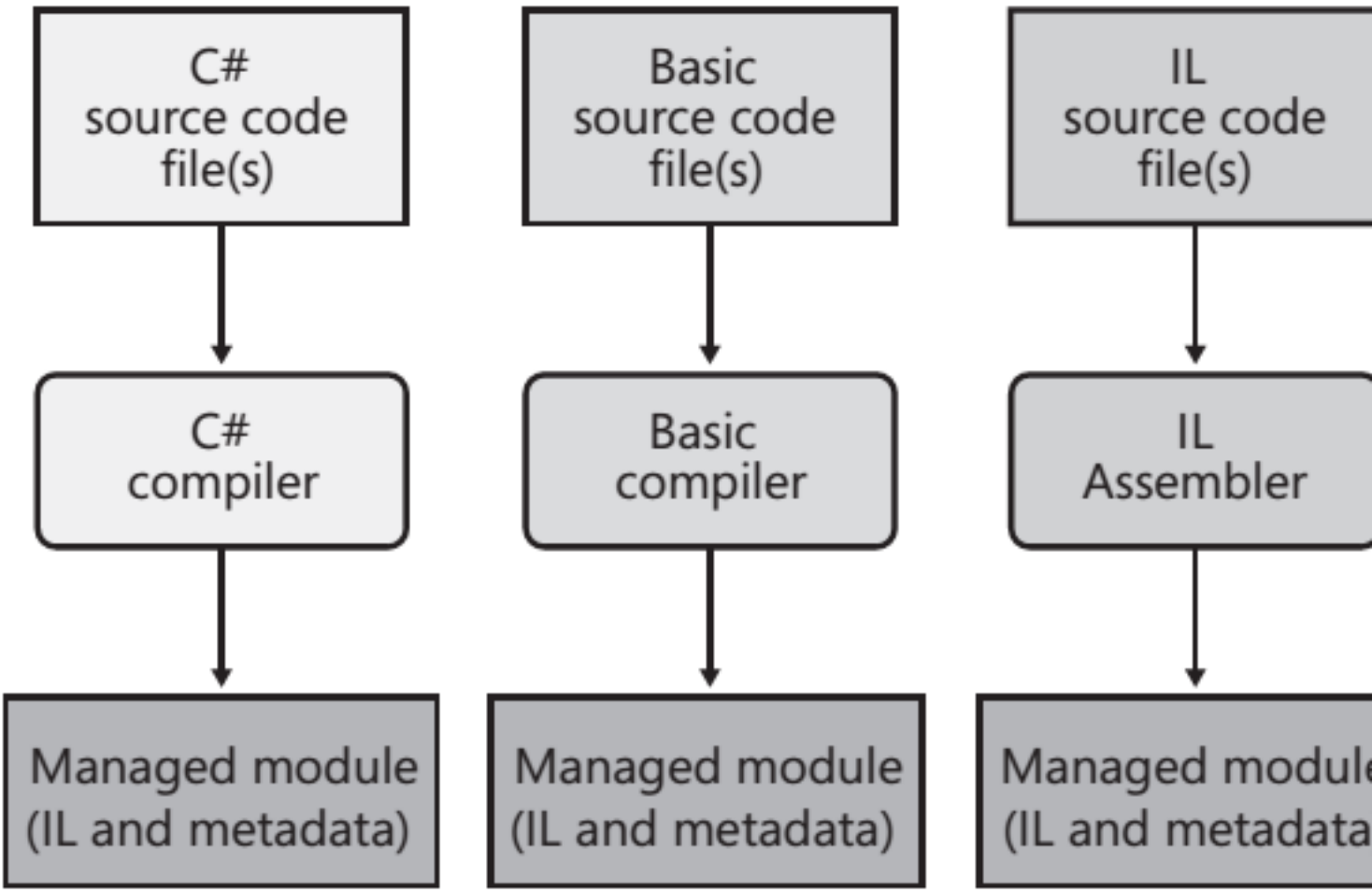
عنوان: **همه چیز در مورد CLR : قسمت اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۲۲ ۱۵:۴۵
آدرس: www.dotnettips.info
گروه‌ها: CLR, .NET Framework

در حال حاضر من کتاب CLR Via Csharp ویرایش چهارم نوشته آقای جفری ریچر را مطالعه می‌کنم و نه قسمت از این مقالات، از بخش اول فصل اول آن به پایان رسیده که همگی آن‌ها را تا 9 روز آینده منتشر خواهم کرد. البته سعی شده که مقالات ترجمه صرف نباشند و منابع دیگری هم در کنار آن استفاده شده است. بعضی موارد را هم لینک کرده‌ام. تمام سعی خود را می‌کنم تا ادامه کتاب هم به مرور به طور مرتب ترجمه شود؛ تا شاید نسخه‌ی تقریبا کاملی از این کتاب را به زبان فارسی در اختیار داشته باشیم.
بعد از اینکه برنامه را تحلیل کردید و نیازمندی‌های یک برنامه را شناسایی کردید، وقت آن است که زبان برنامه نویسی خود را انتخاب کنید. هر زبان ویژگی‌های خاص و منحصر به فرد خود را دارد و این ممکن هست انتخاب شما را سخت کند. برای مثال شما در زبان‌های C/C++ unmanaged، کنترل بسیار زیادی روی امور سیستمی از قبیل حافظه و تردها دارید و به هر روشی که می‌خواهید می‌توانید آن‌ها را پیکربندی کنید. در زبان‌هایی چون Visual basic قدیم و مشابه‌های آن عموما اینگونه بود که طراحی یک اپلیکیشن از رابط کاربری گرفته تا اتصال به دیتابیس و اشیاء COM در آن ساده باشد؛ **ولی در زبان‌های CLR چطور؟**

در زبان‌های CLR شما دیگر وقت خود را به موضوعاتی چون مدیریت حافظه، هماهنگ سازی تردها و مباحث امنیتی و صدور استثناء در سطوح پایین‌تر نمی‌دهید و فرقی هم نمی‌کند که از چه زبانی استفاده می‌کنید. بلکه CLR هست که این امور را انجام می‌دهد و این مورد بین تمامی زبان‌های CLR مشترک است. برای مثال کاربری که قرار است در زمان اجرا استثناءها را صادر کند، در واقع مهم نیست که از چه زبانی برای آن استفاده می‌کند. بلکه آن CLR است که مدیریت آن را به عهده دارد و روال کار CLR برای همه زبان‌ها یکی است. پس این سوال پیش می‌آید که وقتی مبنا و زیر پایه‌ی همه زبان‌های CLR یکی است، چرا تعدد زبان دیده می‌شود و مزیت هر کدام بر دیگری چیست؟ اولین مورد syntax آن است. هر کاربر رو به چه زبانی کشیده می‌شود و شاید تجربه‌ی سابق در قدیم با یک برنامه‌ی مشابه بوده است که همچنان همان رویه سابق را ادامه می‌دهد و یا اینکه نحوه‌ی تحلیل و آنالیز کردن کدهای آن زبان است که کاربر را به سمت خود جذب کرده است. گاهی اوقات بعضی از زبان‌ها با تمرکز در انجام بعضی از کارها چون امور مالی یا ریاضیات، موارد فنی و ... باعث جذب کاربران آن گروه کاری به سمت خود می‌شوند. البته بعدا در آینده متوجه می‌شویم که بسیاری از زبان‌ها مثل سی شارپ و ویژوال بیسیک هر کدام قسمتی از امکانات CLR را پوشش می‌دهند نه تمام آن را.

زبان‌های CLR چگونه کار می‌کنند؟

در اولین گام بعد از نوشتن برنامه، کامپایلر آن زبان دست به کار شده و برنامه را برای شما کامپایل می‌کند. ولی اگر تصور می‌کنید که برنامه را به کد ماشین تبدیل می‌کند و از آن یک فایل اجرایی می‌سازد، سخت در اشتباه هستید. کامپایلر هر زبان CLR، کدها را به یک زبان میانی Intermediate Language به اختصار IL تبدیل می‌کند. فرقی نمی‌کند چه زبانی کار کرده‌اید، کد شما تبدیل شده است به یک زبان میانی مشترک. CLR نمی‌تواند برای تک تک زبان‌های شما یک مفسر داشته باشد. در واقع هر کمپایلر قواعد زبان خود را شناخته و آن را به یک زبان مشترک تبدیل می‌سازد و حالا CLR می‌تواند حرف تمامی زبان‌ها را بفهمد. به فایل ساخته شده managed module گویند و به زبان‌هایی که از این قواعد پیروی نمی‌کنند unmanaged گفته می‌شود؛ مثل زبان سی ++ که در دات نت هم managed و هم unmanaged داریم که اولی بدون فریم ورک دات نت کار می‌کند و مستقیما به کد ماشین تبدیل می‌شود و دومی نیاز به فریم ورک دات نت داشته و به زبان میانی کامپایل می‌شود. جدول زیر نشان می‌دهد که کد همه‌ی زبان‌ها تبدیل به یک نوع شده است.



فایل هایی که ساخته می‌شوند بر دو نوع هستند؛ یا بر اساس استاندارد windows Portable Executable 32bits برای سیستم‌های 32 بیتی و 64 بیتی هستند و یا بر اساس windows Portable Executable 64bits مختص سیستم‌های 64 بیتی هستند که به ترتیب PE32 و +PE32 نامیده می‌شوند که CLR بر اساس این اطلاعات آن‌ها را به کد اجرایی تبدیل می‌کند. زبان‌های CLR همیشه این مزیت را داشته‌اند که اصول امنیتی چون DEP یا Data Execution Prevention و همچنین ASLR یا Address Space Layout Randomization در آن‌ها لحاظ شده باشد.