BOUNATIROU Rodolphe

Centre de readaptation de mulhouse

Fonction flechees

# Introduction

Les fonctions fléchées sont des fonctions générés « à la volée » qui permettent de remplir une action lors d’événement (pour servir de méthode déléguée) par exemple

Elles utilisent entre autre l’opérateur lambda (=>).  
  
Dans ce document je vise à simplifier leur utilisation.

# Action et Func

Action et Func sont des classes dites générique c’est-à-dire qu’elle contienne des types variables placées entre signe d’infériorité et de supériorité.

Une classe générique que vous avez déjà vu est la classe List,  
entre ses signes nous précisons le type de donnée que nous souhaitons stocké

Par exemple List<int> ne stockera que des données compatibles avec le type int.

Ici ces types correspondront dans le cas de Action au type des arguments à passer en paramètre et pour le cas de Func le dernier des élément correspondra au type de retour de la fonction.  
  
La classe **Action** sert à remplir le rôle de **procédure**,  
tandis que la classe **Func** sert à remplir le rôle d’une **fonction**.  
Par conséquent **Action peut être vide** (Action<>) mais **Func devra toujours avoir au moins un type correspondant au type de retour.**

|  |  |
| --- | --- |
| ⚠ | *Attention :*  *Il y’a une limite de 16 arguments (donc 17 types pour Func et 16 pour Action)* |

# Pratique

Pour les ressources nécessaires veuillez vous appuyez sur l’exemple du dépôt git ci-dessous en le clonant :

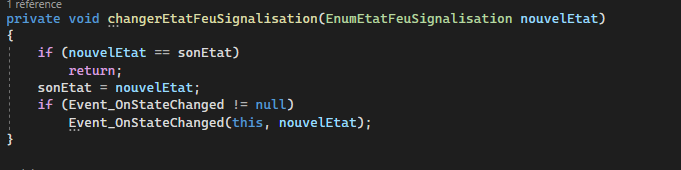
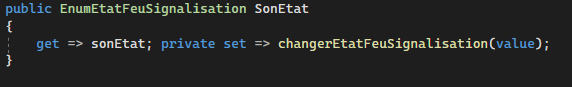
|  |  |
| --- | --- |
| **CMD** | git clone https://github.com/rbounatirou/ExempleFncFleches.git |

Décortiquons la classe métier dans le projet FeuxSignalisations

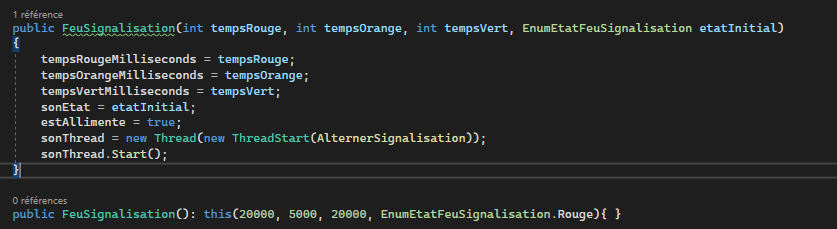
Elle possède un delegate auquel est rattaché un event   
  
Le mutateur privé « SonEtat » (public en accesseur)  
va permettre en plus de changer l’état de la production d’informer les personnes rattaché à l’event du changement d’état



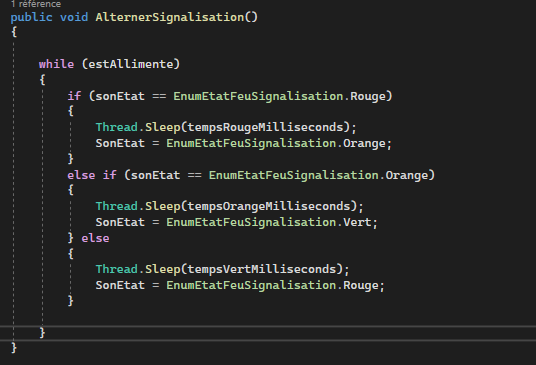
C’est au travers du mutateur que les changement d’état se feront afin d’exécuter le comportement des fonctions remplissant le rôle du délégué pour les abonnée de l’évènement Event\_OnStateChanged.



Le thread qui lance la signalisation quand à lui est initialisé et lancé directement dans le constructeur de la classe



Pour le moment la classe est un thread sans fin.  
Il sera plus tard possible de changer cela en passant estAllimente à l’état false dans une autre méthode



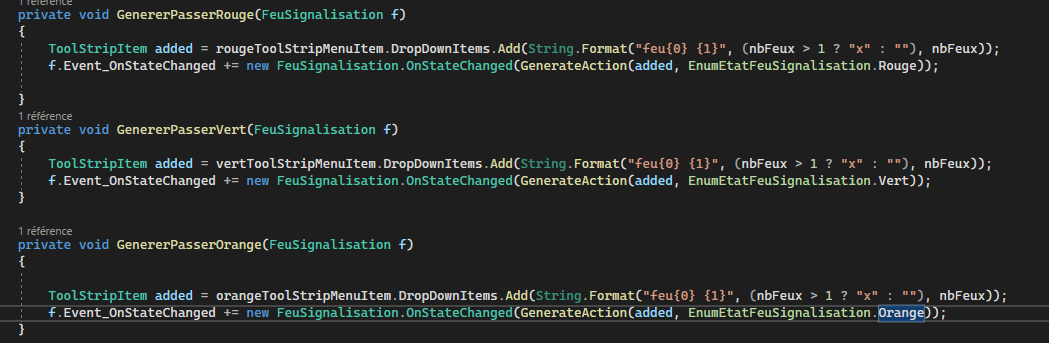
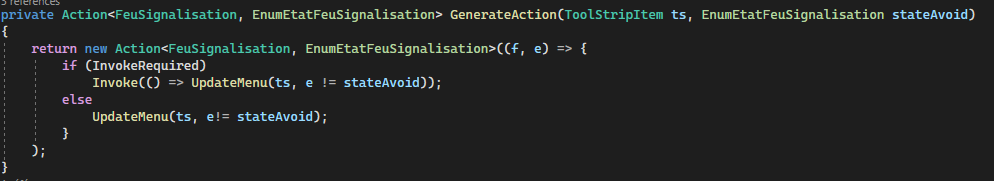
Coté IHM

Les ToolStripItem sont générés dynamiquement lors de la création de feux.  
(et un UserControl est ajouté)  
  
Les ToolStripItem ainsi généré sont rattaché à l’evenement Event\_OnStateChanged de la classe métier ainsi lors du changement d’état du feu de signalisation qui doit leur être attaché, sa propriété enable change en fonction de l’état de ce dernier.

En théorie pour faire cela sans fonction fléché on pourrait mettre à chaque FeuSignalisation un attribut unique permettant de l’identifier, et rattacher via la propriété Tag du menu ToolStripItem le FeuSignalisation correspondant.  
  
Ensuite en abonnant l’IHM à l’event de chacun des feu,   
lors du changement d’état d’un feu (sender), on regarderait l’ensemble des ToolStripItem pour actualiser ceux dont le Tag correspond au Sender.

Bref ce système force un parcours, la création d’un identifiant unique dans la classe métier,  
et par certain aspect il complexifie la programmation des composants.  
  
Afin de limiter toutes ces adaptations on va utiliser les fonctions fléchées

Grace à ces dernières on générera des Action pour remplir le rôle de procédure déléguées abonnée à l’événement



Comme la procédure est générée à la volée. Elle a accès à d’autres paramètres que ceux qu’impose le délégué. **C’est un des grands intérêts des fonctions fléchées, il permet d’avoir accès à des données normalement inaccessible si on passait par une méthode associée à l’évènement déléguée.**  
  
Le code est donc simplifiée d’une certaine façon (même si cela demande un peu d’apprentissage supplémentaire). De plus dans notre cas de figure il va limiter le parcours pour retrouver le Tag… etc.