عنوان: **فلسفه وجودی بخش finally در try catch چیست؟**
نویسنده: فانوس
تاریخ: ۱۳۹۲/۰۷/۰۹ ۱۱:۱۰
آدرس: www.dotnettips.info
برچسب‌ها: Exception, finally, exception handling, استثناء

حتما شما هم متوجه شدید که وقتی رخداد یک استثناء را با استفاده از try و catch کنترل می‌کنیم، هر چیزی که بعد از بسته شدن تگ catch بنویسیم، در هر صورت اجرا می‌شود.

```
try {
     int i=0;
     string s = "hello";
     i = Convert.ToInt32(s);
} catch (Exception ex)
{
     Console.WriteLine("Error");
}
Console.WriteLine("I am here!");
```

پس فلسفه استفاده از بخش finally چیست؟

در قسمت finally منابع تخصیص داده شده در try را آزاد می‌کنیم. کد موجود در این قسمت به هر روی اجرا می‌شود چه استثناء رخ دهد چه ندهد. البته اگر استثناء رخ داده شده در لیست استثناء هایی که برای آنها catch انجام دادیم نباشد، قسمت finally هم عمل نخواهد کرد مگر اینکه از catch به صورت سراسری استفاده کنیم.

اما مهمترین مزیتی که finally ایجاد می‌کند در این است که حتی اگر در قسمت try با استفاده از دستوراتی مثل return یا break یا continue از ادامه کد منصرف شویم و مثلا مقداری برگردانیم، چه خطا رخ دهد یا ندهد کد موجود در finally اجرا می‌شود در حالی که کد نوشته شده بعد از try catch finally فقط در صورتی اجرا می‌شود که به طور منطقی اجرای برنامه به آن نقطه برسد. اجازه بدهید با یک مثال توضیح دهم. اگر کد زیر را اجرا کنیم:

```
public static int GetMyInt()
 {
     try {
          for (int i=10;i>=0;i--)
             Console.WriteLine(10/i);
         return 1;
     } catch
     {
         Console.WriteLine("Error!");
     }
     finally {
         Console.WriteLine("ok");
     }
     Console.WriteLine("can you reach here?");
     return -1;
 }
```

برنامه خطای تقسیم بر صفر می‌دهد اما با توجه به کدی که نوشتیم، عدد -1 به خروجی خواهد رفت. در عین حال عبارت ok و can you reach here در خروجی چاپ شده است. اما حال اگر مشکل تقسیم بر صفر را حل کنیم، آیا باز هم عبارت can you reach here در خروجی چاپ خواهد شد؟

```
public static int GetMyInt()
 {
     try {
          for (int i=10;i>=1;i--)
             Console.WriteLine(10/i);
         return 1;
     } catch
     {
         Console.WriteLine("Error!");
     }
     finally {
         Console.WriteLine("ok");
     }
```

```
    Console.WriteLine("can you reach here?");
    return -1;
}
```

مشاهده می‌کنید که مقدار 1 برگردانده می‌شود و عبارت can you reach here در خروجی چاپ نمی‌شود ولی همچنان عبارت ok که در finally ذکر شده در خروجی چاپ می‌شود. یک مثال خوب استفاده از چنین وضعیتی، زمانی است که شما یک ارتباط با بانک اطلاعاتی باز می‌کنید، و نتیجه یک عملیات را با دستور return به کاربر بر می‌گردانید. مسئله این است که در این وضعیت چگونه ارتباط با دیتابیس بسته شده و منابع آزاد می‌گردند؟ اگر در حین عملیات بانک اطلاعاتی، خطایی رخ دهد یا ندهد، و شما دستور آزاد سازی منابع و بستن ارتباط را در داخل قسمت finally نوشته باشید، وقتی دستور return فراخوانی می‌شود، ابتدا منابع آزاد و سپس مقدار به خروجی بر می‌گردد.

```
public int GetUserId(string nickname)
{
    SqlConnection connection = new SqlConnection(...);
    SqlCommand command = connection.CreateCommand();
    command.CommandText = "select id from users where nickname like @nickname";
    command.Parameters.Add(new SqlParameter("@nickname", nickname));
     try {
        connection.Open();
         return Convert.ToInt32(command.ExecuteScalar());
     }
     catch(SqlException exception)
     {
     // some exception handling
        return -1;
     } finally {
         if (connection.State == ConnectionState.Open)
        connection.Close();
     }
     // if all things works, you can not reach here
}
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۰۹ ۱۱:۲۸

- اینکه شما بروز یک مشکل رو با یک عدد منفی از یک متد بازگشت می‌دید یعنی هنوز دید زبان C رو دارید. در دات نت وجود استثناءها دقیقا برای ننوشتن return 0 یا -1 و شبیه به آن هست. در این حالت برنامه خودکار در هر سطحی که باشد، ادامه‌اش متوقف میشه و نیازی نیست تا مدام خروجی یک متد رو چک کرد.

- اینکه در یک متد کانکشنی برقرار شده و بسته شده یعنی ضعف کپسوله سازی مفاهیم ADO.NET. نباید این مسایل رو مدام در تمام متدها تکرار کرد. میشه یک متد عمومی ExecSQL درست کرد بجای تکرار مدام یک سری کد.

- یک سری از اشیاء اینترفیس IDisposable رو پیاده سازی می‌کنند مثل همین شیء اتصالی که ذکر شد. در این حالت میشه از using استفاده کرد بجای try/finally و اون وقت به دوتا using نیاز خواهید داشت یعنی شیء Command هم نیاز به try/finally داره.

نویسنده: فانوس
تاریخ: ۱۳۹۲/۰۷/۰۹ ۱۱:۵۰

دوست عزیزم. من این رو به عنوان یک مثال ساده برای درک مفهوم مورد بحث نوشتم و نخواستم خیلی برای افرادی که تازه سی شارپ رو شروع می‌کنند پیچیده باشه. قواعدی که شما فرمودید کاملا درست هست. متشکرم.

نویسنده: محمد مهدی
تاریخ: ۱۳۹۲/۰۷/۱۰ ۰:۱

لطف کنید در مورد مدیریت استثناء در لایه‌های مختلف توضیح بدین.اینکه چجوری این استثناءها به لایه بالاتر یا همون اینترفیس منتقل بشه

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۰۷/۱۰ ۱۰:۱۹

«*برای افرادی که تازه سی شارپ رو شروع می‌کنند*» با تشکر از مطلبتون **به نظر من** کسی اینجا تازه سی شارپو شروع نکرده اگه میشه مطالبتونو تخصصی‌تر کنید ممنون

عنوان: **مدیریت AccessViolationException در برنامه‌های دات نت 4 به بعد**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۴ ۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: C#, C++, exception handling

فرض کنید که از یک برنامه‌ی native ویندوز برای تهیه تصاویر سایت‌ها در یک برنامه‌ی وب استفاده می‌کنید و صبح که به سایت سر زده‌اید پیام در دسترس نبودن سایت قابل مشاهده است. مشکل از کجا است؟!

یک مثال ساده

```
using System;

namespace AccessViolationExceptionSample
{
    class Program
    {
        private static unsafe void AccessViolation()
        {
            byte b = *(byte*)(8762765876);
        }

        static void Main(string[] args)
        {
            try
            {
                AccessViolation();
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
            }

            Console.WriteLine("Press a key...");
            Console.ReadKey();
        }
    }
}
```

برنامه‌ی کنسول فوق را پس از فعال سازی Allow unsafe code در قسمت تنظیمات پروژه، کامپایل کرده و سپس آن‌را خارج از VS.NET اجرا کنید. احتمالا انتظار دارید که قسمت catch این کد حداقل اجرا شود و سپس سطر «کلیدی را فشار دهید» ظاهر گردد. اما ... خیر! کل پروسه کرش کرده و هیچ پیام خطایی را دریافت نخواهید کرد. اگر به لاگ‌های ویندوز مراجعه کنید پیام زیر قابل مشاهده است:

```
System.AccessViolationException.  Attempted to read or write protected memory.
 This is often an indication that other memory is corrupt.
```

و این نوع مسایل هنگام کار با کتابخانه‌های C و C++ زیاد ممکن است رخ دهند. نمونه‌ی آن استفاده از WebControl دات نت است یا هر برنامه‌ی native دیگری. در این حالت اگر برنامه‌ی شما یک برنامه‌ی وب باشد، عملا سایت از کار افتاده‌است. به عبارتی پروسه‌ی ویندوزی آن کرش کرده و بلافاصله از طرف ویندوز خاتمه یافته است.

چرا قسمت catch اجرا نشد؟

از دات نت 4 به بعد ، زمانیکه دسترسی غیرمجازی به حافظه صورت گیرد، برای مثال دسترسی به یک pointer آزاد شده، استثنای حاصل، توسط برنامه catch نشده و اجازه داده می‌شود تا برنامه کلا کرش کند. به این نوع استثناءها Corrupted State Exceptions یا CSE گفته می‌شود. اگر نیاز به مدیریت آن‌ها توسط برنامه باشد، باید به یکی از دو طریق زیر عمل کرد:
الف) از ویژگی HandleProcessCorruptedStateExceptions بر روی متد فراخوان کتابخانه‌ی native باید استفاده شود. برای مثال در کدهای فوق خواهیم داشت:

```
[HandleProcessCorruptedStateExceptions]
  static void Main(string[] args)
  {
```

ب) و یا فایل کانفیگ برنامه را ویرایش کرده و چند سطر ذیل را به آن اضافه کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <runtime>
     <legacyCorruptedStateExceptionsPolicy enabled="true" />
  </runtime>
</configuration>
```

در این حالت مدیریت اینگونه خطاها در کل برنامه همانند برنامه‌های تا دات نت 3.5 خواهد شد.

نظرات خوانندگان

نویسنده: سوین
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۹:۴۶

با سلام
من در فایل کانفیگ یه WPF App ، تغییرات گفته شده را قرار دادم و خطای زیر رو در vs داد
.The type initializer for 'System.Windows.Application' threw an exception

و بیرون از vs اصلا اجرا نشد خیلی نیاز دارم به این مورد ، چون یه پروژه دارم که درست اجرا میشه اما بعضی مواقع برنامه کرش میکنه و نمی‌تونم catch کنم . با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۹:۵۹

- مطلب فوق بیشتر مرتبط است به استثناهای کتابخانه‌های native استفاده شده در برنامه‌های دات نت. برای سایر موارد باید در فایل App.xaml.cs موارد ذیل را بررسی کنید:

```
public partial class App
    {
        public App()
        {
            this.DispatcherUnhandledException += appDispatcherUnhandledException;
            AppDomain.CurrentDomain.UnhandledException += CurrentDomain_UnhandledException;
        }
```

+ نمونه تنظیم زیر در فایل app.config یک برنامه WPF کار می‌کند (آزمایش شد):

```
<?xml version="1.0"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
  <runtime>
     <legacyCorruptedStateExceptionsPolicy enabled="true" />
  </runtime>
</configuration>
```

نویسنده: سوین
تاریخ: ۱۳۹۲/۱۲/۰۲ ۹:۳۸

با سلام؛ من یه پروژه با WPF نوشتم اما یه ایراد داره و اونم اینه که مثلا فرم1 رو 20 بار اجرا می‌کنی خطا نمی‌ده اما بار 21 ام برنامه کرش میکنه و اصلا نمیشه catch کرد. متن خطا در Log ویندوز اینه

```
Error 01 :
Application: MyWPFApp.exe
Framework Version: v4.0.30319
Description: The process was terminated due to an unhandled exception.
Exception Info: exception code c0000005, exception address 77D52239

Error 02 :
Faulting application name: MyWPFApp.exe, version: 1.0.0.0, time stamp: 0x52d550ac
Faulting module name: ntdll.dll, version: 6.1.7601.17514, time stamp: 0x4ce7b96e
Exception code: 0xc0000005
Fault offset: 0x00032239
Faulting process id: 0xa28
Faulting application start time: 0x01cf113ae6813d88
Faulting application path: R:\Source\MyWPFApp\bin\Debug\MyWPFApp.exe
Faulting module path: C:\Windows\SYSTEM32\ntdll.dll
Report Id: 460eda62-7d33-11e3-a572-ac220bc99cf8

Information :
```

```
Fault bucket , type 0
Event Name: APPCRASH
Response: Not available
Cab Id: 0

Problem signature:
P1: MyWPFApp.exe
P2: 1.0.0.0
P3: 52d550ac
P4: ntdll.dll
P5: 6.1.7601.17514
P6: 4ce7b96e
P7: c0000005
P8: 00032239
P9:
P10:

Attached files:
C:\Users\Administrator\AppData\Local\Temp\WERE9B3.tmp.WERInternalMetadata.xml
C:\Users\Administrator\AppData\Local\Temp\WER16AC.tmp.appcompat.txt
C:\Users\Administrator\AppData\Local\Temp\WER18A1.tmp.hdmp
C:\Users\Administrator\AppData\Local\Temp\WER3BFA.tmp.mdmp

These files may be available here:
C:\Users\Administrator\AppData\Local\Microsoft\Windows\WER\ReportQueue\AppCrash_MyWPFApp.exe_125fc667a6
9fcc31c463a5e1b4032657c4ce830_cab_0ac03d3e

Analysis symbol:
Rechecking for solution: 0
Report Id: 460eda62-7d33-11e3-a572-ac220bc99cf8
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۲/۰۲ ۱۱:۳۷

از چه کامپوننتی استفاده کردی ؟ بهتره اون فایل‌های کرش دامپ dmp رو براشون ارسال کنی.

نویسنده: سوین
تاریخ: ۱۳۹۲/۱۲/۰۲ ۱۹:۲۵

با سلام
از کامپوننت شرکت‌های ثالث استفاده نکردم . آیا راه حل کلی برای پیدا کردن چنین خطاهایی وجود نداره ، اینترنت رو هم سرچ کردم اما کمک زیادی نکرد که بشه فهمید مشکل از چیه و قبل ارسال این پست 2 ساعت تمام آزمایش کردم خطا نداد اما بعضی مواقع این اتفاق می‌افته .
در ضمن این برنامه WPF App که برای اوتوماسیون اداری نوشته شده و از EF 6.2 ، قفل سخت افزاری (که بدون قفل هم این ایراد رو می‌ده)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۳ ۰:۴۶

مثال WPF ایی که AccessViolation عمدی دارد: WpfApplicationC0000005.zip
به فایل‌های App.config و App.xaml.cs آن دقت کنید.
پروژه را کامپایل کرده و خارج از VS.NET اجرا کنید. خطا را نمایش می‌دهد ولی کرش نمی‌کند.

نویسنده: سوین
تاریخ: ۱۳۹۲/۱۲/۰۳ ۹:۲۱

با سلام
من تگ startup رو به صورت زیر نوشته بودم آیا می‌تونه تاثیر داشته باشه

```
<startup useLegacyV2RuntimeActivationPolicy="true">
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
    <requiredRuntime version="v4.0.20506"/>
</startup>
```

باز تست می‌کنم ببینم چی میشه ، انشاالله که درست بشه . با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۳ ۹:۳۰

تنظیم یاد شده مربوط به تگ runtime است.

عنوان: **first chance exception چیست؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۳۱ ۱۲:۱۰
آدرس: www.dotnettips.info
گروه‌ها: WPF, exception handling

چند سال قبل یک datapicker تقویم شمسی را برای سیلورلایت تهیه کردم. بعد از آن نسخه‌ی WPF آن هم به پروژه اضافه شد . تا اینکه مدتی قبل مشکل عدم کار کردن آن در یک صفحه‌ی دیالوگ جدید در ویندوز 8 گزارش شد. در حین برطرف کردن این مشکل، مدام سطر ذیل در پنجره‌ی output ویژوال استودیو نمایش داده می‌شد:

```
A first chance exception of type 'System.ArgumentOutOfRangeException' occurred in mscorlib.dll
```

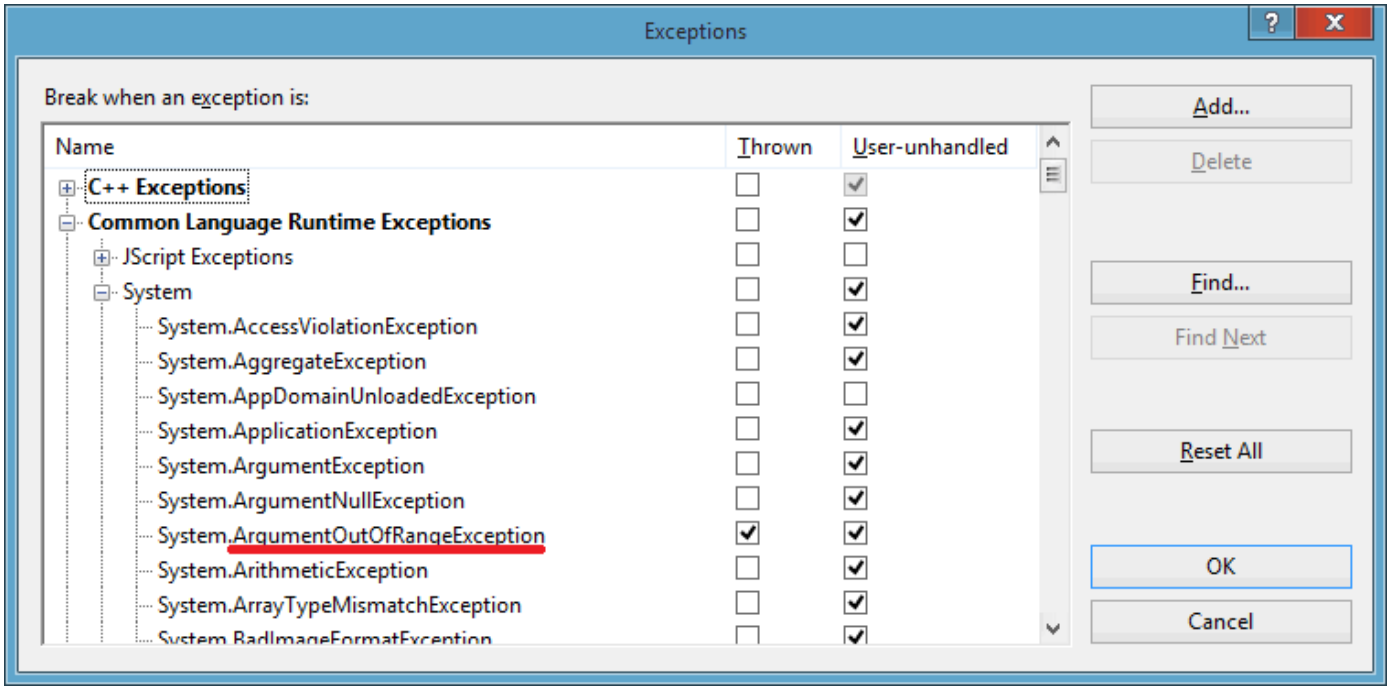
البته برنامه بدون مشکل کار می‌کرد و صفحه‌ی نمایش Exception در VS.NET ظاهر نمی‌شد.

سؤال: first chance exception چیست؟

وقتی استثنایی در یک برنامه رخ می‌دهد، به آن یک first chance exception می‌گویند. این اولین شانسی است که سیستم به شما می‌دهد تا استثنای رخ داده را مدیریت کنید. اگر کدهای برنامه یا ابزاری (یک try/catch یا دیباگر) این اولین شانس را ندید بگیرند، یک second chance exception رخ می‌دهد. این‌جا است که برنامه به احتمال زیاد خاتمه خواهد یافت.
مشاهده‌ی پیام‌های A first chance exception در پنجره‌ی output ویژوال استودیو به این معنا است که استثنایی رخ داده، اما توسط یک استثناءگردان مدیریت شده‌است. بنابراین در اکثر موارد، موضوع خاصی نیست و می‌توان از آن صرفنظر کرد.

سؤال: چگونه می‌توان منشاء اصلی پیام رخ‌دادن یک first chance exception را یافت؟

ویژوال استودیو در پنجره‌ی output، مدام پیام رخ‌دادن first chance exception را نمایش می‌دهد؛ اما واقعا کدام قطعه از کدهای برنامه سبب بروز آن شده‌اند؟ به صورت پیش فرض صفحه‌ی نمایش استثناءها در VS.NET زمانی نمایان می‌شود که استثنای رخ داده، مدیریت نشده باشد. برای فعال سازی نمایش استثناهای مدیریت شده باید تنظیمات ذیل را اعمال کرد:
- به منوی Debug | Exceptions مراجعه کنید.
- گره Common Language Runtime Exceptions را باز کنید.
- سپس گروه System آن‌را نیز باز کنید.
- در اینجا بر اساس نوع استثنایی که در پنجره‌ی output نمایش داده می‌شود، آن استثناء را یافته و Thrown آن‌را انتخاب کنید.



اینبار اگر برنامه را اجرا کنید، دقیقا محلی که سبب بروز استثنای ArgumentOutOfRangeException شده در VS.NET گزارش داده خواهد شد.

عنوان: **تولید MiniDump در حین کرش برنامه‌های دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۱۶ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: Debugging, exception handling, Crash Dump

با مطالعه‌ی سورس‌های محصولات اخیرا سورس باز شده‌ی مایکروسافت، نکات جالبی را می‌توان استخراج کرد. برای نمونه اگر سورس پروژه‌ی Orleans را بررسی کنیم، در حین بررسی اطلاعات استثناءهای رخ داده‌ی در برنامه، متد TraceLogger.CreateMiniDump نیز بکار رفته‌است. در این مطلب قصد داریم، این متد و نحوه‌ی استفاده‌ی از حاصل آن‌را بررسی کنیم.

تولید MiniDump در برنامه‌های دات نت

خلاصه‌ی روش تولید MiniDump در پروژه‌ی Orleans به صورت زیر است:

الف) حالت‌های مختلف تولید فایل دامپ که مقادیر آن قابلیت Or شدن را دارا هستند:

```
[Flags]
public enum MiniDumpType
{
    MiniDumpNormal = 0x00000000,
    MiniDumpWithDataSegs = 0x00000001,
    MiniDumpWithFullMemory = 0x00000002,
    MiniDumpWithHandleData = 0x00000004,
    MiniDumpFilterMemory = 0x00000008,
    MiniDumpScanMemory = 0x00000010,
    MiniDumpWithUnloadedModules = 0x00000020,
    MiniDumpWithIndirectlyReferencedMemory = 0x00000040,
    MiniDumpFilterModulePaths = 0x00000080,
    MiniDumpWithProcessThreadData = 0x00000100,
    MiniDumpWithPrivateReadWriteMemory = 0x00000200,
    MiniDumpWithoutOptionalData = 0x00000400,
    MiniDumpWithFullMemoryInfo = 0x00000800,
    MiniDumpWithThreadInfo = 0x00001000,
    MiniDumpWithCodeSegs = 0x00002000,
    MiniDumpWithoutManagedState = 0x00004000
}
```

ب) متد توکار ویندوز برای تولید فایل دامپ

```
public static class NativeMethods
{
    [DllImport("Dbghelp.dll")]
    public static extern bool MiniDumpWriteDump(
        IntPtr hProcess,
        int processId,
        IntPtr hFile,
        MiniDumpType dumpType,
        IntPtr exceptionParam,
        IntPtr userStreamParam,
        IntPtr callbackParam);
}
```

ج) فراخوانی متد تولید دامپ در برنامه

در اینجا نحوه‌ی استفاده از enum و متد MiniDumpWriteDump ویندوز را مشاهده می‌کنید:

```
public static class DebugInfo
{
    public static void CreateMiniDump(
        string dumpFileName, MiniDumpType dumpType = MiniDumpType.MiniDumpNormal)
    {
        using (var stream = File.Create(dumpFileName))
        {
            var process = Process.GetCurrentProcess();
            // It is safe to call DangerousGetHandle() here because the process is already crashing.
            NativeMethods.MiniDumpWriteDump(
                        process.Handle,
```

```
                            process.Id,
                            stream.SafeFileHandle.DangerousGetHandle(),
                            dumpType,
                            IntPtr.Zero,
                            IntPtr.Zero,
                            IntPtr.Zero);
        }
    }

    public static void CreateMiniDump(MiniDumpType dumpType = MiniDumpType.MiniDumpNormal)
    {
        const string dateFormat = "yyyy-MM-dd-HH-mm-ss-fffZ"; // Example: 2010-09-02-09-50-43-341Z
        var thisAssembly = Assembly.GetEntryAssembly() ?? Assembly.GetCallingAssembly();
        var dumpFileName = string.Format(@"{0}-MiniDump-{1}.dmp",
                    thisAssembly.GetName().Name,
                    DateTime.UtcNow.ToString(dateFormat, CultureInfo.InvariantCulture));

        var path = Path.Combine(getApplicationPath(), dumpFileName);
        CreateMiniDump(path, dumpType);
    }

    private static string getApplicationPath()
    {
        return HttpContext.Current != null ?
            HttpRuntime.AppDomainAppPath :
            Path.GetDirectoryName(Assembly.GetExecutingAssembly().Location);
    }
}
```

متد MiniDumpWriteDump نیاز به اطلاعات پروسه‌ی جاری، به همراه هندل فایلی که قرار است اطلاعات را در آن بنویسد، دارد. همچنین dump type آن نیز می‌تواند ترکیبی از مقادیر enum مرتبط باشد.

یک مثال:

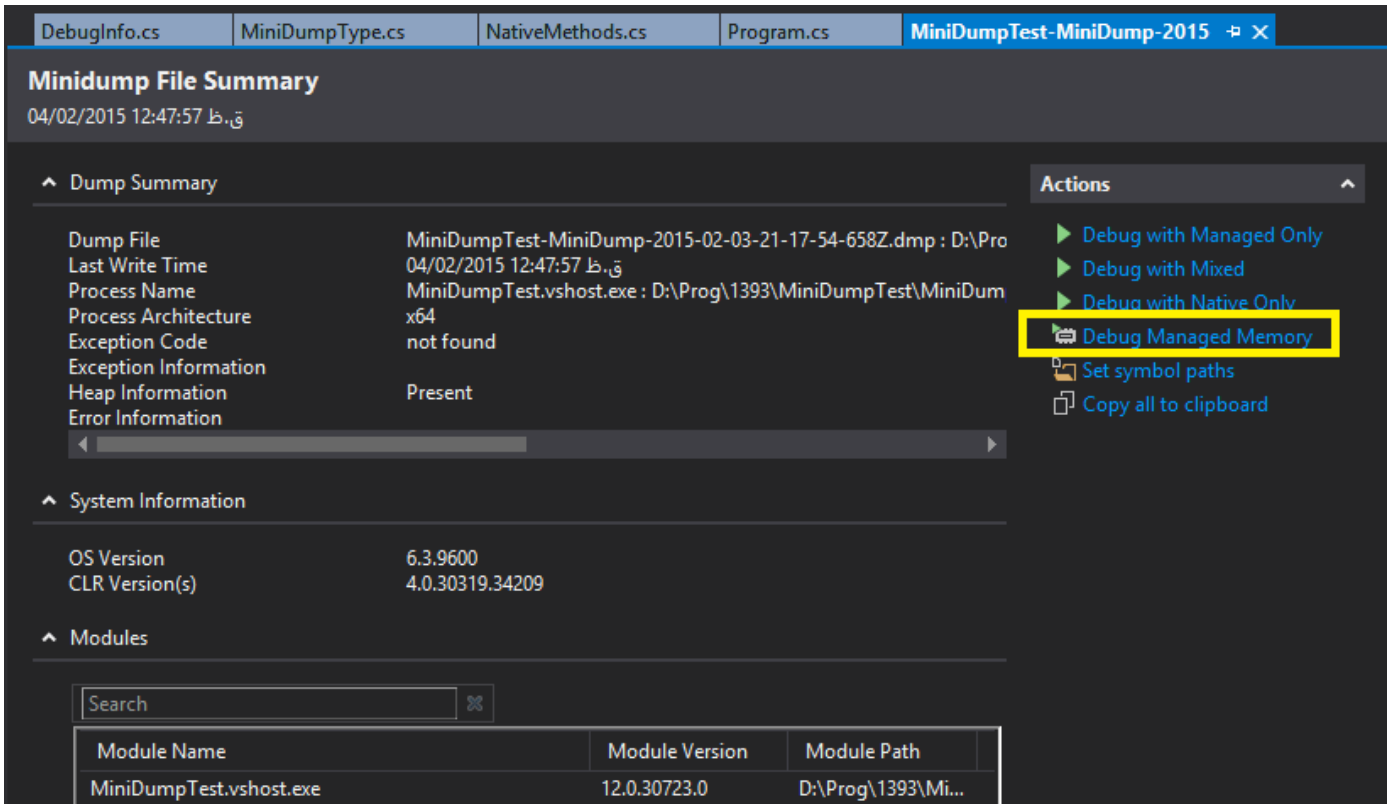
```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            var zero = 0;
            Console.WriteLine(1 / zero);
        }
        catch (Exception ex)
        {
            Console.Write(ex);
            DebugInfo.CreateMiniDump(dumpType:
                                MiniDumpType.MiniDumpNormal |
                                MiniDumpType.MiniDumpWithPrivateReadWriteMemory |
                                MiniDumpType.MiniDumpWithDataSegs |
                                MiniDumpType.MiniDumpWithHandleData |
                                MiniDumpType.MiniDumpWithFullMemoryInfo |
                                MiniDumpType.MiniDumpWithThreadInfo |
                                MiniDumpType.MiniDumpWithUnloadedModules);

            throw;
        }
    }
}
```

در اینجا نحوه‌ی فراخوانی متد CreateMiniDump را در حین کرش برنامه مشاهده می‌کنید. پارامترهای اضافی دیگر سبب خواهند شد تا اطلاعات بیشتری از حافظه‌ی جاری سیستم، در دامپ نهایی قرار گیرند. اگر پس از اجرای برنامه، به پوشه‌ی bin\debug مراجعه کنید، فایل dmp تولیدی را مشاهده خواهید کرد.

نحوه‌ی بررسی فایل‌های dump

الف) با استفاده از Visual studio 2013

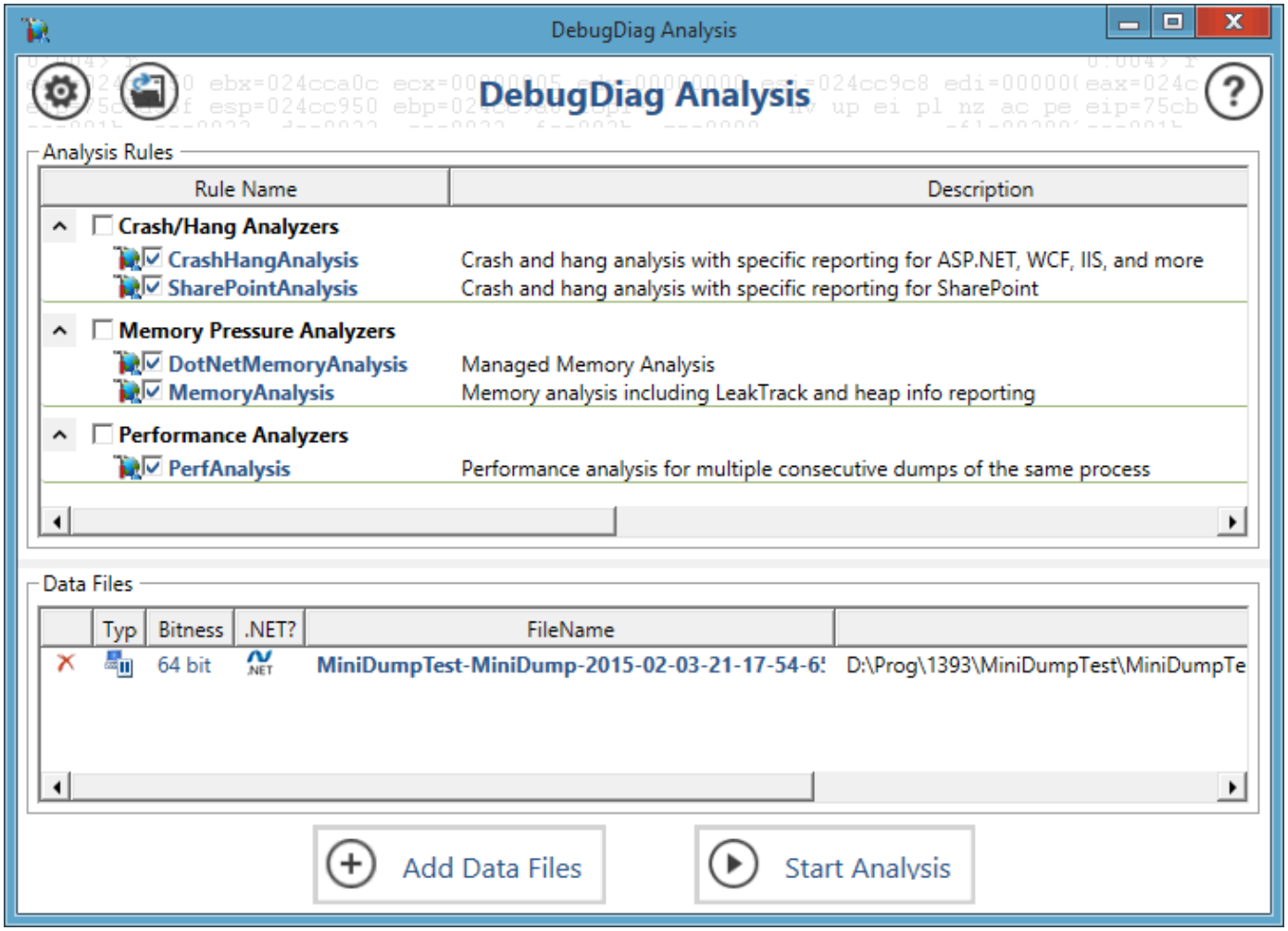
از به روز رسانی سوم VS 2013 به بعد، فایل‌های dump را می‌توان داخل خود VS.NET نیز آنالیز کرد (^ و ^ و ^). برای نمونه تصویر ذیل، حاصل کشودن فایل کرش مثال فوق است:



در اینجا اگر بر روی لینک debug managed memory کلیک کنید، پس از چند لحظه، آنالیز کامل اشیاء موجود در حافظه را در حین تهیه‌ی دامپ تولیدی، می‌توان مشاهده کرد. این مورد برای آنالیز نشتی‌های حافظه‌ی یک برنامه بسیار مفید است.

ب) استفاده از برنامه‌ی Debug Diagnostic Tool

برنامه‌ی Debug Diagnostic Tool را از اینجا می‌توانید دریافت کنید. این برنامه نیز قابلیت آنالیز فایل‌های دامپ را داشته و اطلاعات بیشتری را پس از آنالیز ارائه می‌دهد.



برای نمونه پس از آنالیز فایل دامپ تهیه شده توسط این برنامه، خروجی ذیل حاصل می‌شود:

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

MiniDumpTest.zip