COURS WINDEV NUMERO 5

14/02/2015

Études des Sockets, communication répartie

Rappels de notions réseaux, Tcp/ip, Ports

Cours Windev numéro 5

VERSION 19

Grâce à ce nouveau Tp nous allons rentrer dans le monde merveilleux de la communication distante via réseau. Cette jolie introduction pour vous faire comprendre que ce support va vous apprendre à faire discuter 2 (ou plusieurs) ordinateurs entre eux. Nous allons employer les Sockets et les threads.

Commençons par définir ces 2 termes :

Les Sockets : (Tous ceux qui pensent à une chaussette sont virés) Une Socket est définie comme une extrémité d'une communication.

Une paire de processus (ou de Threads) communiquant sur un réseau emploie une paire de sockets, une pour chaque processus. Une socket est constituée d'une adresse IP concaténée à un numéro de port. En général les sockets utilisent une architecture Client/Serveur. Le serveur attend des requêtes entrantes du client en écoutant un port spécifique. Dès réception d'une requête, il accepte une connexion de la socket du client. Les serveurs implémentant des services particuliers (par exemple, telnet, ftp, mail, http), écoutent des ports bien connus (Telnet écoute le port 23, un serveur ftp le port 21, un serveur web [Http] le port 80). Les ports inférieurs à 1024 sont considérés comme connus et sont utilisables pour les services standards. Lorsqu'un thread client commence une requête de connexion, il se voit assigner un port par la machine hôte. Ce port est un nombre supérieur à 1024.

Par exemple, lorsqu'un client de l'hôte X d'adresse Ip 192.168.5.20 souhaite établir une connexion avec un serveur Web (qui écoute le port 80) d'adresse 192.168.6.10, l'hôte X peut se voir affecter le port 1625. La connexion est constituée d'une paire de sockets : (192.168.5.20 : 1625) sur l'hôte X et (192.168.6.10 : 80) sur le serveur Web.

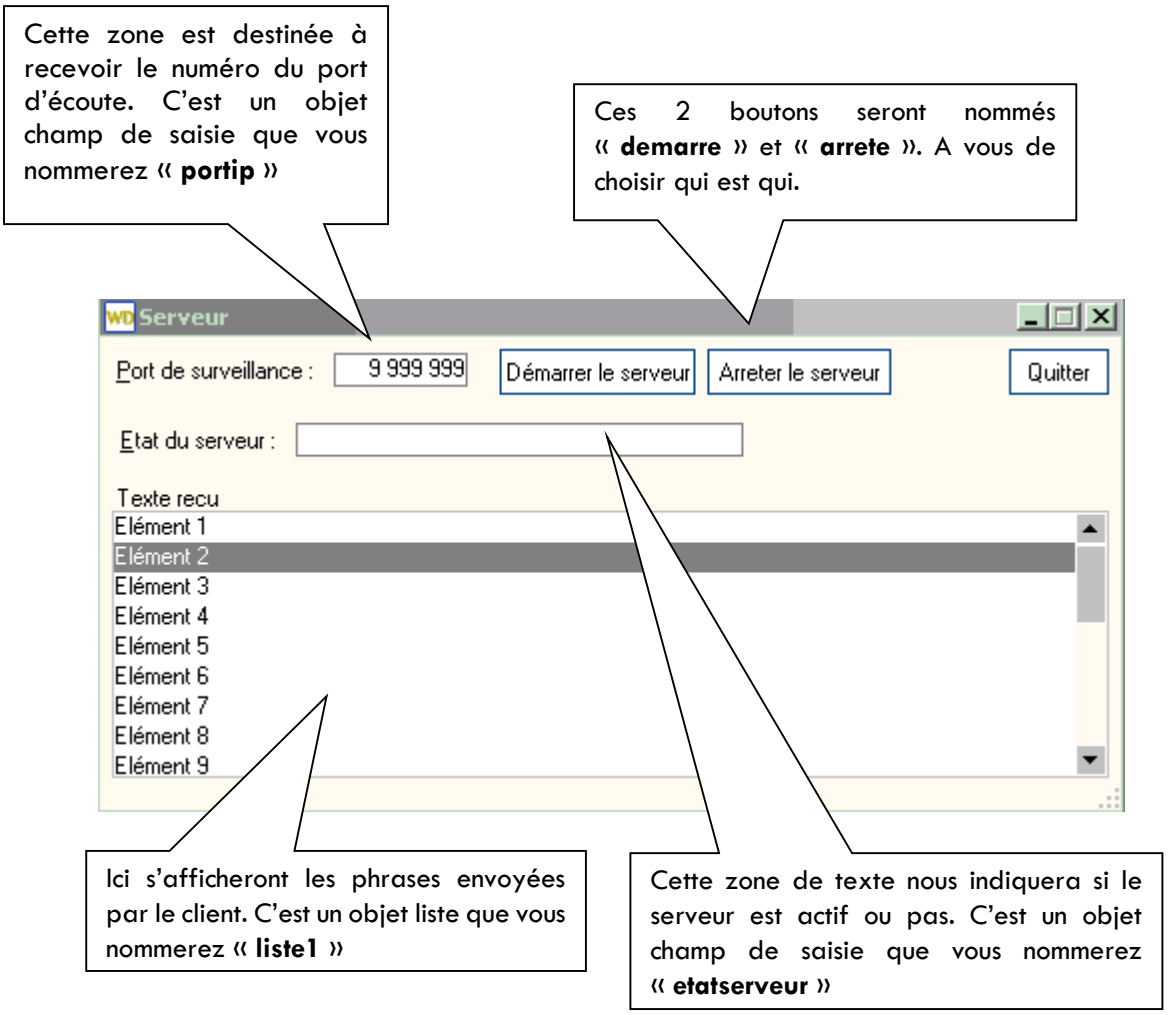
Un thread peut être perçu comme un flot de contrôle à l'intérieur d'un processus. Dans notre cas il joue le rôle d'interface entre les 2 ordinateurs, interceptant les données transmises.

Notre exercice consistera à créer 2 exécutables, un client et un serveur. Le serveur sera en attente de réception de message et le client essayera de se connecter au serveur et de lui envoyer des messages. Pour cela il nous faudra 2 projets un pour le client, un pour le serveur

Je vous rappelle que plus on avance au fil des supports moins je détaille les fonctionnalités que je considère comme devant être acquises. Donc si à ce stade vous éprouvez des difficultés reprenez les cours précédents. Nous montons en gamme en terme d'expertise donc il est probable que vous éprouverez certaines difficultés sur ce support. Restez motivé et recommencez le autant de fois que nécessaire. Le but étant de le réaliser sans avoir recours à ce support.

PARTIE SERVEUR

Vous allez commencer par créer un nouveau projet nommé « **Serveur** ». Nous ne travaillerons pas sur des fichiers donc faites en sorte de n'utiliser aucune analyse. Nous n'aurons besoin que d'une fenêtre que vous nommerez « **Depart** » et qui sera la première fenêtre du projet. Faites en sorte qu'elle ressemble à celle-ci :



Analysons le fonctionnement du serveur :

Il faut lui donner un port a écouter.

Il faut lancer la boucle d'écoute (boucle infinie) et faire en sorte que les événements d'entrés (demande de connexion…) soient traités par des threads.

Une fois la connexion acceptée le texte recu sera inscrit dans la liste déroulante.

Retroussons nos manches et en avant pour le codage. Nous allons commencer par créer une procedure d'attente (la boucle d'écoute). Pour cela créez une procédure globale nommée "**attente**"

Voici le code que nous allons analyser.

```
PROCEDURE attente()

BOUCLE // début de la boucle

        SI SocketAttendConnexion("serveur") ALORS // si une demande de connexion est en attente

        canal est une chaîne

        canal=SocketAccepte("serveur") //Cette fonction permet de créer le canal de communication entre la socket serveur et la socket cliente.

        ThreadExécute("threadcnx",threadNormal,"affichemes",canal) //Lance l'exécution d'un "thread"

        Multitache(-30) //La fonction Multitache avec un negatif suspend l'application

        FIN

FIN
```

Vous pouvez constater que nous nous trouvons devant une boucle sans fin ou en attente dite active. A l'intérieur de cette boucle une fonction WinDev (SocketAttendConnexion), est chargée de verifier si des demandes de connexion se produisent. Comme parametre cette fonction prend un argument qui est le nom de la socket ici "serveur". Ne vous inquietez pas pour l'instant, vous allez voir ou nous allons définir la socket nommée "serveur". Pour l'instant l'essentiel est de comprendre le principe de la boucle d'attente active. Donc si une demande de connexion se produit pour la socket "serveur" on l'accepte en créant un canal de communication. Vous pouvez considérer ce canal comme un tunel ou les données vont transiter.

La ligne : ThreadExécute("threadcnx",threadNormal,"affichemes",canal), est chargée de faire en sorte que le code de la fonction affichemes soit executée comme un Thread normal appelé "threadcnx" utilisant le parametre "canal")

Multitache(-30) : L'exécution de l'application est suspendue durant <Temporisation> 100ème de seconde. D'autres traitements peuvent être exécutés durant cette période de temps (ré-affichage ou exécution d'un code de clic par exemple). Dans notre cas la boucle est gelée pour permettre aux threads de s'exécuter durant leurs quantums.

Donc : la socket s'appelle « serveur », le canal crée s'appelle canal, le thread gestionnaire se nomme « threadcnx ».

J'admets qu'au premier abord cela puisse vous sembler complexe, alors que c'est extrêmement logique. Relisez le paragraphe précédent et imaginez le fonctionnement : la boucle sans fin, l'attente de connexion, le traitement de la connexion….

Maintenant nous allons nous intéresser à la procédure « affichemes ».C'est elle le cœur de notre serveur puisque c'est la gestionnaire d'événement. Pour ce faire créez une procédure globale « affichemes » (affichemes pour affiche messages).

```
PROCEDURE affichemes(canal)
texte est une chaîne
BOUCLE
        texte=SocketLit(canal,Vrai)
        ListeAjoute(""depart.Liste1"",texte)
FIN
ThreadArrête("",500)
Multitache(-30)
```

Comme vous pouvez le constater la procédure prend comme paramètre le nom du canal reliant les 2 sockets. Je vous rappelle que cette procédure est lancée en tant que thread. La fonction WinDev SocketLit lit le contenu du canal et le mets dans la variable texte. Le paramètre vrai signifie à SocketLit que la durée d'attente sur canal est indéfinie. Une fois le message récupéré il doit être mis dans notre liste déroulante « liste1 ». Le reste de la procédure n'appelle pas de commentaires particuliers.

Continuons par le code du bouton "demarre".

```
SI PAS SocketCrée("serveur",PORTIP,NetAdresseIp()) ALORS
        Erreur("Erreur de création " + ErreurInfo(errMessage))
        ETATSERVEUR="Problème lors du démarrage du serveur"
SINON
        ETATSERVEUR="Serveur démarré"
        ThreadExécute("thread1",threadNormal,attente)
FIN
```

Ce code lance l'exécution du serveur. La procédure WinDev SocketCréer utilise plusieurs parametres :

Le nom de la socket que l'on va créer.

Le port sur lequel on écoute.

L'adresse ip du poste serveur.

Comme vous le voyez à la lecture de ce code si la socket "**serveur**" est créée on fait de notre procédure globale "attente" un thread qui appellera lui meme le thread "**affichemes**". C'est pas super l'informatique ?

Voici le code du bouton arreter

```
SocketFerme("serveur")

ETATSERVEUR="Arret du serveur"
```

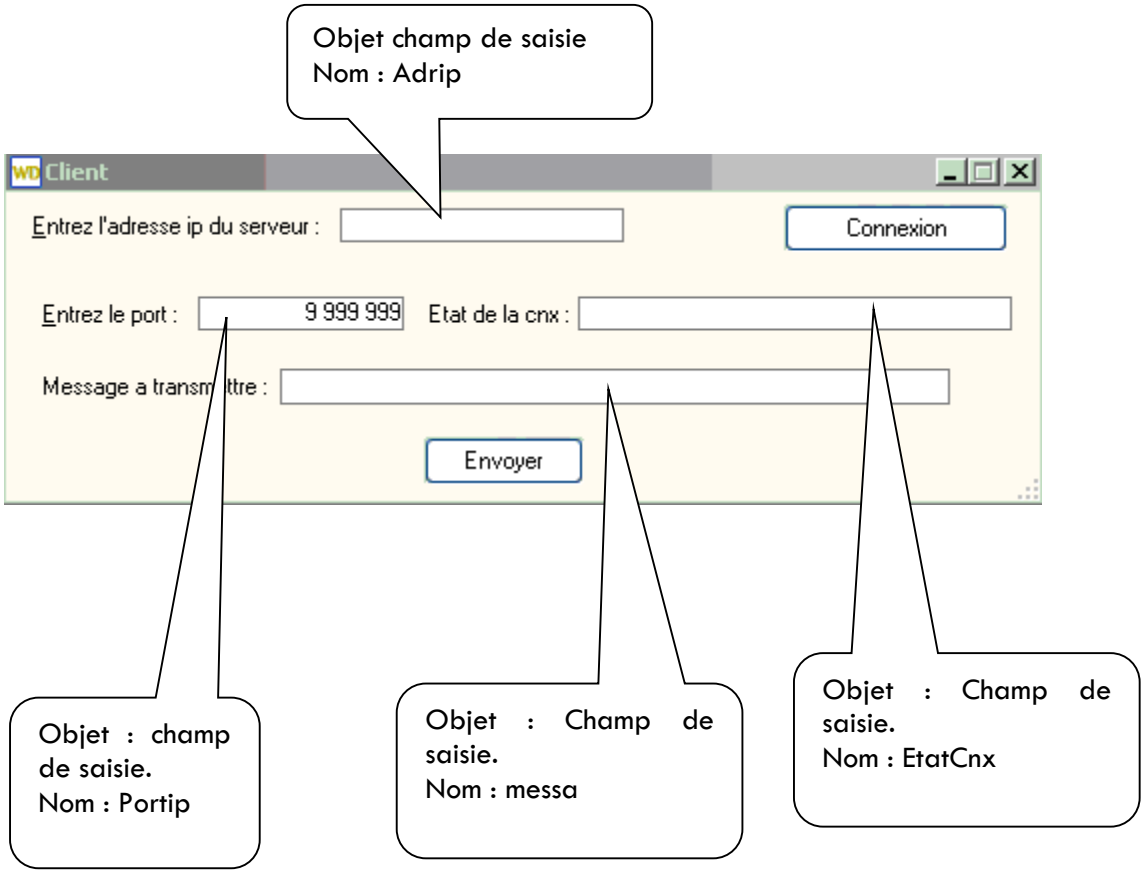
Voilà la partie serveur est maintenant terminée… Il ne vous reste plus qu'a créer l'exécutable (Menu Projet/Créer l'executable).

NB: Si vous avez des Warnings concernant une boucle sans condition de sortie ne vous en souciez pas !

Dés a présent vous avez concu un serveur, il ne nous reste plus qu'a créer le client. Son rôle est d'essayer de se connecter au serveur et de lui envoyer des message. C'est clair ? Alors Avanti !

Partie Cliente

Pour le client, comme ce doit etre une application autonome, il nous faut donc créer un projet, sans analyse ayant pour nom "**Client**". Ce projet ne contiendra qu'une fenetre nommée "**Depart**". Elle ressemblera a ceci :



Comme vous pouvez le constater le client est vraiment minimaliste. On saisit l'adresse ip du serveur, le port d'écoute, le message à transmettre. Le bouton **Connexion** nous servira pour établir la liaison, le bouton envoyer transmettra le message.

Interressons nous au bouton connexion. Son rôle est de nous mettre en relation avec la socket du serveur. Donc il est impératif que le serveur soit actif avant le client, mais ca les plus rusés l'avaient remarqué…

Voici le code du bouton :

```
SI PAS SocketConnecte("serveur", PORTIP,ADRIP) ALORS
        Erreur("erreur de connexion " +  ErreurInfo(errMessage))
SINON
        ETATCNX="Vous etes en ligne"
FIN
```

On essaye de ce connecter à la socket nommée "Serveur" (vous savez c'est celle que l'on vient de créer coté serveur ?!), écoutant le port défini (PORTIP), à telle adresse IP (ADRIP). Si tout ce passe bien on ecrit "Vous etes en ligne dans le champ de saisie Etatcnx. Sinon on envoie le message d'erreur.

Ben oui, c'est tout… Simple, non ?

Maintenant voyons le code du bouton envoyer.

```
SI SocketEcrit("serveur", messa) = Faux ALORS
        Info("Un problème est survenu")
FIN
```

On envoit un message (le texte contenu dans le champ de saisie "**messa**") à la socket "**serveur**". Si ca ne fonctionne pas on affiche une boite de dialogue d'avertissement.

Voilà tout est fini, vous pouvez compiler, créer l'executable et tester votre client/serveur. Pour la mise en œuvre vous avez besoin de lancer le serveur, le mettre en écoute d'un port. Ensuite vous lancez le client que vous branchez sur le port d'écoute et envoyez le message. Si vous etes en réseau utilisez la bonne adresse ip du serveur et have fun !!

Exercices applicatifs

Maintenant que vous êtes un pro de la socket essayez de réaliser l'exercice suivant.

A partir de l'executable faites en sorte que le client oblige le serveur a executer une action (par exemple activer la calculatrice)

Pistes de réalisation :

Coté client creez un bouton qui envoie le chiffre 1 par exemple à la socket serveur.

Coté serveur analysez les mesages recu et si message =1 alors …..

Avec cet exercice vous etes en train d'apprendre les base d'un outils de maintenance à distance. A vous de le perfectionner

Pour faire lancer une application windows par Windev voici la syntaxe : LanceAppli("CALC.EXE »). Calc.exe étant la calculatrice