Contents

Introduction
Read Me First

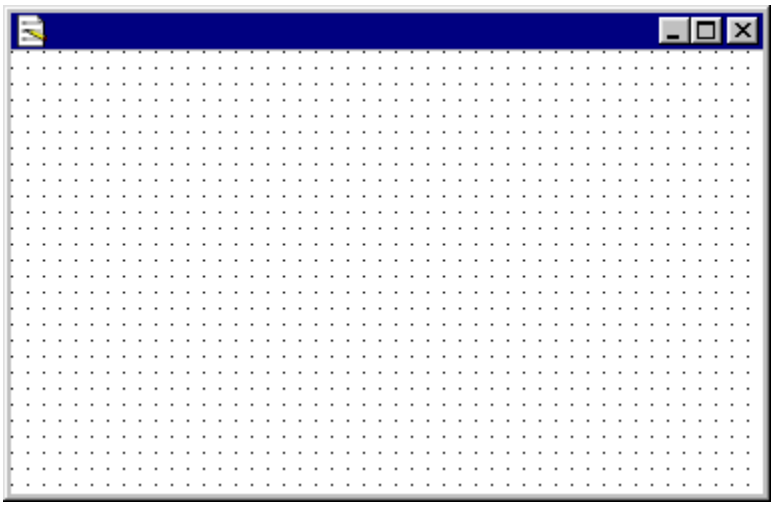
Form Design
Global Declarations
Load Event
PositionFirst Procedure
Vertical Scroll Bar Change Event
Sorting
Return A Value On Double Click

Visual Basic 4.0 and Above

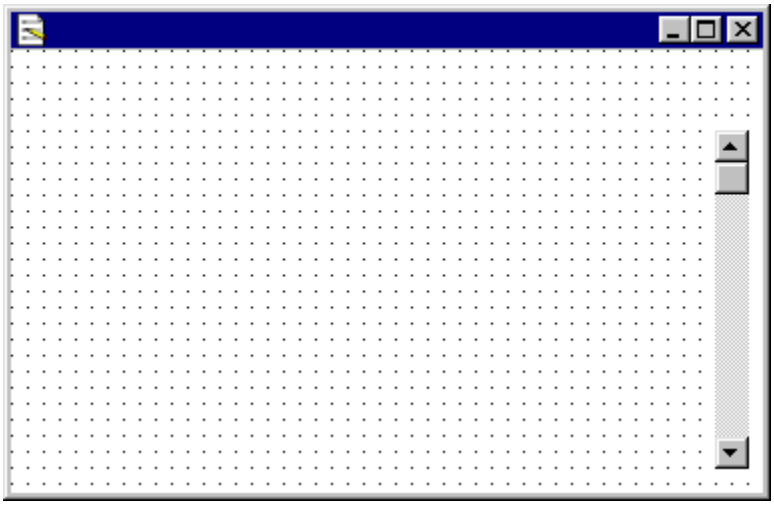
Form Design

It is up to you to determine the form layout. How many (columns) you want to include in the form? and how many (rows)?
I included in my example a simple 2 columns tablular form of Company code and it's name.

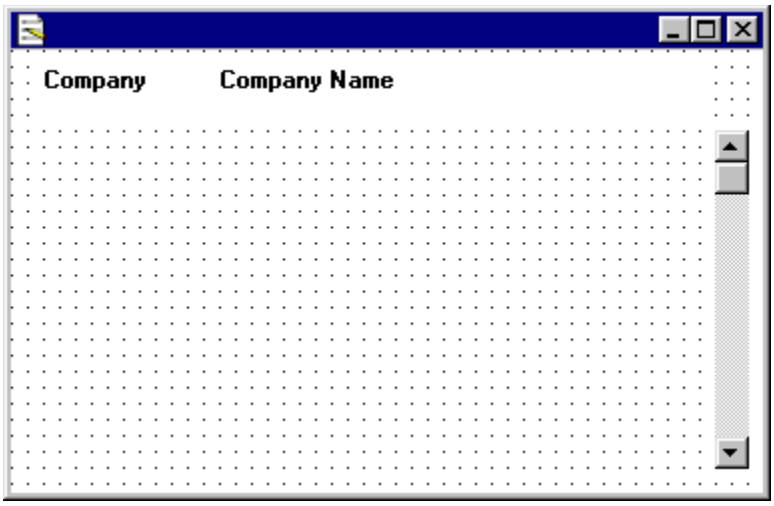
1. Create The Form



2. Add The Vertical Scroll Bar

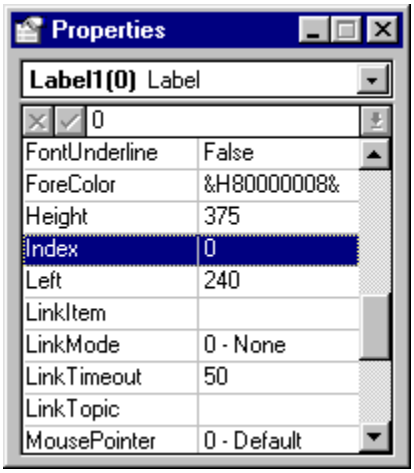


3. Add The Header Label Controls



Change their Caption
Label1.Caption = "Company"
Label2.Caption = "Company Name"

4. Change Label Indexes

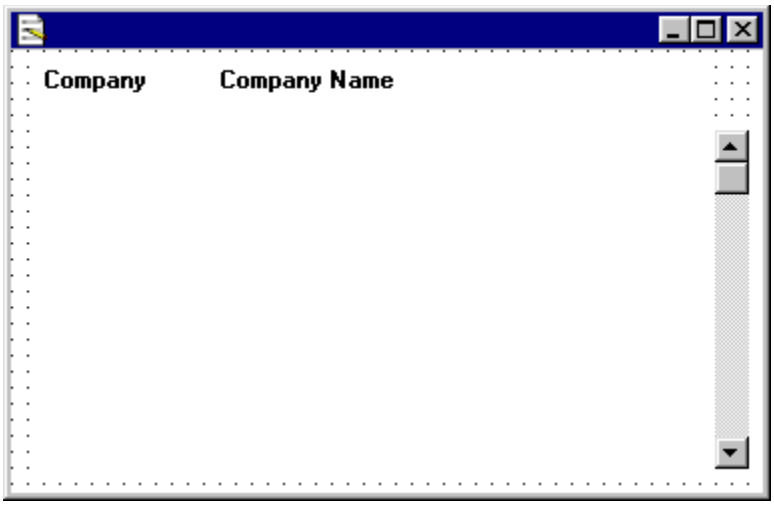


Label1.Index = 0
Label2.Index = 0

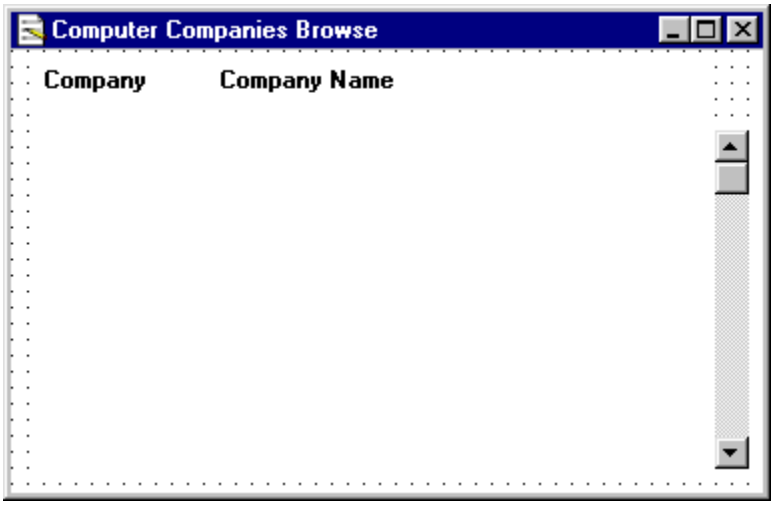
5. Copy and paste Other Label Controls from Label1 and Label2 with indexes starting from 1 To 7 (My example was for 7 rows, you can create more if you wish)

Line them up under their headers starting from Index 1 in an ascending order. (Remember to blank their Captions)

If you don't want to do this step at design time you can create it at run time with a simple procedure named AutoLoad.
Remember to call this Subroutine on the Form Load Event.



6. Change the Form Caption.



That was only what you needed to design the form. I think it was never this simple before to design a browse form.

Global Declarations
Contents

Load Event

1. Know The Record Count

```
Set DB = OpenDataBase ("CUSTOMER.MDB")
T = "Companies"
F1 = "Company_Code"
F2 = "Company_Name"
Cond = ""

If Cond = "" Then
    SQLStmt = "SELECT COUNT(*) AS Count FROM " & T
Else
    SQLStmt = "SELECT COUNT(*) AS Count FROM " & T & " WHERE " & Cond
End If

Set Sn = DB.CreateSnapshot(SQLStmt)
If Sn.RecordCount < 1 Then Exit Sub
```

2. Make the Maximum of the Scroll Bar Equal the Record Count

```
Sn.MoveFirst
VScroll1.Max = Val(Sn.Fields("Count"))
```

3. Prepare The Rows to be loaded and Filtered.

```
If Cond = "" Then
    SQLStmt = "SELECT * FROM " & T
Else
    SQLStmt = "SELECT * FROM " & T & " WHERE " & Cond
End If

Set Sn = DB.CreateSnapshot(SQLStmt & " ORDER BY " & F1)
```

4. Call the PositionFirst Procedure to load controls with data.

```
PositionFirst
```

PositionFirst Procedure
Contents

Global Declarations

This section describes the Form Global Declarations

Everything in your form can be dynamic and user defined, the SQL Statement you will be using, the filteration condition, The headers, the name of the fields. If you declare these variables to be **Global** instead of **Dim**, it will make you capable of passing parameters to the form from an external function.

```
Dim DB As DataBase      ' Database
Dim Sn As SnapShot      ' Snapshot
Dim SQLStmt As String   ' SQL Statment
Dim Cond As String      ' Condition Statement
Dim T As String         ' Table Name
Dim F1 As String        ' Field 1
Dim F2 As String        ' Field 2
Dim R1 As String        ' Return Parameter 1
Dim R2 As String        ' Return Parameter 2
Dim i                   ' Misc.
```

Load Event
Contents

PositionFirst Procedure

The purpose of this Sub is to position the Label controls with the first of the Snapshot data. Assign the Field Data to the Caption of Each Label to show it.
The purpose of the Error handler routine, is that if a Snapshot doesn't contain enough records to fill the rows, it will avoid a Visual Basic Error generation. Then it will assign the caption of the control to be blank as desired.

```
Sub PositionFirst ()

    On Error GoTo PositionError

    Sn.MoveFirst

    For i = 1 To 7
        Label1(i).Caption = Sn.Fields(F1)
        Label2(i).Caption = Sn.Fields(F2)
        Sn.MoveNext
    Next

    Exit Sub

PositionError:

    Label1(i).Caption = ""
    Label2(i).Caption = ""
    Resume Next

End Sub
```

Vertical Scroll Bar Change Event
Contents

Vertical Scroll Bar Change Event

This is the event that is incharge of the scrolling and positioning. The idea is based on always returning to the First record of the Snapshot, then Moving to the Record Order. Knowing the record order inside a snapshot is simply by retrieving the Vertical Scroll Bar Value, since you limited the Maximum to the number of records in the Snapshot.
After positioning on the record in question, you need to fill the label controls with the next 7 records (Based on our example).

An error handler code is added to prevent any error that Visual Basic will generate if the next records don't add up to 7

```
Sub VScroll1_Change ()

    If Sn.RecordCount < 1 Then Exit Sub

    On Error GoTo ErrorHandler

    Sn.MoveFirst
    For i = 1 To Val(VScroll1.Value)
        Sn.MoveNext
    Next
    Sn.MovePrevious

    For i = 1 To 7
        Label1(i).Caption = Sn.Fields(F1)
        Label2(i).Caption = Sn.Fields(F2)
        Sn.MoveNext
    Next

    Exit Sub

ErrorHandler:

    Label1(i).Caption = ""
    Label2(i).Caption = ""
    Resume Next

End Sub
```

Sorting
Contents

Return A Value On Double Click

The remaining part is the return values (R1 and R2). You can assign the return value and unload the from on a Double Click Event.

```
Sub Label1_DblClick (Index As Integer)
    If Index = 0 Then Exit Sub
    R1 = Label1(Index).Caption
    R2 = Label2(Index).Caption
    Unload Me
End Sub

Sub Label2_DblClick (Index As Integer)
    If Index = 0 Then Exit Sub
    R1 = Label1(Index).Caption
    R2 = Label2(Index).Caption
    Unload Me
End Sub
```

Contents

Read Me First

As simple as it may seem, it was difficult for me to generate the idea itself. I need to know what you think about it.

Please send an email.
To: **ayoub@usa.net**
Subject: **VB Browse Form**.

Contents

Visual Basic 4.0 and Above

The previous example was targeted for VB 3.0 If you are going to use this code into Visual Basic 4.0 and above, you should change the Snapshot declaration to a RecordSet of type Snapshot. It is important to use a Snapshot type to eliminate additional time required by a non-snapshot type RecordSet.

Please refer to you Visual Basic 4.0 and above manuals to know exactly what are the other differences from Visual Basic 3.0

Contents

Sorting

In browsing, it is important to identify sorting order, your browse form should support sorting on more than one column.
A simple click on the headers (Index=0) can resort the form by Column as follows:

```
Sub Label1_Click (Index As Integer)
    If Index > 0 Then Exit Sub
    Set Sn = DB.CreateSnapshot(SQLStmt & " ORDER BY " & F1)
    PositionFirst
End Sub

Sub Label2_Click (Index As Integer)
    If Index > 0 Then Exit Sub
    Set Sn = DB.CreateSnapshot(SQLStmt & " ORDER BY " & F2)
    PositionFirst
End Sub
```

Return A Value On Double Click
Contents

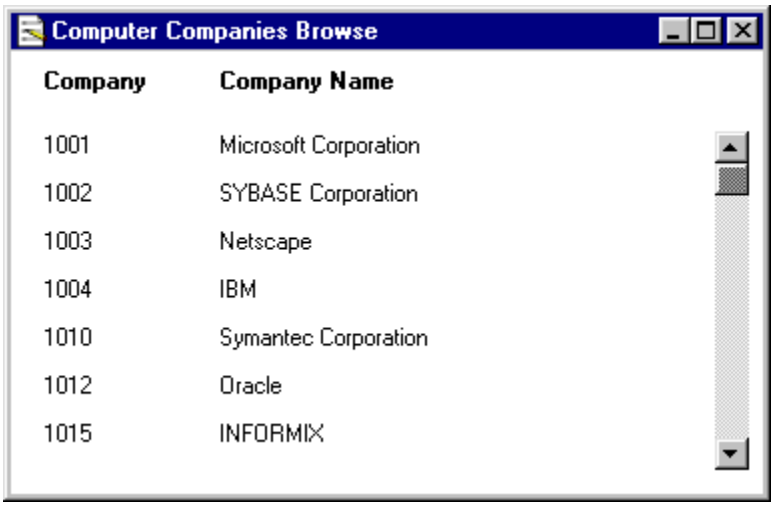
Introduction

This help file demonstrates how to create a fully capable, data aware browse form in Visual Basic, without using any VBX, OCX or DLL.
It is a well known problem that Visual Basic cannot satisfy the programmer need to create a browse form that scrolls up and down changing the *tabular* format of the form. Like Microsoft Access tabular forms, or PowerBuilder Data windows.

To overcome this problem, programmers use data aware third party controls, that are capable of forming a tabular layout and scrolling through it. An example of such controls is Microsoft Grid VBX, OCX control. Others like SpreadSheet and TrueGrid. These controls are perfect for reaching the desired goal.

But, some other problems may appear like, wanting to make each row more than one line!!! or Wanting to add a normal ComboBox in each row.
Draw backs are well known. Rows and Columns replace ordinary Control Names, making it more difficult to handle the data by row and column number.

Now, how would you like to create a form looking like this?



You can scroll up and down as you like, using an *ordinary* scroll bar, and display data using *ordinary* Label control. You can also include any type of control, a TextBox, a CheckBox or a ComboBox.

Don't be alerted, the coding is simple, minimal. Nowhere compared to the coding it you need to fill a Grid or a SpreadSheet control.

Contents

```
Sub AutoLoad ()

    For i = 1 To 7
        Load Label1(i)
        Label1(i).Top = Label1(i - 1).Top + 400
        Label1(i).Caption = ""
        Label1(i).FontBold = False
        Label1(i).Visible = True

        Load Label2(i)
        Label2(i).Top = Label2(i - 1).Top + 400
        Label2(i).Caption = ""
        Label2(i).FontBold = False
        Label2(i).Visible = True
    Next

End Sub
```