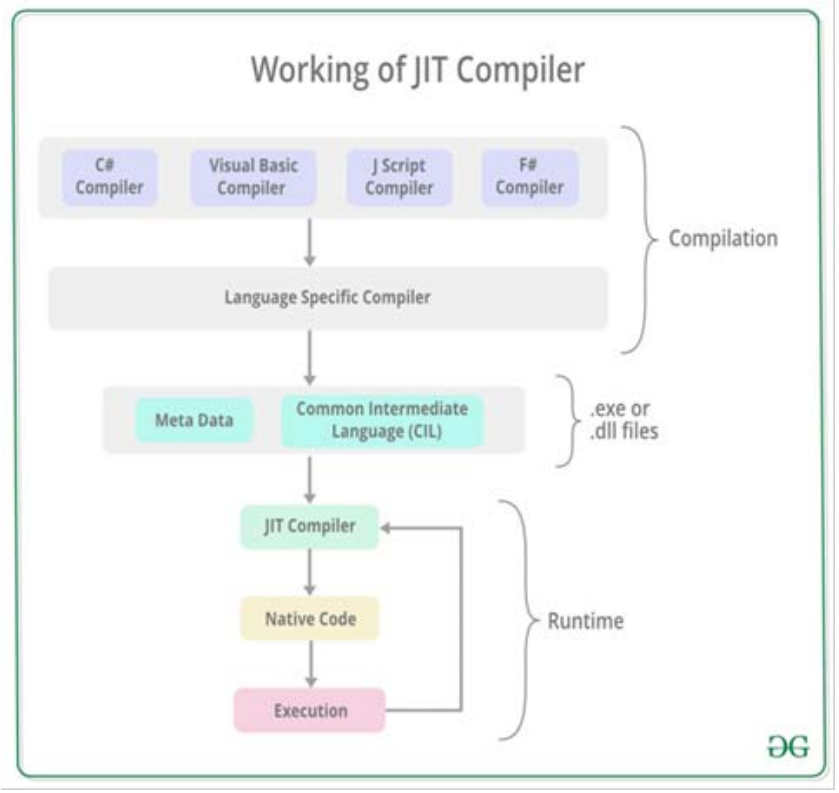
託管程式碼與非託管程式碼

大綱

- .NET 託管程式碼 (managed code)
- Native 非託管程式碼 (unmanaged code)
- .NET Native → Universal Windows Platform
- 託管程式碼 [C#] 如何使用非託管程式碼 [C++]
  - ✓ 利用 P/invoke 呼叫非託管 C++
  - ✓ 利用 C++/CLI 作為代理中間層

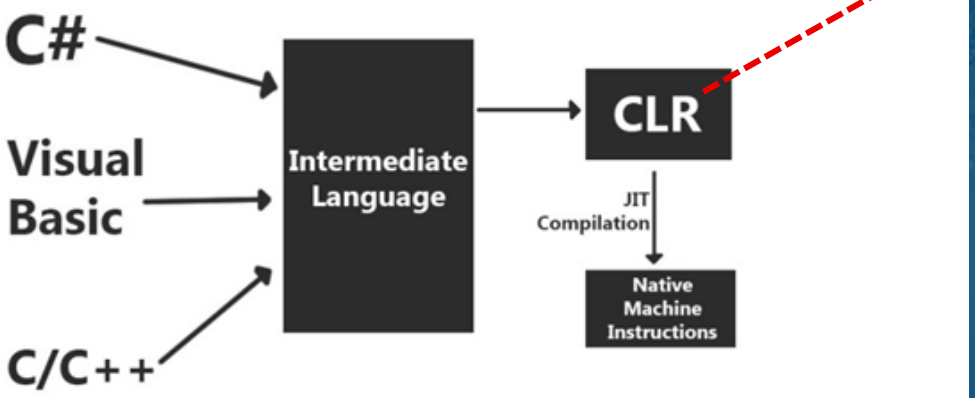
.NET 託管程式碼

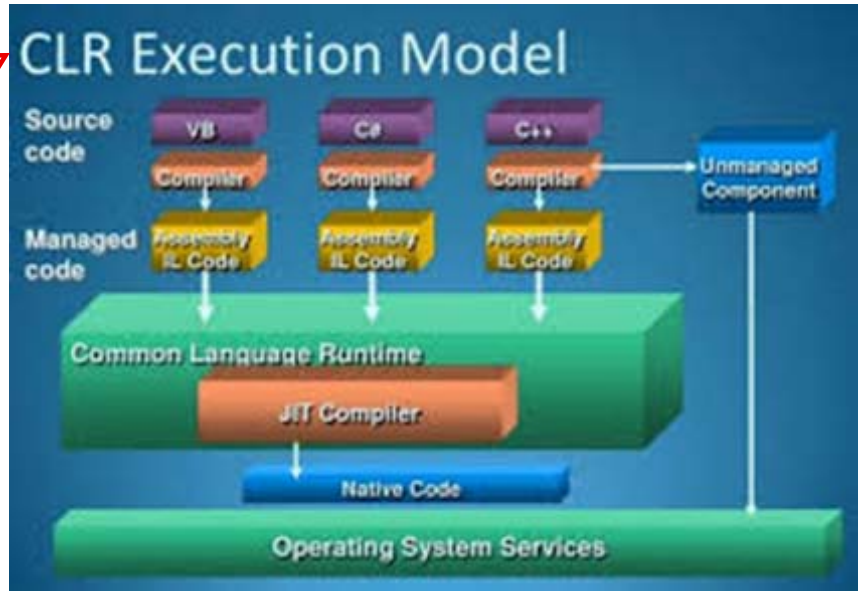
- .NET 應用程式會編譯成中繼語言 (IL)。
- 在執行階段，Just-In-Time (JIT) 編譯器會將 IL 轉譯成機器碼。



.NET 託管程式碼（續）

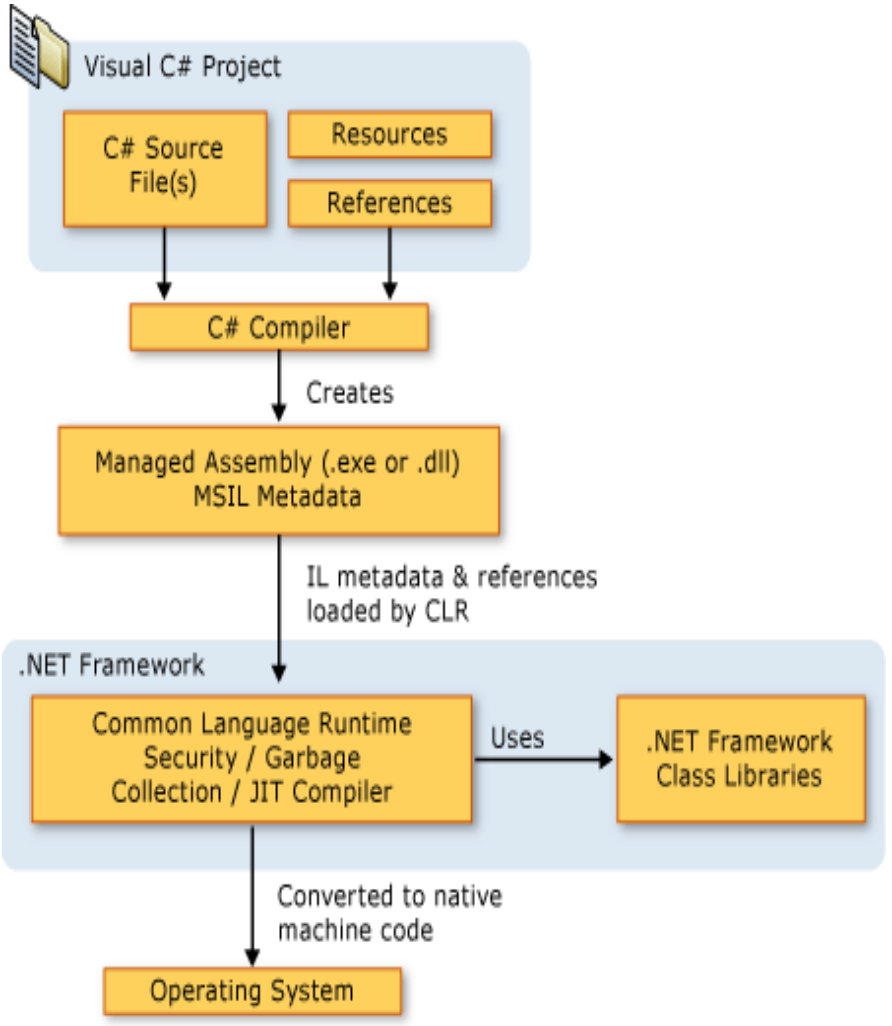
- C# 程式其實經過了兩次編譯，第一次由 C# 編譯器進行編譯，第二次由 JIT 進行編譯。
- C# 編譯器進行編譯的過程稱為「**編譯時期**」，而到 CLR 執行程式 (包含 JIT 編譯) 稱呼為「**執行時期**」。





.NET 託管程式碼（續）

- .NET 程式被載入入記憶體以後，當某段 IL 程式碼被第一次執行的時候，JIT 編譯器就會將這段 IL 程式碼，全部編譯成機器碼，然後再執行。
- 這就是 **.NET 程式第一次執行啟動都很慢的原因**。
- Visual Basic.NET 和 C# **只能產生託管程式碼**。用這類語言寫程式，所產生的程式碼就是託管程式碼。



Native 非託管程式碼

- 非託管程式碼不能享受 .NET Framework 所提供的服務，例如安全和記憶體管理等。
- 非託管程式碼需要進行記憶體管理等服務，就必須**呼叫 Windows SDK 提供的 API 來實現**。

SDK

- 在Windows 這一領域，**SDK** 是Software Development Kit的縮寫，意思是軟體開發工具箱，爲開發者提供開發視窗應用程式的必要檔案(包含API(Application Programming Interfaces)函式庫，標頭檔及例子程式)。(不同的作業系統都可能有自己的SDK)。
- 凡以Windows API (編制32位元應用程式所使用的函數,結構,巨集定義)撰寫的程式我們通常也稱爲SDK 程式。也有人把Windows API 稱爲SDK API。
  - API是Windows提供給應用程式的編程介面,任何用應用程式必須建立在API之上
  - API是一個程式庫,提供各式各樣與Windows系統服務有關的函數,例如某些C的標準函式庫也是呼叫了API來達成其功能的,如create()即是呼叫API 的CreatFile,事時上,在windows下執行的程式最終都是透過API來完成工作的
  - 因此可將Win32 API看成是最底層的服務
- 早期一般稱使用SDK來開發應用程式,即使用C/C++呼叫Windows API函式,來開發視窗應用程式

Native 非託管程式碼（續）

- 與 Visual Studio 中的其他 Microsoft 語言不同，**Visual C++ 可以建立非託管應用程式**。建立專案並選擇名稱以 MFC、ATL 或 Win32 開頭的應用程式型別時，表示建立一個非託管應用程式。。
- 建立託管應用程式時，產出是副檔名為 .exe 的 CIL 程式碼。當非託管應用程式，產出是 Windows 可執行檔案的本機程式碼，其副檔名也同樣為 .exe ，但**這兩個檔案的內部佈局完全不同**。

.NET Native

- .NET Native 包含在 Visual Studio 2015 和更新版本中。它會自動編譯以託管程式碼撰寫的 Universal Windows Platform (UWP) 應用程式發行版本 (C# 或 Visual Basic.Net) 至機器碼。
- .NET Native 能夠為託管程式碼開發人員帶來 C++ 的效能優勢，因為它在幕後使用與 C++ 相同或類似的工具。
- 應用程式會以原生程式碼的效能為特色，而且可以利用 .NET Framework 所提供的資源，包括其類別庫、自動記憶體管理和垃圾收集，以及例外狀況處理。

託管程式碼如何使用非託管程式碼

- 在託管程式碼中可使用 COM Interop 、 P/Invoke 及 C++/CLI 來呼叫非託管程式碼的 COM 元件或 DLL 。
- COM Interop 是一種讓 .NET Framework 的程式能夠和 COM 的程式相互操作的一種橋接技術，是 .NET Framework 互通性的一環。
- COM Interop 可以讓 .NET Framework 的程式使用 COM 元件，也可以讓 COM 程式使用 .NET Framework 的元件。
- 例如讓 .NET Framework 的程式可以呼叫舊有的 ActiveX 元件。

託管程式碼如何使用非託管程式碼

- COM Interop 的服務是由 **System.Runtime.InteropServices** 命名空間中的類別來提供，其中最重要的是 **Marshal 類別**，它提供了託管程式碼和非託管程式碼之間的資料格式與指標轉換。
- **P/Invoke 技術**可從託管程式碼存取非託管函式庫中的函式。
- P/Invoke API 都包含在兩個命名空間中：**System** 和 **System.Runtime.InteropServices**。

託管程式碼如何使用非託管程式碼

- 應用 COM Interop / P/Invoke ，必須要先知道的幾件事：
  - C# 無法直接呼叫 C++ API 的 DLL 檔，這是因為直接用 C++ 產生的 DLL 檔為非託管程式碼。
  - 傳送字串需要注意你的資料格式是 ANSI 還是 Unicode 。
  - **32 位元 / 64 位元 的 DLL 並不存在向上向下相容**，所有程式碼包含呼叫方的程式碼，只能符合一致性，調用正確位元的 DLL ，否則會出錯。

託管程式碼如何使用非託管程式碼

- 在 C#(託管程式碼) 使用 C++ 的 DLL(非託管程式碼)，有兩種方法：
  - **利用 P/Invoke 呼叫非託管 C++ 程式碼。**
  - **利用 C++/CLI 作為代理中間層。**
- 利用 P/Invoke 呼叫非託管 C++ 程式碼
  - C# 編寫的是託管程式，編譯生成中間語言 (CIL)，而 C++ 程式則編譯生成本地機器碼，這兩種語言進行混合程式設計存在一定困難。常用的方法是使用 **DllImport** 的方法，但是這種方法只能匯出函式，卻沒辦法匯出非託管的 C++ 類別。
  - C# 通過 P/Invoke 調用 C/C++ 函式。
  - extern “C” 聲明，表示函式和變數是按照 C 語言的方式編譯和連結的。
  - extern “C” _declspec(dllexport) 的目的，是為了使用 DllImport 引用非託管 C++ 的 DLL 檔；**使用 DllImport 只能引用由 C 語言函式的 DLL。**

託管程式碼如何使用非託管程式碼

利用 C++/CLI 作為代理中間層

- 除了 C# 、非託管 C++ 外，C 系列中還存在一種語言叫做託管 C++ ，這種語言語法上和非託管 C++ 幾乎一樣，但是卻和 C# 一樣編譯成為微軟中間語言，這樣託管 C++ 就可以和 C# 良好地通信，即可以在 C# 中使用託管 C++ 類。
- **託管 C++ 已經被廢棄，C++ 託管程式碼的現代擴展叫做 C++/CLI 。**
- C++/CLI 是 C++ 的 .NET 實現，微軟為了使 C++ 開發人員能更容易掌握託管框架 (managed framework) 設計了 C++/CLI 。
- **C++/CLI 的程式碼可以嵌入非託管 C++ 編譯的機器碼。引用非託管 C++ 的類別和函式；也就是用 C++/CLI 給非託管 C++ 程式碼做一個外殼包裝供 C# 引用。**
- C++/CLI 為 C# / .NET 和非託管 C++ 程式碼提供二者之間的橋樑；但很少看到單獨使用 C++/CLI 的作為應用程式開發工具。