CREER UNE BASE DE DONNEES ACCESS AVEC DAO (étape par étape)

NIVEAU : PREMIERE RENCONTRE AVEC VB INITIES/EXPERIMENTES

Pré requis pour comprendre ce tutorial :

- Connaître les principales commandes de VB
- Connaître la structure générale d'une base de données Access

Auteur : Dark sidious
Date de création : 12/06/2004
Version : 1.0.1

SOMMAIRE

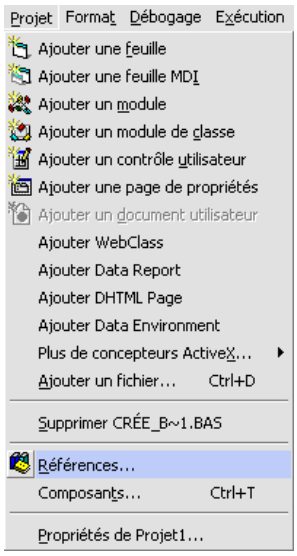
INTRODUCTION

Ce tutorial étape par étape est destiné à vous apprendre à créer une base de données Access avec du code uniquement en utilisant la technologie DAO de Microsoft en vous montrant les différentes étape à faire. Grâce à ce tutorial, vous apprendrez à créer un fichier .mdb au format Access 2000 contenant des tables, des champs, avec des clés primaires, etc. Ceci peut être très utile pour créer un fichier de base de données sans posséder Microsoft Access.
Pour avoir des informations supplémentaires sur les bases de données ou leur création, si vous voyez des erreurs, si vous avez des problèmes ou que vous ne comprenez pas trop bien ce que j'essaie de vous apprendre dans ce tutorial, ou si vous avez des idées pour un nouveau tutorial, vous pouvez m'envoyer un message privé sur le site www.ProgOtoP.com à DARK SIDIOUS.

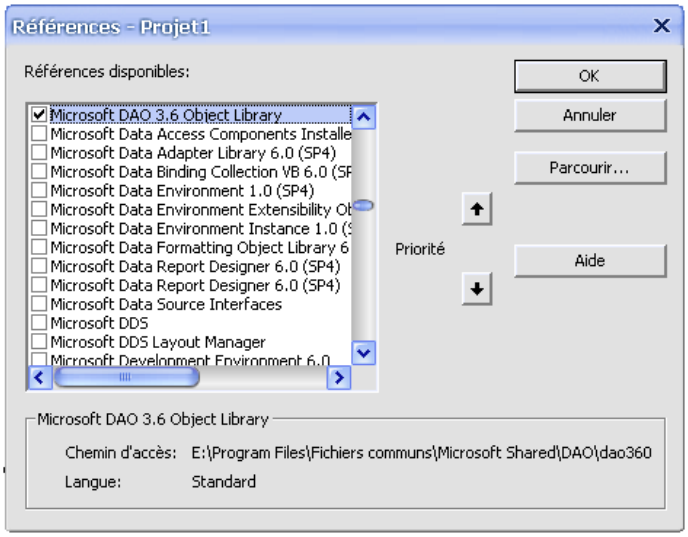
Bonne lecture, et bonne compréhension.

Etape 1 : Les références à DAO

Avant de se lancer dans la création de la base de données, il vous faut définir quelques références à DAO (DataBase Object Library) pour que votre projet puisse créer la base avec DAO. L'utilisation de la technologie DAO vous permet d'accéder à des bases de données très facilement, mais il faut pour cela renseigner VB. Pour ce faire, créer un nouveau projet, de type EXE Standard par exemple. Ensuite, cliquez sur le menu Projet, puis sur Références :



Ensuite, une boite de dialogue s'ouvre vous montrant toutes les références qu'il est possible de faire appel depuis VB, rechercher la référence « Microsoft DAO 3.6 Object Library » et cochez-là :



Remarque : il est possible que vous ne possèdiez pas la même version de DAO, mais ce tutorial devrait être compatible pour toutes les versions de DAO. Si la version 3.6 n'est pas référencée par VB, cochez une autre version de DAO.

Validez alors la nouvelle référence de votre projet en cliquant sur le bouton « Ok » de la boite de dialogue.

Cette étape permet donc à VB de reconnaître la librairie ADO et vous pourrez alors l'utiliser pour accéder à la base de données, déclarer des Database et des Recordset dans votre code, etc.

Etape 2 : Création du fichier de base de données

Maintenant que VB est informé qu'il peut utiliser DAO directement, nous allons commencer à créer la base de données.

Dans un premier temps, il faut créer un fichier de base de données.

Pour cela, nous allons créer un nom de fichier qui sera dans le répertoire du programme, on utilisera donc pour cela l'objet App. Nous stockerons ce nom de fichier dans la variable STR_Nom_Base. Le nom de la base de donnée s'appellera « Base.mdb » :

```
Dim STR_Nom_Base As String

Const STR_Nom_Fichier = "Base.mdb"

STR_Nom_Base = App.Path

STR_Nom_Base = Iif(Right(STR_Nom_Base, 1) = "\", STR_Nom_Base &
    STR_Nom_Fichier, STR_Nom_Base & «\» & STR_Nom_Fichier)
```

Nous pouvons créer le fichier de la base en appelant la fonction CreateDataBase qui est incluse dans l'objet DBEngine. Celle-ci prend plusieurs arguments mais seuls les deux premiers nous intéresse :

- Le nom du fichier de base de données qu'il stockera la nouvelle base de données
- L'ordre du tri de la base de donnée : spécifiez la constante dbLangGeneral si vous voulez que l'ordre du tri utilise les langues générales (Anglais, allemand, français, portugais, italien et espagnol moderne). Vous pouvez également spécifier dans le deuxième argument le mot de passe de la base de données en concaténant la chaine « ;pwd= » suivie du mot de passe.

Pour notre exemple, nous allons créer une base de données avec le nom que nous avons définit et avec un mot de passe qui sera : TestMDP. Nous stockerons cette base de données dans une variable de type DataBase qui est disponible grâce à la référence à DAO.

Les mots de passe d'une base de données Access sont sensibles à la casse, ce qui permet d'avoir un plus grand panel de valeurs, et ce qui en fait des mots de passe assez compliqué à perçer.

Voici le code nécessaire pour créer la base :

```
Dim DAT_Nouvelle_Base As DataBase

Set DAT_Nouvelle_Base = DBEngine.CreateDatabase(STR_Nom_Base, dbLangGeneral
    & «;pwd=TestMDP»)
```

Si vous possédez Microsoft Access et que vous ouvrez le fichier de base de données crée, vous pouvez remarquer qu'il vous demande un mot de passe dès l'ouverture, et si vous spécifiez le bon mot de passe, vous pouvez remarquer que la base de données est vide : il ne contient aucune table.

Etape 3 : Création de tables et de champs

Nous sommes maintenant prêts à créer des tables dans le fichier de base de données.

Comme vous l'avez vu, la création de la base de données a permit de récupérer un objet de type DataBase. C'est par l'intermédiaire de cet objet que nous pouvons dorénavant créer tout ce dont on a besoin : les tables, les champs, etc.

Pour créer une table de données, il nous faut tout d'abord déclarer une variable de type TableDef qui stockera la définition de chaque table à ajouter à la base.

Ensuite, nous devons créer la table en appelant la méthode CreateTableDef de notre objet DataBase. Cette méthode prend 4 paramètres mais seul le premier paramètre nous intéresse : Le nom de la table à créer (ce nom doit être unique) :

Après avoir crée une table, il nous faut déclarer une variable de type Field pour chaque champ à ajouter à la table.

Cet objet Field doit alors être crée en appelant la méthode CreateField de l'objet Table. Cette méthode prend en argument plusieurs paramètres, mais seuls les 3 premiers nous intéressent : le nom du champ, le type de champs (ici, nous spécifions des champs de texte uniquement) et la longueur du champ (ici nous limitons le nombre de caractère de tout nos champs à 200).

Enfin, il faudra définir les champs un par un, les rajouter à la table en appelant la méthode Append de l'objet Field contenu dans l'objet TableDef.

Dans cet exemple, nous allons créer 2 tables : l'une s'appelant « Noms », et l'autre s'appelant « Travail ».

Dans ces tables, nous allons créer 2 champs en plus de la clé primaire : dans la table « Noms » nous allons créer les champs « Nom » et un autre nommé « Prénom », et dans la table « Travail » nous allons créer les champs « Adresse » et « Poste ».

Une fois les champs des tables crées, il faut rajouter la table à la base de données en appelant la méthode Append de l'objet DataBase.

Voici le code complet pour créer les 2 tables contenant les 2 champs :

```
Dim TAB_Table As TableDef

Dim FLD_Champ As Field

   Set TAB_Table = DAT_Nouvelle_Base.CreateTableDef("Noms")

   Set FLD_Champ = TAB_Table.CreateField("Nom", 10, 200)

   FLD_Champ.Attributes = 2 'spécifie qu'il s'agit d'un champ de texte

   TAB_Table.Fields.Append FLD_Champ

   Set FLD_Champ = TAB_Table.CreateField("Prénom", 10, 200)

   FLD_Champ.Attributes = 2 'spécifie qu'il s'agit d'un champ de texte

   TAB_Table.Fields.Append FLD_Champ

   DAT_Nouvelle_Base.TableDefs.Append TAB_Table


   Set TAB_Table = DAT_Nouvelle_Base.CreateTableDef("Travail")

   Set FLD_Champ = TAB_Table.CreateField("Adresse", 10, 200)

   FLD_Champ.Attributes = 2

   TAB_Table.Fields.Append FLD_Champ

   Set FLD_Champ = TAB_Table.CreateField("Poste", 10, 200)

   FLD_Champ.Attributes = 2

   TAB_Table.Fields.Append FLD_Champ

   DAT_Nouvelle_Base.TableDefs.Append TAB_Table
```

Comme vous pouvez le voir, il faut respecter un certain ordre lors de la création d'une base de données complète : tout d'abord il faut créer le fichier de base de données, cela nous permet de récupérer un objet DataBase qui nous servira par la suite pour tout faire. Ensuite, il faut créer une table. Mais avant d'ajouter la table à la base de données, il faut d'abord lui ajouter tous les champs qu'elle doit contenir. Une fois qu'elle est définie, vous pouvez l'ajouter à la base et créer une nouvelle table.

Enfin, une fois que vous avez créé complètement la base, il vous faut fermer la connexion à la base de données en appelant la méthode close de l'objet DataBase :

```
DAT_Nouvelle_Base.Close
```

La création d'une table ne se limite pas qu'à la création de table, vous pouvez également créer des requêtes, des propriétés ainsi que des relations, mais si vous avez compris comment construire une table, vous aurez peu de mal à construire des requêtes, des propriétés ou des relations car c'est le même principe.

Etape 4 : Création d'index

Nous allons voir comment créer des champs indexés pour optimiser les recherches dans la base de données. Un champ indexé est un champ normal qui possède un numéro d'index invisible à l'utilisateur qui permet à Microsoft Access d'accélérer notablement les requêtes de recherches sur la base. Par contre, les index ralentissent la mise à jour des données de la base. Il est donc conseillé de n'utiliser les index que sur les champs où vous pratiquez beaucoup de requêtes de recherches, et de laisser les champs où vous devez faire des mises à jour normaux. De plus, les index font « grossir » la base de données car à chaque valeur d'un champ, un numéro d'index est attribué, ce qui prend un peu plus de place.

Pour créer un index, c'est très peu différent de la création d'un champ, il faut tout d'abord déclarer une variable de type Index. Ensuite, il faut créer un objet Index en appelant la méthode CreateIndex de l'objet TableDef. Cette méthode ne prend qu'un seul argument : le nom de l'index :

```
Dim IDX_Index As Index

Set IDX_Index = TAB_Table.CreateIndex(« IndexNom »)
```

Une fois que l'index est crée, il faut créer les champs qui seront contenus dans cet index. Attention, les champs qui doivent être contenus dans l'index doivent être identique (même type, même taille) que les champs qui sont contenus dans la table !

Pour créer un champ dans l'index, il suffit d'appeler la méthode CreateField de l'objet Index comme précédemment avec l'objet TableDef. Ensuite, il faut l'ajouter à la collection des champs de l'index en appelant la méthode Append de l'objet Index.

Dans cet exemple, nous allons indexer uniquement le champ Nom :

```
    IDX_Index.Fields.Append IDX_Index.CreateField("Nom", 10, 200)
```

Enfin, il nous faut maintenant ajouter l'index à la table en cours pour qu'elle le prenne en compte en appelant la méthode Append de l'objet Indexes contenu dans l'objet TableDef :

```
    TAB_Table.Indexes.Append IDX_Index
```

Comme vous pouvez le voir, une fois que vous avez compris comment fonctionne l'ajout d'un champ dans une table, tout le reste des opérations de créations pour une base de données sont quasi-identique.

Etape 5 : Personnalisation de la base

Dans cet exemple, vous avez vu comment créer une base de données « standard » avec des champs contenant des chaînes de caractères limités à 200 caractères, ce qui permet de construire quasiment n'importe quelle base qui stockera des données, mais cela n'est pas très optimisé : pourquoi stocker un nombre entier sur 200 caractères, en serait-ce pas plus optimisé de ne stocker que le nombre ? Comment faire pour stocker du texte dont on ne connaît pas la longueur et qui peut largement dépasser les 200 caractères ? C'est le but de cette étape : vous montrer comment personnaliser un champs ou un index.

L'avantage de DAO est qu'il est structuré en objet, et que l'on retrouve facilement les propriétés de ceux-ci pour voir comment on peut modifier les champs déjà crée.

Par exemple, si vous voulez modifier la taille du champ Nom, le limiter à 255 caractères et non à 200 (255 caractère est le nom maximal que peut atteindre un champ de type texte, ensuite il faut passer à un champ de type mémo), il vous suffit de modifier la propriété Size de l'objet Field qui est contenu dans l'objet TableDef :

```
TAB_Table.Fields(0).Size = 255
```

Remarque : Les index des champs sont crée dans l'ordre de création des champs, ainsi, si vous avez ajouter le champ « nom » avant le champ « prénom », l'index du champs « nom » sera inférieur à celui du champ « prénom »

De même, si vous voulez modifier le type du champ pour qu'il ne stocke que des nombres entiers, il faut modifier la propriété Type :

```
TAB_Table.Fields(0).Type = dbInteger
```

Vous avez donc accès à un large choix de personnalisation avec l'objet Fields pour faire correspondre les champs à vos attentes.

Vous pouvez, de même, personnaliser les index d'une table pour, par exemple, indexé un champ sans doublons, ce qui optimise encore plus les recherches dans la base, mais qui ne sera valide que si chaque valeur du champ est différente :

```
TAB_Table.Indexes(0).Unique = True
```

Chaque objet de la table possède des propriétés qui vous permettent de personnaliser l'objet, mais une présentation de toutes ses propriétés sortirait du cadre de ce tutorial, et je vous laisse les explorer vous-même avec l'explorateur d'objet de VB.

Le code complet

Voici le code complet pour la création de la base de données d'exemple en prenant en compte la mise en forme du code :

```
Private Sub Creer_Base()


Const STR_Nom_Fichier = "Base.mdb"


Dim STR_Nom_Base As String

Dim DAT_Nouvelle_Base As Database

Dim TAB_Table As TableDef

Dim FLD_Champ As Field

Dim IDX_Index As Index


   STR_Nom_Base = App.Path

   STR_Nom_Base = IIf(Right(STR_Nom_Base, 1) = "\", STR_Nom_Base &
     STR_Nom_Fichier, STR_Nom_Base & "\" & STR_Nom_Fichier)


   Set DAT_Nouvelle_Base = DBEngine.CreateDatabase(STR_Nom_Base,
     dbLangGeneral & ";pwd=TestMDP")


   Set TAB_Table = DAT_Nouvelle_Base.CreateTableDef("Noms")
```

```
    Set FLD_Champ = TAB_Table.CreateField("Nom", 10, 200)
    FLD_Champ.Attributes = 2
    Set IDX_Index = TAB_Table.CreateIndex("IndexNom")
    IDX_Index.Fields.Append IDX_Index.CreateField("Nom", 10, 200)
    TAB_Table.Indexes.Append IDX_Index
    TAB_Table.Fields.Append FLD_Champ

    Set FLD_Champ = TAB_Table.CreateField("Prénom", 10, 200)
    FLD_Champ.Attributes = 2
    TAB_Table.Fields.Append FLD_Champ

DAT_Nouvelle_Base.TableDefs.Append TAB_Table

Set TAB_Table = DAT_Nouvelle_Base.CreateTableDef("Travail")

    Set FLD_Champ = TAB_Table.CreateField("Adresse", 10, 200)
    FLD_Champ.Attributes = 2
    TAB_Table.Fields.Append FLD_Champ

    Set FLD_Champ = TAB_Table.CreateField("Poste", 10, 200)
    FLD_Champ.Attributes = 2
```

```
        TAB_Table.Fields.Append FLD_Champ


    DAT_Nouvelle_Base.TableDefs.Append TAB_Table


    DAT_Nouvelle_Base.Close


End Sub
```

CONCLUSION

La création d'une base de données avec DAO sous VB est relativement aisée, et vous permet de faire des programmes qui automatisent la création d'une base de données par exemple.
Au final, vous obtenez une base de données qui comporte les mêmes attributs qu'une base construite avec Microsoft Access. Bien entendu, il est plus agréable de construire ses bases avec Microsoft Access, mais vous pouvez vous en passer.
J'espère que la lecture de ce tutorial vous a satisfaite, et que vous continuerez à programmer sous Visual Basic. Et si vous avez des commentaires à me faire, n'hésitez pas à m'envoyer vos messages sur le site www.ProgOtoP.com