Développer sous Microsoft Visual Basic 6

7. Le client serveur avec SQL Server

Client-serveur

Historique des méthodes de connexion

En terme de middleware, c'est-à-dire de logiciel de connectivité entre les applications et le serveur de base de données, Sybase et Microsoft ont proposé avec SQL Server 1 un client développé en C : DB-Library.

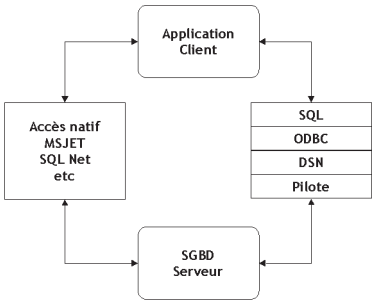
Avec Windows 3.x, Microsoft a développé une méthode universelle et ouverte avec ODBC (Open Data Base Connectivity) qui permet de se connecter à Access et à SQL Server.

Avec DAO (Data Access Objects), Microsoft définit un modèle objet S'appuyant sur ODBC. Le Client devient complètement indépedant de la abse de données. RDO (Remote Data Objects) permer d'accroître les performances de cette méthode.

Visula Basic 6 s'appuie aujourd'hui sur OLE-DB et le modèle objet ADO (ActiveX Data Objects). Avec Visual Studio .Net, les outils De middleware s'appuient sur .Net Framework.

Accès à SQL Server

Il existe deux manières d'accéder aux données sur les machines à base de processeurs Intel compatibles et de systèmes d'exploitation Windows 98-98-Nt Workstation 4. Vous pouvez disposer d'un accès natif, propriétaire et souvent plus rapide. Le système propriétaire de Microsoft repose sur Jet. Il vous permet de vous connecter notamment à une base Access. Ou bien, vous pouvez utiliser ODBC, Open Data Base Connectivity qui vise à une plus grande ouverture et interopérabilité en vous éloignant des couches matérielles. Vous accédez aux ressources par le biais



d'un identifiant logique : le DSN (Data Source Name).

Utiliser le composant Ado

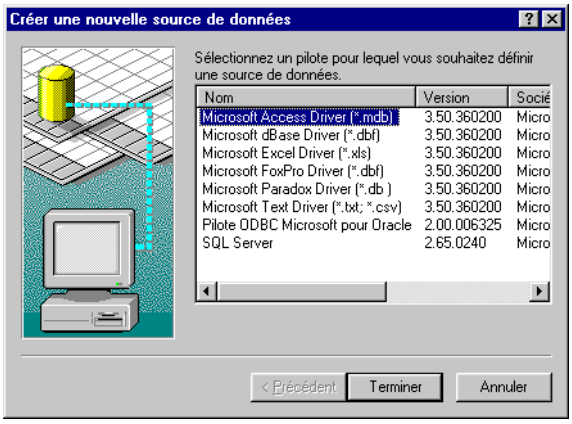
ODBC

A partir du menu **Démarrer**, choisissez **Paramètres | Panneau de configuration.**

Double-cliquez sur l'icône 

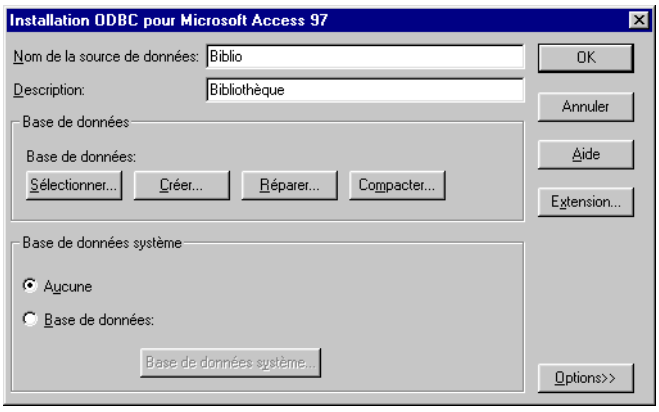
Pour créer votre DSN, vous disposez de deux possibilités : un DSN attaché à l'utilisateur (non accessibles aux autres), un DSN attaché à la machine (accessibles à

tous les utilisateurs).



Cliquez sur le bouton Ajouter... .

Utiliser les Data Environments



La première consiste à déterminer le pilote représentant la nature de la base de données. Dans l'exemple ci-dessous, il s'agit de **Microsoft Access**.

Entrez un alias qui représente la connexion dans le nom de la source de données. C'est le nom que vous emploierez sous VB pour vous connecter. La description explicite sous forme d'un commentaire la source de données.

Dans une seconde phase, cliquez sur le bouton Sélectionner... pour choisir la base de données. Sélectionnez alors la base de données. Dans l'exemple, il s'agit de la base de données Biblio fournie avec la distribution de Visual Basic.

Contenu du fichier UDL

```
[oledb]
```

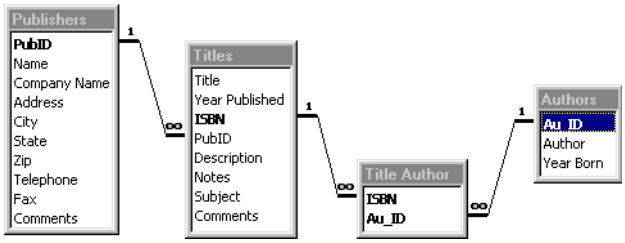
; Everything after this line is an OLE DB initstring

Provider=SQLOLEDB.1;Password="";Persist Security Info=True;User ID=SA;Initial Catalog=Northwind;Data Source=SERVER2\DSFC

Par le code

Connexion en mode SQLOLEDB en définissant les propriétés de connexion

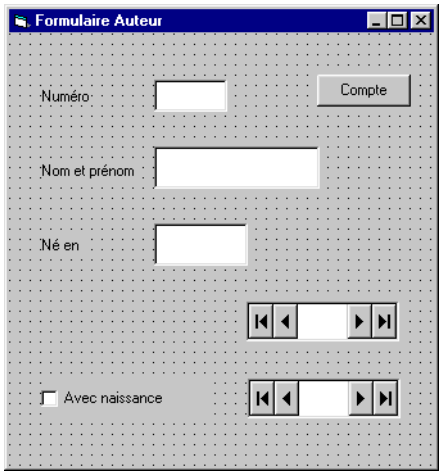
```
' Déclaration des variables
Dim cn As New ADODB.Connection
' Fournisseur OLE-DB.
cn.Provider = «sqloledb»

' Définition des propriétés de connexion
cn.Properties(«Data Source»).Value = «SERVER2\DSFC»
cn.Properties(«Initial Catalog»).Value = «NorthWind»

' Type d'authentification
If optWinNTAuth.Value Then
        cn.Properties(«Integrated Security»).Value = «SSPI»
Else
```




```
        cn.Properties(«User ID»).Value = «sa»
        cn.Properties(«Password»).Value = «»
End If

' Ouverture de la base
cn.Open
```

Connexion en mode SQLOLEDB par un chaîne de connexion

```
' Déclaration des variables.
Dim cn As New ADODB.Connection
```

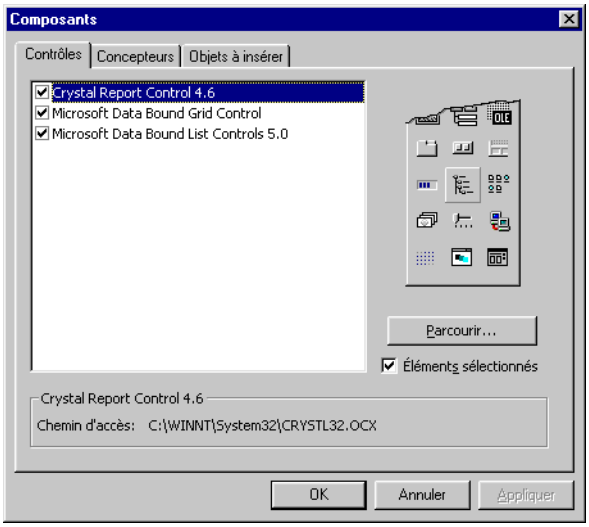


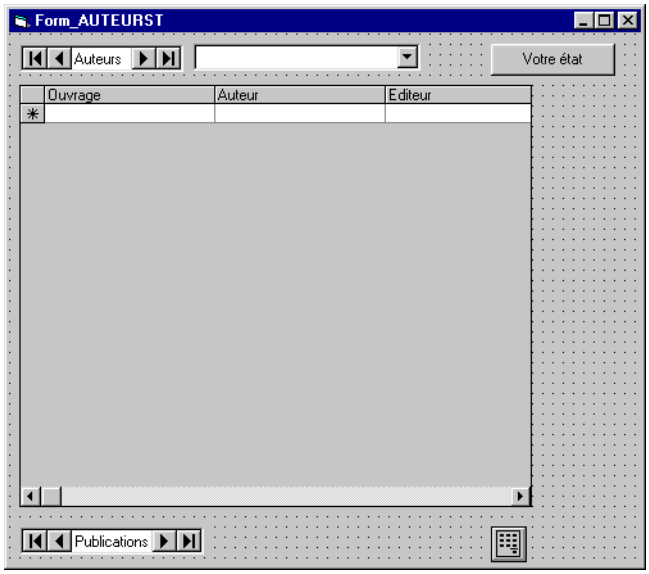
L'exemple

```
Dim provStr As String

' Fournisseur OLE-DB.
cn.Provider = «sqloledb»

' Ouverture de la connexion
ProvStr =
«Server=MyServer;Database=northwind;Trusted_Connection=yes»
cn.Open provStr
```

Utilisation de MSDASQL (via un DSN ODBC)

```
Dim cn As New ADODB.Connection

cn.ConnectionTimeout = 100
' Le DSN Suppose la création préalable d'une connexion ODBC
```





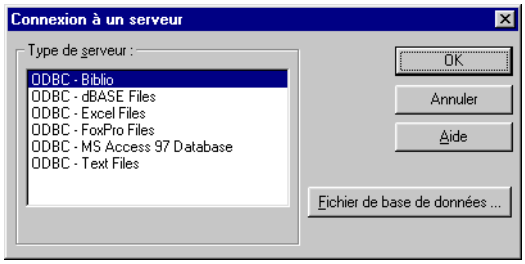

```
cn.Open «MyDataSource», «sa», «MyPassword»
' Autre syntaxe
' cn.Open «DSN=DataSourceName;UID=sa;PWD=Password;»
```

Denis
Szalkowski

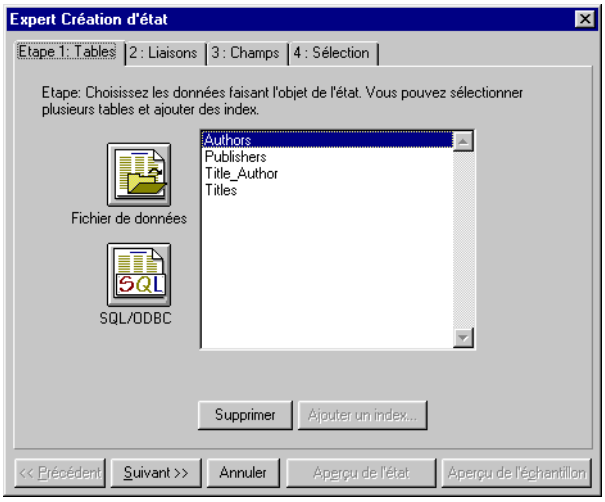
```
cn.Close
```

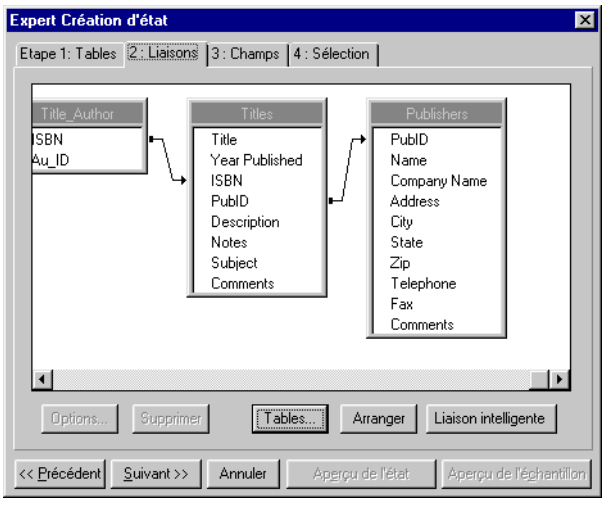
Utilisation de MSDASQL (via ODBC sans DSN)

```
Dim cn As New ADODB.Connection

' Pas de Data Source Name
cn.Open «Driver={SQL
Server};Server=Server1;Uid=SA;Pwd=;Database=northwind»
```

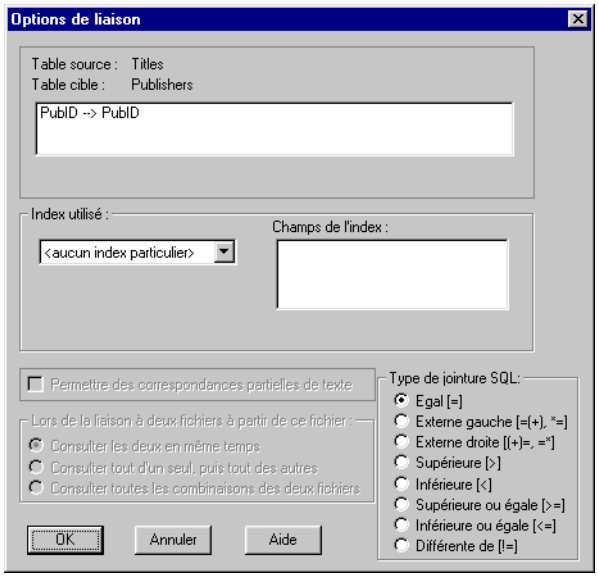



```
cn.Close
```





Le Data Control, contrôle accessible à partir de la boîte à outils, représente le jeu d'enregistrements auquel votre application peut accéder. Il peut s'agir d'une table,



d'un requête sous la forme d'un ordre SQL SELECT en lecture-écriture ou lecture seule. A partir de la boîte à outils, sélectionnez l'outil Data 

et dessinez une surface dans la feuille correspondant à l'emplacement du Data Control.

Au préalable, nommez votre DataControl à l'aide de la propriété **Name** Data_AUTEURS par exemple.

Dans un deuxième temps, vous devez déterminer la nature de la connexion à l'aide de la propriété **Connect**. Pour se connecter à Access via MS Jet, choisissez dans la liste **Access**. Pour se accéder via ODBC, veuillez taper dans la zone **ODBC;DSN=Biblio;[UID=UserName;PWD=Password;]**. Le fait de rentrer en "dur" le mot de passe ne répond à une bonne logique de sécurité. Pour les connexions de type MsJet, sélectionnez directement la base Access par **DataBaseName**.

Si vous choisissez l'un ou l'autre mode, allez préciser la nature de la connexion au niveau de la propriété **DefaultType**. Pour ODBC, prenez **UseODBC**. **UseJet** est réservé à l'emploi de connexions de type MsJet. Pensez aussi au niveau dur **DefaultCursorType** à choisir en mode ODBC à prendre **ODBCCursor**.

Il vous reste encore deux étapes. Précisez le **RecordSetType**. Le type **Table** est réservé à la connexion MsJet. La différence entre le type **Dynaset** et **Snapshot** est que le premier permet des opérations en lecture-écriture alors que le second ne permet que les accès en consultation (lecture seule).

Enfin, entrez l'ordre **SELECT** correspondant à la collection d'enregistrements dans la propriété **RecordSource**, par exemple **SELECT * FROM Authors**. En mode MsJet, vous pouvez sélectionner directement la table.

Structure de la base Biblio

Les contrôles dépendants

Vous pouvez lier des contrôles standards diffusés avec VB comme la zone de texte, la case à cocher, la zone de liste, la zone de liste modifiable et le conteneur OLE pour les images au Data Control précédemment créé.

Denis Szalkowski

Pour le Data Control précédent, vous aurez besoin de trois champs de type **Text**.

Pour chaque zone de texte, après les avoir nommées (**Name**), modifiez tout d'abord la propriété **DataSource** en sélectionnant le Data Control , Data_AUTEURS. Puis, choisissez le champ que vous voulez afficher dans la zone de texte par la propriété **DataField**.

Pour la Data Control :

Connect = "odbc;dsn=biblio;"
DatabaseName = ""
DefaultCursorType = 1 'ODBCCursor
DefaultType = 1 'UseODBC
RecordsetType = 1 'Dynaset
*RecordSource = "SELECT * FROM Authors"*

Pour la zone de texte :

DataField = "Au_ID"
DataSource = "Data_AUTEURS"

Les contrôles de l'édition professionnelle

Il existe trois composants très puissants distribués au niveau de l'édition professionnelle : **Crystal Report** (outil de reporting), **Data Bound List** (listes déroulantes), **Data Bound Grid** (tableaux).

A partir de votre boîte à outils, par un clic droit, ajoutez ces trois composants.

Utilisation du Data Grid

Pour le DataGrid, la seule propriété à renseigner est le DataSource qui doit référence au Control Data. Faites ensuite un clic droit sur le Data Grid et choisissez **Extrait les zones**. Pour donner d'autres titres aux colonnes, par clic droit , accédez à la page de **Propriétés**.

Utilisation du Data List

Pour le DataList, les éléments à renseigner sont les suivants : **RowSource** fait référence au DataControl, **ListField** et **BoundColumn** au champ de la requête ou de la table.

Utilisation du Crystal Report

De tous les contrôles, Crystal Report est sans doute l'un des plus compliqué à utiliser. Il vous permet de faire du reporting : des états sur la base de données.

Pour créer un état, allez dans **Compléments | Créateurs d'états**. En premier lieu, vous devez choisir votre source de données.

Par la suite, vous devez déterminer les tables à partir desquelles vous souhaitez concevoir votre état.

Il peut s'agir d'une requête, comme dans l'exemple ci-dessous.

Vous pouvez modifier la nature de la jointure : équijointure ou jointure externe. Faites un double clic sur le connecteur qui relie les tables pour accéder à l'écran suivant.

La dernière étape génère l'aperçu de l'état. Vous pouvez maintenant l'éditer. Ainsi pour changer les noms des champs, faites un clic droit et choisissez **Editer le champ texte**. Une fois l'état fini, sauvegardez le. L'extension du fichier est de type **RPT**.

Pour utiliser le composant dans votre feuille, dessinez un bouton à l'aide de l'outil Crystal Report. Les propriétés à renseigner sont le **DataSource** correspondant au DataControl et le **ReportFileName** Correspondant au fichier RPT.

Pour lancer l'état, créez un bouton auquel vous associez le code événementiel suivant

:

```
Private Sub Command_ETAT()
        CrystalReport_PUB.SQLQuery = DATA_NEW
        CrystalReport_PUB.Destination = crptToWindow
        CrystalReport_PUB.Action = 1
End Sub
```

Dans l'exemple ci-dessus, DATA_NEW correspond à une requête SELECT.

Quelques instructions sur le RecordSet

Ajout, modification et suppression d'enregistrement

Pour ajouter un enregistrement à partir d'un Data Control, modifiez par programmation la propriété **EOFAction** en lui donnant la valeur **AddNew**, soit la valeur 2.

Vous pouvez aussi directement le faire par programmation. La validation intervient lors d'un nouvel **AddNew** ou d'un ordre **Update** sur le RecordSet. Pensez peut-être à poser une gestion d'erreurs dans de telles procédures, faute de quoi les messages d'erreur risqueraient de pleuvoir dru.

```
Private Sub Command_AJOUT_Click()
        Data1.Recordset.AddNew
End Sub
Private Sub Command_VALIDE_Click()
        Data1.Recordset.UpDate
End Sub
```

Pour la mise à jour, le code est identique à ceci près que vous devez employer la méthode **Edit**. Edit nécessite l'emploi de la commande UpDate.

Enfin pour la suppression, employez la méthode **Delete**. Vous n'avez pas besoin d'utiliser la commande Update?

Les déplacements

Vous pouvez utiliser les méthodes MoveFirst, MoveLast, MovePrevious, MoveNext. Pour cela, vous devez tester l'enregistrement en cours.

```
Private Sub Command_FIN_Click()
        Data1.Recordset.MoveLast
End Sub
Private Sub Command_PREC_Click()
        If Not Data1.Recordset.BOF Then
            Data1.Recordset.MovePrevious
        Else
            Data1.Recordset.MoveLast
        End If
End Sub
Private Sub Command_PREMIER_Click()
        Data1.Recordset.MoveFirst
End Sub
Private Sub Command_SUIV_Click()
        If Not Data1.Recordset.EOF Then
        Data1.Recordset.MoveNext
        Else
        Data1.Recordset.MoveFirst
        End If
End Sub
```

La méthode Find

Pour rechercher des enregistrements, vous pouvez utiliser les méthodes suivantes :

FindFirst, FindLast, FindNext et FindPrevious.

```
Private Sub Command_FIND_Click()
        Data1.Recordset.FindFirst "Revue like '" & Text4.Text & "'"
End Sub
Private Sub Command_CHERCHE_SUIVANT_Click()
        Data1.Recordset.FindNext "Revue like '" & Text4.Text & "'"
End Sub
```

La méthode Seek

Cette méthode s'appuie sur le choix préalable d'un index avec la propriété Index.

```
Dim Table_REVUE As RecordSet
Set Table_REVUE=Data1.RecordSet
Table_REVUE.Index="REVUE"
```

La recherche s'effectue en utilisant un opérateur de comparaison et une valeur numérique ou alphanumérique correspondant au type du champ.

La méthode *NoMatch* permet de rechercher jusqu'à ce qu'aucun enregistrement ne soit trouvé.

```
Table_REVUE.Seek "=","Windows Plus"
Do Until Table_REVUE.NoMatch
```

Les signets

Vous pouvez mémoriser l'enregistrement sur lequel vous vous situez à l'aide de la propriété *BookMark*.

Pour mémoriser la position du pointeur, inspirez-vous du code suivant :

```
Dim MARQUE_PAGE As Variant
MARQUE_PAGE=Data1.RecordSet.BookMark
Data1.Recordset.MoveLast
Data1.Recorset.BookMark=MARQUE_PAGE
```

Les transactions

Les contrôles Data fonctionnement dans un mode de transaction en validation automatique. Vous devez déclarer l'objet correspondant à votre jeu d'enregistrements ainsi que l'espace de travail dans lequel vous évoluez. La méthode **BeginTrans** initie les transactions jusqu'à ce qu'elles soient validées par **CommitTrans** ou annulées par ***RollBack***.

```
Dim TABLE_REVUE As RecordSet, SESSION As WorkSpace
Set SESSION=WorkSpace(0)
Set TABLE_REVUE=Data1.RecordSet
SESSION.BeginTrans
....
SESSION.CommitTrans
....
SESSION.Rollback
```

Les instructions de mise à jour

La méthode **Refresh** permet de mettre à jour les propriétés du contrôle Data. C'est fort utile lorsque vous utilisez des instructions SQL par exemple.

La commande **UpDateRecord** met à jour le jeu d'enregistrements avec les données saisies dans les contrôles dépendants.

La commande **UpdateControls** provoque l'effet inverse à l'instruction précédente.

Utilisation des commandes SQL

Le SQL (le langage de manipulation de données) interne à Visual Basic n'est pas standard. Toutes les instructions qui suivent peuvent être utilisées au niveau de la propriété **RecordSource** de votre contrôle **Data**.

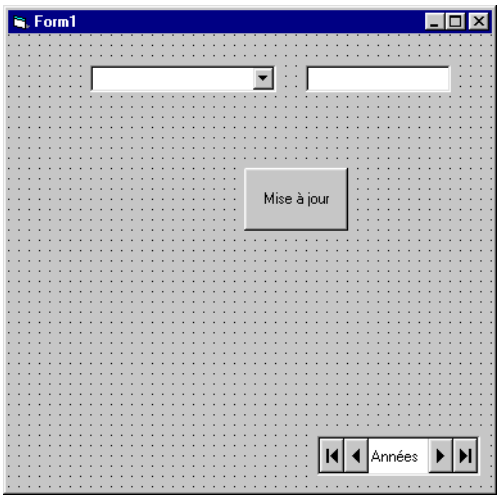
SELECT

L'ordre SELECT permet de retourner un jeu d'enregistrement.

*SELECT [predicate] { * | table.* | [table.]field1 [AS alias1] [, [table.]field2 [AS alias2] [, ...]]}*
FROM tableexpression [, ...]
[IN externaldatabase]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]

predicate	ALL, DISTINCT, DISTINCTROW ou TOP. DISTINCT La clause DISTINCT évite les doublons. Si rien n'est précisé, ALL est choisi par défaut.
*	Indique que tous les champs de la ou des tables spécifiées sont sélectionnés.
table	Nom de la table contenant les champs dans lesquels les enregistrements sont sélectionnés.
field1, field2	Noms des champs contenant les données à extraire. Si vous incluez plusieurs champs, les données seront extraites dans l'ordre indiqué.
alias1, alias2	Noms à utiliser comme en-têtes de colonne à la place des noms de colonnes originaux dans table.
tableexpression	Nom de la ou des tables contenant les données à extraire.
externaldatabase	Nom de la base de données contenant les tables de tableexpression si elles ne se trouvent pas dans la base de données en cours.

*SELECT * FROM Employés;*
SELECT Employés.Département, Superviseurs.NomSupv
FROM Employés INNER JOIN Superviseurs
WHERE Employés.Département = Superviseurs.Département;
SELECT [Date de naissance]
AS Anniversaire FROM Employés;



UPDATE
DELETE

Denis Szalkowski

BeginTrans
CommitTrans
RollBack

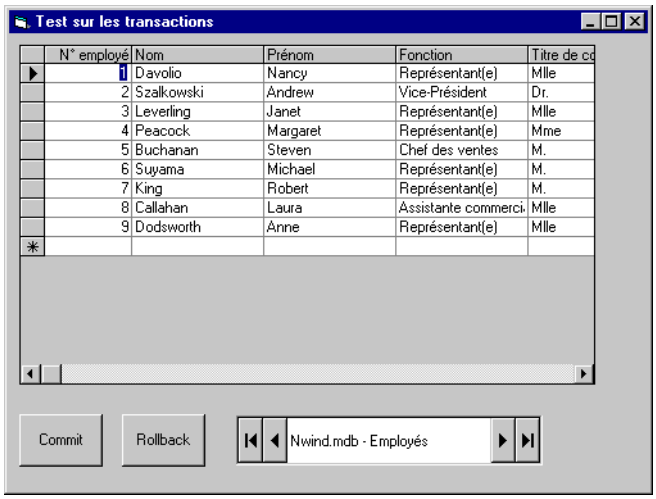
Vous pouvez employer des fonctions de regroupement statistiques :

Avg	moyenne
Count	comptage
Min, Max	Minima et maxima
StDev, StDevP	Ecart-type sur un échantillon , sur une population
Sum	Somme
Var, VarP	Variance sur un échantillon , sur une population

SELECT COUNT([N° employé])
AS Effectif FROM Employés;

L'opérateur INNER JOIN

Cet opérateur permet d'établir entre une à n tables lorsqu'à partir d'un champ commun, ce dernier contient des valeurs identiques. On parle d'équijointure. Cet opérateur s'emploie derrière l'instruction FROM. Si vous employez les opérateurs LEFT



JOIN ou RIGHT JOIN, vous demandez à obtenir tous les enregistrements de la table mentionnée au niveau du membre gauche de la jointure.

FROM table1 INNER JOIN table2 ON table1.field1 compopr table2.field2

Vous pouvez employer comme opérateur de comparaison relationnelle "=", "<", ">", "<=", ">=", ou "<>".

```
SELECT [Nom de catégorie], [Nom du produit]
FROM Catégories INNER JOIN Produits
ON Catégories.[Nom de catégorie] = Produits.[Nom de catégorie];
SELECT fields
FROM table1 INNER JOIN table2
ON table1.field1 compopr table2.field1 AND
ON table1.field2 compopr table2.field2) OR
ON table1.field3 compopr table2.field3)];
SELECT fields
FROM table1 INNER JOIN
(table2 INNER JOIN [( ]table3
[INNER JOIN [( ]tablex [INNER JOIN ...) ]
ON table3.field3 compopr tablex.fieldx)]
ON table2.field2 compopr table3.field3)
ON table1.field1 compopr table2.field2;
```

DELETE

L'odre DELETE permet de supprimer le contenu de champs dans une table à partir de critères logiques. On l'emploie pour détruire un ensemble d'enregistrements.

```
DELETE [table.*]
FROM table
WHERE criteria
```

UPDATE

L'instruction UPDATE vous autorise à mettre à jour une collection d'enregistrement selon une expression soumise à critères.

```
UPDATE table
SET newvalue
WHERE criteria;
```

Dans l'exemple ci-dessous, les valeurs de MontantCommande sont automatiquement multipliées par un coefficient de 1,1, les valeurs du port de 1,03 pour le pays de livraison égal au Royaume Uni.

```
UPDATE Commandes
SET [Montant Commande] = [Montant Commande] * 1,1,
Port = Port * 1,03
WHERE [Pays livraison] = 'RU';
```

INSERT INTO

Cette instruction permet d'ajouter un ou plusieurs enregistrements à une table.

Pour ajouter plusieurs enregistrements :

INSERT INTO target [IN externaldatabase] [(field1[, field2[, ...]])]
SELECT [source.]field1[, field2[, ...]
FROM tableexpression

Pour un un seul enregistrement :

INSERT INTO target [(field1[, field2[, ...]])]
VALUES (value1[, value2[, ...])

Si, lors d'une opération d'ajout, vous ne précisez pas tous les champs, la valeur par défaut ou Null s'insère à la place des colonnes manquantes.

Vous pouvez aussi utiliser INSERT INTO pour ajouter une série d'enregistrements depuis une autre table ou requête, conjointement à la clause SELECT ... FROM comme décrit plus haut dans la syntaxe de requête Ajout avec plusieurs enregistrements. Dans ce cas, la clause SELECT mentionne les champs à ajouter à la table target spécifiée.

SELECT INTO

Cette instruction SQL permet la création d'un table.

SELECT field1[, field2[, ...]] INTO newtable
[IN externaldatabase]
FROM source

Remarques sur les instructions UPDATE et DELETE.

Elle nécessite du code... Rien que du code ! Avec même... quelques surprises ! Les bases de données de type Access exigent un mode d'ouverture de type Jet et non ODBC !

```
Private Sub Command_MAJ_Click()
        Dim dbs As Database
        Set dbs = OpenDatabase("E:\samples\cinq\Biblio97.mdb")
        dbs.Execute "UPDATE Titles " _
            & "SET [Year Published] = " & CInt(Text_NEW.Text) _
            & " WHERE [Year Published] = " & CInt(DBCombo_ANNEES.Text)
        dbs.Close
        Data_ANNEES.Refresh
End Sub
```

Les transactions

Un espace de travail définit un environnement dans lequel vous pouvez valider ou invalider des modifications apportées à vos données : mises à jour, ajout et suppression.

Après avoir déclaré une variable de type **WorkSpace** au niveau d'un module ou d'un module de feuille, initialisez cette variable à l'aide de l'instruction **Set**.

L'espace transactionnel est défini par la méthode **BeginTrans**. Pour valider les modifications apportées, utilisez la méthode **CommitTrans**. Pour invalider, employez **RollBack**. Toute sortie intempestive du programme équivaut à un RollBack.

Une exemple : utilisation de la table Employés de l'application NWind.mdb

Après avoir ajouté le composant **Microsoft Data Bound Grid**, dessinez un Data Control et DBGrid Control. Après avoir défini les propriétés **Connect**, **DataBaseName** et **RecordSource** du Data Control, définissez les propriétés **DataSource** du DBGrid. Par un clic droit, extrayez les champs de la table par **Extraire les zones**.

Le code associé

```
Option Explicit
'
'Définit une variable de type environnement
'
Dim wrkCourant As Workspace

'
'Création de l'espace de travail
'
Private Sub Form_Initialize()
        Set wrkCourant = DBEngine.Workspaces(0)
End Sub
'
'Démarrage de l'espace transactionnel
'
Private Sub Form_Activate()
        wrkCourant.BeginTrans
End Sub
'
'Validation effective des modifications
'
Private Sub Command_COMMIT_Click()
        wrkCourant.CommitTrans
        Data_EMP.Refresh
        wrkCourant.BeginTrans
End Sub
'
'Invalidaton des modifications
'
Private Sub Command_ROLLBACK_Click()
        wrkCourant.Rollback
        Data_EMP.Refresh
        wrkCourant.BeginTrans
End Sub
'
'Fermeture de l'environnement
'
Private Sub Form_Unload(Cancel As Integer)
        wrkCourant.Close
        Set wrkCourant = Nothing
End Sub
```

Annexe : le code associé aux feuilles

Le code associé au formulaire de saisie

```
Private Sub Check_NAISSANCE_Click()
        If Check_NAISSANCE.Value Then
            Data_AUTEURS.RecordSource = «SELECT * FROM Authors WHERE [Year
Born] Is Not Null «
                Else
            Data_AUTEURS.RecordSource = «SELECT * FROM Authors»
        End If
        Data_AUTEURS.Refresh
End Sub

Private Sub Command_COMPTE_Click()
        Data_COMPTE.RecordSource = «SELECT COUNT(Au_ID) As Total FROM Authors»
        Data_COMPTE.Refresh
        MsgBox Data_COMPTE.Recordset.Fields(«Total»)
End Sub
```

La requête sur le formulaire "DataGrid"

```
SELECT Titles.Title,Authors.Author,Publishers.Name
FROM
(Publishers INNER JOIN Titles ON Publishers.PubID=Titles.PubID)
INNER JOIN
([Title Author] INNER JOIN Authors ON [Title Author].Au_ID=Authors.Au_ID)
ON Titles.ISBN=[Title Author].ISBN
```

Le code associé au formulaire "DataGrid"

```
Option Explicit
Dim DATA_ORI As String
Dim DATA_NEW As String
Dim CRITERIA As String

Private Sub Form_Load()
        DATA_ORI = Data_PUBLICATIONS.RecordSource
End Sub

Private Sub DBCombo_AUTEURS_Click(Area As Integer)
        CRITERIA = « WHERE Authors.Author = '« & DBCombo_AUTEURS.SelText & «'«
        DATA_NEW = DATA_ORI & CRITERIA
        Data_PUBLICATIONS.RecordSource = DATA_NEW
        Data_PUBLICATIONS.Refresh
End Sub

Private Sub Command_ETAT()
        CrystalReport_PUB.SQLQuery = DATA_NEW
        CrystalReport_PUB.Destination = crptToWindow
        CrystalReport_PUB.Action = 1
End Sub
```

Le code associé au formulaire "mise à jour"

```
Option Explicit

Private Sub Command_MAJ_Click()
        Dim dbs As Database
        Set dbs = OpenDatabase(«E:\samples\cinq\Biblio97.mdb»)
        dbs.Execute «UPDATE Titles « _
                & «SET [Year Published] = « & CInt(Text_NEW.Text) _
                & « WHERE [Year Published] = « & CInt(DBCombo_ANNEES.Text)
        dbs.Close
        Data_ANNEES.Refresh
End Sub
```

Technologie ActiveX

Property
Get
Let

Présentation de la technologie ActiveX

Active X est une technologie d'interopérabilité applicative. C'est le fer de lance de Microsoft pour contrer Java. Elle est propriétaire et n'est utilisable que sur les plate-formes Win32. Rappelons que Java, s'appuyant sur une machine virtuelle, interprète ses composants ou Applets sur toutes les plates-formes. Ses composants peuvent être appelés à partir d'une page HTML. Seul Internet Explorer permet d'interpréter des ActiveX... en attendant Netscape Communicator 5.

La programmation orienté objet sous VB

En programmation "classique", les traitements servent à modifier les données déclarées préalablement. En programmation objet, données et traitements sont rassemblés en un élément et un seul : l'objet. Les méthodes sont des procédures qui vont modifier le comportement de l'objet. Les propriétés sont des attributs de ces objets modifiables en lecture, en écriture ou en lecture-écriture.

Ajouter un module de classe

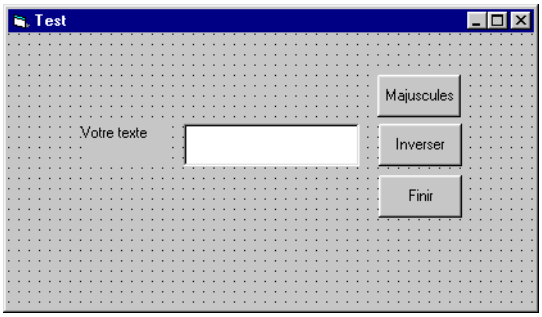
A partir de l'explorateur de projets, par un clic droit, ajoutez un module de classe. Les fichiers de ce type ont pour suffixe le type CLS.

```
Option Explicit
Private Var_TEXTE As String
'
'La procédure Initialize se produit à la création de l'objet.
'
Private Sub Class_Initialize()
        Var_TEXTE = «»
End Sub
'
'La fonction Property Get permet de lire le contenu de la propriété.
'
Public Property Get TEXTE() As String
        '
        'Il faut reprendre le nom de la propriété !
        '
        TEXTE = Var_TEXTE
End Property

'
'La procédure Property Let permet d'écrire le contenu de la propriété.
'
Public Property Let TEXTE(Arg_TEXTE As String)
        Var_TEXTE = Arg_TEXTE
End Property
'
'INVERSE_TEXTE est une méthode attachée à l'objet CLASS_TEXTE.
'
Public Sub INVERSE_TEXTE()
        Dim I As Integer, L As Integer
        Dim CHAINE As String
        L = Len(Var_TEXTE)
        For I = 1 To L
            CHAINE = Mid(Var_TEXTE, I, 1) & CHAINE
        Next I
```

Module de classe

```
        Var_TEXTE = CHAINE
End Sub
'
'MAJ_TEXTE est une méthode attachée à l'objet CLASS_TEXTE.
'
Public Sub MAJ_TEXTE()
        Var_TEXTE = UCase(Var_TEXTE)
End Sub
'
'Instructions qui s'exécutent à la destruction de l'objet.
'
Private Sub Class_Terminate()
        MsgBox «Fini»
End Sub
```

La forme utilisant le module de classe



Le projet consiste à partir de l'événement click des boutons de commande d'exécuter les méthodes (procédures publiques) définies dans le module de classe.

Le module de feuille

```
Option Explicit
Dim Obj_TEXTE As Class_TEXTE
Private Sub Form_Load()
        '
        'Initialise un nouvel objet
        '
        Set Obj_TEXTE = New Class_TEXTE
End Sub
Private Sub Command_MAJ_Click()
        '
        'Mise en majuscules du texte
        '
        Obj_TEXTE.TEXTE = Text_NOM.Text
        Obj_TEXTE.MAJ_TEXTE
        Text_NOM.Text = Obj_TEXTE.TEXTE
End Sub
Private Sub Command_INVERSER_Click()
        '
        'Inversion du texte saisi dans la zone
        '
        Obj_TEXTE.TEXTE = Text_NOM.Text
        Obj_TEXTE.INVERSE_TEXTE
        Text_NOM.Text = Obj_TEXTE.TEXTE
End Sub
```

```
Private Sub Command_FINIR_Click()
        '
        'Remise à zéro de l'objet
        '
        Set Obj_TEXTE = Nothing
        Unload Me
End Sub
```

Créer un composant ActiveX

L'objet de cette partie est double. Elle est de voir la construction de composants employables à l'intérieur de projets de type EXE Standard. Elle est aussi d'intégrer un composant ActiveX au sein d'une page HTML et de lui transmettre un paramètre. Nous verrons que les deux approches comportent de grandes différences.

Nous chercherons à faire un composant qui permet de voir l'heure ou la date en temps réel.

Le code du composant ActiveX

L'exemple

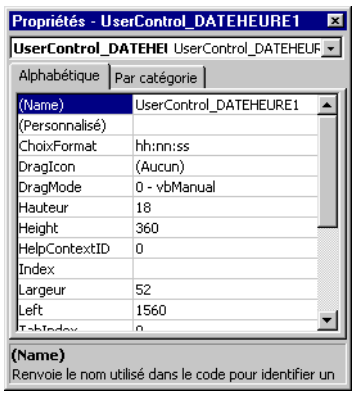
Ajoutez à partir de l'explorateur de projet un contrôle utilisateur. Dessinez alors une étiquette à l'aide du composant Label, ainsi qu'un Timer dont vous réglez la propriété Interval à **1** seconde. Pour utiliser le contrôle au niveau d'un EXE standard, pensez à mettre la propriété Public à False. Pour Internet, vous recompilerez après avoir mis la propriété à True.

```
Option Explicit
'
'La constante COEF permet de repasser en pixels
'
Private Const COEF As Single = 1440 / 72
'
'Les variables TEMP stockent les valeurs des différentes propriétés.
'
Private TEMP_HAUTEUR As Integer, TEMP_LARGEUR As Integer
Private TEMP_FORMAT As String
'
'L'horloge est rafraîchie à partir de la propriété ChoixFormat
'
Private Sub Timer_HORLOGE_Timer()
        Label_DATEHEURE.Caption = Format(Now(), ChoixFormat)
End Sub
'
'Mise en place du composant
'
Private Sub UserControl_Initialize()
        TEMP_FORMAT = «hh:nn:ss»
        Label_DATEHEURE.Caption = Format(Now(), ChoixFormat)
End Sub
'
'Initialisation des propriétés : uniquement pour l'intégration en HTML
'
Private Sub UserControl_InitProperties()
        ChoixFormat = «hh:nn:ss»
End Sub
'
'Lecture des propriétés : uniquement pour l'intégration en HTML
'
Private Sub UserControl_ReadProperties(PropBag As PropertyBag)
```

```
                ChoixFormat = PropBag.ReadProperty(«ChoixFormat», «hh:nn:ss»)
End Sub
'
'Modification des propriétés : uniquement pour l'intégration en HTML
'
Private Sub UserControl_WriteProperties(PropBag As PropertyBag)
                PropBag.WriteProperty «ChoixFormat», ChoixFormat, «hh:nn:ss»
End Sub
'
'Lecture de la propriété ChoixFormat
'
Public Property Get ChoixFormat() As String
                ChoixFormat = TEMP_FORMAT
End Property
'
'Modification de la propriété ChoixFormat
'
Public Property Let ChoixFormat(Nouveau_Format As String)
                TEMP_FORMAT = Nouveau_Format
                '
                'Equivaut à un «refresh»
                '
                PropertyChanged «ChoixFormat»
End Property
Public Property Get Hauteur() As Integer
                UserControl.Height = Label_DATEHEURE.Height
                Hauteur = CInt(UserControl.Height / COEF)
End Property
Public Property Get Largeur() As Integer
                UserControl.Width = Label_DATEHEURE.Width
                Largeur = CInt(UserControl.Width / COEF)
End Property
Public Property Let Hauteur(NouvelleHauteur As Integer)
                TEMP_HAUTEUR = CInt(NouvelleHauteur * COEF)
                Label_DATEHEURE.Height = TEMP_HAUTEUR
                UserControl.Height = TEMP_HAUTEUR
                PropertyChanged «Hauteur»
End Property
Public Property Let Largeur(NouvelleLargeur As Integer)
                TEMP_LARGEUR = CInt(NouvelleLargeur * COEF)
                Label_DATEHEURE.Width = TEMP_LARGEUR
                UserControl.Width = TEMP_LARGEUR
                PropertyChanged «Largeur»
End Property
```

Insertion du composant dans une forme

Fermez préalablement toutes les fenêtres relatives au contrôle utilisateur. Le contrôle apparaît dans la palette des composants.

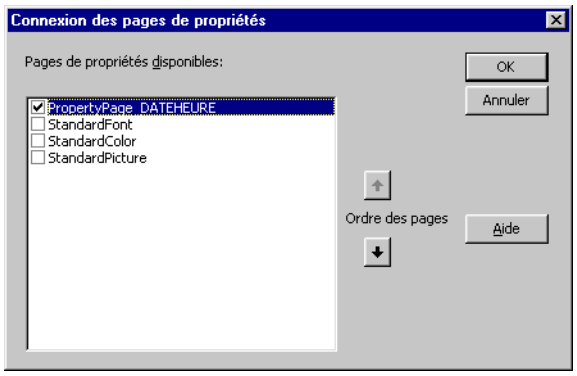


Page de propriétés

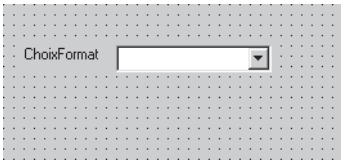
Dessinez alors un rectangle correspondant au composant. Remarquez dans les propriétés l'apparition des propriétés ChoixFormat, Hauteur et Largeur.

Associer une page de propriétés

Vous pouvez accéder à des propriétés supplémentaires à partir des contrôles grâce à un simple clic droit. Lorsque vous définissez vos composants, Visual Basic vous



permet de créer votre propre page de composants. A partir de l'explorateur de projet,



ajoutez une page de propriétés. Au niveau du composant, cliquez dans la fenêtre des propriétés sur PropertyPage. Connectez la page de propriétés à votre composant.

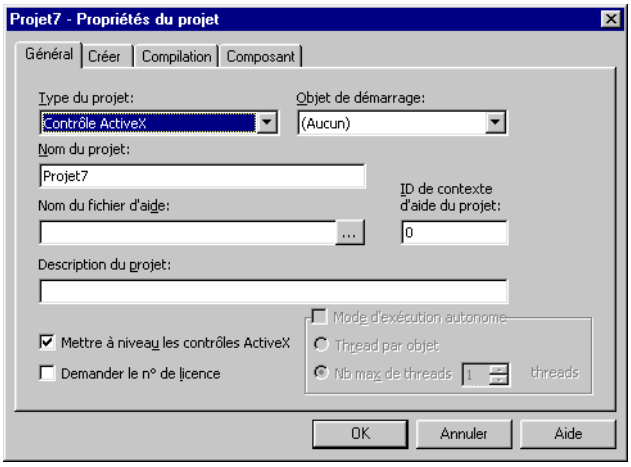
Dans notre exemple, dessinez une combo box.

Voici le code qui lui est associé :

```
Option Explicit

Private Sub PropertyPage_ApplyChanges()
        SelectedControls(0).ChoixFormat = Combo_CHOIXFORMAT.Text
End Sub

Private Sub PropertyPage_Initialize()
        Combo_CHOIXFORMAT.AddItem «hh:nn:ss»
        Combo_CHOIXFORMAT.AddItem «dd/mm/yyyy»
        Combo_CHOIXFORMAT.AddItem «dd/mm/yyyy hh:nn:ss»
End Sub
```

Compiler un composant OCX

Détruisez alors la feuille qui nous a



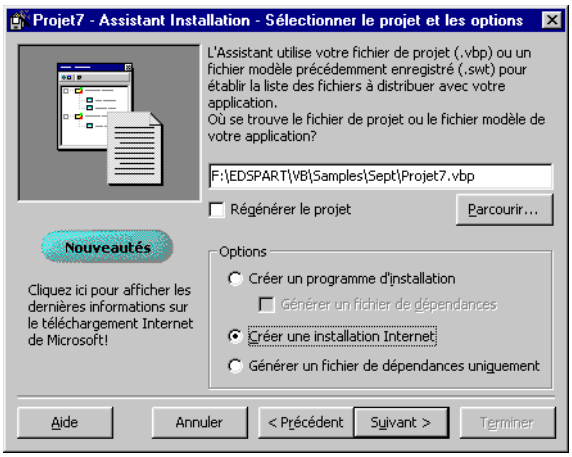
Denis Szalkowski

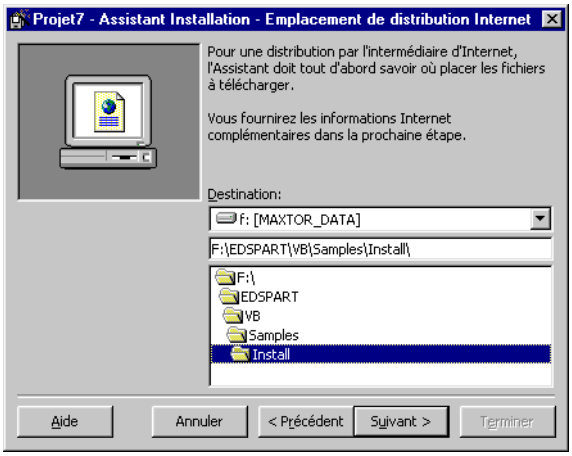
servis à tester notre composants. Repassez la valeur Public du composant à True. Allez dans Projet | Propriétés du projet et choisissez Contrôle ActiveX à Type de projet et (Aucun) à Objet de démarrage.

Enfin, choisissez Fichier | Créer... OCX.

Utiliser un ActiveX à partir d'une page Internet

En fait , l'assistant Installation génère

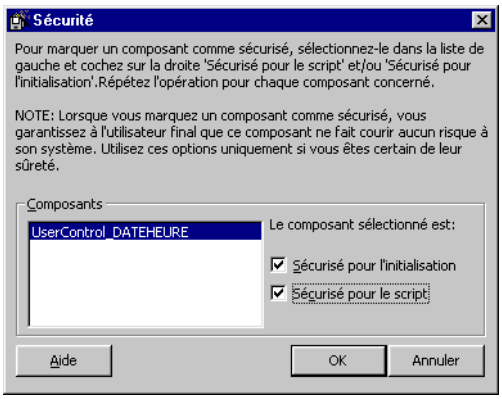


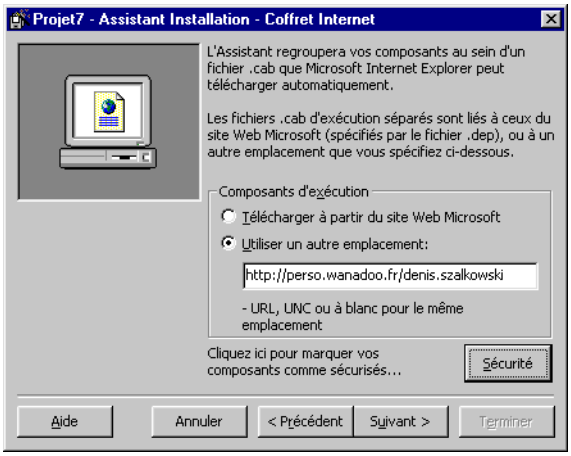


tout le code et tous les fichiers nécessaires au fonctionnement de votre composant au sein d'une page HTML.

Les étapes de l'installation Internet

Au niveau de la première étape de l'assistant, choisissez l'option Créer une installation



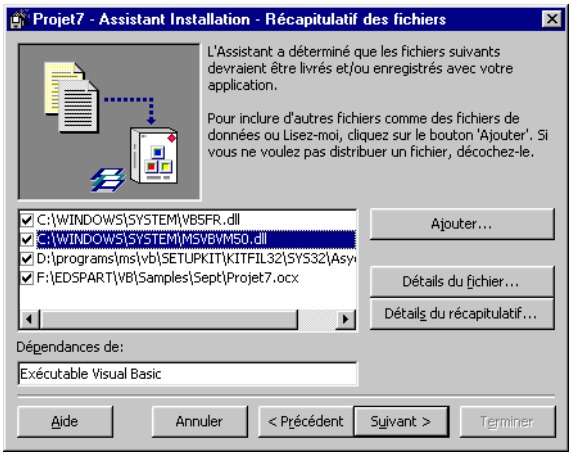


Internet. Sélectionnez votre projet (fichier VBP). Passez à l'étape suivante.

Lors de la seconde étape, vous aurez à

Denis Szalkowski

déterminer le dossier de stockage des fichiers générés par l'assistant installation.

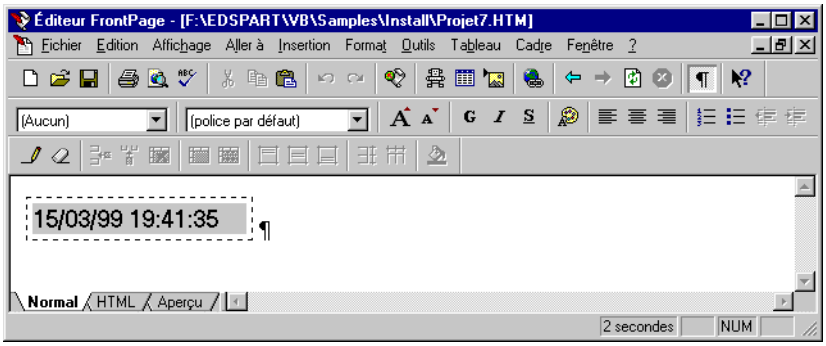


<PARAM ... NAME... VALUE...>

La troisième étape garantit l'imperméabilité de votre contrôle à des paramètres passés par script ou par paramètre.

La quatrième étape permet de spécifier un emplacement alternatif à partir duquel vous pouvez télécharger votre composant.

La cinquième étape Vous donne la liste des dépendances (fichiers nécessaires) pour



l'exécution en mode standard. Après être passé à l'étape suivante, vous en avez terminé.

Modifier le code HTML à partir de FrontPage

Dans le répertoire De l'installation, vous trouverez une page HTML que vous pouvez éditer Avec FrontPage Express ou FrontPage 98... si vous en disposez. Pour visualiser le code sous FrontPage Express, allez dans Affichage | HTML. Remettez par un copier-coller le code de l'objet dans la partie <BODY> </BODY>.

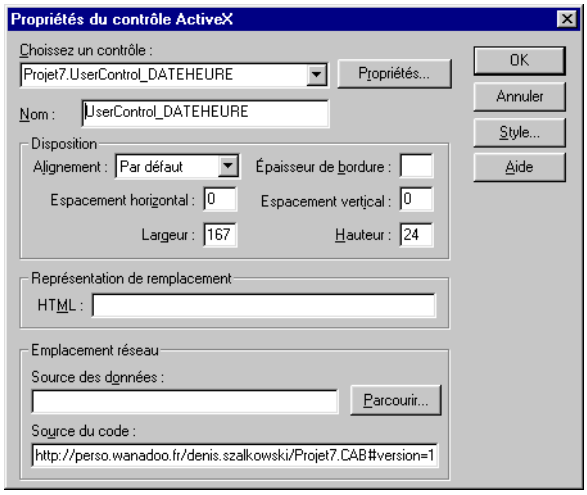
Si vous utilisez FrontPage 98, cliquez sur l'onglet HTML.

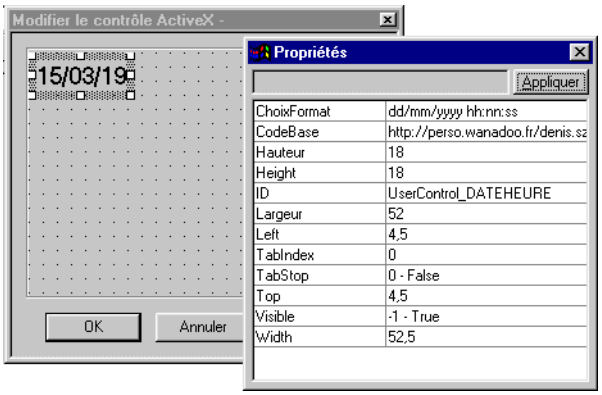
L'utilisation du Control comprend plusieurs paramètres :

CLASSID	Numéro identifiant la classe du contrôle
CODEBASE	Url où se procurer le composant par téléchargement
PARAM NAME... VALUE	Paramètre transmis de la page vers le contrôle

```
<html>
<head>
<title></title>
</head>
<body>
<p>
```

<object
ID=»UserControl_DATEHEURE»
WIDTH=»167" HEIGHT=»24"





CLASSID=»clsid:CBFB9FA8-DAEB-11D2-AFBE-0020AF27536E»
CODEBASE=»http://perso.wanadoo.fr/denis.szalkowski/Projet7.CAB#version=1,0,0,0">
<param name=»_ExtentX» value=»4419">
<param name=»_ExtentY» value=»635">
<param name=»ChoixFormat» value=»dd/mm/yyyy hh:nn:ss»>
</object>

</p>
</body>
</html>

En faisant un double clic sur le composant, vous accédez à la fenêtre suivante. Cliquez alors sur Propriétés... .

Vous accédez à la fenêtre des propriétés.

5. Le client serveur

Client-serveur

Présentation du client-serveur

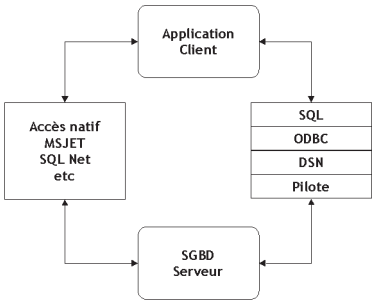
L'architecture client-serveur s'est imposé au monde informatique avec les environnements graphiques. Il ne pouvait plus être question de continuer à faire exécuter les applications à partir d'une seule machine. Cela était trop gourmand en ressources (mémoire, processeurs) et en infrastructure réseau (nécessité d'acheminer à la station cliente les applications) d'autant que les "mainframes" avaient en plus à charge la gestion des moteurs de base de données. A chacun son métier !

Dans l'architecture client-serveur, chaque client dispose d'une station intelligente (et non d'un terminal) dotée de mémoire, d'un processeur, d'un système d'exploitation ou os, d'un disque dur permettant de stocker les applications. Les données, quant à elles, sont stockées sur le serveur souvent sous Unix ou NT Server. Le moteur de base de données, par l'intermédiaire d'un langage universel, le SQL (Structured Query Language), traite les requêtes du client. Les moteurs les plus utilisés sont Oracle, Informix, dB2, Ingres et SQL Server.

Visual Basic permet de gérer des développements de cette nature, tout comme Delphi.

Accès aux bases de données sur les plates-formes Win32

Il existe deux manières d'accéder aux données sur les machines à base de processeurs Intel compatibles et de systèmes d'exploitation Windows 98-98-Nt Workstation 4. Vous pouvez disposer d'un accès natif, propriétaire et souvent plus rapide. Le système propriétaire de Microsoft repose sur Jet. Il vous permet de vous



connecter notamment à une base Access. Ou bien, vous pouvez utiliser ODBC, Open Data Base Connectivity qui vise à une plus grande ouverture et interopérabilité en vous éloignant des couches matérielles. Vous accédez aux ressources par le biais d'un identifiant logique : le DSN (Data Source Name).

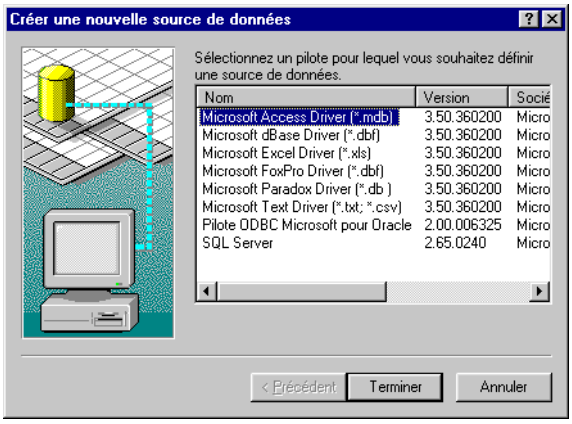
ODBC

Paramétrer ODBC

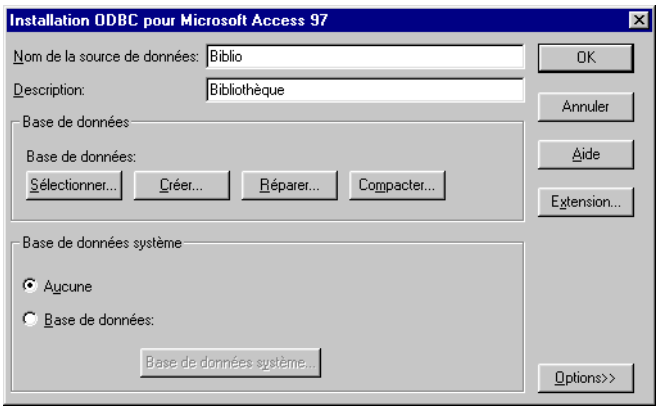
A partir du menu Démarrer, choisissez Paramètres | Panneau de configuration.

DSN

Double-cliquez sur l'icône 
.



Pour créer votre DSN, vous disposez de deux possibilités : un DSN attaché à l'utilisateur (non accessibles aux autres), un DSN attaché à la machine (accessibles à tous les utilisateurs).



Cliquez sur le bouton 
.

La première consiste à déterminer le pilote représentant la nature de la base de

données. Dans l'exemple ci-dessous, il s'agit de Microsoft Access.

Entrez un alias qui représente la connexion dans le nom de la source de données. C'est le nom que vous emploierez sous VB pour vous connecter. La description explicite sous forme d'un commentaire la source de données.

Dans une seconde phase, cliquez sur le bouton 
 pour choisir la base de données. Sélectionnez alors la base de données. Dans l'exemple, il s'agit de la base de données Biblio fournie avec la distribution de Visual Basic.

Utilisation du Data Control

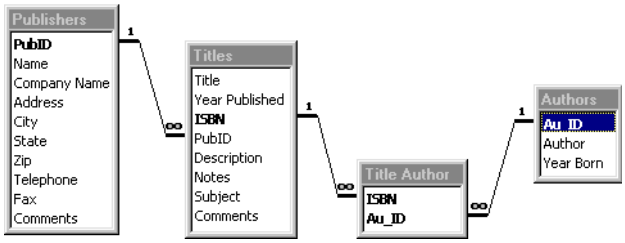
Le Data Control, contrôle accessible à partir de la boîte à outils, représente le jeu d'enregistrements auquel votre application peut accéder. Il peut s'agir d'une table, d'un requête sous la forme d'un ordre SQL SELECT en lecture-écriture ou lecture seule. A partir de la boîte à outils, sélectionnez l'outil Data 

et dessinez une surface dans la feuille correspondant à l'emplacement du Data Control.

Au préalable, nommez votre DataControl à l'aide de la propriété Name Data_AUTEURS par exemple.

Dans un deuxième temps, vous devez déterminer la nature de la connexion à l'aide de la propriété Connect. Pour se connecter à Access via MS Jet, choisissez dans la liste Access. Pour se accéder via ODBC, veuillez taper dans la zone ODBC;DSN=Biblio;[UID=UserName;PWD=Password;]. Le fait de rentrer en "dur" le mot de passe ne répond à une bonne logique de sécurité. Pour les connexions de type MsJet, sélectionnez directement la base Access par DataBaseName.

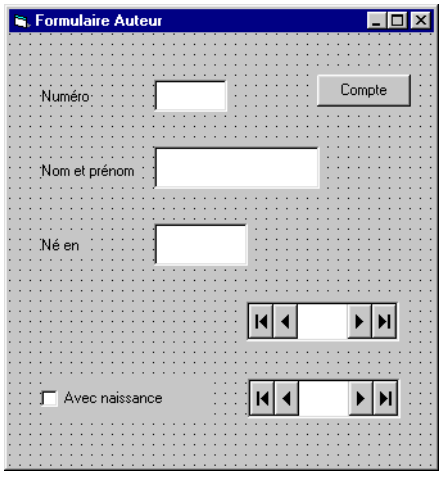
Si vous choisissez l'un ou l'autre mode, allez préciser la nature de la connexion au niveau de la propriété DefaultType. Pour ODBC, prenez UseODBC. UseJet est



réservé à l'emploi de connexions de type MsJet. Pensez aussi au niveau dur DefaultCursorType à choisir en mode ODBC à prendre ODBCCursor.

Il vous reste encore deux étapes. Précisez le RecordSetType. Le type Table est réservé à la connexion MsJet. La différence entre le type Dynaset et Snapshot est que le premier permet des opérations en lecture-écriture alors que le second ne permet que les accès en consultation (lecture seule).

Enfin, entrez l'ordre SELECT correspondant à la collection d'enregistrements dans la propriété RecordSource, par exemple SELECT * FROM Authors. En mode MsJet, vous pouvez sélectionner directement la table.



L'exemple

Structure de la base Biblio

Les contrôles dépendants

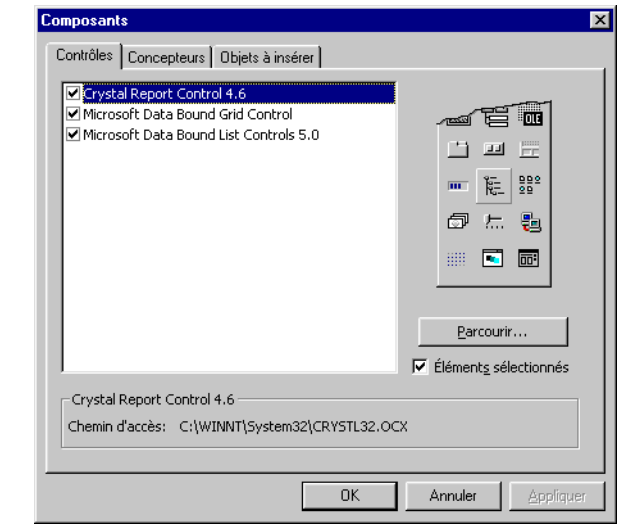
Vous pouvez lier des contrôles standards diffusés avec VB comme la zone de texte, la case à cocher, la zone de liste, la zone de liste modifiable et le conteneur OLE pour les images au Data Control précédemment créé.

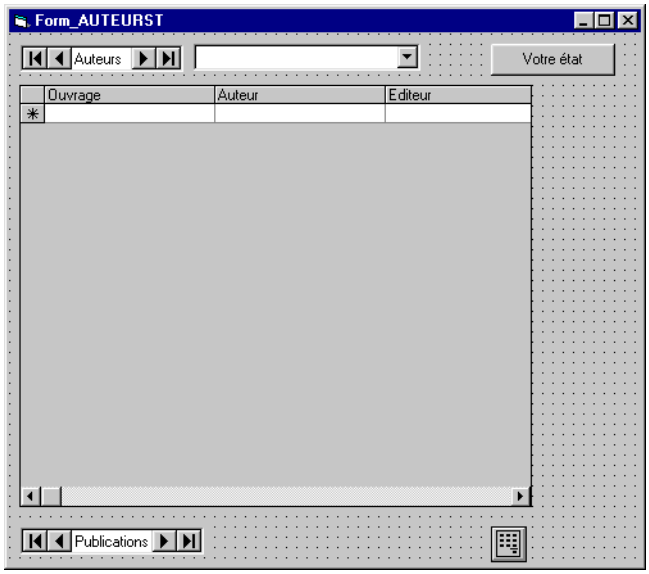
Pour le Data Control précédent, vous aurez besoin de trois champs de type Text.

Pour chaque zone de texte, après les avoir nommées (Name), modifiez tout d'abord la propriété DataSource en sélectionnant le Data Control , Data_AUTEURS. Puis, choisissez le champ que vous voulez afficher dans la zone de texte par la propriété DataField.

Pour la Data Control :

Connect = "odbc;dsn=biblio;"
DatabaseName = ""





DefaultCursorType = 1 'ODBCCursor
DefaultType = 1 'UseODBC
RecordsetType = 1 'Dynaset
*RecordSource = "SELECT * FROM Authors"*

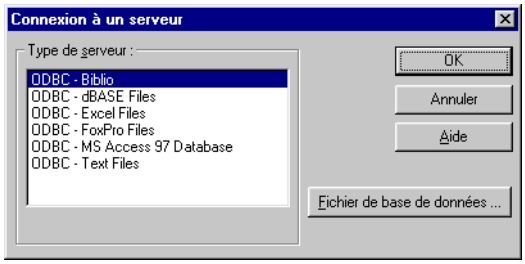
Pour la zone de texte :

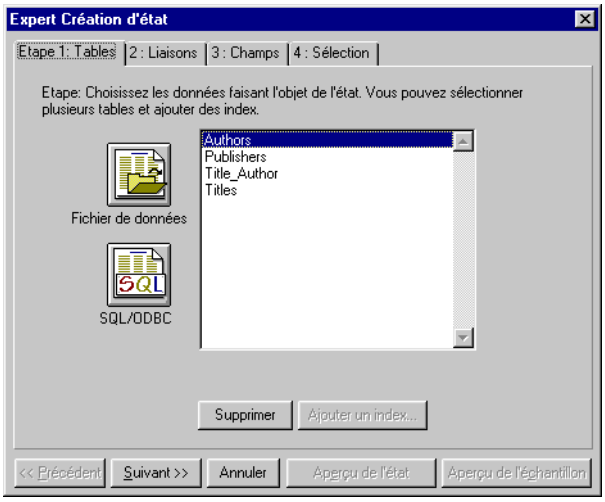
DataField = "Au_ID"

DataSource = "Data_AUTEURS"

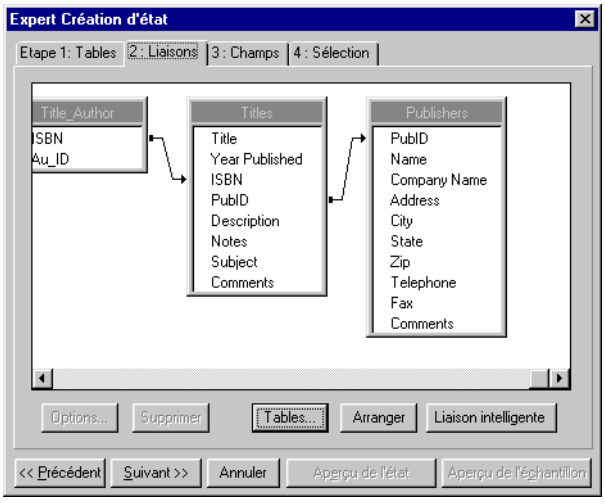
Les contrôles de l'édition professionnelle

Il existe trois composants très puissants distribués au niveau de l'édition professionnelle : Crystal Report (outil de reporting), Data Bound List (listes déroulantes), Data Bound Grid (tableaux).

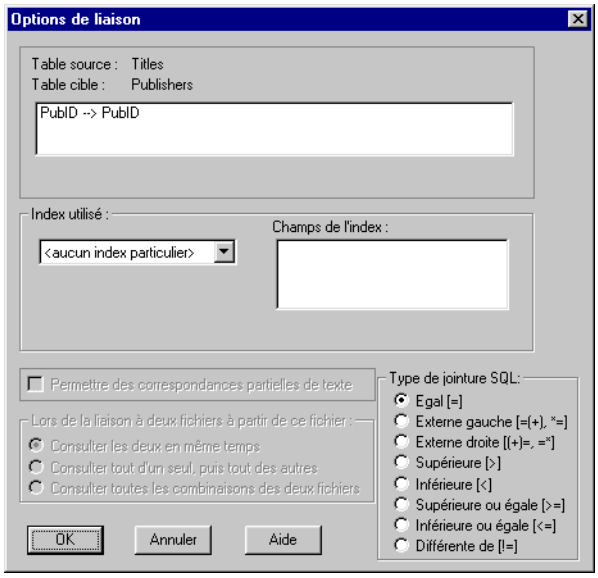




A partir de votre boîte à outils, par un clic droit, ajoutez ces trois composants.



Utilisation du Data Grid



Pour le DataGrid, la seule propriété à renseigner est le DataSource qui doit référence au Control Data. Faites ensuite un clic droit sur le Data Grid et choisissez Extrait les zones. Pour donner d'autres titres aux colonnes, par clic droit , accédez à la page de Propriétés.

Utilisation du Data List

Pour le DataList, les éléments à renseigner sont les suivants : RowSource fait référence au DataControl, ListField et BoundColumn au champ de la requête ou de la table.

Utilisation du Crystal Report

De tous les contrôles, Crystal Report est sans doute l'un des plus compliqué à utiliser. Il vous permet de faire du reporting : des états sur la base de données.

Pour créer un état, allez dans Compléments | Créateurs d'états. En premier lieu, vous devez choisir votre source de données.

Par la suite, vous devez déterminer les tables à partir desquelles vous souhaitez concevoir votre état.

Il peut s'agir d'une requête, comme dans l'exemple ci-dessous.

Vous pouvez modifier la nature de la jointure : équijointure ou jointure externe. Faites un double clic sur le connecteur qui relie les tables pour accéder à l'écran suivant.

La dernière étape génère l'aperçu de l'état. Vous pouvez maintenant l'éditer. Ainsi pour changer les noms des champs, faites un clic droit et choisissez Editer le champ texte. Une fois l'état fini, sauvegardez le. L'extension du fichier est de type RPT.

Pour utiliser le composant dans votre feuille, dessinez un bouton à l'aide de l'outil Crystal Report. Les propriétés à renseigner sont le DataSource correspondant au DataControl et le ReportFileName Correspondant au fichier RPT.

Pour lancer l'état, créez un bouton auquel vous associez le code événementiel suivant :

```
Private Sub Command_ETAT()
        CrystalReport_PUB.SQLQuery = DATA_NEW
        CrystalReport_PUB.Destination = crptToWindow
        CrystalReport_PUB.Action = 1
```

Denis Szalkowski

```
End Sub
```

Dans l'exemple ci-dessus, DATA_NEW correspond à une requête SELECT.

Quelques instructions sur le RecordSet

Ajout, modification et suppression d'enregistrement

Pour ajouter un enregistrement à partir d'un Data Control, modifiez par programmation la propriété EOFAction en lui donnant la valeur AddNew, soit la valeur 2.

Vous pouvez aussi directement le faire par programmation. La validation intervient lors d'un nouvel AddNew ou d'un ordre Update sur le RecordSet. Pensez peut-être à poser une gestion d'erreurs dans de telles procédures, faute de quoi les messages d'erreur risqueraient de pleuvoir dru.

```
Private Sub Command_AJOUT_Click()
        Data1.Recordset.AddNew
End Sub
Private Sub Command_VALIDE_Click()
        Data1.Recordset.UpDate
End Sub
```

Pour la mise à jour, le code est identique à ceci près que vous devez employer la méthode Edit. Edit nécessite l'emploi de la commande UpDate.

Enfin pour la suppression, employez la méthode Delete. Vous n'avez pas besoin d'utiliser la commande Update?

Les déplacements

Vous pouvez utiliser les méthodes MoveFirst, MoveLast, MovePrevious, MoveNext. Pour cela, vous devez tester l'enregistrement en cours.

```
Private Sub Command_FIN_Click()
        Data1.Recordset.MoveLast
End Sub
Private Sub Command_PREC_Click()
        If Not Data1.Recordset.BOF Then
            Data1.Recordset.MovePrevious
        Else
            Data1.Recordset.MoveLast
        End If
End Sub
Private Sub Command_PREMIER_Click()
        Data1.Recordset.MoveFirst
End Sub
Private Sub Command_SUIV_Click()
        If Not Data1.Recordset.EOF Then
        Data1.Recordset.MoveNext
        Else
        Data1.Recordset.MoveFirst
        End If
End Sub
```

La méthode Find

Pour rechercher des enregistrements, vous pouvez utiliser les méthodes suivantes : FindFirst, FindLast, FindNext et FindPrevious.

```
Private Sub Command_FIND_Click()
        Data1.Recordset.FindFirst "Revue like '" & Text4.Text & "'"
End Sub
Private Sub Command_CHERCHE_SUIVANT_Click()
```

```
        Data1.Recordset.FindNext "Revue like '" & Text4.Text & "'"
    End Sub
```

La méthode Seek

Cette méthode s'appuie sur le choix préalable d'un index avec la propriété Index.

```
Dim Table_REVUE As RecordSet
Set Table_REVUE=Data1.RecordSet
Table_REVUE.Index="REVUE"
```

La recherche s'effectue en utilisant un opérateur de comparaison et une valeur numérique ou alphanumérique correspondant au type du champ.

La méthode *NoMatch* permet de rechercher jusqu'à ce qu'aucun enregistrement ne soit trouvé.

```
Table_REVUE.Seek "=","Windows Plus"
Do Until Table_REVUE.NoMatch
```

Les signets

Vous pouvez mémoriser l'enregistrement sur lequel vous vous situez à l'aide de la propriété *BookMark*.

Pour mémoriser la position du pointeur, inspirez-vous du code suivant :

```
Dim MARQUE_PAGE As Variant
MARQUE_PAGE=Data1.RecordSet.BookMark
Data1.Recordset.MoveLast
Data1.Recorset.BookMark=MARQUE_PAGE
```

Les transactions

Les contrôles Data fonctionnement dans un mode de transaction en validation automatique. Vous devez déclarer l'objet correspondant à votre jeu d'enregistrements ainsi que l'espace de travail dans lequel vous évoluez. La méthode BeginTrans initie les transactions jusqu'à ce qu'elles soient validées par CommitTrans ou annulées par *RollBack*.

```
Dim TABLE_REVUE As RecordSet, SESSION As WorkSpace
Set SESSION=WorkSpace(0)
Set TABLE_REVUE=Data1.RecordSet
SESSION.BeginTrans
....
SESSION.CommitTrans
....
SESSION.Rollback
```

Les instructions de mise à jour

La méthode Refresh permet de mettre à jour les propriétés du contrôle Data. C'est fort utile lorsque vous utilisez des instructions SQL par exemple.

La commande UpDateRecord met à jour le jeu d'enregistrements avec les données saisies dans les contrôles dépendants.

La commande UpdateControls provoque l'effet inverse à l'instruction précédente.

Utilisation des commandes SQL

Le SQL (le langage de manipulation de données) interne à Visual Basic n'est pas standard. Toutes les instructions qui suivent peuvent être utilisées au niveau de la propriété RecordSource de votre contrôle Data.

SELECT

L'ordre SELECT permet de retourner un jeu d'enregistrement.

Utilisation de INNER JOIN

*SELECT [predicate] { * | table.* | [table.]field1 [AS alias1] [, [table.]field2 [AS alias2] [, ...]]}*
FROM tableexpression [, ...]
[IN externaldatabase]
[WHERE...]
[GROUP BY...]
[HAVING...]
[ORDER BY...]
[WITH OWNERACCESS OPTION]

predicate	ALL, DISTINCT, DISTINCTROW ou TOP. DISTINCT La clause DISTINCT évite les doublons. Si rien n'est précisé, ALL est choisi par défaut.
*	Indique que tous les champs de la ou des tables spécifiées sont sélectionnés.
table	Nom de la table contenant les champs dans lesquels les enregistrements sont sélectionnés.
field1, field2	Noms des champs contenant les données à extraire. Si vous incluez plusieurs champs, les données seront extraites dans l'ordre indiqué.
alias1, alias2	Noms à utiliser comme en-têtes de colonne à la place des noms de colonnes originaux dans table.
tableexpression	Nom de la ou des tables contenant les données à extraire.
externaldatabase	Nom de la base de données contenant les tables de tableexpression si elles ne se trouvent pas dans la base de données en cours.

*SELECT * FROM Employés;*
SELECT Employés.Département, Superviseurs.NomSupv
FROM Employés INNER JOIN Superviseurs
WHERE Employés.Département = Superviseurs.Département;
SELECT [Date de naissance]
AS Anniversaire FROM Employés;

Vous pouvez employer des fonctions de regroupement statistiques :

Avg	moyenne
Count	comptage
Min, Max	Minima et maxima
StDev, StDevP	Ecart-type sur un échantillon , sur une population
Sum	Somme
Var, VarP	Variance sur un échantillon , sur une population

SELECT COUNT([N° employé])
AS Effectif FROM Employés;

L'opérateur INNER JOIN

Cet opérateur permet d'établir entre une à n tables lorsqu'à partir d'un champ commun, ce dernier contient des valeurs identiques. On parle d'équijointure. Cet opérateur s'emploie derrière l'instruction FROM. Si vous employez les opérateurs LEFT JOIN ou RIGHT JOIN, vous demandez à obtenir tous les enregistrements de la table mentionnée au niveau du membre gauche de la jointure.

FROM table1 INNER JOIN table2 ON table1.field1 compopr table2.field2

Vous pouvez employer comme opérateur de comparaison relationnelle "=", "<", ">", "<=", ">=", ou "<>".

SELECT [Nom de catégorie], [Nom du produit]
FROM Catégories INNER JOIN Produits
ON Catégories.[Nom de catégorie] = Produits.[Nom de catégorie];
SELECT fields
FROM table1 INNER JOIN table2
ON table1.field1 compopr table2.field1 AND
ON table1.field2 compopr table2.field2) OR
ON table1.field3 compopr table2.field3)];
SELECT fields
FROM table1 INNER JOIN
(table2 INNER JOIN [( ]table3
[INNER JOIN [( ]tablex [INNER JOIN ...)]
ON table3.field3 compopr tablex.fieldx)]
ON table2.field2 compopr table3.field3)
ON table1.field1 compopr table2.field2;

DELETE

L'odre DELETE permet de supprimer le contenu de champs dans une table à partir de critères logiques. On l'emploie pour détruire un ensemble d'enregistrements.

DELETE [table.]*
FROM table
WHERE criteria

UPDATE

L'instruction UPDATE vous autorise à mettre à jour une collection d'enregistrement selon une expression soumise à critères.

UPDATE table
SET newvalue
WHERE criteria;

Dans l'exemple ci-dessous, les valeurs de MontantCommande sont automatiquement multipliées par un coefficient de 1,1, les valeurs du port de 1,03 pour le pays de livraison égal au Royaume Uni.

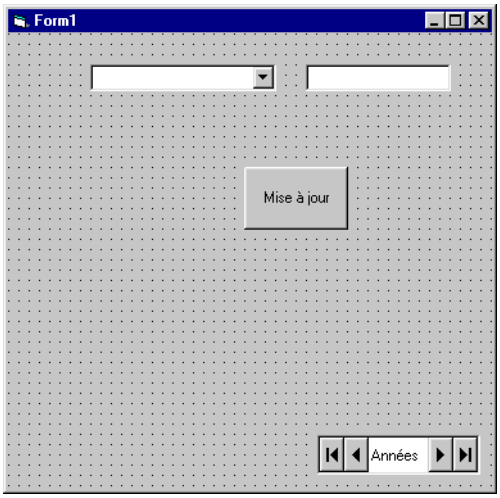
UPDATE Commandes
*SET [Montant Commande] = [Montant Commande] * 1,1,*
*Port = Port * 1,03*
WHERE [Pays livraison] = 'RU';

INSERT INTO

Cette instruction permet d'ajouter un ou plusieurs enregistrements à une table.

Pour ajouter plusieurs enregistrements :

INSERT INTO target [IN externaldatabase] [(field1[, field2[, ...]])]



UPDATE DELETE

BeginTrans
CommitTrans
RollBack

SELECT [source.]field1[, field2[, ...]
FROM tableexpression

Pour un un seul enregistrement :

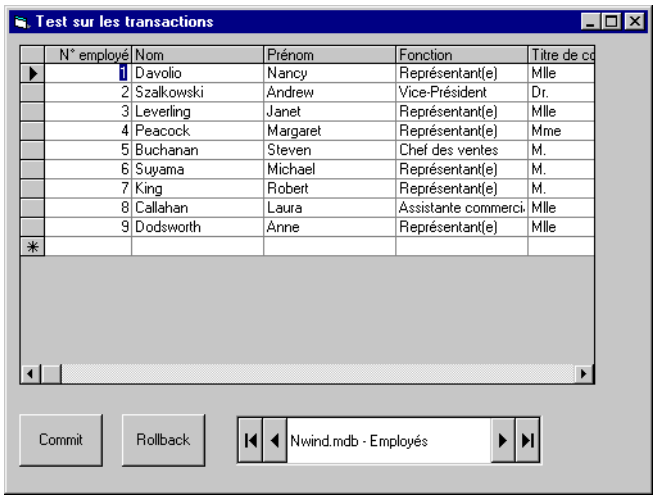
INSERT INTO target [(field1[, field2[, ...]])]
VALUES (value1[, value2[, ...])

Si, lors d'une opération d'ajout, vous ne précisez pas tous les champs, la valeur par défaut ou Null s'insère à la place des colonnes manquantes.

Vous pouvez aussi utiliser INSERT INTO pour ajouter une série d'enregistrements depuis une autre table ou requête, conjointement à la clause SELECT ... FROM comme décrit plus haut dans la syntaxe de requête Ajout avec plusieurs enregistrements. Dans ce cas, la clause SELECT mentionne les champs à ajouter à la table target spécifiée.

SELECT INTO

Cette instruction SQL permet la création d'un table.



SELECT field1[, field2[, ...]] INTO newtable
[IN externaldatabase]
FROM source

Remarques sur les instructions UPDATE et DELETE.

Elle nécessite du code... Rien que du code ! Avec même... quelques surprises ! Les bases de données de type Access exigent un mode d'ouverture de type Jet et non ODBC !

```
Private Sub Command_MAJ_Click()
        Dim dbs As Database
        Set dbs = OpenDatabase("E:\samples\cinq\Biblio97.mdb")
        dbs.Execute "UPDATE Titles " _
            & "SET [Year Published] = " & CInt(Text_NEW.Text) _
            & " WHERE [Year Published] = " & CInt(DBCombo_ANNEES.Text)
        dbs.Close
        Data_ANNEES.Refresh
End Sub
```

Les transactions

Un espace de travail définit un environnement dans lequel vous pouvez valider ou

invalider des modifications apportées à vos données : mises à jour, ajout et suppression.

Après avoir déclaré une variable de type WorkSpace au niveau d'un module ou d'un module de feuille, initialisez cette variable à l'aide de l'instruction Set.

L'espace transactionnel est défini par la méthode BeginTrans. Pour valider les modifications apportées, utilisez la méthode CommitTrans. Pour invalider, employez RollBack. Toute sortie intempestive du programme équivaut à un RollBack.

Une exemple : utilisation de la table Employés de l'application NWind.mdb

Après avoir ajouté le composant Microsoft Data Bound Grid, dessinez un Data Control et DBGrid Control. Après avoir défini les propriétés Connect, DataBaseName et RecordSource du Data Control, définissez les propriétés DataSource du DBGrid. Par un clic droit, extrayez les champs de la table par Extraire les zones.

Le code associé

```
Option Explicit
'
'Définit une variable de type environnement
'
```

```
Dim wrkCourant As Workspace

'
'Création de l'espace de travail
'
Private Sub Form_Initialize()
        Set wrkCourant = DBEngine.Workspaces(0)
End Sub
'
'Démarrage de l'espace transactionnel
'
Private Sub Form_Activate()
        wrkCourant.BeginTrans
End Sub
'
'Validation effective des modifications
'
Private Sub Command_COMMIT_Click()
        wrkCourant.CommitTrans
        Data_EMP.Refresh
        wrkCourant.BeginTrans
End Sub
'
'Invalidaton des modifications
'
Private Sub Command_ROLLBACK_Click()
        wrkCourant.Rollback
        Data_EMP.Refresh
        wrkCourant.BeginTrans
End Sub
'
'Fermeture de l'environnement
'
Private Sub Form_Unload(Cancel As Integer)
        wrkCourant.Close
        Set wrkCourant = Nothing
End Sub
```

Annexe : le code associé aux feuilles

Le code associé au formulaire de saisie

```
Private Sub Check_NAISSANCE_Click()
        If Check_NAISSANCE.Value Then
            Data_AUTEURS.RecordSource = «SELECT * FROM Authors WHERE [Year
Born] Is Not Null «
                Else
            Data_AUTEURS.RecordSource = «SELECT * FROM Authors»
        End If
        Data_AUTEURS.Refresh
End Sub

Private Sub Command_COMPTE_Click()
        Data_COMPTE.RecordSource = «SELECT COUNT(Au_ID) As Total FROM Authors»
        Data_COMPTE.Refresh
        MsgBox Data_COMPTE.Recordset.Fields(«Total»)
End Sub
```

La requête sur le formulaire "DataGrid"

```
SELECT Titles.Title,Authors.Author,Publishers.Name
FROM
(Publishers INNER JOIN Titles ON Publishers.PubID=Titles.PubID)
INNER JOIN
([Title Author] INNER JOIN Authors ON [Title Author].Au_ID=Authors.Au_ID)
```

Sélectionnez avec MAJ

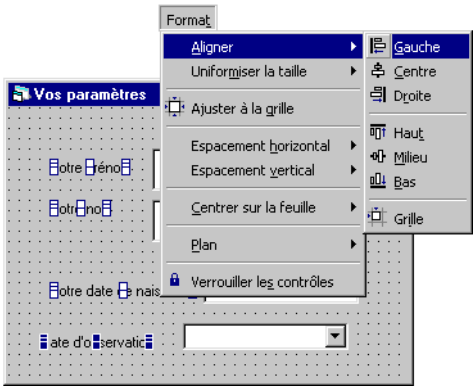
Format

Les outils d'alignement et de centrage

Après avoir incorporé un certain nombre de contrôles à votre feuille et en avoir modifié leurs propriétés, vous chercherez à organiser votre feuille.

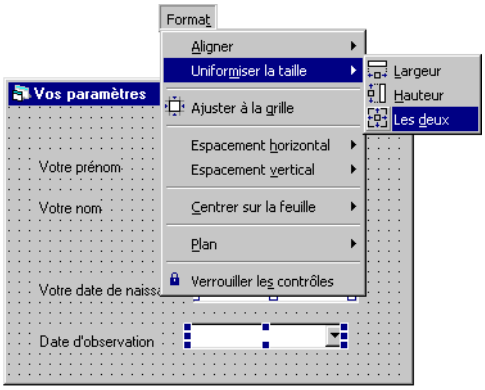
Aligner les contrôles

Sélectionnez les contrôles en maintenant MAJ appuyée ou en dessinant une surface en intersection avec les contrôles à sélectionner. Allez dans Format | Aligner.



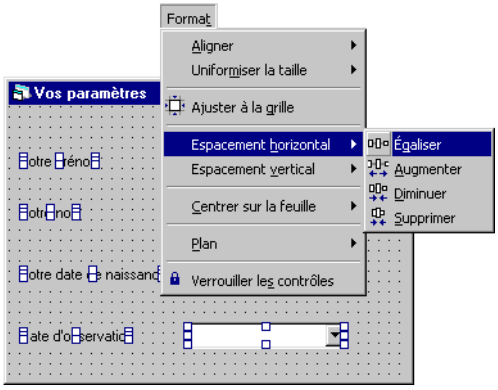
Uniformiser la taille des composants

Pour que les zones de saisie disposent des mêmes propriétés en hauteur et en largeur, allez dans Format | Uniformiser.



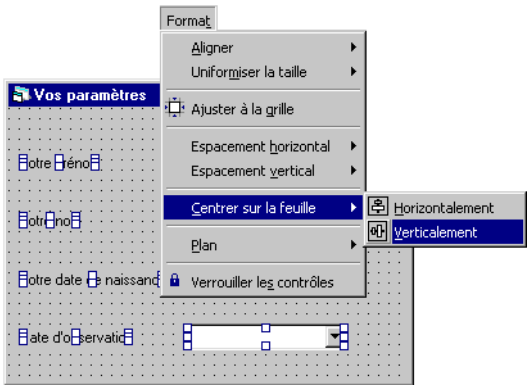
Harmoniser les espacements

Si vous souhaitez automatiquement répartir l'espace entre les composants, allez dans Format | Espacement.



Centrer dans la feuille

Si vous poussez le comble de l'esthétisme à ce que tous les éléments se répartissent harmonieusement dans la feuille, alors choisissez Format | Centrer dans la



feuille.

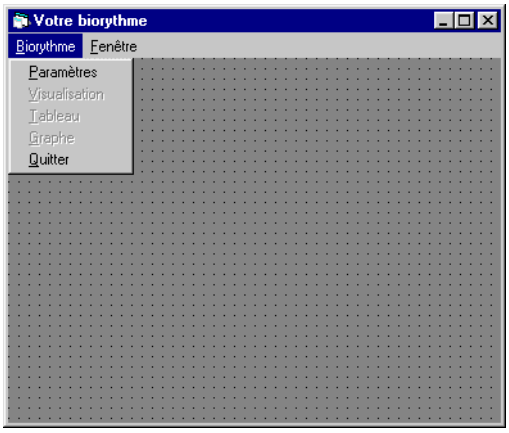
Calculez votre biorythme

Le biorythme est un concept "ésotérique" auquel se réfèrent certains sportifs pour obtenir des "performances". Il est basé sur des cycles vis à vis de la date de naissance.

Les modules

Ils ont pour objectif de stocker les fonctions, procédures, constantes et variables communes (et globales) au projet. Remarquez que l'ordre n'a pas d'importance.

```
'
' Retourne un tableau contenant les différents biorythmes
'
Function RESULTATS_BIO(DN As Date, DJ As Date) As Variant
    Dim Var_PHY As Double, Var_EMO As Double, Var_INT As Double, Var_MOY As
Double
    Var_PHY = CALCUL_BIO(DN, DJ, 23)
    Var_EMO = CALCUL_BIO(DN, DJ, 28)
    Var_INT = CALCUL_BIO(DN, DJ, 33)
    Var_MOY = (Var_PHY + Var_EMO + Var_INT) / 3
    RESULTATS_BIO = Array(Var_PHY, Var_EMO, Var_INT, Var_MOY)
End Function
'
' Le calcul du biorythme
'
Function CALCUL_BIO(DN As Date, DJ As Date, PERIODE As Integer) As Double
    CALCUL_BIO = (Sin(2 * Pi * (DJ - DN) /
PERIODE) + 1) / 2
End Function
'
' La "constante" PI n'existe pas !
'
Function Pi() As Double
    Pi = 4 * Atn(1)
End Function
```



La forme MDI

Elle contient le menu de l'application.

```
Private Sub
```

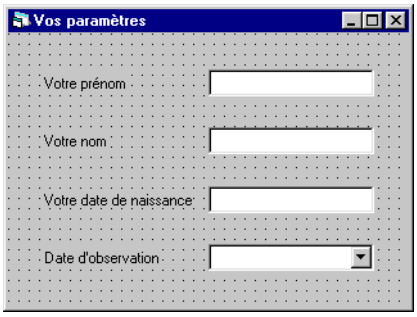
PopupMenu

Show
Hide
Unload

```
CommandeMenu_PARAM_Click()
    Hide
    '
    'Active les autres éléments du menu
    '
    CommandeMenu_VISU.Enabled = True
    CommandeMenu_TAB.Enabled = True
    CommandeMenu_GRAPH.Enabled = True
    Form_PARAM.Show
End Sub


Private Sub CommandeMenu_VISU_Click()
    '
    'La procédure Hide masque la feuille active
    '
    Hide
    'If Form_PARAM.Visible Then
    '   Unload Form_PARAM
    'End If
    Form_VISU.Show
End Sub
Private Sub CommandeMenu_TAB_Click()
    Hide
    Form_TAB.Show
End Sub
Private Sub CommandeMenu_GRAPH_Click()
    Hide
    Form_GRAPH.Show
End Sub
'
'Procédure liée à la commande Quitter du Menu
'
Private Sub CommandeMenu_QUIT_Click()
    '
    'Déchargement de l'application
    '
    Unload Me
End Sub

'
'Gestion du menu contextuel
'
Private Sub MDIForm_MouseDown(Button As Integer, Shift As Integer, X As Single, Y
As Single)
    If Button = 2 Then
      PopupMenu Menu_BIO
    End If
End Sub
'
' Ensemble de procédures liées au menu Fenêtre
'
Private Sub CommandeMenu_CASCADE_Click()
    MDIForm_MAIN.Arrange vbCascade
End Sub

Private Sub CommandeMenu_HORI_Click()
```

Label
Text

```
    MDIForm_MAIN.Arrange vbTileHorizontal
End Sub

Private Sub CommandeMenu_REORG_Click()
    MDIForm_MAIN.Arrange vbArrangeIcons
End Sub

Private Sub CommandeMenu_VERT_Click()
    MDIForm_MAIN.Arrange vbTileVertical
End Sub
```

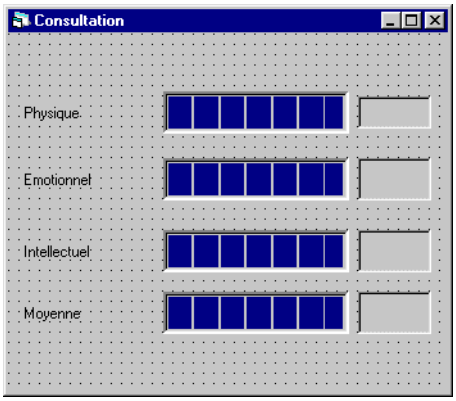


La saisie des éléments nécessaires

Le contrôle de saisie est très insuffisant dans l'exemple. Vous vous emploierez à mieux le gérer par la suite.

```
'
' Chargement d'une combo de 30 éléments
'
Private Sub Form_Load()
    Dim I As Integer
    Dim Tab_OBS(29) As String * 10
    For I = 0 To 29
      Tab_OBS(I) = Format(Date + I, "dd/mm/yyyy")
      Combo_OBSERVATION.AddItem Tab_OBS(I)
    Next I
    Combo_OBSERVATION.Text = Tab_OBS(0)
End Sub
'
' Test et formatage de la date de naissance
'
Private Sub Text_NAISSANCE_LostFocus()
    If IsDate(Text_NAISSANCE.Text) Then
      Text_NAISSANCE.Text =
Format(Text_NAISSANCE.Text, "dd/mm/yyyy")
    Else
      MsgBox "Date erronée", vbOKOnly, «Erreur»
    End If
End Sub
```

Label
Text
ProgressBar

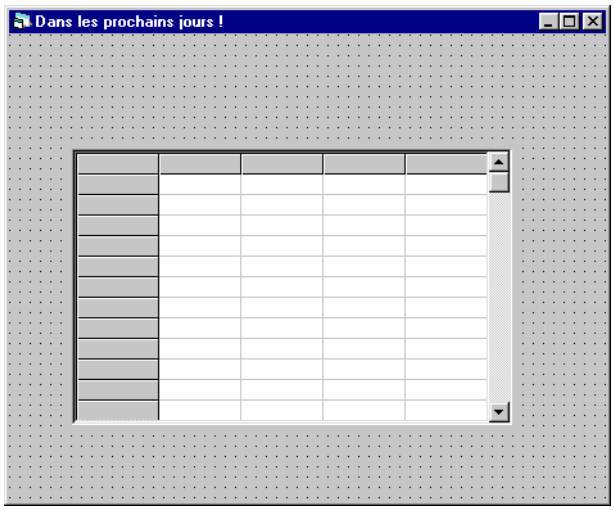


La visualisation des résultats

L'affichage des résultats nécessite l'emploi de composants supplémentaires (Microsoft Windows Common Control 5). Je vous demande de gérer un affichage animé des résultats au niveau des Progress Bars.

```
'
' Affichage des résultats par des étiquettes et des Progress Bar.
'
Private Sub Form_Load()
    Dim Var_DJ As Date, Var_DN As Date
    Dim TAB_RESULTATS As Variant
    Var_DN = CDate(Form_PARAM.Text_NAISSANCE.Text)
    Var_DJ = CDate(Form_PARAM.Combo_OBSERVATION.Text)
    TAB_RESULTATS = RESULTATS_BIO(Var_DN, Var_DJ)
    Label_RESPHY = Format(TAB_RESULTATS(0), "0.00 %")
    Label_RESEMO = Format(TAB_RESULTATS(1), "0.00 %")
```

```
    Label_RESINT = Format(TAB_RESULTATS(2), "0.00 %")
    Label_RESMOY =
Format(TAB_RESULTATS(3), "0.00
%")
    ProgressBar_PHY.Value =
Int(TAB_RESULTATS(0) * 100)
    ProgressBar_EMO.Value =
Int(TAB_RESULTATS(1) * 100)
    ProgressBar_INT.Value =
Int(TAB_RESULTATS(2) * 100)
    ProgressBar_MOY.Value =
Int(TAB_RESULTATS(3) * 100)
End Sub
```

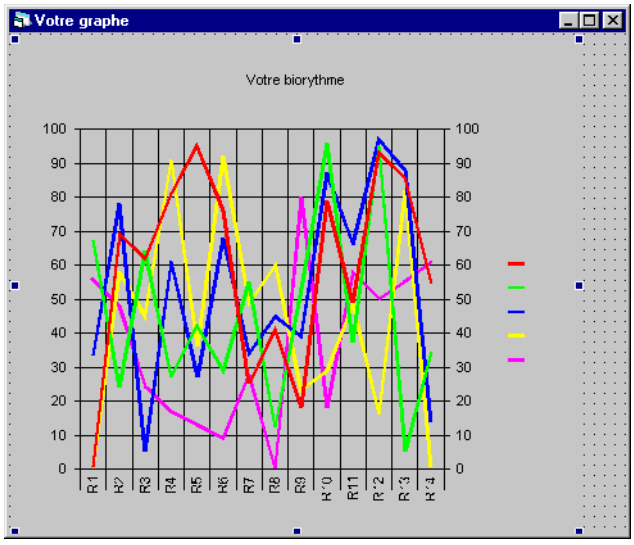


L'affichage dans un tableau de type MSFlexGrid

Pour utiliser ce type de tableau, vous devez l'ajouter à votre boîte à outils (Microsoft Flexgrid Control). La nécessité de passer par la procédure événementielle masque un réel problème. Vous pourrez l'observer dans l'exemple précédent si, après avoir changé les paramètres du premier menu, vous visualisez les résultats à partir du deuxième menu. Pourquoi ? Dommage que nous ne puissions pas "envoyer" un tableau directement dans ce composant.

```
Private Sub Form_Activate()
    MSFlexGrid_BIO.Visible = False
    Dim TAB_RESULTATS As Variant
    Dim Var_DJ As Date, Var_DN As Date
    Dim I As Integer, MAX As Integer
    Var_DN = CDate(Form_PARAM.Text_NAISSANCE.Text)
    MAX = MSFlexGrid_BIO.Rows - 1
    Var_DJ = CDate(Form_PARAM.Combo_OBSERVATION.Text)
    MSFlexGrid_BIO.Row = 0
    MSFlexGrid_BIO.COL = 0
    MSFlexGrid_BIO.Text = Var_DN
    MSFlexGrid_BIO.COL = 1
    MSFlexGrid_BIO.Text = "Physique"
    MSFlexGrid_BIO.COL = 2
    MSFlexGrid_BIO.Text = "Emotionnel"
    MSFlexGrid_BIO.COL = 3
    MSFlexGrid_BIO.Text = "Intellectuel"
    MSFlexGrid_BIO.COL = 4
    MSFlexGrid_BIO.Text = "Moyenne"
    For I = 1 To MAX
      MSFlexGrid_BIO.COL = 0
      MSFlexGrid_BIO.Row = I
      MSFlexGrid_BIO.Text = Var_DJ
      Var_DJ = Var_DJ + 1
      TAB_RESULTATS = RESULTATS_BIO(Var_DN, Var_DJ)
      MSFlexGrid_BIO.COL = 1
      MSFlexGrid_BIO.Text = Format(TAB_RESULTATS(0), "0.00 %")
      MSFlexGrid_BIO.COL = 2
      MSFlexGrid_BIO.Text = Format(TAB_RESULTATS(1), "0.00 %")
```

```
    MSFlexGrid_BIO.COL = 3
    MSFlexGrid_BIO.Text = Format(TAB_RESULTATS(2), "0.00 %")
    MSFlexGrid_BIO.COL = 4
    MSFlexGrid_BIO.Text = Format(TAB_RESULTATS(3), "0.00 %")
```



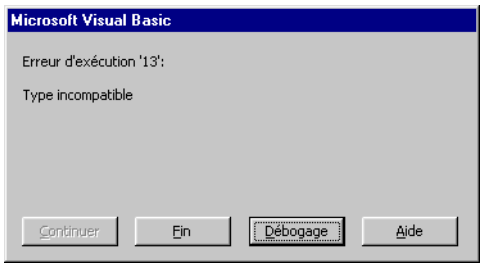

```
    Next I
    MSFlexGrid_BIO.Visible = True
End Sub
```

L'affichage par un graphique

Pour réaliser un contrôle, employez le composant Microsoft Chart Control. Les problèmes sont identiques à ceux que nous avons pu pointer précédemment.

```
'
' Indice des tableaux à 1
'
Option Base 1
Private Sub Form_Load()
    Dim LIG As Integer
    Dim Tab_GRAPH()
    Dim TAB_RESULTATS As Variant
    Dim Var_DJ As Date, Var_DN As Date
    Var_DN = CDate(Form_PARAM.Text_NAISSANCE.Text)
    Var_DJ = CDate(Form_PARAM.Combo_OBSERVATION.Text)
    ReDim Tab_GRAPH(15, 5)
    Tab_GRAPH(1, 2) = "PHY"
    Tab_GRAPH(1, 3) = "EMO"
    Tab_GRAPH(1, 4) = "INT"
    Tab_GRAPH(1, 5) = "MOY"
    For LIG = 2 To 14
      Tab_GRAPH(LIG, 1) = Var_DJ
      TAB_RESULTATS = RESULTATS_BIO(Var_DN, Var_DJ)
      Tab_GRAPH(LIG, 2) = TAB_RESULTATS(0)
      Tab_GRAPH(LIG, 3) = TAB_RESULTATS(1)
      Tab_GRAPH(LIG, 4) = TAB_RESULTATS(2)
      Tab_GRAPH(LIG, 5) = TAB_RESULTATS(3)
      Var_DJ = Var_DJ + 1
```

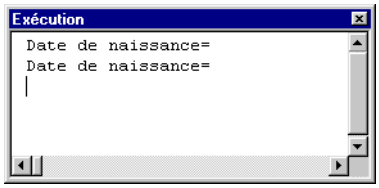
```
Next LIG
'
' Le composant accepte un tableau de données.
'
MSChart_BIO.ChartData = Tab_GRAPH
```




```
End Sub
```

Le débogage

Debug.Print



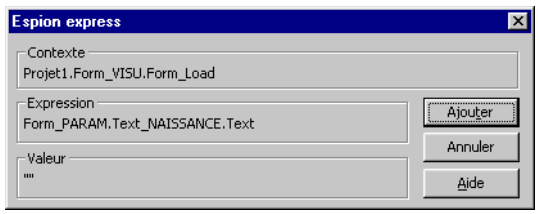
C'est l'opération par laquelle vous tentez de cerner les problèmes de programmation : erreurs de type, division par 0, dépassement de piles, etc. Dans la phase d'édition et de test, évitez le plus possible l'utilisation du On Error Goto.

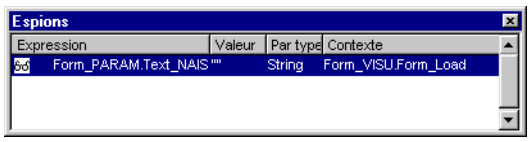
L'instruction Debug.Print

Insérée dans le programme, elle renvoie une chaîne de caractères dans la fenêtre

```
Private Sub Form_Load()
    Dim Var_DJ As Date, Var_DN As Date
    Debug.Print "Date de naissance=", Form_PARAM.Text_NAISSANCE.Text
    Var_DN = CDate(Form_PARAM.Text_NAISSANCE.Text)
    Var_DJ = CDate(Fo Form_PARAM.Text_NAISSANCE.Text = "".Text)
    TAB_RESULTATS = RESULTATS_BIO(Var_DN, Var_DJ)
    Label_RESPHY = Format(TAB_RESULTATS(0), "0.00 %")
    Label_RESEMO = Format(TAB_RESULTATS(1), "0.00 %")
    Label_RESINT = Format(TAB_RESULTATS(2), "0.00 %")
    Label_RESMOY = Format(TAB_RESULTATS(3), "0.00 %")
    ProgressBar_PHY.Value = Int(TAB_RESULTATS(0) * 100)
    ProgressBar_EMO.Value = Int(TAB_RESULTATS(1) * 100)
    ProgressBar_INT.Value = Int(TAB_RESULTATS(2) * 100)
    ProgressBar_MOY.Value = Int(TAB_RESULTATS(3) * 100)
End Sub
```

Exécution. **Evidemment, vous avez intérêt à la positionner avant le "plantage", si possible en la marquant d'un point d'arrêt** (Débogage | Basculer le point

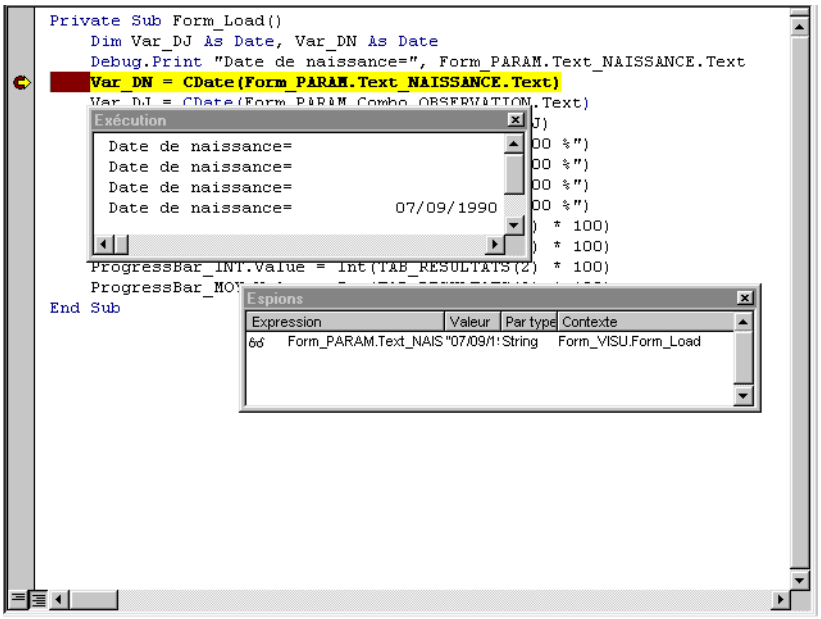




d'arrêt).

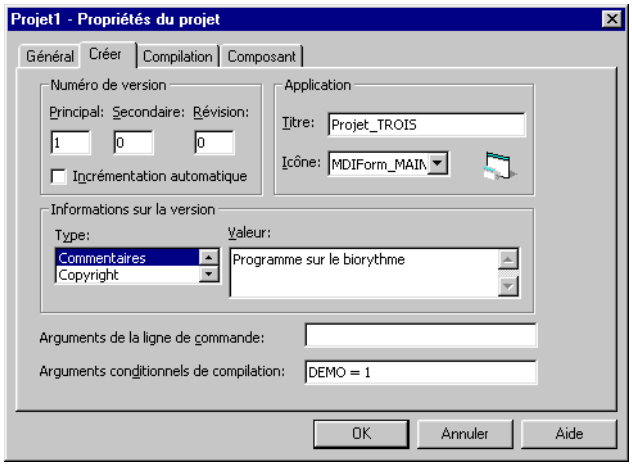
Les espions

Ce sont des expressions de votre programme que vous décidez de suivre durant l'exécution du programme. La première méthode consiste en arrière-plan à placer la



souris sur la valeur. Elle s'affiche alors dans une info-bulle.

Une autre méthode consiste à ajouter un espion en allant dans Débogage | Espion express. Vous pouvez alors ajouter l'espion à une liste qui apparaît en exécution.



Les points d'arrêts

Ils permettent de suspendre l'exécution du programme. Pour positionner ou enlever un point d'arrêt, sélectionnez la ligne et allez dans Débogage | Basculer le point d'arrêt. Pour tous les retirer, choisissez Débogage | Effacer tous les points d'arrêt.

La compilation conditionnelle

A quoi peut-elle servir ? Tout simplement, à distribuer des versions de démonstrations ou à optimiser selon les plate-formes.

#Const
#if...
#else if...
#else...
#end if

Globale au projet

Allez dans Projet | Propriétés du projet. Entrez vos arguments dans la zone Arguments conditionnels de compilation.

Dans le code, observez l'instruction #if...#else...#end if.

```
Private Sub CommandeMenu_PARAM_Click()
    Hide
    '
    'Active les autres éléments du menu
    '
    CommandeMenu_VISU.Enabled = True
    #if not DEMO then
      CommandeMenu_TAB.Enabled = True
      CommandeMenu_GRAPH.Enabled = True
    #end if
    Form_PARAM.Show
End Sub
```

Locale au module

Vous pouvez aussi utiliser une constante contenant la valeur de l'argument conditionnel préfixé par #Const en en-tête du module.

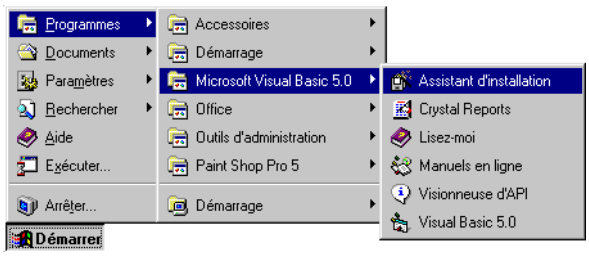
```
#Const DEMO=True
Private Sub CommandeMenu_PARAM_Click()
    Hide
    '
    'Active les autres éléments du menu
    '
    CommandeMenu_VISU.Enabled = True
    #if not DEMO then
      CommandeMenu_TAB.Enabled = True
      CommandeMenu_GRAPH.Enabled = True
    #end if
    Form_PARAM.Show
End Sub
```

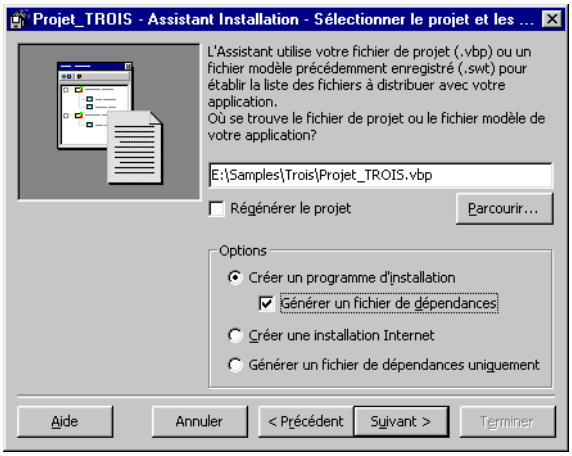
La gestion de la ligne de commande

L'exemple ci-dessous vous permet de voir Comment employer la fonction Command() qui retourne sous forme de chaîne de caractères le contenu d'arguments tapés en suffixe à la ligne de commande. L'exemple ci-dessous est sensé tenir compte du paramètre pour déterminer le mode d'affichage.

```
Sub Main()
    '
    ' Initialisation d'un tableau contenant trois valeurs
    '
    Dim VALEURS
    VALEURS = Array("0", "1", "2")
    '
    ' Récupération du contenu de la ligne de commande
    '
    ligne_COMMANDE = Command()
    '
    ' Balayage du tableau
    '
    For Each ELEMENT In VALEURS
      '
      ' Test pour savoir si la ligne de commande contient l'une des valeurs du tableau
      '
```

```
        If ligne_COMMANDE = ELEMENT Then
            '
            ' Changement du mode de
démarrage en fonction du contenu de
la ligne de commande
            '

MDIForm_MAIN.WindowState =
Command()
            Exit For
        End If
    Next ELEMENT
```



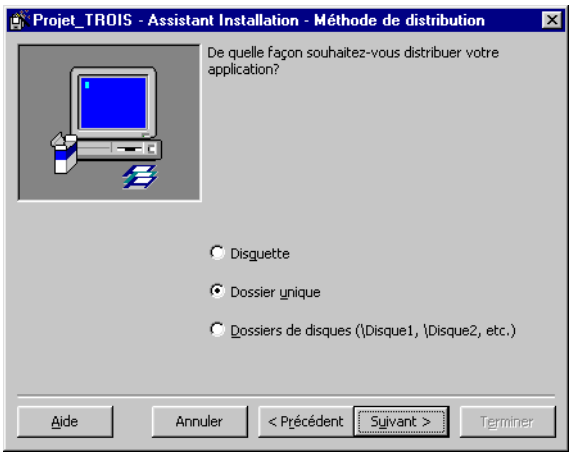


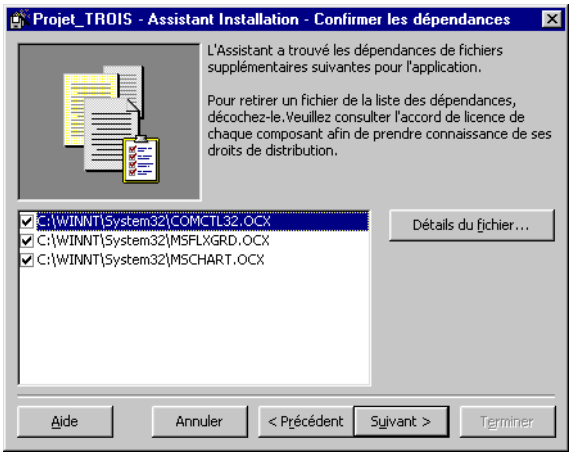

```
    '
    ' Affichage de la feuille
    '
    MDIForm_MAIN.Show
```

```
End Sub
```

L'assistant installation

L'assistant va réaliser l'empaquettage

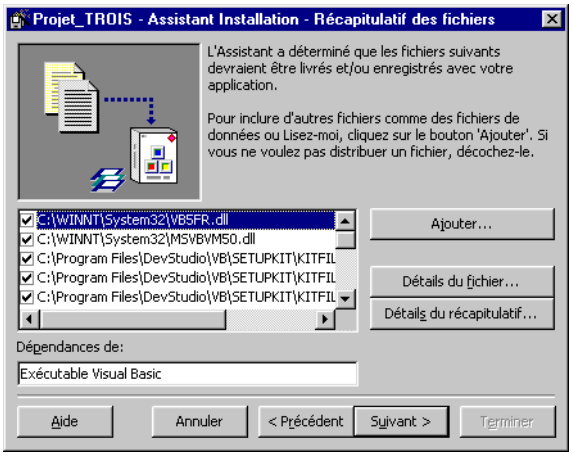


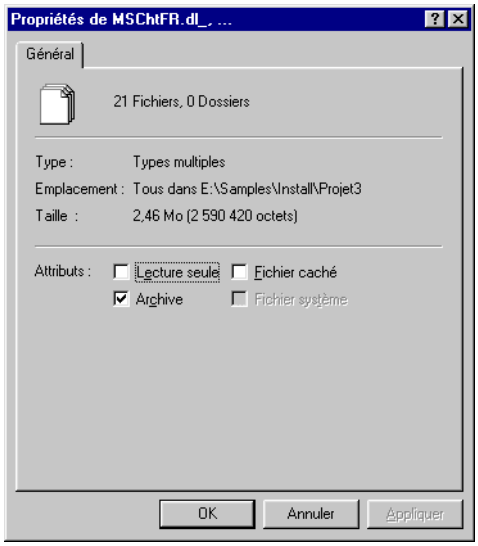


de votre projet de façon à le distribuer par disquettes, CD Rom ou unités de serveurs de réseau local. A partir du menu Démarrer, choisissez

Programmes | Microsoft Visual Basic 5 | Assistant installation.

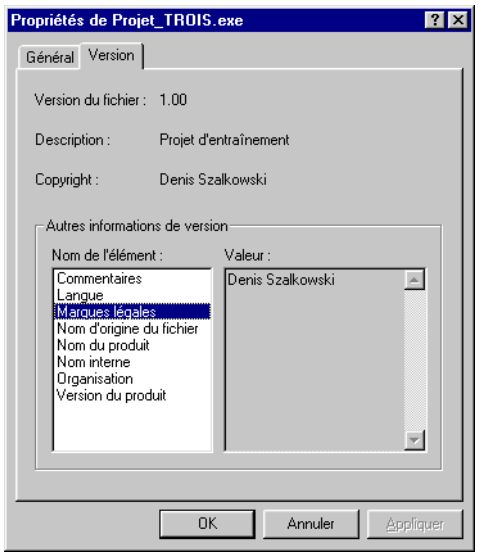
La première étape est de choisir votre projet ou le profil d'un précédent empaquettage (fichier swt).





La deuxième étape est le type de distribution. Vous devez ensuite sélectionner le dossier de votre disque recevant les fichiers nécessaires à l'installation.

Visual Basic Entend par dépendance tout contrôle sur lequel repose votre application. Ce sont donc les composants OCX contenant ces différents contrôles.



L'assistant installation affiche la liste de tous les éléments nécessaires à l'installation .Lors de l'étape suivante, vous pouvez enregistrer un profil dans un modèle. Générez votre empaquettage en cliquant sur Terminer.

Au final, la capacité en partie compressée de votre application est 2,46 Mo ! Il faut dire que nous n'avons pas utilisé beaucoup de composants.

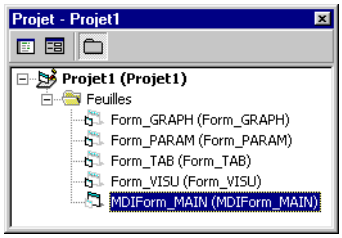
3. Environnement MDI et SDI

Single Multiple Document Interface

Définitions

Un environnement MDI ou Multiple Document Interface signifie que vous pouvez "encapsuler" des feuilles-filles au sein d'une feuille-mère. Lorsque vous souhaitez gérer un composant de type menu dans une application, il vaut mieux l'intégrer dans une feuille de type MDI. Les feuilles par défaut sont de type SDI.

Créez plusieurs feuilles conformément aux spécifications ci-dessous.

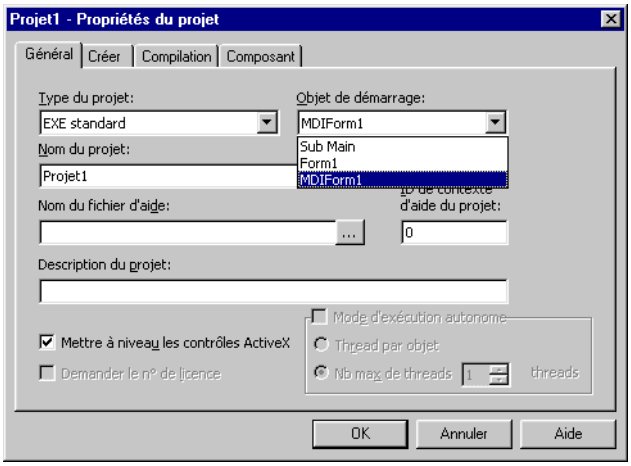


Pour faire en sorte qu'une feuille soit fille d'une feuille MDI, dans les propriétés, modifiez la valeur MDIChild en la passant à True.

Déterminer le lanceur du projet

Projet Propriétés

Le lanceur d'un projet VB peut être ou une feuille ou une procédure Main (sans arguments) appartenant à un module. Pour déterminer le lanceur, allez dans Projet | Propriétés du projet.



Pour lancer l'application à partir d'un module, entrez une procédure du type suivant :

```
Sub Main()
        MDIForm_MAIN.Show
End Sub
```

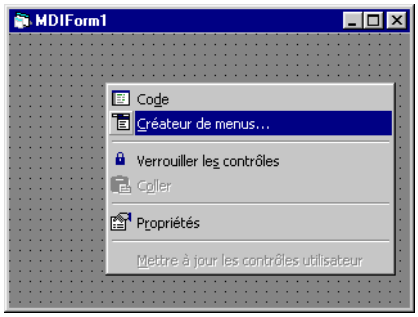
Créateur de menus

Windows List

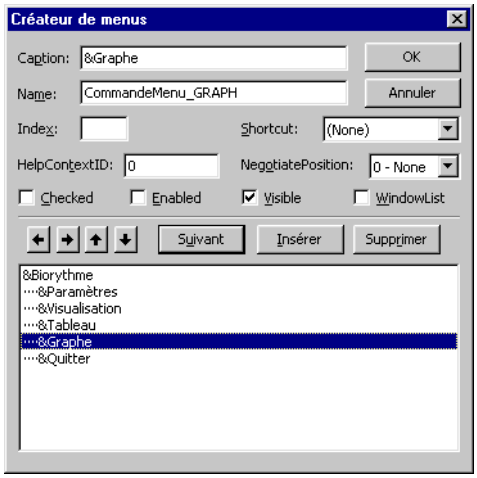
Créer un composant de type Menu

Le créateur de menus

A partir de la feuille MDI, faites un clic droit et choisissez Créateurs de menus.



Tapez le titre de la commande dans la zone Caption. Si vous faites précéder une lettre par &, vous pouvez accéder par raccourci clavier du type ALT+Lettre à la commande. Donnez un nom en le préfixant dans la zone Name.



Les propriétés Visible, Enabled, Checked vous permettent de rendre visible ou invisible, accessible ou inaccessible, coché ou non coché.

La propriété Windows List

Le fait de cocher *Windows List* pour un menu signifie qu'il listera par leur nom toutes les fenêtres actives filles incluses ou liées à la feuille MIDI.

L'organisation des feuilles filles en mosaïque, en cascade ou en organisant les icônes se réalise par le code associé à un élément de menu.

```
Private Sub CommandeMenu_HORIZONTAL_Click()
        MDIForm_MAIN.Arrange vbTileHorizontal
End Sub
Private Sub CommandeMenu_VERTICAL_Click()
        MDIForm_MAIN.Arrange vbTileVertical
End Sub
Private Sub CommandeMenu_CASCADE_Click()
        MDIForm_MAIN.Arrange vbCascade
End Sub
Private Sub CommandeMenu_ORGANISER_Click()
        MDIForm_MAIN.Arrange vbArrangeIcons
End Sub
```

PopupMenu

Show
Hide
Unload

Associer une procédure événementielle à une commande de menu

A partir de la feuille MDI, cliquez sur la commande de menu. L'événement géré est du type NomContrôle_Click. Pour accéder à l'élément et aussi pour qu'il s'exécute, il doit être visible et activé.

Rien ne vous empêche par la fenêtre Code à entre le code lié à l'événement.

Le menu contextuel

Le menu contextuel n'est autre qu'un menu de commandes au sein du créateur de menus auquel vous faites appel par la méthode object.PopupMenu NomMenu. L'événement le plus approprié à la gestion de ce menu contextuel est MouseDown.

```
Private Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
        If Button = 2 Then
            PopupMenu Menu_BIO
        End If
End Sub
```

Afficher les fenêtres

Il existe plusieurs méthodes (procédures liés à un objet) pour manipuler les fenêtres.

Pour afficher (et charger) une feuille, employez la méthode Feuille.Show. Pour masquer une feuille déjà chargée, utilisez la méthode Hide. Enfin pour quitter l'application, déchargez la forme principale par Unload Me.

```
Private Sub CommandeMenu_PARAM_Click()
        Hide
        '
        'Active les autres éléments du menu
        '
        CommandeMenu_VISU.Enabled = True
        CommandeMenu_TAB.Enabled = True
        CommandeMenu_GRAPH.Enabled = True
        Form_PARAM.Show
End Sub
Private Sub CommandeMenu_VISU_Click()
        '
        'La procédure Hide masque la feuille active
        '
        Hide
        'If Form_PARAM.Visible Then
        '   Unload Form_PARAM
        'End If
        Form_VISU.Show
End Sub
Private Sub CommandeMenu_TAB_Click()
        Hide
        Form_TAB.Show
End Sub
Private Sub CommandeMenu_GRAPH_Click()
        Hide
        Form_GRAPH.Show
End Sub
'
'Procédure liée à la commande Quitter du Menu
'
Private Sub CommandeMenu_QUIT_Click()
```

```
        '
        'Déchargement de l'application
        '
        Unload Me
    End Sub
```

Pour fermer un ensemble de feuilles ouvertes, vous pouvez prendre exemple sur le code suivant :

```
    Private Sub Menu_QUITTER_Click()
        Dim I As Integer, N As Integer
        N = Forms.Count - 1
        If N > 0 Then
            For I = N To 1 Step -1
                Unload Forms(I)
            Next I
        End If
    End Sub
```

Feuilles modales

Ouvrir de façon modale une feuille d'un projet signifie qu'elle soit fermée (déchargée) par l'utilisateur afin de revenir à la feuille qui l'a appelée.

Pour ouvrir une feuille en mode modal, utilisez la syntaxe suivante :

```
feuille.show vbModal
```

Boîtes de dialogue standard

Vous pouvez utiliser les fonctions standards InputBox et MsgBox à l'instar de ce qui existe en VBA.

La fonction InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context]) retourne une chaîne correspondant à la valeur saisie si vous cliquez sur OK ou appuyez sur Entrée, la chaîne nulle "" si vous activez Annuler ou Echap au clavier.

L'instruction MsgBox peut-être employée en tant que procédure ou fonction :

```
MsgBox(prompt[, buttons] [, title] [, helpfile, context])
```

Si vous l'employez comme fonction, elle retourne une valeur entière correspondant au bouton cliqué.

' _ :
Commentaire
Instructions

Règles d'écriture

Les commentaires

Ils sont précédés par le caractère apostrophe. Dans l'optique de transmettre un projet compréhensible à une personne tierce, leur usage est plus que recommandé.

Prolonger une instruction sur plusieurs lignes

Pour signifier à l'éditeur de prolonger une commande à la ligne suivante, tapez en bout de ligne un caractère underscore _ ou tiret bas précédé d'un espacement.

Saisir plusieurs instructions sur une même ligne

Bien que cela ne soit pas très recommandé, séparez vos instructions par le signe deux-points :.

Les opérateurs

Liste des opérateurs

Arithmétique	Comparaison	Logique
Puissance (^)	Egalité (=)	Not
Négation (-)	Différence (<>)	And
Multiplication et division (*, /)	Plus petit que (<)	Or
Division entière (\)	Plus grand que (>)	Xor
Modulo (Mod)	Plus petit ou égal à (<=)	Eqv
Addition et soustraction (+, -)	Plus grand ou égal à(>=)	Imp
Concaténation de chaînes (&)	Like	
	Is	

L'opérateur Like

Il permet de comparer deux chaînes de caractères.

resultat = expression Like modèle

Si l'expression est comparable au modèle, le résultat est vrai. Si l'une au moins des deux parties est nulle, le résultat est nulle. Par défaut, Visual Basic fait la distinction entre les minuscules et les majuscules sauf si figure en en-tête de module Option Compare Text.

Caractère	Signification dans le modèle
?	N'importe quel caractère
*	Ignore ce qui suit
#	N'importe quel chiffre (0-9)
[Liste_de_caractères]	N'importe quel caractère dans la liste
[!Liste_de_caractères]	N'importe que autre caractère que ceux dans la liste

Pour rechercher les caractères ?, #, entrez-les entre crochets. Vous pouvez utiliser le signe moins pour spécifier un intervalle de recherche : [A-Z]. Vous pouvez spécifier plusieurs intervalles de Recherche sans délimiteur : [a-zA-Z0-9]. L'opérateur Like est capable de retrouver le caractère «æ» s'il est comparé à «ae».

Conventions de lecture

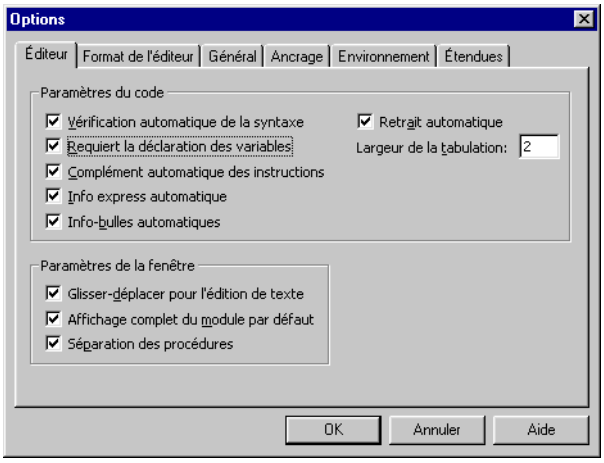
Option Explicit

Les crochets indiquent des éléments facultatifs. Le "pipe" | signifie une alternative. Les accolades signifient un choix obligatoire dans un ensemble de solutions.

Variables, types et constantes

Les identificateurs

Ils ont limités à 255 caractères maximum, le premier d'entre eux étant



nécessairement une lettre. Vous avez le droit aux caractères accentués (quoique dangereux pour le portage de votre application), aux chiffres, au caractères underscore. Sont interdits les espaces, les opérateurs +, -, *, /, \, ^, &

Les types

Type de données	Taille d'enregistrement	Plage
Byte	1 octet	0 255
Boolean	2 octets	True False
Integer	2 octets	-32 768 32 767
Long	4 octets	-2 147 483 648 2 147 483 647
Single	4 octets	-3,402823E38 -1,401298E-45 1,401298E-45 3,402823E38
Double		8 octets -1,79769313486232E308 -4,94065645841247E-324 4,94065645841247E-324 1,79769313486232E308
Currency	8 octets	-922 337 203 685 477,5808 922 337 203 685 477,5807
Decimal	14 octets	+/- 79 228 162 514 264 337 593 543 950 335 +/- 7,9228162514264337593543950335 +/- 0.0000000000000000000000000001
Date	8 octets	1er janvier 100 31 décembre 9999
Object	4 octets	

Le type Variant

String	10 octets + chaîne	0 à environ 2 milliards
String * n	n octets	1 à environ 65 400
Variant	16 octets	Numérique : double
Variant	22 octets + chaîne	String * n

Le type variant

Le type variant est susceptible d'accueillir tout type de données. C'est le mode de gestion des variables si vous ne forcez pas le mode de déclaration explicite. De la même façon, si vous ne typez pas votre variable, elle est considérée dans le type Variant.

Pour passer en mode explicite, allez dans Outils | Options | Editeur et cochez Requiert la déclaration des variables. Une autre solution consiste à écrire en en-tête de module Option Explicit.

La portée

Public	L'élément est alors disponible pour toutes les procédures de l'ensemble des modules. Ce mot clé ne peut pas être utilisé au sein des procédures.
Private	L'élément est alors disponible pour toutes les procédures du module. Ce mot clé ne peut pas être utilisé au sein des procédures.

La déclaration de constantes

[Public | Private] Const constname [As type] = expression

type	Byte, Boolean, Integer, Long, Currency, Single, Double, Decimal, Date, String ou Variant
expression	Valeur littérale, autre constante ou combinaison pouvant contenir tout opérateur arithmétique ou logique à l'exception de Is.

Les énumérations de constantes

Les constantes sont alors de type Long.

```
[Public | Private] Enum name
        membername [= constantexpression]
        membername [= constantexpression]
            . . .
End EnumSet
```

La déclaration de variables

Lorsque les variables sont déclarées à l'intérieur de la forme ou du module, sa portée est locale.

Dim NomVariable [As [New] type]

Lors de l'initialisation des variables, une variable numérique prend pour valeur initiale 0, une chaîne de longueur variable prend pour valeur initiale une chaîne de longueur nulle (""), et une chaîne de longueur fixe est remplie de zéros. Les variables Variant ont la valeur Empty à l'initialisation.

Les fonctions sur la gestion et conversion de type

VarType(varname)	Renvoie une valeur de type Integer qui indique le sous-type d'une variable varname.
TypeName(varname)	Retourne une chaîne correspondant au type d'une valeur varname de type Variant
Fonctions d'information	IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric,

IsObject

Fonctions de conversion Asc, CBool, CByte, CCur, CDate, CDec, CDbl, Chr, CInt, CLng, CSng, CStr, CVar, CVErr, Format, Hex, Oct, Str, Val

Attention ! Les fonctions de conversion provoquent des erreurs si la conversion échoue.

Création de types personnalisés

Le langage ne connaît pas la notion de salarié. Vous pouvez le définir en créant un type composé de différents éléments.

```
[Private | Public] Type varname
        elementname1 [([dimension1])] As type
        [elementname2 [([dimension2])] As type]
        . . .
End Type
```

Pour utiliser le type ainsi défini, déclarez une variable se référant du type créé.

```
Dim Variable As Varname
With Variable
        elementname1=...
End With
ou
Variable.elementname1=...
```

Les définitions de type par défaut

Très pratique, cette possibilité que vous offre le langage associe le type à la première lettre des identificateurs dans les formes et modules de votre projet.

```
DefInt letterrange [,letterrange] . . .
DefLng letterrange [,letterrange] . . .
DefSng letterrange [,letterrange] . . .
DefDbl letterrange [,letterrange] . . .
DefCur letterrange [,letterrange] . . .
DefStr letterrange [,letterrange] . . .
DefVar letterrange [,letterrange] . . .
```

Les tableaux

La déclaration de tableaux

Lorsque les variables sont déclarées à l'intérieur de la forme ou du module, sa portée est locale.

```
Dim NomVariable ([Indice_mini_1 To ] Indice_Maxi_1[,...,[Indice_mini_n To]
Indice_Maxi_n])] [As [New] type]
```

Des parenthèses vides au niveau d'un tableau permettent de créer un tableau dynamique. Utilisez la commande Redim à l'intérieur des procédures pour redimensionner le tableau. Les tableaux s'initialisent à l'instar de variables classiques.

Redim

L'instruction Redim permet de modifier la dimension des tableaux dynamiquement.

```
ReDim [Preserve] NomVariable(Indices) [As type]
```

L'emploi de la clause Preserve permet la Conservation des anciennes valeurs dans le cadre du redimensionnement. La dimension maximale en terme d'indices d'un tableau est de 60 éléments.

```
ReDim TABLEAU(8,3)
ReDim A(0 To 8, 0 To 3)
```

Fonction Array

Denis Szalkowski

Instructions conditionnelles

Instructions répétitives

ReDim A(8, 0 To 3)

Vous pouvez redimensionner un tableau en en changeant son type.

Option Base {0|1}

Par défaut, la borne inférieure est de 0. La commande Option Base {0|1} vous permet de spécifier l'indice de départ.

Array(arglist)

Array est une fonction qui vous permet d'initialiser un tableau sous forme d'une liste. L'argument arglist est une liste de valeurs délimitées par des virgules. Chaque élément charge le tableau. Si l'argument est omis, un tableau de longueur nulle est créé. La fonction renvoi une collection de Variant.

```
Dim TABLEAU As Variant
TABLEAU = Array(1,2,3)
ELEMENT = TABLEAU(2)
```

La limite inférieure d'un tableau créé à l'aide de la fonction Array est toujours égale à zéro. Il n'obéit pas à l'instruction Option Base.

Les fonctions LBound et UBound

Elles renvoient chacune une valeur de type Long contenant respectivement le plus petit et le plus grand indice disponible pour la dimension indiquée d'un tableau.

```
UBound(arrayname[, dimension])
LBound(arrayname[, dimension])
```

L'argument dimension est à préciser pour un tableau à plus de une dimension..

Les instructions de rupture de séquence

if...then...[else...]end if

Il existe deux syntaxes distinctes. La moins courante :

```
If condition Then Instruction_11[:...Instruction_1n] [Else
Instruction_21[:...Instruction_2n]]
```

La plus courante :

```
If condition1 Then
        Bloc_Instructions_1
[ElseIf condition2 Then
        Bloc_Instructions_2]
[Else
        Bloc_Instructions_n]
End If
```

Select Case

```
Select Case test_variable
[Case expression_liste1
        Bloc_Instructions_1]
[Case expression_liste2
        Bloc_Instructions_2]
[Case Else
        Bloc_Instructions_n]
End Select
```

Vous pouvez utiliser le mot To pour spécifier un intervalle de valeurs.

For...next

```
For compteur = mini To maxi [ Step pas ]
        bloc_instructions
        [Exit For]
        bloc_instructions
```

Next [compteur]

While...Wend

Tant que la condition est vraie, les instructions internes à la structure continuent de s'exécuter. Préférez l'emploi du Do...Loop.

While condition
bloc
Wend

Do Loop

Do [{While | Until} condition]
bloc_instructions
[Exit Do]
bloc_instructions
Loop
Do
bloc
[Exit Do]
bloc
Loop [{While | Until} condition]

Exit

Cette instruction permet de sortir d'un bloc de contrôle.

Exit Do *Sortie d'un bloc Do...Loop*
Exit For *Sortie d'un bloc For...Next.*
Exit Function *Sortie d'une fonction*
Exit Sub *Sortie d'une procédure*

Fonctions et procédures

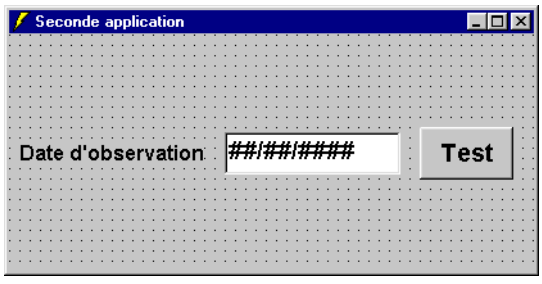
A l'instar du Pascal, Visual Basic différencie les fonctions et les procédures. Les premières retournent une valeur au programme appelant. Les secondes en changent son comportement sans retourner de valeur. Ce sont des "sous-traitants".

Les procédures

[Static] [Private] Sub NomProcédure ([[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type] [= defaultvalue],...])
BlocInstructions
[Exit Sub]
BlocInstructions
End Sub

L'emploi du mot Static signifie que les variables internes à la procédure sont conservées entre les appels successifs.

Le mot Optional permet de rendre un argument optionnel. Tous doivent l'être derrière. L'option Defaultvalue n'est valide que pour les arguments du type Optional. ParamArray permet de transmettre un tableau. L'uitlisation de l'argument est incompatible avec Optional. Il doit être en dernier dans la liste. ByVal et ByRef signifient un argument de type "valeur" pour le premier, de type "variable" pour le second.



Lors de l'appel à une procédure, chaque argument doit être séparé par un point-virgule et sans parenthèse contrairement au fonctions où le séparateur est une virgule. Pour les fonctions, les arguments doivent être encadrés par des

parenthèses.

Les fonctions

```
[Static] [Private] Function Nom_de_la_fonction ([[ByVal] variable_1[( )] [As type] [,...,
[ByVal] variable_n[( )] [As type]]]) [As type]
        [Instructions]
        [Nom_de_la_fonction = expression]
        [Exit Function]
        [Instructions]
        [Nom_de_la_fonction = expression]
End Function
```

Avant de quitter la fonction, il faut donner une valeur à celle-ci.

Des fonctions récursives (qui 'appellent elles-mêmes) peuvent provoquer lorsqu'elles sont mal faites des débordements de piles (stack overflow).

La gestion d'erreurs

Lors de la conversion de type, du remplissage d'un tableau, une erreur de votre part peut causer l'arrêt brutal du programme. Pour illustrer ce point, nous allons nous servir d'un contrôle de saisie sur un contrôle de Type MaskEdit. Ce contrôle ne fait pas partie des composants standards. Vous devez par conséquence l'ajouter dans votre boîte à outils. La gestion d'erreur se fait dans l'exemple ci-dessous sur l'événement LostFocus (sortie de zone).

```
'
'Procédure attribuant la date à la zone MaskEdBox_DATEOBS
'
Private Sub Form_Load()
        MaskEdBox_DATEOBS.Text = Format(Date, "dd/mm/yyyy")
End Sub

Private Sub MaskEdBox_DATEOBS_LostFocus()
        Dim Var_INFOS
'
' La gestion d'erreur est gérée par étiquette !!!
'
On Error GoTo ERREUR
        '
        ' Solution ne nécessitant pas de gestion d'erreur
        '
        'If Not IsDate(MaskEdBox_DATEOBS.Text) Then
        '   GESTION_ERREUR ("Date incorrecte")
        'End If
        '
        ' A éviter : la conversion suppose qu'il s'agit d'un type compatible
        ' Si la zone n'est pas compatible avec le format date, une erreur survient !
        '
        If Not IsDate(CDate(MaskEdBox_DATEOBS.Text)) Then
```

```
            GESTION_ERREUR ("Date incorrecte")
        End If
        '
        ' Sortie de bloc
        '
        Exit Sub
ERREUR:
        Var_INFOS = Err.Description & Chr(13) & _
                Err.Number & Chr(13) & _
                Err.Source
        GESTION_ERREUR (Var_INFOS)
End Sub
'
' Procédure-utilisateur visant à gérer l'erreur
'
Sub GESTION_ERREUR(MESSAGE)
        MsgBox MESSAGE, vbOKOnly, "Erreur"
        Form_Load
        '
        ' Le curseur est positionné dans la zone de saisie !
        '
        MaskEdBox_DATEOBS.SetFocus
End Sub
```

La table ANSI

0		32	[space]	64	@	96	‘	128		160	[space]	192	À	224	à
1		33	!	65	A	97	a	129		161	¡	193	Á	225	á
2		34	«	66	B	98	b	130		162	¢	194	Â	226	â
3		35	#	67	C	99	c	131		163	£	195	Ã	227	ã
4		36	$	68	D	100	d	132		164	¤	196	Ä	228	ä
5		37	%	69	E	101	e	133		165	¥	197	Å	229	å
6		38	&	70	F	102	f	134		166	¦	198	Æ	230	æ
7		39	‘	71	G	103	g	135		167	§	199	Ç	231	ç
8	BS	40	(	72	H	104	h	136		168	¨	200	È	232	è
9	TAB	41	)	73	I	105	i	137		169	©	201	É	233	é
10	LF	42	*	74	J	106	j	138		170	ª	202	Ê	234	ê
11		43	+	75	K	107	k	139		171	«	203	Ë	235	ë
12		44	,	76	L	108	l	140		172	¬	204	Ì	236	ì
13	CR	45	-	77	M	109	m	141		173	-	205	Í	237	í
14		46	.	78	N	110	n	142		174	®	206	Î	238	î
15		47	/	79	O	111	o	143		175	¯	207	Ï	239	ï
16		48	0	80	P	112	p	144		176	°	208	Ð	240	ð
17		49	1	81	Q	113	q	145		177	±	209	Ñ	241	ñ
18		50	2	82	R	114	r	146		178	²	210	Ò	242	ò
19		51	3	83	S	115	s	147		179	³	211	Ó	243	ó
20		52	4	84	T	116	t	148		180	´	212	Ô	244	ô
21		53	5	85	U	117	u	149		181	µ	213	Õ	245	õ
22		54	6	86	V	118	v	150		182	¶	214	Ö	246	ö
23		55	7	87	W	119	w	151		183	·	215	×	247	÷
24		56	8	88	X	120	x	152		184	¸	216	Ø	248	ø
25		57	9	89	Y	121	y	153		185	¹	217	Ù	249	ù
26		58	:	90	Z	122	z	154		186	º	218	Ú	250	ú
27		59	;	91	[	123	{	155		187	»	219	Û	251	û
28		60	<	92	\	124	|	156		188	¼	220	Ü	252	ü
29		61	=	93	]	125	}	157		189	½	221	Ý	253	ý

1. Création d'un premier projet EXE standard 1.1

VBP
FRM
BAS

Les éléments du projet

Le projet

C'est l'ensemble des éléments attachés au développement d'une application visuelle sous Visual Basic. Il s'agit d'un container. L'extension d'un projet VB est par défaut VBP. Il s'agit d'un fichier texte.

La feuille (form)

Elle représente l'interface-utilisateur. Une feuille correspond à une fenêtre. Elle inclut les contrôles et le code "événementiel" attaché aux contrôles. L'extension d'une feuille est FRM. Il s'agit d'un fichier texte.

Une fenêtre comprenant un menu est un exemple de forme.

Les contrôles

Ce sont les composants (OCX Object LinKing and Embedding Controls) d'une forme : boîtes de dialogues, boutons, étiquettes, etc. Ils intègrent des propriétés modifiables. Leur sont associées des procédures événementielles, c'est-à-dire des actions attachées à l'utilisateur : clic souris, touche clavier.

Les modules

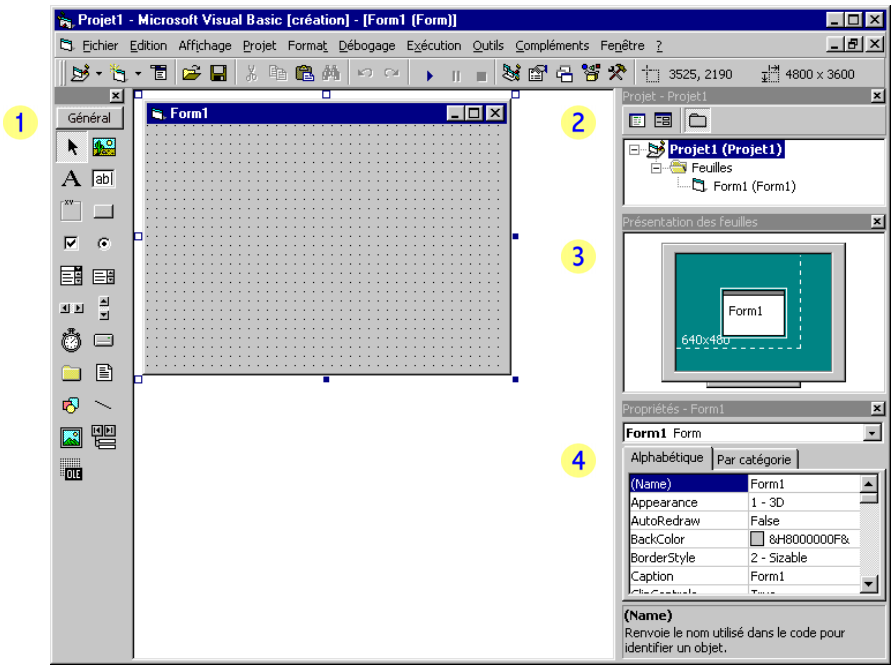
Ce sont des bibliothèques de fonctions et de procédures utilisables au niveau du projet indépendamment des feuilles qu'il comprend. Vous pouvez démarrer votre projet à partir de la procédure Main. L'extension d'un module est BAS. Ce sont des fichiers texte.

CTRL+R
F4
MAJ+F7
F7

Les régions de l'écran

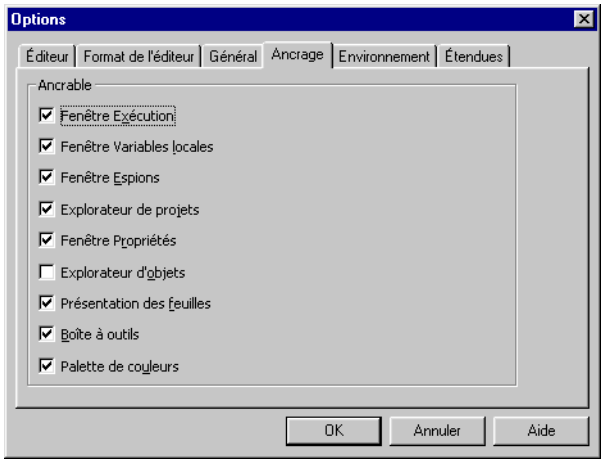
1. la boîte à d'outils : Affichage | Boîte à outils
2. l'explorateur de projets : Affichage | Explorateur de projets ou CTRL+R

   Pour basculer dans les modes d'affichage de la feuille, utilisez ou MAJ+F7 (feuille) et ou F7 (code).



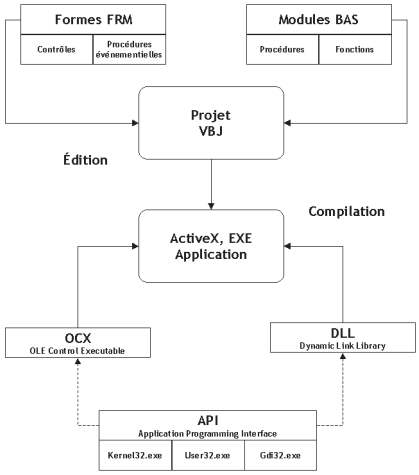
Outils Options Ancrage

3 **les propriétés du composant sélectionné :** Affichage | Fenêtre Propriétés

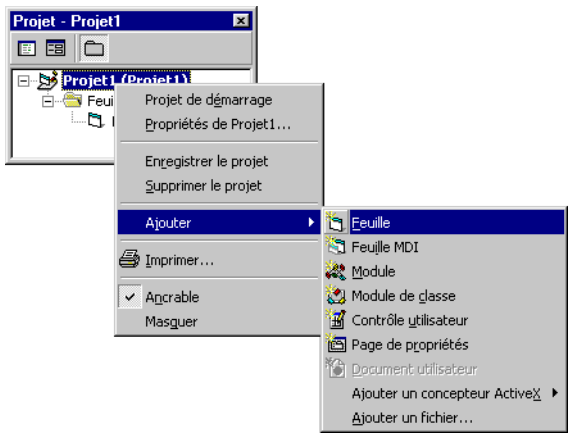


ou F4

4 **la présentation des feuilles :** Affichage | Fenêtre présentation des feuilles



Vous pouvez magnétiser les fenêtres par Outils | Options | Ancrage.



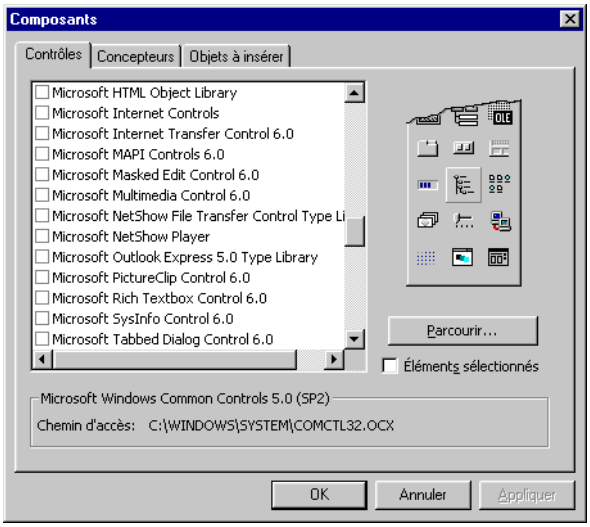
Le fonctionnem ent d'une application EXE Standard

Ajout d'éléments au projet

Ajout d'une feuille ou d'un module

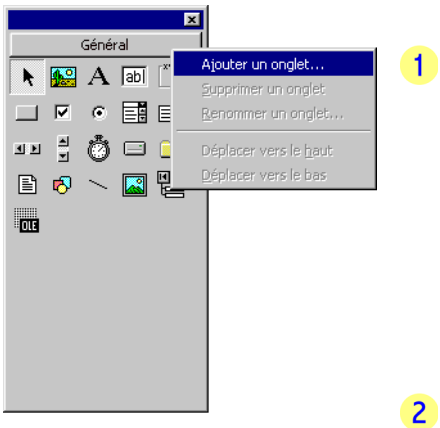
Denis Szalkowski

Ce sont des bibliothèques de fonctions et de procédures utilisables au niveau du projet indépendamment des feuilles qu'il comprend. Vous pouvez démarrer votre projet à partir de la procédure Main. L'extension d'un module est BAS.

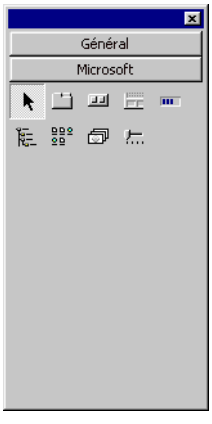


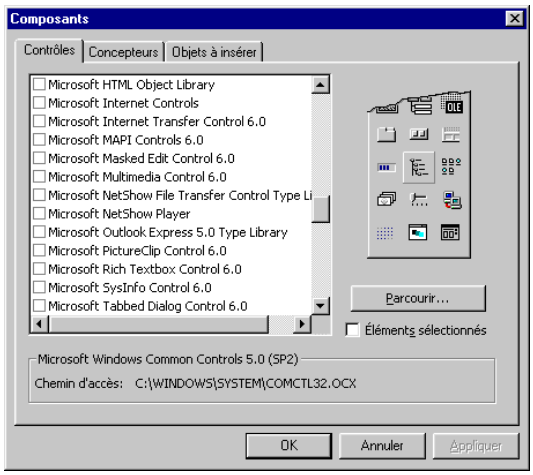
Ajout de composants ActiveX

Un ActiveX est un exécutable mettant ses objets à disposition d'autres applications conformément au modèle d'objet composant distribué ou DCOM (Distributed *Component Object Model*). Le principe de l'utilisation est celui du client-serveur.Il existe différents catégories de composants ActiveX. Un EXE ActiveX s'exécute dans son propre espace mémoire (out-of-process) . Une DLL Active X s'exécute dans l'espace du client. On parle alors *d'exécution in-process*.



1. Créez au préalable par un clic droit un onglet au sein de votre boîte à outils.Choisissez Ajouter un onglet.

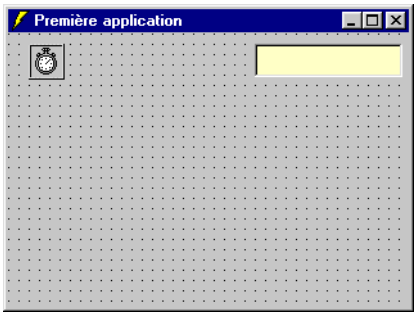




Ajouter un onglet dans la boîte à outils

Denis Szalkowski

2. Donnez un nom de votre choix (éditeur, famille de contrôle, etc).
3. Faites un clic droit dans la boîte elle même et choisissez Composants. Lorsque vous cochez un composant figurant dans la liste, ses éléments apparaissent alors dans l'onglet sélectionné après avoir validé par Appliquer. Pour visualiser les éléments ajoutés, cliquez sur Eléments sélectionnés.



Utilisation des composants

Cette partie s'appuiera sur un exemple concret : affichage de l'horloge en temps réel. Dans cette exemple, nous aurons recours au Timer, au composant Label.

Ajout d'un composant dans une feuille

Sélectionnez dans la boîte à outils le composant de votre choix. Dessinez un espace sur la feuille correspondant à la place que vous souhaitez lui faire prendre. Pour le composant Timer, la place occupée est constante.

La première action est, à partir de la fenêtre des propriétés, de nommer le composant en conservant en préfixe la classe à laquelle il appartient. Par exemple, pour le timer, nommez-le Timer_HEURE.

Associer une procédure événementielle

Un événement peut-être défini comme une déclencheur : un clic, une touche, le temps qui passe. La procédure événementielle, quant à elle, est une action (souvant formaliser par un verbe) associée à l'évenement déclencheur.

Pour initialiser le contenu de la zone Label, par un double-clic au centre du formulaire, vous accédez à la fenêtre relative au code.

```
Private Sub Form_Load()
        '
        ' Attribution au contrôle Label_HEURE de l'heure courante formatée
        '
        Label_HEURE.Caption = Format(Now(), "hh:mm:ss")
End Sub
```

Pour que l'heure s'actualise en permanence, il convient dans un premier temps d'initialiser la propriété Interval à 1 s (1000 ms). Ensuite, par double clic sur le timer, entrez le code suivant :

initialiser le contenu de la zone Label, par un double-clic au centre du formulaire, vous accédez à la fenêtre relative au code.

```
Private Sub Timer_HEURE_Timer()
        '
        ' Attribution au contrôle Label_HEURE de l'heure courante formatée
        '
        Label_HEURE.Caption = Format(Now(), "hh:mm:ss")
        '
        ' Appel à l'a procédure Form_Load : le Call est facultatif
        '
        'Call Form_Load
End Sub
```

Créer l'application

Avant toute chose, pensez à sauvegarder l'ensemble du projet par .

Tester votre "code"

A partir de la barre d'outils Standard, cliquez sur le bouton  . Pour arrêter l'exécution du programme, ou bien fermez la fenêtre principale de l'application par  ou bien arrêter l'application par le bouton  en arrière-plan.

Tester et compiler

L'exécutable n'est généré sur disque que s'il est compilé, c'est-à-dire traduit dans un format "binaire". Vous pouvez ou faire CTRL+F5 ou sélectionner Exécution | Exécution avec compilation complète.

Compiler seulement

Si vous ne souhaitez que compiler, c'est-à-dire créer l'exécutable, choisissez par le menu Fichier | Créer... EXE.