项目 3

连接数据库

随着网络的日益普及，数据越来越多地被存储在网络的数据库服务器中，应用程序对数据库的访问和操作成了一种极为普遍的事情。在本项目中，我们将学习使用 C#连接数据库，实现 ExamSystem 系统的登录问题。

工作任务

- 完成 ExamSystem 系统的登录功能。
- 连接 ExamSystemDB 数据库。
- 查询用户名和密码是否存在。

技能目标

- 了解 ADO.NET 的功能和组成。
- 学会使用 Connection 对象连接数据库。
- 学会使用 try…catch 捕捉异常。
- 学会使用 SqlCommand 对象查询单个值。

预习作业

- ADO.NET 的主要组件有哪些?
- 简述 SqlConnection 与 SqlCommand 对象的作用。
- try…catch 语句的语法格式是什么?
- SQL Server 2008 如何创建数据库?

任务 3.1　连接 ExamSystemDB 数据库

任务描述

在 ExamSystem 系统登录界面单击“连接测试”按钮，如果成功连接数据库，则提示“打开数据库连接成功!”，随后执行关闭数据库连接，提示“关闭数据库连接成功!”。需要注意的是，并不是单击“连接测试”按钮给出相应提示，而是要进行数据库的连接和关闭之后再给出相应提示。

任务分析

其实数据库的操作和我们日常生活中的某些活动非常相似，比如我们可以把图书馆看作一个存放图书的数据库，想象一下：如果我们要去图书馆借书或还书需要几个步骤？在本项目中，首先创建一个字符串，用于描述要对哪个数据库服务器上的哪个数据库进行操作，然后建立数据库连接，最后关闭数据库连接，在打开数据库连接和关闭数据库连接时给出相应的提示即可完成任务。

3.1.1 任务实现代码及说明

实现步骤如下。

（1）建立项目之前先准备一个 SQL Server 2008 数据库，如图 3-1 所示。

图 3-1　准备 SQL Server 2008 数据库

（2）新建一个 Windows 项目，项目名称为 ExamSystem。

（3）在 Windows 项目中新建一个窗体，窗体名称为 ConnectionTestForm。

（4）修改 Program.cs 文件中 Main()方法的代码，使得打开的第一个窗体为 ConnectionTestForm，代码如下：

```
static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        //设置首先运行的 ConnectionTestForm 窗体
        Application.Run(new ConnectionTestForm());
    }
```

（5）在 ConnectionTestForm 窗体中添加引用 SQL Server 的命名空间，代码如下：

```
using System.Data.SqlClient;
```

（6）在 ConnectionTestForm 窗体中添加一个“连接测试”按钮，名称为 btnConTest。

（7）在“连接测试”按钮单击事件中添加如下代码：

```
    private void btnConTest_Click(object sender, EventArgs e)
    {
        // 数据库连接字符串，字符串的内容要根据读者自己电脑的实际情况进行编写
        string strCon= "Data Source=STAR\\SQLEXPRESS;Initial Catalog= ExamSystemDB;
User ID=sa;Pwd=123";
       //创建数据库连接对象
        SqlConnection conn = new SqlConnection(strCon);
        //打开数据库连接
        conn.Open();
        MessageBox.Show("打开数据库连接成功！");
        //关闭数据库连接
        conn.Close();
        MessageBox.Show("关闭数据库连接成功！");
    }
```

小贴士

为了可以正常使用.NET Framework 数据提供程序，用户一定要添加引用命名空间的语句“using System.Data.SqlClient;”。如果没有该语句，则用户将无法使用.NET Framework 数据提供程序。

实际上，Windows 已经帮我们设计好了关于数据库固定的一些操作，现在我们需要使用数据库对象的属性或方法来编写程序达到特定功能，这时需要引入一个数据库的命名空间。换句话说，我们要对数据库进行操作，但是实际上并不知道具体每一步是如何对数据进行操作的，我们是通过微软设计的数据提供程序来对数据库进行具体操作的。

上面程序的运行结果如图 3-2 所示。

图 3-2　运行结果

3.1.2　ADO.NET 概述

有一个普遍现象，现在人们外出旅行到了酒店最先问的问题不是住宿费用是多少元，而是 Wi-Fi 密码是什么，这从侧面说明了人们对于网络的需求，也正是因为网络的迅速发展，有越

来越多的应用程序连接了网络，所以应用程序要处理的大量数据存储在网络服务器中，也就是网络数据库中。

目前常用的数据库有很多种，如 SQL Server、MySQL、Oracle 等。为了使客户端能够访问数据库服务器上的数据，就需要使用数据库访问的方法和技术，ADO.NET 就是一种常见的技术。

ADO.NET 的名称来源于 ADO（ActiveX Data Objects），它是一个 COM 组件库，用于在以往的 Microsoft 技术中访问数据。之所以命名为 ADO.NET，是因为 Microsoft 希望表明，这是在.NET 编程环境中优先使用的数据访问接口。

通过 ADO.NET，应用程序可以很方便地访问和操作数据库。ADO.NET 的功能非常强大，它提供了对关系数据库、XML 及其他数据存储的访问。应用程序可以通过 ADO.NET 技术与这些数据源进行连接，对数据进行增加、删除、修改、查询等操作。

ADO.NET 技术有一个非常大的优点，即断开式连接，也就说当它与数据源断开连接时也可以使用数据。这是怎么做到的呢？以从水源中取水为例进行说明。水源就相当于应用程序中的数据库，这里有大量的数据（水）供我们使用。如果直接从水源中取水这就是传统的数据库编程方法，要求应用程序必须时刻与数据库保持连接，一旦断开必然发生错误。而 ADO.NET 技术可以先把应用程序需要的数据（水）存储在 DataSet（蓄水池）中，当应用程序需要访问和操作数据时可以对 DataSet 进行访问和操作，此时不必保持应用程序与数据库的连接，只要保证临时数据库 DataSet 中有数据即可。

3.1.3 ADO.NET 的组件

ADO.NET 提供了两个组件来访问和处理数据：一个是.NET Framework 数据提供程序（.NET Framework Data Provider）；另一个是 DataSet 数据集。ADO.NET 访问数据库模型如图 3-3 所示。

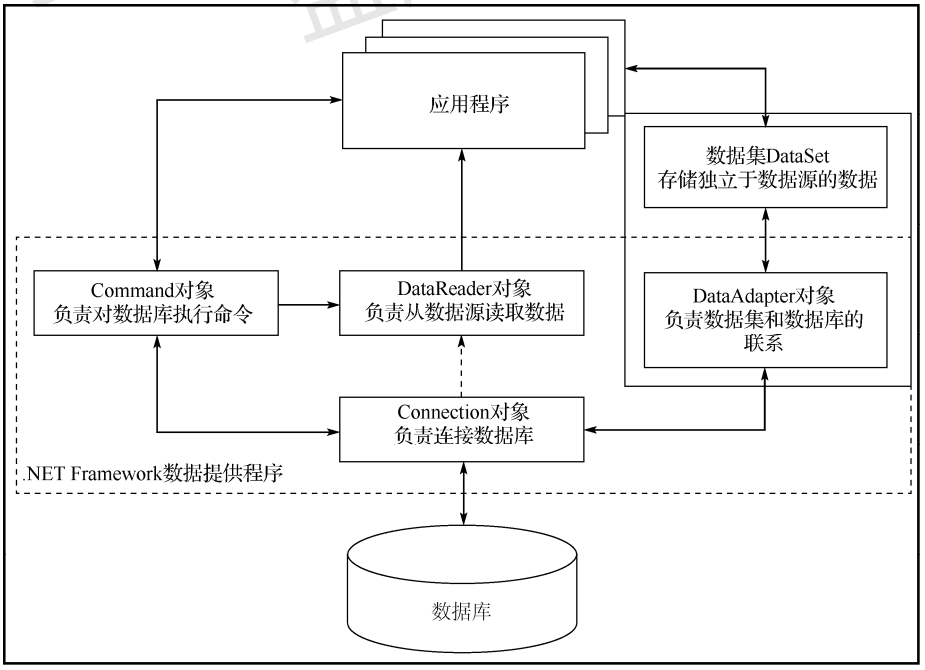

图 3-3　ADO.NET 访问数据库模型

- .NET Framework 数据提供程序提供对关系数据源中的数据的访问。.NET Framework 数据提供程序包含一些类，这些类用于连接数据源，在数据源处执行命令，返回数据源的查询结果；.NET Framework 数据提供程序还能执行事务内部的命令。
- DataSet 是 ADO.NET 的中心概念。可以把 DataSet 当成内存中的数据库，DataSet 是不依赖于数据库的独立数据集合。所谓独立，就是说即使断开数据链路，或者关闭数据库，DataSet 依然是可用的。DataSet 在内部是用 XML 来描述数据的，因为 XML 是一种与平台无关、与语言无关的数据描述语言，而且可以描述复杂关系的数据，如父子关系的数据，所以 DataSet 可以容纳具有复杂关系的数据，而且不再依赖数据链路。在具体使用中可以把对 DataSet 的操作当成是对数据库的操作，其作用相当于前文介绍的蓄水池。

.NET Framework 数据提供程序包含了访问各种数据源的对象，它和数据库类型相关。目前，.NET Framework 数据提供程序支持的提供程序如表 3-1 所示。

表 3-1　.NET Framework 数据提供程序支持的提供程序

.NET Framework 数据提供程序	说　明
.NET Framework 用于 SQL Server 的数据提供程序	提供对 Microsoft SQL Server 的数据访问，使用 System.Data.SqlClient 命名空间
.NET Framework 用于 Oracle 的数据提供程序	适用于 Oracle 数据源，支持 Oracle 客户端软件 8.1.7 或更高版本，使用 System.Data.OracleClient 命名空间
.NET Framework 用于 OLE DB 的数据提供程序	提供对 OLE DB 公开数据源中数据的访问，使用 System.Data.OleDb 命名空间
.NET Framework 用于 ODBC 的数据提供程序	提供对 ODBC 公开数据源中数据的访问，使用 System.Data.Odbc 命名空间
EntityClient 提供程序	提供对实体数据模型（EDM）应用程序的数据访问，使用 System.Data.EntityClient 命名空间

在实际应用中使用哪种数据提供程序，要看应用程序所使用的数据源。Microsoft 的 C# 对 SQL Server 数据库的支持更加稳定和可靠。下面采用 SQL Server 数据库作为应用程序的数据源。

.NET Framework 数据提供程序包含 4 个核心对象，即 Connection、Command、DataReader 和 DataAdapter，如表 3-2 所示。

表 3-2　.NET Framework 数据提供程序的 4 个核心对象

对　象	说　明
Connection	建立与特定数据源的连接
Command	对数据源执行命令
DataReader	从数据源中读取只读的数据流
DataAdapter	用数据源填充 DataSet 并解析更新

在不同的命名空间中都有相应的对象，例如，要操作 SQL Server 数据库，需要使用 System.Data.SqlClient 命名空间，SQL 数据提供程序中的类都以“Sql”开头，所以它的 4 个核心对象分别为 SqlConnection、SqlCommand、SqlDataReader 和 SqlDataAdapter。

3.1.4 使用 Connection 对象

Connection 对象是.NET Framework 数据提供程序的核心对象之一，起到了应用程序与数据库之间建立连接的作用。正如桥梁可以把两岸连接起来，应用程序与数据库之间想要进行数据传递也需要一个类似桥梁的东西把两者连接起来，Connection 对象就起到了这样的作用。

在不同的.NET Framework 数据提供程序中都有自己的连接类，如表 3-3 所示。应用程序可以根据自己所使用的数据库类型来确定使用哪个连接类。

表 3-3 .NET Framework 数据提供程序及相应的连接类

.NET Framework 数据提供程序	连 接 类	命 名 空 间
SQL 数据提供程序	SqlConnection	System.Data.SqlClient
Oracle 数据提供程序	OracleConnection	System.Data.OracleClient
OLE DB 数据提供程序	OleDbConnection	System.Data.OleDb
ODBC 数据提供程序	OdbcConnection	System.Data.Odbc

为了建立应用程序与数据库的连接，Connection 对象提供了一些属性和方法。Connection 对象的常用属性和方法如表 3-4 和表 3-5 所示。

表 3-4 Connection 对象的常用属性

属 性	说 明
ConnectionString	设置/获取应用程序连接数据库的连接字符串

表 3-5 Connection 对象的常用方法

方 法	说 明
void Open()	打开数据库连接
void Close()	关闭数据库连接

建立应用程序与数据库的连接需要按照下面 3 个步骤完成。

1．定义连接字符串

SQL Server 数据库连接字符串的语法格式如下：

```
Data source=服务器名;Initial Catalog=数据库名;User ID=用户名;Pwd=密码
```

- Data source：指定与应用程序连接的数据库服务器的名称或 IP 地址。如果本机作为应用程序的服务器，则参数的值可以是“.”、“(local)”或“127.0.0.1”。建议使用数据库服务器的名称作为参数的值，防止本机有两个以上数据库服务器时引起应用程序无法正确识别数据库服务器的情况。
- Initial Catalog：指定应用程序访问数据库服务器中的数据库的名称。
- User ID：登录 SQL Server 数据库的用户名。
- Pwd：登录 SQL Server 数据库的密码，如果密码为空，则可以省略此项。

例如，应用程序与本机的 ExamSystemDB 数据库连接字符串，代码如下：

```
string connectionString = "Data Source=STAR-PC\\SQLEXPRESSS;Initial Catalog=
ExamSystemDB;User ID=sa;Pwd=123456";
```

2．创建 Connection 对象

使用定义好的连接字符串创建 Connection 对象，语法格式如下：

```
SqlConnection conn = new SqlConnection(connectionString);
```

3．打开数据库连接

使用 Connection 对象的 Open()方法打开数据库连接，语法格式如下：

```
conn.Open();
```

连接字符串比较长，又是写在引号之中的，无法利用智能提示，那么我们怎么才能记住里面的关键字呢？其实我们可以利用 Visual Studio 中的服务器资源管理器来获得连接字符串，实现步骤如下。

（1）在 Visual Studio 中选择菜单栏中的“视图”→“服务器资源管理器”命令，如图 3-4 所示；或者按 Ctrl+Alt+S 组合键。

（2）在打开的“服务器资源管理器”窗口中，右击“数据连接”选项，在弹出的快捷菜单中选择“添加连接”命令，如图 3-5 所示。

图 3-4　选择“服务器资源管理器”命令

图 3-5　选择“添加连接”命令

（3）在弹出的“添加连接”对话框中，选择数据源，输入服务器名称，选择身份验证与要连接的数据库，即可在“服务器资源管理器”窗口中添加一个数据库连接。

（4）选中新添加的数据库连接，在“属性”窗口中就能找到连接字符串了，可以将它复制到代码中，如图 3-6 所示。

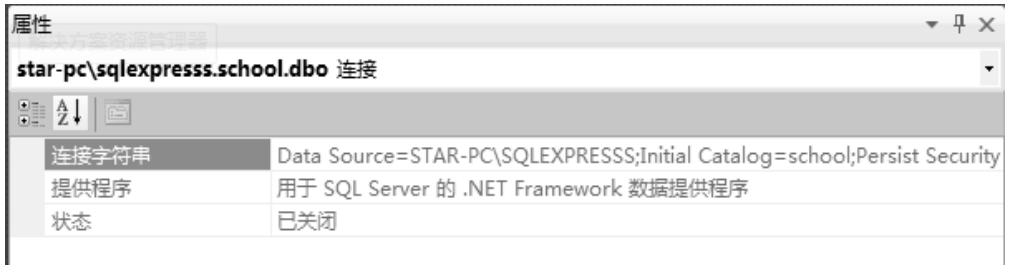

图 3-6　“属性”窗口中的连接字符串

（5）修改连接字符串，如用户名和密码，以达到符合应用程序的要求。

（6）打开数据库连接，执行命令后，要记得关闭数据库连接。

3.1.5 使用 sa 用户登录 SQL Server 数据库

在默认情况下，我们使用 Windows 身份验证的方式登录 SQL Server 数据库，但是在实际项目开发过程中这样做非常不安全，因此我们通常会使用 SQL Server 身份验证的方式登录和操作数据库。实现步骤如下。

（1）在“连接到服务器”对话框中，分别设置服务器类型、服务器名称、身份验证，使用 Windows 身份验证的方式登录 SQL Server 数据库，如图 3-7 所示。

图 3-7 “连接到服务器”对话框

（2）在“对象资源管理器”窗口中，选择“安全性”→“登录名”→“sa”选项，右击，在弹出的快捷菜单中选择“属性”命令，如图 3-8 所示，打开“属性”窗口。

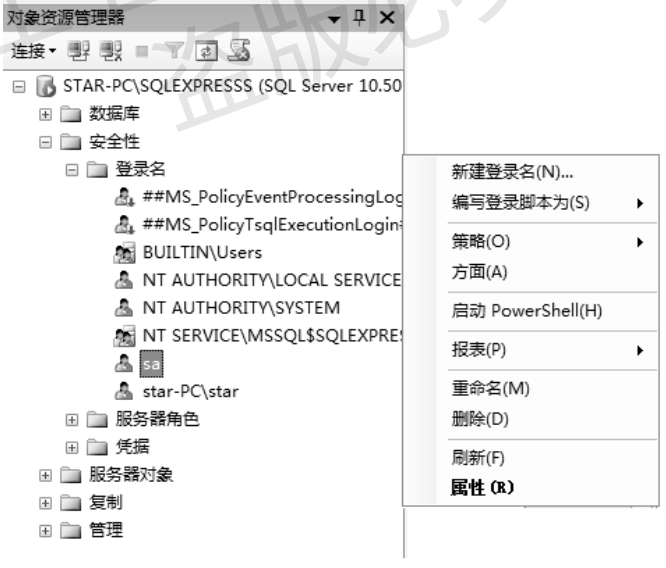

图 3-8 选择“属性”命令（1）

（3）在“属性”窗口修改并设置 sa 用户的密码，密码自行设置，两次输入的密码保持一致即可，如图 3-9 所示。

（4）在“属性”窗口中单击“状态”选项，设置 sa 用户的“登录”状态为“启用”，如图 3-10 所示。单击“确定”按钮，关闭“属性”窗口。

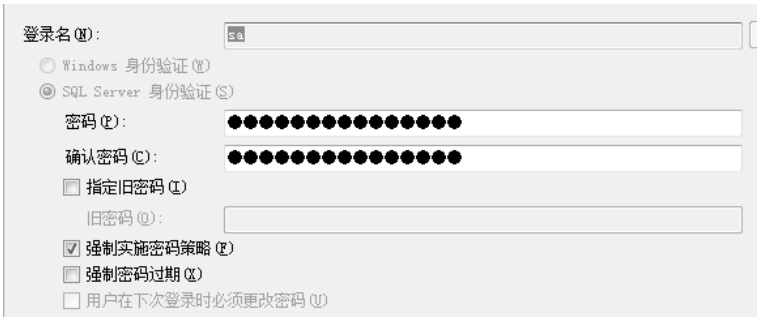

图 3-9　设置 sa 用户的密码

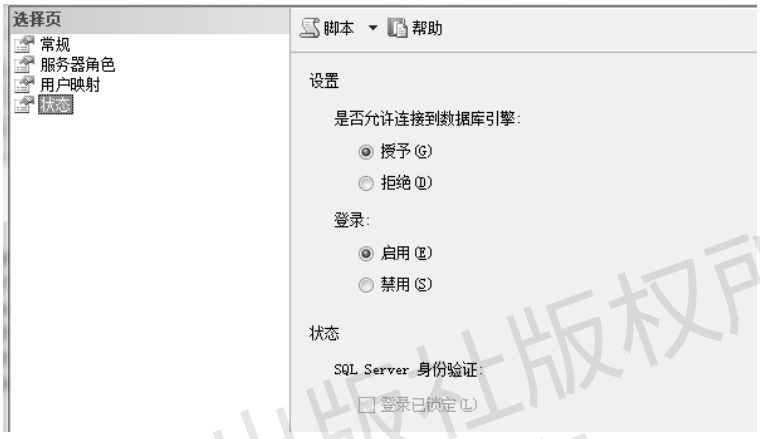

图 3-10　设置 sa 用户的“登录”状态为“启用”

（5）在“对象资源管理器”窗口中，右击服务器选项，在弹出的快捷菜单中选择“属性”命令，如图 3-11 所示，打开“服务器属性-STAR-PC\SQLEXPRESSS”窗口。

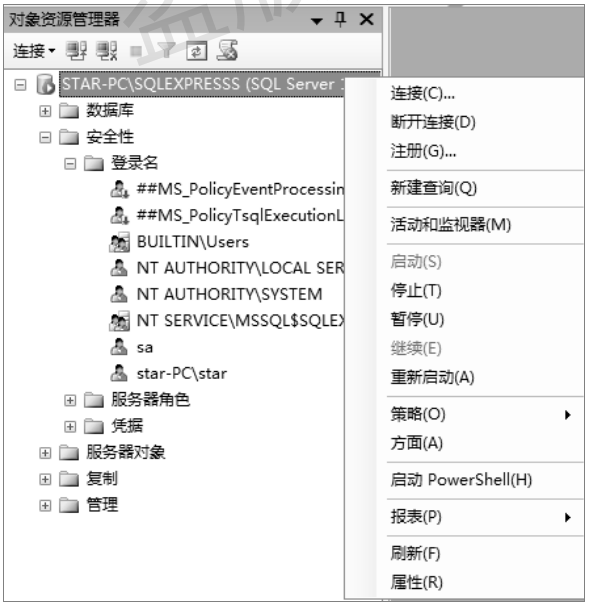

图 3-11　选择“属性”命令（2）

（6）在“安全性”选项卡中，选中“SQL Server 和 Windows 身份验证模式”单选按钮，如图 3-12 所示。单击“确定”按钮，关闭“服务器属性-STAR-PC\SQLEXPRESSS”窗口。

图 3-12　选中“SQL Server 和 Windows 身份验证模式”单选按钮

（7）打开“Microsoft SQL Server Management Studio”消息框，提示“直到重新启动 SQL Server 后，您所做的某些配置更改才生效。”，如图 3-13 所示。

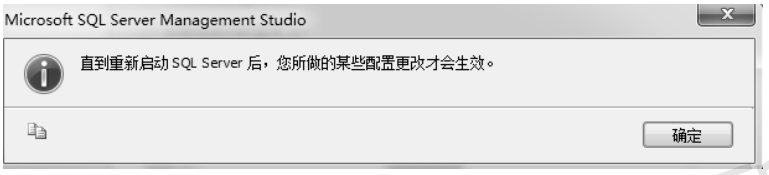

图 3-13　“Microsoft SQL Server Management Studio”消息框

以上操作步骤完成后，我们可以使用 sa 用户及设置的密码登录 SQL Server 数据库检查是否成功。

3.1.6　常见错误与问题

在开始编写数据库相关程序时，初学者很可能会犯如下常见错误。

1．数据库连接字符串中的各参数之间的分隔符拼写错误

下面的代码在运行后，应用程序会提示如图 3-14 所示的错误信息。

```
   string connectionString = "Data Source=STAR-PC\\SQLEXPRESSS,Initial Catalog=
ExamSystemDB,User ID=sa,Pwd=123456";
   //创建数据库连接对象
    SqlConnection conn = new SqlConnection(connectionString);
    //打开数据库连接
    conn.Open();
```

图 3-14　数据库连接字符串中的各参数之间的分隔符拼写错误

造成上面错误的原因是，数据库连接字符串中的各参数之间应该使用“;”而不是“,”。

2．数据库连接字符串中的参数名称拼写错误

下面的代码在运行后，应用程序会提示如图 3-15 所示错误信息。

```
    // 数据库连接字符串
    string connectionString = "DataSource=STAR-PC\\SQLEXPRESSS;Initial Catalog=
ExamSystemDB;User ID=sa;Pwd=123456";
    //创建数据库连接对象
    SqlConnection conn = new SqlConnection(connectionString);
    //打开数据库连接
    conn.Open();
```

图 3-15　数据库连接字符串中的参数名称拼写错误

从上面的错误提示信息中可以发现，数据库连接字符串中的参数“DataSource”不能被编译器识别，将其修改为“Data Source”就可以排除该错误了。

避免上面两种错误发生的方法有两种：第一，认真检查自己书写的连接字符串是否有拼写错误；第二，利用.NET 自动生成连接字符串，再复制和适当修改参数的值。

3．数据库连接字符串中的引号出现位置错误

这类错误在代码进行编译时就会暴露出来，只要读者适当注意就可以避免此类错误。下面的代码在编译时会提示如图 3-16 所示错误信息。

```
    string connectionString = Data Source="STAR-PC\\SQLEXPRESSS;"Initial Catalog=
"ExamSystemDB;"User ID=sa;Pwd="123456";
```

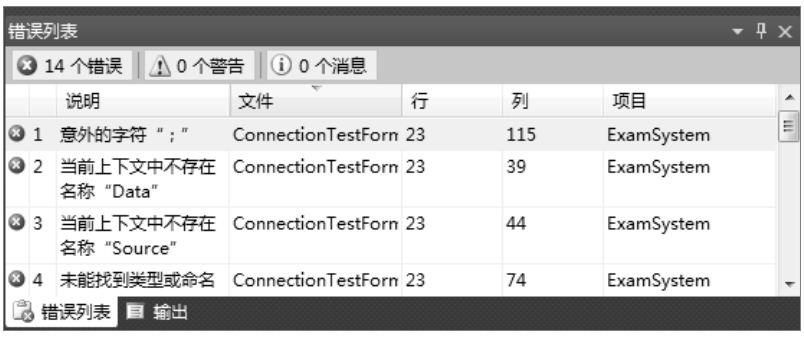

图 3-16　数据库连接字符串中的引号出现位置错误

3.1.7 上机实训

【上机练习 1】

测试 ExamSystemDB 数据库连接。

- 需求说明。
  - 创建 ExamSystem 项目。
  - 创建用于操作 SQL Server 数据库的 DBHelper 类。
  - 在 DBHelper 类中定义测试数据库连接的方法，实现连接和关闭 ExamSystemDB 数据库。

程序运行结果参考图 3-2。

- 实现思路及关键代码。
  - 在创建的 DBHelper 类中打开数据库连接的方法必须设置为 public，否则外部类无法访问 DBHelper 类中的打开数据库连接方法。
  - 在 DBHelper 类中定义数据库操作的两个方法，即数据库连接方法 TestOpenDB()及数据库关闭方法 TestCloseDB()，代码如下：

```
   //数据库连接字符串声明
    string strCon = "Data Source=STAR\\SQLEXPRESS;Initial Catalog=ExamSystemDB;
User ID=sa;Pwd=123";
    SqlConnection conn;
    /// <summary>
    /// 测试 ExamSystemDB 数据库的连接与关闭
    /// </summary>
    public string TestOpenDB()
    {
        // 测试打开数据库的操作
        conn = new SqlConnection(connString);
        // 打开数据库连接
        conn.Open();
        return ("打开数据库连接成功！");
    }
    public string TestCloseDB()
    {
        // 关闭数据库连接
        conn.Close();
        return ("关闭数据库连接成功！");
    }
```

因为我们要得到数据库连接与关闭的信息，所以数据库连接方法 TestOpenDB()及数据库关闭方法 TestCloseDB()具有 string 返回值，这个值传递到调用该方法的窗体中再做进一步处理。

- 删除“连接测试”按钮中的代码，修改后的代码如下：

```
   private void btnConTest_Click(object sender, EventArgs e)
   {
```

```
        DBHelper myDB = new DBHelper();
        MessageBox.Show(myDB.TestOpenDB());
        MessageBox.Show(myDB.TestCloseDB());
    }
```

【上机练习 2】

- 思考和讨论：在什么情况下连接数据库会失败？读者可以主要从程序代码和非程序代码两个方向讨论。

【上机练习 3】

- 思考涉及数据库应用程序的一般编程步骤。
- 如何查找数据库应用程序出现错误的原因？

任务自测表（请在下面记录表中记录自己的学习情况，看看自己掌握的程度）

任务 3.1	看懂本任务的代码	上机练习 1	上机练习 2	上机练习 3

任务 3.2　ExamSystem 系统异常处理

任务描述

在 ExamSystem 系统登录界面单击“登录”按钮，如果出现数据库不存在或无法打开数据库等意外情况，则提醒用户“数据库暂时无法访问”。

任务分析

使用 try…catch…finally 语句对可能发生异常的语句进行处理。

3.2.1　任务实现代码及说明

实现步骤如下。

（1）将任务 3.1 中的代码加入异常处理语句，修改后的代码如下：

```
    private void btnConTest_Click(object sender, EventArgs e)
      {
          // 数据库连接字符串
          string strCon= "Data Source=STAR\\SQLEXPRESS;Initial Catalog= ExamSystemDB;
User ID=sa;Pwd=123";
          //创建数据库连接对象
          SqlConnection conn = new SqlConnection(strCon);
          try
          {
              //打开数据库连接
              conn.Open();
              MessageBox.Show("打开数据库连接成功！");
```

```
        }
        catch (Exception)
        {
            MessageBox.Show("数据库暂时无法访问");
        }
        finally
        {
            //关闭数据库连接
            conn.Close();
            MessageBox.Show("关闭数据库连接成功！");
        }
    }
```

（2）如果我们此时将 SQL Server 数据库的服务器关闭，再次运行程序，当程序试图打开数据库连接时就会出现无法打开数据库连接的异常。但是由于我们做了异常处理，所以程序不会崩溃，而只是根据程序的处理弹出一个“数据库暂时无法访问”消息框，如图 3-17 所示。

图 3-17　“数据库暂时无法访问”消息框

小贴士

在任务 3.1 的基础上快速添加 try…catch…finally 语句的方法如下。

- 微软给我们提供了“外侧代码”功能来添加 try 块，此方法很简单。首先在 Visual Studio 的代码编辑器中选中可能会出现异常的代码，然后右击，在弹出的快捷菜单中选择“外侧代码”命令，如图 3-18 所示。

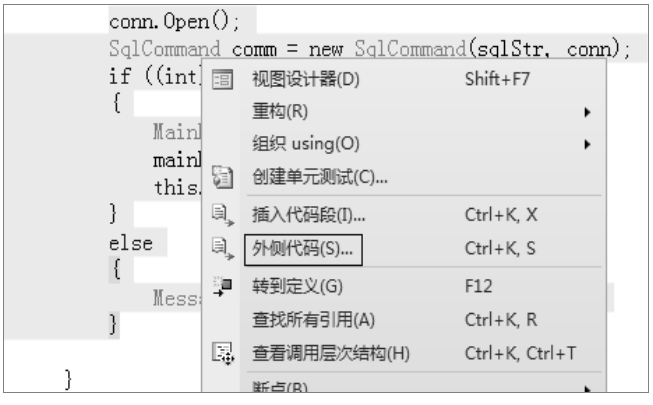

图 3-18　选择“外侧代码”命令

- 在外侧代码编辑器中找到 try 块，如图 3-19 所示，双击 try 块或按两次 Tab 键，代码就会自动添加 try…catch 语句，并且所选中的代码已经被放在 try 块中了，只要增加和修改 catch 块中的代码，添加 finally 块及代码就可以完成任务 3.2。

图 3-19　选择外侧代码 try 块

如果之前是可以正常运行的程序，经过异常处理之后用户可能会发现和之前程序没有太大区别，此时用户可以尝试关闭数据库服务后再次运行程序。对比没有加入异常处理的代码就会发现，整个程序不会产生崩溃的现象，可以说异常处理提供了一个友好的用户交互场景。

3.2.2　什么是异常

在程序设计中，错误通常分为两类，即编译错误和运行错误。编译错误是比较容易发现的，一般在程序运行前就由编译器发现了；而运行错误却很难判断，常常使程序员感到头疼。异常是程序运行中的一种正常的错误，如果异常处理不当，会影响程序的稳定性。

在 C#中，异常是一个在程序执行期间发生的事件，它中断了本该正常执行的程序指令流。在程序开发中，异常产生的范围非常广阔，如编写代码过程、程序调试过程、项目测试过程及用户运行程序过程中，都可能会出现一些不同的异常现象。

虽然程序在运行过程中出现错误是无法避免的，但是可以预知。例如，程序正要对数据库中的数据进行读取，但是网络突然中断了，程序本身无法控制网络是否通畅，程序员却可以预测可能会有这种情况出现。为了保证程序能够正常运行，程序员要对程序运行中可能发生的错误进行编码处理，这就是异常处理。

3.2.3　如何处理异常

在.NET 中我们使用 try…catch 语句捕获和处理异常。

语法格式如下：

```
try
{
   //包含可能会出现异常的代码
}
catch (处理的异常类型)
{
   //处理异常的代码
}
```

try…catch 语句是把可能出现异常的代码放在 try 块中。如果在程序运行过程中发生了异常，则会跳转到 catch 块中进行错误处理，这个过程被称为捕获异常。如果程序在执行过程中没有发生异常，则会正常执行 try 块中的全部语句，不会执行 catch 块中的语句。

异常有很多种类型，本书只关注 Exception 类。这个类是.NET 提供的一个异常类，表示应用程序在运行过程中发生的错误。例如，我们在对数据库进行访问和操作时，很可能会发生不能访问数据的情况（有可能是数据库本身出了问题，也有可能是网络出了问题等），我们就可以把对数据库操作的语句放在 try 块中。

前文也提到过，数据库连接必须显式关闭。但是如果是在数据库连接关闭之前就出现了异常，程序就会跳转到 catch 块中，则 try 块中的数据库连接关闭的方法就不会执行了。这时我们该如何处理呢？其实，这个问题微软早就想到解决的办法了，它提供了一个 finally 块，将代码放在 finally 块中表示无论是否发生异常这部分代码都会执行。我们只需要把关闭数据库连接的语句放在 finally 块中就可以解决这个问题了。无论程序在运行期间是否发生异常，我们可以确保在程序结束运行之前都会关闭数据库连接。代码如下：

```
try
{
   conn.Open();
   //省略其他代码
}
catch (Exception ex)
{
   //处理异常的代码
}
finally
{
    conn.Close();
}
```


3.2.4 上机实训

【上机练习 4】

为 ExamSystem 系统数据库连接操作添加异常处理。

- 需求说明。
  - 在上机练习 1 实现的数据库连接与关闭的基础上增加异常处理。
  - 停止 SQL Server 数据库服务，再次测试数据库连接操作。
  - 改变连接字符串中连接服务器名称，再次测试数据库连接操作。

程序运行结果参考图 3-17。

【上机练习 5】

- 在 DBHelper 类中增加异常处理，使得自己的程序更加稳定。

【上机练习 6】

- 讨论在进行数据库编程时，哪些地方需要进行异常处理。
- 讨论除了数据库编程，还有哪些常见场合需要进行异常处理。

任务自测表（请在下面记录表中记录自己的学习情况，看看自己掌握的程度）

任务 3.2	看懂本任务的代码	上机练习 4	上机练习 5	上机练习 6

任务 3.3　输入用户名及密码登录 ExamSystem 系统

任务描述

在 ExamSystem 系统登录界面输入的用户名及密码通过数据库中的数据判断是否为合法用户，通过检验后登录 ExamSystem 系统，否则提示用户输入错误。

任务分析

登录系统通常是程序的第一步操作，在用户登录系统时我们要对用户输入的用户名及密码进行校验。如果用户输入的数据与之前保存在数据库中的数据一致，则允许用户进行下一步操作，否则我们应该怀疑是否有人正在进行非法访问，此时应该及时给出用户必要的提醒，让用户重新输入用户名及密码。

本任务可以分成以下两个步骤。

- 使用 Connection 对象操作应用程序连接 SQL Server 数据库。
- 将用户的输入数据与数据库中的数据进行比对校验，并提示用户相应的后继操作。

其中第一个步骤在前面的任务中已经完成，现在主要实现第二个步骤。

3.3.1　任务实现代码及说明

根据上面的提示步骤，编码查询 ExamSystemDB 数据库中是否存在输入的用户记录，如果存在，则登录系统，跳转到欢迎窗体；否则提示登录失败。如果数据库发生异常，则提示“数据库暂时无法访问”。实现步骤如下。

（1）命名空间处添加 SQL Server 命名空间，代码如下：

```
using System.Data.SqlClient;
```

（2）“登录”按钮代码如下：

```
private void btnOK_Click(object sender, EventArgs e)
    {
        // 数据库连接字符串
        string strCon =
             "Data Source=STAR\\SQLEXPRESS;Initial Catalog=ExamSystemDB; User
ID=sa;Pwd=123";
        // 创建数据库连接对象
        SqlConnection conn = new SqlConnection(strCon);
        string strSql = string.Format("select count(*) from admin where
```

```
AdminName='{0}' and AdminPwd='{1}'", txtName.Text, txtPwd.Text);
            try
            {
                conn.Open();
                SqlCommand comm = new SqlCommand(strSql, conn);
                if ((int)comm.ExecuteScalar() == 1)
                {
                    MainFrm mainFrm = new MainFrm();
                    mainFrm.Show();//打开管理员主窗体
                    this.Hide();
                }
                else
                {
                        MessageBox.Show("请核对用户名及密码！");
                }
            }
            catch (Exception)
            {
                MessageBox.Show("数据库暂时无法访问" );
            }
            finally
            {
                conn.Close();
            }
        }
```


说明：

```
   string strSql = string.Format("select count(*) from admin where AdminName='{0}'
and AdminPwd='{1}'", txtName.Text, txtPwd.Text);
```

上面的代码通过字符串构造语句实现，因为我们在输入用户名及密码时，需要告诉 SQL Server 数据库用户名及密码两个变量的值，所以 strSql 字符串变量需要使用“+”拼接起来，或者使用此实例中的 string.Format()方法来实现。推荐大家使用 string.Format()方法来实现，使用这种方法不仅修改起来方便，而且程序看起来也会结构清晰，不容易产生拼写错误。

虽然上面的代码不多，尤其是涉及数据库操作的核心代码极少，但是很多人都在此遇到了很大的问题，他们不知道如何根据数据库中的数据进行合法登录系统的判断。下面用尽量通俗易懂的话来描述如何实现上述功能。

代码主要实现根据数据库中的数据及用户在 ExamSystem 系统登录界面输入的数据进行比对，然后判断是否为合法用户（用户输入用户名及密码是否与数据库中保存的用户名和密码一致）。如果是合法用户，则允许登录并打开 MainFrm 窗体；如果不是合法用户，则提示用户输入错误。

因此，基本的数据来源于两个地方：一个地方是之前保存在数据库中的用户名及密码；另一个地方就是用户在 ExamSystem 系统登录界面输入的用户名及密码。用户输入的数据可以直接从文本控件的 Text 属性获取，但是如何获取数据库中保存的数据呢？

实际上在本程序中，我们并未从数据库中获取数据，而是把用户输入的数据放到数据库中进行查找，数据库根据查找的结果返回一个结果而已。听起来好像有点绕，但道理很简单。来做个比喻，假设我们访问一个机密部门，在门口肯定会被门卫拦下，然后告诉门卫已经预约了贵部门的张三，这时门卫就会去部门中询问张三是否有此事，如果部门不存在张三这个人，或者虽然存在张三这个人，但是他说没有预约这回事，那么我们就不能进入该部门；相反，我们就可以进入该部门。

我们可以把数据库看成机密部门的门卫，而进入机密部门就是进入 MainFrm 窗体。简单来说，我们想登录一个系统，要在系统登录窗体提供正确的用户名及密码，数据库用来验证用户名及密码的输入是否正确。

现在关键的问题是如何使用 SQL 语句验证用户名及密码的正确性。

我们可以使用 SQL 语句中的 count()聚合函数完成上述验证。简单来说，就是利用 count()聚合函数根据 where 语句中的条件(用户名是什么？密码是什么？)返回符合条件的记录个数。在正常的情况下，如果符合条件的记录个数为 1，则表示数据库中存在合法记录（用户名及密码均正确的记录个数）；如果符合条件的记录个数为 0，则表示数据库中不存在合法记录。

现在我们再把上面的程序重新整理一遍，为了使人们更容易理解，我们先删除有关异常处理的部分代码。

- 创建数据库连接字符串 strCon（访问数据库时需要）。
- 创建数据库连接对象 conn（根据数据库连接字符串 strCon 生成）。
- 打开数据库连接（conn.Open()）。
- 创建数据库命令对象 comm（根据两个条件创建：一个是数据库要执行的 SQL 语句 strSql；另一个是数据库的连接对象 conn）。
- 关键的 SQL 语句 strSql 是需要我们根据实际情况来编写的（本实例利用 count()聚合函数得到符合条件的记录个数）。
- 由数据库命令对象 comm 的特定方法得到我们想要的数据（本实例可以使用命令对象的 ExecuteScalar()方法得到数据库执行命令的首行首列数据。也就是说，如果我们把数据库的数据看成一个由行和列组成的二维表，则会得到左上角格子中的数据）。
- 因为 ExecuteScalar()方法得到的数据是 object 对象，如果要对 0 或 1 进行数字判断，则必须强制将其转换为 int 类型才能进行数字的对比判断。

上述的过程再加上异常处理部分代码就是整个的程序逻辑实现。

（3）程序运行界面如图 3-20 所示。

（4）输入正确的用户名及密码后，单击“登录”按钮，打开“管理员主窗体”界面，如图 3-21 所示。

图 3-20　“登录”窗体界面

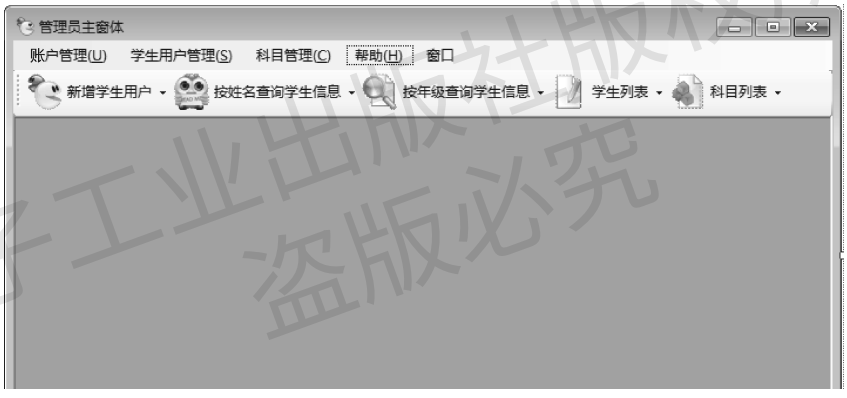

图 3-21　“管理员主窗体”界面

3.3.2　什么是 Command 对象

Command 对象也是.NET Framework 数据提供程序的一个核心对象。Command 对象用于执行面向数据库的一次简单查询。此查询可执行创建、添加、取回、删除或更新记录等操作。

如果该查询用于取回数据，则此数据将以一个 RecordSet 对象返回。这意味着被取回的数据能够被 RecordSet 对象的属性、集合、方法或事件进行操作。

也就是说，Command 对象负责执行命令并从数据源中获得返回结果。如果把 Connection 对象看作连接两岸（数据库与应用程序）的桥梁，则可以把 Command 对象看作运货的汽车，它负责将数据在数据库与应用程序之间移动。

3.3.3　使用 Command 对象

同 Connection 对象一样，Command 对象也属于.NET Framework 数据提供程序，不同的数据提供程序有各自的 Command 对象，如表 3-6 所示。

表 3-6　.NET Framework 数据提供程序及相应的 Command 对象

.NET Framework 数据提供程序	连　接　类	命 名 空 间
SQL 数据提供程序	SqlCommand	System.Data.SqlClient
Oracle 数据提供程序	OracleCommand	System.Data.OracleClient
OLE DB 数据提供程序	OleDbCommand	System.Data.OleDb
ODBC 数据提供程序	OdbcCommand	System.Data.Odbc

在建立了 Connection 对象之后，我们就可以使用相应的 Command 对象来执行应用程序对数据库的操作。创建 Command 对象的语法格式如下：

```
SqlCommand comm=new SqlCommand(string sqlStr,SqlConnection conn);
```

创建一个 Command 对象需要两个参数：第一个参数是要执行的 SQL 语句，第二个参数是已经创建的 Connection 对象。

Command 对象的主要属性和方法如表 3-7 和表 3-8 所示。

表 3-7　Command 对象的主要属性

属　　性	说　　明
Connection	Command 对象使用的数据库连接
CommandText	要执行的 SQL 语句

表 3-8　Command 对象的主要方法

方　　法	说　　明
int ExecuteNonQuery()	返回命令所影响的行数
SqlDataReader ExecuteReader()	返回 DataReader 对象
object ExecuteScalar()	返回执行结果的首行首列

一般在判断用户输入数据是否与数据库数据一致时使用的是 ExecuteScalar()方法，实现方法是：利用 SQL 语句中的 count()聚合函数来判断数据库中应用程序查询语句的记录条数。

使用 Command 对象的步骤如下。

- 创建数据库连接。
- 定义要执行的 SQL 语句。
- 创建 Command 对象。
- 调用 Command 对象中的某个方法执行 SQL 命令。

3.3.4　常见错误与问题

1．没有打开或关闭数据库连接

在执行 Command 对象前必须先打开数据库连接。如果没有打开数据库连接，则会出现下面的异常。

try 部分代码修改如下：

```
try
{
```

```
    //  conn.Open();
      SqlCommand comm = new SqlCommand(sqlStr, conn);
      if ((int)comm.ExecuteScalar() == 1)
      {
          MessageBox.Show("登录数据库成功！");
          Test myForm = new Test();
          myForm.Show();
      }
  }
  catch (Exception ex)
  {
      MessageBox.Show(ex.Message);
  }
```

程序运行后，单击“登录”按钮会弹出异常提示消息框，如图 3-22 所示，提示没有打开数据库连接。

图 3-22　异常提示消息框

删除“//　conn.Open();”语句的注释就可以解决上述问题。如果在程序中忘记编写数据库连接打开命令，则加上打开命令即可。

2．ExecuteScalar()方法的返回值没有进行类型转换

ExecuteScalar()方法的返回值是 object 对象类型，而不是我们主观上认为的数字类型，所以在进行操作时必须将其进行数据类型的转化，否则就会出现数据类型不匹配的错误。没有对 ExecuteScalar()方法进行数据类型转化会发生异常的情况如图 3-23 所示。

图 3-23　没有对 ExecuteScalar()方法进行数据类型转换会发生的异常情况

3.3.5　上机实训

【上机练习 7】

实现 ExamSystem 系统管理员登录功能。

- 需求说明。

- 输入管理员用户名和密码，并选择登录类型为管理员。
- 在数据库中查询管理员用户名和密码是否与用户输入一致，并且登录类型也为管理员。
- 如果验证通过，则跳转登录成功后的窗体，否则给出必要提示。
- 使用 DBHelper 类编写有关数据库的操作方法。
- 对数据库的操作采用调用 DBHelper 类相应方法来实现。

“管理员主窗体”界面如图 3-24 所示。

图 3-24　“管理员主窗体”界面

- 在 DBHelper 类中添加方法 LoginSystem(string strLoginName,string strPwd)，代码如下：

```
public int LoginSystem(string strLoginName,string strPwd)
{
    // 数据库命令
    string strSql = "select count(*) from login where AdminName='"+
strLoginName+"' and AdminPwd='"+strPwd+ "'";
    //打开数据库连接
    TestOpenDB();
    //创建 Command 命令
    SqlCommand com = new SqlCommand(strSql,conn);
    //返回结果
    return((int)com.ExecuteScalar());
}
```

其中，加粗的语句需要格外注意，此语句是对数据库进行命令操作的 SQL 语句，由于存在用户名和密码两个参数，因此需要使用字符串拼接命令将其变成一个新的字符串。注意单引号在字符串中的位置。

【上机练习 8】

实现 ExamSystem 系统学生人数的统计。

- 需求说明。
  - 在“学生用户管理”菜单项中添加一个“统计学生人数”选项。
  - 选择“统计学生人数”选项后，在弹出的相应窗体界面中显示学生人数。
  - 使用 DBHelper 类。

- 实训要点。
  - 打开和关闭数据库连接。
  - 异常处理。
  - 创建 Command 对象。
  - 使用 ExecuteScalar()方法统计学生人数。
- 实现思路及关键代码。
  - 创建 DBHelper 类，提供对数据库的有关操作，并添加异常处理。
  - 创建 SchoolManager 类，实现对 DBHelper 类中数据库操作的方法调用。
  - 使用 DBHelper 类查询学生人数，代码如下：

```
//查询学生人数
//如果返回结果为-1，则表示失败；反之，则表示成功
public int GetStudentNumber()
{
    SqlConnection conn = new SqlConnection(strConn);
    try
    {
        string strSql = "select count(*) from Student ";
        conn.Open();
        SqlCommand comm = new SqlCommand(strSql, conn);
        int iStuNum = (int)comm.ExecuteScalar();
        return iStuNum;
    }
    catch (Exception)
    {
        return -1;
    }
    finally
    {
        conn.Close();
    }
}
```

  - 使用 SchoolManager 类查询学生人数，代码如下：

```
DBHelper myDBHelper = new DBHelper();

// 执行查询学生人数操作并显示结果信息
public string StudentNumber()
{
    int iStuNum = myDBHelper.GetStudentNumber ();
    if (iStuNum == -1)
    {
        return( "查询失败");
    }
```

```
        else
        {
            return ("在校学生人数：" + iStuNum);
        }
    }
    //“统计学生人数”选项单击事件
    SchoolManager mySchool = new SchoolManager();
    MessageBox.Show(mySchool.StudentNumber());
```

【上机练习 9】

实现 ExamSystem 系统显示当前登录用户名。

- 需求说明。
  - 在“账户管理”菜单项中添加一个“当前用户”选项。
  - 选择“当前用户”选项后，在弹出的相应窗体界面中显示当前登录系统的用户名。
  - 使用 DBHelper 类。
- 实训要点。
  - 打开和关闭数据库连接。
  - 异常处理。
  - 创建 Command 对象。
  - 使用 ExecuteScalar()方法获取登录系统的用户名。
- 实现思路及关键代码。
  - 在 DBHelper 类中创建 GetLogingId()方法，该方法实现根据用户名和密码获得登录用户名的主键 AID，代码如下：

```
    public int GetLogingId(string strLoginName, string strPwd)
    {
            try
            {
                // 数据库命令
                string strSql = "select Aid from admin where AdminName='" +
strLoginName + "' and AdminPwd='" + strPwd + "'";
                //打开数据库连接
                TestOpenDB();
                //创建 Command 命令
                SqlCommand com = new SqlCommand(strSql, conn);
                //返回结果
                return ((int)com.ExecuteScalar());
            }
            catch (Exception)
            {
                return -1;
            }
        }
```

  - 在 DBHelper 类中创建 GetLoginName()方法，该方法实现根据 ID 获得登录用户名，

代码如下：

```
public string GetLoginName(int iId)
      {
          try
          {
              // 数据库命令
              string strSql = "select AdminName from admin where Aid='" + iId +"'";
              //打开数据库连接
              TestOpenDB();
              //创建 Command 命令
              SqlCommand com = new SqlCommand(strSql, conn);
              //返回结果
              return ((string)com.ExecuteScalar());
          }
          catch (Exception)
          {
              return "异常";
          }
      }
```

- MainFrm 窗体声明 iCurId 对象，用于保存登录用户的主键，代码如下：

```
public int iCurId;
```

- “登录”窗体中的“登录”按钮代码如下：

```
private void btnLogin_Click(object sender, EventArgs e)
      {
          if (cboLoginType.Text == "")
          {
              MessageBox.Show("请选择登录类型！");
              return;
          }
          if (cboLoginType.Text == "系统管理员")
          {
              DBHelper mydb = new DBHelper();
              // 数据库连接字符串
              try
              {
                  if (mydb.LoginSystem(txtName.Text, txtPwd.Text) == 1)
                  {
                      MainFrm mainFrm = new MainFrm();
                      //记录并保存当前登录用户的主键
                      mainFrm.iCurId  =  mydb.GetLogingId(txtName.Text,  txtPwd.
Text);
                      mainFrm.Show();//打开“管理员主窗体”界面
```

```
                this.Hide();
            }
            else
            {
                MessageBox.Show("请核对用户名及密码！");
            }
        }
        catch (Exception)
        {
            MessageBox.Show("数据库暂时无法访问");
        }
    }
}
```

任务自测表（请在下面记录表中记录自己的学习情况，看看自己掌握的程度）

任务 3.3	看懂本任务的代码	上机练习 7	上机练习 8	上机练习 9

归纳与总结

- ADO.NET 是.NET Framework 中的一组允许应用程序与数据库交互的类。
- ADO.NET 的两个主要组件是.NET Framework 数据提供程序和 DataSet。
- .NET Framework 数据提供程序有 4 个核心对象。
- Connection 对象用于建立应用程序与数据库之间的连接，需要显式打开和关闭数据库连接。
- Command 对象允许应用程序向数据库发送命令请求，可以检索和操作数据库中的数据。
- Command 对象中的 ExecuteScalar()方法返回检索结果的首行首列数据，但返回的是一个 object 类型数据。
- 对于程序中可能发生的异常错误，我们可以使用 try…catch…finally 语句进行处理。
- 创建 DBHelper 类专门处理有关数据库操作的代码。
- 创建 SchoolManager 类处理具体事务代码。