دانشگاه آزاد اسلامی

واحد کرج

تکنولوژی .NET

پروژه دوره کاردانی پیوسته

رشته کاربرد کامپیوتر

استاد راهنما:

جناب آقای مهندس سید مسگری

توسط دانشجو:

فرهاد حیدری

شماره دانشجویی:۷۸۹۱۳۵۳۴۵۵۵

نیمسال اول سال تحصیلی ۸۲-۸۱

فهرست مطالب

فهرست منابع

۱- **کتاب** .NET Frsamework , C# **مترجم: مهنـــدس حوریـــه شـــاه حسینی**

۲- **سایت اینترنتی** WWW.microsoft.com

۳- **سایت اینترنتی** WWW.devn.com

۴- **سایت اینترنتی** WWW.SearchwebSerdices.Com

پیشگفتار

ما بسوی تغییر دیگری در سکو (Platfrom) پیش می رویم. درست همانطور که Dos به Windows تحول پیدا کرد، تکنولوژی .NET نیز تحول دیگراست که کاربران کامپیوتر لزوم یادگیری آنرا بطور آشکار احساس می کنند.

دات نت هم یک استراتژی حرفه ای و تجاری برای شرکت مایکروسافت است و هم مجموعه ای از برنامه های قابل اجرا و پشتیبانی است،برای هر آنچه که بعنوان خدمات وب (Web Services) شناخته می شوند.

دات نت تلاش می‌کند تا در نهایت یک محدوده اجرایی برای تمام زبانها ارائه دهد. تا عمل متقابل (interoperate) به هم پیوسته ای با یک سیستم نوع متداول و کتابخانه کلاس پایه داشته باشند به همراه این Platfrom جدید، مایکروسافت یک زبان برنامه نویسی برای بهره گیری از تکنولوژی .NET طراحی و پیاده سازی نموده که C# نام دارد برخی از مطالب و مثالهایی که در این مجموعه تحقیقی ارائه شده در مورد این زبان است.

بر خلافت زبان Visual Basic، مایکروسافت هر دو زبان C# و CLS را به سازمان استانداردهای ECMA ارائه داده است تا از پذیرش Platfrom و زبان برنامه نویسی C# مطمئن شود. با استفاده از استانداردهای عمومی بعنوان زیر بنای .NET این شرکت اطمینان دارد که قدرت و میزان پذیرش این زبان باعث می‌شود که فراتر از زبانها و سیستمهای صرفاً تجاری گام بردارد.

این مجموعه تحقیقی که بعنوان پایان نامه دانشجویی ارائه شده است. مبتنی برهفت بخش است که شش بخش از آن شامل اطلاعات نسبتاً اجمالی در مورد .NET **و به زبان فارسی می باشد که عبارتند از:**

- .NET Platform
- **زبان میانه** (IL)
- **زبان عمومی** (CLR)
- **توسعه فرمهایی ویندوز با** Visual Studio.NET
- **اساس** ASP.NET
- **خدمات وب** (Web Dervices)

و بخش آخر این مجموعه نیز شامل اطلاعات تفضیلی در مورد .NET **و به زبان انگلیسی می باشد. در خاتمه لازم می دانم از کمکها و راهنمایی های استاد راهنمای اینجانب، جناب آقای مهندس سید مسگری تشکر و سپاسگذاری نمایم.**

فرهاد حیدری

زمستان ۳۸۱

Π

نمای کلاسی (Class View)

نمای منبع (Resourse Viwe)

جستجوگرماکرو (Macro Xeplorer)

–راهنمای دستی

–بخش C: جعبه ابزار و Server Expiorer

بخشD: تکالیف (Taske)**، خروجی** (Output)**، نتایج جستجو و مشاهدات**

بخش E: خصوصیات (Properties) **راهنمای پویا** (Dynamic Help) **و برگزیدگان** (Favorites)

–برنامه‌های اشکال زدایی

–الفبای تصویری نقطه توقف

–الفبای تصویری نقطه پیشرفته

–نقطه‌های توقف شرطی

–شمارش دفعات (Hit Counts)

–هنگام مکث رمز چه باید کرد؟

–ضمیمه شدن اشکال زدا به یک فرآیند

–اشکال زدایی JIT

بخش چهارم : برنامه کاربردی نمونه فرمهای ویندوز (Scribble.NET)

–تعریف

––منابع در .NET

–پیج و مهره‌های محلی سازی

–کلاسهای مدیریت منابع .NET

بخش هفتم: اطلاعات تصیلی در مورد تکنولوژی .NET به زبـان انگلیسی

فهرست منابع

پیشگفتار

ما بسوی تغییر دیگری در سکو (Platfrom) پیش می رویم. درست همانطور کـه Dos به Windows تحول پیدا کرد، تکنولوژی .NET نیز تحول دیگراسـت کـه کاربران کامپیوتر لزوم یادگیری آنرا بطور آشکار احساس می کنند.

دات نت هم یک استراتژی حرفه ای و تجاری برای شرکت مایکروسافت اسـت و هم مجموعه ای از برنامه های قابل اجرا و پشتیبانی است،برای هر آنچه که بعنـوان خدمات وب (Web Services) شناخته می شوند.

دات نت تلاش می‌کند تا در نهایت یک محدوده اجرایـی بـرای تمـام زبانهـا ارائـه دهد. تا عمل متقابل (interoperate) به هم پیوسته ای با یک سیستم نوع متداول و کتابخانه کلاس پایه داشته باشند به همراه این Platform جدید، مایکروسـافت یک زبان برنامه نویسی بـرای بهـره گیـری از تکنولـوژی .NET. طراحـی و پیـاده

سازی نموده که C# نام دارد برخی از مطالب و مثالهایی که در این مجموعه تحقیقی ارائه شده در مورد این زبان است.

بر خلافت زبان Visual Basic، مایکروسافت هر دو زبان C# و CLS را به سازمان استانداردهای ECMA ارائه داده است تا از پذیرش Platfrom و زبان برنامه نویسی C# مطمئن شود. با استفاده از استانداردهای عمومی بعنوان زیر بنای .NET. این شرکت اطمینان دارد که قدرت و میزان پذیرش این زبان باعث می‌شود که فراتر از زبانها و سیستمهای صرفاً تجاری گام بردارد.

این مجموعه تحقیقی که بعنوان پایان نامه دانشجویی ارائه شده است. مبتنی برهفت بخش است که شش بخش از آن شامل اطلاعات نسبتاً اجمالی در مورد .NET. و به زبان فارسی می باشد که عبارتند از:

-.NET Platform

-زبان میانه (IL)

-زبان عمومی (CLR)

-توسعه فرمهایی ویندوز با Visual Studio.NET

-اساس ASP.NET

-خدمات وب (Web Services)

و بخش آخر این مجموعه نیز شامل اطلاعات تفصیلی در مورد NET. و به زبان انگلیسی می باشد. در خاتمه لازم می دانم از کمکها و راهنمایی های استاد راهنمای اینجانب، جناب آقای مهندس سید مسگری تشکر و سپاسگذاری نمایم.

فرهاد حیدری

زمستان ۱۳۸۱

مقدمه ای درباره **.NET**

یقیناً، میکروسافت برجسته ترین تحولات را در صنعت رایانه به وجو آورده است. موفقیت های DOS ، حاصل تلاش بی وقفه بیل گیتس و استیوبالمر بود، وقتی آنها به IBM اطلاع دادند که سیستم عاملی برای فروش دارند حرکت قابل توجهی در دنیای رایانه ایجاد شد. میکروسافت با الهام گرفتن از جذابیت اپل مکینتاش سیستم عاملی ایجاد کرد که به طور گسترده در جهان مورد استفاده قرار گرفت و تحولی دوباره در دنیای رایانه بوجود آورد.

اکنون، میکروسافت با ابداع مجدد روشی برای استفاده و برنامه ریزی رایانه ها تحول دیگری در جهان ایجاد کرده است. اگر شما یک برنامه نویس C++ هستید یا اگر با سیستم عامل ویندوز یا کلاس (رده) خیت بنیادین میکروسافت (MFC) کار می کنید، بدون شک کاملاً تحت تأثیر الزامات NET، خواهید بود.

اخیراً اینترنت به رسانه ای تبدیل شده است که ما از طریق آن میتوانیم تجارت کنیم، دوستان خود را ملاقات کنیم، حساب جاری خود را انجام دهیم، بازی کنیم، گپ بزنیم و با اشخاصی که دوست داریم در تماس باشیم. انجام همه کارها از طریق یک نرم افزار نوشته شده، عملی و امکان پذیر شده اند تا ارتباطات ضابطه اینترنت (IP) برای وب جهانی برقرار می شوند. از آن جا که وب پیشرفت نموده

و پیچیده تر شده است نرم افزار لازم است برای اجرای آن نیز باید میلیون ها داده IP را به رایانه ما انتقال دهد تا از آن دریافت کند. با استفاده از پست الکترونیکی مرور صفحات وب تعامل (فعل و انفعال) با پایگاه دادهع ها و اجرای برنامه های کاربردی توزیع شده بسیار پیچیده تر شده است و در نتیجه مهارت ها و تکنیک های برنامه نویسی نیز به ایجاد و حفظ نرم افزاری نیاز دارد که پیچیده تر باشد. یک برنامه نویس، اغلب مواقع (و به طور روزانه) با XML , SOAP , ASP, COM CoM^{+} . DCOM، و XSL و یک آرایه رو به افزایش از SDK های کامل و نیز راهنماییهایی در رابطه با آنها مواجه می گردد.

میکروسافت، بنابر اصل نیازهای فناوری جاری همانند بسیاری از صنعتکاران به طور روزافزون، برنامه های کاربردی، مجموعه ابزارهای و سیستم های عامل و کتابخانه ها قابل حمل نیستند، زیرا بسیار گسترده تر می باشند و همچنین حاوی افزودنی ها، الحاقات (extensions) و توافق ها (compromisers) هستند. چارچوب NET، در اصل آن را تغییر می دهد. این تغییر نه نمونه ی است و نه برای تحکیم کار، بلکه این یک حرکت بسیار بزرگ، برجسته و قابل تأمل است که همه مطالب مربوط به برنامه نویسی رایانه های شخصی شما (در هر شکل و اندازه ای) را دوباره تعریف می کند و شامل خود زبان ها نیز می باشد.

یکی از مهمترین حقایق موجود در مورد چارچوب NET. (و شاید این نظر از جهانی درست نباشد) آنست که این زبان، همانند جاوا از فناوری ماشین مجازی استفاده نمی کند. نرم افزار قابل اجرا در NET کاملاً کامپایل (همگردانی) می شود و همانند کد (رمز) کامپایل شده از یک برنامه C در سطح ماشین اجرا خواهد شد. این تصور غلط رایج از آن ناشی می شود که NET از یک زبان میانه که گاهی اوقات به عنوان "P – code" تعریف می شود، استفاده می کند. NET در واقع روش کامپایل (همگردانی) چند مرحله ای را به کار می برد. یک کامپایل اولیه به یک فرمت (قالب) میانه قابل حمل و یک کامپایل درجا (JIT) برای یک برگه نهایی قابل اجرا در زمان اجرا.

این سیستم (در حالی که نسبتاً پیچیده است) برای انجمن برنامه نویسی فواید زیادی دارد. یعنی در همه زبان ها دارای یک سطح عمومی هستند که در شکل میانه خود آن را به اشتراک می گذارند. بنابر این ترکیبی از ماجول(پیمانه) های نوشته شده در زبان های COBOL . FORTRAN , Eiffel , Visual Basic , C# یا هر زبانی که در آینده پشتیبانی خواهد شد میتواند برای زبان میانه (IL) کامپایل (همگانی) می شود و همراه با شرح کاملی از خود بسته بندی می گردد. رابط ها،خصوصیات (Prroperties) روش ها و کلاس (رده) های آن (همانند فوق داده ها) برای سایر ماجول (پیمانه) ها با محیط های توسعه برنامه

کاربردی که سریعاً مجتمع شده اند (نظیر Visual Studio. NET) در دسترس می باشند.

هنگامی که زمان اجرای NET برای اجرای یک برنامه تبدیل شده به IL استفاده می شود، NET از یک JITer برای کامپایل (همگردانی) IL به یک کد ماشین اصلی (درتبدیل کامل) که در ریز پردازنده دستگاه اجرا می شود، استفاده می کند. JITer ها حتی برای برخی از بخشهای بی استفاده کامپایل نشده از کد (رمز) میانه نیز بسیار سریع و مؤثر هستند. بدیهی است که این فرایند کامپایل وقت گیر است، بنابر این وقتی زمان های بارگذاری مهم هستند یا مسأله قابلیت حمل ضروری نیست، این سیستم یک کامپایلر pre – JIT را ارائه می دهد که آن رمز مبتنی بر IL را به یک قالب پایدار و اصلی تبدیل می کند.

چهارچوب NET چهار زبان اصلی را ارائه می دهد: C#، ویژوال بیسیک، C++ (همراه با الحاقات اداره شده) و Jscript. شرکت های دیگری نیز در حال ایجاد چند زبان دیگر هستند. مثلاً شرکت فوجیستو، نگارشی از COBOL را ارائه داده است.

مجموعه های اصلی از همگردان ها وجود دارند که زبان های تحت NET را می گیرند و ماجول (پیمانه) های زبان های میانه را ایجاد می کنند. در آنسوی این مجموعه ها، Jiter های IL به رمز ماشین هستند. یک Jiter استاندارد وجود دارد

که عمدتاً توسط سیستم های دارای قدرت و حافظه زیاد استفاده می شود. این نوع Jiter یک شکل استاندارد IL را به شکل استاندارد IL را به یک شکل بهینه شده در ماشین کامپایل (همگردانی) می کند. در نتیجه یک کامپایلر JIT مقرون به صرفه به دست می آید که سرعت زیادی دارد ولی کمی بهینه سازی شده است و وابسته به زیرمجموعه ای از IL از پیش بهینه می باشد. در نهایت per – JIT موجود یک کد (رمز) اصلی قابل اجرا ایجاد می کند که میتواند در شکل آماده برای کامپایل خود توزیع شود. این روشها تنها زمانی استفاده می شود که برنامه مورد نظر نتواند برای سکوی (دستگاه) سخت افزاری دیگری توزیع گردد.

سرانجام زمانی نسبی سخت افزار و سیستم عامل موجود در ماشین همان زمان اجرای زبان عمومی NET(CLR)، می باشد. این بلوک مؤثر خدماتی را ارائه می دهد که مؤلفه ها بتوانند از آنها استفاده کنند تا به رایانه شما یا OS اصلی ماشین تعامل داشته باشد. این نکته، جالب توجه است که میزبان ماشین CLR به عنوان یک PC شناخته نمی شود و سیستم عامل مورد نظر لزوماً نباید DOS یا ویندوز باشد. میکروسافت برای گسترش چارچوب، NET در بسیاری از سکوها (از کوچک ترین وسایل دستی ها، تلفن ها، PDA ها تاسیستم های لینوکس و یونیسک) یک راهکار (استرانژی) کاملی دارد. با وجود این که این راهکار در سیستم هایی غیر از سیستم 1386Windose 2000 سریعاً در دسترس قرار نمی

گیرند، ولی قابلیت حمل آن باعث شد توسعه گران یک محیط برنامه نویسی قابل اجرا در همه جا و یکبار نوشته شده را ایجاد کنند.

زبان میانه (**IL**) میکروسافت

یک نکته قابل توجه در مورد زبان میانه (IL) میکروسافت این است که این زبان در ماشین مزبور پنهان نشده است. IL، یک زبان کاملاً جدید و مبتنی بر stake (پشته) است و شباهت کمی با کد اسمبلی (همگذاری) دارد که در صورت لزوم می توانید آن را با دست بنویسید. همچنین ابزارهایی وجود دارند که شما را قادر می سازد تا IL را از اسامبلی (همگذاری) خارج نموده و محتوای موضوعات سیستم رمز مختص به خود را مشاهده کنید. در فص ۳-۱ زبان میانه IL جزئیات بیشتری از IL را بررسی خواهیم کرد و حتی می توانیم مستقیماً آن را برای نوشتن برخی از برنامه ها مورد استفاده قرار دهیم.

مقدمه ای برای مدیریت حافظه **.NET**

یکی از واقعیت های NET که افراد بسیاری را هیجان زده نگران و یا فقط مات و مبهوت می کند این است که مدیریت حافظه زمان اجرای NET همانند یک سیستم GC (جمع آوری آشغال) می باشد. برنامه نویسان قدیمی در روزهای

اوج زبان Lisp دچار کابوس می شدند، روزهایی وجود داشت که انتظار برای رسیدن به جمع کننده آشغال (garbage - collector) یک تجربه سخت بود زیرا این کار فقط در روزهای سه شنبه انجام می شد. برنامه نویسان C++، مدیریت حافظه ای داشتند که به آنها یادآوری می کرد که از دست دادن نظارت بر تخصیص حافظه ها و حذف ها کار درستی نیست.

سیستم مدیریت حافظه NET، تخصیص منابع حافظه را به شکل دیگری انجام می دهد. یک بلوک حافظه، به آشغال های جمع آوری شده یا کپه (heap) اداره شد، اختصاص دارد که سابقه (رکورد) همه موضوعات ارجاع داده شده به آن را حفظ می کند. فقط زمانی که آن مرجع ها ترخیص می شوند، موضوع از بین می رود. بدین وسیله، برنامه نویس از مسئولیت مدیریت حافظه خلاص می شود. شما دیگر مجبور نیستید که پاکسازی حافظه را به خاطر بسپارید، بلکه فقط باید استفاده از آن را متوقف نمایید. همچنین یک کلاس (رده) دیگر نباید از شمارش های مرجع آگاه باشد بلکه فقط زمان حذف خود را می داند. برای تکه تکه شدن کمتر کپه، GC موضوعات را جابجا می کند تا فضاهای خالی و استفاده شده ای که در ذخیره گاه حافظه اداره شده قرار دارند را یکپارچه نماید.

این امر مانع از نشست های (leaks) حافظه می شود و باعث پیشرفت کارآیی سیستم های خادمی که به شدت فشرده شده اند، می گردد. همچنین کپه اداره شده

باید اطمینان یابد که دستیابی های ناامن [نظیر سرریزش ها و صدمات بافر (میانگاه)] نمی توانند داده های مربوط به سایر اجرای برنامه های روی همان سیستم را اصلاح کنند. این امر کل سیستم عامل را ایمن تر و مطمئن تر خواهد کرد. سیستم های جمع آوری کننده آشغال، به نامؤثر و کند بودن معروف هستند. ولی میکروسافت به مرحله ای رسیده است که کارکرد واقعی جمع آوری آشغال در NET را تضمین می کند. این برنامه دارای سرعت بسیار زیادی است و تأثیر عمیقی در بارگذاری مهم CPU نخواهد داشت. معمولاً در یک برنامه ویندوز ۲۰۰۰ (مبتنی بر ماشین)، GC (جمع کننده آشغال) فقط به حدود ۱/۱۰۰۰ از کل زمان پردازنده نیاز دارد.

در نهایت برای برنامه نویسانی که به استفاده از اشاره گرهای موجود در بلوک های حافظه نیاز دارند وسیله ای در سیستم وجود دارد که کد (رمز) و حافظه نا امن یا اداره نشده ناامیده می شود می شود. این به صورتی است که شما می توانید ساختارهای قدیمی خود یا برنامه های کاربردی قبلی که مبتنی بر C++ هستند به موازات ویژگی های NET استفاده کنید.

سیستم های نوع چارچوب NET

برنامه نویسان C++ خصوصاً، با تغییرات ایجاد شده در نوع های (types) اصلی ذخیره گاه در چارچوب NET، استفاده شدند، شگفت زده خواهند شد. یک عدد صحیح دیگر چندبایتی از بلوک حافظه نیست. البته داده ها در حافظه ذخیره می شوند، ولی حالا می توانید آن عدد صحیح را به عنوان یک موضوع در نظر بگیرید و بر طبق روش های نمایش داده شده آن عمل کنید.

در اینجا دو خانواده کاملاً متفاوت از انواع وجود دارند. نوع های مقدار (vatue types)، شامل نویسه ها (chars)، اعداد صحیح (ints) و مضاعف ها (doubles) هستند که با نوع های مرجع (نظیر آرایه ها، رابط ها، کلاس (رده) ها و یک نوع رشته ای اصلی) همراه می شوند.

از آنجا که چهارچوب NET، خود نوع ها را تعریف می کند، در نتیجه زبان هایی نظیر ویژوال بیسیک، C# و سایر زبان ها می توانند این نوع داده ها را به همان صورت استفاده نمایند. این بدان معناست که عدد صحیح استفاده شده در VB (ویژوال بیسیک) درست همانند عدد صحیحی است که در C# به کار می رود و دیگر تبدیل پر دردسر بین نوع های واقعی و نوع هایی نظیر متغیرها، ضروری نیستند.

موضوعات سیستم چهارچوب **NET**

بخش های فعال چهارچوب NET، در مجموعه های از DLL ها قرار دارند که الگوی موضوع سیستم را حفظ می کند. نامکده (فضای نام) این سیستم مراتب های کلاس (رده) که مختص مجموعه ها، امنیت، I/O پرونده، گرافیک ها (نگاره ها) و دستیابی Win32 API ، چند بخشی کردن XML و بسیاری از تابع های مهم دیگر می باشند را در خود نگه می دارد. کل سیستم NET.

چگونه موضوعات، خود را تعریف می کنند

فوق داده ها (matadata) کلید اصلی عمل متقابل زبان در چارچوب NET هستند. در هر DLL یا اسمبلی (همگذاری) قابل اجرا یک توضیح کامل از کلاس (رده) ها، روش ها و لیست (سیاهه) های پارامتر (معرفه) ها و اعضای داده های کلاس (رده ای) وجود دارد. فوق داده های ذخیره شده در این سمبلی (همگذاری) به موضوع امکان می دهد در ابزارهای توسعه گر (نظیر Visual Studio .NET) مورد استفاده قرار می گیرد. این بدان معناست که موضوعات C# می توانند به راحتی در یک فرم ویژوال بیسیک نیز به کار می روند. به علاوه فوق داده ها در زمان اجرا در دسترس قرار می گیرند. سایر موضوعات و روش های نشان

داده شده را فراخوانی کنند. این امر به شما امکان می دهد تا از فوق داده ها به عنوان یک جایگزین ساده تر برای Idspatch استفاده نمایید.

عمل پذیری متقابل الگوی موضوع مؤلفه (**COM**)

چارچوب NET همه بارها و فشارهای ناشی از مدل (الگوی) موضوع مؤلفه (COM) را تحمل می کند. در مورد سیستم های دستیابی به موضوعات مبتنی بر NET و دوردست (remote)، این موضوعات شبیه به موضوعات COM هستند. همچنین رده های NET و موضوعات شبیه به موضوعات بدون نگرانی در مورد منظم کردن موضوعات COM می توانند آنها را دستکاری نمایند. CLR، همه این کارها را در پشت صحنه برای شما انجام دهد.

Windows From ها، کنترل وب و **CLR**، همه این کارها را در پشت صحنه برای شما انجام دهد.

اگر در طول ده سال گذشته، شما برنامه هایی برای مشتریان یا خادم های ویندوز توسعه داده اید، به احتمال رده های اصلی (Foundation) میکروسافت را نیز به کار برده‌اید. MFC، یک سیستم بسیار عالی برای ایجاد رابطه کاربری ثابت و پیچیده محسوب می شود، این رابط ها، جادوگرهای (wizards) مورد استفاده را

ایجاد می کنند. اشکال MFC، از این امر ناشی می شود که مدیریت آیتم (فقره) های عملیاتی (نظیر مجموعه ها، گرافیک (نگاره) ها، مدیریت پرونده، چندبخشی کردن و غیره) خود در کلاس (رده) های چارچوب MFC، ایجاد می شود. شاید حدود ۸۵٪ از آنچه شما در MFC با آن سر و کار دارید، در ساختار سطح پایین شامل یک لفاف (wrapper) کلاس (رده) موضوع گرا باشد که NET، روی آن تأثیر می گذارد. نامکده سیستم [که دارای بیت ها و قطعات عملیاتی (oprational) نظیر انواع داده های اصلی، مجموعه ها و چند بخشی کردن XML و غیره می باشند]، همانند یک کیف پر از موارد حاضر و آماده است که به طور ذاتی مؤلفه های موضوع گرا هستند. در نتیجه (frameword) مختص ایجاد برنامه های کاربردی ساده تر می باشد.

در مورد مشتری، برنامه Windows Forms همانند یک وسیله برای برنامه کاربردی پنجره ای عمل می کند و بر خلاف MFC می باشد، زیرا یک برنامه Windows Forms یک تمام رمز (کد) محسوب می شود. جعبه های محاوره ای و نماهای فرم (که با MSF در منابع طرح بندی شده اند) توسط کد (رمز) تنظیم کننده خصوصیات در رده های مبتنی بر System windows . Forms . Form ایجاد می شوند. برنامه های کاربردی Windows Forms از یک سلسله

مراتب کلاس ساخته می شوند و برای ایجاد یک برنامه قدرت ولی فشرده ویندوز از قابلیت های CLR استفاده می کنند.

کنترل (ناظم) های وب نوع دیگری از نمونه رابط کاربری را به کار می برند. آنها با اصل GUL غیر متصل کار می کنند. برنامه نویسان جاوا، یک خادمک (servlet) در اختیار داشتند و نیز برنامه کاربردی که گاهی اوقات در خادم (و از سوی مشتریان) اجرا می شود. خادمک، محاسبات را انجام می دهد، به تعامل های کاربری کمک نموده و با منطق تجاری یک برنامه کاربردی وب کار می کند، این در حالیست که کاربر طرف دیگر یک اتصال وب یک مرورگر مشتری کم کار (thin client) را به کار می برد. پیچیدگی یک برنامه کاربردی وب ممکن است طوری باشد که HTML در یک مشتری کم کار و متغیر لحظه لحظه نمایش داده شود (مثلاً زمانی که یک کاربر روی یک دکمهکلیک می کند، تغییری در UL ایجاد خواهد شد). این صفحه در پشت صحنه ها بازسازی می شود و یک صفحه جدید HTML که اختلافات موجود را نشان می دهد، به مرورگر مشتری فرستاده خواهد شد، معمولاً روند HTML بلافاصله پس از ارسال صحنه ایجاد می شود، بنابراین GUL قابل رؤیت برای کاربر پویا و غنی می باشد.

میکروسافت این مسأله را با نگارش جدید خادمک های خود به شکل کنترل (ناظم) های وب مربوط به سرور (خادم) پاسخ داده است. این مولفه ها زیر بخش

هایی از برنامه کاربردی توزیع شده را به همان روش تقویت یک برنامه کاربردی رومیزی توسط کنترل (ناظم) های الحاقی MFC ارائه می دهند. این برنامه کاربردی و کنترل (ناظم) هاییکه در ایجاد آن مؤثر هستند در خادم اجرا می شوند و رابط کاربر (به عنوان یک مسیل ASP شکل از HTML پویا) به ماشین مشتری پروژه (طرحانه) می شود.

GDI^+ همانند GDI سابق خود (رابط دستگاه گرافیکی) یک API گرافیکی نگاره ای با حالت سریع و پالایش شده تر است که شامل جزئیاتی نظیر چرخش دو بعدی و مقیاس

گذاری به اضافه تلفیق آلفا و ترانسپارنت (فرانمایی) می باشد. GDI^+ سرعت بسیار زیادی دارد و از پیشرفت های ایجاد شده در پردازش تصاویر نگاره ای جارویی (raster) که DirectX آن را به سیستم های قدیمی تر ارائه می دهد استفاده می کند.

ابزارها

NET SDK ، اصلی با ابزاری همراه است که کار توسعه گر را راحت تر می کند. برای ایجاد و نمایش IL ابزارهایی وجود دارند که عبارتند از: اسمبل (همگذار) زبان میانه (ILASM) و ناهمگذار زبان میانه (ILDASM) کامپلر (همگردان) سی شارپ (csc) و اشکال زدا (debagger) برنامه Visule

Studio . NET یک محیط توسعه ارجح است و بخش بزرگی از کل تجربه یک توسعه گر NET را تشکیل می دهد. ولی شامل SDK اصلی نمی شود. اگر نمیتوانید با VS . NET کار کنید یا اگر مایلید مفاهیم موجود را فقط امتحان نمایید بدون VS . NET [که از بستر یادداشت (Notepand) یا یک ویرایشگر متنی ساده دیگر استفاده می کند] نیز می توانید یک توسعه سودمند را داشته باشید. ما یک ویرایشگر شاخص کننده دستور نحوی (syntaX) C# نوشته ایم که پایین گذاری (download) از سایت (محله) وب به نام Singray به نشانی زیر می باشد:

http: / / www . stingray . com / csharpeditpr . aap

اسمبلی ها، سیستم بسته بندی **NET**

همگذاری (assembly) یک رسانه بسته بندی مؤلفه NET است. یک اسمبلی کد (رمز) ایجاد شده میانه توسعه منابع مختلف که مختص یک مؤلفه خاصی هستند را شامل می شود (این مؤلفه همان فوق داده های لازم برای مؤلفه ها و همه پرونده های اضافی و اطلاعات مورد نیاز برای اجرای مؤلفه مزبور می باشد.) هر همگذاری دارای یک اظهار نامه (manifest) است که پرونده های موجود در آن اسمبلی (همگذاری) را شمارش نموده و مؤلفه هایی که همگذاری های دیگر به آن

وابسته اند را تعریف می کند. همچنین اظهار نامه مزبور اطلاعاتی در مورد آنچه به کرابران اسمبلی (همگذاری) نمایش داده شده است را در خود حفظ می کند.

برنامه نویسی با صفات

NET از سیستم برنامه نویسی دارای صفت به طور عملی استفاده می کند که به موضوعات با روش ها امکان می دهد به روش خاصی توسعه داده شوند. مثلاً در C# هیچ تدارکی برای قالب آسا ها (templates) یا ماکرو (کلان ها) ندارد. این موارد به عنوان روش کارآیی یا ارائه عملگرهای طبق برندمه برای برنامه نویسان C++ شناخته شده هستند. صفات (همانند ماکروها یا قالب آساها)، کدی (رمزی) را در نرم افزار شما ردج می نمایند نکه در مورد عناصر جانشین پذیر، قوانین معینی را اعمال می کند. مثلاً امکان تبدیل یک کلاس (رده) ظاهراً معمولی C# (که دارای سه یا چهار خط رمز می باشد) به یکی ا ز خدمات وب اینترنتی در بالاترین درجه خود وجود دارد این کار را می توان با استفاده از صفت [Webme thod] موجود در کد (رمز) مبدأ انجام داد. با استفاده از این روش می توان در ظرف چند دقیقه چیزی همانند یک DLL الحاقی سرور (خادم) IIS کامل را ایجاد نمود و از اجرای خدمات اینترنت به طور مستقیم (در طول یک یا دو دقیقه) استفاده کرد. حتی طراحی صفات مربوط به خو.د نیز امکان پذیر است.

امنیت (**security**)

هر سیستم برنامه نویسی که در اینترنت امروزی کار می کند باید ایمن و مطمئن باشد. زمان اجرای NE، سیستمی را به کار می گیرد که مجوزهای مؤلفه را برای دستیابی به سیستم اجرا شده در آن بررسی و تقویت می نماید. این تنظیمات امنیتی را می توان دقیقاً انجام داد و به شما امکان می دهد تا مجوز داده شده به یک برنامه کاربردی (از یک قالب کاملاً خیالی تا هر سطح دستیابی که برای شما مقدور است) راتنظیم کنید.

مرور کلی

زمان اجرای زبان عمومی (CLR) نقش های بسزایی در NET به عهده دارد. CLR بخش مهمیس از سیستم اجرایی است، ولی به عنوان یک بخش فعال در توسعه و گستردگی نرم افزاریکه در یک چارچوب (framwwork) اجرا می شود، عمل نماید. CLR، پشتیبانی چند زبانی (multilanguge) را از طریق مدیریت کامپایلر (همگردان) هایی که برای تبدیل زبان مبدأ (surce) به زبان میانه (IL) و از زبان میانه به کد (رمز) اصلی استفاده می شوند، ارائه نموده و همچنین حفاظت و امنیت برنامه را تقویت می نماید.

بیایید به طور جزئی تر این نقش ها را بررسی کنیم.

ساده سازی توسعه

نقش توسعه CLR در کاهش بار تکلیف روز به روز ما بسیار مهم است. با استفاده از NET، شما نباید در مورد جزئیات ضروری GUIDS (یعنی Idispach , TUnKnown) یا کتابخانه های نوع نگرانی داشته باشید. وجود همه این موارد توسعه CLR$ به شمارش مرجع صریح نیاز ندارند، داده هایی که شما تخصیص می دهید هنگامی مکه دیگر از آنها استفاده نمی کنید، به طور خود کار توسط جمع کننده آشغال (GC) آزاد می شوند. فوق داده ها (metadata) نیز مقید سازی (binding) پویای قابل اجرا را امکان پذیر می کنند. این بدان معناست که در امنیت که مشکلات پیوندی DLL قدیمی، به طور کامل برطرف خواهند شد. همچنین اطمینان پذیری در امنیت نوع بیشتر می شود. هیچ یک از جزئیات اندازه یک ساختار یا سازمان اعضای یک موضوع شناخته نشده نیستند، بنابراین، هیچ مشکلی در مورد بسته بندی یا همترازسازی موضوعات وجود ندارد. به علاوه هرگز نباید در مورد بررسی تقید (bound) های آرایه ها یا بافرها (میانگیرها) نگران باشید.

پشتیبانی ابزار

CLR، با ابزارهایی نظیر Visual studio، کامپایلر (همگردان) ها، اشکال زداها (debuggers) و پروفایل (شرحانه) ها کار می کند تا کار توسعه گر راحت انجام شود. برای کسب تجربه توسعه زیاد، حتی Visual studio . NET نیز ضروری است. یک اشکال زدا SDK ای را ارائه می دهد که تقریباً به خوبی یک SDK مجتمع شده در VS . Net می باشد. سایر ابزارها نظیر ناهمگذاری (disassembler) زبان میانه، (ILDASM) از خدمات ارائه شده توسط CLR استفاده می کند.

پشتیبانی زبان چند گانه

اساس پشتیبانی زبان چند گانه، سیستم نوع عمومی (Common Type System) و فوق داده ها می باشد. انواع داده های اصلی که CLR آنها را مورد استفاده قرار می دهد، در همه زبان ها رایج هستند. بنابراین هیچ یک از نوع های اعداد صحیح اصلی، ممیز اعشاری و نیز نوع داده های رشته تغییر نخواهد کرد.

همه زبان ها به یک روش با تمام نوع داده های داده ای کار می کنند. همچنین برای تعریف و اداره کردن نوع های جدید یک مکانیسم (سازوکار) وجود دارد.

تمام زبانهای سطح بالا، به زبان میانه (IL) کامپایل (همگردانی) می شوند، وقتی عمل کامپایل (همگردانی) انجام می گردد و فوق داده های لازم برای موضوع ایجاد میشوند، همگردانی مزبور از طریق سایر زبان ها در دسترسی قرار می گیرد. این بدان معناست که موضوعات نوشته شده به یک زبان دوم قابل ارث بری هستند. در نتیجه این امکان وجود دارد که یک کلاس (رده) نوشته شده به زبان VB قابل ارث بری در زبان C# باشد.

تاکنون، چارچوب NET، از پانزده پشتیبانی کرده است که برخی از آنها عبارتند از: COBOL , Pascal , Jscript , Python , Perl , C# , Visual Basic , C++ و Smalltalk.

آماده سازی آسانتر می شود

با استفاده از چارچوب NET لازم نیست که مؤلفه (componet) های موجود در سیستم ثبت شوند. شما به راحتی می توانید یک مؤلفه را در یک فهرست (directory) رونوشت بگیرند و در این هنگام ، CLR به شما اطمینان می دهد که این مؤلفه به درستی کامپایل (همگردانی) و اجرا می شود. در صورتی

که کد (رمز) برای ASP . NET نوشته شده باشد، باید از کد مبدأ , Jscript VB یا C# اصلی کپی (رونوشت) بگیرید و سپس برای کامپایلر (همگردان) IL یک مبدأ صحیح فراخوانی خواهد شد.

دو امکان مؤثر وجود دارند که می توان مؤلفه (componest) را در آنها قرار داد. اولین مکان فهرستی است که برنامه کاربردی در آن قرار دارد. این مکان به نگارش های مختلف برنامه (همگذاری) سراسری (Global Assembly Catche) می باشد و ذخیره گاه مرکزی برای اسمبلی هایی است که قابل دسترس همه برنامه های کاربردی NET می باشند.

هر جا که مؤلفه ها ذخیره شوند، رویه بهنگام سازی ساده خواهد بود. شما فقط باید آن همگذاری (assembly) را در یک فهرست صحیح کپی (رونوشت) بگیرد، حتی اگر موضوع در حال اجرا باشد(البته نمی توانید این کار را در مورد یک DLL ویندوز انجام دهید) و سیستم نیز موجب کارکردن آن باشد.

جدا سازی نرم افزار

برنامه های کاربردی با استفاده از DLL از یکدیگر مجزا می شوند، به طوری که نتوانند بر هم تأثیر بگذارند. شاید منابع به اشتراک گذاشته شوند، ولی این اشتراک گذاری باید به طور صریح انجام گردد. این بدان معناست که یک سیستم

NET می تواند در صورت لزوم یک رویه قالبی (sandbox) را برای امنیت سیستم داشته باشد. وقتی امنیت بیشتر با دستیابی به هر منبع سیستم خاص مورد نظر شما باشد، دستیابی به پرونده ها یا سایر منابع موجود در سیستم (طبق اصل برنامه به برنامه) قابل تنظیم است.

وارسی و امنیت نوع

CLR، اطمینان پذیری و امنیت از طریق امنیت نوع، وارسی (verification) و اعتماد انجام می دهد. CLR، کد نوع ایمن (Type - safe) را برای هماهنگ کردن موارد مختلف ایجاد می کند. این رمز فقط از نوع های معرفی شده استفاده میکند، که نه تنها حافظه تخصیص داده شده به آن را از طریق مدیر حافظه به کار می برد و نمی تواند به روش ها یا داده های غیر عمومی (none - public) مربوط به سایر برنامه های کاربردی یا فرایندها دسترسی داشته باشد. یک برنامه نوع ایمن در جایی که به بیرون از فضای خارجی رفته و در همه بلوک های حافظه ای سایر فرآیندها نوشته می شود، دچار قطع سیستم نخواهد شد. وارسی امنیت نوع، زمانی اتفاق می افتد که CLR، کد را بارگذاری (load) و اجرا کند. CLR در طول فرایند هر روش JIT، توضیح فوق داده های آن روش را مورد توجه قرار می دهد و وجود امنیت نوع را وارسی میکند. مثلاً، در شرایطی که این نوع وارسی

غیر ممکن است و CDR باید رمز ادره نشده (unmanaged) را از طریق یک موضوع اداره شده (managed) مورد استفاده قرار دهد. این حالت غالباً زمانی پیش می آید که کلاس (رده) NET، یک Win32 DLL را فراخوانی کند. در این هنگام، کد(رمز) باید مورد اعتماد CLR باشد.

امنیت

موضوع امنیت، در سیستم ها از اهمیت خاصی برخوردار است، به طوری که می تواند محتوای فعال یا اسکریپت (امریه) های کاربردی اجرایی را اجرا کند. شما به عنوان یک مشتری (client) اجازه نخواهید داد که برنامه های مخرب در ماشین (دستگاه) شما اجرا شوند و داده های با ارزش را از بین ببرند. همچنین به عنوان یک ارائه دهنده خدمات اجازه نخواهد داد که حملات یا اشتباهات ساده ای که از جانب کاربران ایجاد می شوند، کل سیستم شما را تخریب کنند.

CLR، امنیت سیستم را از طریق کاربر و هویت (identify) کد (رمز) مربوط به بررسی های مجاز اداره می کند. هویت مرز (مثلاً شامل ناشر و اصل کد) را می توان شناسایی نمود و بنابر این استفاده از منابع صحیح را مجاز کرد. این نوع امنیت، امنیت مبتنی بر صحت (Evidence Based Security) نامیده می شود و یکی از ویژگی های مهم NET، می باشد. چهارچوب NET، با استفاده از گروه

ها و حسابهای ویندوز NET از امنیت مبتنی بر نقش (role - based) نیز پشتیبانی می کند.

رابطه **CLR** و **NET**

نمودار بلوک نشان داده شده در شکل ۱-۲-۱، رابطه CLR با چارچوب عمومی NET را نشان می دهد.

جزئیات **CLR**

به عنوان یک زیرمؤلفه ای از چارچوب NET و CLR از قسمت های جداگانه ای ایجاد می شود، شکل ۲-۲-۱ رابطه این اجزا را با یکدیگر نشان می دهد.

همان طور که در شکل ۲-۲-۱ می بینید تابع اولیه CLC یک بارگذار (loader) کلاس (رده) می باشد. CLR کلاس هایی که شما ایجاد می کنید با آنهایی که به عنوان بخشی از کتابخانه کلاس پایه ارائه می شوند را بارگذاری نموده و برای استفاده آماده می کند و سپس آنها را اجرا نموده و یا کاربرد زمان طراحی آنها کمک می کند.

کلاس ها از اسمبلی (همگذاری) هایی که با EXE یا DLL در NET هم ارز هستند بارگذاری می شوند. گاهی اوقات یک اسمبلی ممکن است حاوی کاد

اصلی باشد که احتمالاً کلاس های کامپایل (همگردانی) شده به IL و فوق داده های مربوط به آنها را شامل می شود. در زمان طراحی، CLR یا ابزارهایی نظیر Visuale Studio. NET ارتباط برقرار می کند تا توسعه سریع برنامه های کاربردی (Papid Appication Development) را ارائه می دهد. برنامه نویسان VB برای مدت زیادی از آنها استفاده کرده اند و این امر در همه ماجول (پیمانه) های موجود در همه زبان ها صدق نمی کند. بارگذار کلاس، کلاس ها را در زمان اجرا (runtime) اجرا می نماید.

CLR در زمان اجرا

وقتی بارگذار رده (class loader) یک کامپایل (همگذاری) را در زمان اجرا شروع می کند و قبل از اجرا شدن رده ها، باید مراحل مهمی را انجام دهند. شکل ۳-۲-۱ ، الگوی اجرای CLR را نشان می دهد.

بارگذار کلاس اداره کننده رمز (Code Manger) را به کار می برد تا حافظه ای را برای موضوعات و داده ها تخصیص دهد. لی آوت (طرح بندی) کلاسهای موجود در حافظه محاسبه می شود و به هر یک از این روش ها (که به عنوان زبان میانه ذخیره می شوند) یک برنامه کمکی ارائه می گردد. این برنامه کمکی برای فراخوانی کامپایلری (همگردانی) که برای اولین بار اجرا شده است مورد استفاده قرار می گیرد.

توجه:

در شکل ۳-۲-۱ می بینید که دو نوع مشخص از زبان میانه وجود دارند (Optil , MSIL) که توسط کامپایلر (همگردان) های JIT مختص به خود کامپایل می شوند. نوع OptIL، زیر مجموعه بهینه شده MSIL می باشد که به ویژه برای استفاده سیستم های میزبان فاقد موارد اضافی با سرعت محاسباتی طراحی شده است. به دلیل کار زیاد که کامپایلر (همگردان) زبان سطح بالا برای این بهینه سازی انجام می دهد. نوع OptIL با FDA ها تلفن های هوشمند و نظیر اینها متناسبتر است. EconoJIT یک کامپایلر (همگردان)؛ قابل حمل و فشرده است که OptIL را به کد (رمز) ماشین تبدیل گردد.

هر وقت یک تابع برای اولین بار به یک کلاس ذخیره شده در اسمبلی (همگذاری) دیگر با نوع داده هایی که قبلاً استفاده نشده اند، ارجاع داده شود، بارگذار (loader) کلاس برای ایجاد موضوعات مورد نیاز مجدداً فراخوانی خواهد شد. این فرایند احتمالاً با بررسی فلوچارت (روند نما) بهتر قابل درک می باشد. سیستم اجرایی مجازی (VES) که به زمان اجرای زبان عمومی (CLR) تعلق دارد و همچنین مسیرهای اصلی نظارت در شکل ٤-۲-۱ نشان داده شده است.

تولید کد زمان نصب

بخشی از VES که در شکل ٤-٢-١ نشان داده نشده است، تولید رمز زمان نصب می باشد. این بخش، زمانی ایجاد می شود که اسمبلی (همگذاری) ها برای اولین بار در یک سیستم میزبان قرار گیرند. این بخش، کد اصلی را اسمبلی ایجاد خواهد کرد که زمان شروع یک موضوع خاص را کاهش می دهد. معمولاً وارسی این کد (رمز) در زمان نصب انجام می شود ولی اگر نگارش یک زمان اجرا یا بررسی های امنیتی ضروری باشند، این کد مجدداً توسط JITer استاندارد و به طور معمول کامپایل (همگردانی) خواهد شد.

انواع داده های پشتیبانی شده توسط **CLR**

همان طور که قبلاً گفته شد چارچوب NET، دارای مجموعه ای از نوع های داده های اصلی می باشد که توسط همه زبان ها استفاده می شوند. این انواع، شامل نوع های صحیحی نظیر نویسه (کاراکتر) ها و اعداد صحیح ۸ و ١٦ و ٣٢ و ٦٤ بیتی هستند. همچنین این لیست (سیاهه) شامل انواع ممیز اعشاری و اشاره گرهای مختص به حافظه اداره شده (managed) و اداره نشده (unmanged) ومی باشد.

شاید همه نوع های داده های اصلی توسط رمز نوشته شده در IL مورد استفاده قرار گیرند و در نتیجه شما می توانید IL را به طور دستی بنویسید. IL شباهت بسیار زیادی با زبان اسمبلی دارد، بنابراین اگر شما تاکنون با یک Z80 یا 6502 کار کرده اید، در مورد کار با IL نیز راحت خواهید بود. در ادامه همین بخش یک IL دست نویس و ابزارهای همراه با چارچوب NET را به شما نشان خواهیم داد، که این ابزارها را می توانید برای اسمبلی (همگذاری) و نیز نامگذاری آن استفاده کنید.

جدول ۱-۲-۲ نوع های اصلی داده های CLR

نام نوع	توضیح
I1	مقدار علامت دار مکمل دو ۸ بیتی
U1	مقدار دودویی بدون علامت ۸ بیتی
I2	مقدار علامت دار مکمل دو ١٦ بیتی
U2	مقدار دودویی بدون علامت ١٦ بیتی
I4	مقدار علامت مکمل دو ۳۲ بیتی
U4	مقدار دودویی بدون علامت ۳۲ بیتی
I8	مقدار علامت دار مکمل دو ٦٤ بیتی

U8 مقدار دودویی بدون علامت ٦٤ بیتی

R4 مقدار ممیز اعشاری ٦٤ بیتی IEEE 754

R8 مقدار ممیز اعشاری ۳۲ بیتی IEEE 754

I مقدار علامت دار مکمل دار دو با اندازه معمولی

U مقداردودویی‌بدون علامت با اندازه معمولی(وهمچنین اشاره گرادار نشده)

نام نوع	توضیح
R4Result	اندازه معمولی برای نتیجه یک محاسبه ممیز اعشاری ۳۲ بیتی
R8Result	اندازه معمولی برای نتیجه یک محاسبه ممیز اعشاری ٦٤ بیتی
Rprecise	مقدار ممیز اعشاری یا حداکثر دقت
O	مرجع موضوع با اندازه معمولی برای حافظه اداره شده
&	اشاره‌گر اداره شده با اندازه معمولی (گاهی به حافظه‌اداره‌شده‌اشاره می‌کند)

جدول ۱-۲-۱، به اندازه های رایج برای انواع داده ها اشاره شده است. در این جا منظور اندازه هایی هستند که توسط سخت افزار اعلام می شوند. مثلاً ماشینی که دارای یک گذرگاه (bus) ۱۶ بیتی است، احتمالاً یک عدد صحیح معمولی ۱۶ بیتی دارد و ماشین با یک گذرگاه۳۲ بیتی که به همین نسبت یک عدد صحیح بزرگ و متناظر خواهد داشت. این امر زمانی مشکل ایجاد می کند که بخواهید با نرم افزار موجود در طرف دیگر یک اتصال اینترنتی گفتگو داشته باشید. بدون اجرای دستورالعمل های صریح شناخت مقدار بزرگی یک عدد صحیح در هر طرف مشکل است و بنابر این عدم هماهنگی اندازه نوع و ناهمترازی هعای ساختار بوجود خواهد آمد. برای برطرف کردن این مشکل و سایر مشکلات مشابه CLR فقط برخی از نوع های داده های انتخاب شده را در هنگام ارزیابی و اجرای نرم افزار استفاده می کند. نوع اصلی که به اندازه های انتخاب نشده هماهنگ نیست، به دقت به انواع قابل جابجا تری بسته بندی می شوند و از حالت بسته بندی خارج می گردند. این عمل در پشت صحنه ها روی می دهد و شما نباید نگران آن باشید.

کد و داده های اداره شده

CLR، شامل اداره کننده رمز (Code Manager) و جمع کننده آشغال (GC) می باشد. این بلوک های تابعی مسئول مدیریت حافظه همه موضوعات اصلی

NET هستند که شامل همه انواع داده های اداره شده می باشند. اداره کننده رمز، فضای ذخیره گاهی را تخصیص می دهد: جمع کننده آشغال آن را حذف نموده و کپه (heap) موجود را فشرده می کند تا میزان گسستگی آن کاهش یابد.

از آن جا که GC مسئول فشرده سازی کپه است داده ها یا موضوعات استفاده شده در فضای اداره شده NET اغلب مواقع در طول عمر خود در دسترس قرار می گیرند. در دنیای C++، یک مرجع تفاوت زیادی با یک اشاره گر ندارد. به عبارت دیگر سازوکار (مکانیسم) مورد استفاده برای پیدا کردن موضوعی که به آن ارجاع شده است با پیدا کردن یک موضوع توسط اشاره گری، در محلی از بلوک حافظه یکسان می باشد. مرجع های C++ ، مرجع های C++ همان روش همگردانی هستند که گویی تضمین می نماید که شما به یک نوع خاصی اشاره می کنید. در C++، امکان ذخیره یک مرجع برای یک موضوع حذف موضوع و سپس استفاده از مرجع قدیمی و ذخیره شده وجود دارد تا بتوانید به مکانی که می خواهید از موضوع آن استفاده کنید، دست یابید اغلب مواقع این امر سبب شکست بزرگی در یک برنامه C++ می باشد.

با این وجود این مشکل در سیستم NET برطرف شده است. هر وقت چیزی به یک موضوع ارجاع داده شود، آن موضوع از مرجع بودن خودآگاهی دارد و GC هرگز آن را حذف نخواهد کرد تا زمانی که مانع برطرف شود. به علاوه، وقتی GC

کپه را مشغول نموده و موضوع را در حافظه جابجا می کند، همه ارجاع های به آن موضوع به صورت خودکار بهنگام می شوند به طوری که مؤلفه های استفاده کننده از موضوع مورد نظر به درستی به موضوع بعدی دست می یابند.

با استفاده از اداره کننده کد (رمز) CLR، مسئول لی آوت (طرح بندی) حافظه واقعی موضوعات و ساختارهایی است که آن را میزبانی می کند. معمولاً، این فرآیند طرح بندی به طور خودکار اجرا می شود، ولی وقتی مجبورید این کار را انجام دهید میتوانید ترتیب، بسته بندی و لی آورت خاص حافظه موجود در فوق داده ها را مشخص کنید. این امر هنگامی امکان پذیر می شود که فوق داده ها از طریق مجموعه رابطهای مشخصه در دسترس برنامه نویس قرار گیرند.

رمز اداره نشده و دستیابی به داده ها

شاید معلوم نباشد که وارسی، حفاظت و امنیت نوع NET چطور به شما امکان می دهند از سرمایه خود در رمز C++ (که به مدت طولانی از آن مراقبت کرده اید) پشتیبانی کنید. میکروسافت نیز دقیقاً همین وضعیت را داشت، در نتیجه چارچوب NET برای استفاده مجدد از رمز قبلی شما دارای قابلیت های بسیار خوبی می باشد.

برای عملیات متقابل اداره شده/نشده تحت NET، سه نوع ساز وکار وجود دارد. Introp COM، به موضوعات COM شما امکان می دهد که توسط چارچوب NET مورد استفاده قرار گیرند، به طوری که گویی آنها موضوعات NET، هستند. روش Platfrom Invoke یا (P/ Invoke) باعث می شوند موضوعات اداره شده، نقاط مدخل ایستایی که در DLL ها وجود دارند را همانند روش های Loadlibrary و GetProcAddress فراخوانی کنند. در نهایت IJW (It Just Work) در ابتدایی ترین شکل خود به شما امکان می دهد تا کد (رمز) خود را کامپایل (همگردانی) مجدد نمایید و ... این فقط عمل می کند. یک فرم پیچیده تر به شما امکانم می دهد تا با استفاده از الحاقات Managed C++ (اداره شده)، رمز خود را بهینه کنید. این امر باعث می شود شما موضوعات اداره شده حافظه و کاملاً پیچیده GC را ایجاد نمایید که از کد مبدأ قدیمی C++ شما استفاده می کند.

COM Interop از طریق **CLR**

در شکل ۲-۲-۱ می بینید که CLR شامل یک منظم کننده COM می باشد. این بلوک تابعی مسئول عملیات متقابل COM /. NET است. دو سناریو وجود دارند که مستلزم عملیات متقابل COM با چارچوب NET هستند. سناریوی اول، زمانی روی می دهد که شما بخواهید از طریق یک کد (رمز) جدید نوشته شده

VB یا #C، به موضوعات قدیمی COM خود دست یابید. دومین سناریو نیز زمانی به وجود می آید که بخواهید یک رابط معروف را اجرا کنید و موضوعات COM شما به آن دسترسی داشته باشند.

در مورد این سناریوها ، clr با ایجاد یک لفاف ویژه در زمان اجرا و منظم کردن داده ها در داخل و خارج از موضوع COM (در صورت لزوم) یک نقش بسیار پویا را ایفا می کند. نمودار موجود در شکل ۵-۲-۱، یک موضوع COM را نشان می دهد که در دسترس یک موضوع مشتری NET، می باشد.

برای استفاده از موضوعات COM با برنامه های NET خود باید تعریف کتابخانه نوع را وارد کنید. وقتی شما یک مرجع را به موضوع COM موجود در IDE اضافه نمایید، VS. NET این کار را به طور خودکار برای شما انجام می دهد (دقیقاً به همان روشی که برنامه نویسان VB در طول سالها متمادی انجام داده اند). همچنین می توانید صریحاً از برنامه سودمند وارد کردن کتابخانه نوع (TLBImp) استفاده کنید. TLBImp، برای شما توشیح های روش COM را به توشیح های روش NET تغییر می دهد. مثلاً توشیح روش COM:

HRSULT MYMethode , [out, retval) (BSTR) b, Int n

به:

int MyMethod (sring b, int n):

تغییر می یابد.

در نتیجه وقتی شما روش مزبور را فراخوانی می کنید، نباید در مورد تفسیر HRESULT ها نگران باشید، بلکه فقط می توانید مقدار بازگشتی عدد صحیح را به یکی از متغیرهای موجود در کد (رمز) خود نسبت دهید. در واقع استفاده از تصویر عملیات متقابل NET به COM بسیار ساده است. در مورد تبدیل های داده ها نگران نباشید. همان طور که در این مثال می بینید، انواع داده های COM (نظیر BSTR) به انواع هم ارز مناسب خود در NET نگاشت می شوند. شما مجبور نیستید شمارش مرجع را اداره کنید و هیچ گونه GUIDS نیز وجود ندارد. اگر شما در مورد HRESULT موفق نشوید لفاف (wrapper) استثنایی را ایجاد می کند که شما بتوانید آن را به دست آورید.

هنگامی که می خواهید یک موضوع NET، را از موضوعات COM موجود فراخوانی کنید. فرایندی که CLR برای تسهیل اتصال مورد استفاده قرار می دهد در شکل ساده تری با فرآیند موجود موجود در شکل ۵-۲-۱ شباهت دارد. شکل ٦-۲-۱، یک اتصال NET به COM را نشان می دهد.

با وجود اینکه موضوعات COM فقط می توانند به روش ه ای عمومی موضوعات NET، شما دسترسی داشته باشند، ولی می توانند از طریق دنیای COM نیز مورد استفاده قرار گیرند.

ابتدا با استفاده از ابزار TipExp موجود در NET یک typelib را از موضوع خود ایجاد کنید. سپس این اسمبلی (همگذاری) ایجاد شده توسط TipExp را با استفاده از RegAsm ابزار ثبت نمایید. دراین زمان فقط باید همه موارد تحت NETh را ثبت کنید. برای اجرای COM در بخش اداره نشده رایانه میزبان شما به فعال نمودن موضوع مزبور با روشی استاندارد لازم است. درنهایت شما می توانید QueryInterface و CoCrreeInstance را از طریق COM، Addref , Release را به طور عادی و HRESULT هایی که یک لفاف ارائه می دهد، را مورد استفاده قرار دهید.

در بخش ۳-۵، یک فصل به عملیات متقابل COM اختصاص داده شده است که جزئیات و مثال های بیشتری را ارائه می دهد.

احضار سکو (**P/ Invoke**) از **CLR**

یکی از عمومی ترین لفاف عملیات متقابل به کارگیری DLL ای می باشد که از قبل در سیستم محلی وجود داشته است. مثلاً شاید شما مجبور شوید از ابزارهایی که دراختیار دارید و فقط در دسترس DLL ها هستند، استفاده کنید.

چارچوب NET، سازوکار P/ Invoke را بدین منظور ارائه می دهد. این سازکار داخلی شباهت زیادی با عملیات متقابل COM دارد، ولی به دلیل این که

DLL ها مورد استفاده شما همیشه (در همان ماشین) محلی هستند، استفاده از این سیستم راحت تر است.

برای استفاده از سازوکار P/ Invoke به لفافی نیاز دارید که از طریق CLR فراخوانی ها را به یک تابع DLL واقعی نسبت می دهد. شما با استفاده از صفات خاص موجود در تعریف های روش خود این صفات خاص را ایجاد می کنید.

کد (رمز) موجود در مثال زیر به شما امکان می دهد که یکی از توابع موجود در یک DLL سفارشی را فراخوانی نمایند.

```
Public class MyDiWrapper
  [ D11Import (“MyDii. D11” , EntryPoint- “) ]
  public atatic extern int MyFunction (int x)؛
```

حال ممکن است شما با استفاده از فرمان های C# به DLL دست یابید:

```
int r=MyD11Wrapper . MyFunction ؛
```

در فصل ٤-١، کار با الحاقات اداره شده (Managed) به C++ به DLL دست یابید:

سازوکار P/ Invoke با جزئیات بیشتر بررسی خواهد شد.

IJW (این فقط عمل می کند)

ساده ترین روش برای مهاجرت برنامه های کاربردی C# شما چارچوب .NET، جمع آوری کل رمز مبدأ برای EXE یا DLL شما می باشد. همه آنها را مجدداً با سوئیچ (دگرگزینی) کامپایلر (همگردان) CLR/ کامپایل (همگردانی) و اجرا کنید. میکروسافت معتقد است این فقط عمل میکند. اگر برنامه نویسی شما بر طبق یک روش منطقی و نوین باشد این نظریه صحیح خواهد بود و در صورتیکه که کد (رمز) شما هنوز هم اعلانی هالی قدیمی و خوب سبک K&R را شامل می شود این گفته صحت نخواهد داشت.

محدودیت هایی برای این فرایند وجود دارند. شما نمی توانید همه کتابخانه های کلاس (رده) خود را مجدداً کامپایل کنید و سپس در دنیای اداره شده (managed) از آنها ارث برید. همچنین نمی توانید به راحتی اشاره گر کد (رمز) اداره شده را به کلاس (رده) ها و توابع خود رد کنید.

هر دو روش های P/ Invoke و IJW طول عمر کد شما را افزایش می دهند و فرصت مهاجرت به چارچوب .NET را برای شما به وجود می آورند.

الحاقات اداره شده (**Managed Extension**) به **C++**

هنگامی که تنها استفاده مجدد از رمز قدیمی شما کافی نیست و یک رویکرد یکنواختی تری از نگرش پرسپکتیو (دورنمای) یک برنامه نویس C++ مورد نیاز

است، الحاقات اداره شده برای C++ یک مسیر مهاجرت یا یک راه حل بسیار مناسب برای این منظور محسوب می شود.

چارچوب NET، برخی از مفاهیم مهم را به مدیریت حافظه و کاربرد داده ها اضافه می کند این ویژگی ها از طریق زبان های VB و C# در دسترس هستند (نه از طریق C++) مواردی که به مجموعه برنامه نویس اضافه شده اند در بخش های بعدی مورد بررسی قرار می گیرند.

کلاس های جمع آوری آشغال

کلاس جمع آوری آشغال (GC)، موضوعاتی هستند که توسط CLR کل مدیریت حافظه برای آنها اجرا می شود. وقتی استفاده از یک کلاس GC را متوقف می کنید، این کلاس (رده) به منظور حذف و بازیابی تسوط کپه GC نشانه گذاری می شود. یک کلاس GC دارای یک روش حذف باشد که می تواند صریحاً فراخوانی گردد.

پشتیبانی از اشکال زدایی

چارچوب NET، دارای نوعی اشکال زدایی است که در یک سطح بسیار پایین ایجاد شده است. برخلاف شما (طرحواره) های اشکال زدا که برنامه نویسان C++ از آنها استفاده می کردند و در اغلب مواقع تحمیلی هستند، CLR (که همه رمز را

اجرا می کند) اشکال زدایی را اداره نموده و در مواقع لازم رخـدادها (events) را به اشکال زدایی متصل شده ارسال می کند.

NET SDK (علاوه بر اشکال زدایی که با Visual Studio . NET مجتمـع شده است) دارای اشکال زداهای مختص به خود می باشد. همـه اشـکال زداهـا از طریق API های عمومی که توسط این چارچوب ارائه می شوند ، عمل می کنند.

خلاصه

زمان اجرای زبـان عمـومی (CLR)، قلـب سیسـتم NET اسـت. CLR کـد (رمز) را بارگذاری و اداره می کند، با ابزارهای توسعه و اشکال زداها فعل و انفعال دارد، امنیت سیتسم شما را تقویت نموده و به نرم افـزار شـما امکـان مـی دهـد بـا سیستم های قبلی (نظیر COM و DLL کاربر) مجتمع شود. از آنجا که CLR به عنوان یک مؤلفه NET در نظر گرفته می شود بعید است که شما در مورد آن زیاد فکر کنید

همان طور که در دنیای زبان میانه (IL) می گردیم و نحوه نوشتن کد را به زبـان اصلی و مناسب NET (به طور دستی)بررسی می نماییم شما هـم ایـن مطالـب را مطالعه کنید.

محیط توسعه مجتمع (**IDF**)

اگر با نگارش ۵/x یا ۶/x برنامه Visual Studio آشنا هستید، Visual Studio. NET برای شما غریب و ناآشنا نخواهد بود. با این وجود تفاوت هایی وجود دارند که باید مورد توجه شما قرار گیرند تا با این ابزار جدید، مسیر صحیح توسعه مؤثر قرار گیرند.

VS . NET همتای بزرگتر و پیشرفته تر VS قدیمی می باشد. در گذشته C++ و ویژوال بیسیک با استفاده از برنامه های کاربردی مختلف توسعه داده شدند. ولی در حال حاضر انواع مختلف زبان ها نظیر C++ ، JavaScript، ویژوال بیسیک و بسیاری از زبانهای دیگر که در این چارچوب اجرا می شوند در اختیار شما قرار می گیرند. همچنین ایجاد یک برنامه کاربردی که مستلزم استفاده بسیاری از زبان ها یا مؤلفه های (components) توزیع شده می باشند، عمومیت پیدا می کند. در نتیجه محیط توسعه پیچیده تر می شود و می تواند همه بخشهای برنامه کاربردی شما را با یک روش یکنواخت اداره نماید.

همانند بیشتر موضوعات این کتاب احتمالاً Visual Studio صفحات بسیاری زیادی را به خود اختصاص می دهد، ولی در این فصل یک مرور کلی روی آن خواهیم داشت تا با این نرم افزار آشنا شود.

بخش A : ناحیه اصلی ویرایش

اولین چیزی که در این ناحیه می بینید ، Start Page است که مانند بسیاری از پنجره های VS. NET، یک پنجره مرور گر HTML می باشد و چند نمونه از آخرین طرحانه (پروژه) ها پیوندهایی به محله پشتیبانی مشتری (client) و اطلاعات سودمند دیگری (در مورد بسته نرم افزاری VS . NET) را در اختیار شما قرار خواهد داد. همچنین این ناحیه محلی است که در آن همه پرونده های مبدأ (surce files) ویرایش می شوند. بالای این پنجره یک نوار زبانه دار (tabbed bar) وجود دارد که به شما امکان میدهد تا به راحتی از پرونده هایی که به طور جاری باز می باشند را برای ویرایش انتخاب کنید.

بخش **B**: راه حال ها (**Solutions**)، کلاس (رده) ها، آیتم (فقره) ها و راهنما (**Help**)

در این قسمت از پنجره IDL یک کنترل (ناظم) زبانه دار دیگر وجود دارد که دارای مقطع های (Panes) متعددی می باشد. این کنترل در واقع یک تم (شاهدک) بازگشتی در VS. NET است. زیرا IDL به قدری شلوغ است که جایی برای قرار دادن همه اطلاعات آن وجود ندارند.

به طور پیش فرض، این مقطع (pane) شامل پنجره های Help Index , Help Search و Contents Help , Class View , Sulition Explorer می باشد. برخی دیگر از زبانه ها که بعداً توضیح داده خواهند شد نیز در این مقطع وجود دارند.

راه حل ها و جستجو گر راه حل (**Solutionn Explorer**)

راه حل (Solution) یکی از بسته های نرم افزاری مهم کار در VS . NET می باشد. یک راه حل می تواند پروژه (طرحانه) های بسیاری از نوع های (tyopes) مختلف را نگهداری کند. مثلاً ممکن است بخواهید یک برنامه کاربردی کامل کلاینت (مشتری)/ سرور (خادم) را ایجاد نمایید که دارای بسیاری از زیرپروژه (طرحانه) های منفرد، (از کد مشتری مبتنی بر MFC تا کد خادم ASP . NET) باشد. این راه حل یک مکان نگهدار (placeholder) سطح بالا همه زیر طرحانه های Sub – project) مربوط به برنامه کاربردی مزبور است. کل یک راه حل را می توان همزمان ایجاد نمود و وابستگی های بین زیرپروژه ها را نیز طوری تعریف کرد که به ترتیبی خاص ایجاد شوند.

با انتخاب زبانه Solution Explorer، یک مقطع همراه با درخت آیتم (فقره) ها ظاهر می شود. این درخت شامل پروژه ها، پوشه ها (folders)، پرونده ها،

منابع (resurses) و سایر داده های مختلف (که قلمرو جاری کار شما را تعریف می کنند) می باشد.

یک راه حل دارای خصوصیاتی است که بر همه پروژه (طرحانه) های آن تأثیر می گذارند. همانند بیشتر آیتم (فقره) های دیگر که در Visual Studio . NET وجود دارد، کلیک راست روی یک Solution و انتخاب گزینه Properties از منوی حاصله به شما امکان می دهد که مشخصات راه حل خود را تنظیم کنید.

پانل (تختانه) Solution Explorer، (در منتهی الیه راست) یک پروژه (طرحانه) به نام "AnApplications" را شامل می شود. جعبه محاوره ای Solution Properties (سمت چپ) به شما امکان می دهد تا بتوانید زیرپروژه ای که پرونده می باشد، نحوه وابستگی پروژه ها به یکدیگر و همچنین جایی که IDE در حین اشکال زدایی پرونده های مبدأ را جستجو نموده و نیز محل ذخیره پرونده های نماد (symbol) اشکال زدایی را انتخاب کنید. این تنظیمات برای همه پروژه های یک راه حل مشتری هستند.

پروژه ها

همان طور که راه حل ها پروژه (طرحانه) هایی را شامل می شوند، پروژه ها نیز حاوی آیتم (فقره) ها هستند. این آیتم ها در برخی از موارد می توانند به عنوان

پوشه هایی باشند که برای سازماندهی مبدأ مورد نظر سرصفحه و پرونده های منبع، مرجع های (references) سایر پروژه ها خدمات وب،ؤ پرونده های سیستم و نیز همه پرونده های مبدأ (که پروژه درون یک راه حل را ایجاد می کنند) استفاده شوند. در شکل ٤-٥-١ مشاهده می کنید که پروژه مزبور دارای دو پرونده به نام های Form1 . cs و AssemblyInfo . cs و مجموعه های از مرجع ها می باشد. این مرجع ها نسبت به رهنمود (رهنمود) #include که در یک پرونده C++ قرار دارد، قیاسی هستند و امکان دسترسی به فوق داده های (metadata) موجود در نامکده (فضای نام) های نشان داده شده را به وجود می آورند. این مرجع ها به پرونده های واقعی (actual) و داده هایی که در ماشین شما وجود دارند وابسته هستند و همچنین می توانند بر شرح زبان توصیف خدمات وب (WSDL) متعلق به یک خدمات وب ارجاع شوند. این نوع از مرجع یک مرجع وب (Web Refrence) نامیده می شود.

پروژه های متعدد موجود در یک راه حل منفرد

افزون یک پروژه به یک راه حل ساده است. بدین صورت که روی این راه حل کلیکر راست کرده و و گزنه Add را از منوی حاصله انتخاب کنید. انتخاب نوع ایجاد یک موضوع ظاهر خواهد شد که در آن صورت می توانید از موضوع محلی (local) موجود در ماشین (دستگاه) خود استفاده نموده، یا یکی از موضوعات

وب را به دست آورید. با افزودن طرحانه جدید جعبه محاوره ای ظاهر می شود که نوع پروژه را در آن انتخاب می کنید.

پس از تأئید این جعبه محاوره ای راه حل مزبور دو طرحانه (پروژه) را شامل می شود دومین پروژه ساختار دیگری غیر از ساختار اصلی خواهد بود و درون ساختار یک طرحانه C++ اداره شده می باشد، بنابراین حاوی پوشه هایی برای پرونده های مبدأ سرصفحه (header) و منبع است.

در هر دوی این پروژه ها پرونده ای به نام "AsseblyInfo" وجود دارد. پرونده مزبور در همه پروژه هایی که از این چارچوب استفاده می کنند وجود دارد و همانند منبع VS – VERSION – Info (که در Visual C++ استفاده شده است) می باشد. یک پرونده Assembly واحد بسته بندی نرم افزاری اصلی این چارچوب است و شامل یک یا چند ماجول (پیمانه) قابل اجرا و منبع می باشد. همچنین یک یک اظهار نامه (manifest) اسمبلی (همگذاری) را در بر می گیرد. این اظهار نامه فوق داده هایی هستند که به همگذاری مربوط می شوند. شما می توانید با پر کردن محتویات پرونده AsseblyInfo اطلاعاتی نظیر شماره نسخه های مکد علائم تجاری، توضیحات و سایر داده ها را به این اظهار نامه اضافه کنید. بعداً کاربرد مؤلفه اسمبلی ها و نحوه اداره کردن آنها را توضیح داده می شوند.

وابستگی های پروژه

حال که دو پرونده در این راه حل وجود دارد می توانیم نحوه مدیریت وابستگی ها (dependencies) را نشان دهیم. با کلیک کردن راست روی یک پروژه و انتخاب گزینه Project Dependencies از منوی مضمونی (context menu) شما می توانید پروژه هایی که باید قبل از پروژه (طرحانه) انتخاب شده ایجاد شوند را انتخاب کنید. جعبه محاوره ای موجود در شکل ۸-۵-۱ نشان می دهد که AnOtherApp به AnApplication وابسته است و پروژه های مزبور با همان ترتیب نشان داده شده ایجاد خواهند شد. این تصویر هر دو مقطع (Pane) جعبه محاوره ای مزبور را نشان می دهد. به خاطر داشته باشید که تنظیمات نیز از طریق خصوصیات راه حل در دسترس قرار می گیرند.

نمای کلاس

Class View

اگر شما با Visual C++6.2 آشنا می باشید، در مورد کار با زبانه Class View مشکلی نخواهید داشت. این یک نمای کلاس (رده) چند پروژه ایست که

به شما امکان می دهد پروژه های خود را توسط کلاس، عضو (member) و تابع ناویری (navigate) کنید. کلیک کردن روی یک مدخل شما را به محل مورد نظر در ویرایشگر منتقل خواهد کرد. با کلیک راست کردن روی یک کلاس (رده) یک منوی مضمونی ظاهر می شود که دارای مدخل هایی برای افزودن روش ها، خصوصیات، فیلد (میدان) ها و ایندکس (نمایه گذار) ها می باشد. کلاس ها نیز پایه و رابط هایی دارند که برای مرور کردن (browshing)، در دسترس قرار می گیرند. شکل ۹-۵-۱ محتویات مقطع Class View

برای راه حل را نشان میدهد.

نمای منبع

Resurce View

کنترل (ناظم) زبانه ای که شامل Sulalation Explorer (جستجوگر راه حل) و Class View است می تواند Resurce View (نمای منبع) رانیز میزبانی می کند. وقتی شما یک برنامه کاربردی Windows Forms را ایجاد می کنید، این زبانه را مشاهده نخواهید مرد ولی می توانید با انتخاب فرمان View= ResurceView از منوی اصلی یا با فشار دادن کلید های میانبر ctrl + Shift + E آن را صریحاً درخواست نمایید. این مقطع به شما امکان می دهد که پرونده های منبع سبک قدیمی (که از توسعه C++ و MFC آنها را می شناسید) را

ویرایش کنید. توجه داشته باشید که VS . NET، به شما امکان می دهد برنامه های کاربردی ATL و MFC را به موازات پروژه های چارچوب NET، خود ایجاد و ویرایش نمایید.

جستجو گر ماکرو

Macro Splorer

گسترده از دستورهای ماکرو (کلان) در توسعه کارهای روز به روز امکان پذیر است. Visaule studio 7.0 دارای یک الگوی موضوع جامع است که امکان امریه نویسی (scripting) در مورد هر تابع مورد نظر شما را به وجوئد می آورد. جستجوگر ماکرو (Macro Explorer) محلی است که می توانید همه دستورهای ماکرو مربوط به نگارش VS . NET مختص به خود را در آن بیابد.

راهنمای دستی

بین راهنمای (help) موجود در سه زبانه باقیمانده در این مقطع و سیستم Dynamic Help موجود یک تمایز وجود دارد. سیستم Help بسیار خوب می کند و شامل زبانه های Search Index و Contents می باشد.

زبانه Index به شما امکان می دهد یک عبارت ساده را تحریر نمایید و فوراً برای یافتن هماهنگ ترین مورد جستجو می کند. زبانه Search امکان اجرای جستجوی پیچیده تری را نیز فراهم خواهد کرد. برای عبارت های مورد جستجو

می توان مدخل های (entry) سند را استفاده نمود. مثلاً در حین جستجو عبارت "Thead AND bloking" همه سندهایی که حاوی کلمات Theard و Broking هستند، جستجو می شوند. همچنین یک جعبه علامت (check box) وجود دارد که در صورت فعال بودن امکان جستجوی کلمات مشابه را نیز فراهم می سازد. بنابر این در این مثال ممکن است در رابطه با کلمات مورد جستجو کلمات threading و blocked نیز پیدا می شوند. شما می توانید کلمه "Near" را به جای کلمه "AND" استفاده کنید. این امر به شما اطمینان می دهد که کلمات یافته شده در یک گروه ۸ کلمه ای قرار دارند. همچنین می توانید بر نزدیک بودن کلمات کنترل (نظارت) داشته باشید، مثلاً به گونه ای باشند که آنها جزو ۳کلمه از گروه دیگر قرار گیرند. بسیاری از جستجوها ۵۰۰ مدخل را بر می گرداند. برای فیلتر نمودن (پالایش) آن می توانید مجدداً نتایج جستجوی قبلی را بررسی کنید. مرتب سازی مدخل ها هنگام یافتن آنها مفید است. مثلاً جستجویی که ۵۰۰ مدخل را ارائه می دهد ظاهراً سودمند نیست و نتایج مزبور از سراسر پایگاه داده های راهنما به دست می آیند. مرتب سازی این جستجو بر حسب Location (مکان) به شما امکان می دهد که بخش در برگیرنده آن عنوان را به دست آورید، بنابراین اگر "text NEAR box" را جستجو نمایید. به محل ۵۰۰ مدخل دست پیدا می کنید. فقط ۳۰ مورد از آنها در کتابخانه رده چارچوب NET

(NET framework Class Library) وجود دارند. در نتیجه مرتب سازی بر حسب مکان فیلد (میدان) موجود را در حد زیادی محدود میکند.

بخش C : جعبه ابزار و **Server Explorer**

حرکت دادن ماوس روی این نوار ابزار (Toolbar) باعث جمع شدن آن می شود. به طوریکه در این حالت میتوانید به مؤلفه های منفرد دسترسی داشته باشید. جعبه ابزار (Tool box) احیا شده (animated) یک دستگاه ذخیره غیر منقول پرده نمایش می باشد. اگر این قدر خوش شانس هستید که یک مبصر مانیتور (مبصر) ۳۶ اینچی دارید. جایی برای ذخیره تغییرات مورد نیاز نخواهد بود. بنابراین شما می توانید همه کنترل (ناظم) ها را با استفاده از آیکن (شمایل) Pushpin (سوزن سر پهن) به طور ثابت در محل مورد نظر الصاق کنید. این آیکن در گوشه راست- بالای مقطع مزبور قابل رؤیت است. همچنین اگر مانیتور (مبصر) بزرگی در اختیار دارید و مایلید خطوط طولانی از خود را بنویسید می توانید سوزن های سرپهن طرف راست را از حالت الصاق خارج نموده و در این صورت سایر مقطع های سمت راست نظیر Sulittion Explorer و غیره جمع می شوند.

جعبه ابزار

Tool box جعبه ابزار دیگری است که بخش های متعددی را شامل می شود و قبلاً این جعبه ابزار جزو مؤلفه ها بوده است. ویرایش لی آوت (طرح بندی) های برنامه کاربردی Windows Forms افزون فقره (آیتم) ها به جعبه های محاوره ای و نیز قرار دادن سایر منابع به این جعبه ابزار بستگی دارد. (همانند کشیدن مبدأ(drag Surce) برای پالت (لوحه ها) آیتم های موجود هستند. محتویات tool box بسیار پویا می باشد. و طبق کاری که شما انجام می دهید تغییر خواهند کرد. هنگام ویرایش یک فرم ویندوز (Windows Form) لیست FORMS را در حین ویرایش یک نمودار UML و همچنین مؤلفه های UML را مشاهده می کنید.

سرور اکسپلورر

در این جعبه ابزار همراه tool box جستجو گر خادم (Server Explorer) می باشد. که یک ویژگی جدید در VS . NET است. و برخی از ویژگی های پیشرفته (که در حین برنامه نویسی برای خادم های پایگاه داده ها یک ذخیره کننده فوری می باشد) امکان میدهد صف ها خدمات سیستم و ثبت های رخداد سیستم را اداره کنند. Server Explorer مجموعه ای از گروه های (Nodes) درخت

شده همه خادم های متصل به آن را نشان می دهد. شما می توانید مقوله های (categories) مختلف گروهها را انتخاب نماید و آنها را به فرم ویندوز خود بکشید. در این حالت VS . NET مجموعه ای تطبیق گر ها (adapters) را در اختیار شما قرار می دهد تا برنامه شما بتواند به گروه انتخاب شده دسترسی داشته باشد. با استفاده از جستجوگر خادم Server Explorer شما میتوانید یکی از شمارنده های کارآئی را به یک فرم ویندوز بکشید. مثلاً بردن شمارنده کارآیی – Global - از گره CCW های C# به یک فرم مؤلفه Performance Counter (شماره کارآئی) را ایجاد می کند، به طوری که شما می توانید برای اطلاع از تعداد دفعات منظم کردن کد اداره شده و نشده آن را استفاده نمایید.

همچنین با کشیدن پایگاه داده های Access از گره Data Connections به یک برگه یک oleDBConnection برای آن پایگاه داده ها ایجاد می شود. گسترش این گره پایگاه داده ها در Server Explorer به شما امکان می دهد جدول ها رویه های (orocedures) ذخیره شده و نماها (Views) را به آن برگه بکشید و تطبیق های مناسب برای آنها را نیز ایجاد کنید.

شما می توانید یک سرور (خادم) را با استفاده از انتخاب منوی Connect to Server از منوی Tools (از نوار ابزار اصلی) به پنجره جستجوگر اضافه نمایید.

پس از اتصال به این خادم (در صورت مجاز بودن) می توانید به شمارنده های کارآیی و ثبت های سیتسم آن دسترسی داشته باشید.

بخش **D**: تکالیف (**Tasks**)، خروجی (Output)، نتایج جستجو و مشاهدات

در این بخش یک پنجره زبانه دار را می بینید که دارای مقطع های متعددی می باشد. این بخش از پرده نمایش با تکالیف خروجی فرآیند ایجاد اطلاعات ردیابی اشکال زدایی نتیاج جستجوها از طریق مقطع های راهنمای Search و Index و نتایج جستجوهای نماد موجود در برنامه های کاربردی شما سروکار دارد.

تکالیف

برنامه نویسانی که با Visual C++ بیش از چند تصویر محدود اخیر کار می کنند توضیحات (comments) TODO / / (که در کد ایجاد شده توسط جادوگر ظاهر میشوند) را استفاده خواهند کرد. معمولاً این توضیحات به شما می گویند که هنوز چه بخش هایی از پیاده سازی خودکار لازم است کامل شوند. شما فقط باید با یک جستجوی سریع پرونده نشان گذارهای TODO را بیابید و نیز نحوه کامل شدن مرز را مشاهده نمایید.

حال VS . NET فرآیند ردیابی تکالیف (task) را برای شما به طـور خودکـار انجام می دهد. هر وقت شما توضیح (comments) TODO / /را در کد (رمز) خود وارد کنید، IDE یک تکلیف را در سیاهه تکالیف شما قرار می دهد. با دو بار کلیک کردن روی آن تکالیف وارد توضیح موجود در آن کد می شـوید. عـلاوه بـر تگ (ضمیمک) های توکار TODE , HACk و UNDONE شـما مـی توانیـد تگ های سفارشی خود را که در لیست تکلیف نیـز نشـان داده خواهـد شـد) درج نمایید.

برای افزودن ضـمیمک سـایه تکلیـف خـود فرمـان ، Tools = Options را از منوی اصلی و سپس Environment= Task List را از درخت متعلق به گزینه های (options) برپایی (setup) موجود در مقطع سمت چپ این جعبه محـاوره ای انتخاب کنید. به طور پیش فرض، ممکن اسـت ایـن تکـالیف در پنجـره پنهـان شوند. برای مشاهده همه این تکالیف روی مقطعـه تکلیـف (task pane) کلیـک راست نمـوده و از منـوی حاصـله، گزینـه Show tasks و سـپس گزینـه ALL انتخاب کنید.

جزئیات خروجی

با انتخاب زبانه Output ممکن است مقطع های متعددی از اطلاعات به شما نشان داده شوند پیام های ایجاد (شامل هشدارها و خطاها) نیز در همان مقطع ها نمایش داده خواهند شد. پیام های ردیابی های اشکال زدایی (علاوه بر پیام های خاص از سایر برنامه ها) نیز در این منطقه در دسترس می باشند و همانند آنها از طریق خط فرمان و توسعه IDE فراخوانی می شوند.

نتایج نماد FIND

با فشار دادن کلید های F12 + Alt یا انتخاب فرمان Edit → Find and Replace → FindeSymbol جعبه محاوره ای Find Symbol ظاهر خواهند شد. با استفاده از این جعبه محاوره ای می توانید نمادی را در پرونده های برنامه کاربردی خود منابع موجود در پروژه یاهر مؤلفه انتخاب شده دیگر جستجو کنید. این نتایج در مقطع نتایج Find Symbol نمایش داده می شوند. کلیک راست روی هر مدخل موجود در آن مقطع به شما امکان می دهد به توضیح ویرایشگر منتقل شوید یا این که توضیحات و مرجع ها را برای آن نماد مرور کنید. با انتخاب گزینه Browse Definition مقطع Object Browser در پنجره

اصلی ظاهر خواهد شد شما میتوانید دراین پنجره کل درخت سلسله مراتبی موضوع و نیز سایر موارد موجود در اسمبلی (همگذاری) آن را برای موضوع انتخاب شده مشاهده نمایید.

رابط مرورگر موضوع کمی با Browser Info در VS 6.0 فرق دارد. در نگارش قبلی اطلاعات مرور گر باید به طور صریح ایجاد می شد. Object Browser در VS . Net همیشه حتی در قبل از اولین ایجاد در دسترس می باشد و برای امور توسعه عمومی روز به روز شما بسیار سودمند است. این مرورگر متشکل از سه مقطع میباشد که عبارتند از: مقطع Object (در سمت چپ) مقطع Menbers (سمت راست) و مقطع descriptipon (در پایین). به علاوه یک نوار ابزار در بالای این مرورگر قرار دارد که به شما امکان می دهد سایر گزینه های موجود را انتخاب کنید. با استفاده از Object Browser می توانید نامکدهها (فضای نام) کلاس (رده) ها نوع ها (type) و رابط ها، enums (شمارشی ها) و ساختارها (Structures) را جستجو نمایید. همچنین موضوعات به طور اختیاری در Brows Pane مرتب می شوند، این کار با استفاده از نوار ابزار Object Browser (برای مرتب سازی اعضا به ترتیب حروف الفبا) بر حسب درج بر حسب دستیابی به موضوع، یا برحسب درج موضوع گروه بندی می شوند.

ایندکس (نمایه) و نتایج جستجو

این دو مقطع (pane) نتایج جستجو های محتویات راهنما (برحسب درخواست یک نمایه ساده یا از طریق یک جستجوی پیچیده تر) را نمایش می دهند. کلیک کردن روی یک نتیجه شما را به مدخل راهنما منتقل می کند.

پنجره های اشکال زدا

برخی از مقطع های دیگر در این بخش ظاهر می شوند، مقطع های مزبور شامل Break Point و پنجره فرمان فوری هستند. این مقطع ها در ادامه همین فصل و در بخش برنامه های اشکال زدایی توضیح داده خواهند شد.

بخش E : خصوصیات (Properties) ، راهنمای پویا (Dynamic Help) و برگزیدگان (Favorites)

حال بیایید آنچه احتمالاً در IDE به نحو احسن مورد استفاده قرار می گیرد را بررسی کنیم.

مرور گر Propertt

تقریباً همه جنبه های NET از طریق سیستم خصوصیت (Property)، ترکیب بندی ،نظارت و بررسی می شوند. موضوعات NET می توانند مؤلفه (پارامتر) های خود را از طریق صفات خاص نمایش دهند. (این صفات به خصوصیت آنها وابسته هستند). اگر شما با ویژوال بیسیک آشنایی داشته باشید، در مورد کار با Property Grid (تورانه خصوصیت) مشکلی نخواهید داشت. در صورتی که یک برنامه نویس با تجربه C++ هستید، خوگرفتن با سیستم خصوصیت ممکن است به مدت زمان کمی نیاز داشته باشد.

مقطع Property Browser تورانه ای از آیتم های مقداری- نامی را نمایش می دهد که می تواند به طور اختیاری به مقوله هایی گروه بندی شود. برخی از آیتم ها نیز ممکن است خصوصیات فرعی (Sub - Properties) مختص به خود را داشته باشد. با استفاده از شمایل "+" (در سمت چپ) می توان این خصوصیات را توسعه و نمایش داد. همچنین همه مقوله ها را می توان با استفاده از شمایل های "+" یا توسعه داده یا جمع نمایید. (collapse) معمولاً تورانه (grid) خصوصیت فقط با مقادیر ساده (نظیر یک عدد یا یک مقدار رشته) سرو کار دارد. با این وجود مقادیری که به صورت عدد نشان داده می شوند، گاهی اوقات با استفاده از یک شکل دیگر (مثلاً با یک جعبه حاوی مسطوره رنگ) راحت

تر نمایش داده خواهند شد. Property Browser می تواند نوع خاص اطلاعات (مثلاً یک مقدار رنگ بر حسب عدد RGB) را به یک عنصر UI (نظیر مسطوره رنگ موجود در شکل ۱۸-۵-۱) تبدیل کند. علاوه بر این یک مقدار enum (شمارشی) ممکن است نظیر مقدار عددی آن ذخیره شود، ولی Property Browser می تواند مقادیر عددی واقعی را نشان دهد.

Property Browser، اطلاعات را فقط با یک روش یادآور (mnemonic) یا دوستانه ای کاربر (user - friensdly) نمایش دهد. همچنین این نوار ابزار به شما امکان می دهد که آن اطلاعات را به یک روش سودمند ویرایش کنید. مسطوره (swatch) های رنگ (شکل ۱۸-۵-۱) می توانند با استفاده از یک انتخابگر (picker) رنگ اختصاصی تغییر داده شوند (این انتخابگر زمانی گشوده می وشد که شما بخواهید یک مقدار را تغییر دهید) در شکل ۱۹-۵-۱، یک ویرایشگر Property Browser مربوط به مقادیر رنگ را نشان میدهد.

چنیدین ویرایشگر های UI برای خصوصیات موضوعات این چارچوب وجود دارند. خصوصیات Anchor (لنگر) و Dock (مقرّ) متعلق به یک ناظم برگه ویندوز دارای ویراشگرهایی با رابط های شهودی (Intuitive) و نگاره ای (grafical) می باشد. نوار ابزار Property Browser که در بالای Property Browser اصلی قرار دارد، برای نمایش انواع مختلف اطلاعات یا ترتیب داده ها

با روشی دیگر استفاده میشود. بسته به محتوای خصوصیاتی که شما می بینید برخی از دکمه های زیر را مشاهده خواهید کرد:

Categrized - یک نمای مرور گر خصوصیت راانتخاب می کند این نما آیتم (فقره) های گروه بندی شده به مقوله ها را نشان می دهد . (نظیر Appearance یا Bihavior).

Alfabetic – همه خصوصیات را صرفنظر از مقوله آن به ترتیب الفبا مرتب خواهد کرد.

Events- همه خصوصیات رخداد یک مؤلفه را لیست (سیاهه) نموده و به شما امکان میدهد تا یک رخدادگردان برای آنها ایجاد کنید). رخدادها در چارچوب موجود در محل نگاشتهای پیام استفاده می شوند. برای ایجاد یا ویرایش یک اداره کننده رخدادی خاص روی نام آن رخداد رد سیاهه کلیک نمایید. Ide به شما امکان می دهد که نامی را برای اداره کننده خود درج نموده یا یکی از اداره کننده های موجود در لیست پایین کشیدنی را انتخاب کنید. پس از ایجاد یا انتخاب این اداره کننده می توانید روی ان دو بار کلیک نمایید تا به کد مورد نظر برسید. اگر می خواهید اداره کننده ای با یک نام از پیش انتخاب شده ایجاد کنید فقط باید روی رخدادی که فاقد اداره کننده است دو بار کلیک کنید و در این حالت IDE

اداره کننده ای را با یک نام نشان دهنده موضوع و نیز یک تابع تعیین شده برای آنها ایجاد خواهد کرد.

Massages - وقتی مقطع Class view یک برنامه کاربردی C++ Windows را نمایش می دهد. این دکمه درنوار ابزار ظاهر خواهد شد. شما می توانید اداره کننده های پیام را نمایش دهید. آنها را ویرایش نموده و از طریق Property Browser به کلاس (رده) های C++ خود اضافه کنید. روش افزون اداره کننده های پیام به کلاس های C++ خود همانند روشی است که قبلاً درمورد Event Editor بیان شد.

Overrides – وقتی Class view کلاسهای C++ را نمایش می دهد شما می توانید با کلیک کردن روی دکمه Overrides (در نوار ابزار Property Browser) لغوهای یک رده انتخاب شده در نمای کلاس را مشاهده نموده وهمچنین ویرایش را اضافه کنید.

Property Pages – وقتی با استفاده از Solution Explorer پروژه یا ترکیب بندی های راه حل را بررسی می کند. نوار ابزار Property Browser این دکمه را نشان می دهد. از این دکمه استفاده نمایید تا یک جعبه محاوره ای که دارای خصوصیت لازم برای آن ترکیب بندی می باشد، ظاهر می شود.

Properties- اگر برای نمایش رخدادها یا سایر اطلاعات از نوار ابزار Property Browser استفاده می کنید. می توانید از طریق این دکمه به نمای معمولی خصوصیت بازگردید.

راهنمای پویا

سیستم جدید راهنمای پویا (dynamic help) در VS . NET برای یک برنامه نویس پر مشغله مهم است. در این صورت اغلب مواقع شما در حین برنامه نویسی به جستجوی پایگاه داده ها نیاز دارید. زیرا این سیستم یک سیاهه اطلاعاتی بهنگام شده (مربوط به آنچه همیشه درج می کنید) را شامل می شود. سیستم مزبور حتی دارای اطلاعاتی در مورد پنجره های مورد استفاده شما در IDE می باشد. مثلاً اگر نخست زبانه (Dynamic Help) در بخش E سپس مقطع Class View را انتخاب نمایید. یک فقره (آیتم) راهنا راجب استفاده از مقطع Class view در دسترس قرار می گیرد.

با این وجود تجربه نشان داده است Dynamic Help به نیروی پردازشی زیادی نیاز دارد. زیرا با هر ضربه کلید (keystroke) داده ها جستجو و تلفیق (collate) می شوند. برای غیر فعال کردن Dynamic Help یک گزینه وجود دارد. در صورت کم بودن سرعت دستگاه شما در حین استفاده از VS . NET ممکن است بخواهید این کار را انجام دهید.

پنجره **Favorites**

سیاهه Interner Favorites از طریق این بخش از IDE در دسترس می باشند. انتخاب یک پسوند برگزیده (favorite) آن پیوند را در رابط مرورگر موجود در مقطع اصلی ویرایش نمایش خواهد داد.

برنامه های اشکال زدایی

حتی بزرگترین برنامه نویسان هم مرتکب اشتباه می شوند بنابراین یک اشکال زدای (debagger) خوب از اهمیت زیادی برخوردار است. اشکال زدایی در مورد Visual Studio کار بسیار راحتی است. و محدوده وسیعی از گزینه های متوقف کننده برنامه ها و امتحان رمز را در اختیار شما قرار می دهد. ساده ترین روش این است که رمز را در حین ایجاد آن اشکال زدایی کنید. همچنین اشکال زدا می تواند وضعیت یک خطای اداره نشده را وقفه نموده و خود را به این فرآیند خطادار ضمیمه نماید با این که شما می توانید به طور صریح آن اشکال زدا را به یک فرآیند ضمیمه کنید. بیایید اولین حالت ساده را بررسی نماییم.

برای اشکال زدایی برنامه حالت Debug (mode) باید در این برنامه ایجاد شده باشد. با استفاده از قابلیت انتخاب مدیر ترکیب بندی (Configiguration Manager) پروژه در منوی اصلی Debug... حالت Debug را انتخاب کنید.

پس از انتخاب ترکیب بندی اشکلا زدا، آمادهاید تا محل توقف برنامه را انتخاب کنید. این امر با تنظیم یک نقطه قطع انجام می شود و شما می توانید با انتخاب خط رمزی که می خواهید متوقف گردد و سپس فشار دادن کلید F9 این کار را انجام دهید. یک نشانه قرمز رنگ در حاشیه ویرایشگر کد (رمز) ظاهر خواهد شد که نشان می دهد نقطه توقف (breakpoint) فعال است. همچنین می توانید باکلیک کردن روی حاشیه خاکستری سمت چپ پنجره ویرایش یا با فشار دادن کلیدهای CTRL + B یک نقطه توقف در محل مورد نظر ایجاد کنید.

در قسمت قبلی همین فصل به مقطع breakpoint که نزدیک پنجره کاری قرار دارد اشاره کردیم. برای فعال کردن این مقطع فرمان

Debug → Windows → Breakpoint

انتخاب نماییم یا کلیدهای Ctrl + Alt +B را فشار دهید.

الفبای تصویری نقطه توقف

دایره توخالی قرمز رنگ، یک نقطه توقف (breakpoint) غیر فعال است و می تواند با مقطع نقاط توقف فعال یا غیر فعال شود. برای انجام این کار فرمان Debug → Windows → Breakpoint را انتخاب کنید یا کلیدهای Ctrl + Alt +B را فشار دهید.

نقطه توقفی که یک علامت سوال در آن قرار دارد ممکن است عمل نکند زیرا فرایند مربوط به آن اجرا نمی شود. مثلاً اگر یک نقطه توقف را در یک برنامه کاربردی قرار دهید و سپس برنامه کاربردی دیگر را از طریق همان راه حل اجرا کنید، این امر اتفاق می افتد. به محض اینکه فرآیندی که دارای نقطه توقف بالقوه ای می باشد بارگذاری شود، این نقطه توقف فعال خواهد شد.

یک نقطه توقف که دارای یک علامت تعجب می باشد، به دلیل خطا نمی تواند تنظیم شود احتمالاً مکان نقطه توقف نامعتبر است با وجود آن نقطه توقف تحت شرطی می باشد که مورد نظر شما نیست.

یک نقطه توقف همراه بایک نقطه در یک صفحه ASP تنظیم می شود و آن توقف به صفحه HTML متناظر (که توسط ASP ایجاد شده است) نگاشت می گردد.

پس از تنظیم موفقیت آمیز یک می تواند با وجود عمل کردن این توقف اجرا شود. کلید F5 را برای ایجاد رمز موجود در حالت اشکال زدایی فشار دهید.

تنظیمات نقطه توقف پیشرفته

گاهی اوقات شما به بیش از یک مکان برای توقف نیاز دارید بنابر این نقاط توقف موجود در VS . NET نیز می توانند طوری تنظیم شوند که پس از چندین عبور از نقطه توقف یا هنگام مواجه شدن با یک شرط روی دهند. برای ایجاد یک نقطه توقف جدید یک خط را رمز را انتخاب نموده و کلید F9 یا کلید های Ctrl + B مستلزم هستند که شما این اطلاعات را به طور دستی پر نمایید. وقتی از کلیدهای Ctrl + B استفاده می کنید جعبه محاوره ای New Break Point را مشاهده خواهید کرد.

در این جعبه محاوره ای دو دکمه وجود دارند که عبارتند از Hit Count و Condition. برای ویرایش یک نقطه توقف که قبلاً تنظیم شده نقطه توقف موجود در مقطع BreakPoints را پیدا کنید و روی آن کلیک راست کنید. این عمل به شما امکان می دهد Properties را از یک منوی مضمونی (جعبه ویرایش BreakPoints را ظاهر می کند و گزینه های مشابهی دارد) انتخاب نمایید.

نقطه های توقف شرطی

انتخاب این گزینه به شما امکان می دهد که وقتی می خواهید بر اساس یک شرط برنامه ای را متوقف کنید یا شرط دیگری برای آن بگذارید تصمیم گیری درستی داشته باشید. مثلاً ممکن است بخواهید وقتی توقف تغییری داشته باشد) آن برنامه را متوقف کنید.

Hit Couns (شمارش دفعات)

این گزینه به نقطه توقف امکان می دهد تا زمانی که به تعداد دفعات معین انجام نشده است به طور معمول اجرا شود. شما می توانید آن را در حالت های Break Always (همیشه متوقف شود) ، Break When Hit – Count is Equal to a Specific Value (وقتی متوقف شود شمارنده مساوی با مقدار خاصی باشد. Break When Hit – Count is a Multiple of Some Number (وقتی متوقف می شود که شمارنده مضربی از عدد مورد نظر باشد) یا در نهایت Break When Hit – Count is Greater Than or aqual To a value (وقتی توقف کند که بزرگتر یا مساوی مقدار مورد نظر باشد) تنظیم کنید.

ترکیب این گزینه ها امکان پذیر است، به طوری که می توانید اعلام کنید که می خواهید این برنامه در پانزدهمین عبور از نقطه توقف قطع شود(به شرط آنکه یک متغیر بزرگتر یا مساوی چهاردهمین عبور باشد).

هنگام مکث رمز، چه باید کرد؟

برخی از گزینه های قدیمی و آشنا در حین مکث (pause) که در دسترس برنامه نویس قرار می گیرند. شما می توانید متغیرها را امتحان نموده، خطوط رمز را یک به یک اجرا کرده و حتی فرمانها را از طریق پنجره فرمان مستقیماً به موضوعات موجود در برنامه کاربردی ارسال نمایید.

ویژگی که برنامه نویسان ویژوال بیسیک (احتمالاً نه برنامه نویسان C++) با آن آشنا هستند، پنجره فرمان (command) است. این مقطع که در زمان اشکال زدایی در سمت راست پایین نوار زبانه قرار دارد، به شما این امکان را می دهد فرمان ها را مستقیماً به موضوعاتی که در اشکال زدا متوقف شده اند، بنویسید. مثلاً در مورد زمان درج عبارت زیر وجود دارد که در نتیجه زمان سنج غیر فعال خواهد شد.

This . timeral . Enablel = flase < enter >

اگر کلید F5، برای اجرای مجدد برنامه از نقطه توقف فشار داده شود نقطه توقف موجود در آن زمان سنج دیگر عمل نمی کند. با استفاده از پنجره فرمان، شما می توانید به راحتی محتویات با ترکیب بندی موضوعات موجود در رمز را تغییر دهید تا سناریوهایی که در برطرف کردن اشکالات به شما کمک می کنند امتحان شوید.

ضمیمه شدن اشکال زدا به یک فرآیند

اگر مجموعه ای از برنامه های کاربردی را از طریق یک راه حل یا احتمالاً از طریق اشکال زدایی یک DLL، اشکال زدایی می کنید، این ویژگی سودمند خواهد بود. شما می توانید این برنامه کاربردی را اجرا نموده، اشکال زدایی را شروع کنید و با متوقف کردن برنامه کاربردی اجرای آن را قطع نمایید. نوع برنامه کاربردی که می خواهید اشکال زدایی کنید از شما سوال می شود. انتخاب های متداول برای آن شامل: یک برنامه کاربردی مبتنی بر CLR (CLR – based app) کی T – SQL میکروسافت (Microsoft T- SQL) یک برنامه کد (رمز) اصلی احتمالاً C++ و یک اسکریپت (امریه) می باشند. ضمیمه شدن یه یک فرآیند دیگر می تواند در یک ماشین (دستگاه) دوردست انجام شود (البته اگر شما اجازه دستیابی به را داشته باشید).

اشکال زدایی **JIT**

اشکال زدایی درجا (JIT) زمانی رخ می دهد که قسمتی از رمز یک استثنای اداره نشده را ایجاد کند و به یک اشکال زدا ضمیمه نشود. هنگامی که این مورد اتفاق می افتد گزینه های متعددی در اختیار شما قرار می گیرند. یک جعبه محاوره ای به شما نشان داده می شود شود که امکان انتخاب نوع اشکال زدایی مورد استفاده را برای شما فراهم سازد. توجه داشته باشید که حتی اگر Visual Studio را نصب نکرده باشید، باز هم گزینه های اشکال زدا را خواهید داشت. در صورتی که Visual Studio نصب کرده باشیم بهترین راه این است که این اشکال زدا را به عنوان اشکال زدای خود انتخاب کنید. زیرا ویژگی هایی که VS . NET دارد، پیشرفته تر از سایر انتخاب ها می باشد.

در این جا می بینید که این گزینه ها باید یکی از اشکال زداهای ذکر شده را استفاده کنند:

اشکال زدایی که در حال اجرا می باشد (این اشکال زدا، برای ویرایش AnApplication که در همین فصل به کار رفته است مورد استفاده قرار می گیرد) اشکال زدای CLR (که یک اشکال زدای خوداتکا) می باشد) با یکی از نمونه های جدید Visual Studio . NET با انتخاب یک اشکال زدا و کلیک

کردن روی دگمه Yes، اشکال زدای انتخاب شده اجرا خواهد شد، این اشکال زدا در جایی که به خطا برسد متوقف می شود، باید بتوانید رمز برنامه را مشاهده نموده و آن را اجرا کنید تا خطای موجود پیدا شود. اگر این برنامه کاربردی خطادار با یک ترکیب بندی ترخیصی ایجاد شود شاید، JIT زیاد سودمند نباشد، زیرا احتمالاً رمز همگذاری X80 برای اشکال زدایی به شما نشان دهده خواهد شد.

برنامه کاربردی نمونه فرمهای ویندوز (**Scribble . NET**)

در این فصل به تعداد بیشتری از خصوصیات فرم های ویندوز توجه می کنیم که لازم است شما هنگام نوشتن برنامه های کاربردی برای چارچوب NET، آنها را بدانید.

به خصوص در این جا با منابع شامل متون تصاویر نگاشت بیتی (bitmap) و آیکون (شمایل) ها سرو کار خواهیم داشت. همچنین در مورد سراسری سازی (globazition) یعنی فرآیند درخواست دوستانه یک حضار (audence) بین المللی بررسی کوتاهی خواهیم کرد.

منابع NET

برخلاف Win32 و MFC برنامه های کاربردی NET جهت محاوره و لی آوت (طرح بندی) فرم، زیاد وابسته به منابع نیستند. به هر حال منابع برای محلی نمودن (localization) برنامه های کاربردی، یکپارچه سازی تصاویر و آیکون (شمایل) ها و برای داده های سفارشی (Custom data) (که می توانند به راحتی و بدون نیاز به کامپایل مجدد برنامه کاربردی اصلاح شوند) مورد نیاز می باشند.

مدل (الگوی) منابع تحت NET، یک منبع پیش فرض برای یک برنامه کاربردی و یک مجموعه اختیاری از منابع می باشد که محلی سازی، یا بین المللی ساز کامل تر و همچنین داده هایی برای یک فرهنگ خاص را فراهم می کند. مثلاً یک برنامه کاربردی ممکن است یک فرهنگ منبع پیش فرض امریکایی داشته باشد ولی لازم باشد که برای انگلیس بریتانیا، که (نمای واحد پول تغییر می کند) یا برای فرانسوی (که رشته های متن وابسته به منوها و جعبه های محاوره ای به زبان صحیح باشد) محلی ساز (localization) شود.

منبع پیش فرض معمولاً شامل اسمبلی (همگذاری) می باشد که کد برنامه کاربردی را در بر دارد. سایر اسمبلی ها که همگذاری های ماهواره ای (satallite assemblies) نامیده می شوند، منابعی دارند که می توانند در صورت لزوم

بارگذاری (load) و استفاده شوند. این رویکرد شبیه به ایده یک برنامه کاربردی است که یک DLL منبعی تحت Win32 یا MFC را استفاده می کند.

یک برنامه کاربردی تابعیت سراسری global citizen خوب باید همیشه قابل محلی شدن باشد و یک منبع برای هر فرهنگ مورد استفاده آن نرم افزار ایجاد کند. این هم درستی می باشد که هر برنامه کاربردی لازم نیست که تابعیت سراسری باشد. بنابراین شما باید سطح فرهنگ رابطه دوستانه را بر اساس نیازمندی های توزیع و حضار مورد هدف انتخاب کنید.

پیچ و مهره های محلی سازی

این تمرین خوبی است که برنامه کاربردی خود را با محلی سازی در ذهن خود بسازیم. مهمترین کاری که می توانید تا برای انجام تمرین مزبور آماده شوید این است که در کد خود هیچ رشته ای را برای مشاهده کاربر به کار نبرید. مثلاً هرگز از عبارت زیر استفاده نکنید:

Massage Box . Show (“Hello world”)

زیرا رشته Hello world در کد مبدأ (source code) نشانده شده است. برای تبدیل این برنامه به یک تابعیت سراسری خوب یک منبع (که رشته توسط یک مقدار تعریف می شود) ایجاد می کند. مثلاً:

Massage Box . Show ((string) resources . Get object (“My Hello world resource string.

در این خط متن واقعی پیغام در کد قرار داده نشده است. بلکه از یک منبع که رشته ای را به شکل: (“My Hello world resouerce") تعریف می کند بازیابی می شود.

اگر برنامه کاربردی لازم باشد که به زبان فرانسه استفاده شود یک اسمبلی ماهواره ای در مورد منابع فرانسوی رشته ای با همان “My Hello world resouerce" ID خواهد بود، به جز ایم که شامل مقدار “Bonjour Le Monde" می باشد.

کلاس های مدیریت منابع **NET**

چارچوب NET. یک خانواده از کلاس (رده) ها را ارائه می دهد که برای مدیریت منابع مدیریتی پیوسته و آسان طراحی شده اند. این کلاسها به صورت زیر می باشند.

- ResouerceManager (مدیر منابع) برای مدریت همه منابع موجود در یک برنامه کابردی به کار می رود.

- Resouerceset نشان دهنده مجموعه ای از همه منابع برای یک فرهنگ بخصوص است.
- Resouerce Reader یک کلاس پیاده سازی برای رابط IresouercReader است. این کلاس برای خواندن منابع از یک مسیل (steam) به کار می رود.
- ResouercWriter یک کلاس پیاده سازی برای رابط I ResouercWriter است که برای نوشتن منابع در یک مسیل (stream) به کار می رود.

برنامه کاربردی شما باید از یک ResouercManager برای بارگذاری اسمبلی (همگذاری) ها و قابل دسترس نمودن آنها برای بقیه برنامه استفاده کند. ResouercManager یک Resouercset را برای فرهنگ تعریف شده بر میگرداند.

به دست آوردن فرهنگ مورد نظر

فرهنگ ها و زبانهای مختلف از طریق کلاس(رده) Culltureinfo مشخص می گردند. این کلاس یک انبار ساده اطلاعاتی است. که شامل داده هایی در مورد فرهنگ انتخاب شده توسط کاربر (پس از نصب سیستم عامل)

می باشد. کلاس مزبور شامل اطلاعاتی در مورد زبان تقویم، واحد پول و سایر پیش گزیده هایی (Preferences) می باشد که ممکن است کاربر انتخاب کرده باشد.

نرم افزار شما به این اطلاعات دسترسی خواهد داشت بنابراین می تواند تعیین کند که آیا توانایی ارائه منابع خاص یک فرهنگ را دارد یا باید به مقدار پیش فرض خنثی را انتخاب نمود. داشتن فرهنگ خاص ممکن است بر پیغامی که شما سعی درانتقال آن دارید خیلی مهم باشد. حکایت زیر علت این امر را به شما نشان میدهد.

اطلاعات فرهنگی در مورد زبان ها توسط یک کد دوحرفی و یک شناسه منطقه ای اختیاری ارائه می شود که همراه باکد دوحرفی می باشد. مثلاً زبان انگلیسی عادی با کد دوحرفی EN نشان داده می شود. ولی گویشهای مختلف انگلیسی از انگلیس کانادا ایالات متحده و سایر مناطق با کد فرعی US و CA و GB و غیره شناسایی می شوند.

ایجاد منافع متنی

روشهای زیادی جهت ایجاد منافع متنی برای برنامه های شما وجود دارد. آسانترین روش از طریق ایجاد فایل متنی دارای مجموعه ای از جفت هاین نام-

مقدار می باشد. این روش یک نامی را تعریف می کند که درکد خود می توانید از آن استفاده نمایید. تا منبع و متن مرتبط با آن نام را پیدا کنید.

وقتی ایجاد پرونده منبع رشته ای پایه تمام شد با استفاده از . RESGEN EXE ابزار NET آن را به یک پرونده RESX یا Resouerc تبدیل کنید. مثال بعدی این فرایند را نشان می دهد.

متن منبع رشته ای

پرونده متن ساده، مجموعه ای از جفتهای نام- مقدار برای منبعی که جهت ایجاد رشته ای پیش فرض برای یک برنامه کاربردی استفاده می شود را نشان میدهد. توجه داشته باشید که علامت های گوئیشن (نقل قول) در رشته ها مورد نیاز نیستند. اگر از آنها استفاده کنید در منبع رشته ای قرار می گیرند.

پرونده متنی می تواند به یکی از این دو شکل تبدیل شود:

یکی اینکه مبتنی بر XML که به عنوان پرونده RESX ذخیره می شود. دیگر اینکه مستقیماً به کامپایل شده آن تبدیل شود.

Default cultro resources

WindowTaxt = Internationalization example

Label taxt = Hello World !!!

پرونده resx یک مجموعه داده هـای XML اسـت کـه بـرای سـریال سـازی اطلاعات منبع در طی توسعه استفاده می شود. تبدیل منابع از شکلی بـه شـکل دیگر با استفاده از برنامه سودمند Resgen انجام می شود. متن منبع ساده قبلـی را درج نموده و به عنوان firstResouerc. Txt ذخیره نماییـد و سـپس یـک پرونده RESX با استفاده از خط فرمان زیر ایجاد کنید:

reagan firstResouerc. Txt firstResouerc. Resx

این برنامه کمکی پرونده را کامپایل (همگردانی) نموده و به شما می گوید به دو منبع تبدیل شده اند.

استفاده از **Visual Studio . NET** برنامه بین المللی کردن Visual Studio . NET منابع را مدیریت نموده و به شما کمک می کند تـا فرم ها و مؤلفه هـای (components) محلـی را ایجـاد کنیـد. شـکل ۳-٤-۳ خاصیت قابلین محلی شدن (Incalizable) را برای تنظیم یک فرم نشـان مـی دهد. قرار دادن مقدار true در این خاصیت می تواند سبب طرح محـیط زمـانی جهت ذخیره تمام اطلاعات مربوط در پرونده منبع شود.

روش IntializeComponent مختص فرم ۱، حالا شامل کدی می باشد که مقدار دهی اولیه یک مدیر منبع و بازیابی هر نوع اطلاعات (که ممکن است در حین انتقال برنامه کاربردی بین محل ها از منابع تغییر کد) را انجام دهد.

همان گونه که در مثال شکل ۲-٤-۳ مشاهده کردید گستردگی فیزیکی رشته های متن می توانند هنگام انتقال تغییر یابند. بنابراین متن محل و اندازه اطلاعات ذخیره شده در منابع را پیدا می کنید.

شما می توانید مشاهده کنید که IDL یک مدیر منبع جدید در خط ۳ ایجاد کرده و خطوط ٦ و ۷ متن و اطلاعات اندازه کلاینت (مشتری) را از منابع بازیابی می کنند.

وقتی یک فرم به این صورت قابلیت محلی پیدا کرده می کند، تمامی مؤلفه هایی (components) که شما در آن قرار می دهید، متن، اندازه و اطلاعات موقعیت آنها را در یک پرونده منبع ذخیره می کنند.

منابع تصویری

تا این جا ما با منابع رشته ای سرو کار داشته ایم ولی برنامه های کاربردی NET همانند برنامه های Win32 تصاویر را در منابع برای شمایل (آیکون) ها پس زمینه ها و سایر چیزها ذخیره می کنند. نهایتاً تصاویر در فرم کامپایل

(همگردانی) شدهResource ذخیره می شوند، ولی وقتی شما آنها را در **Visual Studio** یا به طور دستی ایجاد می کنید، آنها را به فرمResx مبتنی بر XML تبدیل می شوند. بدیهی است در فرم XML ویرایش یک پرونده تصویری به صورت متن به هیچ وجه کار راحتی نمی باشد. روش تعریف شده برای جاگذاری تصاویر در منابع با ابزاری مانند VS . NET توسط میکروسافت ارائه شد.

مثلاً یک مؤلفه ممکن است تصویری را به عنوان یک پس زمینه شامل باشد. شما این مولفه را در فرم قرار داده و خصوصیت شکل پس زمینه مربوط به این مؤلفه را ویرایش می کنید. این ویرایشگر به شما امکان خواهد داد که یک تصویر را انتخاب کنید. سپس تصویر مزبور در یک منبع قرار می گیرد.

استفاده از لیست های تصویری (**Tmage Lists**)

برخی مؤلفه ها برای رابط کاربر خود نیاز به تصاویر دارند. کلاس (رده) های Tree view و Listview می توانند برای بالا بردن کیفیت نمایش خود از تصاویر استفاده کنند. لیست (سیاهه) های تصویری مجموعه هایی از تصاویر نگاشت بینی (bitmam) که در یک پرونده منفرد نگهداری می شوند) می باشند. این تصاویر می توانند انتخاب کلیدی پس زمینه ترانسپارنت (فرانما) از

یک رنگ انتخابی داشته باشند، بنابراین فقط بخش انتخاب شده از تصویر نشان داده شده می شود.

همان طور که پیش می رویم کد(رمز) نمایشی برای این بخش به وجود خواهد آمد. با استفاده از Visual Studio یک برنامه کاربردی جدید C# به وجود می آید. و image List demo نامیده می شود. برای تکمیل این کار بقیه اصل فصل را دنبال کنید.

برای ایجاد یک لیست تصویری توسط Visual Studio ابتدا باید یک مؤلفه ImageList را از جعبه ابزار به درون فرم بکشید (drag). حال تصاویر را ایجاد کنید. برنامه MS Paint برای انجام این عمل مناسب می باشند، فقط به خاطر داشته باشید که تصاویر را در یک دایرکتوری (فهرست) مشترک ذخیره نمایید. شکل ۵-٤-۳ یک آیکون (شمایل) ١٦*١٦ را که در MS Paint ویرایش شده، نشان می دهد.

پس زمینه آیکون (شمایل) با رنگ سرخابی پوشانده می شود. هیچ کجا این رنگ در تصویر آیکون واقعی ظاهر نمی شود، بنابراین استفاده از آن به عنوان کلید ترانسپارنت (فرانما) اشکالی ایجاد نمی کند. برای اضافه کردن یک مجموعه از تصاویر به موضوع MS Paint می توانید مجموعه Images را در خصوصیات موضوع انتخاب نموده و روی دکمه Browse کلیک نمایید. سپس

یک کادر محاوره ای همانند ٦-٤-٣ ظاهر می شود. این جعبه محاوره ای می تواند جهت افزون تصاویر به اندازه دلخواه به لیست مزبور به کار برده می شود. توجه کنید که حتماً همه تصاویر را در یک لیست (سیاهه) تصویری و یک اندازه (در این جا ١٦*١٦ است) نگهداری نمایید.

شما از جعبه محاوره ای ویرایشگر مجموعه تصویر می توانید مشاهده کنید که چهار تصویر مزبور از ۰ تا ۳۰ شماره گذاری شده اند و هر کدام خصوصیاتی دارند که توسط ویرایشگر قابل مشاهده است.

حال برای استفاده از این تصاویر درموضوع List view از جعبه ابزار مزبور یک موضوع List view به درون فرم بکشید. در حال حاضر ما به راحتی از List view برای نمایش آیکن های کوچک استفاده کرده ایم. بنابراین موضوع List view را انتخاب کرده و خصوصیات آن را نمایش دهید. خصوصیتی که SmallImagelists نامیده می شود حال می توان برای استفاده از Imagelists (که ما ایجاد کرده ایم) تنظیم شود.

خطوط ٤ و ٥ EesuurceManager (مدیریت منابع) را برای فرم ایجاد کردند. خط ١٢ تا ١٦ List view را به وجود می آورند و خصوصیت SmallImagelists را برای استفاده مؤلفه ImageList1 برپا (setup) می کنند. خطوط ٢٠ تا ٢٤ لیست تصویری را با گرفتن مقادیر از جریان دهنده

(streamer) لیست تصویری مقدار دهی اولیه نموده و ترانسپارنت (فرانمایی) رنگ و اندازه تصویر را تنظیم می کنند.

برای استفاده از منابع در List view خصوصیات (Properties) موضوع List view را انتخاب کنید. سپس خصوصیت مجموعه ای (Collection) Items را انتخاب نمایید و در نهایت روی دکمه ای که با سه نقطه (...) نشانه گذاری شده کلیک کنید، این عمل موجب نمایش یک جعبه محاوره ای (شکل ۷-٤-۳) می شود که به شما امکان می دهد عناصر را در جعبه سیاهه (list box) اضافه و ویرایش نمایید.

توجه:

توجه کنید که برنامه های شما همچنین می توانند آیتم را به طور پویا به مؤلفه ها اضافه نمایند. Vasual Studio برای انجام این کارها یک روش RAD دیگری ارائه می دهد.

به هر یک از عناصری که به درخت (tree) اضافه می شوند می توان متن و یک تصویر برای نمایش اختصاص داد. همچنین List view دخدادهایی شبیه به آنچه در فصل ۲-۳ مؤلفه های رابط کاربر نشان داده شد (وقت آیتم (فقره) های لیست انتخاب یا ویرایش و غیره می شوند)، ایجاد می کند. اداره کردن این

رخداد به شما امکان می دهد تا با لیست مزبور فعل و انفعال (intraction) داشته باشید.

حال خطوط ٦ تا ١٣ موضوعات List view را به وجود می آورند و متن و فاصله تصویر را در Image List مورد استفاده تنظیم می کنند. خطوط ٣٢ تا ٣٦ آیتم (فقره) ها را به List view اضافه می کنند.

دسترسی برنامه ای به منابع

علاوه بر ایجاد منابع با Vasual Studio یا سایر ابزار ارائه شده شما می توانید منابع را به راحتی از طریق کد (رمز) ایجد مدیریت و استفاده نمایید. یک نمونه کاربرد منابع ذخیره برخی از داده های سفارشی (نظیر ، اندازه پنجره ها و موقعیت ها) برای برنامه کاربردی خود می باشد تا وقتی برنامه دوباره اجرا می شود بازیابی گردد.

خواندن و نوشتن منابع با کلاس (رده) های ResourceReade و ResourceWriter اجرا می شوند. این موضوعات به شما امکان می دهند تا با منابع ذخیره شده در مسیل ها (streams) یا پرونده ها سر و کار داشته باشیم.

در مثال بعدی ما یک برنامه کاربردی ساده Windows Forms را که یک توپ قرمز و زرد را در سطح فرم نمایش می دهد، آماده کرده ایم، شما می توانید این توپ ها را با ماوس بلند کرده و در اطراف جابجا کنید. هنگام بسته شدن برنامه کاربردی یک منبع به نام ball Locations . Resource را ایجاد کرده و می نویسد. این منبع موقعیت های نمایش دو توپ را ذخیره می کند، بنابراین دفعه بعد که بارگذاری (load) می شود، برنامه کاربردی مزبور توپ ها را در همان جایی که قبلاً گذاشته بودید قرار میدهد.

به عنوان یک مزیت این برنامه اداره کردن ماوس (mousehandling) و استفاده از Imag List برای ترسیم تصاویر روی فرم را نشان می دهد.

نوشتن و خواندن پرونده های RESX XML

پرونده های منبع (همان طور که در ابتدای فصل اشاره کردیم) به شکل های مختلف در می آیند که عبارتند از: پرونده های Resource و پرونده های res.... پرونده های Resx خام، XML کامپایل نشده هستند که به عنوان فرم مبدأ اصلی برای منابع استفاده می شوند.

در مثال قبلی ما کلاس (رده) های ResourceReade و ResourceWriter را نشان دادیم. آنها مطابق کلاس های ResourceReade و ResourceWriter هستند که می توانند برای نگهداری فرم های XML منابع استفاده شوند.

در مثال قبل این فصل ما برنامه کاربردی ساده ای ایجاد کردیم که به شما امکان می دهد تا پرونده های نگاشت بیتی (bitmap) متن و آیکون (شمایل) انتخاب شده را بارگذاری کرده و آنها را به پرونده RESX تبدیل نمایید.

خلاصه

در این فصل مشاهده کردید که نحوه ایجاد منابع متنی به طور دستی چیست؟ چگونه برنامه های کاربردی با استفاده از اسمبلی های ماهواره ای بین المللی شدند و همچنین چطور از Visual Studio برای ایجاد برنامه های کاربردی با تصاویر و سایر منابع استفاده کنیم و چگونه پرونده های منابع را در برنامه های خود به طور پویا ایجاد و استفاده نماییم. در فصل بعد گزینه های گرافیکی (نگاره ای) قدرتمند قابل دسترسی را همراه با GDI+ بررسی می کنیم.

وب جدید

در طی سالهای گذشته گرایش عجیب برنامه های کاربردی رومیزی قدیمی به برنامه های کابردی مبتنی بر وب مطلب مهمی بوده است. توانایی ایجاد و آماده سازی برنامه های کابردی مبتنی بر وب (که باعث شدند کاربر همان تجربه های ارزشمند را در برنامه های کاربردی رومیزی به دست آورد) هرگز یک کار ساده نبوده است. همان طور که فناوری های جدید نظیر پشتیبانی امریه نویسی (scripting) پیشرفته ایجاد می شوند توانایی ایجاد یک UL مبتنی بر وب قوی نیز به دست می آید.

کاربران برنامه کاربردی نه تنها انتظار برنامه کاربردی تابعی (functional) را دارند بلکه این برنامه کابردی خود باید از نظر زیاشناسی رضایت بخش نیز باشند.

علت اصلی برای سوئیچ (دگرگزینی کردن به برنامه های کاربردی مبتنی بر وب آماده سازی راحت است. شرکتی که دارای ۱۰۰۰۰ کارمند رسمی می باشد را در نظر بگیرید. آیا مایلید جدیدترین نرم افزار را در ۱۰۰۰۰ رایانه نصب کنید؟ خیر! با پیشروی برنامه های کاربردی با الگوی توسعه و آماده سازی اینترنت/ اکسترانت لازم نیست کد (رمز) مشتری را در یک رایانه کاربران نهایی نصب نمایند. تغییرات وجود در برنامه کاربردی مبزور با انجام تأخیرهای به موقع (که به جمع کردن از حافظه برنامه کاربردی قدیمی مربوط می شود) فوراً در دسترس همه کاربران نهایی قرار گیرند.

بزرگترین عیب توسع وب متداول کارآیی آن بود. از آن جا کـه صـفحات Asp کد (رمز) کامپایل (همگردان) شده نبودند سرور (خادم) وب مجبور به تفسـیر کـد ASP می باشد. با ظهور ASP . NET به مؤلفه های NET کامپایـل (هگـردان) می شوند و به خودی خود در یک سرعت بومی (native) اجرا می شوند. ASP . NET به توسعه گران امکان می دهد تا از هر زبان مورد هدف NET نظیر، C# Manager و حتی COBOL استفاده کند (بله مـن یـک صـفحه ASP . NET نوشته شده در COBOL را دیده ام.)

اساس **ASP . NET**

برای اجرا شدن صفحات ASP . NET لازم است برنامه کابردی IIS کـاملاً با الحاقات (entensions) فرانت پیج ترکیب بندی شوند. در حین نصـب NET SDK یا نصب (Visual Studio . NET) نگارش مورد نیـاز الحاقـات فرانـت پـیج بایـد نصـب شـوند. بـرای وارسـی آن پنجـره Internet Information Services Manager را باز کنید و روی محله وب مزبـور کـه بیشـتر شـبیه بـه محله وب پیش فـرض اسـت کلیـک راسـت کـرده و گزینـه Cheack Server Entensions را از منوی All Task انتخاب کنید. هنگامی که روی خادم محلـی

کلیک راست می نمایید گزینه منویی All Task ظاهر می شود که منوی مضمونی گشودنی (context prp - up) را نشان می دهد.

Hello ASP . NET

معمولاً مایلم از هر IDE خاصی دو باشیم، زیرا چنین مکالمه هایی مـی تواننـد به یک جنگ ملیتی منجر شوند. به هر حال هنگامی که زمان سـاخت برنامـه هـای کابردی ASP . NET می شـود، Visual Studio . NET (VS . NET) بـا آن متناسب خواهد شد. با ترخیص VS , NET تمام توسعه هـای مـی تواننـد در یک IDE منفرد و آشنا رخ دهند. در حال حاضر توسعه برنامه کاربردی در VB . NET و C# و MC++ و FoxPro، و البته ASP . NET می باشد. در مورد شما اطلاعاتی ندارم ولی من در واقع به Intellisense وابسته هستیم.

در پوشه WebApplication 1 پرونده هـای پـروژه (طرحانـه) VS . NET یعنی caproj . * همراه با صفحه ASPX و کد (رمز) C# برای صـفحه Asp . NET وجود دارند. صفحات ASPX دارای دو بخش هستند، HTML و صفحه مبدأ (که در این حالت پرونده cs. می باشند) این جداسازی UI و منطـق تجـاری برای توسعه برنامه های کاربردی ضروی است و گروه ASP . NET مطمئناً ایـن جزئیات را نظارت نمی کند.

از میان برنامه های ایجاد شده برای برنامه کاربردی Asp . NET موارد اصلی و مهم آنها عبارتند از:

- WebForm1 . aspx صفحه ASP که توسط مروگر (browser) کلایت (مشتری) درخواست می شود.
- Web . config پرونده ترکیب بندی XML می باشد. ASP . NET امکان اداره راحت از طریق این پرونده ساده مبتنی بر XML را می دهد.

هنگام مرور Solution Explorer در VS . NET متوجه خواهید شد که پرونده WebForm1 . aspx . cs ظاهر نمی شود. برای مشاهده صفحه پشتیابن کد (رمز) برای WebForm1 . aspx لازم است گزینه View Code را از منوی مضمون انتخاب نمایید.

با تنظیم خصوصیات برچسب پروژه (طرحانه) Web Application را بسازید و اجرا کنید و این کار را می توان با فشار دادن کلید F5 انجام داد. اولین بار که صفحه aspx از IIS درخواست می شود، کامپایل (همگردانی) و اسمبلی (همگذاری) خواهد شد. [که در این موارد WebApplication1. D11 ایجاد می شود و در دایرکتوری و در دایکتوری (فهرست) bin قرار می گیرد]. در تقاضاهای متوالی اگر مبدأ (source) مزبور تغییر نماید. [برای اطمینان یافتن از این که d11 بهنگام شده است و از اسمبلی (همگذاری)

کامپایل (همگردانی) شده استفاده نموده یا مجدداً کامپایل می کند] IIS بررسی خواهد شد.

یکی از مهمترین مزیت های ASP . NET و VS . NET توانایی در اشکال زدایی یک ASPX Web Application ، با استفاده از اشکال زدای VS . NET می باشد. کد (رمز) WebFom1.aspx را ببینید و قعطه کد (رمز) زیر را قرار دهید:

حکم InitializComponent (statement) یک خط انفصال قرار دهید. نقطه توقف را می توان با کلید F6 قرار داد. با قرار دادن نقطه توقف مزبور در جای مناسب همان طور که در شکل ٦-١-٤ شرح داده شد،کلید F5 را فشار دهید تا پروژه Web Application اجرا شود و در نتیجه شما شروع IE را مشاهده خواهید کرد. سپس هنگامی که روش page – Init فراخوانی می شود و خط انصال مزبور به خط ٦ می رسد، انتقال آن به VS . NET Debugger را کنترل (نظارت) کنید.

توسعه گرانی که با نگارش های قبلی Visual studio میکروسافت کاملاً آشنا هستند، در حین توسعه برنامه های کاربردی Asp . NET بسیار احساس راحتی می کنند. قبل از ایجاد NET و VS . NET فرآیند اشکال زدایی یک برنامه کاربردی مبتنی بر ASP شبیه به افتادن یک تیغ تیز در یک محصول الکل بود.

افزودن موارد اصلی

هدف از Asp . NET توسعه سریع برنامه کاربردی مبتنی بر وب است. بنابر این میکروسافت ٤٥ کنترل (ناظم) برای برنامه های کاربردی (Asp . NET) ایجاد کرده است. کنترل (ناظم) های Asp . NET طرف خادم (server - side) را به طور خودکار به سطح مورد تقاضای مشتری پایین می آورند. امروزه اکثر محله های وب برای پشتیبانی IE و Netscape دارای کد (رمز) هستند، بنابراین لازم است که Web Developer دو مجموعه کدی که کارهای یکسانی انجام می دهند را نگهداری کند. با ورود کنترلهای طرف خادم، زود است که این موضوع را کنار بگذاریم.

ASP . NET علاوه بر کنترل (ناظم) های Asp . NET کنترل های استاندارد HTML را نیز پشتیبانی می کند. در حقیقت کنترلهای مزبور می توانند در یک صفحه قرار گیرند و حتی تودرتوی یکدیگر باشند.

یکی از اهداف اصلی هر برنامه کابردی، جمع آوری داده ها و نمایش آنها به یک شکل قابل قبول باشند. یک Web Application داخلی را در نظر بگیرید که برای جمع آوری اطلاعات درباره یک استخدام جدید استفاده می شود. سپس این اطلاعات برای پردازش به HR ارائه می شوند و در پایگاه داده های شرکتی ذخیره می گردند.

برای شروع فرآیند مزبور شناخت اساسی WebFrom ها ضروری است. درست همان طور که برنامه های کاربردی فعلی ASP از سبک تقاضا / پاسخ (Request / response) برای جمع آوری داده ها استفاده می کنند، ASP. NET نیز همین کار را انجام می دهد. این عمل از طریق From Post قدیمی انجام می شود.

امتحان ایده ها

بهترین روش یادگیری یک زبان / سکو (platfrom) توسعه جدید ایجاد برنامه کاربردی مفید است که تا حد امکان زبان و چارچوب مورد نظر را در بر می گیرد. در مورد Asp . NET و چارچوب NET حداقل می توان گفت که یک تکلیف قابل ارزیابی است.

برای جستجوی Asp . NET یک پایگاه داده های Employee مبتنی بر وب، به این روش ادامه می دهد تا راحتی توسعه که به وسیله Asp . NET انجام شده است را نمایش دهد. توانایی خود در مورد ایجاد لی آوت (طرح بندی) های جالب صفحه و طراحی گرافیکی (نگاره ای) را نمی دانم. بنابراین کارهای هنری آن را به عهده شما می گذارم. طبق این روش عناوینی که به آنها پرداخته می شوند شامل کنترل (ناظم) های Asp و Ado . NET توسعه کنترل (ناظم) وب سفارشی، بارگذاری (unpload) پرونده طرف مشتری و سایر چیزهای مورد استفاده می باشد.

کار را با ایجاد پروژه (طرحانه) جدید Web Application باید پیش فرض Gridlyout باشد، این امر قرار دادن مطلق کنترل (ناظم) ها در رابطه با مختصات(\x,y) را امکان پذیر می سازد. همچنین مطمئن باشید که tragetSchema به Internet Explorer 5 تنظیم شده است.

همان طور که طراح Windows Forms برخی از بدترین کدهای شناخته شده برای افراد ار ایجاد می نماید طراح WebFrom نیز این کار را انجام می دهد. در حقیقت کار را به پایان رساندم و تکه گزینی (trimmed) نموده و کد متعلق به DataFrom. Asps را پاک کردم، بنابراین می تواند به طور واقعی خوانده شود. TTML در اصل برای DataFrom. Aspx ، XHTML است. به خط ۲۶ توجه کنید که در آن یک توضیح را درجه نموده ام که اعلان می کند کنترل asp: repeator بعداً اضافه می شود. از کنترل (ناظم) های asp: repeator استفاده می شود تا کارمندها را به محض اضافه شدن نشان دهد.

یک Web From از System . web . UI . Page مشتق می شود و سپس صفحه c همان طور که در خصیصه Inherite صفحه مزبور لیست (سیاهه) شده است) از کلاس DataForm DataFrom. Behind مشتق می شود کلاس (رده) پیش فرض DataFrom. Behind صفحه CodeBehind (پشتیبان رمز) می باشد و از کلاس (رده) پیش فرض Webfom1 مجدداً نام گذاری می شود. این موضوع مهم است که توجه داشته باشید به نظر می رسد که ASP . NET در حین سرو کار داشتن با نام کلاس که همان نام را به عنوان فضای نام موجود (که در آن قرار دارد) به اشتراک می

گذارد، مشکلاتی به همراه دارد. به این دلیل نام کلاس (رده) برای صفحه پشتیبان کد (رمز) با نام پرونده فرق دارد. VS . NET به جای این که نام مناسب را برای شما اعلان کند، عمومی ترین نام ممکن را ایجاد می نماید. مطمئناً این موضوع یکی از دلایل نارضایتی من است! شما اغلب هنگام نامگذاری مجدد کلاس (رده) ها و فرم ها و همه مرجع های ایجاد شده به آنها وقت زیادی را هدر می دهید.

همانند اعلان متغیرهای موجود در صفحه aspx واقعی پرونده Data from aspx . cs . نیز دارای اعلان های متغیر می باشد. این امر دستیابی به خصوصیات کنترل(ناظم) ها و روش ها (در طی بارگذاری صفحه) و سایر رخدادهای اداره شده توسط کلاس DataFrom. Behind را امکان پذیر می سازد. اگر شما نگران این هستید که چه اتفاقی می افتد، اولین باری که صفحه مزبور در دسترس قرار می گیرد، ASP . NET کار ایجاد رمز انجام می دهد که در طی آن زمان کد چسبنده (glue - code) لازم ایجاد می شود تا کنترل های aspx را به متغیرهایی که در صفحه CodeBehind اعلان می شوند، پینوند دهد. بدین ترتیب در لسیت (سیاهه) ۲-۱-٤ کلاس (رده) DataFrom. Behind که درطی این فرآیند استفاده می شود شامل منطق صفحه ای برای عمل هایی نظیر مقیدساز داده ها (datubinding) طرح بندی (layout)،

اعتبار سنجی (validation) و سایر منطق های تجاری مورد نیاز هر برنامه کاربردی می باشد.

من مواردی راکه به کلاس (رده) DataFrom. Behind اضافه نمودم که لازم است شما نیز این کار را انجام دهید. اولین مورد اعلان متغیر EmployeeRepeater است. یک Repeater گسترش قالب آسای (template) کد HTML در طی پردازش صفحه را امکان پذیر می سازد. در این مثال برای هر کارمند سطرهایی به جدول HTML اضافه می شود. در کلاس Repeater بازتابش برای جمع آوری اطلاعاتی مهم درباره موضوع (ها) خصوصیات عمومی آن موضوع را سرشماری می کند.

سپس لازم است تا رده ای که یک کارمند را نشان می دهد وجود داشته باشد. هم اکنون وقتی یک کارمند جدید اضافه می شود کارمند مزبور در یک لیست (سیاهه) آرایه ذخیره می گردد و سپس به عنوان مبدأ داده ها (data surce) برای کنترل Repeater استفاده می شود. یک کلاس (رده) بسیار اساسی به منظور نگداری از اطلاعات کارمند ساخته می شود. لیست ۳-۱-٤، شامل کلاس (رده) Employee می باشد.

اگر چه این کلاس (رده) در واقع کار مهمی انجام نمی دهد، ولی نحوه کنترل (ناظم) Repeater را نشان می دهد. همچنین ارزش ندارد که هر کنترل (ناظم)

asp که از مقیدسازی داده ها استفاده می نماید، این کار را از طریق خصوصیات موضوع مقید انجام دهد به نظر می رسد که ناظم های مقید، فیلد (میدان) های عمومی را نادیده می گیرند و فقط خصوصیات عمومی (publice) را جستجو می کنند.

به منظور مورد استفاده قرار دادن کنترل Repeater، کد موجود در لیست ٤-١-٤ را به پرونده DataForm . aspx (جایی که در اصل توضیح مربوط به کنترل تکرار کننده را درج نمودم). اضافه کنید.

با افزودن قالب آسای asp: Repeater (template) مناسب هر بار که یک کارمند جدید به جدول اضافه می شود یک سطر جدید نیز به آن اضافه می گردد. توسعه گران C++ درحال حاضر با ایده مربوط به قالب آساها و کاربرد آنها در طی توسعه آشنا باشند.

قالب آساها یک چارچوب عمومی را امکان پذیر می سازند که می تواند در صورت لزوم بسط داده شود. شما باید با asp: Repeater احساس راحتی کنید زیرا زمان توسعه را بسیار کم می کند. نتیجه نهایی باید شبیه شکل ٩-١-٤ باشد.

هم اکنون مقدمه اصلی برنامه کاربردی ایجاد شده است. احساس شروع توسعه ASP . NET باید به طور موقت یک احساس دورنی مبهم در شما

ایجاد کند و شما را آماده می کند تابیشتر از عهده Web Appolication برآیید.

خلاصه

در این فصل اساس توسعه وب با استفاده از ASP . NET می پردازند. هر عنوان در رابط با مسافت برنامه کابرد نهایی Employee Browser ایجاد شد. من شما را تشویق می کنم تا هر وقت صرف نمایید و قبل ازرفتن به بخش بعدی بیشتر خودتان، هر عنوانی را جستجو کنید. این امر به شناخت بیشتر ASP . NET کمک می نماید و به شما امکان می دهد که خود به آن پی ببرید.

خدمات وب (Web SErvices)

در طی مدت چند سال دستیابی داده ه یا برنامه های کاربردی توزیعی فناوری های متنوعی را ایجاد نموده است. آنها از سازوکارهای عمومی (Remote Producr

Class(RPC نظیر XML – RPC تا موضوعات توزیع شده کاملی نظیر CORBA و DCOM را شامل می شوند. الگوی Web Services بر مبنای فرضیه ساده از توسعه سازوکار پایه XML – RPC می باشد تا به رویه امکان دهد که در اینترنت با استفاده از ضوابط نقل و انتقال باز (نظیر HTTP در اتصالات TCp/ IP) فراخوانی کند.

NET، احضار Web Services ها را از طریق HTTP POST و HTTP GET و Simple object Access Protocol (SDAP) پشتیبانی می کند. پیش از معرفی پشتیبان خدمات رسان وب (Web Services) د رNET، توسعه میزبانی خدمات رسان وب، وکیل (proxy) و کلاینت (مشتری) یک تکلیف غیر بدیهی بوده است. حتی میکروسافت نیز SOAP Toolkit را برای Visual Studio 6 ترخیص می کند تا به توسعه گر امکان دهد که موضوع COM را در اینترنت ارائه کند. این Toolkit فاقد پشتیبانی ارائه شده از طریق NET و C# می باشد. توسعه Web Services نسبت به تعریف یک مجموعه عمومی از روش هاجهت ارائه به کلاینت ها (همراه با نوع های بازگشتی مناسب و اعمال WebMetodAttributee ساده به روش های مورد نظر کمی ضروری تر می باشد.

در طی این فصل یک EchoService کوچک به عنوان نقطه شروع برای آشنایی شما با مفهوم Web Services عمل نموده و در پایان یک رابط خدمات رسان وب را به EmployeeBrower ASP WebApplication اضافه می کند.

rEcho Sevice (خدمات رسان پژواک)

ایجاد برنامه های کاربردی خادم/ مشتری عموماً شبیه به مسأله مرغ و تخم مرغ است. وجود یک خادم برای امتحان کردن مشتری و یک مشتری برای آزمودن خادم ضروری است. یک خادم پژواک تمام اطلاعات دریافتی برای مشتری که داده ها را ارسال نموده باز می گرداند همچنین به برنامه کابردی مشتری امکان می دهد تا اتصال خود را آزمایش کند و اطمینان یابد که داده های ارسالی صحیح هستند. پیاده سازی Echo Sevice با استفاده از NET WebServices به عنوان نقطه شروع مناسبی عمل می کند.

VS . NET ، ایجاد Web Services را مستقیماً از TDE پشتیبانی می کند. VS . NET با استفاده از الحاقات ErontPage ، جهت باز کردن یک پوشه وب به یک خادم IIS خاصی متصل می شود که درست شبیه

WebApplication توسعه یافته در فصل های گذشته می باشد. شکل ۱-٤-٤ کادر محاوره ای پروژه (طرحانه) جدید را برای ایجاد یک Web Services نشان می دهد.

همان طور که خدمات وب از طریق VS . NET ایجاد می شوند ازدو پرونده نیز تشکیل شده اند: پرونده asmx و یک پرونده مبدأ C# به هر حال به این دلی که کل رمز (کد) C# می توانند در پرونده asmx قرار گیرند فقط پرونده asmx ضروری می باشد. به هر حال رویکردی نیست که ما هنگام توسعه نمونه های موجود در این فص له کار می بریم.ملغی ساختن یا پیاده سازی به کلاش مشتق شده نیاز ندارد و در عوض کلاس (رده) پایه مزبور فقط پیاده سازی لازم باری پشتیبانی احضارهای روش را ارائه می دهد.

کلاس (رده) EnchoServer خود به تنهایی فقط یک رویش مفرد ارائ می دهد که به مشتری ها نظیر Echo ارائه می شود. به صفت مورد استفاده در این روش توجه کنید.صفتwEDmECHOD از طریق زمان اجرا استفاده می شود تا روش هایی که کلاس مورد نظر به مشتری ها ارائه می دهد را مکان یابی کند. در روش مشابهی که کلاسDBAccess برای نمونه EmployeeBrowser پیاده سازی می شود. زمان اجرا (runtime) برای

مکان یابی WedMotode و احضار آن با پارامترهای منساب از Refection API استفده می کند.

برای میزبانی IIS از WebServer وجود پرونده esmx در فهرست (دایکتوری) مجازی وب که شامل WebSErviva EchoServiva می باشد ضروری است. پرونده مزبور اتصال کد خدمات رسان وب به dill و کلاس ارائه دهنده خدمان رسان وب واقعی به کار برده می شود اتصال پایه پرونده asmx برای کد مورد نیاز C# در لیست (سیاه) ۳-٤-٤ نشان داده شده است.

همان طور که مشاهده می کنید در واقع برای پرونده asmx موارد زیادی وجود ندارند. همراه با صفحات asmx یک رهنمود زبان و یک رهمنمود Codebehi (پشتیبانی رمز) وجود دارد که پرونده مبدأ و رهنمود کلاس را که نام مناسب از کلاس ارائه کننده پیاده سازی WebService را ارائه می دهد مشخص می کند.

به این دلیل که EchoService به عنوان پرونده مبدأ C# پیاده سازی WebService شده و به یک اسکرپیت درون برنامه ای لازم است که پرونده EchoService . asmx.cs کامپایل همگردانی شده و نتایج DII حاصل به فهرست bin از فهرست مجاری جاری که EchoService را میزبانی می کند رونوشت گیری شود. VS . NET به طور خودکار فرایند مزبرو را در طی

فرایند ساخت اجرا می کند. آزمایش خدمات رسانی صرفاً مستلزم دستیابی به پرونده asmx (به مرور گر Internet Explorer Web) می باشد.

سیستم خدمات رسانه های وب فرم اصلی HTML را ایجاد می کند که از طریق آن می توان خدمات وب را امتحان کرد. حال جعبه ورودی (Input) را قرار دهید مقداری متن را در آن واردنموده و سپسر روی دکمه Invoke کلیک نمایید.

ایجاد یک کلاس وکیل (Proxy)

برای احضار WebService(خدمات وب) از کد (رمز) C# وجود کلاس وکیل ضروری می باشد. یک وکیل برای کد که خادم (server) واقعی یک پل ارتباطی را نشان می دهد به ترتیب خادم زیر برنامه ای دارد که از طریق آن وکیل به خادم متصل می شود.

برای ایجاد یک وکیل لازم است ضابطه خاصی را که می خواهید به کاربرید تعیین کنید. انتخاب های مرور گر مزبرو HTTP GET m HTTP POST یا COAP هستند. بر مبنای این تصمیم وکیل می تواند از خصوصیت کلاس (رده) پایه System Web. Services . Protocols. X مشتق شود یعنی جایی که x ضابطه متناظر را نمایش می دهد.

یک خبر خوب این است که لازم نیست شما این کد (رمز) را با دست تحریر کنید. به همراه NET SDK ابزاری به نام WSDL. Exe وجود دارد که هدف آن ایجاد کد کلاینت (مشتری) برای خدمات وب می باشد. اجرای WSDL . exe بدون هیچ پارامتری خطاهای فرمان و معانی آنها را نمایش می دهد. فرم اصلی برای WSDL همانند برنامه زیر ظاهر می شود:

WSDL <URL or wad1 document> <Language>
<protocol>

خدمات وب می تواند به طور همزمان یا غیر همزمان احضار شوند. مورد مشترک آنها استفاده از احضار روش همزمانی (synchronously) می باشد. احضار روش همزمانی در مواردی که فراخوانی روش برای پردازش به دوره زمانی وسیع نیاز داشته باشد کلاینت (مشتری) میتواند هنگام انتظار برای بازگشت پاسخ همزمان به کار خود ادامه دهد. این عمل امکان می دهد تا سایر کارها بدون وصل شدن به برنامه کاربردی کلاینت (مشتری) اجرا شوند.

مقید سازی (binding) مختلف HTTP GET, HTTP POST یا SOAP به شما فرصت می دهند تا برنامه کابردی مشتری را (که از factory برای ایجاد و بازگشت مقید سازی درخواستی استفاده می کنند) توسعه دهید. یک خدمان وب برای موضوعات متعددی قابل ارائه می باشد همچنین هر یک

از این موضوعات می توانند در فرم های یک رابط عمومی مشاهده شوند.

رابطی که روش های عمومی یک Web Servuce را تعریف می کند خود نیز می تواند تعریف شود و سپس هر وکیل این رابط را پیاده سازی می نماید که این هم یک موردی است.

درمورد EchoService رابط مورد نظر برای Webservice خود می تواند همانند زیر بیان شود:

Publice Interfact I EchoService

String Encho (string managed) ؛

این رابط فقط مفهومی (conceptual) می باشد و در واقع EchoService رابطی ندارد که بتواند ارجاع داده شود.

ProxyFactoty

انگاره Factory (pattem) کمک کند تا مشتری ها از متن های زیربنایی که یک عامل آنها را ایجاد می کند مجزا شوند. Factory یک متن کامل شناخته شده ای را باز می گرداند (که در این مورد منظور رابط I EchoService می باشد.) هر وکیل به پیاده سازی رابط I EchoService نیاز دارد. در واقع وکیل

های لیست (سیاهه) شده همیشه اجرا می شوند و فقط لازم است که رابط I EchoService به اعلان کلاس (رده) اضافه شود.

کلاس (رده) Proxy Factory یک enum را به عنوان یک پارامتر برای تعیین موضوع زیربنا (به منظور ایجاد و بازگشت آن) می گیرد. لیست (سیاهه) ٦-٤-٤ پیاده سازی Proxy Factory را نشان می دهد.

کلاس (رده) Proxy Factory ایجاد وکیل درخواستی را که مبتنی بر مقید سازی عبور داده شده به روش ConstructProxy می باشد محصور می کند به دلیل این که رابط I EchoService توسط هر یک از پیاده سازی های وکیل زیربنا پیاده سازی می شود برای Proxy Factory فقط لازم است که کلاس (رده) مناسب وکیل ساخته شده و رابط I EchoService نیز برگردانده شود.

مشتری فرم های ویندوز

برای امتحان کردن خدمات وب، کلاینت (مشتری) WinForms رابطه UI رابطه به EchoService ارائه می دهد. با استفاده از VS . NET برنامه کاربردی C# Windows Forms را ایجاد نموده و فرم را بسازید.

به دلیل این که طراح (disigner) فرم ها بیشترین کد (رمز) را ایجاد می کنند فقط بخش هایی مربوط (به جای لیست کردن کامل کد) لازم است که بازبینی شود. حال متغییر حفاظت شده را اضافه کنید:

```
Private ProxyProtocal      SelectedProtocal =
ProxyProtocal. SOAP"
```

انجام این کار به برنامه کاربردی امکان می دهد تا ضابطه درخواستی برای احضار روش WebSevice پیگیری شود. اطمینان یابید کلمه Proxy Using Factory در اعلان های نام کده (فضای نام) قرار گرفته است.

سپس هر دکمه رادیویی را به همان اداره کننده Click متصل نمودم که شامل کدهای زیر می باشد:

```
protected void onopt – Click (object sender , Eventagrs )
else if (sewnder == optSOAD)
this selectedProtocol = ProxyProtocal. SOAP ؛
else if (sender == optHttpPost )
this . Selectedprotocol = ProxyProtocal. HttpGet:
```

کد رمز مزبور برای بهنگام سازی متغیر Selectedprotocol بـر مبنـای دکمـه رادیویی انتخاب شده جاری عمل می کند. به منظور تخصیص اداره کننده هـای delect برای کنترل ها به فصلهای WinForms مراجعه کنید.

سـرانجام وقتـی دکمـه Invoke، رخـداد Click را فعـال مـی کنـد احضـار WebSevice روی خواهد داد.

به همراه کد (رمز) رخدادگران در یک مکـان کلاینـت (مشـتری) Windows Forms آماده حرکت می شوند. شکل ٦-٤-٤ برنامه کاربردی مشتری را نشـان می دهد که برای احضار Encho WebSevice آماده است.

حال که مـورد مشـترک اسـتفاده از احضـار روش همزمـان (synchronous) بحث شد مرحله بعدی این است که احضار روش همزان را درک کنید. احضـار روش غیر همزمان به یک نماینده همراه با توشـیح (signature) روش زیـر نیاز دارد:

< ACCESS MODIFIER > VOID <NAME> (System . IasyncResult)

خوشبختانه یا متأسفانه در نحوه عملکرد این سازوکار هیچ مورد مبهمـی نـدارد. فرایند اصلی برای فراخوانی همانند نسخه همزمانی می باشد و در حـال حاضـر

فقط یک سرنخ (thread) برای اداره کردن فرایند ارسال و دریافت مورد تقاضا ایجاد می شود. پس از تکمیل تقاضای مورد نظر یک رخداد فعال می شود و یک نماینده ضمیمه شده نیز فراخوانی خواهد شد.

به وکیل های مختلف WebSevice ایجاد شده در لیست ۳-٤-٤ و ٤-٤-٤ و ٥-٤-٤ توجه کنید. هر وکیل همانند زیر روشی را تعریف می کند:

publice system . IasyncResult BeginEcho (string message’
System . Asyncallback callback’
Object asynState

دقیقاً همان طور که روش Echo یک پارامتر رشته ای را می گیرد، روش BeginEcho نیز این کار را انجام می دهد. به علاوه روش BeginEcho به یک نماینده برای فراخوانی متقابل (clallback) و یک موضوع که دارای درخواست غیر همزمان است نیاز دارد.

پس از احضار روش Callback موضوع اصلی EchoServiceSoapProxy رابط IasyncResesult به دست می آید. سپس روش EndEcho برای به دست آوردن نتایج فراخوانی غیر همزمان روش WedService فراخوانی می شود.

خوشبختانه EchoService یک مقدمه مناسبی را در رابطه با توسعه آسان و در نظر گرفتن کلاینت مشتری چنین خدمات وبی (EchoService) ارائه می گردد.

برگرداندن نوع هخای تعریف شده توسط کاربر

بیشتر مواضع ضروری است که نوع های تعریف شده توسط کاربر از یک روش (نظیر کارمند یا بخش) برگردانده (return) شوند. هنگام انجام این کار سری سازی خدمات وب نمایش مشابه از عنصر داده ها را برای برگرداندن به مشتری ایجاد می کند. بسیار مهم است که بدانید خدمات وب باید برای بازگشت داده ها و نه موضوعات به کار روند. اصولاً یک یادآور (memento) باید به جای تلاش برای نمونه سازی از یک موضوع و کوشش به منظور شمار ش های مرجع توزیعی برگردانده شود.

یک یادآور (memento) حالت فعلی یک موضوع یا داده های لازم برای توصیف یک موضوع را نشان می دهد. اطلاعات اساسی در رابطه با یک کارمند (نظیر ID نام و بخش و ID عکس) داده های مربوط به کارمند را نشان میدهد.

خدمات سریال سازی تمام فیلد (میدان) های عمومی و خصوصیات عمومی را به همراه دسیابی های GET سریال سازی می کند. برای پشتیبانی از سریال سازی خصوصیات متناظر باید SET را همانند GET در دستیاب پیاده سازی نماید. هر فیلد (میدان) یا خصوصیات به یک عنصر XML نگاشت می شود. این عملکرد پیش فرض ممکن است برای خدمات ساده مناسب بوده و همچنین به عنوان یک نقطه شروع برای استفاده از User Defriend (UDT) Types در محتوای خدمات رسان وب عمل کند.

ایجاد خدمات رسان

با ایجاد یک خدمات رسان وب به نام UDTWebService (با استفاده از VS. NET) کار را شروع نموده و پرونده پیش فرض Servical . asmx را به UDTService.asmx نامگذاری مجدد نمایید. نظیر همه خدمات رسان های ایجاد شده پیاده سازی واقعی در کد صفحه پشتیبان کد (codebehind) قرار می گیرد.

خدمات رسان وب روشی را ارائه می دهد که شخص را بر مبنای نام خانوادگی او مکان یابی می کند، از ا ین رو نام روش مزبور locateperson خواهد شد. قبل از پیاده سازی روش مزبور این عمل برای ایجاد کلاس (رده) Person که

می تواند برای بازگشت اطلاعات به کلاینت کلاس UDTPerson کلاس(رده) اصلی C# را بدون صفت های اعمال شده بر آن نشان می دهد بنابراین چگونه می توان کلاس مزبور را سریال سازی نمود.؟ و آیا Reflection را به یاد می آورید؟ این یک سازوکار (مکانیسم) مورد استفاده برای سریال نمودن اطلاعات را به کار می برد. مجدداً فیلد (میدان) UDTPerson مرحله بعدی افزون روش locateperson به کلاس °×ثقسخد می باشد.

به جای بازیابی از یک پایگاه داده ها مثال مزبور برای نگاه داشتن تعدادی از افراد مختلف (که جستجو می کنید) یک جدول درهم ایجاد می کند. روش locateperson در خط ۵۲ LastName را به عنوان پارامتر رشته ای می گیرد. بسیار مهم است که توجه کنید به نظر می رسد روش مزبور نسبت به Webma thod موجود در EchoService هیچ تفاوتی نداشته باشد. به دلیل این که سازو کار پیش فرض سریال سازی مورد مشترکی را اداره می کند لازم نیست کد (رمز) سریال سازی سفارشی برای کلاس (رده) UDTPerson ارائه دهید.

ایجاد مقید سازی کلاینت (مشتری)

بار دیگر از ابـزار WSDL. Exe از ابـزار WSDL . exe مـی تـوان بـرای ایجاد کلاس ضروری وکیل (proxy) منظـور احضـار روش Locateperson استفاده کرد. WSDL. Exe به همراه کلاس وکیل کـلاس سـادهای کـه نـوع برگشتی UDTPerson رانمایش می دهد.، ایجاد می کند. علت این اسـت کـه کاربران خدمانت رسان وب به پشتیبان کد خدمات رسان (سـرویس) دسترسـی پیدا نمی کنند. در عوض آنها مجبور خواند شد برای ایجاد کـد (رمـز) مشـتری لازم (به منظور احضار خدمات رسان ارائه شده) بـه توصـیف WSDL سـایت (محله) تکیه کنند.

کد ایجاد شده توسط WSDL. Exe در اصل همان است کـه قـبلاً مشـاهده نموده اید، موارد جدید آن کلاس (رده) UDTPerson می باشد. توجـه کنیـد که صـفتهای First Name و LastNanme بـه جـای فیلـد (میـدان) هـای عمومی به عنوان فیلدهای خصوصی ایجاد می شوند. همچنـین WSDL. Exe باید برای ساخت وکیل و نوع های ارجاع شده اطلاعات موجود در سند wsdl را به کار برد.

احضـــار Locateperson . UDTServica از Internet , XML Explorer برگشتی را ایجاد خواهد کرد و آن را در مرورگر نمایش مـی دهـد. (شکل ۸-٤-٤) XML مزبور پیام برگشتی خدمات رسان وب را نشان میدهـد.

مدت زمان کوتاهی را به مطالعه نحوه نگاشت XML از طریق ابزار .WSDL Exe به کلاس (رده) اختصاص دهید.

صفت های XML

این جا نحوه بازگشت نوع های فطری (intrinsic) و UDT ها از خدمات رسان وب (WebService) رامشاهده نمودید سریال سازی ارائه شده توسط خدمات وب دراصل کافی خواهد بود. ولی زمانی فرا می رسد که لازم است نمایش عناصری که برگردانده (return) شده اند کنترل (نظارت) نمایید.

یک سند XML می تواند بیش از یک رشته داشته باشد. ریشه XML را می توان با استفاده از MLRoXotAttibute تعریف نمود. همچنین MLRoXotAttibute برای مشخص نمودن خوصیات موجود در جدول ۱-٤-٤ استفاده می شود.

خصوصیات	مفهوم
DataType	نوع داده های XML مربوط به عنصر ریشه است
Flement Name	نام عنصر ریشه می باشد
Form	نام عنصر ریشه توصیفی یا غیر توصیفی

TsNullable	درمواقعی که null می شود تعیین می کند که آیا لازم است XmlSerializer سریال سازی شود
Namespace	نامکده (فضای نام) ای برای عنصر ریشه

همچنین عنصر مزبور می تواند نوع هویت سند را بررسی نماید.

XmlElementAttibute برای توصیف این عناصر به کار برده می شود.

عناصر می توانند یک کلاس (رده) یا اعضای کلاس باشند. علاوه بر مشخص کردن XmlElementAttibuteAttribute به آنها اعمال شده است.

تاکنون فقط موضوعات تک سطحی با ایجاد خدمات رسان وب برگردانده شدند. XmlSerializer قادر به برگشت ساختارهای پیچیده XML باکلاس (رده) های تودرتو می باشد. همچنین استفاده از صفت های مختلف در ترکیب با ساختارها و آمیخته در موجودیت (entity) ها، صفت های XML غیر صریح امکان پذیر شده است.

به منظور مشاهده قدرت XmlSerializer در عمل یک خدمات رسان ساده وب برای برگرداندن Employee (.که حاوی دو موضوع کلاس Address و ساخت Department می باشد ایجاد می شود.

موجودیت ها در لیست (سیاهه) ۱۱-٤-٤، صفت های مختلف XlLL و ایجاد کلاس مرکب آمیخته و باهم متناسب می کنند Address و Employee

صفت های XML را به کار می برند. در حالی که در ساخت Department فقط از فیلد (میدان) های عمومی استفاده می شود. به طور پیش فرض تمام فیلد(میدان) های عمومی یک کلاس یا ساخت توسط کد سریال ساز XML خدمات رسان سریال سازی می وشد. این توانایی آمیختن و متناسب نمودن قدرت شگفت انگیزی را دراختیار توسعه گر می گذارد و به او امکان می دهد تا کنترل (نظارت) کاملی بر فرآیند سریال سازی داشته باشد.

به خصیصه های مختلف XLL و نحوه تأثیر گذاری آنها بر خروجی XML ایجاد شده در طی سریال سازی موجودیت Employee توجه کنید. همچنین بسیار مهم است که توجه داشته باشید بیش از یک روش آراستن کلاس با خصیصه های XML (که یک قالب سریال سازی را دریافت می کنند) وجود دارد. در نظر داشته باشید که کد وکیل (proxy) از طریق ابزار WSDL. Exe ایجاد شده است.

طبق معمول ابزار WSDL. Exe کد وکیل لازم برای احضار روش مورد تقاضای خدمات رسان وب را ایجاد می کند. WSDL. Exe همراه با کد وکیل موجودیت های Employee و Address و Department قبلاً توسعه یافته و مبدأ (surce) آن نیز با استفاده از ابزار WSDL. Exe ایجاد شده است.

اگر چه هر کلاس/ ساخت برای ایجاد قالب سریال سازی مناسب رویکرد های مختلفی را به کار می برد ولی هر دو پایده سازی در راستای یکدیگر قرار می گیرند. اساساً فقط نام ها در فیلدهاو خصوصیات آراسته شده تغییر کرده اند و در واقع از WSDL. Exe بر هر دو صفات مختلف XML و استفاده از Reflection برای ایجاد قالب سریال سازی مناسب تکیه دارند.

خلاصه

من شما را تشویق می کنم که برای امتحان کردن خدمات رسان وب ادامه دهید، زیرا آنها الگوی جدید توسعه در جداسازی (separation) داده ها و منطق (logic) نمایش را نشان می دهد. در این روش اطمینان یابید که کد وکیل ایجاد شده از طریق WSDL. Exe رابررسی نموده ایم. این امر به توسعه شناخت شما در این رابطه (نه تنها در مورد خدمات رسان وب بلکه XML و XML نگاشت شده بین یک موجودیت خاص و نسخه سریال سازی شده) کمک می کند.

فهرست منابع

۵- کتاب C# , .NET Framework. مترجم: مهندس حوریه شاه حسینی

٦- سایت اینترنتی WWW.microsoft.com

۷- سایت اینترنتی WWW.devn.com

۸- سایت اینترنتی WWW.SearchwebServices.Com