

Języki Programowania na Platformie .NET

LINQ Language-Integrated Query

LINQ pozwala na dostęp do danych w formie podobnej do SQL.
Możliwy jest dostęp do danych:

- znajdujących się w kolekcjach i tablicach,
- w bazie danych SQL,
- w pliku XML,
- w obiekcie DataSet.

Operacja z użyciem LINQ przebiega następująco:

- uzyskanie źródła/źródeł danych
- zbudowanie zapytania
- wykonanie zapytania

Przykład

```
Dim x() As Integer = {5,3,6,3,1,0,9,7,-2,11,43}
```

```
' utworzenie zapytania:
```

```
Dim male = From liczba In x  
           Where liczba < 10  
           Select liczba
```

```
' wykonanie zapytania:
```

```
For Each Dim n In male  
    Console.WriteLine(n)  
Next
```

Zapytanie do tablicy

- zastosowanie LINQ do tablicy jest możliwe, ponieważ tablice implementują interfejs `IEnumerable(Of T)`
- to samo odnosi się do wszystkich typów, które implementują ten interfejs (np. `List(Of T)`, `Dictionary(Of TKey, TValue)`) lub pochodny jak `IQueryable(Of T)`
- w przypadku typów, które nie implementują takich interfejsów potrzebny jest odpowiedni *provider*, który implementuje niezbędną funkcjonalność dla standardowych operatorów w zapytaniu, na przykład `XElement` jest typem pozwalającym na dostęp do plików XML poprzez LINQ

Struktura zapytania - podstawy

- zapytanie zaczyna się od słowa kluczowego `From`. Odpowiada ono za identyfikację typu źródła danych oraz zmiennych, które są używane do odnoszenia się do poszczególnych danych (ang. *range variables* lub *iteration variables*). W niektórych zapytaniach słowo to jest opcjonalne,
- `Where` służy do filtrowania wyników; Przykład:
`From k In klienci Where k.imie = "Ala",`
- `Select` służy do zwracania tylko wybranych 'wierszy':
`From k In klienci Where k.imie = "Ala" Select k.nazwisko`

Wybór określonych pól – przykład

```
Dim wynik = From s In studenci
              Where s.Miasto = "Gdansk"
              Order By s.Nazwisko Ascending
              Select Imie = s.Imie, Numer = s.NumerIndeks

For Each stud In wynik
    Console.WriteLine(stud.Imie & " " & stud.NumerIndeks)
Next
```

Złączenia

Złączenie można uzyskać używając odpowiednio operatora Where:

```
Dim vowels() As String = {"A", "E", "I", "O", "U"}
Dim vowelNames = From student In students, vowel In vowels
                  Where student.Last.IndexOf(vowel) = 0
                  Select Name = student.First & " " &
                  student.Last, Initial = vowel
                  Order By Initial
```

```
For Each vName In vowelNames
    Console.WriteLine(vName.Initial & ": " & vName.Name)
Next
```

<http://msdn.microsoft.com/en-us/library/bb384504.aspx>

... lub korzystając z operatora Join:

```
Dim vowelNames2 = From student In students
                   Join vowel In vowels
                   On student.Last(0) Equals vowel
                   Select Name = student.First & " " &
                           student.Last, Initial = vowel
                   Order By Initial
```

<http://msdn.microsoft.com/en-us/library/bb384504.aspx>

Sortowanie

```
Dim slowa = {"ala", "ma", "kota", "i", "psa"}
```

```
Dim zapytanie = From w In slowa  
                Order By w.Length  
                Select w
```

```
Dim x = From k In klienci Order By k.wiek
```

Sortowanie po kilku kryteriach

Przykład sortowania względem długości (rosnąco), oraz (dla jednakowych długości) po pierwszej literze w słowie (rosnąco)

```
Dim slowa = {"ala", "ma", "kota", "i", "psa"}
```

```
Dim zapytanie = From w In slowa  
                Order By w.Length, w.Substring(0,1)  
                Select w
```

Sortowanie po kilku kryteriach

Przykład sortowania względem długości (rosnąco), oraz (dla jednakowych długości) po pierwszej literze w słowie (malejąco)

```
Dim slowa = {"ala", "ma", "kota", "i", "psa"}
```

```
Dim zapytanie = From w In slowa  
                Order By w.Length, w.Substring(0,1) Descending  
                Select w
```

Inne operatory

- Let pozwala na przypisanie wartości, np.:
From p In produkty Let x = p.cena*0.1 Where x > 100,
- Distinct eliminuje duplikaty:
From k In klienci Select k.Imie Distinct.

Operator grupowania

```
Group [ listField1 [, listField2 [...] ] By keyExp1 [, keyExp2  
      Into aggregateList
```

Przykład:

```
Dim PracownikDzial1 = From p In Pracownicy
```

```
Dim PracownikDzial2 = From p In Pracownicy  
                       Group By NazwaDzialu = p.Dzial  
                       Into Dzial = Group, Count()  
                       Order By NazwaDzialu
```

Przykład dostępu do bazy danych

```
Public Sub LinqToSqlAdo01()
```

```
    ' Create a standard ADO.NET connection:
```

```
    Dim nwindConn As SqlConnection = New SqlConnection(My.S  
    nwindConn.Open()
```

```
    ' ... other ADO.NET database access code ... '
```

```
    ' Use pre-existing ADO.NET connection to create DataCon
```

```
    Dim interop_db = New NorthwindDataContext(nwindConn) W
```

```
    Dim orders = From o In interop_db.Orders _  
        Where o.Freight > 500D _  
        Select o
```

Przykład dostępu do kolekcji

```
Dim customers As List(Of Customer) = GetCustomerList()
```

```
Dim customersByCountry = From cust In customers _  
                        Order By cust.Country, cust.City _  
                        Group By CountryName = cust.CountryName _  
                        Into RegionalCustomers = Group, Customers _  
                        Order By CountryName
```

```
For Each country In customersByCountry  
    Console.WriteLine(country.CountryName & " (" & country.Count & ")") & vbCrLf)
```

```
For Each customer In country.RegionalCustomers  
    Console.WriteLine(vbTab & customer.CompanyName & " (" & customer.City & ")")
```

```
Next
```