تاريخچه ساده و مختصر

وقتي ويندوز 3/0 روش ابتدايي ارتباط درميان برنامه‌ها مبادله پوياي داده‌ها يا DDE بود. DDE داراي منابع فشرده بود، انعطاف‌پذير بود و مستعد ايجاد برخوردهاي سيستمي بود. با اين وجود سالها در رايانه‌ها بطور قابل‌قبولي كار كرد و برنامه‌هاي بسياري با استفاده از اين روش پيام‌ها را بين اين برنامه‌ها ارسال مي‌كردند.

درطي اين سالها ، مايكروسافت روش استفاده از DDE را منع كرد و استفاده از الگوي شيئي مشترك (COM) و COM توضيعي (DCOM) را تشويق نمود. COM براي ارتباط بين برنامه‌هاي مايكروسافت در يك رايانه تنها بكار مي‌رفت. ولي از DCOM براي ارتباط با ميزبانهاي دوردست استفاده مي‌شد.

درضمن مشاركتي كه از چند فروشندة هم‌پيمان تشكيل شده بود (ازجمله آي بي ام، سان و اپل) روش ديگري براي ارتباط با ميزبان‌ها به نام CORBA پيشنهاد كرد. روش CORBA برخلاف COM در مورد عبور دادن پيام‌ها ميان سيستم‌هاي عامل مختلف بهتر عمل مي‌كرد. متأسفانه اين ضابطه داراي منابع فشرده بود و برنامه‌نويسي آن دشوار بود و برخلاف آنچه انتظار مي‌رفت، استفاده از آن مدت زيادي طول نكشيد.

دراين زمان، مايكروسافت فن‌آوري خود را بالاتر برد و فن‌آوري‌هاي COM+ ، سرور تبادل مايكروسافت(MTS) و معماري توضيعي شبكه DNA را معرفي نمود. اين فن‌آوري‌ها امكان تبادل‌هاي پيچيده‌تري را بين اجزاي شبكه فراهم نمود. نظير ذخيره‌سازي شيء ، عملكردهاي كاربر و تبادلات. متأسفانه دراين فناوري‌ها لازم است كه هر برنامه موارد زيادي را راجع به برنامه‌هاي ديگر بدانيم، و درصورت همنوع نبودن سيستم‌هاي عامل، آنها بخوبي كار نمي‌كنند. (مثلاً برقراري ارتباط برنامه‌هاي ويندوز با لينوكس)

همه اين روشها را به سال 2001 و طرح جديد .NET رساند. اين طرح قدرت COM را همراه با انعطاف‌پذيري CORBA يكجا دارد. با وجود اينكه اين فناوري در مرحلة اول همراه با مايكروسافت ارائه شده است، ولي انعطاف‌پذيري و قابليت توسعه‌پذيري آن به معناي آن است كه ازنظر تئوري مي‌تواند در آينده براي سيستم‌هاي عامل ديگر مورد استفاده قرار گيرد (هرچند چارچوب .NET در همة سيستم‌هاي عامل مربوط به ويندوز 95 به بعد اجرا مي‌شود ولي نسخة ديگري به نام چارچوب فشردة .NET فقط براي اجرا در ويندوز CE درنظر گرفته شد).

معماري .NET

چاچوب .NET از سه بخش تشكيل شده است:

1- اجراي زبان مشترك

2- كلاسهاي چارچوب

3- ASP.NET

.NET در راه است

در جولاي سال 2000 اعلام گرديد كه .NET برنامه شركت ميكروسافت براي خدمات وب مبتني برXML ، در حال تهيه (نسخه بتا) براي عرضه در اواخر سال جاري يا اوايل سال 2002 مي‌باشد. .NET داراي نقشي مهم در استراتژي شركت ميكروسافت براي بهبود خدمات وب، اينترنت و ايجاد خدمات بلوكي، ابزارهاي متعدد براي طراحان و بسياري امكانات ديگر مي‌باشد.

اما اين نسل جديد نرم‌افزار نتوانسته است توجه دنياي كامپيوتر را مانند Java در زمان اوج محبوبيت خود، جلب نمايد. .NET پديده مهمتري مي‌باشد. مطبوعات تجاري هنوز اهميت فناوري خود را از دست نداده‌اند. هم week Business و هم The Economist بخش گسترده‌اي از پوشش خبري خود را به .NET اختصاص داده‌اند.

شايد گستره عظيم پوشش دهي .NET بعضي از ناظران فني را سردرگم نموده و انها را وادار به تفكر پيرامون چيزي كه در زير پوشش .NET وجود خواهد داشت، كرده است. يكي از ناظراني كه چنين نمي‌انديشد، John Dvorak است كه در ستون خود در مجله PC Magazin نوشته است كه .NET با شايعات و كليت‌هاي بسياري احاطه شده است كه بايد تمامي آنها را مورد توجه قرار داد.

اساس چنين ادعايي، گزارشي مربوطبه ميكروسافت است كه در آن، آينده هيجان انگيزي به لطف وجود .NET به تصوير كشيده شده است. شما با كمك دوچرخه‌تان كه به وب متصل بوده مي‌توانيد به سهولت رزرواسيون رستوران خود را براي امشب تغيير داده و اين كار بدون از دست دادن لحظه‌اي، وقت ضمن حركت، ممكن است. اگر هدف، ايجاد ايده‌آل‌هايي بزرگ و رفيع براي نسل بعد مي‌باشد، اين مثال مسلماً مبالغه‌آميز خواهد بود ولي از چه زمان روزنامه‌نگاران فني پيرامون فناوريهاي جديد قضاوت و پيشنهاد مي‌كنند؟ بعضي افراد ديگر باهوشتر بوده‌اند. مثلاً Patricia Seybold Group در مقاله‌اي نوشته است كه .NET نمونه‌اي پيشرو از آن چيزي است كه ما معتقديم، مدل ساختاري غالب براي برنامه‌هاي نسل سوم اينترنت خواهد بود. و اين برنامه نشانه‌اي بدشگون براي بسياري از رقباي ميكروسافت خواهد بود.

در تشريح .NET بهتر است ابتدا به مواردي اشاره كنيم كه .NET جزء آنها نمي باشد. .NET يك سيستم عامل و يا يك زبان برنامه‌نويسي نيست. سيستم‌هاي عامل ميكروسافت به تكامل خود ادامه مي‌دهند (CE و XP و Me و Windows 2000 براي دستگاههاي داخلي) .NET براي زبانهاي برنامه‌نويسي، زباني جديد C# را ارائه نموده ولي اين امر مورد توجه اين فناوري نبوده است. بلكه تنها وسيله‌اي براي بيان برنامه‌نويسي زمان اجراي .NET مي‌باشد.

C# در حالي كه به رقيبي جدي، براي جاوا تبديل مي‌شود براي جايگزيني به جاي تمامي زبانهاي موجود، عرضه نشده است. در واقع سرمايه‌گذاري فعلي ميكروسافت در Visual Basic و C++ با چنين هدفي تناقض دارد. .NET علاوه بر زبانهاي تحت پشتيباني ميكروسافت، بسياري از زبانهاي ديگري چون Python ، Perl، fortan، Eiffel ، CoBol ، Smalltalk و زبانهاي تحقيقاتي از ML تا Haskell و Oberon را تحت پوشش قرار مي‌دهد. ميكروسافت برخلاف ساير شركتهاي فعال در صنايع، قصد ندارد دنيا را به زباني جديد تغيير دهد.

ساختار .NET

بنابراين .NET چيست؟ در تعريفي كلي، مي‌توان آن را " يك پلاتفورم آزاد زبان برنامه‌نويسي، براي توسعه وب و شركتها " دانست. هدف فراهم‌سازي ماشيني انتزاعي براي برنامه‌نويسان حرفه‌اي است كه بتواند برنامه‌هاي مبتني بر وب و سرويس‌دهنده و سرويس گيرنده و يا به عبارتي" فناوري اطلاعات " را تحت پوشش قرار دهد.

در زير شش لايه از ساختار كلي برنامه نشان داده شده است:

خدمات وب: لايه فوقاني، كاربران .NET (اشخاص و شركتها) را همراه با خدمات وب براي تجارت الكترونيكي و برنامه‌هاي business to-business شامل مي‌شود.

چهارچوب كاري و كتابخانه‌ها: مجموعه‌اي از چهار چوبهاي كاري و كتابخانه‌ها، سيمايي جذاب براي طراحان فراهم نموده است. اين بخش شامل ASP.NET (صفحات فعال سرويس دهنده براي توسعه سايت‌هاي هوشمند وب و خدمات آن)، ADO.NET نوعي بهبود مبتني بر XML براي اشياء داده‌اي اكتيوايكس براي پردازش بانكهاي اطلاعاتي و مربوط به شيء و فرمهاي ويندوز براي تصاوير، مي‌باشد. .NET در كل شامل هزاران جزء قابل استفاده مجدد مي‌باشد.

استانداردهاي تبادلي: استانداردهاي تبادلي مبتني بر XML به عنوان وسيله‌اي مستقل از برنامه براي تبادل اشياء عمل مي‌كند. مهمترين آنها SOAP (پروتكل دستيابي ساده به اشياء) يكي از روشهايي كه براي كدگذاري اشياء به سرعت رواج يافته است و WSDL (زبان توصيف خدمات وب) مي‌باشند.

محيط توسعه: برنامه جديد Visual Studio.Net صريح‌ترين ابزار را براي طراحان و توسعه‌دهندگان براي طراحي، ترجمه، مرور و اشكال‌زدايي را بطور مشترك براي چندين زبان فراهم نموده است. اين محيط حاصل

Visual Studio توسعه يافته همراه با يك رابط برنامه نويسي كاربردي است كه نه تنها از زبانهاي مورد استفاده ميكروسافت مانند Visual C++ ، C# و Visual Basic پشتيباني نموده بلكه به فروشندههاي ديگر نيز امكان اتصال به اين ابزار و مترجمها را بدست ميدهد.

مدل اجزاء: پيش از انكه .NET مطرح شود، سه رقيب اصلي و مهم براي رياست در زمينه مدلها و استانداردهاي توسعه مبتني بر اجزاء وجود داشت: اين سه عبارتند از: Cobra از Object Management و J2EE از SUN و SOM از ميكروسافت. .NET مدلي ديگر را بر پايه نظريات شيگرا ارائه نموده است. با كمك .NET ميتوانيد شباهت سازيها را انجام دهيد هريك از آنها ميتوانند شامل تعداد زيادي كلاس همراه با رابطهاي كاملاً تعريف شده باشند. اين مدل كاملاً با COM متفاوت است هرچند كه مسير انتقالي را فراهم مينمايد، مهمترين ويژگيهاي آن در جذب طراحان، سادگي و عدم وجود يك IDL (زبان توصيف رابط) ميباشد.

مدل شيء: مدل شيء بيان مفهومي را تشكيل ميدهد كه هرچيزي در آن قرار ميگيرد به خصوص سيستم .NET¢s (ويژگي زبان مشترك)، محدوديتهايي را اعمال ميكند كه قابليت بكارگيري زبان را تضمين ميكند.

زمان اجراي زبان مشترك: زمان اجراي زبان مشترك، مجموعه اصلي مكانيسمهاي اجرايي برنامههاي .NET را فراهم ميكند. اين مكانيسمها بدون توجه به مبدا زبان آنها، ترجمه به زبان ماشين، بارگذاري مكانيسمهاي امنيتي، مديريت حافظه (شامل جمعآوري ضايعات)، كنترل نسخه و برقراري ارتباط با كدهاي غير .NET تعيين ميشوند. .NET اين قابليتها را در طيف گستردهاي از برنامههاي نرمافزاري و سختافزاري از سرويس دهندههاي گسترده تا وب و كامپيوترهاي شخصي تلفنها و PDAها و ساير دستگاههاي بيسيم و وسايل اينترنت، فراهم ميكند.

مزاياي .NET

كاربران و طراحان ميتوانند مزاياي متعددي را براي .NET انتظار داشته باشند. براي بسياري از افراد، حساسترين و مؤثرترين جزء ASP.NET ميباشد. ASP.NET بهبوديافته فناوري ASP (صفحات فعال سرويس دهنده) كه در ويندوز موجود ميباشد، نيست. ASP.NET در واقع طرحي جديد است كه ابزاري را براي ايجاد سايتهاي هوشمند وب همراه با تسهيلات برنامهنويسي مؤثر فراهم ميكند. بعضي از جذابترين زواياي چهارچوب كاري NET عبارتند از:

كنترلهاي وب كه براساس پيشفرض، درسمت سرويسدهنده استفاده ميشوند، امكان انتقال مستقل از مرورگر را بدست ميدهد، يعني خروجي بطور خودكار به مرورگر منتقل ميشود. بعنوان مثال اگر يك بازديدكننده وب سايت از نسخه فعلي Internet Explorer و يا مرورگري كه از HTML داينميكي يا Java Script پشتيباني مينمايد استفاده كند، آنگاه درحالت پيشفرض، سرويسدهنده، محاوره را انجام داده و همه چيز را بصورت HTML منتقل ميكند.

ASP.NET يكي از روشنترين زواياي پردازش درخواستي، يعني حفظ حالت يك سرويسگيرنده را انجام ميدهد. HTTP پروتكلي بدون حالت است ولي هر رابط واقعي وب (مثل سبد خريد) بايد اطلاعات سرويس گيرنده را ازيك صفحه نمايش به صفحه بعد منتقل نمايد. ASP.NET حالت وب را بدون ذخيرهسازي اطلاعات سرويسگيرنده بر روي سرويسدهنده، حفظ مينمايد كه درنتيجه آن نيازي نيست كه طراحان از تكنيكهاي پيچيده دستي مانند رمزگذاري URL ، فيلدهاي مخفي و Cookies استفاده نمايند.

از طريق ارتباط ميان ADO.NET و ASP.NET (ADO.NET ارتباط ميان بانكهاي اطلاعاتي را برقرار ميسازد) ASP.NET ميتواند بخشي از صفحه وب را به منظور بازتاباندن مستقيم محتويات يك جدول بانك اطلاعاتي بدون مداخله دستي تنظيم نمايد. هركسي كه سعي ميكند جداول HTML نشاندهنده محتويات بانك اطلاعاتي را كدگذاري نمايد، از اين امكانات استقبال ميكند.

چونكه ASP.NET مستقيماً به مدل شيءاي .NET مترجمها و مكانيسمهاي زمان اجرا مرتبط است. كد مربوط به يك صفحه وب را ميتوان، بخشي ازيك برنامه دانست كه پيچيده بوده ولي از مكانيسمهايي چون محرمانه بودن و نسخهدار شدن از روش ترجمه .NET و هر نوع زبان تحت پشتيباني .NET بهره ميبرد. تسهيلات نسخهدار شدن .NET امكان به روزدرآوردن را ميدهد. فقط كافي است صفحهاي را با نسخه جديدش جايگزين كنيم، درنتيجه بطور خودكار دفعه بعد، بدون نياز به متوقف نمودن و دوباره راهاندازي سرويس دهنده، ترجمه ميشود.

رفع شكاف موجود
شايد ASP.NET بزرگترين سهم را در رفع اختلافات ميان IT (متداولترين توسعه نرمافزاري) و توسعه وب دارا ميباشد. ايجاد قابليت وب براي يك برنامه موجود، تلاشي مهم تلقي شده و ميتواند يك صفحه وب را با تصاوير زيبا و پردازش پيشرفته تجهيز نمايد. .NET و ASP.NET بطوركلي داراي پتانسيل لازم براي ادغام اين دو نظام ميباشند.

معمولاً شركتها نرمافزار را توسعه دادهاند. در سالهاي اخير انها وب سايتها را طراحي و ايجاد كردهاند ولي به تدريج عناصر پردازشي بيشتري (Scripts , CGT, ASP , Java Script) بدست ميآورند. غالباً اين سايتها به طريقي تخصصي طراحي ميشوند و از تجارب متعارف IT در ساير موارد بهره نميبرند. با كمك .NET يك صفحه وب، برنامه خواهد بود و يك برنامه را ميتوان به راحتي به صفحه وب تبديل نمود.

خدمات وب
.NETمكانيسمهايي را ارائه ميكند كه صفحه وب را به عنوان يك رابط انساني و يك رابط برنامه كاربردي، قابل استفاده مينمايد. از انجايي كه صفحه ASP.NET به مجموعهاي از عناصر برنامهاي .NET مرتبط و متصل است، آن را ميتوان يك API يا اخبار به روز درامده، برنامه زمانبندي يك جلسه براساس اوقات فراغت همكاران خود، يا انجام خريدهاي شركت استفاده نماييد.

ميكروسافت با استفاده از ابتكار " Hailstorm " خود، فشاري مهم و عمده بر فناوري " Passport " خود وارد آورده است. اين فناوري در گذشته در Windows XP قرار داده شده و به كاربران كامپيوتر امكان تشريح يك برنامه شخصي و در دسترس قراردادن آن به عنوان مجموعهاي از خدمات وب و مشخص نمودن اينكه چه كسي ميتواند دران مشاركت داشته باشد را ميدهد. اين هدف، تسهيلسازي مهمي را براي فراكنشهاي مختلفي به همراه داشته است.

شركتهاي ميكروسافت، IBM و چند شركت ديگر، قالب SOAP را كه بر XML مبتني است و اشياء .NET را در مقياس وسيع به سراسر جهان ارسال ميكند، طراحي نمودهاند، ميكروسافت و IBM، زبان Web services Description Language را به عنوان استاندارد جديدي براي خدمات وب طراحي نمودهاند.

شركت SUN نيز اخيراً SOAP و WSDL را تاييد نموده است. اگرچه كه سؤالات زيادي در مورد اين جنبه جديد فناوري .NET هنوز بي پاسخ مانده است ولي اين فناوري روبه توسعه ميتواند استانداردهاي واقعي را كه بعضي از سودآورترين پتانسيلهاي اينترنت را فراهم مينمايد، در اختيار ما بگذارد.

امنيت
بازار .NET بر روي خدمات وب به عنوان بخشي از فناوري تاكيد دارد. ولي رسيدن به اين هدف ، به زمان زيادي نياز دارد. در حالت خاص، مجموعهاي از مكانيسمهاي تعبيه شده براي طراحان نرمافزار به جاي كاربران، تعبيه شده است.

خط مشي امنيتي .NET (مزيتي براي كاربران و طراحان) تلاشي منظم براي انتشار تصويري از ويندوز به عنوان برنامهاي با امنيت ضعيف، ميباشد. دراين سياست چهار تكنيك مهم ادغام شدهاند:

· اصلاح كدها: هنگامي كه مجري سايت اصلاح زيادي را فال ميكند، تمامي كدهاي .NET مورد بررسي قرار ميگيرند اين عمل از قواعد سيستم مدل شيءاي تبعيت ميكند. بعنوان مثال اگر عبارتي را به يك

متغير نسبت دهيد، انواع كدها بايد براساس موارد تعيين شده از سوي ساختارهاي ذاتي مورد تأييد قرار گيرند. اين امر شامل اشيايي است كه مي‌خواهند چيزي به جز آنچه كه هستند، باشند و بر روي تمامي كلاسهاي نقض امنيت عمل مي‌كنند.

· اصلاح مبدأ: هر عنصر برنامه .NET مي‌تواند و معمولاً بايد با استفاده از رمزگذاري كليد عمومي 128 بيتي امضاء شود، اين عمل از شخصي نمودن منبع نرم‌افزار ديگري، جلوگيري به عمل مي‌آورد.

· مكانيسم مجوز: هر عنصر برنامه مي تواند مجوز دقيقي را مشخص نمايد و خواننده برنامه بايد آن را در اختيار داشته باشد، تا بتواند فايل را بخواند، فايل را بنويسد به DNS دسترسي داشته باشد. علاوه براين مكانيسم ¢¢Stack Wall¢¢ تضمين مي‌نمايد كه اگر شما به مجوزي نياز داشته باشيد بتوانيد آن را اطلاع دهيد، بنابراين اگر A اجازه تماس با C را نداشته باشد، نمي‌تواند اين محدوديت را با تماس با B برطرف نمايد. (B داراي مجوز است)

· نظريه اصلي: عناصر نرم‌افزاري مي‌توانند در طول عمر خود داراي وظايف متفاوتي باشند بطوريكه هر وظيفه امكان دستيابي به سطوح معيني از امنيت را بدست دهد. اين نظريه شامل متغيرهاي از پيش تعريف شده و متغيرهايي كه توسط برنامه‌نويسان تعريف مي‌شوند، نيز مي‌گردد.

ايجاد نگارش
.NET براي طراحان به جاي كاربران نهايي (اگرچه كه آنها بطور غيرمستقيم بهره‌مند مي‌شوند)، مكانيسمي ساده ولي قدرتمند فراهم نموده كه به برنامه‌هاي كاربردي امكان مي دهد مشخص نمايد چه نسخه‌اي براي ماژول‌هاي مورد استفاده آنها، قابل پذيرش است. بنابراين تاكنون ويندوز دوخط مشي نگارش گذاري را كه رضايت‌بخش نبوده، ارائه نموده است. با استفاده از كتابخانه‌هاي اتصال ديناميك (DLLها) هرنگارشي مي‌تواند جايگزين نگارش ديگر شود. اين امر به عدم سازگاري قابل ملاحظه‌اي و "Dell Hell" منجر مي‌گردد. برنامه‌اي يك نسخه جديد DLL را نصب مي‌كند كه جايگزين نگارش قبلي مي‌شود. بعد ناگهان با كمال تعجب، بعضي از برنامه‌ها ديگر عمل نمي‌كنند. خط مشي ديگر، COM مي‌باشد كه از اين وضعيت جلوگيري به عمل مي‌آورد البته بايد هزينه اين مسئله كه احتمالاً نسخه‌اي جديد و ناسازگار باشد را نيز پرداخت.

مدل نگارشي .NET خط مشي استاندارد شماره‌گذاري نگارش را تعريف نموده و به شما امكان مي‌دهد تا مشخص نماييد كه چه نگارشي براي عناصر برنامه مورد نياز شما، لازم مي‌باشند.

مدل مؤلفه‌اي

.NET مدل مؤلفه‌اي جديدي را نيز كه براساس مفاهيم شي‌گرا مي‌باشد، ارائه مي‌كند. با حذف تفاوتهاي ميان يك عنصر برنامه و يك مؤلفه نرم‌افزاري، .NET مزاياي مهمي را نسبت به فناوري‌هايي چون COM و Corba دربر خواهد داشت. از آنجايي كه يك عنصر برنامه، مجموعه‌اي كاملاً مشخص از رابطها را فراهم مي‌نمايد، درنتيجه ساير عناصر مي‌توانند از آن استفاده نمايند.

براي تبديل يك عنصر نرم‌افزاري به يك مؤلفه قابل استفاده مجدد، COM و Corba به نوشتن توضيحي براي رابط دريك IDL خاص، نياز دارند. .NET از IDL خلاص مي‌شود. شما مي‌توانيد از يك عنصر برنامه .NET مستقيماً به عنوان يك مؤلفه بدون نياز به بسته‌بندي بعدي، استفاده نماييد چون قبلاً به اطلاعات لازم تجهيز شده‌اند.

مدل مؤلفه‌اي .NET بسيار ساده‌تر از COM، (استاندارد پيشين براي توسعه مبتني بر مؤلفه برنامه‌هاي ميكروسافت) مي‌باشد. اين مدل نه تنها با IDL بلكه با بسته‌بندي‌هاي COM نيز فاصله دارد.

طراحان ازاين تغيير روش نسبت به COM بسيار استقبال كرده‌اند. هرچند كه در كوتاه‌مدت دو روش مي‌تواند موجود باشد: COMInterop ، مكانيسمي با قابليت عملكرد در درون برنامه، اين تغيير را تسهيل مي‌نمايد.

اجزا و مؤلفه‌هاي خود مستندساز
حقه به كار رفته در حذف IDL براي بكارگيري اطلاعات رابط كه قبلاً در كد مبدأ و دريك زبان شي‌گرا موجود بوده است، مورد استفاده قرار مي‌گيرد. دراين شرايط، متن برنامه شامل فهرستي از كلاسهاي موجود دريك عنصر برنامه، فهرست امكانات هركلاس (مسيرها/ روش‌ها و ويژگي‌ها/فيلد) و اطلاعات ضروري براي نام هريك از امكانات ، تعداد و انواع آرگومانها، توابع يا رويه‌ها و نوع نتايج در صورت وجود مي‌باشد.

مترجم‌هاي زبانهاي پشتيباني كننده از سوي .NET ، اين اطلاعات را بصورت فوق داده، در كد ايجاد شده نگهداري مي‌كنند (در واقع اين نظريه ايجاد اجزايخودمستندساز مي‌باشد).

فوق داده به اطلاعات از پيش تعريف شده، محدود نمي‌شود. شما مي‌توانيد از ويژگي‌هاي خاصي براي مشخص نمودن هرنوع اطلاعات موجود در فوق داده استفاده كرده و آن را پس از ترجمه، همراه با آن جزء مؤلفه نگهداري نماييد. دسترسي به فوق داده، از طريق مختلف و متعددي امكان‌پذير است.

ILDASM (برنامه زبان واسطه كه كد منبع را به زبان اسمبلي تبديل مي‌كند) ابزاري گرافيكي است كه به شما امكان بررسي يك رابط اسمبلي را به منظور حصول اطلاعات از فوق داده به طريقه بصري، بدست مي‌دهد. علاوه براين هر برنامه‌اي مي‌تواند از طريق كتابخانه Reflection به فوق داده دسترسي داشته باشد. چونكه فوق داده در قالب XML نيز موجود مي‌باشد. هر برنامه‌اي كه بخشي از .NET باشد، مي‌تواند اطلاعات پيرامون مؤلفه‌ها را بدست آورد.

يكي از مثالهاي يك برنامه فوق داده، " Contract Wizard " است كه برنامه‌نويسان مي‌توانند از آن براي تجهيز عناصر ترجمه شده از زبانهايي به جز زبان Eiffel همراه با قراردادهاي شبيه Eiffel استفاده نمايند. برنامه‌نويسان از ويزارد براي بررسي كلاسها و امكانات آنها استفاده نموده و تصميم مي‌گيرند كه آيا عناصر قرارداد (پيش شرطها، شرايط پس از قرارداد و متغيرهاي كلاس) را بيافزايند يا نه؟ ويزارد نيز از اين ورودیها براي ايجاد مؤلفه‌هاي پروكسي كه قراردادها را اجرا نموده و مؤلفه‌هاي اصلي غيرقراردادي را فرا مي‌خواهند، استفاده مي‌كنند. مشخص نيست كه چگونه مي‌توان بدون استفاده از داده، چنين ابزاري را ايجاد نمود.

قابليت عملكرد ميان زبانها
بسياري از شركتها سرمايه قابل ملاحظه‌اي را در بخش نرم‌افزار براي زبانهاي برنامه‌نويسي مختلف، قرار داده‌اند قابليت عملكرد .NET در ميان زبانهاي مختلف براي جلوگيري از به هدر رفتن سرمايه تعبيه شده است. سطح قابليت عملكرد آن بسيار فراتر از نسخه‌ها و برنامه‌هاي قبلي است. بعنوان مثال ماژولهايي كه به زبانهاي مختلف نوشته مي‌شوند مي‌توانند يكديگر را صدا زده (درحالت كلاسها) و از يكديگر ارث ببرند. برنامه‌هاي اشكال‌زدايي درون Visual Studio.NET به سادگي اين مرزهاي ميان زبانها را برداشته‌اند. اين قابليت در ادغام زبانها بدون صرف هزينه براي بسته‌بندي‌هاي خاص يا IDL ، موجب گرديده تا طراحان بهترين زبان را براي هريك از بخشهاي يك برنامه برپايه انتظاراتي كه از آن مي‌رود و آن را به رابطي خودكار و هماهنگ با ساير اجزاء تبديل مي‌نمايد، انتخاب كنند.

مدل شي‌اي
مدل شي‌اي و ويژگي‌هاي زبان مشترك (CLS) ، اساس قابليت عملكرد .NET را در زبانهاي مختلف تشكيل مي‌دهد. مدل شي‌اي مجموعه‌اي از مفاهيم (شبيه زبان شي‌گرا بدون املاء يا هرچيز ديگري كه ظاهر خارجي برنامه را تحت پوشش قرار مي‌دهد) است كه سيستم نوع را مشخص مي‌نمايد يعني اينكه كلاس چيست؟ شيء چيست؟ و چگونه كلاسها از يكديگر ارث مي‌برند؟ و غيره.

بعضي از گزينه‌هاي آن كه كاملاً تحت تأثير Java قرار دارند، تأسف‌آور هستند. درحالت خاص تمايزي روشن ميان كلاسها و رابطها كه بدون هيچ‌گونه زيربرنامه يا ويژگي خاص مشخص مي‌شود، وجود دارد. بعضي از مهمترين مزاياي فناوري شي‌اي ناشي از قابليت آن در پوشش‌دهي طيف وسيعي از ويژگي‌هاي محض و طراحي جوانب فني‌تر و دقيق‌تر اجرا از طريق نظريه كلاس مي‌باشد. البته شباهت آن به جاوا، محدوديتي در ميراثهاي متعدد رابط، ايجاد مي‌كند، مدل فعلي از عموميت بخشي يا كلاسهايي با انواع پارامتردار پشتيباني نمي‌كند ولي طرحهايي در دست است كه براساس آنها مكانيسم عمومي‌سازي در نسخه بعدي قرار داده شود.

مدل شي‌اي .NET مجموعه‌هايي از مفاهيم پايه را براساس سياست قدرتمند شي‌گرا فراهم مي‌نمايد. تمامي مترجم‌هاي زبانها نوعي كد واسطه، MSIL (زبان واسطه ميكروسافت) يك زبان ماشين براي يك ماشين شي‌گرا و محض كه در مورد اشياء ارث بردن متدها و پيوندهاي ديناميكي و ساير مفاهيم شي‌گرا اطلاعاتي دارد توليد مي‌كنند. هنگامي كه مترجمي MSIL را ايجاد كرده و شبه‌داده را در كنار آن قرار مي‌دهد، عنصر برنامه .NET را توليد كرده كه مي‌تواند با هر عنصر ديگر برنامه محاوره نمايد.

مدل اجرا
زمان اجراي زبان مشترك .NET برنامه اجرايي ابتدايي را فراهم مي‌نمايد. مترجم زبان، كد MSIL را كه تفسير نشده ولي دوباره به كد تبديل مي‌شود را توليد مي‌كند. بنابراين برنامه اصلي تنها كد مشترك را اجرا مي‌كند.

بي‌ثباتي
دراثر بي‌ثباتي‌هاي پيش‌فرض زيربرنامه‌هاي منحصربفرد درصورت تقاضا در اجراي برنامه دچار بي‌ثباتي شده و سپس درحالت بدون ثبات نگهداري مي‌شوند. با استفاده از EconoJit ، گونه‌اي كه براي دستگاههايي با حافظه كوچك طراحي شده است شما مي‌توانيد حداكثر فضا را تعيين نموده و از سياستي پنهان براي حذف يك زيربرنامه بدون ثبات و ايجاد فضا براي استقرار يك زيربرنامه جديد، استفاده نماييد. امكان ديگري كه دراختيار داريد Prejit است كه ترجمه يكباره كل برنامه را انجام مي‌دهد.

كد مرتب شده
كدي كه به مفهوم اجرا توسط زمان اجراي .NET مي‌باشد را "كد مرتب شده " مي‌نامند. اين كد از تمامي مزاياي تسهيلات زمان اجرا مانند جمع‌آوري آشغالها ، انجام استثنائات و مسائل امنيتي، برخوردار مي‌باشد.

جمع‌آوري آشغالها مشكلات را براي C++ افزايش مي‌دهد (C++ يكي از زبانهاي ارائه شده از سوي .NET مي‌باشد) C++ داراي سيستم نوع lax است كه امكان تبديلات را ميان بيشتر گونه‌هاي اختياري بدست مي دهد ولي شروط لازم براي GC مطمئن را نقض مي‌كند. در واقع C++ كلاسيك نمي تواند كد مرتب شده را در .NET ايجاد نمايد.

براي كلاسهاي مربوطه، شما بايد از " Managed C++ " گونه‌اي جديد از زبان كه محدوديت قدرتمندي را در ادغام گونه‌ها اعمال مي‌كند و در واقع بسيار به C# نزديك است، استفاده نماييد.

ميكروسافت، هشدار شديدي براي طراحان C++ ارسال نموده مبني براينكه سازگاري كامل با C ، ويژگي توضيح دهنده C++ كلاسيك را درنظر بگيرند. تركيب كلاسهاي مرتب شده و مرتب نشده، يك مسير انتقالي را ارائه مي‌كند.

فراتر از ويندوز
تنها بعضي از تسهيلات .NET داراي ويژگيهاي ويندوز هستند.چهارچوب كاري فرمهاي ويندوز، طوري است كه امكان جايگزيني API گرافيكي ويندوز و بخش گرافيكي كلاسهاي اصلي ميكروسافت (برايC++) را مي‌دهد و ASP.NET در بالاي سرويس دهنده وب IIS ميكروسافت اجرا خواهد شد. بيشتر قسمتهاي ديگر .NET ، مي‌توانند در بالاي Solaris ، Linux و سيستمهاي ديگر اجزا شوند.

آيا ما .NET را در سيستمهاي عامل غيرميكروسافت خواهيم ديد؟ اگرچه تاكنون براي اجراي آن به ويندوز نياز داريم. ولي در اواخر سال گذشته، ميكروسافت بخش‌هاي كليدي فناوري را براي استاندارد نمودن براساس ECMA (نوعي روش استاندارسازي بين‌المللي) ارسال نمود. عناصري كه هنوز مورد بحث هستند شامل زمان اجراي زبان مشترك، MSIL ، CLS ، C# و بيش از 1000 مؤلفه از كتابخانه‌هاي اصلي مي‌باشند. ميكروسافت با شدت تمام، فعاليت مي‌كند تا اين عمل را سريعتر (پيش از موعد) به پايان ببرد.

اخيراً اعلام شد سكه فعاليت MONO براي توسعه اجراي يك منبع بازار .NET برپايه ويژگي‌هاي ECMA انجام گرفته و براي اجرا بر روي برنامه‌هايي چون Linux مناسب مي‌باشد. چنين تلاشهايي نشان مي‌دهد كه محيط توسعه چند برنامه‌اي زودتر از انتظار به وقوع مي‌پيوندد.

.NET به عنوان يكي از مهمترين طرحهاي فناوري از سرمايه‌گذاري چندين بيليون دلاري دربخش تحقيقات بهره‌مند شده است. رقباي ميكروسافت نيز مسئله را درك كرده‌اند و بهمين علت در اعلاميه‌هاي اخير خود مسائلي را مطرح نموده‌اند. هيچ كس دراين صنعت، تضمين پيشگام بودن درمدت طولاني را ندارد ولي ميكروسافت پتانسيل لازم براي ظهور دائمي درصحنه را دارا مي‌باشد.

آشنائي با برنامه‌نويسي وب در .NET

همانند بسياري از فن‌آوريهاي عمده جديد، .NET يك بهبود قابل توجه نسبت به وضعيت كنوني است و نسبت به فن‌آوريهاي موجود كه مي‌توانيد ازميان آنها انتخاب كنيد، برنامه‌نويسي وب را به يك وظيفه بسيار آسانتر تبديل مي‌كند.

محيط كاري .NET چيست؟

درطول چندسال گذشته اهميت اينترنت تقريباص براي تمام حوزه‌هاي محاسباتي و پردازش اطلاعات با رشد بسيار زيادي روبرو بوده است. يك نتيجه مستقيم اين رشد آن است كه از برنامه‌نويسان خواسته مي‌شود تا كاربردهائي را ايجاد كنند كه از قابليتهاي اينترنت بهره ببرند، خواه براي پردازش گروهي، تحويل مضمون باند پهن، همكاري online و يا هريك از موارد استفاده آن. متأسفانه منشاء ابزارهاي توسعه‌اي كه قبلاً دراختيار برنامه‌نويسان قرار داشتند (نظير نسخه ششم C++ و ويژوال بيسيك)، به دنياي پيش از اينترنت مربوط مي‌شد. هريك از قابليتهاي خاص اينترنتي كه در آنها وجود داشت، فكرهاي بعدي بودند كه براي برطرف نمودن نيازهاي برنامه‌نويسان به آنها ضميمه شده بودند. بعضي از فن‌آوريهاي ابتدائي برنامه‌نويسي وب واقعاً مؤثر بودند و بسياري از كاربردهاي فوق‌العاده با آنها ايجاد شده‌اند. حتي با اين وجود نيز قابليتهاي اينترنت بدون مواجهه با مشكلات اجتناب‌ناپذير در زمينه بازده برنامه‌نويسي، باگها و نگهداري برنامه در اين ابزارهاي برنامه‌نويسي تعبيه نشده بودند. پاسخ مايكروسافت به اين وضعيت دشوار، آغاز يك تلاش از مرحله كاملاً ابتدائي بود؛ ايجاد يك چارچوب جديد از ابزارهاي توسعه كه از روز اول با ادغام پشتيباني كامل از اينترنت در نهادشان طراحي شده بودند. محيط كاري .NET تنها به برنامه‌نويسي وب مربوط نمي‌شود بلكه راهبردهائي را براي توسعه كاربردهاي سنتي‌تر دسك‌تاپ نيز فراهم مي‌كند. با اين وجود شكي نيست كه هيجان مربوط به .NET بيشتر به قابليتهاي اينترنت آن مربوط مي‌شود و ما براساس تجربياتمان مي‌توانيم به شما بگوئيم كه اين قابليتها عالي هستند.

.NET داراي دوبخش اصلي است:

· ابزار توسعه نرم‌افزاري محيط كاري .NET يك مجموعه گسترده از كلاسها و اينترفيسها، همراه با عناصر پشتيباني كننده مختلف است. كه طوري طراحي شده‌اند كه تقريباً براي برآورده ساختن هرنياز برنامه‌نويسي با يكديگر كار كنند. محيط كاري .NET بايستي بر روي هر سيستمي كه براي اجراي كاربردهاي .NET و يا توسعه آنها مورد استفاده قرار خواهد گرفت نصب شده باشد. ..NET SDK شامل كامپايلرهائي براي سه زبان است:

C++ قابل احترام، يك زبان جديد با نام C# و ويژوال بيسيك كه بطور گسترده‌اي تقويت شده است.

· محيط برنامه‌نويسي ويژوال استوديو. اين ابزار پيشرفته، مجموعه فراگيري از ابزارهاي برنامه‌نويسان نظير ويرايشگرهاي كد، ابزارهاي طراحي اينترفيس و ديباگرها را تأمين مي‌كند كه وظيفه ايجاد كاربردهاي .NET را به ميزان زيادي تسهيل مي‌كنند.

مفاهيم بنيادي در برنامه‌نويسي .NET

: Common Language Runtime

CLR زير ساختاري است كه .NET براي اجراي تمام كاربردها از آن استفاده مي‌كند. CLR شامل مجموعه انبوهي از كتابخانه‌هاي كلاس است كه شما مي‌توانيد در كاربردهايتان از آنها استفاده كنيد. بعضي از كلاسهاي جالبي كه ارتباطي به اينترنت ندارند، شامل يوتيليتي‌هائي هستند كه به شما امكان مي‌دهند گزارش رويداد ويندوز .NT اكتيودايركتوري، سرويسهاي Cryptographic ، شمارشگرهاي عملكرد، سرويسهاي COM+ ، ترتيب اشياء و سرويسهاي بومي ويندوز را اداره نمائيد. CLR همچنين مراقب كامپايل نمودن كد IL به كد ماشين است.

CLR سرويسهاي ديگري نظير پشتيباني از اجراي چند كاربرد .NET در داخل يك روند با اين تضمين كه بايكديگر تداخل نخواهند داشت را دربرمي‌گيرد. اين ويژگي نتيجه مفهومي با نام Type Safety است. هر زبان .NET بايستي كد Type-Safe را تضمين نمايد كه اطمينان مي‌دهد آرايه‌ها و مجموعه‌ها هيچگاه بطور غيرمستقيم مورد دسترسي قرار نمي‌گيرند. كامپايلر ويژوال بيسيك هميشه درهنگام كامپايل كد شما به IL ، Type Safety را انجام مي‌دهد.

كلاس‌هاي چارچوب كاري
عجيب است. يكي از دلايلي كه اكنون ويژوال بيسيك .NET بسيار قدرتمندتر است اين است كه كار بسيار كمتري انجام مي‌دهد. تا ويژوال بيسيك6 ، كامپايلر مجبور بود نسبت به كامپايلر زباني مثل كامپايلر زبان C++ كار بسيار بيشتري انجام دهد. دليلش اين است كه مقدار بسياري از عمليات درخود ويژوال بيسيك تعبيه شده بود ولي در C++ اين كار از طريق كلاسهاي اضافي انجام مي‌شد واين موضوع باعث شد كه به روز كردن و افزودن ويژگي‌ها به اين زبان آسانتر شود و سازگاري بين برنامه‌هايي كه داراي كتابخانه‌هاي يكسان و مشترك بوده‌اند افزايش يابد.

اكنون VB.NET ازاين الگو استفاده مي‌كند. بسياري از ويژگي‌هايي كه قبلاً مستقيماً در ويژوال بيسيك بود اكنون در كلاسهاي چارچوب جايگذاري شده‌اند. مثلاً اگر بخواهيد ريشه دوم بگيريد به جاي استفاده از عملگر ويژوال بيسيك از متدي كه در كلاس System.math وجود دارد استفاده مي‌كنيد. اين امر موجب مي‌شود كه زبان بسيار ساده‌تر و توسعه‌پذيرتر شود.

: Assemblies

تمام كد .NET به Assemblies كامپايل مي‌شود، يك assembly از بيرون دقيقاً مانند يك كتابخانه لينك ديناميك(.dll) و يا يك برنامه قابل اجرا (.exe) به نظر مي‌رسد. چند بيت اول درون هر Assembly شامل كد ماشيني است كه در زمانيكه Assembly توسط يك برنامه ديگر و يا يك كاربر مورد دسترسي قرار مي‌گيرد، CLR را درخواست مي‌كند.

بسته به اينكه چقدر از Assembly بوسيله CLR كامپايل شده باشد، درون Assembly تركيبي از IL و كد ماشين است. Assembly همچنين حاوي Metadata است. Metadata براي يك برنامه .NET عبارت است از:

· اطلاعاتي درباره هريك از كلاسهاي عمومي و يا نوع آن، نظير نام، اينكه كلاس از چه كلاسهائي مشتق شده است و هر اينترفيسي كه كلاس بكار مي‌گيرد.

· اطلاعاتي درباره هريك از شيوه‌هاي عمومي درهر كلاس، شامل نام و مقدار برگشتي.

· اطلاعاتي درباره هريك از پارامترهاي عمومي براي هر شيوه، شامل نام و نوع هر پارامتر.

· اطلاعاتي درباره هر سرشماري، شامل نام و مقاديري براي هريك از اعضاي سرشماري.

· اطلاعاتي درباره نسخه خاص Assembly

Metadata تعدادي از مشكلات را حل كرده و بعضي از ويژگيهاي مطلوب را فعال مي‌نمايد. يكي از مشكلاتي كه فايلهاي DLL دارند اين است كه راهي براي بررسي توابعي كه پشتيباني مي‌كنند وجود ندارد. فايلهاي COM نيز كه هميشه براساس فايلهاي DLL ساخته مي‌شوند نيز همين مشكل را دارند.

با داشتن چنان داده‌هاي ارزشمندي در هر Assembly ، محيط كاري .NET براي اجراي Assemblies نيازي به استفاده از رجيستري ندارد. اگر يك برنامه .NET نياز به استفاده از يك Assembly ديگر .NET داشته باشد، آنها تنها بايستي بر روي همان دايركتوري قرار داشته باشند. اين بدان معني است كه اگر مي‌خواهيد برنامه .NET را بر روي كامپيوتر ديگري مستقر نمائيد، كافي است تنها فايلهاي موردنياز را كپي نمائيد.

آيا .NET واقعاً قابل اعتماد است؟

امروزه در حوزة فن آوري اطلاعات ابتكار Microsoft.NET و البته امنيت قابل توجه است.

كلمه عبور
يكي از ويژگيهاي جديد به مديران شبكه امكان مي‌دهد از ورود كاربران با كلمات عبور خالي جلوگيري كنند. به كاربران امكان داده شده است كه بصورت محلي با Accountهايي كه داراي كلمه عبور خالي هستند وارد سيستم شوند، اما از آنجائيكه آنها در آن هنگام به domain متصل نيستند، در معرض حملات اينترنت قرار نمي‌گيرند. بعبارت ديگر كاربران دور نخواهند توانست با accountهائي كه كلمات عبور خالي دارند به Windows.NET Server متصل شوند. مانند هركسي كه دريك بخش IT كار مي‌كند بخوبي مي‌دانيم كه اگر به كاربران اجازه دهيم كه كلمات عبور خالي را براي Accountهائي خود داشته باشند، بسياري از كاربران ترجيح مي‌دهند از كلمات عبور خالي استفاده كنند. اينكار مانع آن مي‌شود كه در داخل يادداشتها و يا در زير صفحه‌كليد، به دنبال كلمه عبور خود بگرديد. با سرور .NET حتي اگر accountهاي سرپرست محلي بر روي ايستگاههاي كاريتان خالي مانده باشند، شركت شما در برابر حملات اينترنتي محافظت شده است. در صورتيكه كاربران از كلمات عبور خالي استفاده كرده باشند، تنها مي‌توانند بصورت محلي وارد سيستم شوند.

تغييرات سرويسهاي اطلاعاتي اينترنت

در Windows.NET Server تقريباً تمامي تركيبات IIS بطور پيش فرض نصب مي‌شوند. با تبديل شدن سرويسهاي وب به منبع عمده ويروسها و ساير حمله‌ها از دنياي خارج، مايكروسافت تصميم گرفته است تا امنيت در windows.NET Server را سخت‌تر نمايد. اين بخش هيچگونه add-ons اضافي را بطور پيش‌فرض نصب نمي‌كند. در واقع بصورت پيش‌فرض هيچگونه پشتيباني براي صفحات Active Server ASP)) وجود ندارد، تنها صفحات HTML ثابت، پشتيباني شده‌اند.

يك ويژگي مطلوب ديگر اين است كه هنگاميكه شما براي اولين بار كنسول مديريت، IIS را اجرا مي‌كنيد، IIS 6.0 Security Lockdown Wizard به شما امكان مي‌دهد، تنها ضمائم دلخواه خود را فعال نمائيد. پيكربندي پيش‌فرض IIS دريك سرور .NET بسيار بيشتر از اسلاف آن، ايمن مي‌باشد.

ابزارهاي خط دستور

يك ويژگي جديد ديگر، توانائي مديريت اختيارات كاربر از طريق يك ابزار خط دستوري است. اين به سرپرستان امكان مي‌دهد براي اضافه كردن نام كاربران و كلمات عبور آنها از اسكريپت‌ها استفاده كنند. استفاده از ابزارهاي خط دستور مي‌تواند براي سرپرستاني كه Enterprise هاي بزرگي را مديريت مي‌كنند ضروري باشد. بعلاوه، ابزارهاي خط فرمان متعدد ديگري نيز قابل دسترسي هستند، نظير takeown.exe كه به سرپرستان امكان مي‌دهد مالكيت فايلهاي بي‌صاحب را به عهده بگيرند و xcacls.exe كه يك ويرايشگر فهرست كنترل دسترسي (ACL) گسترده است.

پيشگيري از سرريزهاي بافر

مشكل سرريز بافر درحال حاضر، يكي از چالشهاي عمده امنيتي محسوب مي‌شود. هكرها سالها است كه از سرريز بافر بهره‌برداري كرده‌اند. براي مثال در نوامبر 2000، يك آسيبپذيري شناخته شده در NetMon مايكروسافت وجود داشت كه درآن يك سرريز بافر مي‌توانست موجب بعضي مشكلات جدي امنيتي شود. اگر كسي يك داده ناهنجار را به قسمتي كه سرپرست با استفاده از NetMon درحال كنترل آن بود مي‌فرستاد، اين مسئله مي‌توانست موجب يك سرريز بافر گردد. در بعضي از موارد اين مسئله تنها موجب عدم كارآئي NetMon مي‌شد. با اينحال دربعضي از موارد نيز اين مسئله مي‌توانست موجب اجراي كد فرستنده بر روي كامپيوتر سرپرست در مضمون امنيتي سرپرست گردد. بدون شك اين مسئله مي‌تواند فاجعه‌آميز باشد.

مايكروسافت در Windows.NET تصميم گرفته است تا بررسي سرريز بافر را در كامپايلر Visual C استفاده شده براي Windows.NET انجام دهد. بعلاوه آنها مراحل ديگري را نيز براي تحليل سرريز بافر براي كد درنظر گرفته‌اند. با اين سنجشهاي احتياطي، Windows.NET به شكل اميدوارکننده‌اي در آسيب پذيري كمتري در برابر حملات سرريز بافر دارد.

سرويسهاي كمتر بر روي منو

Windows 2000 Server در مقايسه با Windows NT شامل سرويسهاي بسيار بيشتري است كه بطور پيش‌فرض نصب شده و آغاز مي‌شوند، خصوصاً بر روي كنترلرهاي حوزه. بر روي يك كنترلر حوزه استاندارد Windows 2000، من توانستم 65 سرويس در حال اجرا را بشمارم. 24 سرويس اضافي نيز نصب شده اما آغاز نشده بودند. اين تعداد سرويس نصب شده بر روي يك كنترلر حوزه بسيار زياد است! مايكروسافت براي سالها به ما تذكر داده است كه براي محفوظ نگهداشتن سرورها، پروتكلها و سرويسهائي كه به آنها نياز نداريم حذف نمائيم. نيازي به گفتن نيست كه اين تعداد كثير از سرويسها، يك نگراني امنيتي جدي براي اگثر مؤسسات است، زيرا هكرها مي‌توانند ازاين سرويسها بهره‌برداري، كنند. در نتيجه مايكروسافت تغييرات متعددي را بر روي مدل امنيتي اعمال نموده است.

اولاً Services.exe ديگر حاوي هيچگونه كدي كه به امنيت مربوط نباشد نيست. بعلاوه مايكروسافت اعلام كرده است كه آنها تعداد سرويسهايي كه به طور پيش‌فرض در سرورهاي Windows.NET اجرا مي‌شوند را كاهش خواهند داد. براساس بعضي پيش‌بيني‌ها، در مقايسه با Windows 2000 تعداد سرويسهائي كه بر روي سرورهاي Windows.NET اجرا مي‌شوند 20 عدد كمتر خواهد بود.

يك گام ديگر بسوي محفوظ نمودن سرويسها و شيوة عملكرد آنها، به ميزان امتياز هر سرويس مربوط مي‌شود. امروزه اكثر سرويسها تحت يك Account Service امتيازبندي شده داخلي با نام سيستم محلي اجرا مي‌شوند. اين Account تاحد زيادي داراي تمامي امتيازات سرپرستي است. كاربران نمي‌توانند تحت اين Account وارد سيستم شوند، اما اين Account Service كنترل كاملي بر روي كامپيوتر شما دارد. چنانچه مزاحمين به اين Account دسترسي پيدا كنند قادرند نه تنها بر روي كامپيوتر شما بلكه در سرتاسر شبكه خرابيهايي را به بار آورند.

فلسفه Service Account در سرور Windows.NET تغيير كرده است. حالا ما دو Service.Account خواهيم داشت: Local Service Account وNetwork Service Account . اكثر سرويسها تحت اختيارات يكي از اين دو Account اجرا خواهند شد. Local Service Account به كامپيوتر محلي محدود خواهد بود و هيچگونه دسترسي به شبكه نخواهد داشت. Network Service Account با استفاده از اختيارات Machine Account كامپيوتر، قادر با تعامل با شبكه خواهد بود. اين مدل، امنيت بهبوديافته‌اي را عرضه خواهد نمود و كنترل بيشتري را بر روي سرويسهاي درحال اجرا در شبكه در اختيار سرپرستان قرار مي‌دهد.

پيشرفتهاي امنيتي در محيط كاري .NET. يك قدم بزرگ دراين تمركز جديد بر روي امنيت است. حتي منتقدين مايكروسافت نيز درحال تحسين تلاشهاي اخير آنها هستند و تحت تأثير پيشرفتهاي امنيتي در Window.NET Serverقرار گرفته‌اند.

.NET از ديدگاه برنامه‌نويسي

نمي‌توان منكر اين شد كه مايكروسافت با ابداع فناوري .NET ، فرآيند توسعه نرم‌افزارهاي ويندوزي را دگرگون كرده است. البته، اختلاف نظر و بحث‌هاي زيادي بر سر خوب يا بد بودن .NET مطرح شده است. بعضي‌ها (در واقع منفي‌بافان) مي‌گويند تغييراتي كه دراين محصول نسبت به محصولات قبل پديد آمده‌اند به حدي زياد و وسيع هستند كه پذيرفتن آن را بسيار سخت كرده است. اين افراد معتقدند كه وسعت اين تغييرات چنان زياد است كه كسي قبول نمي‌كند برنامه‌هاي موجود خود را با استفاده از اين فناوري جديد جديد بازنگري و بازنويسي كند. آن‌ها همچنين از تغييراتي كه در زبان ويژوال‌بيسيك صورت گرفته شكايت دارند.

بايد اعتراف كنيم كه اين افراد زياد هم اشتباه نمي‌كنند. انتقال برنامه‌هاي بزرگ به اين فناوري، بدون طراحي دوباره و بازنويسي آن‌ها اگر غيرممكن نباشد، بسيار دشوار است، ولي به هر ترتيب، براي بهره‌بردن از قابليت‌ها و مزاياي يك سيستم و زبان جديد بايد زحمت طراحي مجدد را بپذيريم. مهندسين بايد وقت زيادي را صرف يادگيري و حرفه‌اي شدن بااين فناوري جديد بكنند. چراكه تغييراتي بنيادي در ساختار زبان برنامه‌نويسي صورت گرفته و ساختمان .NET (با بيش از 6500 كلاس) واقعاً بزرگ است. زبان .NET بسيار قدرتمند و حتي مطمئن‌تر شده است. درعين حال، هنوز هم آسان است. برنامه‌نويساني كه با ..NET كار مي‌كنند برنامه‌هايي توليد خواهند كرد كه تا قبل ازاين حتي فكرش را هم نمي‌كردند.

زبان مشترك در زمان اجرا

ويژگي زبان مشترك در زمان اجرا امكان اضافه شدن هر زباني به محيط ويژوال استوديو و توليد آسان برنامه‌هاي باينري را فراهم ساخته است. مقصود اين است كه رسم الخط زباني است كه استفاده مي‌شود بايد كمابيش يك نظر شخصي باشد و تأثيري روي قابليت‌ها يا عملكرد برنامه نگذارد.

وقتي برنامة VB.NET خود را كامپايل مي‌كنيد، اولين كاري كه صورت مي‌گيرد ترجمة آن به يك «حالت واسطه» است كه «زبان مياني» يا IL (مخفف Intermediate Linguage) ناميده مي‌شود. اين زبان خيلي شبيه اسمبلي است. تمام زبان‌ها در محيط ويژوال استوديو به IL ختم مي‌شوند. و اين زبان دست آخر به يك فرمت باينري اجرا شدندي تبديل خواهد شد.

فايدة اين كار چيست؟ اگر بخواهيد زباني بنويسيد كه در محيط ويژوال استوديو قابل استفاده باشد، فقط بايد تا حد توليد IL پيش برويد. اين كار را خيلي آسان‌تر مي‌كند، چراكه مي‌توانيد از كامپايلر نهايي كه مايكروسافت قبلاً نوشته استفاده كنيد، يعني همان كامپايلر IL. بعلاوه، مي‌توانيد از هر زباني كه فكر مي‌كنيد براي انجام كار شما مناسب‌تر است استفاده كنيد، بدون اينكه انتخاب زبان تأثيري محسوس روي قابليت‌هاي برنامه شما داشته باشد. علاوه بر تمام اينها، يك مزيت بزرگتر هم وجود دارد: قابليت همكاري بين كامپوننت‌ها . هر زباني كه در ويژوال استوديو با زبان مشترك كار مي كند«كد مديريت شده» ناميده مي‌شود. كمپوننت‌هايي كه با كد مديريت شده نوشته شده باشند مي‌توانند بدون هيچ دردسري كامپوننت‌هايي را مورد استفاده قرار بدهند كه به زبان‌هاي ديگر كه مديريت شده نوشته شده باشند. يعني مثلاً تابعي را در ويژوال بيسيك مي‌نويسيم و آن را در C فرا مي‌خوانيم. كلاسي را در C++ مي‌سازيم و در Small Talk مورد استفاده قرار مي‌دهيم. همة اين‌ها مي‌توانند باهم كار بكنند ، چراكه دست آخر همه IL هستند. به همين دليل هركس مي‌توانند واقعاً از هرزباني دوست دارد استفاده بكند.خود زبان عملاً اهميتي ندارد و يك انتخاب شخصي و سليقه‌اي است كه عمدتاً به صرف و نحو آن برمي‌گردد.

كار با ديتا به كمك ADO.NET

دسترسي به بانك اطلاعات هميشه بخش مهمي از فرآيند توسعه نرمافزار بود است. اين كار سابقاً بسيار دشوار بود و اغلب به APIهاي خاص نياز بود. بعد ازيك دوره تاريكي، ODBC به ميدان آمد كه به توليدكنندگان امكان مي‌داد API هاي خود را به صورت استاندارد شده توليد و عرضه كنند. به اين ترتيب مردم مي‌توانستند با درنظر گرفتن يك « واسطه » برنامه خود را بنويسند و بعداً بدون تغيير دادن تمام كد، بين بانك‌هاي اطلاعات مختلف سوئيچ كنند. ولي واسطه ODBC پيچيده بود و يادگيري آن دشوار. راه‌حل مايكروسافت ADO بود، مجموعه‌اي از اشياء اوليه كه دسترسي به داده‌ها را آسانتر مي‌كرد. درست است كه ADO يك پيشرفت محسوب مي شد ولي مشكلات خاص خودش را داشت و درضمن چيز كاملي هم نبود. بالاخره ADO بيرون آمد و واقعاً انقلابي به پا كرد. اين فناوري زود رواج پيدا كرد و مورد استفادة اكثيرت قرار گرفت.

با پيدايش اينترنت و برنامه‌هاي پيچيده مبتني بر وب كه نياز به كار با بانك اطلاعات داشتند، ADO خوب جواب مي‌داد. اگر مي‌خواستيم يك دسترسي ثابت و مستقيم به بانك اطلاعات داشته باشيم. ADO از قابليت‌هاي اساسي « غير اتصالي » برخوردار بود ولي ارصال داده‌ها از طريق يك اتصال HTTP با ADO بسيار مشكل بود. مايكروسافت با ساخت ADO.NET اين مشكل را نيز حل كرد؛ يك ADO جديد كه اينترنت را مي شناسد و مي‌تواند با چارچوب جديد NET . همكاري كند . اين دو فناوري تفاوت هاي زيادي با هم دلرند ، ولي من فقط به دو مورد اشاره مي كنم كه از بقيه مهم تر هستند . اولاً ، ADO.NET بصورت يك DATA SYSTEM غير اتصالي كار مي‌كند . اين به آن معني است كه داده ها ابتدا از بانك اطلاعات به داخل Container ريخته شده و سپس بدست برنامه اي سپرده مي شوند كه به آنها نياز دارد . اتصال به بانك اطلاعات فقط در هنگام ضرورت برقرار مي شود و پس از رفع نياز بلافاصله قطع خواهد شد تا ساير برنامه ها هم بتوانند از منابع بهره برداري كنند . در محيطي مثل اينترنت ، كه درخواست هاي زيادي بطور همزمان در آن صورت مي گيرند ، منابعي چون اتصالات بانك اطلاعات ارزش زيادي دارند ، راحت بدست نمي آيند . توانايي ADO.NET در استفاده از اتصال بانك اطلاعات براي مدت كوتاه و سپس آزاد كردن آن ، امتيازي است كه براي برنامه هاي اينترنتي بسيار مفيد است . دومين اختلاف مهم اين است كه داده ها در Data Set هاي ADO.NET در قالب XML ذخيره مي شوند ، كه يك فرمت استاندارد مورد پذيرش صنايع است . اين در حالي است كه Record Set هاي ADO با فرمت دروني خودشان نگهداري مي شوند . اين فرمت براي برنامه اي كه اصلاً چيزي از Record Set نمي داند هيچ فايده اي ندارد . Record Set هاي مزبور نمي توانند مثلاً براي ارسال داده ها به يك « جاوااپلت » در صفحه اي از وب مورد استفاده قرار بگيرند . ولي ماهيت ايكس ام الي Data Set هاي ADO.NET در هر كلاينتي كه زبان XML را بفهمد قابل استفاده است . ADO.NET اشياء و كامپوننت هاي بسيار مفيد دارد .

ساخت صفحات وب با ASP.NET

ASP امروزه يكي از گسترده‌ترين روشهاي توسعه برنامه‌هاي كاربردي وبي شده است. آسان بودن اين تكنولوژي دنياي وب سرويسهارا كه زماني فقط در اختيار گروه كوچكي از برنامه‌نويسان يونيكس بود كه بلد بودند با PERL يا C به پياده‌سازي CGI بپردازند، پيش روي ميليون‌ها برنامه‌نويس ويژوال بيسيك در سراسر دنيا گشوده است. وي‌بي اسكريپت طرف سرور دركنار اشياء ذاتي ASP (از قبيل Session , Application , Response , Request) به ميزان قابل ملاحظه‌اي از سختي‌ها وزمان آموزش افراد كاسته است. البته اين آسان بودن بدون جنبه منفي به دست نيامده است. وي‌بي اسكريپت يك زبان interpret شده است، يعني وب سرور بايد در پاسخ به درخواستهايي كه دريافت مي‌كند، به Parse كردن و كامپايل كردن صفحات ASP در سرور بپردازد. اسكريپت ASP بصورت inline در خروجي HTML صفحه درج مي‌شود و به سرعت يك كد اسپاگتي گونه را پديد مي‌آورد كه نگهداري و ارتقاء آن كار راحتي نيست. اديتورهاي WYSIWYG مكرراً از اين كد طرف سرور ايراد مي‌گرفتند كه نمي‌فهمند چه مي‌گويد. شيء پرمصرف Session خواست كه امكان« نگهداري وضعيت» (state maintnenace) را دريك محيط ذاتاً بي‌ثبات (ststeless) فراهم كند، ولي اين راه‌حل در يك محيط وب كارساز نشدو هيچ زيرساختار خوبي براي cache كردن خروجي درطرف سرور وجود نداشت. مايكروسافت با آگاهي كامل از تمام اين مشكلات قدم پيش گذاشت و چارچوب ASP.NET را از ريشه خلق كرد تا با اين مشكلات مقابله كند. در نگاه اول به نظر مي‌رسد كه NET.ASP شباهت بسياري به Active Server Page قديمي دارد. مايكروسافت حتي ادعا مي‌كند خيلي راحت پسوند يك صفحه ASP را به ASPX تغيير دهيد و مطمئن باشيد كه آن صفحه در چاچوي جديد كار خواهد كرد، ولي هرچه بيشتر به عمق اين چارچوب وارد شويد، متوجه خواهيد شد كه اين مجموعه يك نسخة كاملاً بازبيني شده و تكميلي ASP است،

ASP.NET چيست؟

يك چارچوب كاملاً پيشرفته است كه براي توسعه برنامه‌هاي قابل بست وب بكار مي‌رود. سيستم مي‌تواند چنان تنظيم شود كه پاسخ به درخواستهايي را كه براي انواع مختلف فايل‌ها انجام مي‌شوند، به زيرسيستم‌هاي مختلف نصب شده در سرور محول كند. با ASP ، درخواست‌هايي كه براي فايل‌هاي با پسوند .asp صورت مي‌گيرند به ASP.DLL واگذار مي شوند. ASP.DLL نسخه اجرايي ASP است كه كد اسكريپت نعبيه شده در HTML درون فايل را Parse كرده و خروجي HTML آن صفحه را (به صورت پويا) توليد مي‌نمايد. چاچوب ASP.NET يك نسخه اجرايي كاملاً جديد ASP است كه درجهت توليد HTML تدارك ديده شده است. درخواست‌هايي كه باري دريافت فايل‌هايي با پسوند .aspx فرستاده مي‌شوند، به aspnet-isapi.dll واگذار مي‌شوند. اين يك كامپوننت مديريت شده است كه در پاسخ به درخواست ، به ايجاد نمونه‌هايي از كلاس‌هاي .NET مي‌پردازد.

چاچوب ASP.NET زيرمجموعه‌اي از كلاس‌هاي سيستم .NET است. دراين زيرمجموعه ، تعداد بي‌شماري كلاس گنجانده شده كه به صورت پويا براي شما HTMLهايي مي‌سازند، مانند انواع و اقسام فهرست، جدول، فرم و منطق اعتبارسنجي ، كلاس‌هايي براي مديريت state و caching و براي تنظيم و ايمن كردن برنامه و مجموعه‌اي از كلاس‌هايي براي حفظ سازگاري با ASP قديمي . با استفاده از ويژوال استوديو دات نت، اين زير ساختار در اختيار شما قرار مي‌گيرد و درنتيجه كدي كه شما مي‌سازيد مي‌تواند خيلي شبيه به Active Server Page ثديمي باشد، ولي هرچه بيشتر با سرويسها و زيرساختار آن آشنا مي‌شويد، كد شما بيشتر به يك فرم ويژوال بيسيك درمي‌آيد، با يك كد براي اجزاي UI و فايلي جداگانه براي كد شما.

ASP.NET و ASP چند تفاوت جال توجه دارند:

-صفحات ASP.NET « كامپايل » مي‌شوند نه « تفسير »: با دريافت اولين درخواست صفحه، يك فايل باينري اجرايي كامپايل مي‌شود. اين فايل ، در يك حافظه cache در وب سرور ذخيره مي‌شود، و درخواست‌هاي بعدي اين صفحه، از اين نسخة اجرايي براي پاسخ به درخواست استفاده مي‌كنند.

- از آنجا كه ديگر به مفسر اسكريپت احتياجي نيست، به ويژگي‌هاي زباني ويژوال بيسيك دات نت دسترسي كامل داريم. در واقع به جاي نوشتن اسكريپتي كه HTML توليد كند، برنامه‌اي مي‌نويسيم كه با اشياء كار مي‌كند.

- چاچوب به روشني كد را از محتوا تفكيك كرده ، با ASP، چون HTML درهنگام interpret شدن صفحه توليد مي‌شود، منطق صفحه شما بايد در داخل صفحه درجايي گنجانده شود كه مي‌خواهيد HTML ساخته شده با آن منطبث در آنجا ظاهر شود، ولي با ASP.NET تا وقتي تمام كد صفحه شما به پايان اجرا نرسيده باشد، هيچ HTML توليد نمي‌شود. كل فرآيند توليد HTML در مرحلة پرداخت صفحه صورت مي‌گيرد، كه از خصوصيات اشيائي استفاده مي‌كند كه براي توليد HTML ايجاد كرده‌اند.

- چارچوب ASP.NET وضعيت را براي شما حفظ مي‌كند: با چارچوب ASP.NET « حفظ وضعيت » اعتبارسنجي روي داده‌هايي كه دريك فرم HTML وارد شده‌اند، خود به خود براي شما انجام خواهد شد و بدون اينكه لازم باشد يك خط كد بنويسيد، فيلدها مقادير خود را حفظ مي‌كنند. اين امكان، نه تنها براي فيلدهاي متني ساده فراهم شده، بلكه روي ليست‌هاي SELECT، چك باكس‌ها، دكمه‌هاي راديويي و هرنوع ورودي ديگر نيز عمل مي‌كند.

- ASP.NET مي‌تواند وقايع را در سرور اجرا كند: در ASP به علت انبوه اسكريپتي كه بايد با HTML تركيب شوند، رايج شده كه يك ناحيه عملياتي را بين چند صفحه تقسيم كنند. با ASP.NET ، اما مي‌توانيم event trap يا « واقعه نگارهايي » را در طرف سرور بنا كنيم. اين چيزي شبيه به Remote Scripting است، ولي با HTML استاندارد كار مي‌كند، بدين ترتيب ، مي‌توانيم يك دكمه HTML را روي صفحه بگذاريم و كاري كنيم واقعه Click آن درطرف سرويس تحريك شود.

- مدل وقايع در ASP.NET بسيار منسجم است: با ASP، اسكريپت به يك حالت «بالا به پايين» در صفحه اجرا مي‌شود، ولي در ASP.NET ، يك مدل رويدادي وجود دارد و مهمتر اين‌كه، واقعه‌اي وجود دارد كه وقتي صفحه شروع به بارشدن مي‌كند، تحريك مي‌شود. اين خيلي شبيه به واقعه From Load در VB است.

چارچوب كاري .NET چيست؟

چارچوب كاري .NET آخرين ارائة مايكروسافت در دنياي چند محيطي (برنامه‌نويسي برنامه‌هاي روميزي و تحت وب) قابل عملكرد در چند سيستم مي‌باشد.

.NET علاوه بر تغييرات قبلي بيشتر، تلاش كرده است تا به ميزان ممكن ساده شود. .NET شامل عملكردي است كه هر برنامه‌نويس مي‌تواند بر سادگي به آن دسترسي داشته باشد. اين عملكرد يكسان در محدوده‌هاي انواع داده استاندارد شده قراردادهاي نامگذاري عمل مي‌كند. اين عملكرد داخلي ايجاد داده‌هاي خاص در يك فايل اسمبلي را نيز دربر مي‌گيرد كه براي عملكرد چندمحيطي امنيت تعبيه شده .NET و مديريت منبع خودكار .NET حياتي است.

بخش ديگر فلسفة حفظ سادگي اين است كه برنامه‌هاي .NET به نصب‌هاي كپي تنها گره خورده‌اند، به عبارت ديگر نياز براي يك بسته نصب خاص براي برنامه شما ، ديگر يك شرط نيست. اكثر برنامه‌هاي .NET در صورتي كار مي‌كنند كه آنها را در يك دايركتوري كنيم. اين ويژگي بار برنامه‌نويسي را كم مي‌كند.

CLR روش برنامه‌ها را تغيير داده است، زيرا برنامه‌نويسان VB محدود به محيط ويندوز نمي‌شوند. برنامه‌نوسان VB همانند ISO C/C++ هم اكنون قادر به مشاهده اجراي برنامه‌هاي خود در هر محيطي هستند كه زمان اجراي .NET روي آن نصب شده است. بنابراين، اگر شما نماينده يك برنامه‌نويس C براي پيش‌بيني آيندة برنامه‌نويسي در برنامه‌هاي VB.NET خود هستيد، سختي يادگيري طبيعي براي اين برنامه‌نويس، بطور روپايي با قابليت‌هاي چندزباني .NET كاهش مي‌يابد.

معرفي زمان اجراي زبان مشترك
CLR اجراي كد .NET را كنترل مي‌كند. CLR گامي بالاتر از COM ، MTS و COM+ است و درطي زمان به عنوان لايه زمان اجراي ويژوال بيسيك جايگزين آنها مي‌شود.

براي برنامه‌نويسان، مفهوم اين مسئله اين است كه كد VB.NET ما هر متد از زبان‌هاي ديگر اجرا خواهد شد، درحالي كه اندازة فايل را يكسان و كوچك نگه مي‌دارد.

CLR محيط زمان اجرايي براي .NET است و اجراي كد را همانند سرويس‌هاي فراهم شده .NET مديريت مي‌كند. CLR مي‌داند كه از طريق داده‌هاي خاصي كه در برنانه‌ها وجود دارند، چه كاري انجام دهد. داده‌هاي خاص در برنامه‌ها ، نقشه مكاني را براي يافتن كلاس‌ها، زمان بار كردن كلاس‌ها و زمان تنظيم كران‌هاي زمينه زمان اجرا توليد كد طبيعي، تقويت امنيت، تعيين كلاس‌هاي مورد استفاده متدها ، و باركردن كلاس‌ها در صورت نياز ذخيره مي‌كنند. باتوجه به اينكه CLR از اين اطلاعات آگاه است، مي‌تواند زمان استفاده يك شيء و زمان آزادسازي آن را تعيين مي‌كند. اين مسئله تحت عنوان كد مديريت شده شناخته مي‌شود.

حذف تمام اين زائده‌ها نيز باعث فراهم شدن يك مدل برنامه‌نويسي مستحكم‌تر مي‌گردد. باتوجه به اين كه CLR تمام عملكردهايي را به همراه دارد كه دركد مديريت نشده داشتيم، نبايد متكي به فايل‌هاي DLL از قبل موجود باشيم كه روي درايو سخت قرار دارند. اين مسأله بدان معني نيست كه آخرين DLLها را ديده‌ايم، بلكه به اين معني است كه چارچوب كاري .NET حاوي سيستمي درخود است كه مي‌تواند محل منابع مورد استفاده ما را نگاشت كند. ما ديگر وابسته به فايل‌هاي زمان اجراي VB نيستيم كه نصب شده‌اند يا وابسته به اجزاء ازقبل موجود خاص نيستيم.

به دليل اين كه مكمل CLR نيز كد مكمل مشخصه زبان مشترك (CLS) است، به كد مبتني بر CLR اجازه اجراي صحيح را مي‌دهد. CLS زيرمجموعه‌اي از انواع CLR تعريف شده در سيستم نوع مشترك (CTS) مي‌باشد. ويژگي‌هاي CLS در فرآيند چند محيطي مفيد است، زيرا آن‌ها حاوي انواع اصلي موردنياز براي قابليت CLR مي‌باشند. اين ويژگي‌هاي مركب به .NET اجازه مي‌دهند تا چندين زبان برنامه‌نويسي را مديريت كند. CLR نگاشت را مديريت مي‌كند، تمام آنچه كه شما نياز داريد كامپايلري است كه بتواند كد و داده خاص موردنياز برنامه را براي عمل كردن CLR توليد كند. اين امر اطمينان مي‌دهد كه هر وابستگي به برنامه شما ممكن است هميشه برآورده شود.

هنگامي كه كامپايلر خود را براي توليد كد .NET تنظيم مي‌كنيد، از طريق CTS اجرا مي‌شود و داده مناسب را در برنامه براي خواندن توسط CLR اضافه مي‌كند، هنگامي كه CLR داده را مي‌يابد، از طريق آن به اجرا ادامه مي‌دهد و هر چيزي را كه نياز دارد، در حافه مي‌چيند، هر شيئي را در هنگام فراخواني مي‌كند (و نه

از قبل). هر محاوره با برنامه، از قبيل ارسال مقادير كلاس‌ها، نيز در داده‌هاي خاص نگاشت مي‌شود و توسط CLR مديريت مي‌شود.

استفاده از زبان‌هاي برنامه‌نويسي مكمل .NET

.NET يك زبان برنامه‌نويسي تنها نيست بلكه مزيت يك سيستم چند محيطي را دارا مي‌باشد. هر زمان اجرا قابليت حمل را ميسر مي‌سازد، اما نيازمند اين مطلب است كه شما از يك مدل برنامه‌نويسي تكي استفاده كنيد. اگر به اين صورت عمل كنيد، اتكاي شما به آن زبان درهنگام عدم برآورده ساختن نيازها براي يك وظيفه خاص توسط آن زبان پاسخگو خواهد بود. ناگهان، قابليت حمل به صورت برعكس عمل مي‌كند. .NET اين مشكل را با مسير ساختن زبان برنامه‌نويسي مكمل .NET براي اجرا حل كرده است. درهنگام كار با VB دچار مشكل نخواهيد شد، اما مي‌دانيد كه مي‌توانيد آن را در C هم كار كنيد؟ از C# براي ايجاد كلاسي استفاده كنيد كه مي‌تواند به سادگي در برنامه Vb شما استفاده شود. كاربران زبان‌هاي برنامه‌نويسي شخص ثالث، نبايد نگران باشند، زيرا شركت‌هاي مختلفي نگارش‌هاي مكمل .NET زبان‌هاي خود را ايجاد مي‌كنند.

درحال حاضر، تنها زبان‌هاي مكمل .NET ، تمام زبان‌هاي مايكروسافت است. براي اطلاعات بيشتر آدرس http://msdn.microsoft.com/net را براي موارد زير بررسي كنيد:

C# 3

C++ 3 با توسعه‌هاي مديريت شده

VB.NET 3

ASP.NET 3 (هرچند اين يكي بيشتر زيرمجموعه‌اي از VB.NET است)

Jscript.NET 3

اين پيشرفت‌ها باعث بهبود و توانايي شما براي كار با چندين زبان خواهد شد. مثلاً، يك بهبود كوبول كه ممكن است در VB.NET وجود نداشته باشد، بدين معني نيست كه برنامه‌نويس VB.NET نمي‌تواند آن مزيت را داشته باشد. مي‌توانيد به آساني روشي را براي استفاده از راه حل كوبول به عنوان مثال بيابيد يا كد را به VB.NET تبديل كنيد.

ايجاد اسمبلي‌ها
هنگامي كه چندين زبان داريد، چگونه با يكديگر جهت اجرا كار مي‌كنند؟ بيشتر زبان‌هاي برنامه‌نويسي ديگر از فرمت اجرايي قابل حمل (PE) براي فايل‌هاي اجرايي خود استفاده نمي‌كنند. درمحيط .NET مورد جديدي وجود دارد: يك روش منطقي براي مالك كلكسيوني از فايل‌هاست كه به عنوان اسمبلي‌هاي ايستا در نظر گرفته مي‌شوند كه CLR از آن‌ها استفاده مي‌كند. اسمبلي‌هاي ايستا مي‌توانند منابع مورد استفاده اسمبلي باشند. ازقبيل فايل‌هاي تصويري يا فايل‌هاي متني كه برنامه از آن‌ها استفاده خواهد كرد. كد واقعي كه اجرا مي‌شود، در اسمبلي با فرمت زبان واسط مايكروسافت (MSIL) مي‌باشد. به عبارت ديگر، هر اسمبلي معادل يك جزء VB 6.0 COM است. هر اسمبلي سه گزينه دارد كه بايستي درهنگام ايجاد آن تنظيم شوند:

بهينه‌سازي بارگننده 3

نام‌گذاري 3

محل 3

بهينه‌سازي بارکننده داراي سه تنظيم مي‌باشد، تك حوزه‌اي، چند حوزه‌اي و ميزبان چند حوزه‌اي. تنظيم تك حوزه‌اي پيش‌فرض است و در بيشتر موقعيت‌هاي سمت کلاينت استفاده مي‌شود. کد JIT معمولاً درهنگام استفاده از تنظيم تك حوزه‌اي کوچکتر است (درمقايسه با دوتنظيم ديگر) و هيچ تفاوت قابل توجه‌اي بين منابع حافظه وجود ندارد. استثناء هنگامي است که برنامه خاتمه به عنوان بخشي ازيك تنظيم چند حوزه‌اي يا ميزبان چند حوزه‌اي به کار مي‌رود، که واقعاً مفيدتر است تا سودمندتر (ازقبيل يك راه حل کلاينت/سرور).

تنظيمات چند حوزه‌اي و ميزبان چند حوزه‌اي براي مفهوم يکساني از کاربرد چند حوزه‌اي به کار مي‌رود تنها تفاوت بين اين دو چگونگي عکس‌العمل CLR در کد است، در چند حوزه‌اي که درکل حوزه يکسان فرض مي‌شود. در ميزبان چند حوزه‌اي هر ميزبان حوزه کد متفاوتي دارد. فرض کنيد که يك برنامه‌نويسي داريد که تمام حوزه‌ها داراي نام فايل اسمبلي هستند، اما هريك کد ميزبان متفاوتي دارند که نشان مي‌دهد هريك چگونه محاوره مي‌کنند. شما بهترين کارآيي را با استفاده از روال بهينه‌سازي ميزبان چند حوزه‌اي خواهيد داشت.

بيشتر منابع را با تنظيم اسمبلي جهت استفادة برنامه‌ها خواهيد داشت. منابع کمتري مصرف خواهد شد زيرا نوع (شيء) بار شده و نگاشت مي‌شود، بنابراين نوع نبايد در هربار مجدداً ايجاد شود. هرچند، نتيجه نهايي کد JIT مقداري افزايش مي‌يابد، و دستيابي به آيتم‌هاي ايستا کندتر مي‌گردد، زيرا مرجع ايستا به طور غيرمستقيم ارجاع مي‌گردند.

نام اسمبلي مي‌تواند توسط چند برنامه بر محدوده و کاربرد تأثير گذارد. برنامه‌اي که از سمت کلاينت استفاده مي‌شود، نام داده شده را درهنگام ايجاد به کار مي‌برد، اما به هيچ وجه از تصادم نام جلوگيري نمي‌شود. بنابراين به منظور کمك به جلوگيري از برخوردهاي نام که يك اسمبلي در يك سناريوي چند اسمبلي مي‌تواند نامي مشترك به اسمبلي بدهيد. داشتن نام مشترك بدين معني است که اسمبلي مي‌تواند در کش اسمبلي سراسري گسترش يابد که دراين صورت به عنوان يك مخزن سراسري از اسمبلي‌ها تصور مي‌شود.

نام مشترك از نام متني اسمبلي (نامي که براي آن ايجاد مي‌کنيد) و يك امضاي ديجيتال تشکيل مي‌شود. نام‌هاي مشترك نام‌هاي منحصر به فردي هستند که وابسته به زوج نام متني و امضاي ديجيتال مي‌باشند. اين سيستم ، به نوبت باعث جلوگيري از برخورد نام مي‌شود و به هر فردي اجازه مي‌دهد تا از نام متني يکسان براي نوشتن در فايل خود استفاده کند، زيرا نام مشترك متفاوت است. هر نام مشترك اطلاعات موردنيازي را فراهم مي‌کند که براي پشتيباني نگارش توسط CLR ضروري هستند. اين اطلاعات مشابه براي تأمين بررسي‌هاي جامعيت به کار مي‌رود.

نام مشترك درابتدا در اسمبلي اصلي ايجاد مي‌شود، سپس مرجعي به نام اسمبلي اصلي بعنوان نشانه‌اي از نگارش در فوق داده اسمبلي ارجاع شده ذخيره مي‌شود و بالاخره از طريق CLR اصلاح مي‌گردد. هر اسمبلي درهنگام ايجاد داراي مشخصه‌هاي زير مي‌باشند.

3 حاوي کدي است که زمان اجرا آن را اجرا مي‌کند که: PE MSIL بدون وجود در فهرست اقلام اجرا نمي‌گردد. به عبارت ديگر، اگر فايل بطور صحيح فرمت نشده باشد، اجرا نخواهد شد.

3 تنها يك نقطه ورودي: هر اسمبلي نمي‌تواند بيش از يك نقطه شروع براي ايجاد اجرا توسط زمان اجرا داشته باشد. مثلاً نمي‌توانيد از WinMain و Main باهم استفاده کنيد.

3 واحدي از اجراي پهلو به پهلو: هر اسمبلي واحد اصلي موردنياز براي اجراي پهلو به پهلو را فراهم مي‌کند.

3 محدوده نوع: هر نوع که دريك اسمبلي معرفي مي‌شود، به عنوان نوع اصلي تشخيص داده مي‌شود و نه به عنوان يك نوع منحصربه‌فرد که در حافظه مقداردهي شده است.

3 محدودة امنيتي اسمبلي درخواستهاي مجوز را ارزيابي مي‌کند.

3 واحد توسعة اصلي: هر برنامه از اسمبلي‌هايي تشکيل مي‌شود که تنها به اسمبلي‌هايي نياز دارد که از توابع هسته‌اي آن تشکيل شده‌اند. هر اسمبلي ديگري که موردنياز باشد، مي‌تواند طبق تقاضا فراهم گردد که باعث مي‌شود تا برنامه‌ها چندين فايل راه‌انداز مرتبط با فايل‌هاي زمان اجراي VB 6.0 نداشته باشند.

3محدوده برد مرجع: اقلام اصلي در اسمبلي نشان مي‌دهند كه به منظور تعيين انواع و منابع چه اتفاقي افتاده و يا نيفتاده است. همچنين وابستگي اسمبلي را محاسبه مي‌كنند.

3 محدوده نگارش: كوچكترين واحد قابل نگارش در CLR مي‌شود ، تمام انواع و منابعي كه داراي نگارش هستند، به عنوان يك واحد درنظر گرفته مي‌شوند. اقلام اصلي هر وابستگي نگارشي را بيان مي‌كنند.

آشنايي با فوق داده
هنگام ايجاد يك اسمبلي دو چيز اتفاق مي‌افتد. كد شما به MSIL تبديل مي‌شود، و تمامي اطلاعات وابسته به آن كه در كد وجود دارد (references , types,...) به عنوان فوق داده در داخل مانيفست آورده مي‌شوند. به دنبال آن، CLR فوق داده را ضميمة داده‌هاي موجود در حافظه كرده و از آن به عنوان يك مرجع در يافتن نيازهاي برنامه استفاده مي‌كند. اين نقشه سهم عظيمي در قابليت عملكرد چند محيطي دارد، زيرا CLR درحقيقت نيازي به آن ندارد كه از محتويات كدي برنامه اطلاع داشته باشد و كافي است كه نگاهي به فوق داده بياندازد و از موضوع موردنياز و مقصد نهايي آن اطلاع پيدا كند. فوق داده مسئول انتقال اطلاعات زير به CLR است:

3 مجوزهاي امنيتي

3 انواع صادر شده

3 هويت

3 مراجع اسمبلي خارجي

3 قابليت ديد رابط

3 عناصر اسمبلي محلي

فواید فوق داده
اقلام موجود در فوق داده درهنگام اجرا، توسط CLR در ساختارهاي اطلاعاتي موجود در حافظه جا داده مي‌شوند. اين امر موجب مي‌شود كه فوق داده در اسرع وقت با آزادي بيشتري مورد استفاده قرار گيرد. اين سيستم با عرضه كردن تمامي اقلام موردنياز اسمبلي، توابع خود توصيف‌گر اسمبلي‌هاي .NET را بهبود بخشد. همچنين به اشياء ديگر اجازه مي‌دهد (البته براساس فوق داده) تا با اسمبلي تعامل و ارتباط داشته باشند. همچنين فوق داده با ايجاد لايه‌اي بين كد اسمبلي و آنچه در CLR مي‌بينيد، قابليت عملكرد چندمحيطي را امكان‌پذير مي‌سازد. CLR از فوق داده به طور وسيع استفاده مي‌كند، بنابراين بار قابليت عملكرد را از دوش /CPU زبان برمي‌دارد. CLR از طريق مجموعه‌اي از APIها، بويژه سرويسهاي پخش بازتاب و بازتاب مديريت شده، فوق داده را مي‌خواند و ذخيره مي‌كند و از آنها استفاده مي‌كند. انتزاع لايه‌اي موجب مي‌شود كه در زمان اجرا به بهينه‌سازي اقلام مانيفست موجود در حافظه بدون مراجعه به كامپايلرهاي اوليه ادامه داده و يك ماندگاري از نوع snap-in را قادر سازد تا به CLR اجازه نمايش دودويي بدهد و در عين حال با انواع مديريت نشده و هر نوع فرمت ديگري ارتباط برقرار كند كه نياز به استقرار در حافظه داشته باشد.

استفاده از ارجاعات ضعيف
ابتكار ديگري كه از مفهوم ريشه‌ها نشأت مي‌گير ارجاعات ضعيف است. ارجاع ضعيف عبارت است از پيوندي ضعيف به يك شيء موجود در حافظه كه در فرآيند finalization بوده است يا مي‌باشد و مانند يك ريشه توسط جمع كن حافظه هرز در نوبت بعدي اجرا جمع‌آوري خواهد شد. يك ارجاع مستحكم و قوي، توليد شيء اوليه را به تصوير مي‌كشد. بدون يك ارجاع قوي، نمي‌توانيد يك ارجاع ضعيف ايجاد كنيد. در زمانيكه شما با اشياء تشديد كنندة حافظه سروكار داريد، از هزينه اشيائي كه به صورت پيوسته مجدداً توليد و تعيين وضعيت اوليه مي‌شوند، پرهيز مي‌كنيد. ارجاعات ضعيف مي‌توانند يك ابزار چاره‌ساز باشند. درنظر بگيريد يك شيء از يك بانك اطلاعاتي عبور مي‌كند و مجموعه‌اي از فيلترهاي مرتب شده را ذخيره مي‌سازد. اگر بانك اطلاعاتي به حد كافي كوچك باشد، مي‌تواند بدون هيچ مشكلي در حافظه بماند. اما اگر بانك اطلاعاتي حجيم باشد، هرگاه نياز به توليد يك منبع جديد داشته باشيم با خطر overload شدن

منابع مواجه هستيم. به كمك يك ارجاع ضعيف، مي‌توانيم از توليد يك شيء جديد صرفنظر كنيم (آن را توليد نمي‌كنيم) و با حفظ اقلامي كه در حالت standby به آنها نياز داريم از مرتب‌سازي مجدد بي‌نياز شويم. به دنبال آن شما مي‌توانيد با اشاره به ارجاع ضعيف، ارجاع جديد را مجدداً توليد كنيد.

خدمات امنيتي (Security Service)

خدمات امنيتي نبايد با مفاهيم امنيتي مخصوص .NET يكي دانسته شود. خدمات امنيتي نوعي بررسي و هماهنگي با كد، فوق داده‌ها و MSIL است و تضمين مي‌كند كه CLR به آنچه كه انتظار داشت، رسيده است، چه از طريق همان برنامه‌نويس يا يك منبع مورد اعتماد، و دستيابي به ارجاعات بعدي نيز غيرممكن مي‌سازد كه بسته به ايزوله‌سازي است كه دستيابي تضمين شده باشد.
در .NET سيستم اجراي مجازي (VES) تمام بازرسي‌هاي امنيتي را اداره مي‌كند. امنيت نوع، از طريق مطابقت دادن انواع مستحكم موجود در فوق داده‌ها با MSIL متناظر (متغيرهاي محلي و شيارهاي پشته) به واسطه VES پياده مي‌شود. شما مي‌توانيد به صورت يك نمودار فني به آن نگاه كنيد، خطي خيلي واضح و پر كشيده شده است كه از فوق داده‌ها به MSIL اشاره مي‌كند و اطمينان پيدا مي‌كند كه همه چيز با تعاريف صحيح و فضاي حافظه مطابقت دارد.

VES همچنين ايمني نسخه‌اي را پوشش مي‌دهد، چون VES همه چيز را مشخص مي‌كند، در ادامه كار بازرسي خود درستي همه اطلاعات را تحقيق مي‌كند، ازجمله آزمايش نسخه را. VES همچنين از اين موضوع اطمينان حاصل مي‌كند كه CLR آنچه را خواهد ديد كه او بدست مي‌آورد. به عبارت ديگر CLR در محدودة فرضياتي كار خواهد كرد كه VES دربارة كد كرده است.

اما براي آنكه درخصوص كد فرضيه‌اي صورت بگيريد CLR بايد مطمئن باشد كه كد بخوبي اجرا مي‌شود. VES با فراهم كردن سه متد مداخله مي‌كند كه كد مي‌تواند از اين سه متد جهت اجرايي شدن استفاده كند. بارگذارندة كلاس، احضار محيطي بر كد قانوني و براي اهداف انتقال يك COM مديريت نشده با عملكرد چندمحيطي استفاده از دو مورد آخر ممكن است منجر به مشكلات كاركردي گردد، بنابراين بهتر است كه موقع نوشتن يا انتقال كد از آنها خودداري كنيد و فقط به بارگذارندة كلاس بچسبيد. بارگذارندة كلاس، مراحل پياده‌سازي اجرا را به اطلاعات مربوط به مراحل اجرا دريك فوق داده وصل مي‌كند. VES همچنين از بارگذارندة كلاس براي شناسايي شخصي استفاده مي‌كند كه سعي در دسترسي به يك نوع دارد ولذا از توانايي تعيين قابليت دسترسي آن بهره مي‌برد.

بعلاوه VES، از طريق CTS، به مجوزهاي دسترسي به كدها كه در فوق داده ذخيره مي‌شوند، دسترسي پيدا مي‌كند. VES هرنوع را مطابق با مجوزها بررسي مي‌كند و هر نوع مجوزدار را با يك stub در لودري علامت مي‌گذارد كه به CLR مي‌گويد تا مجوزها را به آنكه stub اشاره مي‌كند اعمال كند. به چنين امنيتي امنيت اعلاني (declrative) گفته مي‌شود.

امنيت چارچوب كاري (Framework)

امنيت دسترسي كدي و امنيت براساس نقش و وظيفه، دو نوع امنيت ارائه شده توسط خود چارچوب كاري .NET مي‌باشند. اين‌ها مكانيزم‌هايي هستند كه به منظور حفظ يك ذهنيت ساده مربوط به يك كاربرد براي تصميم‌گيري بنا نهاده شده‌اند. نظريه ساده‌انگاري براساس يكنواختي، و امكان‌گذاري از امنيت كدي به امنيت وظيفه‌اي و بالعكس مي‌باشد. اصولي كه به امنيت .NET ثبات و استحكام مي‌بخشد، عبارتند از: مجوز، اصول و خط مشي امنيتي.

امنيت دسترسي كدي، همانطوريكه احتمالاً اطلاع داشته باشيد، درجات متعدد اعتماد را براي يك برنامه فراهم مي‌سازند، اين درجات را براساس اطلاعاتي كه اسمبلي مي‌دهد مي‌تواند تغيير دهد، مانند برنامه‌نويس، نسخه (نگارش) و مانند آن، چون اين اطلاعات در كد ذخيره مي‌شود. هنگامي كه فرآيندي تعيين مي‌كند كه‌آيا مي‌توان به كد خاصي دسترسي داشت، زمان اجراي پشته فراخواني جاري كد را براي مجوز جستجو مي‌كند، هرچند اگر نتواند مجوز را پيدا كند. استثنائي را به وجود مي‌آورد.

امنيت برپايه نقش و وظيفه، تصميمات اختياري و دلخواه را براساس مقدار اصلي مي‌گيرد كه از رشته جاري درخواست كرده است. نقش‌هاي فهرست شده در مقدار اصلي در مرحلة بعد ارزيابي شده و عملكرد توانايي درخواست شده يا داده مي‌شود يا داده نمي‌شود.

برنامه‌نويسان نرم‌افزارهاي مالي و كدنويسان بانك‌اطلاعاتي ممكن است قبلاً با مفهوم امنيت براساس نقش و وظيفه آشنا باشند. معمولاً در اين موارد، هنگاميكه يك كلاينت خواهان دسترسي به قسمت معيني از سيستم يا منبع مي‌باشد، يم وارسي صورت مي‌گيرد كه درنتيجه آن مشخص مي‌شود كه درخواست كلاينت از كدام نقش نشأت گرفته است. فرض كنيم كه عضوي از گروه Alpha تلاش مي‌كند كه به يك منبع واقع در عضوي از گروه omega دسترسي داشته باشد. Alpha شروع به ارتباط مي‌كند و omega اولين اصل از رشته ارتباط را دريافت مي‌كند. در مرحله بعد، اصل ازنظر نقش آناليز مي‌شود و omega متقاعد مي‌شود كه گروه كاري alpha براي همة منابع مجوز ندارد (فقط براي دوتا از آنها مجوز دارد). Omega ارتباط را قبول مي‌كند اما درخواست alpha را به دو منبع محدود مي‌كند. اگر alpha سعي در دسترسي به منابعي جز اين دو مورد بود، درخواست او رد مي‌شد.

اهداي مجوزها

مجوز بدنه اصلي بلوك امنيت را تشكيل مي‌دهد. بعضي‌ها به مجوز به عنوان يك پاسخ منطقي نگاه مي‌كنند، كه به يك استعلام مبني بر كسب دسترسي داده مي‌شود، درحالي كه برخي ديگر اعتقاد دارند مجوز يك كليد مناسب براي يك قفل است. هر دونظريه درست است. مجوزها در .NET ازطريق درخواست‌ها، اهداءها و نيازها استفاده مي‌شود. كد مجوزها را به اين دليل درخواست مي‌كند تا ببيند آيا مي‌تواند به يك فايل دسترسي پيدا كند. اگر كد دراثر اين مجوزها خراب نشود، شما مي‌توانيد توسط يك تابع به كد مجوز اعطا كنيد. اگر با مجوزهاي آماده كنار بياييد ، ممكن است يك لايه اضافه شده از مجوز بنام تقاضا را پياده‌سازي كنيد، بعبارت ديگر هنگامي كه كد ممكن است مجوزهاي اوليه موردنياز براي تأمين خواسته را داشته باشد، مي‌تواند درخواست مجوز يا مجوزهاي ويژه‌اي را بكند. هردو امنيت دسترسي كدي و امنيت برپايه نقش، فهرستي از مجوزها را دارند.

حصول نمايش از طريق يك اصل

آيا تا كنون از واسطه‌اي درخواست كرده‌ايد تا براي كسب دسترسي از برنامه وارد عمل شود؟ اصل دقيقاً همين كار را مي‌كند. يك اصل بسته به شرايط حاكم، مجوز لازم را براساس درخواست شما جهت ورود بدست مي‌آورد. CLR به اصل اجازه مي‌دهد ولي به شما اجازه نمي‌دهد، زيرا CLR تنها آنچه را كه اصل قرار است انجام دهد، به شما اجازه انجام مي‌دهد.

يكي از اصل‌هاي معمولي ارائه اجراي بي‌هدف مي‌باشد كه براي تشخيص آنچه كه يك شخص شناسايي نشده مي‌تواند ببيند، قابل استفاده است. هرچند اين مسأله دريك برنامه روزمره عملي نيست. براي تست كردن و اشكال‌زدايي موقعيت‌ها خيلي مفيد است و براي تعيين تكليف در مواقعي كه مجوز نشاندهنده آن است كه شما برنامه‌ريزي نكرده‌ايد، بي‌نهايت سودمند است. اصل‌هاي سفارشي و سليقه‌اي توسط برنامه كاربردي در شرايط فوري ايجاد مي‌شوند تا يك نياز و فوريت جاري را رفع كنند. اين‌ها قابليت استفاده پايه‌اي يك اصل معمولي را گسترش مي‌دهند، اما وابسته به داشتن ماژول‌هاي شناسايي و انواع داده شده به آنها توسط برنامه مي‌باشند. اين وابستگي عنصري ، امنيتي، اصل سفارشي امنيتي مي‌دهد. چون نيازمند خواسته‌اي است كه براي كاركردن لازم دارد.

خط مشي امنيتي (Securty Policy)

مقرراتي را كه CLR از آنها تبعيت مي‌كند در مجموع خط‌مشي امنيتي گفته مي‌شود. مدير محلي اين قوانين قابل پيكربندي را تعيين مي‌كند. هنگاميكه يك اسمبلي سعي در باركردن دارد، خط‌مشي امنيتي اين موضوع را بررسي مي‌كند كه چه مجوزهايي را CLR مي‌تواند به اسمبلي بدهد. امكانات و حالت‌هاي مختلفي را مشخص مي‌كند، اگر اسمبلي از آنها به سلامت گذشت، مجوزهاي موردنياز را صادر مي‌كند و درغير اينصورت اجازة اجرا شدن برنامه را نمي‌دهد.

سه سطح خط‌مشي امنيتي را مشخص مي‌كنند: خط‌مشي ماشين محلي، خط‌مشي حوزه برنامه كاربردي و خط‌مشي كاربر.runtime از هر سه خط‌مشي براي فيلتر كردن خط‌مشي نهايي استفاده مي‌كند و سپس آن را در اسمبلي وضع و مجوز آن را مشخص مي‌كند. خط‌مش‌هاي كاربردي و حوزه برنامه هردو مجموعه مجوزهاي داده شده را تعيين مي‌كنند، و سپس مجوزهايي كه فيلتر نمي‌شوند خط‌مشي امنيتي مي‌شوند.

حوزه‌هاي برنامه‌هاي كاربردي
هر برنامه‌اي كه در .NET دريك حوزه‌اي اجرا مي‌شود كه توسط يك ميزبان مديريت مي‌شود. اين ميزبان ممكن است يك ميزبان هسته (.EXE ها را از يك هسته اجرا مي‌كند)، يك ميزبان مرورگر (كد را از يك پايگاه وب اجرا مي‌كند) ، يك ميزبان سروري (ASP.NET؛ كدي را اجرا مي‌كند كه درخواست‌هاي سروري را كنترل مي‌كند) و يك ميزبان سفارشي باشد. وقتي كه يكي از اينها حوزه برنامه را مي‌سازد، مثلاً ميزبان هسته، (كه ويندوز خواهد بود) يك خطمشي را تعريف مي‌كند(sets) كه كد بايد تحت آن حوزه با آن درگير باشد. خطمشي توليد شده قابل ازدياد نيست ولي مي‌تواند توسط ميزبان منعطف‌تر و سازگارتر گردد.

پس از تنظيم و تعريف يك خطمشي حوزه برنامه‌اي، خطمشي جديد تنها به اسمبلي‌هايي اعمال مي‌شود كه بعد از آن بار مي‌شوند. اسمبلي‌هايي كه خطمشي قبلي دارند نمي‌توانند از خطمشي جديد استفاده كنند و برهمان خطمشي قبلي خواهند بود مگر آنكه دوباره بار شوند، به محض آنكه اسمبلي اصلي بار شد و اولين ارجاع به اسمبلي ديگر به عمل آمد نوبت به باركننده مي‌رسد و اسمبلي را در حوزه برنامه مناسب قرار مي‌دهد و سپس اطلاعات (به عنوان مدرك) را كه قابل اعتماد بودن آنرا ثابت مي‌كند به runtime باز مي‌گرداند (اطلاعات نسخه‌اي را جهت تحقيق برگشت مي‌دهد).