Tp WinDev Numéro 6

Objectifs : Accès à des bases de données tierces (Mysql)

Langage Sql, Fonctions Sql de WinDev, gestion avancée des tree-view, des tables mémoire, des chaînes de caractères…

A l'heure actuelle, les bases de données de type Clients/Serveurs ont la faveur de plus en plus d'informaticiens. Une de ces bases se détache du lot, il s'agit de Mysql. Cette base de donnée offre plusieurs avantages non négligeables.

1. Elle est gratuite.
2. Elle est rapide
3. Elle est fiable.
4. Elle est **Multi-Plateforme**…

Le fait qu'elle existe sous plusieurs systèmes d'exploitations est appréciable, en effet à l'heure actuelle beaucoup de Fournisseurs d'Accès Internet vous permettent de vous connecter à une base de donnée, dans la majorité des cas cette base de donnée est Mysql. Pourquoi la choisissent elle ? En fait, ils ont fait le choix du logiciel libre, donc le système d'exploitation est Linux, comme les internautes utilisent de plus en plus le couple Php/Mysql, cette base de données était le choix technologique le plus évident à faire.

Maintenant imaginez la situation suivante : Vous êtes développeur dans une société qui a sur Internet un site où les clients peuvent passer des commandes. Le site utilise Php et Mysql, comment faire pour que votre logiciel commercial puisse récupérer les commandes directement dans la base de donnée Mysql qui est sur Internet ?

Je vous sens dubitatif ! Heureusement super Jean-Luc a la solution : Nous allons, ensemble, développer un logiciel de connexion à une base de donnée Mysql, créer des requêtes, récupérer les résultats…..Elle n'est pas belle la vie ? Mais cependant certaines mises en garde sont nécessaires :

Je vous rappelle qu'il est inutile de faire ce TP sans maîtriser les tp précédents. Je ne reviens pas sur les notions abordées dans les TP précédents donc voici l'adresse ou vous trouverez les autres TP : www.btsig.org.

Pour le bon déroulement de ce TP téléchargez et installez easyphp 1.6 que vous trouverez à l'adresse suivante : http://www.easyphp.org/

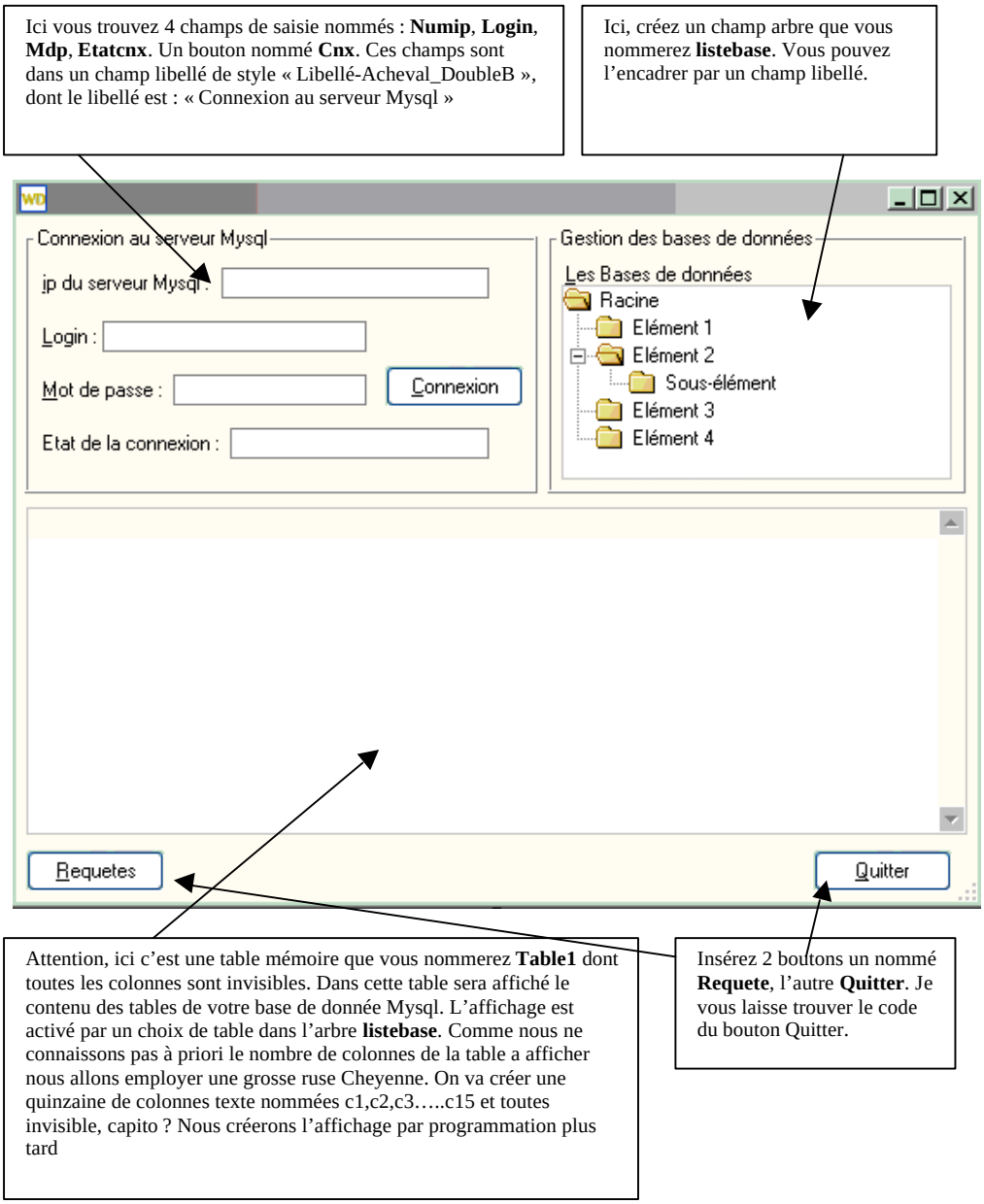
Durant le Tp, prenez le temps de comprendre ce que vous faite, aucun apprentissage efficace ne peut avoir lieu si on se contente de copier ou imiter. Donc si vous vous sentez « largué », pas de souci, apprenez à faire marche arrière et relisez. Je fais toujours en sorte que les explications soient claires, mais n'ayez pas honte de ne pas comprendre immédiatement, faites l'effort et vous serez toujours récompensé. N'oubliez pas l'excellente aide de WinDev (vous savez la touche F1 !)

Durant cet exercice, vous avez le droit de faire des pauses ;-))

Avant de commencer, vérifiez qu'Easyphp est actif (Vous devez apercevoir un E noir avec un point rouge clignotant dans la barre des taches). S'il n'est pas en fonctionnement, vous ne pourrez pas accéder au serveur MySql. Je vous laisse lire la doc fournie avec pour le faire fonctionner, c'est simple.

Nous allons maintenant entrer dans le vif du sujet. Créez un nouveau projet nommé TP6 qui ne comporte aucune analyse, normal puisque nous allons accéder à des données distantes. Ce projet comportera 2 fenêtres : Une orienté gestion de la base Mysql, l'autre sur l'édition de requêtes.

Voici le prototype de la première fenêtre que vous nommerez départ et qui sera la première fenêtre du projet.



Maintenant que la scène est installée, voyons les comportements des objets et ce que l'on attend d'eux. Pour ce connecter à une base de donnée Mysql nous avons besoin de divers renseignements :

L'adresse Ip de l'ordinateur où est située votre base de donnée Mysql. Si vous l'avez sur votre ordinateur, plusieurs possibilités s'offrent à vous, soit vous inscrirez dans ce champ l'adresse ip de votre ordinateur, soit localhost, soit une adresse de bouclage ex : 127.0.0.1.

Le login : Si vous venez d'installer votre base de donnée le login par défaut est : **root**. Sinon demandez un login à votre administrateur réseau.

Le mot de passe : Si vous venez d'installer Mysql sur votre poste le mot de passe par défaut n'est pas défini, donc cette zone restera vide, sinon contactez votre administrateur si vous n'en avez pas.

Donc, lorsque ces renseignements seront saisis nous essayerons de nous connecter à la base de donnée via le bouton de connexion, si la connexion réussie nous remplirons l'arbre avec les bases Mysql et pour chaques bases, les tables constituantes. Dans le champ etatcnx nous mettrons un texte nous indiquant le succès de la connexion sinon l'échec.

On y va ?Vous avez placé tous les objets ? Votre fenêtre ressemble à la mienne ? Vous l'avez enregistrée sous le nom de départ ? Vous l'avez déclarez comme première fenêtre du projet ? Non ! ! Mais qu'est-ce que vous faites ?

Nous allons avoir besoins de 2 variables globales à la fenêtre, allez dans le code de la fenêtre et dans la zone « déclaration globale de départ » inscrivez ceci :

```
GLOBAL
mabase,matable sont des chaînes
```

Ces 2 variables doivent être connues ou vues par tous les objets de la fenêtre, c'est pour cela que l'on les déclare globales au plus haut dans le conteneur principal.
NB : Le mot clé global pourrait dans ce cas là être omis.

Intéressons nous maintenant au comportement du bouton de connexion. Son rôle est d'établir une connexion avec la base de données Mysql. Il a besoin de l'adresse ip de la base, du login et du mot de passe pour établir le contact.

Voici son code :

```
resultat est un entier

resultat=SQLConnecte(Numip,login,mdp,"","MySQL")
SI resultat=0 ALORS
        etatcnx="Votre serveur Mysql n'a pas repondu"
SINON
        etatcnx="Votre connexion est active"
        lesbases()
FIN
SQLDeconnecte()// Une fois les traitements fait, on se déconnecte
```

La procédure WinDev importante est SQLConnecte son rôle est de connecter l'application en cours à une base de données à interroger par SQL. Pour cela elle a besoin de paramètre qui sont le contenu du champ Numip, de login, du mot de passe. Le 4eme paramètre est vide (les 2 apostrophes), il est dévolu au nom de la base de donnée souhaitée. Le 5eme paramètre est le type de base de données attaqué, dans notre cas MySQL. Appuyez sur F1 en étant positionné sur le mot SQLConnecte et regardez dans l'aide toutes les bases de données susceptibles d'être interfacées avec WinDev !

Cette procédure retourne un entier qui vaut 0 si la connexion n'a pas pu être établie (mauvaise adresse ip, mauvais login, mot de passe, serveur Mysql arrêté…etc…). Une bonne habitude a prendre est de tester la réussite ou l'échec d'une fonction. Donc, si resultat=0 on informe l'utilisateur qu'il y a un boulon dans le potage. Sinon c'est que tout va bien : on fait afficher dans le champ de saisie etatcnx que la connexion est active. Pour alléger le code nous allons créer une procédure locale à la fenêtre que nous nommerons lesbase. Le rôle de cette procédure est de nous remplir l'arbre avec les bases de données MySql et les tables les composants. Voyons son code :

```
PROCEDURE lesbases()
resultat est un booléen

resultat=SQLExec("show databases","requete1")

SI resultat ALORS // L'exécution de la requête est réussie
        TANTQUE SQLFetch("requete1") = 0
                ArbreAjoute(listebase,"Bases"+TAB+ SQLLitCol("requete1", 1),aDéfaut,aDéfaut,SQLLitCol("requete1", 1))
                lestables(SQLLitCol("requete1", 1))
        FIN
        SQLFerme("requete1")
        ArbreDéroule(listebase,"Bases")
SINON
        Info("Vous avez un problème de connexion")
FIN
```

Résultat est un booléen qui va recevoir le résultat d'exécution de la procédure SQLExec. Celle ci va exécuter la requête nommée « requete1 », dont le texte associé est « show databases » (cette commande Sql fait retourner l'ensemble des bases de donnée présentent sur votre serveur MySql).Si la requête s'exécute bien resultat vaut vrai sinon faux.

```
TANTQUE SQLFetch("requete1") = 0
```

Cette ligne de code peut vous paraître étonnante, en fait la fonction SQLFetch ne lance pas la récupération de toutes les lignes du résultat de la requête : seul l'enregistrement en cours est récupéré par la fonction SQLFetch. Le bizarre est que tant qu'elle retourne 0 la lecture de l'enregistrement s'est bien passée. Il y a des fois où la logique n'est plus de ce monde ! Mais ne nous laissons pas perturber par cette petite bizarrerie…. !

La procédure ArbreAjoute crée un tree-view avec les éléments que l'on va lui passer.
Dans notre cas il faut remplir avec le contenu résultant de la requête précédente. En fait le résultat de la requête est, dans ce cas là, une table d'une colonne contenant un nom de base par ligne.

SQLFetch parcourt les lignes de la table et SQLLitcol("requete1", 1) lit pour la requête passée en paramètre, le contenu de la colonne passée aussi en paramètre (ici 1).
Je vous laisse regarder l'aide d'ArbreAjoute et comparer avec le code, vous allez vite comprendre son fonctionnement.
Nous allons donc créer un arbre affichant les bases de données disponibles, mais pour chaque base il nous faut aussi insérer dans l'arbre les tables qui la composent. C'est le rôle de la procédure globale lestable que nous allons créer.
Comme vous le remarquez la procédure lestables prend comme paramètre le nom de la base de donnée contenu dans SQLLitCol("requete1", 1)

Créez donc cette procédure

```
PROCEDURE lestables(labase)
resultat est un booléen

SQLConnecte(Numip,login,mdp,labase,"MySQL")
resultat=SQLExec("show tables","requete2")
SI resultat ALORS
        TANTQUE SQLFetch("requete2") = 0
                ArbreAjoute(listebase,"Bases"+TAB+ labase+TAB+SQLLitCol("requete2", 1),aDéfaut,aDéfaut,SQLLitCol("requete2", 1))
        FIN
        SQLFerme("requete2")
        ArbreDéroule(listebase,"Bases"+TAB+labase)
FIN
SQLDeconnecte()
```

Comme vous pouvez le constater cette procédure récupère comme argument une chaîne de caractère (labase) contenant le nom de la base de donnée a traiter. Il faut ensuite se connecter à cette base de donnée (ligne 3)pour demander l'ensemble des tables la constituant (ligne 4).Une fois la requête exécutée, si elle a fonctionné, tant que des lignes existent dans le contenu du résultat de la requête, on les ajoute au bon endroit dans le tree-view (l'arbre). On ferme la requête avec la commande SQLFerme. Et on déroule l'arbre pour des raisons esthétiques. Pour des raisons de sécurité on utilise SQLDeconnecte() qui ferme la connexion en cours et libère l'espace mémoire utilisé par la connexion. La fonction SQLDéconnecte doit être appelée systématiquement pour fermer la connexion, même si cette connexion a échoué

Arrivé à ce stade, vous pouvez tester le résultat. Entrez les bons paramètres de connexion, appuyez sur le bouton connexion et sous vos yeux ébouriffés de surprise le tree-view est rempli de l'arborescence bases de données + tables.
Si sous vos yeux effrayés vous ne voyez rien paraître, vérifiez les points suivants :

- Votre serveur est-il actif ?
- Votre code est-il exempt de Bug ?
- Les paramètres de connexion sont-ils free of âneries ?
- Vos lunettes sont-elles propre ?
- ….

Maintenant, ce serait super de pouvoir lister le contenu d'une table dont on aurait cliqué sur le nom dans l'arbre. Pour cela il faudrait récupérer le nom de la table choisie et remplir la table mémoire. C'est tout simple voici le code que vous allez inscrire dans la zone clic sur **listebase** de l'objet **listebase** (l'arbre).

```
resultat est une chaîne

resultat=ArbreSelect(MoiMême)
mabase=ExtraitChaîne(resultat,2)
matable=ExtraitChaîne(resultat,3)

SI matable<>EOT ET mabase <>EOT ALORS //Nous avons la base et la table
        remplirtable(mabase,matable)
FIN
```

ArbreSelect(MoiMême) renvoie l'élément cliqué sous forme d'une chaine. Le problème c'est qu'il ne renvoi pas la terminaison (la feuille) mais l'arborescence complète. Par exemple si vous cliquez sur la table **clients**, contenue dans la base **gestcom** ayant comme racine **Bases** la chaîne resultat sera comme ceci : « Bases Gestcom clients ». Les éléments susceptibles de nous intéresser sont **Gestcom**, pour le nom de la base et **clients** pour la table. La fonction

ExtraitChaine va nous être d'un grand secourt. On lui donne la chaîne initiale et on lui dit de nous renvoyer le Xieme mot. Dans notre cas nous allons mettre dans la variable globale mabase le deuxième terme de la chaîne et dans la variable matable le troisième terme. Si jamais vous avez cliqué sur le nom de la base au lieu de cliquer sur le nom de la table, resultat sera composé de 2 mots et non de trois, dans ce cas ExtraitChaine(resultat,3) renverra **EOT**. Nous allons pouvoir remplir la table si les 2 variables (mabase et matable) sont différentes de EOT. Pour alléger la lecture et faciliter la compréhension, on va donc créer un traitant de remplissage de table, la procédure remplirtable qui va prendre 2 arguments : le nom de la base et le nom de la table.

Voici le code de remplirtable(mabase,matable)

```
PROCEDURE remplirtable(labase,latable)
resultat est un booléen
numconnexion est un entier
texte est une chaîne

TableSupprimeTout(Table1)

numconnexion=SQLConnecte(Numip,login,mdp,labase,"MySQL")

texte=SQLColonne(numconnexion,latable,Faux)

SQLExec("select * from "+latable,"requete3")
SQLInfoGene()
miseenforme(texte,SQL.NbCol)

SQLTable("requete3", "Table1")
SQLFerme("requete3")
SQLDeconnecte()
```

Voici l'explication du code. On commence par vider la table mémoire nommé table1. Ensuite on se connecte à la base de donnée dont le nom fut passé en paramètre.
Cette ligne : texte=SQLColonne(numconnexion,latable,Faux) renvoie dans la variable « **texte** » le nom des colonnes de la table choisie par l'utilisateur dans le tree-view. Le nom des colonnes nous sera utile pour mettre en forme la table mémoire on mettra en entête de table le nom des colonnes, ce sera plus parlant que c1,c2,c3…c15. La ligne suivante fait une requête select classique qui liste le contenu intégral d'un fichier donné (latable).
SQLInfoGene() va renseigner diverses variables sur la dernière requête lancée (requete3).

Nous ce qu'il nous intéresse, c'est de connaître le nombre de colonnes que va générer notre requête. Notre table sélectionnée contient-elle 5 colonnes, 2, 10 ? En fait, a priori nous n'en savons rien, c'est pour cela que je vous ai demandé de créer une table mémoire de 15 colonnes par défaut. C'est la variable SQL.NbCol qui va nous dire combien la requête a de colonnes. Mais n'oubliez pas que SQL.NbCol ne contient des infos qu'après l'appel de SQLInfoGene()

Nous allons commencer à créer l'entête de la table mémoire(table1) avant d'y transférer les données. C'est le rôle de la procédure « miseenforme(texte,SQL.NbCol) »

```
PROCEDURE miseenforme(lescolonnes,nbcol)
i est un entier=1
nomcol,exnomcol sont des chaînes

TANTQUE i<>nbcol
        nomcol=ExtraitChaîne(lescolonnes,i,RC)
        exnomcol="C"+i
        {exnomcol}..Titre=nomcol
        {exnomcol}..Etat=Visible
        i++
FIN
```

Le but de cette procédure est de remplacer les c1,c2, c3…..nbcol par un le nom de la colonne renvoyé par la requête. Nous allons donc affecter le nouveau nom à l'ancien. Nous avons passé a la procédure 2 paramètres : Une chaîne contenant les noms des colonnes séparés par un espace et le nombre de colonnes. Nous avons comme impératif de renommer la première colonne (C1) par le 1er terme contenu dans la chaîne (lescolonnes), la deuxième colonne (C2) par le 2eme terme de la chaîne et cela jusqu'à nbcol. Comme vous le voyez c'est ce que fait la boucle tantque.

On initialise une variable i à 1, puis tant qu'elle est différente de nbcol on place dans nomcol le terme contenu dans la chaîne lescolonnes à l'indice i. Ensuite on recrée le nom de colonne basé sur l'indice pour être en phase. La partie la plus surprenante est le fait de mettre entre accolades le nom de la colonne. C'est ce que l'on appelle une indirection, comme le nom du champ est dans une variable on fait ainsi comprendre à WinDev de traiter le contenu de la variable comme étant le nom du champ souhaité. On modifie le nom de la colonne par la constante ..Titre et on la rend visible par ..Etat=Visible. Voilà la procédure a fait la mise en forme. Elle s'arrête et repasse la main a la procédure appelante : la procédure remplirtable. L'exécution continue sur la ligne SQLTable(« requete3 », « Table1 »). Cette commande fait un transfert du contenu de la requête dans la table mémoire.

Voilà le tout est joué, sauvegardez et testez votre travail. C'est super non ?

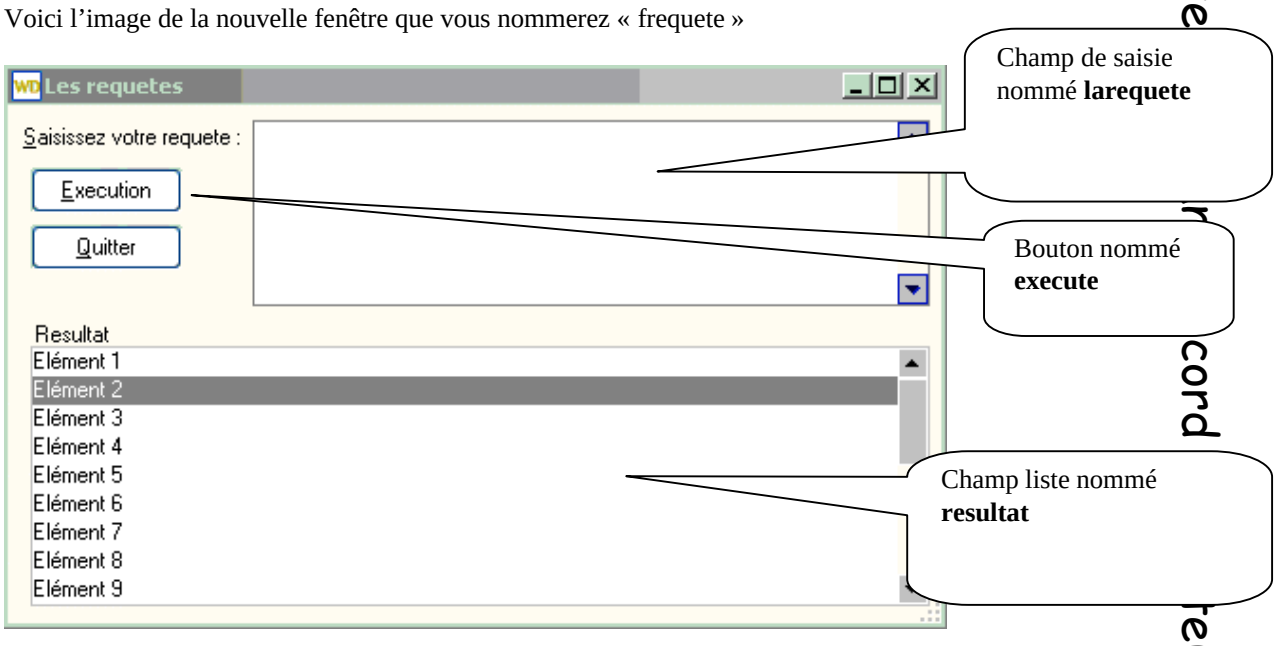
Mais tout cela serait plus magique si vous aviez une zone de saisie de requête sql pour créer des nouvelles bases de données, créer des nouvelles tables, insérer des enregistrements…..

C'est ce que l'on va réaliser maintenant, mais tout d'abord voici le code du bouton requete

```
SI mabase<>"" ALORS
        Ouvre(frequete,Numip,login,mdp,mabase)
SINON
        Info("Choisissez une base de donnée")
FIN
```

Si l'utilisateur a cliqué sur une base du Tree-view alors on peut lancer l'ouverture de la 2eme fenêtre nommée requête. Fenêtre à qui on passe 4 arguments de connexion : le numéro ip, le login, le mot de passe et la base sur laquelle on désire travailler.

Voici l'image de la nouvelle fenêtre que vous nommerez « frequete »



Voyons les éléments :

- Un champ de saisie nommé larequete dans lequel vous taperez le texte de votre requête
- Un champ liste nommé résultat dans lequel le résultat de votre requête apparaîtra.
- Un bouton execute qui contient le code chargé de se connecter à la base MySql, de faire exécuter la requête et d'afficher le résultat.

Voyons de suite le code du bouton **execution**

```
i est un entier
texte est une chaîne

ListeSupprimeTout(resultat) // on vide la liste resultat

SQLConnecte(numip,login,mdp,labase,"MySQL")// on se connecte à la base
SI larequete<>"" ALORS // si du texte a été frappé
	SQLExec(larequete,"requete4") // on fait exécuter la requête par Mysql
	SQLInfoGene() // on fait générer les variables concernant la requête
	SQLPremier("requete4") // on se positionne sur la première ligne des données retournées par la requête

	TANTQUE PAS SQL.EnDehors // Tant qu'il reste des lignes à lire
		i=1
		texte="" // Je vais construire dans texte la ligne en concaténant les colonnes
		TANTQUE i<=SQL.NbCol // de i au nombre de colonnes
			texte=texte+TAB+SQLCol("requete4", i) // Je concatene
			i++
		FIN
		ListeAjoute(resultat,texte) // J'ajoute dans la liste résultat la ligne que je viens de créer
		SQLSuivant("requete4") // je passe à la ligne suivante
	FIN
	SQLFerme("requete4") // je détruis ma requête
SINON
	Info("Veuillez saisir une requête")
FIN
SQLDeconnecte() // Je me déconnecte
```

Que puis-je dire de plus ? Tout est expliqué ! Ah ! Si ! ! Le tp numéro 6 est fini, je vous remercie d'avoir eu la patience d'arriver jusqu'ici…

Je vous dis à bientôt pour un nouveau TP

Exercice applicatif

Créer un système d'administration Mysql.

Imaginez une fenêtre ou l'on puisse créer des bases de données, les supprimer, créer des tables, les supprimer, ajouter des utilisateurs, des droits, des permissions….

Vous voyez ? Essayez de faire le plus simple et le plus ergonomique, n'oubliez jamais que l'informaticien n'est jamais l'utilisateur !

Bien sur vous aurez besoin de commandes Sql simples du type

Drop database
Drop table
Create database…….

A vous de jouer, bon courage !