عنوان: **آشنایی با CLR: قسمت نهم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۳۰ ۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: CLR, .NET Framework

.net framework شامل Framework Class Library یا به اختصار FCL است. FCL مجموعه‌ای از dll اسمبلی‌هایی است که صدها و هزاران نوع در آن تعریف شده‌اند و هر نوع تعدادی کار انجام می‌دهد. همچنین مایکروسافت کتابخانه‌های اضافه‌تری را چون azure و Directx نیز ارائه کرده است که باز هر کدام شامل نوع‌های زیادی می‌شوند. این کتابخانه به طور شگفت آوری باعث سرعت و راحتی توسعه دهندگان در زمینه فناوری‌های مایکروسافت گشته است.

تعدادی از فناوری‌هایی که توسط این کتابخانه پشتیبانی می‌شوند در زیر آمده است:

Web Service : این فناوری اجازه‌ی ارسال و دریافت پیام‌های تحت شبکه را به خصوص بر روی اینترنت، فراهم می‌کند و باعث ارتباط جامع‌تر بین برنامه‌ها و فناوری‌های مختلف می‌گردد. در انواع جدیدتر WCF و Web Api نیز به بازار ارائه شده‌اند.

webform و MVC : فناوری‌های تحت وب که باعث سهولت در ساخت وب سایت‌ها می‌شوند که وب فرم رفته رفته به سمت منسوخ شدن پیش می‌رود و در صورتی که قصد دارید طراحی وب را آغاز کنید توصیه میکنم از همان اول به سمت MVC بروید.

Rich Windows GUI Application : برای سهولت در ایجاد برنامه‌های تحت وب حالا چه با فناوری WPF یا فناوری قدیمی و البته منسوخ شده Windows Form.
Windows Console Application : برای ایجاد برنامه‌های ساده و بدون رابط گرافیکی.
Windows Services : شما می‌توانید یک یا چند سرویس تحت ویندوز را که توسط Service Control Manager یا به اختصار SCM کنترل می‌شوند، تولید کنید.
Database stored Procedure : نوشتن stored procedure بر روی دیتابیس‌هایی چون sql server و اوراکل و ... توسط فریم ورک دات نت مهیاست.
Component Libraray : ساخت اسمبلی‌های واحدی که می‌توانند با انواع مختلفی از موارد بالا ارتباط برقرار کنند.
Portable Class Libary : این نوع پروژه‌ها شما را قادر می‌سازد تا کلاس‌هایی با قابلیت انتقال پذیری برای استفاده در سیلور لایت، ویندوز فون و ایکس باکس و فروشگاه ویندوز و ... تولید کنید.

ازآنجا که یک کتابخانه شامل زیادی نوع می‌گردد سعی شده است گروه بندی‌های مختلفی از آن در قالبی به اسم فضای نام namespace تقسیم بندی گردند که شما آشنایی با آن‌ها دارید. به همین جهت فقط تصویر زیر را که نمایشی از فضای نام‌های اساسی و مشترک و پرکاربرد هستند، قرار می‌دهم.

Namespace	Description of Contents
System	All of the basic types used by every application
System.Data	Types for communicating with a database and processing data
System.IO	Types for doing stream I/O and walking directories and files
System.Net	Types that allow for low-level network communications and working with some common Internet protocols
System.Runtime.InteropServices	Types that allow managed code to access unmanaged operating system platform facilities such as COM components and functions in Win32 or custom DLLs
System.Security	Types used for protecting data and resources
System.Text	Types to work with text in different encodings, such as ASCII and Unicode
System.Threading	Types used for asynchronous operations and synchronizing access to resources
System.Xml	Types used for processing Extensible Markup Language (XML) schemas and data

در CLR مفهومی به نام Common Type System یا CTS وجود دارد که توضیح می‌دهد نوع‌ها باید چگونه تعریف شوند و چگونه باید رفتار کنند که این قوانین از آنجایی که در ریشه‌ی CLR نهفته است، بین تمامی زبان‌های دات نت مشترک می‌باشد. تعدادی از مشخصات این CTS در زیر آورده شده است ولی در آینده بررسی بیشتری روی آنان خواهیم داشت:

فیلد
متد
پراپرتی
رویدادها

CTS همچنین شامل قوانین زیادی در مورد وضعیت کپسوله سازی برای اعضای یک نوع دارد:

private
public
Family یا در زبان‌هایی مثل سی ++ و سی شارپ با نام protected شناخته می‌شود.
family and assembly: این هم مثل بالایی است ولی کلاس مشتق شده باید در همان اسمبلی باشد. در زبان‌هایی چون سی شارپ و ویژوال بیسیک، چنین امکانی پیاده سازی نشده‌است و دسترسی به آن ممکن نیست ولی در IL Assembly چنین قابلیتی وجود دارد.
Assembly یا در بعضی زبان‌ها به نام internal شناخته می‌شود.
Family Or Assembly: که در سی شارپ با نوع Protected internal شناخته می‌شود. در این وضعیت هر عضوی در هر اسمبلی قابل ارث بری است و یک عضو فقط می‌تواند در همان اسمبلی مورد استفاده قرار بگیرد.

موارد دیگری که تحت قوانین CTS هستند مفاهیم ارث بری، متدهای مجازی، عمر اشیاء و .. است.

یکی دیگر از ویژگی‌های CTS این است که همه‌ی نوع‌ها از نوع شیء Object که در فضای نام system قرار دارد ارث بری کرده‌اند. به همین دلیل همه‌ی نوع‌ها حداقل قابلیت‌هایی را که یک نوع object ارئه میدهد، دارند که به شرح زیر هستند:

مقایسه‌ی دو شیء از لحاظ برابری.
به دست آوردن هش کد برای هر نمونه از یک شیء
ارائه‌ای از وضعیت شیء به صورت رشته ای
دریافت نوع شیء جاری

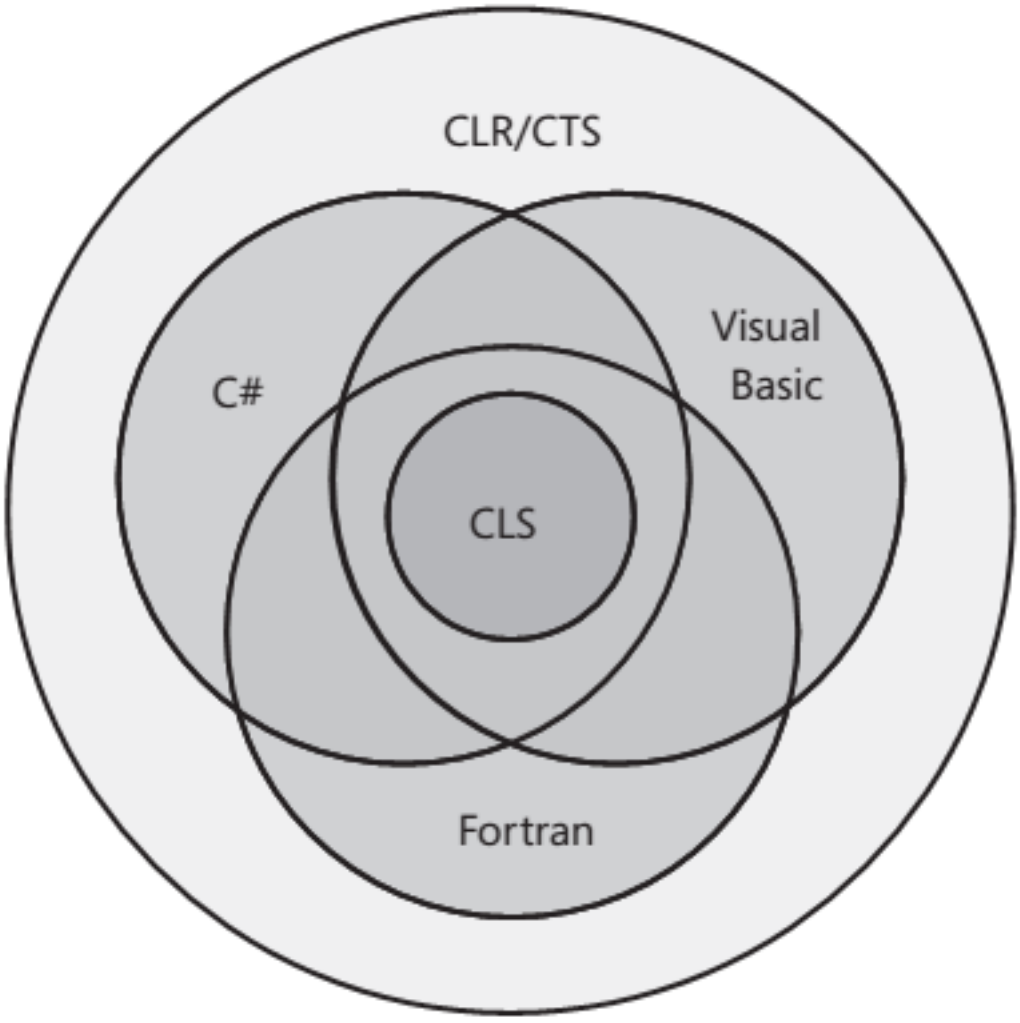
CLS

وجود COMها به دلیل ایجاد اشیاء در یک زبان متفاوت بود تا با زبان دیگر ارتباط برقرار کنند. در طرف دیگر CLR هم بین زبان‌های برنامه نویسی یکپارچگی ایجاد کرده است. یکپارچگی زبان‌های برنامه نویسی علل زیادی دارند. اول اینکه رسیدن به هدف یا یک الگوریتم خاص در زبان دیگر راحت‌تر از زبان پایه پروژه است. دوم در یک کار تیمی که افراد مختلف با دانش متفاوتی حضور دارند و ممکن است زبان هر یک متفاوت باشند.

برای ایجاد این یکپارچگی، مایکروسافت سیستم CLS یا Common Language Specification را راه اندازی کرد. این سیستم برای تولیدکنندگان کامپایلرها جزئیاتی را تعریف می‌کند که کامپایلر آن‌ها را باید با حداقل ویژگی‌های تعریف شده‌ی CLR، پشتیبانی کند.

CLR/CTS مجموعه‌ای از ویژگی‌ها را شامل می‌شود و گفتیم که هر زبانی بسیاری از این ویژگی‌ها را پشتیبانی می‌کند ولی نه کامل. به عنوان مثال برنامه نویسی که قصد کرده از IL Assembly استفاده کند، قادر است از تمامی این ویژگی‌هایی که CLR/CTS ارائه می‌دهند، استفاده کند ولی تعدادی دیگر از زبان‌ها مثل سی شارپ و فورترن و ویژوال بیسیک تنها بخشی از آن را استفاده می‌کنند و CLS *حداقل ویژگی که بین همه این زبان‌ها مشترک است را ارائه می‌کند.*

شکل زیر را نگاه کنید:



یعنی اگر شما دارید نوع جدیدی را در یک زبان ایجاد می‌کنید که قصد دارید در یک زبان دیگر استفاده شود، نباید از امتیازات ویژه‌ای که آن زبان در اختیار شما می‌گذارد و به بیان بهتر CLS آن‌ها را پشتیبانی نمی‌کند، استفاده کنید؛ چرا که کد شما ممکن است

در زبان دیگر مورد استفاده قرار نگیرد.

به کد زیر دقت کنید. تعدادی از کدها سازگاری کامل با CLS دارند که به آن‌ها CLS Compliant گویند و تعدادی از آن‌ها non-CLS-Compliant هستند یعنی با CLS سازگاری ندارند ولی استفاده از خاصیت [assembly: CLSCompliant(true)] باعث می‌شود که تا کامپایلر از پشتیبانی و سازگاری این کدها اطمینان کسب کند و در صورت وجود، از اجرای آن جلوگیری کند. با کمپایل کد زیر دو اخطار به ما میرسد.

```
using System;

// Tell compiler to check for CLS compliance
[assembly: CLSCompliant(true)]

namespace SomeLibrary {

// Warnings appear because the class is public
public sealed class SomeLibraryType {

// Warning: Return type of 'SomeLibrary.SomeLibraryType.Abc()'
// is not CLS-compliant
public UInt32 Abc() { return 0; }

// Warning: Identifier 'SomeLibrary.SomeLibraryType.abc()'
// differing only in case is not CLS-compliant
public void abc() { }

// No warning: this method is private
private UInt32 ABC() { return 0; }
}
}
```

اولین اخطار اینکه یکی از متدها یک عدد صحیح بدون علامت unsigned integer را بر می‌گرداند که همه‌ی زبان‌ها آن را پشتیبانی نمی‌کنند و خاص بعضی از زبان هاست.
دومین اخطار اینکه دو متد یکسان وجود دارند که در حروف بزرگ و کوچک تفاوت دارند. ولی زبان هایی چون ویژوال بیسیک نمی‌توانند تفاوتی بین دو متد abc و ABC بیابند.

نکته‌ی جالب اینکه اگر شما کلمه public را از جلوی نام کلاس بردارید تمامی این اخطارها لغو می‌شود. به این خاطر که این‌ها اشیای داخلی آن اسمبلی شناخته شده و قرار نیست از بیرون به آن دسترسی صورت بگیرد. عضو خصوصی کد بالا را ببینید؛ کامنت بالای آن می‌گوید که چون خصوصی است هشداری نمی‌گیرد، چون قرار نیست در زبان مقصد از آن به طور مستقیم استفاده کند.
برای دیدن قوانین CLS به این صفحه مراجعه فرمایید.

سازگاری با کدهای مدیریت نشده

در بالا در مورد یکپارچگی و سازگاری کدهای مدیریت شده توسط CLS صحبت کردیم ولی در مورد ارتباط با کدهای مدیریت نشده چطور؟
مایکروسافت موقعیکه CLR را ارئه کرد، متوجه این قضیه بود که بسیاری از شرکت‌ها توانایی اینکه کدهای خودشون را مجددا طراحی و پیاده سازی کنند، ندارند و خوب، سورس‌های مدیریت نشده‌ی زیادی هم موجود هست که توسعه دهندگان علاقه زیادی به استفاده از آن‌ها دارند. در نتیجه مایکروسافت طرحی را ریخت که CLR هر دو قسمت کدهای مدیریت شده و نشده را پشتیبانی کند. دو نمونه از این پشتیبانی را در زیر بیان می‌کنیم:
یک. کدهای مدیریت شده می‌توانند توابع مدیریت شده را در قالب یک dll صدا زده و از آن‌ها استفاده کنند.
دو. کدهای مدیریت شده می‌توانند از کامپوننت‌های COM استفاده کنند: بسیاری از شرکت‌ها از قبل بسیاری از کامپوننت‌های COM را ایجاد کرده بودند که کدهای مدیریت شده با راحتی با آن‌ها ارتباط برقرار می‌کنند. ولی اگر دوست دارید روی آن‌ها کنترل بیشتری داشته باشید و آن کدها را به معادل CLR تبدیل کنید؛ میتوانید از ابزار کمکی که مایکروسافت همراه فریم ورک دات نت ارائه کرده است استفاده کنید. نام این ابزار TLBIMP.exe می‌باشد که از Type Library Importer گرفته شده است.

سه. اگر کدهای مدیریت نشده‌ی زیادتری دارید شاید راحت‌تر باشد که برعکس کار کنید و کدهای مدیریت شده را در در یک برنامه‌ی مدیریت نشده اجرا کنید. این کدها می‌توانند برای مثال به یک Activex یا shell Extension تبدیل شده و مورد استفاده قرار گیرند. ابزارهای exe. TLBEXP و exe. RegAsm برای این منظور به همراه فریم ورک دات نت عرضه شده اند.
سورس کد Type Library Importer را میتوانید در کدپلکس بیابید.
در ویندوز 8 به بعد مایکروسافت API جدید را تحت عنوان WinsowsRuntime یا winRT ارائه کرده است . این api یک سیستم داخلی را از طریق کامپوننت‌های com ایجاد کرده و به جای استفاده از فایل‌های کتابخانه‌ای، کامپوننت‌ها api هایشان را از طریق متادیتاهایی بر اساس استاندارد ECMA که توسط تیم دات نت طراحی شده است معرفی می‌کنند.
زیبایی این روش اینست که کد نوشته شده در زبان‌های دات نت می‌تواند به طور مداوم با apiهای winrt ارتباط برقرار کند. یعنی همه‌ی کارها توسط CLR انجام می‌گیرد بدون اینکه لازم باشد از ابزار اضافی استفاده کنید. در آینده در مورد winRT بیشتر صحبت می‌کنیم.

سخن پایانی: ممنون از دوستان عزیز بابت پیگیری مطالب تا بدینجا. تا این قسمت فصل اول کتاب با عنوان **اصول اولیه CLR** بخش اول **مدل اجرای CLR** به پایان رسید.
ادامه‌ی مطالب بعد از تکمیل هر بخش در دسترس دوستان قرار خواهد گرفت.