LINQ

Les délégués

LINQ et les expressions lambda

Délégué (1)

- En VS, un délégué est un type de donnée qui agit en tant que référence sur une méthode.

- Voici des exemples:

```
public int additionner(int x, int y)
{
    return x + y;
}
public int multiplier(int x, int y)
{
    return x * y;
}
```

Délégué (2)

- Déclaration du type délégué:
  ```
  public delegate int TypeOperation(int x, int y);
  ```

- Appel de la méthode déléguée:
  ```
  TypeOperation operation = additionner;
  int resultat = operation(3,4);
  operation = multiplier;
  int resultat = operation(3,4) ;
  ```

Ici, operation est une variable qui fait référence à la méthode qui va être exécutée.

Expression *lambda* (1)

- En *Visual Studio*, une expression *lambda* est une méthode anonyme qui agit en tant que délégué.
- Les expressions *lambda* ne sont pas typés (VS fait de l'inférence de type)

Expression *lambda* (2)

- Sa syntaxe est la suivante:

<paramètre(s) de l'expression> => <expression>

- Par exemples:

  Pour multiplier x à lui-même

  x => x * x

  Pour multiplier x par y

  (x,y) => x * y

  Pour additionner x et y

  (x,y) => x + y

Expression *lambda* (3)

- Déclaration du type délégué:

```
public delegate  int TypeOperation(int  x, int y);
```

- Appel de la méthode déléguée:

```
TypeOperation operation = (x,y)=> x+y;
 var resultat = operation(3,4);
 operation = (x,y) => x*y;
 var resultat = operation(3,4);
```

Ici, operation est une variable qui contient une expression lambda. Cette expression n'a pas de nom.

Expressions *lambda* et *LINQ*

Les expressions lambdas sont à la base de *LINQ*

Par exemples:

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

- Compter le nombre de nombres pairs

```
int  nbPairs = nombres.Count(n => n % 2 == 0);
```

- Faire la somme de tous les nombres plus petits que 6

```
var  sommePPQue6 = nombres.Sum(n => n  < 6);
```

Expressions *lambda* et *LINQ*

Les expressions lambdas sont à la base de *LINQ*

Par exemples:

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

- Mettre dans un tableau tous les nombres plus petits que 6

```
var  ppetitsQue6 = nombres.Where(n => n  < 6);
```

- Mettre dans un tableau tous les nombres aussi longtemps que le nombre est plus petit que 6

```
var  gaucheDe6 = nombres.TakeWhile(n => n <  6);
```

Expressions *lambda* et *LINQ*

- Une requête *LINQ* est convertie en expressions *lambda* avant d'être exécutée.

- Le programmeur pourrait, s'il le voulait, utiliser seulement des expressions *lambda* pour exécuter une requête. Le problème, c'est que c'est relativement complexe à programmer.