

计算机编程技巧 ABC 系列丛书

Visual Basic 编程技巧

(网络与数据库篇)

清宏计算机工作室 编著



机械工业出版社

本书是为了让读者能够掌握 Visual Basic 常用技巧而编写的。包含了数据库与网络编程两大部分,涉及了相关的技巧。其中数据库包括: DAO 和 RDO 的使用、ADO 的运用、SQL 语言、数据的显示和报表的打印。网络编程包括:局域网、广域网、Internet 编程等。

本书包含有大量的实例,便于读者掌握 Visual Basic 编程技巧。适合有一定 Visual Basic 编程基础的计算机爱好者阅读。

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

责任编辑:边 萌 封面设计:姚 毅

责任印制:郭景龙

中国农业出版社印刷厂印刷·新华书店北京发行所发行

2001 年 5 月第 1 版第 2 次印刷

787mm×1092mm 1/16·22.75 张·560 千字

5 001—7 000 册

定价:40.00 元(1CD,含配套书)

ISBN7-900043-97-7/TP·93

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

本社购书热线电话(010)68993821、68326677-2527

前 言

随着计算机技术的飞速发展，计算机的开发迅速得到普及推广。当前有一大批计算机开发人员，虽然已经掌握了某种开发工具的基础知识、甚至达到了相当高的水平，但仍有很多盲区，或者急需更多的编程经验和技巧。为此，我们决定组织编写这套“计算机编程技巧 ABC 系列丛书”，以满足广大读者的需求，帮助读者快速成为计算机编程得行家里手。

这套丛书针对当前最流行的几种开发语言，介绍开发过程中的编程经验和技巧，并解决开发过程中的疑点和难点，主要突出“技巧”二字。针对某些开发语言，根据其内容相关性和独立性，我们编写了两个系列的书籍，即“网络与数据库篇”和“多媒体与系统篇”。这样可以使读者有更大的选择余地，并且在内容编排、读者阅读上都更为合理。

每本书按章节编写，各章由既相对独立，又相互关联的技巧主题组成，并且在顺序编排上考虑了内容的前后连续性。

针对每一个技巧主题，我们将其分为“ABC”三个部分进行讲解。

A——关键所在（或原理方法） 以简单明了的语言指出实现技巧的关键所在或原理方法。

B——实现与应用 以一个简单、完整实例向读者介绍技巧的实现方法或步骤。

C——专家点评 对技巧加以适当说明，包括补充解释，强调注意事项等。比如简要介绍实现该技巧的其他方法，或者介绍关键技术的扩展应用等。

针对每一个技巧主题，力求在内容讲解清楚的前提下，篇幅短小，使读者在有限篇幅内学到更多的技巧，整套书充分体现强调短小精悍、高效率的特点。

读者在阅读本套丛书之前，应该对相应开发语言有了一定了解，但并不要求读者开发水平一定很高，在章节编排和内容讲解过程中，我们都尽量照顾了水平还比较低的读者，因此初学者还是可以阅读这套丛书的。

希望这套丛书能使你很快成为软件开发的高手。

恳请广大读者对书中不足之处提出宝贵建议。

清宏计算机工作室

编者的话

本书是为了让读者能够掌握 Visual Basic 常用技巧而编写的。

本书是一本有关 Visual Basic 6.0 环境下应用程序开发的图书，书中通过一系列的实例向读者介绍如何使用 Visual Basic 6.0 的强大功能来开发各种应用程序，向读者展示了用 Visual Basic 6.0 进行程序开发的基本方法和相关技巧。实际上，当学习一个新的编程课题时，研究一个相关的例子可以在较短的时间内获得较大的效果。

本书深入浅出地主要介绍了 Visual Basic 关于数据库和网络编程的技巧，突出实用性、技巧性。为能充分发挥本书作用，建议读者充分掌握每一章节的实例，并尽可能地将这些技巧与实践结合起来。

由于这是一本介绍 Visual Basic 6.0 应用开发的书，所以书中没有从最基本的一般知识讲起，而是假定读者已经学习过 Visual Basic 编程语言，有过 Windows 编程经验。

关于光盘

光盘中包括本书大部分示例的完整代码，它们都是经过严格调试和测试的。代码按书中章节划分，对应相应的技巧。用户可以直接在工程中加载使用。使用范例如下：在 chapter3 目录下，打开文件夹 Sample02，点击工程 1，即将工程加载。有关的网络与数据库方面的详细设置，请读者参考文档中相应章节的说明。

与作者的联系方式

mtianhua@263.net

由于编者时间关系和水平有限，书中难免出错，希望读者谅解和批评指正。

目 录

前言

编者的话

第 1 章 DAO 和 RDO 的使用.....	1
1.1 DAO 中得到存储过程返回值	1
1.2 用 WithEvents 异步处理 RDO.....	3
1.3 RDO Stored Procedure 的呼叫.....	6
1.4 在 Visual Basic 中直接用 ODBC API 访问数据库	11
1.5 动态新增、删除 ODBC DSN.....	17
1.6 处理加了密码的 MDB 文件	20
1.7 RDO 访问 SQL 数据库出现的缺陷及改正.....	21
1.8 使用 RDO 输入数据	23
1.9 处理 RDO 的 MultiResultset.....	25
1.10 RDOConnection 部件的处理.....	26
1.11 RDO 数据的控制.....	32
第 2 章 ADO 的运用.....	36
2.1 运用 ADO 进行数据库表数据互导	36
2.2 ADO 的常用方法.....	38
2.3 处理 Select 语句中的单引号	41
2.4 用 ADO 创建 Access 数据库	42
2.5 ADO 的参数查询.....	43
2.6 ADO 设定独占性的数据库.....	45
2.7 ADO 对二进制数据的存取.....	49
2.8 使用 ADO 的网络功能	53
第 3 章 SQL 语言技巧.....	56
3.1 用 Visual Basic 和 RDO 访问 SQL Server.....	56
3.2 探讨用 Visual Basic 应用程序访问 SQL Server 的方法	63
3.3 列出 SQL Server 数据库中所有的存储过程	66
3.4 在 Visual Basic 中更改 SQL Server 数据库结构	67
第 4 章 数据显示和报表打印	72
4.1 将其他数据库资料转化为 MDB 文件	72
4.2 透过 Word 97 打印表格	73
4.3 关于数据报表的打印设置	75
4.4 动态更新数据报表	80
4.5 数据报表设计器在多层结构中开发的应用	82
4.6 快速读取 TextBox 第 N 行的数据	87
4.7 使用 FoxPro 数据库	88
4.8 动态调整 DBGrid 列宽度	89

4.9	Grid 控件的打印	90
4.10	Access97 的报表缺陷的解决	95
4.11	大量数据的输入	99
4.12	自动创建数据入口窗体	101
4.13	访问 ORACLE 存储过程	106
4.14	运行 Access 报表	107
第 5 章	网络编程	109
5.1	局域网相关技巧	109
5.2	广域网和 Web	111
5.3	获得主机名和 IP 地址	117
5.4	远程数据库的访问	122
5.5	编写小型的网络系统	129
5.6	通过 Visual Basic 获取网卡地址	136
5.7	开发通信软件的技巧	139
5.8	电话拨号	149
5.9	Visual Basic 串行通信	150
5.10	调用 Win95 API 实现串行通信的查询方法	154
5.11	编写标准 CGI 程序	161
5.12	实现文件上传	171
5.13	自动连接网络	173
5.14	系统登录	177
5.15	建立 Web Server	180
5.16	数据结构的传输转换	183
5.17	完成 PING 的任务	185
第 6 章	Internet 应用	193
6.1	对 Internet 进行操作	193
6.2	开发 IE	196
6.3	利用 IE 控件访问 Internet	200
6.4	设计 Browser 及 FTP 程序	202
6.5	设计 E-mail 程序	207
6.6	Winsock 控制的 UDP 协议的使用	210
6.7	访问 Internet	212
6.8	编写 BO 程序	214
6.9	创建自己的通信程序	217
6.10	设计聊天室	228
6.11	编写网络寻呼机	230
6.12	开发数据库浏览器	235
6.13	查看文件是否在 Internet Explorer 的 Cache 中	238

6.14	发送电子邮件附件	241
6.15	Whois 示例程序	250
6.16	连接默认的 Email 程序	252
6.17	DHTML 程序开发	254
第 7 章	高级应用	258
7.1	访问具有用户级安全的 Access 97 数据库	258
7.2	兼容非 Access 数据库	262
7.3	Excel 中 Basic 程序的运行	269
7.4	实现 Excel 自动获取外部数据	272
7.5	实现类电子表格的数据录入	277
7.6	播放 Flash 动画	280
7.7	操纵 OLE 服务器应用程序	280
7.8	超长 OLE 数据库字段的操纵方法	284
7.9	在 Access 数据库实现密码管理的一种方式	295
7.10	使用 ADO 压缩或修复 Microsoft Access 文件	297
7.11	将表中的数据导出到电子表格中	298
7.12	将文本文件转换为 Access 数据库	300
7.13	建立、改变及重构 Access 数据库	306
7.14	巧用 ClipBoard 建立图像数据库	309
7.15	调用 Office 97 技巧	311
7.16	用 ASP 编写网站统计系统	315
7.17	ASP 调用 SQL Server 存储程序	319
7.18	实现从 Excel 表到 AutoCAD 表转换	320
7.19	得到 Access 文件使用者列表	328
7.20	使 Access 数据库保持同步	330
7.21	ASP 与 Web 数据库查询	334
7.22	ASP 访问 SQL Server 内置对象	337
第 8 章	小技巧	340
8.1	加速数据库的访问速度	340
8.2	提高访问数据库的效率	340
8.3	处理数据库中的空字符	342
8.4	动态改变 Data 控件的属性	343
8.5	数据库的锁定	344
8.6	用类来编写数据库程序	347
8.7	关闭所有数据库对象	350
8.8	求出数据表记录总数	351
8.9	处理 DBF 数据库	352
8.10	压缩数据库	353

第 1 章 DAO 和 RDO 的使用

1.1 DAO 中得到存储过程返回值



原理方法

RDO 能够执行存储过程并得到返回值, DAO 却不行。而只要使用一些有创造力的 SQL 语句, DAO 也可以同样地使用这些存储过程。

RDO 通过 PreparedStatement 对象的 rdoParameters 集合让程序直接得到参数。DAO 没有相应的机制, 可是 SQL 有。下面的 Visual Basic 代码连接到一个 SQL Server, 建立一个带参数的存储过程, 运行该存储过程, 并得到返回值。



实现与应用

新建一个工程。打开菜单“工程”中的“引用”项, 选择一个合适的 DAO 引用(MS DAO 2.5/3.5 兼容 或 MS DAO 3.5, 增加一个 Command 按钮(Command1), 在 Command1_Click() 中增加下面的代码:

```
Private Sub Command1_Click()  
Dim db As Database  
Dim rs As Recordset  
Dim strConnect As String  
Dim strSQL As String  
Dim strResult As String  
'连接串  
  
strConnect = "ODBC;DSN=DSN_SP_TEST;"  
'打开数据库  
Set db = DBEngine.Workspaces(0).OpenDatabase("", False, False, strConnect)  
  
'如果该存储过程存在就删除  
strSQL = "if exists "  
strSQL = strSQL & "(SELECT * from sysobjects "  
strSQL = strSQL & "WHERE id = object_id('dbo.sp_ReturnParams') "  
strSQL = strSQL & "AND sysstat & 0xf = 4) "
```

```
strSQL = strSQL & "DROP PROCEDURE dbo.sp_ReturnParams"
```

```
MsgBox strSQL
```

```
db.Execute strSQL, dbSQLPassThrough
```

'建立存储过程

```
strSQL = "/****** Object: Stored Procedure dbo.sp_ReturnParams */"
```

```
strSQL = strSQL & vbCrLf & "CREATE PROCEDURE dbo.sp_ReturnParams"
```

```
strSQL = strSQL & vbCrLf & "/* Declare Parameters */"
```

```
strSQL = strSQL & vbCrLf & " @intInput int = Null, "
```

```
strSQL = strSQL & vbCrLf & " @strInputOutput varchar(20) = " & "Null OUTPUT,"
```

```
strSQL = strSQL & vbCrLf & " @strOutput varchar(20) = " & "Null OUTPUT"
```

```
strSQL = strSQL & vbCrLf & "AS"
```

```
strSQL = strSQL & vbCrLf & "BEGIN"
```

```
strSQL = strSQL & vbCrLf & " SET NOCOUNT ON /* stops messages */"
```

```
strSQL = strSQL & vbCrLf & " DECLARE @intReturn int "& "/* Declare return */"
```

```
strSQL = strSQL & vbCrLf & " SELECT @intReturn = 43 " & "/* Set the return */"
```

```
strSQL = strSQL & vbCrLf & " SELECT @strInputOutput = " & "'Param InOut Test'"
```

```
strSQL = strSQL & vbCrLf & " SELECT @strOutput = 'Param Out Test'"
```

```
strSQL = strSQL & vbCrLf & " RETURN @intReturn"
```

```
strSQL = strSQL & vbCrLf & "END"
```

```
MsgBox strSQL
```

```
db.Execute strSQL, dbSQLPassThrough
```

'准备调用

'在 SQL 中声明变量

'然后在一个 recordset 中得到返回值

```
strSQL = "SET NOCOUNT ON /* stops unwanted messages */"
```

```
strSQL = strSQL & vbCrLf & "/* Declare the variables */"
```

```
strSQL = strSQL & vbCrLf & "DECLARE @intIn int"
```

```
strSQL = strSQL & vbCrLf & "DECLARE @strInOut varchar(20)"
```

```
strSQL = strSQL & vbCrLf & "DECLARE @strOut varchar(20)"
```

```
strSQL = strSQL & vbCrLf & "DECLARE @intRet int"
```

```
strSQL = strSQL & vbCrLf & "/* Populate In Parameters */"
```

```
strSQL = strSQL & vbCrLf & "SELECT @intIn = 5"
```

```
strSQL = strSQL & vbCrLf & "SELECT @strInOut = 'In/Out In'"
```

```
strSQL = strSQL & vbCrLf & "/* Execute the procedure */"
```

```
strSQL = strSQL & vbCrLf & "EXECUTE @intRet = sp_ReturnParams "
```

```
strSQL = strSQL & " @intIn, @strInOut OUTPUT, @strOut OUTPUT"
```

```
strSQL = strSQL & vbCrLf & "/* Select back the records */"
```

```
strSQL = strSQL & vbCrLf & "SELECT @intRet AS param1, @intIn "  
strSQL = strSQL & "AS param2, @strInOut AS param3, @strOut AS param4"  
Set rs = db.OpenRecordset(strSQL, dbOpenSnapshot, dbSQLPassThrough)
```

'在 MsgBox 中显示

```
strResult = "intReturn = " & rs.Fields("param1")  
strResult = strResult & vbCrLf & "intInPut = " & rs.Fields("param2")  
strResult = strResult & vbCrLf & "strInPutOutput = " & _  
rs.Fields("param3")  
strResult = strResult & vbCrLf & "strOutput = " & _  
rs.Fields("param4")  
MsgBox strResult  
End Sub
```

在控制面板的 ODBC 中, 建立一个名称为 DSN_SP_TEST 的 DSN, 连接到 SQL Server。注意要有相关的权限。运行该工程。将连接到 SQL Server, 建立一个名称为 dbo.sp_ReturnParams 的存储过程, 传递参数并调用后, 在 MsgBox 得到返回值。



专家点评

上面的 SQL 代码使用的是 SQL Server 的 Transact SQL。其他不同数据库服务器可能在语法方面有所不同, 但原理是一样的。

1.2 用 WithEvents 异步处理 RDO



原理方法

平时使用 Dim, Public 等声明时, 可能忽略了 WithEvents 关键字, 本节将介绍该关键字, 并提供了一个简单演示和一个异步处理 RDO 的例子。

简单地说, 使用了 WithEvents 关键字后, 被声明的对象的所有事件都可以使用了。例如, 平时我们定义了一个 Dim t1 As TextBox, 你是不能在下拉框中发现 t1_Change 事件, 而使用了 Dim WithEvents t1 As TextBox 后, 你就能使用 t1_Change 和其他事件了。



实现与应用

1. WithEvents 简单演示

- (1) 新建一个工程。
- (2) 在 Form 中增加两个 TextBox(Text1 和 Text2)。

(3) 在 Form 的代码中增加:

```
Dim WithEvents t1 As TextBox
```

```
Private Sub Form_Load()
```

```
Set t1 = Text1
```

```
End Sub
```

```
Private Sub t1_Change()
```

```
MsgBox "你第一次修改了该文本框!"
```

```
Set t1 = Nothing
```

```
End Sub
```

```
Private Sub Text1_GotFocus()
```

```
Set t1 = Text1
```

```
End Sub
```

```
Private Sub Text2_GotFocus()
```

```
Set t1 = Text2
```

```
End Sub
```

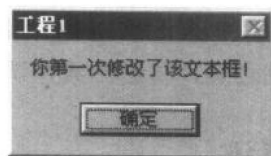


图 1-1 WithEvents 示例

(4) 运行程序。当第一次修改 TextBox 时, 就会出现一个对话框, 如图 1-1 所示。

2. 异步处理的 RDO

(1) 新建一个工程。在菜单“工程”的“引用”项中选中“Microsoft Remote Data Object 2.0”。

(2) 增加一个 CommandButton, 修改名称为 cmdOpenResultset。

(3) 把下面的代码增加到 Form 中。注意修改 gstrConnect 和 gstrSQL 串, 以连接到您的数据库。

```
Const gstrConnect As String = "DRIVER={SQL Server};" & "SERVER=YourServer;" & "DATABASE=pubs;" & "UID=sa;" & "PWD=;"
```

```
Const gstrSQL As String = "SELECT * FROM Authors"
```

```
Dim WithEvents MyConn As rdoConnection
```

```
Dim MyRS As rdoResultset
```

```
Private Sub Form_Load()
```

```
Set MyConn = New rdoConnection
```

```
MyConn.Connect = gstrConnect
```

```
MyConn.CursorDriver = rdUseOdbc
```

```
MyConn.EstablishConnection rdDriverNoPrompt, False, rdAsyncEnable  
cmdOpenResultset.Enabled = False  
Caption = "正在连接中..."  
End Sub
```

```
Private Sub cmdOpenResultset_Click()  
Set MyRS = MyConn.OpenResultset(gstrSQL, _  
rdOpenStatic, _  
rdConcurReadOnly, _  
rdExecDirect + rdAsyncEnable)  
cmdOpenResultset.Enabled = False  
Caption = "查询中.."  
End Sub
```

```
Private Sub MyConn_Connect(ByVal ErrorOccurred As Boolean)  
If ErrorOccurred Then  
Caption = "发生错误!"  
Else  
Caption = "已经建立连接!"  
cmdOpenResultset.Enabled = True  
End If  
End Sub
```

```
Private Sub MyConn_QueryComplete(ByVal Query As RDO.rdoQuery, _  
ByVal ErrorOccurred As Boolean)  
If ErrorOccured Then  
Caption = "发生错误!"  
Else  
Caption = "查询完成!"  
cmdOpenResultset.Enabled = True  
End IF  
End Sub
```

(4) 运行。



专家点评

在我们的第一个演示中，在 Form 中有两个 TextBox，在第一次修改 TextBox 时，就会出现一个对话框。利用了 WithEvents 后，代码变得很少。

在第二个例子中, 建立了一个到数据库的异步连接, 并运行了一个异步查询。当异步处理开始后, 在 Form 中的 Command 按钮就变灰了, 而在处理完毕后, Command 按钮就又恢复了。

应该发现, WithEvents 给我们带来了巨大的灵活性。

1.3 RDO Stored Procedure 的呼叫



原理方法

因为 SQL Server 与 Informix 的 Stored Procedure 语法不同, 使用方式也不太相同, 以致于通过 RDO 来呼叫时, 也有一些不同。

下面通过两个不同的 Procedure 来说明这些不同。



实现与应用

1. Informix 的 Stored Procedure 呼叫

假设有两个 Stored Procedure 如下:

```
CREATE PROCEDURE GETQPPFA (PCASENO CHAR(8),PSEQ INTEGER)
    RETURNING CHAR(1), CHAR(9), CHAR(12), DECIMAL(9,2);
```

```
DEFINE psys_kind CHAR(1);
```

```
DEFINE pnckm_code CHAR(9);
```

```
DEFINE phel_code CHAR(12);
```

```
DEFINE PUNIT_PRICE DECIMAL(9,2);
```

'CASE_NO + SEQ 是 UNIQUE 的 KEY 值

```
SELECT sys_kind, nckm_code, hel_code, unit_price
    into psys_kind, pnckm_code, phel_code, punit_price
    from qppfa
```

```
Where case_no = PCASENO and seq = PSEQ;
```

```
RETURN psys_kind, pnckm_code, phel_code, punit_price;
```

```
END PROCEDURE;
```

```
CREATE PROCEDURE GETQPPFA2 (PCASENO CHAR(8))
```

```
    RETURNING CHAR(1), CHAR(9), CHAR(12), DECIMAL(9,2);
```

```
DEFINE psys_kind CHAR(1);
```

```
DEFINE pnckm_code CHAR(9);
```

```

DEFINE phel_code CHAR(12);
DEFINE PUNIT_PRICE DECIMAL(9,2);

FOREACH
    SELECT sys_kind, nckm_code, hel_code, unit_price
        into psys_kind, pnckm_code, phel_code, punit_price
        from qppfa
        Where case_no = PCASENO
    RETURN psys_kind, pnckm_code, phel_code, punit_price WITH RESUME;
END FOREACH;
END PROCEDURE;

```

如果使用像 SQL Server 一般的方式，即

```

sql = "{ Call GETQPPFA ( ?, ? ) }"
Set qry = cn.CreateQuery("MyQuery",sql)
qry.rdoParameters(0) = "E8701761"
qry.rdoParameters(1) = 1
qry.Execute

```

那么，不会有记录集传回来，这是正常的，因为 sql 字符串中只有两个问号，代表传入的两个参数，而没有传出参数(如果把 OutPut 的参数也加进来，即多 4 个问号，就会出错)。

这时候该以如下方式来做：

```

Private WithEvents cn As rdoConnection
Private en As rdoEnvironment
Private rs As rdoResultset
Private qry As rdoQuery

Dim connstr As String
Set en = rdoEnvironments(0)
Set cn = New rdoConnection

cn.CursorDriver = rdUseServer
'或者设定为 rdUseNone 也可以
connstr = "UID=abc;PWD=abc234;Database=abc@eis;" _
    + "Driver={OpenLink Generic 32 Bit Driver};" _
    + "Host=192.168.0.230;" _
    + ";FetchBufferSize=30" _

```

```

+ ";NoLoginBox=Yes" _
+ ";Options=" _
+ ";Protocol=TCP/IP" _
+ ";ReadOnly=No" _
+ ";ServerOptions=" _
+ ";ServerType=Informix 7.2"
cn.Connect = connstr

cn.EstablishConnection rdDriverNoPrompt, False
Dim sql As String
sql = "Execute procedure getqppfa ('E8701761', 1)"
Set qry = cn.CreateQuery("MyQuery", sql)
Set rs = qry.OpenResultSet(rdOpenForwardOnly, rdConcurReadonly, rdExecDirect)

```

记录可以当作一个 Row 的 Resultset 传回。需要强调, 如果是 rdUseServer 方式, 则只能用 rdOpenForwardOnly 与 rdConcurReadonly 的方式来开启, 否则不会传回任何东西。另外, 有没有设定 rdExecDirect 在此不会有影响, 都会有传回值。

如果想要以参数查询的方式来做, 就得如下操作:

```

Private WithEvents cn As rdoConnection
Private en As rdoEnvironment
Private rs As rdoResultset
Private qry As rdoQuery

Dim connstr As String
Set en = rdoEnvironments(0)
Set cn = New rdoConnection

cn.CursorDriver = rdUseServer
'或者 设定为 rdUseNone 也可以
connstr = "UID=abc;PWD=abc;Database=abc@eis;" _
+ "Driver={OpenLink Generic 32 Bit Driver};" _
+ "Host=192.168.0.230;" _
+ ";FetchBufferSize=30" _
+ ";NoLoginBox=Yes" _
+ ";Options=" _
+ ";Protocol=TCP/IP" _
+ ";ReadOnly=No" _
+ ";ServerOptions=" _

```

```

+ ";ServerType=Informix 7.2"
cn.Connect = connstr

cn.EstablishConnection rdDriverNoPrompt, False
Dim sql As String
sql = "Execute procedure getqppfa2 ( ? )"
'设定是参数查询的方式
Set qry = cn.CreateQuery("MyQuery", sql)
qry.rdoParameters(0) = "E8701761"
'给定参数
Set rs = qry.OpenResultset(rdOpenForwardOnly, rdConcurReadOnly)
'do something
qry.rdoParameters(0) = "E8703069"
'重设参数
rs.Requery
'重新查询

```

在这个例子中，Resultset 的 Row 不只一笔，使用的方式没有什么不同，但这是使用参数查询的方式，而参数查询的 OpenResultset 方法，不可以使用 rdExecDirect 参数，否则没有办法做参数型态的转换工作。如果真的使用了，不会有错，只是没有记录集传回。而如果想改用其他的数查询，那么在改变参数值后使用 rs.Requery 便可以了，不要再使用 OpenResultset 重新开启，否则 RDO 会建立一个新的暂时性 Stored Procedure 并执行该查询(若使用了 rdExecDirect 则不会建立暂时性 Stored Procedure)。

2. SQL Server 6.5 的 Stored Procedure 呼叫

假设有个 Stored Procedure 如下：

```

CREATE PROCEDURE getname
    @name CHAR(8), @pseq INT,
    @OUTVAL INT OUTPUT
AS
    SELECT @OUTVAL = (Select Count(*) from herimit where name= @greatwall)
    IF @OUTVAL > 0
        RETURN 1
    ELSE
        RETURN 0
GO

CREATE PROCEDURE getname2
    @name CHAR(8)

```

```
AS
    SELECT country, name, age ,address
        FROM herimit
        WHERE name = @greatwall
GO
```

相对应的 STORED PROCEDURE 呼叫如下: (传回值的呼叫方式)

```
Private WithEvents cn As rdoConnection
Private en As rdoEnvironment
Private rs As rdoResultset
Private qry As rdoQuery
```

```
Dim connstr As String
Set en = rdoEnvironments(0)
Set cn = New rdoConnection
```

```
cn.CursorDriver = rdUseServer
connstr = "DSN=SQLSRV;UID=abc;PWD=abc234;"
cn.Connect = connstr
```

```
cn.EstablishConnection rdDriverNoPrompt, False
```

```
Dim sql As String
sql = "{ ? = call GETQPPFA (?, ?, ?) }"
```

'一共有四个参数,第一个接收 Stored Procedure 的 Return 之值
'第 2~4 个参数分别对应 @name , @age , @OUTVAL

```
Set qry = cn.CreateQuery("MyQuery", sql)
qry.rdoParameters(1) = "E8701761"
qry.rdoParameters(2) = 1
qry.rdoParameters(3).Direction = rdParamOutput
```

'设定第三个参数是 OUTPUT

```
qry.Execute
```

```
Debug.Print qry.rdoParameters(0), qry.rdoParameters(3)
```

'如果有记录, 则 qry.rdoParameters(0)传回 1, qry.rdoParameters(3)传回记录数

'传回 Resultset

```
Private WithEvents cn As rdoConnection
Private en As rdoEnvironment
```