jBASE Web Builder – VB and SQL

INTRODUCTION

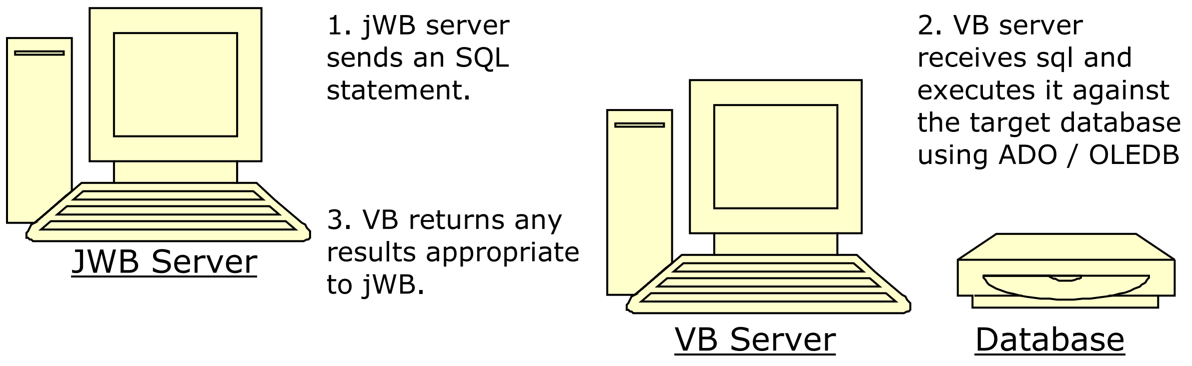
jBASE for web builders is a fully featured environment for creating interactive and transactional web applications. Using the power of Microsoft's Visual Basic may extend this functionality.

One way to extend web builder is to add the facility to execute SQL statements against remote databases. This document explains the principles behind this integration. It includes instructions on how to build a jWB project and a VB project from scratch which can interact together and process SQL statements.

The example projects described in this document require jWB v3.2.1 or above, and Microsoft Visual Basic 6. These products need to be installed on the same machine. If they are on different machines, the IP address in the examples will need to be changed from 127.0.0.1 to the IP address of the VB server.

OVERVIEW

jBASE for web builders versions 3.2.1 and above come with a built in socket calling interface which allows communication via TCP/IP. By writing function calls using this socket interface, jWB can be easily integrated with other applications or programming languages.


1. jWB server sends an SQL statement.
2. VB server receives sql and executes it against the target database using ADO / OLEDB
3. VB returns any results appropriate to jWB.
JWB Server
VB Server
Database

Each conversation must be instantiated by jWB. Although the diagram shows the jWB server, VB server and database server as separate machines, they can co-exist on one machine.

BUILDING THE WEB BUILDER PROJECT

The web builder application is going to consist of a simple page consisting of a query text box, a button to fire off the query, and labels to display the results.

Start up jWB and add a new Application Module. Give it a name of "sql". Add a new page to the module. Name the page "pagsql".

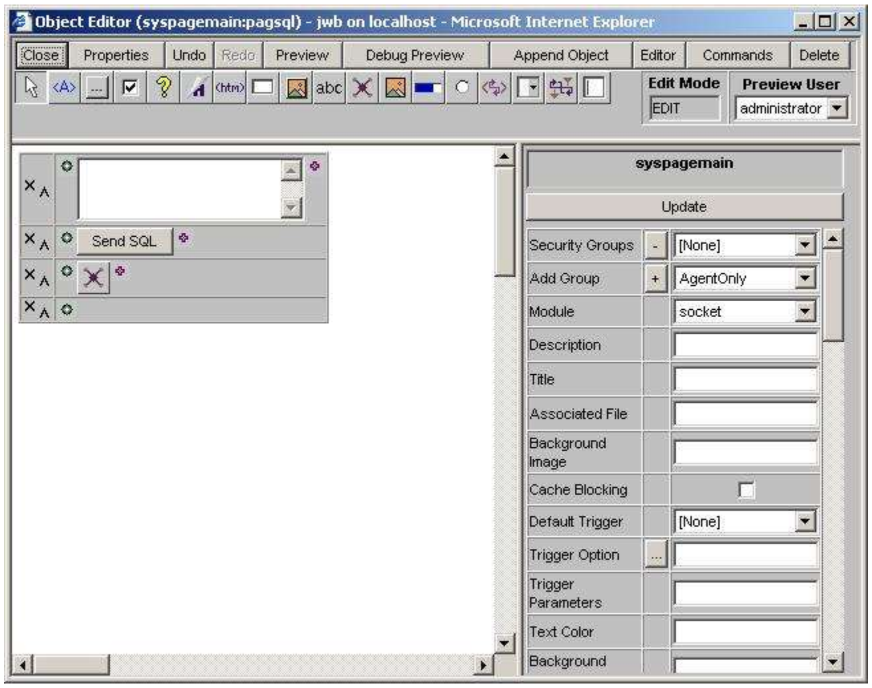
Add a button and a textbox to the page. Change the properties of the button as follows;

Caption	Send SQL
Button Type	Submit
Trigger Type	sub
Trigger Option	sendsqlmsg

Change the properties of the textbox as follows;

Datafield	named,sql

Add a table to contain the results. Give the table a name by changing the object property to tabsql. The screen should look something like this;



Object Editor (syspagemain:pagsql) - jwb on localhost - Microsoft Internet Explorer

Edit Mode	EDIT
Preview User	administrator
Security Groups	[None]
Add Group	AgentOnly
Module	socket
Description	
Title	
Associated File	
Background Image	
Cache Blocking	☐
Default Trigger	[None]
Trigger Option	
Trigger Parameters	
Text Color	
Background	

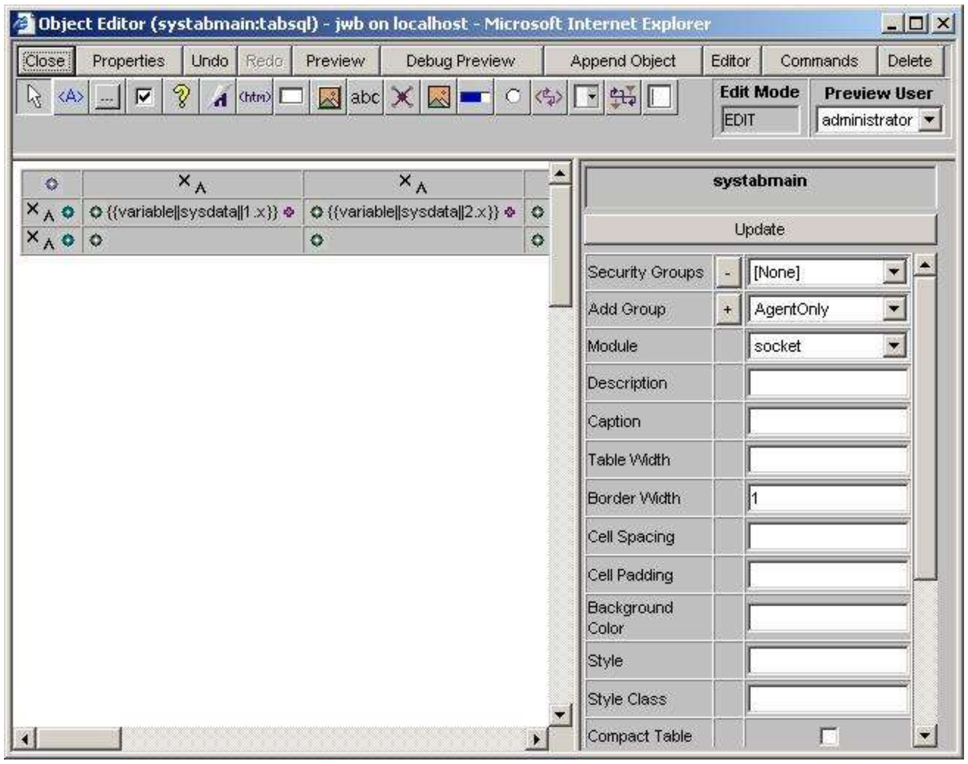
Open up the table in it’s own window by either double clicking on it, or by clicking on the edit object button.

Add a number of labels. These labels are going to store the results of the query. Add one label for each column of data. Change the properties of the labels as follows;

Text	[{{variable||sysdata||1}}]
Dependent field	1
Text	[{{variable||sysdata||2}}]
Dependent field	1
Text	[{{variable||sysdata||3}}]
Dependent field	1

Etc.

The table should now look like this;



Edit Mode: EDIT
Preview User: administrator

{{variable||sysdata||1.x}} | {{variable||sysdata||2.x}}

systabmain

Security Groups	[None]
Add Group	AgentOnly
Module	socket
Description	
Caption	
Table Width	
Border Width	1
Cell Spacing	
Cell Padding	
Background Color	
Style	
Style Class	
Compact Table	☐

On the page, click on the Send SQL button, and then click on the ellipses button next to the trigger option parameter to open the sendsqlmsg routine.

Paste in the following routine;

```
      SUB sendsqlmsg(html)

*--- include the common block
      INCLUDE sysbp syscommon

*--- get the SQL statement
      LOCATE("sql",sysquery,1;pos) THEN
         sql = sysquery<2,pos>
      END

*--- Build parameter list for syssocket call
      syssock<syssockhost> = "127.0.0.1"
      syssock<syssockport> = 8000
      syssock<syssockout> = sql
      syssock<syssocktimeout> = 10

*--- Make socket call
      CALL syssocket

*--- Capture return value (or error)
      sysdata = RAISE(syssock<syssockin>)

*--- Re-build jWB page
      CALL syscreate(syspage,html)

      RETURN
```

This routine reads in whatever text has been typed into the textbox and then builds a set of parameters before calling the socket function. When the socket function returns, it loads sysdata with the values returned.

Once sysdata has been loaded, the page is re-created. The value of sysdata will be displayed in the labels on the page because of the DIRT tags.

The routine to make the socket call is syssocket. All parameters for the syssocket function are contained within the syssock common block.

The parameters for syssock are as follows;

syssock<syssockhost>	Hostname or IP address of machine running the server. For servers running on the same machine, use 127.0.0.1 or localhost.
syssock<syssockport>	TCP/IP port to connect to on the server. It is important that this port is not currently in use. Check http://www.iana.org/assignments/port-numbers for commonly used port numbers
syssock<syssocktimeout>	Timeout value in seconds for the conversation.
syssock<syssockout>	String value which will be sent to the server.
syssock<syssockin>	String value received from the server.
syssock<syssockerr>	If there has been an error interacting with the server, this will be set to non-zero.
syssock<syssockerrmsg>	If there has been an error interacting with the server, this will contain a description of the error.

BUILDING THE VISUAL BASIC SERVER

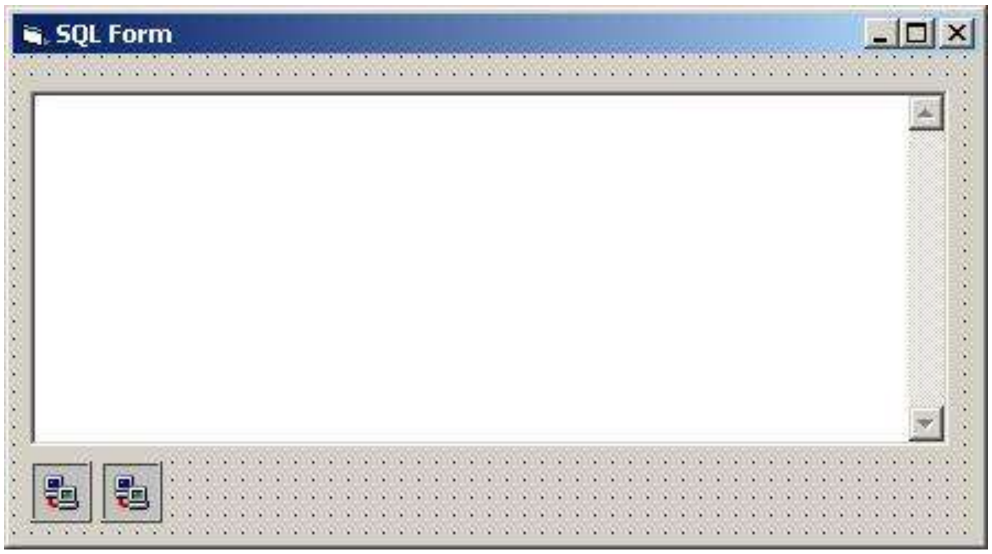
Open Visual Basic and create a new “Standard EXE” project. From the Project menu, choose the components option, and wait for the components dialog to appear. Scroll down, and tick the “Microsoft Winsock Control” component.

From the Projects menu, open up the references option, and from the components dialog, select the Microsoft ActiveX Data Objects option.

Rename form1 to frmSQL and change the caption property to SQL Form. Add a text box to the form. Change the multiline property of the text box to True and change the scrollbars property to “2 – Vertical”. Delete the text property, and change the name to txtReceived.

Add two Winsock controls to the form. Change the names to WinSockGeneral and WinSockListener. On the WinSockGeneral control, change the LocalPort property to 8000.

The frmSQL form should look something like this;



Add the following lines of code to the VB project; (Some lines may have scrolled onto more than one line in this document).

```
Option Explicit
Private DBConnection As ADODB.Connection

Private Sub Form_Load()

    ' Retrieve program settings and st things up
    Initialise

    ' Start listening for socket messages
    txtReceived = txtReceived & Now & " About to listen" & vbCrLf
    WinsockGeneral.Listen
    txtReceived.Text = txtReceived.Text & Now & " Listening" & vbCrLf

End Sub

Private Sub WinSockGeneral_ConnectionRequest(ByVal requestID As Long)

    ' Answer the connection request
    txtReceived = txtReceived & Now & " Connection Request" & vbCrLf
    If WinSockListener.State <> sckClosed Then WinSockListener.Close
    WinSockListener.Accept requestID

End Sub

Private Sub winsocklistener_DataArrival(ByVal bytesTotal As Long)

    Dim sql As String
    Dim ReturnValue As String

    ' Retrieve the SQL and report it
    WinSockListener.GetData sql, vbString
    txtReceived = txtReceived & Now & " DataArrival : " & sql & vbCrLf

    ' Send a reply
    ReturnValue = ProcessSQL(sql)
    txtReceived = txtReceived & Now & " ReturnValue : " & ReturnValue &
vbCrLf
    WinSockListener.SendData ReturnValue

End Sub

Private Sub winsocklistener_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile
As String, ByVal HelpContext As Long, CancelDisplay As Boolean)

    ' There's been an error so report it.
    txtReceived = txtReceived & Now & " Err: " & Description & vbCrLf

End Sub

Private Sub WinSockListener_SendComplete()

    ' Tidy up after message sent.
    txtReceived = txtReceived & Now & " Send Complete" & vbCrLf
    WinSockListener.Close

End Sub

Private Function ProcessSQL(sql As String) As String

    Dim sError As String
    Dim i As Integer
    Dim x As Integer
    Dim y As Integer
    Dim rs As ADODB.Recordset
    On Error GoTo ProcessSQL_Error

    ' Assume it's a SELECT statement for now...
    If DBOpenRecordset(rs, sql, adOpenForwardOnly, adLockReadOnly,
adCmdText) Then

        ' Loop through each field
        For x = 0 To rs.Fields.Count - 1

            ' Add the name of the field first
            ProcessSQL = ProcessSQL & rs.Fields(x).Name & Chr(252)

            rs.MoveFirst

            ' Loop through
            Do While Not rs.EOF

                ' Add each value
                ProcessSQL = ProcessSQL & rs.Fields(x).Value & Chr(252)
                rs.MoveNext

            Loop

            ProcessSQL = ProcessSQL & Chr(253)

        Next x

    Else
        ProcessSQL = "Error Processing SQL"
    End If

Exit Function
ProcessSQL_Error:

    sError = "Error initialising: " & Err.Description
    sError = sError & vbCrLf & Error$
    For i = 0 To DBConnection.Errors.Count - 1
        sError = sError & vbCrLf & DBConnection.Errors.Item(i)
    Next i

    txtReceived.Text = txtReceived.Text & sError
    Resume Next

End Function
```

```
Private Sub Initialise()

    Dim sDataSource As String
    Dim iPort As Integer
    Dim sError As String
    Dim i As Integer

    On Error GoTo Initialise_Error

    ' Get settings from SQL.INI
    sDataSource = GetProfileString("Settings", "DataSource", "",
App.Path & "\sql.ini")
    iPort = GetPrivateProfileInt("Settings", "Port", 4000, App.Path &
"\sql.ini")

    ' Open up ADO connection
    Set DBConnection = New ADODB.Connection
    DBConnection.Open sDataSource
    txtReceived.Text = txtReceived.Text & "Connection to " &
sDataSource & vbCrLf

    ' Set Port number to listen on
    WinsockGeneral.LocalPort = iPort
    txtReceived.Text = txtReceived.Text & "Port number " & iPort &
vbCrLf

Exit Sub
Initialise_Error:
    sError = "Error initialising: " & Err.Description
    sError = sError & vbCrLf & Error$
    For i = 0 To DBConnection.Errors.Count - 1
        sError = sError & vbCrLf & DBConnection.Errors.Item(i)
    Next i

    txtReceived.Text = txtReceived.Text & sError
    Resume Next
End Sub

Public Function DBOpenRecordset(rs As Recordset, sql As String,
CursorType As CursorTypeEnum, LockType As LockTypeEnum, lOptions As
Long) As Boolean
' Opens a recordset with error handling
' The recordset is returned as rs
' The function returns true if successful, in which case the calling
procedure should
' close the recordset when finished.
' sDesc is a description of the calling function, to put in an error
report

    Dim sError As String
    Dim i As Integer
    Dim bSuccess As Boolean

    On Error GoTo DBOpenRecordset_Error

    bSuccess = False
    Set rs = New ADODB.Recordset
    rs.CacheSize = 100

    rs.Open sql, DBConnection, CursorType, LockType, lOptions
    bSuccess = True

    DBOpenRecordset = bSuccess
Exit Function
DBOpenRecordset_Error:
    sError = sError & vbCrLf & "Error opening recordset"
    sError = sError & vbCrLf & Error$
    sError = sError & vbCrLf & "SQL : " & sql
    For i = 0 To DBConnection.Errors.Count - 1
        sError = sError & vbCrLf & DBConnection.Errors.Item(i)
    Next i

    txtReceived.Text = txtReceived.Text & sError
    Resume Next
End Function
```

Add a module to the project, and paste in the following lines of code;

```
Option Explicit

' INI file declarations
Declare Function GetPrivateProfileInt Lib "kernel32" Alias
"GetPrivateProfileIntA" (ByVal lpApplicationName As String, ByVal
lpKeyName As String, ByVal nDefault As Long, ByVal lpFileName As
String) As Long
Declare Function GetPrivateProfileString Lib "kernel32" Alias
"GetPrivateProfileStringA" (ByVal lpApplicationName As String, ByVal
lpKeyName As Any, ByVal lpDefault As String, ByVal lpReturnedString As
String, ByVal nSize As Long, ByVal lpFileName As String) As Long

Function GetProfileString(sSection As String, sEntry As String,
sDefault As String, sFilename As String) As String
' Gets an entry from an 'ini' file
' If sEntry is blank, then it returns all the entries in the section,
separated by null chars
' the maximum length of the returned string will be 128 characters

    Dim R       As Integer
    Dim sReturn As String

    sReturn = String$(128, " ")
    R = GetPrivateProfileString(sSection, sEntry, sDefault, sReturn,
128, sFilename)

    If R = 0 Then
        'MsgBox "Cannot find value [" & sEntry & "] in section [" &
sSection & "] in filename [" & sFilename & "]"
    End If

    sReturn = Trim$(sReturn)
    ' chop off the ascii 0 terminator
    sReturn = Left$(sReturn, R)
    GetProfileString = sReturn

End Function
```

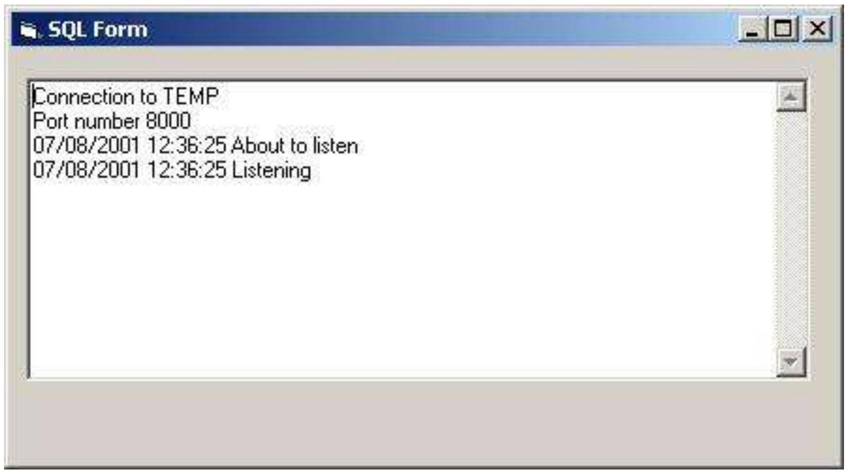
Add a file called sql.ini in the same directory as your VB project, and paste in the following lines;

```
[Settings]
DataSource=TEMP
Port=8000
```

The Datasource setting needs to be changed to the name of a datasource on your machine.

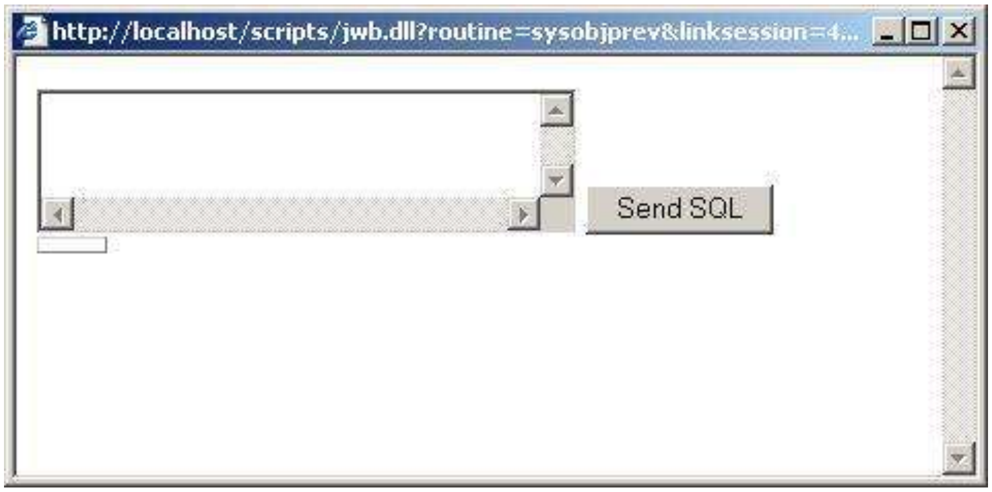
TESTING THE TWO PROJECTS

Start the Visual Basic project. After a brief pause, some text should appear in the text box on frmSQL. If everything is working correctly, the form should appear as follows;

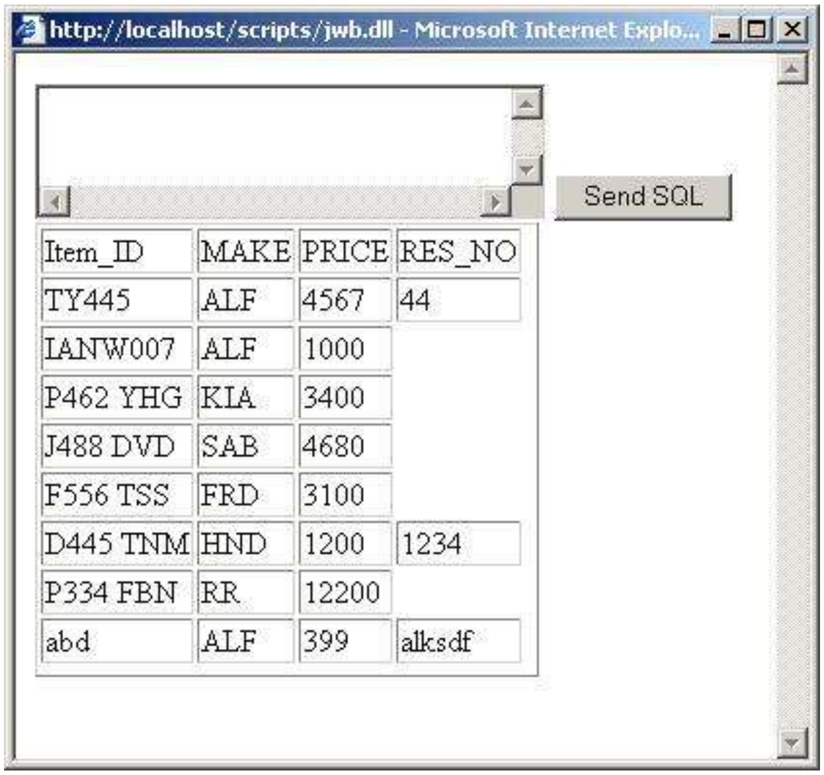


```
Connection to TEMP
Port number 8000
07/08/2001 12:36:25 About to listen
07/08/2001 12:36:25 Listening
```

Open the web builder sql project, and load the pagsql page. Click on the preview button. After a short pause, the page should appear as follows;

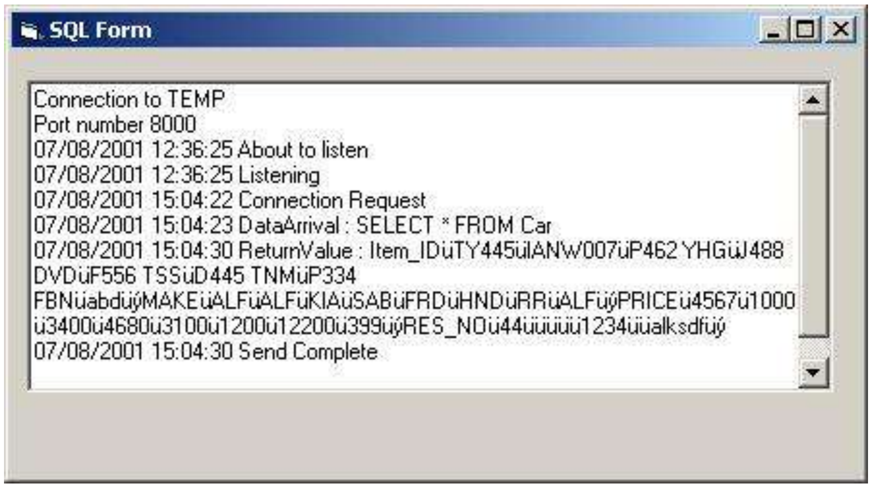

http://localhost/scripts/jwb.dll?routine=sysobjprev&linksession=4...
Send SQL

Type in a valid SQL statement in the text box and click on the “Send SQL” button. After a short delay, some data should appear.


http://localhost/scripts/jwb.dll - Microsoft Internet Explo...
Send SQL

Item_ID	MAKE	PRICE	RES_NO
TY445	ALF	4567	44
IANW007	ALF	1000	
P462 YHG	KIA	3400	
J488 DVD	SAB	4680	
F556 TSS	FRD	3100	
D445 TNM	HND	1200	1234
P334 FBN	RR	12200	
abd	ALF	399	alksdf

Switching to the Visual Basic program on the server should reveal more about the chain of events that caused the interaction between jWB and VB. As can be seen, there was a connection request event, followed by a Data arrival event and finally, there was a Send Complete event.



```
Connection to TEMP
Port number 8000
07/08/2001 12:36:25 About to listen
07/08/2001 12:36:25 Listening
07/08/2001 15:04:22 Connection Request
07/08/2001 15:04:23 DataArrival : SELECT * FROM Car
07/08/2001 15:04:30 ReturnValue : Item_IDüTY445üIANW007üP462 YHGüJ488
DVDüF556 TSSüD445 TNMüP334
FBNüabdüýMAKEüALFüALFüKIAüSABüFRDüHNDüRRüALFüýPRICEü4567ü1000
ü3400ü4680ü3100ü1200ü12200ü399üýRES_NOü44üüüüü1234üüalksdfüý
07/08/2001 15:04:30 Send Complete
```

Document Created 7th August 2001
Author: Martin Bailey