

Laborator 1

Introducere in .NET

.NET este un cadru (*Framework*) de dezvoltare software unitară care permite realizarea, distribuirea și rularea aplicațiilor desktop Windows și aplicațiilor WEB.

Tehnologia .NET pune laolaltă mai multe tehnologii (ASP, XML, OOP, SOAP, WDSL, UDDI) și limbaje de programare (VB, C++, C#, J#) asigurând, totodată, atât portabilitatea codului compilat între diferite calculatoare cu sistem Windows, cât și reutilizarea codului în programe, indiferent de limbajul de programare utilizat.

.NET Framework este o componentă livrată împreună cu sistemul de operare Windows. De fapt, .NET 2.0 vine cu Windows Server 2003, se poate instala pe versiunile anterioare, până la Windows 98 inclusiv; .NET 3.0 vine instalat pe Windows Vista și poate fi instalat pe versiunile Windows XP cu SP2 și Windows Server 2003 cu minimum SP1.

Pentru a dezvolta aplicații pe platforma .NET este bine să avem 3 componente esențiale:

- un set de limbaje (C#, Visual Basic .NET, J#, Managed C++, Smalltalk, Perl, Fortran, Cobol, Lisp, Pascal etc),
- un set de medii de dezvoltare (Visual Studio .NET, Visio),
- o bibliotecă de clase pentru crearea serviciilor Web, aplicațiilor Web și aplicațiilor desktop Windows.

Când dezvoltăm aplicații .NET, putem utiliza:

- Servere specializate - un set de servere Enterprise .NET (din familia SQL Server 2000, Exchange 2000 etc), care pun la dispoziție funcții de stocare a bazelor de date, email, aplicații B2B (*Bussiness to Bussiness* – comerț electronic între partenerii unei afaceri).
- Servicii Web (în special comerciale), utile în aplicații care necesită identificarea utilizatorilor (de exemplu, .NET Passport - un mod de autentificare folosind un singur nume și o parolă pentru toate site-urile vizitate)
- Servicii incluse pentru dispozitive non-PC (Pocket PC Phone Edition, Smartphone, Tablet PC, Smart Display, XBox, set-top boxes, etc.)

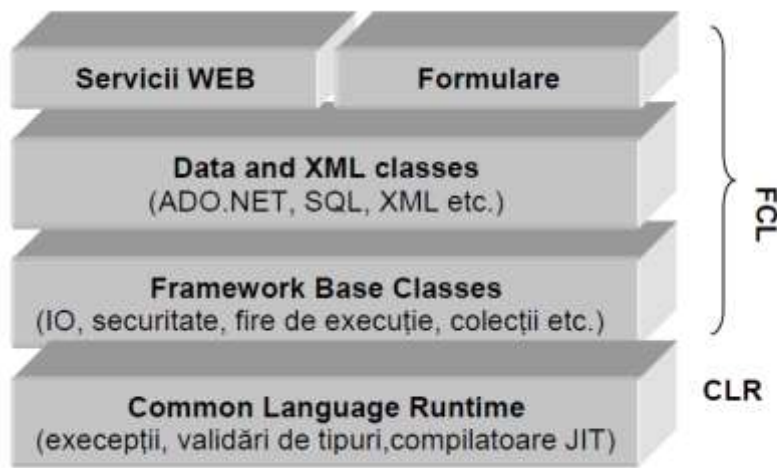
.NET Framework

Componenta .NET Framework stă la baza tehnologiei .NET, este ultima interfață între aplicațiile .NET și sistemul de operare și actualmente conține:

- Limbajele C#, VB.NET, C++ și J#. Pentru a fi integrate în platforma .NET, toate aceste limbaje respectă niște specificații OOP numite *Common Type System* (CTS). Ele au ca elemente de bază: clase, interfețe, delegări, tipuri valoare și referință, iar ca mecanisme: moștenire, polimorfism și tratarea excepțiilor.
- Platforma comună de executare a programelor numită *Common Language Runtime* (CLR), utilizată de toate cele 4 limbaje. CTS face parte din CLR.
- Ansamblul de biblioteci necesare în realizarea aplicațiilor desktop sau Web, numit *Framework Class Library* (FCL).

Arhitectura .NET Framework

Componenta .NET Framework este formată din compilatoare, biblioteci și alte executabile utile în rularea aplicațiilor .NET. Fișierele corespunzătoare se află, în general, în directorul C:\WINDOWS\Microsoft.NET\Framework\V2.0.... (corespunzător versiunii instalate)



Compilarea programelor

Un program scris într-unul dintre limbajele .NET conform *Common Language Specification* (CLS) este compilat în *Microsoft Intermediate Language* (MSIL sau IL). Codul astfel obținut are extensia ".exe", dar nu este direct executabil, ci respectă formatul unic MSIL.

CLR include o mașină virtuală asemănătoare cu o mașină Java, ce execută instrucțiunile IL rezultate în urma compilării. Mașina folosește un compilator special JIT (*Just In Time*).

Compilatorul JIT analizează codul IL corespunzător apelului unei metode și produce codul mașină adecvat și eficient. El recunoaște secvențele de cod pentru care s-a obținut deja codul mașină adecvat, permițând reutilizarea acestuia fără recompilare, ceea ce face ca, pe parcursul rulării, aplicațiile .NET să fie din ce în ce mai rapide.

Faptul că programul IL produs de diferitele limbaje este foarte asemănător are ca rezultat interoperabilitatea între aceste limbaje. Astfel, clasele și obiectele create într-un limbaj specific .NET pot fi utilizate cu succes în altul.

În plus, CLR se ocupă de gestionarea automată a memoriei (un mecanism implementat în platforma .NET fiind acela de eliberare automată a zonelor de memorie asociate unor date devenite inutile – *Garbage Collection*).

Ca un element de portabilitate, trebuie spus că .NET Framework este implementarea unui standard numit Common Language Infrastructure (<http://www.ecma-international.org/publications/standards/Ecma-335.htm>), ceea ce permite rularea aplicațiilor .NET, în afară de Windows, și pe unele tipuri de Unix, Linux, Solaris, Mac OS X și alte sisteme de operare (http://www.mono-project.com/Main_Page).

De ce am alege .NET?

În primul rând pentru că ne oferă instrumente pe care le putem folosi și în alte programe, oferă acces ușor la baze de date, permite realizarea desenelor sau a altor elemente grafice.

Spațiul de nume System.Windows.Forms conține instrumente (controale) ce permit implementarea elementelor interfeței grafice cu utilizatorul. Folosind aceste controale, puteți proiecta și dezvolta rapid și interactiv, elementele interfeței grafice. Tot .NET vă oferă clase care efectuează majoritatea sarcinilor uzuale cu care se confruntă programele și care plictisesc și fură timpul programatorilor, reducând astfel timpul necesar dezvoltării aplicațiilor.

Introducere în limbajul C#

Caracterizare

Limbajul C# a fost dezvoltat de o echipă restrânsă de ingineri de la Microsoft, echipă din care s-a evidențiat Anders Hejlsberg (autorul limbajului Turbo Pascal și membru al echipei care a proiectat Borland Delphi).

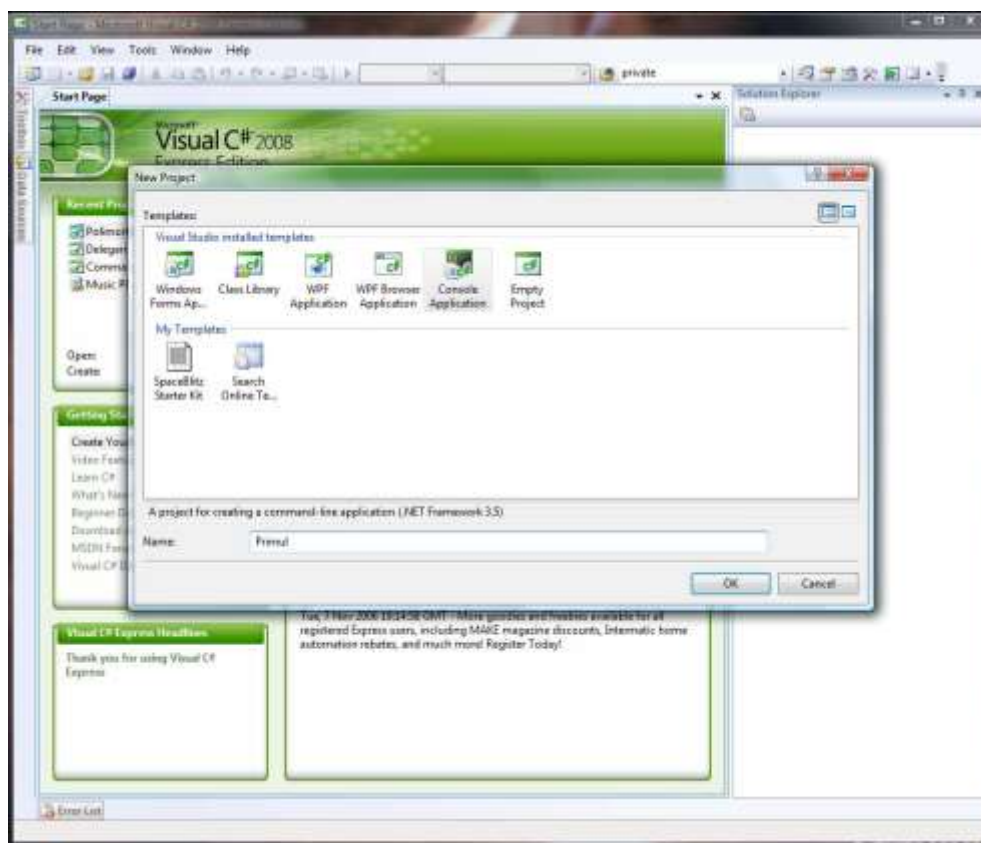
C# este un limbaj simplu, cu circa 80 de cuvinte cheie și 12 tipuri de date predefinite. El permite programarea structurată, modulară și orientată obiectual, conform percepțelor moderne ale programării profesionale.

Principiile de bază ale programării orientate pe obiecte (ÎNCAPSULARE, MOȘTENIRE, POLIMORFISM) sunt elemente fundamentale ale programării C#. În mare, limbajul moștenește sintaxa și principiile de programare din C++. Sunt o serie de tipuri noi de date sau funcțiuni diferite ale datelor din C++, iar în spiritul realizării unor secvențe de cod sigure (*safe*), unele funcțiuni au fost adăugate (de exemplu, interfețe și delegări), diversificate (tipul *struct*), modificate (tipul *string*) sau chiar eliminate (moștenirea multiplă și pointerii către funcții). Unele funcțiuni (cum ar fi accesul direct la memorie folosind pointeri) au fost păstrate, dar secvențele de cod corespunzătoare se consideră „nesigure”.

Crearea aplicațiilor consolă

Pentru a realiza aplicații consolă (ca și cele din Borland Pascal sau Borland C) în mediul de dezvoltare Visual Studio, trebuie să instalăm o versiune a acestuia, eventual mediul free **Microsoft Visual C# 2008 Express Edition** de la adresa <http://www.microsoft.com/express/download/>

După lansarea aplicației, din meniul File se alege opțiunea NewProject apoi alegem ConsoleApplication, modificând numele aplicației în caseta Name.



Când creați o aplicație consolă, se generează un fișier cu extensia .cs. În cazul nostru, s-a generat fișierul **Primul.cs**. Extensia **cs** provine de la **C Sharp**. Redenumirea lui se poate realiza din fereastra **Solution Explorer**, pe care o puteți afișa cu ajutorul combinației de taste **Ctrl+W,S** sau din meniul **View**.

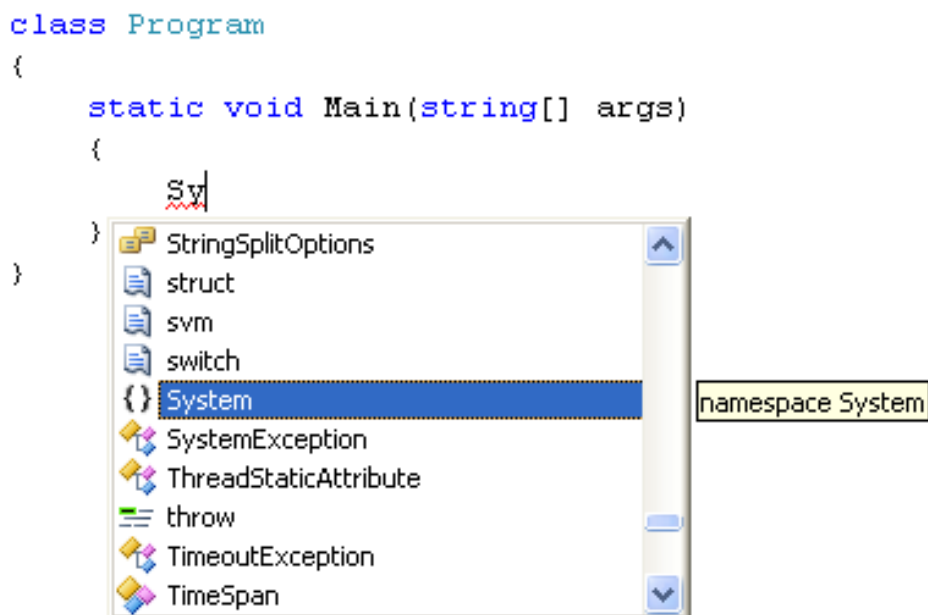
Codul sursă generat este :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Completați funcția Main cu următoarea linie de program:

```
Console.WriteLine("Primul program");
```

Veți observa că în scrierea programului sunteți asistați de **IntelliSense**, ajutorul contextual.





























Pentru compilarea programului, selectați **Build** din meniul principal sau apăsați tasta **F6**. În cazul în care aveți erori, acestea sunt afișate în fereastra **Error List**. Efectuând dublu-clic pe fiecare eroare în parte, cursorul din program se poziționează pe linia conținând eroarea.

Rularea programului se poate realiza în mai multe moduri:

- rapid fără asistență de depanare (*Start Without Debugging* **Ctrl+F5**)
- rapid cu asistență de depanare (*Start Debugging* **F5** sau cu butonul din bara de instrumente)
- rulare pas cu pas (*Step Into* **F11** și *Step Over* **F10**)
- rulare rapidă până la linia marcată ca punct de întrerupere (*Toggle Breakpoint* **F9** pe linia respectivă și apoi *Start Debugging* **F6**). Încetarea urmăririi pas cu pas (*Stop Debugging* **Shift+F5**) permite ieșirea din modul

depanare și revenirea la modul normal de lucru. Toate opțiunile și rulare și depanare se găsesc în meniul *Debug* al mediului de programare.

Icoanele din IntelliSense și semnificația lor

	Assembly		Public enumeration
	Delegate		Protected enumeration
	Event		Private enumeration
	Interface		Constant inside enumeration
	Namespace		Public method
	Structure		Protected method
	Internal class		Private method
	Public class		Public property
	Protected class		Protected property
	Private class		Private property
	Public constant		Public variable
	Protected constant		Protected variable
	Private constant		Private variable

Structura unui program C#

Majoritatea cărților care tratează limbaje de programare încep cu un exemplu, devenit celebru, apărut pentru prima dată în ediția din 1978 a cărții „The C Programming Language” a lui Brian W. Kernighan și Dennis M. Ritchie, „părinții” limbajului C. Vom prezenta și noi acest exemplu adaptat la limbajul C#:

```
1 using System;
2
3 namespace HelloWorld
4 {
5     class Program
6     {
7         static void Main()
8         {
9             Console.WriteLine("Hello World!");
10        }
11    }
12 }
```

O aplicație C# este formată din una sau mai multe **clase**, grupate în **spații de nume (namespaces)**. Este obligatoriu ca doar una din aceste clase să conțină un „punct de intrare” (**entry point**), și anume metoda (funcția) **Main**.

- **Clasa (class)**, în termeni simplificați, reprezintă principalul element structural și de organizare în limbajele orientate spre obiecte, grupând date cât și funcții care prelucrează respectivele date.

- **Spațiul de nume (Namespaces):** din rațiuni practice, programele mari, sunt divizate în module, dezvoltate separat, de mai multe persoane. Din acest motiv, există posibilitatea de a apărea identificatori cu același nume. Pentru a evita erori furnizate din acest motiv, în 1955 limbajul C++ introduce noțiunea și cuvântul cheie **namespace**. Fiecare mulțime de definiții dintr-o librărie sau program este grupată într-un spațiu de nume, existând astfel posibilitatea de a avea într-un program definiții cu nume identic, dar situate în alte spații de nume. În cazul în care, într-o aplicație, unele clase sunt deja definite, ele se pot folosi importând spațiile de nume care conțin definițiile acestora. Mai menționăm faptul că un spațiu de nume poate conține mai multe spații de nume.

Să comentăm programul de mai sus:

linia 1: este o directivă care specifică faptul că se vor folosi clase incluse în spațiul de nume

System. În cazul nostru, se va folosi clasa **Console**.

linia 3: spațiul nostru de nume

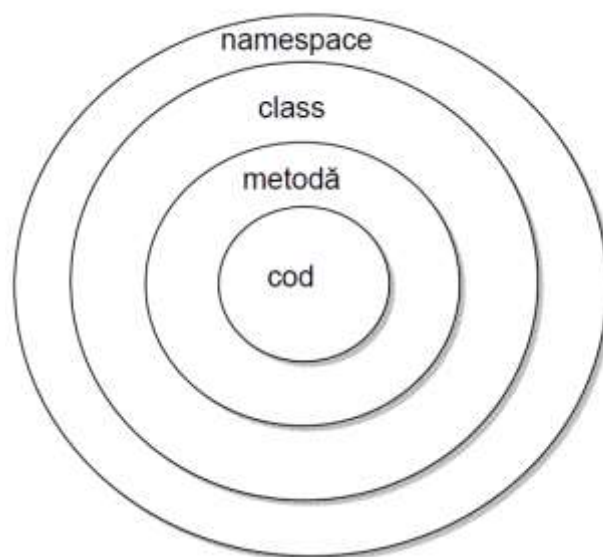
linia 5: orice program C# este alcătuit din una sau mai multe clase

linia 7: metoda **Main**, „punctul de intrare” în program

linia 9: clasa **Console**, amintită mai sus, este folosită pentru operațiile de intrare/ieșire.

Aici se apelează metoda **WriteLine** din această clasă, pentru afișarea mesajului dorit pe ecran.

În C#, simplificat vorbind, un program poate fi privit ca având mai multe „straturi”: avem cod în interiorul metodelor, care, la rândul lor, se află în interiorul claselor, aflate în interiorul **namespaces**-urilor.



Convenție: S-a adoptat următoarea convenție de scriere: în cazul în care folosim nume compuse din mai multe cuvinte, fiecare cuvânt este scris cu majusculă: **HelloWorld**, **WriteLine**. Această convenție poartă numele de **Convenție Pascal**. Asemănătoare este **Convenția cămilă**, cu diferența că primul caracter din primul cuvânt este literă mică.

Sintaxa limbajului

Ca și limbajul C++ cu care se înrudește, limbajul C# are un alfabet format din litere mari și mici ale alfabetului englez, cifre și alte semne. Vocabularul limbajului este format din acele „simboluri” cu semnificații lexice în scrierea programelor: cuvinte (nume), expresii, separatori, delimitatori și comentarii.

Comentarii

Limbajul C# admite trei tipuri de comentarii:

- **comentariu pe un rând** prin folosirea `//`. Tot ce urmează după caracterele `//` sunt considerate, din acel loc până la sfârșitul rândului, drept comentarii.

```
// Acesta este un comentariu pe un singur rand
```

- **comentariu pe mai multe rânduri** prin folosirea `/*` și `*/`. Orice text cuprins între simbolurile menționate mai sus se consideră a fi comentariu. Simbolurile `/*` reprezintă începutul comentariului, iar `*/` sfârșitul respectivului comentariu.

```
/* Acesta este un  
comentariu care se  
intinde pe mai multe randuri */
```

- **creare document în format XML** folosind `///`. Nepropunându-ne să intrăm în amănunte, amintim că XML (eXtensible Markup Language) a fost proiectat în scopul transferului de date între aplicații pe Internet, fiind un model de stocare a datelor nestructurate și semi-structurate.

Nume

Definiție: Prin **nume** dat unei variabile, clase, metode etc. înțelegem o succesiune de caractere care îndeplinește următoarele reguli:

- numele trebuie să înceapă cu o literă sau cu unul dintre caracterele `_` și `@`;
- primul caracter poate fi urmat numai de litere, cifre sau un caracter de subliniere;
- numele care reprezintă cuvinte cheie nu pot fi folosite în alt scop decât acela pentru care au fost definite;
- cuvintele cheie pot fi folosite în alt scop numai dacă sunt precedate de `@`;
- două nume sunt distincte dacă diferă prin cel puțin un caracter (fie el și literă mică ce diferă de aceeași literă majusculă).

Convenții pentru nume:

- în cazul numelor claselor, metodelor, a proprietăților, enumerărilor, interfețelor, spațiilor de nume, fiecare cuvânt care compune numele începe cu majusculă;
- în cazul numelor variabilelor, dacă numele este compus din mai multe cuvinte, primul începe cu minusculă, celelalte cu majusculă.