

کتاب آموزش جامع

زبان برنامه نویسی C#

اسلام احمد زاده

هرگونه انتقاد و پیشنهاد در مورد مطالب کتاب را با شماره ۰۹۱۷۷۱۱۲۱۶۱ در میان بگذارید تا در

نسخه های بعدی اصلاحات لازم صورت گیرد. با تشکر

این کتاب الکترونیکی رایگان است و هرگونه نسخه برداری از مطالب آن بصورت کلی یا جزئی مجاز

میباشد. این کتاب در واقع هدیه ای است به هموطنان عزیز.

(من اسلام احمد زاده در زمان نوشتن این کتاب استاد آموزشکده فنی باهنر شیراز و موسسه غیر انتفاعی زند شیراز میباشم و دست تمام دانشجویانی را میبوسم که با رفتار خود به من بزرگ منشی را یاد داده اند و از تمام دانشجویانی که احساس میکنند در نمره دادن به آنها ظلم کرده ام عذر خواهی میکنم.)

فهرست مطالب

۱-۱-۱ مروری بر ساختار. NET ۱۳	۳-۲-۱-۱ دستورات ۵۸if
۱-۱-۱-۱ NET. - و استانداردهای ۱۴CLI	۳-۲-۲-۲ بلوک های تک دستوری ۵۹if
۱-۱-۲ CLR ۱۵	۳-۲-۳-۳ ارزیابی کوتاه ۶۲
۱-۲-۱-۱ کامپایل کردن کد. NET ۱۵	۳-۲-۴-۴ دستورات ۶۳if.. else
۱-۲-۲-۱ CTS ۱۷	۳-۲-۵-۵ دستورات if تو در تو ۶۴
۱-۲-۳-۱ اسمبلی ها ۱۸	۳-۲-۶-۶ دستورات ۶۵switch
۱-۲-۳-۲ FCL ۲۲	۳-۲-۸-۸ دستورات switch روی رشته ها ۶۸
۱-۴-۱ کار با چارچوب NET و ۲۵SDK	۳-۳-۳-۳ دستورات تکرار ۶۹
۱-۴-۱-۱ بروزآوری چارچوب. NET ۲۵	۳-۳-۱-۱ ایجاد حلقه ها با ۶۹goto
۱-۴-۲-۲ ابزار چارچوب. NET ۲۵	۳-۳-۳-۳ حلقه ۷۱do... while
۱-۴-۳-۳ ابزار پیکربندی چارچوب ۲۹	۳-۳-۴-۴ حلقه ۷۲for
۱-۵-۱ فهم کامپایلر ۳۱C#	۴-۳-۴-۴ خلاصه ۷۷
۱-۵-۱-۱ محل کامپایلر ۳۱	۴-۱-۱-۱ ایجاد مدل ها ۸۰
۱-۵-۲-۲ کامپایل کردن از طریق خط فرمان ۳۱	۴-۲-۲-۲ کلاس ها و اشیاء ۸۰
۱-۶-۱ خلاصه ۳۳	۴-۳-۲-۲-۲ تعریف یک کلاس: ۸۱
۱-۲-۱-۲ چیدمان یک برنامه ۳۶C#	۴-۳-۴-۴ روابط کلاس: ۸۲
۱-۱-۲-۱ تذکرات عمومی برنامه نویسی ۳۷C#	۴-۵-۱-۱ ارکان سه گانه ی برنامه نویسی شی گرا ۸۲
۱-۲-۲-۲ انواع داده اولیه ۳۹	۴-۵-۱-۱-۱ کپسوله کردن ۸۳
۱-۳-۲-۲ عملگرهای ریاضی، منطق و شرطی ۴۲	۴-۵-۲-۲-۲ تخصص ۸۳
۱-۳-۲-۲ عملگرهای ریاضی ۴۲	۴-۵-۳-۲-۲ چندریختی ۸۴
۱-۳-۲-۲ عملگرهای شرطی و رابطه ای ۴۳	۴-۶-۲-۲ تحلیل و طراحی شی گرا ۸۴
۱-۴-۲-۲ راهنماهای پیش پردازش ۴۴C#	۴-۷-۲-۲ خلاصه ۸۴
۱-۴-۲-۲ کامپایل شرطی ۴۴	۴-۱-۲-۲ تعریف کلاس ۸۷
۱-۴-۲-۲ راهنماهای تشخیص ۴۵	۴-۱-۱-۱ نمونه سازی اشیاء ۸۸
۱-۵-۲-۲ نوع داده ی شمارشی ۴۵	۴-۱-۲-۲ ایجاد کلاس ۸۹Time
۱-۵-۲-۲ کار با نوع داده شمارشی ۴۶	۴-۳-۱-۱ معرف های دسترسی ۹۰
۱-۵-۲-۲ متدهای ۴۷System.Enum	۴-۲-۲-۲ آرگومان های متد ۹۱
۱-۵-۲-۲ انواع شمارشی و flag های بیتی ۴۷	۴-۳-۲-۲ سازنده ها ۹۲
۱-۶-۲-۲ انواع داده ی مقداری و ارجاعی ۴۸	۴-۴-۲-۲ مقدار دهنده های اولیه ۹۳
۱-۶-۲-۲ System.Object - و ۴۸System.ValueType	۴-۵-۲-۲ کلمه کلیدی ۹۴this
۱-۶-۲-۲ تخصیص حافظه برای انواع داده مقداری و ارجاعی ۴۸	۴-۶-۲-۲ اعضای نمونه و ایستا ۹۵
۱-۶-۲-۲ جعبه بندی ۴۹	۴-۱-۶-۲ احضار متدهای ایستا ۹۶
۱-۷-۲-۲ فضاهای نامی ۵۰	۴-۲-۶-۲ کاربرد فیلدهای ایستا ۹۸
۱-۷-۲-۲ دستور ۵۱using	۴-۷-۲-۲ خراب کردن اشیاء ۹۹
۱-۷-۲-۲ اسامی مستعار فضای اسمی ۵۲	۴-۸-۲-۲ تخصیص حافظه ۱۰۱
۱-۸-۲-۲ کنسول ۵۳I/O	۴-۹-۲-۲ خلاصه ۱۰۴
۱-۹-۲-۲ خلاصه ۵۴	۴-۱۰-۲-۲ فصل ششم ۱۰۶
۱-۳-۲-۲ دستورات انشعاب غیرشرطی ۵۷	۴-۱۰-۲-۲ وراثت و چند ریختی ۱۰۶
۱-۳-۲-۲ دستورات انشعاب شرطی ۵۸	۴-۱-۶-۲ تخصیص و تعمیم ۱۰۶

۱-۵-کلاس SortedList ۱۵۳

۱-۶-کلاس BitArray ۱۵۴

۱-۷-مقایسه‌ی آرایه‌ها و کلکسیون‌ها ۱۵۷

۱-۸-کاربرد کلاس‌های کلکسیون برای بازی کارت‌ها ۱۵۷

۱-۹-خلاصه ۱۵۷

۱-۱۱-تعریف کل ۱۵۸

۱-۲-پیاده‌سازی کل‌ها ۱۶۰

۱-۱۱-۳-اعمال کردن کل‌ها ۱۶۱

۱-۱۱-۴-انواع داده کل چندگانه ۱۶۱

۱-۱۱-۵-محدودیت‌های کل ۱۶۳

۱-۱۱-۵-۱-محدودیت‌های مشتق ۱۶۴

۱-۱۱-۵-۲-محدودیت سازنده ۱۶۶

۱-۱۱-۵-۳-محدودیت نوع مقداری/ارجاعی ۱۶۶

۱-۱۱-۶-کلاس‌ها و کلکسیون‌های کل در FCL ۱۶۶

۱-۱۱-۶-۱-مروری بر کلکسیون‌های کل ۱۶۷

۱-۱۱-۷-خلاصه ۱۷۰

۱-۱۲-۱-اندیس‌گذار ۱۷۲

۱-۱۲-۱-۱-مثال بدون کاربرد اندیس‌گذار ۱۷۲

۱-۱۲-۱-۲-کاربرد اندیس‌گذارها در مثال قبلی ۱۷۳

۱-۱۲-۲-مقایسه آرایه‌ها و اندیس‌گذارها ۱۷۴

۱-۱۲-۳-خصوصیات آرایه‌ها و اندیس‌گذارها ۱۷۵

۱-۱۲-۴-اندیس‌گذارها در واسط‌ها ۱۷۶

۱-۱۲-۵-خلاصه ۱۷۷

۱-۱۳-۱-کاربرد کلمه‌ی کلیدی operator ۱۷۹

۱-۱۳-۲-پشتیبانی دیگر زبان‌های .NET ۱۷۹

۱-۱۳-۳-ایجاد عملگرهای مفید ۱۷۹

۱-۱۳-۴-عملگرهای دوتایی منطقی ۱۸۰

۱-۱۳-۵-عملگر تساوی ۱۸۰

۱-۱۳-۶-عملگرهای تبدیل ۱۸۰

۱-۱۳-۷-خلاصه ۱۸۴

۱-۱۴-۱-برنامه‌نویسی یک فرم ویندوز ۱۸۶

۱-۱۴-۱-۱-ایجاد دستی یک برنامه کاربردی ویندوز ۱۸۶

۱-۱۴-۲-کلاس‌های کنترل در Windows.Forms ۱۸۸

۱-۱۴-۲-۱-کلاس Control ۱۸۸

۱-۹-خصوصیات Control ۱۸۹

۱-۱۴-۲-۲-کار با کنترل‌ها ۱۹۰

اندازه و موقعیت ۱۹۰

چگونه یک کنترل را لنگر بیاندازیم و بچسبانیم ۱۹۱

ترتیب Tab و کانون ۱۹۲

طی کردن همه کنترل‌های روی یک فرم ۱۹۳

۱-۱۴-۲-۳-رویدادهای Control ۱۹۴

اداره کردن رویدادهای ماوس ۱۹۴

اداره کردن رویدادهای صفحه کلید ۱۹۶

۲-۶-۲-وراثت ۱۰۸

۲-۶-۱-پیاده‌سازی وراثت ۱۰۹

۲-۲-۲-فراخوانی سازنده‌های کلاس پایه ۱۱۰

۲-۶-۳-کنترل دسترسی ۱۱۰

۳-۶-۳-چند ریختی ۱۱۱

۳-۶-۱-ایجاد انواع داده‌ی چندریختی ۱۱۱

۲-۳-۲-نسخه‌سازی با new و override ۱۱۴

۴-۶-کلاس‌های انتزاعی ۱۱۵

۵-۶-کلاس‌های مهرشده ۱۱۷

۶-۶-۶-ریشه‌ی همه کلاس‌ها (Object ۱۱۷)

۶-۷-خلاصه ۱۱۹

فصل هفتم ۱۲۰

متدهای داخلی ۱۲۰

۷-overload-کردن متدها ۱۲۰

۲-۷-کپسوله کردن داده‌ها با خصوصیات ۱۲۲

۲-۲-۱-معاون get ۱۲۴

۲-۲-۲-معاون set ۱۲۵

۳-۷-برگرداندن چندین مقدار ۱۲۵

۳-۷-۱-ارسال انواع داده‌ی مقداری بوسیله ارجاع ۱۲۶

۲-۳-۲-پارامترهای out و انتساب روشن ۱۲۸

۴-۷-خلاصه ۱۲۹

۱-۸-۱-کاربرد آرایه‌ها ۱۳۰

۱-۱-۱-اعلان آرایه‌ها ۱۳۱

۱-۸-۲-فهم مقادیر پیش فرض ۱۳۱

۳-۱-۸-دسترسی به عناصر آرایه ۱۳۲

۲-۸-دستور foreach ۱۳۳

۳-۸-مقداردهی اولیه عناصر آرایه ۱۳۴

۴-۸-کلید کلیدی params ۱۳۴

۵-۸-آرایه‌های چندبعدی ۱۳۵

۱-۵-۸-آرایه‌های مستطیلی ۱۳۵

۲-۵-۸-آرایه‌های ناهموار ۱۳۷

۶-۸-متدهای آرایه ۱۴۰

۷-۸-مرتب کردن آرایه‌ها ۱۴۰

۸-۸-خلاصه ۱۴۱

۱-۹-تعریف ساختارها ۱۴۴

۲-۹-ایجاد ساختارها ۱۴۵

۲-۹-۲-ساختارها به صورت انواع داده مقداری ۱۴۶

۲-۹-۲-ایجاد ساختارها بدون new ۱۴۶

۳-۹-خلاصه ۱۴۸

۱-۱۰-کلاس ArrayList ۱۵۰

۲-۱۰-کلاس Queue ۱۵۱

۳-۱۰-کلاس Stack ۱۵۲

۴-۱۰-کلاس Hashtable ۱۵۲

۱۹۸Form کلاس	۳-۱۴
۱۹۹-۱-تنظیم ظاهر یک فرم	۳-۱۴
کد ری فرم ۲۰۰	
شفافیت فرم ۲۰۰	
تنظیم اندازه و موقعیت فرم ۲۰۲	
۱۴-۳-۲-نمایش فرم‌ها ۲۰۳	
چرخه‌ی زندگی یک فرم ۲۰۳modeless	
ایجاد و نمایش فرم ۲۰۳	
۱۴-۳-۳-فعال سازی و غیرفعال سازی فرم ۲۰۴	
بستن فرم ۲۰۴	
۱۴-۳-۴-فعل و انفعال فرم‌ها - یک برنامه کاربردی نمونه ۲۰۵	
کد فرم اصلی ۲۰۶	
کد فرم جستجو ۲۰۶	
۱۴-۳-۵-فرم‌های مالک و ملک ۲۰۷	
فرم‌های ۲۰۸MDI	
ایجاد یک منو و فرم ۲۰۸MDI	
ایجاد یک منوی MDI با استفاده از ۲۱۰VS.NET	
۱۴-۴-۱-کار با منوها ۲۱۱	
۱۴-۴-۱-۱-خصوصیات ۲۱۱MenuItem	
۱۴-۴-۲-منوهای زمینه ۲۱۱	
۱۴-۴-۳-ساختن یک منوی زمینه ۲۱۲	
۱۴-۵-۱-اضافه کردن کمک به یک فرم ۲۱۳	
۱۴-۵-۱-۴-ToolTip-ها ۲۱۳	
۱۴-۵-۲-پاسخ به ۱F و دکمه ۲۱۴Help	
۱۴-۵-۳-کنترل ۲۱۶HelpProvider	
۱۴-۶-۱-وراثت فرم‌ها ۲۱۶	
۱۴-۶-۱-۱-ایجاد و استفاده یک کتابخانه از فرم‌ها ۲۱۶	
۱۴-۶-۲-کاربرد فرم ارث‌بری شده ۲۱۷	
۳۶-۳-۶-Override کردن رویدادها ۲۱۷	
۱۴-۶-۴-ایجاد فرم‌های ارث‌بری شده با ۲۱۸VS.NET	
۱۴-۷-خلاصه ۲۱۸	
۱-۱-۱-مطالعه کنترل‌های NET فرم‌های ویندوز ۲۲۰	
۱۵-۲-کلاس‌های Button، GroupBox، Panel و Label	
۲۲۲	
۱۵-۱-۲-کلاس ۲۲۲Button	
تنظیم ظاهر یک دکمه ۲۲۲	
اداره کردن رویدادهای ۲۲۳Button	
۱۵-۲-۲-کلاس ۲۲۳CheckBox	
تنظیم ظاهر ۲۲۳CheckBox	
۱۵-۳-۲-کلاس ۲۲۴RadioButton	
قراردادن دکمه‌های رادیویی در یک گروه ۲۲۴	
۱۵-۴-۲-کلاس ۲۲۶GroupBox	
۱۵-۲-۵-کلاس ۲۲۶Panel	
۱۵-۲-۶-کنترل ۲۲۶FlowLayoutPanel	
۱۵-۲-۷-کنترل ۲۲۷TableLayoutPanel	
۱۵-۲-۸-کلاس ۲۲۸Label	
۱۵-۳-۳-کنترل‌های PictureBox و ۲۲۸TextBox	
۱۵-۳-۱-کلاس ۲۲۸PictureBox	
۱۵-۲-۲-کلاس ۲۳۰TextBox	
کادرهای متنی و کاراکترهای بازگشت به سر سطر ۲۳۱	
۱۵-۴-کلاس‌های ۲۳۱CheckedListBox، ListBox و	
۲۳۲ComboBox	
۱۵-۴-۱-کلاس ۲۳۲ListBox	
اضافه کردن اقلام به یک ۲۳۲ListBox	
انتخاب و جستجوی اقلام در یک ۲۳۳ListBox	
سفارشی کردن ظاهر یک ۲۳۴ListBox	
۱۵-۴-۲-کنترل‌های دیگری از لیست: ۲۳۵ComboBox و	
۲۳۶CheckedListBox	
۱۵-۵-کلاس‌های ۲۳۷TreeView و ۲۳۷ListView	
۱۵-۵-۱-کلاس ۲۳۷ListView	
ایجاد یک شی ۲۳۷ListView	
تعریف ظاهر شی ۲۳۷ListView	
تنظیم سرآیندهای ستون ۲۳۸	
ایجاد قلم داده‌های ۲۳۹ListView	
تعیین آیکون‌ها ۲۳۹	
کار با کنترل ۲۴۰ListView	
طی کردن همه قلم داده‌ها یا قلم داده‌های انتخاب شده ۲۴۰	
تشخیص قلم داده انتخاب شده جاری ۲۴۱	
مرتب‌سازی قلم داده‌های یک کنترل ۲۴۱ListView	
۱۵-۵-۲-کلاس ۲۴۲TreeView	
کلاس ۲۴۲TreeNode	
اضافه کردن و حذف کردن گره‌ها ۲۴۳	
طی کردن همه گره‌ها در یک ۲۴۴TreeView	
تشخیص گره انتخاب شده ۲۴۴	
یک مثال ۲۴۴TreeView با کاربرد انعکاس ۲۴۴	
۱۵-۶-کلاس‌های ۲۴۵Timer، ProgressBar و ۲۴۵StatusStrip	
۲۴۶	
ایجاد یک ۲۴۷StatusStrip	
۱۵-۷-۱-ایجاد کنترل‌های سفارشی ۲۴۸	
۱۵-۷-۱-بسط یک کنترل ۲۴۹	
۱۵-۷-۲-ایجاد یک UserControl سفارشی ۲۴۹	
یک مثال از ۲۴۹User Control	
استفاده از ۲۵۰UserControl سفارشی ۲۵۰	
کار با ۲۵۱UserControl در زمان طراحی ۲۵۱	

۱۵-۸- استفاده از کشیدن و انداختن بوسیله کنترل‌ها ۲۵۲

مروری بر کشیدن و انداختن ۲۵۲

مسئولیت‌های کنترل مبدأ ۲۵۴

مسئولیت‌های کنترل هدف ۲۵۴

۱۵-۹- کاربرد منابع ۲۵۶

۱۵-۹-۱- کار با فایل‌های منبع ۲۵۷

ایجاد رشته‌های منبع از روی یک فایل متنی ۲۵۷

کاربرد کلاس ResourceWriter برای ایجاد یک فایل

. Resources ۲۵۸

کاربرد کلاس ResourceManager برای دستیابی به منابع

۲۵۸

کاربرد کلاس ResXResourceWriter برای ایجاد یک فایل

. resx ۲۵۹

کاربرد کلاس ResXResourceReader برای خواندن یک

فایل . resx ۲۵۹

تبدیل یک فایل .resx به یک فایل . resources ۲۵۹

۱۵-۹-۲- VS.NET و منابع ۲۵۹

کاربرد فایل‌های منبع برای ایجاد فرم‌های محلی ۲۶۰

محلی کردن منابع با استفاده از VS.NET ۲۶۰

تعیین منابع محلی در زمان اجرا ۲۶۱

ایجاد یک اسمبلی پیرو بدون VS.NET ۲۶۱

۱۵-۱۰- خلاصه ۲۶۱

۱۶-۱- کادرمحاوره ای MessageBox ۲۶۳

دکمه‌های موجود برای کادر پیغام: ۲۶۴

تنظیم دکمه‌ی پیش فرض: ۲۶۵

گزینه‌های مختلف کادر پیغام ۲۶۵

حالت‌های مختلف استفاده از متد Show ۲۶۵

نمونه‌هایی از کادر پیغام ۲۶۷

۱۶-۲- کنترل OpenFileDialog ۲۶۸

خصوصیت‌های کنترل OpenFileDialog ۲۶۹

متدهای OpenFileDialog ۲۷۱

استفاده از کنترل OpenFileDialog ۲۷۱

بررسی نکات مهم برنامه ۲۷۳

۱۶-۳- کنترل SaveFileDialog ۲۷۴

خصوصیت‌های کنترل SaveFileDialog ۲۷۴

متدهای کنترل SaveFileDialog ۲۷۵

استفاده از کنترل SaveFileDialog ۲۷۵

۱۶-۴- کنترل FontDialog ۲۷۷

خصوصیت‌های کنترل FontDialog ۲۷۷

متدهای کنترل FontDialog ۲۷۸

استفاده از کنترل FontDialog ۲۷۸

۱۶-۵- کنترل ColorDialog ۲۸۰

خصوصیت‌های کنترل ColorDialog ۲۸۱

استفاده از کنترل ColorDialog ۲۸۲

۱۶-۶- کنترل PrintDialog ۲۸۳

خصوصیت‌های کنترل PrintDialog ۲۸۴

استفاده از کنترل PrintDialog ۲۸۴

۱۶-۶-۱- کلاس PrintDocument ۲۸۴

خصوصیات کلاس PrintDocument ۲۸۴

چاپ یک سند ۲۸۵

بررسی مثال چاپ ۲۸۸

۱۶-۷- کنترل FolderBrowserDialog ۲۹۱

خصوصیت‌های کنترل FolderBrowser ۲۹۲

استفاده از کنترل FolderBrowser ۲۹۲

۱۶-۸- خلاصه ۲۹۴

۱۷-۱- تعریف و پیاده‌سازی یک واسط ۲۹۷

۱۷-۱-۱- پیاده‌سازی بیش از یک واسط ۲۹۹

۱۷-۱-۲- بسط دادن واسط‌ها ۲۹۹

۱۷-۲- دستیابی به متدهای واسط ۳۰۳

۱۷-۲-۱- قالب‌بندی به یک واسط ۳۰۳

۱۷-۲-۲- عملگر Fis ۳۰۴

۱۷-۲-۳- عملگر as ۳۰۶

۱۷-۲-۴- مقایسه عملگرهای as و Vis ۳۰۷

۱۷-۲-۵- مقایسه کلاس انتزاعی و واسط ۳۰۷

۱۷-۳- override کردن پیاده‌سازی‌های واسط ۳۰۷

۱۷-۴- پیاده‌سازی صریح واسط ۳۱۰

۱۷-۴-۲- پنهان کردن اعضا ۳۱۲

۱۷-۴-۳- دستیابی به کلاس‌های مهرشده و انواع داده‌ی

مقداری ۳۱۲

۱۷-۵- خلاصه ۳۱۴

۱۸-۱- اعلان و کاربرد نماینده‌ها ۳۱۶

۱۸-۱-۱- سناریوی کارخانه اتوماتیک ۳۱۷

۱۸-۱-۲- پیاده‌سازی کارخانه بدون کاربرد نماینده‌ها ۳۱۷

۱۸-۱-۳- پیاده‌سازی کارخانه با استفاده یک نماینده ۳۱۷

۱۸-۱-۴- متدها و نماینده‌های بی‌نام ۳۱۹

۱۸-۲- اعلان یک رویداد ۳۲۱

۱۸-۲-۱- متعهد شدن به یک رویداد ۳۲۲

۱۸-۲-۲- غیر متعهد شدن از یک رویداد ۳۲۲

۱۸-۲-۳- رها کردن یک رویداد ۳۲۲

۱۸-۳- رویدادهای GUI ۳۲۳

۱۹-۱- مقدمه ۳۲۵

۱۹-۲- کلاس System.Exception ۳۲۶

۱۹-۳- کدنویسی برای اداره کردن استثناها ۳۲۷

۱۹-۴- چگونه یک کلاس استثناء سفارشی ایجاد کنیم؟ ۳۲۹

۱۹-۵- استثناء‌های اداره نشده ۳۳۲

۲۰-۱- کاراکترها و یونیکد ۳۳۳

- ۲۰-۱-۱-یونیکد ۳۳۴
- ۲۰-۲-۱-کار با کاراکترها ۳۳۵
- انتساب یک مقدار به یک نوع داده‌ی char ۳۳۵
- تبدیل یک مقدار Char به یک مقدار عددی ۳۳۵
- ۲۰-۳-۱-کاراکترها و محلی کردن ۳۳۵
- ۲۰-۴-۱-کاراکترها و دسته‌های یونیکد آنها ۳۳۶
- ۲۰-۲-کلاس رشته ۳۳۷
- ۲۰-۱-۱-ایجاد رشته‌ها ۳۳۷
- ۲۰-۲-۲-داخل کردن رشته‌ها ۳۳۸
- ۲۰-۲-۳-مروری بر عملیات رشته‌ها ۳۳۹
- ۲۰-۳-مقایسه‌ی رشته‌ها ۳۳۹
- ۲۰-۳-۱-کاربرد String.Compare ۳۴۰
- ۲۰-۳-۲-کاربرد String.CompareOrdinal ۳۴۱
- ۲۰-۴-جستجو، تغییر و کدگذاری محتوای یک رشته ۳۴۲
- ۲۰-۱-۴-جستجوی محتویات یک رشته ۳۴۲
- ۲۰-۲-۴-جستجوی رشته‌ی جانشین‌دار ۳۴۳
- ۲۰-۳-۴-تبدیل رشته‌ها ۳۴۳
- ۲۰-۴-۴-کدگذاری رشته ۳۴۵
- ۲۰-۵-کلاس StringBuilder ۳۴۶
- ۲۰-۵-۲-مقایسه‌ی StringBuilder و الحاق رشته ۳۴۷
- ۲۰-۶-فرمت‌دهی مقادیر عددی، تاریخ و زمان ۳۴۸
- ۲۰-۱-۶-ساختن یک عنصر فرمت ۳۴۸
- ۲۰-۲-۶-فرمت‌دهی مقادیر عددی ۳۴۹
- ۲۰-۳-۶-فرمت‌دهی تاریخ و زمان ۳۵۰
- ۲۰-۴-۶-تاریخ‌ها و فرهنگ ۳۵۲
- کلاس‌های DateTimeFormatInfo و
- ۳۵۲NumberFormatInfo
- ۲۰-۷-عبارات منظم ۳۵۳
- ۲۰-۱-۷-کلاس Regex ۳۵۴
- ۲۰-۲-۷-ایجاد عبارات منظم ۳۵۷
- ۲۰-۳-۷-مثال‌هایی از کاربرد عبارات منظم ۳۶۱
- ۲۰-۸-خلاصه ۳۶۲
- ۲۱-۱-مدیریت سیستم فایل ۳۶۳
- ۲۱-۱-۱-کلاس‌های مربوط به پوشه‌ها و فایل‌ها در .NET ۳۶۴
- ۲۱-۲-۱-کلاس Path ۳۶۶
- ۲۱-۳-۱-مثال File Browser ۳۶۶
- ۲۱-۲-انتقال، کپی و حذف فایل‌ها ۳۷۰
- ۲۱-۱-۲-مثال FilePeopertiesAndMovement ۳۷۰
- ۲۱-۲-۲-بررسی کد برنامه FilePropertiesAndMovment ۳۷۱
- ۲۱-۳-خواندن و نوشتن در فایل‌ها ۳۷۴
- ۲۱-۳-۱-خواندن یک فایل ۳۷۴
- ۲۱-۳-۲-نوشتن به یک فایل ۳۷۵
- ۲۱-۳-۳-جریان‌های بافر شده ۳۷۸
- کلاس FileStream ۳۷۸
- ۲۱-۳-۶-خواندن و نوشتن در فایل‌های متنی ۳۸۰
- کلاس StreamReader ۳۸۰
- متدهای کلاس StreamReader ۳۸۰
- کلاس StreamWriter ۳۸۱
- متدهای کلاس StreamWriter ۳۸۱
- ۲۱-۳-۷-رمزنگاری با کلاس CryptoStream ۳۸۱
- ۲۱-۴-خواندن اطلاعات درایو ۳۸۱
- ۲۱-۵-امنیت فایل ۳۸۳
- ۲۱-۱-۵-خواندن ACLهای یک فایل ۳۸۳
- ۲۱-۵-۲-اضافه کردن و حذف ACLهای یک فایل ۳۸۴
- ۲۱-۶-خلاصه ۳۸۵
- ۲۲-۱-مقدمه ۳۸۷
- اشیای موجود در Access ۳۸۸
- ۲۲-۳-مقید کردن داده‌ها ۳۹۴
- خلاصه ۴۰۰
- ۲۲-۲-ADO.NET ۴۰۲
- ۲۲-۱-۲-فضای نامی Data ۴۰۳
- ۲۲-۲-۲-کلاس SqlConnection: ۴۰۴
- ایجاد بخش‌های مختلف Fring ConnectionSt ۴۰۴
- ۲۲-۳-۲-کلاس SqlCommand ۴۰۶
- خاصیت Connection ۴۰۶
- خاصیت CommandText ۴۰۶
- خاصیت Parameters ۴۰۷
- متد ExecuteNonQuery ۴۰۸
- ۲۲-۴-۲-کلاس SqlDataAdapter ۴۰۹
- خاصیت SelectCommand ۴۰۹
- تنظیم خاصیت SelectCommand با استفاده از دستور SQL
- ۴۱۰
- تنظیم خاصیت SelectCommand با استفاده از پروسیجر
- ذخیره شده ۴۱۱
- استفاده از CommandBuilder برای ایجاد دستورات SQL
- دیگر ۴۱۱
- متد Fill ۴۱۲
- ۲۲-۵-۲-کلاس DataSet ۴۱۳
- ۲۲-۶-۲-کلاس DataView ۴۱۴
- خاصیت Sort ۴۱۵
- خاصیت RowFilter ۴۱۵

۳۰-۲-۱- ایجاد یک اسمبلی چند ماژولی ۵۶۶

۳۰-۲-۲- آزمایش اسمبلی ۵۷۰

۳۰-۲-۳- اسمبلی‌های خصوصی ۵۷۲

۳۰-۲-۴- اسمبلی‌های اشتراکی ۵۷۲

۳۰-۲-۵- پایان جهنم ۵۷۳DLL

۳۰-۳- نسخه‌ها ۵۷۳

۳۰-۳-۱- اسامی قوی ۵۷۳

۳۰-۳-۲- GAC ۵۷۴

۳۰-۳-۳- ایجاد یک اسمبلی اشتراکی ۵۷۴

۳۰-۳-۴- اسمبلی‌های مورد نیاز دیگر ۵۷۶

ضمیمه ۱ ۵۷۸

نصب ویزوال #C ۵۷۸ ۲۰۰۵

محیط توسعه‌ی #VC ۵۸۰ ۲۰۰۵

۲۷-۱-۰۹/I- ناهمگام ۵۲۲

مثال سرویس گیرنده‌ی ناهمگام ۵۲۵

مثال سرویس دهنده‌ی Tcp ناهمگام ۵۲۷

۲۸-۱- مقدمه ۵۲۹

۲۸-۱-۱- چه زمانی صف‌بندی پیام را به کار ببریم؟ ۵۳۰

۲۸-۱-۲- ویژگی‌های صف‌بندی پیام ۵۳۱

۲۸-۱-۳- محصولات صف بندی پیام ۵۳۱

۲۸-۲- معماری ۵۳۲MQ

۲۸-۲-۱- پیام‌ها ۵۳۲

۲۸-۲-۲- صف پیام ۵۳۳

ایجاد صف‌های پیام ۵۳۴

خصوصیات صف پیام ۵۳۴

۲۸-۳- برنامه‌نویسی صف‌بندی پیام ۵۳۵

۲۸-۳-۱- ایجاد یک صف پیام ۵۳۶

۲۸-۳-۲- برنامه کاربردی CourseOrder ۵۳۶

کتابخانه‌ی کلاس CourseOrder ۵۳۶

ارسال کننده‌ی پیام تکلیف درس ۵۳۸

ارسال پیام‌های قابل ترمیم و اولویت‌دار ۵۳۹

دریافت کننده‌ی پیام تکلیف درس ۵۴۰

صف‌های تصدیق ۵۴۳

صف‌های جواب ۵۴۳

صف‌های تراکنشی ۵۴۴

۲۸-۴- نصب MessageQueue ۵۴۵

۲۸-۵- خلاصه ۵۴۵

۲۹-۱- وارد کردن کنترل‌های ActiveX ۵۴۶

۲۹-۲- وارد کردن یک کنترل به .NET ۵۴۹

۲۹-۲-۱- وارد کردن یک کنترل ۵۴۹

۲۹-۲-۲- وارد کردن کنترل به صورت دستی ۵۵۰

۲۹-۲-۳- اضافه کردن کنترل به فرم ۵۵۱

۲۹-۳- وارد کردن قطعات Com ۵۵۲

۲۹-۳-۱- کدنویسی برنامه‌ی ComTestForm ۵۵۳

۲۹-۳-۲- وارد کردن COM DLL به .NET ۵۵۴

۲۹-۳-۳- وارد کردن کتابخانه نوع داده ۵۵۴

۲۲-۳-۵- ایجاد یک برنامه آزمایشی ۵۵۴

۲۹-۴- صادر کردن قطعات .NET ۵۵۸

۲۲-۴-۱- ایجاد یک کتابخانه‌ی نوع داده ۵۶۰

۲۹-۵- P/Invoke ۵۶۱

۳۰-۱- فایل‌های PPE ۵۶۴

۳۰-۱-۱- فراداده ۵۶۴

۳۰-۱-۲- محدوده‌های امنیت ۵۶۵

۳۰-۱-۳- اظهارنامه‌ها ۵۶۵

۳۰-۲- اسمبلی‌های چند ماژولی ۵۶۶

فصل یک

مقدمه‌ای بر NET و C#

در این فصل یاد خواهید گرفت:

- مروری بر چارچوب NET: معماری و ویژگی‌ها
 - CLR: مروری بر کارهای قابل انجام توسط قسمت زمان اجرای چارچوب.
 - کامپایلر NET^۲ JIT. (فقط در لحظه بارگذاری اسمبلی‌ها و تایید کد لازم است).
 - CTS^۳ و CLS^۴: قوانینی که سازگاری CLR و تعامل بین زبانی را مدیریت می‌کنند.
 - اسمبلی‌ها: نگاهی کوتاه به ساختار یک اسمبلی، فلسفه‌ی پشت آن و تفاوت مابین اسمبلی‌های خصوصی و اشتراکی.
 - FCL^۵: این کتابخانه صدها کلاس پایه‌ی گروه‌بندی شده در فضاها‌ی نامی منطقی را فراهم می‌کند.
 - ابزار توسعه: بوسیله‌ی NET چندین ابزار با هدف توسعه‌ی کد فراهم شده است. این ابزار شامل ILdasm برای نمایش کد، WinCV برای مشاهده خصوصیات یک کلاس و ابزار دیگر پیکربندی چارچوب.
 - کامپایل کردن و اجرای برنامه‌های C#: کاربرد کامپایلر C# از خط فرمان و گزینه‌هایی برای پیکربندی یک برنامه.
- استفاده‌ی کارای یک زبان، یادگیری گرامر و ویژگی‌های زبان را نیاز دارد. در حقیقت بخش اعظم منحنی یادگیری برای تکنولوژی جدید، به محیط برنامه‌نویسی مرتبط است. حرفه‌ای شدن در C# کافی نیست. معمار نرم‌افزار و توسعه دهنده‌ی موفق باید کتابخانه‌های اصیل کلاس و ابزار تولید آنها را بشناسد.
- از منظر برنامه‌نویسی NET، Platform شامل یک محیط اجرایی پیوند خورده به یک کتابخانه کلاس پایه است، که در این فصل CLR، FCL و طرز کار با NET Platform بررسی خواهد شد.

۱-۱ مروری بر ساختار NET

ساختار NET بصورت یک محیط مجتمع برای توسعه و اجرای برنامه‌های اینترنتی، برنامه‌های کاربردی ویندوز (میزکار) و حتی دستگاه‌های موبایل طراحی شده است. اهداف اصلی آن بصورت زیر است:

- فراهم ساختن یک محیط شی‌گرایی مابین دامنه‌ای از کاربردها.
- با فراهم ساختن این محیط، تداخل نسخه‌های DLL را کم کرده و پروسه توزیع و نصب کد را ساده می‌کند.

^۱ Common Language RunTime

^۲ Just In Time

^۳ Common Language Specification

^۴ Common Type System

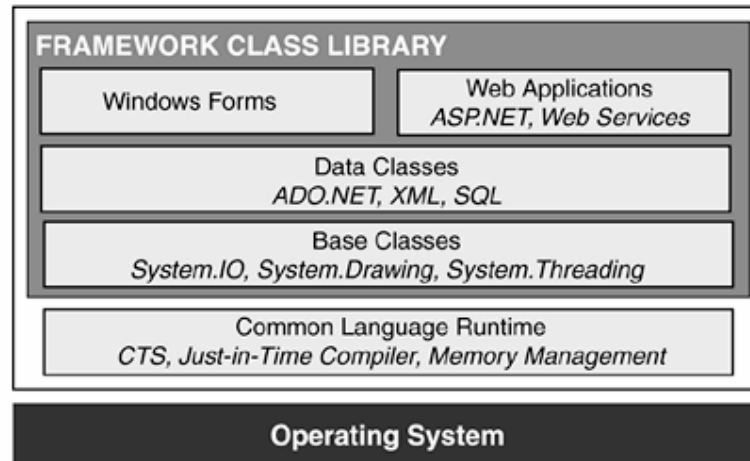
^۵ Framework Common Library

= یک محیط قابل حمل^۱ براساس استانداردهای تایید شده آماده می سازد تا بتوانند توسط هر سیستم عاملی میزبانی شوند. در حال حاضر C# و یک بخش اصلی از زمان اجرای NET به نام CLI^۲ بوسیله ECMA^۳ استاندارد شده اند.

= فراهم ساختن یک محیط مدیریت شده، که اجرای امن را به سادگی تایید می کند.

طراحان چارچوب NET برای رسیدن به این اهداف بزرگ روی یک معماری به توافق رسیدند که چارچوب را به دو بخش تجزیه می کند: CLR و FCL. شکل ۱-۱ آن را ارائه می کند.

شکل ۱-۱ معماری NET Framework.



CLR (پیاده سازی استاندارد CLI توسط میکروسافت) اجرای کد و همه کارهای تخصیص یافته به آن همچون کامپایل، مدیریت حافظه، امنیت، مدیریت ریسمان و ایمنی از نوع داده^۴، کد اجرا شده تحت نظر CLR (که کد مدیریت شده می نامند)، کد مدیریت نشده توسط CLR همچون COM و API را اداره می کند.

FCL قطعه اصلی دیگر می باشد. یک کتابخانه کد قابل استفاده مجدد (شامل کلاس ها، ساختارها و غیره)، که برای برنامه های اجرا شده تحت NET در دسترس است. همه زبان های NET این کتابخانه کلاس مشترک را استفاده می کنند. پس این مفاهیم در همه زبان های NET مشترک خواهد بود.

۱-۱-۱ NET و استانداردهای CLI

یک توسعه دهنده قبل از صرف زمان برای یادگیری NET و C# می پرسد: آیا این مهارت را می تواند به Platform های دیگر تبدیل کند؟ آیا محصول NET، میکروسافت فقط مختص سیستم عامل ویندوز است؟ یا آیا آن قابل حمل اجرایی است و آن برای حمل روی سیستم عامل های دیگر نیز پیاده سازی شده است؟ برای جواب دادن به این سئوال، فهمیدن رابطه ی مابین NET، C# و استانداردهای CLI ضروری است.

CLI یک محیط اجرایی مجازی مستقل از Platform را تعریف می کند. آن هیچ سیستم عاملی را تعیین نمی کند و برای لینوکس همانند ویندوز راحت است. بخش استاندارد مرکزی، تعریف یک CIL^۵ است که باید توسط کامپایلرهای مطیع CLI

^۱ Portable

^۲ Common Language Infrastructure

^۳ European computer Manufactures Association

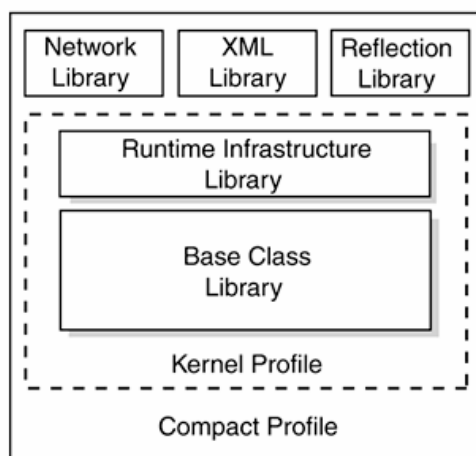
^۴ Type Safety

^۵ Common Intermediate Language

تولید شود. بخش دیگر سیستم نوع^۱ است که همه انواع داده‌ای پشتیبان شده توسط هر زبان مطیع CLI را تعریف می‌کند. این کد میانی به زبان اصلی سیستم عامل میزبان کامپایل می‌شود.

CLI استانداردهایی را برای زبان #C در بر دارد که بوسیله مایکروسافت توسعه و ارتقاء یافته‌اند. (Fortran, Pascal, Pythen, Defacto)

چارچوب NET ارائه شده در شکل ۱-۱، پیاده‌سازی مایکروسافت از استانداردهای CLI است. این پیاده‌سازی ویژگی‌های زیادی دارد که بوسیله معماری CLI مشخص می‌شوند. برای فهم بیشتر، آن را با معماری استاندارد های CLI شکل ۱-۲ مقایسه کنید.



شکل ۱-۲ معماری تعریف شده بوسیله مشخصه CLI

این کتاب پیاده‌سازی مایکروسافت را تشریح می‌کند. فرض بر آن است که می‌توانیم از کدهای قبلی نوشته شده نیز استفاده کنیم و کدهای پیاده‌سازی شده توسط NET به سیستم عامل دیگری منتقل نخواهد شد و محیط مجازی گفته شده نیز شفاف است.

۱-۲- CLR

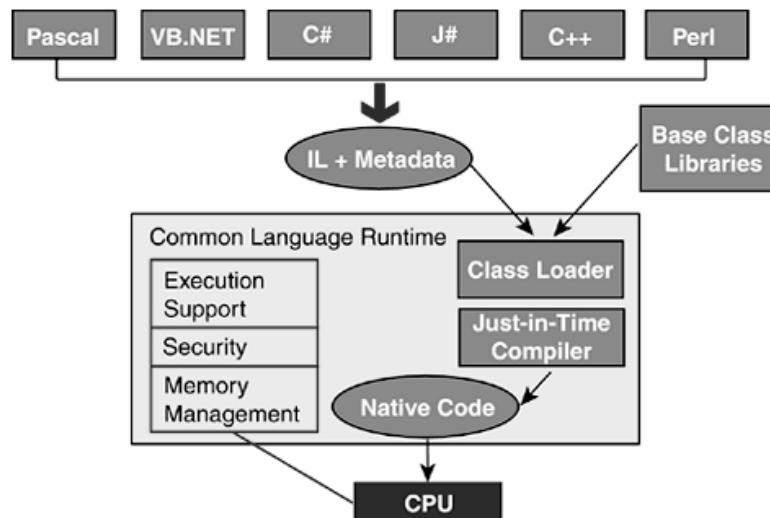
CLR تمام چرخه‌ی زندگی یک برنامه کاربردی را مدیریت می‌کند. آن کد را یافته و کامپایل می‌کند. کلاس‌های تخصیص یافته را بارگذاری می‌کند. اجرایش را مدیریت می‌کند و مدیریت اتوماتیک حافظه را مطمئن می‌سازد. آن ارتباط بین زبانی را پشتیبانی می‌کند، تا تعامل مابین کدهای نوشته شده در زبان‌های مختلف را مجاز دارد. این بخش کارکرد داخلی CLR را نشان می‌دهد. آن یک بحث عمقی نیست و فقط می‌خواهد شما را با اصطلاح آن آشنا سازد.

۱-۲-۱- کامپایل کردن NET.

کامپایلرهای مطیع CLR کدی تولید می‌کنند که کد هدف زمان اجرا بوده و برای یک CPU خاص پیشنهاد شده است. این کد به نام CIL است. IL یا MSIL یک زبان از نوع اسمبلی است که در یک فایل DLL یا EXE بسته‌بندی می‌شود. توجه کنید که اسمبلی‌ها فایل‌هایی با استاندارد اجرایی نیستند. لازم است یک کامپایلر زمان اجرا به نام JIT، IL را به یک کد ماشین خاص تبدیل کند. (زمانی که برنامه واقعاً اجرا می‌شود). چون CLR مسئول مدیریت این IL است، این کد میانی یکی از کلیدهای رویارویی با اهداف اسمی چارچوب NET از نظر سازگاری زبان است. همانطور که شکل ۱-۳ نشان می‌دهد، CLR

^۱ Type System

نمی‌داند چه زبانی این برنامه را ایجاد کرده است. تعامل آن با IL مستقل از زبان می‌باشد. چون برنامه‌ها از طریق IL با هم ارتباط برقرار می‌کنند، پس خروجی یک کامپایلر می‌تواند با خروجی کامپایلر متفاوت دیگر مجتمع شود.



شکل ۳-۱ عملکرد CLR

هدف دیگر .NET، قابلیت حمل Platform است، که با محلی کردن^۱ ایجاد کد ماشین در کامپایلر JIT فراهم می‌شود. بدین معنی که IL تولید شده روی یک Platform می‌تواند روی Platform دیگری که چارچوب خاص خودش و یک کامپایلر JIT با کد ماشین خاص خودش را دارد، اجرا شود.

کامپایلرهایی که کد هدف آنها CLR است، برای هر ماژول علاوه بر تولید IL، باید فراداده^۲ای را نیز صادر کنند. در فراداده‌ها مجموعه‌ای از جداول قرار می‌گیرند تا هر ماژول، کد خود- توصیف^۳ داشته باشد. در جداول علاوه بر توصیف کامل کد، اطلاعاتی درباره اسمبلی‌ها نیز وجود دارد. این اطلاعات شامل موارد دیگر نیز هستند: چه نوع داده‌هایی در دسترس هستند؟ نام هر نوع داده، اعضای نوع داده، دامنه یا میدان دید نوع داده و ویژگی‌های هر نوع داده دیگر. فراداده‌های دیگری که کاربردهای زیادی دارند:

- مهمترین کاربرد آن بوسیله کامپایلر JIT است، که اطلاعات همه‌ی نوع داده‌های مورد نیاز برای کامپایل کردن را مستقیماً از فراکد^۴ جمع‌آوری می‌کند. این اطلاعات را برای بررسی کد بکار می‌برد تا مطمئن شود برنامه عملیات را به درستی انجام می‌دهد. برای مثال، JIT، از طریق مقایسه‌ی پارامترهای متد، فراخوانی صحیح را مطمئن می‌سازد.
- فراداده‌ها در پروسه جمع‌آوری زباله^۵ استفاده می‌شوند. جمع‌کننده زباله، برای شناسایی فیلدها و ارجاعات آنها از فراداده استفاده می‌کند و می‌تواند تعیین کند حافظه‌ی چه اشیایی می‌توانند آزاد شوند یا نه؟
- .NET یک مجموعه از کلاس‌ها برای خواندن فراداده‌های یک برنامه فراهم می‌کند. این توانایی به نام انعکاس^۶ شناخته می‌شود، که یک ویژگی قدرتمند است و اجازه می‌دهد یک برنامه در زمان اجرا، کد را مورد جستجو قرار دهد و براساس اطلاعات یافته شده تصمیم‌گیری کند. می‌توان صفات سفارشی را به فراداده اضافه کرد.

^۱ Localizing

^۲ Metadata

^۳ Self descriptive

^۴ Meta code

^۵ Garbage Collection

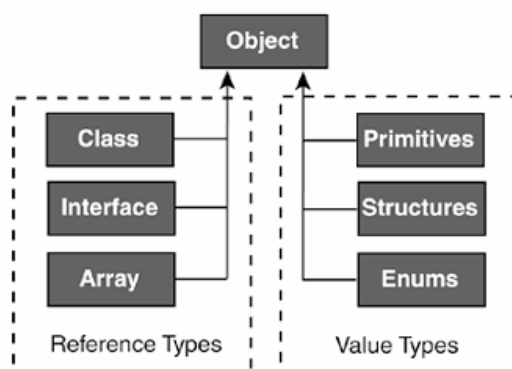
^۶ Reflection

IL و فراداده برای فراهم ساختن ارتباط بین زبانی بسیار مهم هستند. اما دنیای واقعی به همه کامپایلرهای .NET که یک مجموعه‌ی مشترک از انواع داده‌ای و توصیف زبان را پشتیبانی می‌کنند، منوط است. برای مثال، دو زبان در IL سازگار نیستند، اگر یکی عدد صحیح علامت‌دار ۳۲ بیتی را پشتیبانی کند و دیگری آن را پشتیبانی نکند. آنها ممکن است گرامر متفاوتی داشته باشند، اما باید روی انواع داده‌ای پایه که پشتیبانی می‌کنند، توافق داشته باشند.

همانطور که قبلاً بحث شده، CLI یک توصیف رسمی به نام CTS تعریف می‌کند که بخش مکمل CLR است. آن شرح می‌دهد که چگونه انواع داده‌ای تعریف می‌شوند و چگونه باید رفتار کنند تا بوسیله CLR پشتیبانی شوند.

CTS-۲-۲-۱

CTS یک مجموعه‌ی پایه از انواع داده‌ای برای زبان‌های تحت .NET فراهم می‌کند. علاوه بر این نحوه اعلان و ایجاد انواع داده‌ای سفارشی و نحوه مدیریت مدت زمان عمر نمونه‌های این نوع داده‌ها را تعیین می‌کند. شکل ۱-۴ نشان می‌دهد چگونه .NET، CTS را سازمان‌دهی می‌کند.



شکل ۱-۴ - انواع داده‌ای پایه تعریف شده در CTS

از این شکل دو چیز به دست می‌آید. واضح‌ترین مورد اینکه، انواع مقداری^۱ یا ارجاعی^۲ گروه‌بندی می‌شوند. این طبقه‌بندی بر اساس نحوه‌ی ذخیره و دسترسی در حافظه است. انواع ارجاعی در یک ناحیه‌ی خاص حافظه به نام Heap، از طریق اشاره‌گرها دستیابی می‌شوند، در حالیکه انواع مقداری مستقیماً در Stack برنامه قرار می‌گیرند. مورد دیگر اینکه، همه انواع داده‌ای اعم از سفارشی و انواع داده‌ای .NET از یک نوع داده‌ی پیش تعریف شده بنام System.Object ارث‌بری می‌کنند. پس مطمئناً همه انواع داده‌ای، یک مجموعه‌ی پایه از متدها و خصوصیات را ارث‌بری می‌کنند.

در .NET، "نوع داده"^۳ یک عبارت کلی است که به یک کلاس، ساختار، نوع شمارشی یا واسط یا نماینده اشاره می‌کند.

کامپایلری که مطیع مشخصه CTS است، تضمین می‌کند که انواع داده‌ای آن می‌توانند بوسیله‌ی CLR میزبانی شوند. این به تنهایی ضامن ارتباط یک زبان با زبان دیگر نیست. یک مجموعه‌ی محدود کننده از مشخصات به نام CLS وجود دارد که قوانینی را برای ارتباط بین زبانی تعریف می‌کنند. این مشخصات، ویژگی‌های حداقلی تعریف می‌کنند که یک کامپایلر با هدف CLR باید شامل باشد. جدول ۱-۱ بعضی از قوانین CLS را نشان می‌دهد:

جدول ۱-۱ - قوانین و ویژگی‌های CLS

^۱ ValueType

^۲ Reference Type

^۳ Type

میدان دید ^۱	این قوانین فقط به آن اعضای از یک نوع داده اعمال می‌شوند که از بیرون اسمبلی تعریف کننده در دسترس است.
کاراکترها و حالت آنها	در دو متغیر متمایز، باید اختلاف آنها بیشتر از حالت کاراکترها در آنها باشد.
انواع اصلی	انواع داده‌ای اصلی مطیع CLS هستند. Byte, Int۱۶, Int۳۲, Int۶۴, Single, Double, Boolean, char, Decimal, IntPtr, String
سازنده	یک سازنده قبل از دسترسی به هر داده از کتابخانه، باید سازنده کلاس پایه را فراخوانی کند.
محدوده‌های آرایه	همه ابعاد آرایه‌ها باید از اندیس صفر شروع شوند.
نوع شمارشی	نوع داده اصلی یک نوع شمارشی باید از نوع Byte و Int۱۶, Int۳۲, Int۶۴ باشد.
متد	انواع داده‌ی پارامترها و مقدار بازگشتی استفاده شده در متد باید مطیع CLS باشند.

این قوانین مشخص و واضح هستند. قطعه کدی از #C را در نظر بگیرید تا نحوه‌ی اعمال این قوانین را ببینیم:

```
public Class Conversion
{
    public double Metric (double inches)
    {
        return (۲.۵۴ * inches) ;
    }
    public double metric (double miles)
    {
        return (miles/ ,۶۲) ;
    }
}
```

اگرچه با کد #C آشنا نیستید، ولی می‌توانید به راحتی ببینید که با قوانین CLS مغایرت دارد. چون دو متد Metric و metric قانون را نقض کرده‌اند. اگرچه در #C جواب می‌دهد، ولی در صورت تعامل با کدنویسی VB.NET با شکست مواجه می‌شود.

۱-۲-۳-اسمبلی‌ها

همه کدهای مدیریت شده که تحت .NET اجرا می‌شوند، باید در یک اسمبلی قرار گیرند. بطور منطقی اسمبلی یک فایل EXE یا DLL است. از نظر فیزیکی ممکن است شامل کلکسیون‌ی از یک یا چند فایل باشد، که هر فایل می‌تواند شامل کد یا منابعی همچون تصاویر یا XML باشند.

زمانی که یک کامپایلر سازگار .NET یک فایل کد منبع را به یک DLL یا EXE تبدیل می‌کند، یک اسمبلی ایجاد می‌شود. همانطور که در شکل ۱-۵ مشاهده می‌کنید، یک اسمبلی شامل یک اظهارنامه^۲، فرا داده و IL است. این موارد را بیشتر بررسی می‌کنیم.

اظهار نامه

^۱ Visiblity(Scope)

^۲ manifest

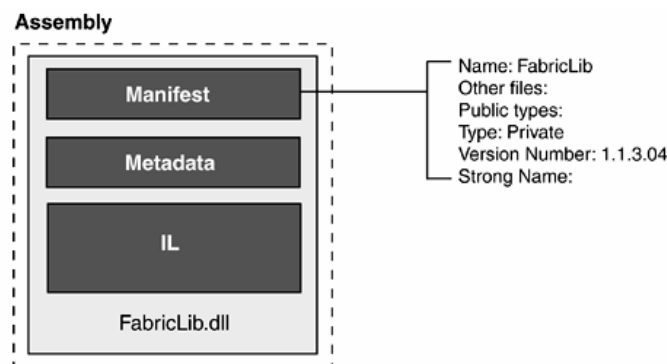
هر اسمبلی باید یک فایل برای در برداشتن اظهارنامه داشته باشد. اظهارنامه شامل جداولی است که در این جداول، اسامی همه فایل‌های موجود در اسمبلی، ارجاعات به اسمبلی‌های بیرونی و اطلاعاتی همچون نام و نسخه اسمبلی لیست می‌شوند. اسمبلی‌های نامگذاری شده، یک امضاء دیجیتالی^۱ منحصر به فرد دارند. زمانی که یک اسمبلی بارگذاری می‌شود، ابتدا فایل manifest توسط CLR بارگذاری می‌شود تا بتواند اعضای اسمبلی را تشخیص دهد.

فرا داده

علاوه بر جداول manifest کامپایلر #C جداول تعریف نوع داده و ارجاع را تولید می‌کند. جداول تعریف، یک توصیف کامل از انواع داده‌ای موجود در IL فراهم می‌کنند. برای مثال، جداولی برای تعریف انواع داده‌ای، متدها، فلیدها و خصوصیات وجود دارند. جداول ارجاع، اطلاعاتی را در مورد ارجاعات به انواع داده‌ای و اسمبلی‌های دیگر شامل هستند. کامپایلر JIT بر پایه‌ی این جداول، IL را به کد ماشین موردنظر تبدیل می‌کند.

IL

نقش IL در حال حاضر بحث شده است. قبل از اینکه CLR بتواند IL را بکاربرد، باید در یک اسمبلی EXE یا DLL بسته‌بندی شود. این دو یکسان نیستند. یک اسمبلی EXE نقطه‌ی ورودی دارد که آن را قابل اجرا می‌سازد. یک اسمبلی DLL بصورت یک کتابخانه از تعاریف انواع داده طراحی می‌شود.



شکل ۱-۵ اسمبلی تک فایلی

اسمبلی، چیزی بالاتر از یک روش منطقی برای بسته‌بندی کد قابل اجرا است. آن قلب مدل NET را برای توسعه‌ی کد، کنترل نسخه و امنیت تشکیل می‌دهد.

- تمام کد مدیریت شده‌ی مربوط به یک برنامه مستقل، یک کنترل یا یک کتابخانه DLL شامل انواع داده‌ای قابل استفاده مجدد، در یک اسمبلی بسته‌بندی می‌شوند. آن تجربه‌ناپذیرترین واحد است که می‌تواند روی یک سیستم بکار گرفته شود. زمانی که یک برنامه آغاز می‌شود، باید فقط اسمبلی‌هایی که برای مقداردهی اولیه لازم هستند، در حافظه باشند. اسمبلی‌های دیگر براساس نیاز بارگذاری می‌شوند. یک توسعه‌دهنده این مزیت را برای تقسیم یک برنامه به چندین اسمبلی براساس فرکانس استفاده آنها بکار می‌برد.

- در NET، اسمبلی یک محدوده‌ی نسخه تشکیل می‌دهد. فیلد نسخه در اظهارنامه روی تمام انواع داده و منابع اسمبلی اعمال می‌گردد. همه فایل‌های تشکیل دهنده اسمبلی بصورت یک واحد منفرد با نسخه‌ی یکسان در نظر گرفته می‌شوند. با جدا کردن بسته فیزیکی از منطقی، NET می‌تواند یک صفت منطقی را مابین چندین فایل فیزیکی به اشتراک گذارد. این یک ویژگی پایه است که یک اسمبلی را از سیستم مبتنی بر DLL سنتی متمایز می‌کند.

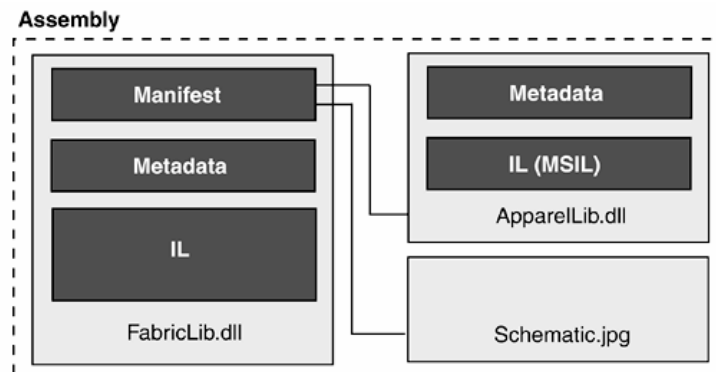
^۱ Digital signature

- اسمبلی یک محدوده امنیت روی جوازهای دسترسی تشکیل می‌دهد. #C معرف‌های دسترسی را برای کنترل نحوه دسترسی انواع داده و اعضای انواع داده در یک اسمبلی بکار می‌برد. دو مورد از کاربرد اسمبلی به عنوان محدوده بصورت زیر است:

— public: دسترسی نامحدود به هر اسمبلی را مجاز می‌شمارد.

— internal: دسترسی را به انواع داده و اعضای داخل آن اسمبلی محدود می‌کند.

همانطور که شرح داده شد، ممکن است یک اسمبلی چندین فایل را شامل شود که این فایل‌ها محدود به ماژول‌های کد نیستند. ممکن است فایل‌های منبع همچون تصاویر گرافیکی و فایل‌های متنی باشند. یک کاربرد عمومی این فایل‌ها، مجاز داشتن منابعی است که یک واسط برای کشور یا زبان یک کاربر فراهم می‌کند. هیچ محدودیتی روی تعداد فایل‌ها در اسمبلی نیست. شکل ۱-۶ طرح یک اسمبلی چند فایلی را در دیگرام اسمبلی چند فایلی نشان می‌دهد. توجه داشته باشید که اظهارنامه‌ی اسمبلی شامل اطلاعاتی است که همه فایل‌های اسمبلی را تعیین می‌کند.



شکل ۱-۶-اسمبلی چند فایلی

- اگرچه بیشتر اسمبلی‌ها، یک فایل منفرد را شامل هستند، ولی در چندین حالت، اسمبلی‌های چند فایلی مزایایی دارند.
- آنها ترکیب ماژول‌های تولید شده در زبان‌های برنامه‌سازی مختلف را مجاز می‌دارند. اگر کدهایی در VB.NET و کدهایی در #C نوشته شده باشند، این دو کد می‌توانند در یک فایل اسمبلی .NET باهم تعامل داشته باشند.
 - برای بهینه کردن نحوه‌ی بارگذاری به CLR، ماژول‌های کد می‌توانند تقسیم‌بندی شوند. بایستی کد پر استفاده و مرتبط بهم در یک ماژول قرار گیرند. CLR ماژول‌ها را در صورت نیاز بارگذاری می‌کند. هنگام ایجاد یک کتابخانه‌ی کلاس، لازم است قطعات کد براساس چرخه زندگی و نسخه و امنیت مشترک در اسمبلی‌های مجزایی گروه‌بندی شوند.
 - فایل‌های منبع می‌توانند در ماژول‌های مجزایی از ماژول‌های IL قرار بگیرند، تا چندین برنامه منابع مشترک خود را به راحتی به اشتراک گذارند.

اسمبلی‌های چند فایلی می‌توانند با اجرای کامپایلر #C از خط فرمان یا برنامه سودمند (Assembly Linker) AL.exe ایجاد شوند. یک مثال کاربرد کامپایلر #C از خط فرمان در بخش‌های بعدی آمده است. توجه داشته باشید که VS.NET ۵ و ۲ ایجاد اسمبلی‌های چند فایلی را پشتیبانی نمی‌کند.

۱-۲-۴-اسمبلی‌های خصوصی^۱ و اشتراکی^۲

^۱ Private

^۲ Shared

اسمبلی‌ها به دو روش ممکن (خصوصی یا سراسری) ایجاد می‌شوند. اسمبلی‌هایی که در فهرست اصلی برنامه یا در یک زیرفهرست آن قرار می‌گیرند، اسمبلی‌های خصوصی خوانده می‌شوند. نصب و بهنگام سازی یک اسمبلی ساده است. فقط لازم است اسمبلی به یک فهرست به نام AppBase در آن برنامه کپی شود. هیچ تنظیم رجیستری لازم نیست. علاوه بر این برای override کردن تنظیمات پیکربندی برنامه می‌توان یک اظهارنامه به برنامه کاربردی اضافه کرد و اجازه داد یک فایل اسمبلی به فهرست AppBase منتقل شود.

یک اسمبلی اشتراکی در یک موقعیت سراسری نصب می‌شود، که ^۱GAC نامیده می‌شود و بوسیله چندین برنامه قابل دستیابی است. مهمترین ویژگی GAC مجاز داشتن اجرای چندین نسخه از اسمبلی در کنار همدیگر می‌باشند. NET برای پشتیبانی از این امر، مشکل تداخل اسمی را با استفاده از چهار صفت جهت شناختن یک اسمبلی رفع می‌کند. نام فایل، مشخصه فرهنگ، شماره نسخه، نشانه کلید عمومی.

معمولاً اسمبلی‌های عمومی در زیرفهرست Assembly از فهرست سیستم عامل (winnt) قرار می‌گیرند. همانطور که در شکل ۱-۷ نشان داده شده است، اسمبلی‌ها در یک قالب خاص لیست می‌شوند که چهار صفت آن نشان داده می‌شود. چارچوب NET یک فایل DLL دارد که Windows Explorer را قادر می‌سازد محتوای GAC را نمایش دهد.

نگاهی سریع به این چهار خصوصیت داریم:

- نام اسمبلی: این همان نام فایل اسمبلی بدون پسوند آن است.
- نسخه: هر اسمبلی یک شماره نسخه دارد که به همه فایل‌های اسمبلی اعمال می‌گردد. آن شامل چهار عدد به قالب زیر است:

<major number>.<minor number>.<build>.<revision>

معمولاً شماره‌های اصلی و فرعی نسخه برای تغییرات بروز می‌شوند، چون سازگاری را تحت تاثیر قرار می‌دهند. شماره نسخه بوسیله یک صفت به نام AssemblyVersion در کد منبع اسمبلی به آن تخصیص داده می‌شود.

- تنظیم فرهنگ: ممکن است محتوای یک اسمبلی به یک زبان و فرهنگ خاصی تخصیص داده شود، که با صفت AssemblyCulture در کد منبع اسمبلی مشخص می‌گردد.

```
[("assembly: AssmbllyCulture ("fr-CA]
```

- نشانه کلید عمومی: برای اطمینان از اینکه یک اسمبلی اشتراکی، منحصر بفرد و تصدیق شده است، در NET باید ایجاد کننده اسمبلی آن را با یک نام قوی نشانه‌گذاری کند. این پروسه را امضاء کردن^۲ گویند که جفت کلید عمومی/خصوصی را نیاز دارد. در زمان کامپایل اسمبلی، کلید خصوصی برای تولید یک نام قوی بکار می‌رود، کلید عمومی برای نشانه بزرگ است، پس با عمل درهم‌سازی کلید عمومی، ۸ بایت آخر آن را انتخاب می‌کنند. این نشانه در اظهارنامه‌ی هر اسمبلی سرویس‌گیرنده که به یک اسمبلی اشتراکی ارجاع دارد جای می‌گیرد و برای تشخیص اسمبلی در حین اجرا بکار می‌رود.

^۱ Global Assembly Cache

^۲ Signing

Assembly Name	Version	Culture	Public Key Token
Accessibility	2.0.3600.0		b03f5f7f11d50a3a
ADODB	7.0.3300.0		b03f5f7f11d50a3a
Apphost	2.0.3600.0		b03f5f7f11d50a3a
AspNetMMCExt	2.0.3600.0		b03f5f7f11d50a3a
CRVsPackageLib	1.0.0.0		692fba5521e1304
CrystalDecisions.CrystalReports.Engine	9.1.3300.0		692fba5521e1304

شکل ۱-۷- بخشی از فهرست اسمبلی سراسری

۱-۲-۴- از پیش کامپایل کردن یک اسمبلی

بعد از بارگذاری یک اسمبلی، بایستی IL آن به کد ماشین جاری کامپایل شود. اگر شما با فایل‌های قابل اجرا در فرمت کد ماشین کار می‌کنید، سئوالاتی در مورد بهره‌وری و اینکه آیا ایجاد فایل‌های قابل اجرای معادل در .NET امکان‌پذیر است، پیش می‌آید. جواب سئوال قسمت دوم بله است. .NET یک روش برای از پیش کامپایل کردن یک اسمبلی فراهم می‌کند.

چار چوب .NET (ابزاری به نام Native Image Generator (Ngen دارد، که برای کامپایل یک اسمبلی به یک "تصویر محلی" بکار گرفته می‌شود، که در کش تصویر محلی (یک فضای رزرو شده از GAC) ذخیره می‌شود. هر زمانی که CLR یک اسمبلی را بارگذاری می‌کند، کش را برای وجود یک تصویر محلی از آن اسمبلی بررسی می‌کند، اگر باشد آن کد از پیش کامپایل شده را بارگذاری می‌کند. ظاهراً، این یک ایده‌ی خوب برای بهبود کارایی به نظر می‌رسد. اما چندین ایراد دارد.

Ngen یک تصویر برای معماری ماشین فرضی اجرا کننده ایجاد می‌کند. برای مثال روی هر ماشین سازگار با پردازنده x86، زمانی که JIT در .NET اجرا می‌گردد، آن از نوع ماشین آگاه بوده و می‌تواند نکات بهره‌وری را در نظر بگیرد. نتیجه اینکه اغلب اوقات خروجی آن خارج از عملکرد اسمبلی از پیش کامپایل شده است. ایراد دیگر کاربرد یک تصویر محلی این است که تغییرات پیکربندی سخت افزار یا سیستم‌عامل یک سیستم، اغلب اوقات اسمبلی از پیش تعریف شده را نامعتبر می‌کند.

تایید کد^۲

به عنوان بخشی از پروسه‌ی کامپایل JIT، CLR دو نوع تایید انجام می‌دهد. تایید IL و ارزیابی فراداده. هدف آن اطمینان از قابل قبول بودن کد نوع امن است. در عمل، بدین معنی است که پارامترهای موجود در یک فراخوانی و متد فراخوانی شده، همنوع هستند یا نوع مقدار بازگشتی یک متد همان نوع برگشتی در اعلان است. خلاصه اینکه CLR از طریق IL و فراداده، سازگاری نوع داده‌ها را مطمئن می‌سازد. اگر به غیر از این باشد، یک خطا رخ می‌دهد.

مزیت کد تایید شده این است که CLR یقین دارد، کد از طریق دسترسی به حافظه‌ی خارج از محدوده‌ی مجاز خود نمی‌تواند برنامه‌های دیگر را تحت تاثیر قرار دهد. بدین ترتیب CLR برای اجرای امن چندین برنامه در یک پروسه یا فضای آدرس واحدی است و کارایی و کاهش استفاده از منابع سیستم عامل را بهبود می‌بخشد.

۱-۳- FCL

FCL کلکسیونی از کلاس‌ها و انواع داده‌ی دیگر (نوع شمارشی، ساختارها، واسط‌ها) است که برای تمام کدهای مدیریت شده‌ی نوشته شده در هر زبانی با کد هدف CLR در دسترس هستند. این بسیار مهم است، بدین معنی که این کتابخانه‌ها مختص کامپایلرهای خاصی نیستند. به عنوان یک توسعه‌دهنده، شما می‌توانید با انواع داده موجود در کتابخانه‌ها آشنا شوید، که این دانش در هر زبان .NET برای شما قابل استفاده است.

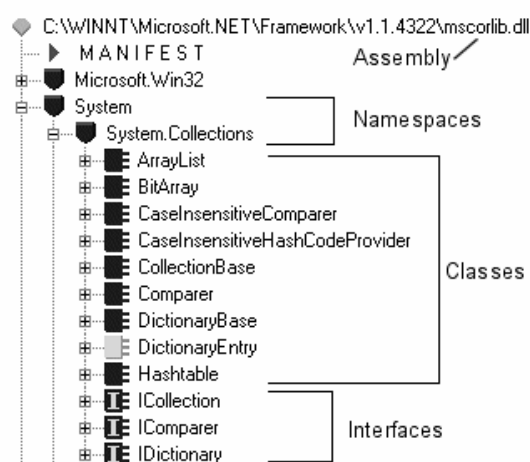
^۱ Native Image

^۲ Code verification

منابع بوسیله FCL^۱ از طریق گروه‌بندی‌های منطقی به نام فضای نامی سازمان‌دهی می‌شوند. این گروه‌بندی‌ها براساس محدوده عملکرد می‌باشند. برای مثال، انواع داده‌ای مورد استفاده برای عملیات گرافیکی در فضاهای نامی `System.Drawing` و `System.Drawing.Drawing2D` گروه‌بندی می‌شوند. انواع داده‌ای مورد نیاز برای ورود و خروج فایل‌ها، اعضایی از فضای نامی `System.IO` هستند. فضاهای نامی، یک مفهوم منطقی نه فیزیکی هستند.

FCL صدها اسمبلی DLL را در برمی‌گیرد. هر اسمبلی ممکن است چند فضای نامی را شامل شود. به علاوه، ممکن است یک فضای نامی چندین اسمبلی را بهم ببافد.^۱ برای ارائه این مطلب به داخل یک اسمبلی FCL نگاه کنید.

شکل ۱-۸ بخشی از خروجی تولید شده با برنامه `ILDasm.exe` جهت کنترل محتوای اسمبلی `mscorlib` را نمایش می‌دهد. اگرچه یک لیست ناقص است، شما می‌توانید ببینید که `mcorlib` فضای نامی `System` را شامل است و انواع داده‌ای پایه‌ی .NET در آن قرار دارند و فضای نامی `System.Collections` را نیز در بر می‌گیرد که کلاس‌ها و واسطه‌های مورد استفاده برای دستکاری کلکسیون‌های داده را شامل می‌شود.



جدول ۱-۲ بعضی از مهمترین فضاهای نامی .NET را لیست می‌کند.

جدول ۱-۲ تعدادی از فضاهای نامی متداول

کاربرد	فضای اسمی
شامل انواع داده‌ی پایه است که بوسیله همه برنامه‌ها استفاده می‌شود. آن کلاس‌های استثناء، خصوصیتی از پیش تعریف شده، کتابخانه‌ی Math و کلاس‌های مدیریت محیط برنامه را نیز شامل است.	System
واسطه‌ها و کلاس‌های استفاده شده جهت مدیریت کلکسیون‌هایی از اشیا. این کلکسیون‌ها شامل ArrayList, Stack, Hashtable و ... هستند.	System.Collections System.Collections.Specialized System.Collections.Generic

^۱ Span

<p>کلاس‌های مورد استفاده برای عملیات پایگاه داده(ADO.NET). فضاهای نامی سرویس گیرنده‌ی Oracle و SQLServer را پشتیبانی می‌کنند و OleDb, Odbc اتصال داده مورد استفاده را تعریف می‌کنند.</p>	<p>System.Data System.Data.OracleClient System.Data.OleDb System.Data.Odbc</p>
<p>کلاس‌هایی را شامل است که می‌توانند اجرای برنامه، اشکال‌یابی، کار با log های سیستم و شمارنده‌های بهره‌وری را پیگیری کنند.</p>	<p>System.Diagnostics</p>
<p>عملکردهای گرافیکی را برای GDI+ فراهم می‌کند. این فضاهای نامی یک کلاس ترسیم به خوبی fonts, pens, geometric shapes, brushes را در بر دارند.</p>	<p>System.Drawing System.Drawing.Drawing2D System.Drawing.Printing System.Drawing.Text</p>
<p>کلاس‌هایی در ارتباط با اطلاعات مرتبط با فرهنگ دارد که روش مقداردهی تاریخ‌ها، واحد پول و سمبل‌های نمایشی را تحت تاثیر قرار می‌دهد.</p>	<p>System.Globalization</p>
<p>عملیات ورود و خروج فایل و جریان داده را فراهم می‌کند. این کلاس‌ها یک روش برای دسترسی به سیستم‌های فایل سیستم عامل میزبان فراهم می‌کنند.</p>	<p>System.IO</p>
<p>کلاس‌هایی که عملیات و پروتکل‌های شبکه را پشتیبانی می‌کنند. برای مثال WebRequest و WebResponse که یک صفحه وب را درخواست و واکنشی می‌کنند.</p>	<p>System.Net</p>
<p>انواع داده‌ای که تغییر فراداده را در زمان اجرا مجاز می‌دارند، شامل است. فضای نامی Emit به یک کامپایلر یا ابزار، تولید پویای IL و فراداده را اجازه می‌دهد.</p>	<p>System.Reflection System.Reflection.Emit</p>
<p>ارتباط داخلی مابین کد مدیریت شده و کد مدیریت نشده همچون DLL یا COM را فراهم می‌سازد.</p>	<p>System.Runtime.InteropServices</p>
<p>کلاس‌های استفاده شده برای مدیریت امنیت .NET. کلاس‌هایی تعریف می‌کنند که دسترسی به عملیات و منابع را کنترل می‌کنند.</p>	<p>System.Security System.Security.Permission System.Security.Cryptography</p>
<p>کلاس‌هایی که موتور عبارت منظم .NET را پشتیبانی می‌کنند.</p>	<p>System.Text.RegularExpressions</p>
<p>فعالیت‌های برنامه‌نویسی ریسمان یعنی ایجاد ریسمان، همگام‌سازی و دسترسی به استخر ریسمان را مدیریت می‌کنند.</p>	<p>System.Threading System.Threading.Thread</p>
<p>کلاس‌های مرتبط با اینترنت که به ASP.NET معروف هستند. آنها نیازهای ارتباط با سرور، دستکاری کوکی‌ها را مدیریت می‌کنند.</p>	<p>System.Web System.Web.Services</p>
<p>Web.UI دارای کلاس‌ها و واسطه‌هایی است که برای ایجاد کنترل‌ها و صفحات مرتبط با فرم‌های وب استفاده می‌شوند.</p>	<p>System.Web.UI System.Web.UI.WebControls System.Web.Security</p>