第5章

实现前台页面①

学习目标

- 了解数据库引擎和 ADO.NET 的作用；
- 掌握组装 SQL 语句的技巧；
- 理解连接字符串的作用，掌握通过 DbConnection 对象连接数据库的方法；
- 了解 try…catch…finally 的异常处理在数据库访问中的应用；
- 理解 DbCommand 对象执行 SQL 语句的三种方法以及使用场合；
- 掌握使用 DataReader 对象的标准步骤；
- 了解 SELECT 语句中使用 TOP 参数限制获取记录的数量；
- 基本掌握使用数据库中的数据动态组装 HTML 代码的技巧；
- 掌握查找功能的实现方法，理解 ASP.NET 事件，深刻理解回发(Postback)的概念；
- 掌握 ASP.NET 控件 Label、TextBox 和 Button 的使用；
- 了解如何使用面向对象编程方法，避免重复书写相同代码片段；
- 了解网站发布的安全性问题。

SQL 语言具有强大的查询能力，但小明只是一名音乐爱好者，让他使用 SQL 语言直接操纵数据库既不现实，也不安全。正确的做法，应该采用 SQL 语言的另一种用法：嵌入到应用开发语言中，通过应用软件来完成对数据库的管理。本章介绍如何在 ASP.NET 中使用 SQL 语言获取查询结果，最终正式实现 MPMM 系统的各个前台页面。

应用程序使用 SQL 语言的基本模式是将命令发送给 DBMS，然后获取 DBMS 返回的结果。但实际上这里面涉及很多问题，例如，如何寻找 DBMS？如何发送命令？如何获取返回的数据？怎么知道这个数据是发给哪个程序(进程)的以及返回的数据是对哪条命令的回复？DBMS 怎么知道应用程序是不是合法的数据库访问者？

ADO.NET 数据库引擎帮助 ASP.NET 的开发人员解决了所有这些问题。对于开发人员来说，所谓 ADO.NET 数据库引擎就是指一组对象，通过对象的属性和方法就可以完成和 DBMS 之间的通信，以及返回结果数据的提取。

5.1 连接数据库

要和 DBMS 进行通信，需要使用数据库连接对象和 DBMS 建立连接。所谓建立连接，相当于在应用程序和 DBMS 之间架设一个“通信管道”。应用程序通过连接对象连接到特定 DBMS，然后向连接对象发送 SQL 语句就可以将命令发送给这个 DBMS，并可以从连接

① 对网站而言，前台指数据展示，普通用户使用的页面；后台指管理员维护数据的页面。

对象获取返回结果。

对于 ADO.NET 而言,不同的 DBMS 对应着不同数据库连接(Connection)对象类,但它们都继承自 DbConnection 这个抽象基类,具有相同的使用的方法。具体如表 5-1 所示。

表 5-1 ADO.NET 数据库连接对象

名 称	命名空间	描 述
SqlConnection	System.Data.SqlClient	表示连接 SQL Server 的连接对象
OleDbConnection	System.Data.OleDb	表示连接 OleDb 数据源的连接对象
OdbcConnection	System.Data.Odbc	表示连接 ODBC 数据源的连接对象
OracleConnection	System.Data.OracleClient	表示连接 Orale 数据库的连接对象

1. Conection 对象基本使用过程

(1) 定义连接字符串;

(2) 创建连接对象;

(3) 建立连接;

(4) 通过连接完成一项或多项数据库操作;

(5) 断开连接;

(6) 释放连接对象。

2. 连接字符串

连接对象可以连接不同的 DBMS,甚至连接一些简单的文件型数据库,如 Excel、特定格式的文本文件等。通常把不同的 DBMS 或文件型数据库叫作数据源。连接对象必须清楚地了解所连接的数据源,这是通过连接字符串来实现的。

连接字符串,告诉 ADO.NET 数据源在哪里,需要什么样的数据格式,提供什么样的访问信任级别以及其他任何包括连接的相关信息。连接字符串由一组元素组成,一个元素包含一个键值对,元素之间由“;”分开,语法为

```
key1 = value1; key2 = value2; key3 = value3…
```

不同的数据源,key 和 value 也不同,实际操作中可用专门的工具来生成,也可用程序代码拼接字符串的方式,或者使用 ADO.NET 提供的 DbConnectionStringBuilder 类来帮助生成。在 VS 中(通过“视图”菜单)打开“服务器资源管理器”,右击“数据连接”节点,在弹出的快捷菜单中选择“添加连接”命令,弹出如图 5-1 所示的“添加连接”对话框。

以连接 SSE 为例。单击“数据源”选项后面的“更改”按钮,在弹出的“更改数据源”对话框中选择 Microsoft SQL Server,然后选择数据提供者为“用于 SQL Server 的.NET Framework 数据提供程序”。

VS 开发环境连接数据库并获取连接字符串

在“服务器名”中输入 Win2k8\SQLExpress。其中,Win2k8 是 DBMS 服务器的计算机名,SQLExpress 则是 DBMS 的实例名,实际使用时要替换成具体的服务器参数,并注意斜杠的方向。如果连接默认的实例,那么斜杠和实例名可以省略。

连接的计算机名不能省略,对于 SQL Server 来说可以有以下几种指定方法。

(1) 指定计算机名。注意在网络环境下,计算机名不一定能被识别。

(2) 如果连接本机的是 SQL Server,则可以用“.”来代替计算机名。

(3) 启用 SQL Server 网络配置中的 TCP/IP,可以用 IP 地址代替计算机名[①]。

图 5-1　“添加连接”对话框

在“登录到服务器”中,对于网站应用通常应该选择默认的“使用 Windows 身份验证”。如果要使用 SQL Server 身份验证,则需要启用 SQL Server 的混合身份验证模式,并配置相应的 DBMS 用户、密码、权限。

输入服务器名后,图 5-1 中“连接到数据库”区域被激活,其中可以指定连接默认操作的数据库。如果不指定,则会默认使用 master 数据库,这里需要选择 MpmmDB 作为默认数据库。

单击“测试连接”按钮,可以确定连接字符串配置是否正确。最后单击“确定”按钮保存连接字符串配置,此时在数据库连接节点下增加一个新的数据库节点。选中这个节点,在属性窗口中可以看到节点的属性,其中的值就是连接字符串。

很多时候,VS 会自动使用服务器资源管理器中的数据库连接,一旦使用,VS 会将连接字符串保存在应用系统的 Web.config 配置文件中。开发工具的自动化能提高开发效率,但不利于初学者掌握原理。所以,练习时建议从属性窗口复制连接字符串粘贴到代码中使用。

3. 建立和断开连接

连接对象创建后并不会连接到数据库,通过调用连接对象的 Open()方法,连接对象才会使用连接字符串所指定的设置建立和数据库的连接。

---

① 如果其他方式连接正常,但用 IP 地址无法连接,可以启动“SQL Browser 服务”后再尝试一下。

完成数据库操作后,应该及时调用连接对象的 Close()方法断开连接,否则 DBMS 会一直等待应用程序发送 SQL 语句,占用 DBMS 的资源。而且,如果长时间不过连接使用数据库,连接可能被自动断开,所以开发人员应该养成每次使用前建立连接,使用后断开连接的良好编程习惯。

至于连接对象的创建和释放,则可以相对提前或延迟。也就是说,可以在程序运行一开始就建立连接对象,等到应用程序被关闭时才释放连接对象。对于 ASP.NET 程序来说,可以在页面初始化 Page_Load()方法中创建连接对象,而释放连接对象的工作则交给.NET Framework 的垃圾回收机制自动完成。

5.2 修改首页布局

1. 布局调整


实现 MPMM 首页

MPMM 首页中有两个区域的内容需要根据数据库中的内容动态生成:一是音乐分类链接清单,二是快速查找音乐资料的结果清单。

从工具箱拖放 Literal 控件到首页取代原来的静态音乐分类列表,设置其 ID 属性值为 litCategoryList;考虑到音乐资料清单默认为空白,不美观,改成默认显示最新的 10 个音乐资料,因此最好将原来显示快速查找结果的控件 ID 属性值修改为 litMusicList。

此外,用 TextBox 控件替换原来的 HTML 表单文本框元素;用 Button 控件取代原来的表单提交按钮。它们的作用和对应的表单元素完全相同,但封装成 ASP.NET 服务控件后,开发人员能够在服务器端以使用对象的方式操控。

注意:ASP.NET 控件在页面代码中的标签都带有"asp:"的前缀,且带有 runat="server"的属性。

设置这两个控件的 ID 属性值分别为 tbKey 和 btQuickSeek,并将 Button 控件的 Text 属性值设置为"查找"。修改后的部分页面代码如下:

```
<table>
  …
  <tr>
    <td>
      <asp: Literal ID = "litCategoryList" runat = "server"></asp: Literal>
    </td>
    <td colspan = "2">
      快速查找:
      <asp: TextBox ID = "tbKey" runat = "server"></asp: TextBox>
      <asp: Button ID = "btQuickSeek" runat = "server" Text = "查找" />
      <br />
      <asp: Literal ID = "litMusicList" runat = "server"></asp: Literal>
    </td>
  </tr>
</table>
```

2. 生成分类列表

为了使代码更加清晰,将生成音乐分类列表的代码封装成页面类的一个私有方法

BuildCategoryList()，方法中的代码为：

```
private void BuildCategoryList()
{
  StringBuilder sb = new StringBuilder();                    //字符串拼装工具
  sb.Append("<ul>");                                         //拼装 HTML 列表开始标记
  string sql = "SELECT * FROM Category WHERE Parent_ID IS NULL"; //SQL 语句
  SqlCommand cmd = new SqlCommand(sql, dbConn);              //创建数据库命令对象
  try
  {
    dbConn.Open();                                           //打开数据库连接
    SqlDataReader dr = cmd.ExecuteReader(); //执行 SQL 语句,获取数据库读取对象
    while (dr.Read())                                        //当成功读取下一行时
    {
        //使用这一行的数据拼装一行音乐分类的 HTML 代码
        sb.AppendFormat("<li><a href='MusicList.aspx?cat={0}'>{1}</a></li>",
                  dr["ID"], dr["Name"]);
    }
    dr.Close();                                              //关闭数据库读取对象
  }
  catch
  {
    sb.Append("<li>读取数据库失败!</li>");                    //失败则拼接,表示失败的 HTML 代码
  }
  finally
  {
    dbConn.Close();                                          //关闭数据库连接
  }
  sb.Append("</ul>");                                        //拼装 HTML 列表结束标记
  litCategoryList.Text = sb.ToString();                      //显示内容
}
```

上述代码是一段标准的读取数据库数据代码，其中使用了 try…catch…finally 的异常捕获机制，catch 用来捕获数据库操作的异常，finally 用来保证关闭数据库连接。一个真正的应用程序不仅要实现功能，而且要保证程序的健壮性和可靠性。使用判断语句判断各种情况，结合异常捕获机制，用日志记录详细的出错信息，这 3 点是编写一个实际应用程序应该要做到的。

3. 数据库连接的处理

上述代码中直接使用了数据库连接对象 dbConn，这是页面类的自定义属性(Property)，可以达到在需要时自动创建数据库连接对象的目的。相应的代码如下：

```
public partial class _Default : System.Web.UI.Page
{
  private SqlConnection _dbConn = null; //保存数据库连接对象的成员变量
  private SqlConnection dbConn          //返回数据库连接对象的只读属性
  {
    get
    {
      if (_dbConn == null)              //如果尚未创建数据库连接对象,则先创建数据库连接对象
      {
        string connString = @"Data Source=127.0.0.1\SqlExpress;
        Initial Catalog=MpmmDB; Integrated Security=True";
```

```
            _dbConn = new SqlConnection(connString);      //使用连接字符串,创建数据库连接对象
        }
        return _dbConn;                                   //返回已经创建好的数据库连接对象
    }
  }
}
```

创建连接对象的代码为 new SqlConnection(<连接字符串>)。只有在访问 dbConn 属性时,该创建代码才会被执行,可以避免在不需要时创建数据库连接对象。而创建代码首先检查_dbConn 是否为空,只在为空的时候才创建,从而保证不会重复创建数据库连接对象。

注意代码中的连接字符串采用了直接代码方式提供。其字符串前有一个“@”符号,这是因为数据库服务器地址和实例名分割符“\”刚好是 C#字符串的转义符,所以用字符串前的“@”符号来取消所有转义。同时,使用“@”符号,还允许在字符串中直接使用回车,而无须使用“\n”。这在编写 SQL 语句时非常有用。

4. 数据库命令对象

Connection 对象只提供了应用程序和 DBMS 之间的通道,实际发送 SQL 语句并取得结果的对象是数据库命令对象。不同类型的数据源对于 SQL 的处理和返回结果的格式也有所不同,需要使用不同的数据库命令对象类,但它们都是 DbCommand 抽象基类的子类。常用 DbCommand 对象如表 5-2 所示。

表 5-2 常用 DbCommand 对象

名 称	命名空间	描 述
SqlCommand	System.Data.SqlClient	表示操作 SQL Server 的命令对象
OleDbCommand	System.Data.OleDb	表示操作 OleDb 数据源的命令对象
OdbcCommand	System.Data.Odbc	表示操作 ODBC 数据源的命令对象
OracleCommand	System.Data.OracleClient	表示操作 Oracle 数据库的命令对象

DbCommand 对象有两个最基本的参数,一个是 DbConnection 对象,另一个是需要执行的 SQL 语句。注意 DbCommand 对象只能使用对应类型的连接对象。在 BuildCategoryList()方法中,通过构造函数同时提供了这两个参数:

```
SqlCommand cmd = new SqlCommand(sql, dbConn);          //创建数据库命令对象
```

使用时,也可以首先创建 DbCommand 对象,然后通过给属性赋值的方式来设置这两个参数:

```
SqlCommand cmd = new SqlCommand();                     //创建数据库命令对象
cmd.CommandText = sql;                                 //设置 SQL 命令
cmd.Connection = dbConn;                               //设置连接对象
```

因此,通过修改 CommandText 或 Connection 属性,可以使用一个命令对象在不同时刻执行不同的 SQL 语句,甚至连接到不同的数据源。

需要注意的是,给 DbCommand 对象设置命令和连接对象并不会导致 SQL 命令的发送和处理,真正执行 SQL 命令需要调用 DbCommand 对象的执行(Execute)方法。根据执行结果的不同,执行方法也有所不同,常用的如表 5-3 所示。

表 5-3　DbCommand 对象的不同执行方法

执 行 方 法	返 回 结 果	描　　述
ExcuteReader()	返回类型为 DataReader，值为向前只读记录集	用于执行 SELECT 语句或其他返回结果集的 SQL 语句
ExcuteNonQuery()	返回类型为 int，值为影响的记录数	用于执行 DDL 类的 SQL 语句或其他无结果集的 SQL 语句
ExcuteScalar()	返回类型为 Object，值为结果表的第一行的第一列，忽略其他列或行	用于获取只有一个返回值的查询结果

BuildCategoryList()方法中，使用下列代码来获取音乐分类记录集：

```
SqlDataReader dr = cmd.ExecuteReader();                    //执行 SQL 语句，获取数据库读取对象
```

5. DataReader 对象

不同类型的 DbCommand 对象的 ExcuteReader()方法返回的记录集也不相同，但它们都是“向前只读记录集”，其中 SqlCommand 对象返回的是 SqlDataReader 对象。

所谓“只读”就是指只能从记录集读取数据，无法写入或修改数据；“向前”则是指只能读取下一行，无法退回去读取前面的记录。使用 DataReader 对象的主要优点是节省资源。

无法直接创建 DataReader 对象，只能通过 DbCommand 对象的 ExcuteReader()方法来获取，获取后的标准用法为：

```
while (dr.Read()) //当成功读取下一行时，将下一行设置为当前行
{
  //读取当前行数据
}
```

Read()方法从数据库获取下一行数据并将其作为当前行，如果获取成功则返回 true，否则返回 false。也就是说，如果 dr.Read()方法返回 false，那么遍历记录集就已经完成了。另外，ExcuteReader()方法创建 DataReader 对象的最初，当前行指向第一行之前，首次执行 dr.Read()方法试图读取的是第一行，因此使用数据前一定要先调用 Read()方法。

成功读取一行数据后，应用程序可以通过两种方法使用当前行的数据，一种称为弱类型数据获取方法，另一种则是强类型数据获取方法。

(1) 使用弱类型数据获取方法：DataReader 对象提供索引器来使用当前行数据，索引既可以是字段序号，也可以是字段名称。例如，dr["ID"]表示获取 dr 对象的 ID 字段值。不建议用字段序号的形式，因为这不容易理解，也不便于数据库定义的修改。使用索引器方式获取的数据一律是 Object 类型，所以称为弱类型。

(2) 使用强类型数据获取方法：通过 DataReader 对象的 Get×××(i)方法也可以使用当前行的数据，其中×××为数据类型的名称，表示返回数据类型。例如，GetString(1)返回的就是第 1 列的 String 类型数据。Get×××()方法实际使用很不方便。

DataReader 对象还有以下几个比较重要的属性或方法。

(1) FieldCount 属性：获取当前行中的列数。

(2) HasRows 属性：返回一个逻辑值，该值指示 DataReader 对象读取的记录集中是否存在行。

(3) IsDBNull(i)方法：判断当前行第 i 个字段的数据库值是否为 NULL。如果该字段值为 NULL，Get×××(i)会抛出异常。

最后，DataReader 对象打开后会一直占用对应 DbCommand 对象的数据库连接，所以一旦完成使用，一定要记得用 DataReader 对象的 Close()方法关闭；否则 DbCommand 对象和数据库连接就无法用于其他操作。

5.3 实现首页音乐列表

1. 默认音乐列表

音乐资料列表的输出涉及多个字段，因此使用表格输出，相应的 BuildMusicList()方法定义如下：

```
/// <summary>
/// 使用指定 SQL 查询获取数据,构造音乐资料列表
/// </summary>
private void BuildMusicList(String sql)
{
  StringBuilder sb = new StringBuilder();                    //字符串拼装工具
  sb.Append("<table>");                                      //HTML 表格开始标记
  SqlCommand cmd = new SqlCommand(sql, dbConn);              //创建数据库命令对象
  try
  {
    dbConn.Open();                                           //打开数据库连接
    SqlDataReader dr = cmd.ExecuteReader(); //执行 SQL 语句,获取数据库读取对象
    while (dr.Read())                                        //当成功读取下一行时
    {
      //使用这一行的数据拼装一行音乐分类的 HTML 代码
      sb.Append("<tr>");
      sb.AppendFormat("<td><a href='Music.aspx?id={0}'>{1}</a></td>",
        dr["ID"], dr["Name"]);
      sb.AppendFormat("<td>{0}</td><td>{1}</td><td>{2}</td>",
        dr["Authors"], dr["publishDate"], dr["MediaTypeName"]);
      sb.Append("</tr>");
    }
    dr.Close();                                              //关闭数据库读取对象
  }
  catch
  {
    sb.Append("<tr><td>读取数据库失败!</td></tr>");
  }
  finally
  {
    dbConn.Close();                                          //关闭数据库连接
  }
  sb.Append("</table>");                                     //HTML 表格结束标记
  litMusicList.Text = sb.ToString();                         //显示内容
}
```

为了让 BuildMusicList()方法同时适用于初始默认 10 个最新音乐资料的显示和快速

查找结果的显示，方法使用参数 sql 来传递不同的 SQL 语句。在 Page_Load()方法中调用 BuildMusicList()方法的代码为：

```
if (!IsPostBack) //非回发访问本页面,说明是新的访问
{
  String sql = @"SELECT TOP 10 ms.ID, ms.Name, ms.Authors,
      ms.PublishDate, cn.Name AS MediaTypeName
    FROM Music ms JOIN CodeNames cn
      ON ms.MediaType = cn.Code AND cn.CodeFor = 'Music.MediaType'
    ORDER BY ms.PublishDate DESC";
  BuildMusicList(sql);                                    //构造默认音乐列表
}
```

其中，sql 字符串的赋值用了“@”前缀，所以可以将 SQL 语句按照 SQL 的习惯分成多行书写。

注意 SELECT 后面紧跟的 TOP 10，这是 SQL Server 特有的限定获取记录数的方法，在 SELECT 后紧跟 TOP n，就可以限制获取最多前 n 条记录。这里通过 ORDER BY 让记录按照发布日期倒序排列，使得获取的记录限制在最新发布的 10 个音乐资料。

2. 快速查找列表

通过使用 ASP.NET 的文本框控件，获取输入的查找关键字变得非常简单，ASP.NET 将其封装在文本框控件的 Text 属性中，相应的访问代码为 tbKey.Text，其中 tbKey 就是文本框控件的 ID 属性值。

当小明单击“查找”按钮时，浏览器将带有表单数据的请求发送到服务器，ASP.NET 分析 Request 对象中的数据时，就能够发现小明当时单击了 ID 属性值为 btQuickSeek 的按钮，因此 ASP.NET 会试图在执行完 Page_Load()方法后去执行按钮绑定的单击事件处理方法。

这就是 ASP.NET 提供的网页事件处理机制，看上去和 WinForm 程序的事件处理机制很像，但一定要注意两者其实完全不同：ASP.NET 的事件处理全部发生在服务器端，并不是事件在客户端触发时进行的处理。每次事件的触发都需要将请求发送给服务端，也就是所谓的页面“回发(PostBack)”。

给按钮绑定“单击事件处理方法”的方法很简单，在 VS 中打开设计页面，双击按钮，VS 就会在页面代码中为按钮添加 onclick 属性和属性值。例如，双击“查找”按钮后 VS 自动生成的代码为：

```
<asp:Button ID="btQuickSeek" runat="server" Text="查找"
onclick="btQuickSeek_Click" />
```

同时，在对应后台页面类代码中自动增加一个名为 btQuickSeek_Click()的方法。

对于 MPMM 来说，需要在这个方法中实现对应音乐资料的查找和显示，完整的代码为

```
/// <summary>
/// 查找指定音乐资料的事件处理方法
/// </summary>
protected void btQuickSeek_Click(object sender, EventArgs e)
{
  String sql = String.Format(@"SELECT ms.ID, ms.Name, ms.Authors,
      ms.PublishDate, cn.Name AS MediaTypeName
    FROM Music ms JOIN CodeNames cn
  ON ms.MediaType = cn.Code AND cn.CodeFor = 'Music.MediaType'
```

```
    WHERE ms.Name LIKE '%{0}%' OR ms.Description LIKE '%{0}%'
      OR Authors LIKE '%{0}%' OR cn.Name LIKE '%{0}%'", tbKey.Text);
  BuildMusicList(sql);                                    //构造音乐资料列表
}
```

上述代码中,先使用查找关键字拼装出 SQL 语句(语句中使用了 LIKE 模糊匹配),然后调用 BuildMuiscList()方法来实现音乐资料的显示。

5.4 实现动态音乐列表

在拼接音乐分类导航项时,使用了<a href="MusicList.aspx? cat={0}">这样的超链接,因此当小明单击“音乐分类”导航项时,浏览器会请求 MusicList.aspx 这个页面,同时使用 GET 方法通过 URL 传递 cat 参数,其值为对应音乐分类的 ID 属性值。

实现分类音乐资料列表页

1. 页面布局

在“解决方案资源管理器”中右击 MPMM 网站,在弹出的快捷菜单中选择“添加新项”命令,然后在对话框中选择“Web 窗体”模板,输入名称 MusicList.aspx,单击“添加”按钮,就会在 MPMM 网站项目下新建一个 ASP.NET 页面,包括前台页面文件 MusicList.aspx 和后台程序代码文件 MusicList.aspx.cs。

该页面需要负责提取 cat 参数的值,获取指定音乐分类中所有音乐资料并展示出来。首先完成页面布局的设计,一般而言一个网站整体布局以及各页面风格应该一样,因此 MusicList 页面保留了首页的页首(Logo 和导航)、页脚(版权),只是将中间的内容替换成了完整的音乐资料列表。中间布局表格部分的 HTML 代码如下:

```
<table>
  <tr>
    <td>
      <table>
        <tr>
          <td><img src="Images/musiccd.jpg" /></td>
          <td><h1>小明的音乐库</h1></td>
        </tr>
      </table>
    </td>
    <td>
      <a href="Default.aspx">首页</a> |
      <a href="CategoryMgr.aspx">分类维护</a> |
      <a href="MusicMgr.aspx">资料维护</a> |
      <a href="SearchMusic.aspx">查找资料</a>
    </td>
  </tr>
  <tr>
    <td colspan="2">
        <a href="Default.aspx">首页</a> &gt;
        <asp:Label ID="lbCategory" runat="server" Text="当前分类" /><br /><br />
        <asp:Literal ID="litMusicList" runat="server"></asp:Literal>
```

```
        </td>
    </tr>
</table>
```

2. 定义通用工具类

后台代码基本上和 Default.aspx 页面的 BuildMusicList()方法代码相同，因此考虑将 BuildMusicList()方法代码挪到一个通用工具类中，增加数据库连接对象和 Literal 对象的参数。非页面代码应该放在 App_Code 文件夹中，如果还没有这个文件夹，则右击解决方案，在弹出的快捷菜单中选择“添加 ASP.NET 文件夹”命令添加。右击 App_Code 文件夹，在弹出的快捷菜单中选择“添加新项”命令，然后在对话框中选择“类”模板，输入名称 CommonTools.cs。

将 CommonTools 类修改成静态类，并将 Default.aspx.cs 中的 BuildMusicList()改造后作为 CommonTools 类的静态方法，代码如下：

```
using System.Data.SqlClient;          //引入处理 SQL Server 数据库的类所在的命名空间
using System.Web.UI.WebControls;      //引入 Literal 控件类所在的命名空间
using System.Text;                    //引入 StringBuilder 类所在的命名空间
/// <summary>
/// 通用工具类
/// </summary>
public static class CommonTools
{
  /// <summary>
  /// 使用指定 sql 查询获取数据,构造音乐资料列表
  /// </summary>
  public static void BuildMusicList(String sql, SqlConnection dbConn,
    Literal litMusicList)
  {
    …//复制 5.3 节 BuildMusicLists()方法中的代码
  }
}
```

调用 CommonTools 类的 BuildMusicList()方法，并传递数据库连接对象和页面显示控件的代码为：

```
CommonTools.BuildMusicList(sql, dbConn, litMusicList);          //构造默认音乐列表
```

3. 数据库页面基类

MusicList 页面同样需要使用数据库连接对象，实际上每个访问数据库的页面都需要使用这个对象。考虑利用 OOP 的继承机制，把数据库连接对象放到一个自定义页面基类中，然后让所有需要用到数据库连接对象的页面类都继承这个基类。例如 MPMM 中，在 App_Code 文件夹中新建一个名为 DbPage 的类，并且继承自 System.Web.UI.Page 类，具体的代码为：

```
using System.Data.SqlClient; //引入 SqlConnection 类所在的命名空间
public class DbPage: System.Web.UI.Page
{
  private SqlConnection _dbConn = null;
  protected SqlConnection dbConn
  {
      …//复制 5.2 节中定义 dbConn 属性的代码
```

```
    }
}
```

注意将代码复制到 DbPage 类中后,修改 dbConn 属性的保护级别为 Protected,否则 DbPage 类的子类将无法访问这个属性。

4. 实现页面

将 Default.aspx 页面和 MusicList.aspx 页面的后台类原来继承的 System.Web.UI.Page 类改为继承 DbPage 类,如 Default.aspx 页面的后台类定义代码修改为:

```
public partial class _Default : DbPage
```

现在,可以编写 MusicList.aspx 页面的后台处理代码,由于该页面没有 PostBack 请求,所以无须区分是否是回发请求,具体代码如下:

```
public partial class MusicList : DbPage
{
  protected void Page_Load(object sender, EventArgs e)
  {
    string catId = Request["cat"];                                    //获取音乐分类 ID
    ShowCategoryName(catId);                                          //显示音乐分类名称
    String sql = String.Format(@"SELECT ms.ID, ms.Name, ms.Authors,
              ms.PublishDate, cn.Name AS MediaTypeName
      FROM Music ms JOIN CodeNames cn
        ON ms.MediaType = cn.Code AND cn.CodeFor = 'Music.MediaType'
      WHERE ms.Category_ID = {0}", catId);                            //拼装获取分类音乐资料的 SQL
    CommonTools.BuildMusicList(sql, dbConn, litMusicList);            //构造默认音乐资料列表
  }
  /// <summary>
  /// 根据指定的音乐分类 ID,显示音乐分类名称
  /// </summary>
  private void ShowCategoryName(string catId)
  {
    String sql = String.Format("SELECT Name FROM Category WHERE ID = {0}", catId);
    SqlCommand cmd = new SqlCommand(sql, dbConn);
    try
    {
      dbConn.Open();
      lbCategory.Text = cmd.ExecuteScalar().ToString();               //获取第 1 行第 1 列的值
    }
    finally
    {
      dbConn.Close();
    }
  }
}
```

5.5 实现动态详细资料页

音乐资料列表中每个音乐资料的名称是一个超链接,链接到 Music.aspx 页面,同时通过 URL 传递 ID 参数,其值为对应音乐资料的 ID 字段值。

1. 页面布局

Music.aspx 页面用来显示音乐资料的详细信息，其布局可以直接使用第 2 章中的静态详细资料页面，将其中的静态文本都替换成对应的 ASP.NET 控件，代码如下：

```
<form id="form1" runat="server">
<div>
  <a href="Default.aspx">首页</a> &gt;
  <asp:HyperLink ID="lnkCategory" runat="server" Text="当前分类" /> &gt;
  音乐详细资料<br /><br />
  <table>
    <tr>
      <td rowspan="4">
        <asp:Image ID="imgPhoto" runat="server" Width="200" />
        <asp:Literal ID="litPlayer" runat="server"/>
      </td>
      <td><asp:Label ID="lbName" runat="server" /></td>
      <td><asp:Label ID="lbMediaTypeName" runat="server" /></td>
    </tr>
    <tr>
      <td colspan="2"><asp:Label ID="lbAuthors" runat="server" /></td>
    </tr>
    <tr>
      <td colspan="2"><asp:Label ID="lbDescription" runat="server" /></td>
    </tr>
    <tr>
      <td colspan="2"><asp:Label ID="lbPublishDate" runat="server" /></td>
    </tr>
  </table>
</div>
</form>
```

上述代码中，Label 控件显示普通文本，Image 控件显示图片，HyperLink 控件显示超链接，而 Literal 控件则用于显示音乐播放器。

2. 显示详细资料

显示音乐资料详细信息的后台代码如下：

```
public partial class Music : DbPage
{
  protected void Page_Load(object sender, EventArgs e)
  {
    String musicId = Request["id"];                    //获取指定音乐资料的 ID 值
    int catId = -1;                                    //初始化音乐分类 ID 值为 -1

    SqlCommand cmd = new SqlCommand();                 //创建 SqlCommand 对象
    cmd.Connection = dbConn;                           //设置 cmd 对象的数据库连接属性
    try
    {
      dbConn.Open();                                   //打开数据库连接
      //获取音乐资料详细信息，并用于设置各详细信息显示用的控件
      String sql = String.Format(@"SELECT ms.ID, ms.Code, ms.Name, ms.Photo,
          ms.Authors, ms.PublishDate, ms.MediaType, ms.MediaFile,
          cn.Name AS MediaTypeName, ms.Description, ms.Category_ID
```

```
            FROM Music ms JOIN CodeNames cn ON ms.MediaType = cn.Code
              AND cn.CodeFor = 'Music.MediaType'
            WHERE ms.ID = {0}", musicId);
          cmd.CommandText = sql;
          SqlDataReader dr = cmd.ExecuteReader();
          if (dr.Read())
          {
            //索引 3 为 Photo 字段.先检测 Photo 字段值是否为空,如果是则显示默认图片
            imgPhoto.ImageUrl = dr.IsDBNull(3) ? "Images/Music.png" : dr.
            GetString(3);
            lbName.Text = dr.GetString(2);                    //获取 Name 字段值
            lbMediaTypeName.Text = dr.GetString(8);           //获取 MediaTypeName 字段值
            lbAuthors.Text = dr.GetString(4);                 //获取 Authors 字段值
            lbDescription.Text = dr.IsDBNull(9)?"": dr.GetString(9);
                                                              //获取 Description 字段值
            lbPublishDate.Text = dr.IsDBNull(5) ? "" : dr.GetDateTime(5).
            ToShortDateString();                  //获取 PublishDate 字段值,格式化日期型为字符串
            if (dr.GetString(6) == "0" && !dr.IsDBNull(7))  //如果是数字化音乐且音乐文件存在
            {
              //嵌入音乐播放器代码(客户端需要安装 Windows Media Player 9.0 或以上版本)
              litPlayer.Text = string.Format(
                      @"< br /> < embed width = 200 height = 50 src = '{0}' hidden = 'no'
                      controls = 'smallconsole'
                      autostart = 'false' loop = 'false' >", dr.GetString(7));
            }
            catId = dr.GetInt32(10);                          //记录音乐资料的音乐分类 ID 字段值
          }
          dr.Close();                                         //关闭 DataReader 对象
          if (catId > 0)                                      //如果成功获取音乐分类 ID 字段值
          {
            //获取音乐分类名,设置返回对应音乐分类的音乐资料列表的超链接
            sql = String.Format("SELECT Name FROM Category WHERE ID = {0}", catId);
            cmd.CommandText = sql;
            lnkCategory.Text = cmd.ExecuteScalar().ToString();          //设置超链接名
            lnkCategory.NavigateUrl = String.Format("MusicList.aspx?cat = {0}", catId);
                                                                        //设置超链接指向的 URL
          }
        }
        finally
        {
          dbConn.Close();
        }
      }
    }
```

对上述代码补充说明几点:

(1) Music 页面类继承了 DbPage 自定义页面基类。

(2) 使用了 Get×××()方法来读取 DataReader 对象返回的数据,对于值可能为 NULL 的字段先用 IsDBNull()方法判断是否为空,如果为空,则值用空串代替。

(3) 因为只获取一条记录,所以用了 if(dr.Read())而不是 while(dr.Read())。

(4) 获取音乐资料和音乐分类的不同 SQL 语句,是使用同一个 DbCommand 对象执行的。

5.6　发布 MPMM 网站

发布 MPMM 到 IIS

完成开发，使用 VS 调试通过后，可以按照第 2 章中发布网站的方法，将新的 MPMM 网站发布到 IIS 中，但可能会出现无法访问数据库的错误。因为 MPMM 的数据库连接选择了“使用 Windows 身份验证”的模式，也就是集成身份验证模式。所谓集成身份验证模式，实际上指用运行应用程序的 Windows 账号作为访问 SQL Server 的账号。

在 VS 中调试网站时，这个账号就是当前登录到 Windows 中的用户账号，通常也是开发人员安装 SSE 的管理员账户。SQL Server 默认已经把 Windows 的管理员账户添加到了数据库系统中，而且设置为 DBMS 的管理员，所以访问数据库不会有任何权限问题。

一旦发布到 Web 服务器上，当前用户就不再是登录到 Windows 中的用户账户了，因为如果不是为了维护服务器，通常是不会有用户登录到服务器中的。所以，Web 服务器会默认以一个 Windows 内置账户来运行，该账户默认无权访问 SQL Server。

解决这个问题有很多途径，例如：

(1) 修改连接字符串，用 SQL Server 身份认证方法，此时需要对 SQL Server 进行相应的配置。

(2) 将该内置账户添加到 SQL Server，并授予访问 MpmmDB 数据库的权限。

网络环境下的权限配置是一个经常需要解决的问题，由于目前尚未涉及数据库权限管理的内容，这里介绍一种开发阶段的临时解决方法。

1. IIS 管理器

IIS 网站的配置需要通过 IIS 管理器进行。打开 IIS 管理器(Windows 管理工具→Internet 信息服务(IIS)管理器)，展开左侧的树形目录，可以看到“应用程序池”和“网站”节点，如图 5-2 所示。

2. 应用程序池

所谓应用程序池，可以简单理解为网站的运行环境，每个网站必须关联到某个应用程序池。网站运行的基本环境，如所使用的.NET Framework 版本、Windows 账户都是由所属的应用程序池决定的。

假设 MPMM 网站发布到了 C:\Inetpub\wwwroot 下，也就是发布成为了默认 Web 站点(Default Web Site)，而不是子站点。选中默认站点，在管理界面右侧出现的选项中，选择“基本设置”命令，出现如图 5-3 所示的对话框。

在图 5-3 中，可以通过单击“选择”按钮来选择网站所属的“应用程序池”，可以看到 MPMM 网站使用了名为 DefaultAppPool 的应用程序池。

3. 修改运行账户

在图 5-2 中，选择左侧的“应用程序池”节点，IIS 管理器列出所有可用的应用程序池，右击 DefaultAppPool 应用程序池，在弹出的快捷菜单中选择“高级设置”命令，弹出如图 5-4 所示的对话框。找到“标识”属性，将其值修改成安装 SSE 时的管理员账户。现在网站运行时就会用该账户去连接 SQL Server，而该账户拥有数据库管理员权限。

图 5-2　IIS 管理器

图 5-3　网站基本设置对话框

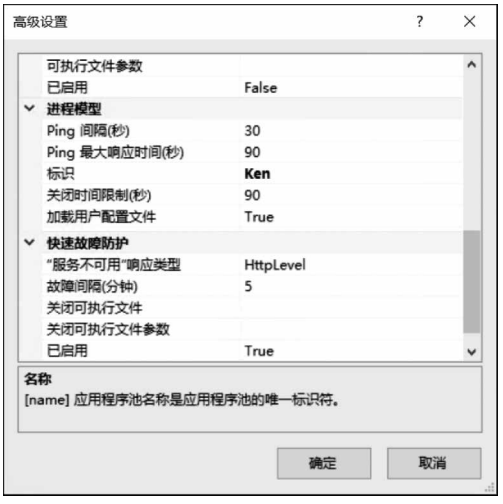

图 5-4　应用程序池高级设置对话框

注意：该账户通常权限过大，因此仅可用于开发阶段的网站运行调试。

习　题　5

一、选择题

1. ADO.NET 是(　　)。

A) Microsoft.NET Framework 的别名

B) .NET 开发环境中的数据库引擎，实现数据库的连接和访问

C) 一组 ASP.NET 专用的数据库访问对象集合

D) 一个高级的 SQL Server 图形化管理工具

2. 连接不同类型的数据源需要使用不同的 DbConnection 对象，其中(　　)用于连接 SQL Server 数据库。

A) SqlConnection　　B) OleDbConnection

C) OdbcConnection　　D) OracleConnection

3. 下面选项哪个不是 DbCommand 的执行方法(　　)。

A) ExcuteReader()　　B) ExcuteUse()

C) ExcuteNonQuery()　　D) ExcuteScalar()

4. MPMM 项目中，负责向 DBMS 发送 SQL 语句并取得返回结果的对象是(　　)。

A) 使用 String.Format()拼装出 SQL 语句的字符串对象 sql

B) 使用连接字符串的 DbConnection 对象 dbConn

C) DbCommand 对象 cmd

D) DbDataReader 对象 dr

5. 用户的(　　)操作会引起回发(Postback)。

A) 单击超链接访问某个 ASP.NET 页面

B) 在浏览器的地址栏中输入 ASP.NET 页面的 URL 地址，并按回车

C) 单击按钮，触发按钮的 Click 事件处理

D) 按住 Shift 按键的同时单击超链接

6. 使用 DataReader 对象读取数据库的标准步骤为(　　)。

①通过数据库读取对象获取数据；②执行 SQL 语句，获取数据库读取对象；③关闭数据库读取对象；④创建数据库命令对象；⑤使用连接字符串，创建数据库连接对象；⑥打开数据库连接；⑦关闭数据库连接；

A) ⑤④⑥②①③⑦　　B) ⑤①②③④⑥⑦

C) ⑤④②③①⑥⑦　　D) ⑤⑥④①②③⑦

7. SQL Server 的 SELECT 语句可以通过(　　)来表示仅获取前 n 行数据。

A) LIMIT n　　B) FIRST n　　C) ONLY n　　D) TOP n

二、填空题

1. 数据库连接对象可以连接不同的 DBMS，通过_____来告诉 ADO.NET 数据源在哪里，需要什么样的数据格式，提供什么样的访问信任级别以及其他相关的连接信息。

2. 数据库连接对象创建后并不会连接到数据库，通过调用数据库连接对象的方法建立

数据库连接。完成数据库操作后,应该及时调用_____方法断开连接。

3. 将数据库连接对象定义为页面基类属性的理由是_____。

三、是非题

(　　)1. 使用 DataReader 对象读取数据库中的数据,对于可能为 NULL 的字段应该先用 IsDBNull()方法判断字段值是否为空。

(　　)2. 应该使用 try…finally 异常处理语句确保执行断开数据库连接的方法被执行。

(　　)3. ASP.NET 可以给控件设置事件处理方法。例如,给按钮设置了 Click 事件处理方法,那么当用户单击按钮的时候,浏览器就会立刻执行该事件处理方法。

四、实践题

实现“个人通讯录”系统的“首页”“查找联系人”“联系人详情”页面。重新发布该系统到 IIS,排除发布后出现的所有错误。