AllModules

The *AllModules* collection describes instances of all **modules** present in the currently loaded Basic libraries.

tags: Collections

Applies to ...

Object	Description
Application	Root class. When present, its name must be "`Application`" but the object name is optional.

Syntax

```
[Application.]AllModules()
[Application.]AllModules(index)
[Application.]AllModules(modulename)
```

Argument #1	Type	Returned value
	absent	A **Collection** object
index	integer long	A **Module** object corresponding to the index-th item in the AllModules() collection. The 1st module is AllModules(0), the 2nd is AllModules(1) and so on ... The last one is AllModules.Count - 1.
modulename	string	A **Module** object having the argument as name. The argument is NOT case-sensitive.

Remarks

- *Access2Base* will scan first the modules present in the current Base document (".odb" file) or current non-Base document containing one or more **standalone forms** (".odt", ".ods", ... file) and continue the search thru all currently loaded libraries. The *Access2Base* library itself however will be skipped.
- The *modulename* argument is not case sensitive.

How to name modules ?

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

```
[[SCOPE.][LIBRARY.]MODULE
```

, the first two being optional.

- 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. This is the default.
- The *LIBRARY* component is the name of the library. The default is "**Standard**".

As such,

```
AllModules("DOCUMENT.STANDARD.myModule")
```

is equivalent to:

```
AllModules("myModule")
```

Error messages

Argument nr. 1 [Value = '...'] is invalid
Out of array range or incorrect array size for collection AllModules()
Module '...' not found in the currently loaded libraries

See also ...

Collection
Module

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()
```