فصل پنجم :

ایجاد و استفاده از Custom Controls

(قسمت دوم)

مقدمه :

بجای ایجاد ظاهر کنترل ها از طریق بکار گیری کنترل های موجود ، می توان کنترل های رندر شده ای پدید آورد که ظاهر خودشان را از طریق تگ های HTML و عناصر آن بوجود می آورند ، بنابراین دیگر یک سری خواص ، متدها و رخدادهای از پیش تعریف شده برای ما مهیا نیست و تمام آنها را باید خودمان ایجاد کنیم.

ایجاد یک کنترل رندر شده:

مراحل ایجاد یک کنترل رندر شده (البته تعدادی از آنها در فصل قبل نیز مرور شدند):

۱- ایجاد یک پروژه ی جدید حاوی custom control project

۲- اضافه کردن یک پروژه ی وب اپلیکیشن جدید به پروژه جاری و تنظیم آن بعنوان پروژه ی آغاز کننده ی برنامه ، از این برنامه برای تست کردن کنترل سفارشی در خلال توسعه استفاده می شود.

۳- اضافه کردن ریفرنس به کنترل سفارشی در وب اپلیکیشن و اضافه نمودن Register و المان ها و تگ های کنترل برای استفاده کردن از آن بر روی یک وب فرم.
۴- ایجاد ظاهر ویژوال کنترل سفارشی با تحریف کردن متد Render .
۵- نوشتن کد برای ارائه دادن خواص get و set .

ایجاد ظاهر ویژوال برای کنترل های رندر شده :

برای ایجاد ظاهر ویژوال کنترل های رندر شده می توان متد Render را تحریف کرد و از طریق HTMLTextWriter آنرا نمایش داد. HTMLTextWriter در جدول زیر بررسی شده است:

جدول ۱- متدهای HTMLTextWriter

متد	توضیح
AddAttribute	خواص HTML را به المان بعدی HTML برای رندر شدن اضافه می کند.
RenderBeginTag	تگ آغازین المان HTML را رندر می کند و از آن بوسیله ی متد WriteLine استفاده خواهد شد.
RenderEngTag	تگ پایانی المان HTML را رندر کرده و المان را نوشته و تکمیل می کند. تمام خواص رندر شده بعد از فراخوانی این تابع خلق می شوند.
Write	بلادرنگ یک رشته را می نویسد.
WriteAttribute	بلادرنگ یک خاصیت HTML را می نویسد.
WriteBeginTag	بلافاصله تگ آغازین المان HTML را می نویسد.
WriteEndTag	بالافاصله تگ انتهایی یک المان HTML را می نویسد.
WriteFullBeginTag	بلافاصله تگ آغازین به همراه براکت پایان (>) المان HTML را می نویسد.
WriteLine	بلافاصله یک خط از محتویات را می نویسد. معادل است با متد Write اما WriteLine یک کاراکتر خط جدید را نیز اضافه می کند.

همانطور که در جدول ۱ مشاهده می شود دو روش برای نوشتن المانهای HTML وجود دارند. پر استفاده ترین روش ، استفاده از متد Write برای اضافه کردن HTML به صورت مستقیم به HTMLTextWriter است.

مثال ۱:

مثال زیر یک دکمه را ایجاد کرده و سپس یک MessageBox را هنگامیکه کاربر روی آن کلیک می کند نمایش می دهد.

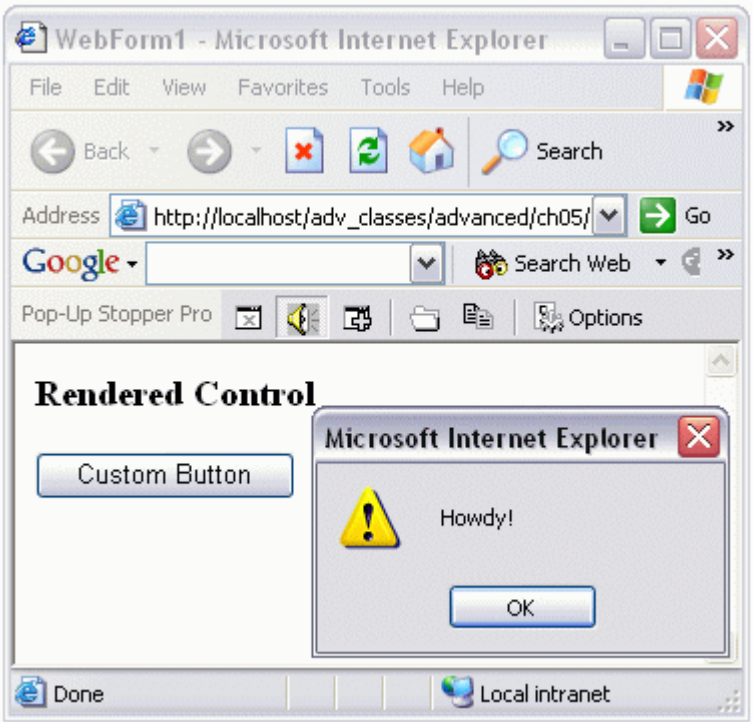
یک پروژه ی جدید web control library را ایجاد نمایید. کد درون کلاس آنرا که به صورت خودکار ایجاد شده است پا ک نموده و سپس کد زیر را به آن اضافه کنید:

```
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
	{

		/// <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)
		{
			// Write a title
			output.Write("<h3>Rendered Control</h3>");
			// Opens an Input HTML tag (inserts "<INPUT").
			output.WriteBeginTag("INPUT");
			// Write some attributes.
			output.WriteAttribute("value", "Custom Button");
			output.WriteAttribute("type", "button");
			output.WriteAttribute("onclick", "javascript:alert('Howdy!')");
```

```
                // Close the Input HTML tag (inserts ">").
                output.WriteEndTag("INPUT");
            }

        }
}
```

سپس همانند مراحلی که در فصل قبل آموختیم ، یک پروژه ی Web application جدید را به پـروژه ی جاری اضافه نموده ، آنرا بعنوان پروژه ی آغازین تنظیم کنید، ریفـرنس لازم بـه کنتـرل را ایجـاد کـرده و سپس با استفاده از تگ های زیر آنرا به صفحه وب اضافه نمایید:

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

<Custom:WebCustomControl1 id="Test1" runat="server" />
```



شکل – نمونه ی از اجرای برنامه .

مثال ۲ :

اگر کد فوق را از روش دیگری بخواهیم ارائه دهیم به صورت زیر خواهد شد (بدیهی است که ایجاد پروژه های جدید و غیره همانند قبل 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
	{
		protected override void Render(HtmlTextWriter output)
		{
			// Write a title
			output.Write("<h3>Rendered Control</h3>");
			// Add some attributes.
			output.AddAttribute("value", "Custom Button");
			output.AddAttribute("type", "button");
			output.AddAttribute("onclick", "javascript:alert('Howdy!')");
			// Opens an Input HTML tag (inserts "<INPUT").
			output.RenderBeginTag("INPUT");
			// Close the Input HTML tag (inserts ">").
			output.RenderEndTag();
		}

	}
}
```

در حقیقت این روش نحوه ی اضافه کردن کدهای جاوا اسکریپت را به کنترل ما نمایش می دهد.

ذخیره سازی تنظیمات خواص:

از آنجائیکه کنترلهای رندر شده ی سفارشی از کنترل های موجود ساخته نمی شوند بنابراین باید با استفاده از ViewSate آنها ، خواصشان را بین نمایش متوالی صفحات حفظ کرد.

مثال ۳ :

یک پروژه ی جدید web control library را ایجاد نمایید. کد درون کلاس آنرا که به صورت خودکار ایجاد شده است پا ک نموده و سپس کد زیر را به آن اضافه کنید.

کد زیر خاصیت Text ایی را نمایش می دهد که مقدار خودش را بین نمایش متوالی صفحات حفظ می نماید.

```
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
	{

		[DefaultValue("")]
		public string Text
		{
			get
			{
				// If the property has been set
				if (ViewState["Text"] != null)
					// Return the settng.
					return ViewState["Text"].ToString();
				else
					// Otherwise return "".
					return null;
			}

			set
			{
```

```
                        // Store the property setting.
                        ViewState["Text"] = value;
                }
        }

        protected override void Render(HtmlTextWriter output)
        {
                // Set the Input control's attributes.
                output.AddAttribute("value", this.Text);
                // (2) You must define this attribute to enable PostBack data.
                output.AddAttribute("name", this.UniqueID);
                // Create an Input control element using above.
                output.RenderBeginTag("Input");
                // Close the Input control element (inserts />).
                output.RenderEndTag();
        }

    }
}
```

سپس همانند مراحلی که در فصل قبل آموختیم ، یک پروژه ی Web application جدید را به پـروژه ی جاری اضافه نموده ، آنرا بعنوان پروژه ی آغازین تنظیم کنید، ریفـرنس لازم بـه کنتـرل را ایجـاد کـرده و سپس با استفاده از تگ های زیر آنرا به صفحه وب اضافه نمایید:

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

<Custom:WebCustomControl1 id="Test1" runat="server" />
```

با تنظیم ViewState ، داده ها درون فیلدهای مخفی بر روی وب فرم ذخیره می شـوند و کـار خوانـدن اطلاعات برای بار بعد نیز از این فیلدهای مخفی صورت خواهد گرفت.

ViewState برای نوع های داده ای string و ArrayList و HashTable بهینه سازی شده است اما هر گونه داده ای را که دارای TypeConverter باشد می توان در آن ذخیره کرد. برای سـایر نـوع هـای داده ایی می توان با تحریف کردن متـدهای SaveViewState و LoadViewState مقـادیر را در ViewState ذخیره و بازیابی کرد.

عملیات بر روی متن روی صفحه :

المان های HTML عموما متون را درون تگ هایی همانند مثال زیر محصور می کنند:

 bold me!

برای انجام عملیات روی یک متن ، دریافت و تغییرات روی آن به صورت زیر عمل می شود:

۱- اضافه کردن خاصیت ParseChildren به تعریف کنترل کلاس پایه.

۲- پیاده سازی اینترفیس INamingContainer در یک کلاس کنترل سفارشی

۳- دریافت متن بوسیله کلکسیون Controls .

مثال ۴ :

کلاس کنترل سفارشی زیر نحوه ی نمایش یک متن را در کنترل سفارشی با رنگ قرمز بیان می کند.

(چون مراحل ایجاد پروژه و اضافه کردن آن به یک وب فرم بارها در این فصل و فصل پیشین تکرار گردیده است از ذکر مجدد آنها صرفنظر می شود و بدیهی است که این مراحل باید انجام شود)

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

namespace WebControlLibrary1
{

      [DefaultProperty("Text"),
      ToolboxData("<{0}:WebCustomControl1 runat=server></{0}:WebCustomControl1>")]

      [ParseChildren(false)]

      public class WebCustomControl1 : System.Web.UI.WebControls.WebControl,
INamingContainer
      {

      // INamingContainer interface makes it possible to get the Controls array.

            protected override void Render(HtmlTextWriter output)
            {
                  // Contained text is returned as a Literal control.
                  LiteralControl litText ;
                  // Get the contained text from the Controls collection.
```

```
                litText = (LiteralControl)Controls[0];
                // Make it red using the <font> element.
                output.Write("<font color='red'>" + litText.Text + "</font>");
            }

        }
}
```

و در یک وب فرم با استفاده از تگ های زیر می توان از کنترل استفاده نمود:

```
<Custom:WebCustomControl1 id="Test1" runat="server">some red text</Custom:WebCustomControl1>
```

مثال ۵ :

با استفاده از کلکسیون کنترل ها می توان هرگونه آیتمی را بین تگ های آغـازین و پایـانی نیـز دریافـت نمود. برای مثال ، کلاس کنترل سفارشی زیر ، کنترل ها را در یـک panel گـرد آوری کـرده و آنهـا را در مرکز پنل در صفحه نمایش می دهد و سپس زمینه ی آنها را قرمز می کند.

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

namespace WebControlLibrary1
{
      [DefaultProperty("Text"),
      ToolboxData("<{0}:WebCustomControl1 runat=server></{0}:WebCustomControl1>")]
      [ParseChildren(false)]
      public class WebCustomControl1 : System.Web.UI.WebControls.WebControl,
INamingContainer
      {
      // INamingContainer interface makes it possible to get the Controls array.


            protected override void Render(HtmlTextWriter output)
            {
                  // Add some attributes for the panel.
                  output.AddAttribute("align", "center");
                  output.AddStyleAttribute("BACKGROUND-COLOR", "red");
                  // Start a panel
                  output.RenderBeginTag("div");
```

```
                // For each contained control.
                foreach (Control ctrItem in Controls)
                {
                      // Render the control
                      ctrItem.RenderControl(output);
                }
                output.RenderEndTag();
          }

     }
}
```

واکنش نشان دادن به عملیات کاربر:

کنترل های رندر شده نیاز به مراحل خاصی برای ایجاد و پاسخ دادن به رویدادها دارند. نحوه ی مواجه شدن با رویدادها بستگی به نوع آن پیدا می کند:

- Cached events : از درون یک کد سفارشی ایجاد می شوند.
- Post-back events : از طرف کلاینت فرستاده می شود و باید توسط اسکریپت های مربوطه انجام شود.

در ادامه این موارد را به تفصیل بررسی خواهیم کرد:

Raising Cached events :

رخدادهای chached بوسیله ی کدهای درون وب فرم ها مدیریت می شوند هنگامکیه وب فرم از طرف کاربر به سرور بازگشت داده داده می شود و بعد از فراخوانی رخداد Page_Load . مدیریت آنها شامل دو مرحله می شود:

۱- تعریف یک رخداد

۲- ایجاد کردن رخداد در جایی درون کد کنترل سفارشی

مثال ۶ :

برای مثال ، کلاس کنترل سفارشی زیر رخداد OnChange را تعریف کرده است بطوریکه هرگاه خاصیت Text آن تغییر کند این رخداد فراخوانی می گردد.

```
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>")]
	[DefaultEvent("Change")]
	public class WebCustomControl1 : System.Web.UI.WebControls.WebControl
	{

		// Declare event.
		public event EventHandler Change;

		[DefaultValue("")]
		public string Text
		{
			get
			{
				// If the property has been set
				if (ViewState["Text"] != null)
					// Return the settng.
					return ViewState["Text"].ToString();
				else
					// Otherwise return "".
					return null;
			}

			set
			{
					// Store the property setting.
					ViewState["Text"] = value;
					// Call event method.
					OnChange(EventArgs.Empty);
			}
		}

		protected override void Render(HtmlTextWriter output)
		{
			// Set the Input control's attributes.
			output.AddAttribute("value", this.Text);
```

```
                // (2) You must define this attribute to enable PostBack data.
                output.AddAttribute("name", this.UniqueID);
                // Create an Input control element using above.
                output.RenderBeginTag("Input");
                // Close the Input control element (inserts />).
                output.RenderEndTag();
            }


            protected virtual void OnChange(EventArgs e)
            {
                if (Change != null)
                    // Raise event.
                    Change(this, e);
            }

        }
}
```

و از کد زیر برای مدیریت رخداد change در وب فرم استفاده می شود:

```
private void Test1_Change(object sender, System.EventArgs e)
{
    Response.Write("Text changed!");
}
```

Raising Post-back events :

یک رخداد Post-back سبب می شود تا وب فرم به سرور برای پردازش آنی فرستاده شود. این رخدادها را می توان توسط Page_Load مدیریت کرد. برای Raising یک رخداد Post-back از یک کنترل سفارشی مراحل زیر باید انجام شود:

۱- پیاده سازی اینترفیس IPostBackEventHandler در کنترل سفارشی

۲- تعریف رخداد

۳- ایجاد المان های HTML برا پاسخ دادن به رخدادهای کلاینت سایدماننd onclick .

۴- اضافه کردن ویژگی های لازم به المان های HTML حاوی اسکریپت تا بتوانند صفحه را به سرور بفرستند.

۵- ایجاد رخداد از متد RaisePostBackEvent که جزئی است از اینترفیس IPostBackEventHandler .

مثال ۷ :

کد زیر این مراحل را در عمل نشان می دهد:

```
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>")]
	[DefaultEvent("Click")]
	public class WebCustomControl1 : System.Web.UI.WebControls.WebControl,
IPostBackEventHandler
	{
		// (2) Declare postback event
		public event EventHandler Click;

		// (3) Render an HTML element that can detect user events.
		protected override void Render(HtmlTextWriter output)
		{
			// Write a title
			output.Write("<h3>Rendered Control</h3>");
			// Add some attributes.
			output.AddAttribute("value", "Custom Button");
			output.AddAttribute("type", "button");
			// (4) Add attribute to raise Postback event on client.
			output.AddAttribute("onclick", "javascript:alert('Howdy!');" +
				Page.GetPostBackEventReference(this));
			// Opens an Input HTML tag (inserts "<INPUT").
			output.RenderBeginTag("INPUT");
			// Close the Input HTML tag (inserts ">").
			output.RenderEndTag();
		}

		// Part of the IPostBackEventHandler interface.
		// you must create this method.
		public void RaisePostBackEvent(string eventArgument)
		{
```

```
            // (5) Call the event method to raise event.
            OnClick(EventArgs.Empty);
        }

        protected virtual void OnClick(EventArgs e)
        {
            // Raise the event.
            if (Click != null)
                Click(this, e);
        }

    }
}
```

هنگامیکه کاربر بر روی کنترل سفارشی کلیک می کند ، رخداد OnClick مربوط به دکمه ، اسکریپت زیر را اجرا می کند:

```
alert('Howdy!');
__doPostBack('winTest','');//generated by Page.GetPostBackEventReference
```

متد _doPostBack صفحه را به سرور می فرستد و توسط متد RaisePostBack مربوط به اینترفیس IPostBackEventHandler پردازش می شود.

از کد زیر برای مدیریت آن استفاده می شود:

```
private void Test1_Click(object sender, System.EventArgs e)
{
    Response.Write("clicked!");
}
```

دریافت داده ها از کاربر:

کنترل های رندر شده به صورت اتوماتیک داده های وارد شده از طرف کاربر را نگهداری نمی کنند. برای مثال اگر توسط المان input مربوط به HTML که یک تکست باکس را ایجاد می کند ، یک کنترل

سفارشی ایجاد نموده و درون آن چیزی را تایپ کنید به محض اینکه صفحه به سرور فرستاده می شود ، تمام متن نوشته شده داخل آن پاک می شود. برای دریافت داده ها از کاربر باید مراحل زیر طی شود:

۱- پیاده سازی اینترفیس IPostBackDataHandler .

۲- اضافه کردن خاصیت name به المان HTML تا منحصربفرد شده و بتوان توسط آن داده ها را دریافت کرد.

۳- تحریف کردن متدهای LoadPostBackData مربوط به اینترفیس IPostBackDataHandler.

۴- تحریف کردن متد RaisePostDataChangedEvent واقع شده در اینترفیس IPostBackDataHandler .

۵- ایجاد یک رخداد تا مشخص شود که data تغییر کرده است (این گزینه اختیاری است).

مثال ۸ :

کنترل سفارشی زیر نشان می دهد که چگونه می توان این مراحل را پیاده سازی کرد و داده های وارد شده از طرف کاربران را در یک تکست باکس رندر شده و ایجاد شده توسط المان input استاندارد HTML ، نگهداری کرد.

```
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>")
	, DefaultEvent("Change")]
	public class WebCustomControl1 : System.Web.UI.WebControls.WebControl,
IPostBackDataHandler
	{
		// Declare event.
		public event EventHandler Change;

		[DefaultValue("")]
		public string Text
```

```
        {
              get
              {
                    // If the property has been set
                    if (ViewState["Text"] != null)
                          // Return the settng.
                          return ViewState["Text"].ToString();
                    else
                          // Otherwise return "".
                          return null;
              }

              set
              {
if ((ViewState["Text"] != null) && (ViewState["Text"].ToString() != value))
                    {
                          // Store the property setting.
                          ViewState["Text"] = value;
                          // Call event method.
                          OnChange(EventArgs.Empty);
                    }
              }
        }

        protected override void Render(HtmlTextWriter output)
        {
              // Set the Input control's attributes.
              output.AddAttribute("value", this.Text);
              // (2) You must define this attribute to enable PostBack data.
              output.AddAttribute("name", this.UniqueID);
              // Create an Input control element using above.
              output.RenderBeginTag("Input");
              // Close the Input control element (inserts />).
              output.RenderEndTag();
        }

        // (3) Get data from the user.
        // You must create this method.
        public bool LoadPostData(string postDataKey,
            System.Collections.Specialized.NameValueCollection postCollection)
        {
              // If the user changes Input value, update the text property.
              if (((ViewState["Text"] != null) &&
                (ViewState["Text"].ToString() != postCollection[postDataKey]))
                    || ((ViewState["Text"] == null) &&
                              (postCollection[postDataKey] != "")))
              {
                    ViewState["Text"] = postCollection[postDataKey];
                    // Returning true invokes RaisePostDataChangedEvent below.
                    return true;
              }
              else
        // Returning False does not invoke RaisePostDataChangedEvent.
                    return false;
        }
```

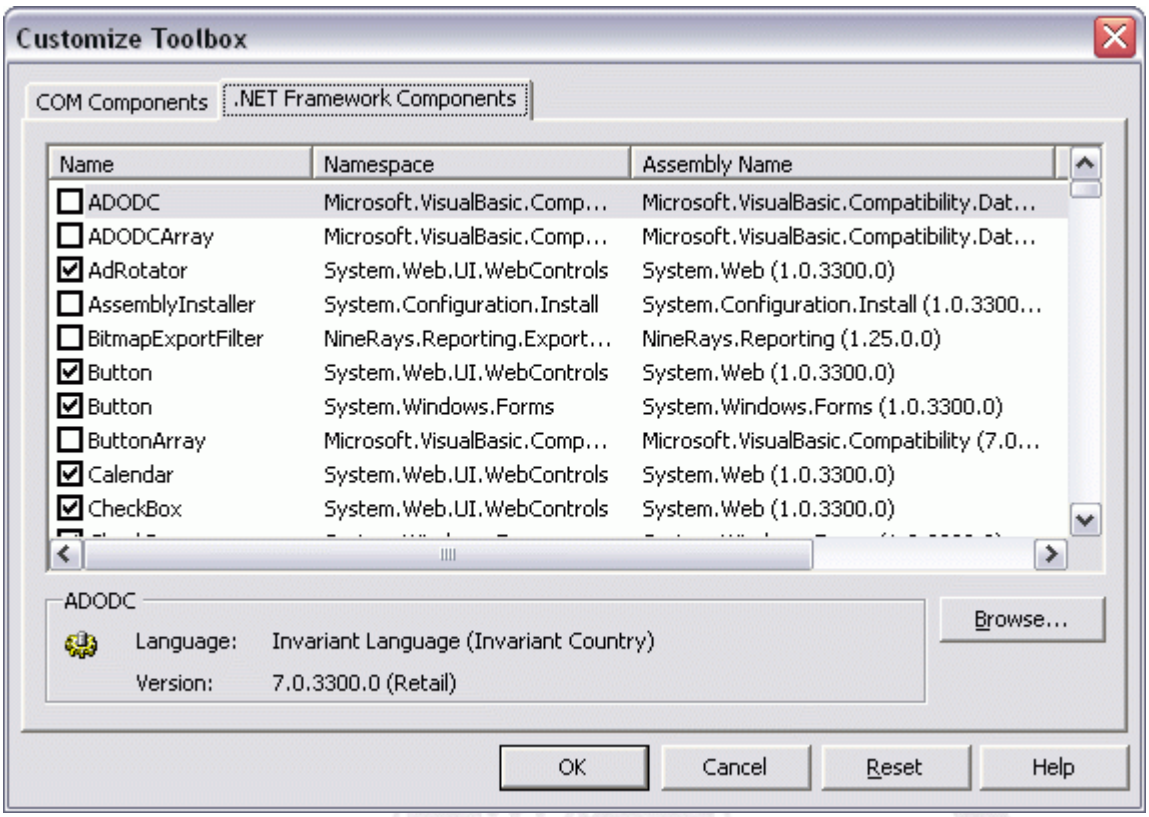
```
        // (4) Override RaisePostDataChangedEvent method.
        // You must create this method.
        public void RaisePostDataChangedEvent()
        {
            // (5) Call the event method. This is optional.
            OnChange(EventArgs.Empty);
        }

        protected virtual void OnChange(EventArgs e)
        {
            if (Change != null)
                // Raise event.
                Change(this, e);
        }


    }
}
```

اضافه کردن کنترل های سفارشی به ToolBox :

همانطور که پیش تر نیز ذکر شد یکی از قابلیت های انواع کنترل های سفارشی ، توانایی اضافه کردن آنها به ToolBar استاندارد VS.NET است. برای این منظور به صورت زیر عمل می شود:

۱- از منوی Tools گزینه ی Customize-toolBox را انتخاب کنید. شکل زیر ظاهر می شود.
۲- tab مربوط به .Net framework components را انتخاب نمایید و سپس بر روی Browse کلیک نمایید. سپس VS.NET دیالوگ باکس Open را نمایش می دهد.
۳- سپس فایل Assembly (.dll) کنترل را انتخاب کرده و در ادامه خودبخود به لیست کنترل ها اضافه می شود.
۴- بر روی دکمه ی Ok کلیک نموده و عملیات را تکمیل نمایید.

اکنون به ToolBox استاندارد مراجعه نمایید ، کنترل شما به آن اضافه شده است. برای استفاده از آن هم مانند سایر کنترل های وب استفاده می شود و تمام تگ های لازم را خود VS.NET به صورت اتوماتیک اضافه می کند و نیازی به نوشتن دستی آنها وجود ندارد.



شکل – اضافه کردن یک کنترل سفارشی به ToolBar از طریق دیالوگ باکس فوق انجام می شود.

تنظیم کردن TagPrefix و آیکون یک کنترل سفارشی هنگامیکه به ToolBar اضافه می شود:

برای اضافه کردن آیکون به کنترل سفارشی مراحل زیر طی می شود:

۱- یک فایل bitmap شانزده در شانزده که حاوی تصویر دلخواهی است را تهیه نمایید. آنرا در دایرکتوری bin پروژه ی کنترل سفارشی کپی نمایید.

۲- خط زیر را به ابتدای کلاس کنترل سفارشی اضافه نمایید :

```
using System.Drawing;
```

۳- ویژگی ToolBoxBitmap را به تعریف کلاس کنترل اضافه کنید:

```
[ParseChildren(False),ToolBoxBitmap("Red")]
```

۴- پروژه را دوباره کامپایل نمایید.

برای ایجاد یک TagPrefix مناسب برای کنترل که به صورت خودکار با هر بار قرار دادن کنترل روی صفحه توسط VS.NET ایجاد شود مراحل زیر طی می شود:

۱- اضافه کردن

```
using System.Web.UI;
[assembly:TagPrefix("namespace","prefix")]
```

۲- کامپایل مجدد

تمرین

۱- یک کنترل رندر شده ی تکست باکس را با استفاده از المان استاندارد input بوجود آورید و از آن در یک صفحه ی لاگین برای پست کردن اطلاعات به سرور استفاده نمایید.

۲- به کنترلی که در سوال یک ایجاد کرده اید یک تصویر برای نمایش در ToolBox و یک TagPrefix مناسب و دلخواه نسبت دهید.