چرا به Microsoft.NET **احتیاج داریم ؟**

به طور معمول نسل های جدید زبان های برنامه نویسی به این دلیل متولد می شوند که زبان های قدیمی تر دارای امکانات محدود بودند و یا قدرت استفاده از تکنولوژی های فعلی را به صورت مطلوب و ساده ندارند .

مهمترین نیازی که به عنوان آخرین تکنولوژی وجود دارد ، برنامه نویسی در محیط اینترنت است . اینترنت در مدت تقریبا ۸ سال جای خود را به عنوان یکی از مهمترین وسایل ارتباطی برای کارهای روزمره و تجارت باز کرده است . سیستم های برنامه نویسی قدیمی تر امکان برنامه نویسی برای اینترنت را فراهم کرده بودند اما هر کدام دارای اشکالات بزرگی هستند ، برای مثال تکنولوژی COM (Component Object Model) اولین بار در ویندوز به کار گرفته شد ؛ در سال ۱۹۷۰ نیز سیستم هایی برای Unix نوشته شده بودند ، جاوا نیز در اصل برای ابزارهای الکترونیکی بود و نه برای اینترنت .

سپس برای اولین بار یک سیستم جامع برای برنامه نویسی تحت اینترنت ایجاد شد ؛ این سیستم- .NET – از مراحل سطح پایین که به زبان ماشین می باشد تا بالاترین سطح که برنامه نویسی ویژوال آن می باشد برای استفاده در اینترنت طراحی شده است . البته .NET فقط برای اینترنت نیست و با استفاده از آن می توان برنامه های کامل تحت Client نیز ایجاد کرد ، اما بزرگترین مزیت آن دربرابر سیستم ها دیگر امکانات اینترنت آن است .

برای اینکه مزایای استفاده از .NET را بهتر متوجه بشویم بهتر است در ابتدا معایب سیستم های پیشین را ذکر کنم . شرکت مایکروسافت تا قبل از سال ۱۹۹۵ به برنامه نویسی در محیط های Client و Server می پرداخت ، اما از آن سال به بعد توجه بیشتری به مساله برنامه نویسی در اینترنت کرد . مایکروسافت COM و COM+ را ایجاد کرد و آنها را در ویژوال استودیو ٦ به کار گرفت . در سال ۱۹۹۹ حدود ۵۰ در صد از بزرگترین سایتهای تجارت الکترونیکی از محصولات مایکروسافت استفاده می کردند .

اما هنوز هم مشکلات بزرگی در سیستم های مایکروسافت وجود داشت که یکی از آنها دشواری نوشتن برنامه در اینترنت با محصولات مایکروسافت بود . شرکت مایکروسافت برای راحتی کار برنامه نویس ها ASP یا Server Page Active را ایجاد کرد . با اینکه این یک قدم بزرگ بود و کارها را بسیار ساده کرد ولی هنوز از برنامه نویسی شی گرا پشتیبانی نمی کرد . همچنین در ویژوال استودیو ٦ قسمتی برای Internet Application ایجاد شده بود و در آنها امکان ساختن Web Class وجود داشت ولی هیچ وقت به عنوان یک ابزار کار آمد برای برنامه نویسی وب درنظر گرفته نشد .

مدل برنامه نویسی DNA

مایکروسافت یک مدل برنامه نویسی به نام Distributed interNet Application دارد که بر پایه برنامه نویسی n-tier و COM بنا نهاده شده است .

مدل DNA از سه بخش اساسی تشکیل شده است.

بخش اول به نام Presentation tire معروف است . در این بخش رابط تصویری کاربر وجود دارد و خود نیز به دو نوع Internet Browser و Win 32 GUI تقسیم می شود که هر کدام مشکلات خاص خود را دارند . در مدلی که از Win32 GUI یا همان نرم افزارهای معمولی استفاده می شود دو مشکل بزرگ وجود دارد ؛ دشواری بروز رسانی نرم افزار و دیگری DLL Hell که در ادامه توضیح داده خواهد شد . در نوع دوم مشکلاتی از قبیل نبود امکانات برنامه نویسی کافی در محیط browser ، نبود رابط قوی با کاربر ، نبودن browser های یکسان و ... وجود دارد . همچنین همیشه یک اتصال به اینترنت یا اینترانت لازم است . در این نوع از برنامه نویسی می توان از Java Applet ها یا ActiveX استفاده کرد ولی browser باید امکان استفاده از آن را داشته باشد ، مخصوصاً هنگام استفاده از ActiveX که باید فقط از IE

استفاده کرد.

بخش دوم که Middle tier نام دارد ، مکانی است که اطلاعات و قوانین تجاری در آن وجود دارد . منظور از قوانین ، متد ها و کامپوننت هایی هستند که اعمال کاربران را کنترل می کنند . مهمترین و آسان ترین زبان برای نوشتن این اجزا از DNA ویژوال بیسیک است . برنامه نویسی که بخواهد در این رده برنامه بنویسد باید آشنایی کاملی با COM و protocol های رایج داشته ، همچنین باید مهارت کافی در استفاده از ADO و ADSI داشته باشد(ActiveX Data Object)
و (Active Directory Service Interface) مشخص است که یک اشتباه در این لایه باعث بروز خطا و نقص در کل سیستم می شود.

بخش سوم یا Data tier مکانی است که اطلاعات سازمان در آن ذخیره می شود . معمولاً در این قسمت از بانکهای پیشرفته رابطه ای مانند SQL Server و Oracle استفاده می کنند.

محدودیت های COM

همانطور که دیدید مهمترین قسمت در DNA همان COM است که در جای جای آن استفاده می شود . در ایجا برخی معایب COM ذکر می شود : (در ابتدای متن ذکر شد که برای درک نیاز به NET. باید ابتدا معایب سیستم های قدیمی را بشناسیم)

DLL Hell

اگر کوچکترین تغییری در یک COM ایجاد شود ، دیگر برنامه هایی که از ورژن قبلی استفاده می کردند قادر به فعال ساختن نسخه جدید نیستند . هنگامی که در ویندوز یک COM نصب شود برایش در رجیستری یک GUID ثبت می شود که اطلاعات آن COM را در خود ذخیره می کند . اگر یک برنامه از نسخه اول یک COM استفاده کند و بعد از مدتی شما تغییراتی در نسخه اول بدهید و بخواهید آن را

دوباره در سیستم نصب کنید ویندوز به شما پیغام خطا می دهد چون ورژن آن تکراری است ، اگر هم آن را به ورژن دوم ارتقا دهید نرم افزار قبلی هنوز به دنبال نسخه اول می گردد. این امر باعث می شود که شما مجبور شوید یکبار دیگر کل برنامه را کامپایل کنید و در کامپیوتر نصب کنید .

کمبود در وراثت

در نسخه های COM که در حال حاضرهستند چیزی به نام وراثتی که در C++ وجود دارد نمی باشد ، بلکه وراثت تنها در interface یک COM می باشد ، استفاده از آن هم چندان کمکی به برنامه نویسی نمی کند .

برخی محدودیت های برنامه نویسی اینترنتی در مدلDNA

وجود دو محیط برنامه نویسی برای اینترنت وClient

نقصان در نوشتن برنامه هایی با رابط گرافیکی خوب که در اینترنت کار می کردند کاملاً مشهود است ، نمونه بارز آن اختلاف در برنامه نویسی در ویژوال بیسیک و ASP است . ویژوال بیسیک با رابط گرافیکی کاملاً سطح بالا و ASP تقریباً رابط گرافیکی ندارد . همین امر باعث می شد که یک برنامه نویس مجبور باشد طیف وسیعی از تکنیک ها و زبان ها را فرابگیرد تا بتواند برنامه ساده ای در اینترنت بنویسد .

نبودن حالت های ذخیره اطلاعات رابط گرافیکی در صفحه های اینترنتی

نمونه این حالت زمانی است که در یک textbox متنی وجود داشته باشد . در برنامه های Win32 GUI متن داخل textbox تا زمانی که کاربر یا برنامه آن را تغییر نداده بر جای خود وجود دارد . اما در محیط اینترنت و نوع ASP با هر بار refresh کردن صفحه کل اطلاعات ازبین می رود . البته این مشکل با استفاده از آبجکتهای Request و Response تقریباً قابل حل است ولی احتیاج به برنامه نویسی برای هر تکه از صفحه ASP است.

نداشتن Event Handler در محیط برنامه نویسی اینترنت

یکی از مهمترین ابزاری که در برنامه نویسی Win32 GUI وجود دارد استفاده از Event ها است . با تکنولوژی که در حال حاضر وجود دارد تنها راه رسیدن به این مهم استفاده از ActiveX است که به علت مسایل امنیتی در بیش از ۹۵ در صد مواقع توسط کاربر استفاده از آن رد می شود.

معایب استفاده از API

APIها توابعی هستند که از ویندوز نسخه ۱ تا امروز در برنامه نویسی کاربرد داشته و دارند . مهمترین کاری که این توابع انجام می دهند انجام کارهای سخت و سطح پایین سیستمی است که احتیاج به برنامه نویسی زیادی دارند و یا حتی امکان ایجاد آن با زبان هایی مثل ویژوال بیسیک نیست . اما ؛ هر APIاز هر نسخه ویندوز تا نسخه دیگر آن می تواند دچار تغییرات بشود . برای مثال برنامه ای که در ویندوز ۹۸ نوشته شده باشد می تواند در ویندوز ۹۵ اجرا نشود . همچنین هم اکنون ابزارهای جدیدی به بازار آمده است که برای آنها نیز می توان برنامه نویسی کرد ، مانند تلفن های سیار ، کیوسک تلفن ، دستگاه های کامپیوتری جیبی و غیره . در این نوع دستگاه ها دیگر ویندوز به مفهومی که در حال حاضر وجود دارد قابل اجرا نیست و در نتیجه API هم وجود ندارد . لازم به ذکر است که ویندوز CE برای دستگاه های مذکور می باشد ولی قابلیت های آن با ویندوز های دیگر تفاوت زیادی دارد .

راه حل Microsoft .NET –

با اینکه مایکروسافت می دانست با ابزارهای قبلی شرکت می توان برنامه های اینترنتی نوشت ولی برای قبضه کردن بازار احتیاج به تکنولوژی جدیدی داشت . مایکروسافت از سال ۱۹۹۸ که ویژوال استودیو ٦ را به بازار وارد کرد در پی حل این مشکلات بود تا در سال ۲۰۰۰ NET. را در کنفرانس برنامه نویسان حرفه ای PDC به جهان معرفی کرد . از آن روز تا به حال مایکروسافت حدود ۸۰ % از توانش را برای تکمیل NET. مصرف کرده است . در همین راه protocol های جدیدی مانند SOAP یا Simple

Object Access Protocolرا ایجاد کرد . همچنین نسل جدیدی از برنامه نویسی به عنوان WebService با نام کد Hailstrom (code name) را تهیه کرده است .

مهم ترین اهداف .NET.

طراحی برنامه های اینترنتی بر سبک برنامه های Win32 GUI

همانطور که اشاره شد برنامه نویسی برای Win32 GUI از قدرت خوبی برخوردار است ، در .NET. برنامه های اینترنتی نیز از همین قدرت برخوردارند .

داشتن رابط گرافیکی خوب اینترنتی

به علت تغییرات اساسی که در برنامه در این سیستم داده شده برنامه اینترنتی قابلیت گرافیکی در حد برنامه های Win32 GUI دارند.

انتقال ساده به سیستم های دیگر

در .NET. براحتی می توان برنامه ها را با یک کپی ساده به کامپیوتر های دیگر انتقال داد .

پشتیبانی از زبانهای مختلف

در .NET. محدود به زبانهای برنامه نویسی مایکروسافت مثل ویژوال بیسیک ، سی شارپ و یا C++ نیستیم . به طوری که در حال حاضر نسخه های Cobol.NET و Pascal.NET در حال ساخته شدن است . اما شرکت مایکروسافت زبان ویژوال بیسیک را به عنوان زبان اصلی برگزیده است . این مساله ریشه در تاریخ مایکروسافت دارد Platform ! های آینده هم اکنون .NET. برای ویندوز نوشته شده است ولی

در آینده نزدیک نسخه های Unix و Linux و همچنین برای Mobile و PDA نیز ارائه خواهد شد . این امر امکان این را می دهد که برنامه ای که برای ویندوز در .NET. نوشتید در تمامی سیستم عامل ها و دستگاه های بالا قابل اجرا باشد .در ادامه توضیحات کامل برای این مبحث ارائه خواهد شد.

ساختار .NET.

مهمترین ویژگی در .NET Framework. این است که تمام لایه های برنامه نویسی را در بالای سیستم عامل در بر می گیرد . که این شامل تمامی تکنولوژی های موجود که از طرف مایکروسافت یا شرکتهای دیگر ارائه شده است ، می شود . در .NET. تمام اعمال تخصیص حافظه و سازماندهی فایل بر عهده .NET Frameworkاست و همین اصل باعث می شود که بتوان برنامه هایی نوشت که به سیستم عامل متکی نباشد.

زیر ساختارCLR.NET.

قلب .NET Framework. همان CLR یا Common Language Runtime می باشد . CLR . مسئول اجرای فایل ها ، فراخوانی آنها به حافظه و کامپایل کردن آنها به زبان MSIL یا Microsoft Intermediate Languageاست . بعداً کدهای IL در هنگام اجرا ، برنامه بوسیله just-in-time کامپایلر ها به زبان ماشین تبدیل می شود . این بدین معناست که در .NET. دو مرحله برای کامپایل شدن وجود دارد . اولین مرحله وقتی است که برنامه به هر زبان .NET. که باشد به IL کامپایل می شود که این کد کامپایل شده به IL قابلیت پخش در تمام .NET Framework. را دارد و بستگی به سیستم عامل ندارد . مرحله دوم زمان اجرا است که just-in-time کامپایلر ها کد IL را به زبان آن ماشینی که برنامه در آن می خواهد اجرا شود کامپایل می کند.

CLRعهده دار برنامه نویسی شی گرا در سطح زبان های .NET. است ، برای مثال شما می توانید یک objectدر سی شارپ داشته باشید و آن را در ویژوال بیسیک فرا بخوانید و همچنین بوسیله وراثت تغییراتی در آن object بدهید . همچنین CLR بر Garbage Collection ها نیز نظارت می کند . بحث کامل درباره CLR در ادامه خواهد آمد.

.NET Framework Base Classes

این لایه حاوی تمامی کلاس ها و آبجکت هایی است که معمولاً مورد نیاز برنامه نویسان می باشد ، از جمله ADO.NET که نسل جدید ADO است ، XMLکه قسمت زیادی از .NET. از این تکنولوژی استفاده می کند ، Threadingیا آبجکت هایی برای برنامه نویسیِ هر. thread

ASP.NET و Windows Forms

در مرحله بعدی دو روش کلی برنامه نویسی تحت اینترنت و تحت client قرار دارد که هر کدام خواص و آبجکت های مخصوص آن روش برنامه نویسی را دارا هستند .

Common Language Runtime

CLRدر مرکز .NET platform. واقع شده است . در حقیقت CLR عملی را که runtime ها در زبان های پیشین انجام می داند انجام می دهد . برای مثال ویژوال بیسیک ٦ دارای یک Runtime مخصوص به خود بود که کارهای مدیریت حافظه ، فراخوانی فایلها و از این دست اعمال را برای برنامه های بیسیک انجام می داد ، همچنین ویژوال سی نیز دارای چنین ابزاری بود CLR . یک runtime مشترک است برای تمامی زبانهایی که بر مبنای .NET. طراحی شده اند . قبل از اینکه بیشتر درباره اجزای CLR توضیح بدهم بهتر است با ساختار برنامه های .NET. آشنا شوید .

مروری بر ساختار برنامه های .NET.

هر برنامه ای که بر مبنای .NET. تعریف می شود از سه قسمت مهم و اصلی تشکیل شده است : Assemblies، Modulesو . Types اسمبلی ها اصلي ترین جز برای انتقال برنامه های .NET. هستند . (Deployment)ماجول ها فایلهایی هستند که اسمبلی از روی آنها ساخته می شود و تایپ ها ، واحد های پایه برای تعریف داده ها ، propertyها و توابع هستند.

اسمبلی ها : اسمبلی تشکیل شده از manifest و یک یا چند فایل ماجول یا XML یا . HTML

Manifestنیز دارای اجزای زیر است:

--اطلاعاتی درباره خود اسمبلی که به صورت text ذخیره شده است . نمونه این اطلاعات ، نام ، ورژن ، عمومی یا غیر عمومی بودن اسمبلی و ... است.

--نوع حفاظتی اسمبلی را توضیح می دهد . هر اسمبلی می تواند برای اجرا شدن نوع خاصی از لایه امنیتی داشته باشد که بر سه نوع است : Required ، Optionalو. Denied

--اطلاعاتی درباره اسمبلی های دیگری که یک اسمبلی به آنها وابسته است از قبیل نام و ورژن آنها.

--اطلاعاتی از قبیل زبان محلی اسمبلی ، تاریخ ، واحد پول و غیره .

ماجول ها:

ماجول ها یا فایلهای DLL هستند یا فایلهای EXE Windows PE (Portable Executable) که حاوی IL ، Meta Dataو به صورت optional دارای manifest می باشد . هر اسمبلی فقط یک manifest می تواند داشته باشد ، بنابراین اگر ماجولی حاوی manifest نیز بود فقط همان ماجول است که manifest دارد . CLRدو روش برای کامپایل هر فایل IL دارد ، یکی install-time است که در زمان نصب برنامه فعال می شود و دیگری JIT یا just-in-time کامپایلر که به صورت method by method برنامه را کامپایل می کند ، یعنی هنگامی که برنامه هر method را صدا می زند کامپایل هم می شود . به صورت عادی برنامه ها به روش JIT کامپایل می شوند Meta Data . حاوی اطلاعات بیشتری درباره تعریف تایپ ها می باشد و به صورت IL است.

تایپ ها:

تایپ ها دو نوع هستند ، Valueو . Reference هر تایپ دارای property ، methodو field است . در ادامه درباره تایپ ها توضیحات بیشتری داده خواهد شد .

انتقال و: versioning versioning

در ویژوال بیسیک 6 و کلاً سیستم های قبلی که بر COM استوارند ورژن یک COM مساله بزرگی برای انتشار آن است . در حالی که شما می توانستید ورژن های مختلفی از یک COM داشته باشید ولی در استفاده از ProgID ها دارای محدودیت بودید . برای مثال در ویژوال بیسیک هنگام استفاده از Word.Applicationنمی توانستید ورژن آن را انتخاب کنید ، یعنی برای مثال . Word.Application.9 در CLRتمام کامپوننت ها در GAC یا Global Assemblies Cache بار می شوند . در GAC اطلاعات اسمبلی ها قرار می گیرند ، در CLR دو ویژگی برای اسمبلی هایی که در GAC قرار می گیرند وجود دارد:

Side-by-side versioning : بدین معنی که ورژن های مختلف یک اسمبلی در کنار هم بدون تداخل وجود دارند.

Automatic QFE : اگر ورژن جدیدی از یک اسمبلی که با ورژن قبلی سازگاری دارد در GAC قرار گیرد ، CLRاین مساله را تشخیص می دهد و از آن لحظه به بعد از ورژن جدید استفاده می کند.

ورژن ها در CLR از ٤ قسمت Major.Minor.Revision.Build تشکیل شده اند . اگر در Major و Minor تغییری داده شود ، این بدین معناست که این ورژن از اسمبلی دیگر با ورژن های قبلی سازگاری ندارد.

انتقال : برنامه هایی که به زبان ویژوال بیسیک قدیمی نوشته می شوند در هنگام انتقال به کامپیوتر های دیگر دچار مشکلات فراوان می شوند . تمامی کامپوننت ها باید در رجیستری ثبت شوند . اگر ورژن جدیدی ایجاد شود امکان دارد برنامه هایی که از ورژن قدیمی استفاده می کردند را از کار بیندازد .

ولی در .NET. هم مساله ورژن ها تقریباً حل شده است و هم نحوه انتقال . کافی است در کامپیوتر مقصد .NET Framework. نصب شده باشد ، به راحتی با استفاده از دستور xcopy داس می توانید برنامه خود را انتقال بدهید!

شایان ذکر است که این نوشته صرفاً توضیح مختصری درباره هر کدام از قسمت های .NET. می دهد و برای هر کدام از این اجزا می توان یک مقاله مفصل نوشت .

مدیریت حافظه:

درباره مدیریت حافظه CLR بیشتر در زمینه VB بحث می کنیم.

مدیریت بهتر بر: Garbage Collection

Runtimeدر ویژوال بیسیک دارای سیستم خود کار برای Garbage Collection ها است . در ویژوال بیسیک ٦ ، runtimeبه شکل خودکار منابع را از آبجکت هایی که دیگر توسط برنامه ای استفاده نمی شوند می گیرد . برای مثال فرض کنید می خواهیم یک LOG فایل ایجاد کنیم ، برای این کار می توان از دو آبجکت Scripting.Stream و Scripting.FileSystemObject استفاده کرد . اگر این دو آبجکت را در تابعی به کار ببریم ، بعد از اتمام عملیات runtime منابع اختصاص داده شده به این دو آبجکت را می گیرد . اما مواردی وجود دارد که runtime به آسانی امکان تشخیص آبجکت و زمان بلااستفاده بودن آن را ندارد.

Cyclical References :

یکی از مهمترین زمانهایی که VB runtime نمی تواند تشخیص بدهد که آیا یک آبجکت به منابعی احتیاج دارد یا اینکه کارش به پایان رسیده زمانی است که در متن برنامه حالتی به نام Cyclical Referenceبوجود بیاید . به عنوان مثال اگر آبجکت A به صورت reference از آبجکت B استفاده کند و همچنین آبجکت B از . A

هر آبجکت COM مسئول نگهداری تعداد دفعاتی است که فعال شده . بدین صورت که هر بار از روی آن ساخته شود با AddRef و هربار که یک برنامه آن را رها کند با Release از IUnKnown interface شمارنده ای را یکی افزایش یا کاهش می دهد.

اگر آن شمارنده به صفر برسد آبجکت تمامی منابع خود را آزاد می کند ، اما اگر هنگامی که یک آبجکت COMغیر فعال شود ولی از Release استفاده نشود یک آبجکت بلا استفاده در حافظه می ماند VB 6 . runtimeاین کار را به صورت خودکار انجام می دهد ، اما هنگامی که دو آبجکت به هم اشاره می کنند بحث فرق می کند و runtime به شکل حالت ساده نمی تواند یک آبجکت را از حافظه بردارد . برای مثال به قطعه برنامه زیر توجه کنید .

```
'Class : CCyclicalRef
Dim m_objRef as Object

Public Sub Initialize(objRef as Object)
Set m_objRef = objRef
End Sub

Private Sub Class_Terminate()
Call Msgbox("Terminating.")
Set m_objRef = Nothing
End Sub
```

در کلاس CCyclicalRef متدی با نام Initialize تعریف شده است که در پارامتر های خود یک آبجکت را می پذیرد و از روی آن یک آبجکت دیگر می سازد . حال از کلاس تعریف شده در کد زیر استفاده می

کنیم.

```
Dim objA as New CCyclicalRef
Dim objB as New CCyclicalRef

Call objA.Initialize(objB)
Call objB.Initialize(objA)
Set objA = Nothing
Set objB = Nothing
```

ابتدا دو نمونه از روی کلاس CCyclicalRef با نامهای objA و objB ساختیم ، حال هر دوی این آبجکت ها یک بار ساخته شده اند و شمارنده آنها یک است . با استفاده از متد Initialize هر دوی آبجکت ها ، آدرس حافظه یکدیگر را مبادله می کنند و به این شکل از روی هر دوی آنها یک بار دیگر ساخته می شود و شمارنده هر دو ، ۲ می شود . شماره یک مربوط به خود برنامه است و شماره دوم مربوط به آبجکت هایی که به هم اشاره می کنند . سپس هر دوی آنها را برابر nothing سِت تا آنها را از حافظه حذف کنیم . در اینجا شمارنده هر دو یکی کم و برابر یک می شود . بنابراین برنامه terminate شده است ولی هنوز آبجکت در حافظه وجود دارد.

Garbage Collector در: CLR

روش (Garbage Collector) GC در CLR با VB 6 Runtime تفاوت دارد . در روش جدید که خودِ CLR و GCمسئول اجرای آن هستند ، آخرین باری که از nothing استفاده می شود ولی هنوز آبجکت در حافظه مانده باشد ، GCاین وضعیت را درک می کند و آن آبجکت را بعد از مدتی از حافظه خارج می

کند . آبجکت هایی هستند که مانند COM مسئول غیر فعال کردن خود نیستند ، GCآنها را نیز تشخیص داده و اگر برنامه ای دیگر از آنها استفاده نکند ، از حافظه حذف می شوند .

Finalize :

GCمتد Object.Finilize را قبل از اینکه هر آبجکت را از حافظه حذف کند صدا می زند . این متد می تواند در هر زمانی بعد از اینکه برنامه دیگر از آبجکت استفاده نکرد صدا زده شود ، لذا در هنگام استفاده از این متد باید نکته را در نظر گرفت . در .NET. بهتر است قبل از اینکه یک آبجکت بوسیله nothing از بین برود بهتر است از متد Dispose یا Close استفاده شود تا سریعاً از حافظه حذف شود .

روش سریعتر تخصیص حافظه به آبجکت ها

هر زمان که یک برنامه یک آبجکت بسازد حافظه ای به آن اختصاص می یابد ، آن حافظه در virtual memoryاست که برای هر برنامه اختصاص می یابد و به آن heap گفته می شود CLR . مفهومی به نام Managed Heap دارد بدین معنی که آبجکت ها را مدیریت می کند و آنها را در یکHeap ِ مدیریت شده قرار می دهد و CLR مسئول حفاظت آنهاست . از مزایای این روش این است که راندمان سرعت اختصاص دادن حافظه بالا می باشد . به طور کلی وقتی کُد های مدیریت نشده در Heap های مدیریت نشده قرار می گیرند ، به دنبال جایی در حافظه می گردند که بتوانند خود را در آن قرار دهند . در CLR آبجکتی که ساخته می شود همیشه در بالای آخرین آبجکتی که ساخته شده در heap قرار می گیرد . به یک مثال توجه کنید:

ابتدا CLR حافظه برای آبجکت های A ، Bو C بر روی Heap در نظر می گیرد . سپس آبجکت B از حافظه حذف می شود ، در ادامه نیز برنامه یک آبجکت دیگری به نام D می سازد و آن را بر روی آخرین آبجکتی که ساخته شده بود یعنی C قرار می دهد . این روش برای اختصاص حافظه سریع است ، اما اگر CLR به انتهای heap برسد چه عملی باید انجام دهد . همان طور که مشاهده کرید این روش به

شکل افزایشی آبجکت ها را در حافظه قرار می دهد ، وقتی دیگر CLR نتواند از حافظه استفاده کند GC را فرا می خواند GC . هم فضا هایی مانند B را کاملاً آزاد می کند و هم آبجکت ها فشرده می کند و در پایین heap قرار می دهد تا بالای heap آزاد باشد .

Namespaces

در .NET framework. برای مشخص کردن آبجکت ها از namespace استفاده می شود namespace . تلفیقی است از نام آبجکت ها و اسمبلی آنها بدین شکل که نام های مختلف را از اسمبلی های مختلف بدست آورده و همه را تحت یک namespace برای استفاده آماده می کند . برای مثال system اصلی ترین پدر در این سلسله مراتب است ، یکی از فرزندانش IO است که خود دارای فرزندان زیادی در شاخه خودش است namespace . ها قابلیت وراثت ندارند و همان طور که گفته شد فقط اسامی آبجکت ها نمایش می دهند . برای مثالnamespace ِ Microsoft.VisualBasic.Compatibility برای استفاده از ویژوال بیسیک قدیمی در ویژوال بیسیک جدید است و دارای مقدار زیادی از توابع و سابروتین های بیسیک قبل می باشد . هر کاربر در .NET. می تواند namespace ایجاد و از آن دربرنامه های مختلف استفاده کند . برای استفاده از namespace در کد نویسی می توان برای هر قسمتی از یک درخت یک Alias تعریف کرد ، تعریف alias برای دو هدف می باشد ، اول کوتاه کردن اسم های طولانی و دوم جلوگیری از تداخل ، برای مثال text در چند namespace وجود دارد ، اگر فقط به گفتن نام text اکتفا شود ممکن است کامپایلر نتواند تشخیص دهد که باید از کدام namespace استفاده کند .

منابع:

Professional VB.NET , 2001 , Wrox Press

MSDN Library

Visual Studio .NET Beta II & I

www.Microsoft.com

MSDN.Microsoft.com

www.GotDotNet.com

Microsoft .NET Shows

.NET Framework SDK Documentations