.NET Framework

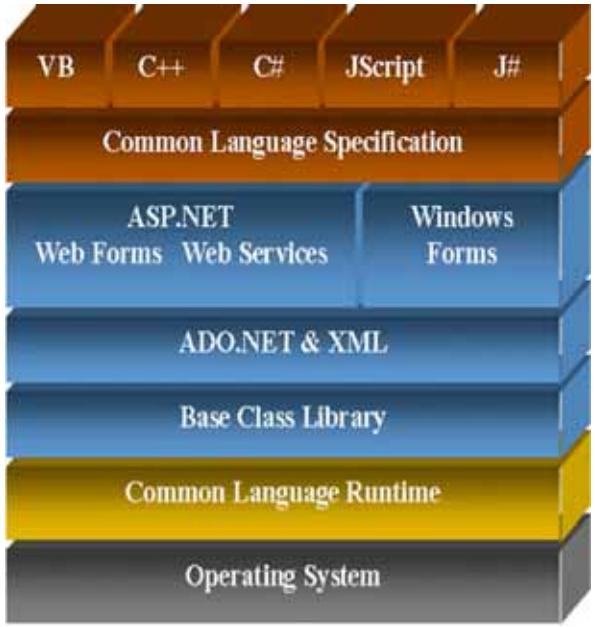
Ing. Enrico Oliva
Phd student, DEIS
eoliva@deis.unibo.it

Outline

- Framework .NET
  - Accenno sulla struttura e obiettivi
- Common Language Runtime
  - Componenti principali, accenni al runtime
- Common Type System
- Esempio Hello word in C#
- Assembly unità di condivisione del codice
- .NET Remoting, ADO.NET, ASP.NET

Architettura .NET Framework

- .NET è un framework di sviluppo per servizi e API di Windows. Integra tecnologie come: COM+, ASP, XML, SOAP, WDSL, UDDI.
- I componenti del framework .NET:
  - Common Language Runtime (CLR)
  - Librerie delle classi base
  - Libreria delle classi estese
    - ASP.NET per i servizi web, ADO.NET e XML per l'accesso e manipolazione dei dati
  - Common Language Specification
    - definisce i requisiti per i linguaggi.NET compatibili
  - Linguaggi e strumenti di sviluppo;
- Obiettivo del framework:
  - Infrastruttura a componenti;
  - Integrazione di linguaggi (VB.NET, C#,...);
  - Interoperabilità Internet (SOAP);
  - Semplicità di sviluppo e rilascio;
  - Affidabilità e Sicurezza;



Infrastruttura a componenti

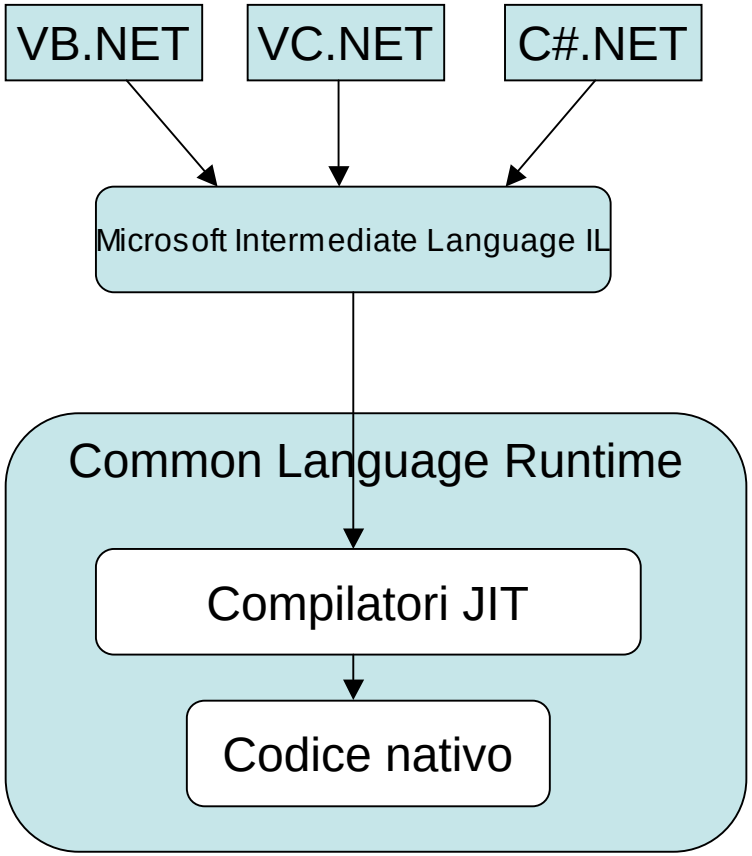
- La tecnologia COM permette l'integrazione di componenti binari. Lo sviluppo su tale tecnologie è particolarmente pesante… (scrittura di codice extra per la gestione)
- In .NET tutte le classi sono pronte al riuso a livello binario "*assembly*".
- .NET elimina l'uso del registro per i componenti e la necessità di codice extra per la loro gestione.

Obiettivi di progettazione CLR

- Semplificare lo sviluppo
  - Definire standard per il riuso del codice
  - Fornire servizi come gestione della memoria e garbage collection
- Semplificare deployment delle applicazioni
  - I componenti usano i metadati anzichè la registrazione
  - Supportare più versioni
  - Deployement da riga di comando XCOPY e disinstallazione DEL
- Supportare i linguaggi di sviluppo
  - Fornendo classi di base per gli strumenti e i linguaggi degli sviluppatori
- Supportare più linguaggi di programmazione
  - Definire i CTS utilizzati da tutti i linguaggi .NET

Common Language Runtime (CLR)

- Il CLR è l'equivalente .NET della Java Virtual Machine (JVM).
  - Attiva gli oggetti, li sottopone a controlli di sicurezza, li dispone in memoria.
- Gli eseguibili .NET contengono codice e metadati.
  - I metadati contengono la definizione di tipo, informazione sulla versione e riferimenti ad assembly esterni.
- Il class loader componente del CLR utilizza i metadati per il caricamento delle classi .NET
- I compilatori Just in Time JIT usano i metadati per compilare il codice Microsoft Intermediate Language (IL)
- CLR utilizza inoltre
  - CTS Common Type System che definisce i tipi a valore e riferimento supportati dal framework
  - CLS Common Languge Specification definisce le regole che un linguaggio deve avere per essere gestito dal framework



Common Language Runtime (CLR)

- Tutti i linguaggi .NET attraverso il CLR accedono al medesimo sistema di tipi e alle stesse classi base
  - È ottenuta cosi una convergenza tra linguaggi e modelli di programmazione
- Il codice eseguito dal CLR viene detto ***codice gestito***
- è memorizzato come codice+metadati in un formato standard Windows PE (portable executable) che consente di
  - Leggere i metadati
  - Fornire lo stack del codice
  - Gestire le eccezioni
  - Recuperare informazioni di sicurezza
- Esistono “versioni del CLR” anche sotto Linux e Mac Os X (non sono supportate da Microsoft)
  - http://www.mono-project.com/Main_Page
  - consentono di eseguire il codice in formato PE anche in macchine non Widows rendendo .NET un framework multipiattaforma

Common Type System (CTS)

- I linguaggi possono interoperare solo se sono basati sul medesimo sistema di tipi
- Nella programmazione ad oggetti un tipo contiene un valore e supporta un'interfaccia per la descrizione del tipo
- In .NET tutti i tipi sono orientati agli oggetti (ogni elemento è un oggetto) e i tipi a valore possono essere convertiti in tipi a riferimento mediante il boxing
- CTS fornisce due tipi principali
  - Tipi a valore derivano dal namespace System.Object e sono utili per rappresentare tipi di dati semplici e quelli definiti dall'utente
  - Tipi a riferimento derivano dal namespace System.Object e sono: tipi di oggetti, tipi di interfacce e tipi di puntatori

Tipi a valore

- I tipi a valore non possono essere null e devono sempre contenere dei dati
  - Il passaggio di un tipo a valore in una funzione avviene per copia
- I tipi a valore sono: primitivi, strutture e enumerazioni

```
Esempio C#:
int i;    //primitivo
Struct Point{int x,y} //strutturato
Enum State{Off,On} //enumerazione
```

- E’ possibile creare un tipo valore anche derivandolo da una classe System.ValueType
  - Quello che fa .NET per i tipi primitivi come int:
    - Int è un alias di System.Int32 che deriva da Sysem.ValueType

Tipi a riferimento

- I tipi a riferimento contengono il riferimento ad oggetti basati sullo heap, che possono essere null
  - Il passaggio avviene per riferimento cioè viene passato un indirizzo o un puntatore ad un oggetto
- Questi tipi vengono gestiti dal CLR e sottoposti a garbage-collection

Sicurezza di tipi

- Il CTS promuove la sicurezza di tipi che migliora la stabilità del codice
  - Le definizioni di tipi sono completamente note e non possono essere compromesse
  - Il codice ASP non supporta variabili strongly-typed. Nascono problemi di controllo del corretto utilizzo dei riferimenti agli oggetti
    - Chiamate a metodi non supportati
    - Assegnamenti non riusciti per incompatibilità di tipo
- Il CTS garantisce che i riferimenti agli oggetti siano strongly-typed
  - In .NET l'oggetto a cui viene fatto riferimento rileva le informazioni del proprio tipo e il compilatore runtime le attiva
    - Si possono solo chiamare i metodi appropriati al tipo: il compilatore runtime non compilerà il codice che tenta di fare riferimento a un metodo inesistente

Common Language Specification (CLS)

- E’ un supporto all’integrazione dei linguaggi
- Il CLS è una specifica che definisce i requisiti minimi per cui un linguaggio possa essere considerato .NET
- Alcune convenzioni
  - Gli identificatori pubblici devono distinguere maiuscole da minuscole
  - I linguaggi devono essere in grado di risolvere identificatori equivalenti alle loro parole chiave
  - …
- Altre informazioni sono contenute nella documentazione del kit SDK di Framework .NET

Metadati

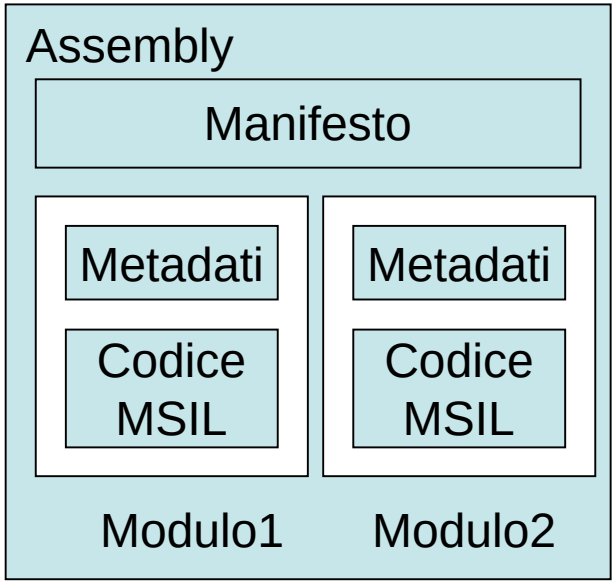
- I metadati sono una descrizione strutturata del codice (di un assembly) e contengono le informazioni sui tipi utilizzati nel modulo
  - Il nome
  - La visibilità
  - Da chi eredita
  - Le interfacce che implementa
  - I metodi che implementa
  - Le proprietà che espone
  - Gli eventi che fornisce
- I metadati sono memorizzati come parte del codice sorgente del componente .NET
  - Quindi i componenti sono a descrizione automatica
    - non necessitano più di voci nel registro di sistema
    - deployment più semplice in .NET
- Sono utilizzati dal CLR per
  - Gestire la memoria / Debugging / Esplorazione degli oggetti / Traduzione dell'IL in codice nativo

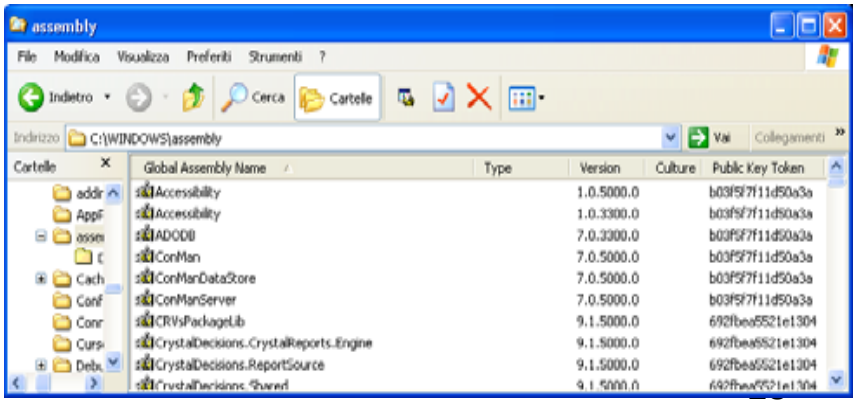
Utilizzo dei metadati

- I metadati sono indipendenti dal linguaggio e tutti i linguaggi che gli accedono e possono interpretarli allo stesso modo
- Visual Studio .NET consente di unire file di codice sorgente di linguaggi diversi all'interno dello stesso progetto.
  - Operazione possibile grazie alle informazioni contenute nei metadati dei componenti
- Il compilatore genera i metadati di un componente dal codice sorgente che vengono memorizzati nel codice compilato in un formato PE (Portable Executable)
- Un tools per la visualizzazione dei metadati è ildasm.exe

Assembly

- Un *assembly* è l'unità elementare di rilascio del codice e di gestione delle versioni.
- E' composta da:
  - *Un manifesto*
  - *Un insieme di uno o più moduli (dll o exe)*
  - *Un insieme opzionale di risorse*
- Nell'esempio visto in precedenza si è generato un *single module assembly* dove tutto viene conglobato in un unico file EXE o DLL;
  - La parola chiave using serve per importare un assembly in C#;
- L'utility di gestione degli assembly condivisi è gacutil.exe
- Gli assembly correttamente installati si trovano nella cartella: c:\windows\assembly





Manifest

- Il manifest rappresenta un metadato per gli assembly contiene le informazioni riguardanti la composizione dell'assembly
  - Il nome dell'assembly
  - La versione dell'assembly formato da
    - <major version>.<minor version>.<build number>.<revision> es. 1.2.1397.0
  - I linguaggi che l'assembly supporta
  - L'elenco dei file contenuti nell'assembly
  - Le dipendenze con altri assembly e i loro numeri di versione

L’esecuzione del codice

- Gli assemblies sono caricati in memoria dal CLR solo all’occorenza
  - prima viene determinata la versione
  - poi viene cercato l’assembly nella Global Assembly Cache (GAC)
  - o nel percorso locale indicato dal codice base
- Compilazione MSIL
  - Il compilatore produce un codice gestito MSIL che non è eseguibile direttamente dal processore
  - Il codice MSIL deve essere compilato nuovamente dal CLR per la generazione di codice nativo. Ci sono due possibilità
    - Compilazione del metodo a tempo di esecuzione Just In Time (JIT)
    - Compilazione di tutto l’assembly prima dell’esecuzione
- Sicurezza negli assemblies
  - Il CLR permette di limitare le funzionalità del codice eseguito

Le classi del framewok .NET

- Le classi le interfacce e i tipi sono organizzate in strutture gerarchiche denominate *namespace* (vedi *package* in Java)
- Il framework fornisce classi estendibili e manipolabili mediante ereditarietà, override di metodi e polimorfismo
- Tutte le classi del framework ereditano da *System.Object*
- *System.Exception* contiene le classi per la gestione delle eccezioni
- Le librerie fornite dal framework .NET
  - ADO.NET fornisce il namespace System.Data per l'accesso ai dati
  - ASP.NET fornisce il namespace System.Web per il codice ASP.NET e per i comandi e le classi che supportano i servizi web
  - XML.NET fornisce il namespace System.Xml per le classi XML. Contiene altri namespace come XPath e XSLT.
  - Windows Forms forniscono i namespace System.Windows.Forms per le classi che supportano i comandi e le funzionalità che gestiscono le finestre

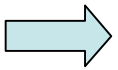
Hello word in C#

Le classi del framework sono organizzate in **namespace** un contenitore per raggruppare in modo logico classi correlate

```
// Allow easy reference to the System namespace classes.
using System;                (in Java imports)
namespace Project1           (in Java package)
{
        // This class exists only to house the entry point.
        class MainApp
        {
        // The static method, Main, is the application's entry point.
                public static void Main()
                {
                        // Write text to the console.
                        Console.WriteLine("Hello World using C#!");
                }
```

Eseguo la compilazione:

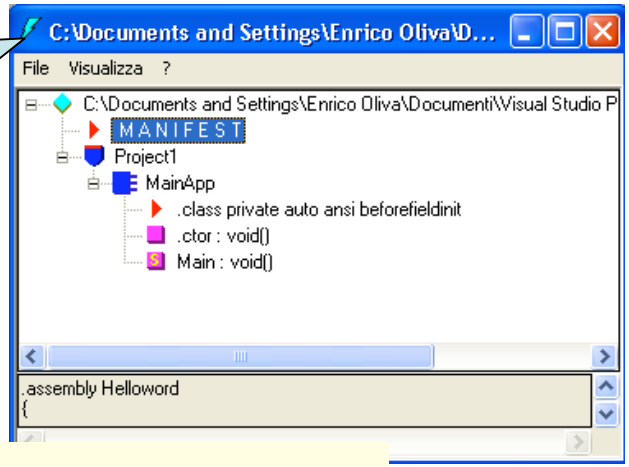
```
>csc Helloword.cs
Compilatore Microsoft (R) Visual C# .NET versione 7.00.9951
per Microsoft (R) .NET Framework versione 1.0.3705
Copyright (C) Microsoft Corporation 2001. Tutti i diritti riservati.
```



Helloword.exe
File eseguibile dal CLR

Hello word *ildasm.exe*

Ildasm.exe è il disassembler che permette di visualizzare a partire da un file .NET valido i metadati e il codice Intermediate Language IL generato.



Manifest

•mscorlib è un assembly esterno
•publickeytoken è l'inforamzione su chi l'ha pubblicato

```
.assembly extern mscorlib
{
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )                        // .z\V.4..
  .ver 1:0:3300:0
}
.assembly Helloword
{
  // --- The following custom attribute is added automatically, do not uncomment -------
  //  .custom instance void [mscorlib]System.Diagnostics.DebuggableAttribute::.ctor(bool,
  //                                                                                 bool) = ( 01 00 00 01 00 00 )
  .hash algorithm 0x00008004
  .ver 0:0:0:0
}
.module Helloword.exe
// MVID: {45F67D1A-5837-4C9B-9FB6-88C2BD8B987C}
.imagebase 0x00400000
.subsystem 0x00000003
.file alignment 512
.corflags 0x00000001
// Image base: 0x03090000
```

C# o VB.NET

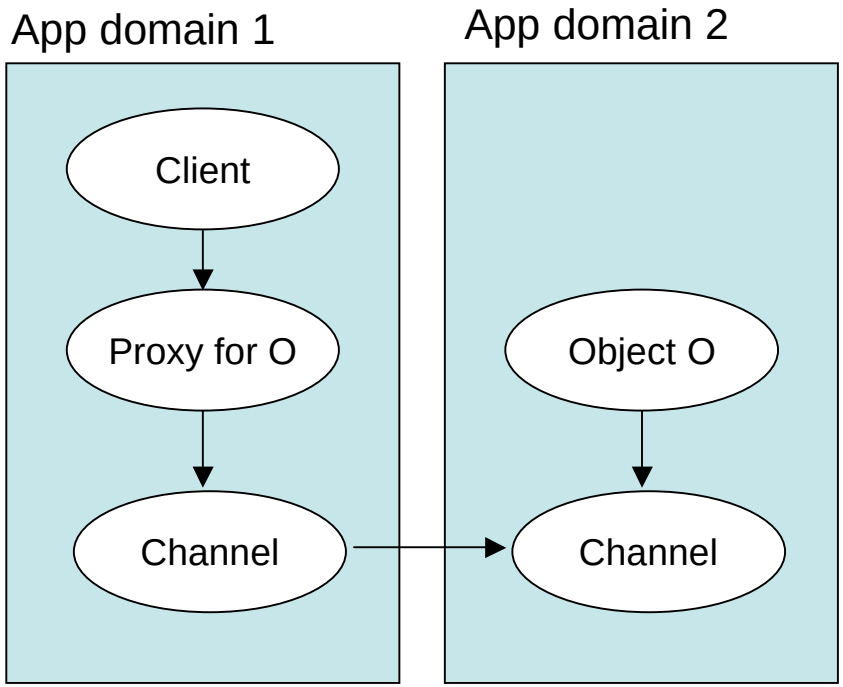
- I due linguaggi hanno le stesse potenzialità e sono molto simili
- Esistono tools di conversione tra codice C# e VB.NET e viceversa
- La scelta tra i due è legata esclusivamente alla familiarità dell'uso di una sintassi like C o like Visual Basic
- Non esistono più ragioni di efficienza nella scelta di uno dei due linguaggi

.NET Remoting

- Fornisce una struttura di gestione della comunicazione interprocesso in .NET
  - Instaurare una comunicazione tra oggetti distribuiti
    - chiamata remota a procedura RPC operazione sincrona
    - chiamate asincrone attraverso l'invio di messaggi
  - Chiamata ad un servizio di sistema
  - Modelli di marshaling e threading per gestione della comunicazione
    - Il marshaling è il processo di impacchettamento dei valori da trasferire
- Domini di applicazione sono unità di elaborazione che isolano le applicazioni
- .NET Remoting fornisce i servizi necessari che permettono oggetti di interagire fra domini di applicazioni differenti
- Consente di determinare il protocollo di trasporto, la durata di attivazione e le modalità di creazione degli oggetti
- Questo tipo di comunicazione è utilizzato quando entrambi gli end point usano .NET Framework (applicazioni basate su CLR)

Processo di Remoting

- Un client crea una istanza di alcune classi e chiama un metodo di un oggetto in un altro dominio di applicazione
- La chiamata è ascoltata da un proxy object caricato nel dominio del client
- Il proxy è creato dal CLR automaticamente attraverso la reflection e i metadata dell'oggetto remoto e lo rappresenta all'interno del client
- Il proxy passerà la chiamata ad un oggetto channel che realizzerà il collegamento (es.: TCP)



Canale

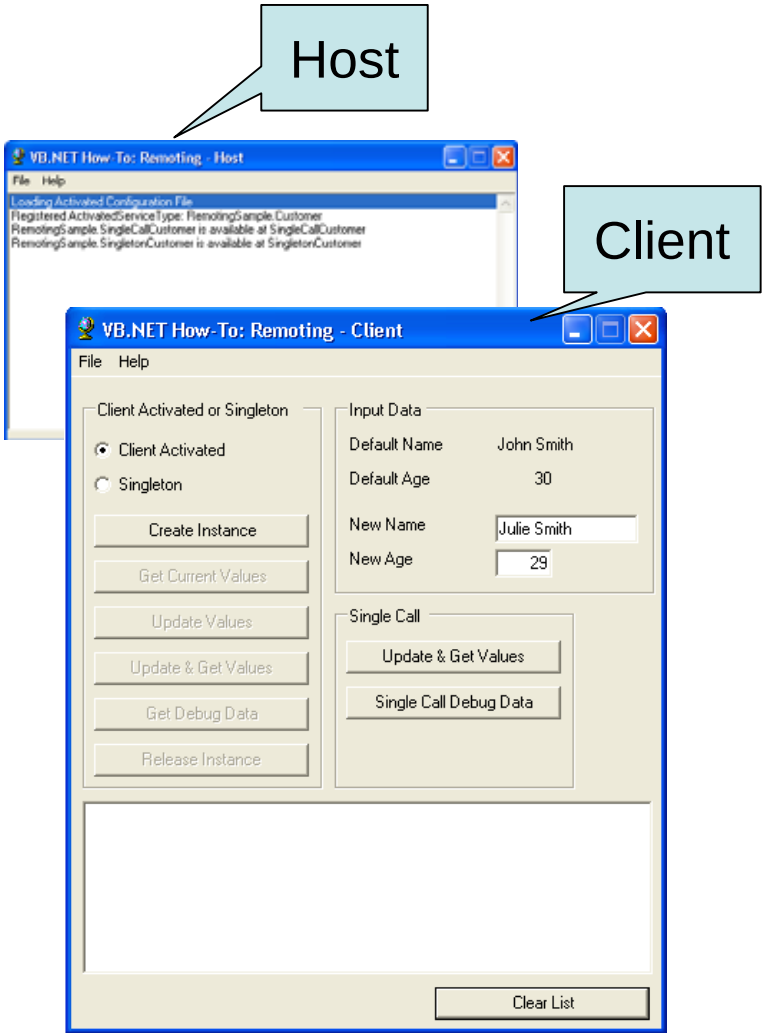
- E’ un meccanismo di trasporto di messaggi (es. messaggi tra due oggetti remoti o tra client e server)
- Nel Framework esistono due implemetazioni di canale
  - TcpChannel: serializza e deserializza le chiamate in un formato binario e le trasmette direttamente come pacchetti TCP
    - Fornisce un supporto per la sicurezza
    - Lo si usa in una rete locale intranet senza firewalls si ha una maggiore efficienza
  - HttpChannel serializza e deserializza le chiamate in formato formato SOAP. Le richieste e risposte SOAP avvengono all’interno di HTTP
    - Lo si usa quando le richieste sono pacchetti che viaggiano sulla rete internet, si ha una minore efficienza
- Possono essere create delle versioni custom del canale ad esempio per particolari esigenze di sicurezza

Attivazione di un oggetto

- Dal tipo di attivazione di un oggetto dipende il momento in cui un oggetto viene creato e la sua durata
  - Lato client
    - La durata dell'oggetto è controllata dal client per default l'oggetto sul server verà eliminato al termine della chiamata dell'oggetto client
      - La creazione dell'oggetto avviene con la **new** è consentito il passaggio di parametri
      - Consente che ad ogni nuovo client venga attivato una istanza nuova dell'oggetto richiesto
  - Lato server
    - La durata dell'oggetto è controllata dal server
      - Singlecall viene creato un oggetto per ogni chiamata e distrutto a fine del metodo
      - Singletons la durata dell'oggetto è configurata dal server, viene creata una sola istanza sulla macchina server
- L'istanza dell'oggetto risiede lato server mentre al client solitamente viene passato un riferimento
  - Il client comunica con l'oggetto di attivazione remota attraverso un proxy lato client generato automaticamente dal framework .NET

Esempio di Remoting

- Server Remoting Customer
  - E' una libreria che risiede lato server e espone dei tipi noti al client che li renderà attivi
  - Tutti i tipi appartengono allo stesso Namespace RemotingSample
- Host
  - Carica il file di configurazione Host.exe.config in formato XML e lo passa a RemotingConfiguration.Configure
  - Espone i tipi disponibili via remoting
    - Sono i tipi contenuti nel namespace RemotingSample
- Client
  - Supporta la connessione di tre diversi tipi di oggetto contenuti nel namespace RemotingSample
  - Crea gli oggetti remoti e ne invoca i metodi



ADO.NET

- E’ una class library che contiene i tipi per la gestione dei dati che risiedono tutti nel namespace System.Data
  - La gestione dei dati è ottenuta attraverso
    - SQL provider viene utilizzato per accedere a Microsoft SQL server DBMS 7.0
    - OLE DB provider per le tradizionali applicazioni ADO dove OLE DB fornisce l’acesso ai dati. Il OLE DB.NET realizza un wrapper delle vecchie classi OLE DB
    - ODBC provider è disponibile ma non ufficialmente nella prima versione di .NET
- Le principali funzioni di gestione dei dati implementate dai provider sono
  - Connessione: permette di stabilire e una connessione
  - Comandi: permette di memorizzare e eseguire comandi come SQL query
  - DataReader: permette un accesso sequenziale e di sola lettura ai dati
  - DataAdapter: permette di costruire partendo da DataReader delle istanze della classe DataSet e popolarle
- Client accede ai dati attraverso
  - DataReader: che realizza un veloce e diretto accesso ai dati con l’utilizzo di poca memoria (legge una riga alla volta)
  - DataSet: che memorizza i dati in cache e ne fornisce un più flessibile accesso. Il DataSet è composto da DataTable e DataRelation

ADO.NET & XML

- Unire una struttura gerarchica (XML) con l'accesso ai dati relazionale è una scelta vincente!
- Entrambe le tecnologie hanno API per lo streaming e per la navigazione:
  - DataReader vs. XmlReader
  - DataSet vs. XmlDocument
- Con ADO.NET si integrano i dati del DataSet con un XML document
  - Dal DataSet è possibile leggere e scrivere dati in forma XML con le classi ReadXml e WriteXml
    - E' possibile anche leggere lo schema XSD del documento XML
  - Serializzando i dati in XML è possibile spedirli a un qualsiasi broswer

Esempio XML to DataSet

```
<employees>
    <employee>
        <name>Andrea</name>
        <age>45</age>
    </employee>
    <employee>
        <name>Enrico</name>
    </employee>
</employees>
```

Name	Age
Andrea	45
Enrico	null

Si inferisce lo schema dalla lettura del XML. Diventa una tabella:

- un elemento con attributi
- e un elemento con figli

ASP.NET

- E’ una class library che supporta la creazione di Web Application e Web Service
- Ad una pagina ASP.NET si può accedere direttamente da un browser ed è memorizzata in un file di testo di estensione .aspx
  - La pagina aspx può contenere testo, HTML, e codice eseguibile
    - Il codice è scritto in blocchi delimitati dai tags
      - <script>… </script> o <% … %>
- La pagina aspx, rispetto a quella asp, viene al momento della chiamata trasformata in una classe e compilata in un assembly

Esempio ASP.NET

```
<html>
    <script runat="server"
    language="vb">
    Sub ShowNumber()
            Dim I As Integer
            For I = 0 to 5
                    Response.Write(I)
            Next
    End Sub
    </script>
La data e l'ora sono: <% =Now() %>
<hr>
I numeri  <% =ShowNumber() %>
</html>
```

Pagina .aspx

```
Class Example_aspx
    Inherits System.Web.UI.Page
Sub ShowNumber()
    Dim I As Integer
    For I = 0 to 5
            Response.Write(I)
    Next
End Sub

Sub Render()
    Response.Write("La data e l'ora
    sono:")
    Response.Write (Now())
    Response.Write ("I numeri")
    Response.Write (ShowNumber())
End Sub
End Class
```

Pagina .aspx convertita in classe

ASP.NET

- ASP.NET usa un modello di esecuzione ad eventi
- La pagina viene richiesta
  - ↳ viene creato l'assembly
    - ↳ viene generata una istanza della pagina
      - ↳ la pagina oggetto riceve degli eventi
- Esempio di eventi
  - Page_Load evento generato immediatamente dopo la creazione dell'oggetto pagina
  - Page _Unload evento generato immediatamente prima della distruzione della pagina
  - Eventi dovuti ai Web Controls (button, check box, list box, radio button…) come click_Button

Riferimenti

- http://msdn.microsoft.com/library/ita/default.asp?url=/library/ITA/cpguide/html/cpconsourcecodelanguage.asp
- D.Chappell. *Undestanding .NET.* Addison-Wesley, 2002
- *Professional .NET Framework*. Wrox 2002