This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework 4 - ASP.NET

**Implementing Client Callbacks Programmatically Without Postbacks in ASP.NET Web Pages**

In the default model for ASP.NET Web pages, the user interacts with a page and clicks a button or performs some other action that results in a postback. The page and its controls are re-created, the page code runs on the server, and a new version of the page is rendered to the browser. However, in some situations, it is useful to run server code from the client without performing a postback. If the client script in the page is maintaining some state information (for example, local variable values), posting the page and getting a new copy of it destroys that state. Additionally, page postbacks introduce processing overhead that can decrease performance and force the user to wait for the page to be processed and re-created.

To avoid losing client state and not incur the processing overhead of a server roundtrip, you can code an ASP.NET Web page so that it can perform client callbacks. In a client callback, a client-script function sends a request to an ASP.NET Web page. The Web page runs a modified version of its normal life cycle. The page is initiated and its controls and other members are created, and then a specially marked method is invoked. The method performs the processing that you have coded and then returns a value to the browser that can be read by another client script function. Throughout this process, the page is live in the browser.

Several Web server controls can use client callbacks. For example, the [TreeView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.treeview.aspx) control uses client callbacks to implement its populate-on-demand functionality. For more information see [TreeView Web Server Control Overview](http://msdn.microsoft.com/en-us/library/e8z5184w.aspx).

There are several options for automating client callbacks in an ASP.NET Web page. AJAX features in ASP.NET such as the [UpdatePanel](http://msdn.microsoft.com/en-us/library/system.web.ui.updatepanel.aspx) server control can automate asynchronous partial-page updates for you, and the Web service communication feature can automate asynchronous Web service calls.

For an overview of AJAX features in ASP.NET that automate client callbacks for you, see the following topics:

* [UpdatePanel Control Overview](http://msdn.microsoft.com/en-us/library/bb386454.aspx)
* [Microsoft Ajax Overview](http://msdn.microsoft.com/en-us/library/bb398874.aspx)
* [Partial-Page Rendering Overview](http://msdn.microsoft.com/en-us/library/bb386573.aspx)
* [ASP.NET AJAX Web Services](http://msdn.microsoft.com/en-us/library/bb398785.aspx)

You can also write your own client script to implement client callbacks directly. This topic discusses how to implement your own client callbacks for asynchronous communication between the client and server.

A Visual Studio project with source code is available to accompany this topic: [Download](http://go.microsoft.com/fwlink/?LinkId=192414).

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifComponents of Client Callbacks

Creating an ASP.NET page that programmatically implements client callbacks is similar to creating any ASP.NET page, with a few these differences. The page's server code must perform the following tasks:

* Implement the [ICallbackEventHandler](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface. You can add this interface declaration to any ASP.NET Web page.
* Provide an implementation for the [RaiseCallbackEvent](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method. This method will be invoked to perform the callback on the server.
* Provide an implementation for the [GetCallbackResult](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method. This method will return the callback result to the client.

In addition, the page must contain three client script functions that perform the following actions:

* One function calls a helper method that performs the actual request to the server. In this function, you can perform custom logic to prepare the event arguments first. You can send a string as a parameter to the server-side callback event handler.
* Another function receives (is called by) the result from the server code that processed the callback event, accepting a string that represents the results. This is referred to as the client callback function.
* A third function is the helper function that performs the actual request to the server. This function is generated automatically by ASP.NET when you generate a reference to this function by using the [GetCallbackEventReference](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method in server code.

Both client callbacks and postbacks are requests for the originating page. Therefore, client callbacks and postbacks are recorded in Web server logs as a page request.

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifImplementing the Required Interfaces in Server Code

To run server code from client script without performing a postback, you must implement several interfaces in server code.

**Declaring the ICallbackEventHandler Interface**

You can declare the [ICallbackEventHandler](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface as part of the class declaration for the page. If you are creating a code-behind page, you can declare the interface by using syntax such as the following.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl01_code');" \o "Copy Code)

Partial Class CallBack\_DB\_aspx

Inherits System.Web.UI.Page

Implements System.Web.UI.ICallbackEventHandler

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl02_code');" \o "Copy Code)

public partial class CallBack\_DB\_aspx :

System.Web.UI.Page, System.Web.UI.ICallbackEventHandler

If you are working in a single-file page or user control, you can add the declaration by using an [@ Implements](http://msdn.microsoft.com/en-us/library/cbsf6k72.aspx) directive in the page, as in the following examples:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl04_code');" \o "Copy Code)

<%@ Page Language="VB" %>

<%@ Implements Interface="System.Web.UI.ICallbackEventHandler" %>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl05_code');" \o "Copy Code)

<%@ Page Language="C#" %>

<%@ Implements Interface="System.Web.UI.ICallbackEventHandler" %>

|  |
| --- |
| **Description: NoteNote** |
| The interface name is case-sensitive. |

**Creating a Server Callback Method**

In server code, you must create a method that implements the [RaiseCallbackEvent](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method and a method that implements the [GetCallbackResult](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method. The [RaiseCallbackEvent](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method takes a single string argument instead of the usual two arguments that are typically used with event handlers. A portion of the method might look like the following example:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl10_code');" \o "Copy Code)

Public Sub RaiseCallbackEvent(ByVal eventArgument As String) \_

Implements System.Web.UI.ICallbackEventHandler.RaiseCallbackEvent

End Sub

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl11_code');" \o "Copy Code)

public void RaiseCallbackEvent(String eventArgument)

{

}

The [GetCallbackResult](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method takes no arguments and returns a string. A portion of the method might look like the following example:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl13_code');" \o "Copy Code)

Public Function GetCallbackResult() As String Implements \_

System.Web.UI.ICallbackEventHandler.GetCallbackResult

Return aStringValue

End Function

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl14_code');" \o "Copy Code)

public string GetCallbackResult()

{

return aStringValue;

}

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifCreating Client Script Functions

You must add client script functions to the page to perform two functions: send the callback to the server page, and receive the results. Both client script functions are written in ECMAScript (JavaScript).

**Sending the Callback**

The function to send the callback is generated in server code. The actual callback is performed by a library function that is available to any page that implements the [ICallbackEventHandler](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface. You can get a reference to the library function by calling the page's [GetCallbackEventReference](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method, which is accessible through the [ClientScript](http://msdn.microsoft.com/en-us/library/system.web.ui.page.clientscript.aspx) property of the page. You then build a client function dynamically that includes a call to the return value from the [GetCallbackEventReference](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method. You pass to that method a reference to the page (**this** in C# or **Me** in Visual Basic), the name of the argument that you will use to pass data, the name of the client script function that will receive the callback data, and an argument that passes any context you want.

When you have built the function, you inject it into the page by calling the [RegisterClientScriptBlock](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.registerclientscriptblock.aspx) method.

The following example shows how to dynamically create a function named CallServer that invokes the callback.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl05_code');" \o "Copy Code)

Sub Page\_Load(ByVal sender As Object, ByVal e As System.EventArgs) \_

Handles Me.Load

Dim cm As ClientScriptManager = Page.ClientScript

Dim cbReference As String

cbReference = cm.GetCallbackEventReference(Me, "arg", \_

"ReceiveServerData", "")

Dim callbackScript As String = ""

callbackScript &= "function CallServer(arg, context)" & \_

"{" & cbReference & "; }"

cm.RegisterClientScriptBlock(Me.GetType(), \_

"CallServer", callbackScript, True)End Sub

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl06_code');" \o "Copy Code)

void Page\_Load(object sender, EventArgs e)

{

ClientScriptManager cm = Page.ClientScript;

String cbReference = cm.GetCallbackEventReference(this, "arg",

"ReceiveServerData", "");

String callbackScript = "function CallServer(arg, context) {" +

cbReference + "; }";

cm.RegisterClientScriptBlock(this.GetType(),

"CallServer", callbackScript, true);

}

The names of the arguments that are accepted by the function you are generating must match the names of the values that you pass to the [GetCallbackEventReference](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method.

The following example shows some markup that could be used to invoke the callback and process its return value:

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl08_code');" \o "Copy Code)

<input type="button" value="Callback"

onclick="CallServer(1, alert('Callback'))"/>

<br />

<span id="Message"></span>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl09_code');" \o "Copy Code)

<input type="button" value="Callback"

onclick="CallServer(1, alert('Callback'))"/>

<br />

<span id="Message"></span>

**Receiving the Callback**

You can write the client function that receives callbacks statically in the page. The function must be named to match the name that you pass in the call to the [GetCallbackEventReference](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method. The receiving function accepts two string values: one for the return value and an optional second value for the context value that is passed back from the server.

The function might look like the following example:

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl58_ctl00_ctl11_code');" \o "Copy Code)

<script type="text/javascript">

function ReceiveServerData(arg, context)

{

Message.innerText = 'Processed callback.';

}

</script>