C# : délégué et évènement

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en Programmation par contrainte (IA)
Ingénieur en Génie logiciel

elmouelhi.achref@gmail.com

Plan

Delegate

Les délégués en C#

- Concept inspiré par les pointeurs de fonction en C et C++.
- Type référence utilisé pour encapsuler une méthode anonyme ou nommée.
- Déclaré avec le mot clé `delegate`
- Défini une signature de méthode
- Toute méthode respectant cette signature peut être appelée par le biais du délégué

Delegate

Considérons la méthode suivante

```
public static void DireBonjour(string nom)
{
    Console.WriteLine($"Bonjour␣{nom}");
}
```

Pour exécuter cette méthode

```
static void Main(string[] args)
{
    DireBonjour("Wick"); // affiche Bonjour Wick
    Console.ReadLine();
}
```

Delegate

Considérons la méthode suivante

```
public static void DireBonjour(string nom)
{
    Console.WriteLine($"Bonjour␣{nom}");
}
```

Pour exécuter cette méthode

```
static void Main(string[] args)
{
    DireBonjour("Wick"); // affiche Bonjour Wick
    Console.ReadLine();
}
```

Question

Comment confier cette mission à un délégué ?

Delegate

Créer un délégué

```
public delegate void PremierDelegate(string s);
```

Déclarer un délégué

```
PremierDelegate d;
```

Créer une instance du délégué

```
d = new PremierDelegate(DireBonjour);
```

Appeler la méthode `DireBonjour` à travers le délégué

```
d("Bob"); // affiche Bonjour Bob
```

Delegate

Créer un délégué

```
public delegate void PremierDelegate(string s);
```

Déclarer un délégué

```
PremierDelegate d;
```

Créer une instance du délégué

```
d = new PremierDelegate(DireBonjour);
```

Appeler la méthode `DireBonjour` à travers le délégué

```
d("Bob"); // affiche Bonjour Bob
```

On peut aussi faire la même chose de deux façons différentes

Delegate

On peut faire aussi

```
PremierDelegate d;

d = DireBonjour;

d("Bob");
```

Delegate

On peut faire aussi

```
PremierDelegate d;

d = DireBonjour;

d("Bob");
```

Ou encore en utilisant les méthodes anonymes

```
PremierDelegate d;

d = delegate (string nom)
{
    DireBonjour(nom);
};

d("Bob");
```

Exemple avec plusieurs méthodes

```
public static void Somme (int a, int b)
{
    Console.WriteLine(a + b);
}
public static void Produit(int a, int b)
{
    Console.WriteLine(a * b);
}
public static void Soustraction(int a, int b)
{
    Console.WriteLine(a - b);
}
public static void Division(int a, int b)
{
    Console.WriteLine(a / b);
}
```

Delegate

Déclarer un délégué

```
public delegate  void Calcul(int x, int y);
```

Delegate

Déclarer un délégué

```
public delegate  void Calcul(int x, int y);
```

Utiliser le délégué

```
Calcul  calcul;

calcul = Somme;
calcul(7, 5);

calcul = Produit;
calcul(7, 5);

calcul = Division;
calcul(7, 5);

calcul = Soustraction;
calcul(7, 5);
```

Trop long ?

Delegate

Solution : utiliser le multicast

```
Calcul calcul;

calcul = Somme;
calcul = calcul + Produit + Division + Soustraction;

calcul(7, 5);
```

Delegate

Solution : utiliser le multicast

```
Calcul calcul;

calcul = Somme;
calcul = calcul + Produit + Division + Soustraction;

calcul(7, 5);
```

Pour connaître le nombre de méthodes abonnées à notre délégués

```
Console.WriteLine(calcul.GetInvocationList().Length)
   ;
```

Event

Les évènements en C#

- L'une des utilisations les plus importantes des délégués est la programmation d'évènements.
- Un évènement est déclaré avec le mot clé `event`
- Les applications à interfaces graphiques sont assez associées aux concepts de programmation évènementielle (`click`, `input`, `focus`...)

Event

Les évènements en C#

- L'une des utilisations les plus importantes des délégués est la programmation d'évènements.
- Un évènement est déclaré avec le mot clé `event`
- Les applications à interfaces graphiques sont assez associées aux concepts de programmation évènementielle (`click`, `input`, `focus`...)

Nomenclature

- L'objet qui déclenche l'évènement est appelé éditeur
- Celui qui capture l'évènement et y réponds est appelé abonné

Event

Déclarer un évènement

```
public static event Calcul MonEvent;
```

Event

Déclarer un évènement

```
public static event Calcul MonEvent;
```

Déclarer un déclencheur

```
public static void MonTrigger()
{
    MonEvent(7, 5);
}
```

Event

Déclarer un évènement

```
public static event Calcul MonEvent;
```

Déclarer un déclencheur

```
public static void MonTrigger()
{
    MonEvent(7, 5);
}
```

C'est quoi le but ?

Exécuter les méthodes d'un délégué lorsqu'un évènement se déclenche (une méthode ici qui sera appelée).

Event

Abonner des méthodes à cet évènement

```
MonEvent += new Calcul(Somme);
MonEvent += new Calcul(Produit);
```

Ou aussi

```
MonEvent += Somme;
MonEvent += Produit;
```

Event

Déclencher l'évènement

```
MonTrigger();
```

Event

Déclencher l'évènement

```
MonTrigger();
```

Résultat

- Les méthodes abonnées à cet évènement sont exécutées
- `12 35` seront affichés
- On n'a pas exécuté les méthodes à travers le délégué