**RowSourceType Property (User-Defined Function) - Code Argument Values [Access 2003 VBA Language Reference]**

2 out of 3 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/office/aa196557%28v%3Doffice.11%29.aspx#feedback)

The Visual Basic function you create must accept five arguments. The first argument must be declared as a control and the remaining arguments as **Variants**. The function itself must return a **Variant**.

**Function** *functionname* (***fld* As Control**, ***id* As Variant**, ***row* As Variant**, ***col* As Variant**, ***code* As Variant**) As Variant

The **Function** procedure has the following five required arguments.

|  |  |
| --- | --- |
| **Argument** | **Description** |
| *fld* | A control variable that refers to the list box or combo box being filled. |
| *id* | A unique value that identifies the control being filled. This is useful when you want to use the same user-defined function for more than one list box or combo box and must distinguish between them. (The example sets this variable to the value of the **Timer** function.) |
| *row* | The row being filled (zero-based). |
| *col* | The column being filled (zero-based). |
| *code* | An intrinsic constant that specifies the kind of information being requested. |

**Note**  Because Microsoft Access calls a user-defined function several times to insert items into a list, often you must preserve information from call to call. The best way to do this is to use Static variables.

Microsoft Access calls the user-defined function by repeatedly using different values in the *code* argument to specify the information it needs. The *code* argument can use the following intrinsic constants.

|  |  |  |
| --- | --- | --- |
| **Constant** | **Meaning** | **Function returns** |
| **acLBInitialize** | Initialize | Nonzero if the function can fill the list; **False** (0) or **Null** otherwise. |
| **acLBOpen** | Open | Nonzero ID value if the function can fill the list; **False** or **Null** otherwise. |
| **acLBGetRowCount** | Number of rows | Number of rows in the list (can be zero); –1 if unknown. |
| **acLBGetColumnCount** | Number of columns | Number of columns in the list (can't be zero); must match the property sheet value. |
| **acLBGetColumnWidth** | Column width | Width (in twips) of the column specified by the *col* argument; –1 to use the default width. |
| **acLBGetValue** | List entry | List entry to be displayed in the row and column specified by the *row* and *col* arguments. |
| **acLBGetFormat** | Format string | Format string to be used to format the list entry displayed in the row and column specified by the *row* and *col* arguments; –1 to use the default format. |
| **acLBEnd** | End (the last call to a user-defined function always uses this value) | Nothing. |
| **acLBClose** | (Not used) | Not used. |

Microsoft Access calls your user-defined function once for **acLBInitialize**, **acLBOpen**, **acLBGetRowCount**, and **acLBGetColumnCount**. It initializes the user-defined function, opens the query, and determines the number of rows and columns.

Microsoft Access calls your user-defined function twice for **acLBGetColumnWidth** — once to determine the total width of the list box or combo box and a second time to set the column width.

The number of times your user-defined function is called for **acLBGetValue** and **acLBGetFormat** to get list entries and to format strings varies depending on the number of entries, the user's scrolling, and other factors.

Microsoft Access calls the user-defined function for **acLBEnd** when the form is closed or each time the list box or combo box is queried.

Whenever a particular value (such as the number of columns) is required, returning **Null** or any invalid value causes Microsoft Access to stop calling the user-defined function with that code.

Tip

**RowSourceType**

**Example**

The following user-defined function returns a list of the next four Mondays following today's date. To call this function from a list box control, enter **ListMondays** as the **RowSourceType** property setting and leave the **RowSource** property setting blank.

Function ListMondays(fld As Control,id As Variant, \_

 row As Variant,col As Variant,code As Variant) \_

 As Variant

 Dim intOffset As Integer

 Select Case code

 Case acLBInitialize ' Initialize.

 ListMondays = True

 Case acLBOpen ' Open.

 ListMondays = Timer ' Unique ID.

 Case acLBGetRowCount ' Get rows.

 ListMondays = 4

 Case acLBGetColumnCount ' Get columns.

 ListMondays = 1

 Case acLBGetColumnWidth ' Get column width.

 ListMondays = -1 ' Use default width.

 Case acLBGetValue ' Get the data.

 intOffset = Abs((9 - Weekday(Now))Mod 7)

 ListMondays = Format(Now() + \_

 intOffset + 7 \* row,"mmmm d")

 End Select

End Function

The next example uses a static array to store the names of the databases in the current directory. To call this function, enter **ListMDBs** as the **RowSourceType** property setting and leave the **RowSource** property setting blank.

Function ListMDBs(fld As Control, id As Variant, \_

 row As Variant, col As Variant, \_

 code As Variant) As Variant

 Static dbs(127) As String, Entries As Integer

 Dim ReturnVal As Variant

 ReturnVal = Null

 Select Case code

 Case acLBInitialize ' Initialize.

 Entries = 0

 dbs(Entries ) = Dir("\*.MDB")

 Do Until dbs(Entries) = "" Or Entries >= 127

 Entries = Entries+1

 dbs(Entries) = Dir

 Loop

 ReturnVal = Entries

 Case acLBOpen ' Open.

 ' Generate unique ID for control.

 ReturnVal = Timer

 Case acLBGetRowCount ' Get number of rows.

 ReturnVal = Entries

 Case acLBGetColumnCount ' Get number of columns.

 ReturnVal = 1

 Case acLBGetColumnWidth ' Column width.

 ' -1 forces use of default width.

 ReturnVal = -1

 Case acLBGetValue ' Get data.

 ReturnVal = dbs(row)

 Case acLBEnd ' End.

 Erase dbs

 End Select

 ListMDBs = ReturnVal

End Function