Threads

Définition

- wikipédia

Utilité

- Concevoir une application en terme de tâches coopérantes
- Ne pas « bloquer » l'IHM lors d'une opération qui peut prendre du temps
  - un thread qui gère l'IHM
  - un thread qui est chargé de la tâche

Mais attention aux opérations inter threads

dans la mémoire partagée

Un exemple

La recherche de fichier dans une arborescence

Exemple de recherche

Etape 1

- Mise en place de l'IHM
  - TextBox
  - Bouton
  - ListView
  - FolderBrowserDialog

Etape 2: La recherche dans une arborescence

```
private void Parcours(string repertoire)
    {
       string[] dirs = null;
       string[] files = null;
       try
       {
          dirs = Directory.GetDirectories(repertoire);
          files = Directory.GetFiles(repertoire);
       }
       catch (UnauthorizedAccessException)
       {
          return ; // cas des répertoires à accès interdit
       }
       foreach (string nomAbsolu in files)
       {
          string nom = nomAbsolu.Substring(nomAbsolu.LastIndexOf('\\')+1);
          if (nom== textBoxFichier.Text)
          {
             // une occurrence du fichier trouvé
             MettreAJourListView(nomAbsolu);
          }
       }
       foreach (string d in dirs)
       {
          Parcours(d);
       }
    }
```

Etape 3 : test avec une seul thread

Que constatez vous?

interface indisponible durant la recherche

=> Aucune action utilisateur possible

Etape 4:

Création d'un thread chargé de la recherche:

Création et lancement d'un thread:

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

Où **NouveauParcours** est une méthode:

```
private void NouveauParcours()
    {
        Parcours(textBoxRepertoire.Text);
        FinRecherche();
    }
```

Etape 5 : problème constaté

Cross thread operation not valid: Control "XXXXXXXXXX" accessed from a thread other …

Explication:

La gestion de l'IHM et le parcours s'exécutent de façon indépendante, dans **deux fils de calcul distincts**.

Or, pour éviter les conflits d'accès:

les contrôles prédéfinis sont protégés contre les accès provenant d'un autre thread que celui qui les a créés.

Etape 6 : résoudre le problème

- Une solution possible, **mais mauvaise**, consiste à **désactiver la sécurité** en modifiant la propriété du contrôle en cause par

```
CheckForIllegalCrossThreadCalls = false;
```

Etape 6 : la bonne solution

Faire exécuter toute méthode qui modifie l'interface par le thread chargé de l'interface

Comment:

Par l'usage :

- d'un délégué
- Et de la méthode Invoke

Mise en place du délégué:

```
public delegate void MettreAJourListViewCallBack(string s);
private void MettreAJourListView(string s)
{
    if (this.InvokeRequired)
     {// code si thread non créateur
        MettreAJourListViewCallBack d = new
            MettreAJourListViewCallBack(MettreAJourListView);
        this.Invoke(d, new object[] { s });
    }
   else
    { // code si thread créateur
            ListViewItem item = new ListViewItem(s);
            listViewRepertoire.Items.Add(item);
    }
}
```

Etape 7 : arrêt d'un thread

Comment arrêter prématurément et proprement un thread ?

Par exemple, l'utilisateur souhaite interrompre la recherche

Voir :

http://fr.wikibooks.org/wiki/Programmation_C_sharp/Threads_et_synchronisation

Arrêt la bonne solution

Le thread est programmé pour s'arrêter lorsqu'une condition est vraie:

Utilisation d'une boucle while (continuer) ou

Dans le cas récursif d'un if (continuer)

Mise en place arrêt prématuré

Définition d’un attribut qui sera partagé par les deux threads:

//Le mot clé volatile alerte le compilateur que plusieurs threads accéderont à l'attribut stop,

// et par conséquent, qu'il ne doit pas émettre d'hypothèses d'optimisation à propos de l'état de ce membre

private volatile bool stop = false;

Utilisation de l'attribut

- Thread interface : arrêt souhaité

  $\Rightarrow$ **stop = true;**

- Thread recherche, modification de la fonction récursive parcours par l'ajout en début de :

```
private void Parcours(string repertoire)
    {
        if (stop) return;
        string[] dirs = null;
        …
```

Etape 8: Prise en compte par l'interface de la fin de la recherche

Reste à écrire « **FinRecherche** »:

```
private void NouveauParcours()
    {
        Parcours(textBoxRepertoire.Text);
        FinRecherche();
    }
```

Qui modifie l'interface, qui est appelé par le thread recherche …

=> Nécessité d'un autre délégué, …