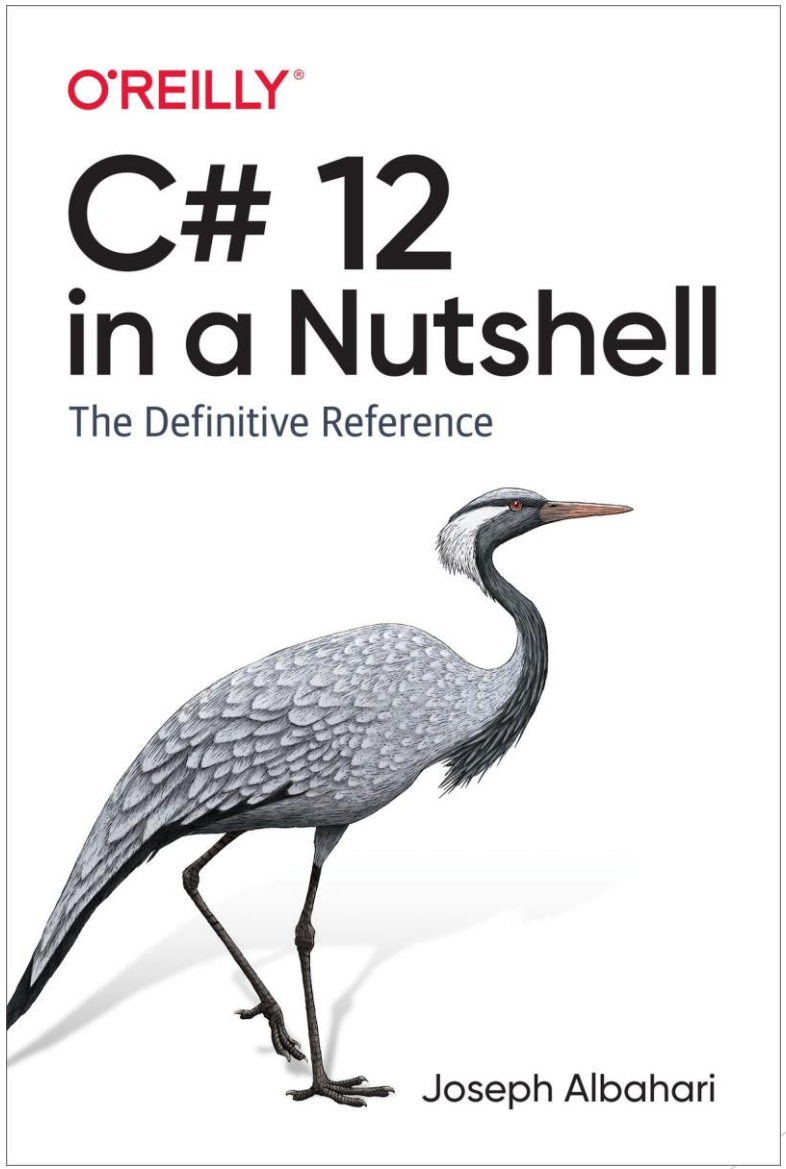
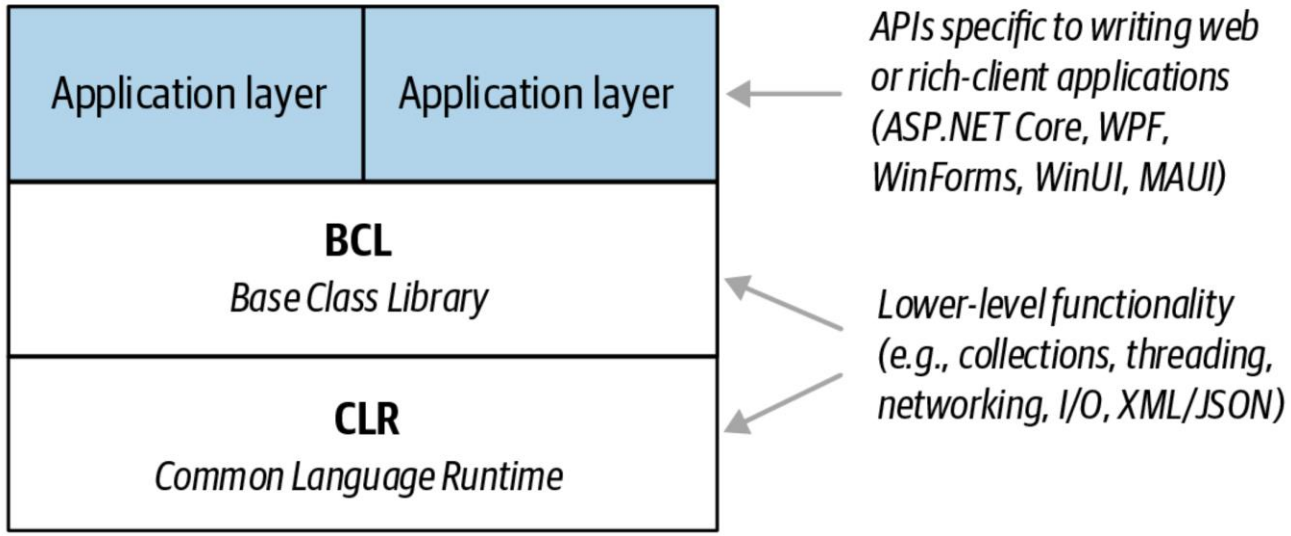
فصل اول - بخش دوم

- CLRs, BCLs, and Runtimes
- CLR
- در حاشیه کتاب
- JIT
- AOT
- Assembly
- Reflection
- BCL
- Runtimes
- .Net 8
- Windows Desktop and WinUI 3
- MAUI
- Avalonia
- .Net Framework



CLRs, BCLs, and Runtimes

Runtime support for C# programs consists of a Common Language Runtime and a Base Class Library. A runtime can also include a higher-level application layer that contains libraries for developing rich-client, mobile, or web applications. Different runtimes exist to allow for different kinds of applications, as well as different platforms.



CLRs, BCLs, and Runtimes

چیزی که کتاب داره میگه و من متوجه شدم اینه که وقتی یه برنامه اجرا میشه ما با یه مفهومی روبرو میشیم به اسم runtime که شامل یه سری چیزها هست از جمله CLR و BCL. همینطور برای برنامه هایی که لایه های بالاتری دارند مثل وب اپلیکیشن ها یا اپلیکیشن های ویندوزی و یا حتی wep api ها یک مفهوم دیگه ای هم در runtime هست به اسم application layer .

Runtime های مختلف برای پشتیبانی از پلتفرم های مختلف به وجود اومدند. (همون عکس اسلاید قبل)

CLR

A Common Language Runtime (CLR) provides essential runtime services such as automatic memory management and exception handling.

CLR سرویسهای اساسی مثل مدیریت حافظه یا هندل کردن خطاها رو فراهم میکنه.

(The word “common” refers to the fact that the same runtime can be shared by other managed programming languages, such as F#, Visual Basic, and Managed C++.)

واژه common یا همون حرف C در CLR اشاره به این موضوع داره که میتونه با سایر زبانهای برنامه نویسی managed مثل visual basic یا F# به اشتراک گذاشته بشه.

CLR

C# is called a managed language because it compiles source code into managed code, which is represented in Intermediate Language (IL).

سی شارپ رو به عنوان یک زبان managed میشناسند، چون کد رو به یک کد managed به اسم IL کامپایل میکنه.

The CLR converts the IL into the native code of the machine, such as X64 or X86, usually just prior to execution. This is referred to as Just-In-Time (JIT) compilation.

در ادامه CLR کد تولید شده IL رو به کد قابل فهم برای ماشین تبدیل میکنه. معمولا این مرحله آخرین مرحله قبل از اجرای برنامه هستش که توسط JIT انجام میشه.

در حاشیه کتاب
managed/ unmanaged code

یه کلمه توی اسلاید قبل ذهنمو درگیر کرد، رفتم یه سرچ کردم ببینم چیه، نتیجه ش شد اینکه:

managed code به کدی میگن که توسط یک runtime مدیریت بشه، که اینجا منظور همون CLR هستش.

در اصل CLR کد سی شارپ رو تبدیل میکنه به یک managed code که بعدش بتونه با کامپایلرهایی که در اختیار داره اونا رو به زبون ماشین تبدیل کنه.

به عبارتی managed code همون کد تبدیل شده به Intermediate Language یا همون IL هستش.

JIT

در اصل JIT یک کامپایلر هستش که وظیفه ش تبدیل کد IL به کد قابل فهم برای ماشین **در زمان اجرا** هستش.

```
1. Source Code (C#)
     |
     | (Compilation to CIL)
     V
2. Common Intermediate Language (CIL) Code (published/deployed)
     |
     | (JIT Compilation at Runtime)
     V
3. Machine Code (Platform-Specific)
     |
     | (Execution)
     V
```

AOT

Ahead-of-time compilation is also available to improve startup time with large assemblies or resource-constrained devices (and to satisfy iOS app store rules when developing mobile apps).

یک کامپایلر دیگه ای که سی شارپ داره اسمش AOT هستش که سرعت اجرا رو برای دیوایس ها با منابع محدود به مراتب بالاتر میبره. (و گویا برای دولوپ برنامه های iOS هم خیلی مناسبه).

AOT

تا اونجا که متوجه شدم، AOT میاد قبل از اجرای برنامه کد رو تبدیل به کد قابل فهم برای ماشین میکنه. در این باره باید بیشتر مطالعه کنم.

```
1. Source Code (C#)
      |
      | (Compilation to CIL)
      V
2. Common Intermediate Language (CIL) Code
      |
      | (AOT Compilation before Runtime)
      V
3. Machine Code (Platform-Specific) (published/deployed)
      |
      | (Execution)
      V
```

Assembly

The container for managed code is called an assembly.

Maneged code ها در یه جایی به اسم assembly نگهداری میشن.

An assembly contains not only IL but also type information (metadata).

البته assembly فقط شامل کد IL نیست و متادیتاها را نیز در خود نگهداری میکند.

The presence of metadata allows assemblies to reference types in other assemblies without needing additional files.

وجود متادیتا به assembly اجازه میده که بدون نیاز به فایل اضافی به همه رفرنس ها دسترسی داشته باشه.

reflection

You can examine and disassemble the contents of an assembly with Microsoft's ildasm tool. And with tools such as ILSpy or JetBrain's dotPeek, you can go further and decompile the IL to C#. Because IL is higher level than native machine code, the decompiler can do quite a good job of reconstructing the original C#.

با استفاده از ابزارهای IL Disassembler شما به راحتی میتونید محتویات یک assembly رو ببینید. همچنین ابزارهای دیگه ای مثل JetBrain's dotPeek هستند که علاوه بر دیدن، میتونید کد IL موجود در assembly رو به c# تبدیل کنید، چرا که IL هنوز سطح بالاتر از زبان ماشین هست و decompiler میتونه برای بازسازی نسخه اصلی کمک زیادی بهمون بکنه.

A program can query its own metadata (reflection) and even generate new IL at runtime (reflection.emit).

یک برنامه میتونه به متادیتای خودش دسترسی داشته باشه یا حتی یک کد IL جدید در زمان اجرا تولید کنه.

BCL

A CLR always ships with a set of assemblies called a Base Class Library (BCL).

CLR همیشه گروهی از assembly ها رو به همراه خودش داره که بهش میگن BCL.

A BCL provides core functionality to programmers, such as collections, input/output, text processing, XML/JSON handling, networking, encryption, interop, concurrency, and parallel programming.

BCL هسته اصلی برنامه نویسی رو برای برنامه نویسان فراهم میکنه، مثل کالکشن ها، ورودی و خروجی، پردازش متن، مسائل امنیتی، شبکه، برنامه های موازی و ... .

A BCL also implements types that the C# language itself requires (for features such as enumeration, querying, and asynchrony) and lets you explicitly access features of the CLR, such as Reflection and memory management.

در کل امکاناتی که یک برنامه نویس نیاز داره رو پیاده سازی میکنه.

runtimes

A runtime (also called a framework) is a deployable unit that you download and install.

Runtime یا فریمورک یک واحد قابل توسعه است که میتونید دانلود و نصبش کنید.

A runtime consists of a CLR (with its BCL), plus an optional application layer specific to the kind of application that you're writing—web, mobile, rich client, etc. (If you're writing a command-line console application or a non-UI library, you don't need an application layer.)

Runtime شامل CLR به همراه BCL هستش. در مواردی که بخواهید نرم افزارهای موبایل یا وب بنویسید، runtime یک بخش application layer هم در اختیارتون میذاره ولی برای برنامه های کنسولی این بخش رو نیاز ندارید.

runtimes

When writing an application, you target a particular runtime, which means that your application uses and depends on the functionality that the runtime provides. Your choice of runtime also determines which platforms your application will support.

وقتی شما میخواهید یه برنامه بنویسید، براساس نیازی که دارید runtime مناسب خودتونو انتخاب میکنید.

Application layer	CLR/BCL	Program type	Runs on...
ASP.NET	.NET 8	Web	Windows, Linux, macOS
Windows Desktop	.NET 8	Windows	Windows 10+
WinUI 3	.NET 8	Windows	Windows 10+
MAUI	.NET 8	Mobile, desktop	iOS, Android, macOS, Windows 10+
.NET Framework	.NET Framework	Web, Windows	Windows 7+

.Net 8

.NET 8 is Microsoft’s flagship open-source runtime. You can write web and console applications that run on Windows, Linux, and macOS; rich-client applications that run on Windows 10+ and macOS; and mobile apps that run on iOS and Android.

دات نت ۸ آخرین runtime متن بازی هست که توسط مایکروسافت ارائه شده که با اون میتونید برنامه های وب و کنسول رو برای سیستم عامل های ویندوز، لینوکس و مک بنویسید. همچنین برنامه های موبایل که روی سیستم عامل های اندروید و ios قابل اجرا هستند.

.Net 8

Unlike .NET Framework, .NET 8 is not preinstalled on Windows machines. If you try to run a .NET 8 application without the correct runtime being present, a message will appear directing you to a web page where you can download the runtime.

دات نت ۸ به صورت پیش فرض روی ویندوز نصب نیست. (برعکس دات نت فریمورک) و اگه بخواهید برنامه ای رو که با دات نت ۸ نوشته شده اجرا کنید با پیغامی روبرو میشید که ازتون میخواد runime دات نت ۸ رو نصب کنید.

You can avoid this by creating a self-contained deployment, which includes the parts of the runtime required by the application.

البته میتونید با ایجاد یک self-contained deployment از این اتفاق جلوگیری کنید. (این همون AOT هستش که گفت نیاز نداره runtime نصب بشه)

.Net 8

.NET's update history runs as follows: .NET Core 1.x → NET Core 2.x → .NET Core 3.x → .NET 5 → .NET 6 → .NET 7 → .NET 8. After .NET Core 3, Microsoft removed "Core" from the name and skipped version 4 to avoid confusion with .NET Framework 4.x, which precedes all of the preceding runtimes but is still supported and in popular use.

This means that assemblies compiled under .NET Core 1.x → .NET 7 will, in most cases, run without modification under .NET 8. In contrast, assemblies compiled under (any version of) .NET Framework are usually incompatible with .NET 8.

خلاصه میخواد اینو بگه که ارتقای ورژن دات نت راحته و باهم سازگارند ولی طبیعتا با دات نت فریمورک سر سازگاری ندارند

Windows Desktop and WinUI 3

For writing rich-client applications that run on Windows 10 and above, you can choose between the classic Windows Desktop APIs (Windows Forms and WPF) and WinUI 3. The Windows Desktop APIs are part of the .NET Desktop runtime, whereas WinUI 3 is part of the Windows App SDK (a separate download).

برای نوشتن برنامه های دسکتاپ که روی ویندوز ۱۰ یا بالاتر اجرا بشه دو انتخاب وجود داره:

- Windows Desktop APIs که بخشی از .NET Desktop runtime هستش(همون ویندوز فرم های قدیمی و همینطور WPF)
- WinUI 3 که بخشی از Windows App SDK هستش و باید جداگانه نصب بشه.

Windows Desktop and WinUI 3

The classic Windows Desktop APIs have existed since 2006 and enjoy terrific third-party library support, as well as offering a wealth of answered questions on sites such as StackOverflow. WinUI 3 was released in 2022 and is intended for writing modern immersive applications that feature the latest Windows 10+ controls. It is a successor to the Universal Windows Platform (UWP).

Windows Desktop از سال ۲۰۰۶ معرفی شه و محتوای خیلی زیادی ازش در سطح وب وجود داره.

WinUI 3 سال ۲۰۲۲ معرفی شده و برای نوشتن برنامه های مدرن برای ویندوز ۱۰ و بالاتر کاربرد داره و یه جورایی جایگزین UWP هستش.

MAUI

MAUI (Multi-platform App UI) is designed primarily for creating mobile apps for iOS and Android, although it can also be used for desktop apps that run on macOS and Windows via Mac Catalyst and WinUI 3.

MAUI در اصل برای ساخت برنامه های موبایل سازگار با اندروید و iOS طراحی شده، اگر چه میتونه برای تولید برنامه های دسکتاپ که روی مک با Mac Catalyst و ویندوز با WinUI 3 هم استفاده بشه.

MAUI is an evolution of Xamarin and allows a single project to target multiple platforms.

MAUI در تکامل زامازین خلق شده و اجازه میده با یک پروژه برای پلتفرم های مختلف برنامه تولید کنیم.

Avalonia

For cross-platform desktop applications, a third-party library called Avalonia offers an alternative to MAUI. Avalonia also runs on Linux and is architecturally simpler than MAUI (as it operates without the Catalyst/WinUI indirection layer).

Avalonia هم یک کتابخانه دیگر است که برای ساخت برنامه های قابل اجرا روی چند پلتفرم مورد استفاده قرار میگیره. این کتابخانه قابلیت اجرا روی لینوکس را هم دارد و از نظر معماری ساده تر از MAUI هست.

Avalonia has an API similar to WPF, and it also offers a commercial add-on called XPF that provides almost complete WPF compatibility.

Avalonia مثل WPF دارای یک api هست و همچنین یک افزونه به اسم XPF را ارائه میدهد که تقریبا سازگاری کاملی با WPF دارد. (اینم جزو بدهی هام در نظر میگیرم که حتما برم و یادش بگیرم)

.NET Framework

.NET Framework is Microsoft’s original Windows-only runtime for writing web and rich-client applications that run (only) on Windows desktop/server. No major new releases are planned, although Microsoft will continue to support and maintain the current 4.8 release due to the wealth of existing applications.

دات نت فریمورک یک runtime مایکروسافت است که برای نوشتن برنامه های تحت وب و دسکتاپ استفاده میشه و فقط روی سیستم عامل ویندوز اجرا میشه. مایکروسافت برای توسعه اون پلنی نداره هرچند به علت وجود برنامه های بزرگ و ارزشمندی که با این فریمورک یا runtime نوشته شده آخرین نسخه ارائه شده یعنی 4.8 رو همچنان پشتیبانی میکنه.

.NET Framework

With the .NET Framework, the CLR/BCL is integrated with the application layer.

با دات نت فریمورک، CLR و BCL با application layer یکپارچه میشود.

Applications written in .NET Framework can be recompiled under .NET 8, although they usually require some modification. Some features of .NET Framework are not present in .NET 8 (and vice versa).

برنامه هایی که با دات نت فریمورک نوشته شده اند میتونند با دات نت ۸ هم مجدد کامپایل بشند، هرچند معمولا نیاز به یک سری تغییرات هست، چرا که بعضی از ویژگی های دات نت فریمورک در دات نت ۸ وجود نداره و برعکس برخی از ویژگی های دات نت ۸ در دات نت فریمورک تعریف نشده اند.

.NET Framework

.NET Framework is preinstalled with Windows and is automatically patched via Windows Update.

دات نت فریمورک همراه با ویندوز نصب میشود و با آپدیت ویندوز هم آپدیت میشود.

When you target .NET Framework 4.8, you can use the features of C# 7.3 and earlier. (You can override this by specifying a newer language version in the project file—this unlocks all of the latest language features except for those that require support from a newer runtime.)

وقتی پروژه ای با دات نت فریمورک 4.8 ایجاد میشه، شما میتونید از قابلیت های C# تا نسخه 7.3 استفاده کنید. البته شما میتونید با اصلاح ورژن در فایل پروژه این محدودیت را بردارید و همه ویژگی های جدید زبان (به جز آنهایی که وابسته به فریمورک است) برای شما فعال شود.