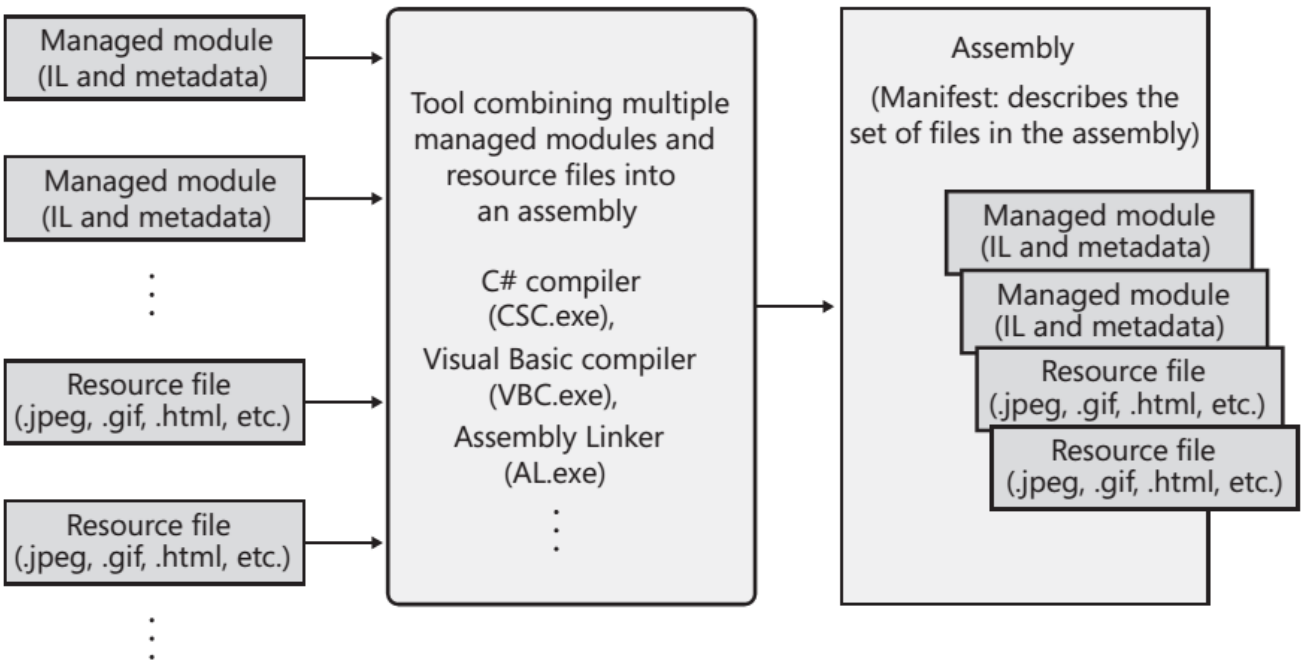
عنوان: **آشنایی با CLR: قسمت سوم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۲۴ ۱:۱۵
آدرس: www.dotnettips.info
گروه‌ها: CLR, .NET Framework

در اینجا ما زیاد بر روی جزئیات یک اسمبلی مانور نمی‌دهیم و آن را به آینده موکول می‌کنیم و فقط مقداری از مباحث اصلی را ذکر می‌کنیم.

ترکیب ماژول‌های مدیریت شده به یک اسمبلی

اگر حقیقت را بخواهید CLR نمی‌تواند با ماژول‌ها کار کند، بلکه با اسمبلی‌ها کار می‌کند. اسمبلی یک مفهوم انتزاعی است که به سختی میتوان برای بار اول آن را درک کرد.
اول از همه: اسمبلی یک گروه منطقی از یک یا چند ماژول یا فایل‌های ریسورس (منبع) است.
دوم: اسمبلی کوچکترین واحد استفاده مجدد، امنیت و نسخه بندی است.
بر اساس انتخابی که شما در استفاده از کامپایلرها و ابزارها کرده‌اید، نسخه‌ی نهایی شامل یک یا چند فایل اسمبلی خواهد شد. در دنیای CLR ما یک اسمبلی را کامپوننت صدا می‌زنیم.

شکل زیر در مورد اسمبلی‌ها توضیح می‌دهد. آنچه که شکل زیر توضیح می‌دهد تعدادی از ماژول‌های مدیریت شده به همراه فایل‌های منابع یا دیتا توسط ابزارهایی که مورد پردازش قرار گرفته‌اند به فایل‌های 32 یا 64 بیتی تبدیل شده‌اند که داخل یک گروه بندی منطقی از فایل‌ها قرار گرفته‌اند. آنچه که اتفاق می‌افتد این هست که این فایل‌های 32 یا 64 بیتی شامل بلوکی از داده‌هایی است که با نام manifest شناخته می‌شوند. manifest یک مجموعه دیگر از جداول متادیتاها است. این جداول به توصیف فایل‌های تشکیل دهنده اسمبلی می‌پردازد.



همه کارهای تولید اسمبلی به صورت خودکار اتفاق می‌افتد. ولی در صورتیکه قصد دارید فایلی را به اسمبلی به طور دستی اضافه کنید نیاز است که به دستورات و ابزارهای کامپایلر آشنایی داشته باشید.

یک اسمبلی به شما اجازه می‌دهد تا مفاهیم فیزیکی و منطقی کامپوننت را از هم جدا سازید. اینکه چگونه کد و منابع خود را از یکدیگر جدا کنید به خود شما بر می‌گردد. برای مثال اگر قصد دارید منابع یا نوع داده‌ای را که به ندرت مورد استفاده قرار می‌گیرد، در یک فایل جدا از اسمبلی نگهداری کنید، این فایل جدا میتواند بر اساس تقاضای کاربر در زمان اجرای برنامه از اینترنت دریافت شود. حال اگر همین فایل هیچگاه استفاده نشود، در زمان نصب برنامه و مقدار حافظه دیسک سخت صرفه جویی خواهد شد. اسمبلی‌ها به شما اجازه می‌دهند که فایل‌های توزیع برنامه را به چندین قسمت بشکنید، در حالی که همه‌ی آن‌ها متعلق به یک مجموعه هستند.

یک ماژول اسمبلی شامل اطلاعاتی در رابطه با ارجاعاتش است؛ به علاوه ورژن خود اسمبلی. این اطلاعات سبب می‌شوند که یک اسمبلی خود تعریف self-describing شود که به بیان ساده‌تر باعث می‌شود CLR وابستگی‌های یک اسمبلی را تشخیص داده تا ترتیب اجرای آن‌ها را پیدا کند. نه دیگر نیازی به اطلاعات اضافی در ریجستری است و نه در Active Directory Domain Service یا به اختصار ADDS.

از آنجایی که هیچ اطلاعاتی اضافی نیست، توزیع ماژول‌های مدیریت شده راحت‌تر از ماژول‌های مدیریت نشده است.

مطلب مشابهی نیز در وبلاگ آقای شهروز جعفری برای توصیف اسمبلی‌ها وجود دارد که خیلی خوب هست به قسمت مطالب مرتبط آن هم نگاهی داشته باشید.