


[Home](#) [Articles](#) [Questions & Answers](#) [Learning Zones](#) [Features](#) [Help!](#) [The Lounge](#)

ActiveX

Not quite what you are looking for? You may want to try:

- [WTL for MFC Programmers, Part VI - Hosting ActiveX Controls](#)
- [A Complete Scriptable ActiveX Web Control Tutorial using ATL](#)

Article	Browse Code	Stats	Revisions					121			4.75 / 5, 90 votes
---------	-------------	-------	-----------	--	--	--	--	-----	--	--	--------------------

[Home](#) » [Platforms, Frameworks & Libraries](#) » [COM / COM+](#) » [ActiveX](#)

# A Complete ActiveX Web Control Tutorial

 By [David Marcionek](#) | 21 Jun 2006
First Posted **21 Jun 2006**Views **325,981**Bookmarked **301 times**

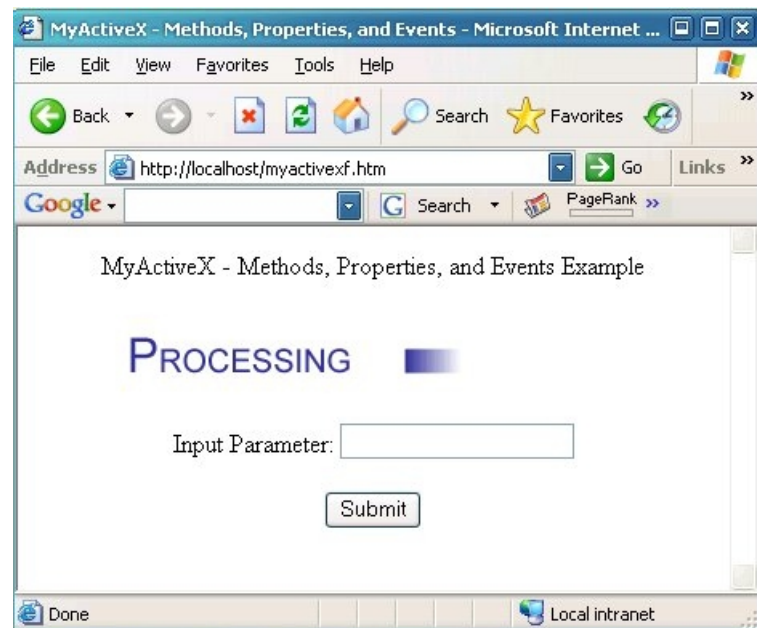
Licence

[VC8.0](#), [Win2K](#), [WinXP](#), [Visual-Studio](#), [Dev](#), [COM](#), [Beginner](#)

This article is intended to help you get up to speed quickly with developing an ActiveX control. It will show you the basic concepts you need to know about ActiveX, such as methods, properties, and events, and how to communicate between an ActiveX control and a web page.



**Is your email address OK?** You are signed up for our newsletters but your email address is either unconfirmed, or has not been reconfirmed in a long time. Please click [here](#) to have a confirmation email sent so we can confirm your email address and start sending you newsletters again. Alternatively, you can [update your subscriptions](#).


[Download demo project - 231 Kb](#)


## Introduction

**ActiveX** is a Microsoft technology developed in the mid 90s, that allows for the creation of applet-like applications that can be downloaded and run within Microsoft's Web browser. This article is intended for Visual C++ developers who are trying to learn how to develop their first **ActiveX** control for a web application but finding it difficult. While trying to learn this technology myself, I found much of the information available on **ActiveX** was either no longer available, out of date, or missing critical information, making it extremely difficult for me to create an **ActiveX** control necessary for my development project. This article is intended to help you get up to speed quickly with developing an **ActiveX** control. It will show you the basic concepts you need to know about **ActiveX**, such as methods, properties, and events, and how to communicate between an **ActiveX** control and a web page. You will learn how to implement the control for use with default security settings in Internet Explorer on Windows XP, without getting unsigned or unsafe control warning messages.

For this tutorial, we will create an **ActiveX** control that displays a progress bar animated GIF when the control is loaded as a way to indicate to users that the control is loading and processing information. The control will contain functionality to demonstrate how to pass information between the control and

the web page. You will be guided step by step on creating the control using Microsoft Visual Studio 2005.

## Creating an ActiveX Control

To create an **ActiveX** control, use Microsoft Visual Studio 2005 to perform the following steps:

1. From *File* menu, select *New*, then *Project*.
2. In the *New Project* dialog, as shown in Figure 1, under *Project types*, select *Visual C++*, *MFC*. Under *Templates*, select *MFC **ActiveX** Control*.
3. Name the project *My**ActiveX***; for *Location*, enter the working folder for the project's source code, and then click the *OK* button.

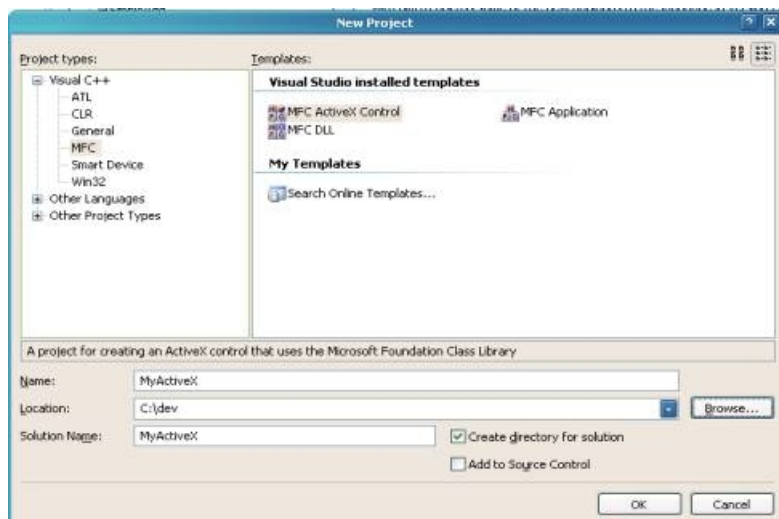


Figure 1. New Project Dialog

4. In the *MFC **ActiveX** Control Wizard* dialog, as shown in Figure 2, select *Control Settings*.
5. Under *Create control based on*, select *STATIC*. We are using a static control, as we are just displaying the output from the control and not accepting input.
6. Under *Additional features*, make sure *Activates when visible* and *Flicker-free activation* are checked and *Has an About box dialog* is **not** checked.

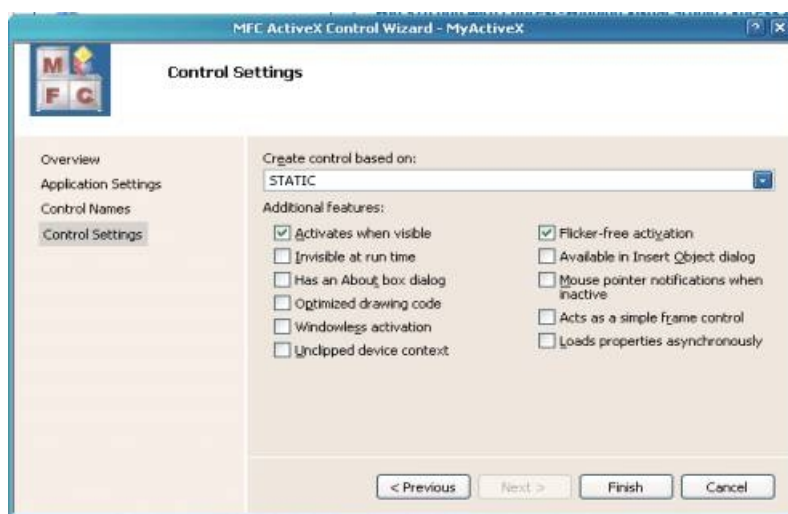


Figure 2. MFC **ActiveX** Control Wizard Dialog

7. Click the *Finish* button to enable the *MFC **ActiveX** Control Wizard* to create the project's source code. By default, the wizard creates the project to use MFC in a shared DLL. We need to change this since the **ActiveX** control will not run unless the required MFC DLLs are already installed on the system where the control is being downloaded and run. This is one of the causes of a red X displayed where an **ActiveX** control should be on a web page. From the Visual Studio menu, select *Project*, *Properties*. Navigate to *Configuration Properties*, *General*. Change the entry ☐ *Use of MFC* to ☐ *Use MFC in a Static Library*.
8. The wizard has created the following three classes:
  - *CMy**ActiveX**App* ☐ This is the **ActiveX** application class derived from *COleControlModule*. It is the base class to derive an OLE control module object that contains the member functions for initialization (*InitInstance*) and code cleanup (*ExitInstance*).
  - *CMy**ActiveX**Ctrl* ☐ This is derived from the base class *COleControl*. This is where we will implement most of the functionality for our control.
  - *CMy**ActiveX**PropPage* ☐ This is derived from the base class *COlePropertyPage*. It is

used to manage the property page dialog for the control. The **ActiveX** Control Wizard has created a default dialog to serve as a property page for the control.

## Adding Support for Animated GIF

In order to implement support for displaying a progress bar animated GIF from the **ActiveX** control, we will use the **CPictureEx** class presented by Oleg Bykov in a CodeProject article. Refer to the *References* section for more details. First, add the source files *pictureex.cpp* and *pictureex.h* to your project, by selecting the *Solution Explorer* tab in VS 2005, then right click on the *Header Files* or *Source Files* in the source tree, and then *Add, Existing Item* to select the appropriate source file.

To add an animated GIF resource to the project, we have to work around a defect in Visual Studio 2005 (and VS 2003) that does not allow importing a GIF image file. If you try it, you will get an error reporting that the file is not a valid GIF file. You can work around this defect as follows:

1. Copy the GIF file *ProcessingProgressBar.gif* to your project working folder, and rename the file to *ProcessingProgressBar.gaf* with a *.gaf* file extension. In *Resource View*, right click on the resource file *MyActiveX.rc*, and select *Add Resource*. In the *Add Resource* dialog, press the *Import* button, and select the file *ProcessingProgressBar.gaf*. In the *Custom Resource Type* dialog, enter *.GIF* for *Resource type*, and press OK. This will import the GIF image file into the project. You will find it listed under *GIF* in *Resources*. Navigate to this item, and change the control ID from the default of *IDR\_GIF1* to *IDR\_PROGRESSBAR*.
2. Now, we need to make things right with the image file. First, save the resource file *MyActiveX.rc*. Navigate to the project working folder, and edit this same resource file with Notepad. Navigate to the line item definition for *IDR\_PROGRESSBAR*, and change the filename in quotes to *ProcessingProgressBar.gif*. Also, change the GIF image filename in the working folder to *ProcessingProgressBar.gif*. From Notepad, save the resource file *MyActiveX.rc*. Visual Studio will then report that the file *myactivex.rc* has been modified outside of Visual Studio, click *Yes* to reload the file. One more correction needs to be made. Select *Solution Explorer*, navigate to the item *ProcessingProgressBar.gaf*, and select it. In *Properties*, select *Relative Path*, and correct the filename to *ProcessingProgressBar.gif*.

## Adding Dialog for Progress Bar Graphic

Now, we will add a dialog for the progress bar graphic.

1. In *Resource View*, right click on the dialog item in the tree, and select *Insert Dialog* to create a default dialog.
2. Delete the OK and Cancel buttons that are not needed, and adjust the size of the dialog to 230 x 40.
3. Change some of the default properties of the dialog to: Border *None*, Style *Child*, System Menu *False*, Visible *True*.
4. Change the control ID to *IDD\_MAINDIALOG*.
5. Insert a picture control into the dialog, by selecting the *Toolbox* tab on the right of Visual Studio, selecting a picture control, and clicking in the dialog. Adjust the size of the control to 200 x 20. Change the control ID to *IDC\_PROGRESSBAR* and the *Color* property to *White*.
6. Create a class for the dialog, by right clicking on the dialog and selecting *Add Class*. This results in the MFC Class Wizard dialog as shown in Figure 3. Name the class *CMainDialog*, with the base class *CDialog*. Click *Finish* for the wizard to create the default source files for the class.

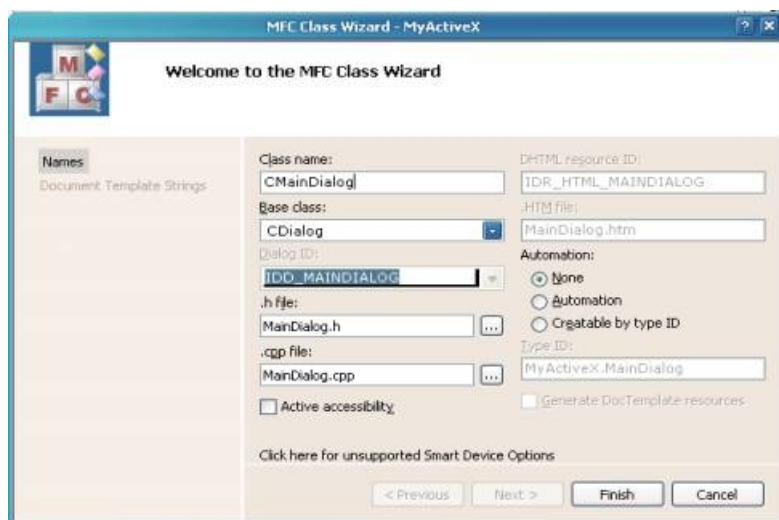


Figure 3. MFC Class Wizard *CMainDialog*

Now, we add the member variables for the classes. The member variable *m\_MainDialog* is associated with the *CMainDialog* class, and *m\_ProgressBar* is associated with the progress bar control we added to the main dialog.

1. Add the member variable *m\_MainDialog* to the class *CMyActiveXCtrl*. Select *Class View*, right click on *CMyActiveXCtrl*, and select *Add, Add Variable*. Enter *CMainDialog* for *Variable type*

and `m_MainDialog` for *Variable name*, and then press the *Finish* button.

- Similar to the above, add a member variable `m_ProgressBar` to the class `CMainDialog`. Enter `CPictureEx` for *Variable type*, `m_ProgressBar` for *Variable name*, and enable the *Control variable checkbox*, and make sure `IDC_PROGRESSBAR` is entered for *Control ID*. Before clicking on the *Finish* button, make sure that *Variable type* is set to `CPictureEx` and not changed to `CStatic`.



Figure 4. Add Member Variable Wizard `m_ProgressBar`

## Adding Support Code

Now, we get our hands dirty with adding some code to support drawing the main dialog and the progress bar control.

- Select the class `CMyActiveXCtrl`. In the *Properties* sheet, select the *Messages* icon, then `WM_CREATE`. Select the listbox to the right of `WM_CREATE`, then `<Add> OnCreate` to add a method for the `WM_CREATE` message. The wizard will add the `OnCreate` method to the `CMyActiveXCtrl` class.
- Edit `MyActiveXCtrl.cpp`, and add the following code to the `OnCreate` method to create the main dialog:

[Collapse](#) | [Copy Code](#)

```
m_MainDialog.Create( IDD_MAINDIALOG, this );
```

Add the following code to the `OnDraw` method to size the main dialog window and fill the background:

[Collapse](#) | [Copy Code](#)

```
m_MainDialog.MoveWindow(rcBounds, TRUE);
CBrush brBackGnd(TranslateColor(AmbientBackColor()));
pdc->FillRect(rcBounds, &brBackGnd);
```

- Select the class `CMainDialog`. In the *Properties* sheet, select the *Messages* icon, then `WM_CREATE`. Select the listbox to the right of `WM_CREATE`, then `<Add> OnCreate` to add a method for the `WM_CREATE` message. The wizard will add the `OnCreate` method to the `CMainDialog` class.
- Edit `MainDialog.cpp`, and add the following code to the `OnCreate` method to load and draw the progress bar animated GIF image:

[Collapse](#) | [Copy Code](#)

```
if (m_ProgressBar.Load(MAKEINTRESOURCE( IDR_PROGRESSBAR ), _T( "GIF" ) ) )
    m_ProgressBar.Draw();
```

Make sure the build configuration is set to the Release configuration, and build the `MyActiveX ActiveX` application.

## Creating a Web Page for an ActiveX Control

The tool of choice for quickly creating a default web page to test your control is Microsoft's `ActiveX Control Pad`. It is available for [download from Microsoft](#).

You will also find it available for download at various other sites on the Internet. Install it and run it on the same system you are using to develop the control with Microsoft Visual Studio. To make it easier for initial testing of the application, you should make sure that the Microsoft IIS web server is installed on this same system.

When you first run the `ActiveX Control Pad`, it will create a default HTML web page for you. To insert



an **ActiveX** control, right click within the `<BODY>` tag of the HTML source, and select *Insert **ActiveX** Control*. In the Insert **ActiveX** Control dialog, scroll down and select *My**ActiveX** Control* that you have created with Visual Studio, and click OK.



Figure 5. **ActiveX** Control Pad □ Insert **ActiveX** Control

Two dialog boxes will be displayed in the **ActiveX** Control Pad, enabling you to modify the control. The *Properties* dialog is for modifying properties of the control, the *Edit **ActiveX** Control* dialog is for manually editing the control. You can close both of these dialogs as we can make any further changes necessary by manually editing the HTML code. You should now find an `OBJECT ID` tag inserted in the HTML code, similar to that shown in Figure 6. Change the size parameters of the control by changing to `WIDTH=350` and `HEIGHT=50` in the `OBJECT ID` tag. Save the HTML file for the web page to the file *myactivex.htm* in the root folder *wwwroot* of IIS web server.

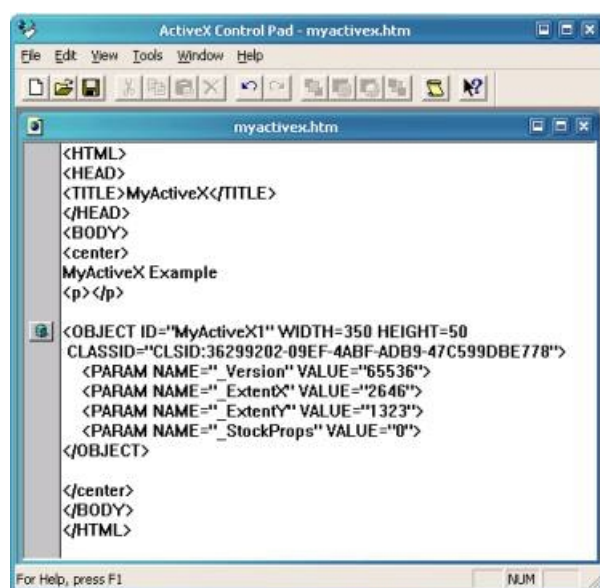


Figure 6. **ActiveX** Control Pad □ My**ActiveX** **ActiveX** Control

To test the **ActiveX** control, load the web page <http://localhost/myactivex.htm> with Internet Explorer. If you get any warning messages, just click OK to proceed. This should result in a progress bar animated GIF displayed within the web page. If not, or if you get a red X displayed where the **ActiveX** control should be, then it is most likely due to the security settings of the browser which is preventing the **ActiveX** control from loading and running. To correct this, modify the security settings in Internet Explorer to change all the settings that have to do with **ActiveX** to *enabled*.

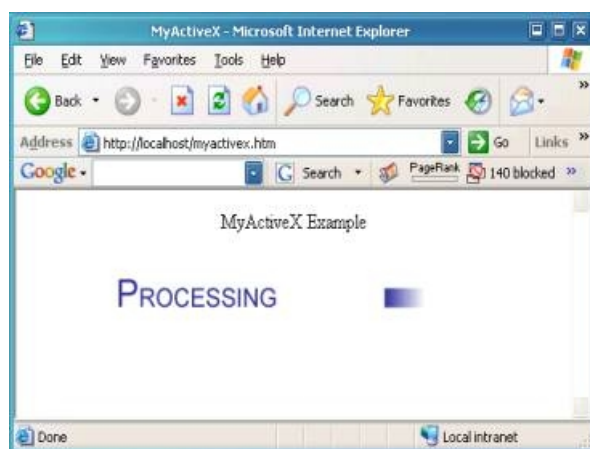


Figure 7. My**ActiveX** Control in Internet Explorer

Next, we need to build the **ActiveX** control so loading it from Internet Explorer browser does not result in annoying error messages complaining that it is an unsigned or unsafe control.

## Building a Signed ActiveX Control

To create a signed **ActiveX** control, you must purchase a Code Signing Certificate from one of the certificate providers such as Thawte, Verisign, or GeoTrust. With this service, they will verify your identity and provide you certificate files you use to sign the **ActiveX** application. I chose Thawte for a Code Signing Certificate, who provided two files for signing the application, *mycert.spc* and *mykey.pvk*.

To sign the **ActiveX** application, we package the components of the application into a CAB file, which is downloaded from the web site and the **ActiveX** control is installed on the system. Part of installing the **ActiveX** control requires registering the control. To enable that to happen, the control must be built with the `OLESelfRegister` value defined in the `VERSIONINFO` structure of the **ActiveX** control. Versions of Microsoft Visual Studio up to VS 2003 inserted this entry, but Visual Studio 2005 does not. To add the entry, edit the resource file *myactivex.rc* to add the `OLESelfRegister` value, as shown below:

[Collapse](#) | [Copy Code](#)

```
VS_VERSION_INFO VERSIONINFO
    FILEVERSION 1,0,0,1
    PRODUCTVERSION 1,0,0,1
    FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
    FILEFLAGS 0x1L
#else
    FILEFLAGS 0x0L
#endif
    FILEOS 0x4L
    FILETYPE 0x2L
    FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904e4"
        BEGIN
            VALUE "CompanyName", "TODO: <Company name>"
            VALUE "FileDescription", "TODO: <File description>"
            VALUE "FileVersion", "1.0.0.1"
            VALUE "InternalName", "MyActiveX.ocx"
            VALUE "LegalCopyright",
                "TODO: (c) <Company name>. All rights reserved."
            VALUE "OLESelfRegister", "\0"
            VALUE "OriginalFilename", "MyActiveX.ocx"
            VALUE "ProductName", "TODO: <Product name>"
            VALUE "ProductVersion", "1.0.0.1"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1252
    END
END
```

Before signing the application, the **ActiveX** control should be packaged into a CAB file. This CAB file will also contain an INF file that is used for installing your **ActiveX** control. To build a CAB file, you need the *cabarc.exe* tool available in the Microsoft Cabinet Software Development Kit. The following is an example of a simple INF file that can be used for packaging the My**ActiveX** control into a CAB file. For the CLSID line item, you should change the value to the same value as that in the `OBJECT ID` tag in the HTML web page you created earlier with the **ActiveX** Control Pad.

[Collapse](#) | [Copy Code](#)

```
[Add.Code]
myactivex.ocx=myactivex.ocx
myactivex.inf=myactivex.inf

[myactivex.ocx]
file=thiscab
clsid={36299202-09EF-4ABF-ADB9-47C599DBE778}
RegisterServer=yes
FileVersion=1,0,0,0

[myactivex.inf]
file=thiscab
```

To create a CAB file, run *cabarc* as shown below. **Important:** Make sure the OCX and INF files are in same directory where you are running *cabarc.exe*, otherwise the CAB will not be extracted correctly after downloading from the web server. This is one of the problems that will cause a red X on the web page where the **ActiveX** control should be.

[Collapse](#) | [Copy Code](#)

```
cabarc -s 6144 N myactivex.cab myactivex.ocx myactivex.inf
```

To sign the CAB file you created, you need the *signcode.exe* tool from Microsoft MSDN. Refer to the [Signing and Checking with Authenticode](#) reference at the end of this article. You use the *signcode* tool with the certificate files you obtained from purchasing a Coding Signing Certificate to sign the CAB file. The following is an example use of *signcode* to sign *myactivex.cab*:

```

signcode -n "myactivex" -i
http://www.myactivex.com -spc mycert.spc -v mykey.pvk -t
http://timestamp.verisign.com/scripts/timestamp.dll myactivex.cab

```

[Collapse](#) | [Copy Code](#)

In the above example, <http://www.myactivex.com> should be replaced with a web page that provides users further information about your signed **ActiveX** control.

To use the signed CAB file in your web page, first copy the *myactivex.cab* to a folder on your web site, then you must modify the **OBJECT ID** tag on your web page with a **CODEBASE** parameter to reference this CAB file. Refer to Figure 8 for an example. If you load this page in Internet Explorer, it should download the CAB file and install your **ActiveX** control with no warning about an unsigned **ActiveX** control.

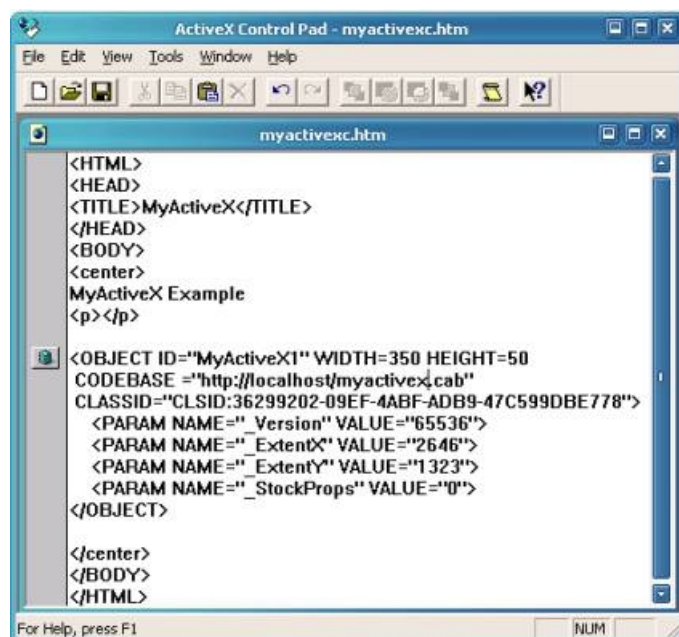


Figure 8. **ActiveX** Control Pad □ My**ActiveX** with CODEBASE

## Building a Safe ActiveX Control

To make a control that will load in Internet Explorer with no unsafe control warning or error messages, you must implement code that ensures safe initialization and safe scripting for an **ActiveX** control. Detailed information for doing that can be found in the article □ Safe Initialization and Scripting for **ActiveX** Controls □ on Microsoft MSDN. Refer to References at the end of this article for details. I found omissions and mistakes in this article that I have corrected for presentation in this article. Basically, all that needs to be done is to add code to the *DllRegisterServer* and *DllUnregisterServer* methods. The following is a step-by-step guide for making your **ActiveX** control safe:

1. Edit *MyActiveX.cpp* and add the following code. The value of *CLSID\_SafeItem* should be taken from *IMPLEMENT\_OLECREATE\_EX* in the *MyActiveXCtrl.cpp* source file or the equivalent for your **ActiveX** control. It will also be the same value as the **CLSID** in the **OBJECT ID** tag on the **HTML** page with your **ActiveX** control.

[Collapse](#) | [Copy Code](#)

```

#include "comcat.h"
#include "strsafe.h"
#include "objsafe.h"

// CLSID_SafeItem - Necessary for safe ActiveX control
// Id taken from IMPLEMENT_OLECREATE_EX function in xxxCtrl.cpp

const CATID CLSID_SafeItem =
{ 0x36299202, 0x9ef, 0x4abf, { 0xad, 0xb9, 0x47, 0xc5, 0x99, 0xdb, 0xe7, 0x78} };

// HRESULT CreateComponentCategory - Used to register ActiveX control as safe

HRESULT CreateComponentCategory(CATID catid, WCHAR *catDescription)
{
    ICatRegister *pcr = NULL ;
    HRESULT hr = S_OK ;

    hr = CoCreateInstance(CLSID_StdComponentCategoriesMgr,
        NULL, CLSCTX_INPROC_SERVER, IID_ICatRegister, (void**)&pcr);
    if (FAILED(hr))
        return hr;

    // Make sure the HKCR\Component Categories\{..catid...}
    // key is registered.
    CATEGORYINFO catinfo;
    catinfo.catid = catid;
    catinfo.lcid = 0x0409 ; // english
    size_t len;
    // Make sure the provided description is not too long.
    // Only copy the first 127 characters if it is.
    // The second parameter of StringCchLength is the maximum
    // number of characters that may be read into catDescription.
    // There must be room for a NULL-terminator. The third parameter
    // contains the number of characters excluding the NULL-terminator.
    hr = StringCchLength(catDescription, STRSAFE_MAX_CCH, &len);
    if (SUCCEEDED(hr))
    {
        if (len > 127)
        {
            len = 127;
        }
    }
    else
    {
        // TODO: Write an error handler;
    }
    // The second parameter of StringCchCopy is 128 because you need
    // room for a NULL-terminator.
    hr = StringCchCopy(catinfo.szDescription, len + 1, catDescription);
    // Make sure the description is null terminated.
    catinfo.szDescription[len + 1] = '\0';

    hr = pcr->RegisterCategories(1, &catinfo);
    pcr->Release();

    return hr;
}

// HRESULT RegisterCLSIDInCategory -
// Register your component categories information

HRESULT RegisterCLSIDInCategory(REFCLSID clsid, CATID catid)
{
    // Register your component categories information.
    ICatRegister *pcr = NULL ;
    HRESULT hr = S_OK ;
    hr = CoCreateInstance(CLSID_StdComponentCategoriesMgr,
        NULL, CLSCTX_INPROC_SERVER, IID_ICatRegister, (void**)&pcr);
    if (SUCCEEDED(hr))
    {
        // Register this category as being "implemented" by the class.
        CATID rgcatid[1] ;
        rgcatid[0] = catid;
        hr = pcr->RegisterClassImplCategories(clsid, 1, rgcatid);
    }

    if (pcr != NULL)
        pcr->Release();

    return hr;
}

// HRESULT UnRegisterCLSIDInCategory - Remove entries from the registry

HRESULT UnRegisterCLSIDInCategory(REFCLSID clsid, CATID catid)
{
    ICatRegister *pcr = NULL ;
    HRESULT hr = S_OK ;

    hr = CoCreateInstance(CLSID_StdComponentCategoriesMgr,
        NULL, CLSCTX_INPROC_SERVER, IID_ICatRegister, (void**)&pcr);
    if (SUCCEEDED(hr))
    {
        // Unregister this category as being "implemented" by the class.
        CATID rgcatid[1] ;
        rgcatid[0] = catid;
        hr = pcr->UnRegisterClassImplCategories(clsid, 1, rgcatid);
    }
}

```



2. Modify the `DllRegisterServer` method to add the highlighted code as shown:

```

if (pccr != NULL)
    pccr->Release();
}
STDAPI DllRegisterServer(void)
{
    return hr;
}
// HRESULT used by Safety Functions

AFX_MANAGE_STATE(_afxModuleAddrThis);

if (!AfxOleRegisterTypeLib(AfxGetInstanceHandle(), _tlid))
    return ResultFromScode(SELFREG_E_TYPELIB);

if (!COleObjectFactoryEx::UpdateRegistryAll(TRUE))
    return ResultFromScode(SELFREG_E_CLASS);

// Mark the control as safe for initializing.

hr = CreateComponentCategory(CATID_SafeForInitializing,
    L"Controls safely initializable from persistent data!");
if (FAILED(hr))
    return hr;

hr = RegisterCLSIDInCategory(CLSID_SafeItem,
    CATID_SafeForInitializing);
if (FAILED(hr))
    return hr;

// Mark the control as safe for scripting.

hr = CreateComponentCategory(CATID_SafeForScripting,
    L"Controls safely scriptable!");
if (FAILED(hr))
    return hr;

hr = RegisterCLSIDInCategory(CLSID_SafeItem,
    CATID_SafeForScripting);
if (FAILED(hr))
    return hr;

return NOERROR;
}

```

3. Modify the `DllUnregisterServer` method to add the highlighted code as shown:

```

STDAPI DllUnregisterServer(void)
{
    HRESULT hr;    // HRESULT used by Safety Functions

    AFX_MANAGE_STATE(_afxModuleAddrThis);

    if (!AfxOleUnregisterTypeLib(_tlid, _wVerMajor, _wVerMinor))
        return ResultFromScode(SELFREG_E_TYPELIB);

    if (!COleObjectFactoryEx::UpdateRegistryAll(FALSE))
        return ResultFromScode(SELFREG_E_CLASS);

    // Remove entries from the registry.

    hr=UnRegisterCLSIDInCategory(CLSID_SafeItem,
        CATID_SafeForInitializing);
    if (FAILED(hr))
        return hr;

    hr=UnRegisterCLSIDInCategory(CLSID_SafeItem,
        CATID_SafeForScripting);
    if (FAILED(hr))
        return hr;

    return NOERROR;
}

```

## ActiveX Control Properties, Methods, and Events

Communication between an **ActiveX** control and a web page is done through **ActiveX** control properties, methods, and events. In order to demonstrate these concepts, we will create a simple web page with a form entry to enter a text string. When a Submit button is pressed, the text entered is passed to the **ActiveX** control through an input parameter custom property. A method of the control is called which copies this text to an output parameter custom property, and then fires an event for this text to be displayed on the web page. Simply follow these steps in Visual Studio to implement this:

1. First, we will create the custom properties for passing text to and from the **ActiveX** control. In Class View, expand the element `MyActiveXLib` to select `_DMyActiveX`. Right click on `_DMyActiveX`, then **Add, Add Property**. In the Add Property Wizard dialog as shown in Figure 9, select **BSTR** for *Property type*, and enter `InputParameter` for *Property name*. The wizard will fill other fields automatically for you with `m_InputParameter` for *Variable name* and `OnInputParameterChanged` for *Notification function*. Click the Finish button where the wizard will automatically create the code to support this property. Do the same for *Property name* `OutputParameter` with the same *Property type* **BSTR**.



Figure 9. Add Property Wizard

- Next, we will create a method to enable the web page to notify the control to transfer the text string input parameter to the output parameter. In Class View, expand the element *MyActiveXLib* to select *\_DMyActiveX*. Right click on *\_DMyActiveX*, then *Add, Add Method*. In the Add Property Wizard dialog, as shown in Figure 9, select *void* for Return type and enter *LoadParameter* for Method name. The wizard will automatically enter *LoadParameter* for Internal name. Click *Finish* where the wizard will automatically create the code to support this method.

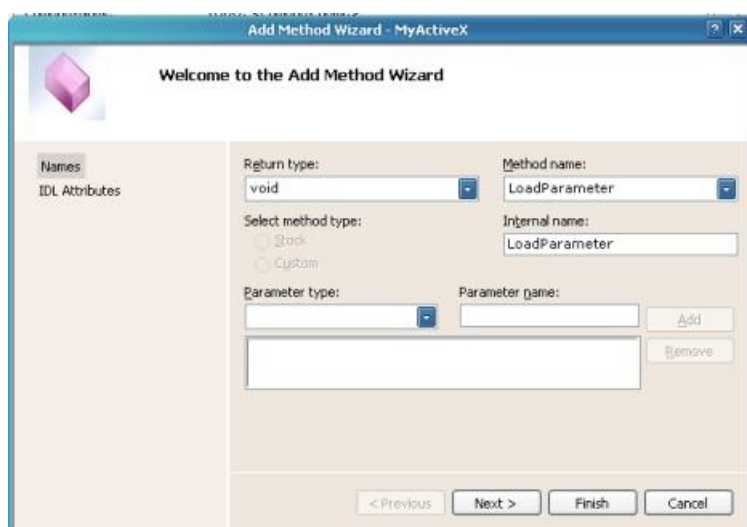


Figure 10. Add Method Wizard

- Now, we will create an event that enables the *ActiveX* control to notify the web page that it is completed transferring the text from the input parameter to the output parameter. Code in the web page will react to this event and respond by displaying the text in the output parameter to verify that this transfer has occurred by the *ActiveX* control. In Class View, right click on *CMyActiveXCtrl*, select *Add, Add Event*. In the Add Event Wizard, as shown in Figure 11, enter *ParameterLoaded* for Event name and change Internal name to *FireParameterLoaded*. Click Finish for the wizard to create the default code to support this event.

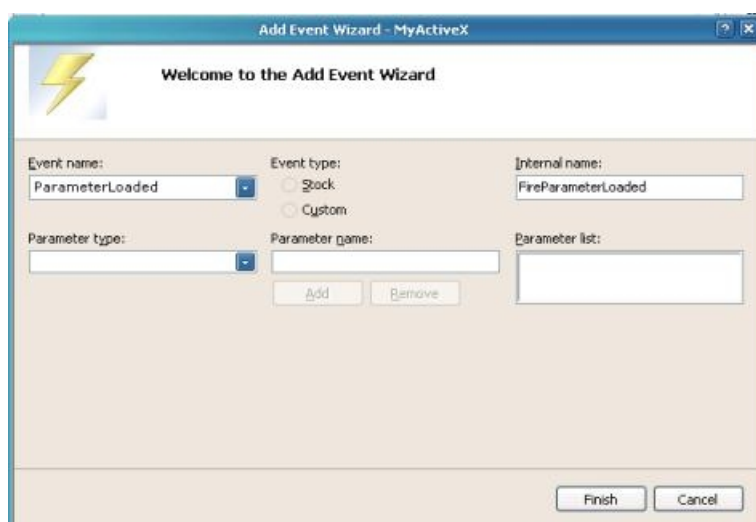


Figure 11. Add Event Wizard

With the above, the wizard has created a majority of the code for you. We only need to add two lines of code to implement the functionality for the **ActiveX** control to copy the text and notify the web page code through an event. Edit the source file *MyActiveXCtrl.cpp*, and add the following code to the *LoadParameter* method.

[Collapse](#) | [Copy Code](#)

```
// Copy text from the input parameter to the output parameter
m_OutputParameter = m_InputParameter;
// Fire an event to notify web page
FireParameterLoaded();
```

To test, use the **ActiveX** Control Pad to create the following HTML code:

[Collapse](#) | [Copy Code](#)

```
<HTML>
<HEAD>
<TITLE>MyActiveX - Methods, Properties, and Events</TITLE>

<SCRIPT LANGUAGE="JavaScript">

function PassParameter()
{
    if (StringInput.value != " ")
    {
        MyActiveX1.InputParameter = StringInput.value;
        MyActiveX1.LoadParameter();
    }
}
</SCRIPT>

</HEAD>
<BODY>
<center>
MyActiveX - Methods, Properties, and Events Example
<p></p>

<OBJECT ID="MyActiveX1" WIDTH=350 HEIGHT=50
    CLASSID="CLSID:36299202-09EF-4ABF-ADB9-47C599DBE778">
    <PARAM NAME="_Version" VALUE="65536">
    <PARAM NAME="_ExtentX" VALUE="2646">
    <PARAM NAME="_ExtentY" VALUE="1323">
    <PARAM NAME="_StockProps" VALUE="0">
</OBJECT>
<p></p>

Input Parameter: <INPUT TYPE="text" NAME="StringInput" VALUE=" ">
<p></p>
<INPUT TYPE="button" NAME="Submit"
    VALUE="Submit" ONCLICK=PassParameter()>

<SCRIPT FOR=MyActiveX1 EVENT=ParameterLoaded()>
<!-- {
    window.document.write("The parameter you entered is:<br> "
        + MyActiveX1.OutputParameter + " ")
-->
</SCRIPT>

</center>
</BODY>
```

Save this HTML code to your web server, and run it. You should see a web page with a progress bar displayed and a form entry to enter the Input Parameter text. Enter text in the field, and press Submit. This should result in a new page with  The parameter you entered is: , followed by the text you entered on the next line. A brief explanation of the HTML code follows.

When you press the Submit button, the JavaScript function *PassParameter* is invoked. This function copies text from the *StringInput* form field to the *InputParameter* property of the **ActiveX** control. It then calls the *LoadParameter* method of the control which copies the text from *InputParameter* to *OutputParameter* and calls *FireParameterLoaded()* to cause an **ActiveX** event. The following HTML code then responds to this event:

[Collapse](#) | [Copy Code](#)

```
<SCRIPT FOR=MyActiveX1 EVENT=ParameterLoaded()>
<!-- {
    window.document.write("The parameter you entered is:<br> " +
        MyActiveX1.OutputParameter + " ")
-->
</SCRIPT>
```

## References:

1. [Add GIF-animation to your MFC and ATL projects with the help of CPictureEx and CPictureExWnd](#) by Oleg Bykov, CodeProject.
2. [Packaging ActiveX Controls](#), Microsoft.

3. [Signing and Checking with Authenticode](#), Microsoft.
4. [Safe Initialization and Scripting for \*\*ActiveX\*\* Controls](#), Microsoft.

## License

This article has no explicit license attached to it but may contain usage terms in the article text or the download files themselves. If in doubt please contact the author via the discussion board below.

A list of licenses authors might use can be found [here](#)

## About the Author

### David Marcionek

Web Developer

 United States

Member

Article **Rate this article for us!** Poor ☐ ☐ ☐ ☐ ☐ Excellent

[Top](#)


Google Data Provider: DataBind to Google Apps!

ADO.NET access to: Gmail, Google Docs, Search, Google Calendar, Adwords, Google Talk, and more.


DOWNLOAD FREE TRIAL: [www.rssbus.com/ado](http://www.rssbus.com/ado)

 rssbus

## Comments and Discussions












































































 **FAQ**

Noise Tolerance  Layout  Per page

 **New Message**

Msgs 1 to 25 of 121 (Total in Forum: 121) ([Refresh](#))

[First](#) [Prev](#) [Next](#)

 <b>My vote of 5</b> 	 <b>sadik</b>	<b>4:27 8 Oct '10</b>
 <b>Can I deploy an ActiveX and call it from server script?</b> 	 Member 2148033	<b>10:48 16 Aug '10</b>
 <b>Passing in/out paramters in Javascript and ActiveX</b> 	 outforlunch	<b>7:28 25 Nov '09</b>
 <b>How do you release ActiveX Control?</b> 	 George Montemayor	<b>10:17 21 Aug '09</b>
 <b>guide lines for creating 64bit activx plugin with MFC</b> 	 hareshel	<b>23:08 5 Aug '09</b>
 <b>What about Vista and IE7 ?</b> 	 <b>Christian Dumon</b>	<b>4:42 26 Jun '09</b>
 <b>Re: What about Vista and IE7 ?</b> 	 Christian Dumon	<b>7:54 26 Jun '09</b>
 <b>Re: What about Vista and IE7 ?</b> 	 mikejobu23	<b>11:04 23 Feb '10</b>
 <b>Great Job</b> 	 <b>carloerei</b>	<b>2:38 7 Apr '09</b>
 <b>Invisible control</b> 	 Gopal.venkatraman	<b>21:00 5 Apr '09</b>
 <b>Event will not be fire if ColeControl method is called at static class method</b> 	 rotalume	<b>23:20 1 Apr '09</b>
 <b>Graphical objects always hidden by ActiveX Control</b> 	 <b>bastardizer</b>	<b>15:48 29 Mar '09</b>
 <b>very good article!</b> 	 rotalume	<b>0:17 23 Mar '09</b>
 <b>Object doesn't support this property or method</b> 	 risinra	<b>13:59 24 Feb '09</b>
 <b>ActiveX web control with DLL files</b> 	 suggam	<b>19:53 10 Feb '09</b>
 <b>Re: ActiveX web control with DLL files</b> 	 Kenneth Claggett	<b>13:40 11 Mar '09</b>
 <b>Re: ActiveX web control with DLL files</b> 	 suggam	<b>15:49 11 Mar '09</b>
 <b>ActiveX Control Pad fails to work with Windows Vista</b> 	 <b>justdan23</b>	<b>9:53 7 Jan '09</b>
 <b>Re: ActiveX Control Pad fails to work with Windows Vista</b> 	 Member 2928947	<b>1:49 23 Apr '09</b>
 <b>How to compose calander (activex control) inside the ActiveX control</b> 	 arkadash	<b>21:45 12 Nov '08</b>
 <b>How to unregistering the ActiveX?</b> 	 <b>pku2009</b>	<b>19:19 22 Sep '08</b>
 <b>Re: How to unregistering the ActiveX?</b> 	 Garth J Lancaster	<b>20:29 22 Sep '08</b>
 <b>Re: How to unregistering the ActiveX?</b> 	 pku2009	<b>0:19 23 Sep '08</b>
 <b>Re: How to unregistering the ActiveX?</b> 	 Garth J Lancaster	<b>3:07 23 Sep '08</b>
 <b>Re: How to unregistering the ActiveX?</b> 	 justdan23	<b>9:12 7 Jan '09</b>

Last Visit: 22:25 18 Nov '10    Last Update: 17:11 21 Nov '10

[1](#) [2](#) [3](#) [4](#) [5](#) [Next](#) »

[General](#)
[News](#)
[Question](#)
[Answer](#)
[Joke](#)
[Rant](#)
[Admin](#)

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+PgUp/PgDown to switch pages.

[PermaLink](#) | 
 [Privacy](#) | 
 [Terms of Use](#)

Last Updated: 21 Jun 2006

Copyright 2006 by David MarcioneK

Everything else Copyright © [CodeProject](#), 1999-2010

Web26 | [Advertise on the Code Project](#)

## Sponsored Links

### ArcGIS Explorer

ArcGIS Explorer is a free downloadable...

[www.esri.com](http://www.esri.com)

### Resco MobileApp Studio

Resco MobileApp Studio is a Microsoft Visual...

[www.resco.net](http://www.resco.net)

### Resco MobileForms Toolkit

The richest, most complete and award winning...

[www.resco.net](http://www.resco.net)

## See Also...

### Creating PDF Documents in ASP.NET

How to create PDF documents in ASP.NET.

### A Fast CSV Reader

A reader that provides fast, non-cached,...

### A Simplified SQL-CSV Import/Export Functionality

A simplified SQL-CSV import/export functionality.

### Importing CSV Data and saving it in database

This article shows how to import CSV data and...

### Transferring Data Using SqlBulkCopy

An article on how to transfer data using...

## Announcements



Register for Our Virtual Tech Summits



WP7 Comp - Post an Article, Share an App



Got an Azure app? Win an Xbox & Kinect

## The Daily Insider

30 free programming books

Daily News: [Signup now.](#)

