



GUIDE DE
FORMATION
Windev

JEAN CLAVER MOUTOH
Ingénieur Informatique & Pétrole
05 162 543/04 329 929

Cours WinDev Numéro 0

Que fait-on avec WinDev ?

WinDev est un AGL (Atelier de Génie Logiciel). Il vous permet de développer des applications dans tous les domaines :

- Gestion (Comptabilité, Paie, Finances, Commerce, Stock, ...)
- Industrie (robots, caisses, automates, balances, lecteur de badge, supervision, ...)
- Médical
- MultiMedia
- Internet
- Accès distant
- ...

Les applications WinDev accèdent à toutes les bases de données, relationnelles ou non du marché. Toutes les bases de données sont supportées. WinDev est livré en standard avec Hyper File , une puissante base de données relationnelle, déjà utilisée sur des millions de sites !

WinDev est un outil de développement complet qui intègre tous les outils nécessaires au cycle de réalisation d'une application.

Contrairement à d'autres langages de développement traditionnels, il n'est pas nécessaire de chercher et de rajouter des modules pour pouvoir concevoir, tester et installer une application.

Le L5G (Langage de 5ème Génération) de WinDev, le W-Langage, vous étonnera par sa simplicité : quelques heures suffisent pour appréhender le langage, une semaine suffit en général pour maîtriser toute sa puissance !

WinDev permet de suivre, étape par étape, de la conception à la finalisation, le cycle complet du développement d'une application.

WinDev propose certainement l'environnement de travail le plus puissant, le plus facile et le plus intégré du marché ! Vos équipes créeront facilement de superbes applications.

L'éditeur de fenêtres de WinDev est 100% WYSIWYG ("Ce que vous voyez est ce que vous aurez"). Il permet de réaliser facilement de superbes fenêtres reliées aux données.

Cours WinDev Numéro 1

Objectifs : Connaître les éléments de base de l'éditeur WinDev.
Création d'un convertisseur Francs / Euro.

Pré requis : Aucun

Pour ce premier cours, nous allons créer un convertisseur Franc / Euro.

Lancer WinDev 9 en double cliquant sur son icône :



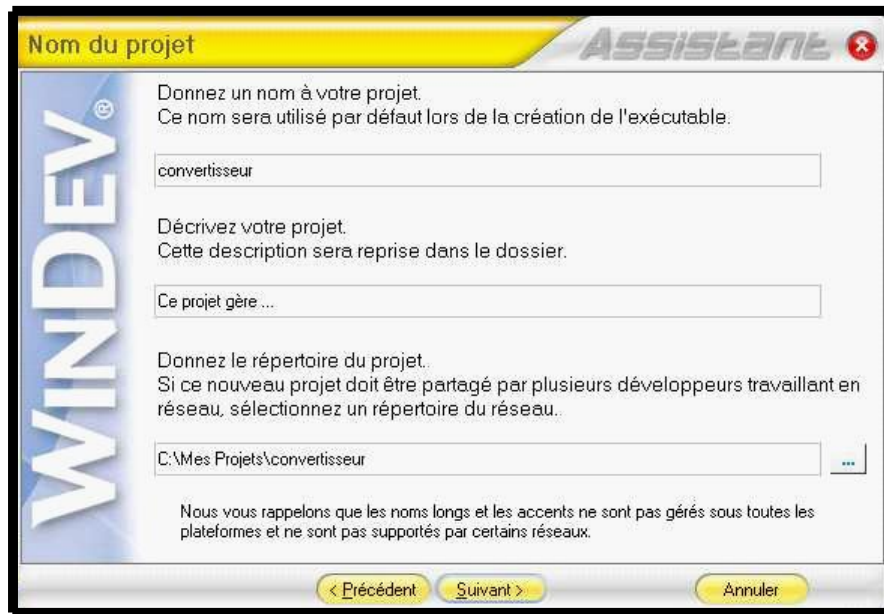
La fenêtre d'accueil apparaît (si ce n'est pas le cas, fermez le projet, quittez puis relancez WinDev) :



Cliquez sur Créer un projet.

La fenêtre Assistant vous indique qu'elle va vous aider en fonction de vos choix. On n'en attend pas moins d'elle.

Cliquez sur Suivant. Cette fenêtre va vous permettre de saisir le nom de ce nouveau projet et le lieu de stockage physique des éléments constitutifs.



Inscrivez « convertisseur » dans la zone nom, il sera repris automatiquement dans le champ du bas. Continuez en cliquant sur Suivant.

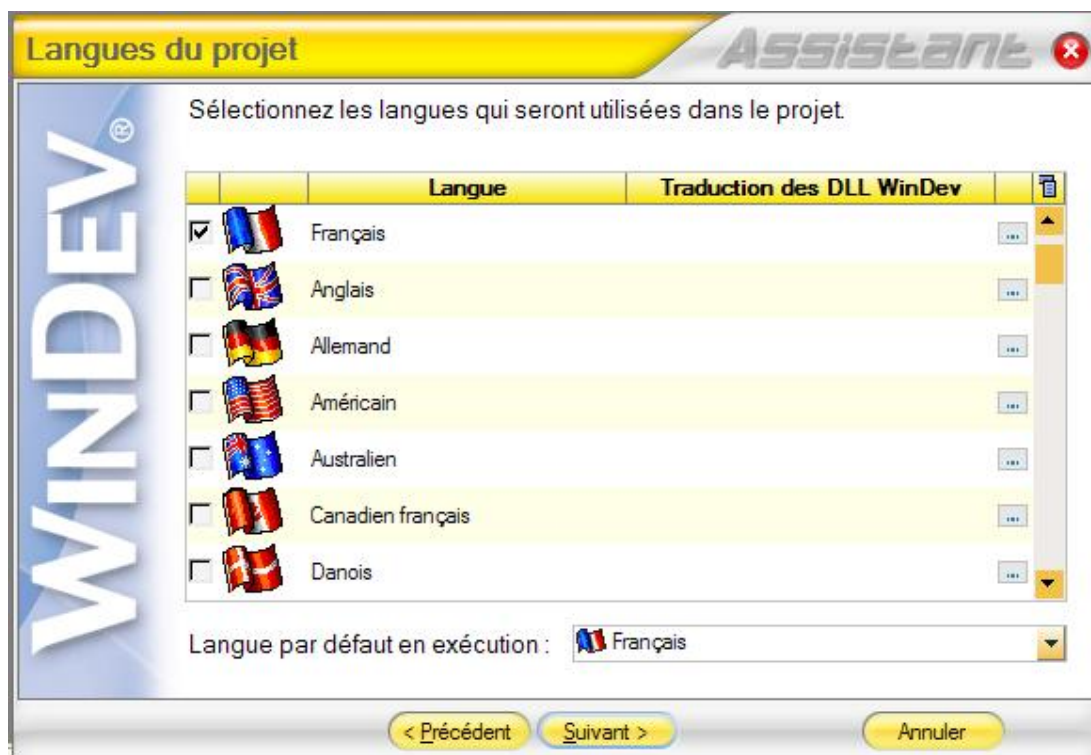
Créez un Type de projet Exécutable puis Suivant.



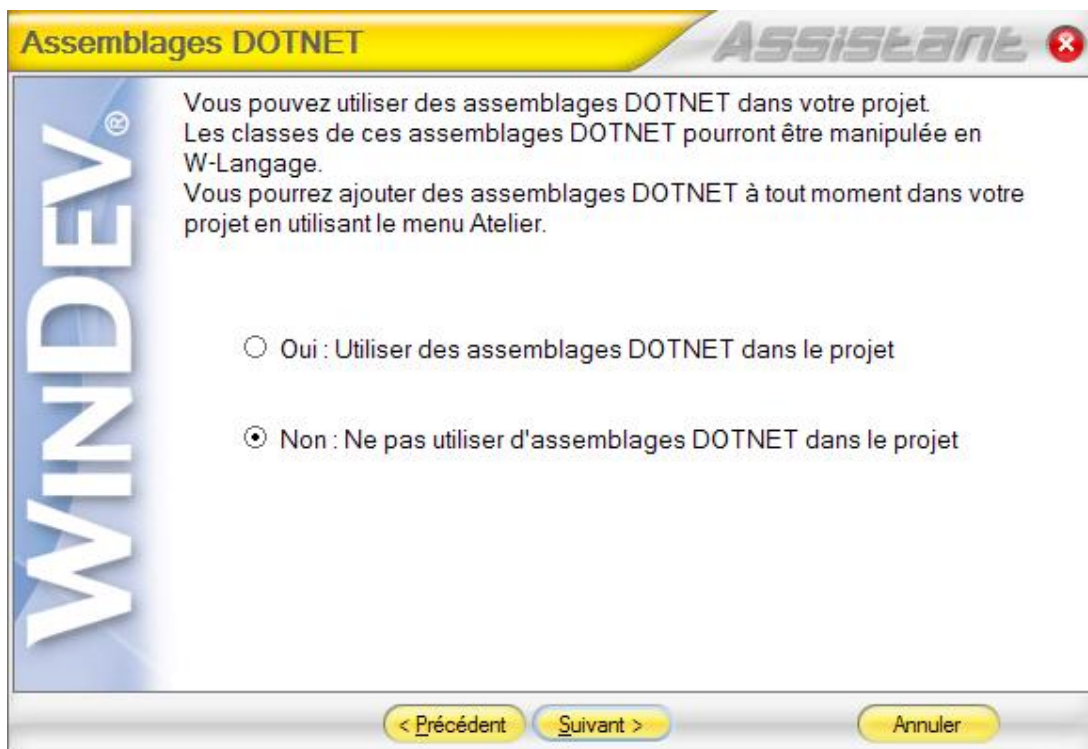
Prenez le Thème de Gabarit par défaut ou choisissez en un autre. Cliquez sur Suivant.



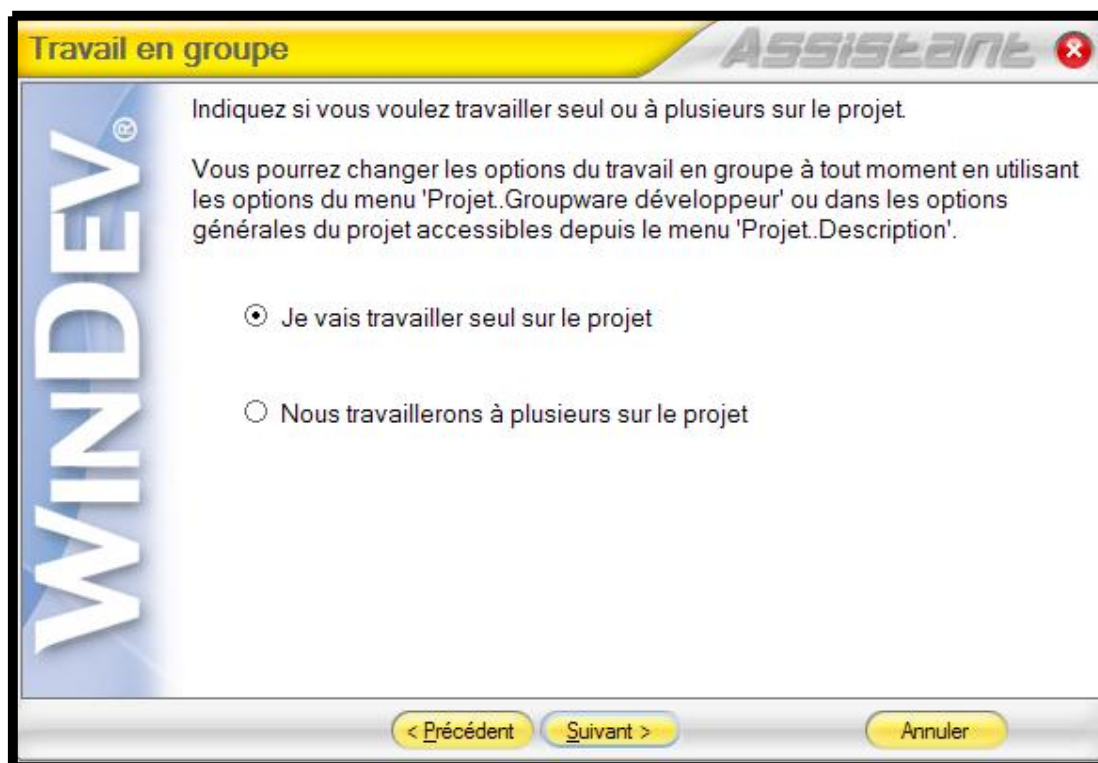
Le choix des langues ne nous intéresse pas pour l'instant. Cliquez sur Suivant



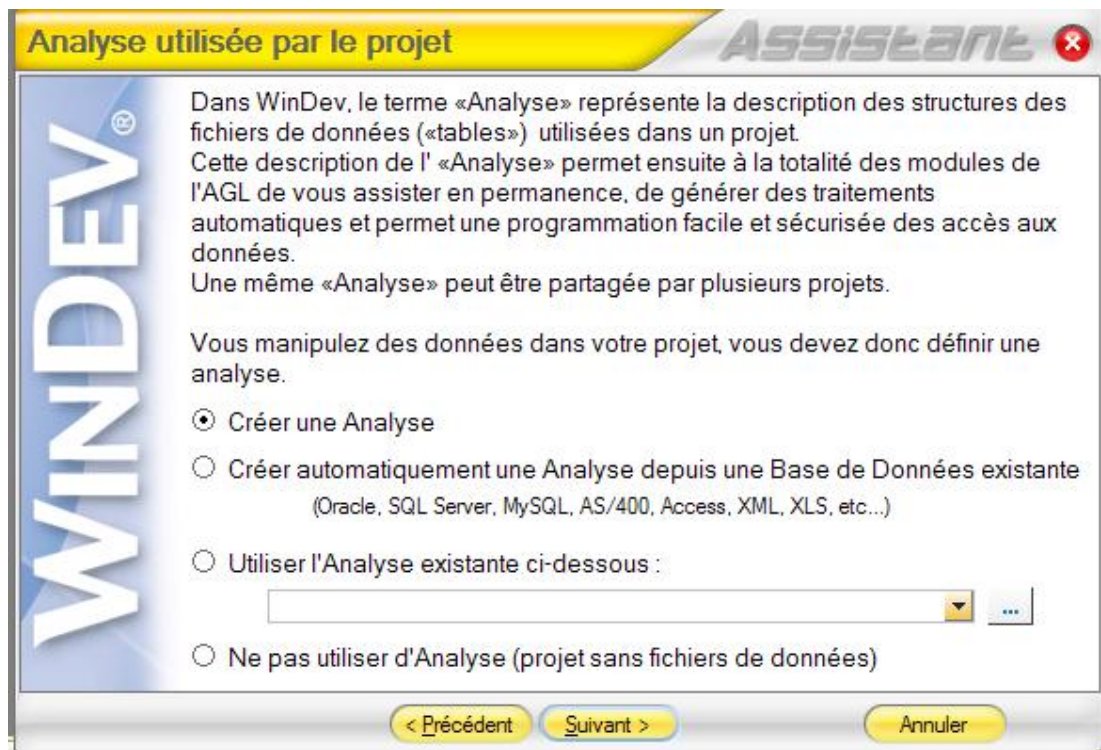
Cliquez sur Suivant et ignorer les assemblages DOTNET dans le projet



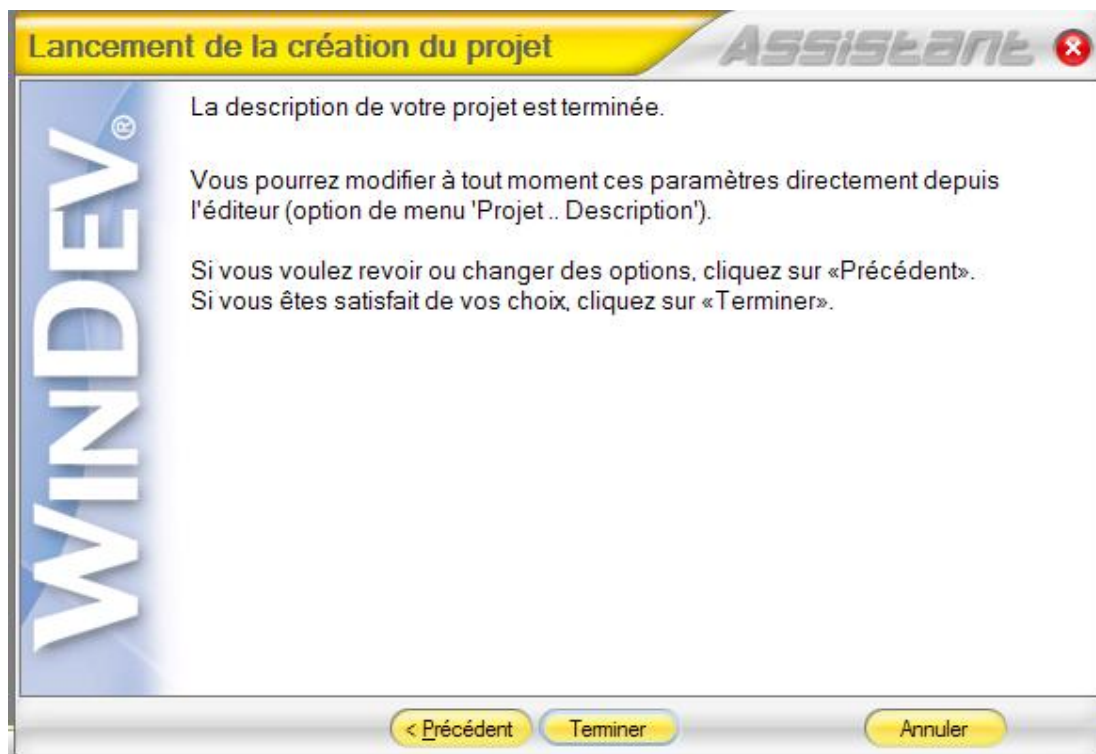
Comme vous travaillez seul pour cet écran, faites un clic sur suivant.



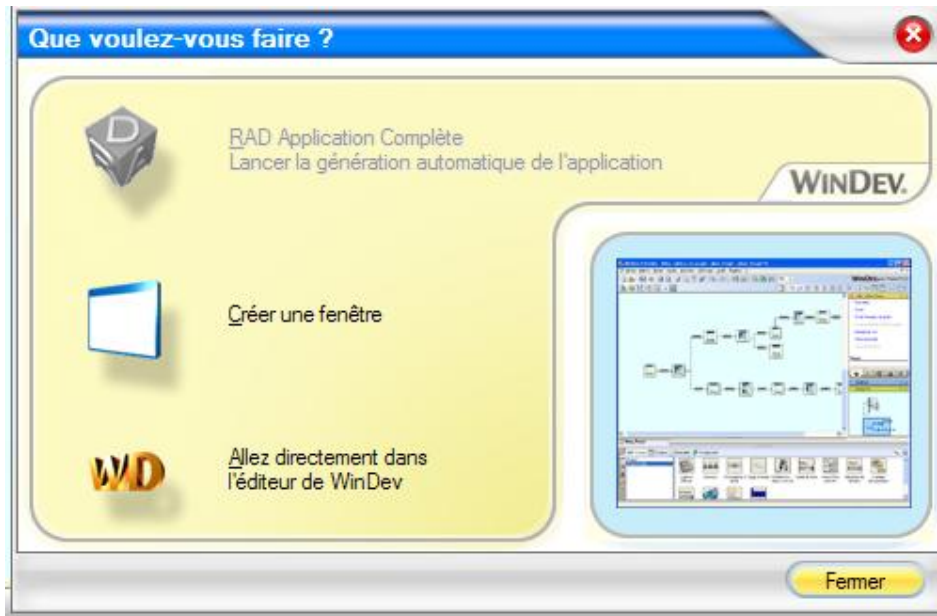
Choisissez Ne pas utiliser d'Analyse puisque pour notre projet nous n'utiliserons pas de fichiers de données.



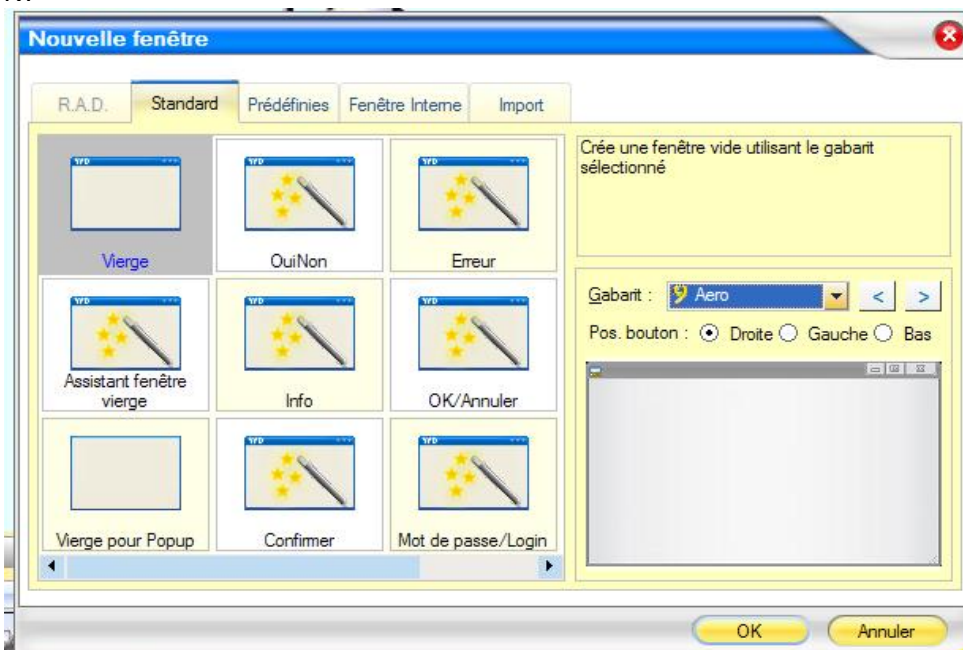
Continuez en cliquant sur Suivant et Terminer.



L'assistant va vous poser la dernière question ? Voulez vous créer une fenêtre ?

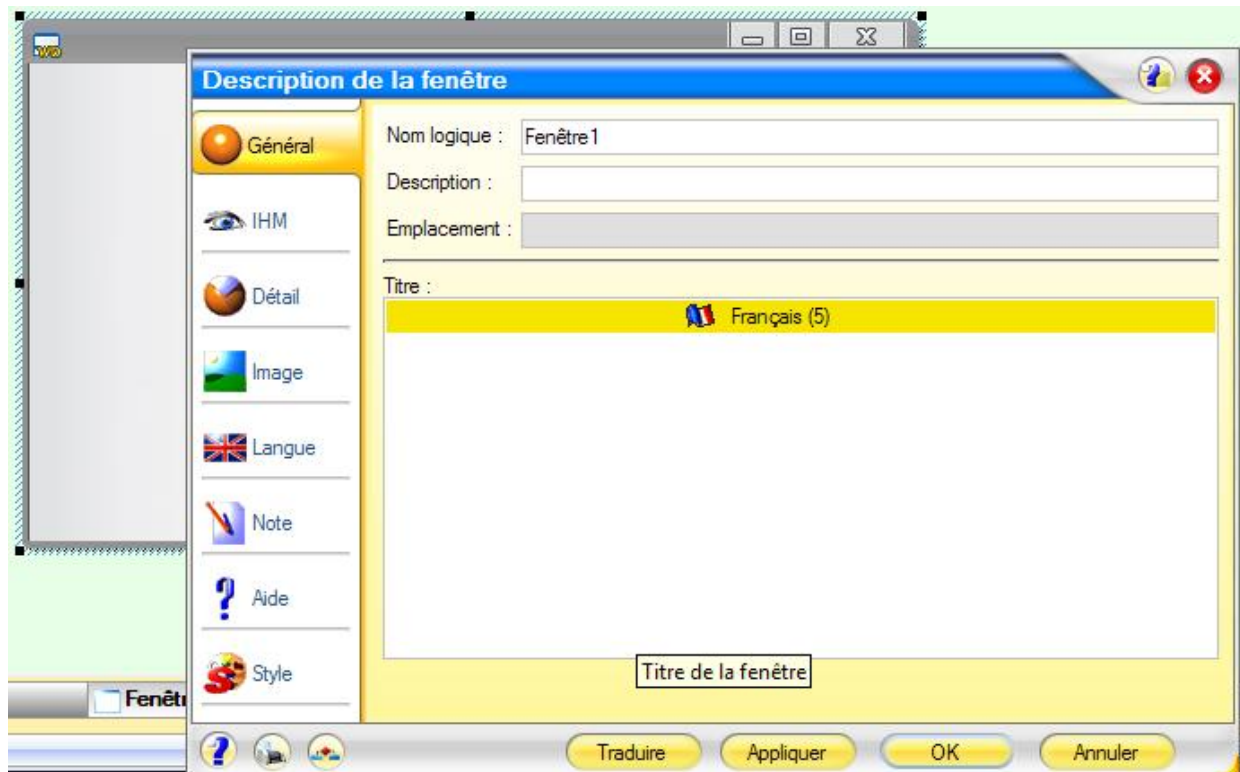


Cliquez Créer une fenêtre puis choisissez Vierge dans l'onglet Standard. Validez par OK.

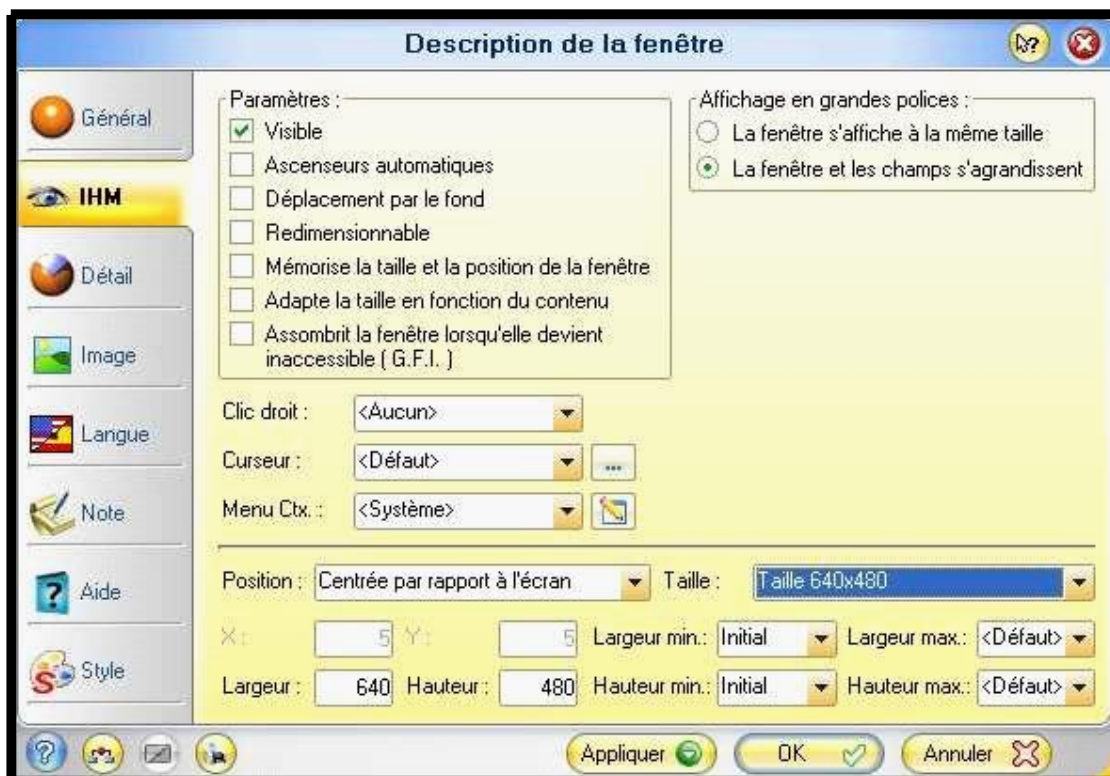


A ce stade nous avons une fenêtre vide dans l'éditeur. Il nous reste à lui donner un nom, une taille et définir quelques comportements.

Placez la souris dans la fenêtre vierge et faites un clic droit. Dans ce menu contextuel, validez Description. Vous obtenez la fenêtre à remplir comme ci-dessous :

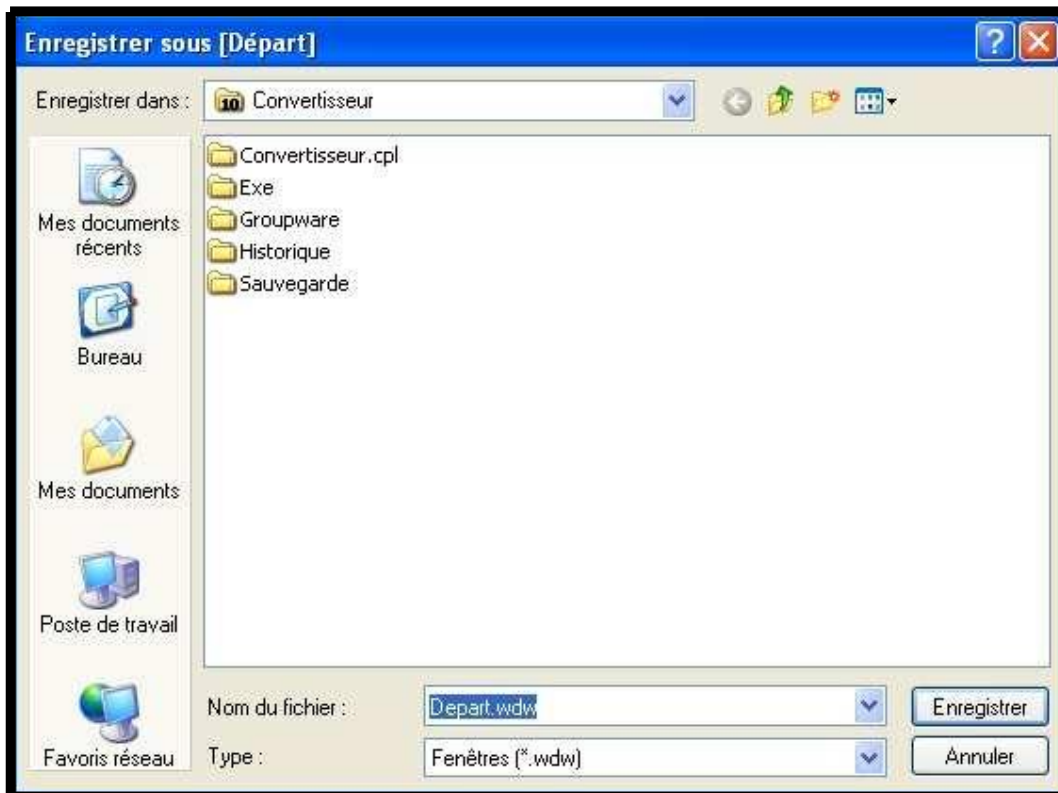


Remplissez les divers champs de façons identiques et sélectionnez l'onglet IHM (Interface Homme-Machine).



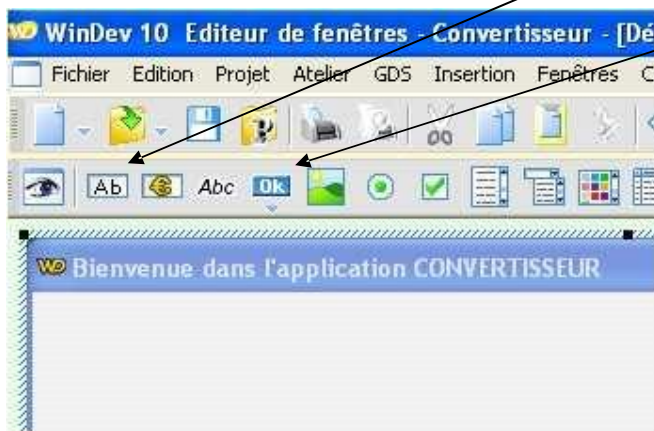
Changez la taille, le fait qu'elle ne sera pas redimensionnable puis validez par OK. Remarquez les différences, vous avez maintenant une fenêtre avec un nom, un titre et une taille définie.

Il est temps de sauvegarder, Cliquez sur Fichier puis Enregistrer.

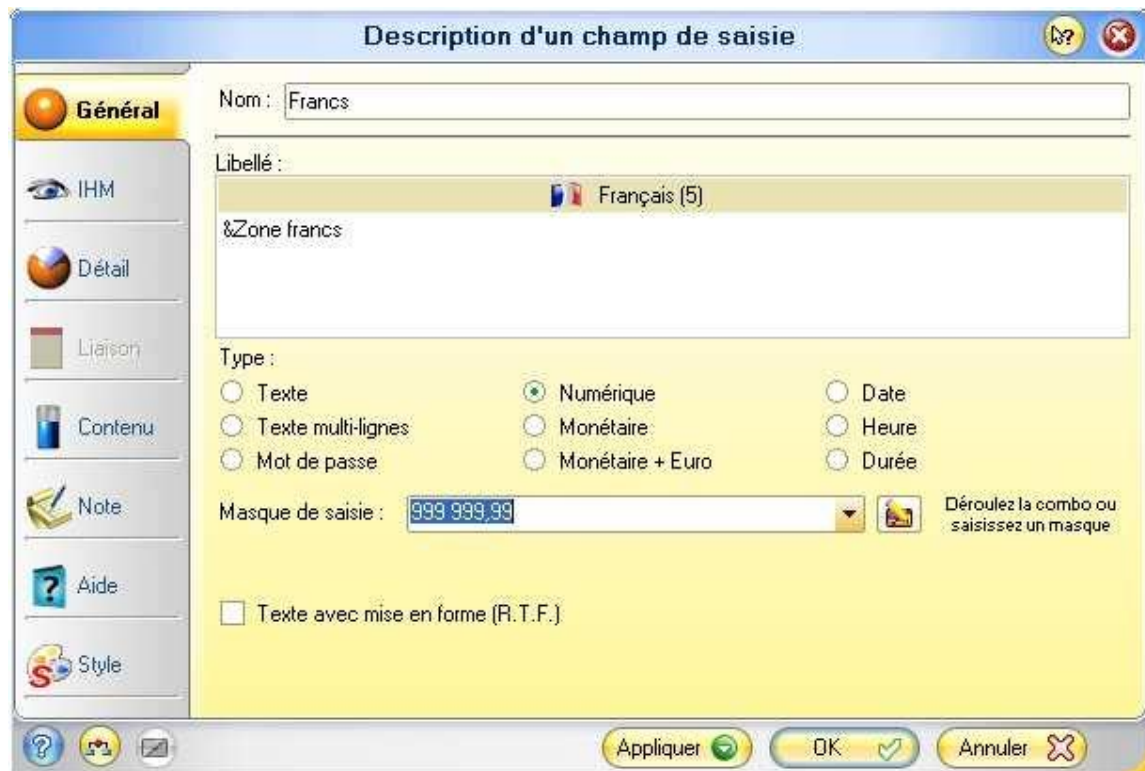


Cliquez sur le bouton Enregistrer.

A l'intérieur de cette fenêtre, placez 4 objets : 3 "champs de saisie" et un bouton (faites un glisser/déposer ou drag & drop des champs dans la fenêtre « Bienvenue dans le méga convertisseur »).



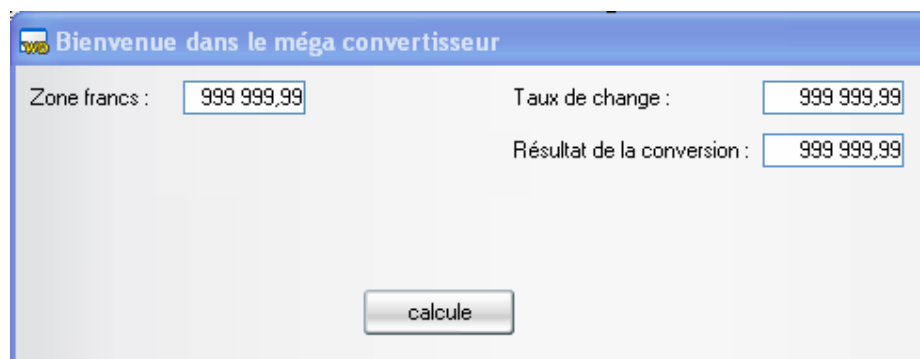
Le premier champ se nommera franc et aura comme libellé "Zone francs :". Cliquez 2 fois dessus pour en modifier les caractéristiques :



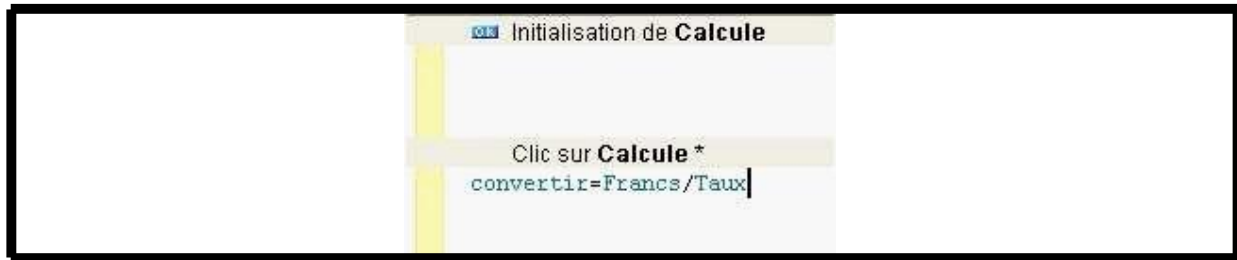
Le second se nommera taux et aura comme libellé "Taux de change :".

Le troisième se nommera convertir et aura comme libellé "Résultat de la conversion :". Le bouton (sans cliquer sur la petite flèche vers le bas placée juste en dessous) enfin se nommera calcule et aura comme libellé "calcule".

Voici à quoi doit ressembler votre fenêtre (avec en surimpression le nom des champs) :



Il nous reste à mettre le code correspondant dans le bouton "calcule". Pour cela, faites un clic droit dessus et choisissez "Code".

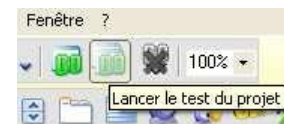


Entrez la séquence comme indiqué ci-dessus puis fermez la fenêtre de l'éditeur de code : cliquez sur l'icône de fermeture en haut à droite de l'écran (croix du bas).



Il est temps d'enregistrer notre projet par le menu Fichier / Enregistrer tout.

Une fois l'enregistrement achevé, nous allons tester le projet, pour cela cliquez sur Lancer le test du projet.



WinDev vous demande de définir la première fenêtre de notre méga projet, choisissez "depart" dans la combo et validez.

Vous avez maintenant devant vous notre super convertisseur. Mais avouez qu'il est franchement moche :

- trop grand ;
- couleurs tristes ;
- et même des comportements par défaut peu pratiques.

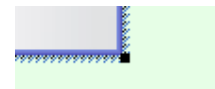
Lesquels ?

C'est simple : Essayez de saisir le taux de conversion de l'euro 6,55957 !

Comme vous le voyez, le champ ne prend que 2 chiffres après la virgule ! De plus vous allez être obligé de le saisir à chaque fois.

Nous allons remédier à tous ces petits détails :

Pour la taille de la fenêtre, placez votre souris sur l'angle inférieur droit de la fenêtre "depart" :



Une fois que le curseur change d'aspect, tenez appuyé le bouton gauche de la souris et remontez vers l'angle supérieur gauche. Relâchez la souris quand la taille souhaitée sera atteinte.

La couleur de fond n'est pas très gaie. Clic droit n'importe où dans la fenêtre : Description, puis onglet Style et Combo "Gabarit en cours". Prenez Orange. Validez 2 fois par Ok.

Changeons le comportement du bouton "taux", faites un clic droit dessus, Description. Vérifiez que le type soit Numérique. Maintenant dans la zone Masque de saisie , frappez 9,99999. Appliquez les modifications puis cliquez sur le bouton Editer le code  (en bas à gauche).

Insérer le code : `Moi-même=6.55957` dans la zone "Initialisation de taux". Ainsi à chaque démarrage du convertisseur la zone de saisie sera remplie. Notez que nous aurions pu écrire : `taux=6.55957`. Moi-même désigne l'objet dans lequel on se trouve.

 Initialisation de **taux**

`MoiMême=6.55957`

Refermez la fenêtre en cliquant sur la croix du bas (en haut à droite). Relancer le test de l'application en cliquant sur  et utilisez votre super convertisseur.

Cours WinDev Numéro 2

Objectifs : Connaître les objets de base de WinDev avec manipulation des :

combo-box,
interrupteurs,
tables mémoires,
...

Pré requis : Cours WinDev Numéro 1.

Pour ce second cours, nous allons créer une fenêtre comportant plusieurs objets de base dont nous allons étudier le comportement.

- Créez un projet nommé TP2 sans Analyse et une fenêtre nommée « sélecteurs » et « Bonjour » pour **Titre**.

Insérez un champ sélecteur à l'intérieur de votre fenêtre.



Allez dans sa description (clic droit / Description) dans la zone **Nom** du champ inscrivez : Civilité. Idem dans le **Libellé** du champ.

Dans la zone **Options** inscrivez :

Mademoiselle
Madame
Monsieur

Rajouter un champ de saisie que vous nommerez **choix** avec comme libellé « Elément sélectionné : ».

Votre fenêtre doit ressembler à ceci :



Vous allez faire en sorte que le champ « Élément sélectionné » se renseigne selon la Civilité. Pour cela, allez dans le **Code** du champ « civilité » (clic droit) dans la zone « A chaque modification de Civilité » et saisissez le code suivant :

```

SELON Civilité
  CAS 1:
    choix="Madame"
  CAS 2:
    choix="Mademoiselle"
  CAS 3:
    choix="Monsieur"
FIN

```

Comme vous le remarquez, WinDev ne renvoi pas le libellé du choix effectué mais l'index (ou position) de l'élément (1, 2 ou 3).

Testez la fenêtre en cliquant sur Go.

Les listes déroulantes

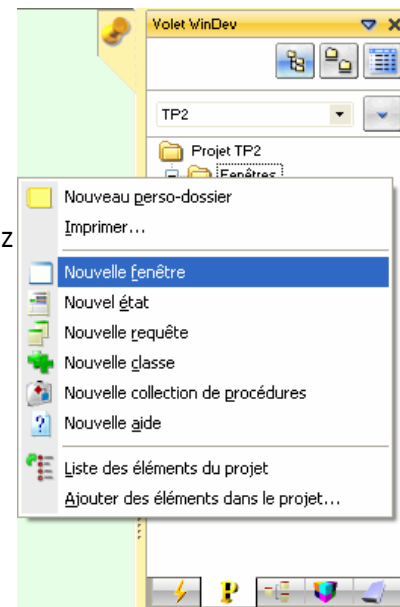
Créez une nouvelle fenêtre que vous nommerez « liste » et « Liste déroulante » en **Titre**. A droite de l'écran, cliquez sur le 2^{ème} onglet  puis sur le bouton droit de la souris pour pouvoir choisir **Nouvelle fenêtre**. A

l'intérieur, placez :

Une combo (**Créer une combo**) que vous nommerez « liste » et qui aura comme libellé « Civilité » (ne pas saisir de liste de valeurs à afficher).

Un champ de saisie nommé « **choix** » et ayant « **Choix** » comme libellé.

Un bouton nommé « remplir » et ayant comme libellé « Remplir la liste Civilité ». La fenêtre ressemblera à ceci :



Nous allons programmer le bouton pour qu'il remplisse la combo avec les éléments souhaités (Madame, Mademoiselle, Monsieur). Puis nous allons définir le comportement de la combo pour qu'elle affecte le champ « choix ».

Dans la zone « Clic sur **remplir** » du **Code** du bouton écrivez le code suivant :

```
ListeSupprimeTout(liste) // Pour éviter d'ajouter les éléments à  
chaque clic sur le bouton  
ListeAjoute(liste,"Madame")  
ListeAjoute(liste,"Mademoiselle")  
ListeAjoute(liste,"Monsieur")
```

Vérifiez que votre liste soit correctement remplie en cliquant sur le bouton **Remplir** puis en l'ouvrant avec la flèche vers le bas :



Dans la zone « sélection d'une ligne de... » du **Code** de la combo **Civilité** écrivez le code suivant :

```
choix=liste..ValeurAffichée
```

Par cette simple ligne vous demandez à la combo de copier la valeur affichée dans le champ « choix ».

Sauvegardez et testez la fenêtre.



N'oubliez pas d'utiliser l'aide pour approfondir vos connaissances !!

Les tables mémoires :

Le champ table permet de simplifier l'affichage et la saisie d'informations stockées en mémoire ou provenant d'un fichier de données, d'une vue ou d'une requête. Une table est composée de lignes et de colonnes. L'intersection d'une ligne et d'une colonne définit une cellule. Une table peut être gérée ligne par ligne, colonne par colonne ou cellule par cellule.

Les informations affichées dans la table peuvent :

- être déterminés par programmation : on parle alors de **Table Mémoire** ;
- provenir d'un fichier de données ou d'une requête : on parle alors de **Table fichier**.

Les tables permettent de sélectionner un ou plusieurs éléments de la table.

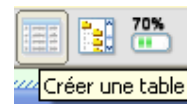
Pour créer un champ de type Table :

1. Sous l'éditeur de fenêtres, cliquez sur l'icône .
2. Cliquez dans la fenêtre à la position où le champ doit être créé. L'assistant de création d'un champ table se lance automatiquement.

Pour afficher les caractéristiques du champ, sélectionnez l'option « Description » dans le menu contextuel du champ. Deux types d'informations peuvent être visualisées :

- les informations concernant la table dans son ensemble (il suffit de sélectionner le nom de la table) ;
- les informations concernant chacune des colonnes de la table (il suffit de sélectionner le nom d'une des colonnes de la table).

Pour notre exemple, créez une fenêtre nommée « **tablemem** » et « Table mémoire » pour Titre, créez un champ de type table et positionnez le sur la fenêtre.



Indiquez à l'assistant que vous allez remplir la table vous-même. Cliquez sur **Terminer** pour sortir de l'Assistant, sans vous soucier des autres choix qu'il vous propose. Nous allons modifier manuellement les propriétés de la table.

Pour ce faire, cliquez sur la table avec le bouton droit de la souris pour faire apparaître le menu contextuel. Choisissez « **Description** ».

Nommez la table « **matable** ».

Créez 3 colonnes en appuyant 2 fois sur le bouton « **Nouveau** ».

Pour la première colonne : Nommez la « **Nom** », son type restera **Texte**, dans la zone « **Titre** » de l'onglet « **Général** » écrivez « **Le Nom** ». Dans la zone « **taille de saisie** » écrivez « **50** ». Ainsi l'utilisateur ne pourra pas inscrire un nom de plus de 50 caractères.

Pour la deuxième colonne : Nommez la « **Prénom** », son type sera **Texte**, dans la zone « **Titre** », écrivez « **Le Prénom** » et 20 caractères de taille de saisie.

Pour la troisième colonne : Nommez la « **Age** », son type sera **Numérique**, dans la zone « **Titre** » écrivez « **Age** », dans la combo « **masque de saisie** » trouvez le masque « **999** » (en haut de liste), cela signifie que seuls des entiers de 3 chiffres maximum seront acceptés.

Une fois ces manipulations réalisées vous pouvez cliquer sur « **Appliquer** » et « **Ok** » pour valider vos choix.

Ajouter 4 Boutons :

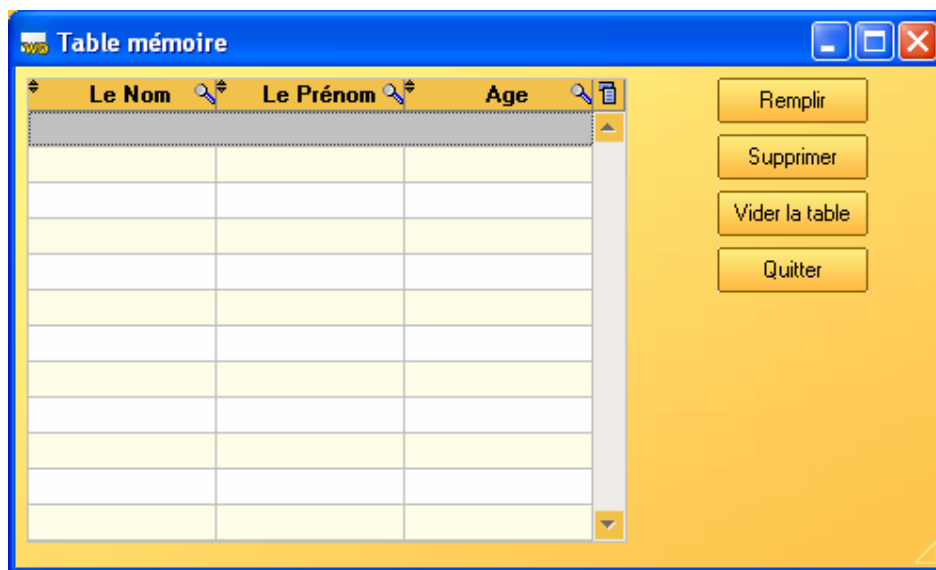
Le premier se nommera « **remplirtable** » et aura comme libellé « **Remplir** » ;

Le second se nommera « **supligne** » et aura comme libellé « **Supprimer** » ; Le

troisième se nommera « **videtable** » et aura comme libellé « **Vider la table** » ;

Le quatrième se nommera « **quitter** » et comme libellé « **Quitter** ».

Voici une représentation de votre fenêtre :



Nous allons étudier les différents codes permettant de remplir la table avec des informations, supprimer la ligne sélectionnée, vider complètement la table et enfin fermer la fenêtre.

Dans la zone « **Clic sur remplirtable** » du bouton « **Remplir** », insérez le code suivant (n'hésitez pas à employer le copier-coller pour éviter de le retaper) :

```
TableAjoute(matable,"Jesus"+TAB+"Christ"+TAB+100)
TableAjoute(matable,"Dieu"+TAB+"Emmanuel"+TAB+10)
TableAjoute(matable,"Moutoh"+TAB+"Claver"+TAB+13)
TableAjoute(matable,"Agnéro"+TAB+"Pascalyné"+TAB+22)
TableAjoute(matable,"Moutoh"+TAB+"Chloé"+TAB+1)
```

« Jésus » correspond au Nom, « Christ » au Prénom et « 100 » à l'âge. TAB indique le changement de colonne.



Utilisez l'aide pour avoir plus de renseignements sur la fonction Tableajoute

Dans la zone « **clik sur supligne** » de « **Supprimer** », insérez le code suivant :

```
TableSupprime(matable)
```

Dans la zone « **clik sur videtable** » de « **Vider la table** », insérez le code suivant :

```
TableSupprimeTout(matable)
```

Dans la zone « **clik sur quitter** » de « **Quitter** », insérez le code suivant :

```
Ferme
```

Testez les différents boutons et appuyez sur la loupe (à coté du nom de la colonne) pour tester son comportement par défaut.

Comme vous pouvez le constater, WinDev est puissant et peu de lignes de codes suffisent. Il est bien évident que le remplissage de la table peut être fait à partir de la lecture d'un fichier.

Pour terminer ce cours, il nous reste à faire une fenêtre de départ comportant 3 boutons qui ouvriront les différentes fenêtres.

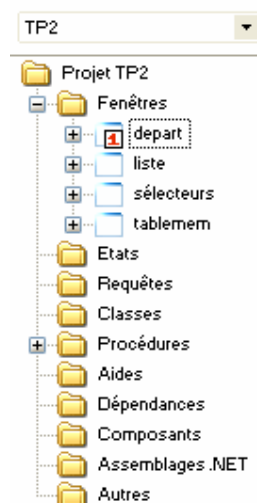
Créez donc une nouvelle fenêtre vierge que vous nommerez « **départ** », son **Titre** sera « Bonjour ». Insérez-y 4 boutons : 3 serviront à lancer les fenêtres, 1 à quitter l'application :



Pour indication, le code d'ouverture d'une fenêtre est : ouvre, le code de fermeture est : ferme. Je vous laisse mettre le code correspondant.

Il ne vous restera qu'à ne pas oublier d'enregistrer cette fenêtre pour pouvoir la déclarer comme première fenêtre du projet.

Dans la zone exploratrice des éléments du projet, faites un clic droit sur la fenêtre départ et choisissez « **Première fenêtre du projet** ».



Cours WinDev Numéro 3



Objectifs : Travailler avec un fichier de données :

Etude du gestionnaire d'analyse,
Manipulation des tables mémoires,
Manipulation de données,
Création d'états,
...

Pré requis : Cours WinDev Numéro 1 et 2

L'objectif de cette leçon est de vous familiariser avec l'utilisation des fichiers, du gestionnaire d'analyse et de la conception d'états.

Vous allez commencer par créer un nouveau projet nommé TP3. Dans l'assistant, vous sélectionnez votre thème préféré, confirmerez le choix de **création d'une Analyse** et finirez le processus de l'assistant. Un nouvel assistant de création d'analyse va apparaître, cliquez sur **Suivant**.

Vous pouvez choisir entre un Modèle Logique de Données et un Modèle Conceptuel de Données. Cliquez sur **Suivant** pour choisir le MLD.

Dans la zone Nom de l'analyse appelez là : **TP3** et vérifiez qu'elle soit bien associée au projet en cours, cliquez ensuite sur **Suivant**.

Nous ne donnerons pas de mots de passe à l'analyse donc cliquez encore sur **Suivant**.

Cette nouvelle fenêtre vous demande de choisir le type de base de données que vous voulez "attaquer". C'est ici par exemple que vous pourriez décider d'utiliser une base de données tierce du type Mysql, Access ou autres. Comme le but de cette leçon est de travailler avec WinDev nous allons faire en sorte de **Créer une nouvelle description de fichier**. Ce choix étant validé nous allons maintenant créer notre structure de fichier.

En fait nous allons programmer une mini gestion de budget familial. Pour cela on va utiliser un seul fichier des mouvements dans lequel on inscrira le descriptif des opérations, la date de l'opération, le montant au débit ou le montant au crédit. Cela devrait vous faire penser à votre relevé de compte bancaire.

Dans la zone **Nom**, vous allez indiquer le nom suivant : **Mouvement**.

Remarquez les champs suivants qui se remplissent automatiquement.

Vérifiez que la case « Le fichier possède un identifiant automatique » soit cochée.

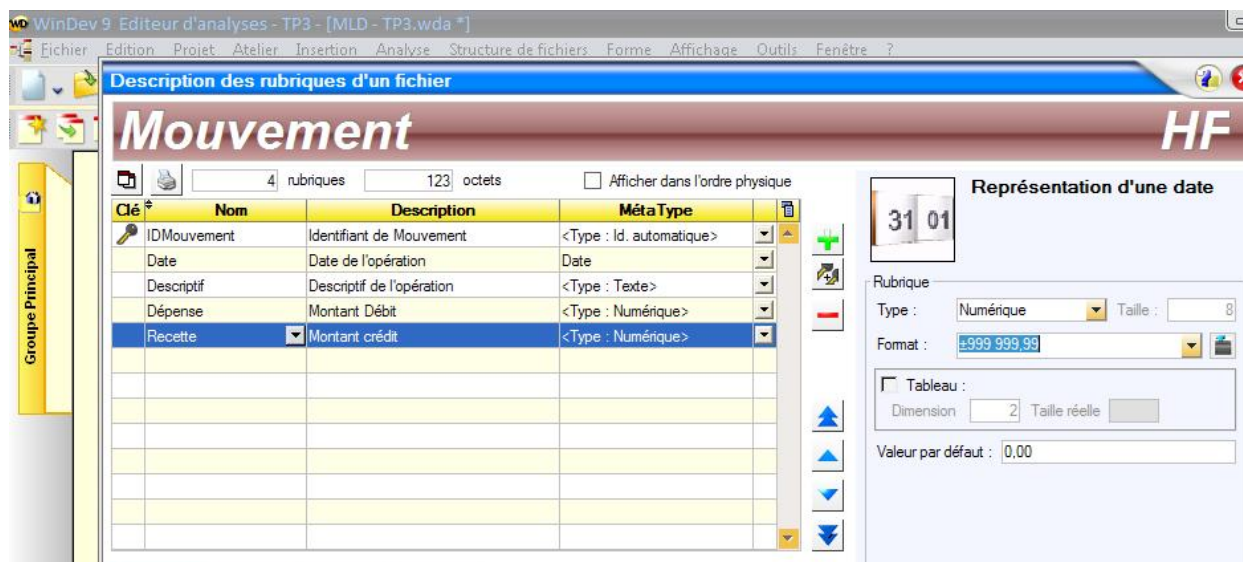
Nous allons ainsi créer un identifiant automatique par WinDev. L'identifiant automatique est comparable à un compteur, c'est lui qui vous garantit l'unicité de vos tuples.

Cliquez sur **Suivant**. La nouvelle fenêtre vous demande de confirmer le fait que vous voulez travailler avec des fichiers de type Hyper file (Format propriétaire WinDev). Cliquez sur **Suivant** puis de nouveau sur **Suivant**. La nouvelle fenêtre vous propose des options RAD (Rapid Application Development). Le RAD est à oublier, c'est WinDev qui fait tout, vous n'apprendrez rien à le voir faire, il vaut mieux que ça soit vous qui créez que lui. Donc décochez ces cases et cliquez sur **Terminer**.

Vous voilà maintenant dans le gestionnaire d'analyse, vous vous trouvez plus exactement dans la fenêtre de description des fichiers. C'est ici que nous allons déterminer la composition de chaque attribut de notre fichier mouvement.

Vous pouvez remarquer que notre identifiant est déjà créé « Idmouvement » et vous voyez la clé jaune à gauche qui symbolise l'identifiant. Nous allons insérer les rubriques suivantes :

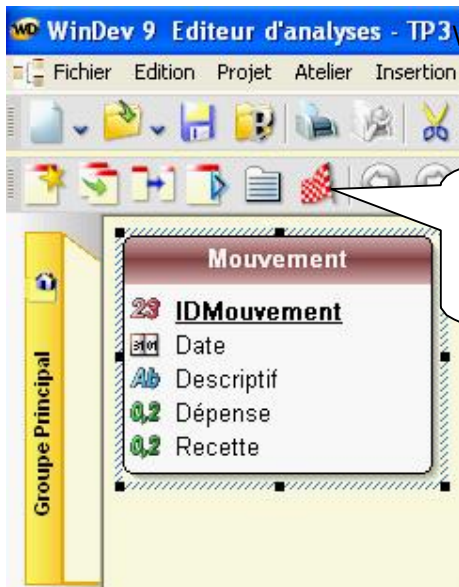
Nom	Libellé	Type	Taille
Date	Date de l'opération	Date	Sera rempli automatiquement
Descriptif	Description de l'opération	Texte	50
Dépense	Montant Débit	Numérique (Format ±999 999,99)	Sera rempli automatiquement
Recette	Montant Crédit	Idem	Idem



Pour insérer, cliquez sur une ligne vide dans la rubrique choisie. A vous de bien remplir vos champs, cela ne devrait vous poser aucun problème. Une fois la saisie terminée, cliquez sur **OK**.

Vous pouvez maintenant **Retourner sous l'éditeur de WinDev**. Choisissez ensuite

Aller directement dans l'éditeur WinDev et OK.



Voici une représentation de ce que vous devriez obtenir

Ce drapeau sert à lancer la génération de l'analyse. En effet, chaque fois que vous créez ou modifiez une structure de fichier, vous devez l'activer pour synchroniser l'analyse et les champs des fenêtres qui accèdent aux données.

Maintenant que votre fichier est décrit, il ne vous reste plus qu'à générer l'analyse.

Nous allons maintenant fabriquer les fenêtres de notre application. Allez dans le menu **Fichier / Nouveau** et cliquez sur **Fenêtre**. Choisissez une fenêtre vierge.

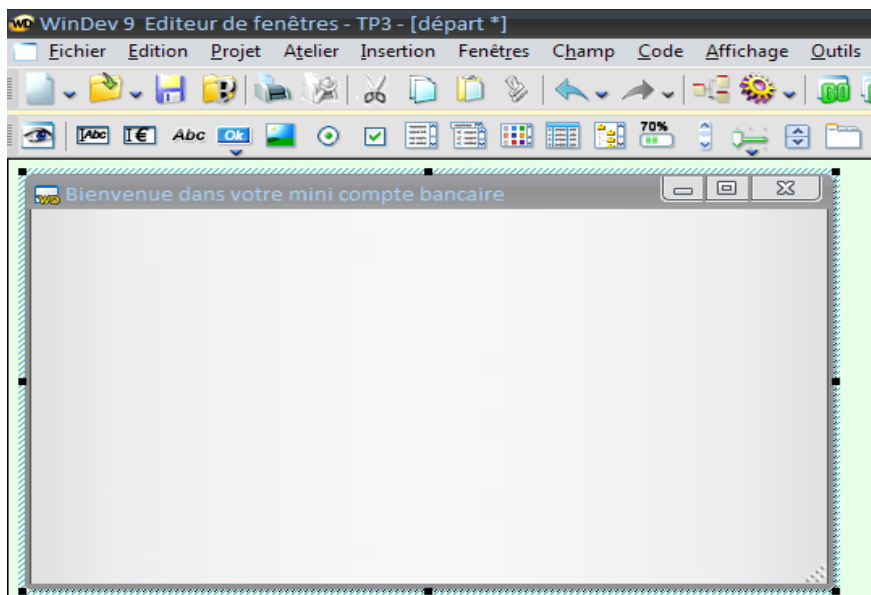
Cette fenêtre sera la première fenêtre de notre application. Dans sa description (clic droit sur la fenêtre), vous lui donnerez les caractéristiques suivantes :

Nom logique : **départ**

Description : **Fenêtre principale de l'application**

Titre : **Bienvenue dans votre mini compte bancaire**

Appliquez les modifications et cliquez sur **Ok**.



Nous allons insérer dans cette fenêtre une table mémoire qui sera le conteneur du fichier. Choisissez un champ table dans la barre d'outils et positionnez le sur votre fenêtre. Dites à l'assistant que vous remplirez la table vous-même, cliquez sur **Suivant**.

Choisissez Type de « **Table en affichage** » puis **Terminer**. La table est définie par défaut, allez dans la **Description** (clic droit sur la table).

Le nom de la table : Tmouv

Nom de la colonne 1 : tdate

Type de la colonne 1 : Date

Titre de la colonne 1 : Date

Nom de la colonne 2 : tdescription

Type de la colonne 2 : Texte

Titre de la colonne 2 : Description de l'opération

Taille de saisie : 50

Respectez les mêmes conventions de nommage que celles indiquées car tout le reste du projet s'appuiera sur ces noms là !

Nom de la colonne 3 : tdébit

Type de la colonne 3 : Numérique

Titre de la colonne 3 : Dépense

Nom de la colonne 4 : tcrédit

Type de la colonne 4 : Numérique

Titre de la colonne 4 : Recette

Nom de la colonne 5 : clé

Type de la colonne 5 : Numérique

Masque de saisie : 999 999 999.

Dans l'onglet **IHM** de cette 5^{ème} colonne, décochez **Visible**, nous mettrons dans ce champ l'identifiant de la ligne. Il n'est pas nécessaire de le montrer à l'utilisateur, c'est pour ça que je vous conseille de le mettre invisible.

Description d'un champ table

Nom table : Tmouv

Nom : tdescription tdébit tcrédit clé

Type : Texte Numérique Texte Numérique

Général

Etat initial :

- ☐ Visible
- ☒ En saisie / actif
- ☐ Affichage seulement
- ☐ Gris

Mode ellipse (tronquer) :

- ☒ Sans ellipse
- ☐ En fin
- ☐ Chemin de fichier

Clic droit : <Aucun>

Une fois tous les champs renseignés, cliquez sur **Appliquer** puis **OK**

Voici ce qu'il vous reste à concevoir pour terminer notre projet :

Nom du champ : Solde
Type : Numérique



Il me semble que ça fait longtemps que vous n'avez pas sauvegardé votre projet ! ce n'est pas très prudent... N'oubliez pas de déterminer cette fenêtre comme la première fenêtre du projet

Maintenant que le décor est planté, nous pouvons commencer la programmation.

La première chose à faire est de dire à WinDev de nous créer le fichier **Mouvement** s'il n'existe pas, pour cela allez dans le menu Projet / Code du projet et dans la zone **Initialisation de TP3** inscrivez le code suivant :

```
HCréationSiInexistant(Mouvement)
```



Remarque : Cette ligne indique à WinDev de commencer à chercher si le fichier **Mouvement** existe, s'il ne le trouve pas il le conçoit. Le code placé dans cette zone est exécuté avant le chargement de la première fenêtre.

Le code du bouton Quitter est très facile : Dans clic sur **Quitter** inscrivez :

```
Ferme
```

Voyons maintenant la décomposition possible des événements. Il faut qu'au Chargement de la fenêtre la table se remplisse avec les enregistrements contenus dans le fichier situé sur le disque dur. Pour cela nous allons parcourir l'ensemble des lignes du fichier « **Mouvement.fic** » et les placer les unes après les autres dans la table mémoire. C'est ce que nous allons faire maintenant.

Allez dans le code de la fenêtre dans la zone « **Initialisation de départ** » et saisissez le code suivant :

```
HLitPremier(Mouvement, IDMouvement)
TANTQUE PAS HEnDehors
    TableAjoute(Tmouv, Mouvement.Date + TAB + Mouvement.Descriptif + TAB + Mouvement.Dépense + TAB + Mouvement.Recette + TAB + Mouvement.IDMouvement)
    HLitSuivant(Mouvement)
FIN
```

Explication du code :

HLitPremier(Mouvement, IDMouvement) // Cette ligne ordonne à WinDev d'ouvrir le fichier Mouvement et de lire la première ligne en plaçant les champs correspondants en mémoire.

TANTQUE PAS HEnDehors // Ici on commence une boucle qui sera vraie tant que l'on reste dans le fichier. Le mot clé Hendehors renvoie Vrai si on est hors du fichier. Donc PAS HEnDehors est vrai tant que la fin du fichier n'est pas atteinte. C'est tout simple en fait, non ?

La séquence tableajoute est connue, elle permet de positionner du texte - ici les rubriques du fichier - à l'intérieur d'un fichier.

Hlitsuivant(Mouvement) // Fait descendre le pointeur du fichier d'une ligne vers

Le bouton **Ajouter** va ouvrir une fenêtre de saisie qui nous permettra de rentrer les informations. Donc le code sera dans **Clic sur ajout** :

Ouvre(saisie)

Il est normal que WinDev vous renvoie un message d'erreur si nous essayons d'exécuter le projet car la fenêtre saisie n'existe pas, créons-la de suite.

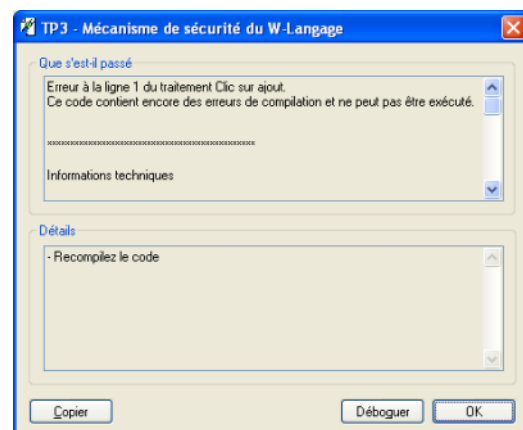
Fichier / Nouveau / Fenêtre

Dans la description de la fenêtre :

Nom logique : saisie

Description : Fenêtre de saisie

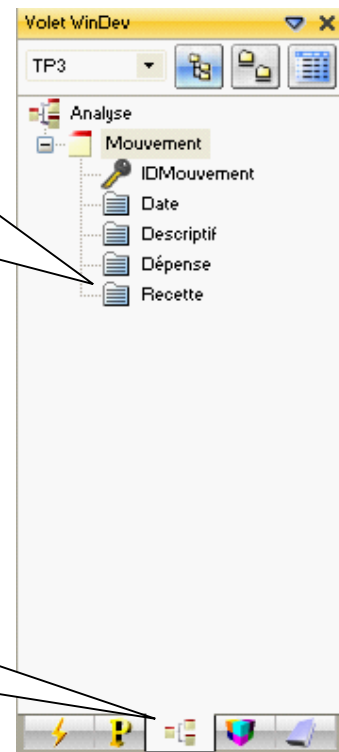
Titre : Saisissez votre opération



Faites glisser les champs **Date**, **Descriptif**, **Dépense**, **Recette** sur la fenêtre **saisie**.

Cliquez sur un champ en tenant le bouton gauche de la souris enfoncé, faite glisser et déposez sur la fenêtre. N'hésitez pas à changer la mise en forme du champ. Le fait de faire glisser le champ vous garanti que WinDev peut faire l'association entre le champ sur la fenêtre et la rubrique du fichier.

Cet onglet vous permet de sélectionner les fichiers constituant votre analyse. Je vous laisse chercher à quoi correspondent les autres.



Votre fenêtre devrait correspondre à celle-ci :

Pour être sûr que chaque champ est bien lié à une rubrique du fichier, cliquez sur l'un d'entre eux et vérifiez dans la barre de message en bas à gauche que le message « Lié à : Mouvement.XXXXXX » soit présent.



Cette fenêtre ne comporte que 2 boutons, le bouton **Annuler** nous servira juste à fermer la fenêtre. Vous connaissez la séquence de code le permettant, ce n'est donc plus la peine que je vous indique la marche à suivre. En cas de problème je vous rappelle que l'aide de WinDev est accessible par la touche **F1**.

Consacrons-nous au bouton **Valider**, la validation consiste à placer les rubriques de la fenêtre dans le fichier et à valider l'ajout. Pour placer les rubriques de la fenêtre dans le fichier, l'ordre est le suivant :

```
EcranVersFichier(saisie) // saisie étant le nom de la fenêtre
```

La validation d'ajout est commandée par l'ordre suivant :

```
HAjoute(Mouvement) // Mouvement étant le fichier dans lequel on ajoute
```

Il nous reste plus qu'à ajouter un ordre de fermeture de la fenêtre.

Voici le code intégral du bouton **Valider** :

```
EcranVersFichier(saisie)
HAjoute(Mouvement)
Ferme
```

Vous pouvez faire en sorte de programmer le contrôle pour qu'il vérifie que l'on n'a qu'un débit ou qu'un crédit, ou alors que l'un des champs n'est pas vide. Le code à placer juste avant celui que l'on voit ci-dessus pourrait ressembler à ceci :

```
SI Date="" OU Descriptif="" OU (Dépense=0 ET Recette=0) ALORS
    Info("L'un des champs n'a pas été rempli !")
    Ferme
FIN
SI Dépense <>0 ET Recette <>0 ALORS
    Info("Vous ne pouvez saisir qu'un montant au Débit ou alors au Crédit !")
    Ferme
FIN
```

Testez votre fenêtre avec le bouton  ou en cliquant à droite de l'écran sur la Fenêtre avec le bouton droit pour choisir **Tester**. Insérez des valeurs dans les champs et validez.

Pour voir si votre nouvelle ligne est présente dans le fichier, allez dans le menu **Outils / WDMAP** et choisissez **Mouvement** comme **Nom du fichier**.

Maintenant que vos lignes s'insèrent, lancez le projet en cliquant sur GO .

Cliquez sur le bouton **Ajouter**, saisissez et validez un nouvel enregistrement. Vous pouvez constater que la table mémoire ne réagit pas correctement : en effet l'insertion n'a pas été détectée et donc la table mémoire n'est pas synchronisée avec le fichier. Nous allons essayer de remédier à ce problème. En fait, il faudrait que lorsque la fenêtre **saisie** se ferme, la fenêtre **départ** recharge la table mémoire.

Placez-vous dans le code de la fenêtre **départ**, vous devez trouver une zone nommée « **prise de focus de départ** ». La prise de focus est le fait de remettre active une fenêtre inactive, en cliquant sur la barre de titre par exemple.

Voici le code à insérer dans cette zone :

```
TableSupprimeTout(Tmouv) // Efface la table mémoire pour éviter d'insérer les
enregistrements à la suite des précédents
HLitPremier(Mouvement, IDMouvement)
TANTQUE PAS HEnDehors
    TableAjoute(Tmouv, Mouvement.Date+TAB+Mouvement.Descriptif+TAB+Mouvement.Dépense+TAB+Mouvement.Recette+TAB+Mouvement.IDMouvement)
    HLitSuivant(Mouvement)
FIN
```

Testez cette modification, comme vous le voyez, les comportements sont maintenant cohérents.

Intéressons-nous au bouton **Supprimer**.

Dans la table, nous avons une rubrique qui est l'identifiant de la ligne.

Pour supprimer cette ligne dans le fichier nous allons donc rechercher cet identifiant dans le fichier et supprimer la ligne correspondante.

Voici la séquence de code nécessaire :

```
HLitRecherche(Mouvement, IDMouvement, clé)
SI HTrouve ALORS
    HSupprime(Mouvement)
    Info("La suppression est effective")
SINON
    Info("Grave problème de l'application")
FIN
```

Explications :

La première ligne fait rechercher dans le fichier Mouvement et sur l'identifiant la valeur du champ **clé** pointée dans notre table mémoire. Ce n'est pas parce que la rubrique **clé** est invisible que nous ne pouvons pas en connaître la valeur.

Si on trouve la ligne ayant le même identifiant que **clé** alors on la supprime et on affiche un message indiquant la bonne marche des opérations. Sinon dans un cas fort improbable où il ne trouve pas l'enregistrement on inscrit un message d'erreur.

Je vous laisse le soin de tester cette nouvelle fonctionnalité de votre programme. Comme vous venez de le remarquer, la mise à jour de la table mémoire ne s'est pas faite. La valeur a été supprimée mais la table ne le sait pas. Nous allons remédier à ce problème.

Le code de réaffichage de la table mémoire existe déjà (ex : dans la zone de **prise de focus** de la fenêtre **départ**) nous allons donc ré-exécuter un traitement existant.

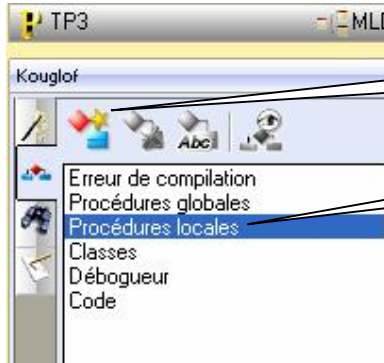
Sous la ligne « Info("La suppression est effective") » inscrivez la commande suivante :

```
ExécuteTraitement(depart, trtPriseFocus)
```

Cette commande fait rejouer un traitement existant.

Testez et vérifiez la cohérence de votre projet.

Maintenant **nous avons à calculer le solde** (Débit – Crédit), pour ce faire nous allons créer une procédure qui scanner le fichier et fera les calculs pour nous.



2) Cliquez sur **Nouvelle procédure** et nommez là **calcsolde**

1) Choisissez **Procédures locales** dans la zone de sélection du bas d'écran



Remarques : Les procédures locales ne sont vues que par les objets de la fenêtre, les procédures globales sont actives pour tous les éléments du projet

Maintenant, vérifiez bien que vous vous trouvez dans la zone **code** de la procédure **calcsolde**.

Le code doit parcourir le fichier Mouvement, affecter le contenu de **débit** dans une variable, le contenu de **crédit** dans une autre et cela jusqu'à la fin du fichier et ensuite affecter la différence entre le débit et le crédit au champ **solde**.

Voici le code de la procédure :

PROCEDURE calcsolde()

sdebit,scredit sont des réels=0 // on affecte la valeur 0 aux deux variables

HLitPremier(Mouvement, IDMouvement)

TANTQUE PAS H.endehors

 Sdebit+=Mouvement.Dépense // += signifie

Sdebit=Sdebit+Mouvement.Dépense

 Scredit+=Mouvement.Recette

 HLitSuivant(Mouvement, IDMouvement)

FIN

solde=sdebit-scredit

Le code est suffisamment simple pour ne pas avoir à l'expliquer. La question qui se pose est où lancer **calcsolde**. Comprenez qu'il nous faut activer cette procédure aux mêmes endroits où l'on a activé le rafraîchissement de la table mémoire.

Je vous laisse modifier les zones de code en conséquence (dans le **code** de la fenêtre **départ**). N'oubliez pas de tester la cohérence de votre projet.

Il ne nous reste plus qu'à fabriquer l'état de sortie. Choisissez **Fichier / Nouveau / Etat**. Choisissez un état de type **Tableau**. En cliquant sur **Suivant** WinDev vous demande la source de donnée, il vous faut préciser que ce sont des enregistrements provenant d'un fichier Hyper file. Sélectionnez le fichier **Mouvement**, la clé de parcours est un identifiant qui sert pour donner l'ordre de tri. Continuez à appuyer sur **Suivant**, je vous laisse découvrir les questions posées et à vous de prendre les bonnes décisions. Vous savez lire, donc vous prendrez les bonnes décisions ! Je vous demande juste d'appeler l'état **Etatmouv**.



Remarques : Les détails du choix du style, de la mise en forme sont à votre discrétion, faites comme bon vous semble. N'oubliez pas que le client n'a pas les mêmes goûts graphiques que vous, donc faites sobre. Eviter les styles « **Noir sur fond Noir** » « **Rose sur fond Vert** » et autres singularités visuelles qui feront penser à l'utilisateur qu'il devient déficient visuel.

Une fois votre état fini et enregistré, entrez dans la zone **code** du bouton **Imprime** :

```
iAperçu(i100) // On enverra l'état à l'écran avec un zoom de 100 %
iImprimeEtat(Etatmouv) // Génération de l'état et utilisation des paramètres définis
par iAperçu
```



Remarques : Les Commandes WinDev sont classées. Celles qui commencent par H sont des commandes d'accès aux fichiers, celles qui commencent par i sont des commandes de pilotage d'état.



N'oubliez pas de **refaire ce TP plusieurs fois**, le but étant de se passer du Guide papier et d'apprécier la facilité avec laquelle on peut travailler avec WinDev.

Cours WinDev Numéro 4



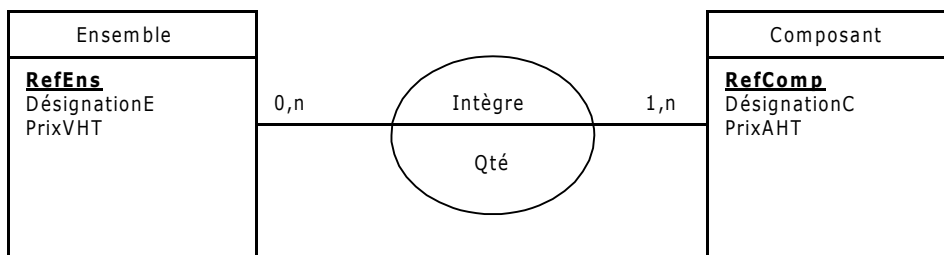
Objectifs : Travailler avec plusieurs fichiers de données

Création d'un MCD,
Gestion des champs indexés,
Manipulation de données,
Liaison des données,
...

Pré requis : Cours WinDev Numéro 1, 2 et 3

L'objectif de ce cours est de vous familiariser avec l'utilisation des fichiers liés.

Nous allons créer une mini GPAO (**G**estion de **P**roduction **A**ssistée par **O**rdinateur). Vous travaillez pour un assembleur informatique, son processus de production est le suivant : Il reçoit les différentes pièces détachées (disque dur, mémoires, cartes mères...) et assemble ces différentes pièces pour en faire un modèle fini. Comme vous pouvez le percevoir, le modèle conceptuel travaillera avec 2 entités (Ensemble fini et composants). Voici une représentation du MCD :



Remarque : Je n'ai pas besoin de vous rappeler que les propriétés soulignées et en gras désignent les identifiants de l'entité, donc je ne le fais pas.

Vous allez commencer par créer un nouveau projet nommé TP4. Dans l'Assistant, vous sélectionnerez votre thème préféré, confirmerez le choix de création d'analyse et finirez le processus de l'assistant. Un nouvel assistant de **Création d'analyse** va apparaître. Vous pouvez ensuite valider les différentes fenêtres de l'Assistant jusqu'à arriver à l'Assistant de **Création de fichier**.

Restez sur **Créer une nouvelle description de fichier**, donnez **Ensemble** comme

Nom et **Ordinateurs assemblés** comme Libellé. Pensez à « décocher » **Le fichier possède un identifiant automatique** puisque nous avons **Refens** qui sera notre identifiant.

Gardez **Hyper File Réseau ou Monoposte** comme **Type de base**

de données et ne laissez coché que l'option **une fenêtre en mode table** dans la dernière partie de l'assistant.

Maintenant la fenêtre suivante apparaît c'est elle qui va vous permettre de saisir toutes les propriétés de votre entité. Remplissez-la pour qu'elle ressemble à ceci :

Description des rubriques d'un fichier

Ensemble HF

0 rubriques 9 octets ☐ Afficher dans l'ordre physique

Clé	Nom	Description	MéTaType

☒ Clé
 ☐ RAD/IHM
 ☐ Notes
 ☐ Avancé

☒ Non clé
 ☐ Clé unique
 ☐ Clé avec doublons

Paramètres de l'index et de la recherche pour clé texte

☐ Sensible à la casse
☐ Sensible à l'accentuation
☐ Sensible aux espaces, ponctuation et car. spéciaux

Infos RAD/IHM

Type de champ

Editer les infos RAD

Description des rubriques d'un fichier

Ensemble HF

3 rubriques 116 octets ☐ Afficher dans l'ordre physique

Clé	Nom	Description	MéTaType
	RefEns	Référence de l'ensemble	<Type : Texte>
	DésignationE	Désignation de l'ensemble	<Type : Texte>
	PrixVHT	Prix de Vente Hors Taxes	<Type : Monétaire>

☒ Clé
 ☐ RAD/IHM
 ☐ Notes
 ☐ Avancé

☐ Non clé
 ☒ Clé unique
 ☐ Clé avec doublons

Paramètres de l'index et de la recherche pour clé

☐ Sensible à la casse
☐ Sensible à l'accentuation
☐ Sensible aux espaces, ponctuation et car. spéciaux
☐ Valeur NULL par défaut

Sens de parcours

☒ Ascendant
 ☐ Descendant

Données personnalisées

Rubrique

Type : Texte Taille : 15

Sous type : Chaîne

Tableau : Dimension 2 Taille réelle

Valeur par défaut :

Editer les infos RAD

Pour **RefEns**, il faut être en **Clé unique** et **Texte** taille **15**, **DésignationE** est en **Texte** taille **80** et **PrixVHT** est en **Monétaire** Windev.

Une fois remplies, cliquez sur **Ok** et retournez sous l'éditeur de Windev. Enregistrez.

Attention un écran va vous demander si vous voulez créer une fenêtre ou allez sous l'éditeur, cliquez sur **Allez directement dans l'éditeur de Windev** pour rester dans l'analyse.

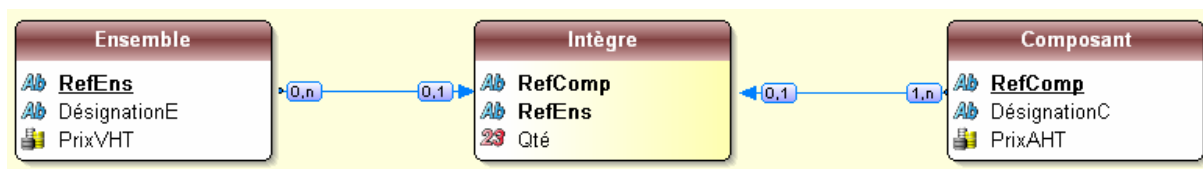
Si vous regardez bien votre éditeur d'analyse en haut vous devez découvrir une barre d'outils comme celle-ci :



Voyez les utilisations des différentes icônes en passant le curseur de la souris dessus.

Comme vous êtes dégourdis et intelligents je vous laisse finir de concevoir le modèle !

Voilà à quoi vous devez arriver :



Vous venez donc de créer le **Modèle Logique de Données (MLD)**. Enregistrez.

Attention : N'oubliez pas que ce n'est pas parce que l'analyse est créée que les fichiers sont physiquement créés sur votre disque dur... Allez dans le code du projet « Projet / Code du projet » et inscrivez la séquence suivante dans la zone « Initialisation de TP4 » :

```

HCréationSiInexistant(Composant)
HCréationSiInexistant(Ensemble)
HCréationSiInexistant(Intègre)
HGèreIntégrité("","*", hCardinalité+hEnModification+hEnSuppression, Faux)

```

La dernière ligne indique à WinDev de ne pas se soucier de gérer l'intégrité référentielle, nous le ferons nous-même.

Maintenant que l'analyse est créée, nous allons commencer à construire notre application. Choisissez **Fichier / Nouveau / Fenêtre**.

Créez une fenêtre nommée **Menu** qui sera la première fenêtre du projet. Faites en sorte qu'elle ait les caractéristiques suivantes :

Onglet **Général** :

Nom logique : Menu

Description : Première fenêtre du projet

Titre : Bienvenue dans la Mini Gp

Onglet **IHM** :

Taille : Taille 640*480

Onglet **Détail** :

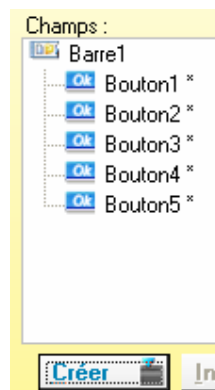
Type de fenêtre : Mère Mdi

	Remarque : Une fenêtre mère MDI est obligatoirement la première fenêtre d'un projet WinDev. Cette fenêtre permet d'afficher toutes les fenêtres de l'application. Une fenêtre mère MDI a les caractéristiques suivantes (non modifiables) : bords modifiables, bouton d'icônisation, d'agrandissement, menu système, barre de titre, barre d'icônes, possibilité d'avoir un menu déroulant et des barres outils. Dans une fenêtre mère MDI, seuls les champs situés dans la zone "barre d'icônes" peuvent être en saisie. En dehors de cette zone, aucun champ ne peut être en saisie, aucun clic souris n'est actif.
---	---

Nous allons placer une barre d'outils dans cette fenêtre **départ**, pour cela cliquez sur

l'objet **Créer une barre d'outils**  et placez le dans le bord supérieur gauche de la fenêtre. Faites un **clic droit** sur ce nouvel objet et allez dans **Description**. Nous allons placer 5 boutons dans cette barre d'outils (un pour saisir les nouveaux composant, un pour créer des ensembles, un pour imprimer la liste des composants, un pour imprimer l'ensemble et ses composants, le dernier pour quitter l'application).

Vous allez donc cliquer sur le bouton **Créer** puis choisissez **Bouton**. Le résultat obtenu devrait ressembler à ceci :



Cliquez sur **Bouton1**, puis sur le bouton **Editer**, dans cette fenêtre **Description d'un bouton** saisissez **nc** dans la zone **Nom**, supprimez le libellé par défaut, cliquez sur **Catalogue...** pour la zone **Image**.

Le catalogue apparaît, sur la gauche choisissez seulement **16*16** (nous n'aurons que des petites images), dans la zone **Recherche** frappez « nouveau » puis appuyez sur le bouton **Rechercher**.

Prenez le 4^{ème} bouton de la 4^{ème} ligne puis validez. Dans l'onglet **Aide** de la fenêtre description, dans la zone **Bulle d'aide** inscrivez **Saisie d'un nouveau composant**

Pour les boutons suivants, faites de même en suivant les instructions suivantes :

Boutons	Nom	Bulle d'aide
Bouton2	Créer	Créer des ensembles
Bouton3	ImprimeC	Imprimer les composants
Bouton4	ImprimeE	Imprimer les ensembles
Bouton5	Fermer	Quitter l'application

Vous choisirez les icônes les plus en phase avec le but du bouton.



Remarque : Si vous avez des difficultés pour placer les boutons (chevauchement par exemple), allez dans leur onglet **Détail** et affectez leurs les dimensions suivantes : largeur 32, hauteur 24.

Voici un exemple de ce que devrait être la barre d'outils :



Nous allons créer les 4 fenêtres filles nécessaires pour faire fonctionner notre application (fenêtre filles car elles doivent s'exécuter à l'intérieur de la fenêtre mère. Pour le bouton **Quitter**, je vous laisse mettre le code Correspondant, vous n'avez plus besoin de mon aide, sinon retour aux TP précédents.

Pour la première fenêtre associée au bouton , respectez les consignes ci dessous :

Onglet **Général de la description**

Nom logique : Gcompo

Description : Gestion des composants

Titre : Gestion des composants

Le code d'ouverture de la fenêtre fille sera :

OuvreFille(Gcompo)

Onglet **IHM**

Largeur : 500

Hauteur : 350

Position : Relatif à la mère

Onglet **Détail**

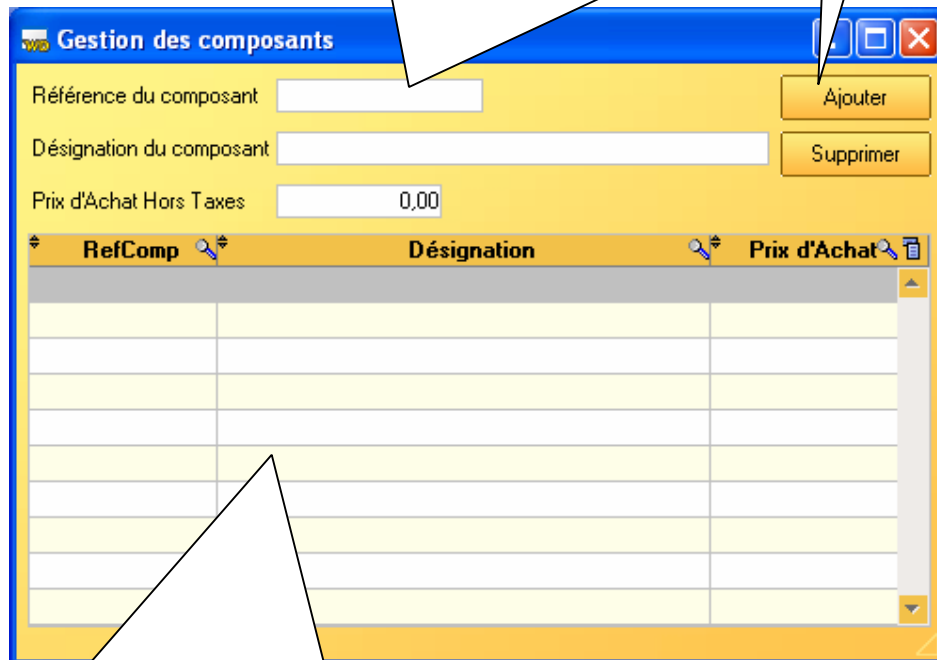
Type de fenêtre : Fille Mdi

Voici à quoi elle devrait ressembler :

Ces 3 champs sont liés avec les champs du fichier Composant

(faites-les glisser depuis l'onglet  à droite de l'écran). Si certains ont des difficultés : retour au TP3. Faites en sorte que RefComp ait un masque de saisie **TOUT EN MAJUSCULES**.

Créez 2 boutons : **BAjout** et **BSup**



Nom de la table mémoire : Table1
 RefComp : type texte longueur 15
 Désignation : type texte longueur 80
 PrixAchat : type numérique (Titre : Prix d'Achat)
 Choisir dans l'assistant **Je veux remplir moi-même la table**
 Mettez la table en **Affichage seulement** (sinon la saisie ne serait pas validée)
 voir TP3

Maintenant analysons le comportement que devra avoir cette fenêtre à l'ouverture : Elle devra parcourir le fichier Composant et afficher les tuples dans la table mémoire s'il y en a.

Le bouton **Ajouter** devra :

- . Inscrire la valeur des 3 champs dans le fichier correspondant en interceptant une erreur de doublonnage si l'utilisateur saisi 2 fois une référence existante,
- . Il devra aussi mettre à blanc les 3 champs pour préparer une nouvelle saisie,
- . Réafficher la table mémoire (supprimer le contenu existant et re-écrire avec le contenu du fichier),
- . Et enfin avertir l'utilisateur que tout s'est bien passé.

Le bouton **Supprimer** devra :

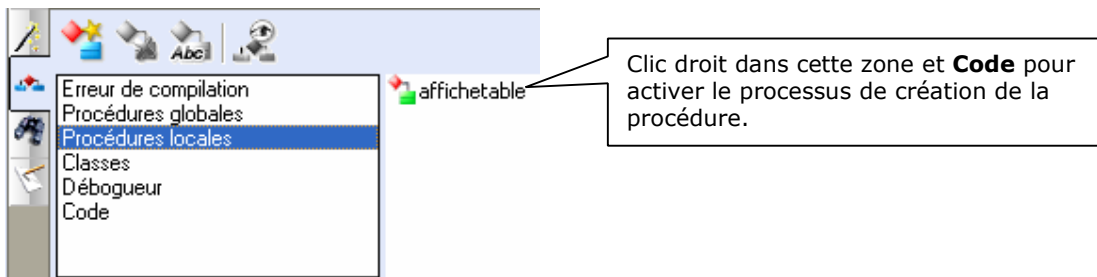
- . Supprimer physiquement la ligne pointée dans la table mémoire
- . Réafficher la table mémoire

Voici les différents codes (Ils ont déjà été étudiés à la leçon 3) : Nous allons créer une procédure locale **affichetable** qui aura pour rôle de réafficher la table mémoire (supprimer le contenu existant et re-écrire avec le contenu du fichier) :

```
TableSupprimeTout(Table1)
HLitPremier(Composant,RefComp)
TANTQUE H.trouve ET PAS H.EnDehors
    TableAjoute(Table1,Composant.RefComp+TAB+Composant.DésignationC+TAB+
Composant.PrixAHT)
    HLitSuivant(Composant,RefComp)
FIN
```

Cette séquence devrait vous être familière donc je ne vous la commente pas, sinon retour au cours 3.

Rappel pour faire une procédure locale cliquez sur « Procédures locales » en bas de l'écran :



Initialisation de Gcompo (la fenêtre) :

```
// Appel de la procédure affichetable
affichetable()
```

Maintenant on est sûr que dès que cette fenêtre s'ouvrira, la table mémoire sera en phase avec le contenu du fichier.



Remarque : la table mémoire vous apportera plus de souplesse et une gestion plus fine des enregistrements et des lignes dans certains cas.

Voici le code du bouton **BAjout** :

```
EcranVersFichier() // Transfère les valeurs contenues dans les champs de la fenêtre
// dans la zone de structure du fichier
HAjoute(Composant) // Passe la structure physiquement dans le fichier
RAZ(Vrai)// Efface les valeurs contenues dans les champs liés pour saisir un nouveau
composant
affichetable()
```


Voici le code du bouton **BSup** :

```

HLitRecherche(Composant,RefComp,RefComp) // Recherche dans la table Composant,
// sur la rubrique RefComp, la valeur contenue dans RefComp de Table1
SI HTrouve ALORS // Si la valeur est trouvée alors
    HSupprime(Composant) // On supprime physiquement la
    ligne dans le fichier Composant
    Info("Suppression réussie") // On informe l'utilisateur de la
    réussite de la suppression
    affichetable()
SINON // Sinon, on n'a pas trouvé ?!!
    Info("Il y a un boulon dans le potage") //
FIN

```

Enregistrez la fenêtre et testez votre projet sans oublier de définir qu'elle est la 1^{ère} fenêtre du projet (**Menu**), normalement les comportements doivent être cohérents. Dans le cas contraire, reprenez le support et vérifiez que vous n'avez pas oublié quelques instructions.

Nous allons maintenant faire la fenêtre la plus intéressante au niveau intellectuel, celle qui gère l'assemblage d'un ordinateur.

Créez une nouvelle fenêtre (Fichier / Nouveau / Fenêtre). Prenez une fenêtre vierge et enregistrez la sous **Gensemble**.

N'oubliez pas d'en faire une fenêtre fille (cf. : étapes identiques à la fenêtre **Gcompo**) et de mettre le code d'ouverture de cette fenêtre dans le deuxième bouton de la barre d'outils de la fenêtre **départ**.

Voici la fenêtre telle quelle devrait être :

The screenshot shows a window titled "Gestion des ensembles". It features a yellow background and a blue title bar. The interface includes the following elements:

- Input Fields:**
 - Field 1: "Référence de l'ensemble" (with a small '1' next to it).
 - Field 2: "Prix de Vente HT" (with a small '2' next to it, showing "0,00").
 - Field 3: "Désignation de l'ordinateur" (with a small '3' next to it).
- Navigation Buttons:** A set of buttons labeled 4, 5, 6, 7, 8, 9, 10, likely for navigating through a list or table.
- Table:** A table with the following columns: "RefComp", "Désignation", "Prix de Vente", and "Qté". The table has several empty rows for data entry.
- Bottom Buttons:** Three buttons at the bottom labeled "Nouveau" (with a small '11' next to it), "Valider" (with a small '12' next to it), and "Supprimer" (with a small '13' next to it).

Objet N°	Nom	Rôle
1	RefEns	Lié à Ensemble.RefEns (TOUT EN MAJUSCULES)
2	PrixVHT	Lié à Ensemble.PrixVHT
3	Désignation	Lié à Ensemble.Désignation
4	BPremier	Bouton nous positionnant sur le premier enregistrement du fichier Ensemble
5	BPrecedent	Bouton nous positionnant sur l'enregistrement précédent
6	BSuivant	Bouton nous positionnant sur l'enregistrement suivant
7	BDernier	Bouton nous positionnant sur le dernier enregistrement du fichier Ensemble
8	Table1	Table mémoire contenant des rubriques similaires à la structure de Composant
9	BAjout	Bouton nous permettant d'ajouter un composant à l'ensemble
10	BSupprime	Bouton nous permettant de supprimer un composant à l'ensemble
11	BNouveau	Bouton nous permettant de créer un nouvel ensemble
12	BValide	Bouton nous permettant de valider un ensemble
13	BSupp	Bouton nous permettant de supprimer un ensemble

Analyse des éléments de la fenêtre **Gensemble**.

A l'ouverture de la fenêtre dans le code d'initialisation, nous devons nous positionner sur le premier ensemble (s'il existe) et rechercher tous ces composants constitutifs.

Voici comment faire :

Il nous faut lire la référence de l'ensemble (RefEns), parcourir le fichier de liaison (Intègre) sur la clé RefEns, tant que l'on trouve Ensemble.RefEns=Intègre.RefEns il nous faut chercher dans la table composant la clé de liaison (Intègre.RefComp = Composant.RefComp) et récupérer Intègre.Qté. Alors relisez en ayant l'analyse en tête.

Nous allons créer une fonction (en fait une procédure, WinDev ne faisant pas de différence entre fonction et procédure) qui, prenant en paramètre le code RefEns, nous remplit la table. Facile et efficace.

```

PROCEDURE remplir (code)
TableSupprimeTout(Table1)
FichierVersEcran(Gensemble) // on affecte aux champs de la fenêtre les données liées
HLitRecherche(Intègre,RefEns,code) // On fait rechercher dans le fichier Intègre une
valeur de RefEns égale à code
TANTQUE HTrouve ET PAS HEnDehors
    HLitRecherche(Composant,RefComp,Intègre.RefComp) // A vous de trouver
    TableAjoute(Table1,Composant.RefComp+TAB+Composant.DésignationC+TAB+
Composant.PrixAHT+TAB+Intègre.Qté)
    HLitSuivant(Intègre,RefEns)
FIN

```

En fait, comme vous pouvez le voir, nous venons de « jongler » avec 3 fichiers sans trop de difficultés. Regardons le code d'initialisation de la fenêtre :

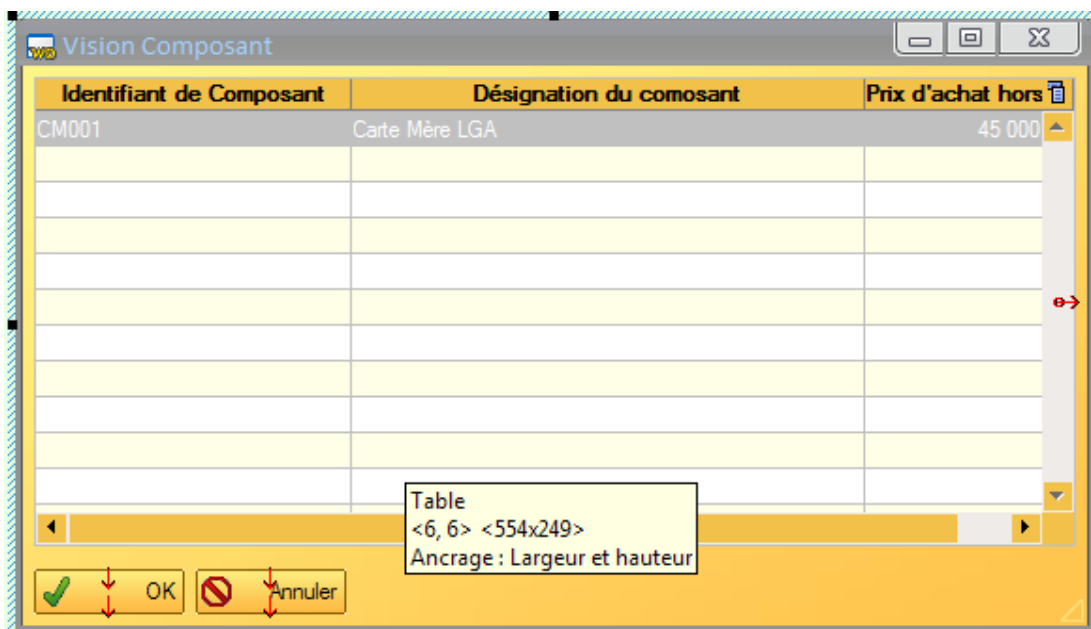
```
HLitPremier(Ensemble,RefEns)
SI HNbEnr(Ensemble)>0 ALORS // si le fichier Ensemble contient au moins un
enregistrement
    remplir(Ensemble.RefEns)
SINON
    Info("Le fichier est vide")
FIN
```

Voyons le code nécessaire aux boutons 4, 5, 6 et 7 (pour les icônes, faites un tour du côté des flèches en 16x16). Je vous donne le code du bouton 4, vous êtes Suffisamment aguerri pour pouvoir créer le code des boutons 5, 6 et 7.
(Si vous ne savez pas... retour, au TP 3).

Nous allons traiter le bouton 9. Ce bouton doit nous permettre de sélectionner dans le fichier **Composant** le composant que nous voulons insérer dans la table mémoire. Nous allons voir si WinDev peut nous générer la fenêtre qui va bien. Allez sur Fichier / Nouveau / Fenêtre. Cherchez une fenêtre nommée **Vision plus** et validez.

Dans l'Assistant qui arrive, décochez **Avec code de rafraîchissement automatique**, puis cliquez sur **Suivant** 2 fois, ensuite cliquez sur **Composant**, cliquez encore sur **Suivant**, vérifiez que tous les champs soient cochés, cliquez encore sur **Suivant** 2 fois. Prenez une taille 640*480. Puis saisissez **selection** comme **Nom**. Enfin cliquez sur **Terminer**.

Vous avez maintenant une fenêtre fonctionnelle dans laquelle on peut supprimer le bouton **Nouveau** (on peut utiliser la création de composants avec notre 1^{er} bouton du menu). Réduisez-la pour quelle soit plus petite. Voici à quoi elle pourrait ressembler :



Regardons comment nous pourrions modifier le code pour qu'il nous soit utile. Allez dans le code du bouton **OK**. Vous pouvez voir que la fenêtre retourne **vrai** si on clique sur **OK**, faux sur **Annuler**. Qui dit bandeau déplacé dit structure mémoire du fichier contenant les valeurs recherchées ! Donc, nous n'aurons qu'à lancer la fenêtre et tester si elle retourne **vrai**.

Fermez la et retournez sur **Gensemble** dans le code du bouton **BAjout**. Saisissez la séquence de code suivante :

```
resultat est un booléen=Faux
resultat=Ouvre(selection)
SI resultat=Vrai ALORS // la fenêtre nous a renvoyé vrai, donc le bandeau a été
déplacé
    TableAjouteLigne(Table1,Composant.RefComp,Composant.DésignationC,Compo
sant.PrixAHT)
FIN
```

Méditez sur la différence entre **tableajoute** et **tableajouteligne**(l'aide, touche F1).

Sauvegardez et exécutez la fenêtre, normalement le clic sur le bouton **BAjout** lance bien la fenêtre **selection** qui nous retourne vrai ou faux selon le bouton cliqué.

Pour saisir la **Qté**, cliquez dans la colonne pour saisir le nombre souhaité. N'oubliez pas de vérifier que la Table est **En saisie** et d'avoir seule la rubrique Qté **En saisie / actif**, les autres restant en mode **Affichage seulement**.

Pour le bouton 10 (**BSupprime**), la séquence de code a déjà été étudiée lors du TP3. Pour le bouton Nouveau (**BNouveau**), il vous faut programmer un traitement qui efface les champs 1, 2, 3 et la table mémoire, vous savez faire. Le bouton Supprime (**BSupp**) va nous obliger à scanner 2 tables et supprimer les références « Ensemble » recherchées. Il faut donc prendre la valeur du champ 1 (RefEns), parcourir la table Intègre sur le champ RefEns et supprimer tous ceux égaux au champ RefEns. Et ensuite le supprimer dans la table Ensemble.

Voici une séquence de code qui devrait être efficace :

```
SI OuiNon("Voulez vous vraiment supprimer cet enregistrement ?")=Oui ALORS
    HLitRecherche(Intègre,RefEns,RefEns)
    TANTQUE HTrouve ET PAS HEnDehors
        HSupprime(Intègre)
        HLitSuivant(Intègre)
    FIN
    HLitRecherche(Ensemble,RefEns,RefEns)
    SI HTrouve ALORS
        HSupprime(Ensemble)
        Info("La suppression est réussie")
    HLitPremier(Ensemble,RefEns)
    FIN
FIN
```

La séquence du bouton **Valider** va être un peu plus rude. En fait, lorsqu'un nouvel ensemble est créé ou modifié par l'adjonction d'un nouveau composant, il va nous falloir faire plusieurs actions différentes.

Il faut supprimer l'ensemble existant dans le fichier pour le recréer avec ses nouveaux composants, donc on commence par une phase de suppression (voir code ci-dessus) et on achève par une phase d'insertion dans le fichier.

Le code est long mais n'est pas complexe, prenez le temps de l'étudier.

i est un entier

```
HLitRecherche(Ensemble,RefEns,RefEns)
SI HTrouve ALORS // l'enregistrement existait dans le fichier, on va le supprimer
    HLitRecherche(Intègre, RefEns, RefEns)
    TANTQUE HTrouve ET PAS HEnDehors
        HSupprime(Intègre)
        HLitSuivant(Intègre)
    FIN
    HSupprime(Ensemble)
FIN
// Maintenant inscrivons l'ensemble dans le fichier
Ensemble.RefEns=RefEns
Ensemble.DésignationE=DésignationE
Ensemble.PrixVHT=PrixVHT
HAjoute(Ensemble)

// Remplissons le fichier Intègre
POUR i=1 A TableOccurrence(Table1) // Pour i=1 aux max de la table mémoire
    Intègre.RefComp=Table1.RefComp[i] // [i] indique l'indice de la ligne
    Intègre.RefEns=RefEns
    Intègre.Qté=Table1.Qté[i]
    HAjoute(Intègre)
FIN
Info("L'ajout c'est bien passé")
```

N'oubliez pas de sauvegarder votre œuvre.



Le troisième bouton de la barre d'outils sert à imprimer la liste des composants, vous avez déjà étudié la procédure dans le TP 3.

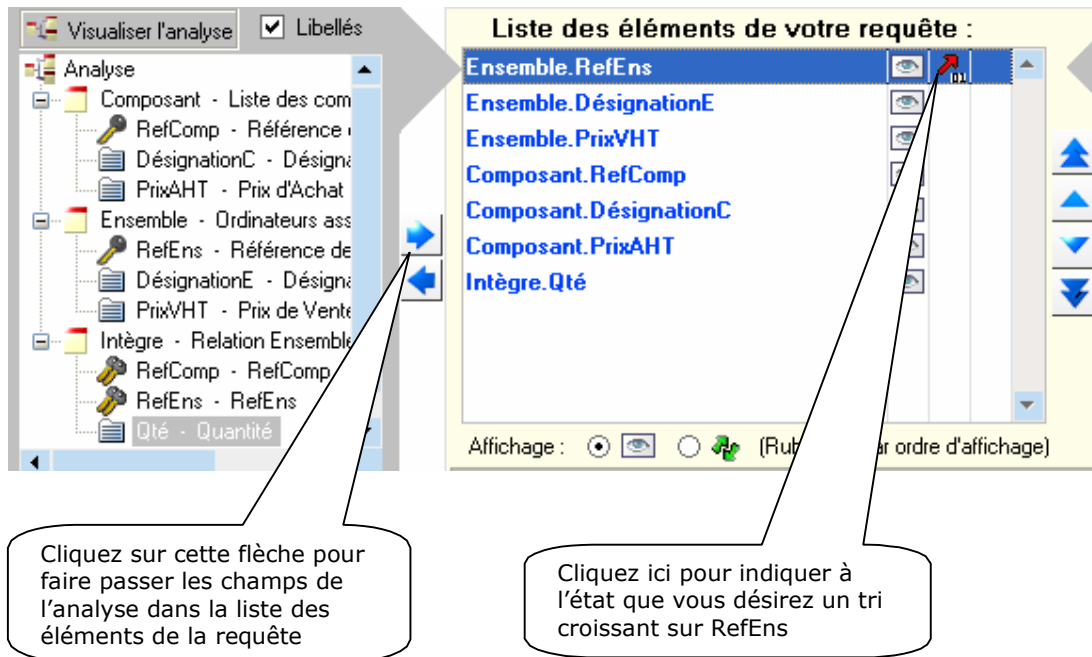
Nous allons nous laisser guider par l'Assistant pour faire un état avec des données provenant des 3 tables. Allez dans **Fichier / Nouveau / Etat**.

Choisissez ensuite un type d'état **tableau**, cliquez sur **Suivant**.

Vérifiez que les données proviennent bien d'une requête et cliquez sur **Suivant**.

Faites lui comprendre que les données proviennent d'une requête que vous voulez définir maintenant, cliquez sur **Suivant**.

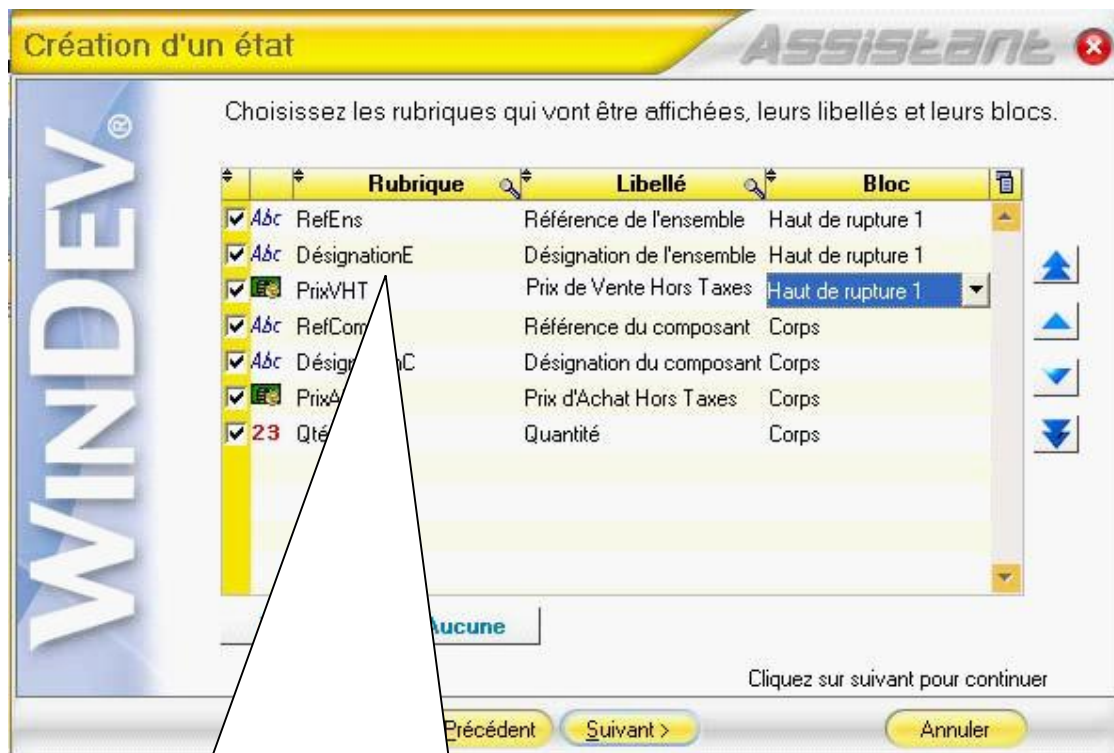
Maintenant vous devez avoir cette fenêtre à l'écran :



Cliquez ensuite sur **Suivant**. A la question « Voulez-vous des ruptures dans l'état » répondez **Oui** et cliquez sur **Suivant**.

La fenêtre suivante vous demande de sélectionner la rupture, vérifiez que **Référence de l'ensemble** soit bien coché, cliquez sur **Suivant**.

Vous devriez avoir maintenant une fenêtre qui ressemble à ceci :



Passez ces 2 champs appartenant à Ensemble en Haut de Rupture. Les éléments en haut de rupture apparaissent une seule fois. Les éléments dans le corps apparaissent autant de fois qu'il y en a dans le fichier

Cliquez ensuite sur **Suivant**. La fenêtre suivante vous demande quels sont les champs numériques sur lesquels vous voulez effectuer un calcul, essayer de faire en sorte quelle ressemble à ceci :



Cliquez sur **Suivant**. Les choix suivants concernent les formats du papier, laissez les par défaut et cliquez sur **Suivant**.

Choisissez un gabarit et **Suivant**. Nommez cet état **Iensemble** (attention, le i est en minuscule), comme titre « Etat des ensembles ». cliquez sur **Terminer**.

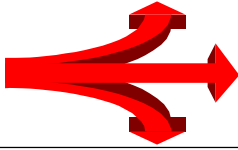
Comme votre état est plus large que la feuille, passez en mode paysage. Votre état apparaît maintenant dans l'éditeur d'état. Si vous voulez faire quelques retouches, c'est maintenant.

Il ne vous reste plus qu'à activer l'état dans le bouton de la barre d'outils avec la séquence de code suivante :

```
iAperçu(i100)  
iImprimeEtat(Iensemble)
```

Voilà, ce modeste "Mini GP" est fini. Vérifiez que tous les boutons fonctionnent, que les traitements sont cohérents.

Cours WinDev Numéro 5



Objectifs : Études des Sockets, communication répartie
Rappels de notions réseaux, Tcp/ip, Ports.

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 nous apprendre à faire discuter 2 (ou plusieurs)

Ordinateurs entre-deux. Nous allons employer les Sockets et les

threads. Commençons par définir ces 2 termes :

Les Sockets : 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. Un socket est constitué 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 du 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.

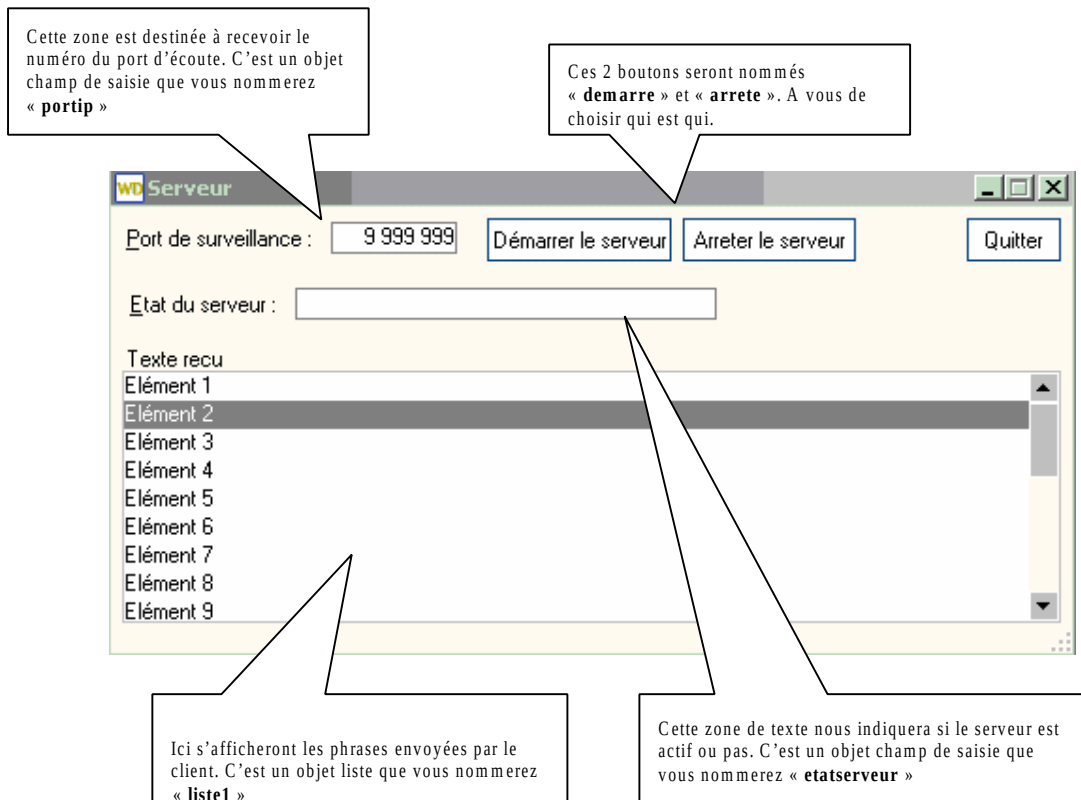
Un thread peut être perçu comme un flot de contrôle à l'intérieur d'un processus. Dans notre cas il joue le rôle d'interface entre les 2 ordinateurs, interceptant les données transmises.

Notre exercice consistera à créer 2 exécutables, un client et un serveur. Le serveur sera en attente de réception de message et le client essaiera 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 TP 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.

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 quelle ressemble à celle ci :



Analysons le fonctionnement du serveur :

1. Il faut lui donner un port à écouter.
 2. Il faut lancer la boucle d'écoute (boucle infinie) et faire en sorte que les événements d'entrées (demande de connexion...) soient traités par des threads.
 3. Une fois la connexion acceptée le texte reçu sera inscrit dans la liste déroulante.
- Nous allons commencer par créer une procédure 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 vérifier si des demandes de connexion se produisent. Comme paramètres cette fonction prend un argument qui est le nom du socket ici "serveur". Vous allez voir où 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 tunnel où 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 exécutée comme un Thread normal appelé "threadcnx" utilisant le paramètre "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 quantum.

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

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("liste1",texte)
    FI
    N
    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 temps que thread. La fonction WinDev SocketLit lit le contenu du canal et le met 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 “démarrer”.

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

Ce code lance l’exécution du serveur. La procédure WinDev SocketCrée utilise plusieurs paramètres :
 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 même le thread “**affiche mes**”.

Voici le code du bouton **arrêter**

```
SocketFerme("serveur")
ETATSERVEUR="Arrêt du serveur"
```

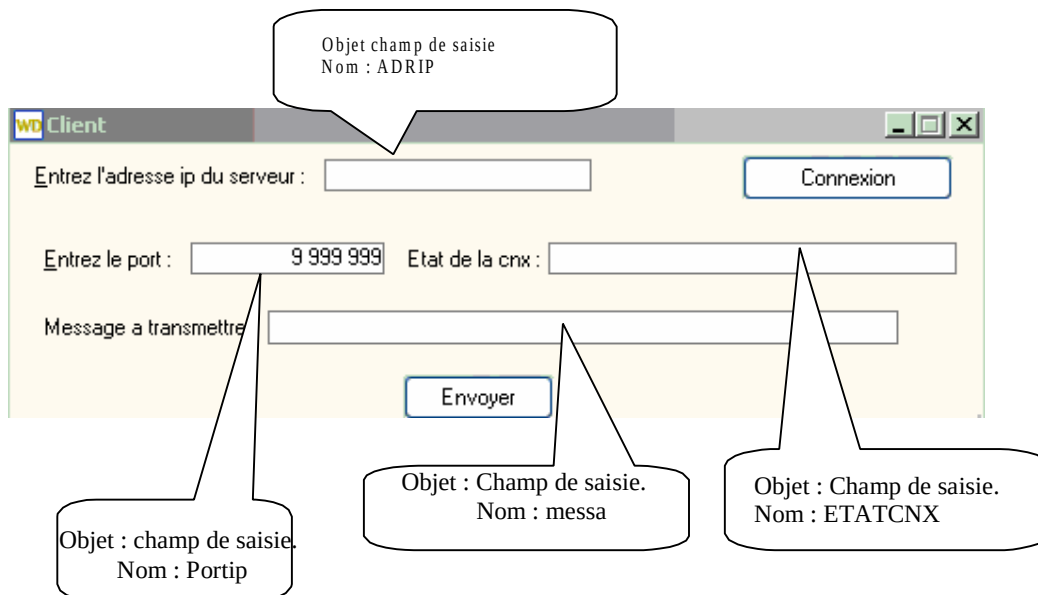
Voilà la partie serveur est maintenant terminée... Il ne vous reste plus qu’à créer l’exécutable (Menu Projet/Créer l’exécutable).

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

Dés à présent vous avez conçu un serveur, il ne nous reste plus qu’à créer le client. Son rôle est d’essayer de se connecter au serveur et de lui envoyer des messages.

Partie Cliente

Pour le client, comme ce doit être une application autonome, il nous faut donc créer un projet, sans analyse ayant pour nom “**Client**”. Ce projet ne contiendra qu’une fenêtre nommée “**Départ**”. Elle ressemblera à 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.

Intéressons nous au bouton connexion. Son rôle est de nous mettre en relation avec la socket du serveur.

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 se connecter à la socket nommée "Serveur, écoutant le port défini (PORTIP), à telle adresse IP (ADRIP). Si tout ce passe bien on écrit "Vous êtes en ligne dans le champ de saisie Etatcnx. Sinon on envoie le message d'erreur.

Maintenant voyons le code du bouton envoyer.

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

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

Voilà tout est fini, vous pouvez compiler, créer l'exécutable 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 êtes en réseau utilisez la bonne adresse IP du serveur

Cours 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 données 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-elles ? 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

Est 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 données MySQL qui est sur Internet ?

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 cette leçon 6 sans maîtriser les Tp précédents. Je ne reviens pas sur les notions abordées dans les TP précédents.

Pour le bon déroulement de ce support téléchargez et installez easyphp que vous trouverez à l'adresse Suivante : <http://www.easyphp.org/>



Attention : Si vous n'êtes pas administrateur de votre ordinateur une librairie de Mysql ne s'installera pas correctement, il s'agit de libmysql.dll, de plus le serveur Mysql s'arrêtera à la moindre tentative de connexion.

Autre point important : Si des erreurs récurrentes se produisent alors que votre code est propre et que vous utilisiez Easyphp, téléchargez la dll libmysql.dll directement chez Mysql.

De plus, allez sur www.pcsoft.fr dans la rubrique téléchargement pour télécharger l'accès natif à Mysql

Avant de commencer, vérifiez qu'Easyphp est actif (Vous devez apercevoir un E noir avec un point rouge clignotant dans la barre des tâches). 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 orientée 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.

Ici vous trouvez 4 champs de saisie nommés : **Numip**, **Login**, **Mdp**, **Etatcnx**. Un bouton nommé **Cnx**. Ces champs sont dans un champ libellé de style « Libellé-Acheval_DoubleB », dont le libellé est : « Connexion au serveur Mysql »

Ici, créez un champ arbre que vous nommerez **listebase**. Vous pouvez l'encadrer par un champ libellé.

Attention, ici c'est une table mémoire que vous nommerez **Table1** dont toutes les colonnes sont invisibles. Dans cette table sera affiché le contenu des tables de votre base de donnée Mysql. L'affichage est activé par un choix de table dans l'arbre **listebase**. Comme nous ne connaissons pas à priori le nombre de colonnes de la table à afficher nous allons employer une grosse ruse Cheyenne. On va créer une quinzaine de colonnes texte nommées c1,c2,c3...c15 et toutes invisible, capito ? Nous créerons l'affichage par programmation plus tard

Insérez 2 boutons un nommé **Requete**, l'autre **Quitter**. Je vous laisse trouver le code du bouton Quitter.

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 données 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.

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 chaque bases, les tables constitutantes. Dans le champ etatcnx nous mettrons un texte nous indiquant le succès de la connexion sinon l'échec.

Verifier que vous avez placé tous les objets et votre fenêtre ressemble à la mienne .Vous l'enregistrez sous le nom de départ .Vous la déclarée comme première fenêtre du projet.

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.

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. Le5eme 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 à 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 les bases.

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")
SINO
N    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 résultat vaut vrai sinon faux.

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

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 hic est que tant qu'elle retourne 0 la lecture de l'enregistrement s'est bien passée. 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).

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 lestables que nous allons créer. Comme vous le remarquerez 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)
FI
N
```

SQLDeconnecte()

Comme vous pouvez le constater cette procédure récupère comme argument une chaîne de caractère (labase) contenant le nom de la base de donnée à 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 SQLDeconnecte 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 vous verrez le tree-view est rempli de l'arborescence bases de données + Tables.

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 chaîne.

La fonction ExtraitChaîne va nous être d'un grand secours. On lui donne la chaîne initiale et on lui dit de nous renvoyer le *Xième* 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 ExtraitChaîne (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 aidemandé 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 1^{er} 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 faitun transfert du contenu de la requête dans la table mémoire.

Sauvegardez et testez votre travail.

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.....

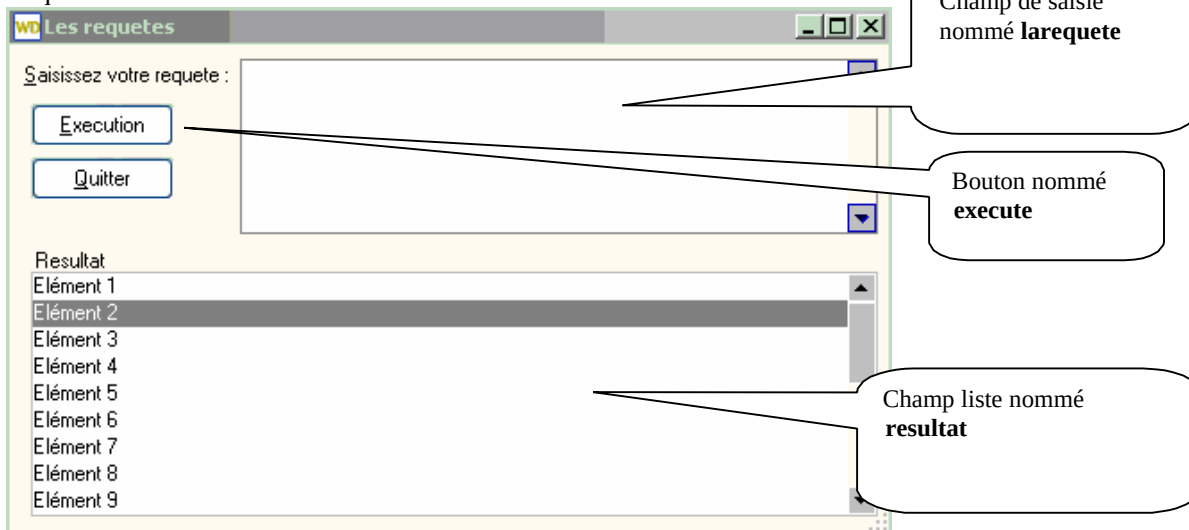
C'est ce que l'on va réaliser maintenant, mais tout d'abord voici le code du bouton requete

```
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 la requête 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.

Pour que les variables passées par le code du bouton requête de la fenêtre « Départ » soient prises en compte par la fenêtre « frequete », il faut saisir dans la zone « déclaration globale de frequete » la ligne suivante :

Procédure frequete (Numip,login,mdp,mabase)

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
```