عنوان: **Roslyn #1**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۶/۲۶ ۱۴:۳۵
آدرس: www.dotnettips.info
گروه‌ها: Visual Studio, Roslyn

معرفی Roslyn

سکوی کامپایلر دات نت یا Roslyn (با تلفظ «رازلین») بازنویسی مجدد کامپایلرهای VB.NET و C# توسط همین زبان‌ها است. این سکوی کامپایلر به همراه یک سری کتابخانه و اسمبلی ارائه می‌شود که امکان آنالیز زبان‌های مدیریت شده را به صورت مستقل و یا یکپارچه‌ی با ویژوال استودیو، فراهم می‌کنند. برای نمونه در VS.NET 2015 تمام سرویس‌های زبان‌های موجود، با Roslyn API جایگزین و بازنویسی شده‌اند. نمونه‌هایی از این سرویس‌های زبان‌ها، شامل Intellisense و مرور کدها مانند go to references and definitions، به همراه امکانات Refactoring می‌شوند. به علاوه به کمک Roslyn می‌توان یک کامپایلر و ابزارهای مرتبط با آن، مانند FxCop را تولید کرد و یا در نهایت یک فایل اسمبلی نهایی را از آن تحویل گرفت.

چرا مایکروسافت Roslyn را تولید کرد؟

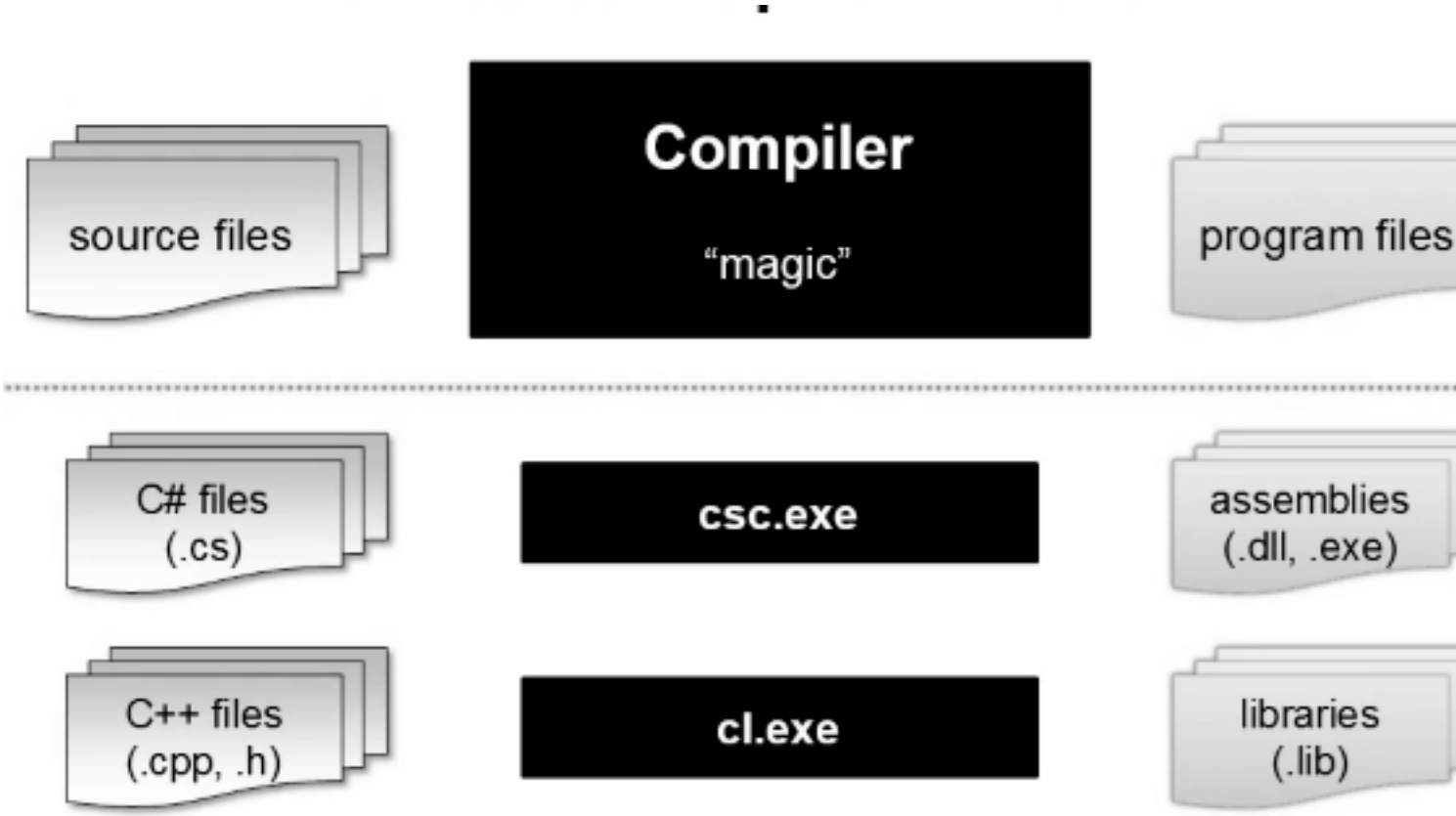
پیش از پروژه‌ی Roslyn، کامپایلرهای VB.NET و C# با زبان C++ نوشته شده بودند؛ از این جهت که در اواخر دهه‌ی 90 که کار تولید سکوی دات نت در حال انجام بود، هنوز امکانات کافی برای نوشتن این کامپایلرها با زبان‌های مدیریت شده وجود نداشت و همچنین زبان محبوب کامپایلر نویسی در آن دوران نیز C++ بود. این انتخاب در دراز مدت مشکلاتی مانند کاهش انعطاف پذیری و productivity تیم کامپایلر نویس را با افزایش تعداد سطرهای کامپایلر نوشته شده به همراه داشت و افزودن ویژگی‌های جدید را به زبان‌های VB.NET و C# سخت‌تر و سخت‌تر کرده بود. همچنین در اینجا برنامه نویس‌های تیم کامپایلر مدام مجبور بودند که بین زبان‌های مدیریت شده و مدیریت نشده سوئیچ کنند و امکان استفاده‌ی همزمان از زبان‌هایی را که در حال توسعه‌ی آن هستند، نداشتند.
این مسایل سبب شدند تا در طی بیش از یک دهه، چندین نوع کامپایلر از صفر نوشته شوند:
- کامپایلرهای خط فرمانی مانند csc.exe و vbc.exe
- کامپایلر پشت صحنه‌ی ویژوال استودیو (برای مثال کشیدن یک خط قرمز زیر مشکلات دستوری موجود)
- کامپایلر snippetها در immediate window ویژوال استودیو

هر کدام از این کامپایلرها هم برای حل مسایلی خاص طراحی شده‌اند. کامپایلرهای خط فرمانی، با چندین فایل ورودی، به همراه ارائه‌ی تعدادی زیادی خطا و اخطار کار می‌کنند. کامپایلر پشت صحنه‌ی ویژوال استودیوهای تا پیش از نسخه‌ی 2015، تنها با یک تک فایل در حال استفاده، کار می‌کند و همچنین باید به خطاهای رخ داده نیز مقاوم باشد و بیش از اندازه گزارش خطا ندهد. برای مثال زمانیکه کاربر در حالت تایپ یک سطر است، بدیهی است تا اتمام کار، این سطر فاقد ارزش دستوری صحیحی است و کامپایلر باید به این مساله دقت داشته باشد و یا کامپایلر snippetها تنها جهت ارزیابی یک تک سطر از دستورات وارد شده، طراحی شده‌است.

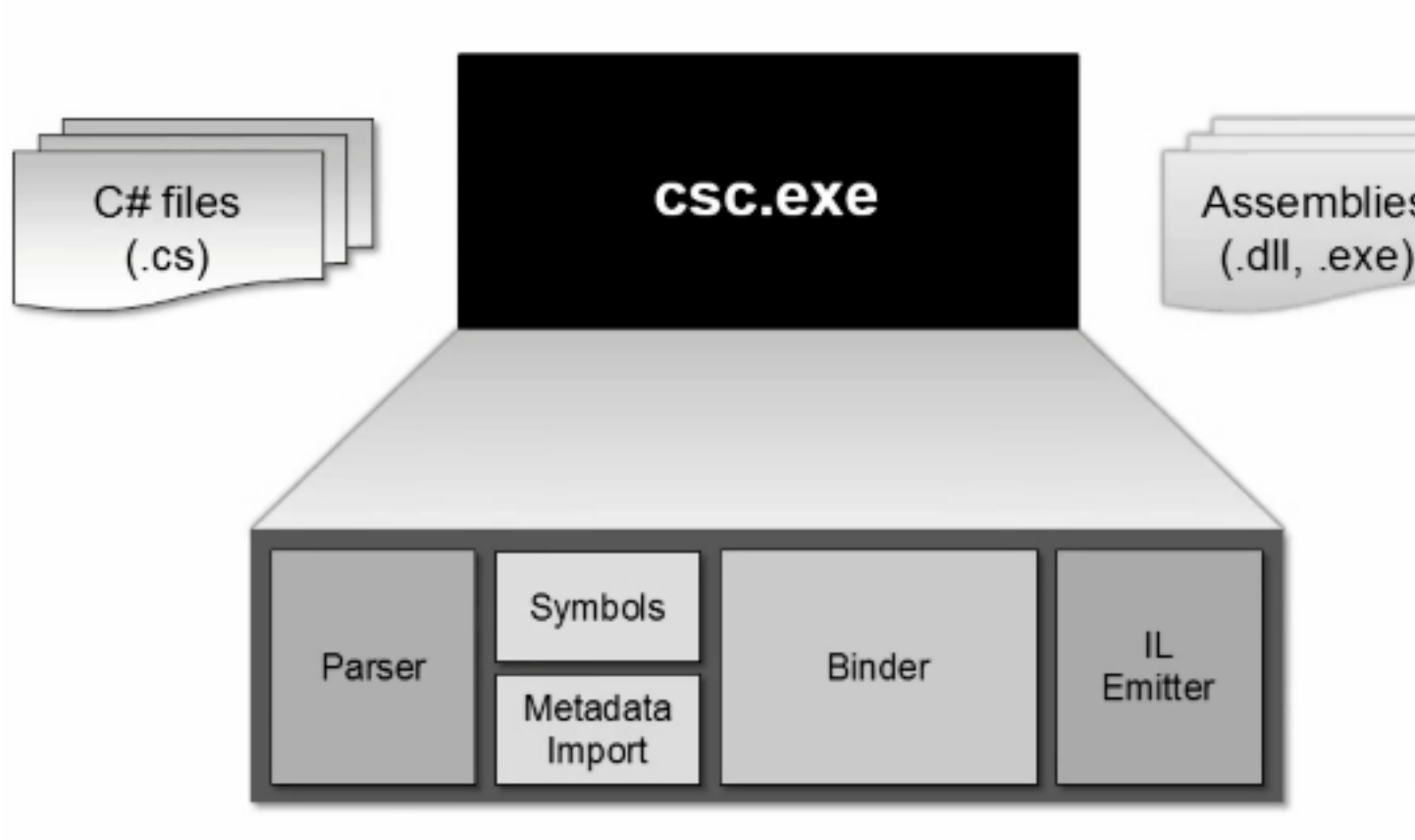
با توجه به این مسایل، مایکروسافت از بازنویسی سکوی کامپایلر دات نت این اهداف را دنبال می‌کند:
- بالا بردن سرعت افزودن قابلیت‌های جدید به زبان‌های موجود
- سبک کردن حجم کاری کامپایلر نویسی و کاهش تعداد آن‌ها به یک مورد
- بالا بردن دسترسی پذیری به API کامپایلرها
برای مثال اکنون برنامه نویس‌ها بجای اینکه یک فایل cs را به کامپایلر csc.exe ارائه کنند و یک خروجی باینری دریافت کنند، امکان دسترسی به syntax trees، semantic analysis و تمام مسایل پشت صحنه‌ی یک کامپایلر را دارند.
- ساده سازی تولید افزونه‌های مرتبط با زبان‌های مدیریت شده.
اکنون برای تولید یک آنالیز کننده‌ی سفارشی، نیازی نیست هر توسعه دهنده‌ای شروع به نوشتن امکانات پایه‌ای یک کامپایلر کند. این امکانات به صورت یک API عمومی در دسترس برنامه نویس‌ها قرار گرفته‌اند.
- آموزش مسایل درونی یک کامپایلر و همچنین ایجاد اکوسیستمی از برنامه نویس‌های علاقمند در اطراف آن.
همانطور که اطلاع دارید، Roslyn به صورت سورس باز در GitHub در دسترس عموم است .

تفاوت Roslyn با کامپایلرهای سنتی

اکثر کامپایلرهای موجود به صورت یک جعبه‌ی سیاه عمل می‌کنند. به این معنا که تعدادی فایل ورودی را دریافت کرده و در نهایت یک خروجی باینری را تولید می‌کنند. اینکه در این میان چه اتفاقاتی رخ می‌دهد، از دید استفاده کننده مخفی است.

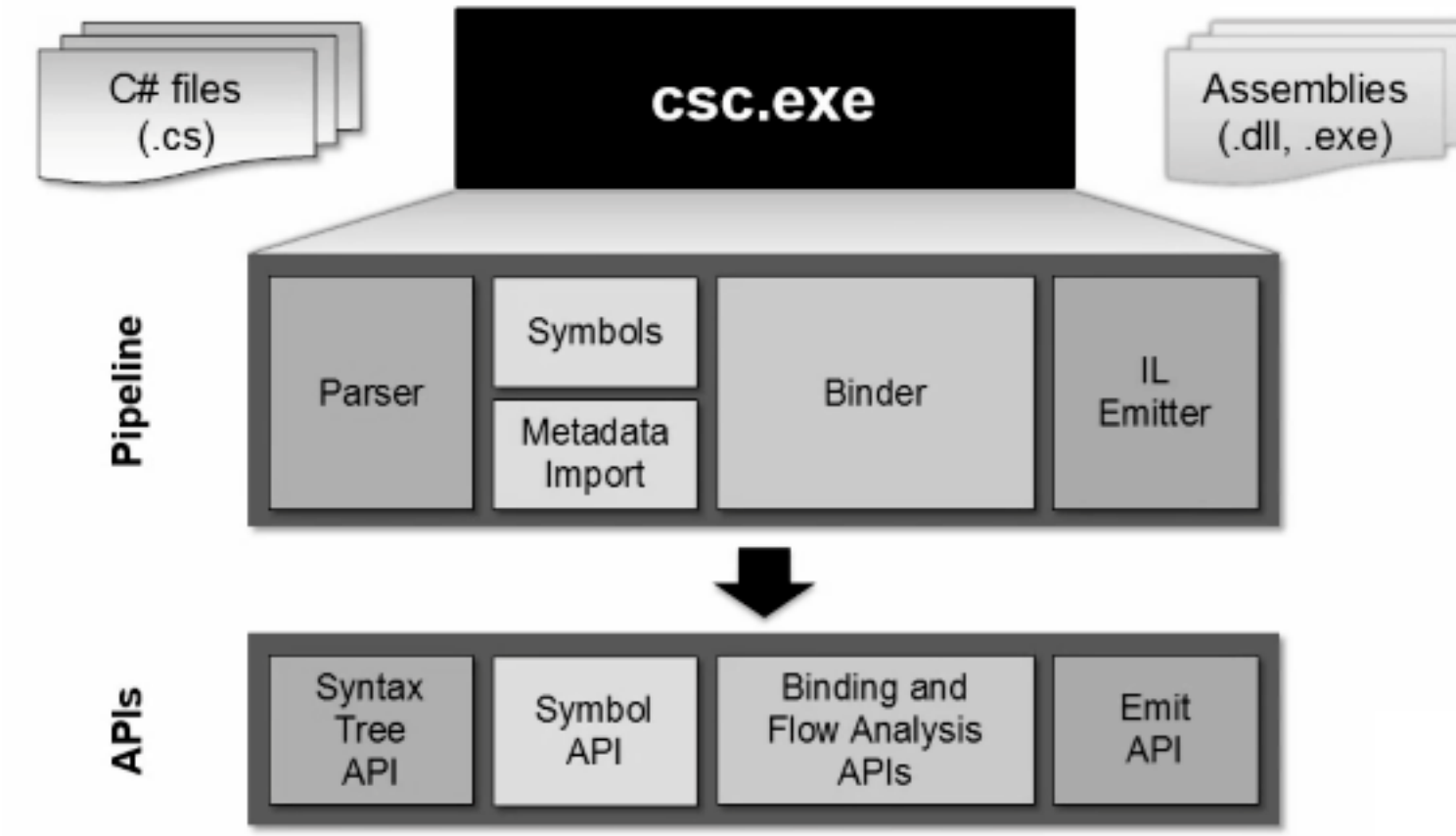


نمونه‌ای از این کامپایلرهای جعبه سیاه را در تصویر فوق مشاهده می‌کنید. در اینجا شاید این سؤال مطرح شود که در داخل جعبه‌ی سیاه کامپایلر سی‌شارپ، چه اتفاقاتی رخ می‌دهد؟

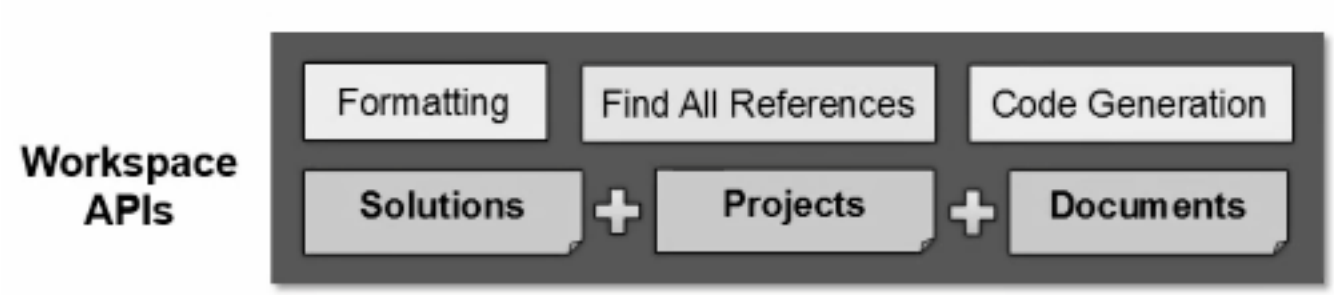


خلاصه‌ی مراحل رخ داده در کامپایلر سی‌شارپ را در تصویر فوق ملاحظه می‌کنید. در اینجا ابتدا کار parse اطلاعات متنی دریافتی شروع می‌شود و از روی آن syntax tree تولید می‌شود. در مرحله‌ی بعد مواردی مانند ارجاعاتی به mscorlib و امثال آن پردازش می‌شوند. در مرحله‌ی binder کار پردازش حوزه‌ی دید متغیرها، اشیاء و اتصال آن‌ها به هم انجام می‌شود. در مرحله‌ی آخر، کار تولید کدهای IL و اسمبلی باینری نهایی صورت می‌گیرد.

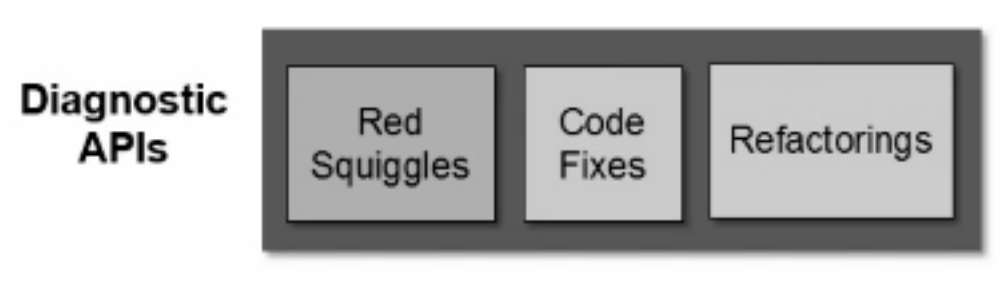
با معرفی Roslyn، این جعبه‌ی سیاه، به صورت یک API عمومی در دسترس برنامه نویس‌ها قرار گرفته‌است:



همانطور که مشاهده می‌کنید، هر مرحله‌ی کامپایل جعبه‌ی سیاه، به یک API عمومی Roslyn نگاشت شده‌است. برای مثال Parser به Syntax tree API نگاشت شده‌است. به علاوه این API صرفا به موارد فوق خلاصه نمی‌شود و همانطور که پیشتر نیز ذکر شد، برای اینکه بتواند جایگزین سه نوع کامپایلر موجود شود، به همراه Workspace API نیز می‌باشد:



Roslyn امکان کار با یک Solution و فایل‌های آن را دارد و شامل سرویس‌های زبان‌های مورد نیاز در ویژوال استودیو نیز می‌شود. برفراز Workspace API، یک مجموعه API دیگر به نام Diagnostics API تدارک دیده شده‌است تا برنامه نویس‌ها بتوانند امکانات Refactoring جانبی را توسعه داده و یا در جهت بهبود کیفیت کدهای نوشته شده، اخطارهایی را به برنامه نویس‌ها تحت عنوان Code fixes و آنالیز کننده‌ها، ارائه دهند.


Diagnostic APIs
Red Squiggles
Code Fixes
Refactorings