

Лекція 3. λ - вирази

Література

1. <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/getting-started-with-linq>
2. Anatomy of the Lambda Expression -
<http://www.tutorialsteacher.com/linq/linq-lambda-expression>
3. Фримен, Адам, Раттц-мл., Джозеф С. Ф88 LINQ: язык интегрированных запросов в C# 2010 для профессионалов. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011
4. Полное руководство по языку программирования C# 6.0 и платформе .NET 4.6, -
<http://metanit.com/sharp/tutorial/15.1.php>

Lambda-вирази

C# 3.0(.NET 3.5) - анонсовано LINQ та засоби для інтерпретації лямбда-виразів.

Лямбда-вираз - це найкомпактніший спосіб запису анонімного методу за допомогою спеціального синтаксису.

Використовують поняття:

анонімні функції, методи

делегат.

Лямбда-вираз – це вираз або блок операторів, який реалізує певні дії

Можуть використовуватись в `Linq query`

Приклад 1. Використання делегатів та анонімних функцій

```
public class Program {  
    public class Student {  
        public int Id { get; set; }  
        public string Name { get; set; }  
        public int Age { get; set; }  
    }  
    delegate bool IsTeenAger(Student stud);  
    public static void Main() {  
        IsTeenAger isTeenAger =  
            delegate(Student s) {  
                return s.Age > 12 && s.Age < 20;  
            };  
        Student stud = new Student() { Age = 25 };  
        Console.WriteLine(isTeenAger(stud));  
    }  
}
```

Anonymous Functions

Анонімний метод перевіряє вік:

```
delegate(Student s)
{
    return s.Age > 12 && s.Age < 20;
};
```

Лямбда-вираз - спрощення опису анонімного методу

- видаляють ключове слово `delegate` та тип вхідного параметра (в прикладі `Student`)
- додають оператор `=>` (наз. оператор lamda)

```
delegate(Student s) { return s.Age > 12 && s.Age < 20; };
```



1 - Remove Delegate and Parameter Type and add lamda operator =>

```
delegate(Student s)={ return s.Age > 12 && s.Age < 20; };
```



```
(s) => { return s.Age > 12 && s.Age < 20; };
```

Lambda Expression from Anonymous Method

Якщо в дужках залишається лише один оператор, то його можна опустити

`(s) => { return s.Age > 12 && s.Age < 20; }`

2 - Remove curly bracket, return and semicolon

`(s) => s.Age > 12 && s.Age < 20;`

3 - Remove Parenthesis around parameter if there is only one parameter

`s => s.Age > 12 && s.Age < 20;`

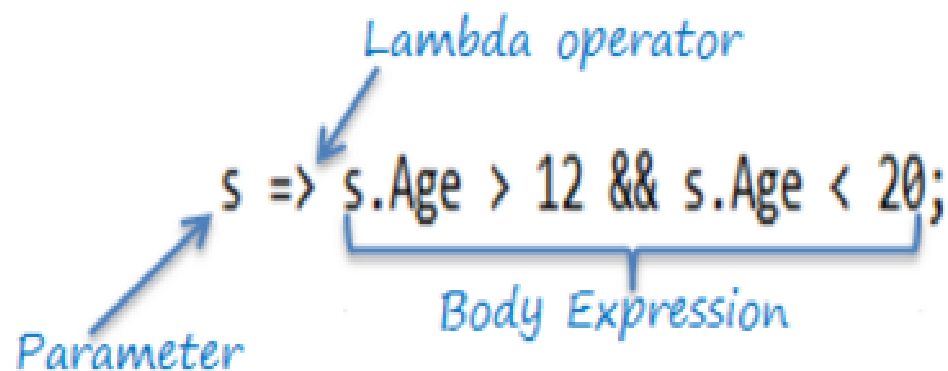
Lambda Expression from Anonymous Method

Lambda Expression `s => s.Age > 12 && s.Age < 20`

`s` is a parameter

`=>` is the lambda operator and

`s.Age > 12 && s.Age < 20` is the body expression



Lambda Expression Structure in C#

Приклад 2. Використання лямбда-виразів

```
public class Program {  
    public class Student{  
        public int Id { get; set; }  
        public string Name { get; set; }  
        public int Age { get; set; }  
    }  
    delegate bool IsTeenAger(Student stud);  
    public static void Main()  
    {  
        IsTeenAger isTeenAger = s => s.Age > 12 && s.Age < 20;  
        Student stud = new Student() { Age = 25 };  
        Console.WriteLine(isTeenAger(stud));  
    }  
}
```


Класифікація Lambda-виразів

Оператор лямбда => відокремлює список параметрів від виразу чи блоку операторів.

Форми лямбда-виразів:

- **права частина містить лише один вираз** або виконує одну дію - фігурні дужки опускаються:

параметр => вираз; - для одного вхідного параметру;

(список_параметрів) => вираз; - для декількох вхідних параметрів:

- **права частина містить багато операторів:**

якщо один вхідний параметр:

```
параметр =>{  
    // оператори  
}
```

якщо декілька:

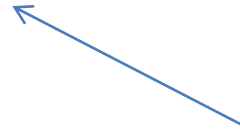
```
(список_параметрів) => {  
    // оператори  
}
```

Lambda-Вираз

$() \Rightarrow ()$



Вхідний параметр, якщо є



Вираз

Приклад: $X \Rightarrow X * X$

Приклад 1. Піднесення до квадрату

```
delegate int mydel(int i);  
static void basic()  
{  
  
    mydel fun = x => x * x;  
    int j = fun(10);  
  
    Console.WriteLine(j);  
  
}
```

Приклад 2. Обчислення факторіалу

```
delegate int fact(int f);  
static void factorial()  
{  
    fact fac = null;  
    fac = n => (n <= 1) ? 1 : n * fac(n - 1);  
  
    Console.WriteLine(fac(3));  
}
```

Lambda-вирази з декількома параметрами

(s, youngAge) => s.Age >= youngage;

(Student s,int youngAge) => s.Age >= youngage;


Приклад 3. Обчислення добутку двох чисел

```
delegate int mydelTwo(int i,int j);  
static void basicTwo()  
{  
  
    mydelTwo fun = (x,y) => x * y;  
    int j = fun(10,20);  
  
    Console.WriteLine(j);  
  
}
```

LINQ і Lambda вирази

```
var students = from s in studentList
                where s.age > 20
                select s;
```

Standard Query Operators



Standard Query Operators in Query Syntax

Standard Query Operators in Method Syntax:

```
var students = studentList.Where(s => s.age > 20).ToList<Student>();
```

*Standard Query Operators
(Extension methods)*



Standard Query Operators in Method Syntax

Приклад 4. Дано два списки користувачів. Написати процедури валідації користувачів та пошуку максимального

Use expressions to manipulate data ..

```
public class Customer
{
    public int ID{ get; set; }
    public String FirstName { get; set; }
    public String LastName { get; set; }
    public int Age { get; set; }
    public List<Invoice> Invoices = new List<Invoice>();
}

static List<Customer> One = new List<Customer>();
static List<Customer> Two = new List<Customer>();
```


Приклад 4. Дано два списки користувачів. Написати процедури валідації користувачів та пошуку максимального за ID, загальний вік, тощо.

Використаємо лямбда-вирази для обробки даних

```
public class Customer
{
    public int ID{ get; set; }
    public String FirstName { get; set; }
    public String LastName { get; set; }
    public int Age { get; set; }
    public List<Invoice> Invoices = new List<Invoice>();
}

static List<Customer> One = new List<Customer>();
static List<Customer> Two = new List<Customer>();
```

//параметром делегата є об'єкт типу клас

```
delegate bool validateCustomer(Customer c);  
static void validateCus()  
{  
  
    validateCustomer check = (cus) => cus.Age > 12 && cus.FirstName == "A";  
  
    Customer c = new Customer();  
    c.Age = 45;  
    c.FirstName = "A";  
  
    if (check(c))  
        Console.WriteLine("Valid");  
    else  
        Console.WriteLine("Not Valid");  
}
```

Методи розширення – як лямбда-вирази

- Max <>

```
// Max Customer ID  
Console.WriteLine(One.Max(c=>c.ID));
```

- Sum <>

```
// Get Sum of age  
Console.WriteLine(One.Sum(c => c.Age));
```

```
// Get Sum of all invoices  
Console.WriteLine(One.Sum(c => c.Invoices.Sum(d => d.amount)));
```

Методи розширення – як лямбда-вирази

- First<>

```
// Get the first Customer  
Console.WriteLine(One.First().LastName);
```

```
// Get the first whos ID >2  
Console.WriteLine(One.First(c => c.ID > 2).LastName);
```

- Last<>

```
// Get the last Customer  
Console.WriteLine(One.Last().LastName);
```

Методи розширення – як лямбда-вирази

- FindAll <>

```
List<Customer> found = One.FindAll(c=> c.LastName.Contains("A"));
```

- Find <>

- ForEach

```
Two.ForEach(c => c.ID = c.ID * 2);
```

get, set і лямбда

```
class Point {  
    int x;  
    int y;  
  
    public int X  
    {  
        get => x;  
        set => x = value;  
    }  
    public int Y  
    {  
        get => y;  
        set => y = value;  
    }  
    public Point(int x, int y, int color)  
    {  
        this.x = x;  
        this.y = y;  
    }  
}
```

Deferred vs Immediate Query Execution in LINQ

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/introduction-to-linq-queries>

<http://qaru.site/questions/18913/what-is-the-difference-between-iqueryablet-and-ienumerablet>

<https://www.dotnetcurry.com/linq/750/deferred-vs-immediate-query-execution-linq>