عنوان: **محاسبه مجدد میزان مصرف حافظه‌ی برنامه‌های دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۰۷ ۲۲:۵۴:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Tips

اگر به میزان مصرف حافظه اولیه‌ی برنامه‌های دات نت دقت کنیم، نسبت به مثلا یک برنامه‌ی MFC چند برابر به نظر می‌رسند و ... این علت دارد:
زمانیکه یک برنامه‌ی مبتنی بر دات نت اجرا می‌شود، ابتدا JIT compiler شروع به کار کرده و شروع به کامپایل برنامه می‌کند. این بارگزاری هم در همان پروسه‌ی اصلی برنامه انجام می‌شود. به همین جهت میزان مصرف حافظه‌ی برنامه‌های دات نت عموما بالا به نظر می‌رسد.
اکنون سؤال اینجا است که آیا می توان این حافظه‌ای را که دیگر مورد استفاده نیست (و توسط JIT compiler اخذ شده) به سیستم بازگرداند و محاسبه‌ی مجددی را در این مورد انجام داد. پاسخ به این سؤال را در متد ReEvaluateWorkingSet زیر می‌توان مشاهده کرد:

```
using System;
using System.Diagnostics;

namespace Toolkit
{
    public static class Memory
    {
        public static void ReEvaluateWorkingSet()
        {
            try
            {
                Process loProcess = Process.GetCurrentProcess();
                //it doesn't matter what you set maxWorkingSet to
                //setting it to any value apparently causes the working set to be re-evaluated and
excess discarded
                loProcess.MaxWorkingSet = (IntPtr)((int)loProcess.MaxWorkingSet + 1);
            }
            catch
            {
                //The above code requires Admin privileges.
                //So it's important to trap exceptions in case you're running without admin rights.
            }
        }
    }
}
```

در این متد ابتدا پروسه جاری دریافت شده و سپس MaxWorkingSet به یک عدد دلخواه تنظیم می‌شود. مهم نیست که این عدد چه چیزی باشد، زیرا این تنظیم سبب می‌شود که در پشت صحنه به شکل حساب شده‌ای حافظه‌ای که مورد استفاده نیست به سیستم بازگردانده شود و سپس عددی که در task manager نمایش داده می‌شود، مجددا محاسبه گردد. همچنین باید دقت داشت که این کد تنها با دسترسی مدیریتی قابل اجرا است و به همین دلیل وجود این try/catch ضروری است.

**نحوه استفاده از متد ReEvaluateWorkingSet در برنامه‌های WinForms :**
فایل Program.cs را یافته و سپس در روال رویداد گردان Idle برنامه، متد ReEvaluateWorkingSet را فراخوانی کنید (مثلا هر زمان که برنامه minimized شد اجرا می‌شود):

```
//Program.cs
namespace MemUsage
{
    static class Program
    {
        /// <summary>
```

```
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            //...

            Application.Idle += applicationIdle;
        }

        static void applicationIdle(object sender, EventArgs e)
        {
            Memory.ReEvaluateWorkingSet();
        }
    }
}
```

**نحوه استفاده از متد ReEvaluateWorkingSet در برنامه‌های WPF :**

فایل App.xaml.cs را یافته و سپس در روال رویدادگردان Deactivated برنامه، متد ReEvaluateWorkingSet را فراخوانی کنید:

```
//App.xaml.cs

public App()
{
    this.Deactivated += appDeactivated;
}

void appDeactivated(object sender, EventArgs e)
{
     Memory.ReEvaluateWorkingSet();
}
```

تاثیر آن هم قابل ملاحظه است (حداقل از لحاظ روانی!). تست کنید!

## نظرات خوانندگان

نویسنده: Ameer Taghavi
تاریخ: ۱۳۹۰/۰۸/۰۸ ۱۲:۱۱:۳۸

تست کزدم از 7524K شد 1604K
واقعا تاثیر داره!

نویسنده: Mohsen
تاریخ: ۱۳۹۰/۰۸/۰۸ ۱۴:۵۲:۰۱

مهندس در ویرایش 2 دات نت از چه متدی می توان برای اینکار استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۰۸ ۱۷:۱۴:۵۳

با دات نت 2 هم کار می‌کنه. مطابق مستندات MSDN کلاس پروسس از زمان دات نت یک اضافه شده:

Process Class Supported in: 4, 3.5, 3.0, 2.0, 1.1, 1.0

زمان دات نت 2 با توجه به اینکه WPF نبوده بنابراین بحث WinForms مطرح است و رویداد Idle هم از زمان دات نت یک وجود داشته:

Idle EventSupported in: 4, 3.5, 3.0, 2.0, 1.1, 1.0

نویسنده: مهمان
تاریخ: ۱۳۹۰/۰۸/۱۱ ۱۲:۲۴:۳۵

برای ASP.NET هم کاربرد داره؟
Silverlight چطور؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۱۱ ۱۲:۳۱:۰۳

خیر. چون هر برنامه‌ای تحت ویندوز حتما باید تحت مجوز یک کاربر اجرا شود. برنامه‌های ASP.NET یا سیلورلایت هم از این مطلب مستثنی نیستند. برای نمونه کاربر پیش فرض ASP.NET یا همان Network service (مثلا)، دسترسی مدیریتی ندارد. بنابراین قادر به اجرای کد فوق نیست.