Home x Database Functions x Definitions x Error handling x Events handling x Methods x
Find x GetChunk x Objects x Module x

Module

A *Module* object represents a module containing a Basic script. It might represent either a standard module or a class module.

tags:
Objects

Function returning a module object

Parent object	Function	Type	Argument	Description
Application	**AllModules**	**Collection**	Integer or String	`Application.AllModules("myModule")` returns an object corresponding with the `myModule` module stored in the current document

Properties

Property	Type	Read only	Description or UNO object
CountOfDeclarationLines		Y	Indicates the number of lines of code in the Declarations section of the module.
CountOfLines		Y	Indicates the number of lines of code in the module.
Lines		Y	Returns a string containing the contents of a specified line or lines in a module.
Name		Y	Specifies the real name of the module
ObjectType		Y	Returns "MODULE"
ProcBodyLine		Y	Returns the number of the line at which the body of a specified procedure begins in the module.
ProcCountLines		Y	Returns the number of lines in a specified procedure in the module.
ProcOfLine		Y	Returns the name of the procedure that contains a specified line in the module.
ProcStartLine		Y	Returns a value identifying the line at which a specified procedure begins in the module.
Type		Y	Indicates whether a module is a standard module or a class module.

Methods

Method	Argument(s)	Return	Description
Find	string to find	Boolean	Return True if the string was found. Other arguments contain its position (line and column).

What does return the *Name* property of a module ?

To manage potential homonyms among libraries, the **name** of a module consists in 3 components:

```
        SCOPE.LIBRARY.MODULE
```

- The *SCOPE* is either
  - `GLOBAL` grouping both the **LibreOffice/OpenOffice Macros and Dialogs** and the **My Macros and Dialogs** catalogs of libraries.
  - `DOCUMENT` grouping the libraries stored in the current document.
- The *LIBRARY* component is the name of the library. It is often equal to "**Standard**".

See also

AllModules

Examples

Query the properties of a Basic module

```
Const cstModule = "myModule"
Const cstProc = "mySub"
Const vbext_pk_Proc = 0                 '       A Sub or Function procedure
Const cstStringToFind = "some string"

Dim oModule As Object, sProc As String, iProcType As Integer
Dim vStartLine As Variant, vStartColumn As Variant, vEndLine As Variant, vEndColumn As Vari

        Set oModule = Application.AllModules(cstModule)
        With oModule
                DebugPrint "Name = " & .Name
                DebugPrint "# of lines = " & .CountOfLines
                DebugPrint "# of declaration lines = " & .CountOfdeclarationLines
                DebugPrint "Lines 26 to 31 = " & .Lines(26, 6)
                DebugPrint "# of lines in proc " & cstProc & " = " & .ProcCountLines(cstPro
                DebugPrint "Start line in proc " & cstProc & " = " & .ProcStartLine(cstProc
                DebugPrint "Start body line in proc " & cstProc & " = " & .ProcBodyLine(cst
                '       Line 35  is located within procedure sProc (of type iProcType)
                sProc = .ProcOfLine(35, iProcType)
                '       Arguments are left uninitialized to consider the whole module
                If .Find(cstStringToFind, vStartLine, vStartColumn, vEndLine, vEndColumn) T
        End With
        TraceConsole()
```

Bookmark this page » » Module