Lambda kifejezések
LINQ

Vastag Atila
2019

Lambda kifejezések

- Új operátor: **=>** (Lambda operátor) , a bemenet és a kimenet összekötésére használt
- A lambda kifejezések tényleges ereje a LINQ módszerekkel együttesen használva derül ki, ahol a paraméter részbe kerülhetnek lambda kifejezések

int[] numbers = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

Az ötnél nagyobbak listája

IEnumerable<int> otnelNagyobb = numbers.Where(n => n > 5);

A páratlanok darabszáma

Int paratlanSzamokSzama = numbers.Count(n => n % 2 == 1);

Nézzük, hogy ez hogyan is történik…

LINQ = Language Integrated Queries

LINQ to … ?

- LINQ To Objects: Memóriában lévő halmazek, listák, tömbök feldolgozását teszi lehetővé (lényegében minden olyan osztállyal működik amely megvalósítja az *IEnumerable<T>* interfészt*).*
- LINQ To XML: XML dokumentumok lekérdezését és szerkesztését teszi lehetővé.
- LINQ To SQL (DLINQ) és LINQ To Entities (Entity Framework, VS2008SP1): relációs adatbázisokon végezhetünk műveleteket (MS SQL-Server, de vannak kiterjesztések más szerverekhez is)

LINQ halmazok

```
public struct Diak
{
    public string Nev;
    public int Kreditek;

    public Diak()
    {}

    public Diak(string nev, int kreditek)
    {
        Nev = nev; Kreditek = kreditek;
    }
}
```

LINQ halmazek

```
int[] elso = new int[] { 2, 4, 6, 8, 2, 1, 2, 3 };


int[] masodik = new int[] { 1, 3, 5, 7, 1, 1, 2, 3 };


string[] strtomb = new string[] { "Béla", "Jolán", "Bill",
 "Shakespeare", "Verne", "Jókai" };


List<Diak> diakok = new List<Diak>();
diakok.Add(new Diak("Első Egon", 52));
diakok.Add(new Diak("Második Miksa", 97));
diakok.Add(new Diak("Harmadik Huba", 10));
diakok.Add(new Diak("Negyedik Néró", 89));
diakok.Add(new Diak("Ötödik Ödön", 69));
```

Használható operátorok

- Rendezés:
  - **OrderBy** – tetszőleges sorrend
  - **Reverse** – sorrend megfordítása
- Halmaz-kezelés:
  - **Concat** – egymás után másolás
  - **Contains** – elem meglétének vizsgálata
  - **Distinct** – ismétlések szűrése
  - **Intersect** – halmazelméleti metszet
  - **Union** – halmazelméleti únió
  - **Except** – Halmazelméleti különbség
- Szűrés:
  - **Where** – Keresés (szűrés), jelentése *ahol*
  - **Any** – Van-e olyan elem a halmazban, amely megfelel a feltételnek

Használható operátorok

- „Számolás” („Aggregate methods”, aggregáló eljárások):
  - **Average** – átlag
  - **Count** – darabszám
  - **Max** – Maximum
  - **Min** – Minimum
  - **Sum** – Összeg
- Csoportosítás:
  - **GroupBy** – valamilyen közös tulajdonság szerint lehet csoportosítani a tagokat

Mi ezeket egyszerű tömbökön / listákon fogjuk használni !!!

LINQ Operátorok 1. – Megfordítás

- Tegyük fel, hogy a halmaz alaptípusa int[], és egyszerűen szűrünk / megfordítunk ➔ tudjuk, hogy a kimenet is int[] lesz
- Kimeneti típusnak EKKOR SEM HASZNÁLHATÓ int[], mert a meghívandó eljárás általánosan **[generic]** van megírva, ugyanaz az eljárás hívódik meg minden típusú halmazra

➔ Mindig **IEnumerable<T>** (a mi esetünkben így **IEnumerable<int>** lesz az eredmény típusa)

➔ Tömbbé konvertálható .ToArray(); függvénnyel

➔ Listává konvertálható a .ToList() függvénnyel

```
IEnumerable<int> forditott = elso.Reverse();
foreach (var szam in forditott) { Console.Write($"{szam}");}
Console.ReadLine();
```

➔3, 2, 1, 2, 8, 6, 4, 2

LINQ Operátorok 2. – Halmazok

- Elem létezésének vizsgálata:

```
bool bennevan=elso.Contains(4);
```

- Két halmaz egymás után fűzése (NEM halmazok!):

```
IEnumerable<int> uj = elso.Concat(masodik);
```

- Ismétlődések kivágása (halmazzá alakítás):

```
IEnumerable<int> uj = elso.Distinct();
```

- Halmazelméleti metszet:

```
IEnumerable<int> uj = elso.Intersect(masodik);
```

- Halmazelméleti únió:

```
IEnumerable<int> uj = elso.Union(masodik);
```

- Halmazelméleti különbség

```
IEnumerable<int> uj = elso.Except(masodik);
```

LINQ Operátorok 3. – Sorrendezés

- OrderBy
  - **Paraméterül egy olyan eljárást vár, amely egy osztályból kiszedi a kulcsot (azt a mezőt, ami alapján rendezni fog) (Ehelyett egy lambda kifejezést szokás írni)**
  - ***Második paraméterként megadható neki egy saját, IComparer interfészt implementáló osztály, ami az összehasonlítást végzi***
  - **int tömb, rendezés az elemek alapján:**

```
IEnumerable<int> uj = elso.OrderBy(x => x);
```

  - **String tömb, rendezés az elemek hossza alapján:**

```
IEnumerable<string> uj = strtomb.OrderBy(x => x.Length);
```

  - **Diákok listája, névsorba rendezés :**

```
IEnumerable<Diak> uj = diakok.OrderBy(x => x.nev);
```

LINQ Operátorok 4. – Szűrés, darabszámolás

- **Where / Count**

  - A paraméterül adott kifejezésnek bool típust kell visszaadni.
  - A *Where* eredménye az a halmaz, ahol ez *true* értéket ad vissza.
  - A *Count* eredménye a darabszám (int!!)
  - A count meghívható paraméter nélkül is ➔ teljes darabszám

  - *int tömb, a páratlanok:*

```
IEnumerable<int> uj = elso.Where(x => x % 2 == 0);
```

  - *string tömb, a négy betűs nevek:*

```
int num = strtomb.Count(x => x.Length == 4);
```

LINQ Operátorok 4. – Szűrés, darabszámolás

- **Any**

  - A paraméterül adott kifejezésnek bool típust kell visszaadni.
  - Nem feltétel a paraméter, olyankor azt vizsgálja, hogy van-e a halmazban elem (min. egy)
  - Az *Any* eredménye mindig egy *bool* kifejezés ➔ van e legalabb egy olyan elem a halmazban, mely megfelel a feltételeknek (ha van feltétel)

  - *van-e olyan diak, akinek a kreditpontja 52:*

```
bool vanE = diakok.Any(diak => diak.Kreditek == 52);
```

  - *van-e elem az **elso** elnevezésű tömben:*

```
bool vanEBenneElem = elso.Any();
```

LINQ Operátorok 5. – Szűrés, részkiválasztás

- Diákok listája, csak név:

```
IEnumerable<string> uj = diakok.Select(x => x.nev);
```

- Diákok listája, ahol a kreditszám prím:

```
var uj = diakok.Where(x =>
{
    for (int i = 2; i <= Math.Sqrt(x.kreditek); i++)
    {
        if (x.kreditek % i == 0) return false;
        return true;
    }
});

// Második Miksa - 97, Negyedik Néró - 89
```

LINQ Operátorok 6. – Több operátor

- Diákok listája, a páratlan kreditszámúak nagybetűs neve név szerinti fordított sorrendben:

```
IEnumerable<string> uj = diakok.Where(x => x.kreditek % 2 == 1)
                               .OrderBy(x => x.nev)
                               .Reverse()
                               .Select(x => x.nev.ToUpper());

// ÖTÖDIK ÖDÖN, NEGYEDIK NÉRÓ, MÁSODIK MIKSA
```

- Ugyanaz az eredmény, ugyanaz a köztes kód:

```
var IEnumerable<string> = from diak in diakok
                          where diak.kreditek % 2 == 1
                         orderby diak.nev descending
                          select diak.nev.ToUpper();
```

LINQ Operátorok 7. – Aggregálás

- Aggregáló metódusok

```
int ossz = elso.Sum(); //28
double atlag = masodik.Average(); //2.875
int parosOssz = elso.Where(x => x % 2 == 0)
                    .Sum(); //24
int paratlanOssz = elso.Where(x => x % 2 == 1)
                       .Sum(); //4
```

- A fenti példa gyakori: valamilyen ismétlődés szerint akarom csoportosítani a halmazemet, és az egyes csoportokra szeretném tudni a darabszámot/összeget
- Csoportonként egy where+aggregálás ➔ zavaró ➔ automata csoportosítás: **GroupBy**

LINQ Operátorok 8. – Csoportosítás

- Csoportosítás, paritás szerinti darabszámok:

```
var csoport = elso.GroupBy(x => x % 2);
foreach (var g in csoport)
{
    Console.WriteLine($"Maradék: {g.Key}, darabszám:
 {g.Count()}");
}

var uj = from x in elso
         group x by x % 2 into g
         select new
          {
            Maradek=g.Key,
            Darab=g.Count()
          };
```

Gyakorlat

```
public struct Jatekos
{
        public string Nev;
        public int Magassag;
        public string Pozicio;
        public string Nemzetiseg;
        public string Csapat;
        public string CsapatOrszaga;
}
```

1 - Keressük ki az ütő játékosokat.

2 - Rendezzük a játékosokat magasság szerint növekvő sorrendbe.

3 - Rendezzük a játékosokat magasság szerint növekvő, majd nev szerint csökkenő sorrendbe.

4 – Keresse ki a Magyaroszáon játszó játékosok nevét.

5 – Mekkora az átlagmagassága a feladóknak?

6 – Hány országból vannak játékosok az ‘adatbázisban’?

7 – Hány olyan játkos van, akinek a neve hosszabb mint 15 karakter?

8 - Mely nemzetiségek képviseltetik magukat a röplabdavilágban mint játékosok és milyen számban.

9 - Keressük azon játékosok nevét és magasságát akik magasabbak mint az „adatbázisban” szereplő játékosok átlagos magasságánál.

10 - Állítsa növekvő sorrendbe a posztok szerint a játékosok átlag magasságát.