عنوان: **آشنایی با CLR: قسمت دوازدهم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۵/۲۷ ۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: CLR, .NET Framework

متادیتاها شامل بلوکی از داده‌های باینری هستند که شامل چندین جدول شده و جدول‌ها نیز به سه دسته تقسیم می‌شوند:

جداول تعاریف Definition Table
جداول ارجاع References Table
جداول manifest

جداول تعریف
جدول زیر تعدادی از جداول تعریف‌ها را توضیح می‌دهد:

ModuleDef	شامل آدرس یا مدخلی است که ماژول در آن تعریف شده است. این آدرس شامل نام ماژول به همراه پسوند آن است؛ بدون ذکر مسیر. در صورتی که کامپایل به صورت GUID انجام گرفته باشد، Version ID ماژول هم همراه آن‌ها خواهد بود. در صورتیکه نام فایل تغییر کند، این جدول باز نام اصلی ماژول را به همراه خواهد داشت. هر چند تغییر نام فایل به شدت رد شده و ممکن است باعث شود CLR نتواند در زمان اجرا آن را پیدا کند.
TypeDef	شامل یک مدخل ورودی برای هر نوعی است که تعریف شده است. هر آدرس ورودی شامل نام نوع ، پرچمها (همان مجوزهای public و private و ...) می‌باشد. همچنین شامل اندیس هایی به متدها است که شامل جدول MethodDef می‌باشند یا فیلدهایی که شامل جدول FieldDef می‌باشند و الی آخر...
MethodDef	شامل آدرسی برای هر متد تعریف شده در ماژول است که شامل نام متد و پرچم هاست. همچنین شامل امضای متد و نقطه‌ی آغاز کد IL آن در ماژول هم می‌شود و آن آدرس هم میتواند ارجاعی به جدول ParamDef جهت شناسایی پارامترها باشد.
FieldDef	شامل اطلاعاتی در مورد فیلدهاست که این اطلاعات ، پرچم، نام و نوع فیلد را مشخص می‌کنند.
ParamDef	حاوی اطلاعات پارامتر متدهاست که این اطلاعات شامل پرچم‌ها (in , out ,retval) ، نوع و نام است.
PropertyDef	برای هر پراپرتی یا خصوصیت، شامل یک آدرس است که شامل نام، نوع و پرچم می‌شود.
EventDef	برای هر رویداد شامل یک آدرس است که این آدرس شامل نام و نوع است.

جداول ارجاعی
موقعی که کد شما کامپایل می‌شود، اگر شما به اسمبلی دیگری ارجاع داشته باشید، از جداول ارجاع کمک گرفته می‌شود که در

جدول زیر تعدادی از این جداول فهرست شده‌اند:

AssemblyRef	شامل آدرس اسمبلی است که ماژولی به آن ارجاع داده است و این آدرس شامل اطلاعات ضروری جهت اتصال به اسمبلی می‌شود و این اطلاعات شامل نام اسمبلی (بدون ذکر پسوند و مسیر)، شماره نسخه اسمبلی، سیستم فرهنگی و منطقه‌ای تعیین شده اسمبلی culture و یک کلید عمومی که عموما توسط ناشر ایجاد می‌گردد که هویت ناشر آن اسمبلی را مشخص می‌کند. هر آدرس شامل یک پرچم و یک کد هش هست که بری ارزیابی از صحت و بی خطا بودن بیت‌های اسمبلی ارجاع شده Checksum استفاده می‌شود.
ModuleRef	شامل یک آدرس ورودی به هدر PE ماژول است به نوع‌های پیاده سازی شده آن ماژول در آن اسمبلی. هر آدرس شامل نام فایل و پسوند آن بدون ذکر مسیر است. این جدول برای اتصال به نوع‌هایی استفاده می‌شود که در یک ماژول متفاوت از ماژول اسمبلی صدا زده شده پیاده سازی شده است.
TypeRef	شامل یک آدرس یا ورودی برای هر نوعی است که توسط ماژول ارجاع داده شده است. هر آدرس شامل نام نوع و آدرسی است که نوع در آن جا قرار دارد. اگر این نوع داخل نوع دیگری پیاده سازی شود، ارجاعات به سمت یک جدول TypeDef خواهد بود. اگر نوع داخل همان ماژول تعریف شده باشد، ارجاع به سمت جدول ModuleDef خواهد بود و اگر نوع در ماژول دیگری از آن اسمبلی پیاده سازی شده باشد، ارجاع به سمت یک جدول ModuleRef خواهد بود و اگر نوع در یک اسمبلی جداگانه تعریف شده باشد، ارجاع به جدول AssemblyRef خواهد بود.
MemberRef	شامل یک آدرس ورودی برای هر عضو (فیلد و متدها و حتی پراپرتی و رویدادها) است که توسط آن آن ماژول ارجاع شده باشد. هر آدرس شامل نام عضو، امضاء و یک اشاره‌گر به جدول TypeRef است، برای نوع‌هایی که به تعریف عضو پرداخته‌اند.

البته جداولی که در بالا هستند فقط تعدادی از آن جداول هستند، ولی قصد ما تنها یک آشنایی کلی با جداول هر قسمت بود. در مورد جداول manifest بعدتر صحبت می‌کنیم.
ابزارهای متنوع و زیادی هستند که برای بررسی و آزمایش متادیتاها استفاده می‌شوند. یکی از این ابزارها ILDasm.exe می‌باشد. برای دیدن متادیتاهای یک فایل اجرایی فرمان زیر را صادر کنید:

```
ILDasm Program.exe
```

صدور فرمان بالا باعث اجرای ILDasm و بارگزاری اسمبلی‌های program.exe می‌شود. برای مشاهده‌ی اطلاعات جداول متا به صورت شکیل و قابل خواندن برای انسان، در منوی برنامه، مسیر زیر را طی کنید:

```
View/MetaInfo/Show
```

با طی کردن گزینه‌های بالا، اطلاعات به صورت زیر نمایش داده می‌شوند:

```
==========================================================
ScopeName : Program.exe
```

```
MVID : {CA73FFE8-0D42-4610-A8D3-9276195C35AA}
============================================================
Global functions
------------------------------------------------------------
Global fields
------------------------------------------------------------
Global MemberRefs
------------------------------------------------------------
TypeDef #1 (02000002)
------------------------------------------------------------
TypDefName: Program (02000002)
Flags : [Public] [AutoLayout] [Class] [Sealed] [AnsiClass]
[BeforeFieldInit] (00100101)
Extends : 01000001 [TypeRef] System.Object
Method #1 (06000001) [ENTRYPOINT]
------------------------------------------------------------
MethodName: Main (06000001)
Flags : [Public] [Static] [HideBySig] [ReuseSlot] (00000096)
RVA : 0x00002050
ImplFlags : [IL] [Managed] (00000000)
CallCnvntn: [DEFAULT]
ReturnType: Void
No arguments.
Method #2 (06000002)
------------------------------------------------------------
MethodName: .ctor (06000002)
Flags : [Public] [HideBySig] [ReuseSlot] [SpecialName]
[RTSpecialName] [.ctor] (00001886)
RVA : 0x0000205c
ImplFlags : [IL] [Managed] (00000000)
CallCnvntn: [DEFAULT]
hasThis
ReturnType: Void
No arguments.
TypeRef #1 (01000001)
------------------------------------------------------------
Token: 0x01000001
ResolutionScope: 0x23000001
TypeRefName: System.Object
MemberRef #1 (0a000004)
------------------------------------------------------------
Member: (0a000004) .ctor:
CallCnvntn: [DEFAULT]
hasThis
ReturnType: Void
No arguments.
TypeRef #2 (01000002)
------------------------------------------------------------
Token: 0x01000002
ResolutionScope: 0x23000001
TypeRefName: System.Runtime.CompilerServices.CompilationRelaxationsAttribute
MemberRef #1 (0a000001)
------------------------------------------------------------
Member: (0a000001) .ctor:
CallCnvntn: [DEFAULT]
hasThis
ReturnType: Void
1 Arguments
Argument #1: I4
TypeRef #3 (01000003)
------------------------------------------------------------
Token: 0x01000003
ResolutionScope: 0x23000001
TypeRefName: System.Runtime.CompilerServices.RuntimeCompatibilityAttribute
MemberRef #1 (0a000002)
------------------------------------------------------------
Member: (0a000002) .ctor:
CallCnvntn: [DEFAULT]
hasThis
ReturnType: Void
No arguments.
TypeRef #4 (01000004)
------------------------------------------------------------
Token: 0x01000004
ResolutionScope: 0x23000001
TypeRefName: System.Console
MemberRef #1 (0a000003)
------------------------------------------------------------
Member: (0a000003) WriteLine:
CallCnvntn: [DEFAULT]
ReturnType: Void
```

```
1 Arguments
Argument #1: String
Assembly
-------------------------------------------------------
Token: 0x20000001
Name : Program
Public Key :
Hash Algorithm : 0x00008004
Version: 0.0.0.0
Major Version: 0x00000000
Minor Version: 0x00000000
Build Number: 0x00000000
Revision Number: 0x00000000
Locale: <null>
Flags : [none] (00000000)
CustomAttribute #1 (0c000001)
-------------------------------------------------------
CustomAttribute Type: 0a000001
CustomAttributeName:
System.Runtime.CompilerServices.CompilationRelaxationsAttribute ::
instance void .ctor(int32)
Length: 8
Value : 01 00 08 00 00 00 00 00 > <
ctor args: (8)
CustomAttribute #2 (0c000002)
-------------------------------------------------------
CustomAttribute Type: 0a000002
CustomAttributeName: System.Runtime.CompilerServices.RuntimeCompatibilityAttribute ::
instance void .ctor()
Length: 30
Value : 01 00 01 00 54 02 16 57 72 61 70 4e 6f 6e 45 78 > T WrapNonEx<
: 63 65 70 74 69 6f 6e 54 68 72 6f 77 73 01 >ceptionThrows <
ctor args: ()
AssemblyRef #1 (23000001)
-------------------------------------------------------
Token: 0x23000001
Public Key or Token: b7 7a 5c 56 19 34 e0 89
Name: mscorlib
Version: 4.0.0.0
Major Version: 0x00000004
Minor Version: 0x00000000
Build Number: 0x00000000
Revision Number: 0x00000000
Locale: <null>
HashValue Blob:
Flags: [none] (00000000)
User Strings
-------------------------------------------------------
70000001 : ( 2) L"Hi"
Coff symbol name overhead: 0
```

لازم نیست که تمامی اطلاعات بالا را به طور کامل بفهمید. همین که متوجه شوید برنامه شامل TypeDef است که نام آن Program است و این نوع به صورت یک کلاس عمومی sealed است که از نوع system.object ارث بری کرده است (یک نوع ارجاع از اسمبلی دیگر) و برنامه شامل دو متد main و یک سازنده ctor. است، کافی هست.

متد Main یک متد عمومی و ایستا static است که شامل کد IL است و هیچ خروجی ندارد و هیچ آرگومانی را نمی‌پذیرد. متد سازنده عمومی است و شامل کد IL است، سازنده هیچ نوع خروجی ندارد و هیچ آرگومانی هم نمی‌پذیرد و یک اشاره‌گر که به یک object در حافظه که موقع صدا زدن ساخته خواهد شد.

ابزار ILDasm امکاناتی بیشتری از آنچه که دیدید ارائه می‌کند. به عنوان نمونه اگر مسیر زیر را در منوها طی کنید:

```
View/statistics
```

اطلاعات آماری زیر نمایش داده می‌شود:

```
File size : 3584
PE header size : 512 (496 used) (14.29%)
PE additional info : 1411 (39.37%)
Num.of PE sections : 3
CLR header size : 72 ( 2.01%)
CLR meta-data size : 612 (17.08%)
CLR additional info : 0 ( 0.00%)
CLR method headers : 2 ( 0.06%)
```

```
Managed code : 20 ( 0.56%)
Data : 2048 (57.14%)
Unaccounted : -1093 (-30.50%)
Num.of PE sections : 3
.text - 1024
.rsrc - 1536
.reloc - 512
CLR meta-data size : 612
Module - 1 (10 bytes)
TypeDef - 2 (28 bytes) 0 interfaces, 0 explicit layout
TypeRef - 4 (24 bytes)
MethodDef - 2 (28 bytes) 0 abstract, 0 native, 2 bodies
MemberRef - 4 (24 bytes)
CustomAttribute- 2 (12 bytes)
Assembly - 1 (22 bytes)
AssemblyRef - 1 (20 bytes)
Strings - 184 bytes
Blobs - 68 bytes
UserStrings - 8 bytes
Guids - 16 bytes
Uncategorized - 168 bytes
CLR method headers : 2
Num.of method bodies - 2
Num.of fat headers - 0
Num.of tiny headers - 2
Managed code : 20
Ave method size - 10
```

اطلاعات بالا شامل نمایش حجم فایل به بایت و سایر قسمت‌های تشکیل دهنده فایل است...

توجه: ILDasm یک باگ دارد که بر نمایش اندازه‌ی فایل تاثیر می‌گذارد و باعث می‌شود شما نتوانید به اطلاعات ثبت شده اعتماد داشته باشید.