فصل چهارم :

ایجاد و استفاده از Custom Controls

(قسمت اول)

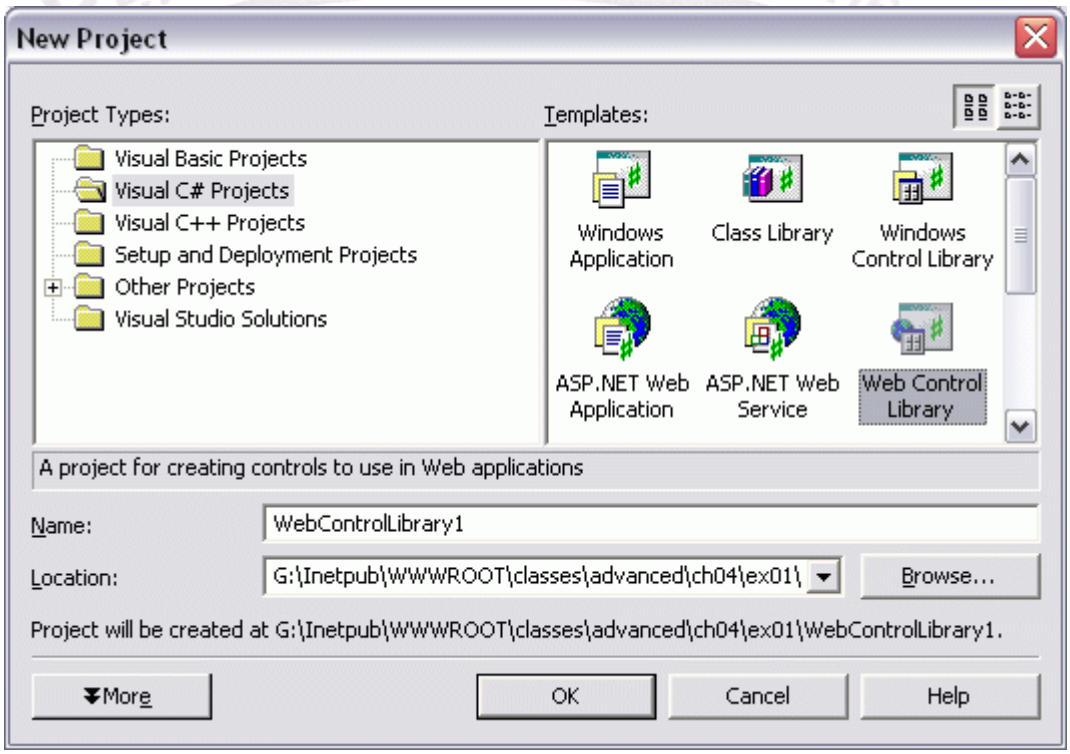
مقدمه :

یکی از مفیدترین جنبه های ASP.NET این است که می توان از کنترل های موجود ، کنترل های جدیدی را خلق کرد. بدین ترتیب به شدت از حجم کدهای تکراری در برنامه های مختلف کاسته می شود. برای مثال یک DropDownList استاندارد را در نظر بگیرید که باید در چندین صفحه قرار داده شود و بدیهی است که در تمام صفحات نیاز به اتصال به دیتابیس و نمایش اطلاعات وجود دارد. بجای تکرار این مساله در صفحات متوالی می توان یکبار برای همیشه کنترلی را خلق کرد که کار دیتابایندینگ را انجام دهد و بارها در برنامه های مختلف از آن استفاده نمود. در این فصل درباره ایجاد اینگونه کنترل های ویژه بحث خواهد گردید.

خلق کنترل های سفارشی :

۶ مرحله برای ایجاد یک کنترل سفارشی در برنامه های وب وجود دارد:

۱- ایجاد یک پروژه ی جدید حاوی یک کنترل سفارشی
۲- اضافه کردن یک WEB application به پروژه ی موجود وتنظیم آن به عنوان پروژه ی startup .
از این برنامه برای تست کنترل سفارشی در حین برنامه نویسی آن استفاده می شود.
۳- در وب اپلیکیشن باید ریفرنسی به کنترل سفارشی ایجاد شود و در سورس HTML آن نیز باید از کلمه ی Register برای استفاده از کنترل سفارشی بر روی وب فرم استفاده کرد.
۴- ظاهربصری کنترل سفارشی، با اضافه کردن کنترل های موجود به متد CreateChildControls ، ایجاد می گردد.
۵- اضافه کردن متدها ، خواص و رخدادهایی که یک کنترل سفارشی مهیا می کند.
۶- ساخت و آزمایش کردن کنترل سفارشی.



شکل ۱- استفاده از Template ساخت کنترل های سفارشی در VS.NET .

مثال ۱ :

ایجاد یک پروژه ی ساخت کنترل سفارشی :

یک کنترل سفارشی کلاسی است که به صورت یک assembly ساخته می شود (یک فایل dll.) . این کنترل سفارشی عمده ی رفتار خودش را از کلاس WebForm به ارث خواهد برد.

ساده ترین راه برای ایجاد یک کنترل سفارشی استفاده از Template فراهم شده توسط Web Control library ویژوال استودیو دات نت است (شکل-۱).

پس از انتخاب نام مناسب و کلیک کردن بر روی دکمه ی Ok ، اسکلت اولیه ی ساخت یک کنترل سفارشی ایجاد می شود که در آن یک خاصیت جدید به نام Text به صورت زیر ایجاد شده است:

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;

namespace WebControlLibrary1
{
	/// <summary>
	/// Summary description for WebCustomControl1.
	/// </summary>
	[DefaultProperty("Text"),
		ToolboxData("<{0}:WebCustomControl1
runat=server></{0}:WebCustomControl1>")]
	public class WebCustomControl1 : System.Web.UI.WebControls.WebControl
	{
		private string text;

		[Bindable(true),
			Category("Appearance"),
			DefaultValue("")]
		public string Text
		{
			get
			{
				return text;
			}

			set
			{
				text = value;
			}
		}
```

```
            /// <summary>
            /// Render this control to the output parameter specified.
            /// </summary>
            /// <param name="output"> The HTML writer to write out to </param>
            protected override void Render(HtmlTextWriter output)
            {
                  output.Write(Text);
            }
      }
}
```

کدی که نمایش داده شد حاوی المانهایی است که تابحال با آن آشنایی نداشته اید و در جدول زیر بررسی شده اند:

جدول ۱- Template یک کنترل سفارشی

قسمت	نام	توضیح
`[DefaultProperty("Text"), ToolboxData( "<{0}:WebCustomControl1 runat=server> </{0}:WebCustomControl1>")]`	ویژگی کلاس	تنظیمات حالت طراحی کنترل را مشخص می کند. این مقادیر مواردی را که در پنجره ی خواص ویژوال استودیو هنگام استفاده از کنترل ظاهر می شوند ، مشخص می کند.
`:System.Web.UI.WebControls.WebControl`	کلاس پایه	کنترل سفارشی ، از یکی از کنترل های پایه ارث خواهد برد.
`[Bindable(true), Category("Appearance"), DefaultValue("")]`	ویژگی های خواص	تنظیمات حالت طراحی یک خاصیت را مشخص می کند.
`public string Text`	تعریف	خاصیت را در زمان اجرای برنامه پدید می آورد.
`protected override void Render(HtmlTextWriter output)`	متد Render	متد Render کنترل سفارشی را نمایش می دهد.

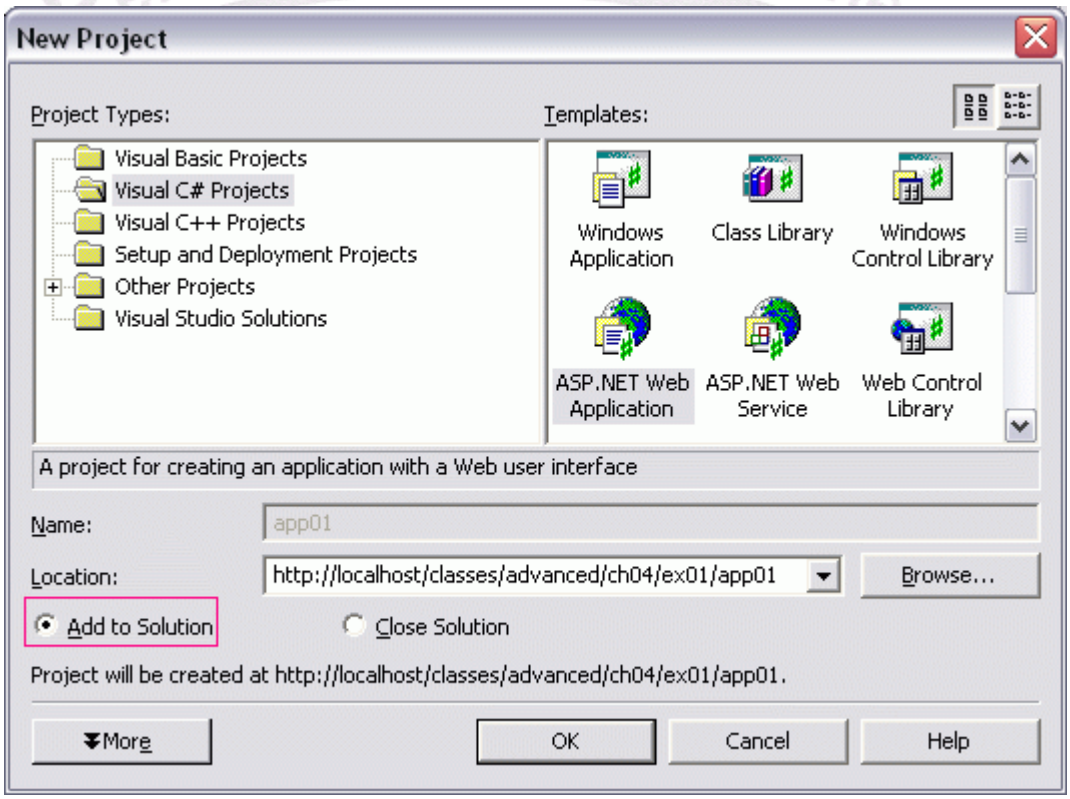
ایجاد یک پروژه برای آزمایش کنترل خلق شده:

چون کنترل سفارشی به صورت یک class ساخته می شود بنابراین امکان استفاده از آن به صورت مستقل وجود ندارد و حتما باید در برنامه ای دیگر مورد استفاده قرار گیرد. برای اجرا و debug کردن کنترل سفارشی نیاز است تا پروژه ی دومی نیز به پروژه ی جاری اضافه شود.

بدین منظور به صورت زیر عمل می شود:

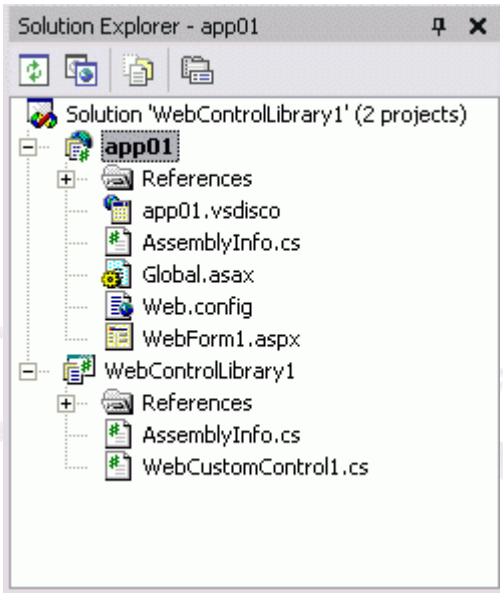
۱- در حالیکه پروژه ی کنترل سفارشی باز است از منوی فایل یک پروژه به برنامه اضافه کنید. در این حالت دیالوگ باکس پروژه ی جدید باز می شود.

۲- یک web application را انتخاب نموده ، نام پروژه را وارد کرده و سپس روی Ok کلیک کنید. در این حالت یک پروژه ی جدید به پروژه ی جاری اضافه می شود.



شکل ۲- هنگام اضافه کردن یک پروژه به پروژه ی جاری ، گزینه ی Add to solution را باید برگزید.

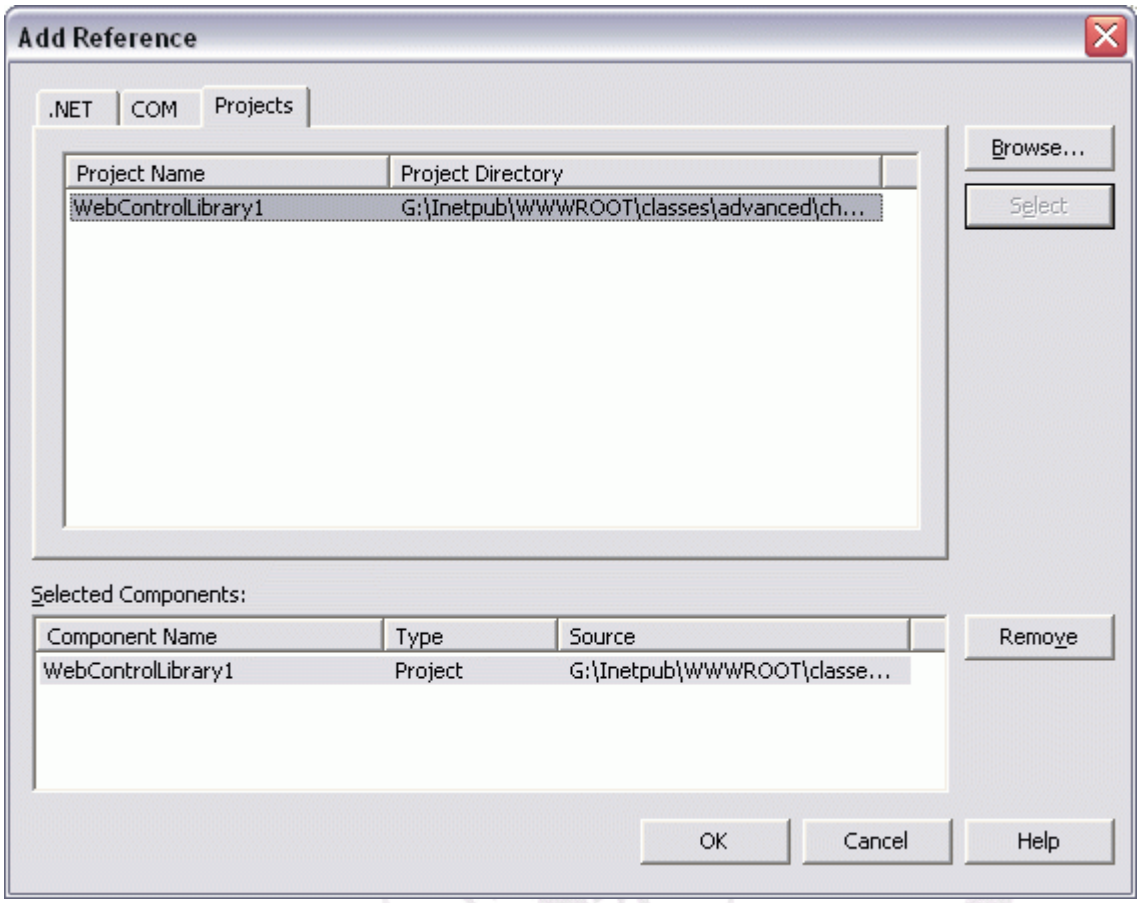
۳- در solution explorer ، روی پروژه ی web application کلیک راست کـرده و از منـوی ظـاهر شده set as start up project را انتخاب نمایید. در این حالت نام پروژه bold خواهد شد (شکل -۳).



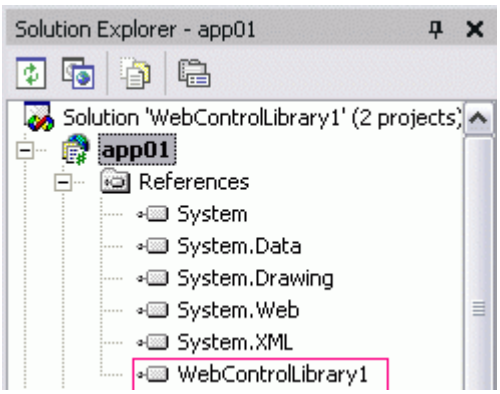
شکل ۳- انتخاب وب اپلیکیشن بعنوان پروژه ی پیش فرض.

۴- روی web application در solution explorer کلیک راست کنید و از منوی ظاهر شده گزینه ی add reference را برگزینید (شکل-۴).

۵- برگه ایی به نام properties را انتخاب نموده ، select را فشرده و ریفرنسی از کنترل سفارشی به برنامه اضافه نمایید . سپس روی Ok کلیک نمایید. VS.NET ریفرنسی را به این کنترل اضافه می کند (شکل-۵).



شکل ۴– دیالوگ باکس اضافه کردن ریفرنس به کنترل سفارشی ایجاد شده.



شکل ۵– پس از اضافه کردن مرجع به کنترل ، این مطلب در Solution explorer مشخص می گردد.

با انجام این کار ، VS.NET یک کپی از اسمبلی کنترل سفارشی (dll.) را درون دایرکتوری bin وب اپلیکیشن کپی می کند و هر تغییری هم در کنترل سفارشی در طول برنامه انجام شود بلافاصله به این دایرکتوری نیز منعکس خواهد گردید.

این نکته را باید بخاطر داشته باشید تنها تغییراتی در وب اپلیکیشن منعکس می شوند که حاصل از کامپایل کردن مجدد کنترل سفارشی باشند. بنابراین پس از انجام تغییرات در کنترل سفارشی ، کامپایل کردن آنرا فراموش نکنید.

اضافه کردن کنترل سفارشی به پروژه ی وب اپلیکیشن آزمایشی :

برای آزمایش کردن و debug نمودن کنترل در حین توسعه ی آن ، باید یک نمونه از آن به وب فرم برنامه اضافه شود. بدین منظور به صورت زیر عمل می شود:

۱- در حالت طراحی WebForm روی دکمه ی HTML کلیک کنید تا سورس صفحه ظاهر شود.

۲- خط رجیستر را به سورس اضافه کنید (شکل- ۶)

۳- یک نمونه از کنترل را در وب فرم با دستور زبان HTML به صورت زیر پدید آورید (شکل- ۶)

معانی ویژگی های تگ Register در جدول زیر بیان شده اند.

جدول۲- ویژگی های تگ Register :

ویژگی	معنا
TagPerfix	گروهی را که کنترل به آن متعلق است مشخص می کند(یک نام دلخواه). در اینجا برای مثال Custom انتخاب شده است.
NameSpace	نام پروژه و فضای نامی می باشد که حاوی کنترل است.
Assembly	نام اسمبلی (dll.) ایی است که حاوی کنترل سفارشی است.

برخلاف user controls ، کنترل های سفارشی همانطور که در شکل-۷ نیز مشخص شده انـد کـاملا در محیط طراحی نمایش داده شده و پنجره ی خواص نیز برای آنها مهیا است.

art Page | WebCustomControl1.cs | **WebForm1.aspx** | WebForm1.aspx.cs |

ent Objects & Events | (No Events)

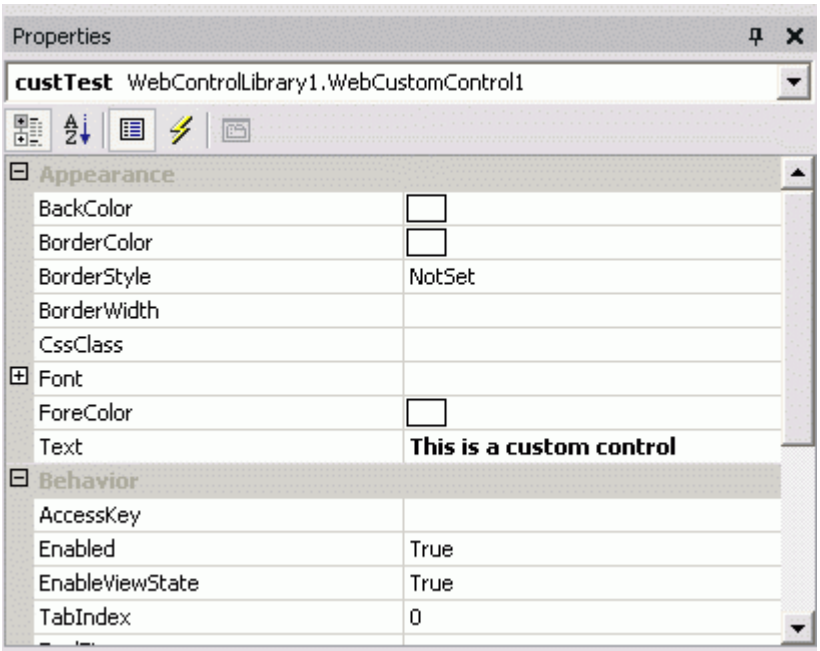
```
<%@ Register TagPrefix="Custom" Namespace="WebControlLibrary1"
    Assembly="WebControlLibrary1" %>

<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
    AutoEventWireup="false" Inherits="app01.WebForm1" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>WebForm1</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio 7.0">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema"
              content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">

            <Custom:WebCustomControl1 id="custTest"
            runat="server" Text="This is a custom control" />

        </form>
    </body>
</HTML>
```

شکل ۶– نحوه ی تعریف و بکار گیری یک کنترل سفارشی بر روی فرم وب . بعد از قرار گیری روی فـرم به راحتی می توان مکان آنرا در محیط ویژوال تنظیم کرد.

Start Page | WebCustomControl1.cs | **WebForm1.aspx**

This is a custom control

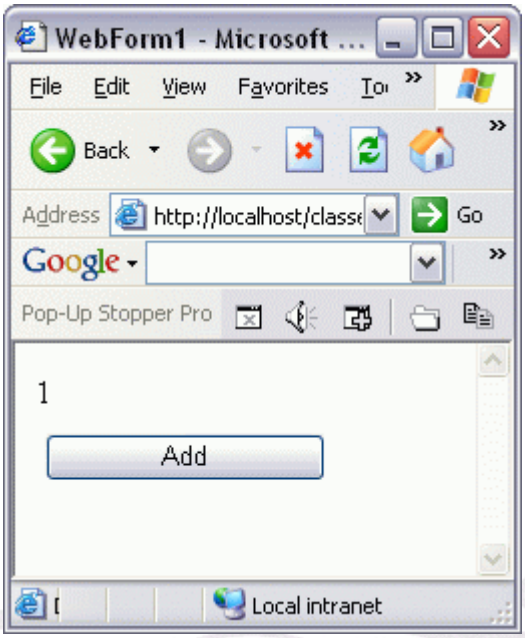
شکل ۷- شکل نهایی کنترل سفارشی ایجاد شده در محیط طراحی.



شکل ۸- خواص منتسب به کنترل سفارشی ایجاد شده.

برای استفاده از آن نیز همانند سایر کنترل های وب عمل می شود. برای مثال یک دکمه روی صفحه قرار دهید و کد زیر رادر آن بنویسید:

```
private void btnAdd_Click(object sender, System.EventArgs e)
{
      // Convert the string to a number.
      int intText = 0;
      try
      {
            intText = int.Parse(custTest.Text);
      }
      catch
      {
            intText = 0;
      }
      // Increment the number.
      intText ++;
      // Display result in control
      custTest.Text = intText.ToString();
}
```



شکل ۹- اجرای برنامه ی حاوی کنترل سفارشی.

اگر چند بار روی دکمه ی Add کلیک نمایید هرگز عدد از یک بیشتر نخواهد شد! یک کنترل سفارشی مقادیر را به صورت پیش فرض ، بین نمایش متوالی صفحات حفظ نمی کند و باید برای اینکار کد نویسی کرد که در ادامه (فصل های آتی) با این مطلب نیز آشنا خواهیم شد.

مثال ۲:

ایجاد کنترل های ترکیبی:

برای ایجاد ظاهر ویژوال یک کنترل سفارشی می توان کنترلهای موجود را به کلکسیون Controls در متد CreateChildControls اضافه کرد.

ایجاد یک MathBox :

یک پروژه ی جدید Web control library را به نام MathBox آغاز کنید.
MathBox از یک textBox ، Button و یک label تشکیل شده است. از کد زیر برای اضافه کردن آنها به کنترل سفارشی استفاده می نماییم.

```
public class MathBox : System.Web.UI.WebControls.WebControl
{
      TextBox txtMath = new TextBox();
      Button butSum = new Button();
      Label lblResult = new Label();


      protected override void CreateChildControls()
      {
            // Add the sub controls to this composite control.
            // Set the TextMode property and add textbox.
            txtMath.TextMode = TextBoxMode.MultiLine;
            Controls.Add(txtMath);
            // Start a new line
            Controls.Add(new LiteralControl("<br>"));
            // Set the Text property and add the Button control.
            butSum.Text = "Sum";
            Controls.Add(butSum);
            // Add Label and Literals to display result.
            Controls.Add(new LiteralControl(" &nbspResult: <b>"));
            Controls.Add(lblResult);
            Controls.Add(new LiteralControl("</b>"));
      }
 }
```

سپس برای استفاده از آن یک پروژه ی WebApplication جدید به پروژه ی جاری همانند مثال قبل اضافه نموده ، ریفرنس لازم را ایجاد کنید ، آنرا بعنوان پروژه ی آغاز شونده انتخاب نمایید و سپس با استفاده از Register آنرا در صفحه ثبت نموده و با استفاده از تگ زیر آنرا به سورس صفحه ی وب فرم اضافه کنید (شکل- ۱۰).

art Page | MathBox.cs | **WebForm1.aspx**

ent Objects & Events | (No Events)

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
    AutoEventWireup="false" Inherits="WebApplication1.WebForm1" %>

<%@ Register TagPrefix="Custom" Namespace="WebControlLibrary1"
     Assembly="WebControlLibrary1" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>WebForm1</title>
        <meta name="GENERATOR" Content="Microsoft Visual Studio 7.0">
        <meta name="CODE_LANGUAGE" Content="C#">
        <meta name="vs_defaultClientScript" content="JavaScript">
        <meta name="vs_targetSchema"
         content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">

            <Custom:MathBox id="mathTest" runat="server" />

        </form>
    </body>
</HTML>
```

شکل ۱۰– چگونگی اضافه کردن تگ های لازم برای استفاده از کنترل سفارشی.

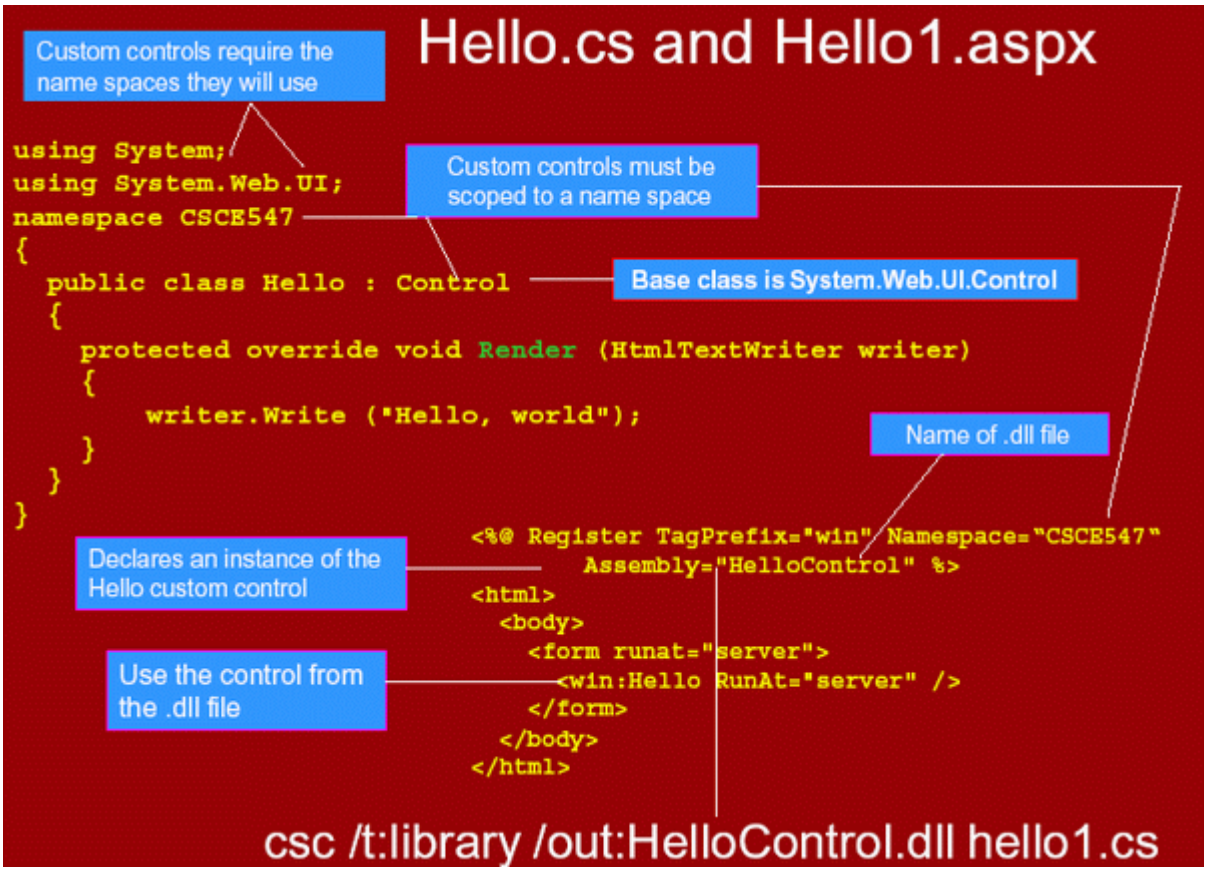
در این حالت اگر شما آنرا به وب فرم در حالت طراحی اضافه کنید همانند یک مستطیل سبز کوچک به نظر خواهد رسید.

در این حالت کنترل ترکیبی سفارشی ما در حالت طراحی ظاهری نخواهد داشت زیرا هیچگونه خواصی را برای آن در نظر نگرفته ایم. این مورد را در قسمت بعد مرتفع خواهیم کرد.

در یک کنترل ترکیبی نیازی به Override کردن متد Render نمی باشد و کد زیر کافی است:

```
protected override void Render(HtmlTextWriter output)
{
      // Render the control.
      base.Render(output);
}
```

نکته ای در مورد نحوه ی اضافه کردن تگ های کنترل به سورس HTML صفحه aspx :



شکل ۱۱- در این تصویر ارتباط بین قسمت های مختلف تگ رجیستر و کلاس حاوی کنترل را می توان به خوبی مشاهده نمود. رعایت نمودن این موارد الزامی است.

اضافه کردن متد و خواص:

MathBox ما لیستی از اعداد را دریافت کرده و توسط TextBox ، آنها را جمع می کند و سپس نتیجه را بر می گرداند. بنابراین نیاز به دو خاصیت Values که حاوی آرایه ای از اعداد ورودی و Result که

حاوی جمع این اعداد است می باشد. همچنین به متد Sum برای جمع زدن نیاز دارد. کد زیر اینکار را انجام می دهد.

```
// MathBox properties and methods.

[DefaultValue("0")]
public string Text
{
      get
      {
            // Make sure child controls exist.
            EnsureChildControls();
            // Return the text in the TextBox control.
            return txtMath.Text;
      }

      set
      {
            // Make sure child controls exist.
            EnsureChildControls();
            // Set the text in the TextBox control.
            txtMath.Text = value;
      }
}

char[] strSep = {'\r'};

public string[] Values
{
      get
      {
            EnsureChildControls();
            // Return an array of strings from the TextBox.
            return txtMath.Text.Split(strSep);
      }
      set
      {
            EnsureChildControls();
            // Set the text in the TextBox from an array.
            txtMath.Text = String.Join(" ", value);
      }
}

public string Result
{
      get
      {
            EnsureChildControls();
            // Return the result from the Label.
            return lblResult.Text;
      }
}
```

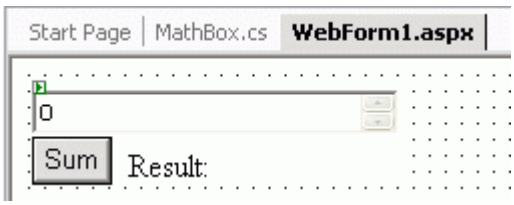
```
public void Sum()
{
      EnsureChildControls();
      // If there is text in the TextBox.
      if (txtMath.Text.Length != 0)
      {
            // Break the text into an array, line by line.
            string[] arrNums;
            arrNums = txtMath.Text.Split(strSep);
            double dblSum = 0;
            // Add each element in the array together.
            foreach (string strCount in arrNums)
            {
                  // Use error handling to ignore non-number entries.
                  try
                  {
                        dblSum += Convert.ToDouble(strCount);
                  }
                  catch
                  {
                  }
            }
            // Display the result in the label.
            lblResult.Text = dblSum.ToString();
      }
      else
            lblResult.Text = "0";
}
```

در کد بالا از EnsureChildControls استفاده گردیده است تا اطمینان حاصل شود که کنترل های روی صفحه insantitated شده اند یا خیر. این متد در ساخت کنترل های ترکیبی قبل از ارجاع بـه هـر کـدام از کنترل ها باید فراخوانی شود.

بعد از اضافه کردن این خواص و متدها و دوباره کامپایل کردن کنترل ؛ این کنترل ظاهر زمان طراحی را نیز بدست خواهد آورد. اضافه کردن خواص و تنظیم آنها نیز به صورت زیر است:

```
<body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
        <Custom:MathBox id="mathTest" runat="server" Text="0" />
    </form>
</body>
```

شکل ۱۲- چگونگی اضافه کردن خواص پیش فرض.



شکل ۱۳- شکل ظاهری کنترل در حالت طراحی.

لازم به ذکر است که به راحتی می توان مکان کنترل ترکیبی سفارشی را روی فرم تغییر داد و هیچگونه محدودیتی در این زمینه وجود ندارد.

اضافه کردن رخدادها به کنترل :

در کنترل MathBox هنگام کلیک روی دکمه ، هیچگونه عملیاتی رخ نمی دهد. برای پیاده سازی رخ داد کلیک به صورت زیر عمل می کنیم:

۱- رویداد گردان مربوط به Click را در متد CreateChildControls اضافه می نماییم.

۲- متدی را برای انجام فرامین مربوط به رخداد خلق می نماییم.

در کد زیر این دو مرحله به صورت ضخیم (Bold) نشان داده شده اند:

```
protected override void CreateChildControls()
{
      // Add the sub controls to this composite control.
      // Set the TextMode property and add textbox.
      txtMath.TextMode = TextBoxMode.MultiLine;
      Controls.Add(txtMath);
      // Start a new line
      Controls.Add(new LiteralControl("<br>"));
      // Set the Text property and add the Button control.
      butSum.Text = "Sum";
      Controls.Add(butSum);
      // Add Label and Literals to display result.
      Controls.Add(new LiteralControl(" &nbspResult: <b>"));
      Controls.Add(lblResult);
      Controls.Add(new LiteralControl("</b>"));

      // Add event handler.
      butSum.Click += new EventHandler(butSumClicked);

}
```

```
void butSumClicked(object sender, EventArgs e)
{
      // Call the Sum method.
      Sum();
}
```

Raising events :

علاوه بر رویداد گردان Click بر روی دکمه ی Sum ، شاید شما بخواهید رخدادی را که در صفحه ی وب فرم ایجاد می شود را نیز مدیریت کنید. برای این منظور این مراحل باید صورت گیرد:

۱- اضافه کردن تعریف رخداد به صورت public در کلاس کنترل سفارشی

۲- ایجاد رخداد درون کد کنترل سفارشی با استفاده از متد رخداد کنترل ها

کد زیر اینکار را نمایش می دهد.

```
// Declare the event.
public event EventHandler Click;

void butSumClicked(object sender, EventArgs e)
{
      // Call the Sum method.
      Sum();
      // Call the event method.
      OnClick(EventArgs.Empty);
}

protected virtual void OnClick(EventArgs e)
{
      // Raise the event.
      Click(this, e);
}
```

برای اینکه رخداد کلیک ، رخداد پیش فرض در کنترل باشد می توان تگ DefaultEvents را به تعریف کلاس به صورت زیر اضافه نمود. این رخداد پیش فرض دقیقا همان چیزی است که هنگامیکه در محیط VS.NET بر روی کنترل دوبار کلیک می نمایید، توسط آن ، متد رخدادش خلق می شود:

```
[DefaultEvent("Click")]
public class MathBox : System.Web.UI.WebControls.WebControl
```

برای استفاده از رخداد MathBox در محیط وب فرم روی کنترل در حالت طراحی دوبار کلیک کنید و سپس از کد زیر در آن استفاده نمایید.

```
private void mathTest_Click(object sender, System.EventArgs e)
{
      Response.Write(mathTest.Result.ToString());
}
```

مدیریت تغییر اندازه ی کنترل:

هنگامیکه اندازه ی کنترل را در زمان طراحی بزرگ و کوچک می نمایید ، رفتار Resizing را باید با کد نویسی کنترل نمود. به صورت زیر:

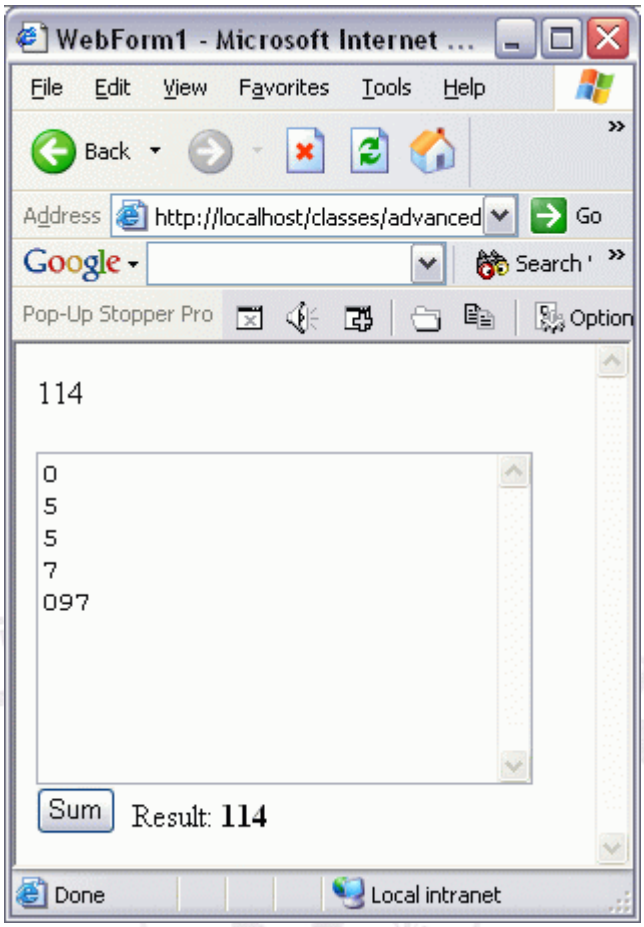
۱- Override کردن متد Render

۲- اضافه کردن کد برای تغییر اندازه ی کنترل ها

۳- استفاده از متد Render برای نمایش کنترل ها

کد زیر اینکار را انجام می دهد.

```
protected override void Render(HtmlTextWriter output)
{
      EnsureChildControls();
      // Resize text box to match control width.
      txtMath.Width = this.Width;
      // Resize text box to match control height.
      double dHeight = this.Height.Value - butSum.Height.Value;
      txtMath.Height = Unit.Parse(dHeight.ToString());
      // Render the control.
      base.Render(output);
}
```



شکل ۱۴- نمایی از اجرای برنامه حاوی کنترل MathBox .

کنترل های SuperClassed :

در قسمت های قبل نحوه ی ایجاد یک کنترل ترکیبی سفارشی را از کنترل های موجود آموختید. می توان از همان روش ها برای ایجاد کنترلی از یک کنترل موجود استفاده کرد. از آنجائیکه این کنترل سفارشی بر مبنای تنها یک کنترل است ، می توان به سادگی با ارث بری از آن کلاس ، کار کد نویسی را کوتاه کرد. گاهی از اوقات به این نوع کنترل ها ، کنترل های SuperClassed هم گفته می شود.

مثال ۳ :

یک پروژه ی جدید Web control library را آغاز کنید.

برای مثال کلاس SuperT زیر، یک کنترل سفارشی را بر مبنای کنترل سرور وب TextBox ایجاد می کند و متدی را برای Sort کردن محتویات آن ارایه می دهد.

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.ComponentModel;

namespace SuperText
{
	/// <summary>
	/// Summary description for SuperText.
	/// </summary>
	public class SuperT : System.Web.UI.WebControls.TextBox
	{

		public virtual void Sort()
		{
			// Create an array.
			string[] arrText;
			// Put the words in the text box into the array.
			char[] strSep = {' '};
			arrText = this.Text.Split(strSep);
			// Sort the array.
			Array.Sort(arrText);
			// Join the string and put it back in the text box.
			this.Text = String.Join(" ", arrText);
		}
	}
}
```

برای استفاده از آن هم می توان همانند قبل assembly آنرا رجیستر نمود و با استفاده از تگ زیر از آن استفاده نمود:

art Page | WebForm1.aspx | SuperText.cs | WebForm1.aspx.cs

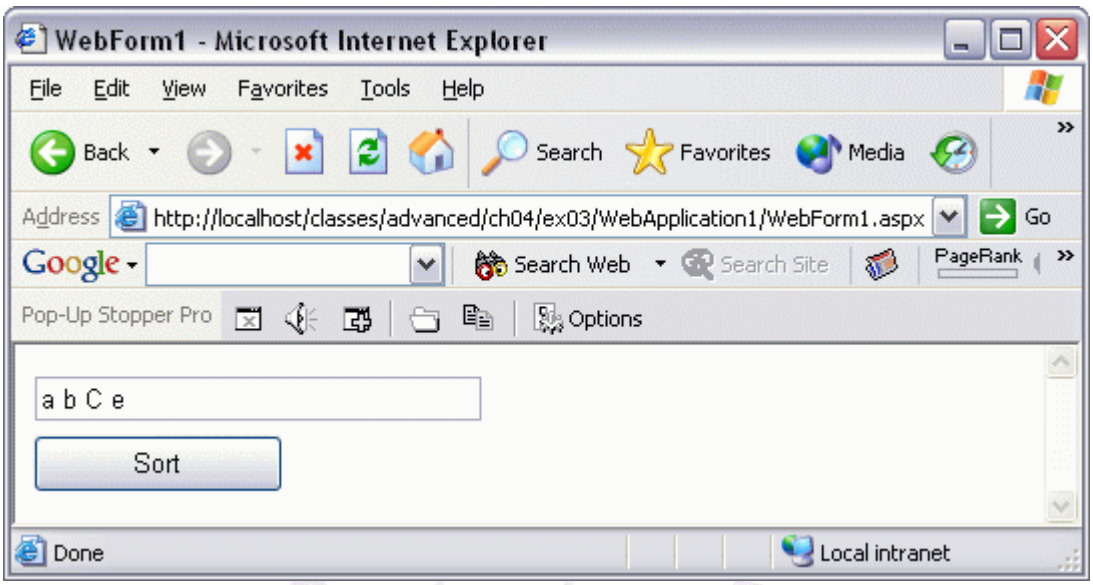
ient Objects & Events | (No Events)

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
    AutoEventWireup="false" Inherits="WebApplication1.WebForm1" %>
<%@ Register TagPrefix="Custom" Namespace="SuperText" Assembly="SuperText" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>WebForm1</title>
        <meta content="Microsoft Visual Studio 7.0" name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript" name="vs_defaultClientScript">
        <meta content="http://schemas.microsoft.com/intellisense/ie5"
        name="vs_targetSchema">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
            <CUSTOM:SUPERT id="superText" runat="server" Width="223px">
            </CUSTOM:SUPERT>
            <asp:button id="btnSort"
            style="Z-INDEX: 101; LEFT: 9px; POSITION: absolute; TOP: 45px"
            runat="server" Width="125px" Text="Sort" Height="29px"></asp:button>
        </form>
    </body>
</HTML>
```

شکل ۱۵- نحوه ی تعریف تگ های لازم برای استفاده از کنترل سفارشی خلق شده.

نحوه ی استفاده از متد Sort آن هم بعد از گذاشتن یک دکمه روی فرم به صورت زیر است:

```
private void btnSort_Click(object sender, System.EventArgs e)
{
   superText.Sort();
}
```



شکل ۱۶- نمایی از اجرای برنامه ی حاوی کنترل سفارشی مشتق شده از تکست باکس معمولی.

تمرین:

۱- یک کنترل سفارشی مشتق شده از کنترل وب تکست باکس ایجاد نمایید که قابلیت تبدیل حروف انگلیسی بزرگ را به حروف کوچک داشته باشد.

۲- کنترل سفارشی تکست باکسی را ایجاد نمایید که حاوی Required field validator نیز باشد.