COM+

NB : ce fascicule fait partie d'un travail de diplôme sur le sujet « Technologie ActiveX & Visual Basic 6 ».
Les autres fascicules peuvent être demandés à fcomte@caramail.com.

Critiques, remarques, questions ? fcomte@caramail.com

1 - Introduction

COM+ (arrivé sur le marché avec Windows 2000) constitue l'étape suivante de l'évolution de COM et de MTS. Le regroupement des modèles de programmation inhérents aux services COM et MTS facilite le développement d'applications distribuées, en supprimant la complexité associée au développement, au débogage, au déploiement et à la maintenance d'une application qui repose sur COM pour certains services et sur MTS pour d'autres.

COM+ offre la possibilité de développer plus rapidement, avec plus de facilité et à moindre coût des applications distribuées, en réduisant la quantité de code nécessaire à l'optimisation des performances des services système sous-jacents.

COM+ est donc construit sur la norme COM et incorpore principalement les deux technologies :

- MTS (Microsoft Transaction Server), qui simplifie la vie du programmeur en lui permettant de créer et utiliser des transactions,
- MSMQ (Microsoft Message Queues), un ensemble d'objets assurant une exécution différée en mode asynchrone lorsque les composants travaillant en coopération sont déconnectés. Cette fonctionnalité vient s'ajouter au modèle de programmation client-serveur synchrone basé sur la session,

ainsi que d'autres outils nouveaux dont le principal est :

- IMDB (In-Memory Database system), qui permet de gérer des informations permanentes et des informations temporaires de façon cohérente. In-Memory DataBase est un système de base de données entièrement transactionnel géré en mémoire et conçu pour fournir un accès extrêmement rapide aux données.

Dans le but d'étendre le modèle COM et les services proposés actuellement par MTS 2.0, COM+ a introduit des améliorations aux services existants, ainsi que de nouveaux services pour la plate-forme d'application Windows.

Ces améliorations sont les suivantes :

Support d'environnements transactionnels hétérogènes
Les composants COM peuvent participer à des transactions gérées par un environnement transactionnel (TP) autre que celui de COM+ par le support du protocole TIP (Transaction Internet Protocol).

Sécurité étendue
La prise en charge de la sécurité est basée sur la notion de rôle ainsi que les permissions d'accès des processus. Dans le modèle de sécurité basé sur le rôle, l'accès aux différentes parties d'une application est accordé ou refusé en fonction du groupe logique auquel l'utilisateur appartient ou du rôle qui lui a été attribué (par exemple, administrateur, employé à temps complet, employé à temps partiel). COM+ implémente en plus de la sécurité basée sur le rôle, la sécurité au niveau des méthodes des interfaces.

Administration centralisée
L'Explorateur des services de composants, qui remplace MTS Explorer et DCOMCNFG, présente un modèle d'administration unifié, qui facilite le déploiement, la maintenance et

la surveillance d'applications à n niveaux, en éliminant le besoin d'utiliser de nombreux outils d'administration distincts.

Notification d' événements
A utiliser lorsqu'un mécanisme de notification d'événements est souhaité. Le système d'événements de COM+ est un mécanisme de publication/abonnement qui permet à des clients de " s'abonner " à des événements " publiés " par plusieurs serveurs. Cette fonctionnalité vient s'ajouter à la structure existante de notification des événements fournie avec les points de connection de COM.

Répartition de charge (MSCS, Microsoft Cluster Server)
Permet aux applications basées sur le composant de répartir leur charge de travail sur un cluster d'applications et en toute transparence vis-à-vis du client.

COM+ s'utilise principalement sous C++ (certaines fonctionnalités ne sont d'ailleurs disponibles que dans ce langage), néanmoins cela ne signifie pas qu'il est impossible d'utiliser COM+ sous Visual Basic. En fait – tout comme le développement de COM sous VB – il est même plus simple de travailler avec Visual Basic.

Malheureusement, l'auteur n'ayant pas réussi à réaliser dans le temps imparti (bien qu'il aie commencé) une application COM+, nous parlerons brièvement de MTS, MSMQ et IMDB et présenterons un modèle d'architecture trois-tiers avec COM+.

2 - MTS

Notions de transaction

Voyons d'abord ce qu'on entend par transaction. Basiquement, il s'agit d'une unité fondamentale de travail dans un système. Une transaction doit compléter son travail sans intervention de l'utilisateur (cela peut être un transfert financier, ou une réservation d'une place dans un avion). Au moment où elle démarre, la transaction doit posséder toutes les informations nécessaires pour que son exécution se déroule normalement.
Sous Windows, les transactions sont exécutées – en utilisant des composants COM+ - sur un serveur d'applications ou de transactions. Il est de la responsabilité de ceux-ci d'exécuter des transactions relatives aux requêtes des utilisateurs, de fournir toutes les sécurités adéquates, et de savoir réagir en cas de panne du système.

Chaque transaction doit remplir un certain nombre de critères connu sous le nom de test ACID : Atomicité, Cohérence, Isolation et Durabilité. Voyons-les de plus près :

Atomicité

Le travail réalisé par la transaction doit être fait entièrement (c'est-à-dire du début à la fin) ou pas du tout, ce qui garantit qu'une transaction ne peut quitter le système dans un état semi-complétée.
Le contenu d'une transaction en peut pas être subdivisé en plusieurs commandes, ainsi si la transaction est menée à son terme, tous les changements qu'elle a apporté sur les données sont validés. Dans le cas contraire, tous les changements sont annulés.

Cohérence

Une transaction doit toujours laisser le système dans un état cohérent après avoir été exécutée.
Prenons l'exemple d'un transfert de fonds d'un compte bancaire à un autre. Si un problème survient durant le transfert même, il est possible qu'un des comptes soit débité d'une somme d'argent, mais que l'autre compte n'ait pas reçu cette même somme. En incorporant ces deux mises à jour dans une transaction, on assure à la base de données de rester dans un état cohérent. Cette dernière, si le problème devait survenir, s'apercevrait que la transaction ne s'est jamais terminée et ne la validerait pas : les deux comptes ne seraient donc pas modifiés.

Isolation

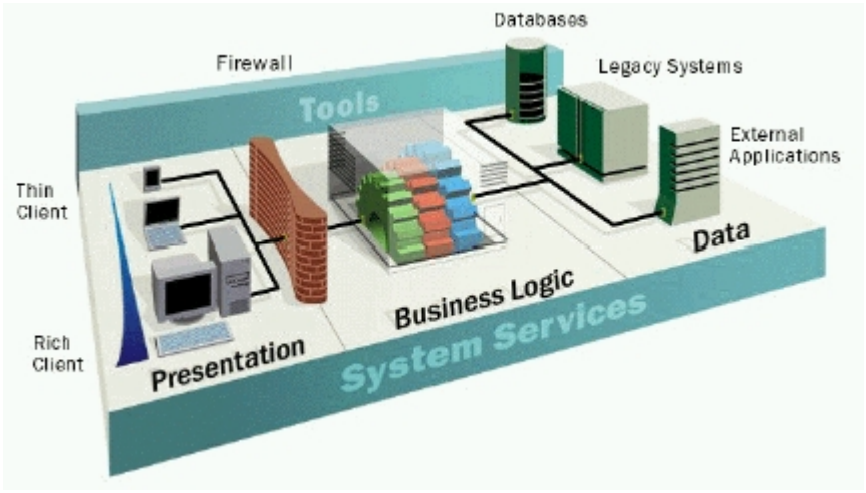
Les autres transactions ne peuvent pas voir les données partiellement modifiées d'une autre transaction en cours.
Ce critère permet une bonne récupérabilité de la base en cas de panne. Il est évident que plusieurs transactions seront activées au même moment. Cependant, l'isolation de celles-ci les unes par rapport aux autres simule le traitement d'une transaction après l'autre (dès que la première est finie, la suivante commence). Ce processus est essentiel pour travailler sur des systèmes de bases de données distribuées.

Durabilité

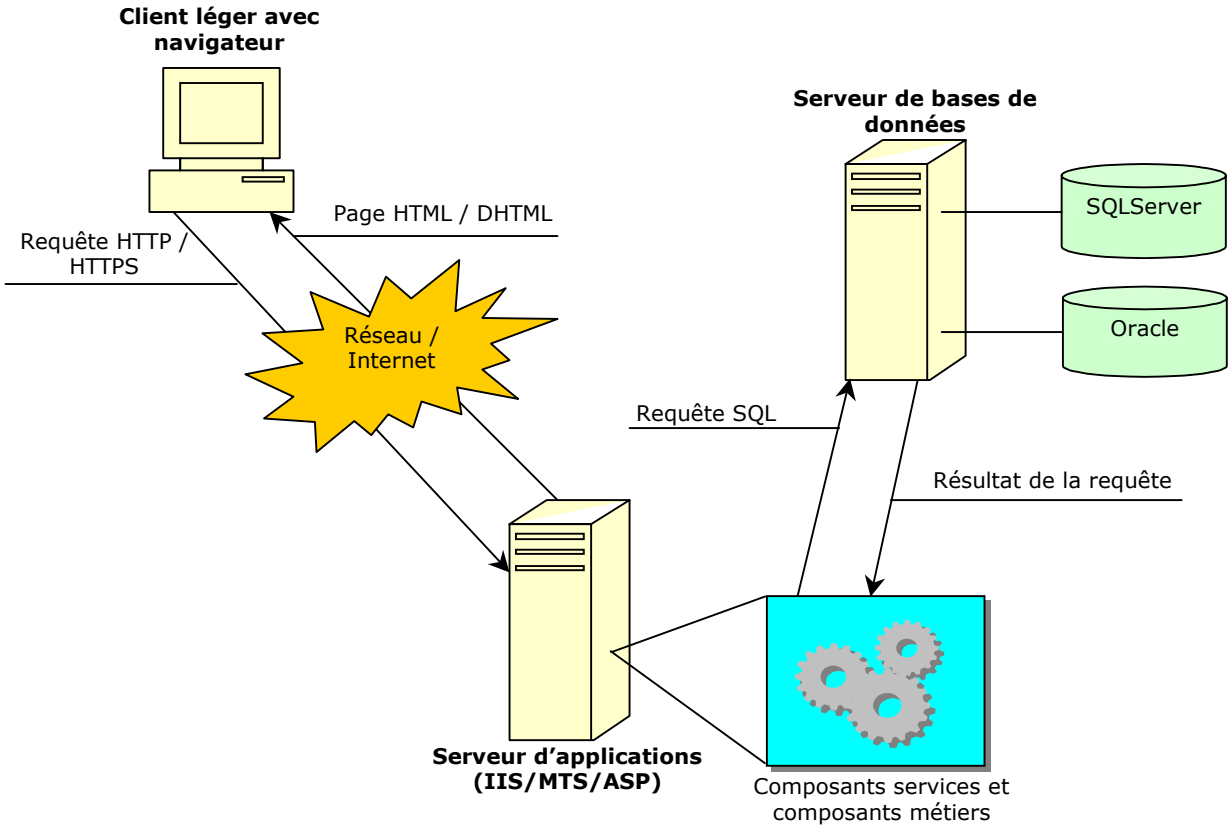
Il doit être possible de récupérer les mises à jour de la base de données et les modifications apportées aux données lors d'un crash système ou d'un problème de matériel.
C'est le travail des fichiers logs et des sauvegardes du système de la base.

3 - Architecture 3-tiers avec MTS

Voyons comment s'insère MTS dans une architecture 3-tiers typique avec un client léger, et quelles sont les fonctionnalités offertes par un tel système, lorsqu'il est mis en relation avec d'autres technologies comme ADO ou SSL.



Architecture 3-tiers (source Microsoft)



3.1 Caractéristiques techniques

- ✔ Transactionnel objet
- ✔ Optimisation des ressources systèmes (connexion réseau, SGBD…etc.)
- ✔ Gestion fixe de la sécurité et des contrôle d'accès à chaque composant (sécurité NT)
- ✔ Interface universelle d'accès aux données (ADO)

3.2 Sécurité

- ✔ Sécurité de communication Client/Serveur d'applications par le protocole SSL.
- ✔ L'accès aux bases de données s'effectue par les objets métiers.
- ✔ MTS permet la notion de "rôle pour gérer la sécurité" : sécurité déclarative basée sur la sécurité NT.
- ✔ On peut implémenter une sécurité plus fine aux niveaux des objets métiers : sécurité programmatique (objet de Context MTS)
- ✔ Hébergement sur des serveurs dédiés.

3 règles assurent la sécurité de l'architecture :

- ✔ L'authentification (référentiel utilisateurs).
- ✔ Le contrôle d'accès au niveau des composants.
- ✔ L'intégrité et la confidentialité (protocole SSL).

3.3 Administration

MTS offre une console d'administration MMC (Microsoft Management Console) permettant d'effectuer les tâche suivantes :

- ✔ Définir des packages ou lots, pour découper de manière logique l'application. Ces packages contiennent les composants MTS, ainsi que l'ensemble de leurs caractéristiques.
- ✔ Modifier de manière graphique la configuration de chaque package.
- ✔ Déployer à distance les packages sur des serveur de production.
- ✔ Visualiser l'état des packages et des composants.

Remarque : Une interface COM permet d'automatiser le déploiement sur des serveurs multiples.

3.4 Modèle d'une application Web

On retrouve trois types d'objets dans le tiers traitement d'une application Web :

- ✔ Les objets "métiers",
- ✔ Les objets "managers",
- ✔ Les objets "services".

Les objets métiers représentent les concepts manipulés par l'application (un client, un contrat, un contact…etc.).

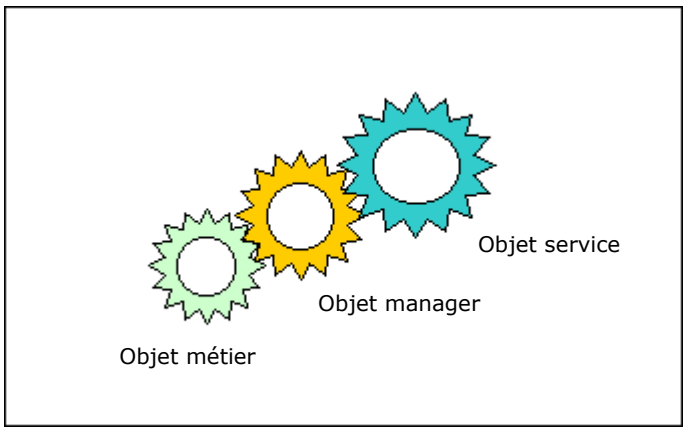
Les objets managers permettent de manipuler les objets métiers pour les objets services. En général, il y a un objet manager par objet métier. Ils sont dotés de deux types de méthodes :

- ✔ Des méthodes de gestion du cycle de vie et de persistance de l'objet métier associé (création, recherche, modification, suppression).

- ✔ Des méthodes métiers effectuant des traitements mettant en jeu plusieurs instances d'objet métier.

Les objets managers encapsulent les requêtes SQL pour les objets métiers, ce qui permet de les faire évoluer plus facilement, en maîtrisant mieux les impacts en cas de modification.

Les objets services offrent des services au tiers client. Chaque objet service regroupe des méthodes publiques fournissant des services métiers (Gestion de portefeuille, Gestion des contacts…etc.). En théorie, il ne devrait y avoir aucune requête SQL dans ces méthodes, puisque les accès aux données se font via la manipulation des objets métiers, dont la persistance est assurée par les objets managers uniquement.



3.5 Structure d'une application

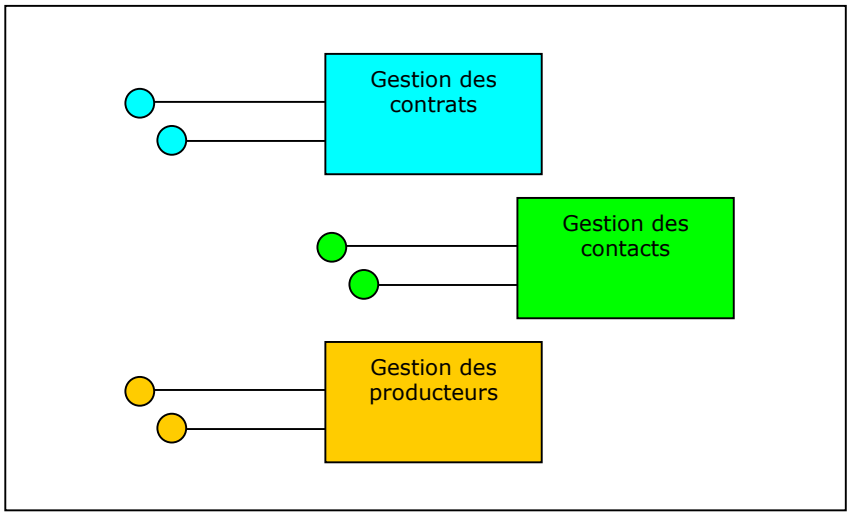
On retrouve deux types de composants dans une application Web :

- ✔ Les composants services, qui encapsulent les objets services.
- ✔ Les composants métiers qui encapsulent un objet métier et son objet manager.

Chacun des trois types d'objets définis dans le modèle d'une application Web se traduit sous forme de classe Visual Basic. Les composants sont regroupés dans des DLL ActiveX. Le modèle d'implémentation des composants est donc important et assure une maintenance ou une mise à jour plus facile.

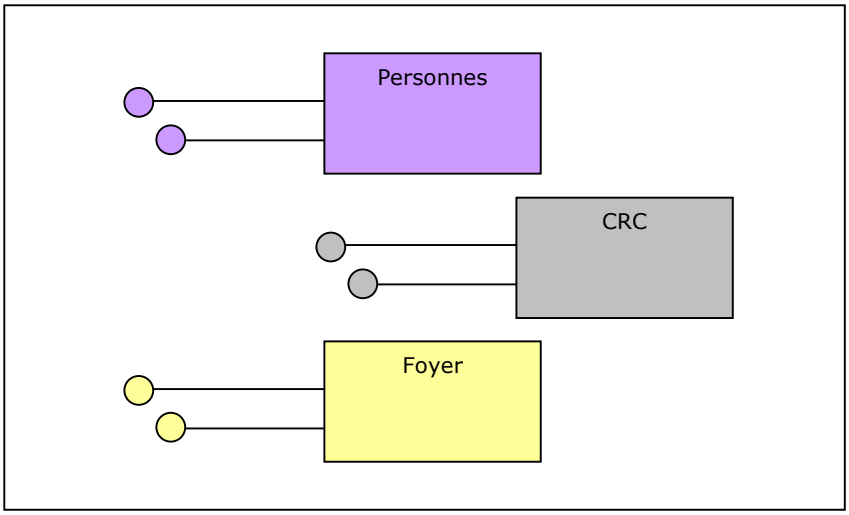
Par exemple, on pourrait :

- ✔ Regrouper les composants ActiveX services dans une seule DLL.



Composants services

- ✔ Regrouper les composants ActiveX métiers appartenant au même modèle conceptuel dans une DLL.



Composants métiers

On comprend aussi toute l'importance d'assurer une compatibilité des composants ActiveX. En effet, si le composants services "Gestion des contrats" n'est pas compatible avec l'interface du composant métiers "Personnes", l'application Web ne fonctionne plus. Afin de résoudre ce problème, il est judicieux que les développeurs maîtrisent les mécanismes de maintenance de la compatibilité en amont dans Visual Basic : la compatibilité des versions et le polymorphisme.

3.6 Avantages

- ✔ Offre complète et intégrée à Windows NT (sécurité et administration).
- ✔ Le soutien de l'éditeur.
- ✔ Le modèle composant COM/ActiveX est robuste et stable, un des plus utilisé, c'est gage de pérennité.

3.7 Inconvénient

- ✔ La solution ne tourne que sous Windows NT.

4 - MSMQ

Les composants en file d'attente assurent une exécution différée en mode asynchrone lorsque les composants travaillant en coopération sont déconnectés. En d'autres termes, vous pouvez utiliser cette technologie pour collecter une série de messages, les envoyer sur un serveur afin de les exécuter, et recevoir les résultats lorsqu'ils ont terminé.

Prenons le cas d'un voyageur de commerce qui se déplace de site en site pour aller chez les clients. Lors de chaque visite, il saisit la commande du client sur son ordinateur portable. A la fin de sa journée, il se connecte sur le réseau de sa compagnie pour envoyer les commandes. Pour chaque ordre envoyé, il reçoit une confirmation, et si le client désire recevoir sa marchandise immédiatement, il le notifie sur la commande.
Grâce à MSMQ, les ordres sont placés dans une file locale qui sera traité la prochaine fois que le voyageur de commerce se connectera sur le réseau d'entreprise. Quand son ordinateur portable est connecté, les ordres sont transmis pour être exécuté sur l'ordinateur distant. Tous les résultats générés par celui-ci sont placés dans la queue et seront envoyés au voyageur à sa prochaine connexion.
Lorsqu'un ordre doit être traité immédiatement, le même processus est utilisé. Néanmoins, et pour autant que le portable soit connecté, l'ordre est placé tout de suite, et la réponse arrive dès que les résultats de la requête sont disponibles.

Même si cet exemple peut sembler simpliste, considérons la situation où un serveur de transactions doit exécuter des requêtes à intervalles irréguliers. Parfois, il ne sera pas capable d'exécuter les requêtes de suite. MSMQ va autoriser l'exécution de requêtes selon le principe « Premier arrivé, premier servi » (FIFO), et l'application devra attendre une seconde ou deux avant de recevoir la réponse. Si les résultats ne sont pas disponibles dans l'immédiat, l'application annoncera par un message que la requête a bien été reçue et qu'il faudra attendre pour obtenir le résultat. Mais lorsque les résultats sont disponibles immédiatement, ils peuvent être envoyés tout de suite.

5 - IMDB

La base de données en mémoire IMDB est conçue pour fournir un accès de haute performance à l'information. Elle ne sert pas à remplacer la base de données, mais offre un cache amélioré permettant de stocker une grande quantité de ces données. IMDB fonctionne le mieux lorsque l'information est stockée sur une autre machine que le serveur et que les accès se font en lecture seule. Comme IMDB garde une copie des données de la base localement, cela signifie que ces données sont transférées en une fois du serveur à la machine sur laquelle fonctionne IMDB. Dans une application qui possède une table où des informations sont régulièrement consultées, le temps passé à obtenir les données ainsi qu'à effectuer les transactions est considérablement diminué.
Il est également possible de construire des tables temporaires sur IMDB. Ces tables existeront le temps de l'application, et lorsque celle-ci se termine, les tables disparaîtront. Cette technique est utile pour garder des résultats intermédiaires de requêtes ou de données couvrant plusieurs transactions. IMDB vit dans le même processus que le composant accédant les données en mémoire, ce qui explique les performances accrues d'un tel système.

IMDB ne supporte pas SQL en raison de la surcharge du langage : l'impact sur les performances s'en ressentirait immédiatement. Cela signifie que vous ne pouvez pas créer d'objet en utilisant des instructions SQL, ni exécuter des requêtes SQL sur vos tables.

Comme tout outil qui améliore les performances des applications, avant d'installer IMDB, il faut se poser quelques essentielles. Si votre serveur est lent en raison de son manque de mémoire, IMDB sera certainement le dernier outil à utiliser. D'un autre côté, si votre serveur est submergé par l'activité des entrées/sorties, IMDB peut certainement changer la situation.
Il vous faut donc pratiquer quelques mesures avant de se lancer avec IMDB : mesures de performances tout d'abord (nombre de transactions par seconde, nombre de requêtes sur la base de données, temps de réponse, …) afin de comprendre à quel moment le pic saturant votre serveur intervient. Mesures de statistiques du système ensuite (utilisation du processeur, utilisation de la mémoire, taux de pagination, activité des entrées/sorties).
Ces statistiques étant connues, vous aurez une meilleure idée de l'impact du changement de votre système si vous utilisez IMDB, et pourrez ainsi comparer comment se déroule les choses avant et après.

Microsoft estimant que IMDB ne répond pas aux attentes des utilisateurs, la compagnie a donc pris la décision de ne la fournir que dans la *Release Candidate 2* de Windows 2000 et non dans la version commerciale.

IMDB est installé et configuré à partir de l'utilitaire Services de composants de Windows NT 4 (Menu Démarrer → Programmes → Outils d'administration → Services de composants), sous l'onglet Options de la fenêtre de propriétés de l'ordinateur.

Dans la mesure où cette technologie semble disparue (elle devrait néanmoins refaire surface sous un autre nom… ☺), nous n'en parlerons pas plus ici.

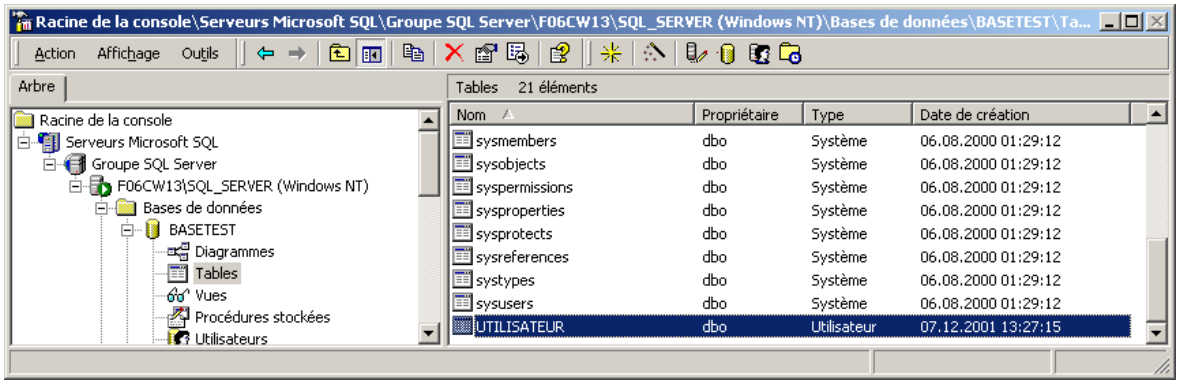
6 - Accès à une base de données distante par ASP / composant COM

Nous allons développer une DLL ActiveX qui contiendra des routines permettant de se connecter à une base de données SQL Server, en appelant une page ASP (sur serveur IIS). Cet exemple, même s'il est simple, permet de se sensibiliser à l'utilisation de composant COM, facilitant la vie du programmeur et offrant une stabilité et une rapidité d'exécution sans comparaison possible avec un programme de script (VBScript avec ADO par exemple).
Nous n'utiliserons donc pas les possibilités offertes par COM+, l'auteur n'ayant pas eu suffisamment de temps pour mettre en œuvre la technologie MTS couplée à une base de données. Néanmoins, l'exemple suivant couvre bien les possibilités d'ASP en relation avec COM offertes au développeur.

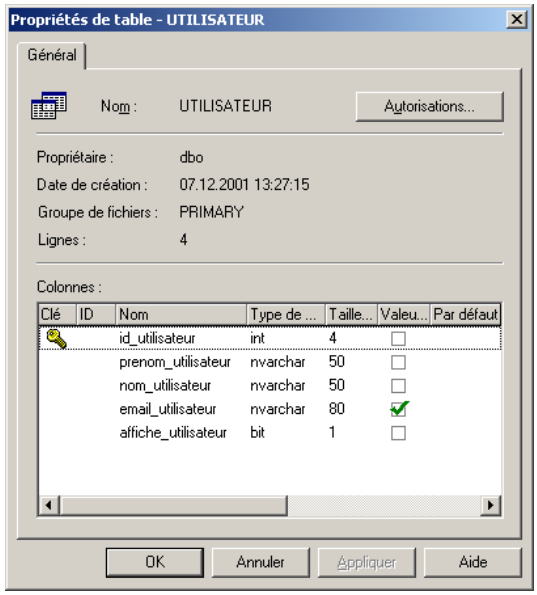
6.1 Mise en place de la base de données

Nous ne rentrerons pas dans les détails de l'installation de SQL Server, celle-ci étant extrêmement vite effectuée, et de surcroît très simple.

Créons une base de données sous SQL Server, nommée `BASETEST`, et possédant une table `UTILISATEUR`. La figure suivante montre l'emplacement de cette table, disponible depuis l'outil de gestion de SQL Server.



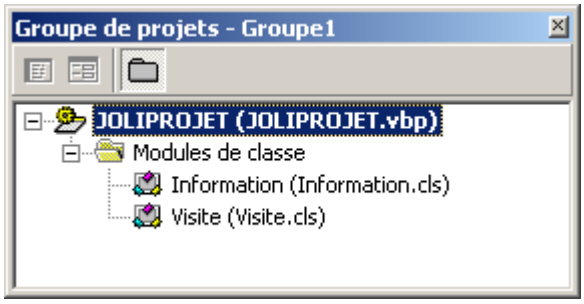
La structure de cette table (notez que tous les champs ne vont pas nous servir dans l'exemple) est la suivante :



Ajoutons ensuite quelques utilisateurs en remplissant au moins les champs nom et prénom, puis fermons l'utilitaire de gestion de SQL Server.

6.2 Conception du composant COM

Sous Visual Basic 6, commençons un nouveau projet DLL ActiveX, et renommons le projet ainsi que le module de classe, comme le montre la figure suivante (nous y avons déjà ajouté un second module de classe que nous détaillerons plus tard) :



Dans les propriétés du projet, sous l'onglet Créer, saisissons `JOLIPROJET` dans le champ Nom du produit de la liste Type.

Définissons à présent la première fonction, appelée `TesteComposant`, qui renverra le nom du composant dans une page ASP. Il s'agit là d'une fonction toute simple mais qui a le mérite de servir de test du composant lors du développement.

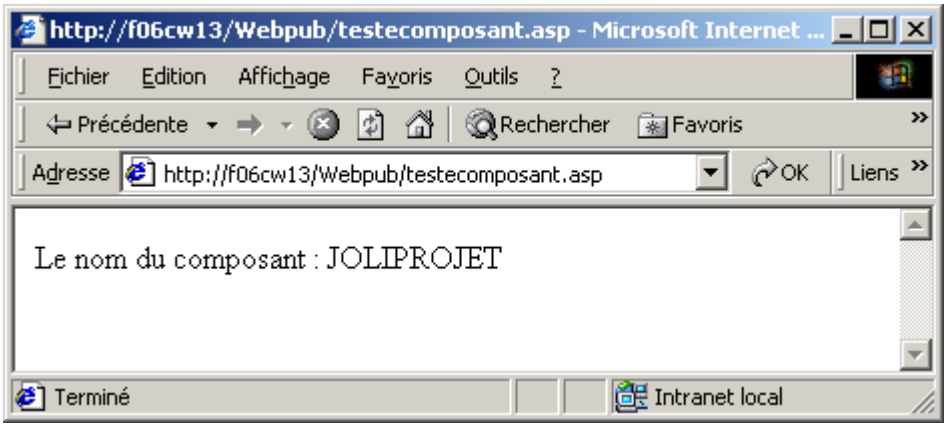
```
Option Explicit

Public Function TesteComposant() As String
    TesteComposant = App.ProductName
End Function
```

Cette fonction renvoie une chaîne de caractères et ne reçoit pas de valeur. Pour renvoyer des informations à l'utilisateur, nous utilisons des fonctions. Si nous avons à faire un traitement divers sans renvoyer d'informations, nous utiliserons une procédure.

Sauvegardez le tout. Si vous développez sur la machine de production, lancez l'exécution, car même si il ne se passe rien, la DLL sera ainsi inscrite. Nous allons maintenant créer une page ASP qui va utiliser notre fonction : testecomposant.asp.
Pour utiliser notre fonction, nous devons créer l'objet, définir le module de classe et appeler la fonction. Le code suivant indique la procédure à suivre :

```
<%
Dim strNom
' créons l'objet
Set objJOLIPROJET=Server.CreateObject("JOLIPROJET.Information")

' affectons le resultat de notre fonction à notre variable strNom
strNom=objJOLIPROJET.TesteComposant
' affichons le resultat
Response.write("Le nom du composant : "&strNom)

'liberons les ressources
set objJOLIPROJET=nothing
%>
```

Pour tester la page depuis la machine de production OU depuis une machine distante, il faut sauvegarder celle-ci dans le répertoire Webpub du serveur IIS. Dans notre cas, le fichier ASP est disponible à l'adresse http://f06cw13/Webpub/testecomposant.asp, et voici l'écran correspondant :



Le nom du composant s'affiche bien, il est transmis depuis la page ASP qui crée un objet du type `JOLIPROJET.Information` et appelle la fonction `TesteComposant`.

6.3 Accès à la base de données à l'aide du composant COM

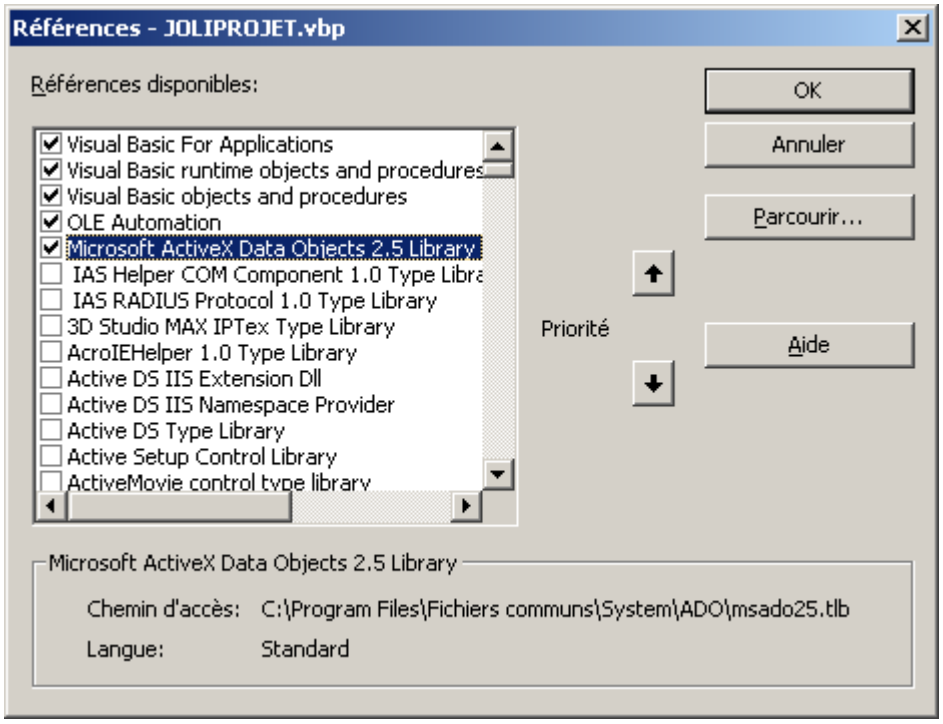
Attaquons nous à la base de données. Pour cela, nous allons définir un nouveau module de classe que nous appellerons `Visite`. Nous définirons ensuite une fonction (`valideUtilisateur`) qui vérifiera si un utilisateur existe ou pas. Mais avant tout, définissons notre connexion à la base, avec une procédure d'initialisation nommée `Class_Initialize()` :

```
Private strDSN As String

Private Sub Class_Initialize()
    strDSN = "driver={SQL Server};" & "server=F06CW13; database=BASETEST"
End Sub

Et le code de la fonction valideUtilisateur est le suivant :


Public Function valideUtilisateur(ByVal nom As String, _
                                  ByVal prenom As String) As Integer

On Error GoTo afferreur

    Dim strSQL As String
    Dim oRS As New ADODB.Recordset

    ' la requete envoyee sur la base
    strSQL = "SELECT * FROM utilisateur WHERE prenom_utilisateur=" _
             & prenom & " AND nom_utilisateur = " & nom & ""

    ' execution
    With oRS
        .CursorLocation = adUseClient
        .Open strSQL, strDSN
    End With
```

```
    ' Si il n'y a pas d'enregistrements correspondants :
    ' la fonction renvoie -1
    ' nous utilisons un integer et pas un booleen pour pouvoir par la suite
    ' modifier le programme de maniere a envoyer d'autres valeurs.
    If oRS.EOF Then
        valideUtilisateur = -1
    Else
        valideUtilisateur = 1
    End If

    oRS.Close
    Set oRS = Nothing

' traitement des erreurs minimal
afferreur:

End Function
```

Cette fonction renvoie un `Integer` et reçoit deux paramètres : le nom et le prénom d'un utilisateur. Nous avons défini une requête SQL qui cherche un enregistrement avec ces deux paramètres. Pour cette vérification, on ouvre donc une connexion et on définit un recordset (`oRS`). La condition `oRS.EOF` teste si le recordset arrive tout de suite à la fin (dans le cas où aucun enregistrement correspondant n'est trouvé). Dans ce cas, la fonction renvoie 1, sinon –1. Puis nous libérons les ressources en fermant le recordset.

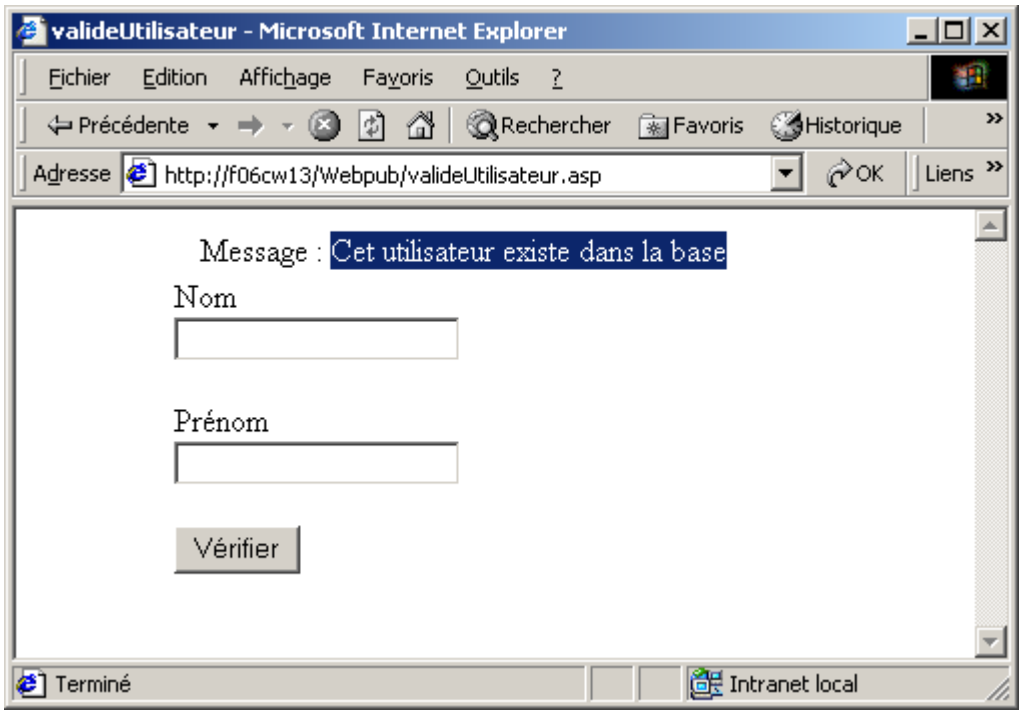
Afin que le projet Visual Basic utilise correctement la technologie ADO, il faut inscrire une référence depuis Projet → Références… sur la librairie ADO, tel que le montre la figure suivante :



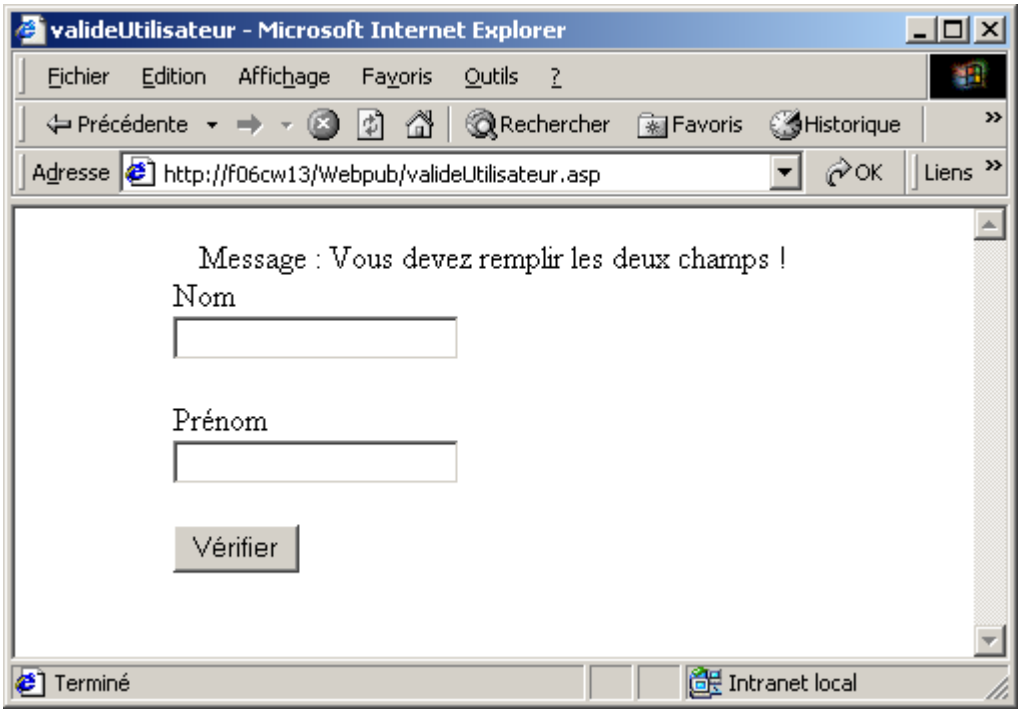
Construisons maintenant la page ASP qui proposera un formulaire contenant deux champs de saisie, pour le nom et le prénom. Le bouton `Vérifier` renverra sur une partie du code qui appellera le composant et la fonction `valideUtilisateur`. Selon la réponse (1 ou –1), nous afficherons un message.

```
<%
' récuperons defaction
'si c'est le premier chargement, l'action de vérification ne sera pas
effectuée
defaction=request.form("defaction")

if defaction="verifier" then
' récuperons le nom et le prenom du formulaire
strPrenom=Server.HTMLEncode(trim(request.form("strPrenom")))
strPrenom=replace(strPrenom,"'","''")
strNom=Server.HTMLEncode(trim(request.form("strNom")))
strNom=replace(strNom,"'","''")
if strNom="" OR strPrenom="" then message="Vous devez remplir les deux
champs !"
      if message="" then
      ' créons l'objet
      set objJOLIPROJET=Server.CreateObject("JOLIPROJET.Visite")
      ' appelons notre fonction valideUtilisateur
      intResultat=objJOLIPROJET.valideUtilisateur(strPrenom,strNom)
      ' définissons un message selon le résultat
      select case intResultat
      case -1
      message="Cet utilisateur n'existe pas dans la base !"
      case 1
      message="Cet utilisateur existe dans la base"
      end select
      end if
end if
%>

<html>
<head><title>valideUtilisateur</title></head>

<body>
<table width="70%" border="0" cellspacing="0" cellpadding="0"
align="center">
  <tr>
    <td>
      <div align="center">Message : <%=message%></div>
    </td>
  </tr>
  <tr>
    <td>
      <form method="post" action="valideUtilisateur.asp">
        <p>Nom<br>
          <input type="text" name="strNom">
        </p>
        <p>Pr&eacute;nom<br>
          <input type="text" name="strPrenom">
        </p>
        <p>
        <input type="hidden" name="defaction" value="verifier">
          <input type="submit" name="Submit" value="V&eacute;rifier">
          <br>
        </p>
      </form>
    </td>
  </tr>
</table>
</body>
</html>
```

Après avoir exécuté le projet pour inscrire la DLL, vous pouvez donc tester la page. Avec de bons paramètres (un nom et un prénom inscrit dans la base de données), la page `valideUtilisateur` est la suivante :



Tentez ensuite de ne remplir qu'un des champs et validez. La page suivante apparaît :



6.4 Déploiement du composant

Notre composant est donc réalisé. Afin de l'installer sur le serveur de production, nous allons l'empaqueter - en utilisant le même outil (Assistant Empaquetage et déploiement) que celui que nous verrons dans les détails pour l'application DCOM (chapitre suivant) - solution plus simple que de devoir inscrire soi-même (avec regsvr32.exe) à la ligne de commande JOLIPROJET.dll !

L'empaquetage se déroule très simplement, en choisissant une installation standard (type d'empaquetage), et en validant tous les écrans les uns après les autres.

Lors du déploiement sur le serveur de production, attention tout de même : vérifiez que la base `BASETEST` existe bien, qu'elle comporte bien la table `UTILISATEUR`, et que les pages ASP sont bien installées au bon endroit sur le serveur IIS.

6.5 Conclusion

L'exemple que nous avons montré ici est très simple. Cela dit, la majorité des projets utilisant un composant pour un tel accès est similaire à ce que nous avons exposé. L'utilisation de composants est très importante en terme de performance et de maniabilité, et il faudrait dès que possible utiliser un composant plutôt qu'un long script ASP, beaucoup plus lent. Les ressources du serveur seront ainsi mieux utilisées.

Bibliographie critiques et liens Internet

- Visual Basic developer's guide to COM and COM+, Wayne S. Freeze, Sybex 2000, 460 pages.
  L'ouvrage qui permet le mieux (parmi ceux cités dans cette bibliographie) de comprendre les technologies de composants de Microsoft, et met en relation simplement et efficacement les différents concepts. La partie consacrée à DCOM est malheureusement très courte, et n'offre qu'une petite introduction au protocole. L'auteur traite également COM+, MSMQ, et IMDB.

- Developping COM/ActiveX components with Visual Basic 6, Dan Appleman, Sams 1999, 860 pages.
  Cet ouvrage s'adresse avant tout aux développeurs professionnels sur VB, et montre les meilleures techniques pour développer des composants COM et DCOM. D'un niveau très nettement plus élevé que le titre précédent, cette « bible » (presque 900 pages !) passe en revue la conception d'objets COM mais aussi et surtout les mécanismes internes lors de l'exécution des composants.

- http://msdn.microsoft.com, le lien INCONTOURNABLE pour développer sous plate-forme Windows.

- http://www.microsoft.com/com/tech/com.asp, la page de Microsoft pour tout ce qui concerne la technologie COM.