

第20章

数据库安全技术

- » 连接加密的数据库
- » 数据库安全在实际中的应用

20.1 连接加密的数据库

实例 585

连接加密的 Access 数据库

光盘位置：光盘\MR\20\585

初级

趣味指数：★★★★

实例说明

开发一些中小型软件时，通常都采用 Access 数据库，因为 Access 数据库体积小、比较方便。但该数据库的安全性比较低，为了防止非法用户的入侵，通常需要为该数据库设置密码，以确保数据库中数据的安全。本实例将对如何使用 C#连接加密的 Access 数据库进行详细讲解。实例运行效果如图 20.1 所示。

关键技术

本实例主要对 Access 数据库的连接方法进行讲解，在连接 Access 数据库时需要用到 OleDbConnection 类。下面对本实例中用到的关键技术进行详细讲解。

（1）连接 Access 数据库的字符串

连接 Access 数据库的字符串代码如下：

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Documents and Settings\Administrator\桌面\测试.mdb;
```


连接加密的 Access 数据库的字符串代码如下：

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Documents and Settings\Administrator\桌面\测试.mdb;JET OLEDB:Database Password=pwd;
```

（2）OleDbConnection 类

OleDbConnection 类主要用来连接 OLEDB 数据源，其 Open 方法用来使用ConnectionString 所指定的属性设置打开数据库连接。例如，本实例中使用 OleDbConnection 类的 Open 方法连接 Access 和 Excel 数据库的实现代码如下：

```
OleDbConnection oledbcon = new OleDbConnection(strCon);  
oledbcon.Open();
```

 **说明：**程序中使用 OleDbConnection 类时，首先需要在命名空间区域添加 System.Data.OleDb 命名空间，下面遇到类似情况时将不再提示。

设计过程

（1）打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 ConProAccess。

（2）更改默认窗体 Form1 的 Name 属性为 Frm_Main，在该窗体中添加两个 TextBox 控件，分别用来显示选择的 Access 数据库和输入密码；添加两个 Button 控件，分别用来执行选择 Access 数据库和连接 Access 数据库操作；添加一个 RichTextBox 控件，用来显示连接 Access 数据库的字符串和连接状态。

（3）程序主要代码如下：

```
private void button3_Click(object sender, EventArgs e)  
{  
    if (textBox1.Text != "") //判断是否选择了数据库  
    {  
        if (textBox2.Text != "") //判断是否输入了密码  
        {  
            //组合 Access 数据库连接字符串  
            strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + textBox1.Text + ";JET OLEDB:Database Password=" + textBox2.Text + ";";  
        }  
        else  
        {  
            //提示密码错误  
        }  
    }  
}
```

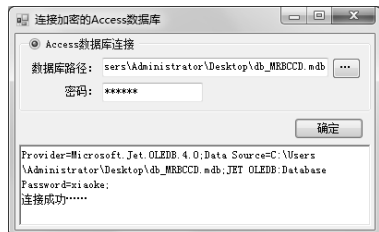


图 20.1 连接加密的 Access 数据库

```

    {
        strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + textBox1.Text + ";"; //连接无密码的数据库
    }
}
OleDbConnection oledbcon = new OleDbConnection(strCon); //使用 OLEDB 连接对象连接数据库
try
{
    oledbcon.Open(); //打开数据库连接
    richTextBox1.Clear(); //清空文本框
    richTextBox1.Text = strCon + "\n 连接成功……"; //显示数据库连接字符串
}
catch
{
    richTextBox1.Text = "连接失败";
}
}

```

秘笈心法

心法领悟 585：如何将数字转换为货币格式？

可以通过调用 ToString("C")方法对输入的数字进行格式化，使其转换为货币格式。将数字转换为货币格式的代码如下：

```
textBox2.Text = Convert.ToInt32(textBox1.Text).ToString("C");
```

实例 586

连接加密的 Excel 文件

光盘位置：光盘\MR\20\586

初级

趣味指数：★★★★

实例说明

Excel 是非常灵活电子表格软件，可以进行复杂的公式计算，那么在程序中如何连接 Excel 文件呢？本实例实现了连接加密的 Excel 文件的功能。运行本实例，首先选择要连接的 Excel 文件，然后输入密码，单击“确定”按钮，即可将连接 Excel 文件的字符串及连接状态显示到下方的文本框中。实例运行效果如图 20.2 所示。

关键技术

本实例主要对 Excel 文件的连接方法进行讲解，在连接 Excel 文件时需要用到 OleDbConnection 类。下面对本实例中用到的关键技术进行详细讲解。

连接 Excel 文件的字符串代码如下：

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Documents and Settings\Administrator\桌面\测试.xls;Extended Properties=Excel 8.0;
```

连接加密的 Excel 文件的字符串代码如下：

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Documents and Settings\Administrator\桌面\测试.xls;JET OLEDB:Database Password=xiaoke;Extended Properties=Excel 8.0;
```



说明：关于 OleDbConnection 类的详细讲解，请参见实例 585 中的关键技术。

设计过程

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 ConProExcel。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm_Main，在该窗体中添加两个 TextBox 控件，分别用来显示选择的 Excel 文件和输入密码；添加两个 Button 控件，分别用来执行选择 Excel 文件和连接 Excel 文件操作；添加一个 RichTextBox 控件，用来显示连接 Excel 文件的字符串和连接状态。

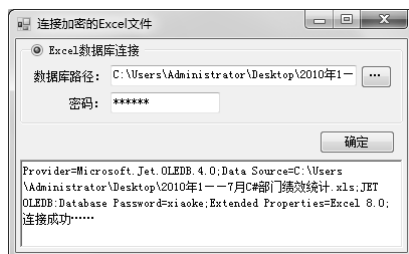


图 20.2 连接加密的 Excel 文件

（3）程序主要代码如下：

```
private void button3_Click(object sender, EventArgs e)
{
    if (textBox1.Text != "") //判断是否选择了 Excel 文件
    {
        if (textBox2.Text != "") //判断是否输入了密码
        {
            strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + textBox1.Text + ";JET OLEDB:Database Password=" + textBox2.Text +
            ";Extended Properties=Excel 8.0;"; //组合 Excel 数据库连接字符串
        }
        else
        {
            //连接无密码的 Excel
            strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + textBox1.Text + ";Extended Properties=Excel 8.0;";
        }
    }
    OleDbConnection oledbcon = new OleDbConnection(strCon); //使用 OLEDB 连接对象连接 Excel 文件
    try
    {
        oledbcon.Open(); //打开数据库连接
        richTextBox1.Clear(); //清空文本框
        richTextBox1.Text = strCon + "\n 连接成功……"; //显示 Excel 文件连接字符串
    }
    catch
    {
        richTextBox1.Text = "连接失败";
    }
}
```

秘笈心法

心法领悟 586：如何获得一个字符串的长度？

获得字符串的长度时，可以直接使用 `string` 类的 `Length` 属性。另外，如果字符串中含有汉字，可以先使用 `System.Text.Encoding` 类中 `Default` 编码方式的 `GetBytes` 方法对字符串编码，然后再使用 `Length` 属性获得其长度。获得字符串长度的代码如下：

```
string str = textBox1.Text;
byte[] strlength = System.Text.Encoding.Default.GetBytes(str);
MessageBox.Show("字符串实际长度: " + str.Length + ",经 Default 解码后的长度: "
+ strlength.Length, "信息", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

实例 587

访问带验证模式的 SQL Server 数据库

光盘位置：光盘\MR\20\587

初级

趣味指数：★★★★

实例说明

在数据库软件的开发过程中，对数据源的指定是必不可少的，而 `SQL Server` 数据库是最常用的一种数据源，本实例将制作一个程序，演示如何在程序中访问带验证模式的 `SQL Server` 数据库。运行本实例，首先输入要连接的 `SQL Server` 服务器名称，然后选择身份验证方式，并且选择要连接的数据库，最后单击“确定”按钮，即可将连接 `SQL` 数据库的字符串及连接状态显示到下方的文本框中。实例运行效果如图 20.3 所示。

关键技术

本实例主要对 `SQL Server` 数据库的连接方法进行讲解，在连接 `SQL Server` 数据库时需要用到 `SqlConnection` 类。下面对本

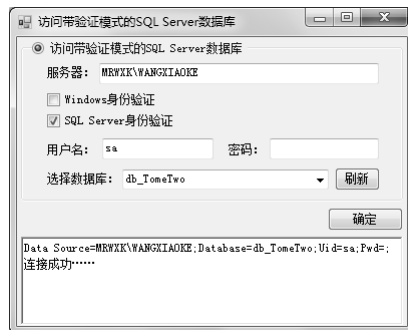


图 20.3 访问带验证模式的 `SQL Server` 数据库

实例中用到的关键技术进行详细讲解。

(1) 连接 SQL Server 数据库的字符串

使用 Windows 身份验证方式连接 SQL Server 数据库的字符串代码如下：

```
Data Source=(local);Initial Catalog =master;Integrated Security=SSPI;
```

使用 SQL Server 身份验证方式连接 SQL Server 数据库的字符串代码如下：

```
Data Source=(local);Database=master;Uid=sa;Pwd=;
```

(2) SqlConnection 类

SqlConnection 类主要用来连接 SQL Server 数据源,其 Open 方法用来使用 ConnectionString 所指定的属性设置打开数据库连接。例如,本实例中使用 SqlConnection 类的 Open 方法连接 SQL Server 数据库的实现代码如下:

```
SqlConnection sqlcon = new SqlConnection(strCon);
sqlcon.Open();
```



说明: 程序中使用 SqlConnection 类时,首先需要在命名空间区域添加 System.Data.SqlClient 命名空间,下面遇到类似情况时将不再提示。

设计过程

(1) 打开 Visual Studio 2008 开发环境,新建一个 Windows 窗体应用程序,并将其命名为 ConSqlServer。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm_Main,在该窗体中添加 3 个 TextBox 控件,分别用来输入 SQL Server 服务器名、用户名和密码;添加两个 CheckBox 控件,用来选择 SQL Server 数据库的访问模式;添加一个 ComboBox 控件,用来选择要连接的数据库名称;添加两个 Button 控件,用来执行刷新数据库列表和连接 SQL Server 数据库操作;添加一个 RichTextBox 控件,用来显示连接 SQL Server 数据库的字符串和连接状态。

(3) 程序主要代码如下:

```
private void button3_Click(object sender, EventArgs e)
{
    if (checkBox1.Checked == true)
    {
        //使用 Windows 身份验证连接 SQL 数据库
        strCon = "Data Source=" + textBox6.Text + ";Initial Catalog = " + comboBox1.Text + ";Integrated Security=SSPI;";
    }
    else if (checkBox2.Checked == true)
    {
        //使用 SQL Server 身份验证连接 SQL 数据库
        strCon = "Data Source=" + textBox6.Text + ";Database=" + comboBox1.Text + ";Uid=" + textBox5.Text + ";Pwd=" + textBox4.Text + ";";
    }
    SqlConnection sqlcon = new SqlConnection(strCon);           //使用 SqlConnection 连接数据库
    try
    {
        sqlcon.Open();                                           //打开数据库连接
        richTextBox1.Clear();                                    //清空文本框
        richTextBox1.Text = strCon + "\n 连接成功……";          //显示数据库连接字符串
    }
    catch
    {
        richTextBox1.Text = "连接失败";
    }
}
```

秘笈心法

心法领悟 587: 如何获得一个字符串中数字的长度?

获得字符串中数字的长度时,可以先使用 CharEnumerator 对象的 MoveNext 方法循环访问字符串中的每个字符,并用 System.Text.Encoding 类中 ASCII 编码方式的 GetBytes 方法对字符进行编码,然后判断经过编码之后的字符的 ASCII 码值是否介于 48 和 57 之间,如果是,则将其添加到一个数组中,最后获得该数组的项数即可。获得字符串中数字长度的代码如下:

```
ArrayList itemList = new ArrayList();
CharEnumerator CEnumerator = textBox1.Text.GetEnumerator();
```

```

while (CEnumerator.MoveNext())
{
    byte[] array = new byte[1];
    array = System.Text.Encoding.ASCII.GetBytes(CEnumerator.Current.ToString());
    int asciiCode = (short)(array[0]);
    if (asciiCode >= 48 && asciiCode <= 57)
    {
        itemList.Add(CEnumerator.Current.ToString());
    }
    textBox2.Text = itemList.Count.ToString();
}

```

20.2 数据库安全在实际中的应用

实例 588

编程修复 Access 数据库

光盘位置：光盘\MR\20\588

中级

趣味指数：★★★★☆

实例说明

Access 数据库操作简单、使用方便，是中小型企业经常采用的数据库，但是它非常容易遭到破坏，并随着时间的增加，数据库文件会变得非常大，那么该如何解决这些问题呢？本实例通过压缩数据库的方法重新组织修复 Access 数据库，减少了 Access 数据库占用的空间。运行本实例，单击“打开”按钮，找到要修复的 Access 数据库；单击“开始修复”按钮，即可完成修复 Access 数据库的操作。实例运行效果如图 20.4 所示。



图 20.4 编程修复 Access 数据库

关键技术

本实例实现时，主要用到了 JRO 命名空间下的 JetEngineClass 对象的 CompactDatabase 方法、System.IO 命名空间下 File 类的 Copy 方法和 Delete 方法，下面对本实例中用到的关键技术进行详细讲解。

（1）CompactDatabase 方法

CompactDatabase 方法主要用来压缩并回收本地数据库中的浪费空间，其语法格式如下：

```
CompactDatabase(string SourceConnection, string DestConnection)
```

参数说明

- ❶ SourceConnection：字符串值，指定与要压缩的源数据库的连接。
- ❷ DestConnection：字符串值，指定与要通过压缩创建的目标数据库的连接。

🔊 注意：添加 JRO 命名空间时，首先需要在“添加引用”中添加 C:\Program Files\Common Files\System\ado\msjro.dll 组件。

（2）File 类的 Copy 方法

该方法为可重载方法，本实例用到的它的重载形式主要用来将现有文件复制到新文件，并且不允许改写同名的文件，其语法格式如下：

```
public static void Copy (string sourceFileName,string destFileName)
```

参数说明

- ❶ sourceFileName：要复制的文件。
- ❷ destFileName：目标文件的名称，不能是一个目录或现有文件。

（3）File 类的 Delete 方法

该方法主要用来删除指定的文件，其语法格式如下：

```
public static void Delete (string path)
```

参数说明

path: 要删除的文件的名称。

设计过程

(1) 打开 Visual Studio 2008 开发环境, 新建一个 Windows 窗体应用程序, 并将其命名为 RepairAccess。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm_Main, 在该窗体中添加一个 TextBox 控件, 用来显示打开的 Access 数据库路径; 添加 3 个 Button 控件, 分别用来选择要修复的 Access 数据库和执行修改 Access 数据、退出窗体操作。

(3) 程序主要代码如下:

```
private void button2_Click(object sender, EventArgs e)
{
    if (!File.Exists(strPathMdb)) //检查数据库是否已存在
    {
        MessageBox.Show("目标数据库不存在, 无法压缩", "操作提示");
        return;
    }
    //声明临时数据库的名称
    string temp = DateTime.Now.Year.ToString();
    temp += DateTime.Now.Month.ToString();
    temp += DateTime.Now.Day.ToString();
    temp += DateTime.Now.Hour.ToString();
    temp += DateTime.Now.Minute.ToString();
    temp += DateTime.Now.Second.ToString() + ".bak";
    temp = strPathMdb.Substring(0, strPathMdb.LastIndexOf("\") + 1) + temp;
    string temp2 = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + temp; //定义临时数据库的连接字符串
    string strPathMdb2 = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + strPathMdb; //定义目标数据库的连接字符串
    JRO.JetEngineClass jt = new JRO.JetEngineClass(); //创建一个 JetEngineClass 对象
    //使用 JetEngineClass 对象的 CompactDatabase 方法压缩修复数据库
    jt.CompactDatabase(strPathMdb2, temp2);
    File.Copy(temp, strPathMdb, true); //拷贝临时数据库到目标数据库(覆盖)
    File.Delete(temp); //删除临时数据库
    MessageBox.Show("修复完成");
}
```

秘笈心法

心法领悟 588: 如何获得字符串中某个数字的位置?

在字符串中获得某个数字的位置时, 可以使用 string 类的 IndexOf 方法, 该方法用来确定指定字符在字符串中的索引, 如果在字符串中能找到指定字符, 则返回其索引, 否则返回-1。在字符串中获得数字位置的代码如下:

```
string str = textBox1.Text.Trim();
int index = str.IndexOf(textBox2.Text.Trim());
if (index >= 0)
    MessageBox.Show("数字 " + textBox2.Text + " 在字符串中的位置为: " + (index+1), "信息", MessageBoxButtons.OK, MessageBoxIcon.Information);
else
    MessageBox.Show("没有要查找的数字", "信息", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

实例 589

Access 数据库备份与恢复

光盘位置: 光盘\MR\20\589

中级

趣味指数: ★★★★★

实例说明

Access 数据库的备份与恢复实质上就是对 Access 数据库文件的复制操作, 本实例就实现了这样的功能。运行本实例, 在“数据库备份设置”栏中选择完要备份的 Access 数据库及备份路径后, 单击“执行备份”按钮, 即可完成 Access 数据库的备份操作。实例运行效果如图 20.5 所示。

在“数据库还原设置”栏中选择完要还原的 Access 数据库及还原路径后，单击“执行还原”按钮，即可完成 Access 数据库的恢复操作。实例运行效果如图 20.6 所示。

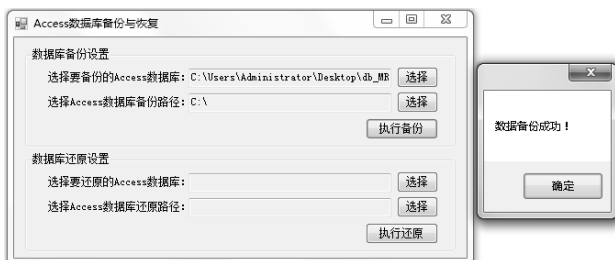


图 20.5 Access 数据库备份



图 20.6 Access 数据库恢复

关键技术

本实例实现时主要通过 File 类的 Copy 方法实现，下面对其进行详细讲解。

File 类的 Copy 方法主要用来将现有文件复制到新文件，它有两种重载形式，分别介绍如下。

（1）第一种用于将现有文件复制到新文件，并且不允许覆盖同名的文件。

语法如下：

```
public static void Copy(string sourceFileName, string destFileName)
```

参数说明

❶ sourceFileName：要复制的文件。

❷ destFileName：目标文件的名称，它不能是一个目录或现有文件。

（2）第二种用于将现有文件复制到新文件，并且允许覆盖同名的文件。

语法如下：

```
public static void Copy(string sourceFileName, string destFileName, bool overwrite)
```

参数说明

❶ sourceFileName：要复制的文件。

❷ destFileName：目标文件的名称，不能是目录。

❸ overwrite：如果可以覆盖目标文件，则为 true；否则为 false。

设计过程

（1）打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 BackupAndRestore Access。

（2）更改默认窗体 Form1 的 Name 属性为 Frm_Main，在该窗体中添加 4 个 TextBox 控件，分别用来显示要备份的 Access 数据库路径、备份路径、要还原的 Access 数据库路径和还原路径；添加 6 个 Button 控件，分别用来选择要备份的 Access 数据库路径、选择备份路径、执行 Access 数据库备份以及选择要还原的 Access 数据库路径、选择还原路径和执行 Access 数据库还原操作。

（3）程序主要代码如下：

```
//备份 Access 数据库
private void button3_Click(object sender, EventArgs e)
{
    string P_str_Path = textBox2.Text + textBox1.Text.Substring(textBox1.Text.LastIndexOf("\\") + 1); //记录备份文件路径
    if (!File.Exists(P_str_Path)) //判断备份文件是否存在
    {
        File.Copy(textBox1.Text, P_str_Path); //备份文件
        MessageBox.Show("数据备份成功!");
    }
    else
    {
        MessageBox.Show("备份文件已经存在，请重新选择路径!");
    }
}
```



```

    }
}
//还原 Access 数据库
private void button4_Click(object sender, EventArgs e)
{
    string P_str_Path = textBox3.Text + textBox4.Text.Substring(textBox4.Text.LastIndexOf("\\") + 1); //记录还原文件路径
    File.Copy(textBox4.Text, P_str_Path, true); //还原文件
    MessageBox.Show("数据还原成功!");
}

```

秘笈心法

心法领悟 589：如何获得字符串中汉字的个数？

获得字符串中汉字的个数时，可以首先定义一个与汉字相匹配的正则表达式，然后使用 CharEnumerator 对象的 MoveNext 方法循环访问字符串中的每个字符，如果访问的字符与定义的正则表达式相匹配，则将其添加到一个数组中，最后获得该数组的项数即为字符串中汉字的个数。获得字符串中汉字个数的代码如下：

```

ArrayList itemList = new ArrayList();
CharEnumerator CEnumerator = textBox1.Text.GetEnumerator();
Regex regex = new Regex("[\u4E00-\u9FA5]{0,}$");
while (CEnumerator.MoveNext())
{
    if(regex.IsMatch(CEnumerator.Current.ToString(),0))
        itemList.Add(CEnumerator.Current.ToString());
    textBox2.Text = itemList.Count.ToString();
}

```

实例 590

加密数据库中的数据

光盘位置：光盘\MR\20\590

中级

趣味指数：★★★★☆

实例说明

本实例演示如何使用 SQL 语言自带的函数，对 SQL Server 2005 数据库中的数据进行加密。运行本实例，首先在窗体左侧的 DataGridView 控件中显示数据库中的数据，然后选中某条记录，单击“加密”按钮，即可将加密后的数据显示在窗体右侧的 DataGridView 控件中。实例运行效果如图 20.7 所示。



图 20.7 加密数据库中的数据

关键技术

本实例实现时，主要用到了 SQL 语言中自带的 pwdencrypt 函数和 pwdcompare 函数，下面分别对它们进行详细讲解。

(1) pwdencrypt 函数

pwdencrypt 函数用来对输入数据进行加密后返回二进制形式的加密内容，其语法格式如下：

```
pwdencrypt('password')
```

参数说明

- ① password：要加密的数据，数据类型为 sysname。
- ② 返回值：varbinary 类型，表示加密后的二进制内容。

（2）pwdcompare 函数

pwdcompare 函数主要用来检查明文是否与加密的二进制数据内容相等，其语法格式如下：

```
pwdcompare('clear_text_password','password_hash')
```

参数说明

- ❶ clear_text_password：未加密的数据。
- ❷ password_hash：密码的加密哈希值。
- ❸ 返回值：int 类型，如果 clear_text_password 参数的哈希值与 password_hash 参数匹配，返回 1；否则返回 0。

🔊 注意：pwdencrypt 函数和 pwdcompare 函数是 SQL Server 未公开的函数，主要用于 SQL Server 的内部调用，它们的优点是使用方便；而缺点是这两个函数没有公开，这就意味着它们有可能改变，并且可能不兼容早期版本，所以在使用上有一定的风险。

设计过程

（1）打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 EncryptDataInDB。

（2）更改默认窗体 Form1 的 Name 属性为 Frm_Main，在该窗体中添加两个 DataGridView 控件，分别用来显示数据库中加密前的数据和加密后的数据；添加一个 Button 控件，用来执行加密数据库中的数据操作。

（3）程序主要代码如下。

Frm_Main 窗体的后台代码中，当在窗体左侧的 DataGridView 控件中选择了某条记录后，单击“加密”按钮，使用 SQL 语言中提供的 pwdencrypt 函数对选中的记录进行加密，并将加密后的数据显示在右侧的 DataGridView 控件中。“加密”按钮的 Click 事件代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
    if (strID != "" && strName != "")
    {
        //创建数据库连接对象
        SqlConnection sqlcon = new SqlConnection("Data Source=MRWXK\\WANGXIAOKE;Database=db_TomeTwo;Uid=sa;Pwd=;");
        SqlCommand sqlcom = new SqlCommand("declare @id int,@name varchar(20) select @id=" + strID + ",@name=" + strName
            + " insert into tb_Business(ID,BusinessName) select @id,convert(varbinary(256),pwdencrypt(@name)) "
            + "select * from tb_Business where ID=@id and pwdcompare(@name,BusinessName)=1 delete from tb_Business where ID="
            + strID + " and BusinessName=" + strName + "", sqlcon);
        sqlcon.Open();
        sqlcom.ExecuteNonQuery();
        sqlcon.Close();
        MessageBox.Show("已成功对数据库中的数据进行了加密！");
    }
    GetData(dataGridView2);
}
```

//创建 SqlCommand 对象
//打开数据库连接
//执行数据库命令
//关闭数据库连接
//对 dataGridView2 控件进行数据绑定

上面的代码中用到了 GetData 方法，该方法为自定义的、无返回值类型的方法，主要用来从数据库中查询数据，并对 DataGridView 控件进行数据绑定。GetData 方法的实现代码如下：

```
private void GetData(DataGridView dgv)
{
    //创建数据库连接对象
    SqlConnection sqlcon = new SqlConnection("Data Source=MRWXK\\WANGXIAOKE;Database=db_TomeTwo;Uid=sa;Pwd=;");
    SqlDataAdapter sqlda = new SqlDataAdapter("select * from tb_Business", sqlcon);
    DataSet myds = new DataSet();
    sqlda.Fill(myds);
    dgv.DataSource = myds.Tables[0];
}
```

//从指定表中查找数据
//创建数据集对象
//填充数据集
//对数据表格控件进行数据绑定

秘笈心法

心法领悟 590：如何获得字符串中指定的后几位字符？

获得字符串中指定的后几位字符时，可以通过调用 string 类的 Substring 方法对字符串进行截取，代码如下：

```
string str = textBox1.Text.Trim().Substring(textBox1.Text.Trim().Length - Convert.ToInt32(textBox2.Text.Trim()), Convert.ToInt32(textBox2.Text.Trim()));
```



实例 591

加密 DataSet 数据集

光盘位置: 光盘\MR\20\591

中级

趣味指数: ★★★★★

实例说明

DataSet 数据集好比内存中的数据库, 其中可以存储各种数据, 本实例将对其中的数据进行加密。运行本实例, 单击“加密”按钮, 将 DataSet 数据集中的数据进行加密, 并写入到 XML 文件中; 单击“解密”按钮, 从 XML 文件中读取数据到 DataSet 中, 并对其进行解密。实例运行效果如图 20.8 所示。



图 20.8 加密 DataSet 数据集

关键技术

本实例实现时, 主要用到了 DESCryptoServiceProvider 类的 CreateEncryptor 方法、CreateDecryptor 方法及 FileStream 类的构造函数和 CryptoStream 类的构造函数, 下面对本实例中用到的关键技术进行详细讲解。

(1) DESCryptoServiceProvider 类的 CreateEncryptor 方法

DESCryptoServiceProvider 类主要用来定义访问数据加密标准 (DES) 算法的加密服务提供程序 (CSP) 版本的包装对象, 其 CreateEncryptor 方法用来使用指定的密钥 (Key) 和初始化向量 (IV) 创建对称数据加密标准 (DES) 加密器对象, 该方法的语法格式如下:

```
public override ICryptoTransform CreateEncryptor (byte[] rgbKey, byte[] rgbIV)
```

参数说明

- ❶ rgbKey: 用于对称算法的密钥。
- ❷ rgbIV: 用于对称算法的初始化向量。
- ❸ 返回值: 对称 DES 加密器对象。



说明: DESCryptoServiceProvider 类位于 System.Security.Cryptography 命名空间下。

(2) DESCryptoServiceProvider 类的 CreateDecryptor 方法

该方法主要用来使用指定的密钥 (Key) 和初始化向量 (IV) 创建对称数据加密标准 (DES) 解密器对象, 其语法格式如下:

```
public override ICryptoTransform CreateDecryptor (byte[] rgbKey, byte[] rgbIV)
```

参数说明

- ❶ rgbKey: 用于对称算法的密钥。
- ❷ rgbIV: 用于对称算法的初始化向量。
- ❸ 返回值: 对称 DES 解密器对象。

(3) FileStream 类的构造函数

FileStream 类的构造函数主要用来初始化 FileStream 类的新实例, 该构造函数可重载, 本实例中用到的它的重载形式用来使用指定的路径、创建模式和读/写权限初始化 FileStream 类的新实例, 其语法格式如下:

```
public FileStream (string path, FileMode mode, FileAccess access)
```

参数说明

- ❶ path: 当前 FileStream 对象将封装的文件的相对路径或绝对路径。
- ❷ mode: FileMode 常数, 确定如何打开或创建文件。
- ❸ access: FileAccess 常数, 确定 FileStream 对象访问文件的方式, 获取 FileStream 对象的 CanRead 和 CanWrite 属性。如果 path 指定磁盘文件, 则 CanSeek 为 true。



说明: FileStream 类位于 System.IO 命名空间下。

(4) CryptoStream 类的构造函数

CryptoStream 类的构造函数主要使用目标数据流、要使用的转换和流的模式初始化 CryptoStream 类的新实例, 其语法格式如下:

```
public CryptoStream (Stream stream, ICryptoTransform transform, CryptoStreamMode mode)
```

参数说明

- ❶ stream: 对其执行加密转换的流。
- ❷ transform: 要对流执行的加密转换。
- ❸ mode: CryptoStreamMode 枚举值之一, 用来指定加密流的模式。CryptoStreamMode 枚举值及说明如表 20.1 所示。

表 20.1 CryptoStreamMode 枚举值及说明

枚 举 值	说 明
Read	对加密流的读访问
Write	对加密流的写访问



说明: CryptoStream 类位于 System.Security.Cryptography 命名空间下。

设计过程

(1) 打开 Visual Studio 2008 开发环境, 新建一个 Windows 窗体应用程序, 并将其命名为 EncryptDataset。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm_Main, 在该窗体中添加一个 DataGridView 控件, 用来显示数据库中的数据; 添加两个 Button 控件, 分别用来执行加密 DataSet 数据集和解密 DataSet 数据集操作。

(3) 程序主要代码如下。

Frm_Main 窗体的后台代码中, 自定义一个静态的、无返回值类型的方法 EncryptDataSetToXML, 该方法用来对 DataSet 数据集中的数据进行加密, 并导出到 XML 文件中。代码如下:

```
public static void EncryptDataSetToXML(DataSet dataSet, string outXMLFilePath)
{
    DESCryptoServiceProvider objDES = new DESCryptoServiceProvider();           //创建 DES 算法提供者对象
    FileStream fout = new FileStream(outXMLFilePath, FileMode.OpenOrCreate, FileAccess.Write); //创建 FileStream 对象
    //用指定的 Key 和初始化向量 (IV) 创建对称数据加密标准 (DES) 加密器对象
    CryptoStream objCryptoStream = new CryptoStream(fout, objDES.CreateEncryptor(desKey, desSIV), CryptoStreamMode.Write);
    StreamWriter objStreamWriter = new StreamWriter(objCryptoStream);           //创建写入流对象
    XmlSerializer objXMLSer = new XmlSerializer(typeof(DataSet));               //创建 XML 序列化对象
    objXMLSer.Serialize(objStreamWriter, dataSet);                             //对数据集进行序列化
    objStreamWriter.Close();                                                    //释放写入流对象
}
```

Frm_Main 窗体的后台代码中, 自定义一个静态的、返回值类型为 DataSet 的方法 DecryptDataSetFromXML, 该方法用来读取 XML 文件中的数据到 DataSet, 并对其进行解密。代码如下:

```
public static DataSet DecryptDataSetFromXML(string inXMLFilePath)
{
    DESCryptoServiceProvider objDES = new DESCryptoServiceProvider();           //创建 DES 算法提供者对象
    FileStream fin = new FileStream(inXMLFilePath, FileMode.Open, FileAccess.Read); //创建 FileStream 对象
    //用指定的 Key 和初始化向量 (IV) 创建对称数据加密标准 (DES) 解密器对象
    CryptoStream objCryptoStream = new CryptoStream(fin, objDES.CreateDecryptor(desKey, desSIV), CryptoStreamMode.Read);
    TextReader objTxrReader = new StreamReader(objCryptoStream);                 //创建读取流对象
}
```

```

XmlSerializer objXMLSer = new XmlSerializer(typeof(DataSet));           //创建 XML 序列化对象
DataSet ds = (DataSet)objXMLSer.Deserialize(objTxrReader);             //对流进行反序列化
return ds;                                                             //返回数据集
}

```

秘笈心法

心法领悟 591：如何获得字符串中大写字母的个数？

在程序中获取字符串中大写字母的个数时，可以先使用 CharEnumerator 对象的 MoveNext 方法循环访问字符串中的每个字符，并将字符用 System.Text.Encoding 类中 ASCII 编码方式的 GetBytes 方法进行编码，然后判断经过编码之后的字符的 ASCII 码值是否介于 65 和 90 之间，如果是，则将其添加到一个数组中，最后获得该数组的项数即可。获得字符串中大写字母个数的代码如下：

```

ArrayList itemList = new ArrayList();
CharEnumerator CEnumerator = textBox1.Text.GetEnumerator();
while (CEnumerator.MoveNext())
{
    byte[] array = new byte[1];
    array = System.Text.Encoding.ASCII.GetBytes(CEnumerator.Current.ToString());
    int asciicode = (short)(array[0]);
    if (asciicode >= 65 && asciicode <= 90)
    {
        itemList.Add(CEnumerator.Current.ToString());
    }
}

```

实例 592

防止 SQL 注入式攻击

光盘位置：光盘\MR\20\592

中级

趣味指数：★★★★★

实例说明

SQL 注入式攻击是指利用软件设计上的漏洞，在目标服务器上运行 SQL 命令以进行其他方式的攻击，动态生成 SQL 命令时没有对用户输入的数据进行验证，本实例为了使程序更加安全，使用 C# 实现防止 SQL 注入式攻击功能。运行本实例，用户只有输入数据库中存在的用户名及对应密码后，才可以通过单击“登录”按钮正常登录程序。实例运行效果如图 20.9 和图 20.10 所示。



图 20.9 登录成功



图 20.10 登录失败

关键技术

要防止 SQL 注入式攻击，首先需要了解 SQL 注入式攻击。

SQL 注入式攻击是指有些人利用软件设计上的漏洞对软件进行恶意攻击，而没有对用户输入的数据进行验证是 SQL 注入攻击得逞的主要方式。

例如，如果用户的查询语句是 `select * from tb_User where name='&user&' and password='&pwd&'`，那么，如果用户名为“1' or '1'='1”，则查询语句将会变成：

```
select * from tb_User where name='1' or '1'='1' and password='&pwd&'
```

这样一来查询语句就通过，其他人就可以进入软件的管理界面，所以防范时需要对用户的输入进行安全性检查。

检查用户输入的最佳途径是通过存储过程实现。存储过程可以在 SQL Server 中创建，也可以通过 C#语言创建。本实例在 C#代码中通过对 SQL 语句使用参数来对用户输入的信息进行安全性验证，实现代码如下：

```
//创建数据库桥接器对象
SqlDataAdapter sqlda = new SqlDataAdapter("select Name,Pwd from tb_Login where Name=@name and Pwd=@pwd", sqlcon);
//为 SQL 语句中的参数赋值
sqlda.SelectCommand.Parameters.Add("@name", SqlDbType.NChar, 10).Value = textBox1.Text;
sqlda.SelectCommand.Parameters.Add("@pwd", SqlDbType.NChar, 10).Value = textBox2.Text;
```

设计过程

(1) 打开 Visual Studio 2008 开发环境，新建一个 Windows 窗体应用程序，并将其命名为 SQLInner。

(2) 更改默认窗体 Form1 的 Name 属性为 Frm_Main，在该窗体中添加两个 TextBox 控件，分别用来输入用户名和用户密码；添加两个 Button 控件，分别用来执行用户登录和清空文本框操作。

(3) 程序主要代码如下：

```
private void button1_Click(object sender, EventArgs e)
{
    //创建数据库连接对象
    SqlConnection sqlcon = new SqlConnection("Data Source=MRWXK\\WANGXIAOKE;Database=db_TomeTwo;Uid=sa;Pwd=;");
    //创建数据库桥接器对象
    SqlDataAdapter sqlda = new SqlDataAdapter("select Name,Pwd from tb_Login where Name=@name and Pwd=@pwd", sqlcon);
    //为 SQL 语句中的参数赋值
    sqlda.SelectCommand.Parameters.Add("@name", SqlDbType.NChar, 10).Value = textBox1.Text;
    sqlda.SelectCommand.Parameters.Add("@pwd", SqlDbType.NChar, 10).Value = textBox2.Text;
    DataSet myds = new DataSet(); //创建 DataSet 数据集对象
    sqlda.Fill(myds); //填充数据集
    if (myds.Tables[0].Rows.Count > 0) //判断数据集中的表中是否有行
        MessageBox.Show("用户登录成功！ ", "提示", MessageBoxButtons.OK, MessageBoxIcon.Information);
    else
    {
        MessageBox.Show("用户登录失败，原因为：用户名或密码错误！ ", "错误", MessageBoxButtons.OK, MessageBoxIcon.Error);
        textBox1.Text = textBox2.Text = ""; //清空文本框
        textBox1.Focus(); //为用户姓名文本框设置输入焦点
    }
}
```

秘笈心法

心法领悟 592：如何获得某字符在字符串中最后出现的位置？

获得某字符在字符串中最后出现的位置时，可以使用 string 类的 LastIndexOf 方法，该方法用来确定指定字符在字符串中最后一次出现的索引位置，如果在字符串中找到指定字符，则返回其索引，否则返回-1。在字符串中获得某字符最后一次出现位置的代码如下：

```
if (textBox1.Text.LastIndexOf(textBox2.Text) == -1)
    MessageBox.Show("在字符串" + textBox1.Text + "中没有" + textBox2.Text
        + "字符", "信息", MessageBoxButtons.OK, MessageBoxIcon.Information);
else
    MessageBox.Show("字符" + textBox2.Text + "在字符串" + textBox1.Text + "中的位置为"
        + (textBox1.Text.LastIndexOf(textBox2.Text) + 1), "信息", MessageBoxButtons.OK, MessageBoxIcon.Information);
```