This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

**Client Callback with Validation Implementation Example**

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. Callback validation involves registering an event for validation during the Web page rendering and then validating the event during the callback.

|  |
| --- |
| **Description: NoteNote** |
| Event validation helps secure your Web application against forged postbacks but does not protect against replay attacks. A more comprehensive event validation scheme should take into account the specifics of your Web application and the permissions of the user accessing its resources. For more information, see [ASP.NET Web Application Security](http://msdn.microsoft.com/en-us/library/330a99hc.aspx). |

The example discussed here extends the [Client-Callback Implementation (C#) Example](http://msdn.microsoft.com/en-us/library/ms178210.aspx) and the [Client-Callback Implementation (Visual Basic) Example](http://msdn.microsoft.com/en-us/library/ms178209.aspx). In those examples, a [ListBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) control named ListBox1 is a server-side control that displays a list of products. An HTML **<button>** element (not a [Button](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.button.aspx) server control) performs a callback to get product inventory information. The example is extended to introduce additional information about whether a product is on sale and to allow this information to be viewed by authenticated users only. A [LoginView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.aspx) control is used with the [LoggedInTemplate](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.loggedintemplate.aspx) property set to display additional content. Anonymous users of the Web page are allowed to execute a callback to get inventory information, whereas logged-in users are also allowed to execute a callback to get sale information. The callback for the sale information is registered for event validation only if the user is authenticated. This prevents execution of the callback by users who are not authenticated.

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

**Description**

In the following example, a Web page emulates a database lookup to determine the number of items that are available and whether an item is on sale. To simplify the example, the data store is represented by two dictionary lists. In a production application, a database would be used instead. The example demonstrates a scenario where validating client callbacks prevents an anonymous user from executing a callback that is intended for authenticated users only.

**Code**

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl39_ctl00_ctl00_code');" \o "Copy Code)

<%@ Page Language="VB" AutoEventWireup="false"

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

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

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

<html >

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

<title>ASP.NET Example</title>

<script type="text/javascript">

function ReceiveServerData(rValue)

{

Results.innerText = rValue;

}

</script>

</head>

<body>

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

<div>

<asp:ListBox id="ListBox1" runat="server"></asp:ListBox>

<br />

<br />

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

<asp:LoginView id="LoginView1" runat="server">

<LoggedInTemplate>

<button id="LookUpSaleButton" type="button" onclick="LookUpSale()">Look Up Back Order</button>

</LoggedInTemplate>

</asp:LoginView>

<br />

Item status: <span id="Results"></span>

</div>

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl39_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>ASP.NET Example</title>

<script type="text/javascript">

function ReceiveServerData(rValue)

{

Results.innerText = rValue;

}

</script>

</head>

<body>

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

<div>

<asp:ListBox id="ListBox1" runat="server"></asp:ListBox>

<br />

<br />

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

<asp:LoginView id="LoginView1" runat="server">

<LoggedInTemplate>

<button id="LookUpSaleButton" type="button" onclick="LookUpSale()">Look Up Back Order</button>

</LoggedInTemplate>

</asp:LoginView>

<br />

Item status: <span id="Results"></span>

</div>

</form>

</body>

</html>

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl39_ctl00_ctl02_code');" \o "Copy Code)

Partial Class ClientCallback

Inherits System.Web.UI.Page

Implements System.Web.UI.ICallbackEventHandler

Protected catalog As ListDictionary

Protected saleitem As ListDictionary

Protected returnValue As String

Protected validationLookUpStock As String = "LookUpStock"

Protected validationLookUpSale As String = "LookUpSale"

Sub Page\_Load(ByVal sender As Object, ByVal e As \_

System.EventArgs) Handles Me.Load

Page.ClientScript.RegisterClientScriptBlock(Me.GetType(), \_

validationLookUpStock, "function LookUpStock() { " & \_

"var lb = document.forms[0].ListBox1; " & \_

"if (lb.selectedIndex == -1) { alert ('Please make a selection.'); return; } " & \_

"var product = lb.options[lb.selectedIndex].text; " & \_

"CallServer(product, ""LookUpStock"");} ", True)

If (User.Identity.IsAuthenticated) Then

Page.ClientScript.RegisterClientScriptBlock(Me.GetType(), \_

validationLookUpSale, "function LookUpSale() { " & \_

"var lb = document.forms[0].ListBox1; " & \_

"if (lb.selectedIndex == -1) { alert ('Please make a selection.'); return; } " & \_

"var product = lb.options[lb.selectedIndex].text; " & \_

"CallServer(product, ""LookUpSale"");} ", True)

End If

Dim cbReference As String

cbReference = "var param = arg + '|' + context;" & \_

Page.ClientScript.GetCallbackEventReference(Me, \_

"param", "ReceiveServerData", "context")

Dim callbackScript As String = ""

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

cbReference & "} ;"

Page.ClientScript.RegisterClientScriptBlock(Me.GetType(), \_

"CallServer", callbackScript, True)

' Populate List Dictionary with invented database data

catalog = New ListDictionary()

saleitem = New ListDictionary()

catalog.Add("monitor", 12)

catalog.Add("laptop", 10)

catalog.Add("keyboard", 23)

catalog.Add("mouse", 17)

saleitem.Add("monitor", 1)

saleitem.Add("laptop", 0)

saleitem.Add("keyboard", 0)

saleitem.Add("mouse", 1)

ListBox1.DataSource = catalog

ListBox1.DataTextField = "key"

ListBox1.DataBind()

End Sub

Public Sub RaiseCallbackEvent(ByVal eventArgument As String) \_

Implements System.Web.UI.ICallbackEventHandler.RaiseCallbackEvent

Dim argParts() As String = eventArgument.Split("|"c)

If ((argParts Is Nothing) OrElse (argParts.Length <> 2)) Then

returnValue = "A problem occurred trying to retrieve stock count."

Return

End If

Dim product As String = argParts(0)

Dim validationaction = argParts(1)

Select Case validationaction

Case "LookUpStock"

Try

Page.ClientScript.ValidateEvent("LookUpStockButton", validationaction)

If (catalog(product) Is Nothing) Then

returnValue = "Item not found."

Else

returnValue = catalog(product).ToString() & " in stock."

End If

Catch

returnValue = "Can not retrieve stock count."

End Try

Case "LookUpSale"

Try

Page.ClientScript.ValidateEvent("LookUpSaleButton", validationaction)

If (saleitem(product) Is Nothing) Then

returnValue = "Item not found."

Else

If (Convert.ToBoolean(saleitem(product))) Then

returnValue = "Item is on sale."

Else

returnValue = "Item is not on sale."

End If

End If

Catch

returnValue = "Can not retrieve sale status."

End Try

End Select

End Sub

Public Function GetCallbackResult() \_

As String Implements \_

System.Web.UI.ICallbackEventHandler.GetCallbackResult

Return returnValue

End Function

Protected Overrides Sub Render(ByVal writer As System.Web.UI.HtmlTextWriter)

Page.ClientScript.RegisterForEventValidation("LookUpStockButton", \_

validationLookUpStock)

If (User.Identity.IsAuthenticated) Then

Page.ClientScript.RegisterForEventValidation("LookUpSaleButton", \_

validationLookUpSale)

End If

MyBase.Render(writer)

End Sub

End Class

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl39_ctl00_ctl03_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 System.Collections.Specialized.ListDictionary saleitem;

protected String returnValue;

protected String validationLookUpStock = "LookUpStock";

protected String validationLookUpSale = "LookUpSale";

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

{

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

validationLookUpStock, "function LookUpStock() { " +

"var lb = document.forms[0].ListBox1; " +

"if (lb.selectedIndex == -1) { alert ('Please make a selection.'); return; } " +

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

@"CallServer(product, ""LookUpStock"");} ", true);

if (User.Identity.IsAuthenticated)

{

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

validationLookUpSale, "function LookUpSale() { " +

"var lb = document.forms[0].ListBox1; " +

"if (lb.selectedIndex == -1) { alert ('Please make a selection.'); return; } " +

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

@"CallServer(product, ""LookUpSale"");} ", true);

}

String cbReference = "var param = arg + '|' + context;" +

Page.ClientScript.GetCallbackEventReference(this,

"param", "ReceiveServerData", "context");

String callbackScript;

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

"{ " + cbReference + "} ;";

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

"CallServer", callbackScript, true);

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

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

catalog.Add("monitor", 12);

catalog.Add("laptop", 10);

catalog.Add("keyboard", 23);

catalog.Add("mouse", 17);

saleitem.Add("monitor", 1);

saleitem.Add("laptop", 0);

saleitem.Add("keyboard", 0);

saleitem.Add("mouse", 1);

ListBox1.DataSource = catalog;

ListBox1.DataTextField = "key";

ListBox1.DataBind();

}

public void RaiseCallbackEvent(String eventArgument)

{

string[] argParts = eventArgument.Split('|');

if ((argParts == null) || (argParts.Length != 2))

{

returnValue = "A problem occurred trying to retrieve stock count.";

return;

}

string product = argParts[0];

string validationaction = argParts[1];

switch (validationaction)

{

case "LookUpStock":

try

{

Page.ClientScript.ValidateEvent("LookUpStockButton", validationaction);

if (catalog[product] == null)

{

returnValue = "Item not found.";

}

else

{

returnValue = catalog[product].ToString() + " in stock.";

}

}

catch

{

returnValue = "Can not retrieve stock count.";

}

break;

case "LookUpSale":

try

{

Page.ClientScript.ValidateEvent("LookUpSaleButton", validationaction);

if (saleitem[product] == null)

{

returnValue = "Item not found.";

}

else

{

if (Convert.ToBoolean(saleitem[product]))

returnValue = "Item is on sale.";

else

returnValue = "Item is not on sale.";

}

}

catch

{

returnValue = "Can not retrieve sale status.";

}

break;

}

}

public String GetCallbackResult()

{

return returnValue;

}

protected override void Render(HtmlTextWriter writer)

{

Page.ClientScript.RegisterForEventValidation("LookUpStockButton",

validationLookUpStock);

if (User.Identity.IsAuthenticated)

{

Page.ClientScript.RegisterForEventValidation("LookUpSaleButton",

validationLookUpSale);

}

base.Render(writer);

}

}

**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. For authenticated users, the page is rendered with two HTML **<button>** elements whose **onclick** events are bound to a client function named LookUpStock and a client function named LookUpSale, respectively. For anonymous users, the page is rendered with only one HTML **<button>** element, whose **onclick** event is bound to LookUpStock. A [LoginView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.loginview.aspx) control is used to specify which buttons are shown. In an overridden [Render](http://msdn.microsoft.com/en-us/library/20b8wk8d.aspx) event for the page, the buttons are registered for validation. If the user is not authenticated, the button that initiates the callback for LookUpSale is not registered and the callback will fail if it is attempted.

The code-behind page adds client-side script to the page through 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 named 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** |
| If your Web page and client callbacks deal with the display of sensitive data or operations that insert, update, or delete data, it is recommended that you validate callbacks to ensure that the intended user interface element is executing the callback. |