

浙江工业大学

本科毕业设计说明书（论文）

（2017 届）



论文题目 基于 quartz 框架作业监控调度
系统的设计与实现

作者姓名 孙瑞丰

指导教师 韩姗姗

学科(专业) 软件工程 1302

所在学院 计算机科学与技术学院

提交日期 2017 年 6 月

摘要

随着计算机行业技术的高速发展，人们的日常生活开始越来越依赖于计算机技术，而这其中就包括作业调度。在未出现作业调度技术之前，人们若想在某一时刻完成特定的作业，只能人为执行或是设置定时器，而人为执行会消耗人力资源，通过设置定时器则又非常复杂不友好。而作业调度技术出现之后，它可以根据开发者指定的时间来执行一些操作，完成开发者想要其在那个时刻完成的作业。

本系统使用 Quartz、Spring 等技术框架设计并实现了作业调度监控系统。本系统可以根据开发者指定的时间来自动执行作业，并且当作业执行异常报错时会发邮件进行告警。首先使用 Quartz、Spring 与 Java 的反射完成基础的作业调度；接着通过异常捕获、Spring 封装的 Java Mail 来实现作业执行时异常告警邮件的发送；然后通过 Redis、Echarts 与 Spring MVC 完成系统的前台展示功能与作业执行结果的图表可视化界面。

本文首先介绍了作业调度的背景、现状、研究意义，接着列举了与本系统相关的技术和方法。然后详细说明了系统整体的框架设计、调度模块功能设计、监控以及可视化功能的模块设计。接着具体介绍了系统各个功能模块的实现，主要包括功能实现、系统中所使用到的数据结构的实现、可视化界面的实现。最后总结了毕设中遇到的问题和不足，并提出下一步工作。

关键词：Quartz，作业调度，异常捕获，图表可视化

Abstract

With the rapid development of computer technology industry, people's daily life began to rely more and more on computer technology, which included job scheduling. In the absence of job scheduling technology, if people want to complete a specific job at a specific time, only human implementation or set a timer can do that, and both this two way are complex and unfriendly. And after job scheduling technology appeared, it can be based on the developer specified time to perform some operations to complete the developer wants to complete the work at that moment.

The system uses Quartz, Spring and other technical framework design and implement of the job scheduling monitoring system. The system can automatically run the job according to the time specified by the developer, and when the job execution error will send an e-mail alarm to a related person. First use Quartz, Sprin and Java reflection to complete the basic job scheduling; then through the exception capture and Spring packaged Java Mail to achieve the implementation of the abnormal operation of the alarm message sent; and then through Redis, Echarts and Spring MVC to complete the system front desk display and graphical visualization interface of the results of job execution.

This paper first introduces the background, current situation and significance of job scheduling, and then lists some technologies and methods related to this system. Then it describes the module design of the whole system design, the function of the dispatching module, the monitoring and the visualization function. Then the implementation of the various functional modules of the system is introduced, including the realization of the function, the realization of the data structure used in the system and the realization of the visual interface. And finally summed up the problems encountered in the set and inadequate, and put forward the next step.

Keywords: quartz, job scheduling, exception capture, graphical visualization

目录

摘要	I
Abstract	II
第一章 绪论	1
1.1 研究背景和意义	1
1.2 国内外研究现状	1
1.3 本文的主要工作	2
1.4 本文的组织结构	3
第二章 技术与方法	4
2.1 Java	4
2.1.1 Java 反射	4
2.1.2 Java 捕获异常	5
2.2 Quartz	6
2.3 Spring	7
2.4 Redis	9
2.5 iBatis	9
2.6 RMI	10
2.7 Echarts	11
2.8 Java Script	11
2.9 jQuery	11
2.10 开发环境与编程语言	11
2.11 本章小结	12
第三章 作业调度监控系统功能设计	13
3.1 系统简介	13
3.1.1 系统需求概述	13
3.1.2 系统功能概述	13
3.2 系统的整体框架设计	14
3.2.1 作业发布系统架构设计	14
3.2.2 监控调度系统架构设计	15
3.3 作业调度功能设计	16

3.3.1 调度器启动	16
3.3.2 作业调度	16
3.3.3 通用作业	16
3.4 远程作业执行设计（作业发布系统）	17
3.5 作业监控功能设计	17
3.6 作业执行记录数据存储功能设计	17
3.7 异常告警功能设计	18
3.8 前台展示功能设计	18
3.9 系统流程	18
3.10 本章小结	19
第四章 作业调度监控系统功能实现	20
4.1 系统开发规范	20
4.1.1 作业发布系统工程结构	20
4.1.2 监控调度系统工程结构	21
4.1.3 代码命名规范	23
4.2 系统数据结构实现	23
4.2.1 BasicQuartzJob 数据结构	23
4.2.2 JobExcuteResult 数据结构	25
4.2.3 UserInfo 数据结构	25
4.2.4 Redis 内作业执行结果数据结构	26
4.3 作业调度功能实现	27
4.3.1 调度器启动	27
4.3.2 作业调度	27
4.3.3 通用作业	28
4.4 远程作业执行功能实现	30
4.5 作业监控功能实现	30
4.6 作业执行记录数据存储功能实现	30
4.7 异常告警功能实现	31
4.8 前台展示功能实现	32
4.9 界面实现	33
4.9.1 作业列表页	33

4.9.2 作业修改页	34
4.9.3 人员列表页	34
4.9.4 人员修改页	35
4.9.5 作业执行结果展示页	35
4.10 本章小结	36
第五章 总结.....	38
5.1 完成的工作	38
5.2 存在的问题及下一步工作.....	38
参考文献.....	39
致 谢	41
附 录	42

图目录

图 2-1Spring 模块图.....	7
图 2-2SpringMVC 设计模式图.....	8
图 2-3RMI 远程代理与对象.....	10
图 3-1 作业发布系统架构设计图	14
图 3-2 监控调度系统架构设计图	15
图 3-3 系统功能流程图	19
图 4-1BasicQuartzJob 对象.....	24
图 4-2JobExcuteResult 对象	25
图 4-3UserInfo 对象	26
图 4-4 成功执行作业 id 数据结构	26
图 4-5 作业成功执行数据结构	27
图 4-6 查出远程作业进行启动代码	28
图 4-7 启动某个作业&设置属性代码	28
图 4-8 通用作业获取执行指定作业信息代码	29
图 4-9 通用作业反射执行本地（监控调度系统内部）作业代码.....	29
图 4-10 作业发布系统配置代码	30
图 4-11 作业成功执行数据入库代码	31
图 4-12GetExceptionDetailUtil 工具类	32
图 4-13 作业成功执行结果 json 拼接代码.....	33
图 4-14 作业列表页	33
图 4-15 作业修改页（1）	34
图 4-16 人员列表页	34
图 4-17 人员修改页	35
图 4-18 作业执行历史记录页	35

第一章 绪论

1.1 研究背景和意义

随着计算机行业的高速发展，计算机应用已经被应用到各行各业。由于计算机应用的稳定性、易用性，人们开始愈发依靠计算机应用提高工作效率用以完成工作。伴随着这种愈发性，计算机应用技术的迭代周期因此变得越来越短，与此同时技术实用性越来越强。作业调度^[1]就是这么一种被人们所需要的应用技术。

作业调度是 J2EE web 项目中常见的应用。它可以按照开发者指定的时间间隔来执行操作，并完成一些无需用户输入的任务^[2]。它可以解放人力，提高工作效率，可以说是一种必需技术。作业调度主要有两大功能：减轻系统负荷、作业项自动定时执行。

减轻系统负荷：当前许多网站的访问量正以成倍的数量级增加，这使得许多网站的网络压力、服务器负载变得相当严重。而为了缓解这一现状，一些实时性不是那么强的作业可以在访问量低谷的时间段执行，如许多社交网站通知邮件的发送，通常在凌晨 2:00 左右发送。

作业自动定时执行：在日常生活中，经常出现有些作业需要在某个特定的时间执行的情况，如果人为执行这些作业，那么这会消耗不必要的人力资源。而如果使用作业调度，那么作业将严格按照设置的时间自动执行，避免人工干预的可能性。比如某些论坛对用户积分的排名、每个半小时对违禁用户的解封操作等。

因此为了更便捷的使用此项技术，本研究最终将开发一个作业调度的系统，可以发布作业以及修改作业，与此同时本系统将对作业的调度情况进行监控。

1.2 国内外研究现状

JDK 中的 Timer, ScheduledExcetuor 是 Java 中实现定时调度作业的基础方法，但有益于其功能较为简单，较难实现复杂的任务调度运行规则^[3]，所以并不适用于实际开发。

Quartz 是 OpenSymphony 开源组织在任务调度领域的一个开源项目，完全基于 Java 实现。作为一个优秀的开源调度框架，Quartz 具有功能强大，应用灵活，易于集成的特点^[4]。Spring 与 Quartz 可以简便的整合，使用最为广泛。Quartz 核心的组件为：trigger（触发器，定义作业执行的规则）、job（作业，即希望执行的业务）、scheduler（作业调度器，trigger 与 job 都需在 SchedulerFactory 注册后才能进行使用）。

Cron4J 同样也是一个适用于 Java 平台的开源作业调度框架，它与 UNIX 的 cron 守护进程非常相似^[5]。它可以根据一些简单的规则，在正确的时间启动想要执行的作业。相比与 Quartz 它的功能非常单一。

在国内，作业调度相关技术出现的较晚。在已问世的技术中最为出名的要属阿里的中间件 SchedulerX。SchedulerX 的前身是 TTM 定时任务触发服务，随着淘宝业务不断发展壮大 TTM 简单的触发功能越来越不满足业务不断增长的需求，在很多场景下 TTM 无法为用户解决问题。于是 SchedulerX 出现了，它提供了自主运维管理后台，让用户能够通过页面来配置、修改和管理定时任务。SchedulerX 还能管理任务执行的生命周期，从每次任务执行开始一直到任务执行结束都有记录，用户能看到每次任务执行的开始和结束时间以及能看到执行成功或者失败，SchedulerX 还会为用户保留过去的执行记录，用户可以查看定时任务历史执行记录。^[6]

如今，人们对作业调度方面的服务需求日益增加，但绝大多数都是基本的服务或者框架，并非是一个完整的系统来提供给用户实用。

1.3 本文的主要工作

本文的主要工作是利用 Quartz、Spring 及 RMI 等相关技术框架搭建一个作业调度监控系统，它能够实现作业在指定时间调度运行、作业异常报错时进行告警、图表可视化作业的执行情况等功能。

1.4 本文的组织结构

本文共分为五章，以“基于 Quartz 框架作业调度监控系统”为背景，研究讨论了 Quartz、Spring、Spring MVC、iBatis 的应用架构，以及所涉及的各种开源框架技术，详细阐述了如何设计与实现作业调度监控系统，各章安排如下：

第一章，简单介绍本次研究的背景和意义，以及国内外相关领域的研究现状，接着介绍了本此研究的主要内容。

第二章，详细介绍了本系统开发过程中所涉及到的技术方法及相关框架，为系统相关功能模块实现做准备。

第三章，重点介绍了本系统相关功能模块的设计，包括系统整体架构设计、调度模块设计、异常监控告警设计、作业执行结果可视化设计。

第四章，具体介绍了本系统功能模块的实现。主要内容包括系统架构、系统中所使用到的数据结构以及系统各个功能模块的实现。

第五章，对全文总结，并指出系统的缺点以及不足，同时展望未来。

第二章 技术与方法

本系统实际可分为两个系统，一个是监控调度系统，一个是作业发布系统。他们使用 Java 语言基于 Quartz 与 Spring 等框架技术进行开发。使用 Quartz 与 Spring 完成作业调度功能，使用 Tomcat 作为服务器，使用 Redis、MySQL、iBatis 作为数据查询与存储，使用 Spring MVC、Echarts、Java Script 作为系统前端展示。

2.1 Java

Java 是一门面向对象的编程语言，不仅吸收了 C++ 等语言的各种优点，同时还去除了 C++ 里难以理解的指针、多继承等复杂概念，因此 Java 语言具有简单易用和功能强大两个特征。Java 语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。^[7]

2.1.1 Java 反射

反射是 Java 程序开发语言的特性之一，它允许运行中的 Java 程序获取自身的信息，并且可以操作类或对象内部的属性。^[8]

程序中一般的对象类型都是在编译期就确定下来的，但是有些对象的类型在编译期是未知的，而 Java 反射机制可以动态的创建对象并调用其方法、访问属性。所以可以通过反射机制直接创建对象，即使这个对象的类型在编译期是未知的。

反射的核心是 JVM 在运行时才动态加载类或调用方法/访问属性，它不需要事先（写代码的时候或编译代码时）知道运行对象是谁。

Java 反射主要提供以下功能：

- 1) 在运行时判断任意一个对象所属的类；
- 2) 在运行时构造任意一个类的对象；
- 3) 在运行时判断任意一个类所具有的成员变量和方法（通过反射甚至可以

调用 `private` 方法）；

4) 在运行时调用任意一个对象的方法。

2.1.2 Java 捕获异常

1) 异常概念

异常情形（`exceptional condition`）是指组织当前方法或作用于继续执行的问题^[9]。异常与普通问题是有区别的，普通问题是指运行时可以获得信息来解决这个错误。而异常则不能继续执行下去，原因是在运行时无法获得信息来处理这一问题。因此出现异常时，所能做的就是从当前环境跳出，将异常抛给上一级调用环境。

异常的出现使得开发人员在做每一件事时都会将其当作一个事物来进行考虑，异常可以看作是保证这些事物完成的基本底线。因为异常的重要作用就是如果发生问题，它将不允许程序沿着正常路径继续执行下去，它将停止程序运行，告诉开发者出现了什么问题，或者强制处理问题，使程序回到稳定状态。

2) 栈轨迹

栈轨迹 `getStackTrace()`方法将返回一个代表该线程的堆栈转储堆栈跟踪元素的数组。

如果返回的数组是非零长度，那么该数组的第一个元素堆栈，则是该堆栈中最新的方法调用的顶部。而数组的最后一个元素代表堆栈，则为该堆栈中最近的方法调用的底部。

3) 异常链

异常链是指在程序捕获到异常后抛出另一个异常，将原始异常的信息保存下来。在 `JDK1.4` 以前，开发人员如果想获得原始的异常欣喜，必须通过自己编写代码来实现。而现在 `Throwable` 子类的构造方法中可以接受 `cause` 对象作为入参。`Cause` 表示为原始异常，这样的话就可以把原始异常传递给新的异常，这样的话就可以通过异常链追踪到原始异常。

2.2 Quartz

当前作业调度框架有很多，如 Quartz^[10]、cron4j、JCronTab^[11]等。Spring 为 Quartz 的组件提供了类似于 Bean 风格的扩展类，使得 Spring 与 Quartz 能够更好地整合在一起。Spring 为创建 Quartz 的 Scheduler、Trigger 和 JobDetail 提供了 FactoryBean 类。有利于在 Spring 环境下创建对应的组件对象，并且和 Spring 容器生命周期进行启动和停止的动作。而 Cron4J 和 JCronTab 在某种程度上来说像是 Quartz 的一个子集，它们拥有的功能 Quartz 也几乎全部拥有。

Quartz 是 OpenSymphony 开源组织的一个开源项目，它能够创建调度来执行一个作业，可以与 Java 项目进行集成。

Quartz 框架的核心是调度器。调度器负责管理 Quartz 应用运行时环境。调度器不是靠自己做所有的工作，而是依赖框架内一些非常重要的部件。为确保可伸缩性，Quartz 采用了基于多线程的结构。启动时，框架初始化一套 worker 线程，这套线程被调度器用来执行预定的作业。Quartz 依赖一套松耦合的线程池管理部件来管理线程环境。Quartz 对任务调度的领域问题进行了高度的抽象，提出了调度器（Scheduler）、任务（Job）和触发器（Trigger）这三个核心的概念。^[12]

Job 接口只有一个简单的方法 `public void execute(JobExecutionContext context)`。当 Job 的触发器被触发时，`execute(...)`方法即由调度其的工作线程之一调用。JobExecutionContext 入参向作业实例提供有关其运行时的环境信息：执行它的调度程序的句柄，触发器执行的句柄，作业的 JobDetail 对象以及其他几个东西。

JobDetail 对象由 Quartz 客户端（Web 程序）在添加作业到调度器时所创建。它包含作业的各种属性设置信息，以及 JobDataMap（JobDataMap 可以用于保存您希望在执行时对作业实例可用的任何数量的可序列化的数据对象）。它本质上是作业实例的定义。触发器用于触发作业执行。当希望调度某一个作业时，可以实例化触发器并“调整”其属性用以转换成期待的调度器。触发器也可以有一个与它们相关联的 JobDataMap（这对于将触发器所携带的参数传递给 Job 非常有用）。Quartz 提供了不同的触发类型，但最常用的类型是 SimpleTrigger 和 CronTrigger。^[13]

2.3 Spring

Spring 是为了解决企业应用程序开发复杂性而创建的一个开源框架，它由七个模块组成^[14]。Spring 模块构建在核心容器上，如下图所示

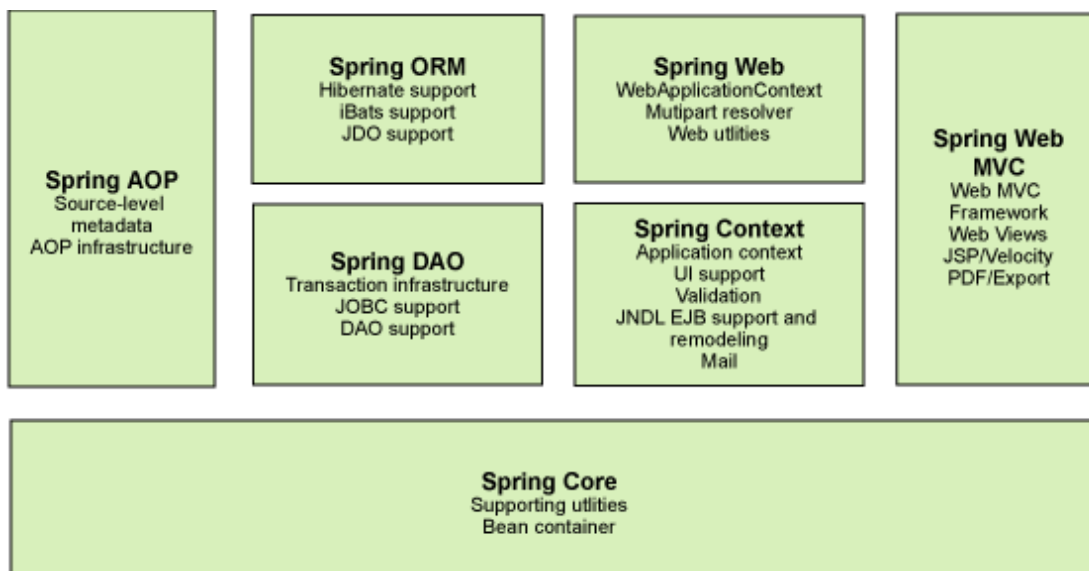


图 2-1Spring 模块图

其中组成 Spring 框架的每个模块都可以单独存在，或者与其他一个或多个模块联合使用。

- 1) **Spring Core:** 提供 Spring 框架的基础功能。它的主要组件是一个通过工厂模式实现的 **BeanFactory**。**BeanFactory** 使用控制反转 (IOC) 解决了程序的依赖问题，将应用的依赖与实际应用程序代码分开。
- 2) **Spring Context:** Spring 上下文是一个配置文件，向 Spring 框架提供上下文信息。Spring 上下文包括企业服务，例如 JNDI、EJB、电子邮件、国际化、校验和调度功能。
- 3) **Spring AOP:** 通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了 Spring 框架中。所以，可以很容易地使 Spring 框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。
- 4) **Spring DAO:** JDBC DAO 抽象层提供了有意义的异常层次结构，可用该

结构来管理异常处理和不同数据库供应商抛出的错误消息。异常层次结构简化了错误处理，并且极大地降低了需要编写的异常代码数量（例如打开和关闭连接）。Spring DAO 的面向 JDBC 的异常遵从通用的 DAO 异常层次结构。

5) Spring ORM: Spring 框架插入了若干个 ORM 框架，从而提供了 ORM 的对象关系工具，其中包括 JDO、Hibernate 和 iBatis SQL Map。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

6) Spring Web: Web 上下文模块建立在应用程序上下文模块之上，为基于 Web 的应用程序提供了上下文。所以，Spring 框架支持与 Jakarta Struts 的集成。Web 模块还简化了处理多部分请求以及将请求参数绑定到域对象的工作。

7) Spring MVC 框架: MVC 模式是一种现在非常普及的软件架构模式，分为 Model、View、Controller 三个部分。Spring 提供了一种高度可佩的 MVC 框架，即 SpringMVC，它很好的实现了 MVC 的核心概念，当向 MVC 添加反转控制时，使得应用程序高度解耦，提供简单的配置更改就可以动态的更改组件的灵活性^[15]。下图是 SpringMVC 的设计模式图:

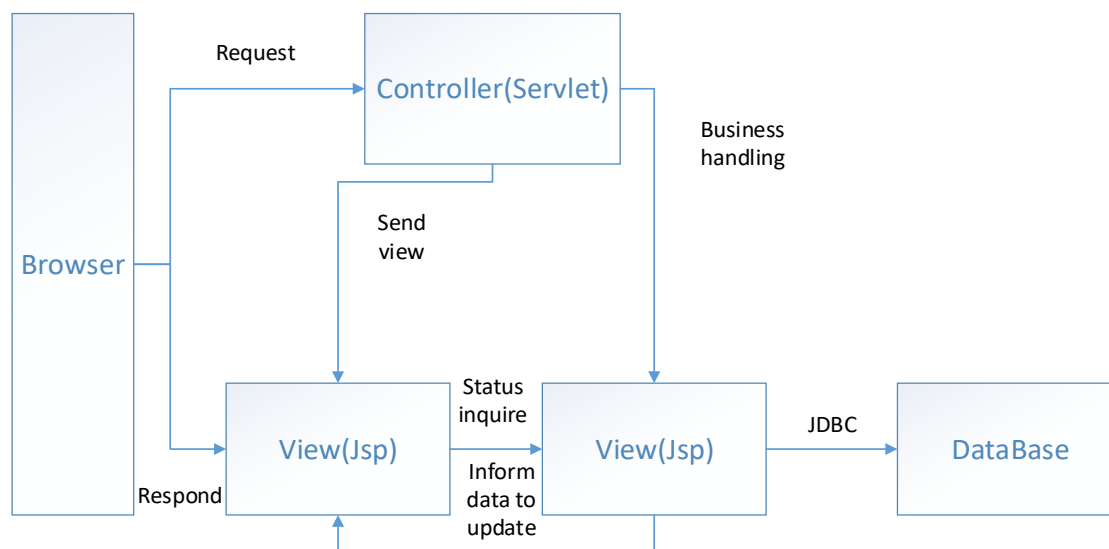


图 2-2SpringMVC 设计模式图

2.4 Redis

Redis^[16]是一个 key-value 存储系统。和 Memcached 类似，它支持存储的 value 类型相对更多，包括 string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和 hash（哈希类型）。这些数据类型都支持 push/pop、add/remove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，Redis 支持各种不同方式的排序。与 memcached^[17]一样，为了保证效率，数据都是缓存在内存中。区别的是 redis 会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件。

Redis 支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。这使得 Redis 可执行单层树复制。存盘可以有意无意的对数据进行写操作。由于完全实现了发布/订阅机制，使得从数据库在任何地方同步树时，可订阅一个频道并接收主服务器完整的消息发布记录。同步对读取操作的可扩展性和数据冗余很有帮助。^[18]

2.5 iBatis

iBatis^[19]是一个 ORM^[20]框架，它可以自动化的将 SQL 数据与对象相关联。在 XML 配置文件中编写 SQL 语句，可以将映射与应用程序逻辑分离。这样显著减少了开发人员使用较低级别的 API（如 JDBC 和 ODBC）访问数据库所需的代码量。

其他持久性框架（如 Hibernate）允许用户创建对象模型，并能自动创建和维护数据库。而 iBatis 采用相反的方法：开发人员通过编写 SQL 语句，iBatis 自动创建 Java 对象。这两种方法都不错，但是当开发人员不能完全控制 SQL 数据库时（例如，应用程序可能需要访问由其他软件使用的现有 SQL 数据库，或访问协议不完全在应用程序开发人员控制下的新数据库，如当专门的数据库设计团队创建协议并仔细优化用于提高性能的数据库），那么 iBatis 是一个不错的选择^[21]。

2.6 RMI

RMI（**Remote Method Invocation**，远程方法调用）是 **Java** 的一组用户开发分布式应用程序的 **API**。在服务端应用如果定义了相关接口并且有相关实现类，那么在客户端只需调用服务端的远程接口即可由代理对象调用执行。

使用 **RMI**，客户端不用与服务端打交道，只需要让客户端与服务端的远程代理对象打交道，因为代理对象同样实现了此接口。当客户端请求此远程接口时，如果代理对象能确认远程对象可以调用，则会委派给远程对象，由它来尽享调用。在这个过程当中，**RMI** 会在客户端生成一个 **Stub**（存根），并且会把 **Stub** 视作远程对象的代理。远程对象的代理需要存在于客户端，所以需要把 **Stub** 复制或下载到本地^[22]。所以客户端实际上是在于远程对象的代理打交道，并非是与远程对象直接打交道，如下图所示：

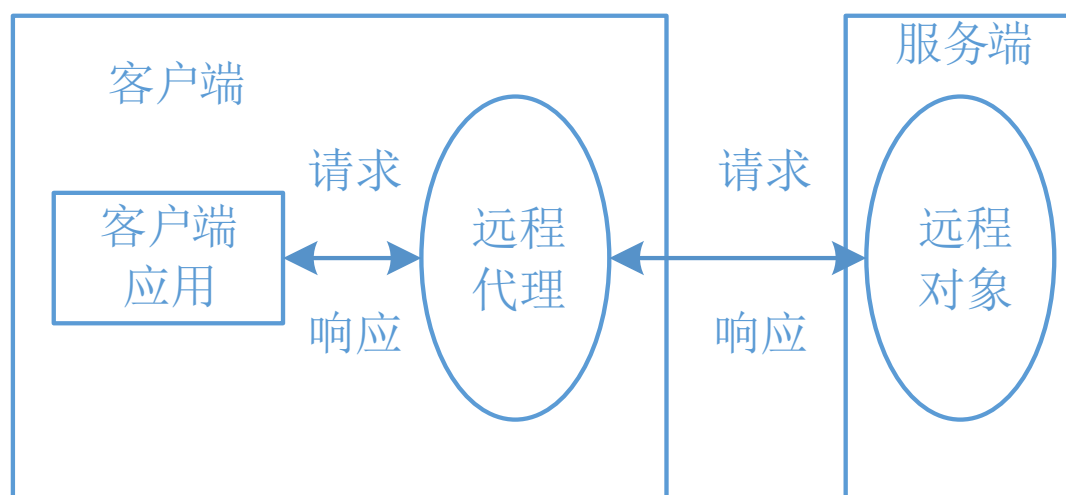


图 2-3RMI 远程代理与对象

Spring 在远程调用服务方面，基于 RMI 的基础，抽象的提供了 RmiProxyFactoryBean 和 RmiServiceExporter。RmiServiceExporter 提供注册远程接口，而 RmiProxyFactoryBean 则会根据指定 URI 去找到相对应的远程接口并获取它的代理对象。使用 Spring 只需配置几个简单的属性，无需继承 Remote 接口抛出 RemoteException 异常，省去了使用原始 RMI 配置的步骤，显得非常便捷。

2.7 Echarts

ECharts^[23]是一个纯 Javascript 的图表库，可以流畅的运行在 PC 和移动设备上，兼容当前绝大部分浏览器（IE8/9/10/11，Chrome，Firefox，Safari 等），底层依赖轻量级的 Canvas 类库 ZRender，提供直观，生动，可交互，可高度个性化定制的数据可视化图表。

2.8 Java Script

JavaScript^[24]一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML（标准通用标记语言下的一个应用）网页上使用，用来给 HTML 网页增加动态功能。

2.9 jQuery

jQuery 是一个快速、简洁的 JavaScript 框架，是继 Prototype 之后又一个优秀的 JavaScript 代码库（或 JavaScript 框架）。jQuery 设计的宗旨是“write Less，Do More”，即倡导写更少的代码，做更多的事情。它封装 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互^[25]。

2.10 开发环境与编程语言

客户端开发环境：Windows10，Eclipse Mars.2。

客户端运行环境要求：Chrome 等其他浏览器。

服务器环境：Tomcat7.0。

本系统使用 Tomcat7.0 作为服务器，采用了 Spring、Spring MVC、iBatis、Quartz 等框架，服务端功能使用 Java 实现，前端功能展示使用 Jsp、jQuery、Echarts 等技术实现。

2.11 本章小结

本章以系统、网站开发的相关理论及技术为基础，介绍本系统开发过程中需要了解和掌握的技术。详细阐释了 Spring 框架、java 反射及异常链、Redis、iBatis 框架等相关技术。

第三章 作业调度监控系统功能设计

3.1 系统简介

3.1.1 系统需求概述

本系统以网页作为组织形式。系统需要实现的功能包括作业发布及修改、作业调度运行、作业异常监控告警，并且在前台展示页将作业调度运行的结果以图表的形式绘出。本系统使用 tomcat 作为服务器，MySQL 及 Redis 作为数据库，使用 Quartz 完成作业调度，使用 echarts 完成作业执行结果可视化。

本系统适用于任何有作业调度需求的企业。当有新的作业调度需求时，只需由开发人员编写业务代码，在作业发布系统上将其发布，之后由超级管理员在监控调度系统进行相关配置之后，就可以按照用户指定的调度运行规则来执行该作业，当作业调度运行过程中出现异常报错时，将会发送邮件告警给该作业的相关负责人。不管作业执行成功与否，都会在数据库产生一条执行结果记录，这些执行结果记录将会在前台以图表的形式展现。

3.1.2 系统功能概述

作业发布系统主要包括以下功能：

- 1) 发布作业：将新的作业相关信息存储到数据库。
- 2) 修改/删除作业：修改已经存在的作业，删除数据库隐藏的作业。
- 3) 远程作业执行：监控调度系统执行本系统内的作业。

调度监控系统主要包括以下功能：

- 1) 修改/删除作业：修改一个作业并配置其调度相关属性。被删除作业在前台不可展示（并没有实际删除，只是在前台不可展示）。
- 2) 调度作业：由 Quartz 框架根据作业的相关属性完成作业的调度运行。
- 3) 监控作业：在作业调度运行过程中，如果执行出现异常，则会将抛出的异

常 Catch。

- 4) 告警：在上一步的监控作业如果 catch 到异常，则将异常信息通过邮件的形式发送给作业相关负责人进行告警，同时暂停该做页的调度运行。
- 5) 作业执行结果图表展示：作业在每次调度运行之后，会将此次的执行结果存放至数据库，然后前台将根据数据库中的执行记录绘制作业调度运行的图表。
- 6) 人员信息管理：在系统提供添加、修改等人员信息管理等功能，该人员可在作业发布系统登陆对自己发布的作业进行管理。

3.2 系统的整体框架设计

3.2.1 作业发布系统架构设计

调度监控系统的核心功能主要集中在业务逻辑层上。它共有三层架构分为数据存储层、业务逻辑层、前端展示层。架构设计图如下图所示：

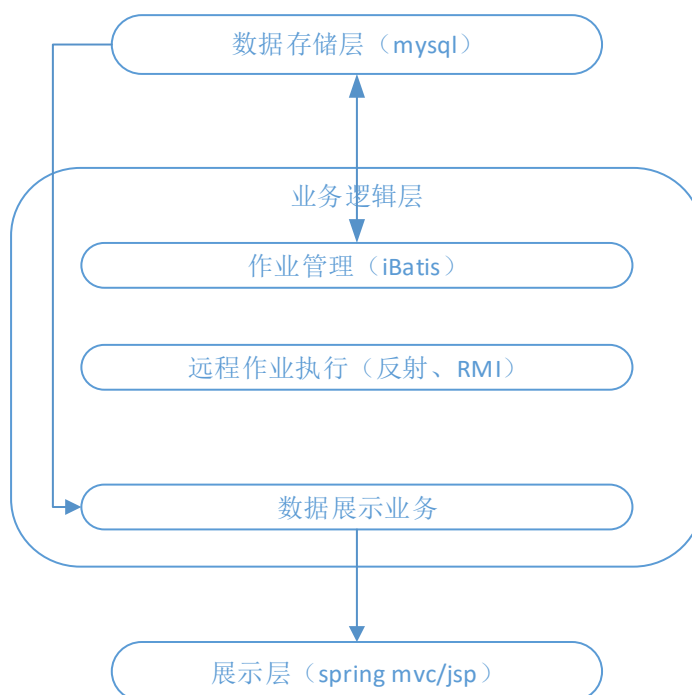


图 3-1 作业发布系统架构设计图

数据存储层负责数据的存储。它使用 iBatis 作为 ORM 框架与 MySQL 完成数据的存储与映射。

业务逻辑层包括作业管理、远程作业执行及数据展示业务。作业管理主要通过 iBatis 对数据库进行操作。远程作业执行使用 Java 的反射特性以及 RMI 来执行监控调度系统调度相关作业的执行。数据展示业务则是将数据进行处理传递给前台。

展示层负责将所存储的数据展示在前台页面，主要展示作业相关信息。

3.2.2 监控调度系统架构设计

调度监控系统的核心功能同样集中在业务逻辑层上。它共有三层架构分为数据存储层、业务逻辑层、前端展示层。架构设计图如下图所示：

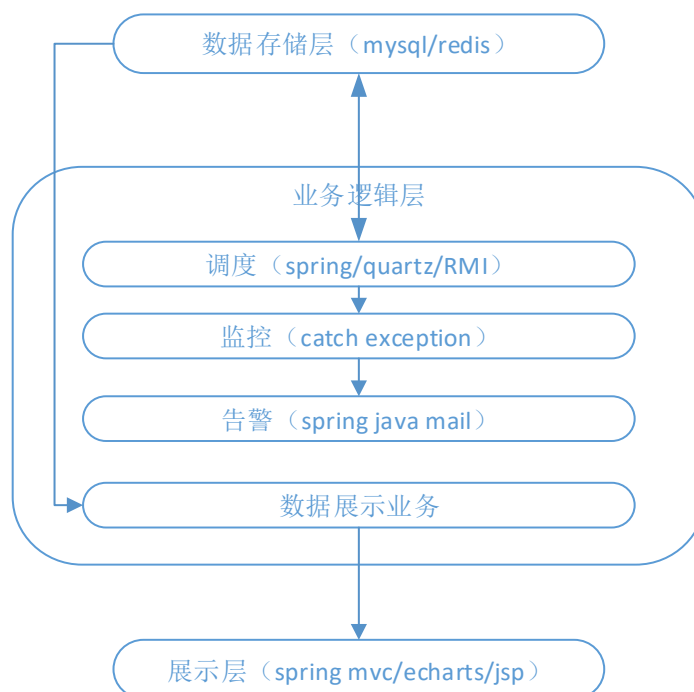


图 3-2 监控调度系统架构设计图

业务逻辑层包括调度、监控、告警以及数据展示业务。调度使用 Quartz、spring 以及 Java 反射等技术实现。监控使用异常抓取实现。使用 Spring 的 JavaMailSender 实现作业异常执行的告警。数据展示业务则是将数据进行特殊规则转为字符串交由前端展示。

数据存储层负责数据的存储。它使用 iBatis 作为 ORM 框架与 MySQL 完成数据的存储与映射，它存储的是历史数据。而 Redis 则存储最近一天作业执行成功的记录。

展示层负责将所存储的数据展示在前台页面，通过 echarts 与 jsp 实现，主要展示作业执行结果。

3.3 作业调度功能设计

3.3.1 调度器启动

调度器的启动伴随着系统的启动而启动，所以当 Spring 在运行之初，调度器也要随之启动。

3.3.2 作业调度

调度器启动后，会根据相关的条件对作业进行调度。调度功能由 Spring 与 Quartz 实现。首先从数据库中获取当前是处于激活的作业，接着根据该作业的相关信息（例如作业类名、作业方法名、运行所需参数、作业分组等相关信息）以及通用作业创建一个属于本作业的 jobDetail，然后将上述作业属性存放至 jobDataMap，接着根据作业类名以及 cron 表达式创建一个 trigger，设置完成之后，交由 Quartz 调度，当作业符合调度条件时就会由 Quartz 进行调度执行。

3.3.3 通用作业

通用作业是监控调度系统中最为重要的一个部分。它实现了 Quartz 的作业接口，是 Quartz 真正执行的类。在通用作业中，首先从 jobDetail 的 jobDataMap 中获取所要执行的作业相关信息（作业类名、方法名、运行参数、是否是调度监控系统内的作业等），接着根据是否是监控调度系统内的作业来执行方式。如果是系统内作业那么根据作业相关信息通过 Java 的反射去获取到作业类，接着通过反射去执行指定的方法，这样就可以通过通用作业完成对监控调度系统内作业的调用执行。如果作业是远程作业（存于作业发布系统）那么则会通过 RMI 方式以接口的形式去远程调用执行作业。

3.4 远程作业执行设计（作业发布系统）

远程作业执行是通过 RMI 的方式来实现。首先在作业发布系统设计且实现了一个执行作业接口。接着在监控调度系统调度执行远程作业时，调用该接口，通过 RMI 即可实现远程调用（即监控调度系统远程调度执行作业发布系统中的作业）。而执行作业接口的实现也是通过 Java 反射去获取到作业发布系统内的作业类进而完成调度执行。

3.5 作业监控功能设计

在通用作业执行指定任务的方法时，会将这整个过程进行 try、catch 检查执行时是否有异常抛出，如果 catch 到异常则将会打印日志、告警、执行失败记录入库以及停止该做页的执行。

3.6 作业执行记录数据存储功能设计

在通用作业类通过 Java 反射或 RMI 执行指定作业时，会将执行结果存放至 Redis 或 MySQL。如果作业执行成功，则会将数据存储至 Redis 中，接着将该作业成功执行所用的时间以及当前的执行时间以 zset 的方式存放至 Redis 中。

成功执行作业 set 设计为：key: job/value: id, id.....。作业成功执行详情 zset 设计为：key: job_id/member: 作业成功执行耗时_作业执行时间戳（此处添加作业执行时间戳是防止同一个作业执行时间相同将数据覆盖的问题出现）/score: 作业执行时间戳。因为这两部分数据在当天会被高频次的访问，所以这两个 key 的失效时间设置为一天。

在系统中有一个作业负责在每天 23: 50 将 Redis 中的数据刷到 MySQL 当中，作为历史数据入库。在该作业执行过程中首先从 Redis 当中获取到成功执行作业的 id，接着遍历这个 id 的 set 获取到每个作业成功执行的详情，然后根据相关规则解析入库。遍历结束后会将 Redis 当中的 key 置为失效状态，以防一直往 Redis 当中 push 数据导致 Redis 奔溃报错。

如果作业执行失败，首先会匹配该作业是否是心跳检测的作业或者是 Redis 数据刷入 MySQL 的作业，如果不是上述两个作业则会将数据入库，否则不入库。

3.7 异常告警功能设计

在通用作业类当中如果 `catch` 到异常，则会将该异常、作业名以及方法名传递给告警接口，在告警接口当中会将这些信息封装发送给作业相关负责人的邮箱。

在告警接口当中使用 `Spring` 的 `mail` 服务来发送邮件。在封装邮件内容过程中会调用一个辅助工具类将异常的堆栈信息遍历打印，这样在告警邮件中就会有一个完整的作业异常信息，方便作业相关负责人及时处理异常。

3.8 前台展示功能设计

系统前台使用 `jsp`、`jQuery` 以及 `echarts` 完成系统前台功能展示。

作业列表页通过超链接、`jQuery` 以及 `ajax` 技术完成作业的操作功能，例如启动、修改以及删除等。同时作业名将超链接到作业的执行结果展示页。作业执行结果页分为饼状图以及折线图两种，这两种图表都是 `echarts` 中的组件。

3.9 系统流程

系统启动时，首先判断是否查看作业调度情况，如果是则展示作业调度情况，否则判断是否对作业进行相关操作（添加、激活、修改、删除、取消激活）。如果操作作业则更新 `DB`，否则由 `Quartz` 来进行调度作业，最后检查作业是否被成功调度执行，如果没有成功调度执行则进行告警。具体流程如下图所示：

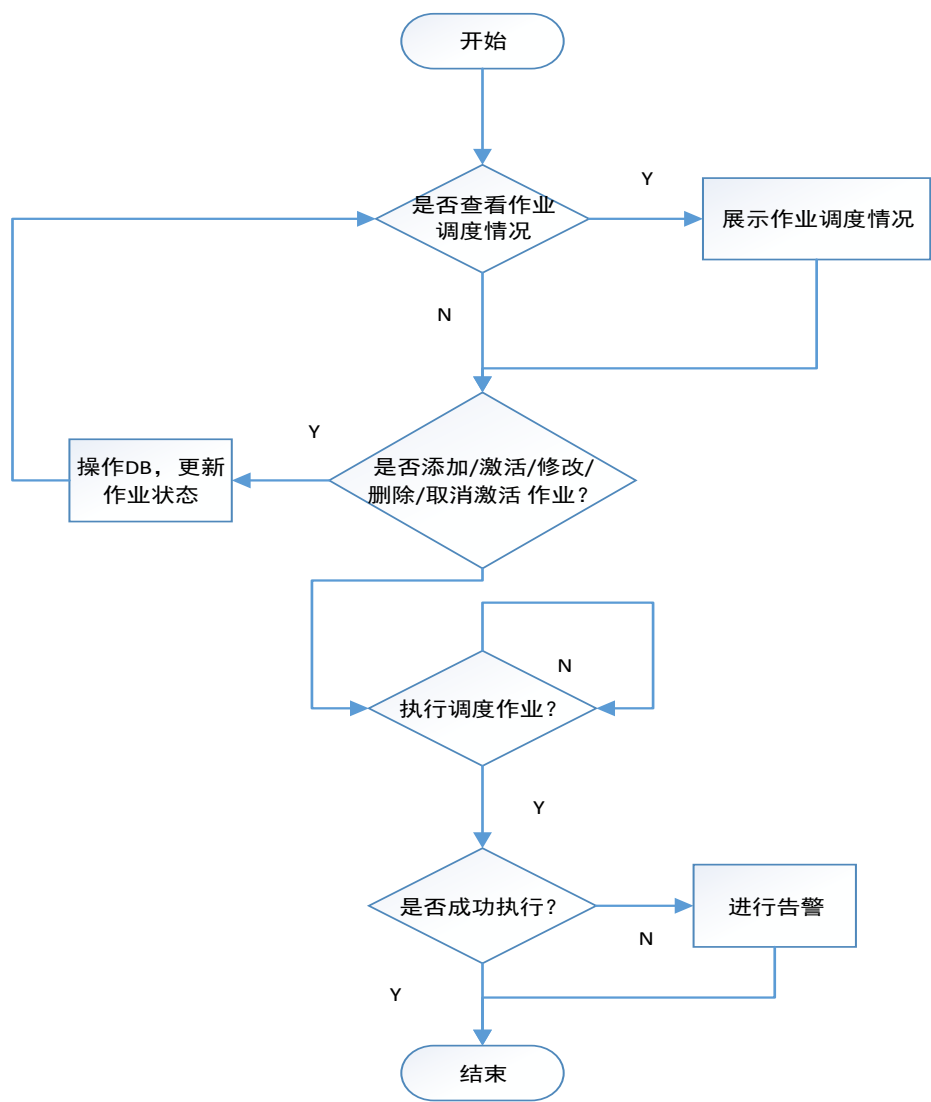


图 3-3 系统功能流程图

3.10 本章小结

本章主要针对系统的功能模块进行设计。首先概述了系统的实现目标，然后确定了系统整体的架构设计。接着对系统的各个功能做了设计阐述，最后对系统的功能流程进行了描述。

第四章 作业调度监控系统功能实现

4.1 系统开发规范

4.1.1 作业发布系统工程结构

- src/main.java
 - com.rephen.controller
 - JobController.java 作业 Controller
 - LoginController.java 登陆 Controller
 - UserController.java 人员 Controller
 - com.rephen.dao
 - BasicQuartzJobDAO.java 作业 DAO 接口
 - DAOException.java DAO 异常类
 - GenericDAO.java 基础 DAO 操作类
 - UserInfoDAO.java 人员 DAO 接口
 - com.rephen.dao.impl
 - BasicQuartzJobDAOImpl.java 作业 DAO 实现类
 - UserInfoDAOImpl.java 人员 DAO 实现类
 - com.rephen.domain
 - BasicQuartzJob.java 作业实体类
 - UserInfo.java 人员实体类
 - com.rephen.general.service
 - GeneralJobService.java 远程作业调用接口
 - com.rephen.general.service.impl
 - GeneralJobServiceImpl.java 远程作业调用实现类
 - com.rephen.interceptor
 - LoginInterceptor.java 登陆验证拦截器

- com.rephen.listener
 - SystemConfigInitListener.java 配置文件读取监听器
- com.rephen.task
 - ExceptionTask.java 测试作业（异常告警）
 - TestTaskOne.java 测试作业（图表可视化）
- src/main/resources
 - spring
 - config.properties 配置文件（jdbc,log 日志级别）
 - spring-*.xml spring 配置文件
 - sqlmaps
 - *SQL.xml iBatis sql 映射文件
 - SqlMapConfig.xml iBatis 配置文件
- src/test/java 该包下存放测试类
- src/main/weapp
 - css
 - js
 - WEN-INF
 - *.jsp

4.1.2 监控调度系统工程结构

- src/main.java
 - com.rephen.component
 - JobSaveResult.java 作业保存结果枚举类
 - com.rephen.controller
 - JobConfigController.java 作业 Controller
 - LoginController.java 登陆 Controller
 - UserInfoController.java 人员 Controller

- com.rephen.dao
 - BasicQuartzJobDAO.java 作业 DAO 接口
 - DAOException.java DAO 异常类
 - GenericDAO.java 基础 DAO 操作类
 - UserInfoDAO.java 人员 DAO 接口
 - JobExcuteResultDAO.java 作业执行结果 DAO 接口
- com.rephen.dao.impl
 - BasicQuartzJobDAOImpl.java 作业 DAO 实现类
 - UserInfoDAOImpl.java 人员 DAO 实现类
 - JobExcuteResultDAOImpl.java 作业执行结果 DAO 实现类
- com.rephen.domain
 - BasicQuartzJob.java 作业实体类
 - UserInfo.java 人员实体类
 - JobExcuteResult.java 作业执行结果实体类
 - QuartzJobBean.java 通用作业类
- com.rephen.general.service
 - GeneralJobService.java 远程作业调用接口
- com.rephen.interceptor
 - LoginInterceptor.java 登陆验证拦截器
- com.rephen.listener
 - SystemConfigInitListener.java 配置文件读取监听器
- com.rephen.query
 - QueryParam.java 作业查询参数封装类
- com.rephen.service
 - MailService.java 作业异常告警邮件接口
- com.rephen.service.impl
 - MailServiceImpl.java 作业异常告警邮件实现类

- com.rephen.task
 - BasicRedisToDBTask.java 本地任务（reids 数据刷到 DB）
 - BasicScheduler.java 调度器相关操作类
 - BasicTaskMonitor.java 本地任务（心跳检测）
- src/main/resources
 - spring
 - config.properties 配置文件（jdbc,log 日志级别）
 - spring-*.xml spring 配置文件
 - sqlmaps
 - *SQL.xml iBatis sql 映射文件
 - SqlMapConfig.xml iBatis 配置文件
- src/test/java 该包下存放测试类
- src/main/weapp
 - css
 - js
 - WEN-INF
 - *.jsp

4.1.3 代码命名规范

系统中涉及到的界面采用英文加下划线的形式命名，如“error_page.jsp”
Spring 配置文件以英文加连接号的形式命名，如“spring-job.xml”，而函数、数据结构等采用驼峰命名方法，如“JobController”、“sendMail”。

4.2 系统数据结构实现

4.2.1 BasicQuartzJob 数据结构

BasicQuartzJob 数据结构如下：

```

private Long id;           //作业 id
private String jobClass;   //作业类名
private String jobMethod;  //作业方法名
private String jobArguments; //作业运行参数
private String jobGroup;   //作业组
private String jobName;    //作业名称
private Integer status;    //作业运行状态
private Integer isHide;    //是否隐藏
private String cronExpression; //作业调度运行的 cron 表达式
private String description; //作业描述
private Date createTime;   //作业创建时间
private Date updateTime;   //作业修改时间
private Long userId;       //作业负责人的人员 id
private String username;   //作业负责人的账号
private Boolean local;     //作业是否是监控调度系统内的任务

```

BasicQuartzJob 是作业的实体类，id 作为它的唯一标识。在这其中，local、username 这两个属性是后期根据 job 信息处理封装进去的，并不是数据库中实际存在的数据。如图 4-1 为 BasicQuartzJob 的一个实例对象。

```

> ■ createTime= Date (id=76)
> ■ cronExpression= "0/30 * * * * ?" (id=79)
> ■ description= "异常测试作业" (id=82)
> ■ id= Long (id=83)
> ■ isHide= Integer (id=84)
> ■ jobArguments= "" (id=86)
> ■ jobClass= "ExceptionTask" (id=87)
> ■ jobGroup= "1" (id=88)
> ■ jobMethod= "sendMail" (id=89)
> ■ jobName= "exceptionTaskf" (id=90)
> ■ local= null
> ■ status= Integer (id=92)
> ■ updateTime= Date (id=93)
> ■ userId= Long (id=94)
> ■ username= null

```

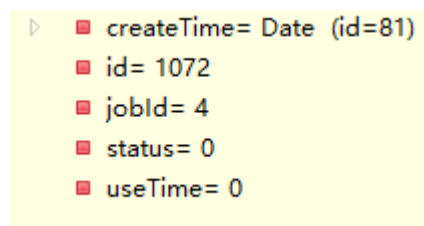
图 4-1 BasicQuartzJob 对象

4.2.2 JobExcuteResult 数据结构

JobExcuteResult 的数据结构如下所示：

```
private long id;           //作业执行记录 id
private long jobId;        //作业 id
private int status;        //执行记录状态（成功/失败）
private long useTime;      //作业执行耗时
private Date createTime;   //作业执行记录创建时间
```

JobExcuteResult 是一个作业执行技术实体类，id 作为一条作业执行记录的唯一标识。status=1 时表示这条记录是一条成功执行的记录，status=0 时表示这条记录是一条执行失败的记录。下图所示为一条作业执行失败的记录：



```
▶ createTime= Date (id=81)
  id= 1072
  jobId= 4
  status= 0
  useTime= 0
```

图 4-2JobExcuteResult 对象

4.2.3 UserInfo 数据结构

UserInfo 的数据结构如下所示，id 作为用户的主键，username 为唯一标识。

```
private Long id;           //用户 id
private String username;    //用户账号
private String password;    //用户密码
private String mailAddress; //用户邮箱
private Date createTime;    //创建用户的时间
private Date updateTime;    //更新用户的时间
```



```
▷ ■ createTime= Date (id=88)
▷ ■ id= Long (id=89)
▷ ■ mailAddress= "452371131@qq.com" (id=90)
▷ ■ password= "aaa" (id=91)
▷ ■ updateTime= Date (id=92)
▷ ■ username= "aaa" (id=93)
```

图 4-3UserInfo 对象

4.2.4 Redis 内作业执行结果数据结构

因为 Redis 是一个基于 key-value 的内存数据库，从中查询数据要比 MySQL 快，所以将访问频次最高的数据即最近一天所有成功执行的作业 id 以及其执行记录（耗时、执行时间戳）存放至 Redis。

本系统使用 Redis 存放两部分数据，分别为成功执行作业的 id 以及每个作业成功执行的记录。

如图 4-4 所示为成功执行作业 id 的数据结构。它使用的是 Redis 当中的 set 作为存储结构，“job”是该记录的 key，“5”、“6”是它的 member。所以它的含义是成功执行的作业 id 为 5、6。

```
127.0.0.1:6379> SMEMBERS job
1) "5"
2) "6"
127.0.0.1:6379> ■
```

图 4-4 成功执行作业 id 数据结构

id 某作业成功执行数据结构如图 4-5 所示。它使用 Redis 的 zset(有序的 set) 作为数据存储结构。使用 zset 的原因是因为它的一个 value 可以存放两部分数据 (member、score) 且取数据是有序的，可满足系统展示作业成功执行记录的条件：按作业成功执行记录的先后顺序展示每条记录的耗时。key: job_成功执行的作业 id, member: 耗时_时间戳（这里加时间戳的原因是防止出现耗时相同的记录覆盖原有数据问题的发生），score: 时间戳。

```
127.0.0.1:6379> ZRANGE job_5 0 -1 withscores
1) "4384_1493955104399"
2) "1493955104399"
3) "2425_1493955107440"
4) "1493955107440"
5) "1847_1493955111865"
6) "1493955111865"
7) "1200_1493955116210"
8) "1493955116210"
9) "2830_1493955122847"
10) "1493955122847"
11) "568_1493955125585"
12) "1493955125585"
13) "2376_1493955132394"
14) "1493955132394"
15) "1544_1493955136558"
16) "1493955136558"
17) "112_1493955140127"
18) "1493955140127"
19) "1266_1493955146282"
20) "1493955146282"
```

图 4-5 作业成功执行数据结构

上图所示为：取 jobId 为 5 的全部成功执行记录，1）（member）表示第一条成功执行记录的耗时_时间戳，2）（score）表示该作业执行时的时间戳。

4.3 作业调度功能实现

4.3.1 调度器启动

当监控调度系统启动时，希望调度功能可以随之开启，所以在 Spring 启动时开启了 Spring 监听（ApplicationListener<ApplicationEvent>），即 basicScheduler 这个 bean 实现了 ApplicationListener<ApplicationEvent>接口，之后只要在系统启动时，就会自动开启调度器。

4.3.2 作业调度

调度器启动后从数据库当中查出当前是运行状态的作业，然后遍历将其 local 属性设置为 false（数据库中存储的作业都是远程作业），设置完成之后将其一一启动（需要注意的是此时调度器并没有真正启动）。

```
// 远程作业从数据库中查询, local为false
for (BasicQuartzJob job : list) {
    if (BasicQuartzJob.JOB_STATUS_ON == job.getStatus()) {
        try {
            job.setLocal(false);
            this.enable(job);
        } catch (Exception e) {
            log.error("[BasicScheduler]error when add job. " + e.getMessage(), e);
        }
    }
}
}
```

图 4-6 查出远程作业进行启动代码

启动后根据作业的名称、分组信息及通用作业类（通用作业具体将在后面阐述）创建一个属于该作业的 jobDetail，创建之后将作业的相关属性存放至 jobDetail 的 jobDataMap（jobDataMap 存放的是 job 的实例信息）中，例如作业类名、作业方法名称、作业参数、以及作业相关负责人的邮箱。然后根据作业的类型、分组以及作业调度的 cron 表达式创建 trigger，之后使用 Quartz 调度该作业，入参是 jobDetail 以及 trigger（此时作业真正被调度）。

```
JobDetail jobDetail = new JobDetail(basicQuartzJob.getJobName(),
    basicQuartzJob.getJobGroup(), QuartzJobBean.class);
jobDetail.getJobDataMap().put(QuartzJobBean.TARGET_CLASS, basicQuartzJob.getJobClass());
jobDetail.getJobDataMap().put(QuartzJobBean.TARGET_METHOD,
    basicQuartzJob.getJobMethod());
jobDetail.getJobDataMap().put(QuartzJobBean.TARGET_ARGUMENTS,
    basicQuartzJob.getJobArguments());
jobDetail.getJobDataMap().put(QuartzJobBean.TARGET_USER_ID,
    basicQuartzJob.getUserId());
jobDetail.getJobDataMap().put(QuartzJobBean.TARGET_JOB_ID,
    basicQuartzJob.getId());
jobDetail.getJobDataMap().put(QuartzJobBean.TARGET_JOB_IS_LOCAL,
    basicQuartzJob.isLocal());

trigger = new CronTrigger(basicQuartzJob.getTriggerName(), basicQuartzJob.getJobGroup(),
    basicQuartzJob.getCronExpression());
scheduler.scheduleJob(jobDetail, trigger);
```

图 4-7 启动某个作业&设置属性代码

4.3.3 通用作业

通用作业是本系统中最为重要的一个部分。首先它实现了 Quartz 的 StatefulJob 的 excute 方法（此方法即为作业被调度运行时，实际执行的方法）。

excute 方法的入参是一个 JobExecutionContext，它存放 job 运行的所有信息包括 trigger、jobDataMap、jobDetail。在 excute 方法中，首先从

JobExecutionContext 的 jobDataMap 中获取作业类名、方法名、方法参数以及作业相关负责人邮箱。

```
String targetClass = (String) context.getMergedJobDataMap().get(TARGET_CLASS);
String targetMethod = (String) context.getMergedJobDataMap().get(TARGET_METHOD);
String methodArgs = (String) context.getMergedJobDataMap().get(TARGET_ARGUMENTS);
Long userId = (Long) context.getMergedJobDataMap().get(TARGET_USER_ID);
Long id = (Long) context.getMergedJobDataMap().get(TARGET_JOB_ID);
Boolean jobIsLocal = (Boolean) context.getMergedJobDataMap().get(TARGET_JOB_IS_LOCAL);

if (StringUtils.isEmpty(targetClass) || StringUtils.isEmpty(targetMethod))
    return;

Object[] args = null;
if (!StringUtils.isEmpty(methodArgs)) {
    methodArgs = methodArgs + " ";
    String[] argString = methodArgs.split("#&");
    args = new Object[argString.length];
    for (int i = 0; i < argString.length; i++) {
        args[i] = argString[i].trim();
    }
}
```

图 4-8 通用作业获取执行指定作业信息代码

如果是监控系统内的作业则通过 applicationContext（spring 上下文）根据类名获取到该作业的 bean，然后通过 Java 的反射机制来执行作业类的指定的任务方法。如果作业存在于作业发布系统（即远程系统）则通过 RMI 调用远程接口执行该作业。

```
// 本地任务
Object target = ac.getBean(targetClass);
if (null != target) {
    Class tc = target.getClass();
    Class[] parameterType = null;
    if (args != null) {
        parameterType = new Class[args.length];
        for (int i = 0; i < args.length; i++) {
            parameterType[i] = String.class;
        }
    }
    Method method = tc.getDeclaredMethod(targetMethod, parameterType);
    if (null != method) {
        method.invoke(target, args);
    }
}
```

图 4-9 通用作业反射执行本地（监控调度系统内部）作业代码

4.4 远程作业执行功能实现

远程作业调度通过 RMI 方式实现。首先在作业发布系统申明一个远程调用接口，接着实现该接口。作业监控调度系统调度该接口时会将作业的类型、方法名以及方法参数以入参的形式传递给接口。接口内部根据反射获取到作业类之后运行，此时完成了监控调度系统远程调度执行作业发布系统内的作业。

最后在 Spring 配置文件中配置 RMI 如下图所示：

```
<bean id="generalJobService" class="com.rephen.general.service.impl.GeneralJobServiceImpl" />
<bean id="generalJobServiceExporter" class="org.springframework.remoting.rmi.RmiServiceExporter">
  <!-- 调用Service -->
  <property name="service" ref="generalJobService"></property>
  <!-- value值是给用户调用 -->
  <property name="serviceName" value="generalJobService"></property>
  <!-- service 接口 -->
  <property name="serviceInterface" value="com.rephen.general.service.GeneralJobService"></property>
  <!-- 注册端口号 -->
  <property name="registryPort" value="1099"></property>
</bean>
```

图 4-10 作业发布系统配置代码

4.5 作业监控功能实现

作业的监控功能是在通用作业的基础上实现的。在通用作业内部会对整个执行作业的过程进行 try、catch，这样 try 的代码块实际上就是被监控的，catch 的代码块就是检测到异常后进行的操作。在监控远程作业的执行时，需要在远程接口 throws Exception，这样才可以实现监控远程作业的执行，否则在这里是无法实现监控的。

监控调度系统有个本地作业：BasicTaskMonitor，它的作业类似于心跳检测。该作业每一分钟该作业会被调度执行一次打印出“DEFAULT TASK IS VIABLE”这样一句日志，所以当系统崩溃后可以根据该条日志来查询系统崩溃的时间。

4.6 作业执行记录数据存储功能实现

在通用作业类中通过 Java 反射或 RMI 执行指定作业时，会将执行结果存放至 Redis 或 MySQL。如果作业执行成功，则会将该作业 id 以 set 的形式存放至 Redis 中，接着将该作业成功执行所用的时间以及当前的执行时间以 zset 的方式

存放至 Redis 中。但是如果作业执行失败即在通用作业执行过程中 catch 到异常，则会将作业执行失败的记录直接写入 MySQL。

成功执行作业 set 设计为：key: job/value: id, id.....。作业成功执行详情 zset 设计为：key: job_id/member: 作业成功执行耗时_作业执行时间戳（此处添加作业执行时间戳是防止同一个作业执行时间相同将数据覆盖的问题出现）/score: 作业执行时间戳。因为这两部分数据在当天会被高频次的访问，所以这两个 key 的失效时间设置为一天。

```
// 成功job id统计
scheduler.getJedisUtil().sadd(JOB, String.valueOf(id));
// 统计job成功结果写入redis zset 结构 score: timeStamp 为防止member重复导致更新此处member也加时间戳
scheduler.getJedisUtil().zadd(JOB + "_" + id,
    String.valueOf(System.currentTimeMillis() - startTime) + "_" + System.currentTimeMillis(),
    System.currentTimeMillis());
// 过期时间为1天
scheduler.getJedisUtil().expire(JOB + "_" + id, ONE_DAY);
scheduler.getJedisUtil().expire(JOB, ONE_DAY);
```

图 4-11 作业成功执行数据入库代码

在系统中有一个作业负责在每天 23: 50 将 Redis 中的数据刷到 MySQL 当中，作为历史数据入库。在该作业执行过程中首先从 Redis 当中获取到成功执行作业的 id，接着遍历这个 id 的 set 获取到每个作业成功执行的详情，然后根据相关规则解析入库。遍历结束后会将 Redis 当中的 key 置为失效状态，以防一直往 Redis 当中 push 数据导致 Redis 奔溃报错。

如果作业执行失败，首先会匹配该作业是否是心跳检测（BasicTaskMonitor）的作业或者是 Redis 数据刷入 MySQL 的作业（BasicRedisToDBTask），如果不是上述两个作业则会将数据入库，否则不入库。

4.7 异常告警功能实现

在通用作业中，如果检测到有异常抛出时会对异常进行判断，异常链的原始异常为空（e.getCause（）==null）那么将异常本身作为异常告警的入参，否则将 e.getCause（）作为异常告警的入参。MailService（异常告警）使用 Spring 的 JavaMailSender 来实现。

在 MailServiceImpl 中首先封装邮件的一些基本内容（如收件人、标题、发送人的地址等）。接下来封装邮件的正文即作业执行过程中发生的异常错误信息，

异常错误信息通过 `GetExceptionDetailUtil` 工具类来获取, 在该类中会再尝试获取一次异常链的原始异常, 如果可以获取到则将整个异常链的信息返回, 否则只返回异常本身信息。

```
public static String getExceptionAllInformation(Throwable ex) {
    String sOut = "cause:"+ex.getMessage()+"\r\n";

    StackTraceElement[] trace;
    if(null != ex.getCause()){
        trace= ex.getCause().getStackTrace();
    }else{
        trace= ex.getStackTrace();
    }

    for (StackTraceElement s : trace) {
        sOut += "\tat " + s + "\r\n";
    }
    return sOut;
}
```

图 4-12GetExceptionDetailUtil 工具类

异常告警反复尝试获取异常链的原因是: 通过反射执行作业, `exception` 本身的异常堆栈只包括了反射部分异常的信息, 并非是所期待的作业异常即原始异常, 所以根据异常链获取原始异常 (`excepiton.getCause()`) 即可解决异常“答非所问”的情况。

4.8 前台展示功能实现

前台展示的是作业的执行记录, 分为饼状图和折线图。

饼状图展示的是某作业执行成功、失败的次数, 同时它也是作业列表页超链接的入口页, 饼状图中的执行成功次数区域绑定了一个点击事件。该点击事件将会触发打开一个新页面并且展示作业最近一天的执行成功记录的折线图, 如果最近一天该做页未被执行或未被成功执行过, 则会跳转到报错页面, 同时会有一个展示历史数据的超链。

最近一天的折线图是从 `redis` 当中获取数据, 然后通过字符串拼接的方式传递给前端 `echarts` 框架, 进行显示。而历史数据是从数据库当中取成功执行的历

史记录进行显示。下图显式的是拼接字符串的代码

```
public static String[] getJobExcuteDetail(List<JobExcuteResult> results) {
    if (CollectionUtils.isEmpty(results)) {
        return null;
    } else {
        String[] content = {"", ""};
        String xAxis = "[";
        String data = "[";
        for (int i = 0; i < results.size(); i++) {
            xAxis += "\'";
            xAxis += yyyyMMddHHmmSS.format(results.get(i).getCreateTime());
            xAxis += "\'";
            data += results.get(i).getUseTime();
            if (i < results.size() - 1) {
                xAxis += ",";
                data += ",";
            }
        }
        xAxis += "]";
        data += "]";
        content[0] = xAxis;
        content[1] = data;
        return content;
    }
}
```

图 4-13 作业成功执行结果 json 拼接代码

4.9 界面实现

系统界面是基于 jsp、css、jQuery、Bootstarp 以及 echarts 等组件实现的。其中监控调度系统与作业发布系统的界面是有相似之处的，所以此处只选择性的阐述几个界面的实现。

4.9.1 作业列表页

作业列表页展示的是作业的信息，在该页面可以对作业进行相关操作（修改、启动/停止、删除），同时还可以查看某作业的执行记录。作业的相关操作以及查看作业执行记录都是使用超链接及 ajax 的方式实现的。

作业调度监控系统									
作业列表		人员列表	添加/修改人员	注销					
作业标识	作业类	作业方法	作业参数	作业组	作业名称	作业状态	作业调度参数	作业描述	负责人
4	ExpectionTask	sendMail		1	expectionTaskf	停止中	0/30 *****	异常测试作业	aaa
5	TestTaskOne	run		1	testTaskOne	运行中	0/5 *****	图表测试作业	aaa

图 4-14 作业列表页

4.9.2 作业修改页

作业实例

注销

job_class:

ExpectionTask

job_method:

sendMail

job_arguments:

job_group:

1

job_name:

expectionTaskf

cron_expression:

0/30 * * * * ?

description:

异常测试作业

username:

aaa

提交

图 4-15 作业修改页（1）

4.9.3 人员列表页

人员列表页展示的是人员信息，此处人员列表不展示超级管理员的信息，以防对管理员信息进行修改。

作业调度监控系统

作业列表

人员列表

添加/修改人员

注销

id	用户名	邮箱地址	动作
3	aaa	452371131@qq.com	修改 重置密码 删除
4	cc	cc	修改 重置密码 删除
5	aaaa	aaaa	修改 重置密码 删除

图 4-166 人员列表页

此处的修改/重置密码/删除同样是基于超链接和 ajax 实现的。

4.9.4 人员修改页

人员修改页同时也可以新增人员，只需点上方菜单的添加/修改人员即可跳转到添加人员页面。人员添加页和人员修改页的区别就是 password 这一栏在人员修改页不现实，并且 username 这一项不可编辑。点击提交后会新增人员或修改，如果存储数据时发生错误则会跳转至 error_page.jsp。

作业调度监控系统

作业列表

人员列表

添加/修改人员

注销

username:

aaa

mail_address:

452371131@qq.com

提交

图 4-177 人员修改页

4.9.5 作业执行结果展示页

图 4-19 显式的是某作业历史的执行情况，分为失败次数和成功次数以及其各自所占全部执行记录的百分比。成功次数的饼状图绑定了一个点击事件，点击之后会跳转至该作业最近一天执行成功的图表页。

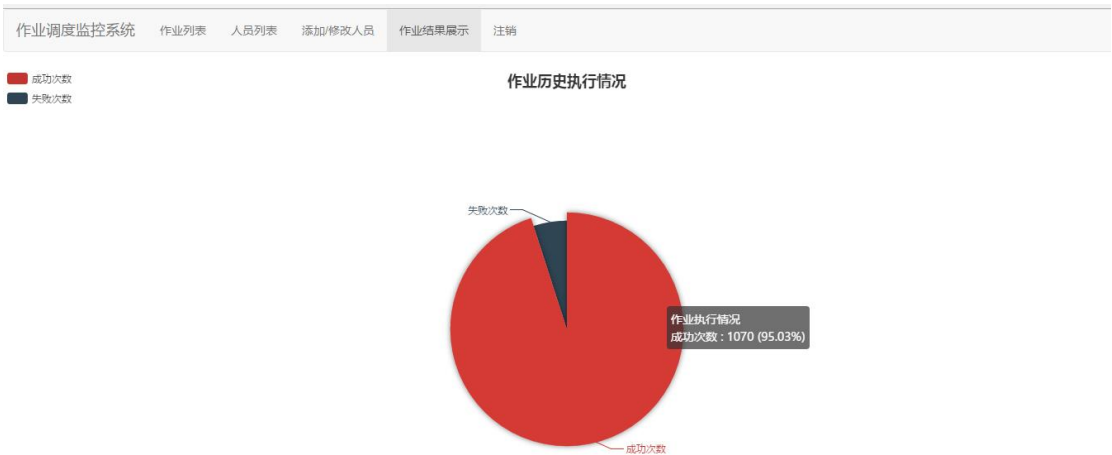


图 4-188 作业执行历史记录页

图 4-19 显式的是某作业最近一天成功执行的记录。它展示了作业在何时成功执行、执行耗时。图标下方是一个可以缩放的 x 轴，图表将根据 x 轴的缩放范

围来展示内容。x 轴的数据和图表数据都存储在页面的隐藏域（input type=“hidden”）。点击图标下方的查看作业全部历史记录可以跳转至该作业称该执行的历史记录页，如图 4-20 所示。



图 4-19 作业最近一天成功执行记录页



图 4-20 作业历史成功执行记录页

4.10 本章小结

本章根据第三章所描述的系统功能设计，给出响应的功能实现。首先介绍了系统的结构以及编码规范。接着详细介绍了系统中所使用到的数据结构，并且介绍了系统中各个功能的实现方式，包括调度功能、远程作业执行功能、监控异常告警功能以及作业执行记录存储功能等。

第五章 总结

5.1 完成的工作

本文介绍了基于 quartz 框架作业监控调度系统。使用 Quartz、Spring 以及 Java 反射完成了基础的作业调度功能，使用 RMI 实现了远程作业调度的执行。本系统目前可基本满足公司企业日常作业调度的需求。

5.2 存在的问题及下一步工作

由于系统使用 RMI 实现远程作业调度执行，所以当作业发布系统未运行时或重启时，由于未能链接导致调度监控系统运行会出错。所以后续希望采用 mq 消息中间件来完成通讯的实现，调度监控系统为消息生产者，作业发布系统为消息的消费者。当监控调度系统希望调度执行某作业时，会发送一个消息（消息体为作业类名、作业方法名、运行参数等）到消息队列，当作业发布系统的消费者收到相关消息时，对这条消息进行解析然后反射执行作业即可代替 RMI 的远程调用，且能规避上述问题。本文的下一步工作是完善系统功能，例如作业管理以及人员密码的安全性存储等。

参考文献

- [1] Sheikhalishahi M, Wallace R M, Grandinetti L, et al. A multi-dimensional job scheduling[J]. Future Generation Computer Systems, 2016, 54(C):123-131.
- [2] 郑雪松, 谭红杨. Java 作业调度应用实例分析[J]. 电脑知识与技术:学术交流, 2007, 3(17):1348-1348.
- [3] 张鹏, 白朝旭, 王锬,等. 基于 Quartz 的集团化调度任务分布部署研究[J]. 现代电子技术, 2014(2):80-83.
- [4] IBMdW. 基于 Quartz 开发企业级任务调度应用 [EB/OL]. http://www.oschina.net/question/129540_111323,2017-2-4.
- [5] Sauron Software. cron4j[EB/OL]. <http://www.sauronsoftware.it/projects/cron4j/index.php>,2017-2-4.
- [6] 银时 . 探秘阿里分布式任务调度服务 SchedulerX[EB/OL]. <https://yq.aliyun.com/articles/57780>,2017-2-5.
- [7] 李刚. 疯狂 Java 讲义.第 2 版[M]. 电子工业出版社, 2012.
- [8] Oracle. Java Reflection API[EB/OL]. <http://docs.oracle.com/javase/8/docs/technotes/guides/reflection/index.html>,2017-4-28.
- [9] BruceEckel. Java 编程思想:第 4 版[M]. 机械工业出版社, 2007.
- [10]Cavaness C. Quartz Job Scheduling Framework: Building Open Source Enterprise Applications[M]. Prentice Hall PTR, 2006.
- [11]Tools S. J2EE Development with Free and Open-Source Tools[J]. 2014.
- [12]丁振凡, 李馨梅. Spring 的任务定时调度方法的研究比较[J]. 智能计算机与应用, 2012, 02(4):55-56.
- [13]Djkin. Quartz 触发器（SimpleTrigger&CronTrigger）配置说明 & cronExpression 表达式[EB/OL]. <http://djkin.iteye.com/blog/1714323>
- [14]郝佳. Spring 源码深度解析[M]. 人民邮电出版社, 2013.
- [15]翟剑锬. Spring 框架技术分析及应用研究[D]. 中国科学院大学(工程管理与信息技术学院), 2013.

- [16]Redislabs. Redis Documentation [EB/OL]. <https://redis.io/>,2017-2-9.
- [17]Brad Fitzpatrick. Memcached [EB/OL]. <http://memcached.org/>,2017-2-17.
- [18]Baike. Redis [EB/OL]. <http://baike.baidu.com/item/Redis>,2017-2-20.
- [19]李澎林, 朱国清, 吴斌. 基于 iBatis SQL Map 的数据持久层实现应用研究[J]. 浙江工业大学学报, 2008, 36(1):72-76.
- [20]梁文菲, 黄厚宽. 对象/关系映射技术与面向对象数据库技术比较分析[J]. 中国科技信息, 2006(21):154-156.
- [21]Wikipedia. iBatis [EB/OL]. <https://en.wikipedia.org/wiki/IBATIS>,2017-2-9.
- [22]陈建华, 胡开明. 基于 RMI 的 Java 远程调用[J]. 电脑编程技巧与维护, 2016(9):34-35.
- [23]王洪九. 运用 Echarts 组件和 Ajax 技术展现商业图表[J]. 中国电子商务, 2014(23):36-36.
- [24]W3School. JavaScript 教程[EB/OL]. <http://www.w3school.com.cn/js/>.
- [25]朱育发. jQuery 与 jQuery Mobile 开发完全技术宝典[M]. 中国铁道出版社, 2014.

致 谢

此处我要感谢我的导师韩姗姗老师。在毕设中期因为设计方面的不合理，导致毕设实现后与题目所定不符合，幸运的是韩姗姗老师及时纠正了我毕设设计的不合理，并给出了一些技术方案。同时感谢韩姗姗老师不厌其烦、细致入微的批改论文，给予我专业的意见。

附 录

附件 1 毕业设计文献综述

附件 2 毕业设计开题报告

附件 3 毕业设计外文翻译（中文译文与外文译文）