Medii vizuale de programare
Curs 12

Conf. dr.ing. GENGE Béla

Universitatea "Petru Maior", Departamentul de Informatică
Tîrgu Mureș, Romania
bela.genge@ing.upm.ro

Operatorul Lambda

- Elementul "=>" poartă denumirea de operator lambda.
- Se utilizează în expresii de tip lamba pentru separarea argumentelor (intrărilor, specificate în partea stângă) de expresia asupra căruia se aplică (partea dreaptă).

Lambda Expresii (expression lambdas)

- În cazul în care în partea dreaptă avem o expresie, avem *lambda expresii*.

C#

```
(input-parameters) => expression
```

C#

```
(int x, string s) => s.Length > x
```

C#

```
(x, y) => x == y
```

C#

```
() => SomeMethod()
```

Lambda Statements

- Aceste lambda-uri generează metode anonmime (fără nume) care pot fi asignate la un delegat.

C#

```
delegate void TestDelegate(string s);
```

C#

```
TestDelegate del = n => { string s = n + " World";
                          Console.WriteLine(s); };
```

Programarea asincronă

- Elimină implementarea comunicațiilor între fire de execuție.
- Programarea asincronă este utilă în foarte multe scenarii (ex. apeluri cu blocare în comunicații), dar în special în cazul accesării UI din alte fire.
- Cuvinte cheie: **async** și **await**.
- Metodele definite prin **async** sunt numite metode asincrone.
- Versiunea minimă: .NET 5 (VS 2012).

Exemplu 1

```
static void Main(string[] args)
{
    // Lansare task async
    Task<int> task = TestAsyncTask();

    // Controlul este returnat fara blocarea firului principal
    Console.WriteLine("Executie operatii pe firul principal");

    // Optional: se poate bloca firul principal pentru asteptarea rezultatului
    task.Wait();
    var x = task.Result;
    Console.WriteLine("Rezultatul returnat: " + x);
}
```

Exemplu 1

```
private async static Task<int> TestAsyncTask()
{
    // WriteLine se executa inca pe firul principal
    Console.WriteLine("TestAsyncTask1: " + Thread.CurrentThread.ManagedThreadId);
    int res = await DoAsync();
    // WriteLine se executa pe un fir ales din ThreadPool
    Console.WriteLine("TestAsyncTask2: " + Thread.CurrentThread.ManagedThreadId);
    return res;
}

private async static  Task<int> DoAsync()
{
    // WriteLine se executa inca pe firul principal
    Console.WriteLine("DoAsync1: " + Thread.CurrentThread.ManagedThreadId);
    await Task.Delay(5000);
    // WriteLine se executa pe un fir ales din ThreadPool
    Console.WriteLine("DoAsync2: " + Thread.CurrentThread.ManagedThreadId);
    return 10;
}
```

Exemplu 2

```
private async static Task<int> DoAsyncEx()
{
    // WriteLine se executa inca pe firul principal
    Console.WriteLine("DoAsyncEx1: " + Thread.CurrentThread.ManagedThreadId);
    // ComputeSomething() se executa pe un fir ales din ThreadPool
    int xx = await Task.Run(() => ComputeSomething());
    // WriteLine se executa pe un fir ales din ThreadPool
    Console.WriteLine("DoAsyncEx2: " + Thread.CurrentThread.ManagedThreadId);
    return xx;
}

private static int ComputeSomething()
{
    Console.WriteLine("ComputeSomething1: " + Thread.CurrentThread.ManagedThreadId);
    Thread.Sleep(4000);
    Console.WriteLine("ComputeSomething2: " + Thread.CurrentThread.ManagedThreadId);
    return 100;
}
```

LINQ

- .NET Language Integrated Query (LINQ).
- LINQ reprezintă un instrument puternic integrat în limbajul C# pentru interogarea unei palete diversificate de resurse.
- LINQ definește un set de operatori care permit interogarea, stocarea și traversarea datelor - toate acestea fiind realizate prin intermediul limbajului C#.

Exemplu - limbajul LINQ

```
using System;
using System.Linq;
using System.Collections.Generic;

class app {
  static void Main() {
    string[] names = { "Burke", "Connor", "Frank",
                       "Everett", "Albert", "George",
                       "Harris", "David" };

    IEnumerable<string> query = from s in names
                                where s.Length == 5
                                orderby s
                                select s.ToUpper();

    foreach (string item in query)
      Console.WriteLine(item);
  }
```

Exemplu - metode anonime

```
Func<string, bool>   filter  = delegate (string s) {
                                   return s.Length == 5;
                               };

Func<string, string> extract = delegate (string s) {
                                   return s;
                               };

Func<string, string> project = delegate (string s) {
                                   return s.ToUpper();
                               };

IEnumerable<string> query = names.Where(filter)
                                 .OrderBy(extract)
                                 .Select(project);
```

Structura generală a unui query LINQ

```
query-expression ::= from-clause query-body

query-body ::=

      query-body-clause* final-query-clause query-continuation?

query-body-clause ::=
  (from-clause
      | join-clause
      | let-clause
      | where-clause
      | orderby-clause)
```

Structura generală a unui query LINQ

```
from-clause ::= from itemName in srcExpr

join-clause ::= join itemName in srcExpr on keyExpr equals keyExpr
       (into itemName)?

let-clause ::= let itemName = selExpr

where-clause ::= where predExpr

orderby-clause ::= orderby (keyExpr (ascending | descending)?)*

final-query-clause ::=
  (select-clause | groupby-clause)

select-clause ::= select selExpr

groupby-clause ::= group selExpr by keyExpr

query-continuation ::= into itemName query-body
```

Exemple

```
var query1 = from p in people
             where p.Age > 20
             orderby p.Age descending, p.Name
             select new {
                 p.Name, Senior = p.Age > 30, p.CanCode
             };

var query2 = from p in people
             where p.Age > 20
             orderby p.Age descending, p.Name
             group new {
                p.Name, Senior = p.Age > 30, p.CanCode
             } by p.CanCode;

var query = from s1 in names
            where s1.Length == 5
            from s2 in names
            where s1 == s2
            select s1 + " " + s2;
```

LINQ și XML

- LINQ include operatori și clase pentru a opera asupra elementelor unui fișier XML.
- În principiu sunt definite trei clase principale: `XName`, `XElement` și `XAttribute`.
- Exemplu de generare a unui string XML:

```
var e = new XElement("Person",
                     new XAttribute("CanCode", true),
                     new XElement("Name", "Loren David"),
                     new XElement("Age", 31));

var s = e.ToString();
```

- Rezultatul:

```
<Person CanCode="true">
  <Name>Loren David</Name>
  <Age>31</Age>
```

LINQ și XML

- Suport definit în System.Xml.Linq.
- Citirea și accesarea elementelor dintr-un XML:

```
XDocument doc = XDocument.Load("XMLFile1.xml");
var people = from p in doc.Descendants("Persoana")
             select new
             {
                 Nume = p.Element("Nume").Value,
                 Prenume = p.Element("Prenume").Value,
                 CNP = p.Element("Detalii").Attribute("CNP"),
                 Adresa = p.Element("Detalii").Attribute("Adresa")
             };
foreach (var p in people)
{
    System.Console.WriteLine(p.Nume + p.Prenume + p.CNP + p.Adresa);
}
```

LINQ și XML

- Citirea și accesarea elementelor dintr-un XML (cu clauze `Where`):

```
XDocument doc = XDocument.Load("XMLFile1.xml");
var people = from p in doc.Descendants("Persoana")
             where (string)p.Element("Detalii").Attribute("CNP") == "212121"
             select new
             {
                 Nume = p.Element("Nume").Value,
                 Prenume = p.Element("Prenume").Value,
                 CNP = p.Element("Detalii").Attribute("CNP"),
                 Adresa = p.Element("Detalii").Attribute("Adresa")
             };
foreach (var p in people)
{
    System.Console.WriteLine(p.Nume + p.Prenume + p.CNP + p.Adresa);
}
```