Gestion de favoris sous Access 2007

par Christophe WARIN

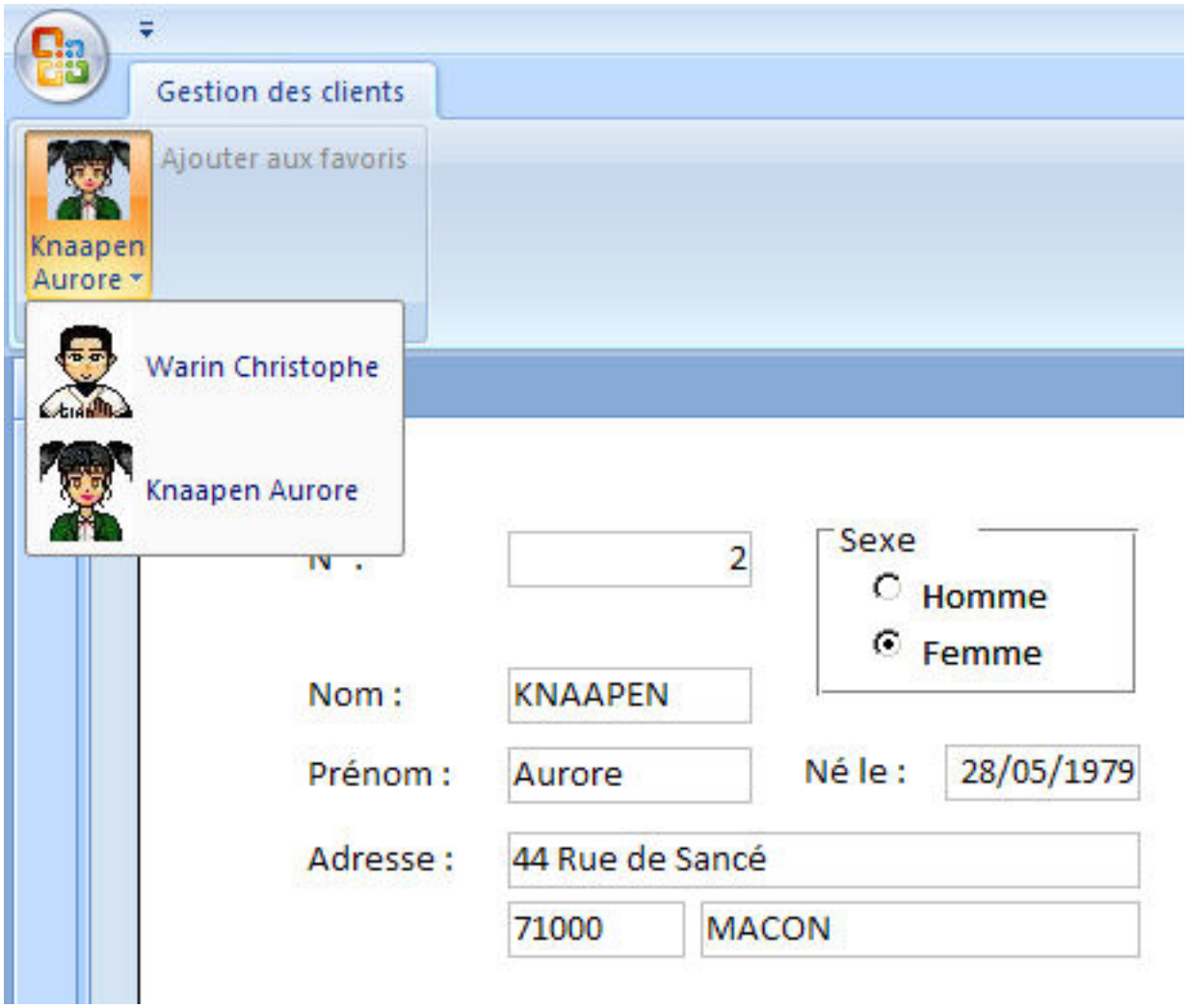
Date de publication : 24/11/2006

Dernière mise à jour :

Un exemple d'utilisation de la nouvelle interface de Microsoft Access. Niveau : Intermédiaire.

I - Introduction

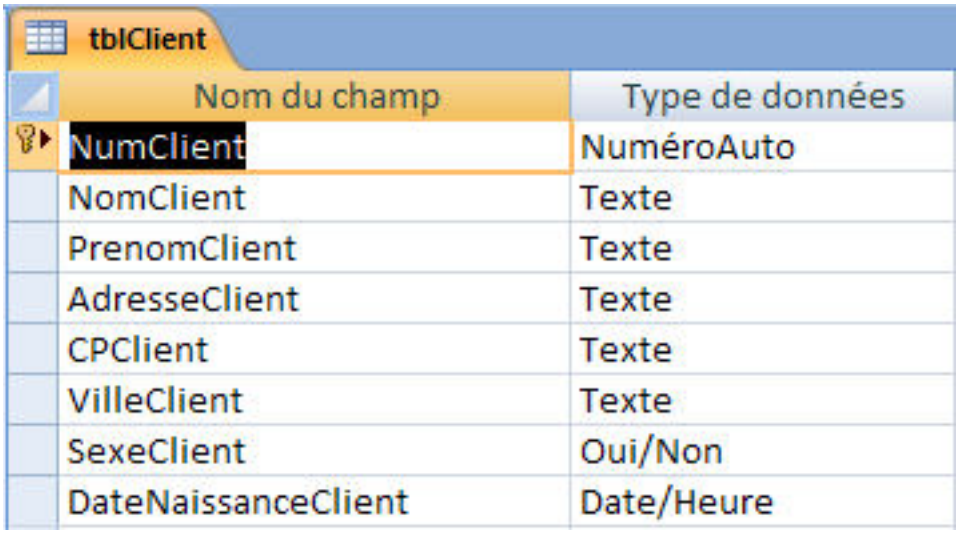
A travers ce tutoriel, je vous propose de découvrir pas-à-pas une utilisation possible des nouvelles fonctionnalités du ruban d'Access 2007. Il s'agit de concevoir une gestion de favoris dans un formulaire de gestion de client. L'utilisateur pourra à sa guise (comme sous un navigateur Web) garder une liste des enregistrements les plus fréquemment accédés. Pour enrichir d'avantage l'application, le ruban affichera une icône différente suivant le sexe du client.



Ce document requiert les compétences illustrées dans le tutoriel : Programmation et personnalisation du ruban sous Access 2007.

II - Structure des tables

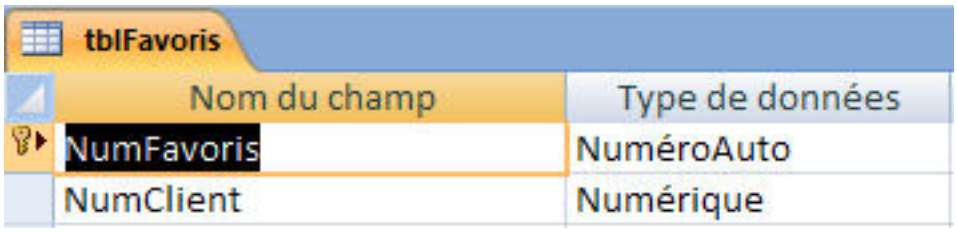
Comme indiqué en introduction, l'exemple considéré dans ce tutoriel concerne une gestion de clients. Une table **tblClient** est donc nécessaire :



tblClient

Nom du champ	Type de données
NumClient	NuméroAuto
NomClient	Texte
PrenomClient	Texte
AdresseClient	Texte
CPClient	Texte
VilleClient	Texte
SexeClient	Oui/Non
DateNaissanceClient	Date/Heure

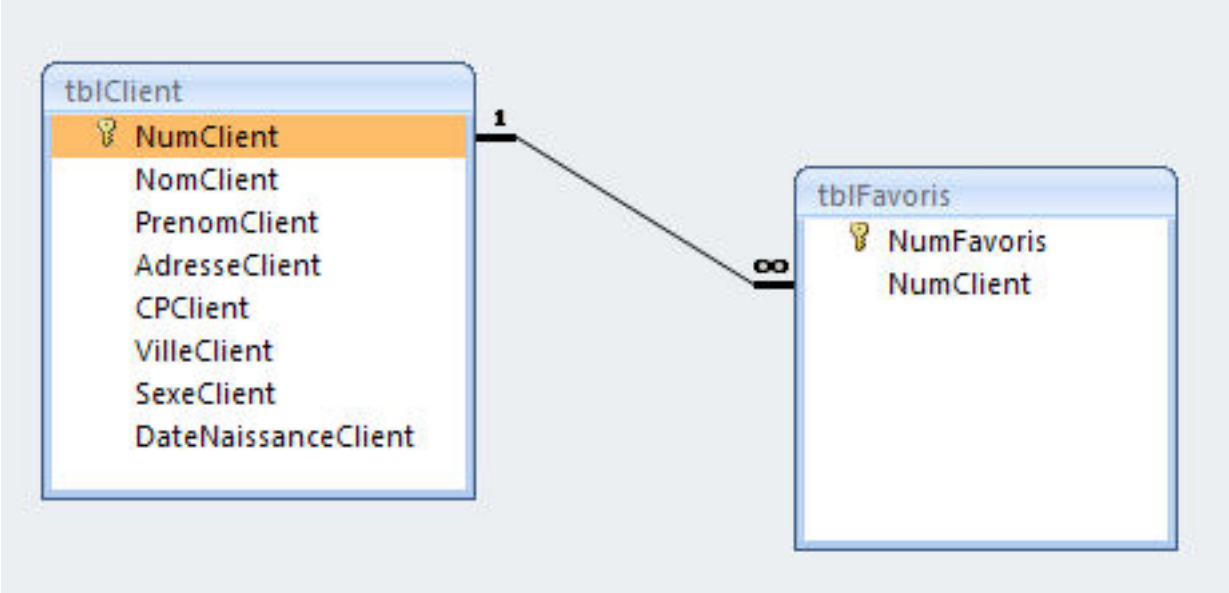
Afin d'enregistrer les clients préférés de l'utilisateur, une autre table (**tblFavoris**) doit être construite. Elle recensera uniquement les numéros des clients favoris. Un champ **NumFavoris** permettra d'identifier de manière unique et incrémentale les enregistrements de cette table.



tblFavoris

Nom du champ	Type de données
NumFavoris	NuméroAuto
NumClient	Numérique

Bien entendu, ces deux tables peuvent être mises en relations :



L'intégrité référentielle, et plus particulièrement la suppression en cascade permettra de supprimer automatiquement le favori en cas de suppression du client. En effet, il serait dangereux de garder des favoris indisponibles.

III - Requête et formulaire

Maintenant que les tables sont créées, il faut écrire une requête permettant de lister les informations des clients stockés dans la table **tblFavoris**.

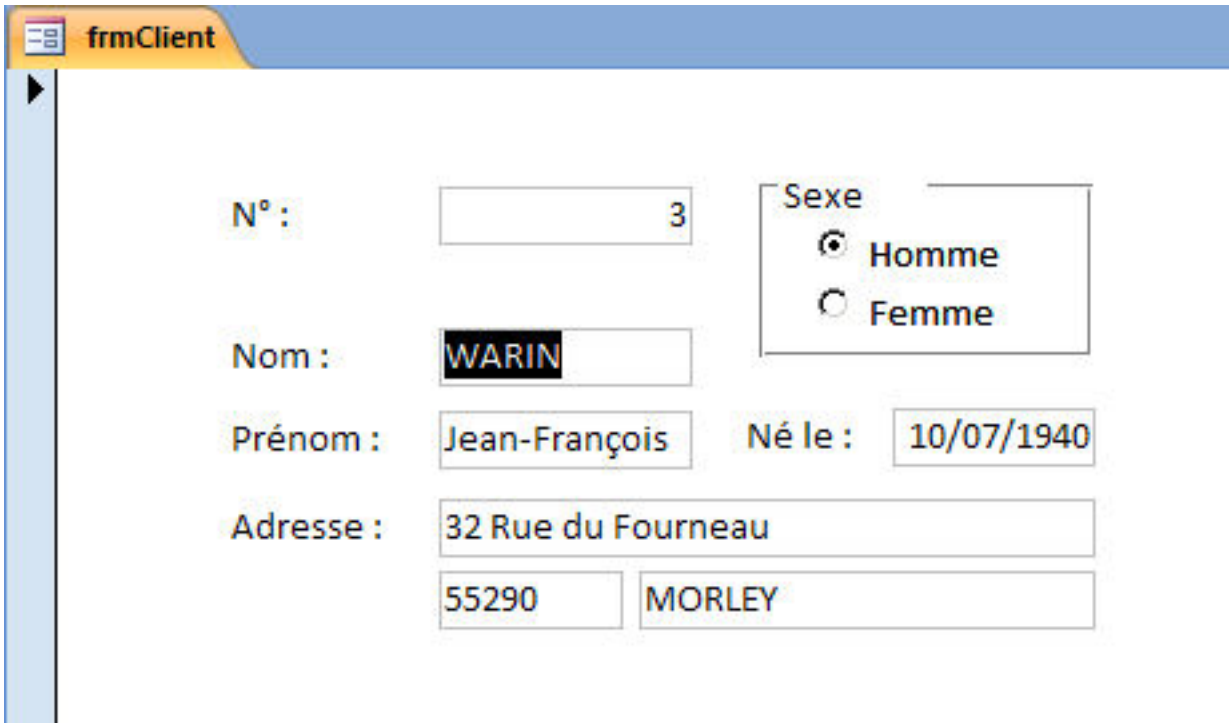
Cette étape nécessite de déterminer quelles données sont nécessaires. Ici, on souhaite afficher le nom et le prénom des clients. On doit aussi être en mesure de changer d'icône en fonction du sexe du client.

Ce qui donne :

```
SELECT tblFavoris.NumClient, tblClient.NomClient, tblClient.PrenomClient, tblClient.SexeClient
FROM tblClient
INNER JOIN tblFavoris ON tblClient.NumClient=tblFavoris.NumClient
ORDER BY NumFavoris DESC;
```

La clause **ORDER BY** permettra d'afficher les favoris les plus récents en premier.

Le formulaire servant de support pour cet exempe est très simple, nous ne nous attarderons pas sur son apparence. Il est nommé **frmClient**.



IV - Programmation

IV-A - Analyse

Programmer en VBA est un peu comme partir en vacances, il faut établir un itinéraire bien précis avant le départ pour éviter de se perdre. Cette étape est la plus complexe car elle nécessite de connaître les possibilités techniques offertes par le langage. Ici, j'ai choisi d'utiliser une collection pour alimenter le contrôle **gallery** du ruban. Pourquoi ? Comme énoncé dans le tutoriel Programmation du ruban, le seul moyen d'afficher les favoris que l'utilisateur va ajouter consiste à rafraîchir le contrôle. Ce mécanisme aura pour objectif d'exécuter les fonctions de rappels afin de reconstruire chaque élément (Item) du **gallery**. Cela signifie que si à 10H00 le contrôle possède 10 favoris et qu'à 11H00 l'utilisateur décide d'en rajouter un autre, il faudra réinsérer aussi ceux datant de 10H00. Il faudra donc à priori relire toute la table **tblFavoris**. Je trouve dommage d'être contraint à relire des informations sur disque qui ont déjà été stcokées en mémoire. Une collection permettra donc d'y remédier en conservant une liste de ces objets. Dès lors, en cas d'ajout, il suffira de parcourir uniquement celle-ci pour reconstruire rapidement le contrôle **Gallery**.

Qui dit collection, dit objets. En effet, il n'est pas possible de stocker des enregistrements dans une collection. Notre collection sera donc une liste d'objets contenant les données des enregistrements de la table **tblFavoris**.

IV-B - La classe classClient

D'autres auteurs ont déjà publié de nombreux articles sur la programmation orientée objet. Je vous invite à consulter celui de Xavier VLIEGHE : Introduction à la POO illustrée par VB6, pour de plus amples renseignements.

Voici le contenu du module de classe **classClient**

```
Option Compare Database

'Déclaration des noms d'images :

Const STRIMAGEHOMME_BLANC = "boy_white.jpg"
Const STRIMAGEHOMME_BLEU = "boy_blue.jpg"
Const STRIMAGEFEMME_BLANC = "girl_white.jpg"
Const STRIMAGEFEMME_BLEU = "girl_blue.jpg"


'Déclaration des attributs de la classe
Private p_intNumClient As Integer
Private p_strNomClient As String
Private p_strPrenomClient As String
Private p_bolSexeClient As Boolean


'Méthode qui permettra d'initialiser l'objet avec les valeurs adéquates
Public Sub Personnaliser(intNumClient As Integer, strNomClient As String, _
    strPrenomClient As String, bolSexeClient As Boolean)
    p_intNumClient = intNumClient
    p_strNomClient = strNomClient
    p_strPrenomClient = strPrenomClient
    p_bolSexeClient = bolSexeClient
End Sub

'Méthode qui retourne le nom formaté du client
Property Get NomComplet() As String
    NomComplet = StrConv(p_strNomClient & " " & p_strPrenomClient, vbProperCase)
End Property

'Méthode qui retourne le chemin de l'image blanche à utiliser
Property Get ImageBlanche() As String
    ImageBlanche = CurrentProject.Path & "\" & IIf(p_bolSexeClient, STRIMAGEHOMME_BLANC,
STRIMAGEFEMME_BLANC)
End Property
'Méthode qui retourne le chemin de l'image bleue à utiliser
```

```
Property Get ImageBleue() As String
    ImageBleue = CurrentProject.Path & "\" & IIf(p_bolSexeClient, STRIMAGEHOMME_BLEU,
STRIMAGEFEMME_BLEU)
End Property
'Méthode qui retourne le numéro du client
Property Get Num() As Integer
    Num = p_intNumClient
End Property
'Méthode qui retourne un identifiant alphanumérique du client
Property Get Cle() As String
    Cle = "cli" & p_intNumClient
End Property
```

Quelques explications :

Les constantes déclarées en haut de module correspondent aux noms des icônes à afficher. Bien entendu, ces images devront être présentes sur votre disque dur. Pourquoi des images blanches et d'autres bleues ? Les images affichées dans la liste du **Gallery** sont sur fond blanc ; en revanche, l'image représentant le contrôle en lui-même est sur fond bleu.

Chaque objet de la collection exposera l'ensemble de ses propriétés permettant ainsi d'afficher le nom complet du client, de récupérer l'identifiant du client, et de spécifier le chemin de l'image à charger.

IV-C - Accès à la table tblFavoris

Maintenant que la classe **classClient** est finalisée, il faut écrire les procédures qui permettront de créer autant d'objets classClient que d'enregistrements figurant dans la table **tblFavoris**. Afin de conserver ces objets en mémoire, ils seront ajoutés à une collection nommée **colFavoris**.

Ces procédures seront écrites dans un module **mduFavoris**.

```
Public Sub ConstructionFavoris()
'Déclarations des variables
Dim oDb As DAO.Database
Dim oRst As DAO.Recordset
Dim oClient As classClient
'Instancie la base de données
Set oDb = CurrentDb
'Ouvre un recordset sur la requête
Set oRst = oDb.QueryDefs(STRREQUETE).OpenRecordset
'Initialise la collection
Set colFavoris = New Collection
With oRst
    'Pour chaque enregistrement, crée un nouvel objet classClient
    'et l'ajoute à la collection
    While Not .EOF
        Set oClient = New classClient
        oClient.Personnaliser .Fields("NumClient").Value, _
                              .Fields("NomClient").Value, _
                              .Fields("PrenomClient").Value, _
                              .Fields("SexeClient").Value

        'Ajoute le client à la collection
        colFavoris.Add oClient, oClient.Cle
    .MoveNext
    Wend
End With
'Ferme les objets et libère la mémoire
oRst.Close
Set oRst = Nothing
Set oDb = Nothing
End Sub
```

La procédure **ConstructionFavoris** sera appelée lors du premier chargement du ruban afin de recenser les clients de la table **tblFavoris**. Les ajouts suivants seront gérés par la procédure ci-dessous.

```
Public Sub AjouterClient(intNumClient As Integer)
'Déclarations des variables
Dim oDb As DAO.Database
Dim oRst As DAO.Recordset
Dim oClient As classClient
'Instancie la base de données
Set oDb = CurrentDb
Set oRst = oDb.QueryDefs(STRREQUETE).OpenRecordset
'Ouvre un recordset sur la requête
With oRst
    'Ajoute le client dans la table
    .AddNew
    .Fields("NumClient").Value = intNumClient
    .Update
    .MoveLast
    'Crée l'objet classClient correspondant
    Set oClient = New classClient
    oClient.Personnaliser .Fields("NumClient").Value, _
                          .Fields("NomClient").Value, _
                          .Fields("PrenomClient").Value, _
                          .Fields("SexeClient").Value

End With

With colFavoris
    'Si la collection contient déjà des clients, alors ajouter au début
    If .count > 0 Then
        .Add oClient, oClient.Cle, .Item(1).Cle
    Else
        .Add oClient, oClient.Cle
    End If
End With
End Sub
```

Pour qu'elles fonctionnent, ces deux ***Sub*** ont besoin de la constante **STRREQUETE** et de la variable **colFavoris**. Elles seront déclarées en entête du module :

```
Const STRREQUETE = "R01_FAVORIS"
Public colFavoris As Collection
```

IV-D - Programmation du ruban

A ce stade, il est possible d'écrire le fichier XML (*ribbon.xml*) du ruban. Le groupe se composera de deux contrôles : le **Gallery** ainsi qu'un bouton qui permettra d'ajouter l'enregistrement courant dans la table **tblFavoris**.

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" onLoad="getMonRuban">
   <ribbon startFromScratch="true">
      <tabs>
        <tab id="tabEvenement" label="Gestion des clients" visible="true">
                        <group id="grpfavoris" label="favoris">

                        <gallery
                                id="galFavoris"
                                label="Mes Favoris"
                                columns="1"
                                getImage="getImageGalleryFavoris"
                                getItemCount="getNBFavoris"
                                getItemLabel="getLabelFavoris"
                                getItemImage="getImageFavoris"
                                getItemID="getIdFavoris"
                                size="large"

                        />

                        <button
                                id="btnAjout"
                                label="Ajouter aux favoris"
                                size="normal"

                        />
                        </group>
```

```
          </tab>
        </tabs>
    </ribbon>
</customUI>
```

Rien de bien compliqué, la complexité va se situer plutôt au niveau des fonctions de rappels destinées à alimenter le contrôle **Gallery**.

Pour rappel, le contrôle **Gallery** puisera ses données dans la collection **colFavoris**. La fonction de rappel **getNBFavoris** retournant le nombre d'éléments du **Gallery** sera donc architecturée comme suit :

```
Sub getNBFavoris(control As IRibbonControl, ByRef count)
    count = colFavoris.count
End Sub
```

Le texte de chaque élément sera géré par la fonction **getLabelFavoris** :

```
Sub getLabelFavoris(control As IRibbonControl, index As Integer, ByRef label)
    Dim oClient As classClient
    Set oClient = colFavoris(index + 1)
    label = oClient.NomComplet
End Sub
```

Quant à l'identifiant de l'élément, il sera défini par :

```
Sub getIdFavoris(control As IRibbonControl, index As Integer, ByRef ID)
    Dim oClient As classClient
    Set oClient = colFavoris(index + 1)
    ID = oClient.Cle
End Sub
```

Il ne reste plus qu'à modifier l'image des éléments et le contrôle Gallery sera capable d'afficher sa liste de boutons :

```
Sub getImageFavoris(control As IRibbonControl, index As Integer, ByRef image)
    Dim oClient As classClient
    Set oClient = colFavoris(index + 1)
    Set image = stdole.LoadPicture(oClient.ImageBlanche)
End Sub
```

```
Sub getImageGalleryFavoris(control As IRibbonControl, ByRef image)
    Set image = stdole.LoadPicture(CurrentProject.Path & "\no_blue.jpg")
End Sub
```

no_blue.jpg est une image de votre choix symbolisant un favori (il s'agit souvent d'une étoile)

IV-E - Interaction ruban / application

Le ruban est maintenant créé (même s'il peut être amélioré), il reste maintenant à le rendre opérationnel.

Dans un premier temps, il faut qu'il soit chargé au démarrage. Pour cela nous allons reprendre l'exemple du tutoriel qui nous sert de référence depuis le début à savoir :

Définir une macro **AutoExec** qui lancera une fonction VBA écrite par nos soins qui lira le contenu du fichier ribbon.xml et chargera le ruban en dernier lieu.

```
Public Function initRibbon()
Dim strXML As String
Dim oFso As New FileSystemObject
Dim oFtxt As TextStream
ConstructionFavoris
'Charge le fichier XML en mémoire
Set oFtxt = oFso.OpenTextFile(CurrentProject.Path & _
      "\ribbon.xml", ForReading)
'Récupère le contenu
strXML = oFtxt.ReadAll
'Charge le rubban personnalisé correspondant
Application.LoadCustomUI "rubanFavoris", strXML
End Function
```

```
Sub getMonRuban(ribbon As IRibbonUI)
Set oMonRuban = ribbon
End Sub
```

Ne pas oublier de déclarer Public oMonRuban As IRibbonUI en entête de module.

Comme vous pouvez le remarquer, le ruban sera accessible par Access sous le nom de **rubanFavoris**. C'est ce nom qui est à saisir dans la propriété **ruban personnalisé** du formulaire **frmClient** pour lier le formulaire à la nouvelle barre de menu.

Dans un second temps, il faut donner une utilité au ruban. Pour l'instant il ne fait office que d'élément de décoration. En effet, aucun évènement n'y est associé. Afin qu'il réponde aux actions de l'utilisateur, nous allons modifier le XML pour ajouter les propriétés adéquates aux contrôles.

Ce qui donne :

```
<gallery
                                id="galFavoris"
                                label="Mes Favoris"
                                columns="1"
                                getImage="getImageGalleryFavoris"
                                getItemCount="getNBFavoris"
                                getItemLabel="getLabelFavoris"
                                getItemImage="getImageFavoris"
                                getItemID="getIdFavoris"
                                size="large"
                                onAction="Selectionner"
                        />

                        <button
                                id="btnAjout"
                                label="Ajouter aux favoris"
                                size="normal"
                                onAction="Ajouter"
                        />
                        </group>
```

Les procédures événementielles VBA **Ajouter** et **Selectionner** figurent ci-dessous :

```
Public Sub Selectionner(ByVal control As IRibbonControl, _
    selectedId As String, selectedIndex As Integer)
On Error GoTo err
    Dim oClient As classClient
    Dim oFrm As Form
    Dim strFiltre As String
    'Accède au formulaire
    Set oClient = colFavoris(selectedId)
    'Cherche le client correspondant dans le formulaire
    Forms(STRFORMULAIRE).Recordset.FindFirst "NumClient=" & oClient.Num
err:
End Sub
```

```
Public Sub Ajouter(ByVal control As IRibbonControl)
    'Appel la procédure AjouterClient
    AjouterClient Forms(STRFORMULAIRE)![NumClient]
End Sub
```

Quelques précisions sur la procédure **Selectionner** :

Le paramètre selectedId retourne l'ID de l'élément sélectionné. Or, lors de l'enrichissement du contrôle **Gallery**, la fonction de rappel **getIDFavoris** définit l'**ID** comme étant égal à la *clé* de l'objet correspondant dans la collection.

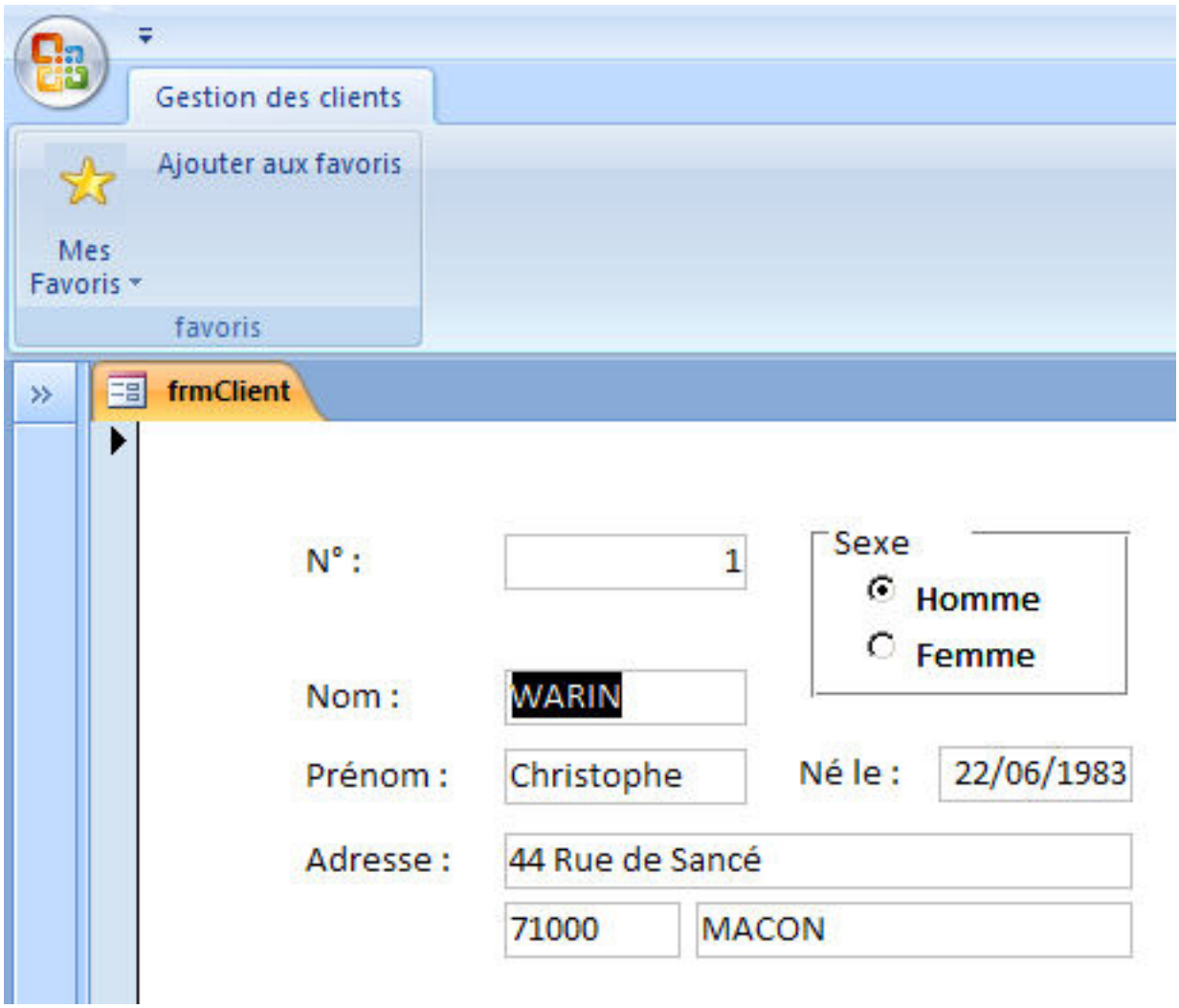
On a donc l'égalité suivante :

colFavoris(xxx).Cle=selectedID

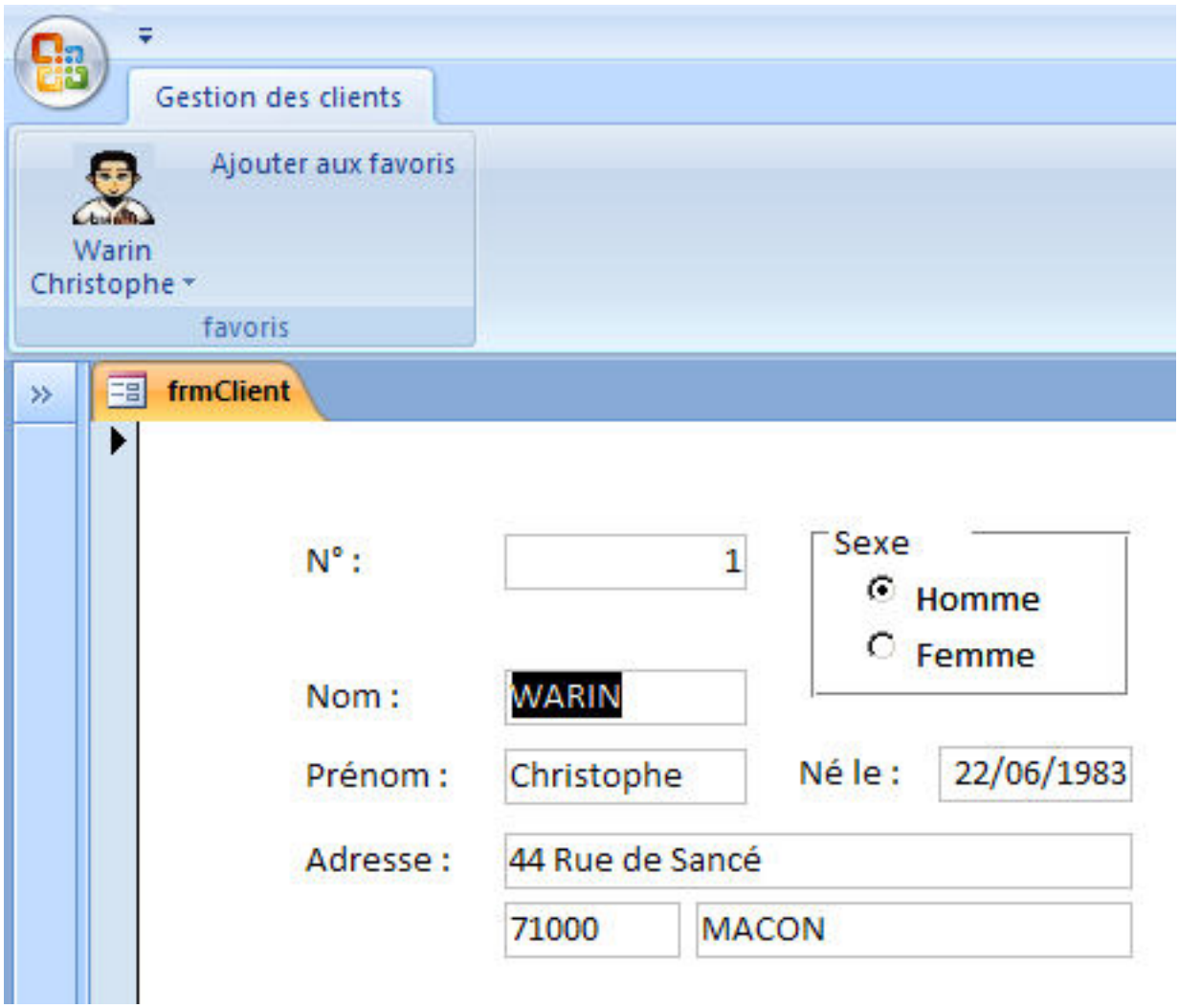
De ce fait, **colFavoris(selectedId)** retourne directement l'objet **classClient** souhaité.

IV-F - Synchronisation

Jusque là, le contrôle **Gallery** est figé. Quelque soit l'enregistrement sélectionné dans le formulaire le ruban affiche *Mes favoris* :



L'ergonomie du ruban (et donc de l'application) sera nettement améliorée si le contrôle **Gallery** devient interactif et affiche le favori courant dans son libellé et son image.



Pour cela, il faut modifier **getImageGalleryFavoris** et créer **getLabelGalleryFavoris** de telles sorte qu'elles vérifient si l'enregistrement courant du formulaire correspond à un favori et si c'est le cas, qu'elles modifient l'intitulé et l'image du contrôle **Gallery** pour afficher le bon client.

```
Sub getLabelGalleryFavoris(control As IRibbonControl, ByRef label)
On Error GoTo err
    Dim oClient As classClient
    Dim oFrm As Form
    'Accède au formulaire
    Set oFrm = Forms(STRFORMULAIRE)
    'Vérifie si le client existe dans les favoris
    Set oClient = colFavoris("cli" & oFrm![NumClient])
    'Si oui, affiche son nom complet
    label = oClient.NomComplet
fin:

Exit Sub

err:
    label = "Mes Favoris"
    Resume fin
End Sub
Sub getImageGalleryFavoris(control As IRibbonControl, ByRef image)
On Error GoTo err
    Dim oClient As classClient
    Dim oFrm As Form
```

```
        'Accède au formulaire
        Set oFrm = Forms(STRFORMULAIRE)
        'Vérifie si le client existe dans les favoris
        Set oClient = colFavoris("cli" & oFrm![NumClient])
        'Si oui, affiche son image
        Set image = stdole.LoadPicture(oClient.ImageBleue)
    fin:

    Exit Sub

    err:
        Set image = stdole.LoadPicture(CurrentProject.Path & "\no_blue.jpg")
        Resume fin
    End Sub
```

Le XML du contrôle **Gallery** devient :

```
                            <gallery
                                    id="galFavoris"
                                    getLabel="getLabelGalleryFavoris"
                                    columns="1"
                                    getImage="getImageGalleryFavoris"
                                    getItemCount="getNBFavoris"
                                    getItemLabel="getLabelFavoris"
                                    getItemImage="getImageFavoris"
                                    getItemID="getIdFavoris"
                                    size="large"
                                    onAction="Selectionner"
                            />
```

L'évènement **Form_Current** du formulaire sera chargé de rafraîchir le ruban afin qu'un déplacement manuel dans les enregistrements synchronise la liste des favoris. En d'autres termes, si l'utilisateur se déplace manuellement sur un enregistrement favoris, alors l'afficher dans le ruban, sinon, afficher *Mes Favoris*.

```
    Private Sub Form_Current()
        If Not oMonRuban Is Nothing Then
            oMonRuban.InvalidateControl "galFavoris"
        End If
    End Sub
```

V - Améliorations

Plusieurs améliorations peuvent être envisagées. L'une d'entre elles consiste à désactiver le bouton *Ajouter aux favoris* si l'enregistrement courant dans le formulaire est déjà dans les favoris. Il s'agit de modifier dynamiquement la propriété **Enabled** du bouton, c'est-à-dire de programmer la fonction de rappel **getEnabled**.

Le code XML du bouton **btnAjout** doit être modifié :

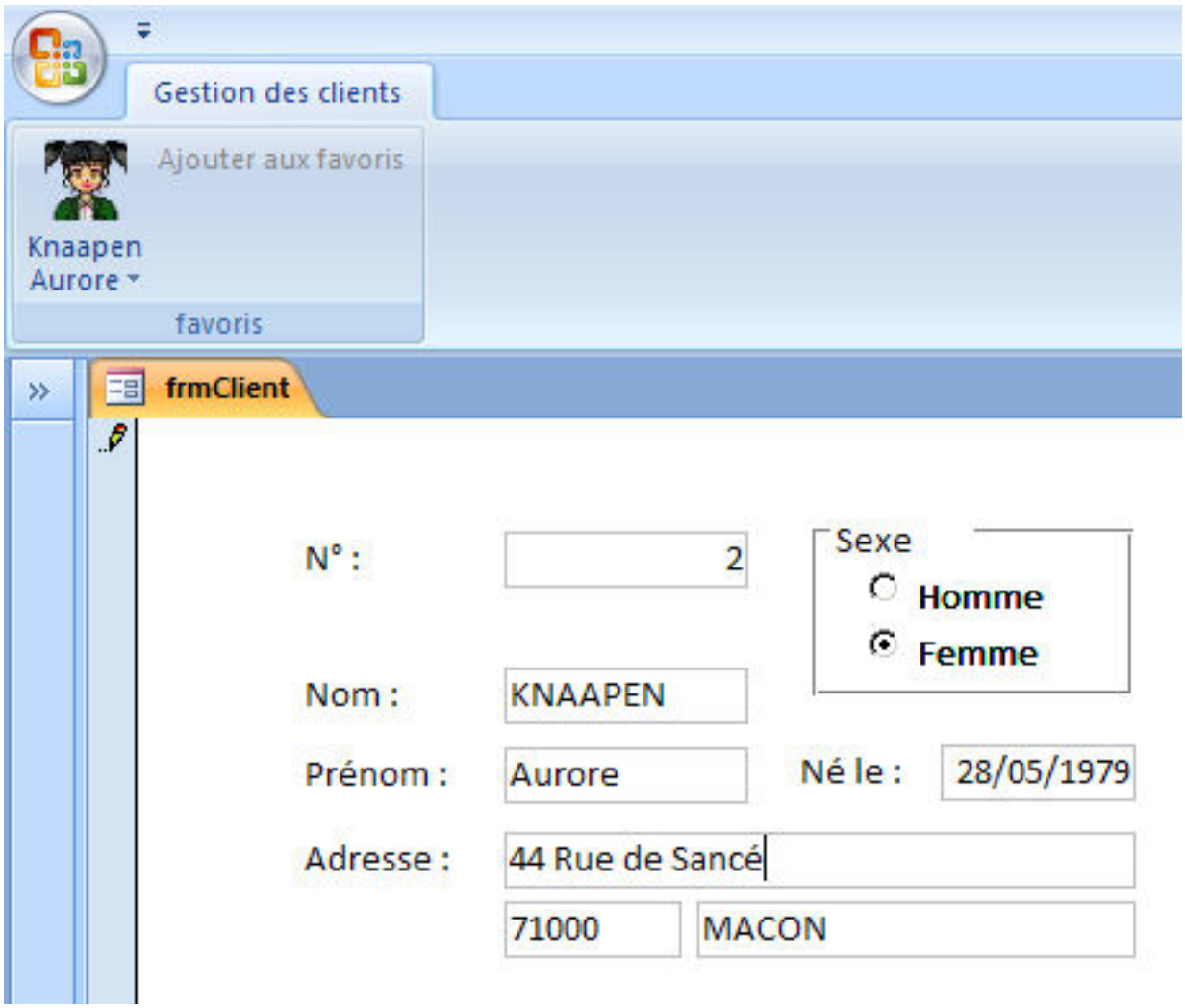
```
                        <button
                                id="btnAjout"
                                label="Ajouter aux favoris"
                                size="normal"
                                onAction="ajouter"
                                getEnabled="getAjoutEnabled"
                        />
```

Ci-dessous le code de la fonction de rappel :

```
Public Sub getAjoutEnabled(ByVal control As IRibbonControl, ByRef enabled)
On Error GoTo err

    Dim oClient As classClient
    'Vérifie si le client existe dans les favoris
    Set oClient = colFavoris("cli" & Forms(STRFORMULAIRE)![NumClient])

    'Si oui, désactive le bouton
    enabled = False

fin:

Exit Sub

err:
    'En cas d'erreur 5 , (client ne figure pas dans les favoris)
    'active le bouton, sinon désactive
   enabled = err.Number = 5
   Resume fin
End Sub
```

Enfin, il faut que l'évènement **form_current** du formulaire rafraîchisse le contrôle **btn_ajout** (jusque là il ne rafraichissait que le contrôle **Gallery**)

```
Private Sub Form_Current()
    If Not oMonRuban Is Nothing Then
        oMonRuban.InvalidateControl "galFavoris"
        oMonRuban.InvalidateControl "btnAjout"
    End If
End Sub
```



VI - Conclusion

Comme vous avez pu le constater à travers cet exemple, la programmation du ruban sous Microsoft Access 2007 offre de nouvelles possibilités et tend à améliorer considérablement l'ergonomie des applications développées ce qui ne saurait déplaire aux utilisateurs.

Bien qu'elle puisse sembler complexe car abordant différentes technologies (XML, DAO, Fichiers, etc.) elle n'en reste pas moins assez facile d'accès à condition d'être méthodique et de procéder par étape.

Cette nouveauté restera sans conteste mon coup de c#ur pour cette nouvelle version d'Office.