BCSCHOOL

Empowering Digital Skills For The Jobs Of The Future

by

BC Soft
consulenza sistemi informativi

AiRED
Artificial Intelligence Ricerca e Sviluppo

Academy .NET

.NET Framework

Sommario

- Microsoft .NET Framework
- Caratteristiche principali
- Indipendenza della piattaforma
- Architettura .NET Framework
- Il CLR
- Common Type System(CTS)
- Common Language Infrastructure (CLI)
- just-in-time compilation (JIT)
- Virtual Execution System (VES)
- .NET assemblies
- Base Class Library (BCL)
- Managed Code
- Architettura .NET Framework
- .NET Native

Microsoft .NET Framework

- **Microsoft .NET Framework** è un componente software che può essere aggiunto o incluso nel sistema operativo Microsoft Windows.

- Fornisce una vasta gamma di soluzioni precodificate ai requisiti dei programmi comuni e gestisce i programmi scritti appositamente per il framework.

Microsoft .NET Framework

.NET Framework è concepito per essere utilizzato dalla maggior parte delle nuove applicazioni create per la piattaforma Windows.

Caratteristiche principali

Sicurezza: .NET consente l'esecuzione del codice con diversi livelli di affidabilità

Distribuzione semplificata: l'installazione del software per computer deve essere gestita con attenzione per garantire che non interferisca con il software installato in precedenza.

Il framework .NET include funzionalità di progettazione e strumenti che aiutano a soddisfare questi requisiti.

Caratteristiche principali

- **Common Runtime Engine** - I linguaggi di programmazione su .NET Framework vengono compilati in un linguaggio intermedio noto come Common Intermediate Language o CIL;

- L'implementazione di **CIL** da parte di Microsoft è nota come Microsoft Intermediate Language o MSIL.

Caratteristiche principali

- Questo linguaggio intermedio non viene interpretato, ma piuttosto compilato in un modo noto come compilazione just-in-time (**JIT**) nel codice nativo.

- La combinazione di questi concetti è chiamata Common Language Infrastructure (**CLI**)

- L'implementazione della CLI da parte di Microsoft è nota come Common Language Runtime (**CLR**).

Caratteristiche principali

- **Indipendenza dal linguaggio**: .NET Framework introduce un Common Type System, o CTS.

- **La specifica CTS** definisce tutti i possibili tipi di dati e costrutti di programmazione supportati dal CLR e il modo in cui possono o non possono interagire tra loro.

- A causa di questa funzionalità, .NET Framework supporta lo sviluppo in **più linguaggi di programmazione**

Caratteristiche principali

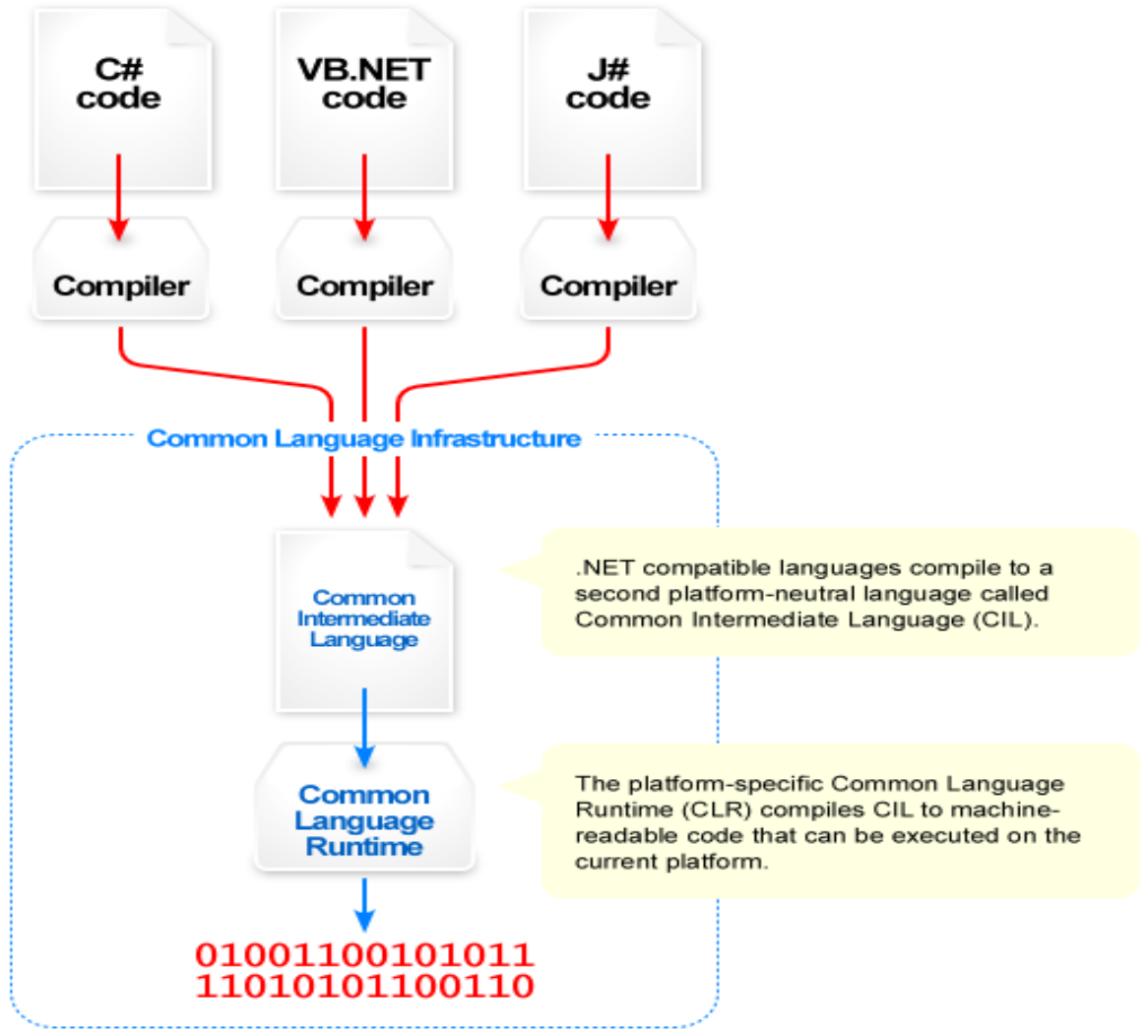
- **Libreria di classi di base** - La libreria di classi di base (BCL), a volte indicata come **Framework Class Library** (FCL), è una libreria di tipi disponibili per tutti i linguaggi che utilizzano .NET Framework.

- Il **BCL** fornisce classi che incapsulano una serie di funzioni comuni, tra cui lettura e scrittura di file, rendering grafico, interazione con database e manipolazione di documenti XML.

Architettura .NET Framework

- **Common Language Infrastructure (CLI)**
- **Assemblies**
- **Metadati**
- **Libreria di classi di base (BCL)**

Architettura .NET Framework



Il CLR

- I programmi scritti per .NET Framework vengono eseguiti in un ambiente software che gestisce i requisiti di runtime del programma.

- Questo ambiente di runtime, che fa anche parte di .NET Framework, è noto come **Common Language Runtime** (**CLR**).

Il CLR

- CLR fornisce importanti servizi come **meccanismi di sicurezza, gestione della memoria e gestione delle eccezioni**.

- La libreria di classi e CLR insieme compongono il .NET Framework.

Il CLR

- L'implementazione della CLI da parte di Microsoft è denominata **Common Language Runtime o CLR**.

- Il CLR è composto da quattro parti principali:

- Common Type System(CTS)
- Common Language Specification (CLS)
- Just-In-Time Compiler (JIT)
- Virtual Execution System(VES)

Common Type System(CTS)

- Il **Common Type System (CTS)** viene utilizzato da ogni linguaggio costruito su .NET Framework.

- Una parte fondamentale di .NET Framework è il Common Language Runtime (CLR),

- il **CTS** non specifica alcuna sintassi o parole chiave particolari, ma definisce invece un **insieme comune di tipi** che possono essere utilizzati con molti linguaggi diversi.

Common Type System(CTS)

Ogni linguaggio è libero di definire qualsiasi sintassi, ma se quel linguaggio è costruito sul CLR, utilizzerà almeno alcuni dei tipi definiti dal CTS.

Common Type System(CTS)

Mentre il creatore di un linguaggio basato sul CLR è libero di implementare solo un sottoinsieme dei tipi definiti da CTS. Visual Basic.NET, C # e quasi tutti gli altri linguaggi utilizzati con .NET Framework si basano molto sul CTS.

Common Type System(CTS)

- Il CTS fornisce a ogni linguaggio in esecuzione sulla piattaforma .NET un **set di base di tipi di dati**.

- Sebbene CTS sia responsabile della definizione dei tipi che possono essere utilizzati nei linguaggi .NET, la maggior parte dei linguaggi sceglie di implementare alias per quei tipi.

Common Language Infrastructure (CLI)

Il Common Language Infrastructure (CLI) è una specifica sviluppata da Microsoft che **descrive il codice eseguibile e l'ambiente di runtime** che costituiscono il nucleo di Microsoft .NET Framework.

Common Language Infrastructure (CLI)

Definisce un ambiente che consente di utilizzare più linguaggi di alto livello su diverse piattaforme di computer senza essere riscritti per architetture specifiche.

Common Language Infrastructure (CLI)

La specifica CLI descrive i seguenti quattro aspetti:

Il Common Type System (CTS)

- Un insieme di tipi e operazioni condivisi da tutti i linguaggi di programmazione compatibili con CTS.

Metadati

- Le informazioni sulla struttura del programma sono indipendenti dalla lingua, quindi è possibile fare riferimento a linguaggi e strumenti, semplificando il lavoro con codice scritto in una lingua che non si utilizza.

just-in-time compilation (JIT)

- In informatica, la **compilazione just-in-time (JIT)**, nota anche come traduzione dinamica, è una tecnica per migliorare le prestazioni di runtime di un programma per computer.

- **Converte, in fase di esecuzione, il codice da un formato a un altro**, ad esempio bytecode in codice macchina nativo.

just-in-time compilation (JIT)

JIT si basa su due idee precedenti in ambienti run-time: **compilazione bytecode** e **compilazione dinamica**

just-in-time compilation (JIT)

In un ambiente JIT, la **compilazione del bytecode** è il primo passo.

Riduce il codice sorgente a una rappresentazione intermedia portatile e ottimizzabile.

- Il bytecode viene distribuito sul sistema di destinazione.
- Quando il codice viene eseguito, il compilatore dell'ambiente di runtime lo traduce in **codice macchina nativo**.

just-in-time compilation (JIT)

Questo può essere fatto per file, per funzioni o anche su qualsiasi frammento di codice arbitrario;

Il codice può essere compilato quando sta per essere eseguito (da cui il nome "just-in-time").

just-in-time compilation (JIT)

JIT offre prestazioni di gran lunga migliori rispetto agli interpreti poiché il codice compilato viene memorizzato nella cache di memoria in fase di esecuzione.

just-in-time compilation (JIT)

Tuttavia JIT potrebbe avere uno svantaggio provocando un leggero ritardo nell'esecuzione iniziale di un'applicazione, a causa del tempo impiegato per compilare il bytecode.

just-in-time compilation (JIT)

Altri vantaggi di un runtime JIT includono:

- La compilazione può essere ottimizzata per la CPU di destinazione e il modello di sistema operativo in cui viene eseguita l'applicazione.

- Il sistema è in grado di raccogliere statistiche su come il programma è effettivamente in esecuzione nell'ambiente in cui si trova.

just-in-time compilation (JIT)

La compilazione JIT è considerata vantaggiosa in molti scenari grazie alla sua portabilità, flessibilità e prestazioni.

L'unico inconveniente principale è il ritardo del tempo di avvio.

Virtual Execution System (VES)

Il Virtual Execution System (VES) fornisce un ambiente per l'esecuzione di codice gestito.

Lo scopo del VES è fornire il supporto necessario per eseguire il set di istruzioni del **Common Intermediate Language**.

.NET assemblies

- Il codice **MSIL** intermedio è ospitato negli assembly .NET, che per l'implementazione di Windows significa un file PE (Portable Executable) (EXE o DLL)

- Gli **assembly** sono l'unità .NET di distribuzione, controllo delle versioni e sicurezza.

.NET assemblies

- L'assembly è costituito da uno o più file, ma uno di questi deve contenere il **manifest**, che contiene i metadati per l'assembly.

- Il nome completo di un assembly contiene il **nome**, il **numero di versione**, **culture** e il **token della chiave pubblica**; il nome è obbligatorio mentre i restanti sono facoltativi.

.NET assemblies

Il token della chiave pubblica viene generato quando viene creato l'assembly ed è un valore che rappresenta in modo univoco il nome e il contenuto di tutti i file dell'assembly e una chiave privata nota solo al creatore dell'assembly.

.NET metadata

- Tutti i CIL si autodescrivono tramite **metadati .NET**.
- Il CLR controlla i metadati per garantire che venga chiamato il metodo corretto.
- I metadati vengono generalmente generati dai compilatori di linguaggio, ma gli sviluppatori possono creare i propri metadati tramite attributi personalizzati.
- I metadati contengono anche tutte le informazioni sull'assembly.

Base Class Library (BCL)

- La Base Class Library (BCL) è una libreria di classi disponibile per tutte le lingue che utilizzano .NET Framework.

- La BCL fornisce classi che incapsulano una serie di funzioni comuni come la lettura e la scrittura di file, il rendering grafico, l'interazione con il database, la manipolazione di documenti XML e così via.

Managed Code

Nella terminologia di Microsoft Windows, il codice gestito è costituito da istruzioni del computer.

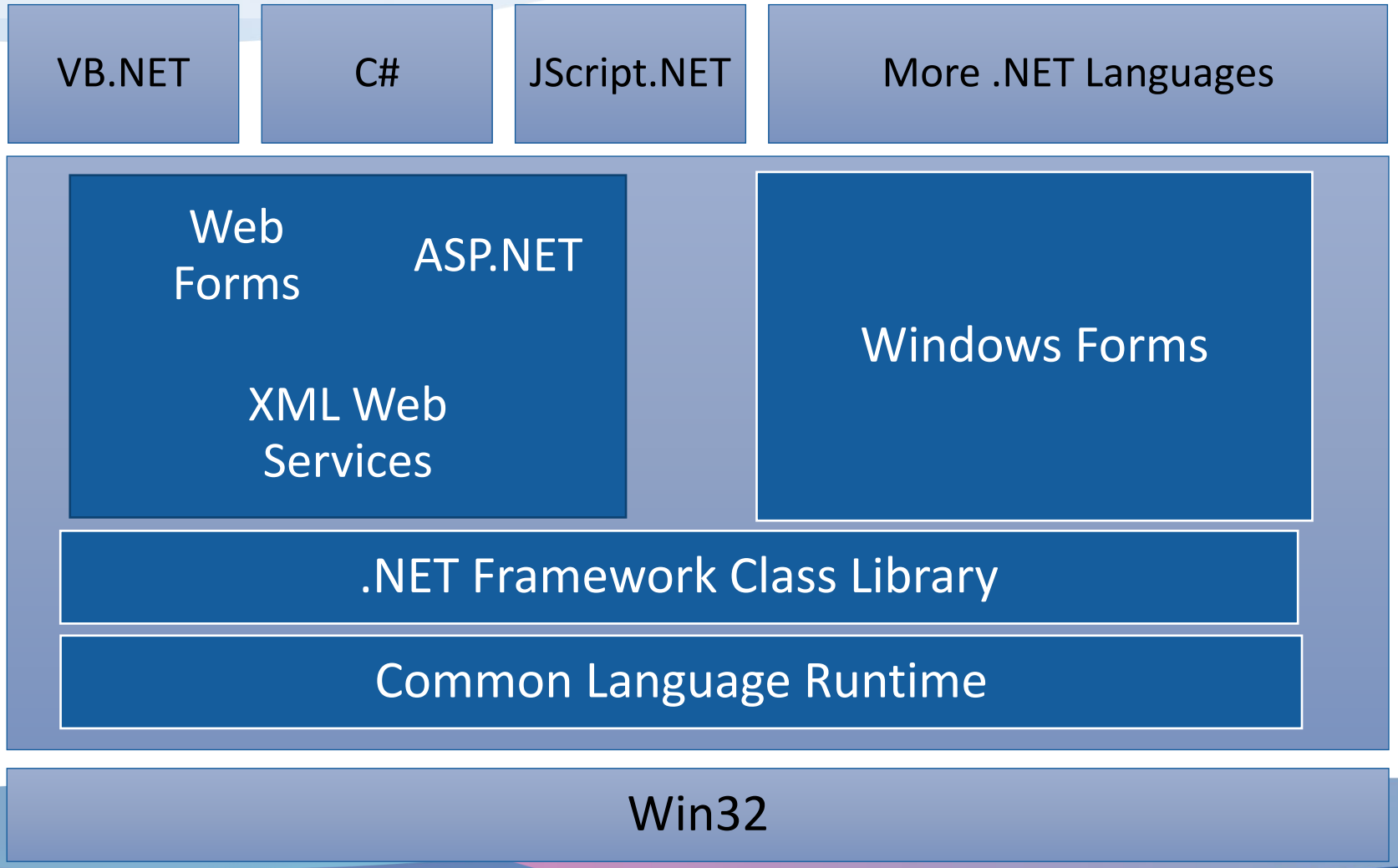
Ovvero "codice", eseguito da una macchina virtuale conforme a CLI, come .NET Framework **Common Language Runtime** di Microsoft o altre implementazioni CLI dal The Mono Project o DotGNU Project.

Managed Code

Prima che il codice venga eseguito, l' **Intermediate Language** viene compilato in **codice macchina nativo**.

Questa viene tradizionalmente definita compilazione Just-in-time.

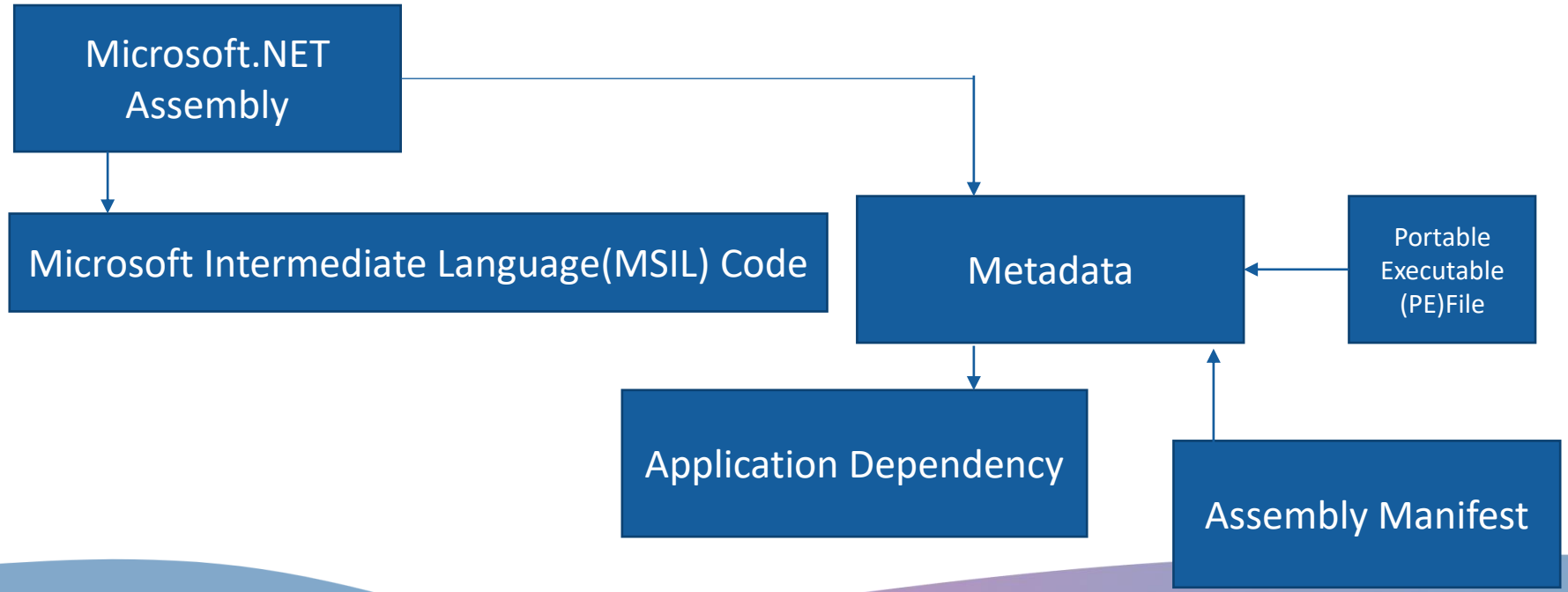
Architettura .NET Framework



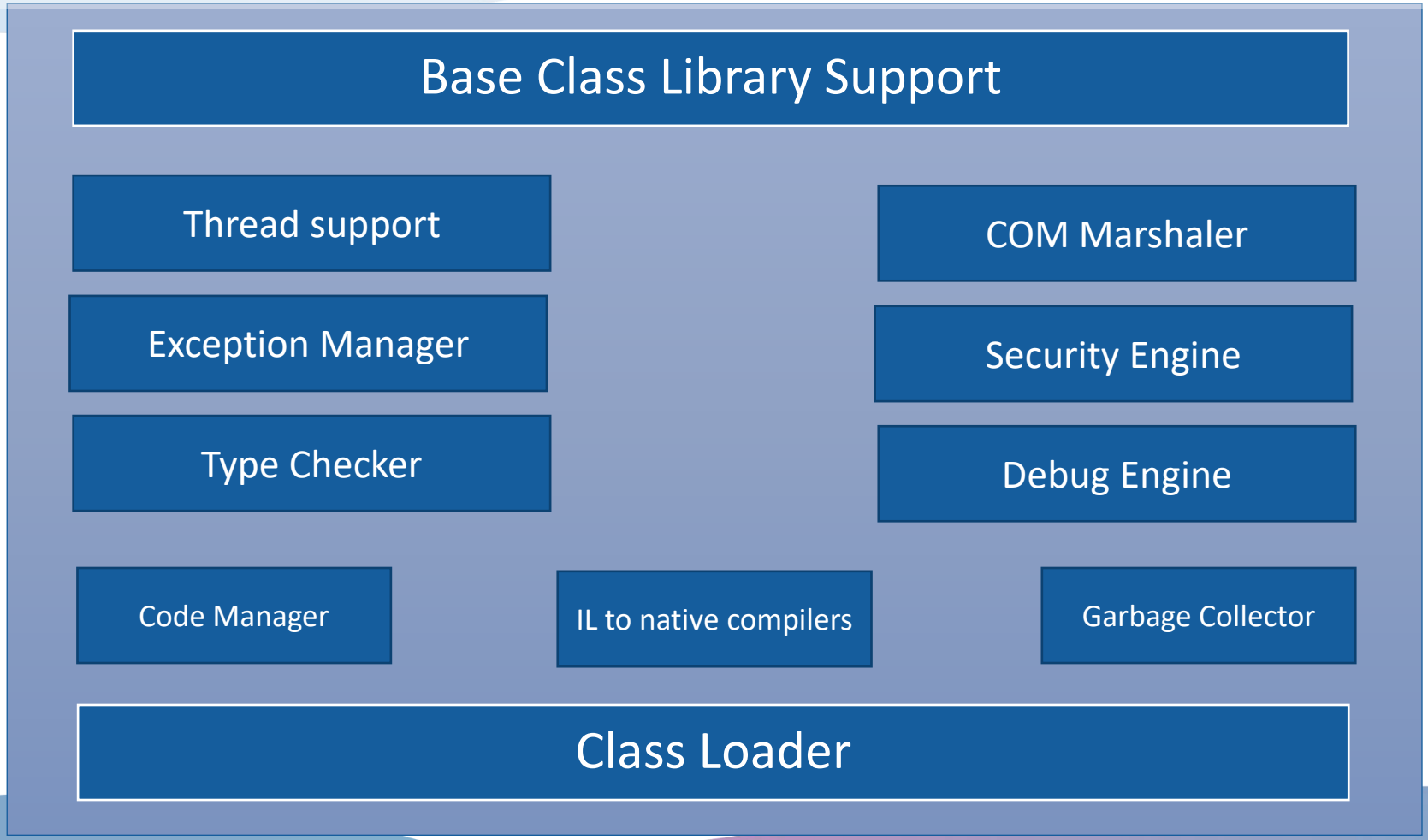
.NET Assembly

Gli assembly sono costituiti da codice IL e metadati.

I metadati determinano le dipendenze dell'applicazione.



Architettura .NET Framework CLR



Architettura .NET Framework CLR

Il **Class Loader**, che carica le classi in CLR:

- **Compila da MSIL a codice nativo**
- **Code manager:** gestisce il codice durante l'esecuzione.
- **Allocazione della memoria e Garbage Collector:** esegue la gestione automatica della memoria.
- **Type checker**: che impone un rigoroso controllo del tipo

Architettura .NET Framework CLR

- **Motore di sicurezza**: questo applica restrizioni di sicurezza come sicurezza a livello di codice a livello di cartella e sicurezza a livello di macchina.

- **Gestore eccezioni:** fornisce un meccanismo per gestire le eccezioni in fase di esecuzione.

Architettura .NET Framework CLR

- **Motore di debug**: consente allo sviluppatore di eseguire il debug di diversi tipi di applicazioni.
- **COM Marshaler**: consente alle applicazioni .NET di scambiare dati con le applicazioni COM.
- **Supporto della libreria di classi di base**: fornisce le classi (tipi) necessarie alle applicazioni in fase di esecuzione.

Caratteristiche del Common Language Runtime

- Gestisce la memoria: Allocazione della memoria, De-Allocation of Memory (garbage collation)
- Supporto per l'esecuzione dei thread
- Esecuzione del codice
- Verifica della sicurezza del codice

Caratteristiche del Common Language Runtime

- Compilazione Da MSIL a codice nativo

- Protezione del codice basata sull'attendibilità (autorizzazione concessa per eseguire codice: Livello codice, livello cartella, livello macchina)

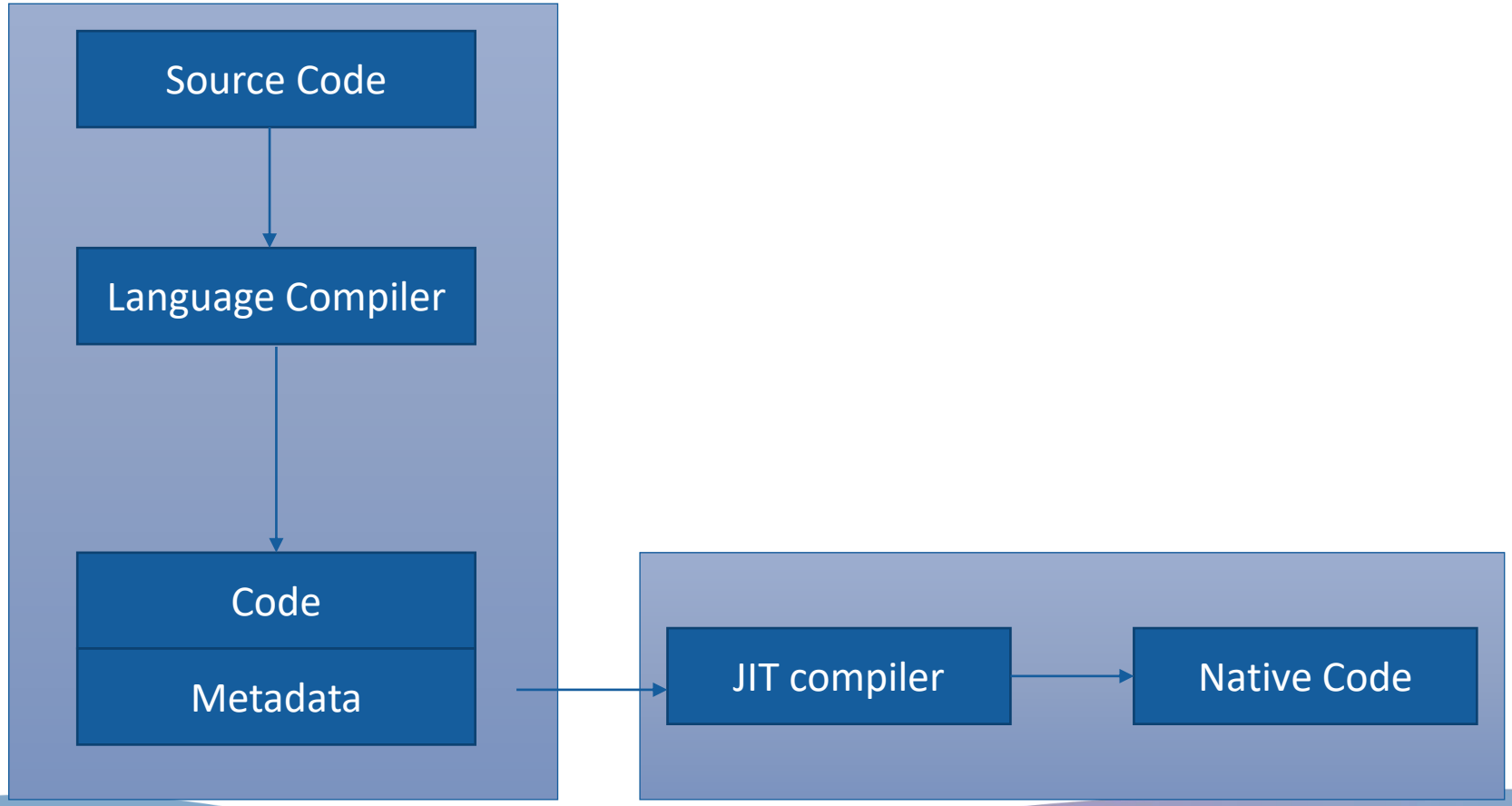
Caratteristiche del Common Language Runtime

Il **compilatore** del linguaggio compila il codice sorgente nel codice MSIL che consiste in codice indipendente dalla CPU e istruzioni indipendenti dalla piattaforma.

MSIL si compone di quanto segue:

- Istruzioni che consentono di eseguire operazioni aritmetiche e logiche
- Accesso diretto alla memoria
- Controlla il flusso di esecuzione
- Gestisce le eccezioni

Caratteristiche del Common Language Runtime



Caratteristiche del Common Language Runtime

- Il compilatore **JIT** è parte integrante del CLR.
- Il compilatore compila il codice MSIL in codice nativo ed esegue il batch di codice Just in time che verrà memorizzato nella cache
- La volta successiva il codice verrà eseguito dalla cache invece di essere compilato di nuovo.
- Il CLR Class Loader carica il codice MSIL e i metadati

Caratteristiche del Common Language Runtime

- Il code manager chiama il metodo del punto di ingresso
- Il compilatore JIT compila il metodo prima dell'esecuzione del metodo entry point.
- Il code manager inserisce gli oggetti in memoria e controlla l'esecuzione del codice.
- Il Garbage Collector esegue controlli periodici sull'heap gestito per identificare gli oggetti che non sono in uso dall' applicazione.

Caratteristiche del Common Language Runtime

Al momento dell'esecuzione del programma, il **type checker** garantisce che tutti gli oggetti, i valori e le referenze degli oggetti e dei valori siano di tipo valido.

Il **type checker** assicura inoltre che sul codice vengano eseguite solo operazioni valide, altrimenti verrà generata l'eccezione.

.NET Native

.NET Native è la tecnologia di precompilazione che permette la **generazione automatica di codice nativo** anziché codice intermedio IL (che viene di seguito tradotto in coice nativo dal compilatore JIT).

.NET Native include staticamente nelle applicazioni le parti necessarie di .NET, ossia tutte le librerie e componenti utilizzati, generando un singolo file eseguibile.

.NET Native

Permette di migliorare le prestazioni in fase di avvio e di esecuzione., ottimizza la memoria e semplificandone distribuzione e aggiornamento dell'applicazione.

.NET Native è disponibile per applicazioni Windows 10 (UWP), Hololens e dispositivi IoT.

Domande & approfondimenti

Academy .NET