第5章　使用数据库

本章要点：

- ADO简介
- 如何应用ADO使用数据库

5.1　ADO 简介

在当今提出的多种动态网页 (DHTML)解决方案中都强调了与数据库的连接，其实网页接挂后台数据库也是当前的热门应用，在电子商务等领域有着广泛的应用，就是说，如果你不能掌握在ASP中使用数据库，那么你就不能编写出功能强大的 ASP应用程序。ASP用Database Access组件与数据库进行连接，Database Access 组件通过 ActiveX Data Objects (ADO) 访问存储在数据库或其他表格化数据结构中的信息。

现在，Microsoft对应用程序访问各种各样的数据源所使用的方法是 OLEDB，OLEDB 介于ODBC层和应用程序之间。在你的 ASP 页面中，ADO介于ODEDB之上的“ 应用程序 ”。你的ADO调用首先被送到OLEDB，接着被送到ODBC层。OLEDB是一套组件对象模型 (COM)接口，但它是相当复杂的。这样，你需要一个连接应用程序与 OLE DB的桥梁，这就是ADO。而且，它支持开放式数据库连接 (ODBC) 标准的关系型数据库。其主要优点是易于使用、高速度、低内存支出和占用磁盘空间较少。 ADO 支持用于建立基于客户端/服务器和 Web 的应用程序的主要功能。

ADO 提供执行以下操作的方式：

5.1.1　连接到数据源

连接可以使应用程序访问数据源，这是交换数据所必须的环境。同时，可确定对数据源的所有更改是否已成功或没有发生。使用 Connection对象实现这一操作。

Connection对象代表与数据源进行的唯一会话。如果是客户端 /服务器数据库系统，该对象可以等价于到服务器的实际网络连接。

使用Connection 对象的集合、方法和属性可执行下列操作：

- 在打开连接前使用 ConnectionString、ConnectionTimeout 和 Mode 属性对连接进行配置。
- 设置 CursorLocation 属性以便调用支持批更新的“ 客户端游标提供者 ”。
- 使用 DefaultDatabase 属性设置连接的默认数据库。
- 使用 IsolationLevel 属性为在连接上打开的事务设置隔离级别。
- 使用 Provider 属性指定 OLE DB 提供者。
- 使用 Open 方法建立到数据源的物理连接。使用 Close 方法将其断开。

- 使用 Execute 方法执行对连接的命令，并使用 CommandTimeout 属性对执行进行配置。
- 可使用 BeginTrans、CommitTrans 和 RollbackTrans 方法以及 Attributes 属性管理打开的连接上的事务（如果提供者支持的话则可包括嵌套的事务）。
- 使用 Errors 集合检查数据源返回的错误。
- 通过Version属性读取使用中的 ADO执行版本。
- 使用 OpenSchema 方法获取数据库模式信息。

5.1.2 操作数据源

连接到数据库后，使用 Command 对象查询数据库并返回 Recordset 对象中的记录，以便执行大量操作或处理数据库结构。

使用 Command 对象的集合、方法和属性进行下列操作：

- 使用 CommandText 属性定义命令（例如，SQL 语句）的可执行文本。
- 通过 Parameter 对象和 Parameters 集合定义参数化查询或存储过程参数。
- 可使用 Execute 方法执行命令并在适当的时候返回 Recordset 对象。
- 执行前应使用 CommandType 属性指定命令类型以优化性能。
- 使用 Prepared 属性决定提供者是否在执行前保存准备好（或编译好）的命令版本。
- 使用 CommandTimeout 属性设置提供者等待命令执行的秒数。
- 通过设置 ActiveConnection 属性使打开的连接与 Command 对象关联。
- 设置 Name 属性将 Command 标识为与 Connection 对象关联的方法。
- 将 Command 对象传送给 Recordset 的 Source 属性以便获取数据。

5.1.3 得到数据

通过Command对象对数据源的操作，返回的记录全集用 Recordset 对象表示。Recordset 对象所指的当前记录均为集合内的单个记录。使用 ADO 时，通过 Recordset 对象可对几乎所有数据进行操作。所有Recordset对象均使用记录（行）和字段（列）进行构造。打开 Recordset 时，当前记录位于第一个记录（如果有），并且 BOF 和 EOF 属性被设置为 False。如果没有记录，BOF和EOF属性设置是True。可以使用MoveFirst、MoveLast、MoveNext 和 MovePrevious 方法以及Move方法和 AbsolutePosition、AbsolutePage和Filter属性来重新确定当前记录的位置。当使用Move方法访问每个记录（或枚举 Recordset）时，可使用BOF和EOF属性查看是否移动已经超过了Recordset的开始或结尾。

Recordset对象可支持两类更新：立即更新和批更新。使用立即更新，一旦调用 Update 方法，对数据的所有更改将被立即写入基本数据源。也可以使用 AddNew 和 Update 方法将值的数组作为参数传递，同时更新记录的若干字段。

如果提供者支持批更新，可以使提供者将多个记录的更改存入缓存，然后使用 UpdateBatch 方法在单个调用中将它们传送给数据库。这种情况应用于使用 AddNew、Update 和 Delete 方法所做的更改。调用 UpdateBatch 方法后，可以使用 Status 属性检查任何数据冲突并加以解决。

5.1.4 使用数据

Recordset 对象含有由 Field 对象组成的 Fields 集合。每个 Field 对象代表了 Recordset 对象中的一列。使用 Field 对象的 Value 属性可设置或返回当前记录的数据。

使用Field对象的集合、方法和属性可进行如下操作：

- 使用Name属性可返回字段名。
- 使用Value属性可查看或更改字段中的数据。
- 使用Type、Precision和NumericScale属性可返回字段的基本特性。
- 使用 DefinedSize 属性可返回已声明的字段大小。
- 使用 ActualSize 属性可返回给定字段中数据的实际大小。
- 使用 Attributes 属性和 Properties 集合可决定对于给定字段哪些类型的功能受到支持。
- 使用 AppendChunk 和 GetChunk 方法可处理包含长二进制或长字符数据的字段值。
- 如果提供者支持批更新，可使用 OriginalValue 和 UnderlyingValue 属性在批更新期间解决字段值之间的差异。

在打开Field对象的Recordset前，所有元数据属性（Name、Type、DefinedSize、 Precision 和 NumericScale）都是可用的。在此时设置这些属性将有助于动态构造其格式。

5.1.5 检测错误

ADO对象的操作也会产生一个或多个错误。每个错误出现时，一个或多个 Error 对象将被放到 Connection 对象的 Errors 集合中。当另一个 ADO 操作产生错误时，Errors 集合将被清空，并在其中放入新的 Error 对象集。可以及时地访问这个错误的集合，以便需要的时候进行更正。

Error对象的属性可获得每个错误的详细信息，包括以下内容：

Description属性，包含错误的文本。

Number属性，包含错误常量的长整型整数值。

Source属性，标识产生错误的对象。在向数据源发出请求之后，如果 Errors 集合中有多个 Error 对象，则将会用到该属性。

SQLState和NativeError属性，提供来自SQL数据源的信息。

某些属性和方法返回的警告以 Errors 集合中的 Error 对象的方式出现，但并不中止程序的执行。在调用 Recordset 对象的 Resync、UpdateBatch或CancelBatch方法，或Connection对象的 Open 方法，或者在设置 Recordset 对象的 Filter 属性之前，可通过调用 Errors 集合的 Clear 方法。这样就可以读取Errors集合的Count属性，并得到所返回的错误信息。

ADO通过Connection对象、Command对象、RecordSet对象来实现上面的操作。 ADO的对象模型如下所示：

Connection(对象)：—Errors(集合)—Error(对象)

Command(对象)：—Parameters(集合)—Parameter(对象)

RecordSet(对象)：—Fields(集合)—Field(对象)

其中Connection对象、 Command对象、 RecordSet对象和Field对象又分别具有Properties集合

而产生Property对象。下面是以上三大对象的相互关系：

```
Command.ActiveConnection->Connection
RecordSet.ActiveConnection->Connection
Connection.Excute->RecordSet
Command.Excute->RecordSet
RecordSet.Source->Command
```

下面分别介绍这三个对象。

5.2 Connection 对象

ASP使用ADO对各种数据源进行各种操作，其中， Connection对象是必不可少的，我们用Connection对象与各种数据源进行连接。

5.2.1 属性

1. CursorLocation属性

它的取值有两个，一个是 adUseClient,一个是adUseServer（默认），前者是使用客户端的游标，而后者是使用服务器端的游标。二着的差别在于 adUseClient游标可以提供供应商所没有提供的额外的属性，因而灵活性更大。需要注意的是 Connection对象与RecordSet对象均有此属性，由Connection对象产生的RecordSet对象会自动继承这个属性。另外，要让此属性对 Connection和RecordSet对象的实例起作用的话，必须在打开它们之前先作定义。下面看一个例子：

```
Set conn=Server.CreateObject("ADODB.Connection")
conn.CursorLocation=adUseClient
strConn ="driver={SQL Server};server=srv;"&_
"uid=sa;pwd=;database=pubs"
conn.open strConn
set rs= Server.CreateObject("ADODB.RecordSet")
rs.open "",conn,,,
```

采用此种方式则conn与rs的游标均为adUseClient了。

2. Attributes属性

它是Connection对象的特征，可读写。在Connection对象中，这个属性可以设置两个值：

adXactCommitRetaining 在调用CommitTrans方法后，自动启动新事务

adXactAbortRetaining 在调用RollbackTrans方法后，自动启动新事务

3. CommandTimeout属性

Connection对象命令执行所等待的时间，默认30秒，超过这个时间，将取消操作，可读写。

4. ConnectionString属性

在使用Connection对象的Open方法打开数据源时，连接参数的字符串，可读写。字符串中包含以下内容：

Provider	OLEDB提供者的名字
Data Source	指定数据源的名字

User ID	指定连接数据源时的用户ID
Password	指定连接数据源时用户的密码
File Name	指定要连接的数据库名字

5. ConnectionTimeout属性

创建连接时所等待的时间，默认15秒，可读写。

6. DefaultDatabase属性

当前连接的数据库的缺省名称，可读写。

7. Mode属性

这个属性指定了打开OLEDB数据处理源时读、写和共享权限。它可以是下列值之一：

adModeUnkown	未指定权限。
adModeRead	数据源被只读打开。
adModeWrite	数据源被只写打开。
adModeReadWrite	数据源被只读写打开。
adModeShareDenyRead	数据源以共享模式打开；而且不允许其他用户对数据源行读打开。
adModeShareDenyWrite	数据源以共享模式打开；而且不允许其他用户对数据源行写打开。
adModeShareExclusive	数据源以共享模式打开；而且不允许其他用户对数据源行读写打开。
adModeShareDenyNone	数据源以排它模式打开，即不允许共享。

8. Version属性

返回ADO的版本号。

5.2.2 方法

1. BeginTrans, CommitTrans, and RollbackTrans方法

BeginTrans方法用于开始一个新事务；CommitTrans方法，在使用这个方法之前，所有事务都在缓冲区，用以提高程序的工作效率，调用这个方法后，数据保存到数据库中；RollbackTrans方法用于取消当前的事务，就是说取消缓冲区中的数据，不把数据修改保存到数据库中。

2. Open，Close方法

Open方法用来打开一个对象与数据源的连接，而 Close方法用来关闭一个对象与数据源的连接。

Open语法为：

```
dbcon.Open Connectionstring,Username,Password
```

dbcon为我们创建Connection对象；Connectionstring是连接字符串；Username为用户名；Password为密码。

其中Connectionstring为我们所建立的数据源名，如果我们没有提供这些参数，则 ADO就用

下面的参数建立这些参数：

Provider	OLEDB提供者的名字
Data Source	指定数据源的名字
User ID	指定连接数据源时的用户ID
Password	指定连接数据源时用户的密码
File Name	指定要连接的数据库名字

Close语法为： dbcon.Close

Close方法关闭后，dbcon对象并不消失，只是释放资源。

3. Execute方法

这个方法来执行一个查询命令，如： dbcon.Execute SQLstr SQLstr是我们建立的查询字符串。

下面语句演示Execute方法的使用：

```
<%@ LANGUAGE = "VBScript" %>
<!-- #Include file="ADOVBS.INC" -->
<HTML><HEAD>
<BODY>

<%
Set OBJConnection = Server.CreateObject("ADODB.Connection")
OBJConnection.Open "Works"
```

注释：建立Connection对象，用Open方法建立与数据源"WORKS"的连接，Works是已经建立好的DSN文件。

```
SQLQuery = "SELECT * FROM TSGL"
Set RSCustomerList = OBJConnection.Execute(SQLQuery)
```

注释：用Execute方法执行上一行的SQL语句，得到记录集RSCustomerRList.

```
%>

<% Do While Not RSCustomerList.EOF %>
  <TR>
  <TD BGCOLOR="f7efde" ALIGN=CENTER>
     <FONT SIZE=1>
        <%= RSCustomerList("TSGL_MC")%>
  </FONT></TD>
<TD BGCOLOR="f7efde" ALIGN=CENTER>
  <FONT SIZE=1>
     <%= RSCustomerList("TSGL_CBS") %>
  </FONT></TD>

<%
RSCustomerList.MoveNext
```

注释：用循环的方法显示记录集RSCustomer中字程TSGL_MC和字段TSGL_CBS的所有内容。

```
Loop
RSCustomerList.Close
Set RSProductList = Nothing
Set OBJConnection = Nothing
%>
```

注释：释放对象。

```
</BODY>
</HTML>
```

5.3 Error对象

前面讲到了Connection对象是用于与各类的数据库进行挂接的，但在此过程中将会出现一些不可预测的错误，因而有了 Error这个对象。首先要清楚一个概念， Error对象是在连接数据库时产生的，而并非那些运行时的实时错误。也就是我们常用 On Error Resume Next来忽略的错误。这些错误将在 Error对象中用一个统一的模板来集中处理，后面会给出一个实例。下面先来看Error对象的属性和方法：

- Count属性：用来统计Errors集合的数目，它的特点与前面讲到的 Property对象的Count对象相同。
- Clear方法：写法为 Error.Clear，是用来清除 Errors集合中的原有对象的，在统计新的 Error对象时应该先使用此语句。
- Item方法：用来指定特定的一个错误，语法为 Error.Item(number)，其中number为一数字。由于Item为默认的方法，所以 Error(number)的写法与前面的写法是等价的。下面是一段程序，用来列举Error的所有对象。

```
<%
```

这个程序用来测试ADO的Error对象

```
DIM i
Set conn=Server.CreateObject("ADODB.Connection")
conn.ConnectionString="Driver={Microsoft Access Driver (*.mdb)};DBQ=" _
&Server.Mappath("/source_asp")&"/property/employee.mdb;"
conn.open
IF conn.errors.count>0 THEN
     Response.Write "connection to datebase cause problem!"&"<br>"
FOR i =0 to conn.errors.count-1
     Response.Write conn.errors.item(i)&"<br>"
     NEXT
ELSE
     Response.Write "connection to datebase successfully!"
END IF
conn.close
%>
```

5.4 Recordset对象

5.4.1 方法

1. Open，close方法

Open方法的用法：rs.Open sqlstr,con,Cursor,lock,Options

其中：rs是我们生成的Recordset对象；sqlstr是查询字符串；con是连接字符串，也可以是Connection对象；cursor是游标类型(后面属性里会讲到)；lock是加锁类型(后面属性里会讲到)；options是参数类型，可取俩个值:adCmdtext是SQL查询字符串，adCmdStoredProc 是存储过程。

Close方法用法为：rs.close　关闭Recordset对象。

2. AddNew方法

这个方法在数据库中添加一个空记录： rs.addnew

要想向空记录中加入数据，用以下方法：

```
rs(字段1)=值1
rs(字段2)=值2
......
rs.update
rs为建立好的Recordset对象。
```

3. Delete方法

rs.delete删除当前记录

4. Move方法

这个方法可以在记录集内漫游，用法为： rs.move nmb,Startrow。

其中：rs是Recordset对象，nmb是指你要移动多少行， Startrow是开始的行标签。

5. Movefirst、Movelast、Movenext、Moveprevious方法

Movefirst	把当前记录指针移动到表的开始。
Movelast	把当前记录指针移动到表的末尾。
Movenext	把当前记录指针向后移动一条记录。
Moveprevious	把当前记录指针向前移动一条记录。

6. Supports方法

这个方法是判断Recordset对象是否支持某个功能，用法为： Set boolstr=rs.Supports(options)。

其中：boolstr是返回的判断值，如支持某个功能则为“ true ”,否则为“ false ”。rs是Recordset对象，options取值如下：

Adaddnew	检查是否支持Addnew方法。
AdBookmark	检查是否支持书签。
Addelete	检查是否支持delete方法。
AdMovePrevious	检查记录集中指针是否可以向后移动。
AdResync	检查记录集是否可以被最新数据源更新。
AdUpdate	检查是否支持Update方法。
AdUpdateBatch	检查是否支持UpdateBatch方法。

7. Update、CancelUpdate方法

这俩个方法一个是更新数据，一个是取消更新。前者是对所做的修改保存，而后者是从缓冲区中除掉数据，取消更新。

8. UpdateBatch、CancelBatch方法

这俩个方法一个是成批更新数据，一个是取消成批更新。前者是对所做的所有修改做保存，

而后者是从缓冲区中除掉所有数据，取消更新。

9. Requery方法

对最初执行的查询再执行一遍。

5.4.2 属性

1. Bookmark 属性

我们在记录中有时需要快速定位某个访问过的记录，那么这个属性就是用来标志某个记录的，我们称它为书签属性。利用该属性我们可以很快地找到某个记录。它的用法是：

```
Set bookmark1=rs.bookmark
```

下次我们可以很快找到这个记录，如： rs.move 0,bookmark1

这样我们就把记录指针定位在上次设置书签的记录上。其中 :rs是Recordset对象。

2. CursorType属性

它返回当前记录集使用的游标类型，也可以由我们给当前记录集设置一个游标类型。在 ASP编程里与在VB中毕竟不太一样。在VB编程中，Recordset对象的所有属性和方法在默任状态下都可以使用，而在ASP编程中则不行。如：Recordset对象的Movelast() 方法和RecordCount属性在默任状态下是不能使用的，只有设置 CursorType属性为adOpenForwardOnly或adOpenDynamic时才可以使用。这个属性的取值如下：

adOpenForwardOnly	指针只能向前移动，其他人做的修改不可见。
adOpenKeyset	其它人的修改和删除是可见的，但添加的数据是不可见的。
adOpenDynamic	动态类型，其它人的修改是不可见的，但可以向前和向后移动指针。
adOpenStatic	静态类型，其它人的修改是不可见的，但可以向前和向后移动指针。

3. EditMode属性

这个属性返回当前的编辑模式，可取下列值之一：

adEditNone	当前没有编辑操作。
adEditinprogress	当前记录集中的数据以被修改。
adEditadd	当前正在进行添加记录操作。

4. Filter属性

这个属性用来过滤当前记录集中的记录，它的值可以取常量，也可以取字符串值。当它取字符串值时，相当于SQL的Select语句中的Where后面的条件，可以是任何表达式。当它取常量时是下面的值之一：

adFilterNone	当前的任何过滤都被取消。
adFilterPendingrecords	只返回修改被挂起的记录，即已修改但没有被保存的记录。
adFilterFetchedrecords	只返回最近获取的记录。

5. LockType属性

在ASP编程中缺省状态下是不能用 AddNew()这个方法的，必须把这个属性设置为adLockOptimistic，我们才可以用AddNew()方法向数据库中添加记录。它的取值如下：

adLockReadOnly	当前记录只读，不能修改。

adLockPessimistic 使用悲观锁。
adLockOptimistic 使用乐观锁。
adLockBatchOptimistic 对成批更新使用乐观锁。

6. Status属性

这个属性是用来检测当前的记录状态的，这个属性是只读的。它的取值如下：

adRecOk 记录修改成功。
adRecNew 表示这是一个新记录。
adRecModified 记录被修改。
adRecDeleted 记录被删除。
adRecUnModified 记录未被修改。
adRecPendingChanges 因为有挂起的插入操作，记录未被称为保存。
adRecCantRelease 因为加锁，所以记录不能更新换代。
adRecPermissionDenied 由于用户权限不允许修改记录。
adRecSchemaviolation 因结构不符，不能更新记录。

5.5 Command对象

一个Command对象代表了对数据源的操作命令的定义，利用它可以简化操作并提高效率。

我们可以用一个 Command对象查询数据库并将结果返回给一个 Recordset对象，执行一批命令或者是对数据库的结构进行操作。

应用Command对象的集合、方法和属性，我们可以做以下的事情：

用CommandText属性定义执行命令的文本（例如，一个 SQL语句）。
用Parameter对象和Parameters集合定义带参数的查询或存储过程。
用Execute方法执行一个命令并返回一个 Recordset对象。
用CommandType属性在执行命令之前确定命令的类型以优化性能。
用Prepared属性在执行命令之前确定命令是否已有编译好的版本。
用CommandTimeout属性设置执行一个命令时 provider等待的时间的秒数。
用ActiveConnection的设置将一个Command对象和一个打开的连接关联起来。
用Name属性将Command对象设置为相关 Connection对象的一个方法。

如果要在没有预先定义一个 Connection对象的情况下独立地创建一个 Command对象，可以将它的ActiveConnection属性设为一个有效的连接字符串。 ADO仍然创建了一个Connection对象，但没有赋予它一个对象变量。但是，如果要将多个 Command对象和一个对象连接关联起来，你必须显式地创建并打开一个 Connection对象，这会赋予该 Connection对象一个对象变量。如果没有将Command对象的 ActiveConnection属性设为此对象变量， ADO将会为每一个 Command对象创建一个新的Connection对象，即使它们使用了同样的连接字符串。

要执行一个定义好的命令，只需要在相关的 Connection对象中调用该命令的 Name属性就可以了。该命令的 ActiveConnection属性必须设为该 Connection对象。如果命令有参数，用该方法的参变量将值传递给它。

Command是可以用来做参数传递的。Command对象的批量参数传递，存储过程的执行等灵活而强大的功能也是它受到青睐的原因。Command对象主要是向SQL语句、存储过程传递参数，依靠SQL的强大功能来完成数据库的操作；而RecordSet对象可以说是微软重新封装了数据对象，并提供了一系列的方法和属性来简化数据库的编程。我们看下面的一个例子，它用了两种不同的方法实现了向数据库中增加一新的记录条。从中可以清楚地看到Command对象与RecordSet对象的不同点。

方法1（Command）：

```
Const adCmdText=&H0001
Const adInteger=3
Const adVarChar=200
Const adParamInput = &H0001
Set conn=Server.CreateObject("ADODB.Connection")
Set comm=Server.Createobject("ADODB.Command")
conn.open "Driver={ Microsoft Access Driver};DBQ="& _
Server.Mappath("/source_asp")&"/property/employee.mdb;"
Comm.ActiveConnection=conn
Comm.CommandType=adCmdText
Comm.CommandText="insert into employee (Job_ID,Fri_Name,Last_name)"& _
&"values(?,?,?)"
Set param=comm.CreateParameter("ID",adInteger,adParamInput,3,4)
Comm.Parameters.Append param
     Set param=comm.CreateParameter("FN",adVarChar,adParamInput,255,"bill")
Comm.Parameters.Append param
    Set param=comm.CreateParameter("LN",adVarChar,adParamInput,255,"Gates")
Comm.Parameters.Append param
Comm.Execute
conn.close
```

方法2（RecordSet）：

```
Const adCmdTable=&H0002
Set conn=Server.CreateObject("ADODB.Connection")
Set rs=Server.Createobject("ADODB.RecordSet")
conn.open "Driver={Microsoft Access Driver (*.mdb)};DBQ="& _
Server.Mappath("/source_asp")&"/property/employee.mdb;"
rs.ActiveConnection=conn
rs.open "employee",,,adCmdTable
rs.addnew
rs("Job_ID")=4
rs("Fri_Name")="bill"
rs("Last_Name")="Gates"
rs.update
rs.close
conn.close
```

从上面的例子就可以看出，这两个对象在处理一些问题上所用的方法不同的。RecordSet对象似乎更加好理解一些,但在性能上来讲Command的性能相对要优越些，但如果你是批量地加入

记录，你就能体会到第一种方案的好处了。下面详细介绍 Command对象的属性、方法和集合。

5.5.1 集合

Command对象有两个集合：Parameters和Properties。

5.5.2 方法

Command对象有两个方法：CreateParameter和Execute方法。

1. CreateParameter方法

该方法用指定的属性创建一个新的 Parameter对象。

语法：

```
Set parameter
=command.CreateParameter (Name,Type,Direction,Size,Value)
```

返回值：返回一个Parameter对象。

参数：

Name可选。一个代表Parameter对象的名字的字符串。

Type可选。一个Long值，确定Parameter对象的数据类型。请参看 Type属性。

Direction可选。一个Long值，确定Parameter对象的类型。请参看 Direction属性。

Size可选。一个Long值，确定参数值的最大长度，单位为字符或字节。

Value,varValue可选。一个Variant值，指示Parameter对象的值。

备注：

- 我们用 CreateParameter方法创建一个新的带有指定的名称、类型、方向、大小和值的 Parameter对象。你传递的参量都被写到相应的 Parameter属性中。该方法并不自动将该 Parameter对象附加到Command对象的Parameters集合中去。它只是设置附加的属性，在将该Parameter对象附加到集合中去时 ADO会验证它。
- 如果在Type参数中指定了一个可变长的数据类型，在将该 Parameter对象附加到Parameters集合之前，必须传递一个 Size参数或设置该对象的 Size属性，不然就会产生一个错误。

2. Execute方法

执行在CommandText属性中定义好的查询、 SQL语句或存储过程。

语法：

对一个返回行的命令：

```
set recordset = command.Execute(RecordsAffected,Parameters,Options)
```

对一个不返回行的命令：

```
command.Execute RecordsAffected.Parameters,Options
```

返回值：返回一个Recordset对象指针。

参数：

RecordsAffected可选。一个Long变量，provider用它返回该操作影响的记录的数量。

Parameters可选。一个 Variant数组，包含被 SQL语句传递的参数值。（在这个参数中传递，

输出参数不会返回正确值。）

Options可选。一个Long值，指示provider应该怎样看待Command对象的CommandText属性。这些属性有：

AdCmdText表明provider应该将CommandText看作一个命令的文本定义，例如一个 SQL语句。

AdCmdTable表明provider应该将CommandText看作一个表名。

AdCmdStoredProc表明provider应该将CommandText看作一个存储过程。

AdCmdUnknown表明CommandText中的命令是未知的。

请参考CommandType属性的说明以了解以上4个常数的详细解释。

备注：

- 我们用 Command对象的 Execute方法来执行在 CommandText属性中定义的查询。如果CommandText属性指定了一个返回行的查询，这个查询的结果保存在一个新的 Recordset对象中。有的编程语言当你不想得到 Recordset时允许忽略此返回值。如果该查询有参数，该Command对象的参数一直保持直到调用 Execute方法时传来新的值。在调用 Execute方法的时候，可以通过省略一些参数的新值的方法只覆盖参数集的一部分。定义参数和该方法传递参数的顺序应该一致。例如，如果有 4个（或更多）的参数，你只想给第一个和第四个参数传递新值，应该以（varl,,,var4）的形式传递数组作为 Parameters集合的参数。
- 当被放在Parameters集合中传递时，输出参数不会返回正确值。

5.5.3 属性

Command对象有 ActiveConnection、CommandText、CommandTimeout、CommandType、Name、 Prepared和State共7个属性，以下分别介绍之。

1. ActiveConnection

请参看Recordset对象的ActiveConnection属性说明。

2. CommandText

包含想要提交给provider命令的文本。

设置和返回值：设置或返回一个字符串变量，包含 provider命令例如SQL语句、数据表名或存储过程调用。默认为“ ”（零长度字符串）。

备注：

- 我们用CommandText属性设置或返回一个Command对象的方便。通常是一个SQL语句，但也可以是其它能被 provider识别的语句，例如存储过程调用。该 SQL语句一定要能被provider的查询处理器正确识别处理。
- 在设定CommandText属性的时候如果Command对象的prepared属性为True并且该Command对象被绑定到一个打开的连接，当调用 Execute或Open方法时ADO会准备好该查询（也就是说，一个该查询的存储好的已编译形态）。
- 根据CommandText属性的设置，ADO也许会更改CommandText属性。你能在任何时候读取CommandText属性以便查看在执行时ADO会使用的准确的命令内容。

3. CommandTimeout

指示在执行一个命令的时候，在终止尝试并报错之前要等待多长的时间。

提供给Command和Connection对象

设置和返回值：设置或返回一个 Long值，指示为一个命令等待多久，单位为秒，默认值为30。

备注：

- 我们用Connection对象或Command对象的CommandTimeout属性来允许一个Execute方法调用作废，原因可能是网络传输问题或服务器忙。如果在 CommandTimeout属性允许的时间间隔过去之后命令还没有执行完， ADO将会报错并终止该命令的执行。如果将该属性设为0，ADO将一直等待下去直到命令执行完成。要使用该属性，请确认使用的 provider和数据源支持CommandTimeout功能。
- Connection对象的 CommandTimeout属性设置对同一连接中的 Command对象的CommandTimeout设置没有影响；也就是说， Command对象的CommandTimeout属性不继承Connection对象的CommandTimeout属性的值。
- 在一个Connection对象上，CommandTimeout属性在打开连接之后保持可读写。

4. CommandType

指示Command对象的类型。

设置和返回值：设置或返回如表 5-1所列的CommandTypeEnum值：

表5-1 设置或返回的Command Type Enum

常 量	描 述
AdCmdText	将CommandText看作命令的文本定义
AdCmdTable	将CommandText看作数据表名
AdCmdStoredProc	将CommandText看作存储过程
AdCmdUnknown	默认。CommandText属性的命令类型未知

备注：

- 我们用CommandType属性来优化对CommandText属性的预处理。
- 如果CommandType属性值为adCmdUnknown（默认值），执行效率可能会降低，因为 ADO必须询问provider以确定CommandText属性是SQL语句、存储过程还是数据表。如果你知道使用的命令的类型，设置正确的 CommandType属性直接引导 ADO到相关的代码进行处理。如果CommandType属性不符合CommandText属性中的命令的类型，当调用 Execute方法时就会产生一个错误。

5. Name

指示一个对象的名称。

提供给Command、Field、Parameter和Property对象

设置和返回值：设置或返回一个字符串值。 Command和Parameter对象的该属性值可读写，Property和Field对象的该属性值只读。

备注：

- 我们用Name属性来赋予或取得一个Command、Field、Parameter或Property对象的名称。
- 对还没有附加到 Parameters集合的 Parameter对象， Name属性可读写。对已附加的Parameter对象和其它对象，Name属性只读。一个集合中的名称不一定唯一。
- 你能够通过顺序访问取得Name属性，之后就可以直接用名称访问该对象。

6. Prepared

指示在执行这前是否创建一个该命令的编译好的语句。

设置和返回值：设置或返回一个Boolean值。

备注：

- 我们用Prepared属性在一个Command对象第一次执行之前使 provider根据CommandText属性储存一个该查询的已准备（已编译）版本。这有可能延迟一个命令的一次执行，但一旦provider编译了一个命令，它就会在以后执行该命令时都使用这个已编译好的版本，这将会大大提高性能。
- 如果该属性为False,provider，则将会在不创建已编译版本的情况下直接执行该命令。
- 如果provider不支持该命令的执行，它会忽略任何要求编译该命令的请求并将 Prepared属性设为False。

7. State

请参看Connection对象中该属性的说明。

5.5.4 存储过程

在讨论StoredProcedure（存储过程）对象之前，我还要对 Command对象的Execute方法的作用进行一下阐述，一般来说使用 Command的Execute方法有3个目的。

- 用于进行一些简单的处理，例如删除一条记录：

```
Comm.CommandType=AdCmdText
Comm.CommandText="Delect From employee Where Job_ID=1"
Comm.execute
```

这样的工作不需要返回什么东西。

- 用于进行一些复杂的处理，这类一般都是和 StoredProcedure一同工作的，而且有输出和输入参数，这也是我们本节的讨论主题。
- 用于返回一个RecordSet对象，用于其它的处理，例如：

```
Comm.CommandType=AdCmdText
Comm.CommandText="Delect From employee Where Job_ID=1"
Set rs=comm.execute
DIM I
WHILE not rs.EOF
  FOR I=0 to rs.fileds.count-1
    Response.Write rs.fileds.item(I).value&","
  NEXT
  Response.Write "<br>"
```

```
  rs.MoveNext
WEND
```

那么StoredProcedure是什么呢？它是一个预先存储的数据库执行动作集，在 SQL的管理结构中，对于一个数据库有 2 部分：一个是数据表的集合，一个是 StoredProcedure的集合。将两者结合可以完成很多强大的功能。 StoredProcedure其实是对传统的SQL语句的一种扩展，主要是在参数的输入与输出上。下面我大致介绍一下 StoredProcedure的语法结构和与 Command对象的参数的传递问题。

StoredProcedure的标准写法：

```
Create Procedure Procedure_Name
   Define Parameter
As
   SQL Structure
```

上面的语法结构中， Procedure_Name为存储结构的的名字，也是你将在 Command中引用的名字。然后是定义输出和输入的参数。最后是一个 SQL结构化语句。下面是一个StoredProcedure的例子，它无需输入参数，也没有输出。

```
Create Procedure Del_User
As
   Delect From Employee Where Job_ID=1
```

如果我们要删除指定的 Job_ID，则我们需要给这个StoredProcedure输入参数。

```
Create Procedure Del_User1
         @intID int
As
  Delect From Employee Where Job_Id = @intID
```

这里的@intJob就是一个输入参数，它可以从外部接受输入的值，下面是给它输入的 asp程序：

```
Set conn=Server.CreateObject("ADODB.Connection")
Set comm=Server.CreateObject("ADODB.Command")
conn.ConnectionString="Driver={SQL Server};Server=ser;"&_
"uid=sa;psss=;database=employee"
conn.open
Comm.ActiveConnection=conn
Comm.CommandType=adCmdStoredProc
Comm.CommandType="Del_User这里的名字就是前面定义过的StoredProcedure的名字。
'下面就是参数的输入
param=comm.CreateParameter("ID",adInt,adParamInput,4)
'这里的adParamInput定义是最重要的。
Param.Value=1 这里的值可以输入你想要的值，也可以用 Request来获得
Comm.Parameters.Append param
Comm.Execute
```

这样我们就可以向StoredProcedure传递参数了。

有时在一个StoredProcedure中，还存在有输出的参数，下面是一个例子它返回一个 Job_ID确

定的Fri_Name的值

```
Create Procedure Get_fName
  @intID int
  @fName varChar Output说明为输出的参数
As
  Select @fName = Fri_Name Where Job_ID = @intID
它相应的asp程序也要改写为下面的形式
Set conn=Server.CreateObject("ADODB.Connection")
Set comm=Server.CreateObject("ADODB.Command")
conn.ConnectionString="Driver={SQL Server};Server=ser;"&_
"uid=sa;psss=;database=employee"
conn.open
Comm.ActiveConnection=conn
Comm.CommandType=adCmdStoredProc
Comm.CommandType="Get_fName"
'这里的名字就是前面定义过的StoredProcedure的名字。
'下面就是参数的输入
param=comm.CreateParameter("ID",adInt,adParamInput,4)
'这里的adParamInput定义是最重要的。
Param.Value=2 这里的值可以输入你想要的值，也可以用 Request来获得
Comm.Parameters.Append param
param=comm.CreateParameter("fName",adVarchar,adParamOutput,255,"")
'这里的adParamOutput定义是最重要的。说明它是一个输出的参数，默认的值      为一空的字符串
Comm.Parameters.Append param
Comm.Execute
Response.Write "Job_为ID&param(0)&"的员工的姓为"&param(1)
```

我给大家简单介绍了一下 StoredProcedure的基本概念，但StoredProcedure比较复杂，如果你想进一步的深入，可以参看 SQL Server的手册。

5.6 ASP与数据库连接

在Web服务器上运行的应用程序需要进行大量的服务器端数据库操作，而 ASP通过内嵌 Database Access组件实现对任何支持 ADO的数据源进行操作，包括 MS SQL Server、Access、Oracle等。

要使用ADO，必须用服务器端的包含 (SSI) 语句在 .asp 文件中包含进 ADO 常量的包含文件。在服务器端配置好 ASP后，ADO的常量被放置在“ \PROGRAM FILES\COMMON FILES\SYSTEM\ADO ”下。

如果用 VBScript 作为主脚本语言的，则包含文件为 ADOVBS.INC。如果用 JScript，则包含文件为 ADOJAVS.INC。

可采用下列语句将包含文件包含进 .asp 文件中：

```
<!--#include virtual="/PROGRAM FILES/COMMON FILES/SYSTEM/ADO/ADOVBS.INC"-->
或
<!--#includvei rtual="/PROGRAMES/COMMOFNILES/SYSTEM/ADO/ADOJAVAS.I
->
```

5.6.1 使用ODBC与数据库连接

要与ODBC兼容的数据库进行连接，必须建立一个 Data Source Name (DSN) 用以定位和标识特定的 ODBC 兼容数据库

通过 ODBC，您可以选择希望创建的 DSN 的类型：用户、系统 或 文件。用户和系统 DSN 存储在 Windows NT 注册表中。系统 DSN 允许所有的用户登录到特定的服务器上去访问数据库，而用户 DSN 使用适当的安全身份证明限制数据库到特定用户的连接。文件 DSN 用于从文本文件中获取表格，提供了对多用户的访问，并且通过复制 DSN 文件，可以轻易地从一个服务器转移到另一个服务器。

通过在 Windows 的“开始”菜单打开“控制面板”，您可以创建基于 DSN 的文件。双击“ODBC Data Sources”图标，然后选择“文件 DSN”属性页，单击“添加”，选择数据库驱动程序，然后单击“下一步”。按照后面的指示配置适用于您的数据库软件的 DSN。（见图 5-1）

图5-1 ODBC数据管理器

1) 配置 Microsoft Access 数据库的文件 DSN

在“创建新数据源”对话框中，从列表框选择“ Microsoft Access Driver”，然后单击“下一步”（见图5-2）。

键入您的 DSN 文件名，本处输入AccessDatabase，然后单击“下一步”。

单击“完成”创建数据源（见图 5-3）。

在“ODBC Microsoft Access 97 安装程序”对话框中，单击“选择”。选择 Microsoft Access 数据库文件 (*.mdb)，然后单击“确定”（见图5-4）。

本处选择的数据库文件包含一个表，名字叫“表 1”，包含两个字段“ Name”和“ Age”。

图5-2 选择ODBC数据源

图5-3 输入DSN文件名

2) **建立连接并输出所有记录的代码如下：**

```
<%@ LANGUAGE = "VBScript" %>
<%
Set objConnection = Server.CreateObject("ADODB.Connection")
objConnection.Open "Filedsn=AccessDatabase.dsn"
```

```
SQLQuery = "SELECT * FR表1"
Set rsCustomersList = objConnection.Execute(SQLQuery)
%>
<%Do Until rsCustomersList.EOF%>
<tr>
  <td bgcolor="f7efde" align=center>
    <%= rsCustomersList("Name")%>
    <%= rsCustomersList("Age") %>
    <br>
  </td>
</tr>

<%
rsCustomersList.MoveNext
Loop
%>

<%objConnection.close%>
```

以上代码建立了一个 Connection**对象，用** Connection**对象的** Open**方法打开在** ODBC**中文件** DSN**设置好的** Access**数据库，用** Connection**对象的** Execute**方法查询数据库中表** 1**的记录，用循环输出所有记录的内容。**

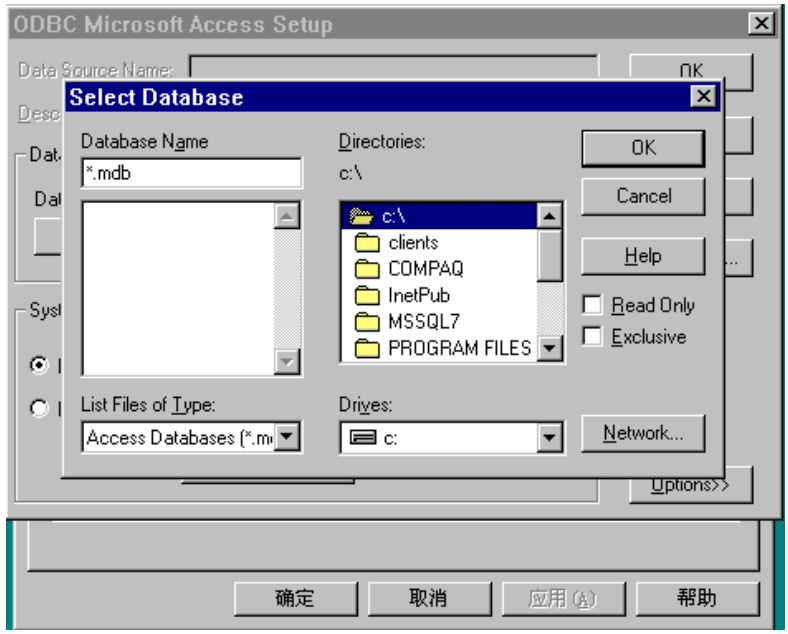

图5-4 选择要连接的数据库

在记事本中输入以上代码，保存为 List.asp**，用**HTTP**的方式打开浏览。**

3) **配置** SQL Server **数据库文件** DSN

如果数据库驻留在远程服务器上，请与服务器管理员联系，获取附加的配置信息；下面的过程使用 SQL Server 的 ODBC 默认的设置，它可能不适用于您的配置。

在“创建新数据源”对话框中，从列表框中选择“ SQL Server”，然后单击“下一步”。

键入 DSN 文件的名称，然后单击“下一步”。

单击“完成”创建数据源。

键入运行 SQL 服务程序的服务器的名称、登录 ID 和密码（见图5-5）。

在“创建 SQL Server 的新数据源”对话框中，在“服务器”列表框中键入包含 SQL Server 数据库的服务器的名称，然后单击“下一步”（见图5-6）。

选择验证登录 ID 的方式（见图5-7）。

如果要选择 SQL 服务器验证，请输入一个登录 ID 和密码，然后单击“下一步”。

在“创建 SQL Server 的新数据源”对话框中，设置默认数据库、存储过程设置的驱动程序和 ANSI 标识，然后单击“下一步”（见图5-8）。

在对话框（同样名为“创建 SQL Server 的新数据源”）中，选择一种字符转换方法，然后单击“下一步”。

在下一个对话框（同样名为“创建 SQL Server 的新数据源”）中，选择登录设置。

在“ODBC Microsoft SQL Server 安装程序”对话框中，单击“测试数据源”。如果 DSN 正确创建，“测试结果”对话框将指出测试成功完成（见图 5-9）。

图5-5 输入DSN文件名

4) 建立连接并输出所有记录的代码如下：

```
<%@ LANGUAGE ="VBScript"%>
<%
```

```
Set objConnection = Server.CreateObject("ADODB.Connection")
objConnection.Open"Filedsn=SqlServer.dsn"
SQLQuery = "SELECT * FROM Tsgl"
Set rsCustomersList = objConnection.Execute(SQLQuery)
%>

<%Do Until rsCustomersList.EOF%>

<tr>
  <td bgcolor="f7efde" align=center>
    <%= rsCustomersList("Tsgl_MC")%>
    <%= rsCustomersList("Tsgl_CBS") %>
    <br>
  </td>
</tr>

<%
rsCustomersList.MoveNext
Loop
%>

<%objConnection.close%>
```

本例中，表 Tsgl**包含** Tsgl_MC**和**Tsgl_CBS**两个字段，在记事本中输入以上代码，保存为** List.asp**，用**HTTP**的方式打开浏览。**

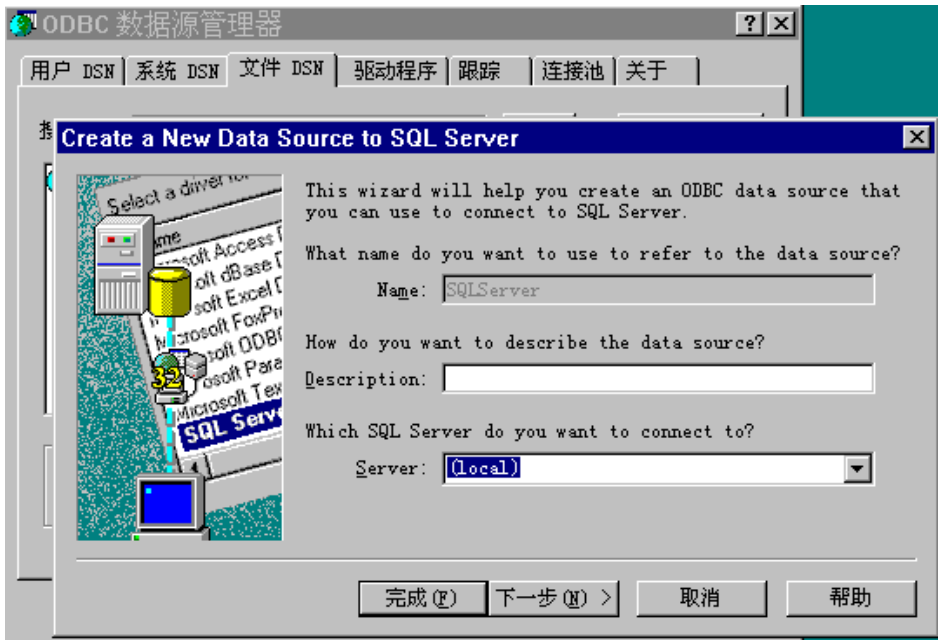

图5-6 选择SQL Server数据源

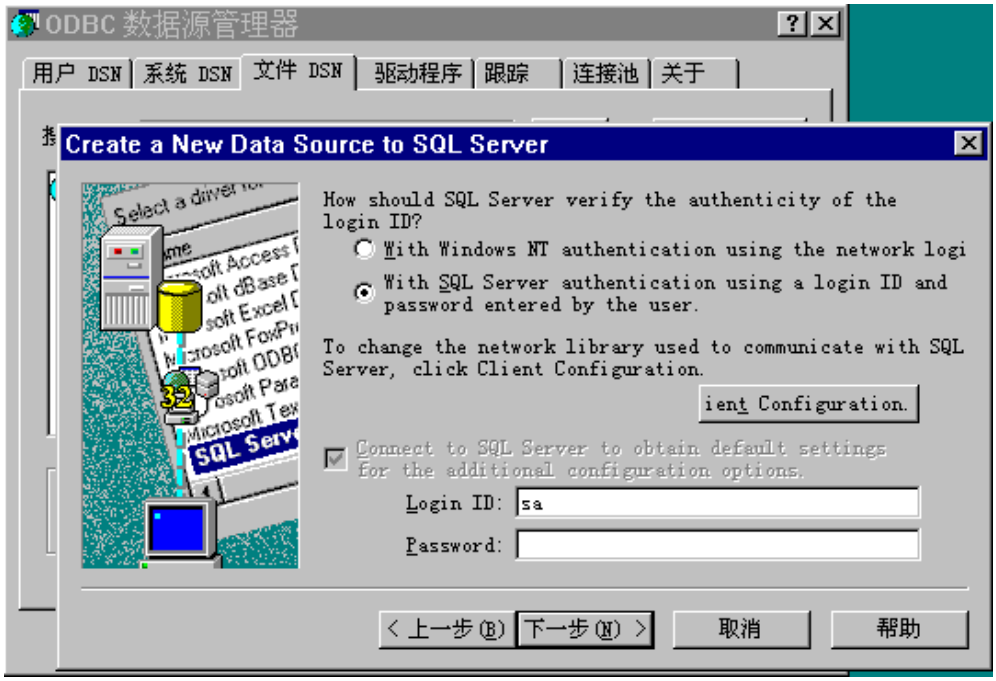

图5-7 选择SQL Server的登陆方式

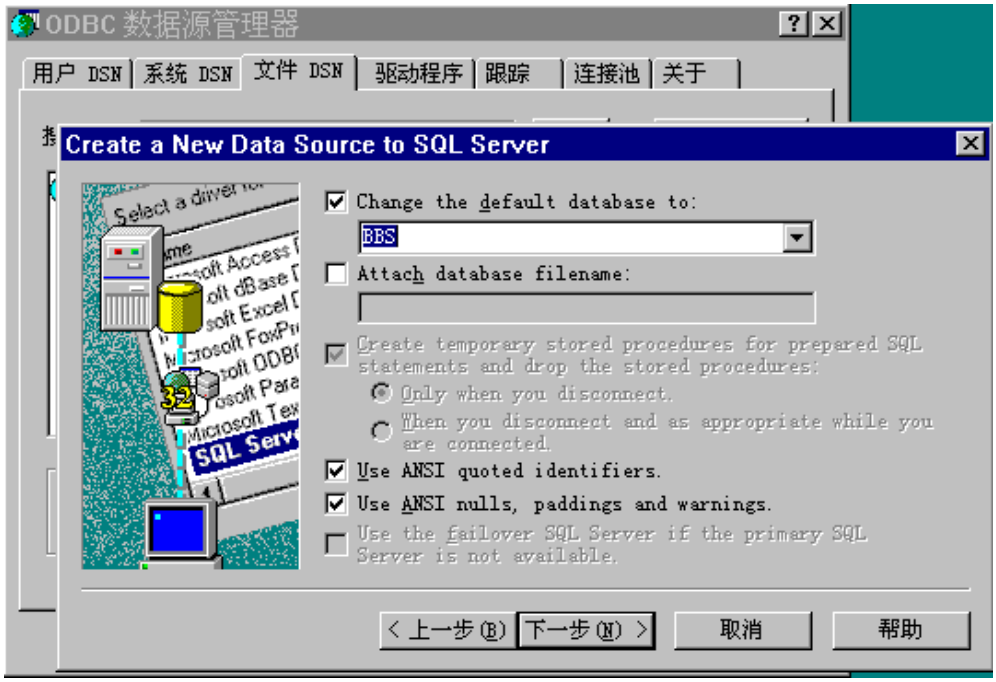

图5-8 选择数据库

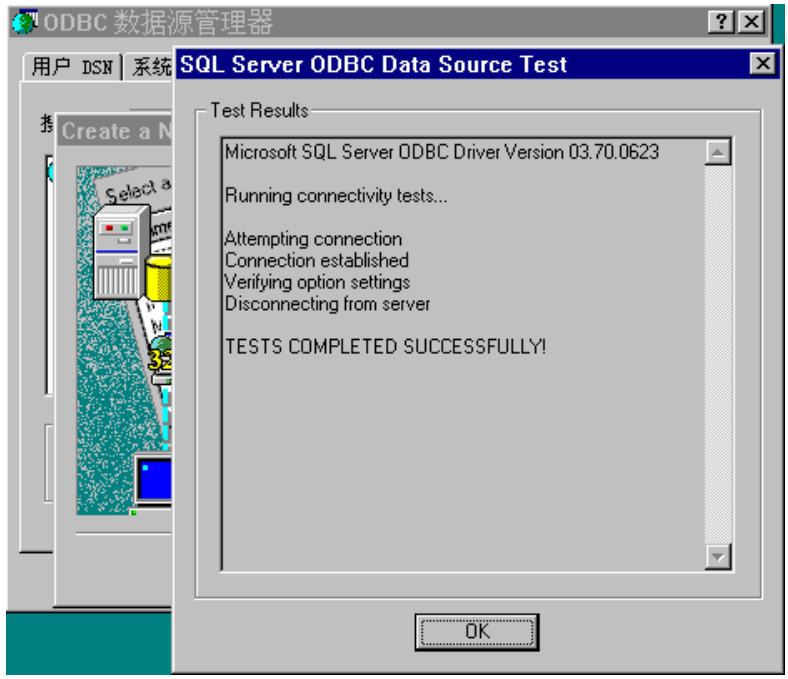

图5-9 检测SQL Server数据源

5.6.2 不使用ODBC与数据库连接

通过建立Connection对象及Recordest对象，ASP可以方便地与任何与 OLD DB兼容的数据库进行连接并进行查询、修改、更新等操作。

1) 与ACCESS数据库进行连接，代码如下：

```
<%@ LANGUAGE = "VBScript" %>
<!--#include file="ADOVBS.INC"-->
<%
Set objConnection = Server.CreateObject("ADODB.Connection")
DBPath = Server.MapPath("db1.mdb")
objConnection.Open "driver={Microsoft Access Driver (*.mdb)};dbq=" & DBPath
SQLQuery = "SELECT * FR表1"
Set rsRecordset = Server.CreateObject("ADODB.Recordset")
rsRecordset.Open SQLQuery,objConnection, adOpenKeyset, adLockReadOnly
%>
<%Do Until rsRecordset.EOF%>
<tr>
  <td bgcolor="f7efde" align=center>
    <%= rsRecordset("Name")%>
    <br>
    <%= rsRecordset("Age") %>
    <br>
  </td>
```

```
</tr>

<%
rsRecordset.MoveNext
Loop
%>

<%objConnection.close%>
```

在记事本中输入以上代码，保存为 list.asp**，用**HTTP**的方式打开并浏览。**

2) **与**SQL SERVER**数据库进行连接，代码如下：**

```
<%@ LANGUAGE = "VBScript" %>
<!--#include file="ADOVBS.INC"-->
<%
Set objConnection = Server.CreateObject("ADODB.Connection")
objConnection.Open "driver={SQL Server};server=jcserver;uid=sa;database=MIS;
SQLQuery = "SELECT * FROM TSGL"
Set rsRecordset = Server.CreateObject("ADODB.Recordset")
rsRecordset.Open SQLQuery,objConnection, adOpenKeyset, adLockReadOnly
%>
<%Do Until rsRecordset.EOF%>
<tr>
  <td bgcolor="f7efde" align=center>
    <%= rsRecordset("Tsgl_MC")%>
    <br>
    <%= rsRecordset("Tsgl_CBS") %>
    <br>
  </td>
</tr>
<%
rsRecordset.MoveNext
Loop
%>

<%objConnection.close%>
```

在记事本中输入以上代码，保存为 list.asp**，用**HTTP**的方式打开并浏览。**