

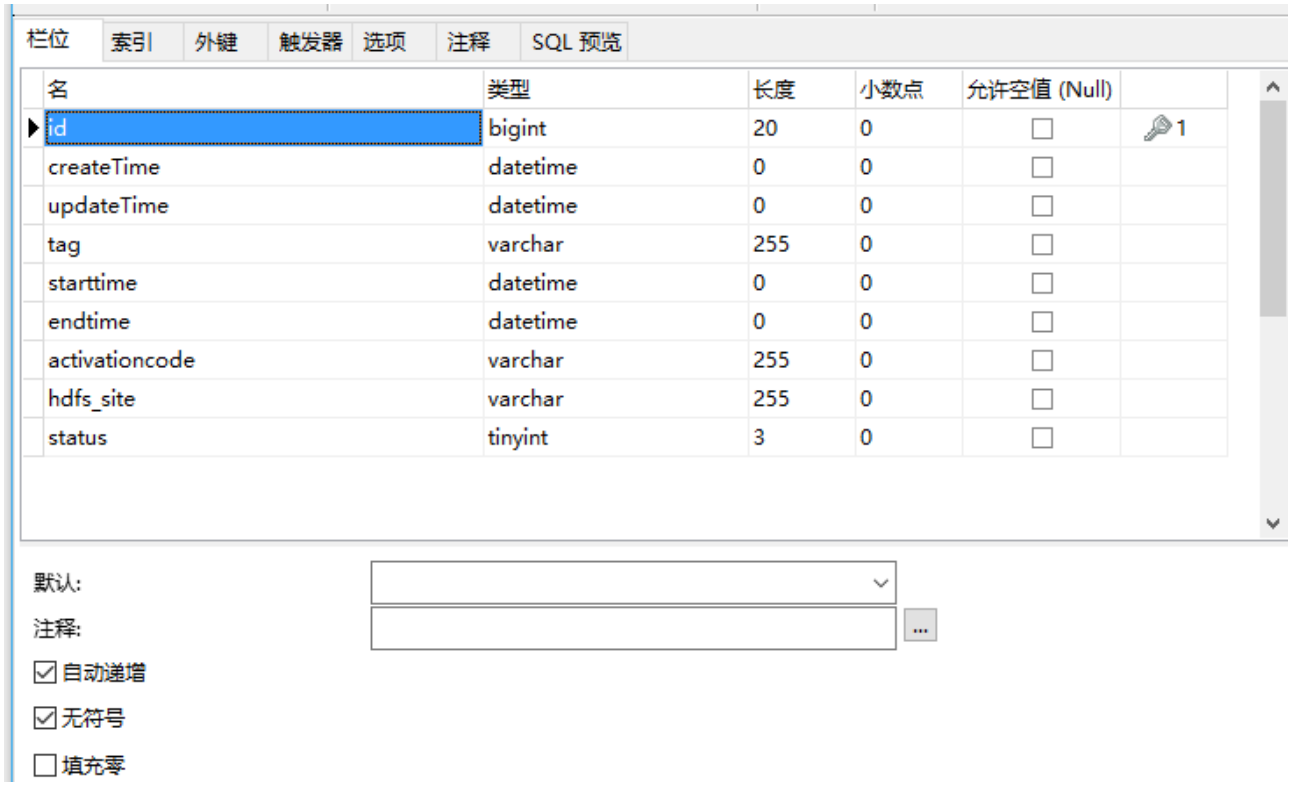
而对于非关系型数据库，hbase 是跟 hdfs 以及 mapreduce,Spark 等结合的最好的，不但可以方便地存，更可以方便地算，而且对于一个用户来说，当然是不仅可以领多张优惠券而且还可以打上多个用户标签，对于有多个版本的场景，mongoDB 在小规模下也可以适应这种场景，不过随着数据增长，会涉及到 sharding 和 gridfs，让人痛苦的要命，所以在这里我们选择 HBASE 来作为我们的非关系型数据库。

另外，由于我们还需要对日志文件进行清洗处理等，所以还需要 HDFS 来做数据存储的作用。

3.4.3 数据库的设计

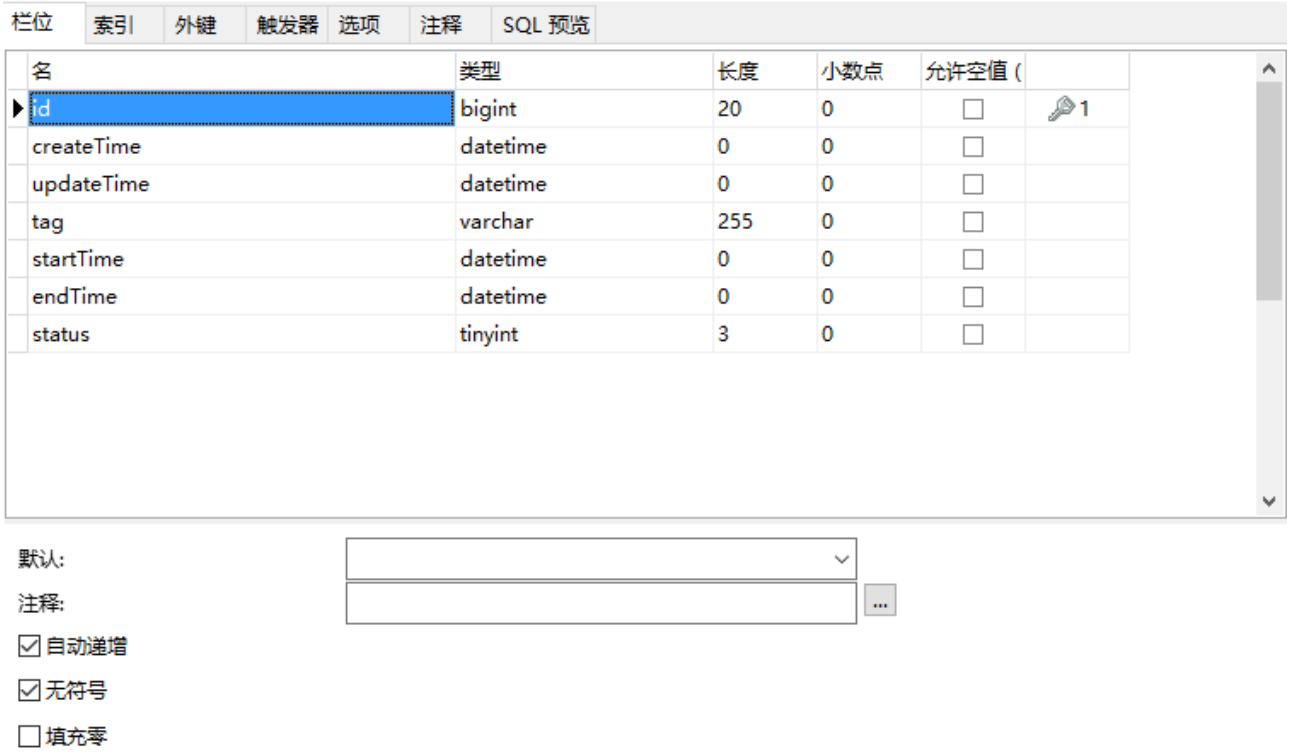
本电商后台设计一共需要4张表，分别是两张mysql表和两张HBASE表，由于HDFS是一个分布式文件存储系统，没有表格的概念，这里就不做展示，由于本系统数据库表不多，这里可以把所有的数据表列出来，具体数据表如下表3.1-表3.3所示。

表3.1 优惠券信息表以及建表语句如下：



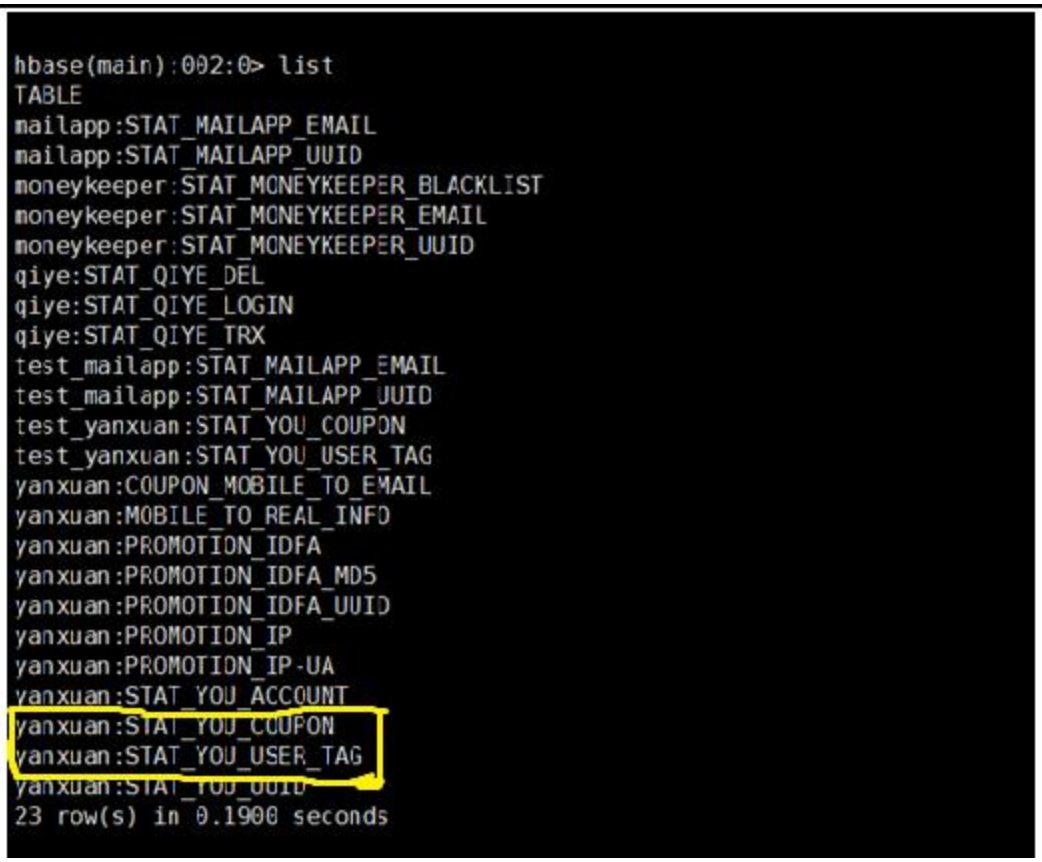

```
CREATE TABLE `tb_hquery_you_coupon` (
`id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
`createTime` datetime NOT NULL,
`updateTime` datetime NOT NULL,
`tag` varchar(255) NOT NULL COMMENT '优惠券码',
`starttime` datetime NOT NULL COMMENT '优惠券生效时间',
`endtime` datetime NOT NULL COMMENT '优惠券失效时间',
`activationcode` varchar(255) NOT NULL COMMENT '优惠券激活码',
`hdfs_site` varchar(255) NOT NULL COMMENT 'hdfs路径信息',
`status` tinyint(3) NOT NULL COMMENT '0 - 生效; 1 - 失效',
PRIMARY KEY (`id`),
UNIQUE KEY `tag` (`tag`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COMMENT='严选优惠券白名单配置文件';
```

表3.2 用户标签信息表以及建表语句如下：




```
CREATE TABLE `tb_hquery_you_coupon_tag` (
`id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
`createTime` datetime NOT NULL,
`updateTime` datetime NOT NULL,
`tag` varchar(255) NOT NULL COMMENT '用户标签',
`startTime` datetime NOT NULL COMMENT '标签生效时间',
`endTime` datetime NOT NULL COMMENT '标签失效时间',
`status` tinyint(3) NOT NULL COMMENT '0 - 生效; 1 - 失效',
PRIMARY KEY (`id`),
UNIQUE KEY `tag` (`tag`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COMMENT='严选用户标签配置文件';
```

表3.3 两张hbase表以及建表语句如下：



create 'yanxuan:STAT_YOU_COUPON', {name=>'C',VERSIONS>=99}

create 'yanxuan:STAT_YOU_USER_TAG', {name=>'T',VERSIONS>=99}

两张 hbase 表就是上面两张标黄的部分：yanxuan:STAT_YOU_COUPON 和 yanxuan:STAT_YOU_USER_TAG。

3.4.4 服务器与数据库的数据交互设计

上面已经讲到服务端我们使用的是 SSM 框架和 mysql 数据库，按分层的思想看和数据库的数据交互的是 DAO 层，对 DAO 层而言，我们使用的是框架中的 Mybatis，所以我们重点理解 mybatis 与 mysql 数据库的数据交互设计。

Mybatis 其实是对 ORM（Object Relation Mapping）规则的实现，jdbc 代码的封装，其 jdbc 与数据库的交互步骤是[23]：

1. 根据数据库方言，加载对应数据库自己的驱动；
2. 获取数据库的连接，这个连接对象（Connection）可以通过 jdbc 中的驱动管理器（DriverManager）获取到；
3. 获取 Statement/PreparedStatement，可以通过第二步中的连接（Connection）对象得到该对象；
4. 将 SQL 语句绑定到 Statement/PreparedStatement 中去，准备向数据库发送 SQL

语句；

5. 执行完 SQl 语句后，返回相应的结果；
6. 如果是查询的话，迭代结果集进行处理；如果是非查询的话，还需要事物支持；
7. 依次关闭连接对象，释放连接资源；

Mybatis 是对 jdbc 的封装，所以与数据库的数据交互的工作流程和以上的流程大体上是一样的，但是也有差异，其工作流程是：

1. 通过 Reader 对象读取 src 目录下的 mybatis.xml 配置文件；
2. 通过 SqlSessionFactoryBuilder 对象创建 SqlSessioFactory 对象；
3. 从当前线程中获取 SQLSession 对象；
4. Mybatis 中默认开启事物；
5. 通过 SqlSession 对象读取映射文件中的 SQl 语句；
6. 执行完 SQl 语句后，做相应处理，再事物提交；
7. 关闭 SqlSession 对象，并且分离当前线程与 SqlSession 对象，让 GC 尽早回收；

在实际项目中，其都会和 spring 进行整合使用，所以不但是 mybatis.xml 配置文件的读取，还是事物的支持，都可在 spring 的配置文件中进行配置，这样大大的降低了开发的难度和提高了开发的效率，减小了大量服务器与数据库的数据交互代码。

3.5 浏览器客户端与服务器的通信设计

客户端与服务器间的通信方式大致有两种：基于 HTTP（即超文本传输协议 Hypertext Transfer Protocol）的通信方式和基于 Socket 套接字的通信方式，在本章中，研究的是浏览器与服务端的交互，我们重点是理解 http 协议，HTTP 协议是在 TCP 协议层之上的，而 HTTPS 还会在 TCP 协议层之外再加入 TLS 或 SSL 安全保密协议。

对于 TCP 协议的传输层来说，任何系统在通信传输数据时必然要划分网络体系结构，而不同的系统在划分网络体系时都大同小异。OSI（Open System Interconnection），意为开放式系统互联，是国际标准化组织（ISO）制定的经典体系模型。OSI 模型把网络传输工作划分为 7 层，分别是物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。OSI 模型虽然分层明确，理论也比较完整，但这种网络分层结构却并不实用。随着人们对网络传输体系认知的加深，现在广泛使用的是五层协议体系模型。五层协议结构分别是物理层、数据链路层、网络层、传输层和应用层，将会话层、表示层和应用层合并为一层，统称应用层，大大简化了体系结构。

HTTP 通信方式与其他不同的是，只要客户端发送请求，都得得到服务器端的响应，其响应可以是 web 页面也可以是 json 数据，客户端和服务器端的连接会在请求结束一段时间后自动断开，从客户端建立与服务器端的连接到服务端断开连接的过程称为“一次连接”，而为了进行可靠传输，在客户端与服务器开始传输数据之前，TCP 协议层要求建立连接，这个连接过程需要经过三次握手，三次握手过程如下[24-26]：

第一次：客户端发起和服务端建立连接，这时客户端会向服务端发送 syn 包（Synchronize Sequence Numbers）同步序列编号(seq=x)到服务器。

第二次：接收客户端发送的 SYN 包后，服务器必须确认回应客户端之外同时自己发送一个 SYN 包。

第三次：在接收到服务器的数据包后，客户端将向服务器发送加一后的确认包 ACK，当这个确认包发送后客户机和服务器就进入成功的连接建立状态了。

其三次握手连接示意图如下：

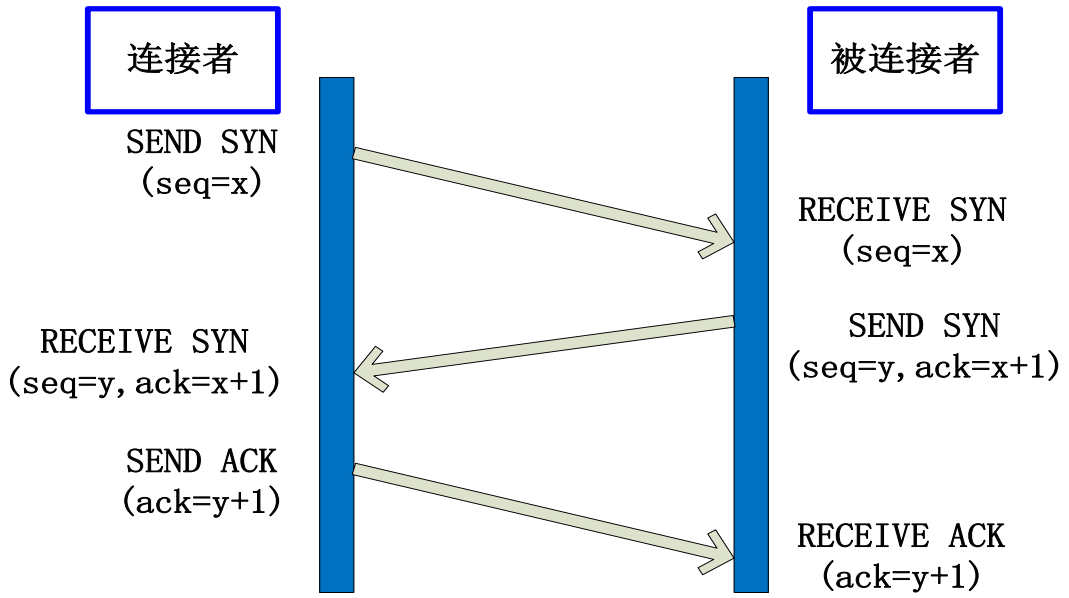


图 3.5 三次握手连接示意图

HTTP 连接，就需要先让客户端发送请求给服务器端，才能让服务器端返回数据给客户端。HTTP 协议的请求响应模式一直都是客户端先请求，然后服务器再响应的模式。如图3.6 所示 HTTP 协议的请求响应模型。

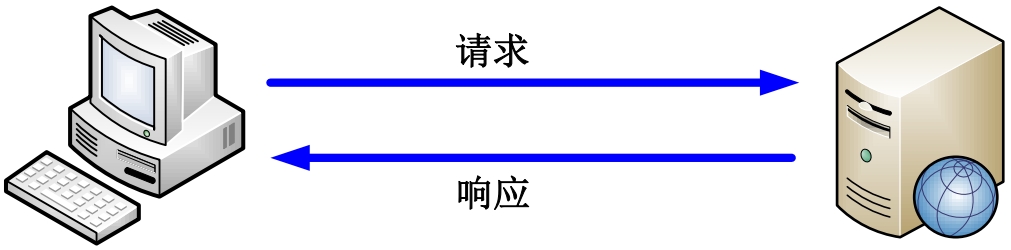


图 3.6 HTTP 请求响应模型

HTTP 通信协议目前有三个版本：HTTP/0.9，HTTP/1.0 和 HTTP/1.1。在早期，HTTP 并没有正式标准，现在我们常见 HTTP/1.0 也是在早期版本演变而来的，在 0.9 版的 http 中，没有现在这么多的请求方式，其只支持最最常见的 get 请求方式。正因为他刚刚兴起，所以也就没有多个版本的说法，因此在请求行中就没有版本号字段，除此之外，早期版本也还没

有请求头。1996 年 5 月，HTTP 正式被作为通信传输标准被公布，版本被命名为 HTTP/1.0，目前还有很多地方会使用，但是在代理服务器中，1.0 版的兼容性还是不太好。在 1997 年 1 月，采用久连接的升级版 HTTP/1.1 诞生，到目前为止，他已经成功超越前两个版本，成为了应用最广泛的版本，并在代理服务器中也能很好的工作。

为了使客户端和服务器端有良好的通信，HTTP 通信协议使用 8 种请求方法用来规定 HTTP 请求的具体内容，在此介绍最常用的三种：

(1)GET 请求方法：用来请求指定的 URL 资源，它是目前 HTTP 请求中用到最多的，GET 请求需要对请求的网络资源进行定位和数据传送；

(2)HEAD 请求方法：HEAD 请求的规则和（1）中的方法一样，区别在于 HEAD 请求没有响应体，没有实体内容，并且只能得到文件长度、时间等信息。HEAD 方法在开发人员只需要得到内容的信息时会用到；

(3)POST 请求方法：POST 请求可以向所需的网络地址上传数据信息，不同于（1）中的方法，POST 请求上传的数据放在 HTTP 请求体中；

服务器端在接收和处理客户端的发送的请求后，会返回一个 HTTP 响应消息。响应消息由状态行、消息报头（也成为响应头 Response Header）和响应正文三部分组成。状态行中有个字段 Status-Code，表示服务器返回的响应状态代码，状态码是一个三位数字，第一个数字定义了响应的类别，有五类可能取值，状态码的第一位数字定义了应答类型，如表3.4。

表 3.4 状态码及其含义

状态码值	含义
第一位是1，后面两个数字随机	请求已接收，还需处理
第一位是2，后面两个数字随机	请求已被成功接收
第一位是3，后面两个数字随机	重定向，需要进行更进一步的操作
第一位是4，后面两个数字随机	客户端错误
第一位是5，后面两个数字随机	服务器端错误

本设计中客户端向服务器端请求的方法只需要用到 GET 和 POST 方法。

3.6 网页前端的设计

为了使代码更加简洁，可读性、可维护性更强，这里使用各种标签库 taglib 和 EL 表达式。如果不使用这些的话，需要用较为复杂的 JS 代码来实现页面逻辑，比如要显示所有优惠券的所有信息，可以使用如下代码：

```
<c:forEach items="${searchResult}" var="result" varStatus="status">
                    <tr <c:if test="${(status.index % 2) ==
1}">bgcolor="#CCFFFF"</c:if>>
                        <td>${result.id }</td>
                        <td>
```

```
                    <c:if test="${result.status == '0'}"><font color="red"><b>
已启用</b></font></c:if>
                    <c:if test="${result.status == '1'}"><font color="red"><b>
已禁用</b></font></c:if>
                </td>
                <td>${result.tag }</td>
                <td><fmt:formatDate value="${result.startTime}"
pattern="yyyy-MM-dd HH:mm:ss" /></td>
                <td><fmt:formatDate value="${result.endTime}"
pattern="yyyy-MM-dd HH:mm:ss" /></td>
                <td>${result.activationCode }</td>
                <td>${result.hdfs_site }</td>
            </tr>
            </c:forEach>
```

这里的 result 直接就是域中的一个对象。如果用 HTML 直接写，就没办法使用 for 循环来遍历所有对象，因此，效率会大大降低。

总体的网页功能和结构比较简单，其中最重要并且最复杂的是当全部获取所有的优惠券和用户标签的时候，因为数据量比较多，需要做下分页技术，因为大部分的分页处理逻辑也是在服务器端，这里就不做深入探讨，我们将在下一章的具体实现中分析，这里我们只展示 web 前端的分页代码，代码如下：

```
<table align="center">
        <tr>
            <%--总共  第i页 --%>
            <td>
                总共：${pageResult.pageCount}页      第
<s:if test="#request.pageIndex <= 0">1</s:if><s:else>${pageIndex}</s:else>页
            </td>
            <%-- 首页 --%>
            <td>
                <a
href="${pageContext.request.contextPath}/coupon/action.do?userAction=${request.userAction}&p
ageIndex=1">首页</a>
            </td>
            <%-- 上一页 --%>
            <td>
```

```
                    <a
href="${pageContext.request.contextPath}/coupon/action.do?userAction=${request.userAction}&p
ageIndex=<s:if test="#request.pageIndex <=1">1</s:if><s:else>${pageIndex-1}</s:else>">上一页
</a>
                </td>
                <%-- 下一页 --%>
                <td>
                    <a
href="${pageContext.request.contextPath}/coupon/action.do?userAction=${request.userAction}&p
ageIndex=<s:if
test="#request.pageIndex >=#pageResult.pageCount">${pageIndex}</s:if><s:else>${pageIndex+1
}</s:else>">下一页</a>
                </td>
                <%-- 尾页 --%>
                <td>
                    <a
href="${pageContext.request.contextPath}/coupon/action.do?userAction=${request.userAction}&p
ageIndex=${pageResult.pageCount}">尾页</a>
                </td>
            </tr>
            </table>
```

由前端分页代码可以看出，我们使用的还是标签库 taglib 和 EL 表达式。

3.7 本章小结

本章首先详细介绍了电商后台的总体设计方案和 B/S 架构，服务器的选择以及两种主流框架的整合等；然后对数据库设计和数据库编码技术进行了解析；最后大致介绍了一下数据交互和 web 网页前端代码，其中的数据交互包括服务器与数据库的数据交互和客户端与服务器的数据交互。

第四章 电商后台的具体实现

电商后台是整个研究系统的的“中心环节”，需要接收并且存储客户端发送的服务请求和请求的具体内容，然后通过接口的调用来判断存储的优惠券和用户标签信息产生相应的日志，经过日志的清洗后又更好的服务于后台系统，这样就形成了一个良性循环。电商后台的的设计涉及到三大部分：控制器、数据库和 Web 页面。本章首先简单的介绍一下后台系统和后台核心分层，技术方案选择我们在上一章已经介绍过，这里就不再做介绍，最后分模块介绍后台系统的具体实现与接口的开发。

4.1 后台系统简介

各种各样的后端软件的开发和我们之前常用的电脑软件的开发不同，前者一般会被安装在各种服务器上，更新也更加频繁。根据本设计要求的软件特点和开发成本，这里选择 linux 操作系统作为后端平台，Linux 操作系统是一个完全开源免费的服务器端操作系统，目前 Linux 操作系统在服务器市场中占据着绝对优势，市场占有率稳居第一。Linux 系统由四个部分组成。其中 linux 内核是整个系统底层最核心的一部分，用户可以利用 linux 系统运行程序、管理文件等[27-28]。Linux 操作系统的技术优势主要体现在下面几点：

- 操作系统开源
- 可兼容多种机型
- 规范了众多的标准化技术
- 核心技术完整统一
- 拥有良好的界面
- 多用户、多任务

本设计的电商后台其实就是为公司运营人员开发的配置系统，可以实现对电商中的优惠券和用户标签进行操作。在进行配置之前，数据开发者需要对以前的优惠券使用情况和用户标签信息进行分析和存储，得到最佳的配置方案后，反馈给运营人员，然后运营人员通过页面上的按钮来进行最佳的配置。因此，我们的电商后台和数据开发都是一环扣一环，紧密联系的。

4.2 电商后台工程目录介绍

整个优惠券和用户标签的配置后台包括服务端开发和前端配置页面的开发。服务器端为客户端提供相应的功能接口，前端页面发起请求调用后端接口。整个后台需要实现优惠券和用户标签的增删改查和发券之前的领券接口测试等功能，并在登录时对用户密码进行 MD5 加密处理，其数据传输是通过 HTTP/1.1 版本协议进行交互的。

整个优惠券和用户标签配置后台工程项目用 ivy 插件来进行 jar 包的管理，因为在实际项目开发中，一个项目会好几十个 jar 包甚至更多，而且还会出现间接依赖等多种依赖关系，如

果没有 ivy 这样的管理工具，项目开发 jar 包的导入就会占用大量时间，所以一般都会用这样的工具，这时你只需要把你用到的 jar 包在 xml 配置文件中配置好即可。如图 4.1 所示，显示了 Web 后台工程项目的整体结构。

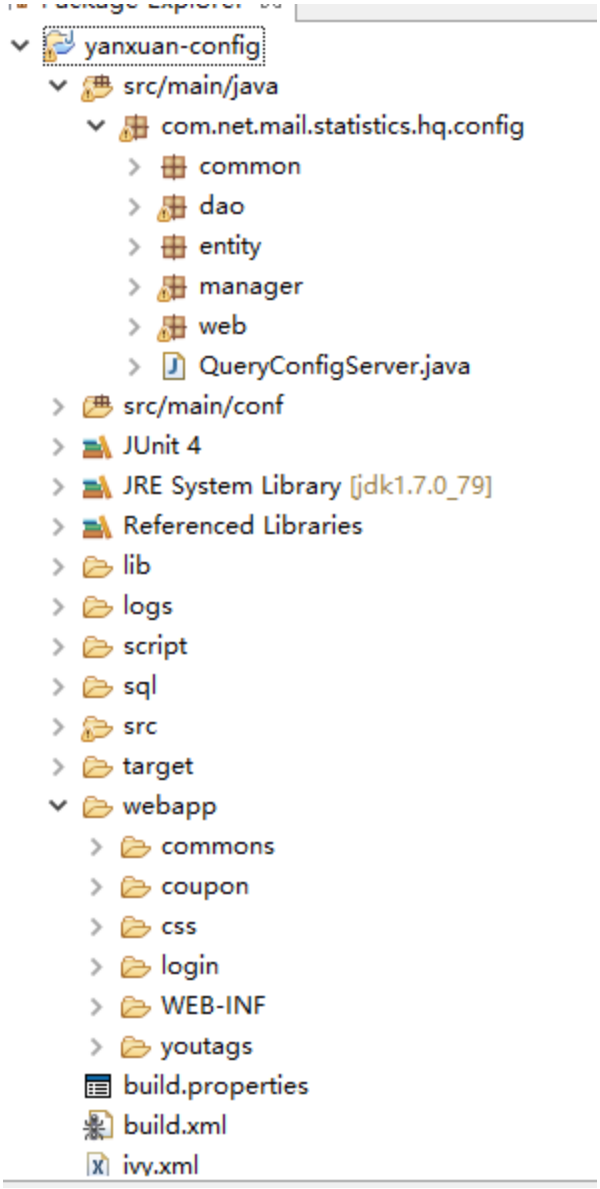


图 4.1 web 后台工程目录结构

各个包及几个重要的文件进行简单介绍，详细的思路和代码编写将在后面章节中具体介绍：

1）src/main/java　所有功能的源代码都放在这下面，其中：

com.net.mail.statistics.hq.config.entity 用于放置定义数据类型。

com.net.mail.statistics.hq.config.web 用于放置 web 前端页面的请求数据和指定 jsp。

com.net.mail.statistics.hq.config.manager 用于放置 service 代码。

com.net.mail.statistics.hq.config.dao 用于放置 dao 代码。

com.net.mail.statistics.hq.config.common 用于放置封装异常的代码和特殊功能的代码。

com.net.mail.statistics.hq.config 用于放置 Server 代码。

2) src/main/conf 各种配置文件都放在这下面，其中：

xml 文件是各种框架的配置文件，例如 applicationContext.xml、Struts.xml 等。

properties 文件是各种属性文件和 log4j 的属性文件。

3）webapp\coupon 放置所有 Web 前端访问的页面。

4）webapp \css 放置所有 Web 前端访问的 css 文件。

5）webapp\WEB-INF 放置 web.xml 文件。

6）ivy.xml 中是 ivy 工程的所有依赖关系。

4.3 后台核心分层

4.3.1 控制（Action）

在这个优惠券和用户标签配置后台的后端开发中，控制层我们是用 Struts2 框架开发的，控制层也是调用接口时后端的入口层，对调用方来说，后端只需要向调用者提供一个接口即可。因此，自然而然整个后台的开发的核心就是接口的开发，每一个接口都对应着单独的一个功能。

前面已经介绍过，Struts2 是一个 webmvc 开发框架，也符合请求响应模式。Struts2 处理前端请求的基本过程如图 4.2 所示：前端控制器 StrutsPrepareAndExecuteFilter 接收到请求后，StrutsPrepareAndExecuteFilter 为请求匹配合适的页面控制器 controller，然后走多个 Interceptor 拦截器，走 18 个默认拦截器和自定义拦截器，再走页面控制器对应的业务方法进行处理，处理完毕后返回一个字符串对象给核心控制器 StrutsPrepareAndExecuteFilter 对象，然后 StrutsPrepareAndExecuteFilter 对象再根据返回的字符串将对应的视图 view 返回给用户。

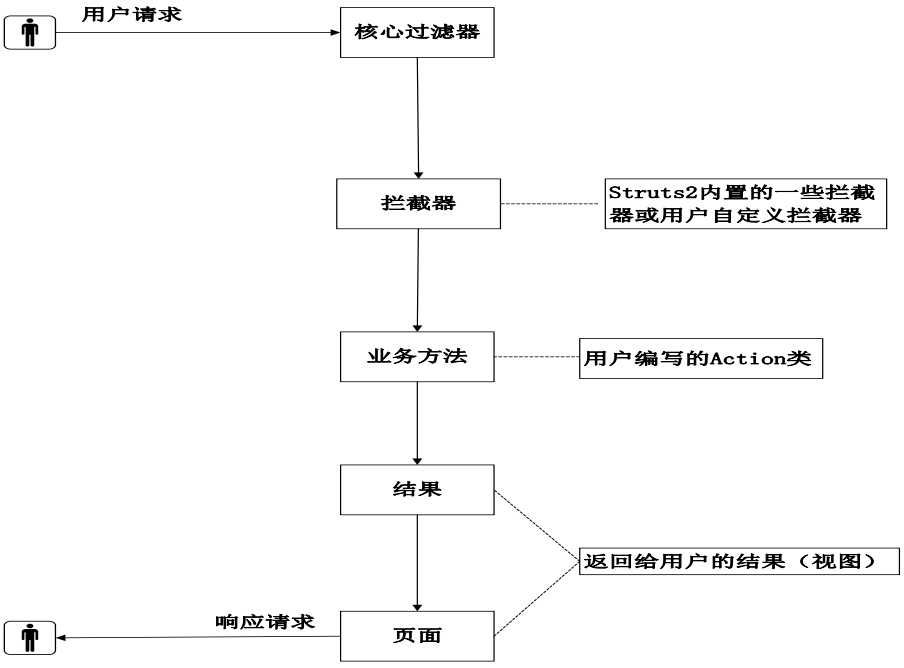


图 4.2 Struts2 处理前端请求的基本过程

控制层（Web 层）应用框架是一种 Web 应用程序开发的 mvc 框架，用来支持动态网站、前端页面参数的收集及响应服务的开发，它提供很多操作前端页面和管理会话等的接口，帮助开发人员在与前端网页交互时更加轻松，并且标准化的接口可以让代码更通用。自从基于 MVC 分层结构的 Web 设计思想普及以来，开发人员在开发时对 Web 层应用框架的选择会对系统开发的难易和质量造成很大影响。现在 Struts、Springmvc 等众多 Web 层框架的问世，目的都是为了使开发人员在开发时能够更方便，逻辑更清楚。

该项目的后台是一个模拟严选电商优惠券和用户标签为例，其 Web 层如图 4.3 所示：

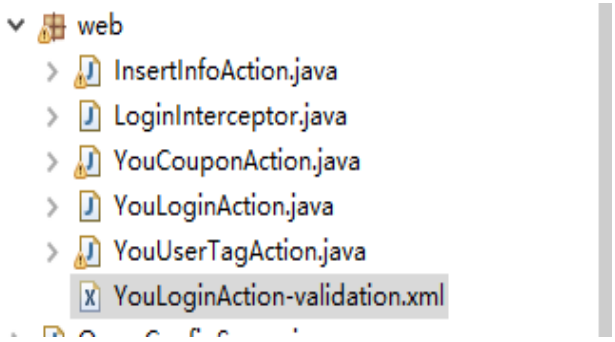


图 4.3 电商后台 Web 层

其各个文件的作用如下：

1. com.net.mail.statistics.hq.config.web.YouLoginAction.java 登录接口 Action
2. com.net.mail.statistics.hq.config.web.LoginInterceptor.java 登录拦截器
3. com.net.mail.statistics.hq.config.web.YouLoginAction-validation.xml 验证用户名密码
4. com.net.mail.statistics.hq.config.web.YouCouponAction.java 优惠券的 CRUD 接口
5. com.net.mail.statistics.hq.config.web.YouUserTagAction.java 用户标签的 CRUD 接口
6. com.net.mail.statistics.hq.config.web.InsertInfoAction.java 插入数据接口

4.3.2 服务（Service）

如果说电商后台系统是整个电商系统的“中心环节”，那么控制器就是整个电商后台系统的核心，控制逻辑的业务流程即服务层就是处理整个后台系统的工作内容。其后台的业务逻辑层（服务层）处理的是各种业务逻辑，比如事物、异常、日志打印、权限检查等。

服务层有个主要的任务是去维护数据库的事物，来保证不出现幻读或脏读，事物通过在 Spring 的配置文件中以 tx+aop 的方式来进行了配置，委托给了 jdbc 的事务管理器 DataSourceTransactionManager，而且整个后台系统我们是用 Spring 这个轻量级框架来进行开发的，用到了 Spring 的 IOC（inverse of control）和 AOP(Aspect Oriented Programming)模块特性,Spring 中的 IOC 就是控制反转，也就是说，整个后台系统中的对象和给对象的属性注入值的工作不再又程序员自己来控制，而是交由了 Spring 容器来维护；Spring 中的 AOP 就是面向切面编程，和面向对象编程不一样的是，AOP 它的切入点是一个方法，关注的是一个一个的

函数，也就是对象的动作，这对于业务逻辑层的某些处理逻辑像权限检查等是很适合的。

Spring 可以管理 web 层，持久层，业务层，dao 层，Spring 可以配置各个层的组件，并且维护各个层的关系，Spring 框架的 7 个模块如图 4.4 所示：

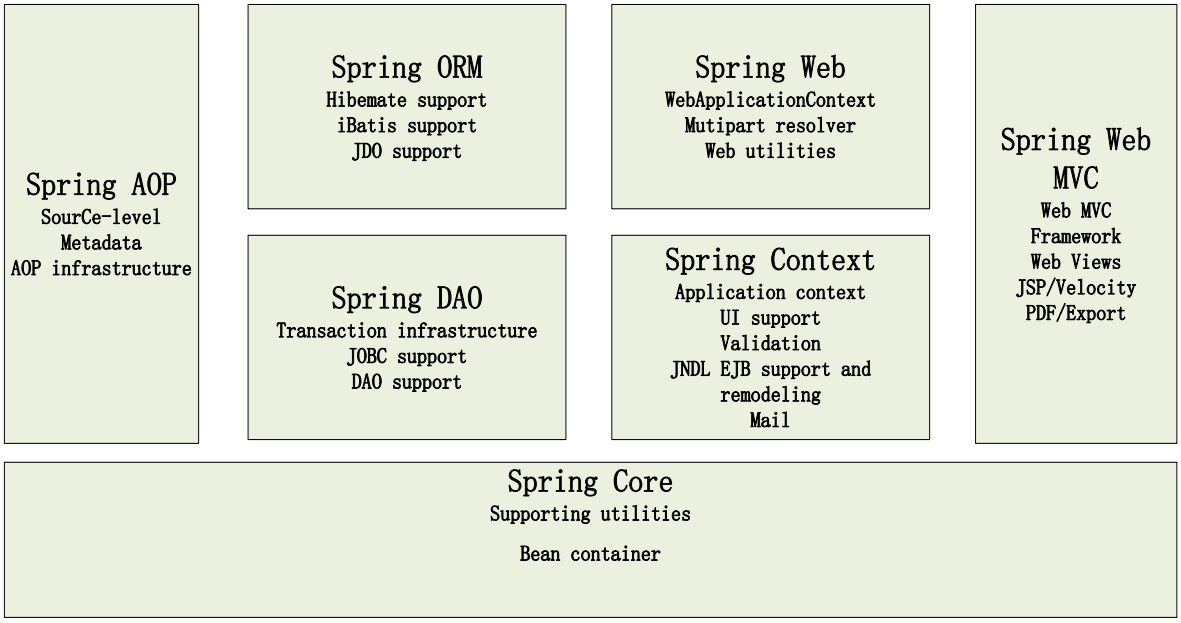


图 4.4 Spring 框架各模块

该项目的后台是一个模拟严选电商优惠券和用户标签为例，其主要用到 Spring 框架的 IOC 和 AOP 模块，其服务层如图 4.5 所示：

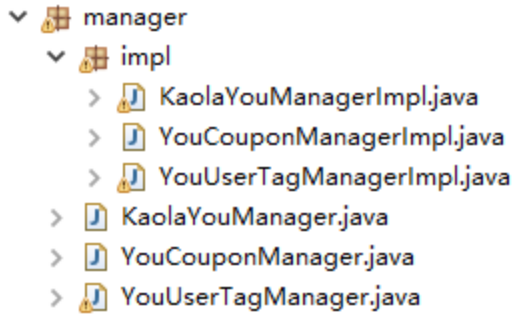


图 4.5 电商后台服务层

其各个文件的作用如下：

1. com.net.mail.statistics.hq.config.manager.YouCouponManager.java 优惠券服务层接口
2. com.net.mail.statistics.hq.config.manager. YouCouponManagerImpl.java 优惠券服务层实现
3. com.net.mail.statistics.hq.config.manager.YouUserTagManager.java 用户标签服务层接口
4. com.net.mail.statistics.hq.config.manager.YouUserTagManagerImpl.java 用户标签服务层实现

5. com.net.mail.statistics.hq.config.manager.KaolaYouManager.java 插入第三方平台数据服务层接口
6. com.net.mail.statistics.hq.config.manager.KaolaYouManagerImpl.java 插入第三方平台数据服务层实现

4.3.3 数据访问（Dao）

数据访问层是电商后台系统与数据库直接交互的一层，所有的优惠券和用户标签的增删改查都在这一层。这一层的处理直接影响与数据库交互的效率，如果处理不当，会大大降低程序的性能。

数据访问层有个主要的任务是去操作数据库。这里用到了 ORM 框架（对象关系映射框架），其最核心的作用就是把表中的一行数据记录和某个实体类的对象之间进行一个映射，利用 ORM 框架，在开发时如果需要操作数据对象，就可以避免使用复杂繁琐的 jdbc 代码，程序员只需要以面向对象的思想操作相应框架的 API 即可[29-30]。ORM 框架有两个最鲜明的特点：第一，对数据进行访问时可以不用管访问细节，封装好的通用数据库交互使得开发人员在与通用数据库交互时比传统的 SQL 语句更简洁方便。从而使整个系统的开发更快速。第二，开发人员使用 ORM 来创造固化数据结构可以更简洁方便。在传统的 Web 应用开发中，开发人员需要把用到的对象模型逐条转化为相应的数据库操作语句，然后在数据库中建立所需的数据库系统。到目前为止，几乎所有的对象关系映射框架都可以利用对象模型创建关系数据库系统。但是，如果以传统的方式开发，正是因为自己需要写 SQL，所以效率一般会更高。

一般在对时间效率要求比较高的互联网公司，一般都是选择以传统的方式来访问数据库。在我们的这个电商后台系统和接口的开发中，我们也选择用传统的 Spring 中的 JdbcTemplate 来访问数据库，程序员自己来写 SQL 代码，另外，为了不经常性的创建数据库连接，我们用到了 DBCP 连接池，因为建立连接是一个非常耗时的过程，从而这样我们可以达到高效。

该项目的后台是一个模拟严选电商优惠券和用户标签为例，其主要用到 Spring 框架的 DAO 模块，在上一节已经画过了 Spring 的各个模块，这里就不再画图，其数据访问层如图 4.6 所示：

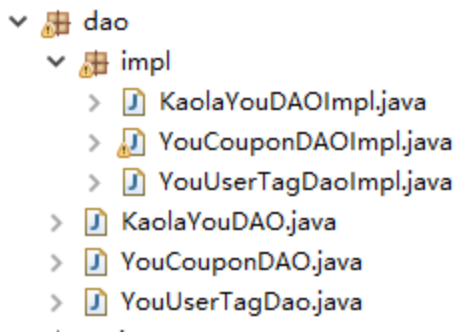


图 4.6 电商后台数据访问层

其各个文件的作用如下：

1.com.net.mail.statistics.hq.config.manager.YouCouponDAO.java 优惠券 DAO 层接口

2.com.net.mail.statistics.hq.config.manager. YouCouponDAOImpl.java 优惠券 DAO 层实现

3.com.net.mail.statistics.hq.config.manager.YouUserTagDao.java 用户标签 DAO 层接口

4.com.net.mail.statistics.hq.config.manager.YouUserTagDaoImpl.java 用户标签 DAO 层实现

5.com.net.mail.statistics.hq.config.manager.KaolaYouDAO.java 插入第三方平台数据 DAO 层接口

6.com.net.mail.statistics.hq.config.manager.KaolaYouDAOImpl.java 插入第三方平台数据 DAO 层实现

4.4 电商后台各模块实现

4.4.1 登录功能模块

虽然本应用后台属于公司内部人员使用的电商后台产品，目的是让运营人员进行优惠券和用户标签的配置。但是为了保证用户的信息安全以及控制哪些运营人员有权限可以来查看和配置，因此需要有一个登录的功能。此外，当系统出现异常时，通过看日志可以查看到具体是哪个同学的操作出现了异常，便于开发人员对异常进行追踪。登录功能的基本流程如图 4.7 所示。

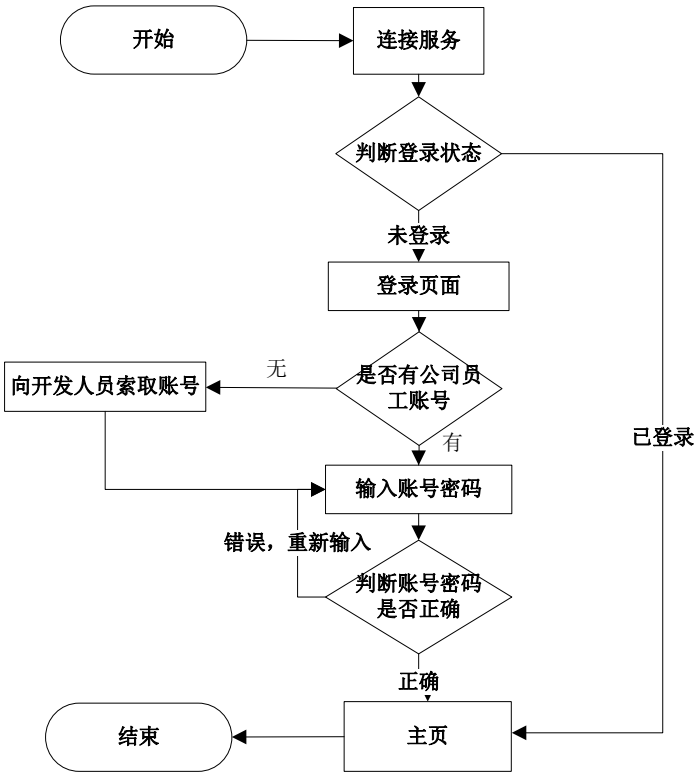


图 4.7 后台登录功能模块基本流程图

(1) 进入后台应用程序之后系统首先会判断用户账号是否已经登录，如果用户已经成功登录，并且没有退出登录，则连接后台后就会直接跳过登入界面，进入主页。否则，如果已经断开连接或关闭浏览器，则先进入登录界面。

(2) 输入正确的账号和密码是登录的必须条件，如果是新员工，还没有账号，则可以向 HR 申请一个新账号或向开发人员索取超级账号，然后再进行后台系统登录。

(3) 如果忘记密码也可以找 HR 来进行找回，这个是公司的员工管理系统，不在本论文中讨论。

(4) 如果在输入登入信息没有输入直接输入，则出现相应的错误提示，以便于用户修正；如果输入错误，则又跳到登录页面，让员工重新输入。

在用户登录以后，会把用户名和密码放入 HttpSession 域中。每次的请求都判断 Session 域中有没有这个用户名和密码，所以在一次会话中，登录后就可以一直正常的使用这个后台系统了。该模块主要的代码截图如图 4.8-4.10 所示：

```
 9  */
10 public class LoginInterceptor extends AbstractInterceptor {
11     private static final long serialVersionUID = -2316687692233075392L
12
13     @Override
14     public String intercept(ActionInvocation actionInvocation)
15         throws Exception {
16         if (ActionContext.getContext().getSession()
17             .get("youLoginInfo") != null) {
18             return actionInvocation.invoke();
19         } else {
20             return "input";
21         }
22     }
23 }
24
```

图 4.8 后台登录功能模块核心代码

```
35
36     public String login() throws Exception {
37         String username = youLoginInfo.getUsername();
38         String password = youLoginInfo.getPassword();
39         if (username == null || username.length() <= 0 || password == null
40             || password.length() <= 0) {
41             return INPUT;
42         }
43         if (adminUsername.equals(username) && adminPassword.equals(password)) {
44             ActionContext.getContext().getSession().put("youLoginInfo",
45                 youLoginInfo);
46             return SUCCESS;
47         } else {
48             return INPUT;
49         }
50     }
```

图 4.9 后台登录功能模块核心代码

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC
        "-//Apache Struts//XWork Validator 1.0.3//EN"
        "http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">
<validators>
    <field name="username">
        <field-validator type="requiredstring">
            <param name="trim">true</param>
            <message>用户名不能为空!</message>
        </field-validator>
    </field>
    <field name="password">
        <field-validator type="requiredstring">
            <param name="trim">true</param>
            <message>密码不能为空!</message>
        </field-validator>
    </field>
</validators>
```

图 4.10 后台登录功能模块核心代码

4.4.2 增删改查功能模块

公司运营人员进行优惠券和用户标签的配置。这就需要和数据库打交道，不单单是插入和查询，因为当运营人员配置出错的时候，还需要进行删除甚至是修改。这就需要相应的增删改查这些所有的操作。增删改查功能的基本流程如图 4.11 所示。

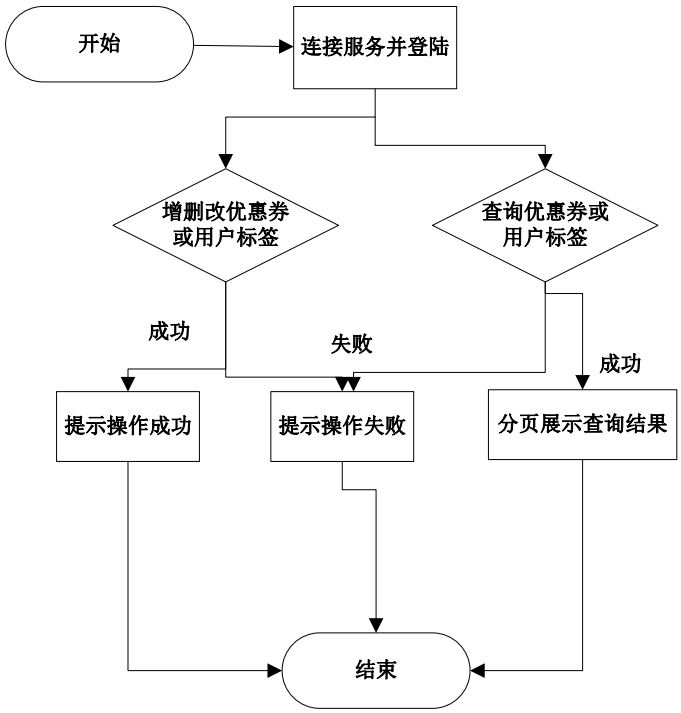


图 4.11 后台增删改查功能模块基本流程图

(1)进入后台应用程序之后并登录，自动会跳到主页，其有增删改查所有造作，而其中的查询有单个优惠券和用户标签的查询也有所有用户标签和优惠券的查询。

(2)其增删改都会提示操作成功或失败，特别需要提出的是，在增加成功后还有展示其所有的优惠券或用户标签并把刚刚配置的展示在最前面一行，而删除和更新是不会展示所有优惠券和用户标签的，为了方便和流程图好看，在上述流程图中并没体现出来。

在公司运营人员做相应的任何操作时，后端代码都在相应的与数据库交互以达到运营人员想要的结果。在这里，由于优惠券和用户标签的逻辑处理很类似，只不过是操作的数据库表不一样而已，由于代码量比较大，所以我在这里只展示查询所有优惠券代码作为代表，该模块主要的代码截图如图4.12-4.13所示：

```
357
358     public void searchAllCoupon() {
359         Set<SearchResult> searchResult = new TreeSet<SearchResult>();
360         logger.info("[invoke method searchAllCoupon...]");
361         try {
362             searchResult = youCouponManager.getAllResult();
363         } catch (YouCouponManagerException e) {
364             logger.error("coupon search all  error,{}", e);
365         }
366         ActionContext.getContext().put("searchResult", searchResult);
367         if (searchResult == null || searchResult.size() == 0) {
368             ActionContext.getContext()
369                     .put("searchAllStatus", "searchAll_empty");
370
371             logger.debug("search result is empty");
372         } else {
373             ActionContext.getContext().put("searchAllStatus",
374                     "searchAll_success");
375         }
376     }
377
```

图 4.12 后台增删改查功能模块核心代码

```
33 public String SEARCH_ALL_SQL = "select * from TB_HQUERY_YOU_COUPON";
34
35 public String SEARCH_BY_TAG_SQL = "select * from TB_HQUERY_YOU_COUPON where tag=?";
36
37 public String DELETE_SQL = "delete from TB_HQUERY_YOU_COUPON where id=?";
38
39 public String DISABLE_SQL = "update TB_HQUERY_YOU_COUPON set status=abs(status-1),updateTime=? where id=?";
40
41 public String ADD_SQL = "insert into TB_HQUERY_YOU_COUPON (createTime,updateTime,tag,startTime,endTime,activationCode,hdfs_s
42
43 public String UPDATE_SQL = "update TB_HQUERY_YOU_COUPON set updateTime=?,tag=?,startTime=?,endTime=?,activationCode=?,hdfs_s
44
45 /* 查询所有优惠券信息 */
46 @Override
47 public List<SearchResult> searchAllCouponInfo() {
48     List<SearchResult> searchResult = this.getJdbcTemplate()
49         .query(SEARCH_ALL_SQL, new SearchResultRowMapper());
50     return searchResult;
51 }
```

图 4.13 后台增删改查功能模块核心代码

4.4.3 内存缓存功能模块

为了在开发优惠券和用户标签相关的接口中，接口的响应速度更快，我们需要定期把数据库中的优惠券和用户标签信息加载进内存当做缓存，在这里，第一，我们就需要用到线程池技术，为了使用习惯，我在这里对线程池进行了二次封装；第二，为了再项目启动时内存缓存功能就有效，在这里我利用了 Spring 框架中的初始化 Bean 即 InitializingBean 来实现。内存缓存功能的设计基本流程如图 4.14 所示：

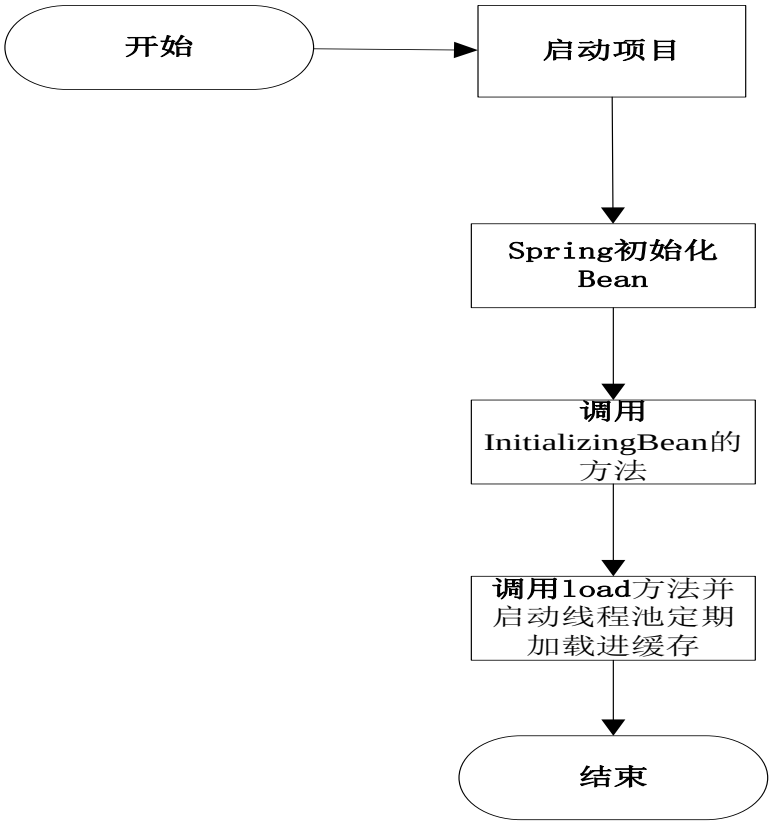


图 4.14 内存缓存功能的设计模块基本流程图

(1)整个项目开发好后，启动应用程序之后，自动会进行 Spring 容器的初始化工作。

(2)其优惠券和用户标签的接口主要是对普通用户生效，但是在公司，有时出现异常时或者在发送优惠券之前，运营人员都需要进行测试确认，所以在后台中加入了该接口的测试功能，而该接口就需要用到内存缓存的优惠券信息。用户标签也是一样，这里就不做解析了。

(3)不管是优惠券的内存缓存还是用户标签的内存缓存，其实都是通过线程池定时把存储在数据库中的相关信息加载到HashMap对象或者ConcurrentHashMap对象中去。

在进行相关接口的开发时，虽然优惠券和用户标签的处理逻辑不一样，但是优惠券和用户标签的内存缓存功能是很类似的，由于代码比较多，所以我在这里只展示优惠券内存缓存功能的核心代码作为代表，该模块主要的代码截图如图4.15-4.18所示：

```
@Service
public class YouCouponCheckServiceImpl implements YouCouponCheckService,
    InitializingBean {

    private Logger logger = LoggerFactory.getLogger(getClass());

    public static ConcurrentHashMap<String, HashSet<CouponInfoSQL>> CACHE = new

    @Autowired
    public YouCouponCheckDAO youCouponCheckDAO;

    @Override
    public void afterPropertiesSet() throws Exception {
        // 执行定时任务
        BackgroundHelper.initialize();
        LoadFromDBTimer.getInstance().load();
    }
```

图 4.15 内存缓存功能模块核心代码

```
    public void load() {
        /* 执行延时加载前先加载 */
        QueryRunner queryRunner = new QueryRunner(YouJDBCUtils.getDataSource());
        List<CouponInfoSQL> couponInfoResult;
        Properties properties = new Properties();
        try {
            properties.load(LoadFromDBTimer.class.getClassLoader()
                .getResourceAsStream("config.properties"));
            DELAY = Integer.parseInt(properties.getProperty("delay"));
            couponInfoResult = queryRunner.query(SQL_QUERY,
                new BeanListHandler<CouponInfoSQL>(CouponInfoSQL.class));
            YouCouponCheckServiceImpl.CACHE.clear();
            for (int i = 0; i < couponInfoResult.size(); i++) {
                //加入缓存
                CouponInfoSQL CouponInfoSQL = couponInfoResult.get(i);
                /* status=0生效，status=1失效 */
                int status = CouponInfoSQL.getStatus();
                if (status == 0) {
                    //解析成功,只有没有禁用的记录才会加载到缓存
                    HashSet<CouponInfoSQL> hs = YouCouponCheckServiceImpl.CACHE
                        .get(CouponInfoSQL.getTag());
                    if (hs == null || hs.size() == 0) {
                        HashSet<CouponInfoSQL> CouponInfoSet = new HashSet<CouponInfoSQL>();
                        CouponInfoSet.add(CouponInfoSQL);
                        YouCouponCheckServiceImpl.CACHE.put(
```

图 4.16 内存缓存功能模块核心代码

```
                        YouCouponCheckServiceImpl.CACHE.put(
                            CouponInfoSQL.getTag(), CouponInfoSet);
                    } else {
                        HashSet<CouponInfoSQL> CouponInfoSet = YouCouponCheckServiceImpl.CACHE
                            .get(CouponInfoSQL.getTag());
                        CouponInfoSet.add(CouponInfoSQL);
                        YouCouponCheckServiceImpl.CACHE.put(
                            CouponInfoSQL.getTag(), CouponInfoSet);
                    }
                }
            }
        } catch (Exception e) {
            logger.debug("loadFromDBTask error", e);
        }
        /* 进行延迟加载 */
        BackgroundHelper.scheduleTaskWithFixedDelay(new LoadFromDBTask(),
            DELAY, DELAY, TimeUnit.SECONDS);
    }
```

图 4.17 内存缓存功能模块核心代码

```
     */
    public static synchronized void initialize() {
        if (backgroundExecSrv != null)
            return;

        // 默认5个核心线程
        int coreSize = 10;

        // 默认5个核心线程
        maxWaitingTaskCount = 20;

        LOG.info("configuration <{}>={}",
            CONF_BACKGROUND_MAX_WAITING_TASK_COUNT, maxWaitingTaskCount);

        backgroundExecSrv = Executors.newScheduledThreadPool(coreSize,
            new DefaultDaemonThreadFactory());
    }
```

图 4.18 内存缓存功能模块核心代码

4.4.4 传输数据加密校验功能模块

考虑到在电商系统中，用户的某些信息非常敏感，为了提高通信传输中的安全性，可以在传输数据之前，就用加密算法对原始数据进行加密，这样在传输中的数据就是加密后的数据；同时，有时在写接口时加密算法也用作校验功能的作用，用来确定约定好的人才可以调用我的接口。比如电商公司在第三方平台售卖的流水数据就用到了这个校验功能，其主要的代码截图如图 4.19-4.20 所示：

```
public boolean isConfig(String sign, String data, String notify_type,
    String notify_typeProp) {
    // 对data进行md5加密
    MessageDigest messageDigest = null;
    try {
        messageDigest = MessageDigest.getInstance("MD5");
    } catch (Exception e) {
        logger.error("get MD5 code error,{}", e);
        throw new RuntimeException("Get Md5 digest fail", e);
    }
    //使用指定的字节数组更新摘要
    messageDigest.update(data.getBytes());

    //通过执行诸如填充之类的最终操作完成哈希计算。
    byte b[] = messageDigest.digest();
    String hs = "";
    String stmp = "";
    for (int n = 0; n < b.length; n++) {
        stmp = Integer.toHexString(b[n] & 0XFF);
        if (stmp.length() == 1) {
            hs = hs + "0" + stmp;
        } else {
            hs = hs + stmp;
        }
    }
```

图 4.19 传输数据加密校验功能模块核心代码

```
        for (int n = 0; n < b.length; n++) {
            stmp = Integer.toHexString(b[n] & 0XFF);
            if (stmp.length() == 1) {
                hs = hs + "0" + stmp;
            } else {
                hs = hs + stmp;
            }
        }

        Calendar cal = Calendar.getInstance();
        cal.setTime(new Date());
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        cal.set(Calendar.MILLISECOND, 0);
        String signProp = hs + cal.getTime().getTime();

        if (signProp.equals(sign) && notify_type.equals(notify_typeProp)) {
            if (logger.isInfoEnabled()) {
                logger.info("[Parameters is matche]");
            }
            return true;
        } else {
            return false;
        }
    }
```

图 4.20 传输数据加密校验功能模块核心代码

另外，数据传输的加密算法一般是对用户密码等敏感信息进行加密，其主要代码截图如图 4-21 所示：

```
public class MD5Utils {
    /* 16进制的各个字符 */
    private static final char[] digits = { '0', '1', '2', '3', '4
            'f' };

    public static String getMD5(String input) { // 数据进行MD5加密
        MessageDigest digest; // 调用Java中MessageDigest类
        try {
            digest = MessageDigest.getInstance("MD5");
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException();
        }
        digest.update(input.getBytes());
        byte[] hash = digest.digest();
        StringBuilder StringBuilder = new StringBuilder();
        for (int i = 0; i < hash.length; i++) {
            StringBuilder.append(getByteAsHexString(hash[i]));
        }
        return StringBuilder.toString();
    }
```

图 4.21 传输数据加密功能模块核心代码

4.4.5 分页功能模块

在开发优惠券和用户标签后台的过程中，有时需要查看所有的优惠券和用户标签信息，因为优惠券和用户标签信息会越来越来多，所以不太可能一次性把所有信息全部展示出来，在个时候我们就需要把优惠券和用户标签数据分页进行展示在前端页面，后端就需要做相应的逻辑处理，然后把查询出来的排好序的数据分页传递给前端页面。

当用户想看某页数据时，后端就传递哪一页的数据给他，为了做好分页工作，就需要处理每页十条记录的展示处理逻辑，还需要对展示的数据按特定字段进行排序，也就是自定义顺序，这时我们可以想到用 TreeSet 集合自定义比较器就可以做到这一点。在这里，我是按照创建时间进行排序，当创建时间相同时，我们以更新时间进行排序。分页功能的设计基本流程如图 4.22 所示：

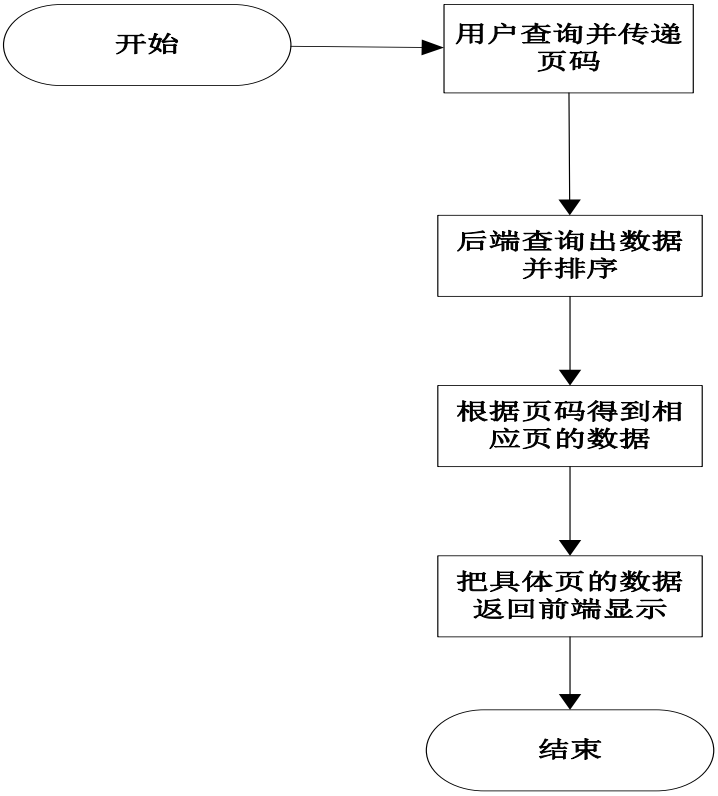


图 4.22 分页功能的设计模块基本流程图

(1)在这个项目中，用户第一次查询时，如果没有选择任何页码，我们在后端的处理逻辑是默认成第一页。

(2)我们的排序是按时间顺序倒序排的，所以用户看到的将是最新配置的用户标签和优惠券信息。

在查询优惠券和用户标签时，虽然查询的具体数据库表不一样，但是在分页时排序规则和处理逻辑都是相同的，所以在这里，由于代码比较多，这里只展示优惠券的分页功能的核心代码作为代表，该模块主要的代码截图如图4.23-4.25所示：

```
public Set<SearchResult> getAllResult() throws YouCouponManagerException {
    List<SearchResult> allRecords = youCouponDAO.searchAllCouponInfo();
    int recordNum = allRecords.size();
    logger.debug("[record size is:{}]", recordNum);
    if (recordNum == 0) {
        return new TreeSet<SearchResult>();
    }
    /* 将查询结果放到set中 */
    Set<SearchResult> recordsSet = new TreeSet<SearchResult>(
        new SearchComparator());
    for (int i = 0; i < allRecords.size(); i++) {
        recordsSet.add(allRecords.get(i));
    }
    /* 将调整好顺序的set放到原来的allRecords中 */
    int count = 0;
    Iterator<SearchResult> recordsSetIt = recordsSet.iterator();
    while (recordsSetIt.hasNext()) {
        allRecords.set(count, recordsSetIt.next());
        count++;
    }
    int pageCount = (recordNum - 1) / DEFAULT_PAGESIZE + 1;
    pageResult.setPageCount(pageCount);
    /* 分页显示 */
    Set<SearchResult> records = new TreeSet<SearchResult>(
        new SearchComparator());
    int pageIndex = pageInputInfo.getPageIndex();
    if (pageIndex < 1) {
        pageIndex = 1;
    }
```

图 4.23 分页功能模块核心代码

```
67          if (pageIndex < 1) {
68              pageIndex = 1;
69          }
70          int startIndex = Math.min((pageIndex - 1) * DEFAULT_PAGESIZE,
71              recordNum - 1);
72          int endIndex = Math.min(pageIndex * DEFAULT_PAGESIZE - 1,
73              recordNum - 1);
74
75          for (int i = startIndex; i <= endIndex; i++) {
76              records.add(allRecords.get(i));
77          }
78          ActionContext.getContext().put("pageResult", pageResult);
79          return records;
80      }
```

图 4.24 分页功能模块核心代码

```
public class SearchComparator implements Comparator<SearchResult> {
    @Override
    public int compare(SearchResult searchResult1, SearchResult searchResult2) {

        if (searchResult1.getCreateTime().getTime() > searchResult2
            .getCreateTime().getTime())
            return -1;
        else if (searchResult1.getCreateTime().getTime() < searchResult2
            .getCreateTime().getTime())
            return 1;
        else if (searchResult1.getUpdateTime().getTime() == searchResult2
            .getUpdateTime().getTime())
            return 0;
        else
            return searchResult1.getUpdateTime().getTime() > searchResult2
                .getUpdateTime().getTime() ? -1 : 1;
    }

}
```

图 4.25 分页功能模块核心代码

4.4.6 properties 文件自动载入功能模块

在项目的实际开发中，为了开发效率的提高、代码的清晰度和后期维护等等诸多原因，有很多的 properties 配置文件，但是这些配置文件在开发中可能经常需要更改，如果每次修改都需要重启项目的话，将会大大降低程序员的开发效率，因此 properties 文件自动载入功能对程序开发来说是很有必要的，其功能的实现的基本流程如图 4.26 所示。

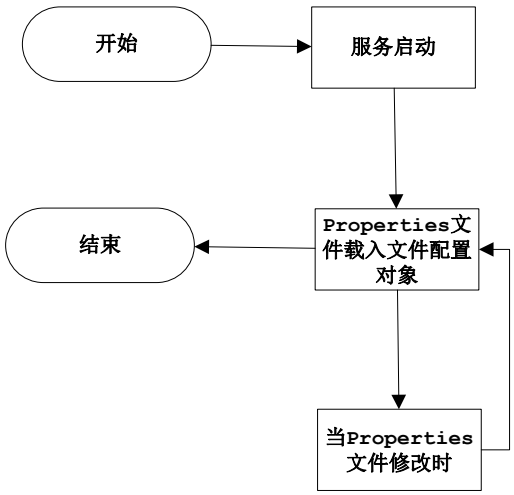


图 4.26 properties 文件自动载入功能模块基本流程图

(1)在进行 properties 文件自动加载的开发中，用到了一个核心类 PropertiesConfiguration，该类是 commons-configuration.jar 包中的其中一个类，专用来处理 properties 文件。

(2)为了自己开发的方便，我自己对 PropertiesConfiguration 这个类进行了二次封装，其会自动去类路径下加载相应的指定文件。

(3)通过该类 PropertiesConfiguration 可以设置加载不同的策略，非常灵活好用，这也是许多程序员喜欢的地方。

在实际项目开发中，有了这个 properties 文件自动载入功能，我们只需去该对象中获取相应配置数据即可，这样会大大节省开发时间，该模块主要的代码截图如图 4.27-4.28 所示：

```
public class PropertiesAutoLoad {

    /**
     * Singleton
     */
    private static PropertiesAutoLoad AUTO_LOAD;

    private static PropertiesConfiguration propConfig;

    private static boolean autoSave = true;

    private static Object syncobj = new Object();

    /**
     * @param propertiesFile
     * @return
     */
    public static PropertiesAutoLoad getInstance(String propertiesFile) {
        if (AUTO_LOAD == null) {
            synchronized (syncobj) {
                if (AUTO_LOAD == null) {
                    AUTO_LOAD = new PropertiesAutoLoad(propertiesFile);
                }
            }
        }
        return AUTO_LOAD;
    }
```

图 4.27 properties 文件自动载入功能模块核心代码

```
    private PropertiesAutoLoad(String propertiesFile) {
        try {
            propConfig = new PropertiesConfiguration(propertiesFile);

            propConfig.setReloadingStrategy(new FileChangedReloadingStrategy());

            propConfig.setAutoSave(autoSave);
        } catch (ConfigurationException e) {

        }
    }
```

图 4.28 properties 文件自动载入功能模块核心代码

4.5 服务器接口开发

根据前面数据库表设计，其实我们知道表中的每条记录都可以和某个实体类的对象进行一一对应，而接口有一个核心部分主要就是实现这个对应关系。YouCouponDAOImpl.java 实现了一个接口 YouCouponDAO，接口包含的方法是各种数据库操作函数。用户标签的操作也是如此，其实后端的所有开发都可以算是接口的开发，不但包括优惠券的领取等，还包括与安卓客户端交互的后端接口开发，只不过除了配置后台页面相关的其他接口，我把它们单独放在另一个项目中。比如：判断优惠券是否可以领取的接口、优惠券和用户标签加载进内存缓存等，项目名是 rtqs-yanxuan-coupon，目的是为了代码清晰，易于维护。其项目目录结构图如图 4.29 所示：

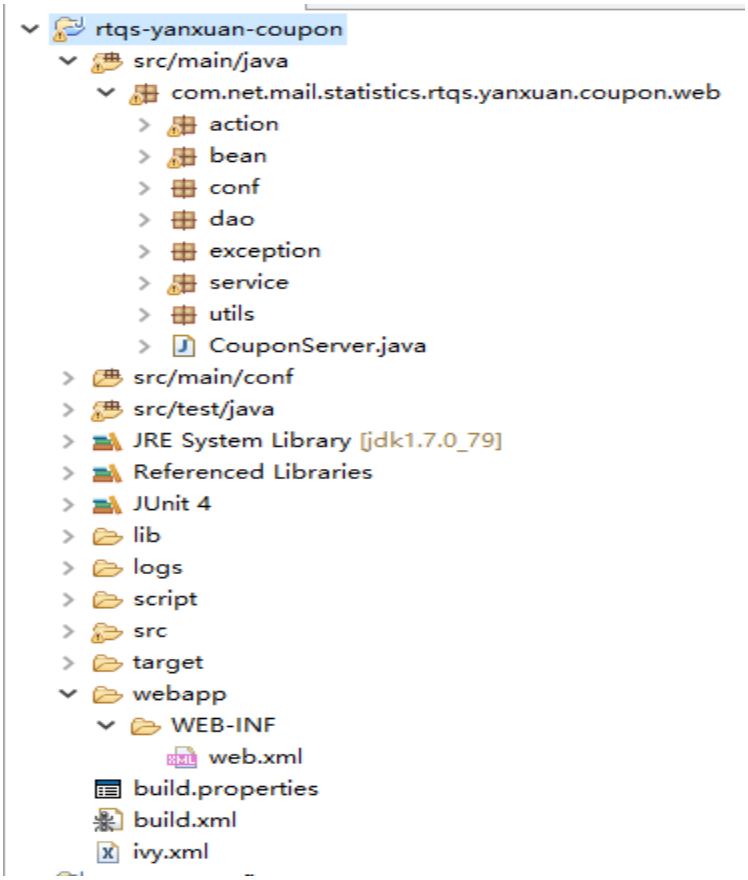


图 4.29 开发与后台无直接关系的接口项目结构图

由于目录结构在前面已经介绍过，这里就不再赘述，其中，一个接口的调用，大致走了如下三步：

1）调用接口，发起请求，把数据按照一定的协议格式提交到后端；

2）后端获取提交过来的数据，并走与 URL 匹配的业务方法；

3）后端处理完相应的操作后再按一定的格式响应给调用方；

比如在领券时，领券页面用户名 username 和优惠券编码 tag 是作为请求参数（请求为 post 方法），如果是服务器在本地的话，请求 URL 为 http://127.0.0.1:18081/you/ YouCouponCheck/ checkCouponByEmail。

由于把所有接口代码和对应的数据库表（用户表、领券详情表等）都截图贴出会比较多，所以在这里只粘贴领券接口各分层核心代码如图 4.30-4.32 所示：

```
public String checkCouponByEmail() {
    // 登录用户名
    email = ServletActionContext.getRequest().getParameter("email");
    tag = ServletActionContext.getRequest().getParameter("tag");
    logger.info("[invoke method checkCouponByEmail, tag:{},email:{}]", tag,
        email);
    String tableName = "";
    String family = "";
    String qualifier = "";
    String result = "";
    HttpServletResponse response = ServletActionContext.getResponse();
    /*************** (1)判断tag和email是否为空 ***************/
    if (tag == null || tag.equals("")) {
        try {
            response.getWriter().append(
                "{\"code\":401, \"msg\": \"tag is empty.\"}");
            logger
                .info("[code:401,tag:,email:,activationcode:,starttime:,endtime:]");
        } catch (IOException e) {
            logger.error("", e);
        }
        return NONE;
    }

    if (email == null || email.equals("")) {
```

图 4.30 领券接口核心代码

```
public String youCouponCheckByEmail(String email, String tag,
    String tableName, String family, String qualifier) {
    HashSet<CouponInfo> couponResultSet = getCouponInfo(tag);
    if (couponResultSet.size() == 0)
        return "403";
    //验证有效期
    HashSet<String> hdfsSiteSet = new HashSet<String>();
    CouponInfo couponInfo = new CouponInfo();
    for (CouponInfo couponInfoResult: couponResultSet) {
        hdfsSiteSet.add(couponInfoResult.getHdfssite());
        couponInfo.setTag(couponInfoResult.getTag());
        couponInfo.setEffectiveDate(couponInfoResult.getEffectiveDate());
        couponInfo
            .setUneffectiveDate(couponInfoResult.getUneffectiveDate());
        couponInfo.setActivationCodeStr(couponInfoResult
            .getActivationCodeStr());
    }
    if (hdfsSiteSet.size() == 0)
        return "403";
    // 获取优惠券信息
    Date effectiveDateRaw = couponInfo.getEffectiveDate();
    Date uneffectiveDateRaw = couponInfo.getUneffectiveDate();

    /* 将优惠券有效期进行格式化 */
    DateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
```

图 4.31 领券接口核心代码

```
25      public boolean checkWhiteList(String email, String tableName,
26          String family, String qualifier, HashSet<String> hdfsSiteSet)
27          throws HbaseTableException {
28          SingleFamilyHBaseTable hbaseTable = null;
29          int maxVersions = 99999;
30          try {
31              logger.debug("hBaseTableFactory connection error:{}",
32                  hbaseTableFactory == null);
33
34              hbaseTable = hbaseTableFactory.getHTable(tableName,
35                  SingleFamilyHBaseTable.class);
36              DataRecord<String, String, String, String> record = hbaseTable
37                  .getDateRecord(email, family, qualifier, maxVersions);
38              //若为空则用户不在白名单中
39              if (record == null || record.isEmpty())
40                  return false;
41
42              Map<Long, String> values = record.getValues(family, qualifier);
43              if (values == null || values.isEmpty())
44                  return false;
45              //检验email是否在白名单中
46              for (Map.Entry<Long, String> entry: values.entrySet()) {
47                  String hdfssiteStored = entry.getValue();
48                  // 查看hbase中的hdfssiteStored是否为当前需要的hdfssite匹配,hdfssite保存在hdfsSiteSet中
49                  for (String hdfssite: hdfsSiteSet) {
```

图 4.32 领券接口核心代码

HttpClient 类是 java 中的 HTTP 协议的封装类，其实就相当于网络通信传输参考模型中的传输层。他可以实现在两个服务器之间的通信，实现服务器 1 与服务器 2 的请求响应，只不过在这个地方，服务器 1 就相当于了一个发起请求的客户端，服务器 2 相当于响应请求的服务端，原理还是和上面中调用接口是一样的。

4.6 本章小结

本章主要解决了电商后台系统的具体实现设计，后台系统使用了 Struts2 MVC 框架和 Spring 框架，为了提高服务器的并发能力，使用 jetty 服务器部署后台应用，再用 Nginx 做反向代理，静态资源由 Nginx 处理，动态资源由后台的 jetty 处理；在数据库操作方面，自己手动编写 SQl 代码来实现高效。再加上前端页面，实现了电商后台系统的开发。并从分层和各个模块功能的实现介绍了该后台系统；最后，为了代码的清晰可维护，单独新建了一个专门编写与页面不直接打交道的接口项目，并对其中的一个接口进行了介绍。

第五章 安卓客户端与大数据

完成了整个后台系统和接口的开发以后，我们需要用户来购物，这时就需要开发面向用户的客户端程序，当客户端和服务端接口进行交互时，又会产生许多的日志文件，经过日志文件的清洗工作，又能更好的为电商公司服务，在这一章，前三节将以优惠券为例介绍安卓客户端，后三节简单介绍了大数据，然后以领取优惠券的张数为例用次数统计做了一个实际应用。

5.1 Android 系统架构

Android 系统采用了分层架构，如图 5.1 所示：

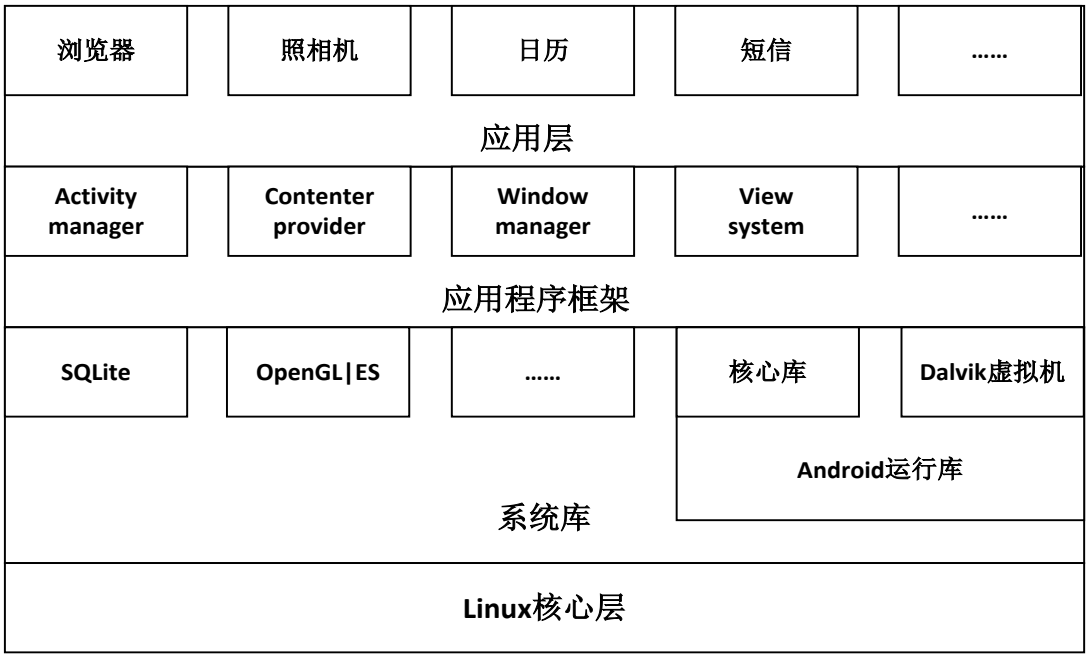


图 5.1 Android 系统架构

应用层包括 Android 自带的浏览器、照相机、短信等核心应用；下一层的应用程序框架层包含了可以供 Android 开发人员直接调用的 API 框架，对于开发人员来说，这大大降低了开发难度[31]。

Dalvik 虚拟机是安卓系统的重要组成部分，Android 的 dalvik 虚拟机不同于 Java 的 JVM 虚拟机，dalvik 虚拟机采用了即时编译技术（JIT 技术）可以使虚拟机执行效率提高 5 倍[32]。

每个 Android 应用在运行时，系统都会为它分配一个唯一的 ID，一个 ID 可以识别一个应用程序，应用程序代码由对应的一个 dalvik 虚拟机执行，也就是说，应用程序和虚拟机是一对一的关系[33]。

Android 系统是基于 Linux 的，Linux 核心层为上层系统提供内存管理、进程/线程管理等机制。

5.2 Android 四大核心组件

安卓系统有四大核心组件，分别是：活动（activity）、服务（service）、内容提供者（contentprovider）、广播（broadcast），在这里我只对活动（activity）和服务（service）做简单理论介绍，更多的介绍留给以优惠券为例的安卓客户端的具体功能实现。

活动 Activity 是 Android 中最常用的应用组件，是 Android 应用中重要的组成部分。Android 是使用任务（Task）来管理活动的，一个任务就是一组存放在栈里的活动集合，这个栈也被称作返回栈（Back Stack）。栈是一种后进先出的数据结构，在默认情况下，每次启动的活动都会压入返回栈的栈顶，而每次销毁一个活动时，处于栈顶的活动都会出栈，这时前一个入栈的活动就会重新处于栈顶的位置，系统总是会显示处于栈顶的活动给用户[34-35]。

活动 Activity 的生命周期有运行状态、暂停状态、停止状态和销毁状态四种，七个重要方法，分别为 onCreate()、onStart()、onResume()、onPause()、onStop()、onDestroy()、onRestart()，图 5.2 描述了 Activity 活动组件的生命周期[34-35]。

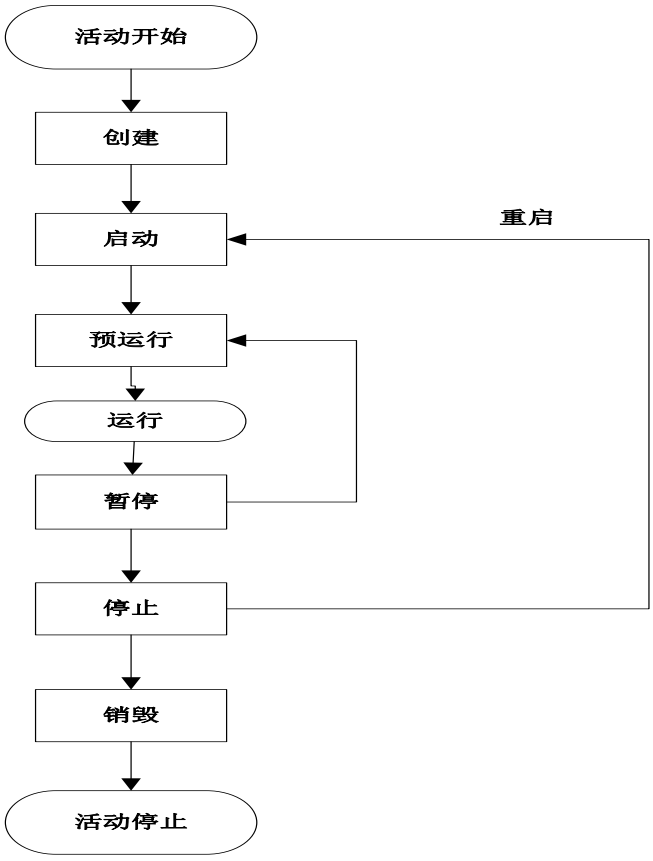


图 5.2 Android 活动生命周期

服务（service）是处于主线程的 Android 组件之一，一方面拥有 service 的进程优先级高，当安卓系统内存不足时，一般来说，系统最先回收的系统资源不会是 service 存在的进程所占用的系统资源；另一方面其一般是在后台执行一些耗时的操作，但依然可以处于不允许耗时太久的主线程中，这是因为其实 service 是新建子线程来执行这些耗时的操作的，这些子进程只是依附于 service 而已，即 service 只是在管理耗时操作的子进程，并没有本身执行这个耗

时过程。服务有两种状态，启动态和绑定态，都可以通过相应的 API 来启动，具体服务生命周期如图 5.3 所示：

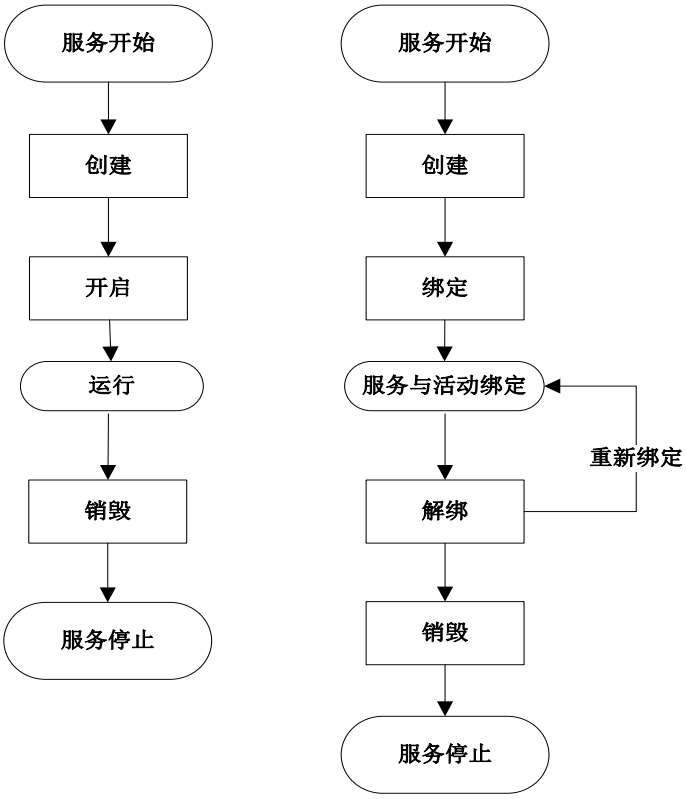


图 5.3 Android 服务生命周期

5.3 Android 客户端各模块设计

由于在下一章将直接粘贴客户端的调试结果，所以，在这一小节，我们只讨论安卓客户端的设计思想，不粘贴客户端的具体实现代码。

5.3.1 登录注册功能的设计

虽然该论文在后台部分模拟了严选电商中的优惠券和用户标签，但是在安卓客户端我主要做了优惠券界面和后端进行交互，为了该系统今后的可扩展性，可能会加入支付、聊天等功能，首先就需要有一个登录的功能，以保证用户的信息安全。此外，用户使用自己的信息来注册也可以便于对用户的信息化管理。登录注册功能的基本流程如图 5.4 所示：

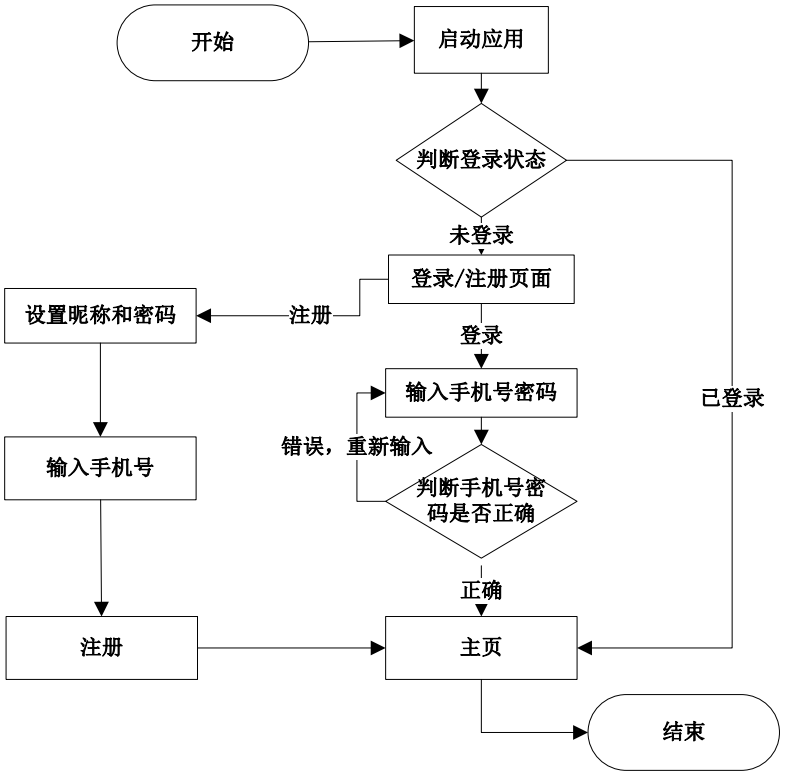


图 5.4 登入注册功能模块基本流程图

(1) 进入应用程序之后系统首先会判断用户账号即个人手机号是否已经登录，如果用户在上次使用此软件时成功登录，并且没有退出登录，则打开本软件后就会直接跳过登入界面，进入主页。如果是首次安装软件，或者之前安装了但从来没有登录过亦或是上次登录了但执行了注销账号操作，则先进入登录界面。

(2) 输入正确的账号和密码是登录的必须条件，如果是新用户，还没有账号，则可以点击“注册”按钮来申请一个新账号。注册账号时需要输入正确的电话号码和设置自定义的昵称、密码等信息。

(3) 如果用户手机号已经注册过本软件的账号，再次注册时，系统将会提示已经注册过。

(4) 如果在输入登入信息或者注册信息时有错误，则出现弹框提示错误所在，以便于用户修正。

在用户登录以后，会保持一段时间的在线状态。而本设计在 Android 端和后端使用 HTTP 协议，连接后不久就会断开。因此，登录功能设计成 30 秒向服务器重新发送一次请求。

5.3.2 优惠券兑换和展示功能的设计

在我的严选电商模拟中，安卓客户端优惠券界面和后端进行交互是我的重中之重，当用户在兑换时，后端接口就要判断该用户是否能领取该张优惠券，如果能领取，要让用户看到领取到的优惠券，否则，提示不能领取该优惠券的具体原因。所以客户端需要传递用户名和优惠券编码给后端服务器进行处理，所以在兑换优惠券之前一定要先登录客户端。具体优惠券兑换功能的基本流程如图 5.5 所示：

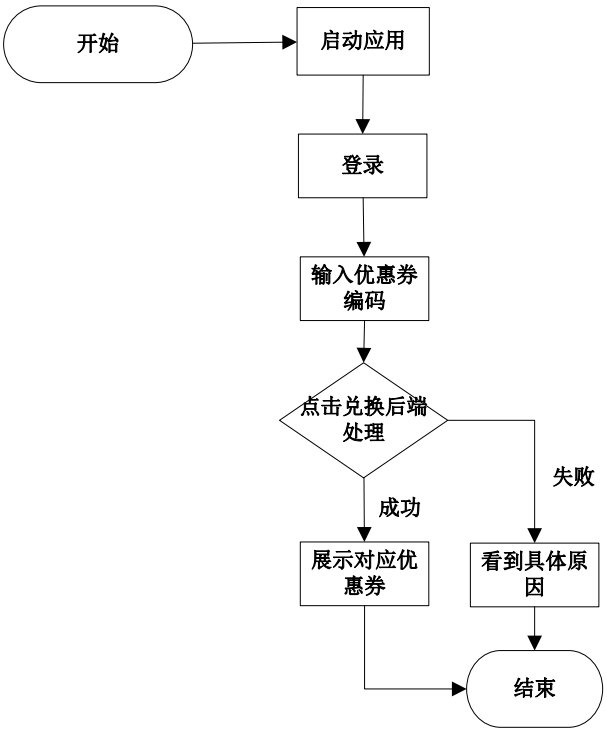


图 5.5 优惠券兑换功能模块基本流程图

(1) 在搭建安卓开发之后，任何界面都可以通过一个 xml 文件生成，放在 src-》main-》res-》layout 目录下。

(2) 登录后点击优惠券兑换就能跳到优惠券兑换界面，输入优惠券码，当点击兑换按钮时就提交了该编码和用户名给后端进行了处理。

(3) 处理后的结果返回相应优惠券，然后客户端通过相应的 java 逻辑代码处理后激活对应的优惠券展示给用户。

由以上可知，一张优惠券的领取客户端和后端都要起到作用，后端更注重逻辑处理，判断某个用户是否能领取某张优惠券，而客户端注重优惠券的展示和美观，当然也能处理一点简单的逻辑问题，比如去加载具体哪些优惠券等等。

5.4 大数据之计算 MapReduce

使用 Hadoop 来分析处理数据，为了充分利用 Hadoop 提供的并行处理优势，我们需要利用 MapRedce 作业。因为作业部署到在集群上运行，这样可以大大提高并行计算处理能力和速度，在进行处理之前，还需要做一些准备，比如由输入格式化 InputFormat 负责产生输入分片并将他们分割成记录[37]。

MapReduce 计算框架分为两个计算步骤，第一步先经过 map 阶段处理，第二步再经过 reduce 阶段的处理，输入输出的键值对具体类型由程序员自己来选择[36]。在 map 阶段转到 reduce 阶段的过程中，有一个非常重要的 shuffle 过程，也就是常说的排序和分组，这一过程是对 map 输出的键值对按键进行排序和分组的，经过这个 shuffle 过程的处理，就变成了

reduce 阶段的输入[37]。同时在大规模数据处理计算时，也有着非常强大不同等级的的容错机制。

在实际的开发的 Java MapReduce 中，程序员需要重写两个函数：map 函数和 reduce 函数，这两个函数都定义在两个不同的接口中。总而言之，mapreduce 其实就是一个移动式的基于 key-value 形式的分布式计算框架。整个数据流如图 5.6 所示[52-56]：

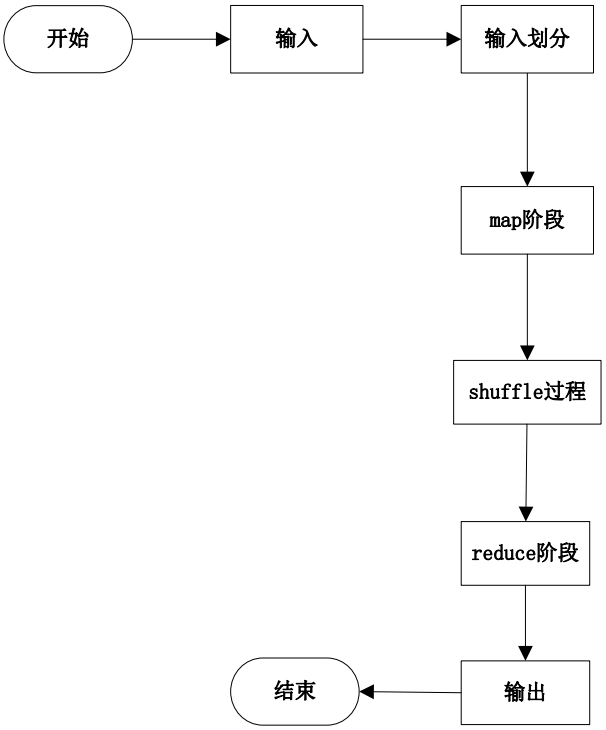


图 5.6 MapReduce 的逻辑数据流

MapReduce 框架的作业运行流程如下[38-40]：

第一步：首先如果要跑 MR 任务，那么这个计算平台的客户端就必须开启一个任务。

第二步：该 MR 任务向作业跟踪器申请一个作业 ID。

第三步：把这个 MR 任务需要的已知文件数据、输入数据的划分信息和各种与其相关的配置文件加载进 HDFS。

第四步：作业跟踪器接收到 MR 任务后，作业调度器会根据自己的算法从任务队列里调度任务，并且为每一个文件块分配一个 map 任务给相应的任务跟踪器去执行该任务。

第五步：任务跟踪器会定期上报该任务的相关执行信息给作业跟踪器，在 MR 中，这个上报过程称之为心跳。

第六步：当作业跟踪器得知该 MR 任务完成时，会自动将该 MR 任务的状态设置成成功状态，这时，当在客户端查询该 MR 任务状态时，就能看到 MR 任务已跑成功。

5.5 大数据之存储 HDFS

HDFS(Hadoop Distributed File System)是一个分布式文件系统,一方面具有高度容错性的特点，另一方面当数据集的大小超过一台物理计算机的存储能力时，就有必要存储到不同

的计算机上，大的数据量就需要提供高吞吐量的数据访问，并且这个分布式文件系统可以部署在廉价商用计算机上，进而可以大大降低企业成本。HDFS 是根据流式数据访问模式来存储大文件的,运行在商用硬件集群上。

HDFS 集群含有 1 个名称节点(NameNode)和 N 个数据节点(DataNode)，名称节点 NameNode 是在一台单独机器上运行的程序,负责 HDFS 文件系统的名称空间的管理与外部客户机访问的控制，以及决定是否将文件映射到 DataNode 数据节点上,数据节点根据具体请求做出相应响应[41] ,HDFS 的主要概念如下:

(1)Block: 不管是磁盘还是文件系统的，都会有块的概念，文件系统是构建于磁盘之上的，可以通俗的理解，文件系统的块是通过磁盘块来进行管理文件的，因此，文件系统中块的大小一般是磁盘块的整数倍，在现在的一般计算机中，磁盘块的大小一般为 512 个字节。在文件系统中,大文件会被存储在多个文件块 Block 块,在默认情况下,文件块的大小为 64M,每一个 Block 块在多个 Datanode 上存储着副本。也就是说，当某个硬件节点出现故障时，该文件系统还能正常进行工作，一般来说，对存储文件而言,为了存储 Block 块的操作更加简单，Block 块的大小一般是大于磁盘块大小的，这是因为这样可以降低寻址时间。

(2)名称节点和数据节点:文件系统以树的形式存在，并由名称节点维护文件和索引目录信息，这些信息以名称空间镜像保存在名称节点的磁盘中，而文件系统的元数据存储在名称节点的内存中，因此不适合存储大量的小文件，这是因为文件总数是受到名称节点内存大小的限制；另一方面名称节点(NameNode)记录存放块的数据节点(DataNode)，数据节点才是是真正存储数据信息的节点，这就相当于名称节点是指挥者,数据节点是执行者一样，当名称节点发出调用命令后，数据节点就会提供块的位置发送存储块的信息[42]。

(3)命令行接口: HDFS 有许多接口，但命令行是最简单的，也是许多开发者最熟悉的，在配置好 HDFS 文件系统后，可以执行所有常用的文件系统操作，例如读取、新建、移动、删除等。

HDFS 也是按照主从结构思想来进行设计的，分为主节点（NameNode）、（二级名称节点）SecondaryNameNode 和（数据节点）DataNode，如图 5.7 所示：

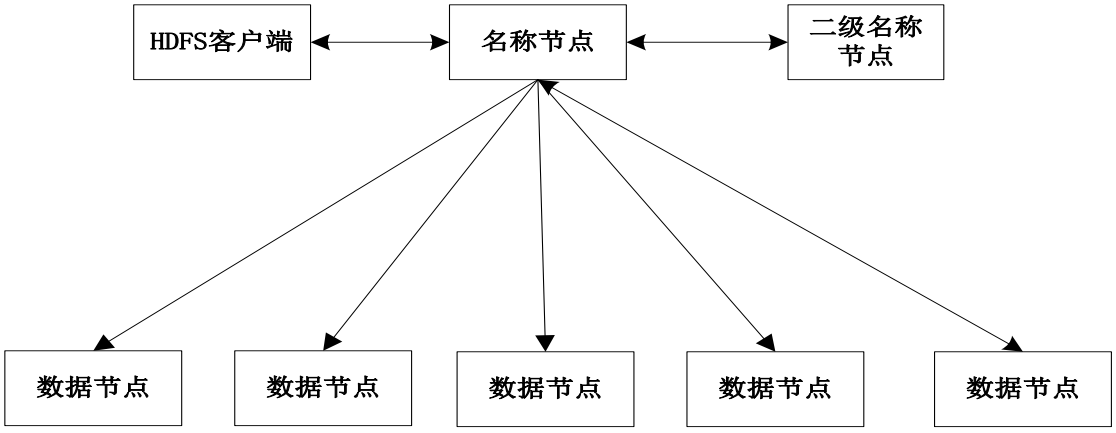


图 5.7 HDFS 的结构图

上图解析如下：

1. 名称节点：是个主节点，主要负责对客户端的读写请求进行调度处理和管理数据块的映射关系等[43]。
2. 二级名称节点：这是一个主节点的二级辅助节点，为主节点做事情后再响应给主节点，主要是对元数据的操作日志和镜像文件进行处理[43]。
3. 数据节点：我们称之为从节点，不管是读操作还是写操作，在真正读写的时候，都是这个节点以 block 的方式来执行的[43]。
4. HDFS 客户端：是用户和 HDFS 交互的主要形式。

5.6 大数据之数据库 HBase

HBase 是基于 HDFS 上开发出来的面向列的分布式数据库，在海量数据的实时查询分析方面，有着得天独厚的优势，例如对消息进行实时分析等。对于 HBase 的部署来说，HBase 不但要求要 HDFS，还需要 Zookeeper 的支持。在分布式环境中，ZooKeeper 就相当于一个指挥官的作用在为分布式环境进行服务，例如同步锁服务和配置服务等[44-45]。HBase 不直接支持 SQL 的语句查询，本身只提供了 Java 的 API 接口。

前面我们提到过，HBase 是构建于 HDFS 之上，当然也可以写入本地文件系统，其实在默认情况下就是写入本地的，不过这样的 HBase 就成了单机模式，只能运行在一台机器上，那么对于分布式大数据的环境来说，单机模式是没有意义的。如果要使用 HBase 集群，也就是分布式的生产环境中，HBase 的存储配置需要 HDFS 作为其基础的存储设施。在 HBase 的上层中，提供了访问的数据库数据的 Java API，供应用程序访问存储在 HBase 数据库中的数据。

在 HBase 集群中，主要由 Master 和 Region Server 组成，同时也依赖于 Zookeeper，hbase 的相关模块如下图 5.8 所示[57-60]。

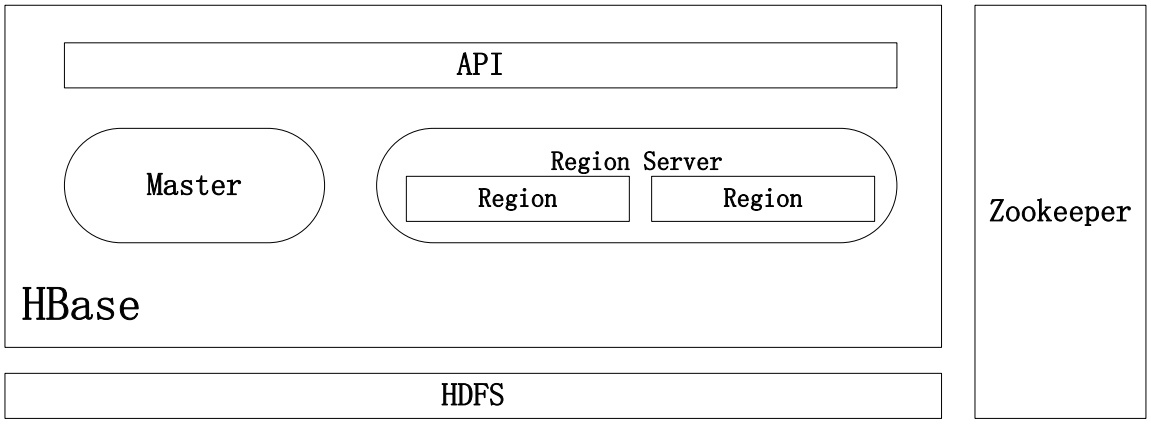


图 5.8 HBase 的相关模块

HBase 特点：

1. 他是一种自底向上进行构建的数据库，能通过增加节点来进行线性拓展。
2. 对表的访问要通过表的主键，而表中的行都是根据主键的字节序进行排序的。

3. 一张表的列族在定义表时就必须预先给出，而具体的列可以后面按需要加入，同一张表中，不同的行可以有不同的列，所以准确的说，HBase 是一个面向列族的存储器。
4. HBase 中的数据都是字符串类型，没有类型，所以 HBase 的数据类型单一。

5.7 大数据的应用之统计与分析

经过上几节的理论，这一节我们以次数统计为例进行实际应用的开发。在电商的日常中，有许多次数统计这样的需求，比如某个 ip 的访问次数、某个按钮的点击次数等等。假设通过优惠券兑换按钮后端接口的调用已经产生了很多的日志文件，现在要对这些日志文件进行清洗，来统计兑换按钮点击的次数，首先我们应该开发一个 MapReduce 任务来解决这类问题。

次数统计的 MapReduce 任务也分为两个阶段，map 阶段和 reduce 阶段，在此之前包括 HDFS 集群在内已经搭建好，两个阶段的核心代码如图 5.9 和 5.10 所示：

```
1294        @Override
1295        public void map(LongWritable key, Text value, Context context)
1296            throws IOException, InterruptedException {
1297            try {
1298                Map<String, String> logMap = getLogMap(key, value, logParser);
1299                if (logMap == null) {
1300                    return;
1301                }
1302
1303                Set<String> keys = count(logMap);
1304                if (keys != null && !keys.isEmpty()) {
1305                    for (String outTag: keys) {
1306                        wirteOutTag(context, outTag, logMap);
1307                    }
1308                }
1309            } catch (Exception e) {
1310                String log = new String(getBytesForText(value), charsetName);
1311                throw new IOException(log, e);
1312            }
1313        }
1314
```

图 5.9 map 阶段的核心代码

```
1456        @Override
1457        public void reduce(GBKText key, Iterable<LongWritable> values,
1458            Context context) throws IOException, InterruptedException {
1459            long count = 0;
1460            for (LongWritable val: values) {
1461                count += val.get();
1462            }
1463
1464            GBKText writeKey = null;
1465            if (addto != null && !addto.isEmpty()) {
1466                writeKey = new GBKText(key.toString() + " " + addto);
1467            } else {
1468                writeKey = new GBKText(key.toString());
1469            }
1470
1471            if (count > minLimit) {
1472                if (isPrintTotal) {
1473                    context.write(writeKey, new Text(StaticFlags.TOTAL_FLAG
1474                        + count + StaticFlags.END_CHAR));
1475                } else {
1476                    context.write(writeKey, null);
1477                }
1478            }
1479
1480        }
```

图 5.10 reduce 阶段的核心代码

由以上可以看出 MapReduce 统计任务的工作流程，先通过日志解析器进行解析返回一个 map 集合，如果有需要可以通过统计规则进行过滤，转换洗牌后给 reduce 去进行整合，结果写到 HDFS 上去即可。次数统计也是如此，不同的统计对应的 map 集合中的键值对也不一样。

5.8 本章小结

本章主要以优惠券为例介绍了安卓客户端和大数据的理论知识和在本论文的具体应用，前三小节阐述了安卓客户端的系统架构与核心组件，以及各个模块的具体实现，从不同的角度理解客户端和后端的交互；后四小节中，前三小节介绍了大数据的基本框架原理和各自的工作流程，紧接着一小节，以点击兑换按钮的次数统计为例，简单介绍了基于大数据的数据清洗过程。

第六章 系统的测试和整体流程

在完成了整个系统的开发工作以后，我们不得不做的事情就是对其进行测试，来保证整个系统能正常的进行工作。整个系统的测试分为功能测试和性能测试，性能测试包括后端接口的压力和 Android 客户端测试，同时对测试结果进行分析也会同步进行，再下一节，我将对整个系统的整体功能结果做简单介绍分析。

6.1 系统的功能测试

功能测试的主要作用是对软件的各个功能进行多次测试并且验证。根据自己编写的测试用例逐条执行，当然也可以不编写测试用例使用 debug 模式直接调试测试，以验证软件是否达到预期要求的功能。功能测试有黑盒测试和白盒测试两种，测试之前只要想好哪些功能需要测试，如果整个软件的内部结构及代码不在测试范围内，这就叫黑盒测试，相反就是白盒测试。一般从软件产品的功能模块出发，在上线之前都要进行功能的测试检测，对于一个开发程序员来说，直接进行功能测试即白盒测试是最方便的，不管写不写测试用例，只需要把观察到的实际结果和预期结果进行对比，如果对比结果相同就说明我们完成了这个功能，当然在功能上完成基本功能的基础上能提出更好的意见和建议就更好了。

6.1.1 电商后台功能测试

以优惠券和用户标签为例的电商后台系统的主要模块分为三个：后台页面的开发、后端接口 API 开发、数据库设计，这三个模块覆盖了后台这个大模块的所有的功能点，为了测试各个模块的具体功能，我在测试之后截取了图片，由于前面章节做实现时已经对实现过程进行了详细的说明，也放了实现过程的相应图片，所以这里我直接放一部分的结果图，其查询全部数据部分和领券接口测试结果部分如图 6.1-6.3 所示：

图 6.1 查询所有用户标签调试结果

图 6.2 查询所有优惠券调试结果

图 6.3 领券接口测试结果

后台系统全部调试完成以后，如果截图全部贴出来会比较多，所以只放了全部查询和根据 tag 和 email 查激活码的领券接口截图结果，比如增删改优惠券和用户标签、按 email 查询某个用户的用户标签等这些都没有贴出。该后台经过多次所有功能的测试，得到的结果与预期基本符合（能改的 bug 都改了）。因此，该系统中后台系统部分基本上实现了预定目标。

6.1.2 Android 客户端和后端接口功能测试

以优惠券为例的 Android 客户端的按功能细分的话，主要的功能模块分为四个：注册、登录、优惠券兑换和查询我的优惠券，这四个功能模块的实现不但有 Android 客户端的开发，也含有对应的后端接口的开发，实现了客户端请求服务端接口响应，其各个模块的测试结果如图 6.4-6.7 所示：

图 6.4 注册模块测试结果

图 6.5 登录模块测试结果

图 6.6 优惠券兑换模块测试结果

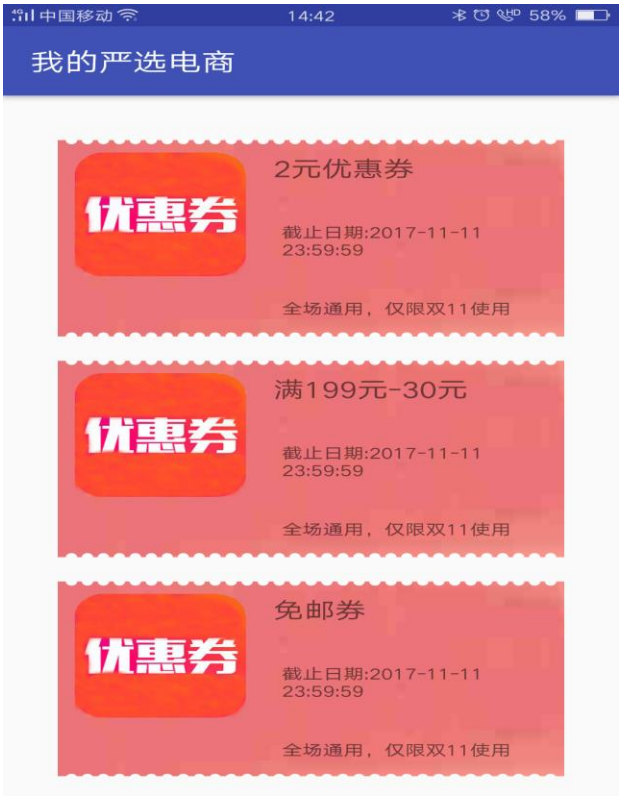


图 6.7 我的优惠券模块测试结果

Android 客户端全部调试完成以后，有以上核心截图可知，Android 客户端与后端服务接口基本功能需求已经实现，该 Android 客户端与后端接口经过多次所有功能的测试，进行了多次代码的修改后最终得到的实际结果与预期结果基本符合。因此，该系统中后端接口和 Android 客户端部分基本上实现了初期设定的目标。

6.1.3 大数据的应用功能测试

在前后端进行交互时产生的日志数据，经过次数统计任务后，某个手机号用户兑换的优惠券张数就被写到了 HDFS 上，得到如下结果图 6.8 所示：

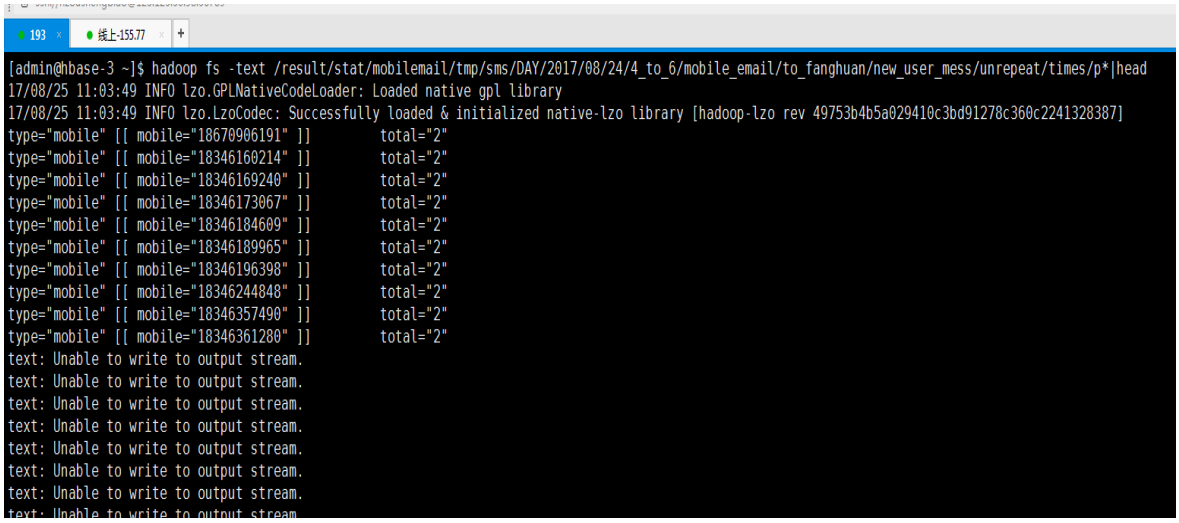


图 6.8 手机号用户兑换的优惠券张数测试结果

由以上结果可知，MapReduce 任务的次数统计已经跑成功，达到了我的预期要求。当运营人员需要查看核心 kpi 时，就能把核心数据反馈给他，从而进一步指导优惠券的配置工作。例如，一天中总共发出的优惠券张数。

6.2 服务端性能测试

性能测试分为负载测试和压力测试，其有多种自动化测试工具，主要是用来模拟真实生活中用户一般性的操作和边界操作等的使用环境，一般来说，在实际软件测试中，负载测试和压力测试会一起使用。利用负载测试，可以得到被测系统在不同的工作负载下的性能，如使工作量逐渐增加，来测试系统是否能按照预期的那样进行工作；压力测试简单的理解就是这个系统的极限所在，比如在该系统中，最多能有多少用户能够同时在线[46-47]。

由于考虑到本系统负载最大的时候可能是因为有很多电商用户同时在线（几百人同时兑换优惠券的这种可能性远小于多用户同时在线），因此测试的主要内容就是测试该系统登录并且保持在线的并发能力。本设计采用 LoadRunner 测试工具来模拟大概 6 分钟左右完成两千次用户登录并且保持多用户在线的实际环境。

作为一个自动化负载测试工具，LoadRunner 通过模拟大量用户来实现并发环境，从而来达到识别和发现问题的目的，所以用这个测试工具能预估出系统性能的大致走势图[48,51]。

在拿到由该测试工具测试出来的测试结果后，可以从多个方面对系统进行分析，具体可以按照如图 6.9 所示的几个方面逐个进行分析，如果项目时间比较仓促，测试时可以只分析并发数和响应时间这两个重要指标[49-50]。但最重要的是，性能测试结果分析的一个重要的原则是以性能测试的需求指标为导向。

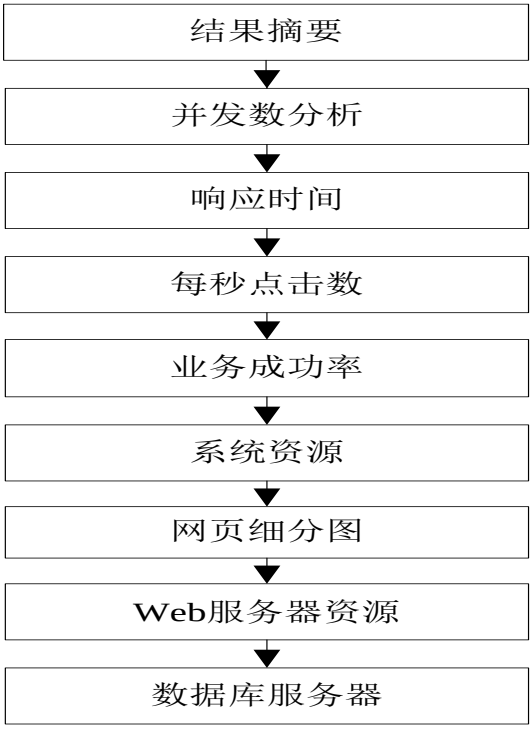


图 6.9 性能测试结果分析过程

如图 6.10 示，LoadRunner 的测试结果首先显示的是该结果的一个摘要信息，包含有测试场景名称“Scenario Name”、存放路径“Results in Session”、持续时间“Duration”、并发数和响应时间等，具体如下：

由图 6.10 可知持续时间为 6 分钟 2 秒，与开始预设的场景持续时间基本相符。Statistics Summary（统计信息摘要）中给出了并发数和平均每秒的请求数等数据信息。

由图 6.10 可知，各项性能指标在下图中一目了然，在这里我就不一一列举了，但值得一提的是，在并发数这一项中，700 对于一个电商服务来说还是小了点的，但是我们在开发和测试时服务器都部署在自己的笔记本电脑上，性能不够强大，因此也可以接受。每秒约 3M 的吞吐量对于登录操作也比较满意。

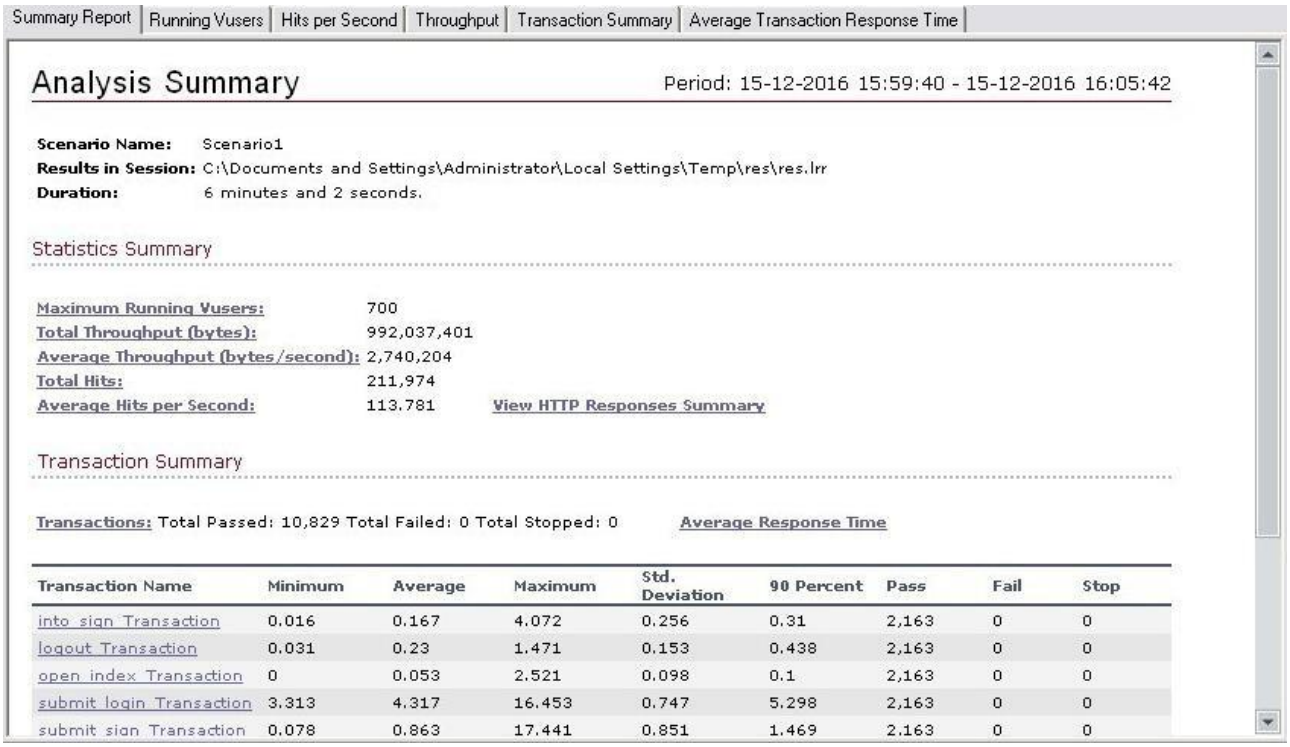


图 6.10 性能测试结果摘要

由图 6.10 可知本次登录响应时间，其中第四行就是 Android 客户端在登录的登录事件。这里设定的响应时间目标值不超过 3 秒，表中 Average 表示的是 Average Time（平均响应时间），其中第四行这一栏中登录部分对应的事件就是“submit login Transaction”，可以看到他的平均响应时间是 4.317s，登录的响应时间长达 4.317 秒，超出了要求。但在理论上，我们应该减去用户输入自己个人数据的时间才是系统真正的登录响应时间，因为在测试时并没有去除，所以，实际响应时间应该是 1.317 秒，小于预期的 3 秒。因此登录业务的事务响应时间也达到了要求。一般在做性能测试时，统计结果会去掉思考时间，加上思考时间能够更实际地模拟实际生活中的真实情况，而在性能测试时，为了能真实测试出服务器的性能，就应该减掉用户输入信息的时间。

6.3 整体功能流程

由于前面章节已经对原理和实现已经做了详细的介绍，并且该章的第一节以图文结合的方式，详细测试了后台的配置系统、Android 客户端的请求和后端接口的响应，并以次数统计为例简单测试了大数据统计分析的日志清洗。因此本节只从整体上对整个系统的整体功能流程进行描述，大致如下：

首先，当然是运营人员利用软件开发人员开发出来的后台系统进行优惠券和用户标签的配置。通过后台的登录页面，输入 username 和 password 后，点击登录就会进入优惠券配置主页，如果上次使用后台登录过账号且没有注销，则直接进入主页，在优惠券主页中，可以对优惠券进行增删改查操作，当然也有一些经常用到的接口测试，例如兑换优惠券的接口，因为公司在每次发券之前，为了保证用户一定能领券，运营人员都会自己注册一个账号来进行测试，同理，用户标签也是这样。

其次，Android 客户端的用户登录后，就可以在优惠券兑换界面输入优惠券码，通过后端优惠券兑换接口的逻辑处理来领取优惠券，并且用户在优惠券界面可以看到自己历史领取的所有优惠券信息。

最后，利用 Android 客户端和后端交互后端接口打印的日志文件数据，通过 Hadoop 的大数据工具对日志文件进行清洗提取，就能看到每天甚至是最近一段时间的优惠券兑换数据。这样可以根据这个兑换数据又指导未来时间的配置工作，一方面公司提高了运营效果，另一方面消费者得到了自己感兴趣的优惠，对于公司和消费者用户来说，都是一件美好的事情。

6.4 本章小结

本章通过功能测试和服务器性能测试对整个系统进行了测试。功能测试所测的所有测试点保证了系统业务流程的正确性。在对服务器性能测试中，主要对并发性、事务响应时间和吞吐量进行测试，其并发能力、事务响应时间和吞吐量都基本满足需求。另外在测试完了以后，我在整体上对整个系统的工作流程进行了介绍。

第七章 总结与展望

7.1 总结

在一次座谈会中，习总书记强调：“在践行新发展理念上先行一步，让互联网更好造福国家和人民”。因此，在互联网服务中，电商服务极大的方便了人民的生活、节约了人民的时间，应该是重中之重。基于 java 的电商软件开发与大数据研究就是以电商公司的优惠券和用户标签为重点进行功能设计和研究。为了满足我所研究的需求，本文做了如下工作：

(1) 对现阶段的电商服务现状及发展趋势进行调查，发现虽然我国的电商事业走在了许多发展中国家的前列，但也存在许多问题，例如假货盛行、海淘电商和品质电商做得不好等。

(2) 为公司运营人员开发了一套优惠券和用户标签后台管理系统，该平台以 Web 网页的形式展现给公司运营人员，运营人员能增删改查优惠券和用户标签，并在真正发送优惠券之前可以在后台先进行测试等。

(3) 根据需求开发了一个安卓客户端 APP，用以提供给电商安卓用户使用，结合前面运营人员配置的优惠券，开发了后端领券的接口，用户注册登录后通过领券界面自己可以领取运营人员配置的优惠券。

(4) 通过邀请一些校内同学参与使用 APP，产生的日志文件，利用公司的分布式 Hadoop 环境和我开发的大数据的次数统计任务，跑出来了所有用户自己兑换的优惠券的兑换结果。这样一来，根据最终的兑换结果又可以更好的指导运营人员进行下一阶段的配置工作。

(5) 开发完后进行测试，分为功能测试、服务端性能测试。对各部分功能测试之后，在服务器的性能测试中，使用 LoadRunner 测试工具进行服务器的并发性和事务响应时间等的测试，测试结果基本达到预期。

(6) 后端系统具有很好的可扩展性，比如后期需要实现与支付宝、微信等的对接的话，在系统的后端可以随时加接口。

7.2 展望

基于 java 的电商软件开发与大数据研究可以实现预期的功能，但是相对于电商软件功能来说，这只是很小的一小块，对于功能的全面性、系统性能等方面还有很多可以提高的地方：

(1) 对安卓客户端来说，我是以电商优惠券为例，但是在现实生活中，还需要很多其他的功能才能真正实现用户购物的需求，例如支付等。

(2) 整个系统主要优化了电商优惠券的配置，其实对于其他类似的配置，也能进行相应的优化，例如，后台中所说的用户标签配置。

(3) 对于大数据的日志清洗工作而言，不但统计的数据类型可以更加多样，而且其本身的统计任务也可以多样化，例如对一天的登录用户进行去重统计等。另外，统计出来的数据可以和 Oracle BI 进行结合，把公司关心的 kpi 数据展示在 BI 页面上，让公司有这个 BI 权限

的员工都可以看到这些核心数据。

(4) 本设计使用的服务器在性能上难以满足大电商，因为经过服务器的性能测试最大并发数只有 700，吞吐量也不够高，应用到实际生活中可以考虑换个服务器。

(5) 随着公司的发展，如果公司有大量的电商交互数据，还可以搭建公司数据仓库等。

(6) 对于像阿里、网易大型电商公司而言，不但可以利用 java 语言来做大数据还可以做云计算，个性推荐等。

致　　谢

忽然一天发现，经常走的那条道枫树红了，枫叶也在往下掉了。让我想到落叶归根，更让我不禁想到研究生的两年半生活就要结束了，一切都好像才刚刚开始，所有的事情回首起来都还历历在目，是的，时光飞逝，不知不觉，已经临近毕业。在此论文完成之际，我不得不一一感谢一些人对我的帮助和关心。

首先，特别要感谢乐观开朗、学术上严谨求真的研究生导师李金新老师。他教会我很多专业知识，许多做人的道理，这份师恩，我永记于心。李老师他是一个以身作则的著名学者，对工作认真负责、对学术尊敬严谨、对生活乐观热爱，所有的方方面面让我受益不菲。在平常学习相处过程中，他的处世态度和专业知识，让我不得不佩服。正是这样，研究生期间，我不但学到了很多的专业知识，也对为人处世这四个字有了新的认识。在研究生期间，在李老师的指导下，我总能比同届的孩子走的更快，学的更多，同时也完成了不少事情。当然，这篇毕业论文也是在他的耐心指导下完成的。在这里，我再次向李老师表示我真诚的谢意！

然后，我要感谢我的实验室同门和科技馆的同学们，大家共同营造了良好的学习气氛，平时生活与他们待在一起，既学到了宝贵的知识，也为我的研究生生涯增添了不少笑声，这让我的研究生生活充满色彩，总之，和他们在一起的这段时光，必定是我今后人生中最宝贵的财富。

特别地，我得感谢我的父母以及爱我的人，感谢他们默默的支持和最无私的精神帮助。

最后，我感谢的是我的母校，是它提供给了我这个优秀的学习平台，在此，祝愿母校在现在辉煌的成绩下能够越来越好、再造辉煌！

参 考 文 献

[1] 李国征. 旅游目的地智能手机 APP 旅游信息服务研究[D]. 东北财经大学, 2013.

[2] 丁天豪. “一带一路”战略下我国跨境电商发展研究[J]. 经营与管理,2017,(07):101-103.

[3] 李莉. 中国跨境电子商务发展现状及对策[J]. 现代营销(下旬刊), 2017, (09):17.

[4] . 大数据概念与发展[J]. 中国科技术语,2017,19(04):43-50.

[5] 张宇敬, 许美玲, 臧丽娜, 刘敏. 京津冀协同发展背景下的大数据人才培养研究[J]. 经营管理者, 2016, (12):115-116.

[6] 高海建. 基于大数据视角的电子商务产业研究[D]. 首都经济贸易大学, 2015.

[7] 莫岱青. “双 11”电商全网首度突破 1000 亿元大关[J]. 计算机与网络,2015,41(22):10-12.

[8] 冯霄霞. 亚马逊大数据成打开用户心门的钥匙[N]. 中国信息化周报, 2016-09-12(017).

[9] 谢朋峻. 基于 MapReduce 的频繁项集挖掘算法的并行化研究[D]. 南京大学, 2012.

[10] 徐春. 基于倒排索引的增量更新关联挖掘算法的研究[D]. 广西师范学院, 2016.

[11] 杨旭. 小型企业办公自动化系统设计与实现[D]. 大连海事大学, 2013.

[12] Hassan Artail, Ammar El Halabi, Ali Hachem, Louay Al-Akhrass. A framework for identifying the linkability between Web servers for enhanced internet computing and E-commerce[J]. Journal of Internet Services and Applications, 2017, 8(1):.

[13] 刘珍. Web 服务器故障的分析与处理[J]. 电脑编程技巧与维护, 2017, (21):87-89.

[14] 刘冶一. 互联网金融环境下农商行直销银行系统的设计和实现[D]. 浙江工业大学, 2016.

[15] 韩旭. 大学足球联赛信息共享与分析平台的设计与实现[D]. 北京工业大学, 2014.

[16] 刘霜霜. Java EE 应用中权限系统的研究与实现[D].湖南大学,2011.

[17] 张佳, 曹悦, 刘冲, 范雷, 吕利娜. 基于 Spring+MyBatis 的高校工资信息管理系统的设计与实现[J]. 科技展望, 2017, 27(15):11.

[18] 杨清茹. 山东省学校建设规划地理信息系统的设计与实现[D]. 山东大学, 2015.

[19] 江志刚. 基于 SSM 框架的网上题目录入答题系统设计[J]. 无线互联科技,2017,(20):62-63.

[20] 叶小艳, 张芒, 顾奕腾. 基于 SSH 框架的 OA 系统设计与实现[J]. 计算机时代,2017,(10):47-50.

[21] 岳巍. 高校图书馆与互联网知识服务的比较及发展策略[J]. 河南图书馆学刊,2016,36(12):57-59.

[22] 吴龙, 吴健, 任红民. 基于双数组 Trie 树的嵌入式 TTS 系统研究[J]. 现代机械,2010,(04):67-70+93.

[23] 何锡标, 陈淑荣. 一种基于无线定位技术的 LBS 应用[J]. 微型机与应用,2014,33(09):7-10.

[24] 朱晶. TCP 协议简述与三次握手原理解析[J]. 电脑知识与技术, 2009, 5(05):1079-1080.

[25] 蒋啸天. 深空网络文件传输协议研究[D]. 西安电子科技大学, 2010.

[26] 谢芳清. 在实验中利用 ethereal 网络协议分析工具体现 TCP 协议三次握手的实现[J]. 福建电脑,2010,26(02):181+159.

[27] 陈远鹏,李永忠. Linux 平台下 Rootkit 木马分析与检测[J]. 电子设计工程,2017,25(01):39-42.

[28] 刘磊. Linux 操作系统学习方向与方法探讨[J]. 电子世界,2012,(24):121-122.

[29] 肖沅峰. 面向计算机信息技术学习交流社区的设计与开发[D].天津大学,2013.

[30] 陈晢. 一款铁路客货运服务调查评价系统设计与实现[J]. 电子技术与软件工程,2017,(04):207-209.

[31] 张小菲.Android 平台上音视频系统的研究及播放器开发[D].导师：王书振;尹德云.西安电子科技大学,2012.

[32] 冯克环.基于 Dalvik 虚拟机自适应编译系统的分析与优化[D].东南大学,2012.

[33] 马哲.基于虚拟机架构的软件保护方法研究与实现[D].西安理工大学,2012.

[34] 蒋亦奇. 基于安卓平台的事件日历系统的设计与实现[D].南京大学,2013.

[35] 白东强. 基于 Android 平台的个人出行助手的设计与实现[D].江西理工大学,2013.

[36] 虞倩倩,戴月明. 基于 MapReduce 的并行模糊 C 均值算法[J]. 计算机工程与应用,2013,49(14):133-137+151.

[37] 郭钊.基于 Hadoop 的数据挖掘在电商环境的研究与应用[D].湖南大学,2016.

[38] 赵建红. Hadoop 模型研究及其作业调度算法的改进[D].首都经济贸易大学,2013.

[39] 李坤.面向舆情分析的分布式数据采集及处理技术研究[D].北京信息科技大学,2013.

[40] 亢丽芸.基于 Heritrix 与 Hadoop 的海量网络学术文献获取及并行处理研究[D].山东理工大学,2012.

[41] 杨柳. 基于 Hadoop 平台的频繁项数据挖掘算法研究[D]. 大连理工大学, 2014.

[42] 柴学智. 面向云计算的工作流系统设计与实现[D].上海交通大学,2011.

[43] 黄斌杰. 基于云平台的移动视频播放系统的设计与实现[D].广东工业大学,2015.

[44] 袁子淇. 基于 ZooKeeper 的集群应用配置管理的设计与实现[D].内蒙古大学,2015.

[45] 唐海东,武延军. 分布式同步系统 Zookeeper 的优化[J]. 计算机工程,2014,40(04):53-56.

[46] 秦丹. 软件测试与质量控制[J]. 信息与电脑(理论版),2012,(05):130-132.

[47] 沈明辉,冯昌琪,冯丽,毛云鹏,邓韧,王帅,吴结凤,姚歆,于志华. 第三方测试在四川省基层医疗卫生机构管理信息系统建设中的应用[J]. 中华医学图书情报杂志,2014,23(03):15-18+22.

[48] 王蕾. 基于 LoadRunner 的负载压力测试[J]. 现代计算机,2013,(18):52-54+73.

[49] 曲芳晓. 基于主数据管理的数据中心系统设计与实现[D].天津大学,2016.

[50] 刘爱云. 东胜集团成本预算管理信息系统的设计与实现[D].电子科技大学,2010.

[51] 姚昕. 基于 Loadrunner 技术的 Web 性能测试的研究与实现[D].哈尔滨商业大学,2013.

[52] Jyoti V. Gautam,Harshadkumar B. Prajapati,Vipul K. Dabhi,Sanjay Chaudhary. Empirical

Study of Job Scheduling Algorithms in Hadoop MapReduce[J]. Cybernetics and Information Technologies,2017,17(1).

[53] Feng Ping Chen,Li Miao,Yue Gao Tang. Research of Hadoop Parameters Tuning Based on Function Monitoring[J]. Applied Mechanics and Materials,2014,3368(621).

[54] B.K. Tripathy,Dishant Mittal. Hadoop based uncertain possibilistic kernelized c-means algorithms for image segmentation and a comparative analysis[J]. Applied Soft Computing,2016,46.

[55] Ivanilton Polato,Reginaldo Ré,Alfredo Goldman,Fabio Kon. A comprehensive view of Hadoop research—A systematic literature review[J]. Journal of Network and Computer Applications,2014,46.

[56] Mukhtaj Khan,Zhengwen Huang,Maozhen Li,Gareth A. Taylor,Mushtaq Khan. Optimizing hadoop parameter settings with gene expression programming guided PSO[J]. Concurrency and Computation: Practice and Experience,2017,29(3).

[57] Yu Su,Wan Lin Gao,Li Na Yu,Hui Hu,Xuan Luo. Wireless Body Area Network Data Storage Method Based on HBase[J]. Applied Mechanics and Materials,2013,2748(427).

[58] Jun Wu Xu,Jun Ling Liang. Research on a Distributed Storage Application with HBase[J]. Advanced Materials Research,2013,2200(631).

[59] An Ping Xiong,Lei Wang. Building Schema of Assorted Index Based on Huffman Encoding in HBase[J]. Applied Mechanics and Materials,2014,3207(556).

[60] Lei Rao,Fan De Yang,Xin Ming Li,Dong Liu. A Storage Model of Equipment Data Based on HBase[J]. Applied Mechanics and Materials,2015,3744(713).

附录

作者在读研期间申请的软件著作权

[1] 欧盛彪，master，小汽车理论考试系统 V1.0，China patent,2017SR225369.