目 录

摘 要

目前，我国信息化程度不断加快，尤其是各企事业单位对物资的管理，大多数都应用到了信息管理系统，可以说企业的信息化系统的建设程度将成为企业现代化的标志。

本文研究的电子元器件管理系统是以南京某研究所的实际需要为背景，对所内元器件实现信息化管理，长期以来人们使用传统人工的方式管理库存中的电子元器件，这种管理方式存在着许多缺点，如效率低等，本系统对传统人工方式的诸多缺点均加以改正，实际应用到工作中，取得良好的效果。首先对系统的开发环境及关键技术进行了详细的研究与探讨，论述了 Microsoft.NET 技术、SQL Server 2005 技术、AJAX 技术，随后对电子元器件管理系统的需求进行分析与设计，从总体上给出了系统的设计框架和数据库设计方案，并给出了每个模块将要实现的功能。论述了电子元器件管理系统的实现，说明了系统的功能实现情况，并从系统的架构出发，论述了系统每一层的实现过程。最后从系统安全角度，对系统安全进行了设计，在系统中主要应用了 RBAC 模型和用户身份验证两项安全技术。本文设计电子元器件管理系统具有良好的扩展性，对于一般的电子类企业均可加以应用。

关键词：信息管理系统，ASP.NET3.5，SQL Server2005，B/S，ADO.NET

Abstract

At present, the degree of information continues to accelerate, especially in the management of enterprises and institutions of the material, the information management system are applied to most enterprises. It Can be said that the construction level of enterprise information system will become a symbol of modernization of business.

In this paper, the electronic components management system is based on the actual needs of a Nanjing Institute, let the information management of the components come true. It has long been using traditional manual management of inventory of electronic components. This management approach, there are many shortcomings, such as low efficiency. The system's many shortcomings of traditional manual are to be corrected, the application to work obtain good results. First, research and study the system development environment and key technology in detail, discuss the Microsoft.NET technology, SQL Server 2005 technology and AJAX Technology, then analyze and design for the demand of electronic components management system. Giving the overall system design framework and database design, and each module is given the function to be achieved. Discussing the management system of electronic components, explained the achievement of the system's functionality, discussed the system implementation process of each layer from architecture of system. Finally, design the system security from the perspective of system security, two security technology are main applied in the system , they are RBAC model and user authentication. This design of electronic components management system has good scalability, for the average electronic enterprises in general can be applied.

Key words: Information Management System, ASP.NET3.5, SQL Server2005, B/S, ADO.NET

第一章 绪论

1.1 项目背景

目前，管理信息系统已经应用于我国各行各业之中，虽然目前仍处于发展阶段，但随着市场经济的发展，企业将面临更加激烈的市场竞争，信息将成为企业从竞争中突围的关键，因而企业的信息化系统的建设程度将成为企业现代化的标志。因此，实现管理的信息化是企业的必然选择。

企业的仓库管理对管理人员来说经常是相当繁琐、相当复杂的。由于企业所需要的物资种类繁多，生产的产品也是不断变化，采购、管理、发货的渠道也是各不相同，各个企业所使用的管理系统也是完全不同，库存管理员制作的各类统计报表也是种类繁多，所以库存管理需要制定一套库存管理信息系统，通过计算机操作实现库存信息化、共享化，而且制定方案应根据企业的具体情况而定。

对于一个科学的库存管理系统，必须将所掌握的所有物资类别，相应分成几个部门来进行货物的计划，采购，核销托收，最后验收入库，根据企业不同部门的不同需求来配送相应物资和设备，并可以通过库存系统随时进行库存盘点，出账，根据企业自身需要按月、按季度、按年度进行统计和分析，产生相应统计分析报表。而为了加强关键货物、设备的管理，要定期了解其储备，使用情况，根据计划定额和实际定额的比较，进行定额管理，使得资金得到最合理的使用，物资设备的储备达到最佳。

库存 ABC 分类是根据商品的货物数量比例和资金占用比例来划分的。

A 类物品： 高值——价值占库存总价值 70-80%的相对少数物品。通常为物品总数量的 15-20%。

B 类物品：中值——总值占库存总价值的 15-20%。物品数相对居中，通常占物品总数量的 30-40%。

C 类物品：低值——库存总值占库存总价值相对较小，几乎可以忽略不计，只占 5-10%。是物品的绝大多数，通常占总数的 60-70%。

显然，A 类物品是库存管理的关键；如果我们将大部分精力集中于 A 类物品的管理，使其库存量可以压缩 10-40%，那对于总库存的来说是一笔相当可观的压缩。

关于 A、B、C 分类方法有几条基本法则：

a. 控制的程度：

严加控制 A 类物品，包括做完备准确的记录、高层监督和经常评审，从供应商按大合同订单频繁交货，对车间紧密跟踪以压缩提前期。

正常控制 B 类物品，包括良好的记录与常规的关注。

尽可能简便地控制 C 类物品，例如定期目视检查库存实物、简要记录或以简便标志法表示补充存货已经在订货，采用大库存量与订货量以避免缺货，并给以低优先级。

b. 优先级

在所有活动中给予 A 类物品以最高优先级以便压缩其提前期与库存，实现精确控制。

对于 B 类物品给予正常优先级即可，仅在需要的关键时给以高优先级。

对于 C 类物品以最低优先级。

c. 订货过程

对于 A 类物品提供及时、准确的采购信息和状态查询。需要进行人工核对、定期的阶段性盘点，以及及时的评审计算机数据以压缩库存。

对于 B 类物品，按照定期或当发生重要改变时评审一次库存信息和订货数量，按照实际需要处理。

对 C 类物品可以进行盘点处理或订货点计算。订货往往仅凭业务人员的经验即可加以控制从而保证库存。

本研究课题来源于南京某研究所对于所内元器件实现信息化管理的实际需要。对于存量巨大，种类繁多的电子元器件而言，要提高存放仓库的利用率，保持高效的运转，时刻满足实际生产需求，精确控制库存数量，如果没有信息化的管理系统是几乎不可能实现的。其实，仓库库存的信息化管理主要包括通过计算机和相关信息的录入，对货物识别、分类、入库、存放、出库，进行操作管理，进行账目处理、结算处理，提供适时的数据查询，进行货位管理、存量控制，制作各种单证和报表，甚至于进行自动化控制等。所以说，仓储管理要提高效率、降低损耗、节约成本就必须实现信息化[1]。

1.2 论文的目的和意义

由上可以看出仓储是企业库存管理的核心环节，能够及时准确地反映存货信息以便企业做出生产、施工和经营的决策。而电子元器件的库存信息管理是电子类企业的重要组成部分，只有在充分掌握电子元器件的存量、存放地点、储备、使用程度的情况下才能进行准确的生产和经营决策，提高企业的效益。有效的库存管理是建立在对元器件的实时控制和支配的基础上，一般说来，管理的决策应及时到达库存管理员，由库存管理员对物流进行控制和组织。为了达到上述目的，往往需要库存管理员、元器件供应商、物资需求者、

运输者之间建立一个有效的信息网络，有关管理电子元器件的进库、出库和库存信息，实现库存信息共享的计算机管理信息系统，用来合理地控制电子元器件的库存种类和数量及库存成本。

本文所设计的系统以计算机为基础、以信息为对象，以数据库为中心集成了研究所各类电子元器件信息，是一个网络化的系统，能够满足必要的元器件信息处理和信息传递[2]。一个优秀的管理系统可以很好的解决传统的管理方式中存在的问题：

1、简化操作，安全可靠。传统的方式中，库存管理员需要对每一次的元器件使用情况做纸质记录存档，这对于事后相关信息的查询带来不便，时间长远的档案也容易遗失。而是用信息管理系统后所有的库存信息和使用信息都存于数据库中，对于查询者而言至于通过网络而不必到管理员处查询，且操作简单，信息也相对更安全、可靠。

2、管理科学、信息共享。管理员可以通过管理系统了解库存信息，进行合适的采购，保证库存种类和数量的合理性，从而在满足生产需求的同时控制库存成本。而元器件需求者通过网络查询到的共享信息选择合适产品，保证了生产效率。

信息管理系统可以帮助决策者根据过去的和当前的数据从整体上了解研究所的元器件存量和使用情况，辅助决策者制定合理的生产计划。

1.3 论文的主要研究内容

本文主要以南京某研究所对于所内元器件实现信息化管理的实际需要为背景，详细探讨了 ASP. NET 技术以及软件的 B/S 设计模式下的三层体系结构在具体构建库存管理信息系统中的应用。论文的主要研究内容如下：

第一章绪论。简要介绍本文课题来源、研究的目的和意义；主要介绍电子元器件库存管理信息系统的研究背景以及信息化管理现状和发展趋势，分析和明确了课题研究与开发的基本思路。

第二章对系统的开发环境及关键技术作了详细的阐述。论述了 Microsoft.NET 技术、SQL Server 2005 技术、AJAX 技术和几种计算机机器语言。

第三章论述了电子元器件管理系统的需求分析与设计。从总体上给出了系统的设计框架和数据库设计方案，并给出了每个模块将要实现的功能。

第四章详细阐述了电子元器件管理系统的实现。说明了系统的功能实现情况，并从系统的架构出发，论述了系统每一层的实现过程。

第五章论述了系统的安全设计。介绍了系统安全的基本原则以及系统中应用的主要安全技术。

第二章 系统的开发环境及关键技术

2.1 Microsoft.NET 概述

2.1.1 .NET Framework 简介

.NET Framework 被习惯性的翻译成.NET 框架，它是微软公司用于生成和运行下一代应用程序和 XML Web services 内部 windows 组件，它代表了未来技术发展的趋势，简化了在高度分布式 Internet 环境下开应用程序的开发。XML Web services 允许应用程序通过 Internet 进行通信和数据共享，而且不论所采取的是何种操作系统、设备或者编程语言。

在微软的设计师们的理想中，.NET Framework 主要是为了实现下列几个目标[3]：

(1) 提供一个面向对象的编程环境，且不管对象代码是在本地存储和运行，还是在本地运行但在互联网上发布，或者是在远程运行。

(2) 提供一个可以提高执行安全性的代码执行环境。

(3) 提供一个将软件的部署和版本的控制冲突降低到最小化的代码执行环境。

(4) 提供一个可以消除脚本环境或解释环境性能问题的代码执行环境。

(5) 使开发人员的经验在买你对类型大不相同的应用程序时保持一致。

(6) 按照工业标准生产所有通信环境，从而确保了基于.NET Framework 的代码可与其他任何代码集成。

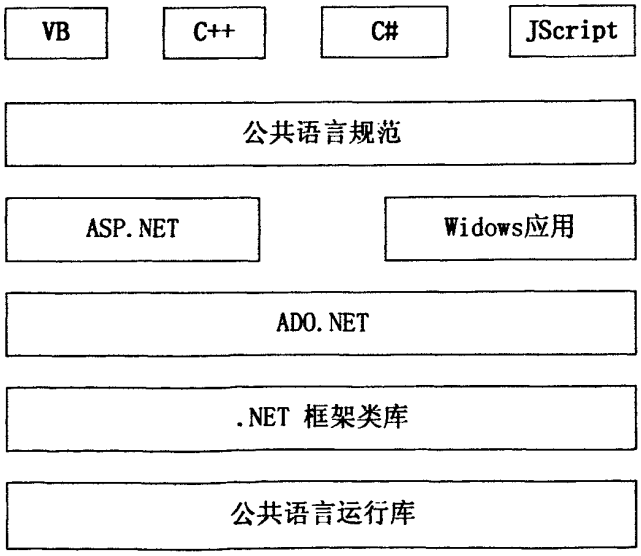

图 2-1 .NET 开发平台结构图

如图 2-1 所示，.NET Framework 主要包含的组件有如下两个：公共语言运行库和.NET Framework 类库。

（1） 公共语言运行库（CLR）

公共语言运行库是用于执行应用程序代码，是.NET Framework 的基础。用户可以将运行库当作是一个在执行时管理代码的代理工具，它主要提供内存管理、线程管理和远程处理等多种核心服务，并且还会强制性的实施严格的类型安全，以提高代码的安全性和可靠性[3]。

事实上，.NET Framework 只理解一种语言—MSIL（微软中间语言），但可以在.NET Framework 上使用 C#、Visual Basic .NET 语言编写应用程序，因为.NET Framework 包含能把这些语言编写的代码编译成 MSIL 的编译器。当使用想 C#和 Visual Basic .NET 这样的语言编写.NET Framework 应用程序时，源代码从不会直接编译成机器码。相反，C#编译器或 Visual Basic .NET 编译器把代码转换成 MSIL 这种特殊语言。MSIL 非常像以一种面向对象的汇编语言，但与传统的汇编语言不同，MSIL 不与特定的 CPU 相关，MSIL 是一种底层的、平台无关的语言。当应用程序正式运行时，JITTER(Just-In-Time 编译器)把 MSIL 代码“实时（just-in-time）”编译成机器码。通常情况下，应用程序并不会整个地从 MSIL 编译成机器码，只有在编译执行中被实际调用的方法才会被编译成机器码[4]。

CLR 中的自动垃圾收集器负责.NET 应用程序运行时的内存分配、对象布局、内存释放等内存管理问题，彻底解决了多年来困扰程序员的内存泄漏问题，大大增强了应用程序的健壮性[5]。

（2） .NET Framework 类库

.NET Framework 的另外一个非常重要的组件就是类库，.NET Framework 类库包含大量常用功能代码库，每一个类都可以有属性、方法和事件，并且作为类的成员公开给外界，是一个综合性的面向对象的可重用类型的集合，用户可以通过继承来使用代码库中已经有的代码开发传统的命令行和图形用户界面（GUI）应用程序以及基于 ASP.NET 提供的最新的应用程序，而且任何.NET 语言都可以使用.NET Framework 类库的特征与正确的对象进行交互，这有助于不同的.NET 语言之间保持一致性，因而就不必在同一台计算机或者网络服务器上安装多个组件。

2.1.2 ASP.NET 简介

ASP.NET 是微软公司推出的新一代软件开发平台.NET 的组件之一，是一种功能强大

的用于创建动态 Web 页面的服务器端技术，它是一个统一的 Web 开发模型，包括用户使用尽可能少的代码生成企业级 Web 应用程序所必需的各种服务，集中体现了微软产品一贯的功能强大、用户界面友好、相关资源丰富的特点。ASP.NET 自发布之日到现在总共发布了 ASP.NET1.x、ASP.NET2.0、ASP.NET3.0、ASP.NET3.5 等四个版本，这四个版本具有以下特点[6]：

1. 性能强大

ASP.NET 是一种编译好的公共语言运行库代码，通常在服务器上运行。与 ASP 不同，ASP.NET 可以利用早期的绑定、实时的编译、本机的优化以及盒外缓存服务，这相当于在编写代码行之前就已经显著地提高了系统的性能。ASP.NET 提供的缓存主要有三种形式：一种是页面级的输出缓存、第二种是用户控件级的输出缓存（或称为片段缓存）、第三种是缓存 API。输出缓存和用户控件级输出缓存都非常容易实现。而缓存 API 则提供了相当大的灵活性，可用于在应用程序的每一层利用缓存。

2. 提供了强大的平台与支持工具

ASP.NET 运行的基础是公共语言运行库，.NET 框架类库、消息处理和数据访问解决方案都可从 Web 无缝访问。ASP.NET 编程与计算机语言无关，此种编程可以根据自己的需要选择最适合应用程序的语言，或者是通过跨多种语言的方式分割应用程序。此外，公共语言运行库的交互性可以保证在程序迁移到 ASP.NET 时保留了基于 COM 开发中的现有投资；而 ASP.NET 框架应用了 Visual Studio.NET 集成开发环境中的工具箱以及设计器，从而可以获得强大的支持工具。

3. 编译和部署简易

ASP.NET 使常见的任务例如简单的窗体提交，客户端的登录身份验证部署以及站点配置的执行变得容易。例如，ASP.NET 自带的页面验证功能， 使 ASP.NET 应用程序的部署变得十分简单，经过编译后的组件可以直接复制到其它计算机上运行，站点的配置也很方便，可以在类似于 Visual Basic 的简单窗体处理模型中来处理事件。另外，公共语言运行库通常可以通过托管代码服务（如自动引用计数和垃圾回收）来进一步简化开发。

4. 缩放性和可用性

微软的设计师们在设计 ASP.NET 时更是考虑了其可缩放性，同时增加了在聚集环境以及多处理器环境下用于提高性能的功能。除此之外，进程也受到 ASP.NET 运行库的严密监视和管理，当进程行为处于不正常（泄漏、死锁）的时候，可以即时创建新进程，从而可以确保应用程序始终可用于处理请求状态。

5. 自定义性和扩展性

ASP.NET 还配备了一套设计非常周到的结构，它可以使开发人员在适当的级别“插入”

代码。实际上，ASP.NET 用户可以通过自己编写的各种自定义组件来扩展或者是替换其运行库中的任何一个子组件。但是如果要实现自定义的身份验证或者是状态服务至今都没有变得很容易。

6. 可管理性强

ASP.NET 实现了采用基于文本的分层系统配置，从而简化了将设置应用于服务器环境和 Web 应用程序的过程。由于各类配置信息都是以纯文本的形式储存的，所以可以在没有任何本地管理工具帮助的情况下应用各类新设置。ASP.NET 用户只需将所有必要的文件复制到指定服务器中，无需重新启动服务器即可将 ASP.NET 框架应用程序部署到服务器，。

7. 安全性高

ASP.NET 借助自身内置的 Windows 身份验证控件以及基于对每个应用程序的配置就可以保证应用程序是处于安全状态下的。此外，ASP.NET 还提供了丰富且容易被访问的安全能力级，方便了 Web 应用程序的安全创建。ASP.NET 经设计后与 InternetInformation Services(IIS)的现有安全能力实现了兼容，但同时它又具有很强的灵活性，并且可以进行扩展。这就意味着可以构建与应用程序紧密集成的各种自定义安全机制。

8. 数据访问性强

几乎所有的 ASP.NET 应用程序，对数据库访问是一个必不可少、至关重要的组成部分。而 ADO.NET 是.Net 框架内的内置数据库访问技术，是一个面向未来的数据访问组件，它可以通过使用.Net 框架内的命名空间以及几十个类，并且同时通过使用 DataSet 对象、DataAdapter 对象以及 DataGrid 控件、Datalist 控件、Repeater 控件等各种数据访问控件来实现和提高数据库的访问能力。

9. 代码与表现分离

基于 Web 的开发企业级的各种应用软件，应使表现设计与业务逻辑相分离，以实现项目的快速开发目的。在使用 ASP.NET 编写程序时，通常是把业务逻辑代码放到一个相对独立的后台代码文件中，从而可以实现代码与页面的分离，这样可以提高应用程序的层次性、可读性、可维护性以及可重用性，并且最终可以达到加快项目开发进度的目的。

2.1.3 ADO.NET 简介

.NET 开发平台，正被广大程序员所青睐，大量的软件类项目中，管理信息系统占了七成以上，各种数据库的访问成为必然[7]。一般而言,影响数据库访问性能的主要因素有三个:数据库服务器的硬件性能、网络性能以及数据库访问程序的设计。目前,前两项的性能普遍得到提高和增强,所以数据库访问的程序设计将直接影响到应用程序的最终性能[8]。顾名思义，ADO.NET 是基于 ADO 的，但在数据访问技术的演化中，它超越了基于 COM 的 ADO

而迈出的新的一步[9]。ADO.NET 是.NET 开发技术中应用非常广泛的核心开发技术，是.NET 框架中的一个重要组件，它以一种统一的访问方式来连接多种形式的数据源，其中最常见的就是关系型数据库。

设计 ADO.NET 组件是为了从数据操作中分解出数据访问，完成此任务的是 ADO.NET 的一下两个核心组件：

1、.NET 数据提供者：实现数据操作和对数据的快速读、写访问[10]。

数据提供者实际上是 ADO.NET 提供的一组封装好的类，用于操作不同类型的数据库系统，可以把这些数据提供者想象成程序和数据库之间的一座桥梁，我们的程序通过它和数据库打交道[3]。

2、DataSet 数据集：在内存中的、有着丰富功能的数据缓冲区，是一个记录集的集合[10]。

DataSet 实质上是已经从数据库中检索的记录的记录的缓冲，可以把 DataSet 看作一个小型数据库。DataSet 中的数据长久保存为 XML 格式。图 1 高度概括了 ADO.NET 应用程序体系结构[9]。

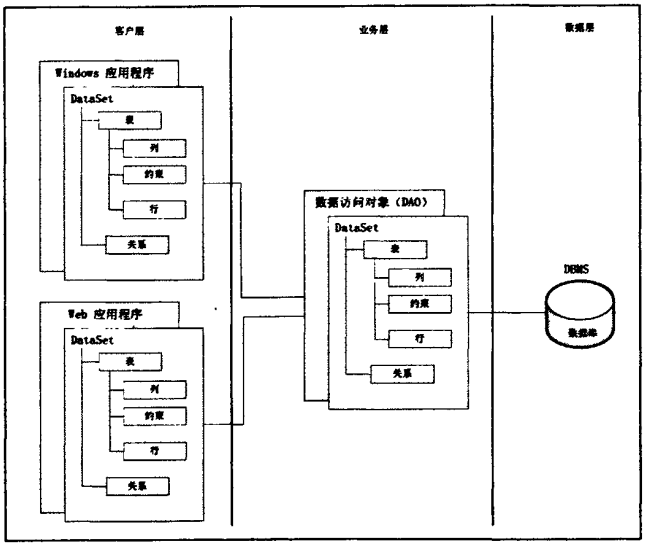

图 2-2 ADO.NET 应用程序体系结构

下面逐个介绍 ADO.NET 中主要的数据对象[3]：

1、 Connection

Connection 类的功能非常明确，它所提供的属性和方法用来建立一个数据库连接。只有建立了连接之后，对数据诸如添加、删除、修改、读取之类的操作才有可能实现。

2、 Commend

Commend 类是一个对 SQL 语句和存储过程进行执行的类，通过它，用户可以实现对数据的添加、删除、更新、查询等各种操作。

3、 DataReader

DataReader 类是一种非常有用的数据读取工具，DataReader 提供的是一种连接着数据库、只能向前的记录访问方式，它可以执行 SQL 语句或存储过程。

4、 DataAdapter

DataAdapter 类也是一个非常常用的类，因为 DataSet 必须通过 DataAdapter 才能得到数据，所以有 DataSet 的地方就有它，它也是专门为 DataSet 服务的。

2.1.4 .NET 与 J2EE

目前在 B/S 模式下的开发平台主要是分为以下两种:一种是 Sun 公司的 J2EE；另一种则是微软公司提出的.NET 开发平台。这两种技术平台各具优势，需要认真分析研究二者的差异才能选择适用于仓储管理系统的最佳方案。仓储管理系统通常需要一个相对稳定且高效能的开发和应用平台，需要相对较低的成本投入和集成的开发环境。而 J2EE 和.NET 这两种开发平台都是经过了市场上众多企业和开发者实践检验过的成熟、高效的开发平台，虽然二者对于 MXL、Web Service 等的支持也相差不多，但还是存在着很多的区别[11]。

(1) 开发环境的区别：.NET 具有非常强大的程序开发工具 VisualStudio.NET，Java 也有 Borland、Sun、Bea、BIM 等厂商的各种整合式开发工具可以选择使用。然而，相比较而言， VisualStudio.NET 的集成开发环境更加有利于实现快速高效的系统开发。

(2) 系统设计以及开发过程的区别：两者虽均采用了面向对象的开发技术，而且在系统架构的设计上，也都采用 UML、Design Pattern 等方式。但是开发者花费在 J2EE 上的学习时间要相对更长一些。

(3) 开发语言的区别：一方面，J2EE 只支持 Java 一种开发语言，而.NTE 虽然最主要支持的开发语言是 C#，但除此之外，还支持 C++、VB、Pear、COBOL 等多种计算机开发语言,这样对开发人员而言的可选择开发语言就相对较广。而且 C#支持 JTI (Just—in—time)的编译方式，而 Java 则是基于解释的方式。在另一方面，C#也正在成为一种工业标准，已经渐渐地被 ECMA(欧洲计算机制造商协会)所接纳。

(4) 支持标准的区别：J2EE 支持 Java、EBJ，而.NET 则是支持 MXL/SOAP。仅仅从标准的开放性上来看，MXL/SOAP 就明显要优于前者。MXL 正逐渐成为 Internet 上内容表示的标准，它代表了下一代网络上互操作的前景是相当光明的，而且 SOAP 协议本身也能够保证其他平台上的各种应用组件能够与.NET 平台上的应用组件进行信息交换，而 Java/EJB

模式却在这方面没有实质上的技术进步，并不能完全实现标标准的统一计算机平台。

(5) 代码通用性的区别：在.NET 平台上开发可以真正实现代码的重用，因为微软是设计师们在设计.NET 平台时的一个非常重要的思想就是在运行时把环境和具体的计算机语言相分离。所有的资源管理、内存分配、变量类型等都由运行时的环境处理，所以用 C# 编写的各种类就都可以直接用在 C/C++程序中，而在 J2EE 平台上，Java 将运行时环境和具体语言混合在一起，在代码重用方面没有任何优势。目前，.NTE 平台和服务器的稳定性也都表现得很好，并且.NET 技术还在不断改进更新之中。鉴于 Microsoft 具备借鉴、吸收其他优秀技术养分的能力和雄厚的技术实力，因此我们有理由相信.NET 平台会很好地满足我们的各种开发需求。所以我们和自然的选择了.NET 作为开发电子元器件管理系统的基础平台。

2.2 SQL Server 2005 概述

2.2.1 SQL Server 2005 简介

SQL Server 是 Microsoft 公司在数据库领域中非常重要的产品，支持 XML 和 Internet 标志，具有基于 Web 的分析能力，是关系型数据库的代表产品，很多大型数据库应用系统都采用 SQL Server 作为后台数据库。

SQL Server 采用二级安全验证、登录验证及数据库用户帐号和角色的许可验证[12]。SQL Server 支持两种身份验证模式：Windows NT 身份验证和 SQL Server 身份验证。“角色”概念的引入方便了权限的管理，也使权限的分配更加灵活[12]。

2005 年美国微软公司推出了 Microsoft SQL Server 2005，它扩展了 SQL Server 2000 的可靠性、可编程性，程序员在系统开发过程中可构建.NET SQL Server 专有对象，原因就是 SQL Server 2005 引入了.NET Framework。SQL Server 2005 还包含了多项新功能，这使得它与以往版本相比具有了更多灵活性和更强大功能[13]。

2.2.2 SQL Server 2005 的特点

SQL Server 2005 是大型的关系数据库，适用于大型或者超大型数据库服务器端，深受广大公司和开发者的喜爱，这主要取决于它的特点和定位。

SQL SERVER 2005 具有如下特点：

1、简单和易用：

SQL SERVER 2005 还是继承了微软一贯的风格，简单、友好、易用、可视化的操作界面，同时还具备了丰富的编程接口，为用户从事程序开发提供了更多的方便。

2、 客户-服务器体系结构

SQL Server 使用客户-服务器体系结构把工作负载划分成在服务器上运行的任务和在客户机上运行的任务，程序也分为客户程序和服务程序。客户程序负责业务逻辑和向用户显示数据，客户程序通常运行于客户机上，但也可以运行于安装有 SQL Server 的服务器上[14]。

3、 独特的安全认证技术

支持用户的 SQL Server 认证和 Windows 域账户认证，以及系统登录账户、数据库账户和角色管理等账户认证方式[14]。

4、 良好的兼容性

今天 Windows 操作系统在全世界都占据着主导地位，因此选择 SQL Server 2005 一定会在兼容性方面具有一定的优势。除此之外，SQL Server 2005 还具备了可以快速开发新的因特网系统的功能，尤其是它具备可以直接存储 XML 数据的功能，可以将搜索的结果以 XML 的格式输出，这样更有利于构建一个异构系统的互操作性，从而奠定了一个面向互联网的企业应用和服务的基石。这些特点都在.NET 战略中发挥着极其重要的作用[6]。

5、 提高生产效率

通过全面的 BI 功能和 Microsoft Office 系统之类的工具集成，SQL Server 2005 为组织内信息工作者提供了关键的、及时的业务信息从而可以满足他们特定的需求。SQL Server 2005 的主要目标是将 BI 扩展到组织内的所有用户，并且最终帮助所有具备不同权限的用户能够以他们最有价值的资产——数据为基础来做出更好的业务决策，从而提高生产效率[15]。

6、 数据仓库处理能力

Microsoft SQL Server 2005 最明显的改进之一就是增加了联机分析处理的功能,这可以让许多中小企业用户也可以使用数据仓库的一些特性来进行分析处理。联机分析处理可以通过多维存储技术完成对于比较大型而且较复杂数据集执行快速、高级的分析处理工作。而数据挖掘功能能够揭示出隐藏在大量数据中的数据倾向和趋势，它允许组织和机构从数据中获得最大价值。通过对现有数据进行的有效分析，可以实现对未来的趋势进行预测[6]。

7、 减少 IT 复杂性

SQL Server 2005 很大程度的降低了开发、部署以及管理业务线和分析应用程序的复杂度，它为开发人员提供了一个相当灵活的开发环境，而且为数据库管理员也提供了一个集成的高效的自动管理工具[15]。

8、 降低总体拥有成本(TCO)

SQL Server 2005 中集成的方法和对产品易用性和部署上的关注提供了行业上最低的

规划、实现和维护成本，使数据库投资能快速得到回报[15]。

9、支持自定义函数

用户可以编写自己的数据处理函数，并且使用自定义的函数就像使用系统函数和程序中的函数一样方便，这就大大提高了编程的灵活性和运行效率[14]。

2.3 ASP.NET AJAX

微软 ASP.NET 是一门终将过时的技术，在 2005 年 2 月 18 日，Jesse James Garrett 发表了他的文章：Ajax：A New Approach to Web Applications，这是对 ASP.NET 的致命一击[16]。

ASP.NET 是一项用来创建 Web 应用的服务器端技术，绝大部分的工作都在 Web 服务器端发生而不是 Web 浏览器端。与服务器端 Web 应用不同的是，Ajax 应用能够对用户交互做出非常及时的相应。Ajax 技术的关键在于异步发送请求，使用户从请求—响应—再请求的循环中解脱出来，是一种具有高度互动性和丰富用户体验的网络应用程序[17]。在 Ajax 应用中，用户界面位于浏览器中，业务逻辑层和数据访问层位于服务器上，用户界面通过 Web 服务访问业务逻辑层。

2.3.1 服务器端 AJAX

ASP.NET AJAX 的设计目标主要有以下的两点[3]：

（1）对现有的 ASP.NET 服务器端模型进行扩展，让其具备能够自动生成支持客户端的 JavaScript 代码的功能；

（2）为 ASP.NET 增加客户端更多的编程模型，从而可以实现进一步简化客户端编程的目标。

因此，对于整个 ASP.NET AJAX 框架而言实质是包含了两种截然不同但又不会相互排斥的 API：服务器端 API 和客户端 API。用户可以通过传统服务器端编程、直接的客户端编程或者是将这两者任意的组合来实现对 AJAX 应用的构建。实质上，所有基于 AJAX 的页面都是需要通过一些客户端 JavaScript 代码来实现对于浏览器的文档对象模型（DOM）的处理和所有特定于应用程序的扩展。然而，也不必将这一类脚本代码的编写工作丢给 ASP.NET 的程序员。实际上，框架可以生成作为服务器端控件的输出而专门设计的脚本语言。通过这种形式的间接页面更新方法是到目前为止将 AJAX 功能添加到新的和现有的 ASP.NET 3.5 页面的最简单、最有效的方法。在 ASP.NET AJAX 中，页面更新可由服务器控件（UpdatePanel 控件）所自动插入的一段客户端代码来管理[18]。

2.3.2 客户端 AJAX

ASP.NET AJAX 支持远程端点（通常为 ASP.NET Web 服务和 Windows 通讯基础服务，但也可能是其它内容）调用的能力被认为是以客户端为中心的编程模型的重心。直接从客户端浏览器启动时，对远程端点的调用需要 JavaScript 代理和 JavaScript 代码片段。我们可以把“客户端数据绑定”视为传统 JavaScript 运行时和 DOM 的扩展。在客户端编程中，首先连接到远程端点，然后下载数据，之后将下载好的数据绑定到 DOM 子树。此时，模板结构和一些状态信息保存在客户端上，只有原始数据从服务器移动到客户端上[18]。

MS AJAX 客户端脚本库包括以下几层[3]：

1）浏览器兼容层：为 MS AJAX 脚本提供了跨浏览器的兼容性的功能。

2）MS AJAX 核心服务：包括了一些 JavaScript 扩展，如类、继承、命名空间、事件处理、数据类型与对象序列化。

3）MS AJAX 基础类库：包含 StringBuilder、Timer、Debugger、Tracing 等新组件。

4）网络层：负责处理与 Web 服务及应用程序之间的沟通，以及管理异步远程方法调用。

5）UI 层：提供一些 MS AJAX 客户端的能力，这些能力有行为、MS AJAX 声明性语法、UI 组件和数据绑定。

6）MS AJAX 控件层：为客户端开发提供了特定 MS AJAX 控件，这些控件具有数据绑定、脚本化、绑定到 MS AJAX 行为，如拖放功能等等，这一层包含常规类型控件（对应于 HTML 常用控件）、数据导航控件，以及具有数据绑定能力的 ListView 控件。

2.3.3 ASP.NET AJAX 常用控件简介

一些最基础的 AJAX 技术，用户可以通过直接选用 JavaScript 和 XMLHttpRequest 对象来为项目构建 AJAX 特性。除此之外，很多公司提供了封装的 AJAX 集合，这些集合可以完成常用的特定的功能，而不用自己再去直接编码,ASP.NET AJAX 控件就是典型的代表，以下是有关他们的一个简单介绍：

1、 ScriptManager 控件

ScriptManager 控件是 ASP.NET 中 AJAX 功能的中心，主要负责管理页面中的所有脚本资源，其中包括 Microsoft AJAX Library 脚本下载到浏览器和协调通过使用 UpdatePanel 控件启用的部分页面更新。当部分页面呈现受支持时，ScriptManager 控件会呈现脚本以启用异步回发和部分页面更新，可使用 UpdatePanel 控件来指定要更新的页面区域。

2、 UpdatePanel 控件

UpdatePanel 控件是一个服务器控件，可帮助用户开发具有复杂的客户端行为的网页，

使网页与最终用户之间具有更强的交互性[3]。使用 UpdatePanel 控件，用户无须编写任何客户端脚本，就能实现页面的局部刷新功能，只需在一个页面上添加几个 UpdatePanel 控件和一个 ScriptManger 控件就能实现。

3、 Timer 控件

Timer 控件是一个服务器控件，它会将一个 JavaScript 组件嵌入到网页中。当经过 Interval 属性中定义的时间间隔时，该 JavaScript 组件将从浏览器启动回发。用户可以在运行于服务器上的代码中设置 Timer 控件的属性，这些属性将传递到该 JavaScript 组件[3]。

一方面，Timer 控件往往和 UpdatePanel 结合起来以实现定势异步更新页面一部分功能。另一方面，也可以使用这个控件定期 PostBack 这个页面。此控件在两种情况使用，一种是该控件位于 UpdatePanel 控件的内部；另一种是该控件位于 UpdatePanel 控件的外部[19]。

4、 UpdateProgress 控件

用户经常会遇到网页刷新速度比较慢的时候，一个空白的页面呈现在人们眼前，或者在进行注册时，明明已经单击了“提交”按钮，可由于网络通信速度或其他问题，页面看上去好像停在那里不动了，这让用户的体验很差，而如果这时网页上出现一句简短而温馨的提示，或者提供一个类似于浏览器的状态栏那样的进度条，那用户的感觉就会好很多。

在 ASP.NET AJAX 中，可以通过 UpdateProgess 控件来轻松地完成这种人性化的设置。当网页包含一个或多个用于部分页呈现的 UpdatePanel 控件时，UpdateProgress 控件可以帮助用户设计更为直观和友好的交互界面，可以是图片信息，也可以是文本信息，用户可以根据自己的项目需要或意愿进行选择。

本系统通过对 Ajax 技术的应用，实现了减少服务器和浏览器间的数据交换，加快了系统响应；将很多处理工作在发出请求的客户端完成，减少了 Web 服务器的处理时间，提高了客户端的响应速度；在不更新整个页面的前提下实现部分数据的维护和更新，改善了用户体验。

2.4 计算机语言简介

2.4.1 C#语言

Microsoft Visual C#是 Microsoft 开发的一种强大的、面向组件的语言，C#在 Microsoft .NET Framework 中有着重要的地位，一些人甚至将它与 C 在 UNIX 开发中的地位相提并论[20]。C#拥有像 Visual Basic 一样的简单易用性，同时兼具了 C++的强大功能，C#也借鉴了 Java 语言的许多优点，并对 Java 语言的一些不足之处作了改进。C#看起来与 Java 有着惊人的相似，包括了诸如单一继承、接口、与 Java 几乎相同的语法和编译成中间

代码再运行的过程。但是 C#与 Java 有着明显的不同，它借鉴了 DelPhi 的一个特点，与 COM(组件对象模型)是直接集成的，而且它是微软公司.NET Windows 网络框架的主角[21]。

2.4.2 HTML 语言

HTML（Hyper Text Markup Language，超文本标记语言）是一种文本类、解释执行的标记语言，是定义网页结构和信息内容的一种简单语言，用来编制通过万维网显示的超文本文件。用 HTML 编写的超文本文档称为 HTML 文档，也就是人们常说的"网页"，它能独立于各种操作系统平台。

HTML 文档就是各种各样的标签的集合，这些标签告诉浏览器应该如何显示网页。HTML 文档由<html>开始，由</html>结束，通常包括两个部分：包括在<head></head>中的抬头部分和包括在<body></body>中的文体部分，被<>包起来的部分称为标签，HTML 标签是成对出现的。

2.4.3 XML 语言

XML（Extensible Markup Language，可扩展标记语言）是目前发展非常迅速的一门语言，且已经形成完整的标准体系。XML 也是.NET 平台的重要组成部分，因此 XML 编程是.NET 编程中不可或缺的一部分。

XML 有如下几种用途[3]：

1、XML 可以将 HTML 与数据分离

当用户使用 HTML 来显示数据时，数据存储于 HTML 中。通过使用 XML，数据可以被存储在单独的 XML 文件中。这样用户就可以把注意力集中在使用 HTML 进行数据布局和显示上，并确保底层数据的改变不会牵扯到 HTML 的改变。XML 数据也可以作为数据岛存储于 HTML 页面内部，用户仍然可以转语音使用 HTML 对数据进行格式化和显示。

2、XML 用于交换数据

在现实世界中，计算机系统和数据库通过互不兼容的格式来容纳数据。对开发人员来说，最大的挑战一直是在因特网上的系统之间交换数据。通过将数据转换为 XML，可以极大地降低这种复杂性，并创建可被许多不同类型的应用程序读取的数据。

3、XML 可用于共享数据

由于 XML 数据存储为纯文本格式，XML 提供了独立于软硬件的数据共享解决方案，这使得不同的应用程序都可以更容易地创建数据，也更容易把某系统扩展或更新为新的操作系统、服务器、应用程序以及浏览器。

4、XML 可用于创建新的语言

XML 是 WAP 和 WML 之母。无线标记语言（WML）是有 XML 编写的。

2.4.4 JavaScript 语言

JavaScript 是一种由 Netscape 和 LiveScript 发展而来的原型化继承的面向对象的动态类型的区分大小写的客户端脚本语言，主要目的是为了解决服务器端语言，比如 Perl，遗留的速度问题，为客户提供更流畅的浏览效果。

现今的 JavaScript 的应用，使得信息和用户之间不仅是一种显示和浏览的关系，而是实现了实时性、动态性和可交互性的作用。JavaScript 与 HTML 超文本标记语言、Java 脚本语言一起完成一个 Web 页面中链接多个对象、与 Web 客户交互的过程。它弥补了 HTML 语言的缺陷，又结合 Java 和 HTML 语言的使用，具有以下特点[22]:

(l)基于对象

JavaScript 是基于对象的语言，同时也代表它是一种面向对象的语言。这说明它能使用自己创建过的对象，所以，可以利用脚本环境中对象的方法和和脚本实现诸多功能并达到相互作用的目的。

(2)简单性

简单性主要体现在: JavaScript 是一种基于 Java 基本语句和控制流上的简单又紧凑的设计，对于学习 Java 是一种很好的过度;它的变量类型都是弱类型，并未使用严格的数据类型。

(3)动态性

JavaScript 是动态的，可以直接对用户的输入做出响应，无须经过 Web 服务程序。它对用户做出的反映响应，是采用以时间驱动的方式进行的。

(4)跨平台性

JavaScript 仅依赖于浏览器本身，而与操作系统无关。只要运行的计算机支持 JavaScript 的浏览器即可。它作为一种脚本语言，都采用小程序段实现编程，可以被嵌入到 HTML 文件中，运用范围很广。

第三章 系统分析与设计

3.1 系统需求分析

系统分析是整个电子元器件库存管理信息系统开发过程中至关重要的一个环节，只有通过完整的系统分析才能把整个系统的功能和性能的整体概念描述为一个具体的系统需求说明，以便为整个系统的完整开发打好基础。仓储物流就是对整个库存产品的状态进行跟踪和管理，本系统通过与企业部门管理者的长时间的分析研究，确定了电子元器件管理信息系统的多个功能模块，其中主要包括：登录注册模块、基本信息模块、出入库模块、库存信息管理模块、历史出入库统计模块、留言板模块、帮助模块和权限设置模块等。

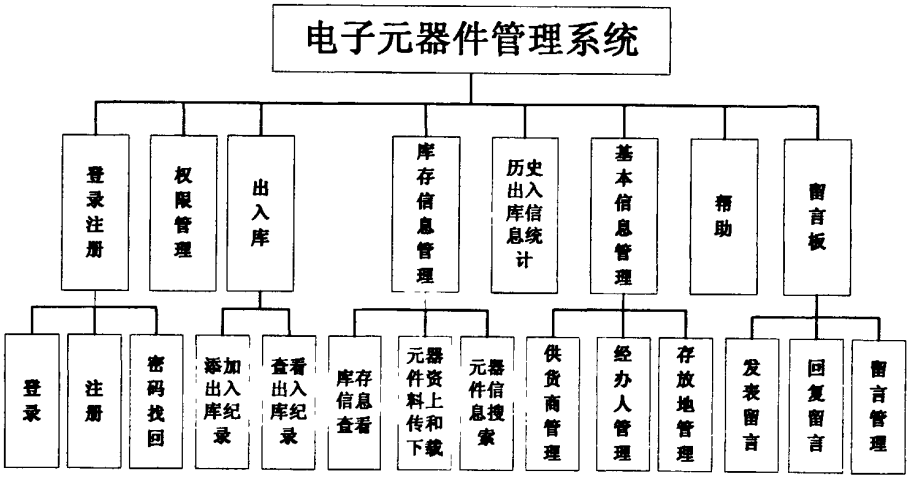

图 3-1 系统功能模块图

如图 3-1 所示，各功能模块的需求分析如下：

（1）登录注册

主要实现用户登录、注册和密码找回功能。用户登录时需要输入系统随机产生的验证码，防止恶意登录，登录时根据用户名的不同权限赋予用户不同的功能；注册时需要填写正确的邮箱，用于用户激活，将用户密码加密后存于数据库；密码找回时系统通过验证用户输入的密码保护答案将解密后的密码发送到用户邮箱，3 次输入错误答案后 24 小时内无法再使用密码找回功能。登录后系统将自动记录 cookie 值 20 分钟，20 分钟内用户在没有

注销的情况下可直接进入系统；30 分钟不操作，系统将自动清空 session 值并自动退出。

（2）出入库管理

主要实现出入库信息的添加和相关记录的管理、出入库清单查询及将出入库记录导入到 Excel 中以便保存。出库时管理员可以直接通过点击信息管理模块的领料按钮跳转并将相关信息传递到出库菜单，只需填写数量和领料人即可，当出库数量超过库存量时，系统将显示提示对话框并终止此次出库行为；入库时通过填写入库单和信息管理模块中的数据导入两种方式实现数据库更新，更新时通过 SQL 查询语句判断库存中是否已存在相同型号的元器件，若存在则更新原纪录的数量，若不存在则插入新纪录。普通用户只能本模块的出入库纪录查看及信息导出。

（3）库存信息管理

主要实现现有库存信息的查看以及相关信息的设定，是本系统的核心模块。管理员可以根据实际需要设定每一种元器件合适的存量范围，存量不符合此范围的元器件编号将会自动显示红色以作提示；通过上传按钮可将元器件的相关 PDF 格式资料上传，以便普通用户下载查阅；用户可以根据型号、名称、供货单位和存放地点中的任何一项或多项进行搜索库存中的相关元器件信息；通过 Excel 导出按钮将现有库存信息导入到 Excel 文档中，以便保存；通过 Excel 导入文件将 Excel 文档中的信息导入到数据库中。普通用户只能使用本模块的信息查看、文档下载、数据导出和搜索功能。

（4）历史出入库信息统计查询

主要实现所有用户对系统历史出入库信息（包括年、月、日、阶段统计）的查询和统计，并可以将需要的数据以 Excel 文档格式打开或保存。

（5）留言板

主要实现留言的发表、回复和管理功能，方便本系统所有用户之间的信息交流，以便将意见和建议及时反馈给研究所电子元器件库存管理员。

（6）权限管理

将管理员权限分为：库存管理员、普通用户二种。

并在用户权限管理页面显示各权限级别说明：

1、库存管理员：1-2 人。

2、普通用户：除超级管理员和库存管理员外的所有注册用户。

在设置权限时应遵循以下两点要求：

1. 低等级用户不能为高等级用户设定权限；同等级用户也不能为或同等级用户设定权限。

2. 权限设置应该：先设定用户组(2 种类型)，再设定用户组操作权限。

权限管理模块主要完成以下功能：查看用户操作记录、新增用户、禁用用户、删除用户、修改用户权限、查询用户信息。

3.2 系统的性能目标

1、安全性

任何一个信息管理系统，都会将系统的安全性放在首要位置。因此本系统在软件平台的选择和数据库平台的选型上都充分考虑了其安全性，使最终完成的系统能够提供灵活合理的权限控制体系，特别是在系统用户身份认证、用户授权权限、防止非法注入上能够提供有力的保证，其中在信息处理和信息传输的这两个环节中，尤其注意了控制和区分用户的权限，充分考虑了系统的安全性。

2、可靠性

对于一个信息管理系统而言，可靠性也极其重要的，同时它也可以作为提高系统安全性一个手段。一个优秀的系统首先要能够保证不间断的运行;将系统的平均故障时间间隔、平均修复时间、系统的最高错误或缺陷率都应该尽量减到最小。

3、可操作性

一个优秀合理的系统要求系统的整个界面能够风格统一、操作尽量简便，使用户;经过短暂的培训就能够对系统进行熟练的操作，使系统的管理员可以掌握整个系统的管理和维护。

4、先进性

开发完成的系统应该尽量能够体现出象征着现代计算机及网络发展趋势的最新技术及应用成果: 标准性、开放性、支持分布应用、高速性等，能够最终实现数据信息共享，大容量的数据传输，能够支持的不同的开发环境等特点。

3.3 UML 系统建模

对系统的需求分析完成之后，下一步我们将对系统进行 UML 建模。选择 UML 的原因是它很适合对物理数据库模式和逻辑数据库模式进行建模。下面是对电子元器件管理信息系统的一个完整的建模过程。

（1） 用例视图

用例模型是主要的 UML 示例，也是行为的建模的焦点[24]。我们已经通过需求分析确定了整个信息管理系统的各功能模块（用例），包括登录注册模块、出入库模块、基本信息

模块、历史出入库统计模块、库存信息管理模块、帮助模块、留言板模块和权限设置模块等，其用例图如图 3-2 所示。

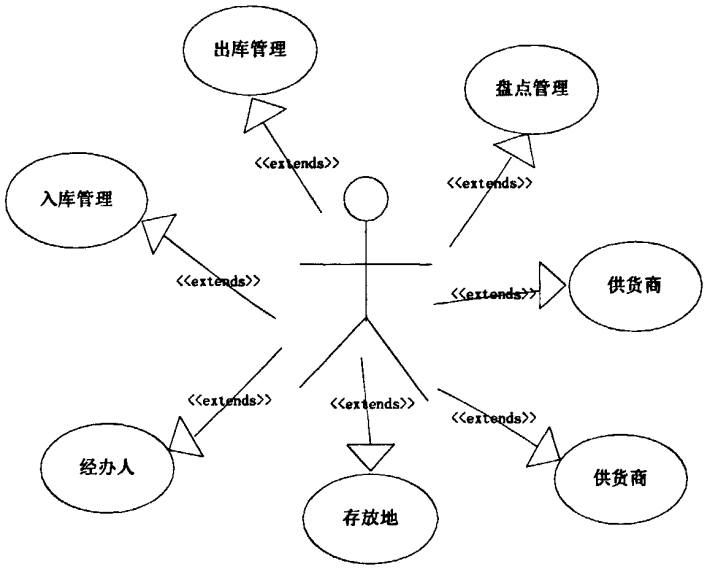

图 3-2 系统用例图

用例图将用例赋给了参与者，还允许用户在用例之间建立关系。图 3-2 显示了参与者库存管理员与所有用例的通信。

（2） 域类分析

类图是面向对象设计的心脏和灵魂[24]。由于类图用于对系统的静态设计视图建模，支持系统的功能需求，我们在此用类图进行域类分析。域类分析是建立在用例分析基础上的，我们根据用例来确定电子元器件管理信息系统当中用到的类和类之间的关系，从而得到类图。在域类分析时，应注意如下两个方面：第一，对数据对象类的确定。第二，对各类间的关联的确定。本系统的类图如图 3-3 所示。

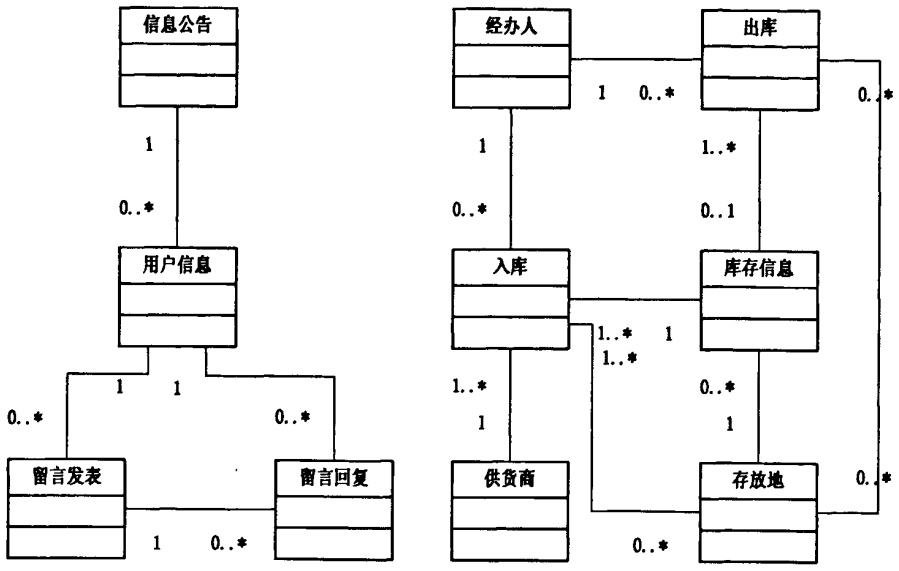

图 3-3 电子元器件管理系统类图

（3） 系统设计

在 UML 中，对系统的动态方面建模的 5 个图有：用例图、活动图、状态图、顺序图和协作图。在设计阶段，我们需要使用状态图来描述类的对象的行为和状态，显示对象的完整的生命周期[25]。在本系统中，有状态图的类有货物信息，货物信息的状态图如下图所示。

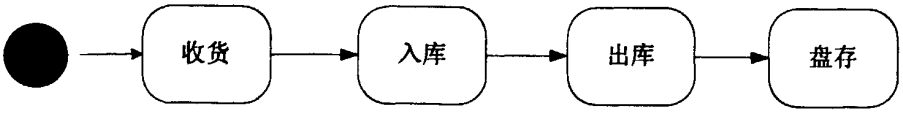

图 3-4 货物信息状态图

另外还用到了顺序图，顺序图是二维图，在水平维上显示角色（对象），在垂直维上从上到下显示消息的顺序。每一条垂直线称为对象的生命线。在生命线上被激活的方法称为激活，它作为垂直的高矩形被显示在顺序图上。顺序图可以单独使用，以可视化、详述、构造和文档化一个特定对象群体的动态信息，对一个实例的特定控制流进行建模。下面是货物入库时的顺序图。

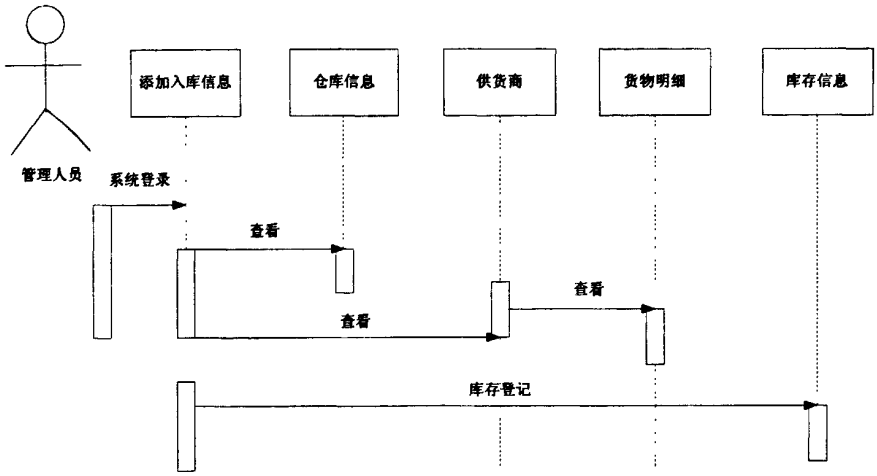

图 3-5 货物入库时的顺序图

3.4 系统体系结构设计

系统结构决定了客户端与应用服务系统的交互方式；开发技术手段指系统平台、开发工具的选型及其采用的相关开发技术，开发技术决定了系统的使用特征和用户的使用环境。目前比较成熟的信息管理系统不外乎以下两种系统结构：B/S(浏览器/服务器)和 C/S(客户端/服务器)，它们各具特色，是当前信息管理系统开发的主流模式。

3.4.1 C/S 体系结构

在以前，开发人员对数据库的设计主要选择的是 C/S 体系结构，即 client/server（客户机/服务器）结构，这主要是由于网络资源的不对等以及为了实现数据信息的共享而提出来的。连接到网络中的计算机一般都分为两个联系在一起的部分，其最主要的特点就是：应用程序逻辑分布在客户端和服务器端，其中客户端主要负责发出对访问数据资源的请求，服务器端则是负责将请求的结果返回到客户端。C/S 体系结构和传统的数据处理体系结构相比较，主要具有以下几个优点：

1、服务器应用程序的规模明显缩减，某些和具体应用相关的部分被分散到客户机端，从而使服务器的相对性能有所提高，可以同时处理更多的请求。

2、C/S 结构便于在不同的应用程序之间进行通讯。使用不同通讯协议的两个客户机应用程序不能直接通讯，却可以通过同时支持两种协议的服务器进行通讯。

但是随着信息管理系统结构的规模和复杂度的不断提高和变化，还有就是客户的高度

分散使得传统的 C/S 体系结构的扩展不易、安全性能低下、部署相对较困难等弱点也逐渐被暴露了出来，其存在的缺点主要包括以下几个方面[2]：

1、伴随着技术的不断进步，系统的规模也在不断扩大，系统的功能也在不断的增强，随着而来的便是客户数量的不断增加。由于 C/S 体系结构采用的是客户端和服务器端直接相连接的方式，导致了服务器端需要花费过多的资源来处理这些连接，从而使其他的服务不能够正常的进行或者是使得响应的速度变得很慢，更严重的是会导致使系统的崩溃。

2、由于 C/S 模式的系统需要将客服端程序安装到客户端，但是随着系统结构的规模和客户群数量的逐渐变大，这使得系统的部署变得相当的困难，而且客户端的应用程序的安装也变得十分的繁琐。因而在系统部署的时候，就必须要为每一台客户机上都安装好客户端程序的执行文件和相关动态连接库文件(*.DLL)，程序初始化文件（*.INI）等相关的文件。除此之外，还需要为客户机配置连接数据库的相关数据信息。

3、由于 C/S 的体系结构一般对数据库的应用都采用了两层结构，都在数据库层完成系统对数据的处理，而且所有的处理过程都必须要采用专用的计算机语言，因此，在系统开发人员需要更好数据库时，这些过程就变得非常难移植，要求程序员一定要重新开发新的数据库处理代码，这样就大大增加了系统的开发成本，C/S 系统可移植性差。

C/S 模式中二层结构是比较具代表性、应用也最为广泛的，它是分布式计算模式中应用最为普遍的典型结构之一，其体系结构示意图如图 3-6 所示。

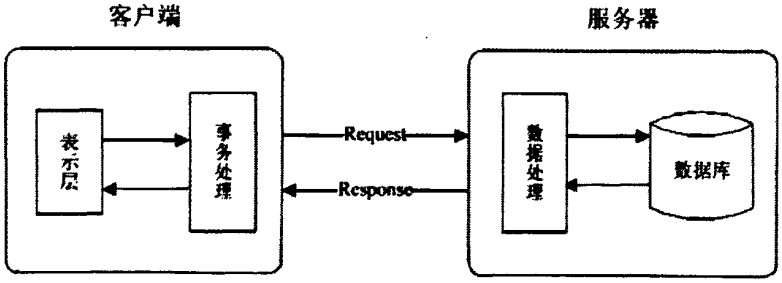

图 3-6 C/S 体系结构图

3.4.2 B/S 体系结构

B/S 体系结构，即 Browser/Server（浏览器/服务器）体系结构，是对 C/S 结构相对的一种变化和改进的体系结构。B/S 系统通常都要求由浏览器和包括 Web 服务、数据库服务、应用服务、中间件等功能的服务器组成。系统数据和程序代码可以都在服务器端设计完成，服务器端也同时可以按照功能模块的不同划分成多层体系结构，系统的大多数运算任务都

在服务器端上完成，服务器还需要负责实现与数据库的交互，然后将交互结果发送给客户端。B/S 模式相对于 C/S 模式的一个很明显的优势是降低了原 C/S 模式中客户端和服务器端之间紧密的联系，从而使程序员把主要的精力都集中放到服务器端的应用软件的开发上。B/S 体系结构示意图如图 3-7 所示。

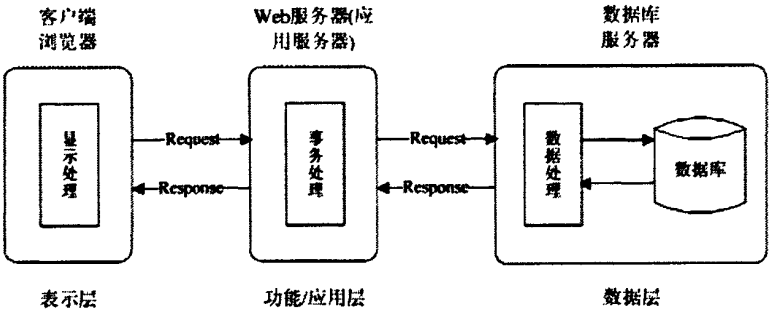

图 3-7 B/S 体系结构图

应用 B/S 体系结构开发的系统具有如下几个优点：

1、充分满足了用户在客户端的访问几乎不受任何限制的要求。

2、不受时间、地域的限制，满足了用户随时随地上网实现对信息系统的业务处理的要求。

3、系统具有良好的可扩展性，方便实现信息系统增加功能模块、完成系统的升级等多种系统扩展要求，这些都仅需要通过对服务器端网页的增加和修改就可以完成。

4、网站的部署和管理相对于 C/S 模式而言都显得简单地多、而且可维护性强。只需要更改服务器端就可实现应用程序的更新；但是在 C/S 模式下需要对所有的客户机上都安装客户端应用程序才能实现更新。

5、因为系统中所有复杂的数据计算和操作都在服务器端完成了，只有计算条件和数据通过浏览器和服务器实现传送，因此大大降低了网中的通信量，很大程度的减轻了网络的负载，极大的提高了客户浏览速度。

5、采用 B/S 结构只需在开发初期一次性投入成本购买需要的软件，后期可以根据实际需要不断的扩展系统功能，但是 C/S 系统结构下伴随着应用范围的不断扩大，整个软件的运营必须通过不断的资金投入来实现。

6、B/S 体系结构的用户界面都都安装在统一的浏览器上，浏览器使用方便、界面相对友好，而且浏览器不再负责数据的存取以及复杂数据的计算等相关任务，只需显示系统界面即可，从而极大程度的降低了系统对客户端的要求。

本系统平台采用 Web 流行的 Browser/Server 模式，以 ASP.NET 和 SQL Server2005 等开发，运行于“Windows Server 2003+IIS6.0”之上。从结构和功能上，系统可以分为表现层、业务逻辑层、数据访问层和数据层四层体系结构[26]（如图 3-8 所示）。表现层相当于用户界面，包含 ASP.NET 网页、图片、样式等界面元素，主要实现与客户端的交互；业务逻辑层是各功能的逻辑代码，是系统的核心处理部分，担当主要的应用处理任务；数据访问层主要通过 ADO.NET 实现对数据层的访问；数据层位于最底层，主要处理业务逻辑层对数据事物的请求，可独立于系统。

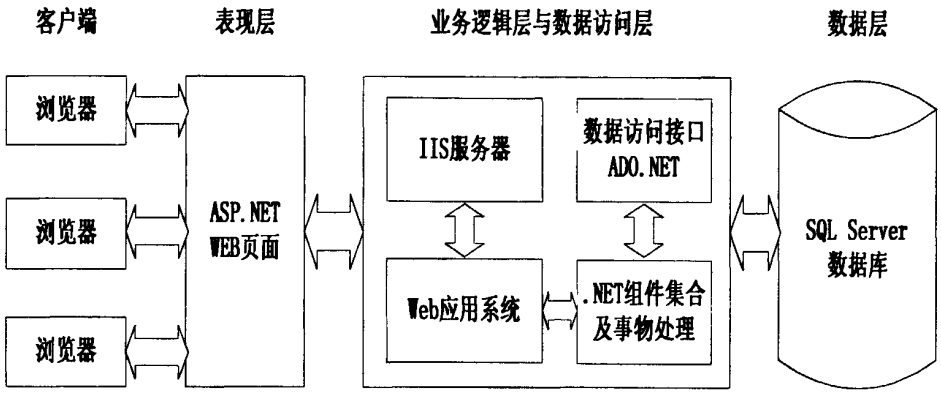

图 3-8 基于 ASP.NET 的应用系统体系架构

3.5 系统数据库设计

数据库设计的好坏将在很大程度上决定此信息管理系统是否是成功的，数据库的设计直接影响了信息管理系统的效率、安全、扩展性、可维护等重要方面，所以，数据库的设计在信息管理系统的开发过程中占有非常重要的比重。数据库设计过程一般分为 6 个阶段：需求分析、概念结构设计、逻辑结构设计、物理设计、数据库实施、数据库运行和维护[14]。

为了设计出好的信息管理系统数据库，除了要严格按照信息系统数据库的设计步骤来设计外，本系统在设计时还遵守一些必要的原则：

1、为了减少寻表的时间，基本表个数应尽量的少；

2、为了减少表与表之间的复杂度，主键的个数应尽量的少；

3、基本表的字段个数应尽量的少；

4、由于数据库管理大量的数据，为了我们能快速的读存数据，表与表之间的关系必

须明了。

3.5.1 需求分析阶段

需求分析是数据库设计的第一个阶段，从数据库设计的角度看，需求分析的任务是对现实世界要处理的对象进行详细的调查了解，通过对原有系统的了解，收集支持新系统的基础数据，并对其进行处理，在此基础上确定新系统的功能。简言之，就是获得用户对所要建立的数据库的信息内容和处理要求的全面描述[14]。

在仔细调查收集了企业库存电子元器件管理过程各种信息的基础上，设计出本系统所处理的主要数据流程图如图 3-9 所示。

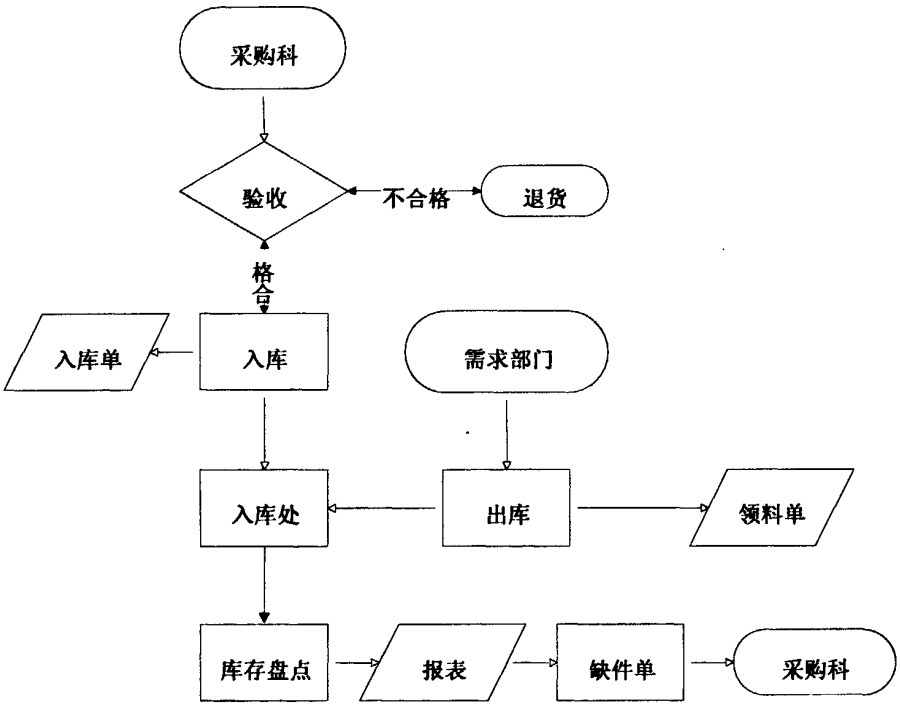

图 3-9 库存管理系统业务流程图

通过对企业库存管理内容和数据流程分析，设计的数据项合数据结构如下：

1、元器件代码信息。包括的数据项有元器件型号、名称、封装类型、生产厂商。

2、现有库存信息。包括的数据项有现有元器件、现有数目、最大库存、最小库存、存放地点、元器件简介文档等。

3、元器件入库信息。包括的数据项有入库的元器件、经办人、数目、供应商、采购单价、入库时间、地点等。

4、元器件出库信息。包括的数据项有出库的元器件、经办人、数目、出库时间等。

有了上面的数据结构、数据项和数据流程，就可以进行下面的数据库设计。

3.5.2 数据库概念结构设计阶段

在需求分析阶段，数据库设计人员充分调查分析了用户的需求，并对分析结果进行了详细的描述，但这些需求还是现实世界的具体需求。接下来应该用过选择、命名、分类等操作抽象为信息世界的结构，便于设计人员更好地用某一个 DBMS 来实现用户的需求。

将需求分析得到的用户需求抽象为信息世界结构（概念模型）的过程就是概念结构设计。简言之，数据库概念设计的任务就是根据需求分析所确定的信息需求，建立概念模型，如 E-R 模型。

根据上面的设计规划出的实体有库存实体、出库实体、入库实体、需求实体。各个实体的 E-R 图及其关系描述如下：

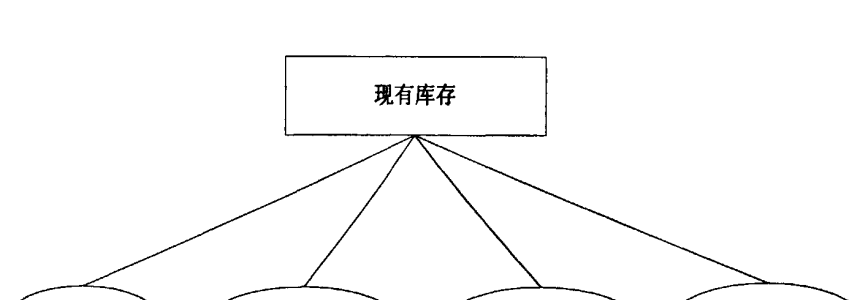

图 3-10 现有库存实体 E-R 图

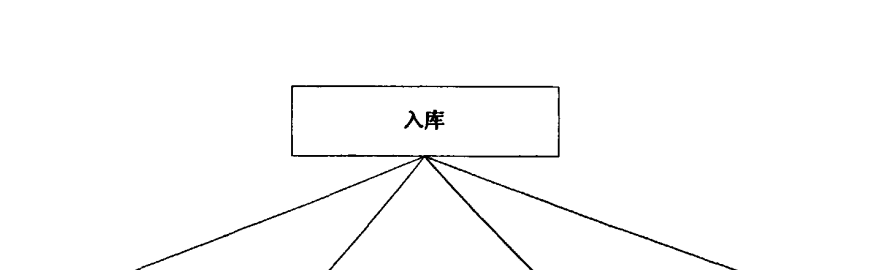

图 3-11 入库实体 E-R 图

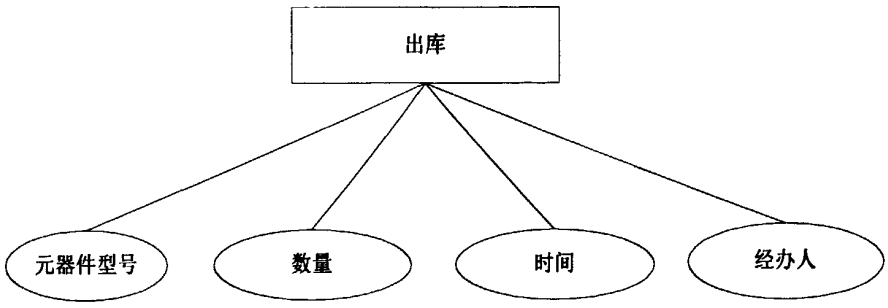

图 3-12 出库实体 E-R 图

实体与实体间的关系 E-R 图如下所示：

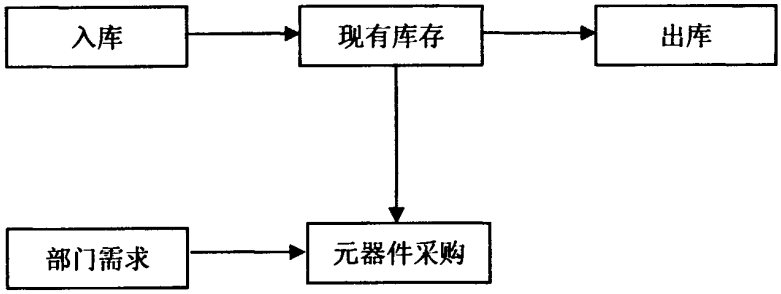

图 3-13 实体之间 E-R 关系图

通过分析实体以及实体之间关系，本系统建立的数据表及各表之间的关系如图 3-14 所示。

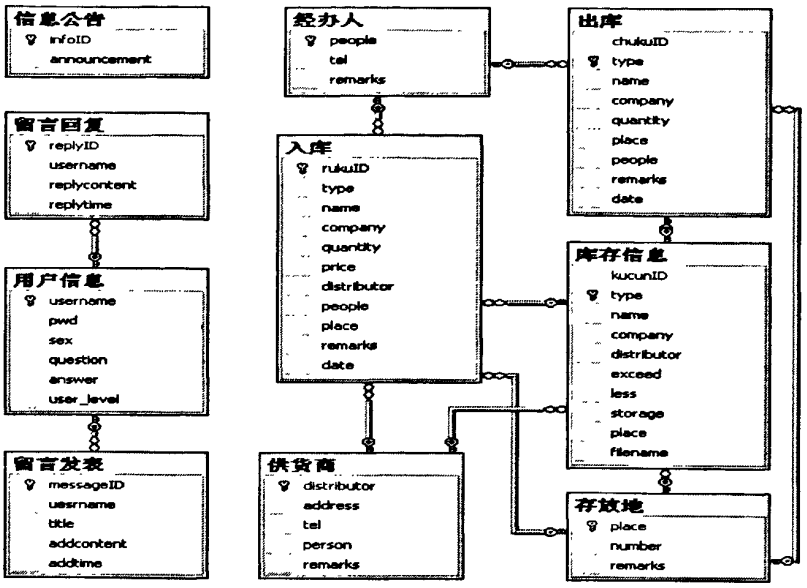

图 3-14 数据表及各表之间的关系图

3.5.3 逻辑结构设计阶段

数据库逻辑结构设计的任务是把概念结构设计阶段所得到的与 DBMS 无关的数据模型，转换成某一个 DBMS 所支持的数据模型表示的逻辑结构。数据库的逻辑设计也不是简单地将概念模型转化成逻辑模型的转换过程，而是要进一步深入解决数据库设计中的一些技术问题，如数据模型的规范化、满足 DBMS 的各种限制。

本系统的逻辑结构设计阶段主要分三部分完成[14]：

1、 由 E-R 图向关系模型的转换。遵循的原则如下：

① 一个实体型转换为一个关系模式

② 一个 m:n 联系转换为一个关系模式

③ 一个 1:n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并

④ 一个 1:1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并

⑤ 3 个或 3 个以上实体间的一个多元联系转换为一个关系模式

⑥ 同一实体集的实体间的联系，即自联系，也可按 1:1、1:n 和 m:n 三种情况分别处理

2、用关系数据理论对关系模式的规范化。

3、关系模式的优化。关系模式的优化通常以规范化理论为指导，采用合并和分解的方法。

3.5.4 物理设计阶段

数据库物理设计是对给定的关系数据库，根据计算机系统所提供的手段和施加的限制确定一个最合适应用环境的物理存储结构和存取方法。设计物理数据库结构的准备工作如下：

1、充分了解应用环境，详细分析要运行的事物，以获得选择物理数据库设计所需参数。

2、充分了解所用 DBMS 的内部特征，特别是系统提供的存取方法和存储结构。

本系统数据库中几张关键表的设计结果如下列图片所示，每张图片表示数据库中的一张表。

列名	数据类型	允许空
kucunID	int	☐
xinghao	nvarchar(50)	☐
name	nvarchar(50)	☐
ghdw	nvarchar(50)	☐
gaoxian	int	☑
dixian	int	☑
cunliang	int	☐
cunfangdi	nvarchar(50)	☐
company	nvarchar(50)	☐
filename	nvarchar(50)	☑
number	int	☐
		☐

图 3-15 库存表

列名	数据类型	允许空
ukuID	int	☐
date	char(10)	☐
xinghao	nvarchar(50)	☐
name	nvarchar(50)	☐
shuliang	int	☑
jinjia	decimal(10, 2)	☑
ghdw	nvarchar(50)	☐
jingbanren	nvarchar(50)	☐
beizhu	nvarchar(50)	☑
cunfangdi	nvarchar(50)	☐
company	nvarchar(50)	☐
fengzhuang	nvarchar(50)	☑
		☐

图 3-16 入库表

列名	数据类型	允许空
chukuID	int	☐
date	char(10)	☐
xinghao	nvarchar(50)	☐
name	nvarchar(50)	☐
company	nvarchar(50)	☐
beizhu	nvarchar(50)	☑
shuliang	int	☐
cunfangdi	nvarchar(50)	☐
jingbanren	nvarchar(50)	☐
fengzhuang	nvarchar(50)	☑
		☐

图 3-17 出库表

第四章 电子元器件管理系统的实现

实际开发过程中，首先在 Visual Studio 2008 里添加一个空白解决方案。添加好之后，依次再添加五个文件夹，用来表示系统的五个层次，分别为：数据访问层（DataBase）、业务逻辑层（Business）、表现层、系统公共类库和工具类库。系统解决方案项目示意图如图 4-1 所示。

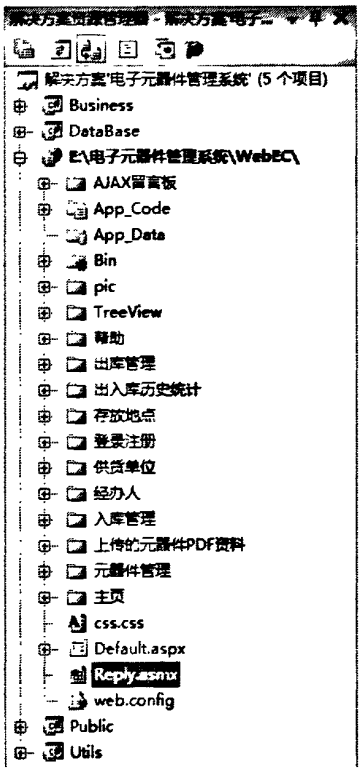

图 4-1 系统解决方案项目示意图

一般认为，好的构架是软件开发成功与否的关键。系统公共类库包括一些对话框定义、会话状态数据等，工具类库主要包括一些字符串处理、类型转换、加密解密等。下面具体论述系统数据访问层、业务逻辑层、表现层的实现过程。

4.1 数据访问层的实现

数据访问层的主要任务就是对实际的物理数据库进行操作，它需要完成的最基本的数据操作，就是隔离系统其他层次和实际物理数据库的联系，数据访问层只涉及最基本的操作而不涉及任何的逻辑处理，因为逻辑处理工作是由业务逻辑层来完成的。因此，数据访问层只需完成如何访问数据库的工作，并留出接口提供给其他层次使用就可以了。

这一层通常执行的操作是：连接数据库、执行数据库操作、数据库事务操作等。本系统的数据访问层的主要功能如下：

(1) 用户表的添加用户、更改密码、查询等的类名为 User，文件是 DataBase/User.cs。

(2) 入库表的添加、删除、查询等的类名为 Enter，文件是 DataBase/Enter.cs。

(3) 出库表的添加、删除、查询等的类名为 Out，文件是 DataBase/Out.cs

(4) 库存表的查询、删除、元器件资料的上传下载、信息设定等的类名为 ElementManage，文件名是 DataBase/ElementManage.cs。

(5) 留言表的添加、删除、更新、查询等的类名为 Message，文件名是 DataBase/Message.cs。

该层的结构图如图 4-2 所示。具体实现代码见附录。

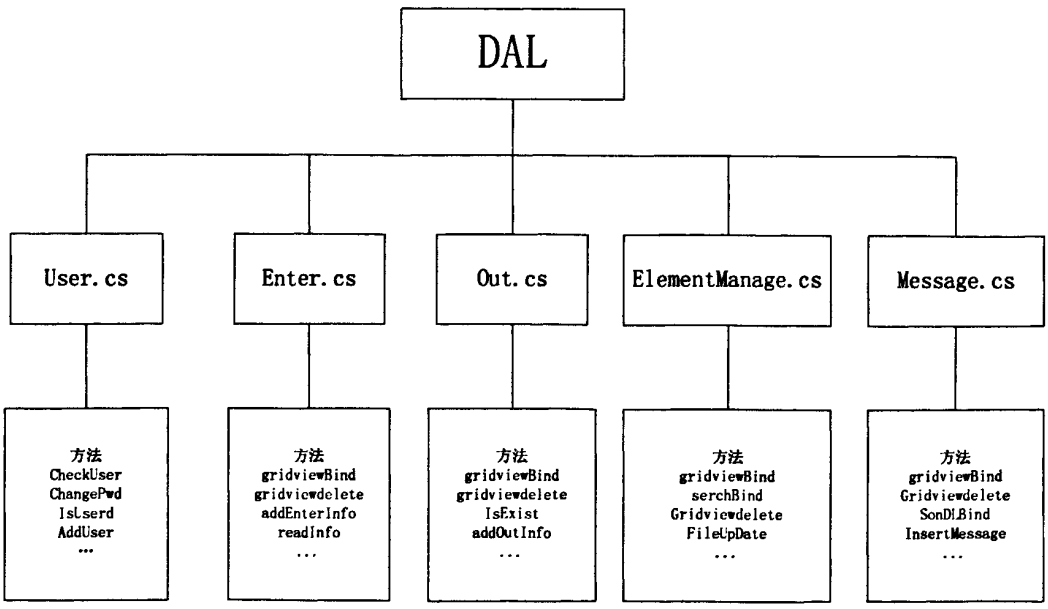

图 4-2 数据访问层结构图

4.2 业务逻辑层的实现

业务逻辑层是各功能的逻辑代码，是系统的核心处理部分，担当主要的应用处理任务。业务逻辑层可以看成数据访问层和系统界面层之间进行数据交换的桥梁[1]。在面向对象的系统中，业务逻辑是通过对象间的消息传递来实现的。在这个部分，为了保证逻辑处理的正确性和可靠性，还必须支持事务处理的能力。

在电子元器件管理系统中，业务逻辑层包含了系统所有功能模块的逻辑代码，提供界面调用的方法，如查询库存信息、添加入库信息等。业务逻辑层把操作逻辑和数据层分开，对于代码的维护和重用以及功能的扩充是有帮助的。

系统业务逻辑层各功能块的逻辑类的主要方法如下列表格所示：

（1）系统用户表的操作逻辑类的主要方法如表 4-1 所示

表 4-1 User 类的方法

方法	*功能描述*
CheckUser	用户登录时检查用户名密码
ChangPwd	修改用户密码
IsUsed	检查用户名是否已被使用
AddUser	注册新用户
IsExist	密码找回时检查用户名是否存在
IsRight	检查用户名的密码找回问题的答案是否正确

（2）系统入库表的操作逻辑类的主要方法如表 4-2 所示

表 4-2 Enter 类的方法

方法	*功能描述*
gridviewBind	入库表数据绑定
gridviewdelete	入库信息删除
addEnterInfo	添加入库表信息
readInfo	添加入库信息时查看库存表中是否存在此元器件
addElementInfo	添加库存表信息
updateElementInfo	更新库存表中相关信息

（3）系统入库表的操作逻辑类的主要方法如表 4-3 所示

表 4-3　Out 类的方法

方法	*功能描述*
gridviewBind	出库表数据绑定
gridviewdelete	出库表信息删除
IsExist	添加出库信息时查看库存表中是否存在此元器件
addOutInfo	添加出库信息
updateOutInfo	更新库存表中的相关信息

（4）系统库存表的操作逻辑类的主要方法如表 4-4 所示

表 4-4 ElementManage 类的方法

方法	*功能描述*
gridviewBind	库存表信息数据绑定
searchBind	查询数据绑定
gridviewdelete	库存表数据删除
heighBind	高警戒线元器件查询数据绑定
lowBind	低警戒线元器件查询数据绑定
IsExist	查询时检查所选元器件是否存在
WarnUpdate	设定所选元器件高低警戒线
FileUpdate	上传元器件资料
down	下载元器件资料
Export	Excel 文件导出
Import	Excel 文件导入

（5）系统留言表的操作逻辑类的主要方法如表 4-5 所示

表 4-5 Message 类的方法

方法	*功能描述*
gridviewBind	留言表数据绑定
gridviewdelete	留言表数据删除
SonDLbind	留言回复数据绑定
InsertMessage	添加留言
DateRead	回复留言时读取原数据
ReplyMessage	添加留言回复
search	查询留言
replydelete	删除回复数据

业务逻辑层的各操作逻辑类的具体实现代码见附录。

4.3 表现层的实现

表现层是直接面对用户的一层，包含 ASP.NET 网页、图片、样式等界面元素，主要功能是接受客户的各种操作，完成与客户端的交互。

界面模块作为所有模块的最高一层，基本上没有太多的代码编写，只需要调用相关的用户自定义控件到 ASPX 界面上即可完成专用工具管理子系统表现层的设计工作。系统界面层的具体 ASPX 页面利用 ASPX 页面利用用户自定义控件，只需要直接注册控件位置到 ASPX 页面即可使用。

该层为客户端提供相应的应用程序访问。系统界面层由 ASP．Net Web 窗体或者是基于 Web Service 的客户端程序实现。在 Web 窗体中，系统界面编程分为两个不同的部分：可视组件和逻辑。在 Web 窗体的逻辑文件中完全支持面向对象程序设计。可以直接声明引用．Net 框架中的类库。在不使用 Web Service 技术的情况下，通过相应的 HTML 标记和 CSS 模式来实现。Web 界面层中是用 HTML 以及各种 Web 控件来提供用户操作界面，而代码隐藏文件实现各种控件的事件处理。

下面将主要介绍本系统多个主要界面的实现，由于在第三章中已经对系统的功能作了较为详细的需求分析，因此下面的章节配合系统界面图作简要的介绍。

4.3.1 登录界面的实现

本系统的登录界面如图 4-3 所示，不同的用户权限进入系统在登录系统后可以使用的系统功能也将有所不同。登录后系统将自动记录 cookie 值 20 分钟，20 分钟内用户在没有注销的情况下可直接进入系统；30 分钟不操作，系统将自动清空 session 值并自动退出。

图 4-3 系统登录界面

4.3.2 系统登录首页界面的实现

库存管理员登录后进入的系统首页如图 4-4 所示，管理员可以根据系统提供的个各种功能进行任何合理的操作，包括修改密码；更改公告、添加查看出入库信息、查看更新库存信息、管理一些基本信息（包括供货单位、经办人、元器件存放地点等）、管理留言板等。

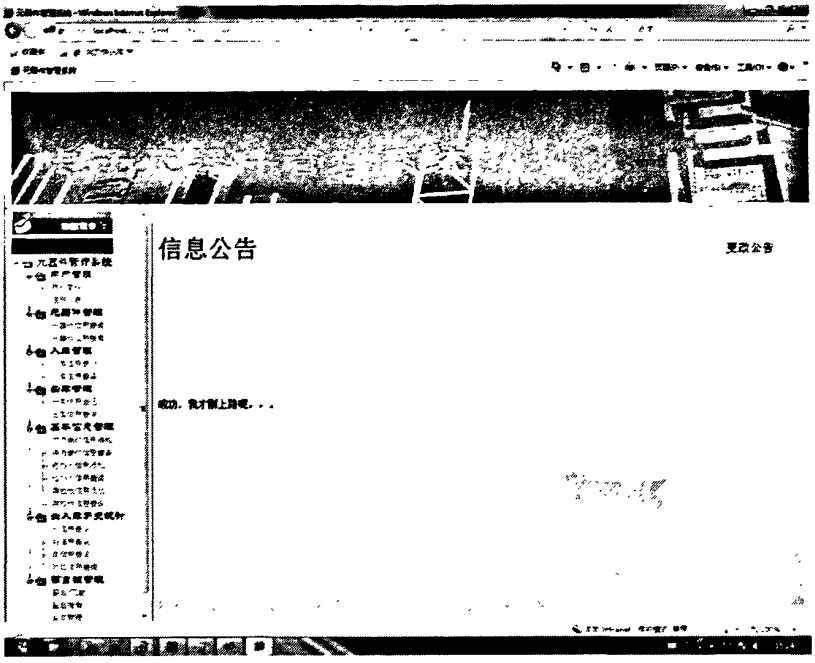

图 4-4 管理员登录首页

系统普通用户登录后进入的系统首页如图 4-5 所示，用户可以通过系统提供的各种查询和搜索功能了解电子元器件的实际库存信息，也可以通过留言板模块与管理员和其他用户交流各种信息。

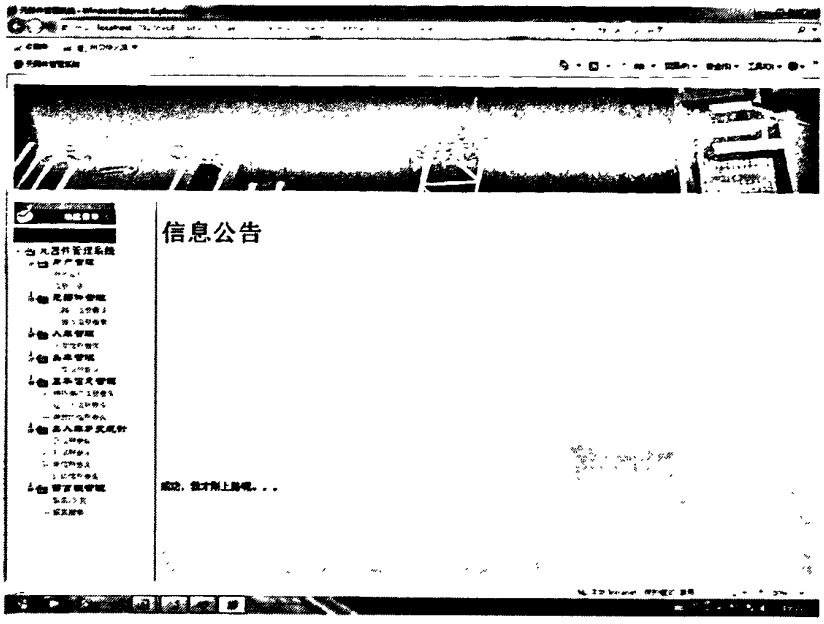

图 4-5 普通用户登录首页

4.3.3 系统库存查询界面的实现

系统管理员登录系统后点击功能菜单中的"元器件管理"按钮后进入的库存信息部分的界面如图 4-6 所示。管理员可以根据库存信息列表清楚了解库存情况，如现有元器件库存的种类、数量、存放地；查看存量不在要求范围内的各种元器件(编号为红色)；点击"详情"可以查看该元器件入库时的各种详细信息(如供货单位、经办人、单价等)；点击"领料"可以直接跳转到出库界面；点击"上传""下载"可以完成该元器件资料的上传和下载；用户可以根据型号、名称、供货单位和存放地点中的任何一项或多项完成支持通配符的库存中相关元器件信息搜索。

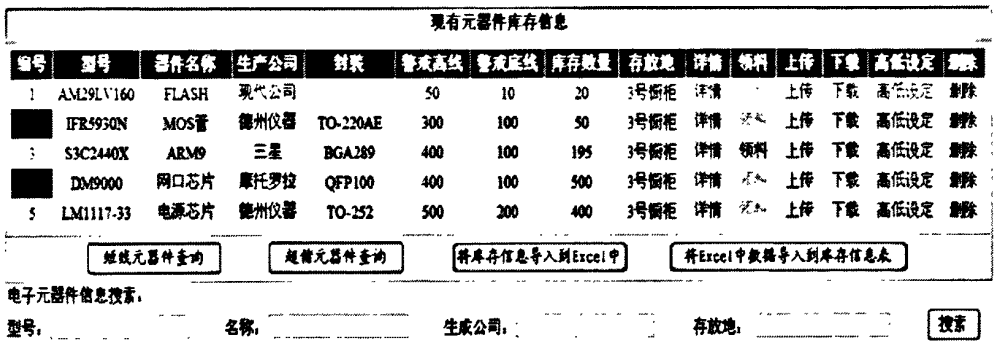

图 4-6 管理员查看库存信息界面

由于权限有限，普通用户登录系统后点击功能菜单中的“元器件管理”按钮后进入的库存信息部分的界面如图 4-7 所示。用户只能完成库存元器件信息的查询、搜索和资料下载等基本功能。

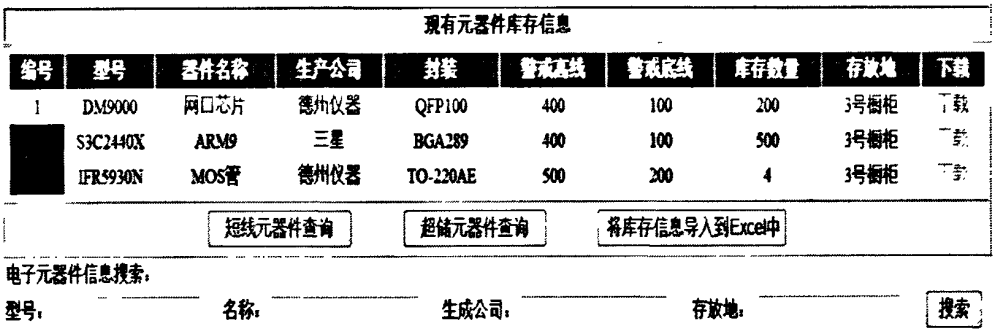

图 4-7 普通用户查看库存信息界面

4.3.4 系统出入库界面的实现

管理员登录系统后点击功能菜单中的“入库信息登记”按钮后看到的入库登记表如图 4-8 所示，有红色星号的表示必填项。当所有入库信息登记完成后点击“入库”按钮后，在入库表中添加入库记录；并系统将自动比对库存表中的库存信息，若库存中已存在此元器件，则更新库存表中元器件库存数量；若库存中不存在此元器件，则添加新纪录到库存表中。

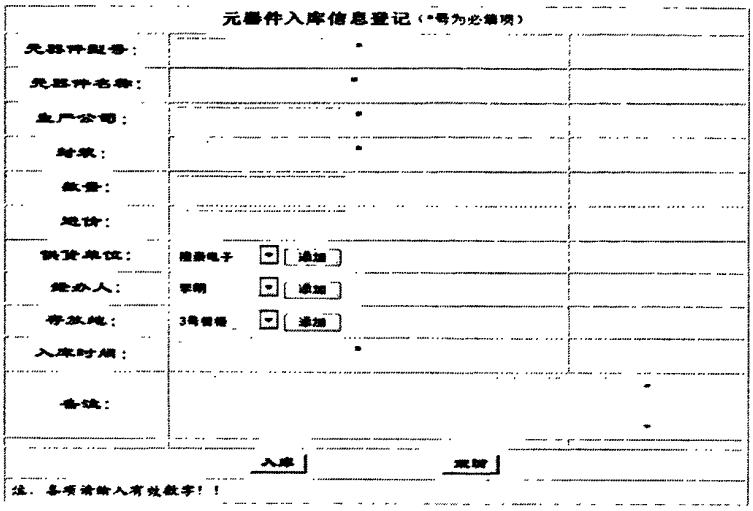

图 4-8 入库登记表

入库信息管理界面如图 4-9 所示。由于出库模块的界面和功能与入库模块基本相同，在这里就不在展开介绍了。

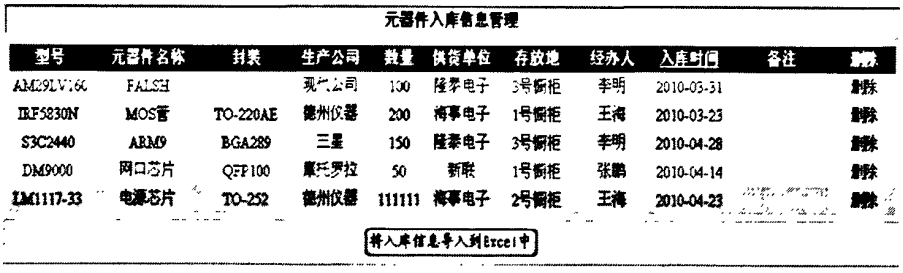

图 4-9 入库管理界面

4.3.4 出入库历史数据查询实现

主要实现所有用户对系统历史出入库信息（包括年、月、日、阶段统计）的查询和统计，并可以将需要的数据以 Excel 文档格式打开或保存。 用户登录系统后点击功能菜单中“出入库历史统计”中的“阶段信息查询”按钮后进入的界面如图 4-10 所示。

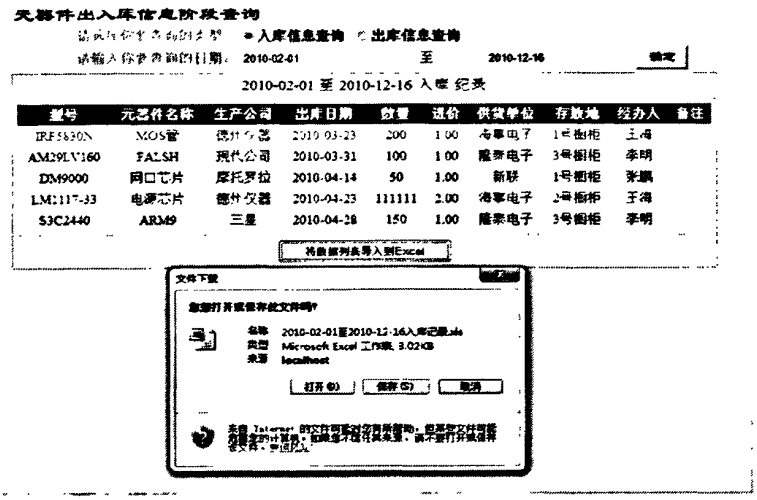

图 4-10 元器件出入库信息阶段查询界面

4.3.5 留言板界面的实现

留言板主要实现留言的发表、回复和管理功能，方便本系统所有用户之间的信息交流，以便将意见和建议及时反馈给研究所电子元器件库存管理员。用户登录系统后进入的留言板界面如图 4-11 所示。

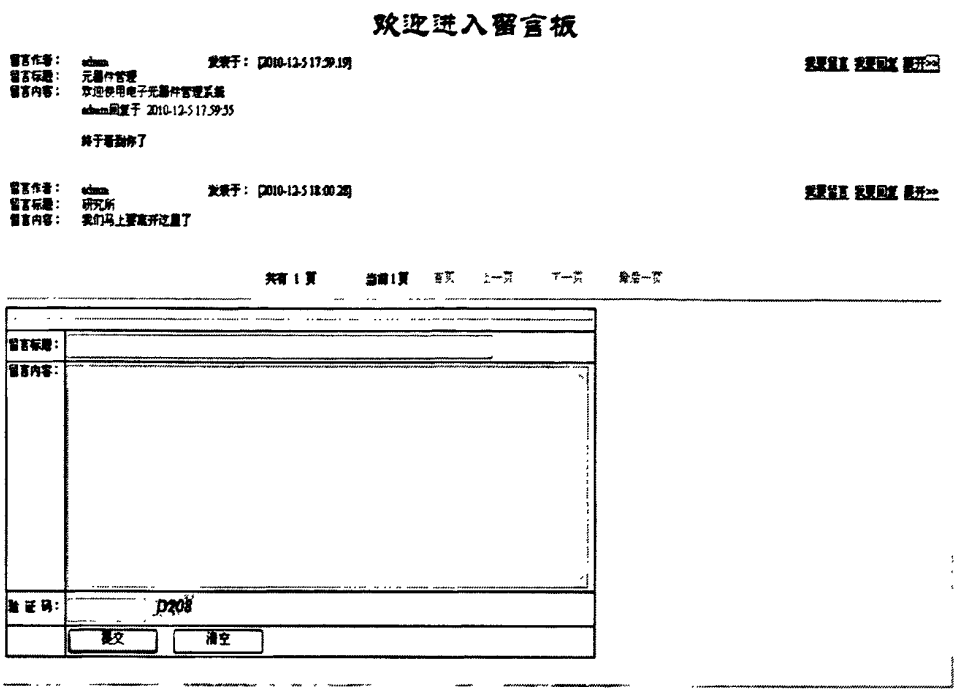
欢迎进入留言板
留言作者：
留言标题：
留言内容：
发表于：
留言标题：
留言内容：
共有 1 页
当前1页
首页
上一页
下一页
最后一页
留言标题：
留言内容：
验证码：
D208
提交
清空

图 4-11 留言板界面

第五章 系统安全设计

5.1 系统安全的基本原则

电子元器件管理系统属于一种管理信息系统，采用了 B/S 模式的三层架构的方式，用户登录时需要将用户信息发送到服务器端验证，从而极大程度的降低了系统的安全漏洞的风险，于此同时，系统还采用了防 SQL 注入式攻击的方式，这些措施对于系统数据的安全性和完整性提供了极大的保证。

(1)信息管理系统一般都需要后台数据库管理系统的支持，一个安全的数据库管理系统将直接决定一个信息管理系统应用程序及其数据文件的安全防护等级。在选择数据库管理系统时，还需要充分考虑其自身的安全策略以及安全能力。信息管理系统可以通过用户识别和访问控制的方式，保证所有的合法用都户能安全的访问系统管理员授权的各种数据，不会出现任何拒绝服务等异常情况。

(2)信息系统的一个重要组成部分便是数据库系统，对数据库信息的任何攻击，修改或者是不能及时提供相关的服务都会给用户带来非常严重的损失。所以，对数据库系统的安全保护对系统开发者而言都是至关重要的。保证数据库系统的安全主要是为了保证系统数据的独立性，保密性，完整性和可用性。通常采用的技术便是通过对数据库的访问控制、权限管理、数据库加密等方式来实现对数据库系统的保护。

(3)对数据输入途径的控制也是非常必要的。所有用户输入系统的各种数据必须都采用规范的格式，保证数据在输入后能够被系统识别。此外，还可以通过程序化的编辑检查来发现数据输入过程中存在的各种问题。

(4)到目前为止，我们为提高数据库安全所采用的技术已经显得非常重视，但在某些突发事件情况下，还是有可能对信息管理系统的正常工作造成一定的的破坏性影响。为了尽可能的避免这类事件的发生，应该将系统中最需要保护的数据库信息进行定期的备份，明确恢复系统运行的各种硬件和软件条件，并且要求系统管理员能够熟练运用系统恢复的方法。

(5)防火墙技术也是网络安全的一项十分重要的技术手段，其最主要的作用是在网络的接入点检查网络通信。主要根据用户事先设定好的各种安全规则，以保护内部网络安全为基础，实现内外网络通信。防火墙其实就是在 Intranet 和 Internet 之间设置了一种过滤器，它的设置其实是在保护网络与外界之间设置了一道安全屏障。

(6) 生成活动日志。许多信息系统程序都会自动生成各类活动日志，例如网络服务器

会生成大量的网络活动日志，计算机的操作系统会生成登录和防火墙活动日志等。但是如果人们不去查看这些日志，那么所有的日志都不会给用户带来任何的价值，。所以，一项非常必要要的安全职能就是分析这些日志，找出所有的威胁类型，成功或者是不成功的恶意攻击，以及所有安全漏洞等。

(7)管理信息系统安全高效的运作。除了从技术的角度出发采取一系列必要的安全措施之外，还应该在管理方法上多运用一些安全措施，所以，制定和完善相关的信息安全法律法规，凭借着社会的监督，提高所有计算机信息系统用户的素质，以及建立和健全信息系统安全制度和体系等就变得十分重要了。

5.2 系统中应用的主要安全技术

系统采用用户组/用户的方式进行权限管理，将系统所有的功能全部都进行拆分，每个子系统、每个子模块、每个子功能，系统中的每一个不可再分的元操作都可以进行权限分配，用户只能通过分配到的不同权限完成相应的操作。

5.2.1 身份验证

身份验证技术是检测用户否是合法的一个必要过程，在系统中登录、运行、访问系统都需要通过身份的验证。

(1)登录系统

用户取得运行程序的权限之后，还需登录系统才能进行业务的操作，离线工作时，如果客户端存在缓存的用户信息，则通过验证缓存的用户信息进行身份验证，在线工作时，通过调用检测用户身份的 Web 服务实现身份验证。

(2) 访问模块

用户通过身份验证后，成功登录系统，业务数据的操作，涉及到与远程服务器的交互，调用其他的模块之前，仍需首先调用身份验证。

(3) 基于表单的实现

首先创建一个登录页面，其中的表单允许用户输入他们的用户名和口令。然后修改应用程序的根 Web.Config 文件中的 authentication 部分，为应用程序设置身份验证模式。最后修改目录中的 Web.Config 文件中的 authorization 部分，以便在这些目录中拒绝匿名用户。

5.2.2 基于 RBAC 模型的权限管理

基于角色的访问控制方法是目前公认的解决大型系统的统一资源访问控制的有效方法。其基本原理是在用户和访问权限之间加入角色这一层，实现用户和权限的分离，用户

只有通过激活角色才能获得访问权限，其显著的两大特征是：(1)减小授权管理的复杂性，降低管理开销；(2)灵活地支持企业的安全策略，并对企业的变化有很大的伸缩性。

在 RBAC 模型应用到系统中，详细处理流程如下：

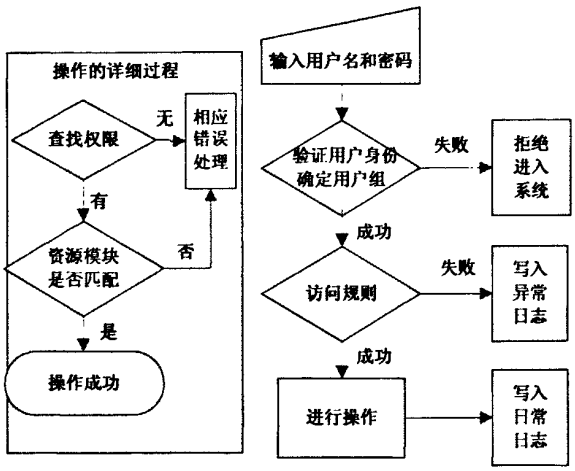

图 5-3 RBAC 模型的安全访问控制的流程图

(1) 用户登录，如果通过身份验证,则加载该用户的基本信息RT、R_IP等,并放入会话session中。将访问规则参数通过访问规则进行验证以便系统给予后续步骤，否则拒绝进入系统。

(2) 验证成功后，获得访问规则参数再调用访问规则算法。如果符合规则，查找对应角色权限表是否有操作的许可,再在资源权限表中是否有在功能资源模块操作的权限,否则返回作相应错误处理。

(3) 不符合规则，根据登录时间参数和激活角色时间参数（如none）选择审计模式。将其操作写入异常操作日志

(4) 符合规则，根据登录时间参数和激活角色时间参数（如all）选择审计模式。最后其操作写入日常日志。

(5) 安全审计员查看日常和异常操作日志。

第六章 结论与展望

6.1 全文总结

随着信息技术与 Web 技术的不断发展，仓储的信息化、信息的网络化对于存量巨大、存货品种繁多的库存管理有着重大的意义。建立基于 Web 的仓储管理信息系统可以极大地提高仓储管理的水平与效率。因此运用最新的 Web 编程技术结合以 B/S 三层结构为基础的多层体系结构建立计算机化的库存管理信息系统是本文研究的重点。

本文深入研究了 ASP.NET 的核心技术以及体系结构，以南京某研究所的电子元器件库存管理信息系统为例，详细阐述了 ASP.NET 技术以及多层结构在建立仓储 MIS 中的具体应用。其中作者做了如下的工作：

（1）研究并总结了电子元器件库存管理信息系统的开发背景和需求；

（2）参阅大量文献，了解并掌握系统开发中的各种相关技术及其优势；

（3）对系统进行了用例分析，并在分析的基础上，应用.NET 架构技术对系统 进行了框架设计和实现；

（4）根据具体的设计原则，在分析了系统功能需求的基础上，划分并且实现了系统的如下功能模块：登录注册模块、基本信息模块、出入库模块、库存信息管理模块、历史出入库统计模块、留言板模块、帮助模块和权限设置模块等；

（5）系统前台采用.NET 2008 开发工具，服务器端采用 SQL Server 2005 数据库，实现了上述设计的功能模块；

（6）在完成系统的实现之后，又应用软件测试的原理和技术，全面的对系统进行了测试。

6.2 工作展望

目前仓储管理理论在不断的成熟，涌现出了大量崭新的研究领域，各种仓储模型、算法层出不穷。在今后的工作中，将对如何利用编程语言对适用的仓储模型，最优的算法进行描述，进一步提高仓储管理者的工作效率以及决策水平作更深入的研究。

系统的模块化设计使得系统具有很大的拓展和升级维护空间，可以在应用的过程中随时根据电子元器件库存管理的不同需求增加和修改模块，从用户的需求上进一步完善系统的设计。

提高系统的网络安全是全世界共同的一个研究课题。有时系统涉及大到国家安全，小到单位机密，不仅要提高网络的安全性，还要建立更严格的系统用户身份认证制度。

上面主要就本文的研究工作和未来的研究方向作了总结和展望，由于时间仓促，作者水平非常有限，文中定有许多不足之处，恳请批评指正。

参考文献

[1] 朱吉. 基于 ASP.NET 的仓储管理信息系统的设计与实现[D]. 华中科技大学, 系统工程, 2007

[2] 王蓉. 基于.NET 的编辑部稿件管理系统[D]. 电子科技大学, 软件工程, 2010

[3] 王杰瑞, 孙更新等. ASP.NET3.5 从入门到精通[M]. 科学出版社, 2009

[4] Stephen Walther 著, 谭振林, 黎志等译, ASP.NET3.5 揭秘(卷 1), 人民邮电出版社, 2009

[5] Reilly,Douglas J. Design Microsoft ASP.NET Applications. Seattle:Microsoft Press, 2002

[6] 毛向荣. 基于 ASP.NET 和 Web 服务的物流仓储管理系统的设计与实现[D]. 湖南大学, 软件工程

[7] 叶安胜, 周晓青. ADO. NET 通用数据库访问组件构建与应用[J]. 现代电子技术, 2009(18):102-104

[8] 华国栋, 刘文予. 基于 ADO. NET 的数据库访问及其性能优化[J]. 计算机应用研究, 2004(6):215-218

[9] Michael Otey, Denielle Otey 著. 史创明, 崔金铃译. ADO.NET 技术参考大全[M]. 清华大学出版社, 2003

[10] 廖勇, 李新峰, 亮剑. NET:图解 ASP.NET 网站开发实战[M]. 电子工业出版社, 2009

[11] 石申红. 基于.NET 平台的库存装备管理信息系统的研究和设计[D]. 南京理工大学, 系统工程

[12] 胡燕. 数据库技术及应用[M]. 清华大学出版社, 2005

[13] 袁宏伟. 基于ASP.NET的现代远程教育网站设计[D]. 中国地质大学, 计算机应用技术

[14] 马涛. 数据库技术及应用[M]. 电子工业出版社, 2007

[15] 李园媛. 基于.NET 的网上购书系统设计与实现[D]. 电子科技大学, 软件工程

[16] Stephen Walther 著. 谭振林, 黎志等译. ASP.NET3.5 揭秘(卷 2), 人民邮电出版社, 2009

[17] 明日科技. ASP.NET 编程全能词典. 人民邮电出版社, 2010

[18] ASP.NETAJ 框架详细说明[OL]. http://www.mcxb.com/NetProgram/cNet/158567.html. 2007

[19] 陈开玉. 基于 ASP.NET 的学科建设管理系统[D]. 西安电子科技大学, 电路与系统

[20] Jone Sharp 著. 周靖译. Visual C# 2008 从入门到精通[M]. 清华大学出版社, 2009

[21] 陈开莲. 基于ASP.NET材料力学在线作业提交系统的开发[D]. 大连理工大学, 结构工程

[22] 王俊杰. 精通 Javaseript 动态网页编程[M]. 北京邮电出版社, 2007

[23] 汤巧英. 基于 UML 的仓储管理系统的分析设计[J]. 现代计算机,2008,8

[24] Leszek A.Maciaszek 著. 马素霞, 王素琴, 谢萍等译. 需求分析与系统设计[M]. 机械工业出版社, 2009

[25] 赵辉. ASP_NET 的冷库管理信息系统开发[D]. 河北农业大学, 农业电气化与自动化

[26] 马燕, 王文发, 李红达. 基于.NET 的四层结构研究及其应用[J]. 微电子与计算机, 2008, 25(11):188-190,194

[27] 唐金鹏, 李玲琳, 杨路明. 面向用户属性的 RBAC 模型[J]. 计算机工程与设计, 2010,31(10):2184-2186.

[28] 邢汉发, 许礼林, 雷莹. 基于角色和用户组的扩展访问控制模型[J]:计算机应用研究, 2009, 26(3):1098-1100.

科研工作及发表论文

个人简介：

张斌斌，男，1985 年 12 月 10 日出生于江苏省吴江市；

2004 年 9 月-2008 年 6 月就读于南京信息工程大学滨江学院电子信息工程专业，获得工学学士学位；

2008 年 9 月-2009 年 6 月，就读南京信息工程大学电子与信息工程学院信息与信息处理专业，完成基础理论课程的学习；

2009 年 7 月至今，参加南京中船重工第 724 研究所的“电子元器件管理系统”的设计与实现。

发表的论文：

张斌斌，陈建军，姚珅.基于 ASP.NET 的电子元器件管理系统的设计与实现.《信息技术》.

致 谢

三年的研究生学习和生活行程即将结束，我也即将顺利完成毕业设计及论文的撰写工作。首先衷心感谢我的导师陈建军研究员几年来在课题研究中给予的悉心指导和生活上的关心。从论文的选题、研究的方法、创新的思路以及论文的写作，陈老师都提供了建设性的指导和宝贵的意见，使我在硕士阶段的学习和研究得以顺利的完成。陈老师严谨认真的治学态度、宽广渊博的专业知识、对事业孜孜以求的献身精神以及平易谦逊的待人风格使我终身收益。从他那里，我不仅学会了许多知识，更重要的是学到了一丝不苟的治学态度和无私奉献的热情。在此，向陈老师表示深深的谢意和敬意。

衷心感谢实验室的所有成员以及我们的师兄、师姐们在学习上给予的帮助及一同分享学习与研究的快乐。

感谢一直以来，在我成长道路上对我默默关怀和无私付出的我的父亲、母亲以及其他亲人在我多年的求学中给予的支持与帮助。因为有了这么多老师、同学、亲人、朋友无私的爱和奉献，我才得以茁壮成长，对你们的感激之情难以言表。

附 录

1 数据库操作辅助类 SQLserver

```
namespace DB
{
    /// <summary>
    /// SQLServer类
    /// </summary>
    public class SQLServer:MyDataBase
    {
        private SqlConnection conn = null;  //数据库连接
        /// <summary>
        /// 连接字符串
        /// </summary>
        string connString =
System.Configuration.ConfigurationManager.ConnectionStrings["Pubs"].ConnectionString;
        /// <summary>
        /// 选择出结果集
        /// </summary>
        public override SqlDataReader Select(string selectSQL)
        {
            SqlConnection conn = new SqlConnection();
            conn.ConnectionString = this.connString;
            SqlCommand command = new SqlCommand(selectSQL, conn);
            try
            {
                conn.Open();
                SqlDataReader dr=command.ExecuteReader();
                return dr;
            }
            catch (SqlException ex)
            {
                this._Message = ex.Message;
                return null;
            }
            finally
            {
                conn.Close();
            }
        }
```

```
///<summary>
///更新
///</summary>
      public override bool Update(string sql)
{
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = this.connString;
    SqlCommand command = new SqlCommand(sql);
    command.Connection = conn;
    try
    {
        conn.Open();
        int result = command.ExecuteNonQuery();
        if (result > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    catch (SqlException ex)
    {
        this._Message = ex.Message;
        return false;
    }
    finally
    {
        conn.Close();
    }

}
///<summary>
///调用存储过程
///</summary>
///<param name="name"></param>
///<param name="paramList"></param>
///<returns>-1:失败</returns>
public override int CallProcedure(string name, SqlParameter[] paramList)
```

```
{
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = this.connString;
    SqlCommand command = new SqlCommand(name,conn);
    command.CommandType = System.Data.CommandType.StoredProcedure;
    //command.Parameters.AddRange(paramList);
    if (paramList != null)
    {
        foreach (SqlParameter parameter in paramList)
            command.Parameters.Add(parameter);
    }
    try
    {

        conn.Open();
        command.ExecuteNonQuery();
        return 1;

    }
    catch (SqlException ex)
    {
        this._Message = ex.Message;
        return -1;
    }
    finally
    {
        conn.Close();
    }
}
public override int CallProcedure(string name,SqlParameter parameter)
{
    SqlConnection conn = new SqlConnection();
    conn.ConnectionString = this.connString;
    SqlCommand command = new SqlCommand(name,conn);
    command.CommandType = System.Data.CommandType.StoredProcedure;
    command.Parameters.Add(parameter);
    try
    {
        conn.Open();
        SqlDataReader dr = command.ExecuteReader();
```

```
            if (dr.Read())
            {
                return 1;
            }
            else
            {
                return 0;
            }
        }
        catch (SqlException ex)
        {
            this._Message = ex.Message;
            return -1;
        }
        finally
        {
            conn.Close();
        }

    }
    /// <summary>
    /// 选取单个值
    /// </summary>
    /// <param name="selectSQL"></param>
    /// <returns></returns>
    public override object SelectValue(string selectSQL)
    {
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = this.connString;
        SqlCommand command = new SqlCommand(selectSQL, conn);
        try
        {
            conn.Open();
            return command.ExecuteScalar();
        }
        catch (SqlException ex)
        {
            this._Message = ex.Message;
            return null;
        }
```

```
        finally
        {
            conn.Close();
        }
    }
    /// <summary>
    /// 打开连接
    /// </summary>
    /// <returns></returns>
    public override bool OpenConnection()
    {
        this.conn = new SqlConnection();
        this.conn.ConnectionString = this.connString;
        try
        {
            conn.Open();
            return true;
        }
        catch (SqlException ex)
        {
            this._Message = ex.Message;
            return false;
        }
    }
    /// <summary>
    /// 关闭连接
    /// </summary>
    public override void CloseConnection()
    {
        if (this.conn != null)
        {

            this.conn.Close();
            this.conn = null;
        }
    }
    #region 快速操作，单一连接
    public override DataSet QuickSelect(string sql)
    {
        DataSet ds = new DataSet();
```

```
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = this.connString;
        SqlDataAdapter adapter = new SqlDataAdapter(sql, conn);
        try
        {
            conn.Open();
            adapter.Fill(ds);
            return ds;

        }
        catch (SqlException ex)
        {
            this._Message = ex.Message;
            return null;
        }
        finally
        {
            conn.Close();
        }

    }
    public override bool QuickUpdate(string sql)
    {
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = this.connString;
        SqlCommand command = new SqlCommand(sql,conn);

        try
        {
            conn.Open();
            int result = command.ExecuteNonQuery();
            if (result > 0)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
```

```
            catch (SqlException ex)
            {
                this._Message = ex.Message;
                return false;
            }
            finally
            {
                conn.Close();
            }
        }
        #endregion

    }
}
```

2 数据访问层实现代码

以 ElementManage 为例说明元器件管理模块中数据访问层的实现。

```
namespace DataBase
{
    public class ElementManage:BaseBN
    {

        private DataBase.MyDataBase db = DataBase.DataBaseFactory.GetMyDataBase();

        /// <summary>
        /// 元器件信息数据绑定
        /// </summary>
        public DataSet gridviewBind()
        {

            string Sql = "select * from kucun order by kucunID desc";
            return db.QuickSelect(Sql);
        }
        /// <summary>
        /// 元器件搜索信息数据绑定
        /// </summary>
        public DataSet searchBind(string type, string name, string company, string place)
        {
            ElementManageBN Bind = new ElementManageBN();
            string Sql = "select xinghao as '型号',name as '元器件名称',company as '生产公
司',cunliang as '库存数量',cunfangdi as '存放地点',dixian as '低警戒线',gaoxian as '高警戒线
```

```
',ghdw as '供货单位' from kucun where xinghao like '" + type + "%' and name like '" + name +
"%' and company like '" + company + "%'and cunfangdi like '" + place + "%'";
            return db.QuickSelect(Sql);
        }
        /// <summary>
        /// 数据删除
        /// </summary>
        public bool gridviewdelete(string ID)
        {

            string Sql = "delete from kucun where kucunID='" + ID + "'";
            return db.Update(Sql);
        }
        /// <summary>
        /// 高警戒线查询
        /// </summary>
        public DataSet heighBind()
        {
            string Sql = "select * from kucun where   cunliang>gaoxian order by kucunID
            desc";
            return db.QuickSelect(Sql);
        }
        /// <summary>
        /// 低警戒线查询
        /// </summary>
        public DataSet lowBind()
        {
            string Sql = "select * from kucun where cunliang<dixian order by kucunID desc";
            return db.QuickSelect(Sql);
        }

        ///<summary>
        ///检查所选元器件是否存在
        ///</summary>
        public DataSet IsExist(string ID)
        {
            StringBuilder Sql = new StringBuilder("select * from kucun where
            kucunID='").Append(ID).Append("'");
            return db.QuickSelect(Sql.ToString());
        }
```

```
        ///<summary>
        ///更新所选元器件高低限
        ///</summary>
        public bool WarnUpdate( string highlimit, string lowlimit,string ID)
        {
            StringBuilder Sql = new StringBuilder("update kucun set
            gaoxian='").Append(highlimit).Append("'").Append(",
            dixian='").Append(lowlimit).Append("'").Append(" where
            kucunID='").Append(ID).Append("'");
            return db.QuickUpdate(Sql.ToString());
         }
        ///<summary>
        ///更新上传文件
        ///</summary>
        public bool FileUpdate(string filename, string ID)
        {
            //更新Number为1，更新filename为文件名
            string Sql = "update kucun set number=1,filename='" + filename + "' where
            kucunID='" + ID + "'";
            return db.QuickUpdate(Sql);
        }
        ///<summary>
        ///查看是否已存在要下载的文件
        ///</summary>
        public DataSet down(string ID)
        {
            string Sql = "select * from kucun where kucunID='" + ID + "'";
            return db.QuickSelect(Sql);
        }
    }
   }
```

3 业务逻辑层的实现代码

以 ElementManage 为例说明元器件管理模块中业务逻辑层的实现。

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        string RowID = this.GridView1.DataKeys[e.RowIndex].Value.ToString();
        Business.ElementManageBN GVdelete = new Business.ElementManageBN();
        GVdelete.gridviewdelete(RowID);
        GridView1.DataBind();
        gridviewBind();
```

```
    }
    //分页
    protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        GridView1.PageIndex = e.NewPageIndex;
        GridView1.DataBind();
        gridviewBind();
    }
    //删除
    protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        string RowID = this.GridView1.DataKeys[e.RowIndex].Value.ToString();
        Business.ElementManageBN GVdelete = new Business.ElementManageBN();
        GVdelete.gridviewdelete(RowID);
        GridView1.DataBind();
        gridviewBind();
    }
    //低存储元器件查询
    protected void btn_di_Click(object sender, EventArgs e)
    {
        Business.ElementManageBN bind = new Business.ElementManageBN();
        DataSet ds = bind.lowBind();
        GridView1.DataSource = ds.Tables[0].DefaultView;
        GridView1.DataKeyNames = new string[] { "kucunID" };
        GridView1.DataBind();
    }
    //高存储元器件查询
    protected void btn_gao_Click(object sender, EventArgs e)
    {
        Business.ElementManageBN bind = new Business.ElementManageBN();
        DataSet ds = bind.heighBind();
        GridView1.DataSource = ds.Tables[0].DefaultView;
        GridView1.DataKeyNames = new string[] { "kucunID" };
        GridView1.DataBind();
    }
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        //实现自动编号
        if (e.Row.RowIndex != -1)
        {
```

```
            int id = e.Row.RowIndex + 1;
            e.Row.Cells[0].Text = id.ToString();
        }
        //高亮显示鼠标指定行数据
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            e.Row.Attributes.Add("onMouseOver",
"Color=this.style.backgroundColor;this.style.backgroundColor='lightBlue'");
            e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
        }
    }
//查询
 protected void bt_search_Click(object sender, EventArgs e)
    {
        if ((tbx_company.Text == "") && (tbx_name.Text == "") && (tbx_xinghao.Text == "")
&& (tbx_address.Text == ""))
        {
            lblmessage.Visible = true;
        }

        else
        {
            lblmessage.Visible = false;
            gridviewBind();

        }
    }
    //数据绑定
    private void gridviewBind()
    {
        string name = tbx_name.Text;
        string place = tbx_address.Text;
        string type = tbx_xinghao.Text;
        string company = tbx_company.Text;
        Business.ElementManageBN GVbind = new Business.ElementManageBN();
        DataSet ds= GVbind.searchBind(type,name,company,place);
        GridView1.DataSource = ds.Tables[0].DefaultView;
        //GridView1.DataKeyNames = new string[] { "kucunID" };
        GridView1.DataBind();
    }
```

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {
        string ID = Request.QueryString["id"];//获取从managekucun中传过来的id,
        Boolean fileOK = false;
        String path = Server.MapPath("~/上传的元器件PDF资料/");//定义文件保存路径
        if (FileUpload1.HasFile)
        {
            String fileExtension =
                System.IO.Path.GetExtension(FileUpload1.FileName).ToLower();
            String[] allowedExtensions = { ".pdf" };//允许上传的文件格式为PDF
            for (int i = 0; i < allowedExtensions.Length; i++)
            {
                //判断格式是否符合要求
                if (fileExtension == allowedExtensions[i])
                {
                    fileOK = true;
                }
            }
         }
            if (fileOK)
            {
                Business.ElementManageBN upload = new Business.ElementManageBN();
                try
                {
                    //文件保存形式
                    FileUpload1.PostedFile.SaveAs(path + FileUpload1.FileName);
                    if (upload.FileUpdate(FileUpload1.FileName, ID))
                    {
                        Response.Write("<script language='JavaScript'>alert('上传成功!
                        ');window.top.location.href('InfoInquire.aspx')</script>");
                    }
                    else
                    {
                        Response.Write("<script language='JavaScript'>alert('上传失败!
                        ');window.top.location.href('InfoInquire.aspx')</script>");
                    }
                }
                //捕捉异常
                catch (Exception ex)
                {
```

```
                    Response.Write("<script language='JavaScript'>alert('上传失败！
                    ');window.top.location.href('InfoInquire.aspx')</script>");
                }
            }
            //文件格式不符合要求
            else
            {
                Response.Write("<script language='JavaScript'>alert('上传文件格式不正
                确！');window.top.location.href('InfoInquire.aspx')</script>");
            }
        }
        else
        {
            Label1.Text = "请添加要上传的PDF文件！";
            Label1.Visible = true;
        }
    }
public void download()
    {
        Business.ElementManageBN dload = new Business.ElementManageBN();
        string ID = Request.QueryString["id"];//获取点击列ID值；
        DataSet ds = dload.down(ID);
        if (ds.Tables[0].Rows.Count != 0)
        {
            int number =Convert.ToInt32(ds.Tables[0].Rows[0][10].ToString());
            //数据库中Namber默认值为0，上传文件后更新为1
            if (number == 0)
            {
                Response.Write("<script language='javascript'>alert('对不起，你要下载的
                资料暂时无人上传！');window.location.href('InfoInquire.aspx')</script>");
            }
        }
    }
    private void dfile()
    {
        string ID = Request.QueryString["id"];//获取从managekucun中传过来的id
        Business.ElementManageBN filedown = new Business.ElementManageBN();
        DataSet ds = filedown.down(ID);
        if (ds.Tables[0].Rows.Count!=0)
        {
```

```
        string filename = ds.Tables[0].Rows[0][9].ToString();
        //在数据库中文件名的默认值为1，上传成功后更新为文件名
        if (filename != "1")
        {
            try
            {
                //获取文件路径
                string path = Server.MapPath("~/上传的元器件PDF资料/") + filename;
                //初始化 FileInfo 类的实例，它作为文件路径的包装
                FileInfo fi = new FileInfo(path);
                //判断文件是否存在；
                if (fi.Exists)
                {
                    //将文件保存到本机上
                    Response.Clear();
                    Response.AddHeader("Content-Disposition", "attachment;
                    filename=" + Server.UrlEncode(fi.Name));
                    Response.AddHeader("Content-Length", fi.Length.ToString());
                    Response.ContentType = "application/octet-stream";
                    Response.Filter.Close();
                    Response.WriteFile(fi.FullName);
                    Response.End();
                }

            }
            //捕捉异常
            catch (Exception ex)
            {
                Response.Write("<script language='javascript'>alert('对不起，下载失
                 败！');window.location.href('InfoInquire.aspx')</script>");
             }
        }
        //无上传文件
        else
        {
            Response.Write("<script language='javascript'>alert('对不起，你要下载的
            文件不存在！');window.location.href('InfoInquire.aspx')</script>");
            Server.Transfer("managekucun.aspx");
        }
    }
```

```
public void find()
    {
        Business.ElementManageBN warnSet = new Business.ElementManageBN();
        string ID = Request.QueryString["id"];
        DataSet ds = warnSet.IsExist(ID);
        tbx_address.Text = ds.Tables[0].Rows[0][7].ToString();
        tbx_company.Text = ds.Tables[0].Rows[0][8].ToString();
        tbx_name.Text = ds.Tables[0].Rows[0][2].ToString();
        tbx_xinghao.Text = ds.Tables[0].Rows[0][1].ToString();
    }
    protected void bt_queding_Click(object sender, EventArgs e)
    {

        Business.ElementManageBN warnSet = new Business.ElementManageBN();
        string ID = Request.QueryString["id"];
        string Gaoxian, Dixian;
        Gaoxian = this.tbx_gaoxian.Text;
        Dixian = this.tbx_dixian.Text;
        if (warnSet.WarnUpdate(Gaoxian, Dixian ,ID))
        {
                Response.Write("<script language='javascript'>alert('更改成功!
                ');window.location.href('InfoInquire.aspx')</script>");
        }
        else
        {
                Response.Write("<script language='javascript'>alert('更改失败!
                ');window.location.href('InfoInquire.aspx')</script>");
        }

    }
    protected void bt_quxiao_Click(object sender, EventArgs e)
    {
        Response.Redirect("InfoInquire.aspx");
    }
```