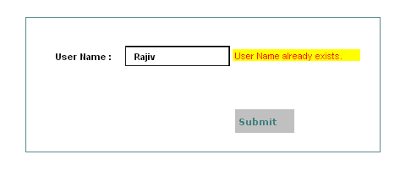
[blogindexer](http://blogindexer.blogspot.in/)

Wednesday, October 13, 2010

AjaxValidator Control in ASP.Net ASPX

"24x7aspnet.blogspot.com Blog says Like the CustomValidator control, the AjaxValidator control enables to create a custom server-side validation function. Unlike the CustomValidator control, however, the AjaxValidator control enables to call the custom validation function.  
  
The AjaxValidator control uses AJAX (Asynchronous JavaScript and XML) to call the server-side validation function from the client. The advantage of using AJAX is that no postback to the server is apparent to the user.  
  
Example : Imagine that you are creating a website form and you need to validate a User Name field. You want to make sure that the User Name entered does not already exist in the database. The AjaxValidator enables you to call a server-side validation function from the client to check whether the User Name is unique in the database.  
  
AjaxValidator.cs  
  
using System;  
using System.Web;  
using System.Web.UI;  
using System.Web.UI.WebControls;  
namespace myControls  
{  
  
public class AjaxValidator : BaseValidator, ICallbackEventHandler  
{  
public event ServerValidateEventHandler ServerValidate;  
string \_controlToValidateValue;  
protected override void OnPreRender(EventArgs e)  
{  
String eventRef = Page.ClientScript.GetCallbackEventReference(this,““,““,““);  
  
String includeScript =Page.ResolveClientUrl(“~/ClientScripts/AjaxValidator.js”);  
Page.ClientScript.RegisterClientScriptInclude(“AjaxValidator”,includeScript);  
  
String startupScript = String.Format(“document.getElementById(‘{0}’).evaluationfunction = ‘AjaxValidatorEvaluateIsValid’;”, this.ClientID);  
Page.ClientScript.RegisterStartupScript(this.GetType(),“AjaxValidator”, startupScript, true);  
base.OnPreRender(e);  
}  
  
  
protected override bool DetermineRenderUplevel()  
{  
return Context.Request.Browser.SupportsCallback;  
}  
  
public string GetCallbackResult()  
{  
return ExecuteValidationFunction(\_controlToValidateValue).ToString();  
}  
  
public void RaiseCallbackEvent(string eventArgument)  
{  
\_controlToValidateValue = eventArgument;  
}  
  
protected override bool EvaluateIsValid()  
{  
string controlToValidateValue = this.GetControlValidationValue  
(this.ControlToValidate);  
return ExecuteValidationFunction(controlToValidateValue);  
}  
  
private bool ExecuteValidationFunction(String controlToValidateValue)  
{  
ServerValidateEventArgs args = new ServerValidateEventArgs  
(controlToValidateValue, this.IsValid);  
if (ServerValidate != null)  
ServerValidate(this, args);  
return args.IsValid;  
}  
}  
}  
  
The control inherits from the BaseValidator class. It also implements the ICallbackEventHandler interface. The ICallbackEventHandler interface defines two methods that are called on the server when an AJAX request is made from the client.  
In the OnPreRender() method, a JavaScript include file and startup script are registered. The JavaScript include file contains the client-side functions that are called when the AjaxValidator validates a form field on the client. The startup script associates the clientside AjaxValidatorEvaluateIsValid() function with the AjaxValidator control. The client-side validation framework automatically calls this JavaScript function when performing validation.  
  
AjaxValidator.js  
  
  
function AjaxValidatorEvaluateIsValid(val)  
{  
var value = ValidatorGetValue(val.controltovalidate);  
WebForm\_DoCallback(val.id, value, AjaxValidatorResult, val,  
AjaxValidatorError, true);  
return true;  
}  
  
function AjaxValidatorResult(returnValue, context)  
{  
if (returnValue == ‘True’)  
context.isvalid = true;  
else  
context.isvalid = false;  
ValidatorUpdateDisplay(context);  
}  
  
function AjaxValidatorError(message)  
{  
alert(‘Error: ‘ + message);  
}  
  
The AjaxValidatorEvaluateIsValid() JavaScript method initiates an AJAX call by calling the WebForm\_DoCallback() method.  
  
<%@ Page Language=”C#” %>  
<%@ Register TagPrefix=”custom” Namespace=”myControls” %>  
<%@ Import Namespace=”System.Data.SqlClient” %>  
<%@ Import Namespace=”System.Web.Configuration” %>  
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>  
<script runat=”server”>  
  
protected void AjaxValidator1\_ServerValidate(object source,ServerValidateEventArgs args)  
{  
if (UserNameExists(args.Value))  
args.IsValid = false;  
else  
args.IsValid = true;  
}  
>  
private bool UserNameExists(string userName)  
{  
if username found in database return "true" else return "false"  
}  
return result;  
}  
  
protected void btnSubmit\_Click(object sender, EventArgs e)  
{  
  
}  
</script>  
  
<html xmlns=”http://www.w3.org/1999/xhtml” >  
<head runat=”server”>  
<title>Demo AjaxValidator</title>  
</head>  
<body>  
<form id=”form1” runat=”server”>  
<div>  
<asp:Label id=”lblUserName” Text=”User Name:” AssociatedControlID=”txtUserName” Runat=”server” />  
<asp:TextBox id=”txtUserName” Runat=”server” />  
<custom:AjaxValidator id=”AjaxValidator1” ControlToValidate=”txtUserName” Text=”User name already taken!” OnServerValidate=”AjaxValidator1\_ServerValidate” Runat=”server” />  
<asp:Button id=”btnSubmit” Text=”Submit” Runat=”server” OnClick=”btnSubmit\_Click” />  
</div>  
</form>  
</body>  
</html>

Posted by [csharphelper](https://www.blogger.com/profile/13598957688466963652" \o "author profile)at [9:54 AM](http://blogindexer.blogspot.in/2010/10/ajaxvalidator-control-in-aspnet-aspx_13.html)

[Email This](https://www.blogger.com/share-post.g?blogID=4224381246143981431&postID=6666659026515962646&target=email)[BlogThis!](https://www.blogger.com/share-post.g?blogID=4224381246143981431&postID=6666659026515962646&target=blog)[Share to Twitter](https://www.blogger.com/share-post.g?blogID=4224381246143981431&postID=6666659026515962646&target=twitter)[Share to Facebook](https://www.blogger.com/share-post.g?blogID=4224381246143981431&postID=6666659026515962646&target=facebook)[Share to Pinterest](https://www.blogger.com/share-post.g?blogID=4224381246143981431&postID=6666659026515962646&target=pinterest)

DEMO 2

Client-Callback Implementation (C#) Example

**.NET Framework 3.0**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Demonstrates an ASP.NET Web page that implements a client callback. For more information, see[Implementing Client Callbacks Without Postbacks in ASP.NET Web Pages](https://msdn.microsoft.com/en-us/library/ms178208(v=vs.85).aspx).

Example

The following code example is in two parts. The first part of the example shows an ASP.NET Web page (the .aspx page). The second part shows the corresponding code-behind file (the .aspx.cs file).

C#

<%@ Page Language="C#" AutoEventWireup="true"

CodeFile="ClientCallback.aspx.cs" Inherits="ClientCallback" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML

1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head id="Head1" runat="server">

<title>Client Callback Example</title>

<script type="text/ecmascript">

function LookUpStock()

{

var lb = document.getElementById("ListBox1");

var product = lb.options[lb.selectedIndex].text;

CallServer(product, "");

}

function ReceiveServerData(rValue)

{

document.getElementById("ResultsSpan").innerHTML = rValue;

}

</script>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:ListBox ID="ListBox1" Runat="server"></asp:ListBox>

<br />

<br />

<button type="Button" onclick="LookUpStock()">Look Up Stock</button>

<br />

<br />

Items in stock: <span id="ResultsSpan" runat="server"></span>

<br />

</div>

</form>

</body>

</html>

C#

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

public partial class ClientCallback : System.Web.UI.Page,

System.Web.UI.ICallbackEventHandler

{

protected System.Collections.Specialized.ListDictionary catalog;

protected String returnValue;

protected void Page\_Load(object sender, EventArgs e)

{

String cbReference =

Page.ClientScript.GetCallbackEventReference(this,

"arg", "ReceiveServerData", "context");

String callbackScript;

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

"{ " + cbReference + ";}";

Page.ClientScript.RegisterClientScriptBlock(this.GetType(),

"CallServer", callbackScript, true);

catalog = new System.Collections.Specialized.ListDictionary();

catalog.Add("monitor", 12);

catalog.Add("laptop", 10);

catalog.Add("keyboard", 23);

catalog.Add("mouse", 17);

ListBox1.DataSource = catalog;

ListBox1.DataTextField = "key";

ListBox1.DataBind();

}

public void RaiseCallbackEvent(String eventArgument)

{

if (catalog[eventArgument] == null)

{

returnValue = "-1";

}

else

{

returnValue = catalog[eventArgument].ToString();

}

}

public String GetCallbackResult()

{

return returnValue;

}

}

The Web page emulates a database lookup to determine the number of items that are available, or in stock, for a series of products (monitors, keyboards, and so on). To simplify this code example, the database is represented by a dictionary list that contains a small set of items. For each item in the table, the key is the item name (such as monitor) and the value is the number of items that are in stock. In a production application, a database would be used instead.

When the page runs, a [ListBox](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox(v=vs.85).aspx) control is bound to the hash table so that the **ListBox** control displays the list of products. The page also contains a **button** element (not a **Button** Web server control), whose **onclick** event is bound to a client function named **LookUpStock**. When users click the button, the button executes the **LookUpStock** function, which gets the current selection from the list box and then performs the client callback by calling the **CallServer** function.

The code-behind page adds client-side script to the page via the [RegisterClientScriptBlock](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.registerclientscriptblock(v=vs.85).aspx) method. The script that is added to the page includes a function called **CallServer**, which gets the name of the method that will post back to the server from the [GetCallbackEventReference](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference(v=vs.85).aspx) method.

The client callback invokes the [RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent(v=vs.85).aspx) method, to determine the available stock for the product passed to it. The [GetCallbackResult](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult(v=vs.85).aspx) method returns the value. Note that the arguments sent between the client script and the server code can only be strings. To pass in or to receive multiple values, you can concatenate values in the input or return string, respectively.

# How to: Implement Callbacks in ASP.NET Web Pages

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

In a client callback, a client script function sends a request to the ASP.NET Web page, which then runs an abbreviated version of its normal life cycle to process the callback. To ensure that callback events originate from the expected user interface (UI), you can validate callbacks. In callback validation, you register an event for validation during the Web page rendering and then validate the event during the callback. For an overview of callbacks, see [Implementing Client Callbacks Programmatically Without Postbacks in ASP.NET Web Pages](https://msdn.microsoft.com/en-us/library/ms178208.aspx).

### To implement the ICallBackEventHandler interface

1. For a single-file page or user control, implement the [ICallbackEventHandler](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface using an[@ Implements](https://msdn.microsoft.com/en-us/library/cbsf6k72.aspx) directive in the page, as shown in the following example.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms366518.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

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

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

|  |
| --- |
| **Note Note** |
| If you are using a code-behind page model, implement the [ICallbackEventHandler](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface for your partial class. |

1. Implement the [RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method of the [ICallbackEventHandler](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface. The[RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method takes a single argument that represents the event arguments, as shown in the following example.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms366518.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

public void RaiseCallbackEvent(String eventArgument)

{

}

1. Implement the [GetCallbackResult](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method of the [ICallbackEventHandler](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.aspx) interface. The[GetCallbackResult](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method takes no arguments and returns a string that represents the result of the callback. In the following example, a string called returnValue is returned.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms366518.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

public String GetCallbackResult()

{

return returnValue;

}

### To register the callback for event validation

* Override the [Render](https://msdn.microsoft.com/en-us/library/20b8wk8d.aspx) method of the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class and use the [RegisterForEventValidation](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.registerforeventvalidation.aspx) method of the [ClientScriptManager](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.aspx) class to register an event for validation. You can get a reference to the [ClientScriptManager](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.aspx) class by using the [ClientScript](https://msdn.microsoft.com/en-us/library/system.web.ui.page.clientscript.aspx) property of the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) class. In the following example, a callback named Callback1 is registered for event validation.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms366518.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

protected override void Render(HtmlTextWriter writer)

{

Page.ClientScript.RegisterForEventValidation("ClientCallback1");

base.Render(writer);

}

### To validate the callback and return the callback result

* In the [RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method, use the [ValidateEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.validateevent.aspx) method of the [ClientScriptManager](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.aspx)class to validate the event. Use the same method signature as the one used when registering the event for validation. In the following example, the method signature that takes one argument is used.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms366518.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

public void RaiseCallbackEvent(String eventArgument)

{

try

{

Page.ClientScript.ValidateEvent("ClientCallback1");

// Callback logic goes here.

returnValue = "callback result";

}

catch

{

// Failed callback validation logic.

}

}

If validation passes, your code should proceed with the callback event logic. After the[RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method has completed, the [GetCallbackResult](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method is invoked to return the callback result as a string to a client script function.

For more information about creating and implementing the client script functions to support client callbacks, see [Implementing Client Callbacks Programmatically Without Postbacks in ASP.NET Web Pages](https://msdn.microsoft.com/en-us/library/ms178208.aspx) and [Client Callback with Validation Implementation Example](https://msdn.microsoft.com/en-us/library/ms366515.aspx).

# BaseValidator.DetermineRenderUplevel Method

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Determines whether the validation control can perform client-side validation.

**Namespace:**  [System.Web.UI.WebControls](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.determinerenderuplevel(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.determinerenderuplevel(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.determinerenderuplevel(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.determinerenderuplevel(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

protected virtual bool DetermineRenderUplevel()

#### Return Value

Type: [System.Boolean](https://msdn.microsoft.com/en-us/library/system.boolean(v=vs.110).aspx)  
**true** if the validation control can perform client-side validation; otherwise, **false**.

## [Remarks](javascript:void(0))

The DetermineRenderUplevel method is a helper function that is used primarily by the[RenderUplevel](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.renderuplevel(v=vs.110).aspx) property to determine whether the client's browser supports client script. For this property to return **true**, the following conditions must be true:

[Developer Network](https://msdn.microsoft.com/en-us)

[Sign in](https://login.live.com/login.srf?wa=wsignin1.0&rpsnv=12&ct=1439821676&rver=6.0.5276.0&wp=MCMBI&wlcxt=MSDN%24MSDN%24MSDN&wreply=https%3a%2f%2fmsdn.microsoft.com%2fen-us%2flibrary%2fms178208.aspx&lc=1033&id=254354&mkt=en-US)

[MSDN subscriptions](https://msdn.microsoft.com/subscriptions/manage/hh442900)

[Get tools](http://go.microsoft.com/fwlink/?LinkId=309297&clcid=0x409&slcid=0x409&campaign=o~msft~msdn~gettools-header~dn308572)

* <http://www.facebook.com/microsoftdeveloper>
* <https://twitter.com/msdev>
* <http://plus.google.com/111221966647232053570/>

* [Technologies](javascript:void(0))
* [Downloads](javascript:void(0))
* [Programs](javascript:void(0))
* [Community](javascript:void(0))
* [Documentation](javascript:void(0))
* [Samples](https://code.msdn.microsoft.com/)

Top of Form

Bottom of Form

**Was this page helpful?**

Your feedback about this content is important. Let us know what you think.

YesNo

[MSDN Library](https://msdn.microsoft.com/en-us/library/ms178208.aspx)

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

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

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

## [Components of Client Callbacks](javascript:void(0))

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](https://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](https://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](https://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](https://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.

## [Implementing the Required Interfaces in Server Code](javascript:void(0))

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

### [Declaring the ICallbackEventHandler Interface](javascript:void(0))

You can declare the [ICallbackEventHandler](https://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.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms178208.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

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](https://msdn.microsoft.com/en-us/library/cbsf6k72.aspx) directive in the page, as in the following examples:

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms178208.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

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

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

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

### [Creating a Server Callback Method](javascript:void(0))

In server code, you must create a method that implements the [RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method and a method that implements the[GetCallbackResult](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.aspx) method. The [RaiseCallbackEvent](https://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:

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms178208.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-3)

public void RaiseCallbackEvent(String eventArgument)

{

}

The [GetCallbackResult](https://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:

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms178208.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-4)

public string GetCallbackResult()

{

return aStringValue;

}

## [Creating Client Script Functions](javascript:void(0))

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](javascript:void(0))

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](https://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](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method, which is accessible through the [ClientScript](https://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](https://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](https://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.

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms178208.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-5)

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](https://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:

C#

[**VB**](https://msdn.microsoft.com/en-us/library/ms178208.aspx?cs-save-lang=1&cs-lang=vb#code-snippet-6)

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

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

<br />

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

### [Receiving the Callback](javascript:void(0))

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](https://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:

<script type="text/javascript">

function ReceiveServerData(arg, context)

{

Message.innerText = 'Processed callback.';

}

</script>

## [See Also](javascript:void(0))

#### Reference

[ClientScriptManager](https://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.aspx)

#### Concepts

[ASP.NET Client Script](https://msdn.microsoft.com/en-us/library/dd410060.aspx)

[Client-Callback Implementation (Visual Basic) Example](https://msdn.microsoft.com/en-us/library/ms178209.aspx)

[Client-Callback Implementation (C#) Example](https://msdn.microsoft.com/en-us/library/ms178210.aspx)

[Any Suggestions?](javascript:void(0)) [Print](https://msdn.microsoft.com/en-us/library/ms178208(d=printer).aspx) [Export (0)](https://msdn.microsoft.com/en-us/library/export/help/?returnUrl=%2fen-us%2flibrary%2fms178208.aspx)

Dev centers

* [Windows](https://dev.windows.com/)
* [Office](http://dev.office.com/)
* [Visual Studio](https://msdn.microsoft.com/vstudio)
* [Microsoft Azure](http://azure.microsoft.com/en-us/documentation/)
* [More...](https://msdn.microsoft.com/aa937802)

Learning resources

* [Microsoft Virtual Academy](http://www.microsoftvirtualacademy.com/)
* [Channel 9](http://channel9.msdn.com/)
* [Interoperability Bridges](http://www.interoperabilitybridges.com/)
* [MSDN Magazine](https://msdn.microsoft.com/magazine/)

Community

* [Forums](https://social.msdn.microsoft.com/forums/en-us/home)
* [Blogs](http://blogs.msdn.com/b/developer-tools/)
* [Codeplex](http://www.codeplex.com/)

Support

* [Self support](https://msdn.microsoft.com/hh361695)

Programs

* [BizSpark (for startups)](https://www.microsoft.com/bizspark/)
* [DreamSpark](https://www.dreamspark.com/)
* [Imagine Cup](http://www.imaginecup.com/)

Top of Form

[United States (English)](https://msdn.microsoft.com/en-us/library/ms178208.aspx)

Bottom of Form

* [Newsletter](https://msdn.microsoft.com/newsletter.aspx)
* [Privacy & cookies](https://msdn.microsoft.com/dn529288)
* [Terms of use](https://msdn.microsoft.com/cc300389)
* [Trademarks](https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/EN-US.aspx)

© 2015 Microsoft

# CustomValidator.EvaluateIsValid Method

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.

Overrides the [BaseValidator.EvaluateIsValid](https://msdn.microsoft.com/en-us/library/system.web.ui.mobilecontrols.basevalidator.evaluateisvalid(v=vs.110).aspx) method.

**Namespace:**  [System.Web.UI.WebControls](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.evaluateisvalid(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.evaluateisvalid(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.evaluateisvalid(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.evaluateisvalid(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

protected override bool EvaluateIsValid()

#### Return Value

Type: [System.Boolean](https://msdn.microsoft.com/en-us/library/system.boolean(v=vs.110).aspx)  
**true** if the value in the input control is valid; otherwise, **false**.

## [Examples](javascript:void(0))

The following code example demonstrates how to override the EvaluateIsValid method in a custom server control.

|  |
| --- |
| **Security noteSecurity Note** |
| This example has a text box that accepts user input, which is a potential security threat. By default, ASP.NET Web pages validate that user input does not include script or HTML elements. For more information, see [Script Exploits Overview](https://msdn.microsoft.com/en-us/library/w1sw53ds(v=vs.110).aspx). |

C#

[**VB**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.evaluateisvalid(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

<%@ Register TagPrefix="aspSample" Namespace="Samples.AspNet.CS.Controls" Assembly="Samples.AspNet.CS" %>

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head>

<title>Custom CustomValidator - EvaluateIsValid - C# Example</title>

</head>

<body>

<form id="Form1" method="post" runat="server">

<h3>Custom CustomValidator - EvaluateIsValid - C# Example</h3>

<asp:Label id="Label1" runat="server" Text="Enter an even number:"

AssociatedControlID="TextBox1" /><br />

<asp:TextBox id="TextBox1" runat="server" />&nbsp;

<aspSample:CustomCustomValidatorEvaluateIsValid

id="Customvalidator1"

runat="server"

ControlToValidate="TextBox1"

Display="Static"

ErrorMessage="Not an even number!" /><br /><br />

<asp:Button id="Button1" runat="server" Text="Validate" />

</form>

</body>

</html>

...

using System.Web;

using System.Security.Permissions;

namespace Samples.AspNet.CS.Controls

{

[AspNetHostingPermission(SecurityAction.Demand, Level = AspNetHostingPermissionLevel.Minimal)]

public sealed class CustomCustomValidatorEvaluateIsValid : System.Web.UI.WebControls.CustomValidator

{

protected override bool EvaluateIsValid()

{

bool isValid = false;

// Get the name of the control to validate.

string controlToValidate = this.ControlToValidate;

if (controlToValidate.Length > 0)

{

// Get the control's value.

string controlValue = GetControlValidationValue(controlToValidate);

// If the value is not null and not empty, test whether

// check if the value entered into the text box is even,

// if so return true, else return false in all other cases.

if ((controlValue != null) && (!controlValue.Trim().Equals(System.String.Empty)))

{

try

{

int i = int.Parse(controlValue);

isValid = ((i%2) == 0);

}

catch

{}

}

}

return isValid;

}

}

}

# ClientScriptManager.GetCallbackEventReference Method (Control, String, String, String)

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Obtains a reference to a client function that, when invoked, initiates a client call back to a server event. The client function for this overloaded method includes a specified control, argument, client script, and context.

**Namespace:**  [System.Web.UI](https://msdn.microsoft.com/en-us/library/system.web.ui(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/ms153103(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/ms153103(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/ms153103(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/ms153103(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

public string GetCallbackEventReference(

Control control,

string argument,

string clientCallback,

string context

)

#### Parameters

*control*

Type: [System.Web.UI.Control](https://msdn.microsoft.com/en-us/library/system.web.ui.control(v=vs.110).aspx)

The server [Control](https://msdn.microsoft.com/en-us/library/system.web.ui.control(v=vs.110).aspx) that handles the client callback. The control must implement the [ICallbackEventHandler](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler(v=vs.110).aspx) interface and provide a[RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent(v=vs.110).aspx) method.

*argument*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

An argument passed from the client script to the server

[RaiseCallbackEvent](https://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent(v=vs.110).aspx) method.

*clientCallback*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

The name of the client event handler that receives the result of the successful server event.

*context*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

The client script that is evaluated on the client prior to initiating the callback. The result of the script is passed back to the client event handler.

#### Return Value

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)  
The name of a client function that invokes the client callback.

# Control.ResolveClientUrl Method

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Gets a URL that can be used by the browser.

**Namespace:**  [System.Web.UI](https://msdn.microsoft.com/en-us/library/system.web.ui(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/system.web.ui.control.resolveclienturl(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/system.web.ui.control.resolveclienturl(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/system.web.ui.control.resolveclienturl(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/system.web.ui.control.resolveclienturl(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

public string ResolveClientUrl(

string relativeUrl

)

#### Parameters

*relativeUrl*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

A URL relative to the current page.

#### Return Value

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)  
A fully qualified URL to the specified resource suitable for use on the browser.

#### Implements

[IUrlResolutionService.ResolveClientUrl(String)](https://msdn.microsoft.com/en-us/library/system.web.ui.iurlresolutionservice.resolveclienturl(v=vs.110).aspx)

## [Exceptions](javascript:void(0))

# ClientScriptManager.RegisterClientScriptInclude Method (Type, String, String)

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Registers the client script include with the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page(v=vs.110).aspx) object using a type, a key, and a URL.

**Namespace:**  [System.Web.UI](https://msdn.microsoft.com/en-us/library/system.web.ui(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/kx145dw2(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/kx145dw2(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/kx145dw2(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/kx145dw2(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

public void RegisterClientScriptInclude(

Type type,

string key,

string url

)

#### Parameters

*type*

Type: [System.Type](https://msdn.microsoft.com/en-us/library/system.type(v=vs.110).aspx)

The type of the client script include to register.

*key*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

The key of the client script include to register.

*url*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

The URL of the client script include to register.

# lientScriptManager.RegisterStartupScript Method (Type, String, String, Boolean)

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Registers the startup script with the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page(v=vs.110).aspx) object using a type, a key, a script literal, and a Boolean value indicating whether to add script tags.

**Namespace:**  [System.Web.UI](https://msdn.microsoft.com/en-us/library/system.web.ui(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/z9h4dk8y(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/z9h4dk8y(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/z9h4dk8y(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/z9h4dk8y(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

public void RegisterStartupScript(

Type type,

string key,

string script,

bool addScriptTags

)

#### Parameters

*type*

Type: [System.Type](https://msdn.microsoft.com/en-us/library/system.type(v=vs.110).aspx)

The type of the startup script to register.

*key*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

The key of the startup script to register.

*script*

Type: [System.String](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx)

The startup script literal to register.

*addScriptTags*

Type: [System.Boolean](https://msdn.microsoft.com/en-us/library/system.boolean(v=vs.110).aspx)

A Boolean value indicating whether to add script tags.

# ServerValidateEventHandler Delegate

**.NET Framework 4.6 and 4.5**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635745069398291661

Represents the method that will handle the [ServerValidate](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator.servervalidate(v=vs.110).aspx) event of a [CustomValidator](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.customvalidator(v=vs.110).aspx) control.

**Namespace:**  [System.Web.UI.WebControls](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols(v=vs.110).aspx)  
**Assembly:**  System.Web (in System.Web.dll)

## [Syntax](javascript:void(0))

C#

[**C++**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventhandler(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1)

[**F#**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventhandler(v=vs.110).aspx?cs-save-lang=1&cs-lang=fsharp#code-snippet-1)

[**JScript**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventhandler(v=vs.110).aspx?cs-save-lang=1&cs-lang=jscript#code-snippet-1)

[**VB**](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventhandler(v=vs.110).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

public delegate void ServerValidateEventHandler(

Object source,

ServerValidateEventArgs args

)

#### Parameters

*source*

Type: [System.Object](https://msdn.microsoft.com/en-us/library/system.object(v=vs.110).aspx)

The source of the event.

*args*

Type: [System.Web.UI.WebControls.ServerValidateEventArgs](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventargs(v=vs.110).aspx)

A [ServerValidateEventArgs](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.servervalidateeventargs(v=vs.110).aspx) that contains the event data.