# 《WEB应用综合课程设计》

# 项目报告

**项目名称：仓库管理系统**

**班 级：**22网络1

**学 号：**2022146101

**姓 名：\_\_\_\_\_**齐智暄**\_\_\_\_\_\_\_\_\_\_**

**报告成绩：**

2024年 12 月 19日

**项目名称**

**1. 实训目的**

在现代软件开发中，大量应用程序都依赖于数据驱动的架构，这使得数据库技术成为不可或缺的一部分。无论是企业级应用，还是互联网产品，数据库都承载着重要的功能——存储、管理和检索数据。没有强大的数据库支持，应用程序将无法高效地运行和提供服务。因此，掌握数据库的基本原理和应用至关重要。

通过结合具体的案例，将能够掌握如何设计和建立一个适用于Web应用的数据库。这一过程不仅要求理解数据库结构、关系模型、SQL语句的使用，还要学会如何根据实际需求来优化数据库设计，提高查询效率和数据存储的可靠性。

此外，还能帮助掌握服务器控件的使用。服务器控件是Web开发中常见的组件，通过这些控件，开发者可以更加便捷地处理与数据库的交互，如数据的展示、表单提交和数据验证等。通过实例操作，能够理解如何利用这些控件来提高开发效率，并在实际项目中加以应用。

**2. 实训内容**

我将重点利用SQL Server来构建和管理后台数据库，同时结合具体的Web数据库实例进行操作。我学习了如何设计和创建数据库，包括定义表结构、设置索引、以及编写存储过程和触发器等。此外，我还会了解如何使用ADO.NET数据提供程序访问数据库，实现Web应用中的数据交互。

首先，我将学习在SQL Server中创建和配置数据库，了解如何设计合适的数据库结构，以确保数据的高效存储和查询性能。通过实际的案例，我将能够应用这些知识，构建一个具体的Web数据库应用，掌握如何通过前端提交表单数据到数据库，或者从数据库获取数据并展示到网页上。

在数据库访问方面，我会专注于如何使用ADO.NET，它是一个.NET平台的数据库访问框架，通过ADO.NET的数据提供程序（比如SQL Server提供程序），我能够实现与数据库的高效交互。

此外，我还会通过GitHub来管理我的项目代码和数据库相关文件。在学习过程中，我将将Web项目的代码和数据库设计整合到GitHub仓库中，掌握如何通过版本控制管理数据库的SQL脚本和应用代码。GitHub不仅能帮助我跟踪数据库和代码的版本变化，还可以让我与团队成员更好地协作。通过GitHub，我还可以管理数据库的迁移脚本和配置文件，确保数据库架构与应用代码的一致性，避免在不同开发阶段出现版本不匹配的问题。

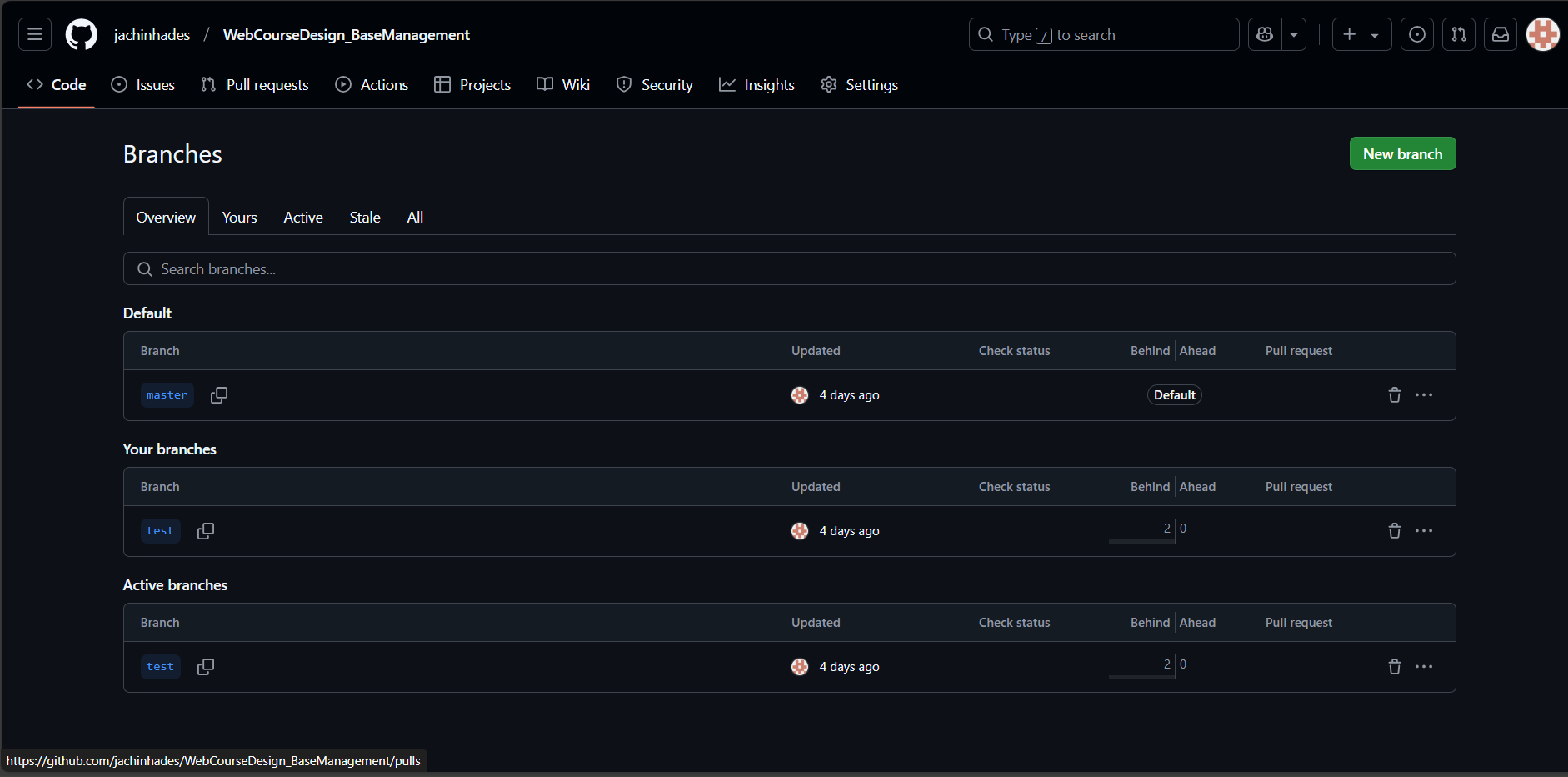
通过这一系列的学习与实践，我将能够全面理解Web数据库开发的全过程，提升自己的开发效率，并能够有效地管理和协作数据库与代码的版本控制。

**2.1 github准备工作**

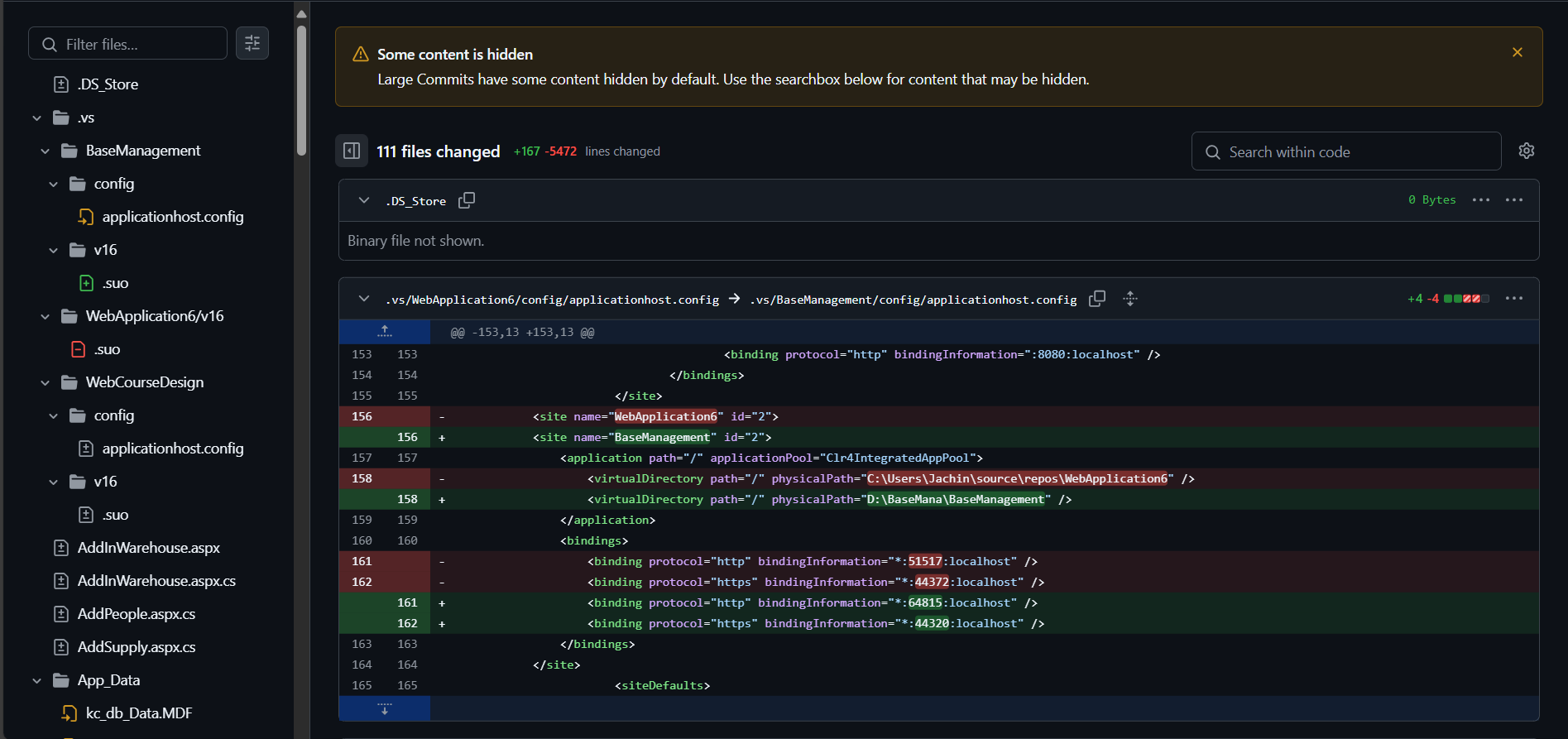
2.1.1建立了如图所示的github仓库

|  |  |
| --- | --- |
|  |  |

2.1.2建立test分支



2.1.3每次修改文件后将文件push到gihub仓库上进行版本控制



2.2 业务逻辑功能的描述

我负责的业务逻辑功能涉及四个主要操作对象：收货单位、供货单位、仓库和经手人。对于每个操作对象，都有两个基本操作：添加和管理（包括修改和删除）。下面是各个操作对象的业务逻辑功能描述：

2.2.1 收货单位（Receiver Unit）

添加功能：用户可以通过填写表单来新增一个收货单位。表单中的字段包括收货单位的名称、联系方式、地址等。系统会对输入数据进行验证，确保数据格式正确且无遗漏，如收货单位名称必须唯一，联系方式符合规定的格式。

管理功能：管理操作包括修改和删除收货单位。

修改：用户可以选择一个现有的收货单位进行修改，修改后的信息会更新到数据库中。系统需要确保在修改时，原数据的完整性不受影响，例如，修改收货单位名称时，要确保不会和其他单位的名称重复。

删除：用户可以删除某个收货单位。在删除前，系统会进行检查，确保该收货单位没有关联到任何未处理的订单或入库记录，否则无法删除。删除操作应具备确认机制，并提供软删除功能，便于恢复删除的数据。

2.2.2. 供货单位（Supplier Unit）

添加功能：通过类似收货单位的添加流程，用户可以新增供货单位信息，填入供货单位名称、联系方式、地址等数据。系统会进行重复性检查，例如防止重复添加相同名称的供货单位。

管理功能：包括修改和删除供货单位。

修改：允许用户对供货单位的基本信息进行修改，如更新联系方式、地址或负责人等。修改时系统会验证修改的数据是否合法，确保供货单位在修改后依然符合业务需求。

删除：在删除供货单位时，系统会首先检查该单位是否存在未完成的供货订单。如果存在未完成的订单，系统将禁止删除，或者提供相应提示。删除时会有确认步骤，并且实施软删除功能。

2.2.3. 仓库（Warehouse）

添加功能：用户可以新增仓库，填写仓库名称、位置、库存容量等信息。新增仓库时，系统会确保仓库名称的唯一性，且位置或容量信息符合实际要求。

管理功能：包括修改和删除仓库。

修改：仓库的修改操作主要涉及仓库信息（如位置、库存容量等）的更新。在修改过程中，系统会进行校验，确保修改后的仓库信息符合实际要求。

删除：删除仓库时，系统需要检查该仓库是否还有存货或未处理的库存记录。如果仓库中还有物品或待处理数据，系统会禁止删除。删除时系统会提供确认提示，并考虑软删除的功能，以防误操作。

2.2.4. 经手人（Handler）

添加功能：系统允许添加经手人（即负责操作和管理仓库的人员）。输入内容包括经手人的姓名、联系方式、职务等信息。系统会验证经手人信息的完整性和正确性。

管理功能：经手人的管理功能同样包括修改和删除操作。

修改：用户可以对经手人的基本信息进行修改，如联系方式、职务变动等。系统会确保修改后的数据完整且合法。

删除：删除经手人时，系统会检查该经手人是否有正在处理的任务或记录。如果该经手人负责的事务还未完成，系统会提示无法删除。删除时，用户需要确认操作，并可能需要将其相关任务转交给其他人员。

2.2.5业务逻辑整合

数据验证：对于添加和修改操作，所有输入的数据都需要进行严格的验证，如字段不能为空、格式必须符合要求等。系统还需要检查数据的唯一性，例如收货单位和供货单位的名称、仓库的ID等要确保不重复。

操作权限控制：系统需要根据用户的权限进行操作限制，例如，普通用户只能查看和修改自己权限内的数据，而管理员可以对所有操作对象进行添加、修改和删除。

数据一致性和事务管理：在涉及多个对象和数据表的操作中，如修改仓库与经手人信息时，系统需要确保数据的一致性和事务的原子性。例如，修改操作和删除操作通常需要在数据库中进行事务管理，确保一旦操作失败，所有变动都可以回滚。

日志记录与审计：所有的管理操作（修改、删除）都需要记录在系统日志中，以便后续审计和问题追溯。日志中应包括操作时间、操作用户、操作对象及具体内容等。

2.3 功能实现

以供货单位为例分析实现方式

1. 添加收货单位功能 - AddSupply.aspx.cs

业务需求： 用户需要输入供货单位的基本信息（编号、名称、地址、电话、负责人），然后将这些信息保存到数据库中。

实现过程：

Page\_Load:

该方法是 ASP.NET 页面生命周期的一部分，通常用于初始化页面内容或进行一次性操作。在此处，Page\_Load 方法并没有做复杂操作，通常可以为空，除非需要加载初始数据或进行页面状态恢复。

Button1\_Click:

用户点击“添加”按钮时，触发该事件进行添加操作。

字段验证 (IsNull):

首先调用 IsNull() 方法，检查 TextBox1（供货单位编号）和 TextBox2（供货单位名称）是否为空。如果其中任一字段为空，则弹出相应的提示框并返回 false，阻止继续执行。

IsNull() 方法会对每个必填字段进行简单的空值检查，若有字段未填，则通过 Response.Write 弹出 JavaScript 警告框提示用户。

插入数据库:

如果输入有效，接下来从页面中的文本框获取用户输入的数据（编号、名称、地址、电话、负责人）。

使用 SqlConnection 类与数据库建立连接：

SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["ConnectionString"]);

其中，ConfigurationManager.AppSettings["ConnectionString"] 会从配置文件中读取数据库连接字符串。

打开数据库连接：

con.Open();

创建 SqlCommand 对象执行 SQL 插入操作：

SqlCommand cmd = new SqlCommand("insert into SupplyUnit(Number,Name,Address,tel,person) values ('" + number + "','" + name + "','" + address + "','" + tel + "','" + person + "')", con);

这里的 SQL 语句是一个简单的插入语句，将用户输入的数据插入到 SupplyUnit 表中的相应字段。

执行插入操作：

cmd.ExecuteNonQuery();

ExecuteNonQuery() 用于执行 SQL 语句且不会返回结果集，通常用于数据的增、删、改操作。

关闭连接：

con.Close();

插入成功后，弹出提示框，提醒用户“供货单位记录添加成功”：

Response.Write("<script language='javascript'>alert('供货单位记录添加成功！');</script>");

使用 Server.Transfer 方法跳转到 ManagerSupplyUnits.aspx 页面，显示供货单位列表：

Server.Transfer("ManagerSupplyUnits.aspx");

2. 显示供货单位列表功能 - ManagerSupplyUnits.aspx.cs

业务需求： 显示供货单位列表，并提供分页和删除功能。

实现过程：

Page\_Load:

页面加载时，调用 Bind() 方法从数据库中获取所有的供货单位数据，并绑定到 GridView 控件上。该方法只有在页面首次加载时执行（!IsPostBack），防止重复绑定。

Bind:

执行 SQL 查询语句，获取所有的供货单位数据：

string s = "select \* from SupplyUnit order by GId desc";

这里的查询语句会按 GId（供货单位的唯一标识符）降序排列，以显示最近添加的供货单位。

使用 SqlDataAdapter 获取数据并填充到 DataSet 中：

SqlDataAdapter da = new SqlDataAdapter(s, con);

DataSet ds = new DataSet();

da.Fill(ds);

将获取的数据绑定到 GridView 控件：

GridView1.DataSource = ds.Tables[0].DefaultView;

GridView1.DataBind();

分页 (GridView1\_PageIndexChanging1):

当用户在 GridView 控件中分页时，PageIndex 会更新为新的页码，并重新绑定数据。

删除操作 (GridView1\_RowDeleting):

用户点击删除按钮时，通过 GridViewDeleteEventArgs 获取被删除项的 GId。

执行 SQL 删除操作：

string s = "delete from SupplyUnit where GId='" + this.GridView1.DataKeys[e.RowIndex].Value.ToString() + "'";

删除成功后，重新绑定数据来更新页面内容。

3. 修改供货单位功能 - UpdateBuyUnit.aspx.cs

业务需求： 用户可以选择一个供货单位进行修改，并保存修改后的信息。

实现过程：

Page\_Load:

获取 URL 中的 ID 参数，并通过 Bind() 方法从数据库中加载该供货单位的详细信息：

id = Request["ID"].ToString();

该 ID 是供货单位的唯一标识符，通常是通过链接或按钮传递过来的。

Bind() 方法根据传递的 ID 参数从数据库中查询供货单位信息，并填充到页面中的各个 TextBox 控件。

string str = "select \* from SupplyUnit where GId='" + ID + "'";

SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["ConnectionString"]);

SqlCommand cmd = new SqlCommand(str, con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

Button1\_Click:

点击“修改”按钮时，首先调用 IsNull() 方法验证输入的有效性。

获取修改后的数据，并执行 SQL 更新操作：

SqlCommand cmd = new SqlCommand("UPDATE SupplyUnit SET Number=@number, Address=@address, Name=@name, person=@person, tel=@tel WHERE GId=@id", con);

cmd.Parameters.AddWithValue("@number", number);

cmd.Parameters.AddWithValue("@name", name);

cmd.Parameters.AddWithValue("@address", address);

cmd.Parameters.AddWithValue("@tel", tel);

cmd.Parameters.AddWithValue("@person", person);

cmd.Parameters.AddWithValue("@id", bid);

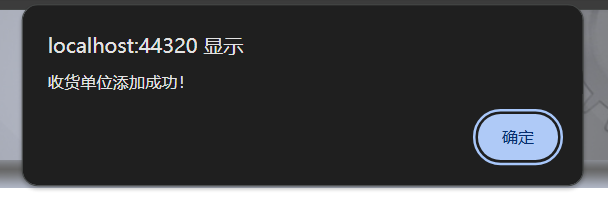
IsNull:

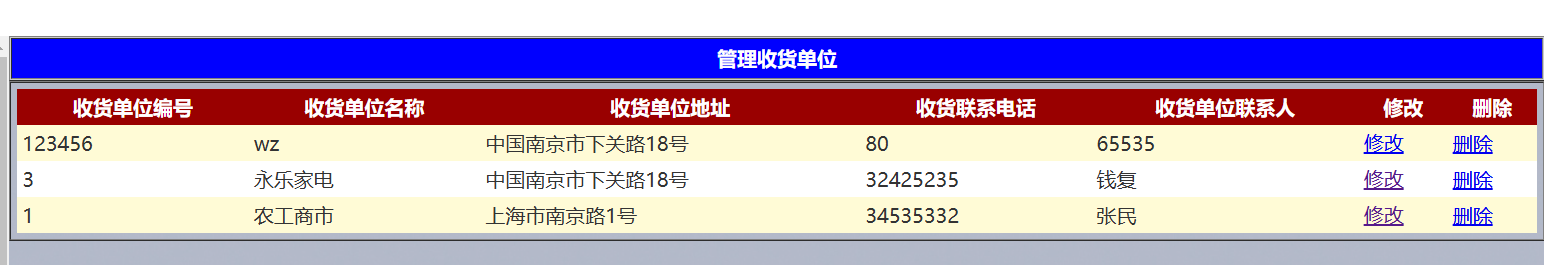
输入验证与添加功能类似，确保关键字段（如编号和名称）不为空。

实现了一个基本的供货单位管理功能，包括添加、显示、修改和删除操作。其核心技术是使用 SqlConnection 和 SqlCommand 来执行 SQL 查询和命令。

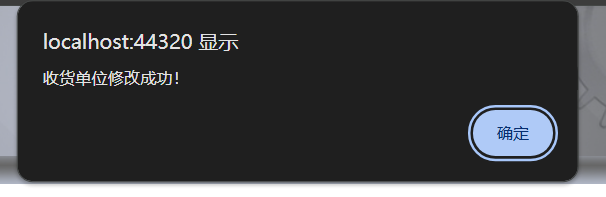
**页面效果：**





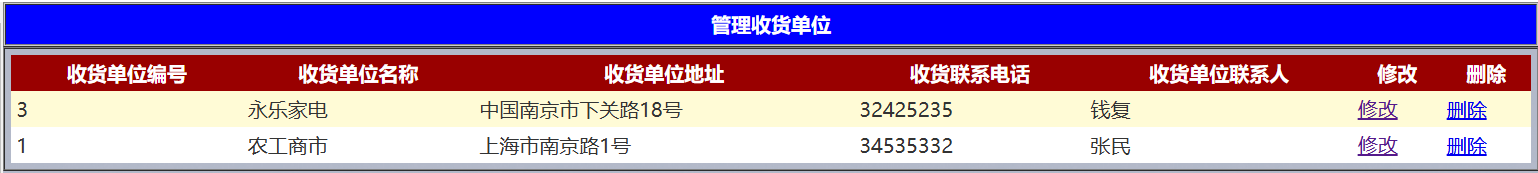












3.小结与体会

在本次对仓库管理系统中管理功能的分析和实现过程中，我深入理解并实践了多个重要的开发技能，特别是在 ASP.NET Web 应用程序的后端数据库操作方面。在这个过程中，我不仅学到了如何通过 SQL 语句与数据库进行交互，还掌握了如何在 Web 页面上实现增、删、改、查等基础操作，同时也深刻体会到了编码过程中细节的重要性和潜在的安全风险。

1. 数据库交互与 SQL 操作

在实现添加、修改、删除功能时，我通过 SqlConnection 和 SqlCommand 类与 SQL Server 数据库进行交互，成功完成了用户输入的供货单位数据的插入、更新和删除操作。通过这些操作，我对 SQL 查询、插入、更新以及参数化查询有了更加深入的理解。

插入操作：我通过构造 SQL INSERT 语句，将用户输入的数据保存到数据库中。在实际开发中，参数化查询能够有效防止 SQL 注入漏洞，这是非常关键的安全措施。

更新操作：在修改操作中，利用了 SqlCommand 的 UPDATE 语句更新了特定 GId 的记录。这让我认识到 SQL 更新操作不仅仅是改变表中的数据，还需要对用户输入进行有效验证，确保数据的合法性。

删除操作：对于删除操作，我掌握了如何在 GridView 控件中实现数据的删除，通过获取相应的 GId 值并执行 SQL DELETE 语句。

2. 页面数据展示与交互

通过对 GridView 控件的使用，我学会了如何在页面中展示数据库中的多条记录，并通过分页功能优化用户体验。尤其是在 ManagerSupplyUnits.aspx 页面中，采用了 SqlDataAdapter 将查询结果绑定到 GridView 控件，动态展示供货单位信息。分页功能的实现也让我更好地理解了如何通过 PageIndex 属性和 GridView 的 PageIndexChanging 事件来处理大量数据的展示问题。

3. 表单验证与用户体验

在所有的表单操作中，用户输入的验证是一个非常重要的环节。通过 IsNull() 方法，我保证了用户在提交数据前填写了所有必要的字段。这一过程虽然简单，但却是确保系统数据完整性和防止错误数据进入系统的关键步骤。同时，表单验证也能有效提高用户体验，避免无效或错误的提交。

4. 安全性问题的认识

在代码中，我意识到 SQL 注入是一个潜在的安全风险，尤其是使用字符串拼接 SQL 语句时。这让我深刻体会到参数化查询的重要性，使用参数化查询不仅能够防止 SQL 注入攻击，还能提高代码的可维护性和可读性。通过这个实践，我进一步认识到在开发过程中始终需要考虑安全性，尤其是涉及到数据库操作时。

5. 代码优化与重构的必要性

在实际的项目开发中，代码的复用性和可维护性非常重要。在这段代码中，插入、更新、删除等数据库操作中存在一定的重复代码。通过重构，将数据库操作提取到一个独立的方法中，可以减少代码的重复性，并提升代码的模块化和可维护性。此外，通过合适的异常处理来提升系统的健壮性，能够帮助开发者及时捕获和处理潜在的错误，避免系统崩溃或数据丢失。

6. 实践中的不足与改进

在这次实践中，我也遇到了一些问题和挑战。例如，初期没有考虑到 SQL 注入的风险，导致代码中存在安全隐患。另外，虽然完成了基本功能的实现，但在性能和用户体验上仍有提升空间。例如，数据库查询的优化和前端分页的增强，都会显著提升系统的性能和可扩展性。

改进建议：

使用参数化查询：防止 SQL 注入攻击，确保数据库操作的安全性。

增强表单验证：加入更多字段的验证，使用正则表达式验证输入的正确性（如电话、邮箱等格式）。

分页优化：使用 SQL 语句的 OFFSET 和 FETCH 子句进行更高效的分页处理，避免一次性将大量数据加载到内存中。

代码重构：将重复的数据库操作提取成公共方法，减少冗余代码，提高代码的可维护性。

异常处理：加入适当的异常处理，防止因数据库连接失败等问题导致程序崩溃，并向用户提供友好的错误提示。

通过此次对仓库管理系统中供货单位管理模块的开发与实现，我深刻体会到了数据库操作的复杂性和细节之处，特别是在安全性、性能优化和用户体验方面的挑战。在今后的开发中，我将继续提升自己的技术能力，注重代码的可读性和可维护性，并时刻关注安全性和性能的提升。