学号 2012301220041
密级

武汉大学本科毕业论文

服务器端 Socket 编程实现
网络数据汇聚

院（系）名称 ：电子信息学院

专业名称 ：光信息科学与技术

学生姓名 ：王维恒

指导教师 ：孙涛 教授

二〇一六年五月

BACHELOR'S DEGREE THESIS OF WUHAN UNIVERSITY

Data Aggregation for Server with Socket Programming

College : Wuhan University

Subject : Photonic Science & Technology

Name : Wang Weiheng

Directed by : Prof. Suntao

May. 2016

郑重声明

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

本人签名：_____　　日期：_____

摘　要

论文的研究来源于973课题 :“空天地一体化传感网”的子模块。

本项目以前的版本中存在着使用Java服务器在通信中经常出现丢包现象的问题和数据在采集后的存储与分析工具缺失的问题，这些问题对在实际科研当中的数据分析造成了极大的困扰。经过对日志的查看，笔者发现是基于传统的Java BIO(block-I/O)通信机制的性能较差，无法面对高并发情况，造成服务器临时拒绝访问。这是由于传统的Java BIO机制是采用同步的流I/O和多线程的并发网络I/O处理方式。这种同步阻塞模型凸显了I/O速度和CPU速度差异的矛盾。此外, 虽然多线程模型使用了线程池来合理分配资源，但是线程的创建和上下文的切换的消耗还是很大，很容易到达机器的物理极限。针对上述传统Java服务器的问题，本文研究和实现了基于Netty的非阻塞式I/O服务器搭配MySQL数据库，提高了服务端并发能力，解决了多线程数据传输丢包问题和数据本地持久化问题。

此外,传统的传感网系统技术的项目版本中，对于数据的处理未做可视化处理，数据包管理不善对于后期的数据维护没有做到很好的持久化和可视化。

本文的目的和意义是解决以上两个问题，即保证相关科研项目的数据安全和数据正常处理。本文首先改进了基于Socket通信模型的服务端，引入了在JDK 1.4提出的NIO(non-block I/O)。NIO是一种来自Linux的I/O模型中的epoll模型的非阻塞I/O通信机制。本文则是采用了在业界比较成熟的基于NIO的Netty框架，作为了服务器的Socket通信框架。其次，选用MySQL数据库作为数据持久化工具，选用JavaEE作为规范，使用它的JSP和Servlet创建WEB应用来作为项目管理平台，并在PC端编写了可视化软件方便使用者查看数据。此外，还实现了管理者对传感器的操作功能，即实现双向通信。

本文完成了测试部署系统到服务器，系统使用效果符合预期，数据能够稳定传输，传感网与系统使用者之间的双向通信稳定，且数据本地分析功能符合需求。

关键词：传感器网络；NIO；Netty；MySQL；JavaEE；网络数据采集汇聚

ABSTRACT

This research papers is from 973: "Empty Earth Integrated Sensor Network" submodules.

Exist in previous versions of the project using the Java server often appear in the communication packet loss problems and issues in data storage and analysis tools lack of post-acquisition, these problems in the actual research data were analyzed caused great problems. After viewing the log, we found that the performance is based on the traditional Java BIO (block-I/O) communication mechanism is poor, unable to face the high concurrency, cause the server to temporarily deny access. This is due to the traditional Java BIO mechanism is the use of synchronous stream I/O and multi-threaded concurrent network I/O handling. This model highlights the synchronous blocking I/O speed and CPU speed contradictory differences. In addition, although the use of multi-threading model thread pool to a reasonable allocation of resources, but the consumption of switching thread creation and context of the still very large, it is easy to reach the physical limits of the machine. For the above-mentioned problems of traditional Java server, we studied and implemented based Netty non-blocking I/O server with MySQL database, improving the concurrency server to solve the problem of multi-threaded data loss and data local persistent problem.

The purpose and significance of this paper is to solve the above two problems, namely to ensure that research projects related to data security and data processed normally. Firstly, improved server Socket communication based on the model introduced in JDK 1.4 appears NIO (non-block I/O). NIO is a type of non-blocking I/O communication mechanism for I/O models from Linux epoll model. This article is the use of a relatively mature industry Netty NIO-based framework, as the Socket server communication framework. Secondly, the choice of MySQL database as data persistence tools, as chosen JavaEE specification, use it to create a JSP and Servlet WEB application as a project management platform, and on the PC side write user-friendly visualization software to view the data. In addition, the sensor also

enables managers operating functions, namely, two-way communication.

This paper completed the deployment of the system to the test server, the system uses the effect as expected, data transmission can be stably stable two-way communication, sensor network system between users and data analysis capabilities in line with local needs.

Key words: Sensor Networks；NIO；Netty；MySQL；Java EE；Network Data Collection Aggregation

目　录

1 绪论

1.1 本文的选题背景.....1

1.2 国内外研究现状.....2

1.3 项目整体架构和实现概述.....3

1.4 本文结构安排.....6

2 基于 Netty 的服务端设计

2.1 传感网数据汇集服务端设计.....9

2.1.1 BIO 模型、NIO 模型和 AIO 模型简介.....9

2.1.2 基于 NIO 的 Netty 框架的介绍.....12

2.1.3 服务端设计思路和架构.....12

2.2 数据库搭建与表设计.....14

2.2.1 MySQL 数据库简介.....14

2.2.2 表设计与优化.....15

3 基于 Java EE 规范的 Web 应用设计

3.1 本文涉及的 Java EE 规范.....17

3.1.1 Servlet 技术.....18

3.1.2 JSP 技术.....18

3.1.3 JDBC 技术.....19

3.2 Web 应用设计.....20

3.2.1 Web 应用的业务逻辑设计.....20

3.2.2 Web 应用实现流程和细节.....21

3.2.3 Web 应用测试效果.....22

4 客户端可视化应用设计

4.1 客户端业务逻辑简介.....26

4.2 客户端实现过程.....28

4.2.1 基于 XML 的数据传输和解析.....28

4.2.2 基于 JFreeChart 的数据可视化 29
5 结论与展望
5.1 项目成果总结与贡献 31
5.2 展望 32
参考文献 33
致　谢 34

1 绪论

1.1 本文的选题背景

随着传感器和无线通信的快速发展，无线传感器网络的应用领域越来越广泛。无线传感网指由大量尺寸较小，通过廉价电池供电的低成本传感器节点组成。每个传感节点至少由四部分构成：传感器，MCU，电源和无线通信模块。传感器和MCU提供数据采集和数据本地处理功能。电源模块为其它三个模块提供所需能量。无线通信模块使传感节点通过自组网构成无线传感网，通过节点间相互协作完成特定任务[1]。

与传统网络相比，传感器网络在监测或数据收集应用中有很多优势。其中，易于部署是众多具有吸引力的优势之一。在传统网络中，传感节点被放置在特定位置，通过有线连接，传感节点的部署和配置极其繁琐。然而，无线传感网传感节点价格低廉且尺寸很小。而且，无线通信不需要固定的基础设施。这些特性使得即使在环境恶劣的野外当中，大量传感节点也可以很快得到部署。例如，传感节点可以通过飞机投放而部署到一些对人类来说危险的区域[2]。

无线传感网另一个吸引人的特性是它的自组织性。即替代了手动配置，传感节点能够自动地形成无线传感网络。该特性支持新增节点快速部署，新节点可以通过简单的路由配置方便地加入到已有无线传感网中。此外，该特性还使得传感网络能够适应设备故障和动态变化。

无线传感网的上述特性使得无线传感网有着广泛的应用。快速部署，自组织，故障容忍等特性将使得无线传感网在环境监测，目标追踪，地震探测，军事侦察，核反应控制，火灾追踪等应用中大展拳脚[3]。

本文项目是来源于973课题，为“空天地一体化传感网”的子模块。主要为解决在实际项目中布置的传感器网络的信息采集和展示问题。

在该课题中，笔者面对这样一个需求：在每个传感器检测项目中，分布有大量传感器节点，这些节点定时定量检测这些项目诸如水库或是铁路的状况，保证质量控制人员和科学家可以实时分析自然灾害亦或是工程出现的问题，及时修复。而由于这些项目的地点可能是位于野外，所以不可能让研究人员实时当场监。针对这样的需求，就需要一种系统对这些传感器检测到的数据进行实时监控。而

本文的项目就是要实现这些传感器网络的数据汇集，在传感网并发发送信息的情况下，系统能及时地稳定地收集传感器网络的检测数据，并能及时反馈给研究人员，并对数据做一定的保存和处理。而对于研究人员，可以根据实际需要来和传感网的某一传感器进行通信，设置其采集选项，实现传感网和人的双向通信。

1.2 国内外研究现状

本题所面对的主要问题是工程界经常会遇见的问题:在超大数据的传感器网络下，有限的硬件资源无法承载高并发的数据汇聚造成的数据丢包。这时候,使用算法和高效的I/O机制以充分利用机器的物理性能就显得很重要了。

在网络通信的服务器编写上，一直以来都是C++占据着主导地位,其凭借着高性能深受追捧。而Java由于在编译过程中需要翻译为字节码再通过虚拟机和Linux交互，其从一开始就被打上了性能的短板。但是，这个情况在JDK1.4后发生了改变。传统的基于Socket的网络编程通信方式BIO是同步且阻塞的，其使用线程池维护着处于accept状态的Socket线程，这种方式在长连接众多的情况下将造成大量的不必要线程资源开销。而JDK1.4推出的NIO技术的基于epoll模型的非阻塞I/O特性能提高网络服务器的通信性能和文件I/O能力。目前基于Java NIO的开源框架有 Netty、MINA、XSocket、Grizzly等等[5]。这些框架对NIO的封装并结合线程池模型来满足服务器对高并发的需求。

Netty是由JBOSS开发的一个基于Java NIO的开源通信框架，用于处理和管理Socket。Netty提供有异步的、事件驱动的网络应用程序框架和工具，它提供了对TCP、UDP和文件传输的支持，作为一个异步NIO框架，Netty的所有I/O操作都是异步非阻塞的，用户可以方便的主动获取或者通过通知机制获得I/O操作结果。用其可以快速开发高性能、高可靠性的网络服务器和客户端程序。

Apache MINA(Multi-purpose Infrastructure for Network Applications) 是一个基于Java NIO的网络通信框架，它隶属于Apache软件基金会。MINA可以帮助我们以一种简单高效的方式的开发出高性能的网络通信程序。它对Java中的Socket和NIO进行了有效和明确的封装，从而使开发面向Socket编程的程序员的只需要关注业务逻辑而不用在意底层，简化了程序员的工作，大大减少了使用原始的Socket时可能遇到的各种繁杂的线程创建、会话维护、性能调优等问题。目前发行的MINA 2.0.x已经支持有基于TCP/UDP协议的网络服务器开发。

在互联网行业中，互联网架构的推进历史已经经历了从ORM(Object Relational Mapping)到MVC(Model-View-Controller)到RPC(Remote Process Call)，虽然RPC支持的分布式系统在现在的架构中占据主流地位，但随着互联网应用规模的增大，这些传统的垂直应用架构面临着前后台耦合、集群通信效率低下、管理困难的困境。

在国内,阿里巴巴研发开源的关系型数据的分布式处理系统Cobar和分布式服务框架Dubbo底层就是基于NIO实现的，用于淘宝天猫等电商平台每日数十亿级的并发。并且Dubbo已经被京东、去哪儿等公司使用。其中Dubbo是基于Netty加二进制编解码框架的内部私有协议，其代替原来的RMI、HTTP+XML等协议，提升节点之间的通信性能[5]。

由上可见,在对于性能有高要求的情况下,传统的Java BIO已经渐渐被抛弃,在面对高并发时,越来越多业界公司开始使用NIO作为通信底层。而对于一些中小型项目,远远用不上Dubbo这样的分布式框架。这也是我在本题中选择Netty的原因，作为一个轻量级的开源框架，Netty和本项目十分契合。

1.3 项目整体架构和实现概述

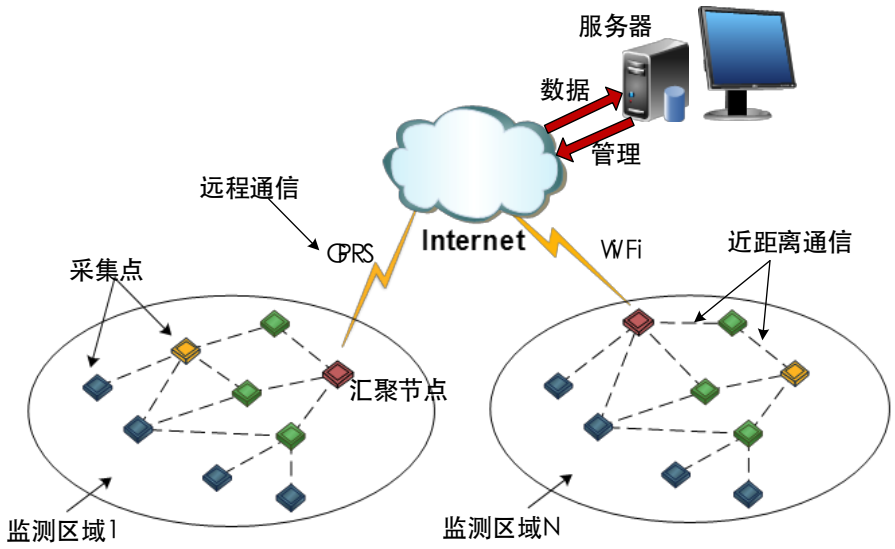

图**1.1** 常见无线传感器网络系统基本架构

图1.1展示了常见的无限传感网的组织结构，本项目负责的是解决数据传输到Internet及其后续的处理。为了解决本项目实际中遇到的问题，本文对项目整体架构做了一个分层处理。项目架构图如图1.2。

在本项目的架构中，按照功能分为了四层，其由低到高分别是：

（1）数据源层。这层抽象了传感网，将其看作一块数据源，后续的层只要将传感网作为一个接口进行交互，而不需要关心传感网内部的拓扑结构。

（2）数据处理层。这层的功能是直接与数据源、数据库交互，对数据进行首次封装。在这层将完成繁杂的数据解析、通信协议判断、数据处理、数据库操作。此外，该层相对于上层就像是封装过的数据源，上次通过该层的间接接口数据库即可获取经过处理的数据。

（3）数据应用展示层。这层的功能是提供人机交互接口，将经过下层封装的数据根据实际的业务逻辑和需求展示给用户、提供给用户分析使用。该层相当于项目整体的GUI(Graphical User Interface)即图形用户接口。

（4）用户层。即可以理解为逻辑层，该层为设计者抽象了用户的实际行为，为下层提供业务逻辑。

基于以上的分层设计，本项目需要设计以下功能模块：

首先，是一个用于接收传感器网络数据并能和传感器通信的服务端其在架构中相当于图1.2的Netty Server。对服务端的要求是能够稳定地接收数据，即保证数据的完整性和准确性，与此同时也可以准确地、及时地发送数据到传感网。为此，本文设计了基于Socket的面向连接协议TCP/IP的服务端用于与传感器网络通信来保证传输层的稳定。对于服务端的设计语言，本项目使用Java而非C++，这是考虑到两方面：一是C++在网络编程方面十分繁琐；二是在JDK1.4，Sun公司提出的NIO机制大大提高了Java网络通信的性能，加上Java本身对开发者的友好，所以选用Java开发服务端程序。

其次，与服务端配套，需要一个数据库作为服务端的数据持久化载体，其在架构中相对于图1.2中的MySQL。本项目选用了在工业界较为流行的MySQL数据库，这是一个轻量级的开源数据库，就并发能力和存储能力而言，已经可以满足本项目的需求。本文根据实际传感网项目需求，还需为项目合理设计表格，建数据库。

再次，在数据应用展示层还需要为工作人员提供GUI。为了更好地方便用户，本项目设计了两个GUI模块，一是基于Java EE规范的Web应用即图1.2中的Web Application，使用它是由于Web应用轻量级且可跨平台，Web端也有许多公有API可供使用，如本文使用的某公司的地图功能可以实现传感网节点在地图上标示，这些都大大方便了工作人员的使用。二是基于Java Swing的PC端GUI程序即图1.2

中的Client Application，这里将对数据处理和与传感网通信的功能抽离到本地来处理，一方面是减轻Java Web服务器的数据处理压力，另一方面是将双向通信的权限限制到个人电脑，防止有人恶意攻击Web应用而造成传感器工作异常。

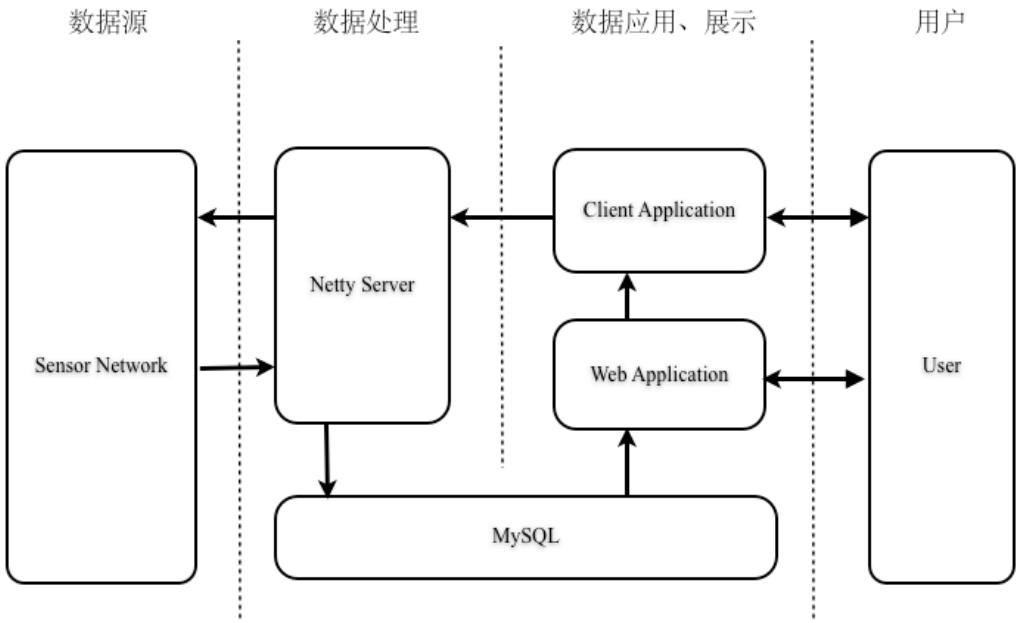

图**1.2** 项目架构图

图1.2描述了项目架构分层实现。

项目整体业务逻辑如图1.3所示，这是整个项目的工作流图。整体分为两个部分。

图1.3中标示的A工作流为服务端接收数据和处理数据部分。传感器网络在平常的工作中收集数据，到一定时刻，则通过Internet与Netty服务端建立长连接，Netty服务端负责接收传感器数据，做数据校验和数据解析。

这里将传感网发送的数据按时效性分为两类：一类是以文件为载体，例如以txt文件格式传输一段时间的记录数据集，由于不是实时传输记录，所以称为非实时的传输方式。另一类是实时传输方式，该工作方式下的传感网每次采集一条数据就会立即发送给服务端，服务端实时接收。

如果数据以非实时的传输方式传输，根据协定的通信协议，服务端将解析文件数据后将打包数据成XML文件存储在服务器上，而如果数据校验后为一条记录，则为实时传输方式，将符合通信协议的该条有效数据插入MySQL数据库中。

图1.3中标示的B工作流则是处理用户与系统交互部分，由于本项目主要目的之一是方便用户去查看、研究、使用传感网的数据，所以这部分功能的业务逻辑

较繁重。首先，用户可以通过浏览器打开Web应用，查看传感网数据变化，如果需要更加直观地分析这些数据，用户有两种方式来获取数据，一是获取传感网非实时文件数据；二是获取MySQL数据库中存放的实时方式采集的数据，在该方式下，用户需要打开客户端，通过在Web上提交表单，Web应用会在检测本地客户端打开之后，将数据打包为XML文件发送到本地。当本地客户端收到数据后，可以解析并在客户端画出对应的折线图或是曲线图。此外，用户还可以通过客户端向其查看的传感器发送指令，该信息同时也会被存储到数据库中，方便管理员审核。

表1.1列举了本项目使用的整体技术特性与创新。

1.4 本文结构安排

本文第1章为绪论，叙述项目背景、实施方案、介绍基于Java NIO服务器的国内外发展和系统总体概述。

第2章介绍了基于Java NIO框架Netty的服务端设计和数据库部署与设计。

第3章介绍GUI模块第一部分即基于Java EE的Web应用设计。

第4章介绍了GUI模块的PC端应用设计。

第5章为本文总结与展望。

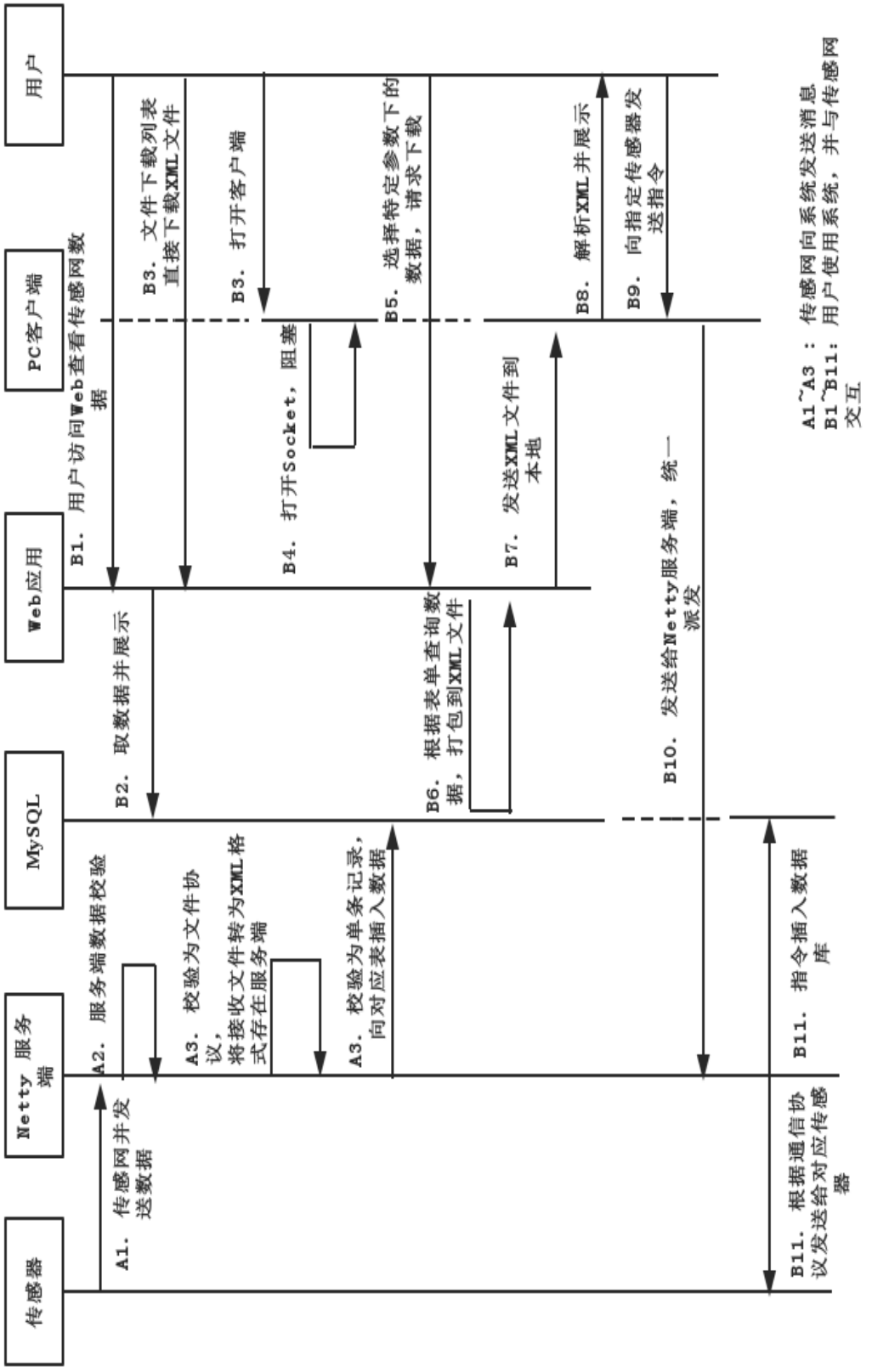

图**1.3** 项目业务逻辑工作流

表 1.1 项目整体技术简述表

功能名称	传统实现	本文实现	比较
传输层协议	UDP 或 TCP	TCP	TCP 较 UDP 传输更加稳定。
服务端通信模型	阻塞式 I/O，Java BIO	非阻塞式 I/O，Java NIO	后者减少了保持长连接带来的资源占用。
服务端通信框架	基于 BIO 封装的线程池模式	基于 Netty 框架的 I/O 多路复用模型	后者使用简单，有开源支持，效率高。
数据存储	Oracle 或 SQL Server 2000	MySQL	Oracle 费用昂贵，SQL Server 2000 闭源商业化，而 MySQL 免费且开源可定制。
Web 应用服务器	Tomcat	Tomcat	
Web 应用网页与逻辑	HTML + CSS + JavaScript	JSP + Servlet + 少量 JavaScript	前者组合轻量级但其是静态网页，后者是符合 Java EE 企业级应用规范的，且是动态网页。
客户端设计	C# 、Objective-C 、Java	Java	前者分别对应软件设计在 Windows 与 Max OS 下的语言基础，而本项目使用 Java，由于基于 Java 的软件是跨平台的，一次书写，随处可用。
客户端与服务器数据交互	XML、Json、字符串	XML	XML 是一种高效的数据协议，且 Java 目前有比较完备的库，其在数据格式上优于字符串，在数据存储上优于 Json。
客户端可视化展示	调用系统画图函数	JFreeChart	后者使用简单，功能完备，节省开发时间。

2 基于 Netty 的服务端设计

2.1 传感网数据汇集服务端设计

2.1.1 BIO 模型、NIO 模型和 AIO 模型简介

网络编程的基本模型是Client/Server模型，也就是两个进程间通信，其中服务端提供位置信息即IP地址和端口，客户端通过连接操作向服务端监听的地址发起连接请求，通过三次握手建立连接，如果连接成功，双方就可以通过Socket进行通信。

在基于传统的同步阻塞模型的开发中，如图2.1，是采用BIO(block-I/O)通信模型的服务端。其通常由一个独立的Acceptor线程负责监听客户端的连接，当它接收到客户端连接请求后，将为每个请求连接的客户端创建一个新的线程进行对应的连接处理，连接结束后，通过四次握手与客户端断开连接，当对应连接的Socket资源释放后，该对应线程销毁。这就是阻塞式通信模型[6]。该模型问题在于线程在连接过程中的阻塞造成内存资源调度不合理，线程资源分配缺乏弹性伸缩能力，一旦客户端并发访问量增加，服务端的线程个数和客户端并发访问数呈线性的正相关，由于线程在JVM(Java虚拟机)中的开销极大，线程不断地创建，系统的内存将急剧下降，并且每个线程得到的CPU时间片也将更少，CPU效率大大降低。一旦并发数超过一个阈值，系统将可能发生因线程过多的堆栈溢出或内存不足，这时服务器进程阻塞或者被操作系统清理，其将不能正常被外界访问。

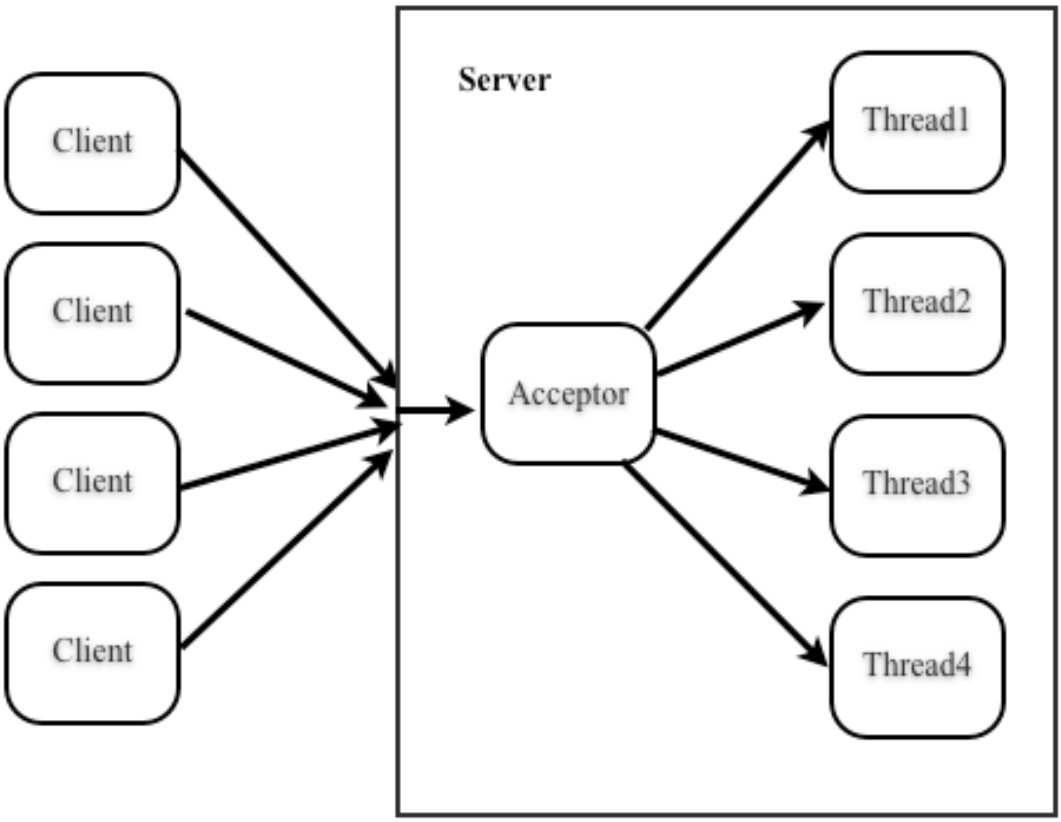

图**2.1** BIO通信模型

以上，认识了传统BIO在网络通信中的性能瓶颈，针对这个问题，Java NIO克服了阻塞式I/O的缺陷的。NIO全称 New I/O,业界又称Non-block I/O即非阻塞式I/O[7]。NIO工作原理图见图2.2，其中Channel是一个通道，可以通过它读取和写入数据，其于Java传统的流不同，它是全双工的，所以它可以更好地映射底层操作系统API，特别是在UNIX网络编程模型中，底层操作系统的通道都是全双工的。而Selector是一个多路复用器，提供选择已经就绪的任务的能力。简单来讲，Selector会不断轮训注册在其上的Channel，如果某个Channel上有新的TCP连接接入、读和写事件，这个Channel就处于就绪状态，会被Selector轮询出来，进行后续I/O。这样，与BIO不同，BIO需要为每个长连接分配线程，而NIO相当于使用了一个线程去管理多个Socket连接，当有正式数据准备写入时再接入对应的连接。NIO的这种机制对应的正是UNIX网络模型中的I/O多路复用模型，在UNIX系统中，这种轮询机制是在内核进行的，效率比用户态的线程要高很多。

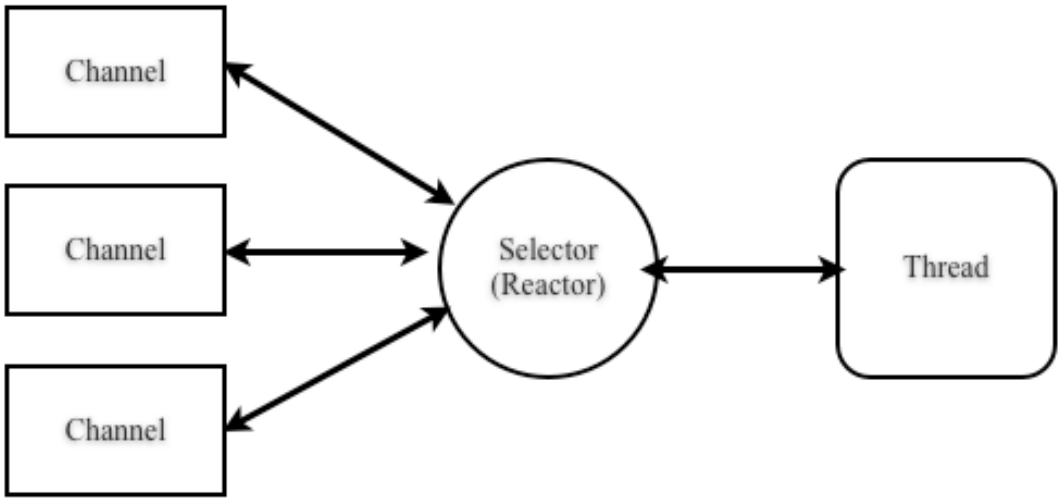

图**2.2** NIO通信模型

JDK7推出了AIO(Asynchronous I/O)，即异步I/O[8]。其异步主要是针对进程在调用I/O获取外部数据时，是否阻塞调用进程而言的。AIO原理图见图2.3，每当一个客户端连接到服务器，主线程将预注册一个回调(Callback)在子线程，然后让设备进行I/O，主线程继续等待连接，一旦对应I/O结束，主线程自动调用对应子线程的回调，这一过程，没有任何阻塞和同步，故称为异步非阻塞I/O。

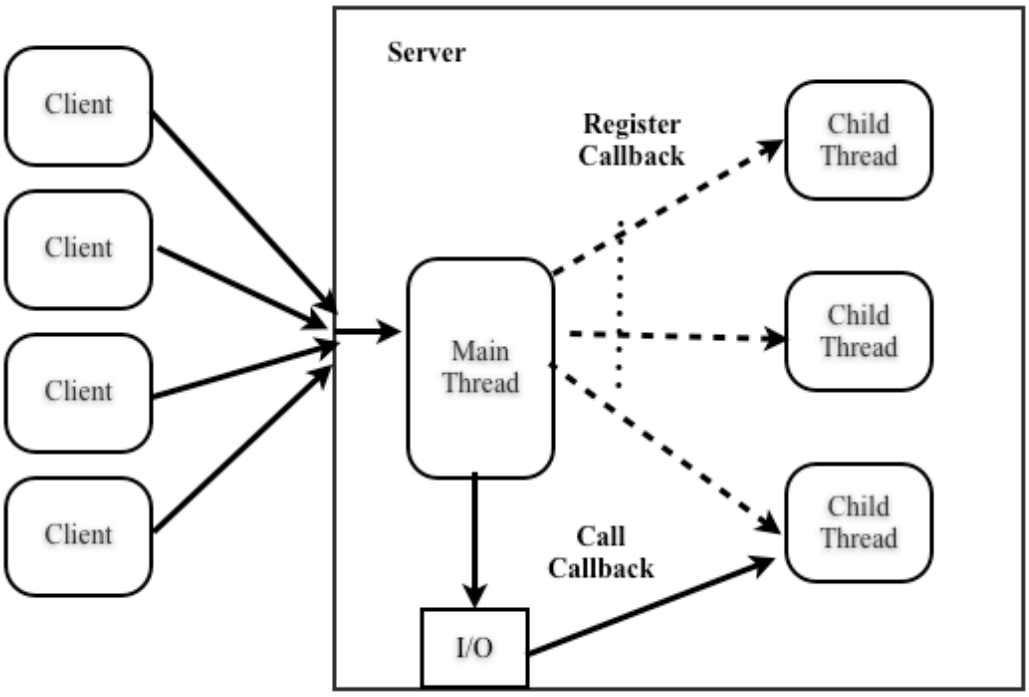

图**2.3** AIO通信模型

以上，本文对比了Java中的I/O模型，而本文选用的是NIO。这是因为：

（1）在处理网络长连接的性能上，从上述模型叙述可知，NIO、AIO的CPU利用率和内存利用率明显高于BIO。

（2）NIO相对于AIO来说有更加成熟的框架和稳定的社区支持。

（3）NIO模型适用场景是大批量轻量级连接，例如作为聊天服务端。而本文项目是用于接收传感网信息，而这类信息一般报文长度短，且连接时间不需要很长，连接频繁，符合NIO的使用场景。

（4）AIO模型适用场景是大批量重量级连接，例如大文件传输这种需要较多I/O资源的连接类型。

综上，本项目的服务端通信模型选择NIO。

2.1.2 基于 NIO 的 Netty 框架的介绍

其实在使用Netty之前，笔者设计这个服务器是准备直接使用Java NIO原生API进行开发的。但当笔者的一个Demo进行调试后，我发现这是群很复杂的API，而且由于这是底层API，对于网络通信的一些诸如网络闪断、连接重入、安全认证、编码解码、粘包拆包等问题的解决需要开发者自己进行封装，这样，就大大增加了开发成本。于是我尝试去找寻业界的成熟解决方案，最后我找到了Netty。

Netty是目前工程界最流行的Java NIO框架之一，它具有健壮性、多功能性、高性能、可定制性和可扩展性等特点。它已经在许多商用项目和开源项目中得到使用，例如分布式数据框架Hadoop的RPC(远端过程调用)框架的avro组件的底层通信就是基于Netty；此外，由于Netty优秀的异步通信能力，其他很多业界主流的RPC框架，也开始使用Netty来构建底层的高性能通信模块。

Netty总结来说具有五个优点有：

1. API使用简单，对开发者较友好开发门；
2. 功能多且强大，例如预置了多种编解码功能，支持多个主流通信协议；
3. 在业界的主流NIO框架中，性能综合最优；
4. 成熟、稳定，Netty修复了已经发现的所以JDK NIO BUG；
5. 社区活跃，版本更新后的BUG能快速得到反馈等。

2.1.3 服务端设计思路和架构

服务端负责与传感器网的通信和存储接收的数据和发送的指令到数据库。其工作流和架构见图2.4。

在图2.4中，首先的场景是传感器网络向服务端发送数据，这里在传输层使用面向连接的TCP协议保证了数据在传输过程中的稳定性[9-10]。此外，介于传输层与应用层，使用Socket作为抽象的流处理通道，将数据准确送达服务端对应的主机的端口上[15-17]。服务端的主机的通信接收端是使用Netty编写的通信程序，图中略去了Netty的通道(Channel)标示，在实际中，每一个长连接通过通道在系统的选择器(Selector)注册有唯一标识。当传感器数据真正到达后，选择器马上通知对应的数据处理模块(Handler)派发一个线程进行处理，而处理模块中封装了对于接收的数据的后续处理过程。

接着，处理模块对数据进行校验，这里基于了一个私有通信协议，通过对报文头的token校验来判断数据类型和合法性，这样可以防止有人恶意连接服务器，发送垃圾数据造成数据不实。

这里将传感网发送的数据按时效性分为两种方式：一类是以文件为载体，例如以txt文件格式传输一段时间的记录数据集，由于不是实时传输记录，所以称为非实时的传输方式。另一类是实时传输方式，该工作方式下的传感网每次采集一条数据就会立即发送给服务端，服务端实时接收。

如果数据以非实时的传输方式传输，根据协定的通信协议，服务端将解析文件数据后将打包数据成XML文件存储在服务器上，而如果数据校验后为一条记录，则为实时传输方式，将符合通信协议的该条有效数据插入MySQL数据库中。

当数据通过校验后且认定为实时传输方式，处理模块将调用负责服务的接口(Service)，在服务接口中已经定义了数据解析和处理的方法，这个接口会接着去调用接口DAO(Data Access Object)中的数据操作方法，在DAO模块，使用Java数据库交互规范JDBC与数据库交互。

这里调用多层接口而不是在处理模块中直接处理数据是考虑到整个项目的扩展性，这是一种面向接口编程的设计思想，若项目后续有功能扩展，只要在对应的数据模块进行修改，减少了模块间的耦合性。

若是接收的数据无法通过校验或是数据的格式有问题，都将在服务接口或DAO对应的Implement模块中进行异常处理，异常处理(Exception Handler)模块会适当抛出异常通知使用者。

此外，在处理模块添加了基于Netty支持的对于TCP粘包和拆包的解决，防止在一次长连接中大量数据的持续传输造成的包错乱问题。

接着，同理，如果是管理人员需要对传感器网络某一节点进行控制，则可以在客户端上直接传输指令到服务端，由其校验后向指定传感器网的IP地址指定端口发送指令信息，至于指令的解析有传感网端根据商议的协议进行解析。

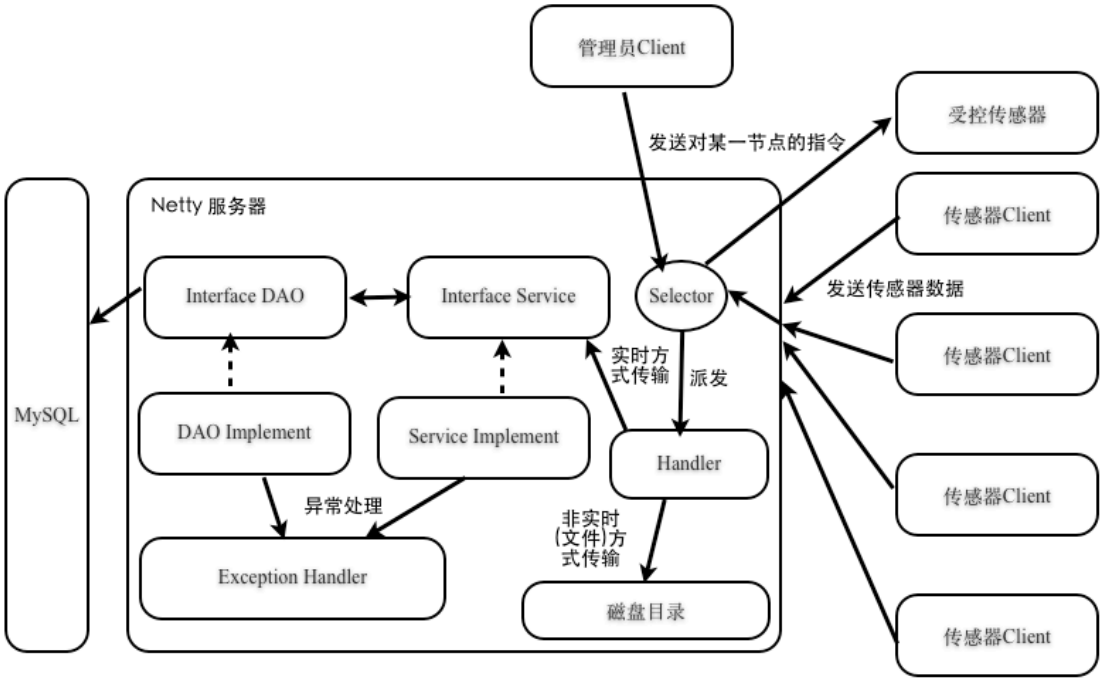

图**2.4** 服务端设计架构图

2.2 数据库搭建与表设计

2.2.1 MySQL 数据库简介

MySQL是一种开源的关系型数据库管理系统，MySQL数据库使用结构化查询语言(SQL)进行数据库管理。由于MySQL是开源的，因此任何人都可以下载其源码并根据实际需求对其进行个性化定制。

MySQL因为其事务速度、可靠性和适应性而受到开发者的青睐。业界普遍认为在不需要事务化处理的需求下，MySQL是数据持久化最好的。所谓事务化，指的是将几个数据库操作封装为一个事务，进行原子性操作。

在本项目中，MySQL是作为一种数据本地持久化的工具和项目管理工具，即作为各个传感网项目的数据表设计载体、数据存储区、WEB应用和客户端的数

据来源。正如上面所说，MySQL作为一款轻量级数据库，其配置简单、免费等特性十分契合本项目需求。

2.2.2 表设计与优化

数据库的设计过程简单概括为：从业务模型入手，先将各种实例进行抽象，再到数据模型即建立对应的表描述这些抽象。所以，设计的核心是表的设计。

在设计表之前，需要先来了解数据库设计的三大范式[11]。

第一范式：要求表的每个字段是不可分割的独立单元。从存储上看，如果字段多重语意，虽然节省字段资源，但是存储时候的格式没有一个统一规范，且语意不明。在查询时，SQL语句书写困难，且查询出的数据有冗余，还需要二次处理，整体上影响表的使用效率。

第二范式：在第一范式的基础上，要求每张表只表达一个意思。表的每个字段都和表的主键有依赖。这个范式是从抽象合理的角度来规定的，如果一个表表达多重意义，虽然可以正常查询，但是其实际意义是很难抽象出的，且多重意义的表其必有字段和主键没有依赖关系，但需要使用这些字段时，与主键的无关联性容易造成数据的不唯一性。

第三范式：在第二范式基础上，要求每张表的主键之外的其他字段都只能和主键有直接决定依赖关系。要求表的字段与主键有直接依赖关系是考虑到了数据冗余的问题，如果没有直接依赖关系而仍然存在此表中，由于其他某一表的主键与该字段有直接依赖关系，那么此表的该字段数据就是重复存储，造成了数据冗余，增加了数据库存储负担。

接着，在设计项目的表时，还需要了解下业务逻辑，让需求驱动设计。

本项目中，我根据“空天地一体化传感网”项目的实际需求，模拟设计了一个水库传感器网络项目，即在一个水库中部署有传感器网络，针对这个网络的数据的收集与存储，据此本章来展开数据库设计的过程：

1. 项目表(图2.5中的project_list)：业务上简单来说，水库项目只是传感器网络项目的一个部分，所以首先需要的是项目层次上的记录表，用于项目的详细记录，根据三大范式，项目记录表是用于记录项目的基本信息的。

2. 项目节点表(图2.5中的node_list):对于每个项目传感网，都有大量节点，所这里需要第二个表来记录项目节点层次，用于记录该项目对应的所以节点的基本信息。

3. 节点记录表(图2.5中的node_detail_list)：对于每个节点，只要在工作状态，都会记录大量数据，所这里需要第三个表来记录节点记录层次上的数据，即传感器节点时间序列上的测量值。

4. 用户表(图2.5中的user)：除了上述的涉及业务的表以外，还需要为WEB应用的登陆设计一个用户表用于权限设置。

5. 指令表(图2.5中的node_command)：用于用户和传感网通信的每条指令的记录，方便管理员查看当前传感网节点状态。

由于以上五个表都是基于三大范式设计，极大地减少了数据冗余，接着，笔者使用了MySQL的外键约束，就可以很好地实现联合查询[12]。

数据库表关系见图2.5。

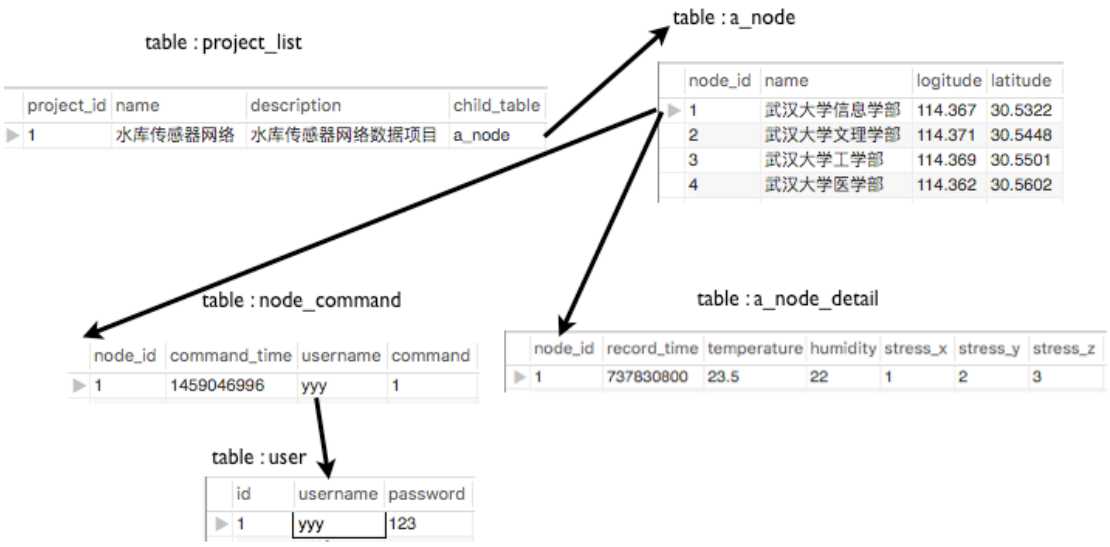

图**2.5** 水库项目表设计图

3 基于 Java EE 规范的 Web 应用设计

3.1 本文涉及的 Java EE 规范

JavaEE又称J2EE，是一种基于Java 2平台的开发、部署和管理相关企业的Web的体系结构[12]。J2EE技术的基于Java 2平台的标准版，其不仅具有Java的“编写一次、随处运行”的特性、方便存取数据库的JDBC规范、CORBA技术以及能够在Internet应用中保护数据的安全模式等等，同时还提供了对 EJB（Enterprise JavaBeans）、Java Servlets API、JSP（Java Server Pages）以及XML技术的全面支持。其最终目的就是成为一个能够使企业开发者大幅缩短投放市场时间的体系结构。

图3.1描述了传统J2EE服务器架构体系，J2EE使用多层的分布式应用模型，应用逻辑按功能划分为组件，各个应用组件根据他们所在的层次分布在不同的设备上。Sun公司设计J2EE的初衷是为了解决C/S模式(Client/Server)的弊端[13]。在传统C/S模式中，客户端集成了过多的功能而显得重量级，在这种模式中，第一次部署较容易，但对客户端难于版本控制和改进。它使得重用客户端的业务逻辑和界面逻辑非常困难。而J2EE的多层企业级应用模型将两层化模型中的客户端分成许多层。这是一种服务拆分的思想，一个多层化应用能够为不同层次提供不同的服务，本章提及的Web应用主要使用了J2EE服务器上的Web容器中的Servlet作为业务逻辑层、使用JSP作为Web前端层、使用了EJB容器作为数据层并搭配JDBC作为统一接口与数据库交互。而这整个服务器均是搭建在符合J2EE规范的开源服务器Tomcat上。

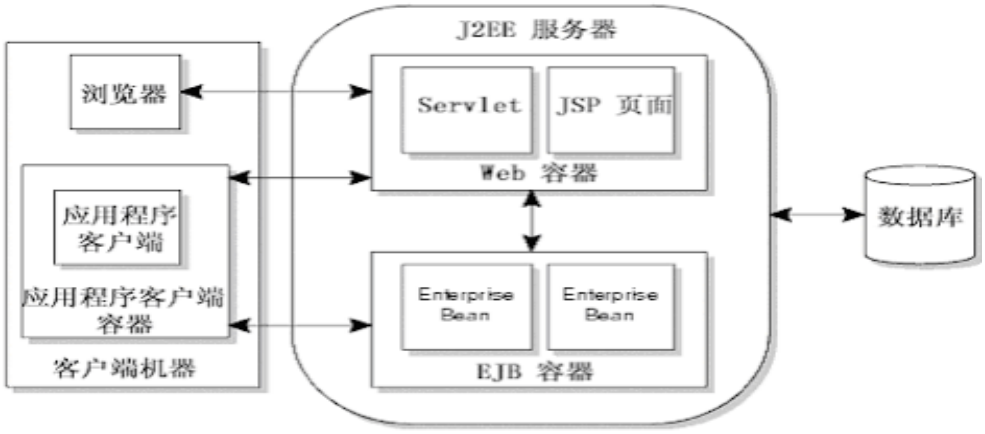

图**3.1** J2EE服务器开发架构

3.1.1 Servlet 技术

Java Servlet运行在Web服务器上，它处于HTTP请求端和服务器数据访问层(DAO)之间的中间层。使用Servlet提供了接收用户请求输入的统一接口，并可以据此来动态产生HTML即动态创建网页。Servlet有以下几点优势：

1. Servlet在Web服务器的地址空间内执行。即它不是一个单独的进程，而是依托于Web容器运行，具有可移植性和可拓展性。
2. Serlet用Java编写，Java类库的全部功能对Servlet来说都是可用的。它可以通过Socket和RMI机制与Applets、数据库或其他软件进行交互。它可以在JVM上跨平台运行

Servlet作为Web应用的中间层，其主要功能有：

1. 读取HTTP请求参数。并根据参数做业务逻辑的处理。
2. 读取隐式HTTP请求数据。诸如cookies、流数据和浏览器支持的压缩格式等等。
3. 发送显式的HTTP响应到客户端。
4. 发送隐式的HTTP响应到客户端。这包括响应文件类型、cookies和缓存参数，以及其他类似的任务等。

3.1.2 JSP 技术

JSP(Java Server Pages)是一种动态网页开发技术。它能够使Java代码运行在HTML代码中。JSP基于Servlet，主要用于实现Java Web应用程序的用户界面部分。

JSP通过HTML上表单提交获取用户输入数据、访问数据库及其他数据源，然后动态地创建网页。JSP的这些功能基于JSP标签，JSP标签能够让Java代码在HTML上访问数据库、数据集处理、记录用户选择等，还支持不同的网页中传递控制信息和共享信息。

JSP的优势可以总结如下：

1. 与静态HTML相比：HTML不包含动态信息。
2. 与纯Servlet相比：JSP专注于页面开发，而在Servlet中写HTML代码是特别繁琐的事。
3. 与.NET ASP相比：JSP由于是用Java书写，具有很好的可移植性。
4. 与SSI相比：SSI无法使用表单数据、无法进行数据库链接。

5. 与JavaScript相比：虽然JavaScript可以在浏览器动态生成HTML，只能发挥前端的功能，因此很难与服务器交互，比如访问数据库和图像处理等等。

3.1.3 JDBC 技术

JDBC(Java Database Connectivity)是Sun公司提出的一种数据库访问规范。为访问不同的数据库提供了一种统一的接口而屏蔽了一些底层细节，具有跨平台和跨数据库的使用特性。

图3.2描述了Application是如何调用JDBC与数据库交互的过程，应用直接调用JDBC接口提供的API，JDBC底层由一个统一的Manager控制数据库驱动，而数据库具体的交互实现根据各个公司推出的数据库不同，设计自己数据库驱动，而这些各异的数据库公司都遵守着JDBC的接口规范，所以调用者就不需要考虑繁琐的数据库语句而是可以调用简单的API进行业务逻辑开发。

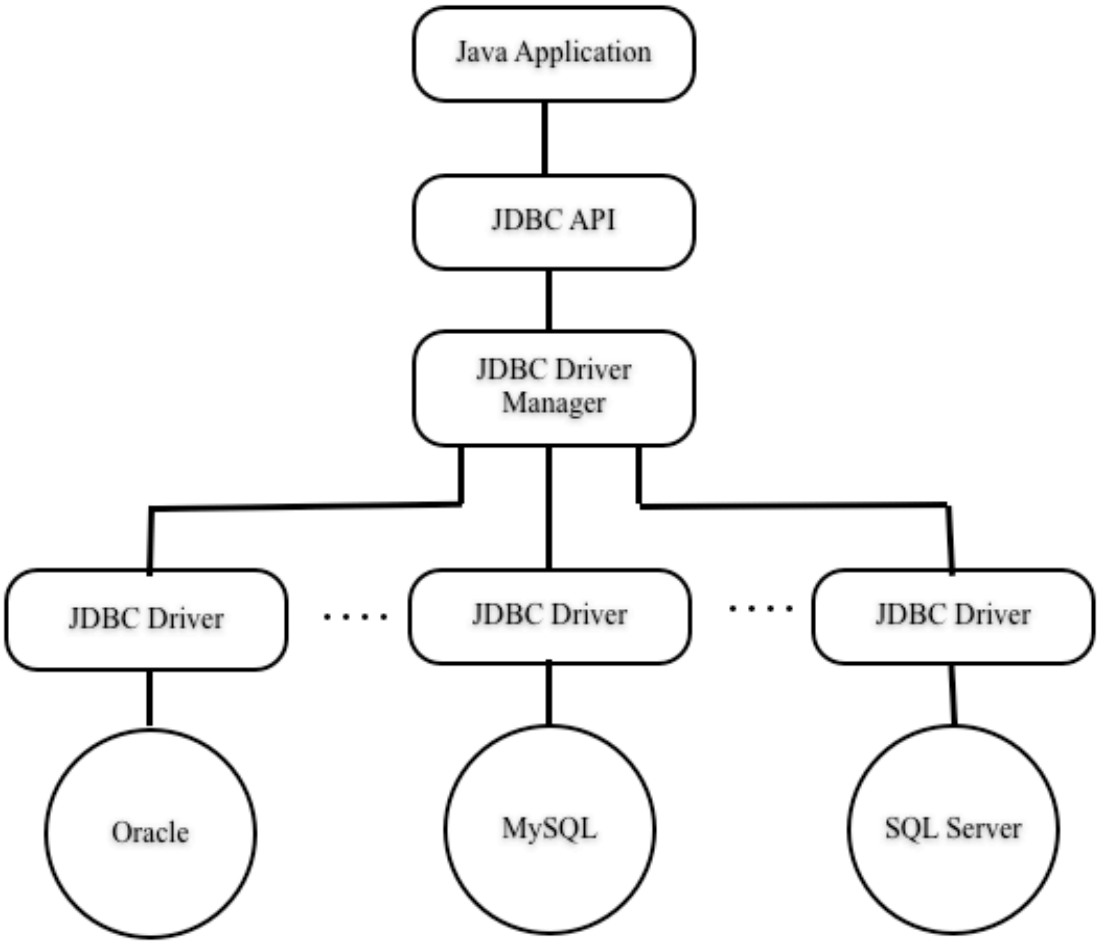

图**3.2** JDBC原理图

3.2 Web 应用设计

3.2.1 Web 应用的业务逻辑设计

本项目的Web应用的业务逻辑大体如图3.3，可以分为以下几个部分：

首先展现给用户的是登陆界面，设计登陆界面作为识别用户的入口，同时还需要提供了一个注册入口，方便后续添加用户。

当用户登陆成功后，展现的是传感器网络项目列表，该列表内容与图2.4中的project_list相对应，该界面展示了所有部署有传感器网的项目以及项目的基本信息介绍，供用户使用。

当用户选择一个项目点击查看后会展示这个项目的传感器节点列表，这个列表的结果与图2.4中的a_node类似，会显示有该项目每个传感器的坐标以及简述，方便用户进一步选择需要的节点查看。此外，为了让用户可以更加直观地选择自己所需要的节点，这里添加了一个界面，显示地图模式下的节点位置，让用户可以根据节点在地图上的标识选择节点查看。

当用户选择一个节点进行查看，进入一个展示该节点的记录界面，该界面列表展示元素内容类似于图2.4中的a_node_detail，在这里，用户可以查看节点记录的所有状态，此外，如果用户需要特定时间和特定参数的数据进行进一步数据分析，在这个界面有一个表单供用户选择，并且可以下载数据到本项目指定的客户端进行查看。

如果用户需要下载非实时的数据，则可以在项目节点记录详情页面进入文件下载列表，选择对应节点对应时间段的数据文件，直接下载到本地，使用客户端查看。

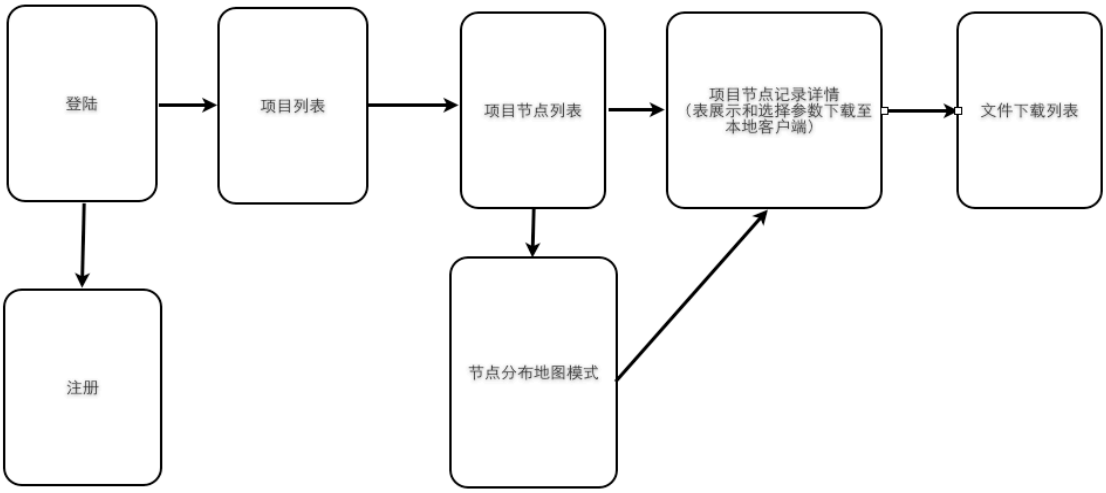

图 **3.3** Web 业务逻辑图

3.2.2 Web 应用实现流程和细节

以上简单阐述了Web应用的业务逻辑，接下来就是根据业务逻辑进行实际开发了。整个Web应用的架构图见图3.4。这里依据了MVC(Model-View-Controller)设计模式，将应用分为三层，其中JSP与Filter作为View层，负责为用户展示网页内容和数据的编解码、登陆校验。Servlet作为Controller层，即控制器层，处理应用内部的业务逻辑和操作数据结构。其中图中未标示的JavaBean作为Model层，负责抽象各种数据。

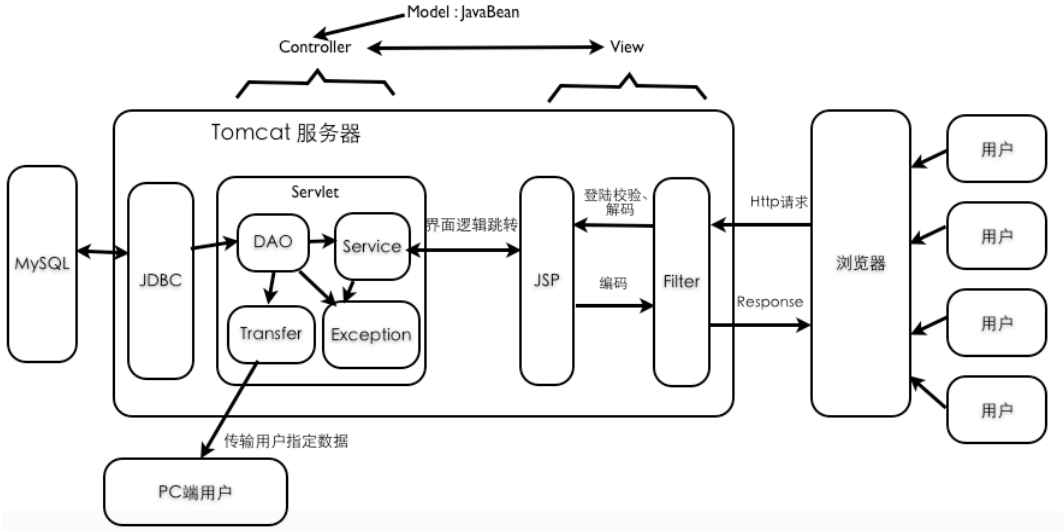

图 **3.4** Web 应用架构示意图

先来看看View层的实现。

View作为与用户直接交互的模块，其有两个功能：监听用户的操作以及展示数据。在本应用中，这块功能主要由JSP完成，上文介绍过JSP是一种动态网页技

术，既能通过JSP标签生成HTML标签，又能实时与服务器进行交互。这里将JSP搭载在符合J2EE标准的Tomcat服务器容器上，实现了动态网页展示。Tomcat服务器是Apache软件基金会的一个免费的开源Web应用服务器。其提供了轻量级HTTP应用服务，故在并发访问用户不是很多的场合下被普遍使用，其符合Java EE规范，是开发和调试Java Web应用的首选。此外，为了实现上文提到的地图标示节点坐标功能，在部分JSP中内嵌了JavaScript代码，调用了某公司的地图API在JSP中嵌入了地图。

除了JSP，View还负责对数据的编解码问题和登陆校验问题。这里使用了J2EE的Filter，顾名思义，这是一种筛选器。当外部Http请求到达是，根据是否在服务端维护着对应的登陆Session(会话)，判断是否进行跳转。若没有检测到对应的Session，则不论请求的URL，都转发到登陆界面的JSP处理。此外，由于浏览器的编码和Web应用编码不同可能造成乱码问题，使用Filter可以对接收数据、返回数据和Servlet中转发的数据进行统一编码，本例使用统一的UTF-8编码。

再来看看Controller层的实现。

Controller层主要作为应用的业务逻辑处理模块，处理JSP传递的用户交互行为，并根据这些交互行为通过Service、DAO去执行相应数据操作和界面跳转。这块的模块分工与上文提到过的Netty服务端是类似的，所不同的是，Netty服务端是没有前端的，而Web应用将JSP作为前端。

除了Service接口封装了功能方法、DAO接口封装了数据处理方法、Exception的异常处理以外，这里还添加了一个Transfer模块，这是一个用于与PC客户端程序通信的模块，通过这个模块，可以将用户在网页表单中选择的特定参数的数据从数据库取出并通过一定协议下载到本地客户端进行显示和分析。

本项目使用XML作为数据传输的格式，当用户在Web上选择其需要的数据后，在确认用户本地的接收客户端已经打开的情况下，Web应用会根据选择的参数去数据库取出相应数据，打包到一个XML文件中，并和本地客户端建立长连接，开始发送文件数据。

3.2.3 Web 应用测试效果

以下图3.5-图3.10为Web应用测试效果图。

图3.5为登陆界面，当用户在浏览器上输入本项目任意URL时，Servlet会根据Session来判断用户是否有权限进入，如果没有对应的Session数据，将会直接跳转到这个界面，强迫用户登陆后才能操作。

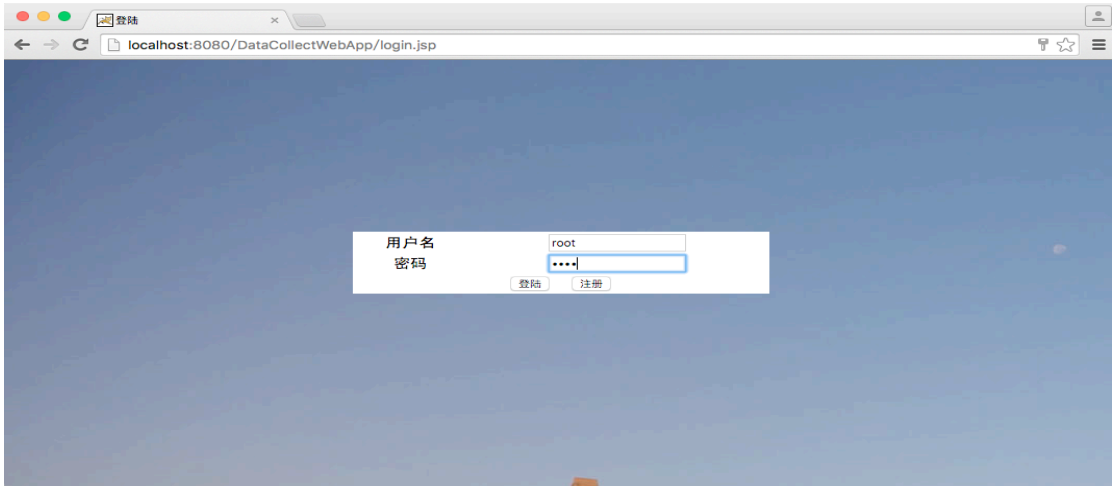

图 **3.5** 登陆界面

图3.6为注册界面，如果需要添加新用户，在获得管理员同意后，可以在这个界面添加新用户。

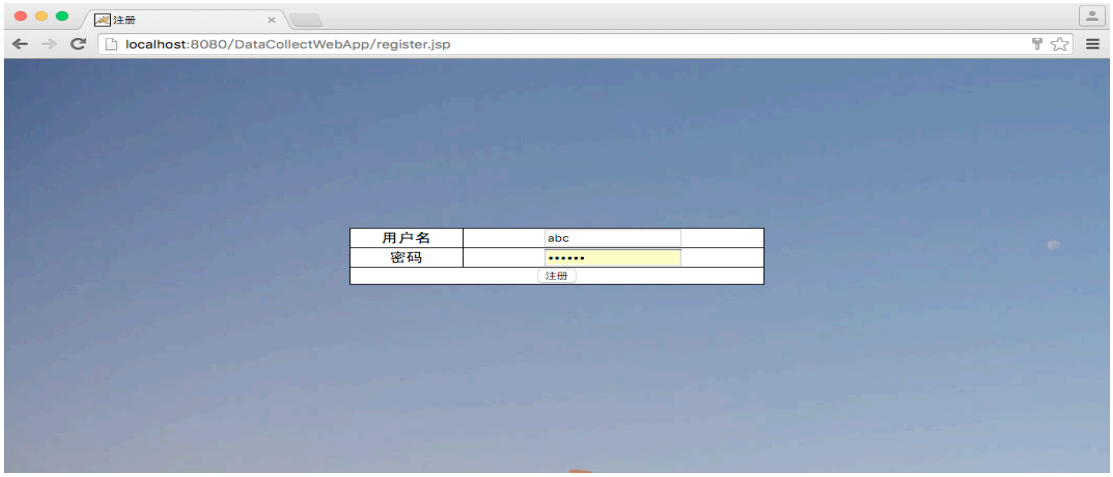

图 **3.6** 注册界面

图3.7为登陆后见到的第一个界面，展示了项目列表，列举了所有传感网项目以及简介。

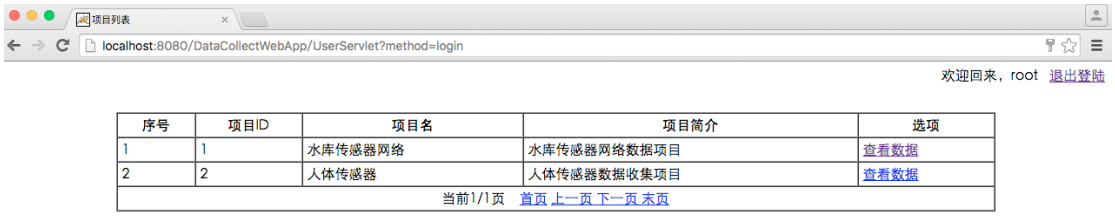

图 **3.7** 项目列表界面

在图3.7界面点击查看数据后进入图3.8所示界面，该界面列举了对应项目的所有节点以及节点基本信息。

序号	节点ID	节点名	节点经度	节点纬度
1	1	武汉大学信息学部	114.367	30.5322
2	2	武汉大学文理学部	114.371	30.5448
3	3	武汉大学工学部	114.369	30.5501
4	4	武汉大学医学部	114.362	30.5602
当前1/1页 首页 上一页 下一页 末页 地图模式				

图 **3.8** 项目节点列表

图3.9所示界面是图3.8界面点击地图模式进入的，在项目节点列表的基础添加了地图来标示节点位置。

序号	节点ID	节点名	节点经度	节点纬度
1	1	武汉大学信息学部	114.367	30.5322
2	2	武汉大学文理学部	114.371	30.5448
3	3	武汉大学工学部	114.369	30.5501
4	4	武汉大学医学部	114.362	30.5602
当前1/1页 首页 上一页 下一页 末页				

图 **3.9** 项目节点地图模式

图3.10为点击某个节点后展示的该节点的传感器记录详情，并且在这个界面可以选择下载特点参数的该节点记录到本地，通过PC客户端接收后，可以解析并展示。

图 **3.10** 节点数据列表和操作界面

传感网在大多数情况下是收集一段时间的传感器数据后打包汇总到服务端，传感网发送的文件数据会在服务端解析后以XML格式保持，提供给用户下载到本地PC客户端解析。下载文件界面如图3.11。

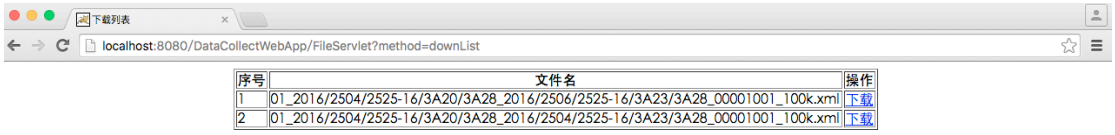

图 **3.11** 文件数据下载界面

4 客户端可视化应用设计

4.1 客户端业务逻辑简介

这里所谓的客户端细指上文提到的GUI中的第二部分，即PC客户端，主要功能有接收服务端的数据并解析、展示，发送用户的指令到传感器上。

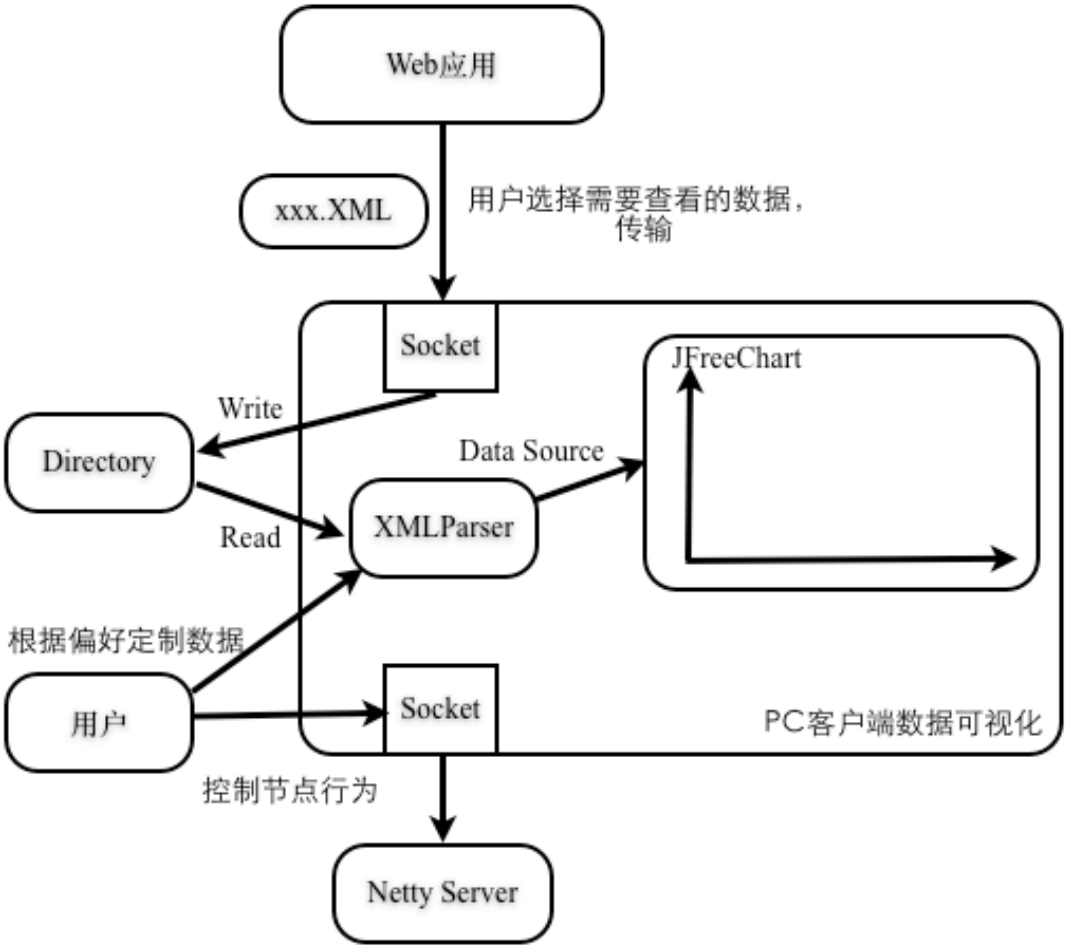

图 **4.1** 客户端架构简图

图4.1描述了客户端应用的简单架构，据此来说明客户端实现的业务逻辑。由第3章已经了解了Web应用中有一个功能是与PC客户端进行交互，即用户通过Web应用向客户端发送特定数据，由本地进行进一步解析。这里首先要解释说明的是为什么要使用一个额外的客户端来解析数据而不是直接在网页内嵌一个图形化界面和指令界面？这里基于三点考虑：

1. 前后端分离。由于数据分析是需要占用CPU时间片和内存的，而服务器的内存十分宝贵，当面对高并发时，由于服务器本身物理性能受限，可

能无法同时满足大量用户的计算需求。建立客户端正是为了缓解服务器的压力，服务器将数据打包到本地，就不需要管用户的进一步分析了。此外，如果将数据在服务端计算并通过JSP返回结果，结果的形式是图表，但本质返回的是png或是jpg等格式的图片，相对于已存文本返回数据，图片占据更大的存储空间和内存，这样也会加大服务器的压力。

2. 数据安全。建立PC客户端来接收数据，可以在一定程度上防止非正常用户通过Web获取数据，由于只能依靠客户端获取数据，非用户不具备使用客户端的条件。此外，把发送操作传感器指令的功能转移到客户端也保证了非用户不能改变传感器网的工作状态，保护了传感器网络的正常工作。
3. 数据分析的可拓展性。由于Web应用以XML格式传输数据给客户端，XML文件的标签模式是一种通用性极强的数据结构，所以接收的XML文件不仅可以通过本项目的客户端打开，同样可以被其他软件解析，数据有很好的通用性，数据分析同时具有拓展性。

回归业务逻辑，在Web应用上根据用户选择的偏好Web服务器以XML格式发送数据给客户端，客户端通过一个Socket阻塞式等待接收，当XML文件数据流到达，唤醒Socket，开始接收数据，并首先进行本地存储即写到本地特定文件夹中。客户端启动界面测试效果图见图4.2。

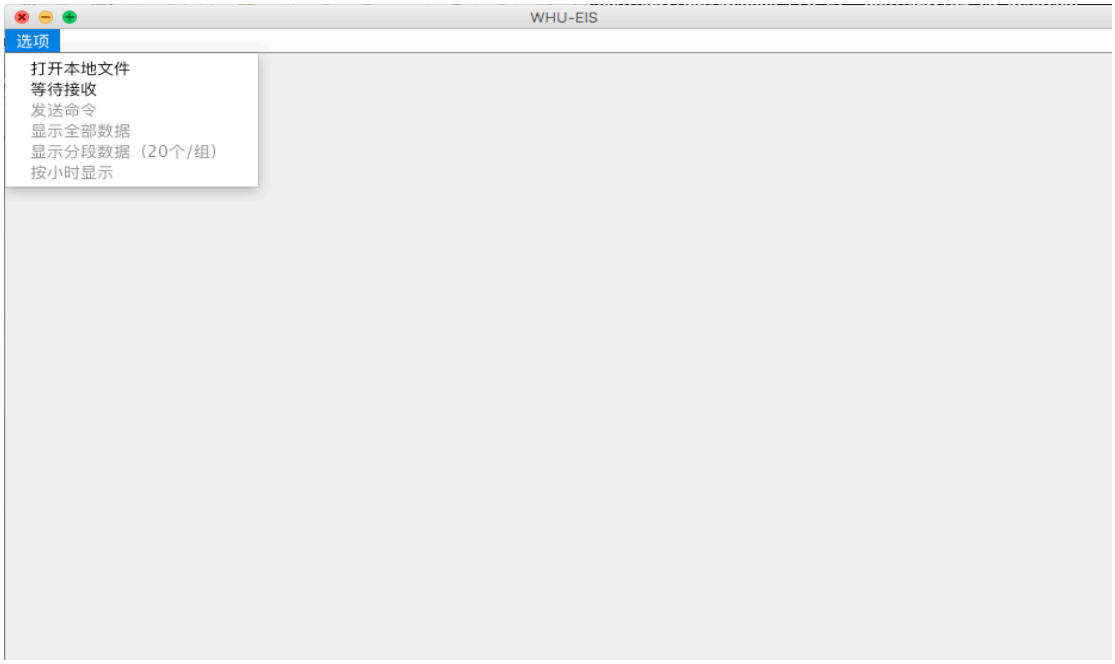

图 **4.2** 客户端启动界面

当XML接收完毕，用户可以选择读取，客户端内部有一个XMLParser模块，其功能就是解析XML文件并进行封装，将封装的结果作为数据源提供给JFreeChart插件。

JFreeChart是一个Java开源插件，专门用于数据可视化，本项目对其进行一定封装，定制了适合本项目的图表控件来展示数据。关于JFreeChart和XML将会在下文介绍。

客户端的另一个功能是使用户有控制传感器的渠道。用户通过在界面上选择，可以给Netty Server发送指令，通过Server发送指令到传感网，以此来控制传感器的检测参数。发送指令界面测试效果见图4.3。

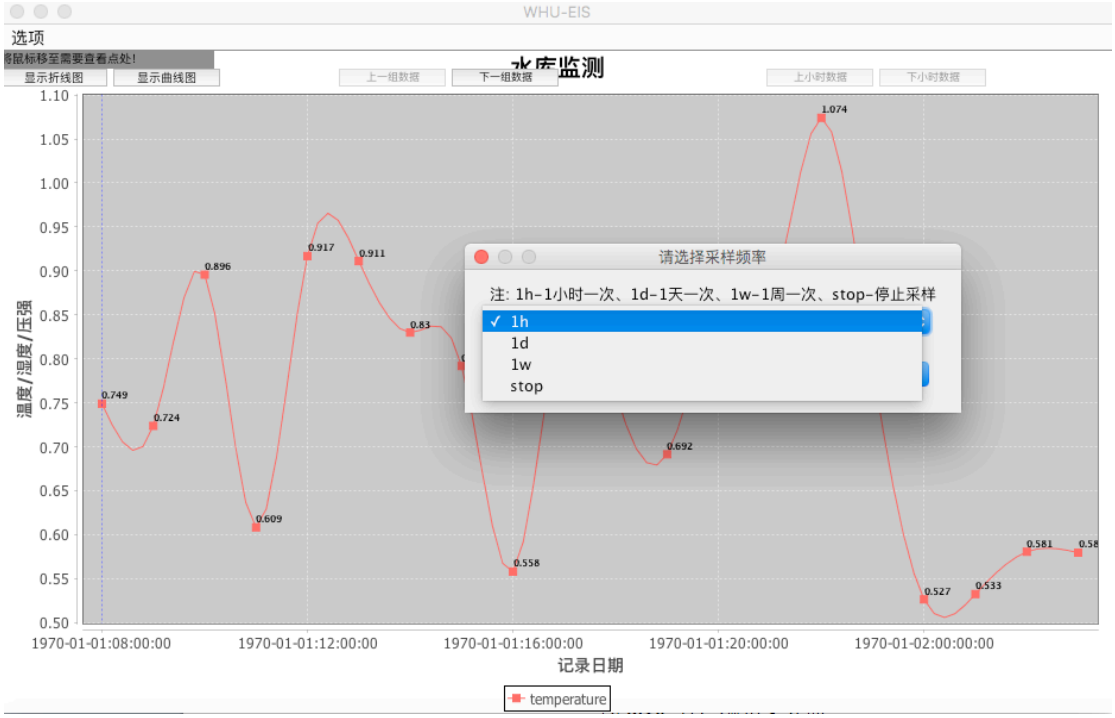

图 **4.3** 客户端指令界面

4.2 客户端实现过程

关于客户端的实现还是比较简单的，界面主要是由Java Swing的控件搭建的。主要难点在于数据的可视化和接收并解析数据方面。以下分别介绍了使用XML数据格式传输数据和使用JFreeChart实现数据可视化。

4.2.1 基于 XML 的数据传输和解析

XML(Extensible Markup Language) 即可扩展标记语言[13]。其可以用来标记数据、定义数据类型，是一种数据存储格式，其数据结构简单直观，可以作为小型数据库或者配置文件的首选。此外，XML是标准通用标记语言 (SGML) 的子集，提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。

XML文件由许多标签组成，父标签可能有子标签，其数据结构类似于数据结构中的树。由于其结构特性，解析XML衍生两种方式。

XML的DOM解析方式。所谓DOM解析，就是在解析XML树时，一次性将其加载到内存，然后在内存中构建一棵Document的对象树，通过Document对象，得到树上的节点对象，通过节点对象访问到XML文档的内容。这种方式在面对大数据文件时一次性读取可能会造成内存溢出。DOM是W3C处理XML的标准API，它是许多其它与XML处理相关的标准的基础，不仅是Java，其它诸如JavaScript，PHP，.NET等等语言都实现了该标准，成为了应用最为广泛的XML处理方式。本项目考虑到数据量不大，使用的正是DOM解析方式，使用了Java著名DOM解析工具dom4j。

XML的SAX解析方式。所谓SAX解析，即在读取XML树时按照树的前序遍历的方式，逐个节点读入数据到内存，这样在解析过程中，内存的使用是常数级别的。但这种方式也带来缺陷，即其查找方式过于繁琐，API没有DOM模式下友好，考虑到初次使用，所以不予考虑。

4.2.2 基于 JFreeChart 的数据可视化

JFreeChart图表库是一个100%免费的开源项目,使开发人员容易专业质量图表显示在他们的应用程序[14]。 JFreeChart的特性包括:

1. 一致的和证据确凿的API,支持多种图表类型;
2. 一个灵活的设计,很容易扩展,和目标服务器端和客户端应用程序;
3. 支持多种输出类型,包括Swing组件、图像文件(包括PNG和JPEG)和矢量图形文件格式(包括PDF、EPS和SVG);
4. JFreeChart是“开源”的。

本项目使用JFreeChart进行数据可视化开发，在时间序列画出了传感器数据对应的折线图。

此外，针对折线图在某些情况下不符合分析场景，本项目进一步开发了插值后的曲线图显示图。客户端可视化测试效果图见图4.4。

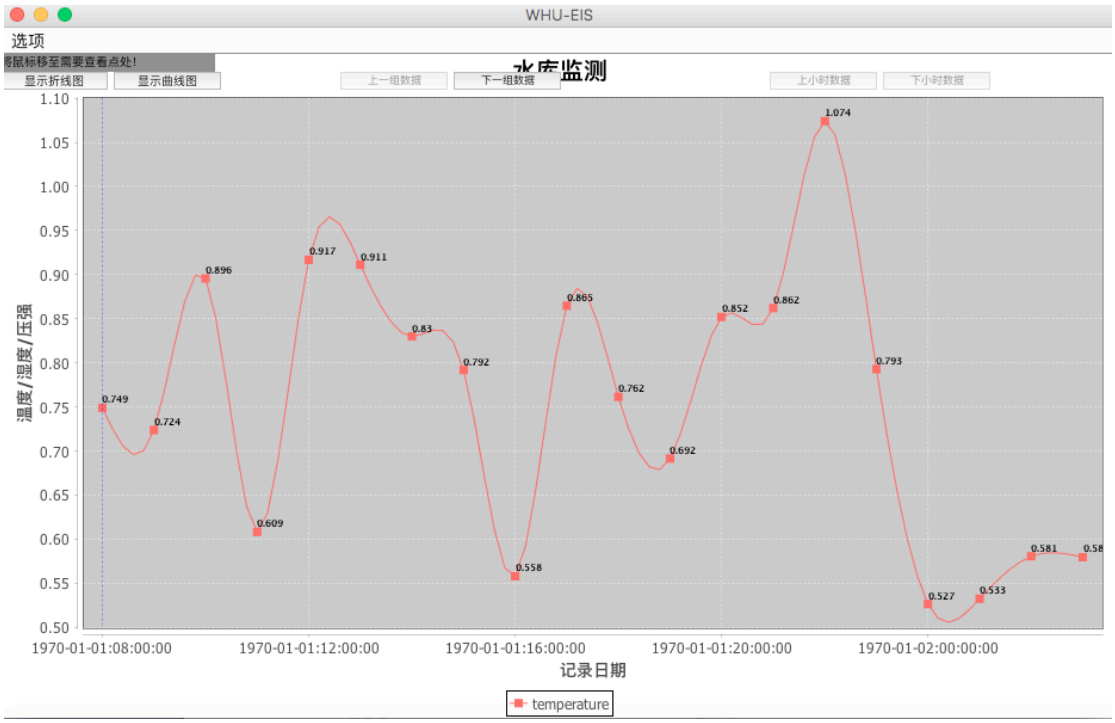

图 **4.4** 客户端展示数据界面（插值拟合后）

5 结论与展望

5.1 项目成果总结与贡献

上文已经分模块介绍了本项目涉及到的各个部分的详细功能、大致实现方案、业务逻辑。本节将说明项目实施的最后成果：

1. 完成了图1.2提到的各层架构的设计与代码上的实现。具体来说：
    a) 完成了对各个模块功能与业务逻辑的抽象；
    b) 使用Java编写了基于Socket的通信服务器与传感网通信；
    c) 搭建稳定可靠数据库用于存储传感网数据；
    d) 使用Java编写了基于Tomcat服务器的符合J2EE规范的Web应用，用于用户在浏览器上与系统进行交互；
    e) 使用Java编写了PC端的客户端程序，用于用户在本地查看数据、进行数据分析以及向传感网发送信息。
2. 完成业务流程设计与实现：
    a) 用户管理业务：涉及用户信息管理与用户在系统中的操作权限管理；
    b) 数据汇集业务：涉及高并发情况下数据稳定采集、解析与存储[18]；
    c) 双向通信业务：涉及传感网与服务器通信协议设计、通信数据解包和设置传感网配置的业务；
    d) 传感网实时方式工作业务：涉及传感网实时数据传输至服务端的业务；
    e) 传感网非实时方式工作业务：涉及传感网以文件传输非实时数据集至服务端的业务；
    f) 数据查询业务：涉及在Web应用上使用数据库数据的查询服务和在客户端上解析XML文件的数据查询服务；
    g) 数据可视化分析业务：涉及在客户端上解析XML数据以图表形式多元化展示数据的业务和数据插值业务；
3. 经过多次测试，Web应用业务逻辑严谨，逻辑符合一般用户的需求，并拓展了地图可视化功能；PC客户端部分业务逻辑设计合理，提供数据的可视化分析功能，其中支持曲线拟合，分段查看，分时查看，传感网双向通

信等功能；在测试环境下(4G内存，通过Eclipse启动服务器)，服务器在面对并发量1000的情况下仍能保证数据稳定且正确接收并存储到数据库；在数据库部分，数据表的设计符合三大范式，MySQL合理配置使用连接池缓存技术，保证了同时3000个数据库连接情况下数据库不会死锁。

4. 在项目测试完成后，各模块稳定工作后，部署代码到实际项目服务器上，投入真正使用。
5. 本人在做这个毕业设计的过程中，学习了Java SE、Java Web基础、HTML、XML、数据库、设计模式、操作系统、网络通信、Eclipse和NetBean等IDE的使用等知识，感觉受益匪浅。

5.2 展望

本项目代码已经部署在服务器上。所有源代码均托管在版本控制系统GitHub上。在改系统的对应代码库中可以查看本项目从创建以来的历次改动，各个模块的代码添加均有一定简单的注释。

代码链接：https://github.com/wwhisdavid/Data-Aggregation-by-Java-Socket。

本系统虽然实现了业务需求的功能，但仍有一些不足之处，这包括用户界面UI设计美感欠缺、缺少服务器自动化部署脚本、缺少详细的使用文档、缺少规范的日志系统和代码耦合度过高等问题。这些问题需要随着系统投入使用后，根据实际的需求来进行改进和补充。

参考文献

[1] 万时光, 马小铁, 李凯. 星型无线传感器网络的应用研究[J]. 通信技术, 2009, 42(3):173-176.

[2] 李宇清. 星型网络下无线传感器网络拥塞控制机制的设计与研究[D]. 北京邮电大学, 2015.

[3] 卢崇. 无线传感器网络节点研制及网络管理软件开发[D]. 西北工业大学, 2007.

[4] 韩安波. 无线传感网数据传输研究[D].北京邮电大学,2015.

[5] 李林锋. Netty 权威指南[M]. 电子工业出版社, 2014.

[6] 布赖恩特. 深入理解计算机系统[M]. 机械工业出版社, 2011.

[7] 刘蓬. NIO 高性能框架的研究与应用[D]. 湖南大学, 2013.17.曾自强. 基于 NIO 的 java 高性能网络应用的技术研究[D]. 北京邮电大学, 2009.

[8] WILLIAM STALLINGS[美]. 操作系统--精髓与设计原理(第 5 版)[M].电子工业, 2006.

[9] 谢钧;谢希仁. 计算机网络教程(第 4 版)[M]. 人民邮电出版社, 2015.

[10] 卡尔弗特. Java TCP/IP Socket 编程[M]. 机械工业出版社, 2009.

[11] 西尔伯沙茨. 数据库系统概念[M]. 机械工业出版社, 2008.

[12] 胡志华. 基于 Web 服务的多数据库集中查询系统的研究与应用[D]. 国防科学技术大学, 2003.

[13] 曾长清. 基于 XML 及中间件技术的数据汇集平台研究与开发[D]. 南昌大学, 2009.

[14] 侯俣, 刘万军. JFreeChart 在 Java Web 项目中的应用[J]. 科学技术与工程, 2008(10):2699-2701.

[15] Sierra K, Bates B. Head First Java[J]. Oreilly Media, 2003.

[16] Eckel B. Thinking in Java[J]. Thinking in Java, 2006, 117(667):212.

[17] (美)卡拉罗(Carrano, F.M.), 金名. 数据结构与算法分析(Java 语言描述)[M]. 清华大学出版社, 2007.

[18] (美)盖茨(Goetz, B.). Java 并发编程实战[M]. 机械工业出版社, 2012.

致　谢

大学四年一晃而过，把最美好的青春留给了武汉大学。这四年自己成长了很多，一路上也经历了很多。现在，站在大学的尾巴上，希望以这篇毕业论文作为大学的圆满结尾。

在此，我要首先感谢我的父母，在我人生的前二十几年无私地爱我，教导我，给予我精神上与物质上的支持。其次，我要感谢指导我毕业论文的孙涛老师，他在我毕业设计的过程中给我很多专业性的指导，他的认真负责让我难忘。接着，我要感谢本科教过我的老师们，是你们为我打开了知识的大门，虽然我以后可能不会从事本专业的工作，但我从你们身上学到的不仅是知识，还有作为一名武大人的品格。最后，我要感谢我的室友熊畅、郭阳、杨驰和我的女朋友房祥香。在我的大学生活中，是你们包容我，鼓励我，和我携手面对生活和学习的困难，谢谢你们的支持。

来了武大，学了光科。我，不后悔，山水一程，三生有幸！