Capitolo 1
Il linguaggio C#

Dopo quanto visto nel capitolo precedente è arrivato il momento di presentare **C#** vero linguaggio principe di .Net. Esso è stato concepito e creato come strumento per programmare specificatamente utilizzando il framework di Microsoft giunto come detto alla sua versione 4.0. E' abbastanza naturale capire il perchè di questa scelta o meglio di questo "parto". Creare un nuovo linguaggio di programmazione è una una mossa comunque pesante ma in questo caso spontanea in quanto .Net è un qualche cosa che si sposta in maniera vistosa dalle precedenti tecnologie di sviluppo; in questo senso è senza dubbio assai produttivo ideare un nuovo strumento che si adatta completamente alla nuova tecnologia. L'adattamento faticoso e laborioso di Visual Basic e C++ al mondo .Net è la riprova che una novità radicale in questo senso è stata una buona idea. Si è trattato una evoluzione cominciata nel 2001 ed in questo periodo il linguaggio è cresciuto fino a diventare uno strumento maturo ed utilizzabile per i più svariati compiti. Esso trae la sua ispirazione a livello sintattico da C/C++ e da Java oltre che da Object Pascal, sia pure per quest'ultimo in maniera meno evidente; tuttavia il la mente principale che si cela dietro C# è **Anders Hejlsberg** che, prima di lavorare in Microsoft, era a capo del progetto Delphi in Borland (dopo alcune vicissitudini Delphi ora è un sempre valido prodotto di Embarcadero Software). Logicamente qualche cosa se la è portata dietro, vista anche la bontà di quel prodotto che il tecnico danese seguì quasi fino alla uscita della versione 3.0. L'evoluzione pubblica di C# ha preso il via come detto nell'anno 2001, anche se il progetto originale pare datasse 1998, ed è continuata ininterrotta fino ai giorni nostri ricevendo nel frattempo il riconoscimento come standard **ECMA** e **ISO**.

C# nasce quindi con un background notevole e la sua sintassi non rinnega le origini; va però tenuto presente che il team di sviluppo ha lavorato in modo da renderlo il più semplice possibile ma soprattutto cercando di modellare la sua espressività in maniera che non si distaccasse troppo da quanto largamente utilizzato dai programmatori Java e C/C++ i quali avranno una curva di apprendimento molto dolce, almeno a livello per così dire grammaticale. Un modo per attrarre programmatori anche da quei linguaggi. La semplicità era comunque sicuramente uno dei target dei progettisti.

Ma quali sono le caratteristiche di C#? Non dimentichiamoci che esso gode di tutte le features di .Net alcune delle quali abbiamo già sfiorate nel capitolo di introduzione al framework. Tuttavia in questa sede le reincontriamo per completezza descrittiva:

- Abbraccia vari paradigmi di programmazione, come la maggior parte dei nuovi linguaggi. Certamente l'aspetto object oriented è quello più utilizzato e conosciuto ma si può usare C# anche per programmare usando l'approccio funzionale (così non era nella versione 1.0), event-driven, generico e anche, perchè no, imperativo. Tutto questo grazie appunto all'evoluzione avutasi nel tempo che ha notevolmente arricchito il linguaggio di costrutti ed espressività.
- E' a tipizzazione statica. Ovvero una variabile nasce e muore rimanendo dello stesso tipo e la verifica avviene a compile-time. Scrivere ad esempio il comando:
  int x
  equivale ad attribuire ad i il tipo intero. Non sarà possibile assegnare ad x successivamenteun altro tipo.
  La scelta della tipizzazione statica o di quella dinamica (in cui il type-check avviene a run-time ed è utilizzato da molti linguaggi anche ad ampia diffusione ad esempio Python, Ruby o PHP) è una questione in un certo qual modo filosofica tuttora senza una risposta certa su quale delle due sia la migliore. C# come detto abbraccia la prima sicuramente meno duttile però, forse, più sicura. Come anticipato nel capitolo precedente tuttavia con la versione 4.0 del linguaggio è stata introdotto la possibilità di una tipizzazione dinamica che, non rinnegando le origini, amplia, come vedremo, la fruibilità del linguaggio.
- La tipizzazione è anche "forte". Questo in pratica pone dei vincoli sull'uso delle variabili di modo che siano consentite solo le operazioni legali su ciascuna variabile. La definizione non è facilmente esprimibile ed è stata oggetto di varie discussioni.
- Concludendo il discorso relativo alla tipizzazione sottolineiamo che essa in C# è "sicura". In questo caso si può in breve affermare che non sono permesse conversioni implicite che possono portare a risultati errati o meglio a perdita di informazione. L'esempio tipico è il seguente:
  ```
  string x = "44";
  int i = 0;
  int z = i + x;
  ```

L'ultima riga non è permessa, ovvero non può esservi conversione implicita della stringa "44" nel suo equivalente numerico. Ovviamente tutti i discorsi relativi ai tipi sono collegati alla necessità di conformarsi alle specifiche .Net CTS e CLS introdotte nel capitolo precedente.

- Dispone, in breve, pienamente di tutte le features proprie di .Net quindi un efficiente meccanismo di garbage collection, gestione delle eccezioni avanzata, ecc... segnalo che dalla versione 4.0 in avanti è possibile usare anche il design by contract e sfruttare le CPU multi core grazie al parallelismo (il progetto si chiama Parallel FX), tutti argomenti già accennati nel capitolo precedente
- Permette solo l'ereditarietà singola da classi ma è possibile ereditare da quante interfaccie si vuole. Se siete dei fan dell'ereditarietà multipla (come me, forse a torto) con C# rimarrete delusi. Ovviamente questo non preclude nulla in termini di potenzialità ed espressività del linguaggio e d'altronde è innegabile che l'ereditarietà singola sia l'approccio a oggi di maggior successo.
- Non viene fatto uso dei puntatori, croce (soprattutto) e delizia dei programmatori C/C++, se non in blocchi di codice marcati come unsafe.
- Può facilmente cooperare con altri linguaggi .Net oriented, vale la pena ripeterlo perchè è un concetto pilastro per .Net.
- Offre un valido supporto per l'internazionalizzazione.
- Supporta nativamente la generazione di documentazione XML e anche scrittura di servizi Web XML Based.
- Tramite il progetto Mono gira anche su piattaforme Linux e Mac.
- E' stato pensato affichè fosse in grado di generare applicazioni adatte anche per i sistemi embedded.
- Dispone di un avanzato strumento di sviluppo come Visual Studio disponibile anche gratuitamente in una serie di versioni ridotte chiamate Express che risultano comunque strumenti completi e assai validi. Questa non è una carrateristica del linguaggio in senso stretto ma insomma chi usa C# sa che ha un vantaggio in questo potente strumento.

Tutto questo naturalmente detto per sommi capi e restando un po' sulle linee generali; soprattutto preferisco evitare confronti diretti con altri linguaggi, confronti di cui il web è pieno zeppo anche da fonti molto più qualificate di me. I dettagli "veri" del linguaggio li vedremo strada facendo. In realtà infatti le caratteristiche di C# sono molteplici ed in particolare è profondo l'intreccio con le tecnologie che Microsoft ha sviluppato negli ultimi anni. Parole come Linq o Wpf, piuttosto che SilverLight saranno oggetto di dicussione (molto più avanti...), capiremo meglio perchè C# è nato per .Net e come da questo potentissimo motore tragga risorse e potenzialità enormi. C# è nato con .Net ed è il linguaggio che in assoluto meglio lavora in congiunzione con questo ambiente. Tra l'altro pare che la gran parte della BCL sia stata scritta proprio in C# e scusate se è poco. Ogni evoluzione della piattaforma sottostante aumenta le possibilità del linguaggio. Altro punto da non sottovalutare è la grande disponibilità di risorse presenti sul Web in termini di codice, tutorials, materiale vario e communities con il sito di MSDN che è un reference enorme ed insositutibile. Va anche aggiunto che, negli anni, Microsoft ha investito molto su C# cercando di arricchirlo ad ogni release di quelle features che gli sviluppatori richiedevano. Insomma c'è un supporto di alto livello alle spalle di tutto. Ed esistono dei progetti sperimentali che hanno dato origine a dialetti tramite i quali sono state sperimentate nuove features. Cω, Spec#, Polifonic C# sono alcuni di questi dialetti in reqaltà veri e propri linguaggi completi la cui nascita ed analisi ha permesso di riversare ulteriore conoscenza in C#.

Il target di C# in definitiva è lo sviluppo di applicazioni di qualsiasi tipo nell'ambiente Windows, che si parli di database, n-tier, GUI, si tratta comunque di termini e sigle che non sfuggono al potenziale applicativo di questo linguaggio; ma anche in se si guarda al Web esso è fortemente utilizzato nell'ambito di Asp.Net. Certamente non è il linguaggio "definitivo", che non esiste ancora e fose non esisterà mai, ma sicuramente è un compagno di viaggio che può dare molte soddisfazioni.

Mi permetto di aggiungere a questa presentazione una mia piccola esperienza personale: ho chiesto a vari sviluppatori che cosa ne pensassero sinceramente di C# e, tolto qualche raro interlocutore palesemente schierato da una parte o dall'altra, ovvero un po' troppo pro o un po' troppo contro, ho avuto la sensazione che, mediamente, sia considerato un linguaggio molto buono, anche da parte di altri "language writer".

Ci sono delle controindicazioni? Beh, si, d'altronde come abbiamo detto il linguaggio perfetto non c'è. Tralasciamo l'appartenenza alla galassia Microsoft che ad alcuni non piace "per principio". Le critiche maggiori derivano proprio da questa paternità, e il fatto che sia stato comunque standardizzato e, di fatto, aperto, non lo ha salvato da lugubri sospetti che vedono tra i principali sostenitori Stallman (che avrebbe detto in sostanza: "dipendere da C# è pericoloso e quindi è bene non usarlo" non tanto temendo il linguaggio quanto l'uso che Microsoft potrebbe farne) e la Free Software Foundation. Qui però siamo sul politico piuttosto che sul tecnico e certamente non mi addentro nel discorso.

Ritornando ad aspetti eminentemente pratici va evidenziato come ultimamente si stanno diffondendo linguaggi come Python e Ruby interpretati, più leggeri, per alcuni versi più semplici. Questo però è un aspetto di cui diffidare: non esiste un linguaggio che renda semplice tutte le cose che non lo sono. Alcuni sono più predisposti verso alcune problematiche, ma insomma, programmare non è facile specialmente se lo si vuol fare davvero bene. La sintassi C based di C# può non piacere, così come in alcuni casi le sue performance sono controverse, d'altronde in fase di design Hejlsberg e i suoi colleghi non sono andati alla ricerca delle prestazioni da record. Per sistemi real-time forse non è il linguaggio ideale, come non lo è nessuno di quelli general purpose. Però sinceramente, anche da questo punto di vista, non ho mai riscontrato criticità negative nell'ampio ventaglio di tipologie di problemi che ho dovuto affrontare. Insomma C# è veloce il giusto, secondo me. Provare è comunque sempre il modo migliore per scegliere e valutare.

Per chiudere una piccola curiosità: il nome C# si vuole contenga un richiamo, abbastanza ovvio, alla "alma mater" ovvero il linguaggio C unito ad un simbolo musicale, appunto #... secondo alcuni proprio l'ultimo simbolo, vista la conformazione, che lo fa assomigliare alla composizione di 4 simboli "+", farebbe si che C# sia in realtà C++++ (!). Altra piccola curiosità: pare che prima di C# uno dei nomi che si pensava di attribuire al nuovo linguaggio era "Cool", poi vi furono dei problemi di registrazione di tale nome e non se ne fece nulla. Per noi italiani forse è meglio: non è carino dire "programmo in Cool".....