ScriptControl

7. september 2016 22:22

https://msdn.microsoft.com/en-us/library/t0aew7h6(v=vs.84).aspx - VBScript
https://msdn.microsoft.com/en-us/library/hbxc2t98(v=vs.84).aspx - JScript (ECMAScript3)

Using the ScriptControl

Visual Studio 6.0

Using the ScriptControl is merely a matter of adding the control to your program, defining the objects the script programs can access, and then running the scripts as needed.
Before we go any further, I should warn you that using the ScriptControl is not for everyone. The ScriptControl is one of the least-documented controls available in Visual Basic. Most of the documentation for MSScript and VBScript was developed for people building web applications. You can incorporate the ScriptControl into your own programs, but expect to spend some time getting the feel of this control and its quirks. Also, be sure to save your programs (both Visual Basic and VBScript) often.

Adding a ScriptControl

Visual Studio 6.0

If you have tried the ScriptControl already, you may have had trouble finding information about it because Microsoft's documentation is really hidden. To find the documentation for this control, type **WinHelp MSScript** from a DOS session, or choose Start Run from the Taskbar.
Table 13.1 lists the key properties associated with the ScriptControl.

Table 13.1: Selected ScriptControl Properties

Properties	Description
AllowUI	When set to True, means that the script program can display user interface (UI) elements such as a MsgBox
CodeObject	Returns the set of objects that were created with the AddObject method using the script name for the object
Error	Returns an Error object containing information about the script error
Language	Contains either VBScript or JScript
Modules	Contains a collection of Module objects
Procedures	Contains a collection of Procedure objects
SitehWnd	Contains a reference to a hWnd that will be used to display GUIs
State	Describes how events of objects added with the AddObject method will be handled
Timeout	Specifies the maximum number of milliseconds the script will run before an error will be generated
UseSafeSubset	Prevents access to selected objects and procedures that can compromise an application's security

The **Language** property determines whether you use VBScript or JavaScript. The default value is VBScript, and if you are a Visual Basic advocate like me, no other option really exists. The **AllowUI** property determines if your script program can display visual elements like **InputBox** and **MsgBox**. The **Modules** and **Procedures** properties return object references to the **Modules** and **Procedures** collections. The **Modules** collection contains the name and object reference for each module available in the ScriptControl. There is always at least one module in every ScriptControl called the **Global** module. Within each module is a collection of **Procedures** and a collection of object references (**CodeObject**) available to the procedures in that module. The **Procedures** collection

contains the name, number of arguments, and whether the procedure returns a value or not for each procedure in the module.
The **CodeObject** property contains all of the routines defined with the **AddCode** method. The objects are referenced using the name of the subroutine or function; however, you must know the name of the routine at design time.
TIP: Don't hard-code: Although you can hard-code references to your script programs using the **CodeObject** property, you probably shouldn't bother. One of the reasons for using the ScriptControl is to allow the user to change the application without recompiling the application. Given the dynamic nature of script programs, you will be better served using the **Run** and **Execute** methods.
The **Timeout** property offers a safety shield to prevent a script program from going into an infinite loop and locking up the system. You can specify the maximum amount of time that a script can run before a warning message is displayed (if the **AllowUI** property is **True**). If the user chooses the End option, the **Timeout** event will occur. If you specify a value of **–1**, no timeouts will occur. A value of **0** means that the ScriptControl will monitor the execution of the script and will trigger the **Timeout** event if it determines that the script is hung.

Using ScriptControl Methods

Visual Studio 6.0
The ScriptControl contains methods to execute code, add code and objects to the scripting engine, and reset the scripting engine to its initial state. Table 13.2 lists the ScriptControl methods. These methods apply to either the global module or any of the local modules that may be defined.
Table 13.2: ScriptControl Methods

Methods	Description
AddCode	Adds a subroutine to the ScriptControl
AddObject	Makes an object available for the script programs
Eval	Evaluates an expression
ExecuteStatement	Executes a single statement
Reset	Reinitializes the scripting engine
Run	Executes a subroutine

There are four different ways to execute a program using the ScriptControl. The simplest way is with the **Eval** method. This method returns the value of the specified expression. For instance **x = ScriptControl1.Eval "1+2"** will assign a value of **3** to the variable **x**. The Eval method can also reference functions and variables that are defined in either the global module or the local module, if the method was invoked from a local module. It also can access any resource declared as public in any module.
NOTE: Wait for me to finish: When you run a script using the ScriptControl, you can't change most of the properties or use any of the methods until the script has finished. Trying to do so will result in the error "Can't execute; script is running."
You can also execute a single statement by using the **ExecuteStatement** method, as in:
ScriptControl1.ExecuteStatement "MsgBox ""VBScript is fun"""""

This method works just like the **Eval** method and can access resources in the module it was declared, in public variables declared in any module, and in the global module.
Another way to execute script code is to use the **Run** method. This method allows you to execute any subroutine declared in the ScriptControl. The subroutine may call any other subroutine or access any objects according to the rules that are used to create modules. You also can specify an array containing the parameters to be passed to the subroutine.
The **AddCode** method adds a block of code of code to the ScriptControl. During this process, the syntax of the code is checked, and the first error found will trigger the **Error** event.
WARNING: One-way street: Be sure to keep a separate copy of the code to which you added the ScriptControl. There is no way to retrieve code from the control once it has been added.

Using ScriptControl Events

Visual Studio 6.0

Table 13.3 lists the only two events that are available with the ScriptControl. The **Timeout** event occurs after the user chooses End from the dialog box, after the script program has timed out. The **Error** event occurs whenever an error is encountered in the script program. You should use the **Error** object, described next, to determine the cause of the error and take the appropriate action.

Table 13.3: ScriptControl Events

Event	Description
Error	Occurs when the scripting engine encounters an error condition
Timeout	Occurs in timeout conditions when the user selected End from the dialog box

WARNING: No runs, no hits, no Error event: If you don't have an **Error** event in your application, any errors found by MSScript will trigger a runtime error in your application. Even a little syntax error while trying to add a script to a module can cause your application to end with a runtime error.

From <https://msdn.microsoft.com/en-us/library/aa227635(v=vs.60).aspx>

Getting Error Information

Visual Studio 6.0

The **Error** object contains information about error conditions that arise while using the ScriptControl. Table 13.4 lists the properties and methods for the **Error** object.

Table 13.4: Error Object Properties and Methods

Property/Method	Description
Clear method	Clears the script error
Column property	Contains the source code column number where the error occurred
Description property	Describes the error
HelpContext property	Contains a help context reference describing the error
HelpFile property	Contains a help filename containing the help file context
Line property	Contains the source code line number where the error occurred
Number property	Contains the error number
Source property	Describes the general type of error
Text property	Contains the line of source code where the error occurred

The **Source** property describes the error as a runtime or compile-time error and the language as VBScript or JScript. The **Text** property contains the line of source code where the error condition was found. The **Line** and **Column** properties contain the exact location of the error in the script. The actual error number is available in the **Number** property, and the standard description of the error is in the **Description** property.

If you want to provide your users with a more detailed explanation, you can use the CommonDialog control with the **HelpContext** and **HelpFile** properties to display the Visual Basic help page for that error. (Note that you will need to install the associated help file on your system for this to work.)

The **Clear** method is used to reset the **Error** object. Using the **AddCode**, **Eval**, **ExecuteStatement**, or **Reset** methods will also clear the **Error** object before these methods begin processing. For more information about error handling, see Chapter 20.

From <https://msdn.microsoft.com/en-us/library/aa227430(v=vs.60).aspx>