This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework 4 - ASP.NET

**Client-Callback Implementation (C#) Example**

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

Description: http://i.msdn.microsoft.com/Global/Images/clear.gifExample

**Description**

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).

|  |
| --- |
| **Description: NoteNote** |
| The example requires the page to be named ClientCallback.aspx and the code-behind file to be named ClientCallback.aspx.cs. |

**Code**

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl01_code');" \o "Copy Code)

<%@ 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 >

<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#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl20_ctl00_ctl02_code');" \o "Copy Code)

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;

}

}

**Comments**

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](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) control is bound to the hash table so that the [ListBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) 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](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.registerclientscriptblock.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](http://msdn.microsoft.com/en-us/library/system.web.ui.clientscriptmanager.getcallbackeventreference.aspx) method.

The client callback invokes the [RaiseCallbackEvent](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.raisecallbackevent.aspx) method, to determine the available stock for the product passed to it. The [GetCallbackResult](http://msdn.microsoft.com/en-us/library/system.web.ui.icallbackeventhandler.getcallbackresult.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.

|  |
| --- |
| **Description: Security noteSecurity Note** |
| When you use this feature, there are potential security threats. Callback arguments are not validated and therefore should be considered unsafe. You should always check the contents of the arguments before using them. For details, see [Script Exploits Overview](http://msdn.microsoft.com/en-us/library/w1sw53ds.aspx). |