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 Sockets

Les threads avec les Windows Forms

Soit le programme suivant qui crée manuellement un thread qui modifie la valeur d'un TextBox:

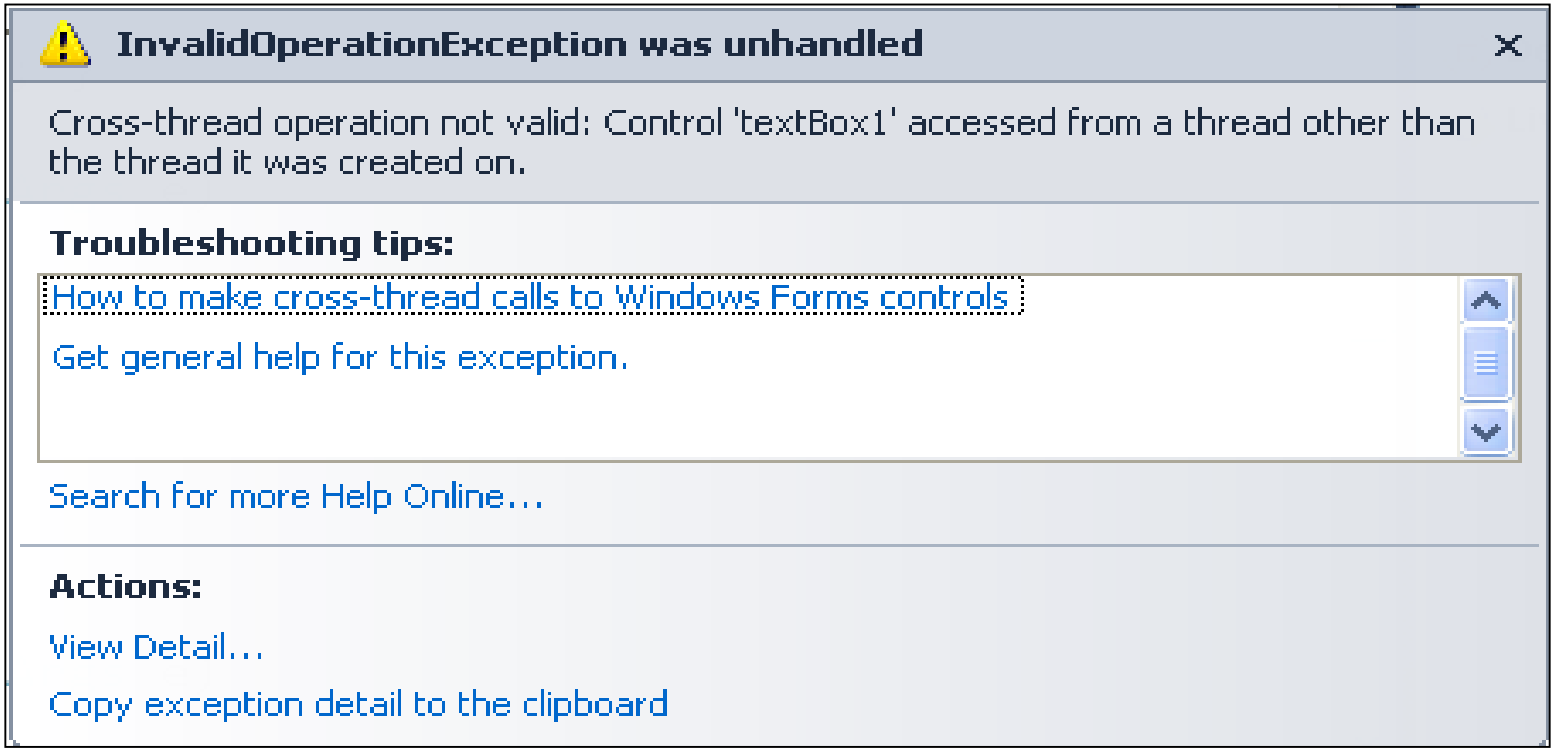
```
public void test ( ) {
   textBox1.Text = "salut";
                    }
```

Les threads avec les Windows Forms

```
private void button1_Click(object sender, EventArgs e)
    {
        Thread t = new Thread(test);
        t.Start();
     }
```

Les threads avec les Windows Forms

- L'exécution de ce programme déclenche cet erreur



- Càd: le contrôle TextBox a fait l'objet d'un accès à partir d'un thread autre que celui sur lequel il a été créé.

Les threads avec les Windows Forms

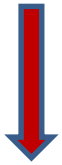
- **Justification:**

-L'accès aux contrôles Windows Forms n'est pas fondamentalement thread-safe.

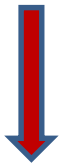
-Une méthode est appelée thread-safe lorsqu'elle peut fonctionner correctement et sans risque lorsqu'elle est exécutée par plusieurs threads simultanément(multithread).

Les threads avec les Windows Forms

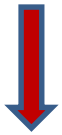
-Les objets Windows Forms, ne sont pas **thread-safe** et donc .NET limite leurs possibilités en multi-thread.



-Cela signifie qu'on pourra pas accéder à leurs propriétés qu'en lecture seule si on ne fait pas parti du même thread.



-Trouver un moyen pour assurer que l'accès à nos contrôles est exécuté de manière thread-safe.



La méthode **Invoke**

Les threads avec les Windows Forms

-La méthode Invoke: sert à demander à l'autre thread de s'occuper d'une action dans un moment libre.

-public Object Invoke(Delegate method) : Exécute le délégué spécifié sur le thread qui crée le contrôle.

-public Object Invoke(Delegate method, Object[] args): Exécute le délégué spécifié sur le thread qui crée le contrôle, avec la liste d'arguments spécifiée.

Les threads avec les Windows Forms

Exemple1:

- public void test1() { textBox1.Text = "salut"; }
- public delegate void methode();
- public methode myDelegate;

```
private void button1_Click(object sender, EventArgs e)
  {    myDelegate = new methode(test1);
       textBox1.Invoke(myDelegate);
  }
```

Les threads avec les Windows Forms

Exemple2:

- public void test2(string t) { textBox1.Text = t; }
- public delegate void methode(string s);
- public methode myDelegate;

```
private void button1_Click(object sender, EventArgs e)
   {    myDelegate = new meth(test2);
        string myString = "Salut";
        Object[] k = { myString };
        textBox1.Invoke(myDelegate, k);      }
```

Introduction au socket

- L'API "sockets" est une bibliothèque de Classes de communication entre machines sur TCP/IP contenu dans le namespace System.Net.Sockets
- Le mode connecté(Stream socket) => protocole TCP :Le protocole établit une connexion virtuelle et se charge alors de maintenir l'intégrité de la communication et de gérer les erreurs de transmission.
- Le mode non connecté(Datagram socket) => protocole UDP: mode non connecté,moins fiable et plus rapide

Le Port

•Il est possible d'avoir plusieurs services (programmes qui fonctionnent en permanence) sur une même machine.

• Pour les différencier, nous utilisons un numéro qui est appelé ***numéro de port***

- Les ports numérotés de 0 à 511 sont les plus connus :ex. (FTP port 21),(HTTP port80).

- De 512 à 1023, on trouve les services Unix.

- Au delà, (1024 ...) ce sont les ports "utilisateurs" disponibles pour placer un service applicatif quelconque.

Principe de fonctionnement de socket

1. Serveur: enregistrer le service.
2. Serveur : attente de connexion.
3. Client : établir la connexion.
4. Utilisation du socket: le client et le serveur peuvent échanger les données

La classe Socket

•Constructeur de la classe Socket :

Socket(AddressFamily,SocketType, ProtocolType);

- *AddressFamily :* un type d'adresse, en general InterNetwork (adresses IPv4) ou InterNetworkV6.

-*SocketType :* le type de socket (Dgram, Stream…).

-*ProtocolType :* le protocole (Icmp, IP, Tcp, Udp...).

Methode	Description
void Bind(EndPoint localEP);	sur le serveur qui associe un socket avec ce que l'on appelle un end-point (formé d'une adresse IP et d'un numéro de port).
void Listen(int n);	Place le socket en attente de connexions. N = la taille de la file de connexions qui pourraient être en attente de traitement. (sur le serveur)
Socket Accept();	Renvoie un socket à utiliser pour la communication avec le client qui vient de semanifester. Cette opération est effectuée sur le serveur.
void Connect(EndPoint remoteEP);	Client, Établit une liaison avec un ordinateur distant. Le serveur doit avoir exécuté Bind avant que le client n'exécute Connect.
int Send(byte[]b);	Envoie des caractères au correspondant. Les données proviennent du tableau b d'octets
int Receive(byte[] b);	Reçoit des caractères provenant du correspondant

La classe Socket

-Convertir string en byte[ ]:

ASCIIEncoding.Default.GetBytes(string)

-Convertir byte [ ] en string

ASCIIEncoding.Default.GetString(byte [ ])

Socket TCP

- **Coté serveur:**

```
Socket   sock;

Sock= new
    Socket(AddressFamily.InterNetwork,SocketType.Stream,
            ProtocoleType.Tcp);


sock.Bind(new IPEndPoint(IPAddress.Parse("127.0.0.1"),123));


sock.Listen(1);

Socket   sockServeur = sock.Accept( );
```

Socket TCP

- **Coté client:**

```
Socket   sockclient;

sockclient= new
    Socket(AddressFamily.InterNetwork,SocketType.Stream,
            ProtocoleType.Tcp);


sockclient.Connect(new
    IPEndPoint(IPAddress.Parse("127.0.0.1"),123));
```

Socket TCP

- **Le serveur encode et envoie un message :**

String message=‘’réponse de serveur’’;

byte[ ] buf= ASCIIEncoding.Default.GetBytes(message);

sockServeur.Send(buf);

- **Le client reçoit et décode le message :**

byte[ ] buf=new byte [50] ;

sockClient.Receive(buf);

String msg= ASCIIEncoding.Default.GetString(buf)

Socket UDP

- **Coté serveur:**

```
Socket   sockServeur;


 sockServeur= new
    Socket(AddressFamily.InterNetwork,SocketType.
    Dgram,ProtocoleType.Udp);


sockServeur.Bind(new
    IPEndPoint(IPAddress.Parse("127.0.0.1"),123));
```

Soket UDP

- **Coté client:**

```
Socket   sockclient;

sockclient= new
    Socket(AddressFamily.InterNetwork,SocketType. Dgram,
            ProtocoleType.Udp);


IPEndPoint x = new IPEndPoint(IPAddress.Parse("127.0.0.1"),
    123);

sockclient.Connect(x);
```

Socket UDP

- **Le serveur encode et envoie un message :**

String message=‘’réponse de serveur’’;

byte[ ] buf= ASCIIEncoding.Default.GetBytes(message);

sockServeur.Send(buf);

- **Le client reçoit et décode le message :**

byte[ ] buf=new byte [50] ;

EndPoint y = (EndPoint)(x);

sockClient.**ReceiveFrom**(buf, ref y);

String msg= ASCIIEncoding.Default.GetString(buf)

Les opérations asynchrones

-Un socket asynchrone n'interrompt pas l'application en attendant des opérations de réseau soit terminée exemple: *Beginconnect,*

BeginReceiveFrom,BeginSend…

-Elles utilisent un délégué de type **AsyncCallback** qui est automatiquement exécutée lorsque l'opération lancée de manière asynchrone se termine.

-**IAsyncResult.AsyncState**:Obtient les résultats d'une opération asynchrone.

Les opérations asynchrones

Exemple:

Public BeginReceiveFrom(byte[] buffer,

int offset, //Position de base zéro dans le paramètre *buffer* à laquelle stocker les données

int size,

SocketFlags socketFlags,

ref EndPoint remoteEP, *//la source des données.*

AsyncCallback callback,

Object state //Objet contenant les informations d'état de cette demande(buffer)

Les opérations asynchrones

Exemple:

```
private void Form2_Load(object sender, EventArgs e)
{   byte[] buffer=new byte[1500];;
    EndPoint  epRemote;
    Socket sck = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
        ProtocolType.Udp);
    epRemote = new IPEndPoint(IPAddress.Parse("127.0.0.1"),123);
    sck.Connect(epRemote);

    sck.BeginReceiveFrom(buffer, 0, buffer.Length, SocketFlags.None,
      ref epRemote, new AsyncCallback(MessageCallBack), buffer);
 }
private void MessageCallBack(IAsyncResult AResult)
{   byte[] ReceivedData = new byte[1500];
    ReceivedData = (byte[])AResult.AsyncState;
    ASCIIEncoding aEncoding = new ASCIIEncoding();
    string receivedMessage = aEncoding.GetString(ReceivedData);
     //...
}
```