عنوان: **به دام انداختن خطاهای مدیریت نشده در برنامه‌های Windows forms دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۴ ۱۴:۰۳:۵۱
آدرس: www.dotnettips.info
برچسب‌ها: Windows forms

شبیه به نحوه‌ی به دام انداختن خطاهای مدیریت نشده در Web forms و روال استاندارد Application_Error ، در برنامه‌های Windows forms نیز این امر به صورت زیر ممکن است:

```
using System;
using System.Threading;
using System.Windows.Forms;

namespace testWinForms87
{
 static class Program
 {
    /// <summary>
    /// The main entry point for the application.
    /// </summary>
    [STAThread]
    static void Main()
    {
        // handling UI thread exceptions
        Application.ThreadException += uIThreadException;

        // force all Windows Forms errors to go through our handler.
        Application.SetUnhandledExceptionMode(UnhandledExceptionMode.CatchException);

        // handling non-UI thread exceptions.
        AppDomain.CurrentDomain.UnhandledException += currentDomainUnhandledException;


        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
    }

    private static void currentDomainUnhandledException(object sender, UnhandledExceptionEventArgs e)
    {
        MessageBox.Show(((Exception)e.ExceptionObject).Message, "currentDomainUnhandledException");
    }

    private static void uIThreadException(object sender, ThreadExceptionEventArgs e)
    {
        MessageBox.Show(e.Exception.Message, "uIThreadException");
    }
 }
}
```

چند نکته:
الف)همانطور که ملاحظه می‌کنید سطرهای فوق باید قبل از Application.Run در روال اصلی برنامه تعریف شوند.
ب) این متدها استاتیک هستند و توصیه شده است در پایان برنامه ارجاعات آنها را حذف کنید تا نشتی حافظه رخ ندهد. دقیقا به همین صورت =+ که اضافه شدند با =- هم قابل حذف هستند.
ج) در حالت اجرا شدن uIThreadException ، برنامه بسته نخواهد شد (و بدیهی است در صورت عدم بکار گیری این روش، حتما برنامه کرش خواهد کرد). برای مثال شاید علاقمند نباشید که بخاطر عدم دسترسی نوشتن در پوشه‌ای خاص، خطای حاصل سبب بسته شدن کل برنامه شود. به این صورت این موارد را می‌توان به دام انداخت. اما currentDomainUnhandledException که حاصل از خطاهای ایجاد شده برای مثال در یک ترد دیگر بجز ترد اصلی برنامه هستند، حتما سبب بسته شدن برنامه خواهند شد. بنابراین اینجا تنها شانس لاگ کردن خطای مدیریت نشده حاصل را خواهیم داشت. به همین منظور همیشه توصیه می‌شود که در تردهای ایجاد شده در برنامه، حتما موارد مدیریت خطاها را لحاظ نمائید، زیرا خطاهای حاصل شده در آن‌ها غیرقابل اغماض بوده و حتما سبب کرش برنامه می‌شوند.

پ.ن.
دقیقا در برنامه‌های Win32 دلفی هم چنین قابلیتی به همین شکل و تقریبا با همین نام‌ها وجود دارد. فقط کافی است روالی را جهت Application.OnException ایجاد کنید: ;)

```
procedure TmyFrmMain.FormCreate(Sender: TObject);
begin
  Application.OnException := MyExceptionHandler;
end;
procedure TmyFrmMain.MyExceptionHandler(Sender: TObject; E: Exception);
begin
  ShowMessage(e.Message);
end;
```

نظرات خوانندگان

نویسنده: Alex's Blog
تاریخ: ۱۳۸۷/۱۰/۱۴ ۱۴:۴۴:۰۰

سلام. ممنون از مطالبتون. خیلی مفید هستند.
در ضمن قالبتون فکر کنم عوض شده مبارکه.

نویسنده: عرفان طاهری
تاریخ: ۱۳۸۷/۱۰/۱۴ ۲۱:۰۵:۰۰

به به قالب جدید مبارکه :D...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۴ ۲۲:۵۴:۰۰

مرسی. این قالب قبلی که پیش فرض گوگل بود اصلا رنگ و رویی نداشت :)

نویسنده: مهدی
تاریخ: ۱۳۸۷/۱۰/۱۵ ۱۰:۵۰:۰۰

قالبه بهتره، ولی با اپرا، عنوان وبلاگ
(تازه‌های دنیای برنامه نویسی)به هم می ریزه.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۵ ۱۱:۰۸:۰۰

فعلا با این مرورگرها تست شده:
IE7
فایرفاکس 3
کروم گوگل

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۵ ۱۲:۱۲:۰۰

با تشکر از دقت نظر شما. مشکل با opera هم برطرف شد.

عنوان: **مشکل چشمک زدن (Flicker) در کنترل ListView**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۰۹ ۱۰:۲۴:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: Windows forms

با هر بار اضافه کردن یک سطر به ListView ، تمام ناحیه پس زمینه کنترل به روز شده و مشکل چشمک زدن (Flicker) آزار دهنده‌ای را پدید می‌آورد. راه حل‌های زیادی برای رفع این مشکل وجود دارد. برای مثال استفاده از متدهای BeginUpdate و EndUpdate قبل و پس از افزودن تعداد زیادی رکورد به یک ListView . اما اگر این کنترل توسط چند ترد در حال به روز رسانی باشد و هربار هم تعداد آیتم‌های اضافه شده آنچنان زیاد نباشد، این روش اثری نداشته و باز هم مشکل flickering وجود خواهد داشت.
رفع این مشکل راه حل بسیار ساده‌ای دارد که به شرح زیر است:

یک user control جدید ایجاد کنید، آن‌را از ListView به ارث برده و سپس سطر زیر را به constructor آن اضافه کنید:

```
this.DoubleBuffered = true;
```

اکنون از این ListView سفارشی بجایlistView استاندارد استفاده کنید، مشکل برطرف می‌شود!

```
public partial class CustomListView : ListView
   {
       public CustomListView()
       {
           this.DoubleBuffered = true;
       }
   }
```

شبیه به همین مورد را جهت کنترل ListBox نیز می‌توان پیاده سازی کرد

نظرات خوانندگان

نویسنده: هدی شاهزاده احمدی
تاریخ: ۱۳۸۷/۱۲/۱۰ ۲۳:۰۸:۰۰

فکر میکنم این DoubleBuffered از چیزهایی باشد که از دلفی وارد سی شارپ شده است.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۱ ۰۲:۰۵:۰۰

فرقی نمی‌کند برای استفاده از کنترل‌های استاندارد ویندوزی از چه محصور کننده‌ای استفاده شود. شما حتی اگر از QT نسخه‌ی ویندوزی هم استفاده کنید، این کتابخانه محصور کننده‌ای خواهد بود برای API ویندوز.

عنوان: **مروری بر کاربرد DoEvents**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۴ ۱۶:۱۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Windows forms

چند روز قبل هنگام استفاده از DoEvents در یک برنامه windows forms ، ناگهان پیغام stack overflow ظاهر شد! برای علت یابی و رفع آن کمی جستجو کردم که خلاصه‌ی آن به شرح زیر است:

DoEvents چیست؟

DoEvents یکی از متدهای کلاس Application در فضای نام System.Windows.Forms است.
ویندوز جهت مدیریت رخدادهای مختلف از یک صف استفاده می‌کند. رخدادهایی مانند کلیک ماوس، تغییر اندازه‌ی یک فرم و مواردی شبیه به آن ابتدا در یک صف قرار می‌گیرند و سپس پردازش می‌شوند. زمانیکه کنترلی مشغول پاسخ دهی به یک رخداد می‌گردد، سایر رخدادها هنوز در صف هستند و پردازش نخواهند شد. بنابراین اگر برنامه‌ی شما در یک روال رخدادگردان کلیک، عملیاتی طولانی را در حال انجام باشد، بدلیل عدم پردازش سایر رخدادها اینطور به نظر خواهد رسید که هنگ کرده است.
روش صحیح پردازش یک عملیات طولانی استفاده از یک ترد دیگر می‌باشد تا ترد اصلی برنامه که کار مدیریت رابط کاربر برنامه را به عهده دارد، درگیر این عملیات طولانی نشده و پاسخگوی رخدادهای رسیده باشد.
راه میان‌بر و ساده‌ای که اینجا وجود دارد استفاده از DoEvents می‌باشد (بدون ایجاد یک ترد جدید). برای مثال اگر در روال رخ دادگردان کلیک یک برنامه، حلقه‌ای طولانی در حال پردازش است، هر از چندگاهی این متد فراخوانی شود، رخدادهای در صف قرار گرفته فرصت ارسال به ترد اصلی برنامه را یافته و برنامه در حالت هنگ به نظر نخواهد رسید.
برای نمونه مثال زیر را در دو حالت با Application.DoEvents و بدون آن اجرا کنید:

```
private void btnProcessWithDoEvents_Click(object sender, EventArgs e)
{
for (int i = 0; i < 100000; i++)
{
  TextBox1.Text = "Processing " + i.ToString();
  Application.DoEvents();
}
}
```

در حالت بدون استفاده از Application.DoEvents ، تنها آخرین عبارت پردازش شده را در TextBox1 مشاهده خواهید کرد و همچنین در این حین، برنامه در حالت هنگ به نظر می‌رسد و برعکس.

مشکلات احتمالی حاصل از استفاده از Application.DoEvents :

الف) حس غلط پایان یافتن عملیات پیش از موعد
در مثال فوق در حین استفاده از Application.DoEvents ، دکمه‌ی btnProcessWithDoEvents مجددا فعال شده و قابل کلیک کردن می‌شود ولی آیا این بدین معنا است که پردازش قبلی به پایان رسیده است؟ به یک سری از کاربرها هم click-happy user گفته می‌شود! یعنی از کلیک کردن مجدد لذت می‌برند! در این حالت حتما باید دکمه‌ی btnProcessWithDoEvents را در ابتدای پردازش غیرفعال کرد و سپس در انتهای آن باید مجددا فعال شود.
مورد مشکل کلیک مجدد حتی می‌تواند منجر به تخریب اطلاعات در حال پردازش شود. فرض کنید برنامه در حال ذخیره‌ی اطلاعات در یک فایل است و کاربر مرتبا بر روی دکمه‌ی پردازش مربوطه کلیک کنید. فایل نهایی از یک سری اطلاعات ناهماهنگ و بی‌ربط پر خواهد شد.

ب) مشکل stack overflow
اگر علاقمند باشید، این مورد را می‌توان به صورت زیر شبیه سازی کرد:

یک تایمر را به برنامه اضافه کنید و یک دکمه. در روال رخدادگردان کلیک مربوط به دکمه، دستورات زیر را اضافه کنید:

```
private void btnStartTimer_Click(object sender, EventArgs e)
{
  this.timer1.Enabled = true;
  this.timer1.Start();
  this.timer1.Interval = 20;

}
```

و در روال tick مربوط به تایمر، دستورات زیر را اضافه کنید:

```
private void timer1_Tick(object sender, EventArgs e)
{
  Thread.Sleep(50);
  Application.DoEvents();
}
```

برنامه را اجرا کرده و یکی دو دقیقه صبر کنید، حتما با پیغام خطای stack overflow مواجه خواهید شد. چرا؟
فواصل زمانی اجرای تایمر به 20 میلی ثانیه تنظیم شده است اما در روال رخداد گردان tick آن، نیاز به 50 میلی ثانیه (بیش از 20 میلی ثانیه) یا بیشتر برای اجرا دارد. با رسیدن به Application.DoEvents ، رخداد در صف قرار گرفته‌ی دیگر tick بلافاصله اجرا می‌شود و همینطور الی آخر، تا بالاخره stack overflow حاصل خواهد شد.

پس چه باید کرد؟

الف) هنگام استفاده از Application.DoEvents به موارد فوق حتما دقت داشته باشید.
ب) بجای استفاده از این روش که در بیشتر موارد یک ضعف برنامه نویسی محسوب می‌شود، شروع به استفاده از روش‌های غیرهمزمان نمائید. برای مثال استفاده از :

BackgroundWorker

Asynchronous delegates

Threads

تنها موردی را که هنگام کار با تردها باید در نظر داشت این است که امکان دسترسی به کنترل‌های یک فرم را از ترد دیگری که آن کنترل را ایجاد نکرده است، ندارید و برای این مورد راه حل‌های زیادی موجود است.
همچنین بخاطر داشته باشید در یک ترد استفاده از Application.DoEvents هیچ معنایی ندارد. ترد اصلی برنامه وظیفه‌ی به روز رسانی رابط کاربر برنامه و پاسخگویی به رخدادهای رسیده را به عهده دارد. زمانیکه پردازش در تردی دیگر صورت می‌گیرد، ترد اصلی برنامه تا پایان پردازش متد شما قفل نخواهد شد که نیازی به استفاده از این متد باشد. در این حالت استفاده از Application.DoEvents ، سبب بالا رفتن مصرف حافظه‌ی برنامه و همچنین بالا رفتن میزان مصرف CPU خواهد شد.

جهت مطالعه بیشتر

Keeping your UI Responsive and the Dangers of Application.DoEvents

نظرات خوانندگان

نویسنده: افشار محبی
تاریخ: ۱۳۸۷/۱۲/۱۷ ۰۹:۰۵:۰۰

در wpf هم مشکل freez شدن UI وجود دارد با این تفاوت که استفاده از DoEvents امکان پذیر نیست و راه حل‌های مشابه هم چندان چنگی به دل نمی‌زنند. آیا راهی اصولی برای wpf وجود دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۷ ۰۹:۳۰:۰۰

راه اصولی همان استفاده از ترد است. در wpf یک سری نکته ریز در این مورد هست که در مقاله زیر به آن اشاره شده است:
http://ascendedguard.com/2007/11/proper-multi-threading-in-wpf.html

عنوان: **اطلاع از بروز رسانی نرم افزار ساخته شده**
نویسنده: رضابیات
تاریخ: ۱۳۹۲/۰۵/۱۶ ۱۴:۵۵
آدرس: www.dotnettips.info
برچسب‌ها: Windows forms, xml, Software deployment, Check update application

برای شما هم پیش آمده که نرم افزاری را تهیه و منتشر کرده باشید و تمایل داشته باشید که استفاده کنندگان از وجود نسخه بروز شده مطلع شوند. یک راه ساده این است که اطلاعات نسخه جدید نرم افزار را داخل فایلی ذخیره کنیم و در وب سایت پشتیبانی نرم افزار قرار دهیم. حال بایستی اطلاعات این فایل را در زمان اجرای برنامه بررسی کنیم و در صورت وجود نسخه جدید از نرم افزار به کاربر اطلاع رسانی کنیم.
ابتدا فایل اطلاعات بروز رسانی نرم افزار را تهیه می‌کنیم و در وب سایت پشتیبانی نرم افزار قرار میدهیم. در اینجا از قالب Xml استفاده شده. که در آن Vertsion نسخه در دسترس نرم افزار است و URL هم مسیر وب سایت و یا فایل بروز رسانی است.

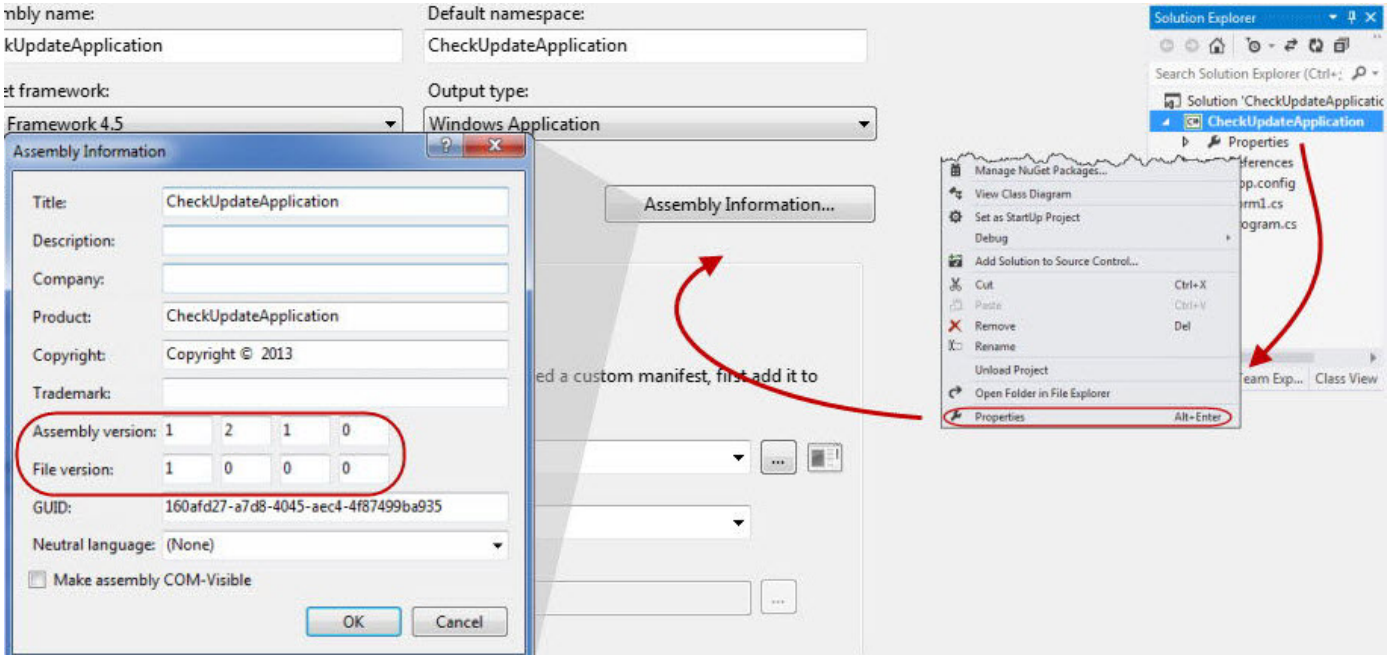
```
<?xml version="1.0" encoding="utf-8"?>
<AccountingApplication>
  <Version>1.5.2</Version>
  <URL>http://www.myappsupport.ir</URL>
</AccountingApplication>
```

نرم افزار را ساخته و کد زیر را در محل مناسبی کد نویسی می‌کنیم. این کد در ابتدا فایل Xml را خوانده و اطلاعات مورد نیاز را از آن دریافت می‌کند. سپس با استخراج نسخه اسمبلی برنامه و مقایسه این دو با هم از وجود نسخه جدید نرم افزار مطلع میشود.

```
...
using System.Xml;
namespace CheckUpdateApplication
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void CheckUpdate_Click(object sender, EventArgs e)
        {
            Version NewVersion = null;
            string DownloadPath = "";
            try
            {
                XmlTextReader xmlRead = new
XmlTextReader("http://www.myappsupport.ir/AccUpdateVersion.xml");
                xmlRead.MoveToContent();
                string elmName = "";
                if ((xmlRead.NodeType == XmlNodeType.Element) && (xmlRead.Name ==
"AccountingApplication"))
                {
                    while (xmlRead.Read())
                    {
                        if (xmlRead.NodeType == XmlNodeType.Element)
                        {
                            elmName = xmlRead.Name;
                        }
                        else
                        {
                            if ((xmlRead.NodeType == XmlNodeType.Text) && (xmlRead.HasValue))
                            {
                                switch (elmName)
                                {
                                    case "Version":
                                        NewVersion = new Version(xmlRead.Value);
                                        break;
                                    case "URL":
                                        DownloadPath = xmlRead.Value;
                                        break;
                                }
                            }
                        }
                    }
                }
                Version AppVertion =
```

```
System.Reflection.Assembly.GetExecutingAssembly().GetName().Version;
                if (AppVertion.CompareTo(NewVersion) < 0)
                {
                    DialogResult Result = MessageBox.Show("نسخه " +
                        NewVersion.Major.ToString() + "." +
                        NewVersion.Build.ToString() + " نسخه" ,"؟در دسترس میباشد مایل به دانلود هستید
",جدید
                        MessageBoxButtons.YesNo,MessageBoxIcon.Question);
                    if (Result == DialogResult.Yes)
                    {
                        System.Diagnostics.Process.Start(DownloadPath);
                    }
                }
                else
                {
                    MessageBox.Show("نرم افزار بروز میباشد");
                }
            }
            catch (Exception E)
            {
                MessageBox.Show(E.Message);
            }
        }
    }
}
```

به روش زیر هم نسخه اسمبلی برنامه را می‌شود تغییر داد.



سورس برنامه نمونه [CheckUpdateApplicationSample.rar]

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۵/۱۶ ۱۸:۹

ممنون از شما. یک روش برای اینکه مستقیما با XML Reader کار نکنیم می‌تونه استفاده از روش‌های سریالایز کردن کلاس‌ها باشه. دردسرش کمتره.

یک سؤال: این فلش‌های انحنا دار رو با چه برنامه‌ای ایجاد کردید؟

نویسنده: رضابیات
تاریخ: ۱۳۹۲/۰۵/۱۷ ۰:۳۴

با تشکر از توجه و راهنمایی شما. نرم افزارهای زیادی برای این کار وجود داره ولی من خیلی وقته از Snagit و Snagit Editor استفاده میکنم. این نرم افزار بیشتر برای فیلم و عکس گرفتن از دسکتاب استفاده میشه ولی امکانات فراوانی دیگری هم در این نرم افزار وجود داره. من خودم نسخه Snagit 11.2.1 را از سایت http://www.softgozar.com دانلود کردم.

نویسنده: مصطفی
تاریخ: ۱۳۹۲/۰۶/۰۹ ۹:۳۴

سلام
میخواستم بدونم فرق File Version و Assembly Version چیه؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۰۹ ۲۲:۱۲

Assembly Version برای مصرف کنندگان اسمبلی شما مهمه (و فقط در دنیای CLR دارای اهمیت هست). مثلا شخصی ارجاعی به اسمبلی نگارش خاصی داره. AssemblyFileVersion در قسمت خواص فایل در ویندوز قابل مشاهده است و بیشتر برای برنامه‌های ست آپ مفیده. اطلاعات بیشتر

عنوان: **حرکت دادن و تغییر اندازه کنترل‌های فرم در زمان اجرا با استفاده از ماوس**
نویسنده: زمرد2020
تاریخ: ۱۳۹۲/۱۱/۲۰ ۱۷:۲۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Windows forms, Controls

برخی اوقات نیاز است در یک فرم ویندوزی، کنترل‌های آن‌را در حال اجرا با استفاده از ماوس جابجا کنیم و یا اندازه‌ی آن‌ها را تغییر بدیم.
در وب راهکارهای مختلفی برای این کار ارائه شده، ولی این راه‌ها معمولا یا فقط برای تغییر مکان و یا فقط برای تغییر اندازه کنترل‌ها ارائه شده‌اند. من یکی از مقالات کد پروجکت را که به جابجا کردن کنترل‌ها پرداخته بود، توسعه دادم که امکان تغییر اندازه هم به آن اضافه شود. مقاله‌ی من (به زبان انگلیسی) در اینجا قرار دارد.
چون از کلاس و متدهای استاتیک استفاده کردم، روش استفاده از این کلاس ساده بوده و افزودن قابلیت تغییر اندازه و جابجایی زمان اجرا با ماوس برای هر کنترل فقط با یک خط کد قابل انجام است:

```
ControlMoverOrResizer.Init(button1);
```

نحوه‌ی استفاده از کلاس:
برای فعال کردن قابلیت تغییر اندازه و جابجایی یک کنترل در حال اجرای برنامه با موس ما باید متد Init از کلاس MoveAndResizeControls را فراخوانی کنیم و کنترل را به عنوان پارامتر به آن بفرستیم.

```
ControlMoverOrResizer.Init(button1);
```

اگر که ما بخواهیم به همراه تغییر کنترل ، خواص container آن را هم تغییر دهیم. باید کنترل container را به عنوان پارامتر دوم به متد مذکور ارسال کنیم.

```
ControlMoverOrResizer.Init(button2,panel1);
```

برخی اوقات ممکن است که ما فقط بخواهیم که یا کنترل‌ها را جابجا کنیم و یا اندازه‌ی آنها را تغییر دهیم؛ در این مواقع ما باید خاصیت WorkType کلاس MoveAndResizeControls را تغییر دهیم به یکی از مقادیر ذیل تغییر دهیم .

```
internal enum MoveOrResize
        {
            Move,
            Resize,
            MoveAndResize
        }
```

مثالی از نحوه‌ی کار با کلاس :

```
using System;
using System.Windows.Forms;
using ControlManager;
namespace MoveAndResizeControls
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            ControlMoverOrResizer.Init(button1);
            ControlMoverOrResizer.Init(groupBox1);
            ControlMoverOrResizer.Init(textBox1);
            ControlMoverOrResizer.Init(button2,panel1);
            comboBox1.SelectedIndex = 0;
        }
```

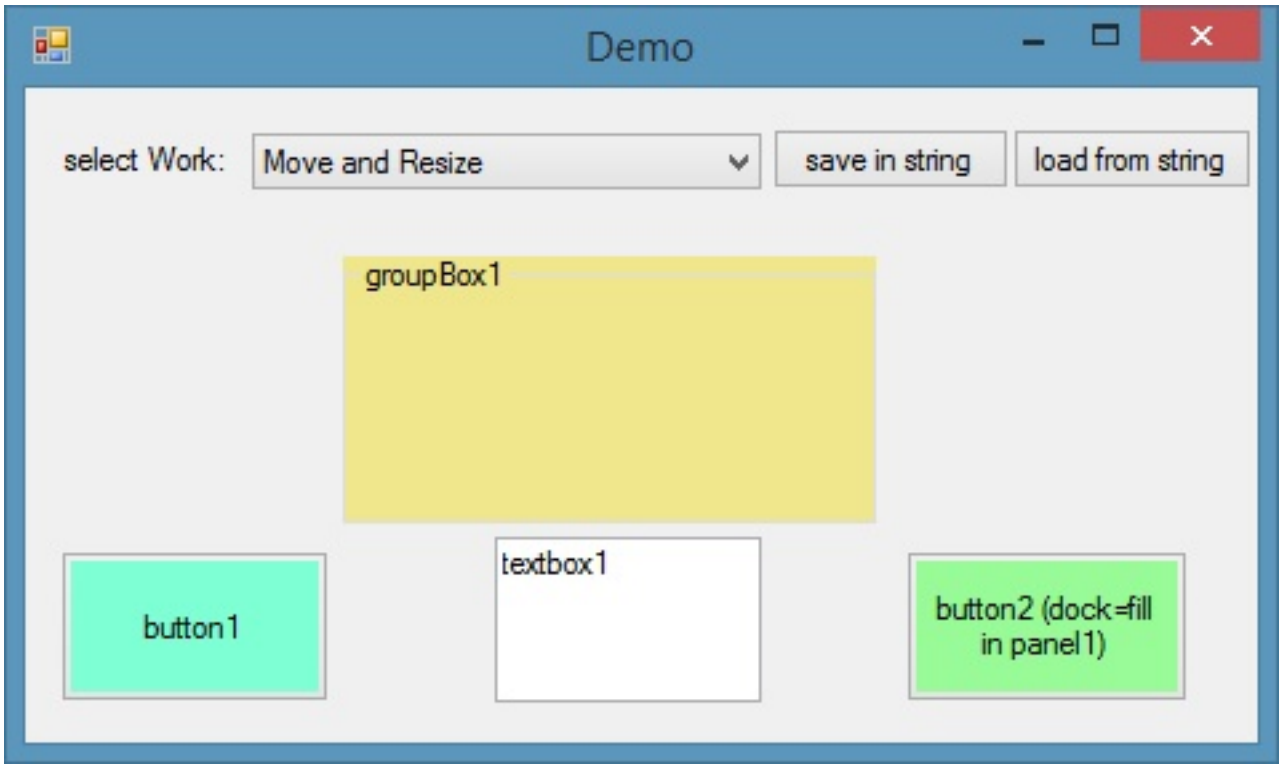
```
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (comboBox1.SelectedIndex)
            {
                case 0:
                    ControlMoverOrResizer.WorkType=ControlMoverOrResizer.MoveOrResize.MoveAndResize;
                    break;
                case 1:
                    ControlMoverOrResizer.WorkType = ControlMoverOrResizer.MoveOrResize.Move;
                    break;
                case 2:
                    ControlMoverOrResizer.WorkType = ControlMoverOrResizer.MoveOrResize.Resize;
                    break;
            }
        }
    }
}
```

نکته :بعد از انجام تغییرات، جهت ذخیره وضعیت کنترل‌ها و بازیابی مجدد آنها می‌توان از متدهای زیر استفاده کرد:

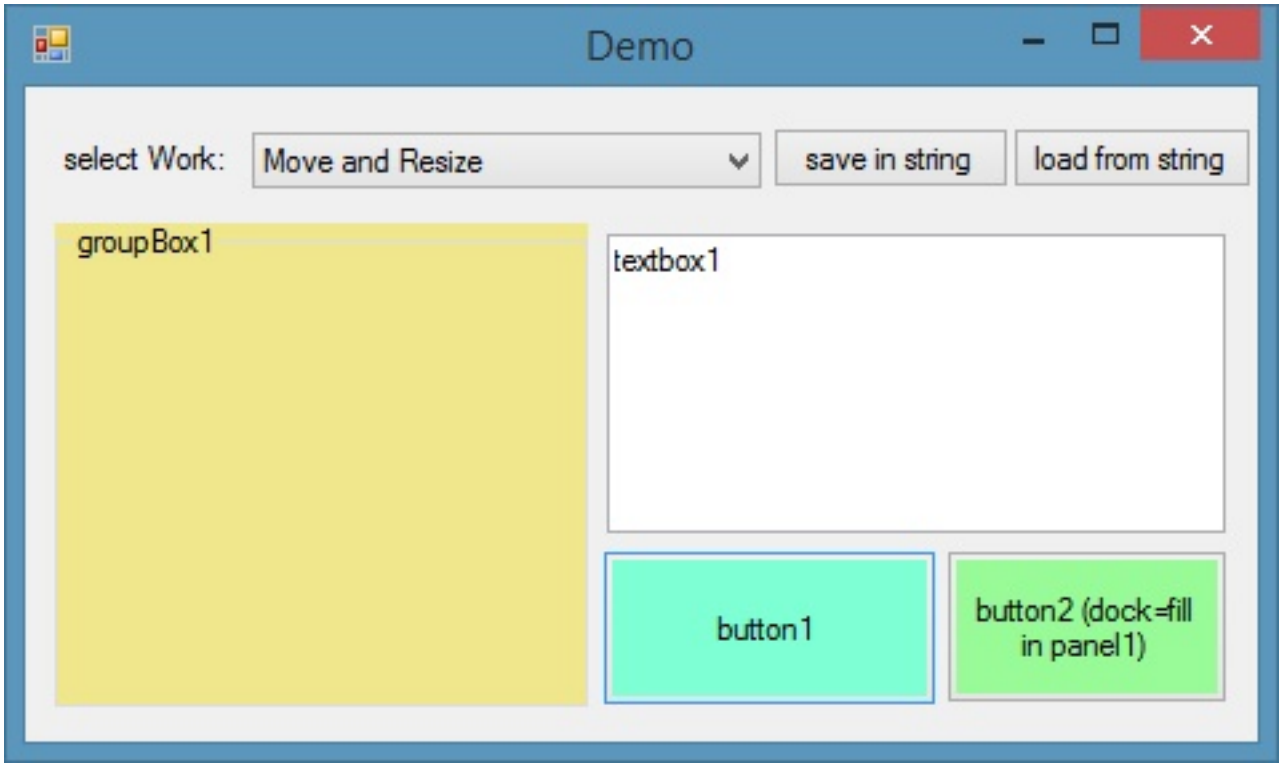
```
GetSizeAndPositionOfControlsToString  , SetSizeAndPositionOfControlsFromString
```

دانلود سورس

شکل حالت اولیه:



شکل حالت نتیجه:


Demo
select Work:
Move and Resize
save in string
load from string
groupBox1
textbox1
button1
button2 (dock=fill in panel1)