第 10 章　访问其他数据源

在 ADO.NET 体系中，非常重要的组件就是.NET Data Provider，它负责建立与数据库之间的连接并执行数据操作。ADO.NET 提供了多种.NET Data Provider，负责连接不同的数据库。在前面的章节中，通常使用的是 SQL Server .NET Data Provider，使用其他的.NET Data Provider 能够访问其他类型的数据库。

10.1　使用 ODBC .NET Data Provider

ODBC（Open Database Connection，开放式数据互连）是访问数据库的一个统一的接口标准。在 C++开发中，经常使用 ODBC 来与数据库互连，.NET 同样提供了连接 ODBC 的方法。ODBC 可以让开发人员通过 API 来访问多种不同的数据库，包括 SQL Server、Access、MySql 等。

10.1.1　ODBC .NET Data Provider 简介

ODBC（Open Database Connection，开放式数据互连）是访问数据库的一个统一的接口标准，它允许开发人员使用 ODBC API（应用程序接口）来访问多种不同的数据源，并执行数据操作。

当使用应用程序时，应用程序首先通过使用 ODBC API 与驱动管理器进行通信。ODBC API 由一组 ODBC 函数调用组成，通过 API 调用 ODBC 函数提交 SQL 请求，然后驱动管理器通过分析 ODBC 函数并判断数据源的类型。驱动管理器会配置正确的驱动器，然后将 ODBC 函数调用传递给驱动器。最后，驱动器处理 ODBC 函数调用，把 SQL 请求发送给数据源，数据源执行相应操作后，驱动器返回执行结果，管理器再吧执行结果返回给应用程序，如图 10-1 所示。

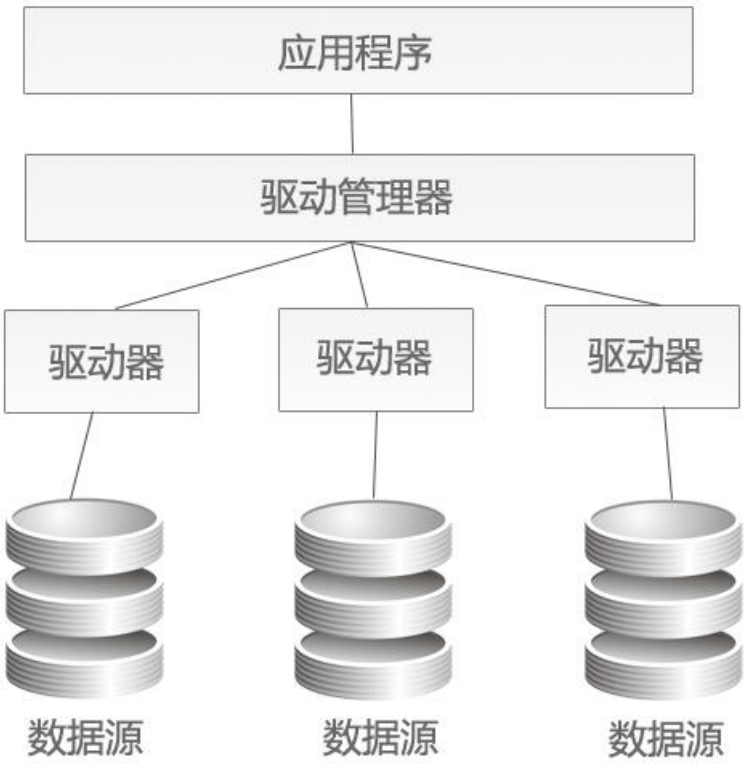

图 10-1　ODBC 原理图

使用命名空间 System.Data.Odbc 才能够使用 ODBC .NET Data Provider 来访问 ODBC 数据源，并且支持对原有的 ODBC 驱动程序的访问。通过 ODBC 能够连接和执行数据操作，其访问方式和 SQL Server .NET Data Provider 相似，都需要先与数据源建立连接并打开连接，然后创建 Command 对象执行相应操作，最后关闭数据连接。

通过 ODBC 驱动程序访问数据源与 SQL Server .NET Data Provider 相同，ODBC .NET Data Provider 同样包含 Connection、Command、DataReader 等类为开发人员提供数据的遍历和存取等操作，这些类和功能如下所示。

- OdbcConnection：建立与 ODBC 数据源的连接。
- OdbcCommand：执行一个 SQL 语句或存储过程。
- OdbcDataReader：与 Command 对象一起使用，读取 ODBC 数据源。
- OdbcDataAdapter：创建适配器，用来填充 DataSet。
- OdbcCommandBuilder：用来自动生成插入、更新、删除等操作的 SQL 语句。

上述对象在 ADO.NET 中经常遇到，在前面的章节中，SQL Server .NET Data Provider 同样包括这些对象，使用 ODBC 操作数据源的操作方法与 SQL Server .NET Data Provider 基本相同，使得开发人员无需额外的学习即可轻松使用。

10.1.2　建立连接

ODBC .NET Data Provider 连接数据库有两种方法，一种是通过 DSN 连接数据库，第二种就是使用 OdbcConnection 对象建立与数据库的连接。

1．使用 DSN 的连接字符串进行连接

使用 DSN（Data Source Name，数据源名）连接数据库，必须首选创建 ODBC 数据源，当创建一个 ODBC 数据源时，需要在管理工具中配置。在开始菜单中找到并打开【控制面板】，然后在【控制面板】中选择【管理工具】选项，在【管理工具】选项中选择【数据源（ODBC）】选项，选择后还需要选择【系统 DSN】标签用于系统 DSN 的配置，如图 10-2 所示。

单击【添加】按钮，在弹出的对话框中选择合适的驱动程序，由于这里需要使用 DSN 连接 ACCESS 数据库，就需要选择 ACCESS 数据库的相应的驱动，这里选择【Microsoft Access Driver(*mdb)】选项，如图 10-3 所示。

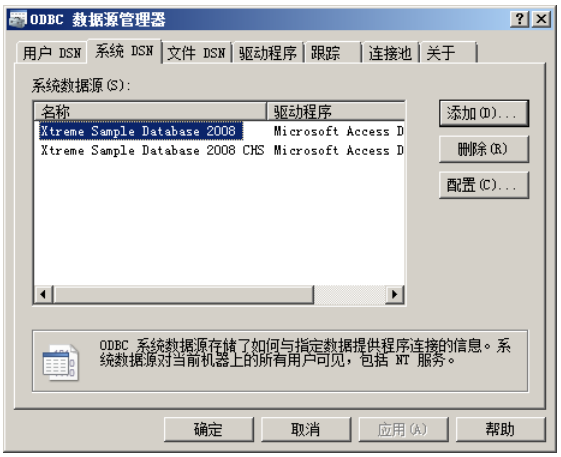

图 10-2　数据源管理器

图 10-3　创建新数据源

单击【选择】按钮可以为需要使用数据库的应用程序选择相应的数据库，在选择完成相应的数据库后就能够在应用程序中使用 DSN 连接该数据源，如图 10-4 所示。

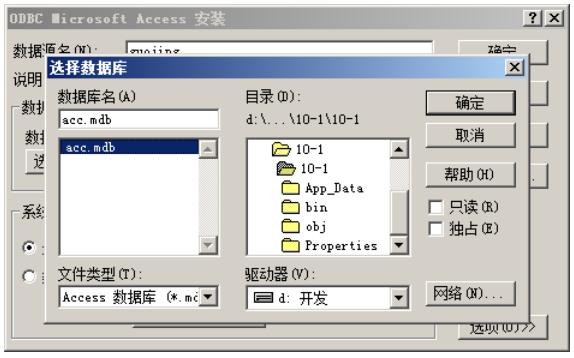

图 10-4　选择数据源

选择好相应的驱动程序后，系统会弹出【ODBC Microsoft Access 安装】对话框并为驱动设置数据源名和说明，如图 10-5 所示。单击【确定】按钮就完成了数据源的配置，如图 10-6 所示。

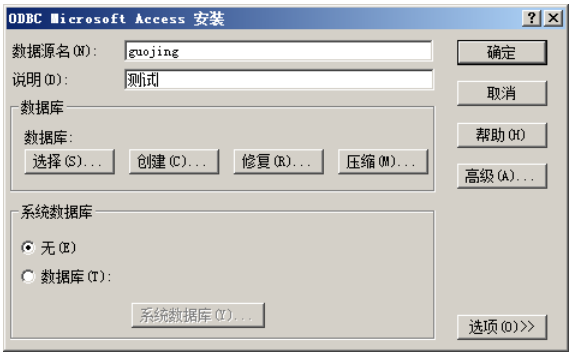

图 10-5　命名数据源

图 10-6　数据源配置完毕

当配置完成数据源后，就可以编写.NET 应用程序来访问数据源。打开 Visual Studio 2008，选择【ASP.NET Web 应用程序】选项，如图 10-7 所示。

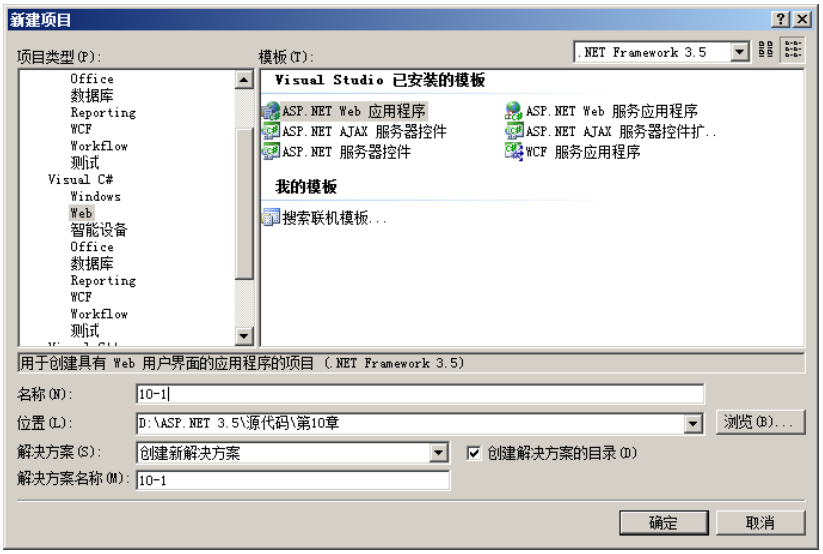

图 10-7　创建 ASP.NET 应用程序

当创建完成数据源之后，就可以使用 OdbcConnection 对象连接应用程序和数据库，与连接字串一样，OdbcConnection 对象需要使用 Open 方法才能打开与数据库之间的连接。在使用 OdbcConnection 对象同样需要使用命名控件 System.Data.Odbc，示例代码如下所示。

```
using System.Data.Odbc;
```

引用了 System.Data.Odbc 命名空间后，就可以创建 Connection 对象进行数据连接，示例代码如下所示。

```
string str = @"DSN=guojing";                          //使用 ODBC 数据源
OdbcConnection con = new OdbcConnection(str);         //创建 OdbcConnection 对象
con.Open();                                           //打开数据库连接
```

上述代码使用了 ODBC 数据源，数据源的名称和刚才创建的名称相同，数据库连接字串直接使用“DNS=数据源名称”即可。打开了与数据库的连接后，即可对数据库进行操作，操作方法同样和普通的方法没有区别。如果希望执行查询语句并填充数据集，则需要创建 DataAdapter 对象和 DataSet 对象，示例代码如下所示。

```
string strsql = "select * from mytable";
OdbcDataAdapter da = new OdbcDataAdapter(strsql,con);          //创建 DataAdapter 对象
DataSet ds = new DataSet();                                    //创建 DataSet
da.Fill(ds, "tablename");                                      //填充数据集
```

若需要执行插入、更新、删除等操作，可以使用 Command 对象执行相应的操作，示例代码如下所示。

```
OdbcCommand cmd = new OdbcCommand("insert into mytable values ('title')",con);
cmd.ExecuteNonQuery();                                         //执行 SQL 语句
```

当需要对其他的数据源执行操作时，在配置 DSN 时配置其他数据源和数据源驱动程序即可，如图 10-8 所示。

图 10-8　配置其他数据源

使用 ODBC 配置数据源有一些好处，就是能够为其他类型的数据库配置数据驱动而不用考虑驱动如何进行手动方式连接。

2．使用接字符串进行连接（ACCESS）

ACCESS 数据库是桌面级的数据库，是以一种文件形式保存的数据库。在使用 ACCESS 数据库时，很多情况下都不能依靠 ODBC 建立数据驱动来连接数据库，在这种情况下，需要使用连接字符串连接 ACCESS 数据库，示例代码如下所示。

```
string str = "provider=Microsoft.Jet.OLEDB.4.0 ;
Data Source=D:\ASP.NET 3.5\源代码\第 10 章\10-1\10-1\acc.mdb";          //配置数据库路径
```

上述代码使用了 Micorsoft.Jet.OLEDB 4.0 驱动进行 ACCESS 数据库的连接。在上述代码中，ACCESS 文件的地址为“D:\ASP.NET 3.5\源代码\第 10 章\10-1\10-1”。但是这样编写代码有若干坏处，最大的坏处就是暴露了物理路径，当非法用户访问或获取代码后，能容易的就能够获取数据库的信息并通过下载工具下载数据库，这样是非常不安全的。为了提高应用程序的安全性，开发人员可以使用 Server.MapPath 方法指定相对路径，示例代码如下所示。

```
string str = "provider=Microsoft.Jet.OLEDB.4.0 ;Data Source=" + Server.MapPath("acc.mdb") + "";
```

上述代码指定了数据库的文件，以及相对路径，当 acc.mdb 与文件夹路径相同时，系统会隐式的补完绝对路径，而不会轻易的暴露物理路径，如果 acc.mdb 在文件夹的上层路径，则只需要使用“../”来确定相对路径。当指定数据库文件地址后，可以使用 Connection 对象进行数据库连接操作并使用 Open 方法打开数据连接，示例代码如下所示。

```
OdbcConnection con = new OdbcConnection(str);                  //配置连接对象
con.Open();                                                    //打开连接
```

3．使用接字符串进行连接（SQL Server）

SQL Server 数据库可以采用两种不同的连接方式，正如 SQL Server Management Studio 中注册连接一样，包括 Windows 安全认证和 SQL Server 验证两种验证方式，SQL Server Management Studio 中注册连接的方式如图 10-9 所示。

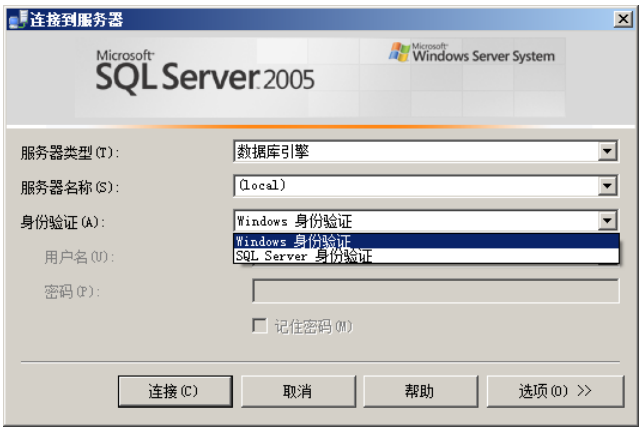

图 10-9　SQL Server 的两种连接方式

当使用 Windows 安全认证进行数据连接时，SQL Server 无需用户提供连接的用户名和密码即可连接，因为 Windows 安全认证是通过 Windows 登录时的账号来登录的。开发 ASP.NET 应用程序时，需要显式的声明这是一个安全的连接，示例代码如下所示。

```
@" Driver={SQL Server}; Server=(local);Database=mytable;Trusted_Connection=Yes";
```

如果需要使用 SQL Server 验证方式连接数据库，就不能够使用 Trusted_Connection 属性进行数据连接，而需要配置 User=用户名;Password=密码，示例代码如下所示。

```
@"Driver={SQL Server}; Server=(local);Database=mytable;User ID=sa;Password= ";
```

10.2　使用 OLE DB.NET Data Provider

OLE DB 是访问数据库的另一个统一的接口标准，它建立在 ODBC 基础之上，通过 OLE DB 可以访问关系型数据库和非关系型数据库，OLE DB 不仅使应用程序和数据库之间的交互减少，还能够最大限度的提升数据库性能。

10.2.1　OLE DB.NET Data Provider 简介

OLE DB（Object Link and Embedding Database，对象连接与嵌套数据库）是访问数据库的另一个统一的接口标准，它建立在 ODBC 基础之上，不仅能够提供传统的数据库访问，并且能够访问关系型数据库和非关系型数据库。

OLE DB 其本质就是一个封装数据库访问的一系列 COM 接口，使用 COM 接口不仅能够减少应用程序和数据库之间的通信和交互，也能够极大的提升数据库的性能，让数据库的访问和操作更加便捷。

OLE DB.NET Data Provider 是 OLE DB 数据源的托管数据提供者，OLE DB.NET Data Provider 能够在不更改 COM 组件的情况下，使用 COM Interpol 来使用 OLE DB.NET Data Provider 访问数据源，如果需要使用 OLE DB.NET Data Provider 访问数据源，则必须存在一个支持 OLE DB.NET Data Provider 的 OLE DB Provider。OLE DB.NET Data Provider 访问原理图如图 10-10 所示。

图 10-10　OLE DB.NET Data Provider 原理图

OLE DB.NET Data Provider 支持 OLE DB Provider 包括：

- SQLOLEDB：用来访问 SQL Server 数据库。
- MSDAORA：用来访问 Oracle 数据库。
- Microsoft.Jet.OLEDB.4.0：用于访问 Access 等使用 Jet 引擎的数据库。

使用 OLE DB.NET Data Provider 访问数据库同 ODBC .NET Data Provider 相同，具有 Connection 对象、Command 对象等用于数据库的连接和执行，以及数据存储等，常用对象如下所示。

- OleDbConnection：建立与 ODBC 数据源的连接。
- OleDbCommand：执行一个 SQL 语句或存储过程。
- OleDbDataReader：与 Command 对象一起使用，读取 ODBC 数据源。
- OleDbDataAdapter：创建适配器，用来填充 DataSet。
- OleDbCommandBuilder：用来自动生成插入、更新、删除等操作的 SQL 语句。

这些常用的对象使用方法同其他 ADO.NET 数据操作类相同，使用 Connection 对象进行数据库连接，连接后，使用 Command 对象进行数据库操作，并使用 Close 方法关闭数据连接。

10.2.2　建立连接

使用 Ole Db 连接数据库基本只需要使用字符串连接方式即可，当需要使用 Ole Db 中的对象时，必须使用命名空间 System.Data.OleDb，示例代码如下所示。

```
using System.Data.OleDb;                                    //使用 OleDb 命名空间
```

使用命名空间后，就能够创建 Ole Db 的 Connection 对象，以及 Command 对象对数据库进行连接和数据操作。

1．使用接字符串进行连接（ACCESS）

Access 数据库是一种桌面级的数据库，同文件类型的数据库类似，所以连接 Access 数据库时，必须指定数据库文件的路径，或者使用 Server.MapPath 来确定数据库文件的相对位置。示例代码如下所示。

```
string str = "provider=Microsoft.Jet.OLEDB.4.0 ;
Data Source=" + Server.MapPath("access.mdb") + "";                    //设置连接字串
OleDbConnection con = new OleDbConnection(str);                       //创建连接对象
try
{
    con.Open();                                                       //打开连接
    Label1.Text = "连接成功";                                          //显式提示信息
    con.Close();                                                      //关闭连接
}
catch(Exception ee)                                                   //抛出异常
{
    Label1.Text = "连接失败";
}
```

上述代码设置了 Access 数据库连接字串，并使用 OleDb 的 Connection 方法创建数据连接，创建连接后打开数据连接，如果数据连接打开成功，则返回成功，否则返回连接失败。

2．使用接字符串进行连接（SQL Server）

当需要连接 SQL Server 时，无需对连接、执行等操作进行修改，只需要对数据库连接字串进行修改即可，示例代码如下所示。

```
OleDbConnection con= new OleDbConnection();
con.ConnectionString="Provider=SQLOLEDB;Data
Source=(local);Initial Catalog=mytable;uid=sa;pwd=sa";                //初始化连接字串
try
{
    con.Open();                                                       //尝试打开连接
    Label1.Text = "连接成功";                                          //显式连接信息
    con.Close();                                                      //关闭连接
}
catch
{
    Label1.Text = "连接失败";                                          //抛出异常
}
```

上述代码只需修改连接字串的格式，而无需修改其他 ADO.NET 中的对象，以及对象执行的方法即可。

10.3　访问 MySql

MySql 是一个开源的小型关系型数据库，MySql 数据库功能性强、体积小、运行速度快、成本低和安全性强，并且广泛的被中小型应用所接受。MySql 通常情况下和 PHP 一起开发使用，在 ASP.NET 中，同样能够使用 MySql 进行数据库的存储。

10.3.1 MySql 简介

MySql（http://www.MySql.com）是一套开源的小型关系型数据库，MySql 能够执行标准的 SQL 语句进行数据操作。MySql 能够支持多种操作系统，包括 Windows、Linux、Mac OS 等等，是一种跨平台的数据库产品，并且 MySql 还为多种语言提供了 API，这些语言包括传统的 C/C++也包括新近的 Python 和 Ruby，MySql 具有功能性强、体积小、运行速度快、成本低和安全性强等特点，MySql 还具有以下特性：

- MySql 具有客户端/服务器结构的分布式数据库管理系统。
- 支持多个操作系统。
- 为多种编程语言提供 API。
- 支持多用户，多线程操作数据库。
- 提供了 TCP/IP，ODBC 和 JDBC 等多种数据库连接方式。
- 优化 SQL 语句能力，提升性能。
- 支持 ANSI 标准的所有数据类型。
- 开放源代码，并且可以免费下载安装。

MySql 不仅具有良好的性能表现，同时 MySql 还能够执行复杂的 SQL 语句操作，但是 MySql 的缺点就在于无法创建和使用存储过程，缺少一些大型数据库所必备的功能。

10.3.2 建立连接

ASP.NET 应用程序需要使用 ODBC .NET Data Provider 连接 MySql 数据库。在连接数据库之前，MySql 数据库能够被.NET Data Provider 识别和驱动必须首先安装 MySql ODBC 驱动程序（MySql-connector-odbc-5.1.5），开发人员可以在官方网站免费获取 MySql ODBC 驱动程序并免费下载（http://dev.MySql.com/downloads/connector/odbc/5.1.html#win32）。单击下载的安装程序，MySql ODBC 驱动程序就会弹出安装向导，并提示安装。通常情况下，只需要选择典型安装即可如图 10-11 和图 10-12 所示。

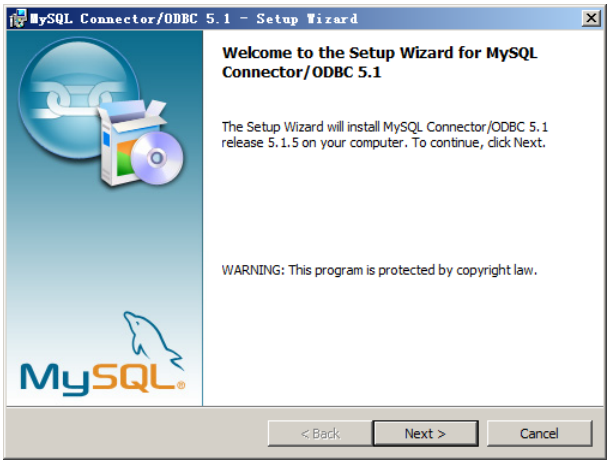

图 10-11 安装 MySql ODBC 驱动程序

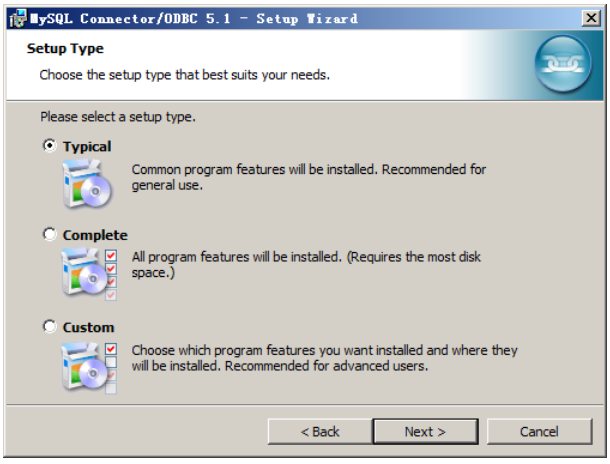

图 10-12 选择安装类型

选择完成后，单击【Next】按钮后即可完成安装。安装完成 MySql ODBC 驱动程序后，单击【开始】菜单找到【控制面板】，在【控制面板】的【管理工具】中选择【数据源 ODBC】选项，并在【数据源

ODBC 选项】对话框中选择【驱动程序】选项卡，如果【驱动程序】选项卡中已经存在 MySql ODBC Driver 选项，则说明 MySql ODBC 驱动程序已经安装完成。当安装完成后，需要新建 DSN，如图 10-13 和图 10-14 所示。

图 10-13　驱动程序已经安装完毕

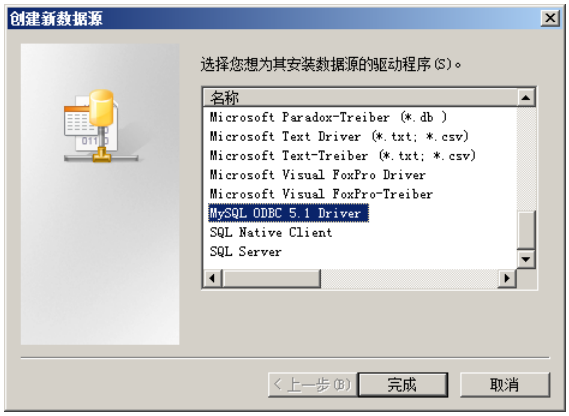

图 10-14　新建 MySql DSN

单击完成，系统会弹出对话框用于指定 DNS 名、MySql 服务器名、数据库名、密码、端口等信息，如图 10-15 所示。

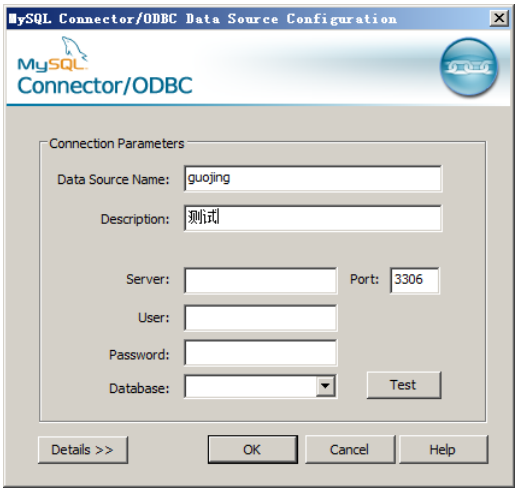

图 10-15　为 MySql ODBC 配置 DSN

配置完成后，即可通过使用 ODBC 类库进行数据库操作，示例代码如下所示。

```
string str = @"DSN=guojing";                                    //设置 Connection 属性,使用 MySql DSN
OdbcConnection con = new OdbcConnection(str);                   //设置 Connection 对象
con.Open();//打开连接
OdbcDataAdapter da = new OdbcDataAdapter("select * from mytables", con);//创建 DataAdapter
DataSet ds = new DataSet();                                     //创建 DataSet 对象
da.Fill(ds, "MySqltable");                                      //填充 DataSet 数据集
```

同样可以创建 Command 对象进行数据操作，示例代码如下所示。

```
OdbcCommand cmd = new OdbcCommand("insert into mytables values ('MySql title')", con);
cmd.ExecuteNonQuery();                                          //执行 Command 方法
con.Close();
```

上述代码同连接和使用 SQL 数据库基本相同，其中 str 变量是配置的是 DSN 的属性。当需要脱离

DSN 连接 MySql 数据库时，可以不需要配置 DSN 来访问 MySql 数据库，在编写 MySql 数据库连接字符串时，需要指定驱动程序名称、IP 地址或数据库名，常用的关键字如下所示。

- Driver：设置驱动程序名。
- Server：设置服务器的 IP 地址或者是本地主机。
- Database：设置数据库名称。
- Option：设置选项值。
- UID：设置连接用户名。
- PWD：设置连接密码。
- Port：设置连接端口，默认值为 3306。

编写 MySql 数据库连接字串代码如下所示。

```
string strbase = @"Driver=MySql ODBC 5.1 Driver;Server=localhost;Database=test;UID=guojing";
```

上述字串能够连接 MySql 数据库，开发人员在连接数据库后就能够使用 Command 对象进行相应的数据存储和操作。

10.4 访问 Excel

Excel 同 Access 数据库一样，都是 Microsoft Office 办公软件中的一个组件，Excel 主要用来处理电子表格，同时 Excel 也能够方便的进行数据存储，并提供强大的运算能力和统计功能，经常使用于办公环境。

10.4.1 Excel 简介

在办公环境中，大部分的办公人员都使用 Excel 进行报表处理，所以，Excel 中存储着大量的信息。这些信息对决策者或者是办公自动化管理而言，都比较重要，在办公室应用中，很多文档都是通过 Excel 保存在计算机中的，所以在编写应用程序时，常常需要访问 Excel 来访问和存储数据。但是，开发人员会发现通过应用程序访问 Excel 是一件非常困难的事情。

因为 Excel 并不是数据库，所以 Excel 不支持相关的数据结构，所以当开发人员需要对 Excel 进行数据访问时，会变得比较困难。但是从另一个角度来看，Excel 文件是由一张张工作表组成，其结构很像数据库中的表，所以，应该能够通过相应的手段让应用程序访问 Excel。

ASP.NET 提供了一些类和方法用于连接和访问 Excel 数据库，极大的方便了开发人员的开发，简化了开发代码。

10.4.2 建立连接

ASP.NET 访问 Excel 通常有两种方法，一种是使用 ODBC .NET Data Provider 进行访问，另一种则是使用 OLE DB .NET Data Provider 进行访问。这两种访问方式在原理上基本相同，同 ADO.NET 其他对象一样，访问和操作 Excel 文件时，都必须使用 Connection 对象进行连接，然后使用 Command 对象执行 SQL 命令。

1．使用 DSN 连接 Excel 数据源

首先，必须创建一个 Excel 数据源，例如 data.xls，并手动增加若干数据，如图 10-16 所示。

	A	B	C	D	E
1	编号	学号	年龄	姓名	性别
2	1	20051183049	21	小红	女
3	2	20051183050	21	小白	男
4	3	20051183051	20	小刘	男
5	4	20051183052	22	小赵	男
6					

图 10-16　创建 Excel 数据源

数据源创建完毕后，则需要在【数据源（ODBC）】中添加支持 Excel 的数据源，如图 10-17、10-18 所示。

图 10-17　创建数据源

图 10-18　数据源安装

添加数据源之后，需要配置数据源，如图 10-17 所示，则需要选择相应数据文件，单击【选择工作簿】按钮选择数据文件的位置，如图 10-19 所示，配置完毕后，单击【确定】按钮，即可配置成功。

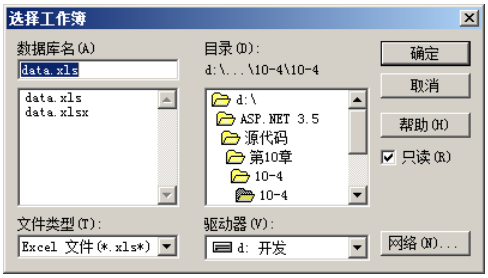

图 10-19　选择数据文件

配置完成后，就可以通过 ASP.NET 应用程序访问 Excel 数据源，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    string str = @"DSN=myexcel";                          //使用 ODBC 连接数据源
    OdbcConnection con = new OdbcConnection(str);         //新建连接对象
    try
    {
        con.Open();                                                  //尝试打开连接
        Label1.Text = "连接成功";                                    //显式连接信息
        con.Close();                                                 //关闭连接
    }
    catch
    {
        Label1.Text = "连接失败";                                    //抛出异常
    }
```

```
        }
```

执行数据连接后，就可以通过 SQL 语句执行数据源遍历，示例代码如下所示。

```
        protected void Page_Load(object sender, EventArgs e)
        {
            string str = @"DSN=myexcel";
            OdbcConnection con = new OdbcConnection(str);
            try
            {
                con.Open();                                                                    //打开连接
                Response.Write("连接成功<hr/>");                                               //输出 HTML
                OdbcDataAdapter da = new OdbcDataAdapter("select * from [Sheet1$]",con);       //创建适配器
                DataSet ds = new DataSet();                                                    //创建 DataSet 数据集
                int count=da.Fill(ds, "exceltable");                                           //填充数据集
                for (int i = 0; i < count; i++)                                                //遍历数据
                {
                    Response.Write(ds.Tables["exceltable"].Rows[i]["姓名"].ToString()+"<hr/>");   //输出数据
                }
            }
            catch(Exception ee)                                                                //抛出异常
            {
                Response.Write(ee.ToString());
            }
        }
```

上述代码使用了 SQL 对 Excel 数据源中的数据进行查询和遍历，运行结果如图 10-20 所示。

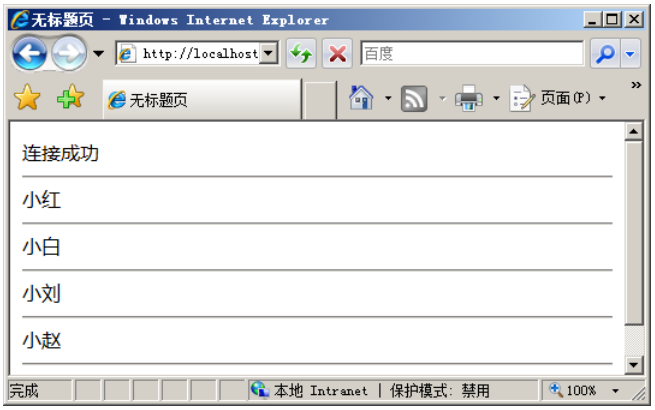

图 10-20　遍历 Excel 数据源

2．使用 OLE DB .NET Data Provider 连接 Excel 数据源

使用 OLE DB .NET Data Provider 连接和操作 Excel 数据源，同其他 ADO.NET 数据源访问方法类似，同样是使用 OleDbConnection 对象进行数据连接，使用 OleDbCommand 对象进行数据访问，示例代码如下所示。

```
        protected void Page_Load(object sender, EventArgs e)
        {
            string str =
            @"Provider=Microsoft.Jet.OleDb 4.0;Data
            Source="+Server.MapPath("data.xls")+";Extended Properties= Excel 8.0;";  //设置 Excel 连接串
            OleDbConnection con = new OleDbConnection(str);                          //创建连接对象
            try
```

```
        {
            con.Open();                                          //尝试打开连接
            Label1.Text = "连接成功";                             //显式连接信息
            con.Close();                                         //关闭连接
        }
        catch
        {
            Label1.Text = "连接失败";                             //抛出异常
        }
    }
```

上述代码通过使用 OLE DB .NET Data Provider 连接字串进行 Excel 数据源的连接，在连接完成后，其数据操作的方法都与 ADO.NET 对象的操作方法相同。

10.5 访问 txt

文本文件（.txt）是一种最基本的文件类型，访问 txt 的方法比较多，不仅能够通过使用 ODBC .NET Data Provider 进行访问，或者使用 OLE DB .NET Data Provider 进行访问。而可以通过 System.IO 进行文本文件的访问。

10.5.1 使用 ODBE.NET Data Provider 连接 txt

使用 ODBE.NET Data Provider 建立与 txt 文件的连接需要在连接字符串中指定驱动器名，同样可以在管理工具中创建【数据源（ODBC）】来访问 txt 文本文件，如图 10-21 和图 10-22 所示。

图 10-21 建立数据源

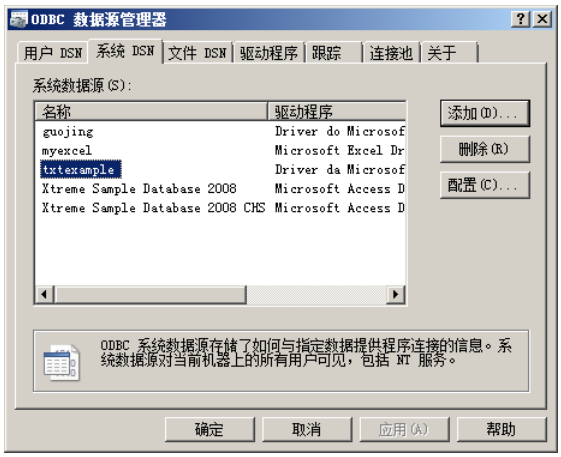

图 10-22 完成配置数据源

上面创建了 txt 数据源的 ODBC 数据源，当连接 txt 数据源时，可使用 DSN 连接数据源，其中 txt 文本中的字符如下所示。

```
title
连接 2
连接 3
连接 4
连接 5
连接 6
```

连接 7

当通过 ODBC 连接 txt 数据源时，只需使用 Connection 对象即可，示例代码如下所示。

```
OdbcConnection con = new OdbcConnection(@"DSN=txtexample");
try
{
    con.Open();                                          //尝试打开连接
    Label1.Text = "连接成功";                             //显式连接信息
    con.Close();                                         //关闭连接
}
catch
{
    Label1.Text = "连接失败";                             //抛出异常
}
```

成功创建连接后，就可以对数据源进行操作了。与数据库的结构和 Excel 结构不同的是，txt 基本上没有类似与表的数据结构，所以在使用 SQL 语句时，基本上是通过查询 txt 文件来实现的，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    OdbcConnection con = new OdbcConnection(@"DSN=txtexample");          //创建连接
    try
    {
        con.Open();                                                      //打开连接
        OdbcDataAdapter da = new OdbcDataAdapter("select * from data.txt", con);   //创建适配器
        DataSet ds = new DataSet();                                      //创建数据集
        int count=da.Fill(ds, "txttable");                               //填充数据集
        for (int i = 0; i < count; i++)                                  //遍历输出
        {
            Response.Write(ds.Tables["txttable"].Rows[i]["title"].ToString()+"<hr/>");   //输出数据
        }
    }
    catch
    {
        Response.Write("连接失败");                                       //抛出异常
    }
}
```

上述代码遍历了 txt 中的数据，运行结果如图 10-23 所示。

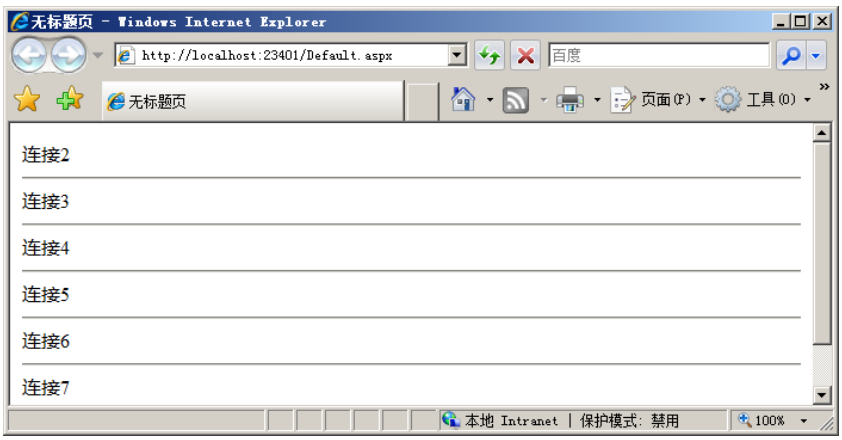

图 10-23 遍历 txt 中的数据

10.5.2 使用 OLE DB .NET Data Provider 连接 txt

使用 OLE DB .NET Data Provider 建立与 txt 文件的连接，只需要在连接字串中指定提供程序名、数据源名、扩展属性、HDR 和 FMT 等参数即可，示例代码如下所示。

```
OleDbConnection olecon = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB 4.0;
Data Source=c:\sample\;Extended Properties=text;HDR=yes;FMT=Delimited");    //创建连接对象
```

当完成连接后，在执行查询等操作时需要指定 txt 文件名，示例代码如下所示。

```
OdbcDataAdapter da = new OdbcDataAdapter("select * from data.txt", con);        //创建适配器
DataSet ds = new DataSet();                                                     //创建数据集
int count=da.Fill(ds, "txttable");                                              //填充数据集
for (int i = 0; i < count; i++)                                                 //遍历数据集
{
    Response.Write(ds.Tables["txttable"].Rows[i]["title"].ToString()+"<hr/>");  //输出数据
}
```

注意：txt 文件的数据连接字串中，数据库结构的中“数据库”的概念对于 txt 文件而言应该是文件所在的目录，而不是具体的某个文件。而具体的某个文件，相当于是数据库中“表”的概念。

10.5.3 使用 System.IO 命名空间

System.IO 能够创建文件，从而与 txt 文件进行交互，在使用 System.IO 命名空间时，通常需要使用各种类来进行文件操作，常用的类如下所示：

- File：提供用于创建、复制、删除、移动和打开文件的静态方法。
- FileInfo：提供创建、复制、删除、移动和代开文件的实例方法。
- StreamReader：从数据流中读取字符。
- StreamWriter：向数据流中写入字符。

在进行文件交互操作时，通常使用 File 类和 FileInfo 类来执行相应的操作。File 类和 FileInfo 类的基本用法基本相同，但是 File 类和 FileInfo 类有一些本质的区别。File 类和 FileInfo 类虽然基本用法相同，但是 File 类不用创建类的实例，而 FileInfo 类需要创建类的实例，所以 File 类可直接调用其类的静态方法执行文件操作，效率也比 FileInfo 类高。

File 类包含的主要方法如下所示：

- OpenText：打开现有的文件进行读取。
- Exists：判断一个文件是否存在。
- CreateText：创建或打开一个文件以便写入文本字符串。
- AppendText：将 txt 文本追加到现有的文本。

在执行 txt 操作时，首先需要判断文件是否存在，如果文件存在，则可用 File 类的 OpenText 方法打开 txt 文件。首先，需要创建一个 txt 文件并编写一些内容，内容如下所示。

```
something happend
in my restless dream,i see that place
silent hill
```

编写完成 txt 文件后，则可以通过 File 类读取 txt 文件内容，示例代码如下所示。

```
if (File.Exists(Server.MapPath("data.txt")))                                    //判断文件是否存在
```

```
{
    Response.Write("文件存在");                                    //输出文件存在
    File.OpenText(Server.MapPath("data.txt"));                    //打开文件
}
```

如果文件存在，并打开了文件，则需要创建 StreamReader 对象，使用 StreamReader 对象的 ReadLine 方法读取一行或 ReadToEnd 方法读取整个文本文件。示例代码如下所示。

```
StreamReader rd = File.OpenText(Server.MapPath("data.txt"));    //StreamReader 对象
while (rd.Peek() != -1)                                         //如果没有读完
{
    Response.Write(rd.ReadLine() + "<hr/>");                    //输出信息
}
```

上述代码中，Peek 方法用于返回指定的 StreamReader 对象流中的下一个字符，但是不会把这个字符从流中删掉。如果流中没有文本字符，则会返回-1，运行结果如图 10-24 所示。

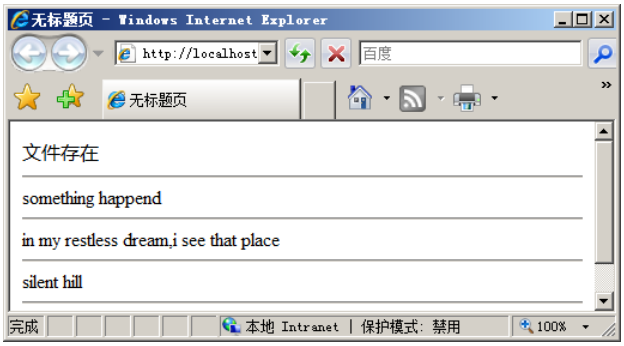

图 10-24　读取 txt 文本文件

上述代码运行后，页面会从 txt 文件中读取内容，并循环遍历输出。

10.6　访问 SQLite

SQLite 是一款轻量级数据库，其类型在文件形式上很像 Access 数据库，但是相比之下 SQLite 操作更快。SQLite 也是一种文件型数据库，但是 SQLite 却支持多种 Access 数据库不支持的复杂的 SQL 语句，并且还支持事务处理。

10.6.1　SQLite 简介

SQLite 数据库具有小巧和轻量的特点，在 SQLite 数据库开发时，SQLite 是为嵌入式特别准备的，所以 SQLite 具有小巧、资源占用率低等特点。在嵌入式设备中，只需要几百 k 的内存即可。而同时 SQLite 支持多种操作系统，包括 Windows、Linux 等主流操作系统。

SQLite 能够与多种语言结合，包括.NET、PHP、Java 等。同样 SQLite 能够支持 ODBC 接口，相比于 MySql 数据库而言，SQLite 执行效率更快。虽然 SQLite 小巧，但是 SQLite 同样能够支持 SQL 语句，这些支持的 SQL 语句相比其他的数据库产品，毫不逊色。SQLite 支持的常用 SQL 语句如下所示。

- CREATE INDEX：创建索引。
- CREATE TABLE：创建表。
- CREATE TRIGGER：创建触发器。

- CREATE VIEW：创建视图。
- DELETE：执行删除操作。
- DROP INDEX：删除索引。
- DROP TABLE：删除表。
- DROP TRIGGER：删除触发器。
- DROP VIEW：删除视图。
- INSERT：执行插入操作。
- SELECT：执行选择，查询操作。
- UPDATE：执行更新操作。

SQLite 不仅能够支持常用的 SQL 语句，SQLite 还包括其他 SQL 语句方便开发人员高效的执行数据库操作。

10.6.2 SQLite 连接方法

通过访问 http://sourceforge.net/project/showfiles.php?group_id=132486 页面下载 SQLite for ADO.NET，下载完成后，在应用程序中添加引用即可，如图 10-25 所示。

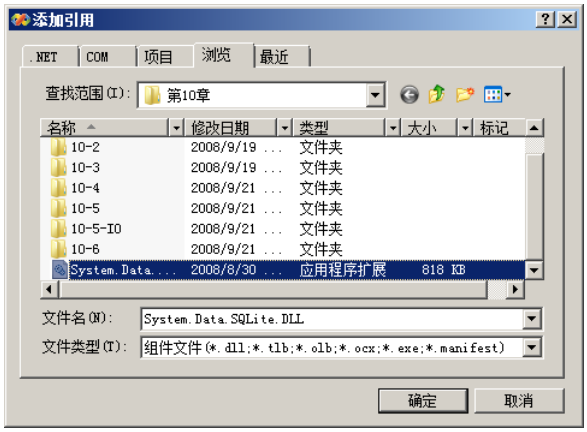

图 10-25　在项目中添加引用

在添加引用后，可以使用命名空间来为相应的操作提供支持，示例代码如下所示。

```
using System.Data.SQLite;                                    //使用 SQLite 命名空间
```

使用命名控件后，就可以像 ADO.NET 其他对象一样，创建数据库并执行数据库操作。在连接 SQLite 数据库之前，首先需要创建 SQLite 数据库，通过编程的方法可以创建数据库，也可以手动创建 SQLite 数据库，示例代码如下所示。

```
        SQLiteConnection.CreateFile(Server.MapPath("sqlite.db"));          //创建数据库
```

SQLite 文件后缀可以直接指定，或者不为 SQLite 文件指定后缀，示例代码如下所示。

```
        SQLiteConnection.CreateFile(Server.MapPath("sqlite"));             //创建无后缀名的数据库
```

SQLite 创建成功后，就可以通过 Connection 对象连接 SQLite，示例代码如下所示。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        SQLiteConnection con = new SQLiteConnection("Data Source="+Server.MapPath("sqlite.db"));
        try
        {
            con.Open();                                              //打开连接
```

```
            Response.Write("连接成功");                                    //提示连接成功
        }
        catch(Exception ee)
        {
            Response.Write(ee.ToString());                                 //连接错误则抛出异常
        }
    }
```

上述代码创建了与 SQLite 数据库的连接，并尝试打开连接。SQLite 数据库同样支持 DataAdapter 对象、Command 对象，其操作与 ADO.NET 其他对象基本上没有任何区别，.NET 平台下的开发人员能够很容易的上手 SQLite。

10.7 小结

本章介绍了 ADO.NET 访问其他数据源的知识，这些数据源包括 MySql、Excel、txt、SQLite 等常用的数据源，这些数据源虽然在性能和功能上都与 SQL Server 有一段距离，但是在小型、轻便的数据操作和应用中，这些数据库都起着非常重要的作用。本章还介绍了如何使用 ODBE.NET Data Provider 连接数据库和使用 OLE DB .NET Data Provider 连接数据库，以及 ODBE.NET Data Provider 和 OLE DB .NET Data Provider 的作用和区别。本章还包括：

- 使用 ODBC.NET Data Provider：讲解了如何使用 ODBC.NET Data Provider 进行数据库存储。
- 使用 OLE DB.NET Data Provider：讲解了如何使用 OLE DB.NET Data Provider 进行数据库存储。。
- 访问 MySql：讲解了如何通过 ADO.NET 访问 MySql。
- 访问 Excel：讲解了如何通过 ADO.NET 访问 Excel。
- 访问 txt：讲解了如何通过 ADO.NET 访问 txt。
- 访问 SQLite：讲解了如何通过 ADO.NET 访问 SQLite。

通过本章的知识介绍，可以比较深入的了解 ADO.NET 和其他数据源的连接、访问、操作等方法，熟练掌握 ADO.NET，能够在.NET 平台下的大部分的数据开发中事半功倍。