Les délégués

Un délégué est un type représentant une référence à une méthode. Grâce aux délégués, il est possible de spécifier qu'un paramètre de méthode doit être une fonction possédant une liste de paramètres et un type de retour précis. Il est ensuite possible d'appeler cette fonction dans le corps de notre méthode sans la connaître à l'avance.

1. Création

La déclaration d'un délégué utilise le mot-clé `delegate` suivi d'une signature de procédure ou fonction. Le nom spécifié dans cette signature est le nom du type délégué créé.

```
//Création d'un délégué pour une fonction prenant
//2 paramètres de type double et renvoyant un double
public delegate double OperationMathematique(double operande1,
double operande2);
```

Le code ci-dessus crée un nouveau type utilisable dans l'application : `OperationMathematique`

2. Utilisation

Le type `OperationMathematique` est utilisable pour toute variable ou tout paramètre de méthode. Une variable de ce type peut être utilisée comme une méthode, c'est-à-dire que l'on peut lui fournir des paramètres et récupérer sa valeur de retour.

```
private static double
ExecuterOperationMathematique(OperationMathematique
operationAEffectuer, double operande1, double operande2)
{
     return operationAEffectuer(operande1, operande2);
}
```

Il convient ensuite d'appeler la méthode `ExecuterOperationMathematique` en lui fournissant des paramètres convenables. Pour cela, il faut créer une méthode correspondant à la signature du délégué, et passer le nom de la méthode comme valeur du paramètre `operationAEffectuer`.

```
private static double Additionner(double operande1, double
operande2)
{
    return operande1 + operande2; s
}
```

```
double resultatAddition =
ExecuterOperationMathematique(Additionner, 143, 18);
Console.WriteLine(resultatAddition);
```

© Editions ENI – Tous droits réservés – Copie personnelle de Christophe EGEA

3. Expressions lambda

Une expression lambda est une fonction anonyme pouvant être utilisée à tout endroit où une valeur est attendue pour un délégué.

Les expressions lambda utilisent l'opérateur `=>` pour leur déclaration. À gauche de cet opérateur, on trouve une liste de noms de paramètres correspondant aux paramètres que le délégué spécifie. Cette liste de paramètres se trouve entre parenthèses. Il y a deux cas particuliers pour cette liste :

- Lorsque l'expression lambda n'a pas de paramètres, le couple de parenthèses doit quand même apparaître.
- Lorsqu'il n'y a qu'un paramètre, les parenthèses sont facultatives.

La partie droite de l'expression est un bloc d'instructions qui peut, ou non, retourner une valeur. Ce bloc doit être encadré par des accolades, sauf dans le cas où il n'y a qu'une instruction. Dans ce dernier cas, elles sont facultatives.

La syntaxe pour la déclaration d'une expression lambda est :

```
(paramètre1, paramètre2, ...) => { [instructions pouvant utiliser
les valeurs parametre1, parametre2, ...] }
```

L'appel de la méthode `ExecuterOperationMathematique` écrite plus haut pour l'exécution d'une addition peut être réécrit de la manière suivante :

```
OperationMathematique addition = (op1, op2) => { return op1 + op2; };

double resultatAddition = ExecuterOperationMathematique(addition, 143,
18);
Console.WriteLine(resultatAddition);
```


© Editions ENI – Tous droits réservés – Copie personnelle de Christophe EGEA