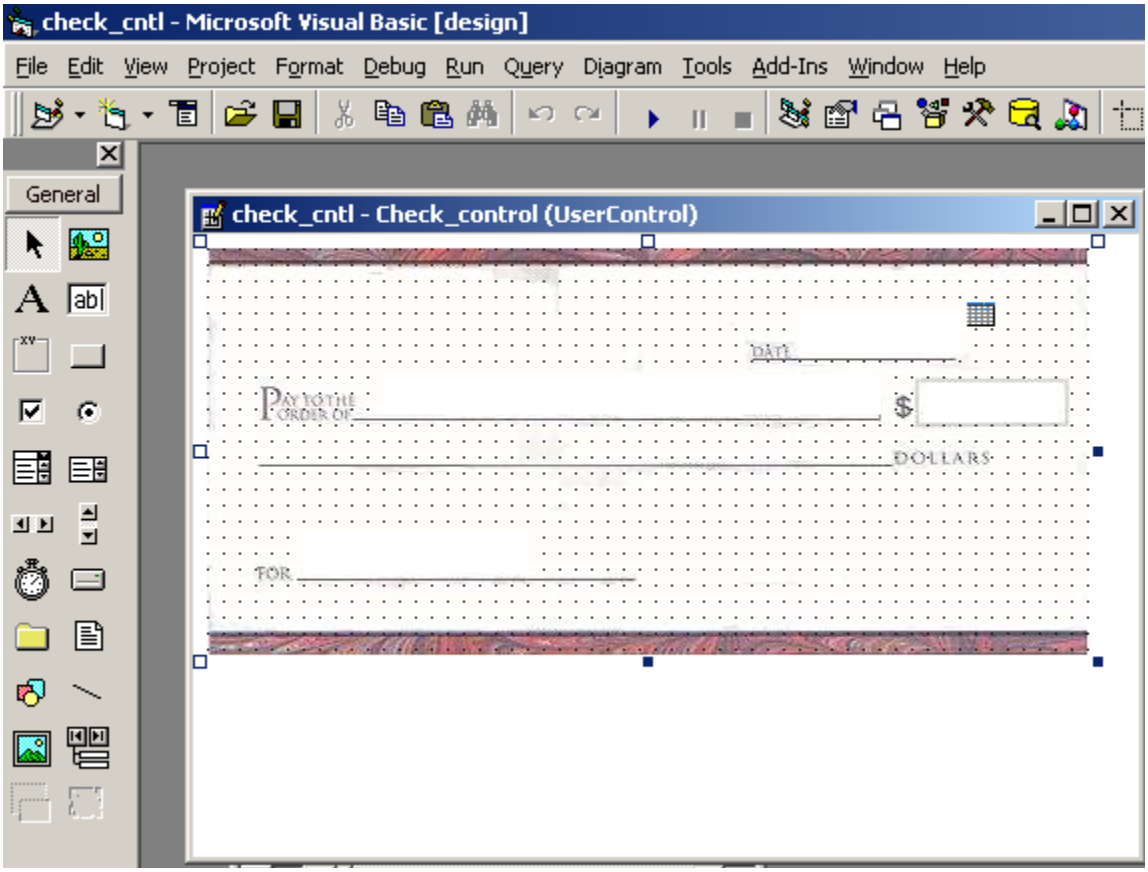
When a web page is loaded in the browser, the advantage is the commonality of anything that will be displayed. Every browser knows what a button looks like, and how to receive and send data to that button. The .aspx pages may introduce additional coding such as XML, scripting, etc, but on the simplest level this is the common denominator in the browser world.
If custom interfaces are needed – a unique button, in both behavior and appearance, some coding may be needed. This is where Windows OCXs come into play. DLLs are in the same class in the Windows world.

DLL = dynamic link library – these have no visual component, but contain callable functions.

OCX = Active X Object – a visual object, that has coding on what parameters are received, what to display, and what to return to the parent. Heavy coding for database calls, etc, should be placed outside of the component due to the overhead cost. (stands for OLE Control eXtension)

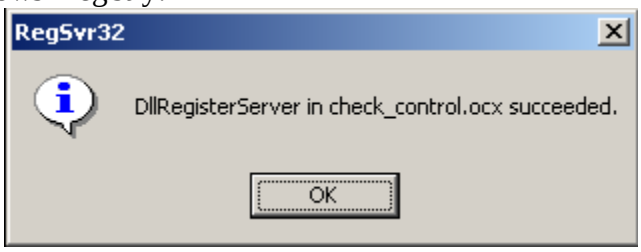
To show what an OCX is, I'll use this example from my personal library at home, from when I was learning these things. The check_cntl control is an object representing a check for entering or displaying information.



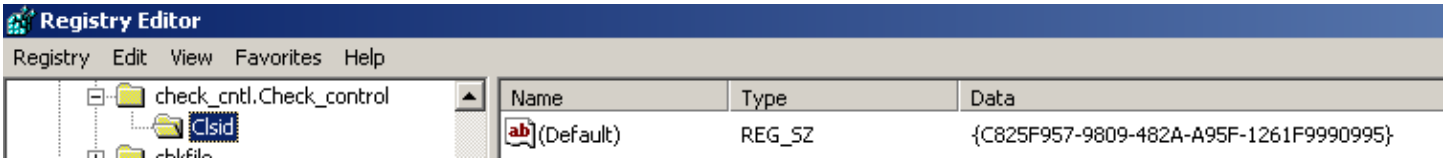
The Visual Basic compiler creates an .ocx file as output from this process, rather than a windows executable program (.exe). The ocx file has no visibility in itself – if you double-click it in windows explorer, nothing will occur, other than a question of what program handler should be used to run it.
This is part of the whole object-oriented world, where we need to have a parent Window that will host this object, and allow it to be displayed. That parent window can be:

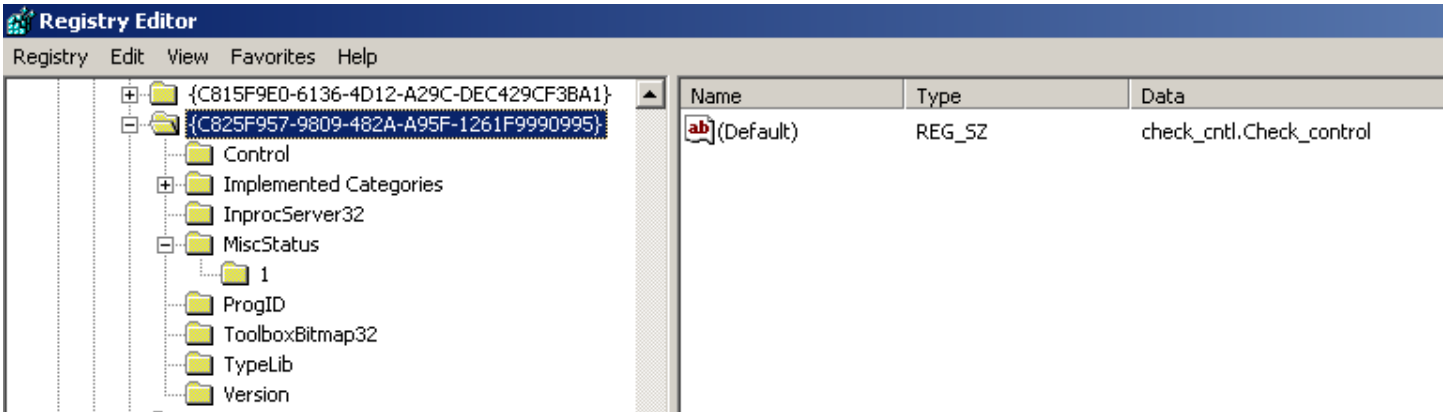
1) A Windows program that is a .exe that can be run on the local machine. Within visual basic, the user would select the check_cntl object from the toolbar on the left with all of the other common controls-buttons, pulldowns, etc. An instance of this object will then be created in the new Windows .exe program that has been built.
2) Alternatively, the Internet Explorer browser can serve as the parent Window for displaying objects that only the local machine knows about, such as this check_cntl.ocx object. The requirement for this in the web page source is that the coding be encapsulated within the <OBJECT section.

For scenarios **1** and **2**, an additional requirement is that the object be "registered" locally, primarily so Windows knows how to communicate with the object. This is accomplished via the regsvr32 command during product installation.
The simplest format of the regsvr32 command will display a box like that shown below when the entries have been added to the Windows Registry:

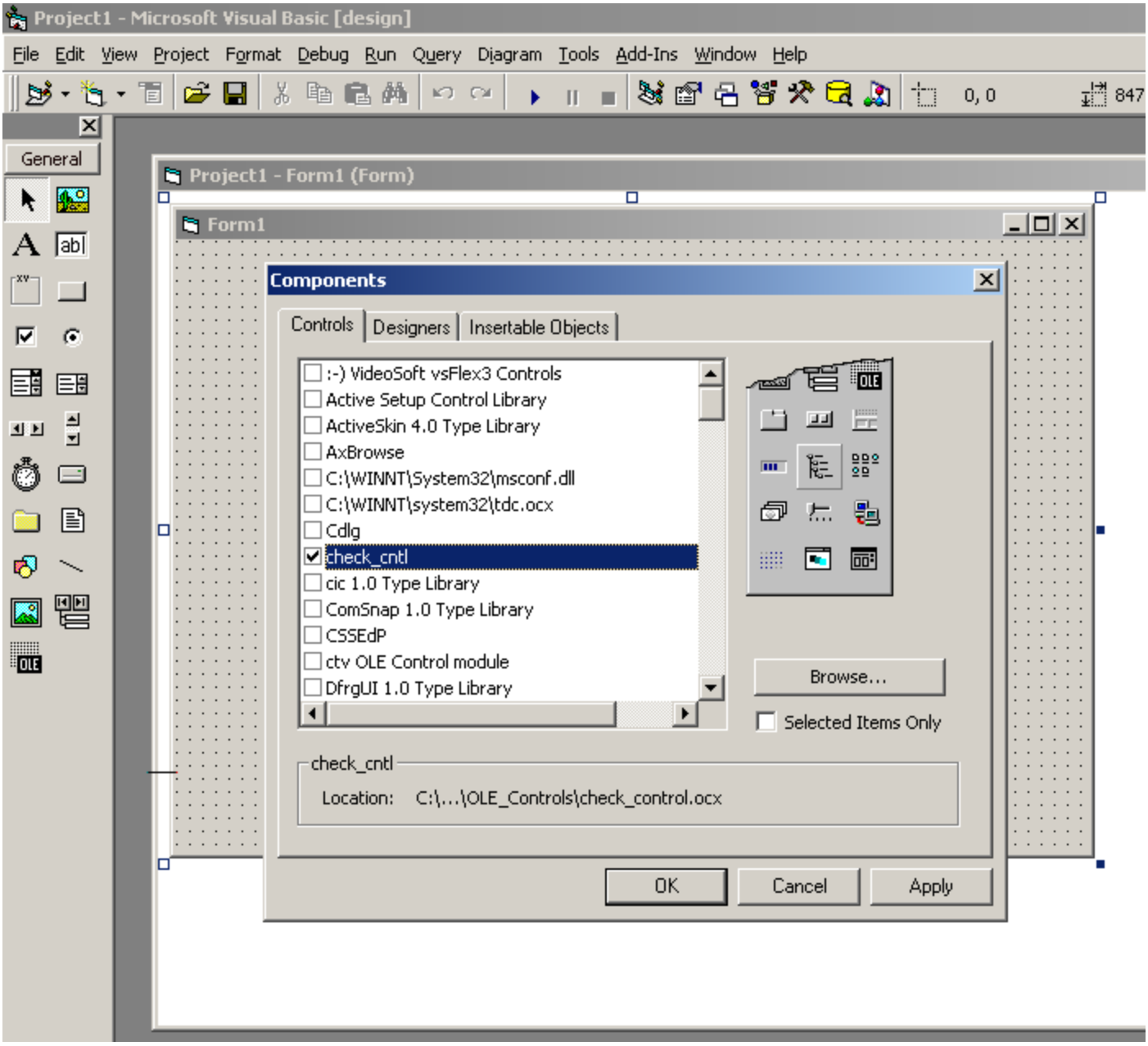


A silent version of the regsvr32 command is often used in .inf files to register components in the background during product installation. The way entries are added, allows for reference of the object either by its name (check_cntl.check_control), or by the the GUID (globally unique identifier) of 36 characters.

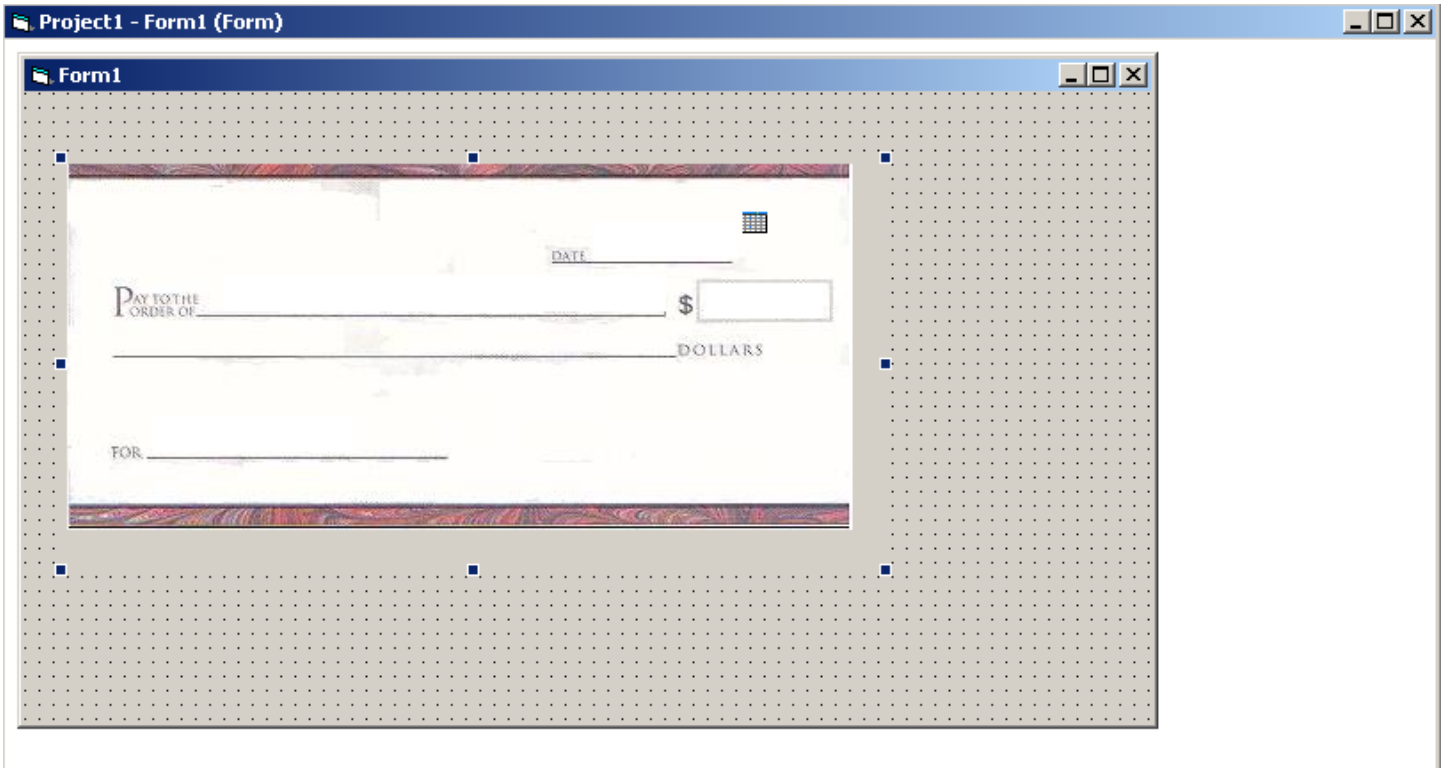




To add a .ocx to a Window (method **1** of allowing a control to be displayed), I started a new program in Visual Basic, and added a reference to the check_cntl to be listed as one of the controls that are on the toolbar – with buttons, pulldowns, etc:



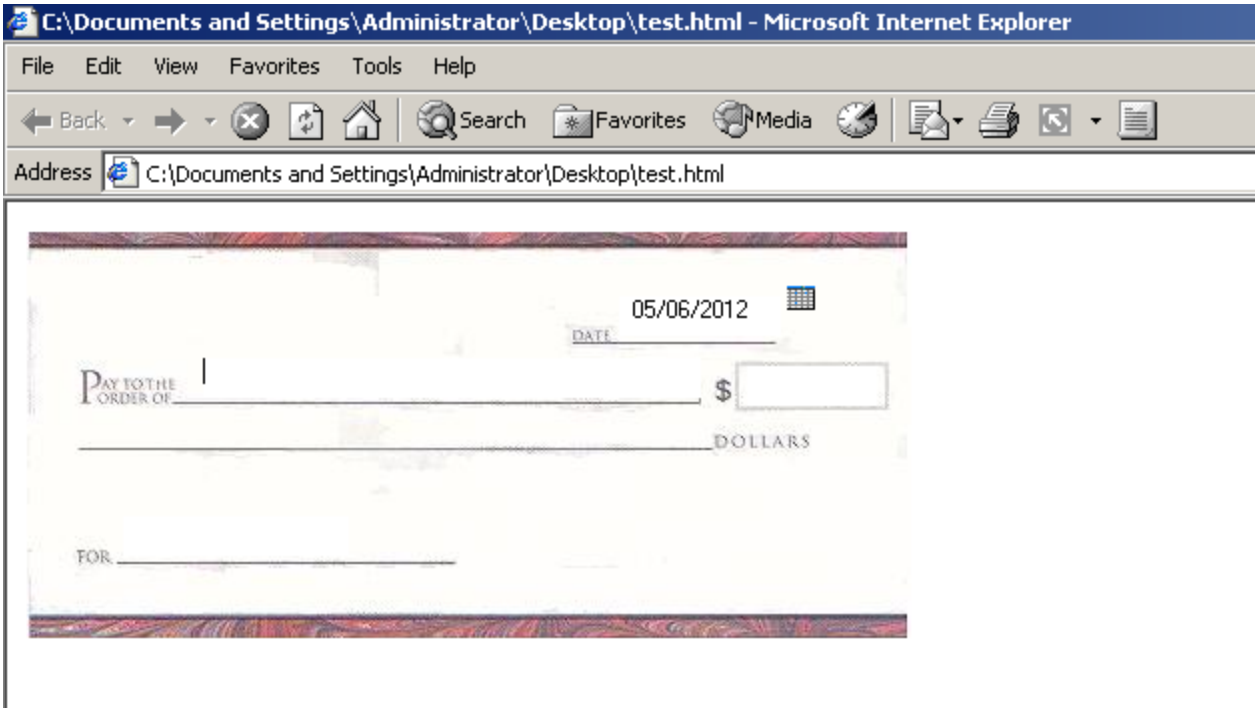
Next, I can drag the control to the main form, and it will display the check_cntl.ocx in the main window. When the .exe is created and subsequently run, it assumes that check_cntl.ocx has been registered on the local machine.



The program form1.exe is run, and the object is displayed.

For method **2**, displaying the object with Internet Explorer as the parent Window:
-We need just a simple html page with the Object tag, to specify the registration information, and where to load a copy of the object if it can't be found at a location that has been recorded in the registry. No parameters are being passed to the object, although it can receive data in any of the fields where data is entered.

```
<html>
<OBJECT ID="checkcntl1" CLASSID="CLSID:C825F957-9809-482A-A95F-1261F9990995"
CODEBASE="C:\Documents and Settings\Administrator\My
Documents\OLE_Controls\check_cntl.ocx">
</OBJECT>
</html>
```

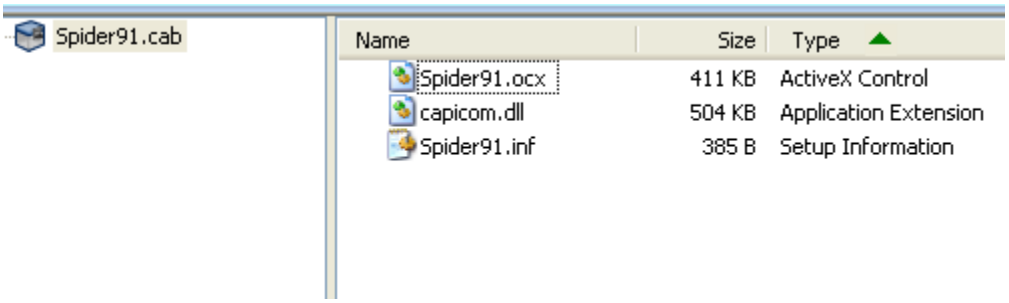


Viewing the Page Source of the Quality Center Webpage that launches:

- The first required object, is CAPICOM. This is a Microsoft ActiveX control created by Microsoft to expose a select set of cryptographic application programming interface functions. It refers to a DLL. Although it has a fancy name, I think it is only used to retrieve user name, IP address, and other local machine information to be kept in Quality Center's history database, for tracking ticket updates.

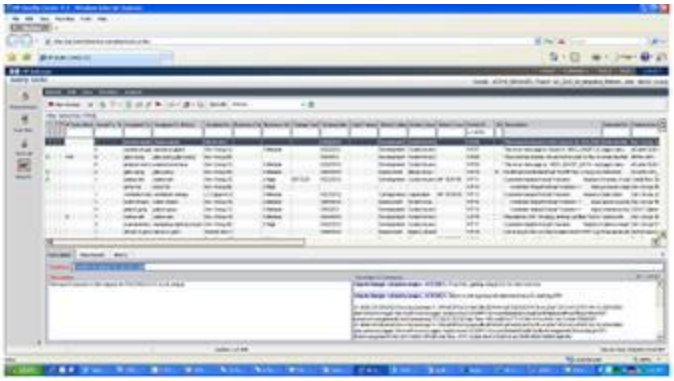
- The MSXML 3 DLLs in the second reference allows for XML parsing by webpages. The cabinet file contains 3 DLLs that can be downloaded from Microsoft if it isn't present on the current machine.

The 3rd Object can be downloaded from the qc.[your company name].com/bin/ location, if the specified version is not present on the local machine. This cabinet file contains:



- The last object (Spider91.ocx) is a monster - b2fc031d-8c74-46ae-8042-bcf4fc03c1ef is a guid that refers in the registry to "Loader Class v4", to be found at C:\WINNT\Downloaded Program Files\Spider91.ocx. If the Spider91.ocx is not yet installed, the .inf file will register it with a silent regsvr32 command.

  From a user perspective, the 2 screens below are different – but they are both handled by the ActiveX object spider91.ocx changing what fields are presented – perhaps through additional children ocx objects, in addition to DLLs providing the database-retrieved information:



The Spider91.ocx is the loading program, but there is a Test Director collection of objects that are brought in for the main defect page. All of the fields that are presented, are nothing more than controls such as the one shown earlier in this document. This, along with numerous database connections being kept open, accounts for the slowness of the product.

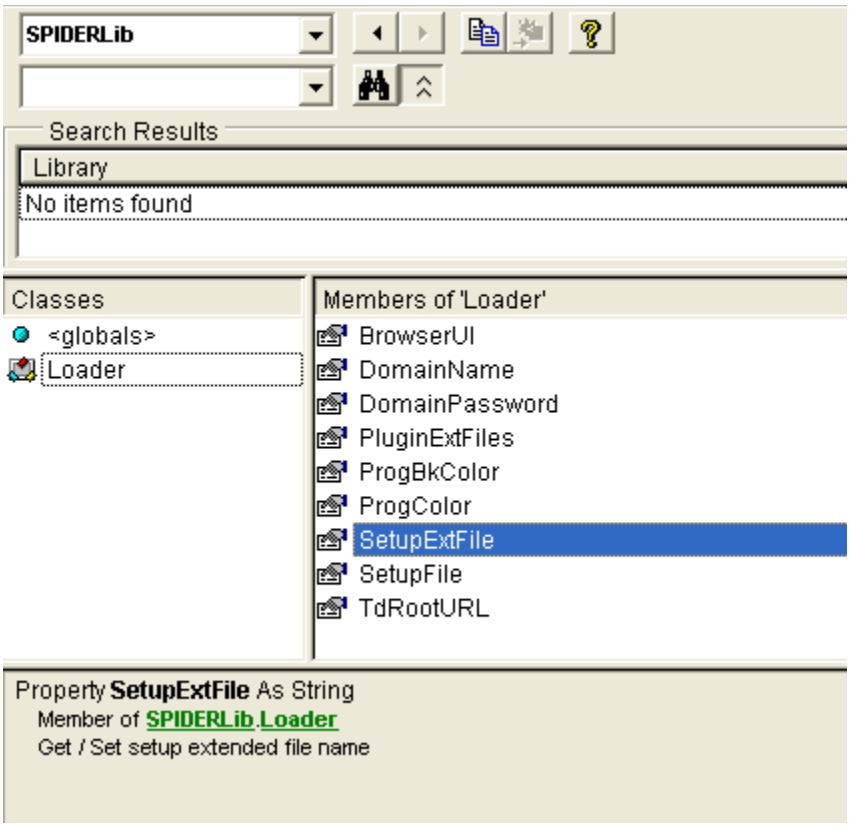
The Internet Explorer Web Page source for start_a.htm:

```
function write_ie_object()
      {
            // write capicom Object
            document.writeln(' <object declare="declare" id="CAPICOM" standby="Loading.
Please Wait..."');
            document.writeln(' classid="CLSID:3605B612-C3CF-4ab4-A426-2D853391DB2E" ');
            document.writeln(' codebase="capicom.dll#Version=2,1,0,2"');
            //document.writeln(' width="100%" height="100%"> ');
            document.writeln(' type="application/x-oleobject" ');
            document.writeln(' STYLE="display: none"> ');
            document.writeln(' </object>');

            // write MS XML Object
            document.writeln(' <object id="MSXML3" ');
            document.writeln(' CLASSID="clsid:f5078f32-c551-11d3-89b9-0000f81fe221" ');
            document.writeln(' CODEBASE="msxml3.cab#version=8.20.8730.1" ');
            document.writeln(' type="application/x-oleobject" ');
            document.writeln(' STYLE="display: none"> ');
            document.writeln(' </object> ');

            // write TestDirector Object
            document.writeln(' <OBJECT ID="MQC" ');
            document.writeln(' CLASSID="CLSID:b2fc031d-8c74-46ae-8042-bcf4fc03c1ef" ');
            document.writeln(' CODEBASE="Spider91.cab#Version=9,2,0,5141"');
            document.writeln(' WIDTH=100% ');
            document.writeln(' HEIGHT=100%> ');
            document.writeln(' <PARAM NAME="SetupFile" value="' + geturl()
+'setup_a.cab"> ');
            document.writeln(' <PARAM NAME="SetupExtFile" value="' + geturl()
+'custom_modules.cab"> ');
            document.writeln(' <PARAM NAME="PluginExtFiles" value="' + geturl()
+'servlet/tdservlet?method=ListExtensionFolder"> ');
            document.writeln(' <PARAM NAME="BrowserUI" value="0"> ');
            document.writeln(' <PARAM NAME="ProgColor" value="#663300"> ');
            document.writeln(' <PARAM NAME="ProgBkColor" value="#FFFFFF"> ');
            document.writeln(' <PARAM NAME="DomainName" value="Default"> ');
            document.writeln(' <PARAM NAME="DomainPassword" value=""> ');
            document.writeln(' <PARAM NAME="TdRootURL"  value="' + getTdRootURL() +'">
');
            document.writeln(' </OBJECT>');
```

The parameters that are in the OBJECT section for Spider91.ocx, pass data to "exposed" properties for that control. Using the VBA editor from MS Excel, I can use the object viewer to locate SPIDERLib, which will list what type of data each variable is expecting to receive.



The .ocx extension obviously doesn't match ActiveX very well. The actual definition is OLE Control eXtension. OLE, the Object Link and Embed technology, is a method for passing data to objects in Windows. It falls under the Common Object Model of using the Windows Registry to track objects, as shown earlier. The problem, is that a canyon developed between Visual Basic and C++ when object development began. DLLs written in VB, could not necessarily be called by a C++ program. It is similar to the control blocks that are shown when a COBOL II program abended, versus a Fortran or PL/I program. All of the storage areas were quite different. This led to the next generation of COM, under the .NET Platform. When Visual Basic .NET was introduced, along with C++ and C#, all of these languages began using the same umbrella environment – similar to IBM's Language Environment on the mainframe including all of the IBM languages. When Visual Studio 2002 arrived, it delivered the .NET platform that has been used for the last 10 years.