عنوان: **آشنایی با CLR: قسمت چهاردهم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۲۹ ۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: CLR, .NET Framework

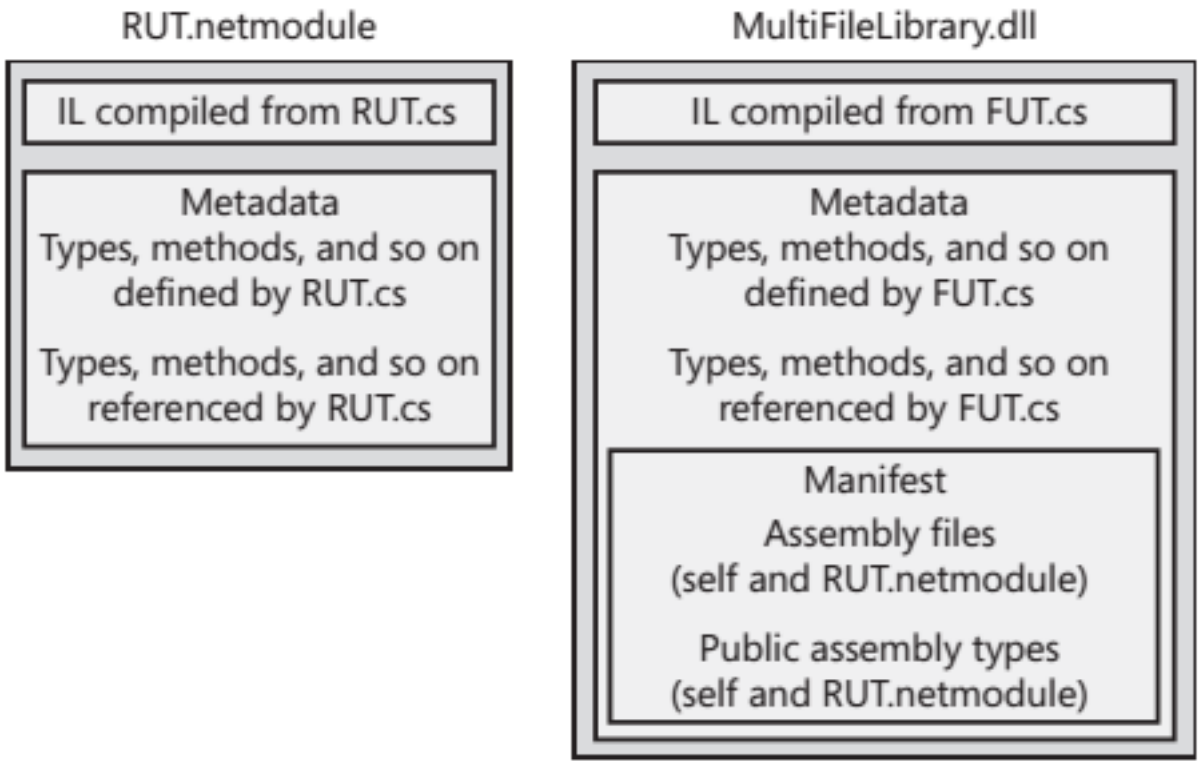
در ادامه قسمت قبلی روش‌های زیادی جهت اضافه شدن یک ماژول به یک اسمبلی وجود دارند. اگر شما از کامپایلر سی‌شارپ برای ساخت یک فایل PE با جدول مانیفست استفاده می‌کنید، می‌توانید از سوئیچ /AddModule استفاده کنید. برای اینکه بدانیم چگونه می‌توان یک اسمبلی چند فایله ساخت بیاید فرض کنیم که دو فایل سورس کد با مشخصات زیر داریم: RUT.cs: این سورس شامل کدهایی است که به ندرت در برنامه استفاده می‌شود. FUT.cs: این سورس شامل کدهایی است که به طور مکرر مورد استفاده قرار می‌گیرد.
ابتدا به صورت زیر کد سورسی را که به ندرت استفاده می‌شود، به عنوان یک ماژول جداگانه کامپایل می‌کنم:

```
csc /t:module RUT.cs
```

اجرای این خط سبب ایجاد یک فایل به نام RUT.netmodule می‌گردد که یک DLL استاندارد است؛ ولی CLR به تنهایی توانایی بارگیری آن را ندارد. دفعه‌ی بعد سورس کدی را که مکرر استفاده می‌شود، به صورت یک ماژول کامپایل می‌کنیم و از آنجائیکه این ماژول استفاده‌ی زیادی دارد، آن را نگهدارنده‌ی جدول مانیفست معرفی می‌کنیم و به این دلیل که این ماژول نماینده‌ی کل اسمبلی است، نام خروجی آن را به جای FUT.dll به MultiFileLibrary.dll تغییر می‌دهیم:

```
csc /out:MultiFileLibrary.dll /t:library /addmodule:RUT.netmodule FUT.cs
```

خط بالا به علت سوئیچ \t:library فایل MultiFileLibrary.dll را ایجاد می‌کند. این فایل شامل جدول متادیتای مانیفست می‌شود و سوئیچ به آن می‌گوید که باید ماژول RUT.netmodule را جزئی از اسمبلی بداند. این سوئیچ به کامپایلر اعلام می‌کند که ارجاع این فایل در جدول FileDef و ExportedTypesDef ثبت شود. بعد از اتمام عملیات کامپایل، مطابق شکل زیر دو فایل ایجاد می‌شود که فایل سمت راست شامل جدول مانیفست است. فایل RUT.netmodule شامل کد IL و جداول متادیتاهای مربوط به خواص و رویدادها و مواردی از این قبیل است که در این ماژول یافت می‌شود. فایل بعدی MultiFileLibrary.dll هست که شامل کد IL کد FUT.CS می‌شوذ بعلاوه جداول متادیتا مثل ماژول قبلی و جدول متادیتای مانیفست که باعث می‌شود به عنوان یک اسمبلی شناخته شود.



البته توجه داشته باشید که جدول مانیفست ارجاعی به نوع‌های عمومی استخراج شده داخل فایل خودش ندارد، زیرا که در جداول اختصاصی خودش موجود است و در ذخیره سازی صرفه جویی می‌گردد.
بعد از اینکه MultiFileLibrary.dll ساخته شد، به منظور آزمایش کردن جداول متادیتا می‌توانید از ابزار ILDasm.exe استفاده کنید تا ارجاع به فایل RUT.netmodule به شما ثابت شود. آنچه در زیر می‌بینید نمایی از جداول FileDef و ExportedTypesDef است:

```
File #1 (26000001)
-------------------------------------------------------
Token: 0x26000001
Name : RUT.netmodule
HashValue Blob : e6 e6 df 62 2c a1 2c 59 97 65 0f 21 44 10 15 96 f2 7e db c2
Flags : [ContainsMetaData] (00000000)


ExportedType #1 (27000001)
-------------------------------------------------------
Token: 0x27000001
Name: ARarelyUsedType
Implementation token: 0x26000001
TypeDef token: 0x02000002
Flags : [Public] [AutoLayout] [Class] [Sealed] [AnsiClass]
[BeforeFieldInit](00100101)
```

همانطور که در بالا می‌بینید فایل RUT.netmodule با شناسه‌ی (توکن) 0x26000001 به عنوان بخشی از اسمبلی شناخته می‌شود و به نوع کد IL آن اشاره می‌کند.

قابل توجه افراد کنجکاو: توکن‌های جداول متا، مقادیر 4 بایتی است که بایت پر ارزش آن اشاره می‌کند که برای یافتن آن باید به چه جدولی ارجاع کرد. مقادیر زیر این نکته را روشن می‌کند که هر کد ابتدایی به چه جدولی اشاره می‌کند:

| | |
|---|---|
| **TypeRef** | **0x01** |
| **TypeDef** | **0x02** |
| **AssemblyRef** | **0x23** |
| **File**<br>*file definition* | **0x26** |
| **ExportedType** | **0x27** |

برای دیدن لیست کاملی از این کدها فایل Corhdr.h را که به همراه فریم ورک دات نت نصب می‌شود، مطالعه فرمایید. سه بایت باقیمانده هم بر اساس جدولی که به آن ارجاع شده است مشخص می‌گردد؛ مثلا در مثال بالا کد 0x26000001 به اولین سطر جدول File اشاره می‌کند. برای اکثر جدول‌ها شماره گذاری سطرها از عدد 1 آغاز می‌شود نه صفر یا برای برای جداول TypeDef عموما از عدد 2 آغاز می‌شود.
برای اجرای اسمبلی، کامپایلر نیاز دارد که همه‌ی فایل‌های اسمبلی، نصب شده و قابل دسترس باشند و در صورتیکه شما فایل RUT.netmodule را حذف کنید کامپایلر سی شارپ خطای زیر را صادر می‌کند:

```
fatal error CS0009: Metadata file 'C:\ MultiFileLibrary.dll' could not be opened—'Error importing
module 'RUT.netmodule' of assembly 'C:\ MultiFileLibrary.dll'—The system cannot find the file
specified'
```

و این خطا بدین معنی است که برای ساخت اسمبلی باید تمامی فایل‌ها حاضر و مهیا باشند. هر کد کلاینتی که اجرا می‌شود آن متد را صدا می‌زنند. موقعی که یک متد برای اولین بار فراخوانی می‌شود، CLR عملیات شناسایی جهت شناسایی ارجاعات آن در

پارامترها، نوع خروجی متد و متغیرهای محلی آن اجرا می‌کند. سپس تلاش می‌کند تا فایل اسمبلی ارجاع شده را که شامل مانیفست هست، بار کند. اگر نوعی که لازم داریم در همین فایل متد وجود داشته باشد، اجرای عملیات را به سمت آن آغاز می‌کند ولی اگر جدول مانیفست ارجاع را به فایل دیگری بدهد، آن فایل در حافظه بار شده و سپس آن نوع را در دسترس قرار می‌دهد. خطوط بالا این نکته را روشن می‌کند که فایل‌های اسمبلی را تنها موقعی در حافظه بار میکند که ارجاعی از نوع موجود در آن صدا زده شده باشد؛ یعنی اینکه در زمان اجرای برنامه، لازم نیست که همه‌ی فایل‌ها حاضر و مهیا باشند.