Ecole Nationale des Sciences Appliquées de Safi

A.U:2019-2020

Développement en c#.NET

Ing. Meryem OUARRACHI

Plan du module

Langage C#

- L'environnement .Net
- Initiation à la programmation C#
- Programmation Orienté Objet C#

Programmation avancée en .Net ,C#

- Programmation distribuée
- Gestion de base de donnée
- Application WPF

Les Threads

Introduction

❑ Définition de Processus:

-C’est une instance d'exécution d'un programme.

-S’exécutent indépendamment les uns des autres, dans des espaces de mémoire distincts.

Exemple: L'exécution de plusieurs instances du programme « Calculateur ». Chacune des instances sont appelés comme un processus.

Introduction

❑ Définition de thread:

-toutes les opérations étaient effectuées, jusqu'à mnt, sur un seul processus (Programme principal).

- Si on lance une tâche lourde ,on ne peut rien faire d'autre qu'attendre que le processus soit terminé.

-Les Threads sont devenus indispensables aux développeurs souhaitant développer des applications performantes.

Introduction

❑ Définition de thread:

- Permet d'effectuer des actions en parallèles dans une machine afin d'en accélérer le traitement.

- Il est qualifié comme un « processus léger ».

Notion de base

- Espace de noms System.Threading: fournit les outils permettant la programmation multithread .
- La classe Thread nous permet de créer simplement un objet qui va exécuter le contenu d'une méthode.
- Pour la gestion de multi-threading en .Net :

✓Utiliser un delegate ThreadStart.

Notion de base

- **Membres d'instances**

Propriétés	Description
IsAlive	Indique si le thread est en cours d'exécution ou non
ManagedThreadId	Retourne un nombre permettant d'identifier le thread
Name	Permet de connaitre ou d'indiquer le nom du thread
ThreadState	Retourne une des valeurs de l'énumération ThreadState reflétant l'état du thread.
Propriété statique	
CurrentThread	Retourne le thread en cours d'exécution

Notion de base

- Enumération ThreadState

Valeur	Description
`Aborted`	Le thread est interrompu
`AbortRequested`	Le thread à reçu une demande d'interruption mais l'exception `ThreadAbortException` n'a pas encore été levée.
`Background`	Le thread tourne en arrière plan.
`Running`	Le thread est en cours d'exécution
`Stopped`	Le thread est arrêté.
`StopRequested`	Le thread a reçu une demande d'arrêt
`Unstarted`	Le thread a été crée mais n'a pas encore était démarré.
`WaitSleepJoin`	Le thread est bloqué. (Par exemple par la méthode `Join`)

Lancer un thread

- Pour une méthode sans paramètres

```
//la méthode qui va étre exécuter par le thread
ThreadStart    delegate = new ThreadStart(fonction);
Thread    t = new Thread(delegate); //Créer le thread
 t.Start();  //lancer le thread
```

ou

```
Thread    t = new Thread(fonction);
t.Start();
```

Lancer un thread

❑ Pour une méthode sans paramètres

- Exemple1:

```
• static void affichage()
  { Console.WriteLine(Thread.CurrentThread.ManagedThreadId
    + "  " + Thread.CurrentThread.ThreadState); }

• static void Main(string[] args)
  {  ThreadStart d = new ThreadStart(affichage);
     Thread t = new Thread(d);
      t.Start(); }}
```

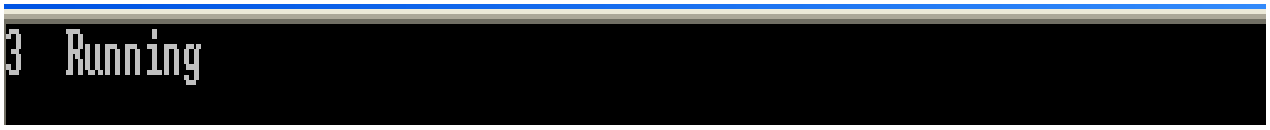
Lancer un thread

❑ Pour une méthode sans paramètres

- Exemple1:

- la fonction affichage, qui affiche le numéro du thread courant et son état.
- délégué de type ThreadStart qui pointe sur notre fonction affichage.
- Création du thread prenant notre délégué en paramètre.

-Résultat d'exécution :



❑ Exemple2

```
static void affichage()
{for (int i = 0; i < 5; ++i)
    { Console.WriteLine("Thread n° {0} - Etat : {1}",
      Thread.CurrentThread.ManagedThreadId, Thread.CurrentThread.ThreadState); } }

static void affichage2()
{Console.WriteLine("Second Thread est exécuté");
 Console.WriteLine("Second Thread se termine");
}
static void Main(string[] args)
{Thread t1 = new Thread(affichage);
 Thread t2 = new Thread(affichage2);
  t1.Start();
  t2.Start();
  Console.ReadKey();
}
```

❑ Exemple2 :

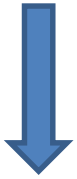
Résultat d'exécution

```
Thread n° 3 - Etat : Running
Second Thread est exécuté
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Second Thread se termine
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
```

```
Second Thread est exécuté
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Second Thread se termine
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
```

```
Second Thread est exécuté
Second Thread se termine
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
```

les threads sont imprévisibles



La notion de priorité

Lancer un thread

❑ Fonctionnement de thread

-L’appel de la méthode start provoque la mise en état d’ exécutable de nouveau thread;et si d’autres threads sont déjà dans cet état, le nouveau thread passe en fil d’attente des exécutables jusqu’à ce qu’arrive son tour et devenir le thread courant.

-La notion de priorité:Chaque thread a une priorité.

-Lorsque plusieurs threads sont démarrés,celui qui a la priorité la plus forte s’exécute en premier ainsi de suite

Lancer un thread

❑ Les threads à méthode paramétrée

- On doit utiliser un délégué de type ParameterizedThreadStart au lieu de ThreadStart. Ou utiliser directement le Thread.

- Le paramètre doit être obligatoirement de type Object.

```
static void affichage3(Object o)
{
    Console.WriteLine("{0}", (string)o);
}
static void Main(string[] args)
{
    ParameterizedThreadStart delegue_parametres = new
ParameterizedThreadStart(affichage3);
    Thread monThread3 = new Thread(delegue_parametres);
    monThread3.Start("Coucou");
    Console.Read();
}
```

Lancer un thread

❑ Les threads à méthode paramétrée

-On ne peut passer qu'un seul argument à un Thread

```
class parametres
{ public string x;
  public string y;
  public parametres(string x, string y) { this.x = x; this.y = y; }
}
```

```
static void affichage3(Object o)
{ parametres prm = (parametres)o;
  Console.WriteLine(prm.x+prm.y);
}

static void Main(string[] args)
{ ParameterizedThreadStart delegue_parametres = new
  ParameterizedThreadStart(affichage3);
  Thread monThread3 = new Thread(delegue_parametres);
  parametres p = new parametres("mmm", "oooo");
  monThread3.Start(p);
  Console.Read();    }
```

La méthode Join

-Join va bloquer le thread courant, jusqu'à ce que le thread ciblé ait fini son exécution.

```
static void Main(string[] args)
{Thread t1 = new Thread(affichage);
 Thread t2 = new Thread(affichage2);
  t1.Start();
  t1.Join();
  t2.Start();
  Console.ReadKey();
}
```

```
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Thread n° 3 - Etat : Running
Second Thread est exécuté
Second Thread se termine
```

Gérer les arrêts des threads

- Pour arrêter un thread, il faut utiliser la méthode Abort.

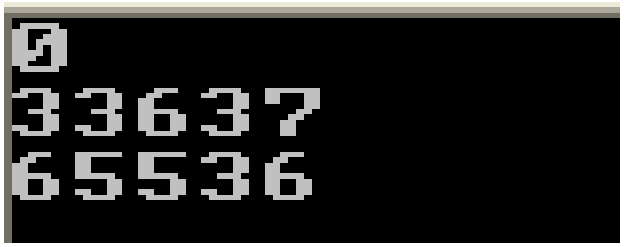
```
static void maListe(Object o)
{
    List<string> liste = o as List<string>;
    for (int i = 0; i < 10000000; i++)
    {
        liste.Add("Test");
    }
}
static void Main(string[] args)
{
    List<string> liste = new List<string>();
    ParameterizedThreadStart delegue_parametres2 = new
ParameterizedThreadStart(maListe);
    Thread monThread4 = new Thread(delegue_parametres2);
    monThread4.Start(liste);
    Console.WriteLine("{0}", liste.Count);
    Console.WriteLine("{0}", liste.Count);
    monThread4.Abort();
    Console.WriteLine("{0}", liste.Count);

    Console.Read();
}
```

Gérer les arrêts des threads

- Résultat sans Abort :

- Résulat avec Abort



Gérer les arrêts des threads

- Suspend: Suspend le thread ou, s'il est déjà suspendu, n'a aucun effet.

-Sleep: Suspend le thread en cours pendant une durée spécifiée.

-Thread.sleep(1000): arrêter le programme pour une seconde

-Resume: Reprend un thread qui a été suspendu.

Synchronisation

-Dans le cas ou des threads partagent les mêmes ressources ,il faut appliquer une règle de synchronisation: indiquer qu'une donnée ne peut pas être modifiée par deux threads en même temps.

-NET Framework propose une solution pour verrouiller les données : Le mot clé lock

Synchronisation

```
private static Object _lock = new Object();

 private static void Initialiser()
 {   lock (_lock)
     {
        _nominateur = 0;
        _denominateur = 0;

     }
 }

 private static void Diviser()
 {lock (_lock)
  {    if (_denominateur == 0)
           Console.WriteLine("Division par 0");

       else
       {
    Console.WriteLine("{0} / {1} = {2}", _nominateur, _denominateur, _nominateur / (double)_denominateur);
       }
    }

  }
```