Passing data from content page to master page-

In the example below,

1. We have a textbox in the master page. The ID of the textbox is txtBoxOnMasterPage

2. On the Content page, we have another textbox and a Button. The ID of this textbox is TextBox1 and the ID of the button is Button1

3. When you type some text in the textbox on the content page and when you hit Set Text button, we want to display the text entered in the textbox on the content page in the textbox that is present in the master page.

Here are the steps:

Step 1: Place a TextBox control with ID="txtBoxOnMasterPage" on the master page. Here is the HTML

Content Page Data:<br />

<asp:TextBox ID="txtBoxOnMasterPage" runat="server"></asp:TextBox>

Step 2: In the code-behind file of the master page, include a public property that returns the textbox control. Content pages will use this property to set the text of the textbox on the master page.

// The property returns a TextBox

public TextBox TextBoxOnMasterPage

{

get

{

// Return the textbox on the master page

return this.txtBoxOnMasterPage;

}

}

Step 3: Include a TextBox and a Button control on the content page. Here is the HTML.

<b>Type the text in the textbox that you want to display

in Master Page TextBox and click Set Text Button</b>

<asp:TextBox ID="TextBox1" runat="server">

</asp:TextBox>

<asp:Button ID="Button1" runat="server" Text="Set Text"

onclick="Button1\_Click" />

Step 4: Copy and paste the following Button1\_Click() event handler method in the content page

protected void Button1\_Click(object sender, EventArgs e)

{

// Retrieve the master page associated with this content page using

// Master property and typecast to the actual master page and then

// reference the property

((Site)Master).TextBoxOnMasterPage.Text = TextBox1.Text;

}

If you want Master property to return a strongly typed reference of the associated master page, include the following MasterType directive on the content page.

<%@ MasterType VirtualPath="~/Site.Master" %>

Once you have included the above MasterType directive on the content page, you will now have a strongly typed reference of the master page and can access it's properties without having to typecast.

protected void Button1\_Click(object sender, EventArgs e)

{

Master.TextBoxOnMasterPage.Text = TextBox1.Text;

}

To retrieve a master page associated with a content page, we can use either

1. Master propery

2. Page.Master property

Page.Master property always returns an object of type MasterPage and we need to explicitly typecast it to the actual master page, if we need to access it's properties and methods.

Where as Master property returns a strongly typed reference of the actual master page if the content page has MasterType directive specified. Otherwise Master property works in the same way as Page.Master property.

Passing data from master page to content page

In the example below,

1. We have a search textbox and a search button in the master page.

2. On the Content page, we have a GridView control that displays students from a database table.

3. When we type some text in the search textbox on the master page and when we hit "Search" button, we want to display all the students whose name contains the search term entered in the search textbox.

Here are the steps

Step 1: Include a panel, textbox and a button control on the master page.

<asp:Panel ID="panelSearch" runat="server">

<b>Search : </b>

<asp:TextBox ID="txtSerach" runat="server"></asp:TextBox>

<asp:Button ID="btnSearch" runat="server" Text="Search"/>

</asp:Panel>

Step 2: Include the following 3 properties in the code-behind file of the master page. We will use these properties in the content page.

public Panel SearchPanel

{

get

{

return panelSearch;

}

}

public string SearchTerm

{

get

{

return txtSerach.Text;

}

}

public Button SearchButton

{

get

{

return btnSearch;

}

}

Step 3: We will be using table tblStudents. Here is the SQL Script to create and populate this table with sample data.

Create Table tblStudents

(

ID int Identity Primary Key,

Name nvarchar(50),

Gender nvarchar(20),

TotalMarks int,

)

Insert into tblStudents values ('Mark Hastings','Male',900)

Insert into tblStudents values ('Pam Nicholas','Female',760)

Insert into tblStudents values ('John Stenson','Male',980)

Insert into tblStudents values ('Ram Gerald','Male',990)

Insert into tblStudents values ('Ron Simpson','Male',440)

Insert into tblStudents values ('Able Wicht','Male',320)

Insert into tblStudents values ('Steve Thompson','Male',983)

Insert into tblStudents values ('James Bynes','Male',720)

Insert into tblStudents values ('Mary Ward','Female',870)

Insert into tblStudents values ('Nick Niron','Male',680)

Step 4: Search stored procedure below, returns all students whose name contains @SearchTerm parameter.

Create Proc spSearch

@SearchTerm nvarchar(50)

as

Begin

Select \* from tblStudents where Name like '%' + @SearchTerm + '%'

End

Step 5: Add a content page, and include a GridView control to display the students.

<asp:GridView ID="GridView1" runat="server">

</asp:GridView>

Include the following MasterType directive on the content page

<%@ MasterType VirtualPath="~/Site.Master" %>

Step 6: Copy and paste the following code in the code-behind file of the content page.

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

{

Master.SearchButton.Click += new EventHandler(SearchButton\_Click);

}

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

{

GetData(null);

}

protected void SearchButton\_Click(object sender, EventArgs e)

{

GetData(Master.SearchTerm);

}

private void GetData(string searchTerm)

{

string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

using (SqlConnection con = new SqlConnection(cs))

{

SqlCommand cmd = new SqlCommand("spSearch", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter searchParameter = new SqlParameter("@SearchTerm", searchTerm ?? string.Empty);

cmd.Parameters.Add(searchParameter);

con.Open();

GridView1.DataSource = cmd.ExecuteReader();

GridView1.DataBind();

}

}

Note: Please include the following USING declarations

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

At this point, run the application and enter a search term in the search textbox on the master page, and click Search button. Notice that the matching students are displayed in the gridview control on the content page.

Let us look at another example of implementing search functionality on another content page. This time we will use table tblEmployee. Here is the sql script to create and populate this table.

Create Table tblEmployee

(

Id int Identity Primary Key,

Name nvarchar(50),

Email nvarchar(50),

Age int,

Gender nvarchar(50),

HireDate date,

)

Insert into tblEmployee values

('Sara Nan','Sara.Nan@test.com',35,'Female','04/04/1999')

Insert into tblEmployee values

('James Histo','James.Histo@test.com',33,'Male','12/07/2008')

Insert into tblEmployee values

('Mary Jane','Mary.Jane@test.com',28,'Female','11/11/2005')

Insert into tblEmployee values

('Paul Sensit','Paul.Sensit@test.com',29,'Male','10/10/2007')

Insert into tblEmployee values

('Todd Grove','todd.grove@test.com',31,'Male','11/11/2008')

Search stored procedure below, returns all employees whose name contains @SearchTerm parameter.

Create Proc spSearchEmployees

@SearchTerm nvarchar(50)

as

Begin

Select \* from tblEmployee where Name like '%' + @SearchTerm + '%'

End

Add a content page, and include a GridView control to display the employees.

<asp:GridView ID="GridView1" runat="server">

</asp:GridView>

Include the following MasterType directive on the content page

<%@ MasterType VirtualPath="~/Site.Master" %>

Copy and paste the following code in the code-behind file of the content page.

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

{

Master.SearchButton.Click += new EventHandler(SearchButton\_Click);

}

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

{

GetData(null);

}

protected void SearchButton\_Click(object sender, EventArgs e)

{

GetData(Master.SearchTerm);

}

private void GetData(string searchTerm)

{

string cs = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;

using (SqlConnection con = new SqlConnection(cs))

{

SqlCommand cmd = new SqlCommand("spSearchEmployees", con);

cmd.CommandType = CommandType.StoredProcedure;

SqlParameter searchParameter = new SqlParameter("@SearchTerm", searchTerm ?? string.Empty);

cmd.Parameters.Add(searchParameter);

con.Open();

GridView1.DataSource = cmd.ExecuteReader();

GridView1.DataBind();

}

}

Note: Please include the following USING declarations

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

At this point, run the application and enter a search term in the search textbox on the master page, and click Search button. Notice that the matching employees are displayed in the gridview control on the content page.

Let's say on a specific content page, we don't want this search functionality. If that's the case, we can very easily hide the search interface on the master page. Here are the steps.

Step 1: Add a new content page

Step 2: Include the following MasterType directive on the content page

<%@ MasterType VirtualPath="~/Site.Master" %>

Step 3: Copy and paste the following code in the code-behind file of the content page

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

{

Master.SearchPanel.Visible = false;

}

Postback-

The IsPostBack property can be used to determine if the page is submitted to itself.

When a form is submitted back to the same page that contains it, it's called a post back.

ASP.NET provides a property called IsPostBack that is TRUE when the page is being loaded as a result of a post back, and is FALSE otherwise.

A PostBack is any request for a page that is not the first request.

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />

<asp:Calendar ID="Calendar1" runat="server"></asp:Calendar><br />

<asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1\_Click" /><br />

<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

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

{

if (IsPostBack)

{

TextBox1.Text = "some value";

Calendar1.SelectedDate = DateTime.Now.Date.AddDays(2);

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Label1.Text = TextBox1.Text + " - " + Calendar1.SelectedDate.ToShortDateString();

}

----------------------------------

string result=string.format("thankyou,{0} for business,your ssn no. is:{1:000-00-0000}",name.text,ss);

--------------------------------------------------------------------------

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<asp:DropDownList ID="DropDownList2" runat="server"></asp:DropDownList>

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

...........

if (Page.IsPostBack == false) {

DropDownList2.Items.Add(new ListItem("Agra"));

DropDownList2.Items.Add(new ListItem("Bangalore"));

DropDownList2.Items.Add(new ListItem("Delhi"));

DropDownList2.Items.Add(new ListItem("Chandigarh"));

DropDownList2.Items.Add(new ListItem("Kolkatta"));

DropDownList2.Items.Add(new ListItem("Ahmedaba"));

DropDownList2.Items.Add(new ListItem("Mumbai"));

}

Another way to solve, this problem is to simply disable the ViewState of the DropDownlist control.

send data from one page to another--

Different techniques to move data from one webform to another

1. Cross Page Postback:

2. Context.Handler object -

In general, the members of one Web form are unavailable from a subsequently displayed Web form.

However, when navigating between Web forms using the Transfer or Execute method, data can be retrieve from the previous Web form using Context.Handler object.

Points to remember Context.Handler object

1. Context.Handler returns WebForm1 as the previous page, only the first time when you land on WebForm2 from WebForm1.

If there is a button on WebForm2, and if you click the button, the page will postback, and Context.Handler will return WebForm2 instead of WebForm1.

2. For the Context.Handler to return WebForm1 as the previous page, you should have landed on WebForm2, using Server.Transfer or Server.Execute method from WebForm1.

3. The control values from the previous page, can be accessed using FindControl() method or using public properties.

The problem with FindControl() method is that, if you mis-spell the ControlID, we could get a runtime NullRefernceException.

Using public properties, instead of FindControl() method, can eliminate runtime NullRefernceExceptions.

WebForm1.aspx HTML source:

<div style="font-family: Arial">

<table>

<tr>

<td colspan="2">

<h1>

This is WebForm1</h1>

</td>

</tr>

<tr>

<td>

<b>Name</b>

</td>

<td>

:<asp:TextBox ID="txtName" runat="server">

</asp:TextBox>

</td>

</tr>

<tr>

<td>

<b>Email</b>

</td>

<td>

:<asp:TextBox ID="txtEmail" runat="server">

</asp:TextBox>

</td>

</tr>

<tr>

<td colspan="2">

<asp:Button ID="Button1" runat="server"

Text="Go to WebForm2" onclick="Button1\_Click" />

</td>

</tr>

</table>

</div>

WebForm1.aspx.cs Code:

protected void Button1\_Click(object sender, EventArgs e)

{

Server.Transfer("~/WebForm2.aspx");

}

public string Name

{

get

{

return txtName.Text;

}

}

public string Email

{

get

{

return txtEmail.Text;

}

}

WebForm2.aspx HTML source:

<div style="font-family: Arial">

<table>

<tr>

<td colspan="2">

<h1>This is WebForm2</h1>

</td>

</tr>

<tr>

<td>

<b>Name</b>

</td>

<td>

:<asp:Label ID="lblName" runat="server">

</asp:Label>

</td>

</tr>

<tr>

<td>

<b>Email</b>

</td>

<td>

:<asp:Label ID="lblEmail" runat="server">

</asp:Label>

</td>

</tr>

</table>

</div>

WebForm2.aspx.cs Code:

//On postback Context.Handler returns WebForm2

if (!IsPostBack)

{

Page lastpage = (Page)Context.Handler;

if (lastpage is WebForm1)

{

//Use FindControl() if public properties does not exist on the

//previous page(WebForm1). FindControl() may cause

//NullRefernceExceptions due to mis-spelled conrol Id's

//lblName.Text = ((TextBox)lastpage.FindControl("txtName")).Text;

//lblEmail.Text = ((TextBox)lastpage.FindControl("txtEmail")).Text;

//Using public properties can eliminate NullRefernceExceptions

lblName.Text = ((WebForm1)lastpage).Name;

lblEmail.Text = ((WebForm1)lastpage).Email;

}

}

-----------------------------------------------------------------------------------------------------------

Cross page post back-

Cross page posting allows to post one page to another page. By default, when you click a button, the webform posts to itself.

If you want to post to another webform on a button click, set the PostBackUrl of the button, to the page that you want to post to.

Ex:1-

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

<div>

User ID :

<br />

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<br />

<br />

Password :

<br />

<asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>

<br />

<br />

<asp:Button ID="Button1" runat="server" Text="Submit" PostBackUrl="~/result.aspx" />

</div>

</form>

...............

in case of dropdown or list item it is better approach

TextBox tx1 = (TextBox)PreviousPage.FindControl("TextBox1");

TextBox tx2 = (TextBox)PreviousPage.FindControl("TextBox2");

Response.Write("ID : " + tx1.Text + "<br/>");

Response.Write("Password : " + tx2.Text + "<br/>");

-----------------------------------------------------------------------------

default.aspx

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

<div id="div1" runat="server">

Background Color&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:DropDownList ID="DropDownList1" runat="server" Height="16px" Width="141px"></asp:DropDownList>

<br />

<br />

TextBox Fore Color&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:DropDownList ID="DropDownList2" runat="server" Height="16px" Width="142px"></asp:DropDownList>

<br />

<br />

TextBox Back Color&nbsp;&nbsp;&nbsp;&nbsp;

<asp:DropDownList ID="DropDownList3" runat="server" Height="17px" Width="136px"></asp:DropDownList>

<br />

<br />

Button Back Color&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<asp:DropDownList ID="DropDownList4" runat="server" Height="17px" Width="133px"></asp:DropDownList>

<br />

<br />

<br />

<asp:Button ID="Button1" runat="server" PostBackUrl="~/result.aspx" Text="Submit" Width="105px" />

<br />

</div>

.........

default.aspx.cs

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

{

string[] colors = Enum.GetNames(typeof(KnownColor));

foreach (string s in colors)

DropDownList1.Items.Add(s);

foreach (string s in colors) DropDownList2.Items.Add(s);

foreach (string s in colors) DropDownList3.Items.Add(s);

foreach (string s in colors) DropDownList4.Items.Add(s);

}

............

result.aspx

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>

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

......

result.aspx.cs

if (PreviousPage != null)

{

DropDownList d1 = (DropDownList)(PreviousPage.FindControl("DropDownList1"));

DropDownList d2 = (DropDownList)(PreviousPage.FindControl("DropDownList2"));

DropDownList d3 = (DropDownList)(PreviousPage.FindControl("DropDownList3"));

DropDownList d4 = (DropDownList)(PreviousPage.FindControl("DropDownList4"));

div1.Style.Add(HtmlTextWriterStyle.BackgroundColor, d1.SelectedValue);

TextBox1.ForeColor = System.Drawing.Color.FromName(d2.SelectedValue);

TextBox2.ForeColor = System.Drawing.Color.FromName(d2.SelectedValue);

TextBox1.BackColor = System.Drawing.Color.FromName(d3.SelectedValue);

TextBox2.BackColor = System.Drawing.Color.FromName(d3.SelectedValue);

Button1.BackColor = System.Drawing.Color.FromName(d4.SelectedValue);

}

Ex:2-

WebForm1.aspx code: Notice that, the PostBackUrl property of the button with ID=btnCrossPagePostback is set to WebForm2.aspx.

When this button is clicked WebForm1.aspx gets posted to WebForm2.aspx.

<div style="font-family: Arial">

<table>

<tr>

<td colspan="2">

<h1>

This is WebForm1</h1>

</td>

</tr>

<tr>

<td>

<b>Name</b>

</td>

<td>

:<asp:TextBox ID="txtName" runat="server">

</asp:TextBox>

</td>

</tr>

<tr>

<td>

<b>Email</b>

</td>

<td>

:<asp:TextBox ID="txtEmail" runat="server">

</asp:TextBox>

</td>

</tr>

<tr>

<td colspan="2">

<asp:Button ID="btnCrossPagePostback" runat="server"

Text="Cross Page Postback - WebForm2"

Width="250px" PostBackUrl="~/WebForm2.aspx"/>

</td>

</tr>

<tr>

<td colspan="2">

<asp:Button ID="Button1" runat="server" onclick="Button1\_Click"

Text="Server.Transfer - WebForm2" Width="250px" />

</td>

</tr>

</table>

</div>

WebForm1.aspx.cs code:

protected void Button1\_Click(object sender, EventArgs e)

{

Server.Transfer("~/WebForm2.aspx");

}

WebForm2.aspx code:

<div style="font-family: Arial">

<table>

<tr>

<td colspan="2">

<h1>This is WebForm2</h1>

</td>

</tr>

<tr>

<td>

<b>Name</b>

</td>

<td>

:<asp:Label ID="lblName" runat="server">

</asp:Label>

</td>

</tr>

<tr>

<td>

<b>Email</b>

</td>

<td>

:<asp:Label ID="lblEmail" runat="server">

</asp:Label>

</td>

</tr>

<tr>

<td colspan="2">

<asp:Label ID="lblStatus" runat="server"

ForeColor="Red" Font-Bold="true"></asp:Label>

</td>

</tr>

</table>

</div>

WebForm2.aspx.cs code: Page.IsCrossPagePostBack Property is used to indicate whether the page is involved in a cross-page postback.

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

{

Page previousPage = Page.PreviousPage;

if (previousPage != null && previousPage.IsCrossPagePostBack)

{

lblName.Text = ((TextBox)previousPage.FindControl("txtName")).Text;

lblEmail.Text = ((TextBox)previousPage.FindControl("txtEmail")).Text;

}

else

{

lblStatus.Text = "Landed on this page using a technique other than cross page post back";

}

}