عنوان: **کامپایل پویای کد در دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۲۵ ۱۶:۱۷:۰۰
آدرس: www.dotnettips.info
گروه‌ها: #C

در دات نت فریم ورک امکان کامپایل پویای یک قطعه کد دریافت شده از یک رشته، توسط فضای نام CodeDom مهیا است که قدرت قابل توجهی را در اختیار برنامه نویس قرار می‌دهد.

مثال یک:
رشته زیر را کامپایل کرده و تبدیل به یک فایل exe کنید:

```
string source =
           @"
               namespace Foo
               {
                    public class Bar
                    {
                        static void Main(string[] args)
                        {
                            Bar.SayHello();
                        }

                        public static void SayHello()
                        {
                            System.Console.WriteLine(""Hello World"");
                        }
                    }
               }
            ";
```

روش انجام کار به همراه توضیحات مربوطه به صورت کامنت:

```
using System;
using System.Collections.Generic;
//دو فضای نامی که برای این منظور اضافه شده‌اند
using Microsoft.CSharp;
using System.CodeDom.Compiler;

namespace compilerTest
{
   class Program
   {
       static void compileIt1()
       {
           //سورس کد ما جهت کامپایل
           string source =
           @"
               namespace Foo
               {
                    public class Bar
                    {
                        static void Main(string[] args)
                        {
                            Bar.SayHello();
                        }

                        public static void SayHello()
                        {
                            System.Console.WriteLine(""Hello World"");
                        }
                    }
               }
            ";

           //تعیین نگارش کامپایلر مورد استفاده
           Dictionary<string, string> providerOptions = new Dictionary<string, string>
                {
                    {"CompilerVersion", "v3.5"}
```

```
            };
            //تعیین اینکه کد ما سی شارپ است
            CSharpCodeProvider provider = new CSharpCodeProvider(providerOptions);

            //تعیین اینکه خروجی یک فایل اجرایی است بعلاوه مشخص سازی محل ذخیره سازی فایل نهایی
            CompilerParameters compilerParams = new CompilerParameters
            {
                OutputAssembly = "D:\\Foo.EXE",
                GenerateExecutable = true
            };

            //عملیات کامپایل در اینجا صورت می‌گیرد
            CompilerResults results = provider.CompileAssemblyFromSource(compilerParams, source);

            //اگر خطایی وجود داشته باشد نمایش داده خواهد شد
            Console.WriteLine("Number of Errors: {0}", results.Errors.Count);
            foreach (CompilerError err in results.Errors)
            {
                Console.WriteLine("ERROR {0}", err.ErrorText);
            }
        }

        static void Main(string[] args)
        {
            compileIt1();

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

مثال 2:

کد مورد نظر را به صورت یک فایل dll کامپایل کنید.

برای این منظور تمامی مراحل مانند قبل است فقط GenerateExecutable ذکر شده به false تنظیم شده و نام خروجی نیز به foo.dll باید تنظیم شود.

مثال 3:

کد مورد نظر را در حافظه کامپایل کرده (خروجی dll یا exe نمی‌خواهیم)، سپس متد SayHello آن را به صورت پویا فراخوانی نموده و خروجی را نمایش دهید.

در این حالت روش کار همانند مثال 1 است با این تفاوت که GenerateInMemory = true و GenerateExecutable = false تنظیم می‌شوند. همچنین جهت دسترسی به متد کلاس ذکر شده، از قابلیت‌های ریفلکشن موجود در دات نت فریم ورک استفاده خواهد شد.

```
using System;
using System.Collections.Generic;
using Microsoft.CSharp;
using System.CodeDom.Compiler;
using System.Reflection;

namespace compilerTest
{
   class Program
   {
       static void compileIt2()
       {
           //سورس کد ما جهت کامپایل
           string source =
           @"
               namespace Foo
               {
                    public class Bar
                    {
                        static void Main(string[] args)
                        {
                            Bar.SayHello();
                        }

                        public static void SayHello()
                        {
```

```
                            System.Console.WriteLine(""Hello World"");
                        }
                    }
                }
             ";

            //تعیین نگارش کامپایلر مورد استفاده
            Dictionary<string, string> providerOptions = new Dictionary<string, string>
                 {
                     {"CompilerVersion", "v3.5"}
                 };
            //تعیین اینکه کد ما سی شارپ است
            CSharpCodeProvider provider = new CSharpCodeProvider(providerOptions);

            //نحوه تعیین مشخص سازی کامپایل در حافظه
            CompilerParameters compilerParams = new CompilerParameters
            {
                GenerateInMemory = true,
                GenerateExecutable = false
            };

            //عملیات کامپایل در اینجا صورت می‌گیرد
            CompilerResults results = provider.CompileAssemblyFromSource(compilerParams, source);

            // اگر خطایی در کامپایل وجود نداشت متد دلخواه را فراخوانی می‌کنیم
            if (results.Errors.Count == 0)
            {
                //استفاده از ریفلکشن برای دسترسی به متد و فراخوانی آن
                Type type = results.CompiledAssembly.GetType("Foo.Bar");
                MethodInfo method = type.GetMethod("SayHello");
                method.Invoke(null, null);
            }
        }


        static void Main(string[] args)
        {
            compileIt2();

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

نکته: نحوه‌ی استفاده از اسمبلی‌های دیگر در رشته سورس کد خود

مثال:

اگر رشته سورس ما به صورت زیر بوده و از اسمبلی System.Drawing.Dll نیز کمک گرفته باشد،

```
string source =
            @"
              namespace Foo
              {

                  public class Bar
                  {
                      static void Main(string[] args)
                      {
                          Bar.SayHello();
                      }

                      public static void SayHello()
                      {
                          System.Console.WriteLine(""Hello World"");
                          var r = new System.Drawing.Rectangle(0,0,100,100);
                          System.Console.WriteLine(r);
                      }
                  }
              }
             ";
```

هنگام کامپایل آن توسط روش مثال یک، با خطای زیر مواجه خواهیم شد.

```
Number of Errors: 1
```

```
ERROR The type or namespace name 'Drawing' does not exist in the namespace 'System' (are you missing an assembly reference?)
```

برای رفع این مشکل و معرفی این اسمبلی، سطر زیر باید پس از تعریف compilerParams اضافه شود.

```
compilerParams.ReferencedAssemblies.Add("System.Drawing.Dll");
```

اکنون کد کامپایل شده و مشکلی نخواهد داشت.
نمونه‌ای دیگر از این دست، استفاده از LINQ می‌باشد. در این حالت اسمبلی System.Core.Dll نیز به روش ذکر شده باید معرفی گردد تا مشکلی در کامپایل کد رخ ندهد.

کاربردها:
1- استفاده در ابزارهای تولید کد (برای مثال در برنامه Linqer از این قابلیت استفاده می‌شود)
2- استفاده‌های امنیتی (ایجاد روش‌های تولید یک سریال به صورت پویا و کامپایل پویای کد مربوطه در حافظه‌ای محافظت شده)
3- استفاده جهت مقاصد محاسباتی پیشرفته
4- دادن اجازه‌ی کد نویسی به کاربران برنامه‌ی خود (شبیه به سیستم‌های ماکرو و اسکریپت نویسی موجود)
و ...

نظرات خوانندگان

نویسنده: میثم جوادی
تاریخ: ۱۳۸۸/۰۶/۲۵ ۲۰:۲۳:۵۹

تو VB هم همچین چیزی به اسم eval بود و به خاطر این من دنبال یه همچین معادلی تو C# بودم که پیدا نکردم!(البته تو برنامه نویس یکی از قدیمی ها گفت قراره تو 4 بیاد دیگه دنبالش نگشتم.)

ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۲۵ ۲۰:۳۳:۴۹

البته این فراتر است از eval وی بی و شما کد کامل رو می‌تونید توسط آن کامپایل کنید.
برای کامپایل سورس از نوع VB.Net هم VBCodeProvider بجای CSharpCodeProvider در مثال‌های بالا قابل استفاده است.

نویسنده: Mohammad Shams Javi
تاریخ: ۱۳۸۸/۰۶/۲۵ ۲۲:۰۹:۱۴

سلام

خیلی مفید و جالب بود، خصوصا که قابلیتها و امکانات Reflection و CodeDom، بحث روز امنیت نرم‌افزارهای .Net مثل انواع روشهای SelfObfuscation و SelfPatching و ... هستند.

نویسنده: Kianoosh
تاریخ: ۱۳۸۸/۰۶/۲۶ ۰۴:۴۹:۵۶

با سلام

مقاله خیلی مفید و زیبائی بود. به زیبائی در این مقله در باره تولید خودکار کد و استفاده از reflection صحبت کرده بودید.

مثالهای انتهای مقاله هم بسیار جالب بودند.
لطفا باز هم درباره reflection و CodDom مطالب بیشتری بنویسید.
ممنون.

نویسنده: ..::A-3BT::..
تاریخ: ۱۳۸۸/۰۹/۰۳ ۲۱:۱۰:۵۷

ممنون ، جناب نصیری من یه مشکلی دارم اینکه یه اسمبلی که بصورت جداست می خوام رفرنس کنم بهش ولی نتونستم میشه یه راهنمایی بکنید؟
بسیار ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۰۴ ۱۰:۴۳:۴۹

سلام
شما در حین کامپایل اولیه در قسمت ReferencedAssemblies.Add مسیر کامل اسمبلی مورد نظر را ذکر کنید تا عملیات کامپایل با موفقیت به پایان برسد.
هنگام اجرای پویای کد، اسمبلی مورد نظر یا باید در GAC باشد یا کنار فایل اجرایی اصلی یا سایر مسیرهای استانداردی که دات

نت فریم ورک در حین اجرا به دنبال اسمبلی‌ها می‌گردد.

نویسنده: reza
تاریخ: ۱۳۸۸/۰۹/۱۷ ۱۸:۳۰:۳۹

سلام
آقای نصیری من چندتا مشکل با این موضوع دارم اگه کمکم کنید ممنون میشم.
-1 من زمان اجرای فایل تولید شده یه فرم ظاهر بشه در حالی که اضافه بر فرم یه صفحه Command هم باز میشه! چطور میشه کاری کرد که این صفحه باز نشه؟
-2 چطور میشه برای فایل های اجرایی تولید شده آیکون در نظر گرفت.
-3 توی کدی که قرار است توسط CodeDom کامپایل شود یه متغیر از نوع آرایه ای از Byte دارم که باید توسط برنامه اصلی مقدار دهی شود روشی که استفاده کردم به اینصورته
";{" + [byte[] a = new byte[]{"+MyByteArray[0] + "," + MyByteArray[1"
ولی این روش هم سرعت برنامه رو پایین میاره هم محدودیت برای تعداد کاراکتری که باید کامپایل شوند ایجاد میکند. شما چه روشی رو برای مقدار دهی این آرایه پیشنهاد میکنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۱۷ ۱۹:۴۶:۳۰

-1 احتمالا برای اجرا از کلاس Process مربوط به فضای نام System.Diagnostics کمک می‌گیرد. اگر اینطور است باید خاصیت process.StartInfo.CreateNoWindow به true تنظیم شود.
-2 متد parameters.EmbeddedResources.Add هم موجود است. کمی در مورد آن تحقیق کنید.
-3 سرعت این روش فقط در حد زمان انجام کامپایل کامل، کند است؛ مابقی آن تفاوتی با اجرای یک برنامه واقعی ندارد. ضمنا محدودیتی هم من ندیدم. محدودیت‌های آن همان محدودیت‌های برای مثال کامپایلر سی شارپ است. مثلا یک سطر نباید از 16777214 کاراکتر بیشتر باشد و امثال آن.
ضمنا استفاده از + هنگام چسباندن رشته‌ها به هم در حجم کم تاثیر آنچنانی روی کارآیی ندارد. ولی اگر تعداد زیاد است بهتر است از StringBuilder استفاده شود.

نویسنده: ..::A-3BT::..
تاریخ: ۱۳۸۸/۰۹/۱۸ ۱۷:۵۲:۴۲

راستی گزینه هایی که برای Optimize کردن کد کامپایل شده هست ، تو این روش هم قابل استفاده است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۱۸ ۱۸:۵۷:۰۳

بله. خاصیت CompilerOptions آن‌را مساوی "optimize/" قرار دهید و یا سایر موارد مورد نظر.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۰۹/۲۲ ۱۵:۳۱

اوایل کارم با سی شارپ بود ، یک پروژه توی codeproject قرار گرفته بود که یک برنامه برای ساخت slideshow با تمامی امکانات لازم ساخته بود که خروجیش هم exe بود
وارد کردن و ترتیب تصاویر و موسیقی و تکرار و حرکت خودکار یا با کلیک ماوس تصاویر و تنظیمات دیگه
ذخیره کار به صورت پروژه و بازیابی اون به صورت serialization
و همینطور کد خروجی exe
نحوه کدنویسی شکیل و ساخت یافتش به قدری کامل و شیوا بود که باعث شد بیش از پیش به این زبان هم علاقه مند بشم و هم به برنامه نویسی
خدا طرف رو خیر بده ، یه زندگی رو با این کدش دگرگون کرد