Microsoft Visual Studio 2005 msdn

Introduction à ADO .NET avec Visual Basic 2005

1) Présentation ADO.NET 2.0

1.1) Architecture

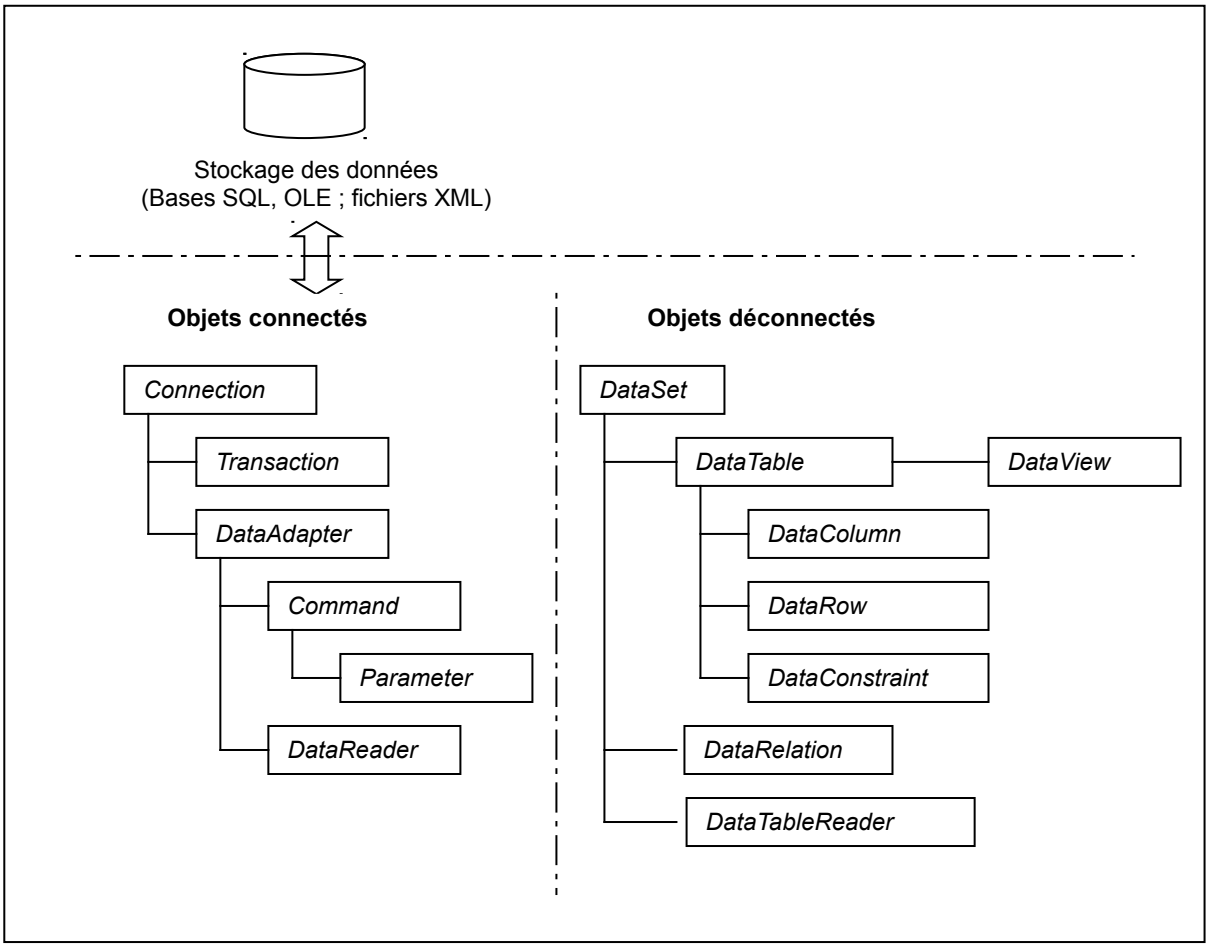
ADO.Net 2.0 est une **technologie**, **intégrée au Framework Net**, offrant des **méthodes d'accès aux données**. Elle dérive de la technologie ADO.Net, qui lui-même succédait à ADO. Cette bibliothèque de classes permet de récupérer, manipuler, ou mettre à jour des données provenant d'une source de données (base de données, fichier XML, etc…)

La grande révolution d'ADO.Net est qu'elle repose sur un **modèle d'accès aux données déconnectés** par défaut. Contrairement aux technologies précédentes tel que ADO, qui accédaient aux données de façon connectés en continu, trop coûteuse en ressource et peut recommandable pour des applications complexes.

http://msdn.microsoft.com/library/fre/default.asp?url=/library/FRE/cpguide/html/cpconwhyadonet.asp

ADO.NET a été conçu pour répondre aux besoins de ce nouveau modèle de programmation : architecture de données déconnectée, intégration poussée avec XML, représentation commune des données avec possibilité de combiner les données de plusieurs sources de données différentes et fonctionnalités optimisées pour permettre l'interaction avec une base de données, tout cela étant natif du .NET Framework.

Voici ci-dessous le schéma de l'architecture ADO.Net :



ADO.Net 2.0 peut se « découper » en 2 groupes principaux qui sont :

- Le **groupe de données** (objets déconnectés) : stockant les données sur la machine locale
- Le **fournisseur** (objets connectés) : gérant la communication entre le programme et la base de données

Le Framework .Net met à disposition **différents types de fournisseurs** pour accéder à diverses sources de données :

- `System.Data.SQLClient` : gestion d'accès aux bases SQL Server (min V7)
- `System.Data.OleDb` : gestion d'accès aux sources de données via le driver OleDb
- `System.Data.Odbc` : gestion d'accès aux sources de données via le driver Odbc
- `System.Data.OracleClient` : gestion d'accès aux bases Oracle
- `System.Data.Xml` : Accès aux fonctionnalités de SQL Server (min 2000)

1.2) Les objets ADO.Net 2.0

Ci-dessous un tableau décrivant rapidement la fonctionnalité des principaux objets ADO.Net 2.0 :

Objet	Description
Connection	Gère les connections vers la base de données
Transaction	Gère les transactions vers la base de données. Un objet `Transaction` peut être récupérer à partir d'un objet `connection` via l'appel à la méthode `beginTransaction()`
DataAdapter	Sert de liaison entre la base de données et un `DataSet`. Vous pouvez remplir un DataSet avec les données de la base via la méthode `Fill` et utiliser l'appel à la méthode `Update` pour la mettre à jour en base les données modifiées en local dans le `DataSet`. Les propriétés `SelectCommand`, `UpdateCommand`, `InsertCommand`, `DeleteCommand` du `DataAdapter` vous permettes de définir vos traitements SQL.
Command	Gère les commandes SQL vers la base de données (SELECT, INSERT, UPDATE, DELETE) en appel in-line (code SQL en dur) ou procédure stockée (définit par les propriétés `CommandText` et `CommandType`). Vous avez la possibilité de passer un ensemble de paramètres (la propriété `Parameters` attend une collection d'objet `Paramater`). L'exécution de la requête s'effectue via différentes méthodes suivant le traitement à effectuer : `ExecuteReader`, `ExecuteScalar`, `ExecuteNonQuery`, …
DataReader	Permet de lire 1 à 1, en mode connecté, les résultats retournés par l'exécution de la commande SQL (comme un recordset connecté en lecture seule en avance seul `MoveNext`)
DataSet	Container de `DataTable` et `DataRelation`, stocké en mémoire. C'est l'objet central de manipulation des objets déconnectés.
DataTable	Table de jeux de données résidant en mémoire. Il comporte une collection de colonnes `DataColumn` et éventuellement de contraintes `DataConstraint`. L'ensemble des données sont stockées dans une collection de `DataRow`. Chaque `DataRow` comporte un état (`RowState`) qui permettra de détecter les données modifiées, supprimées, ajoutées localement et de mettre à jour correctement en base ultérieurement.
DataView	Offre un mécanisme pour trier, filtrer un `DataTable`. On peut comparer l'objet `DataView` à une vue SQL.
DataTableReader	Table de jeux de données, en mode déconnecté, qui permet de lire 1 à 1 les éléments le composant

Voici un exemple de code sur récupérer des données en base :

```
Imports System.Data.SqlClient
…

    Public Function GetDsContacts() As DataSet

        ' Objet SqlConnection avec la chaîne de connection récupérée
        ' dans le app.config
        Dim conn As New SqlConnection(My.Settings.DBContactsConnectionString)

        ' Objet SqlDataAdapter instancié avec la requête Select
        ' et notre objet conn
        Dim adap As New SqlDataAdapter("SELECT * FROM Contact", conn)

        ' Déclaration d'un DataSet pour stocker les données de la table Contact
        ' en base qui seront récupérées (résultant du "SELECT * FROM Contact")
        Dim ds As New DataSet

        Try
```

```
            ' Ouverture de la connection en base
            conn.Open()

            ' Remplissage du DataSet avec les données récupérées.
            ' Un DataTable sera créé automatiquement pour contenir
            ' les données de la table Contact
            adap.Fill(ds)

        Catch ex As Exception

            ' Fermeture de la connection en base
            If conn IsNot Nothing Then
                conn.Close()
            End If

            ' Libération des objets en mémoire
            If adap IsNot Nothing Then
                adap.Dispose()
            End If

        End Try

        Return ds

    End Function
```

Cet exemple vous donne un aperçu de l'utilisation des objets ADO.Net. Mais vous verrez au travers du lab associé que VS2005 offre de nombreux assistants, qui vous aideront à construire l'architecture d'accès aux données de votre application rapidement et efficacement ; sans écrire une ligne de code.

1.3) Couper les liens entre la technologie ADO.Net avec celle d'ADO

Ne comparez pas la technologie ADO.Net avec celle d'ADO. Ces technologies sont très différentes.

En plus de reposer sur un modèle déconnecté, il est important de bien comprendre qu'**ADO.Net ne dispose plus de notion de curseur** comme en ADO. Les méthodes *MoveFirst,*etc… n'existe plus.

Attention les données, récupérées d'une source, dans un **DataTable**, correspondent à une **copie, en mémoire, à un instant de l'état des données**. **Toutes modifications apportées par d'autre utilisateur sur la source de données initiale, ne sont pas directement perçus dans le DataTable**. Toutefois, avec **SQLServer 2005**, vous avez la possibilité de **notifier** toutes modifications apportées en base par d'autre utilisateur et de venir avertir au niveau de votre application ces changements afin de gérer, par code, le traitement à effectuer en conséquence.

2) Construction d'une base SQL Server 2005 au travers de l'environnement Visual Studio 2005

2.1) Les nouveautés de SQL Server 2005

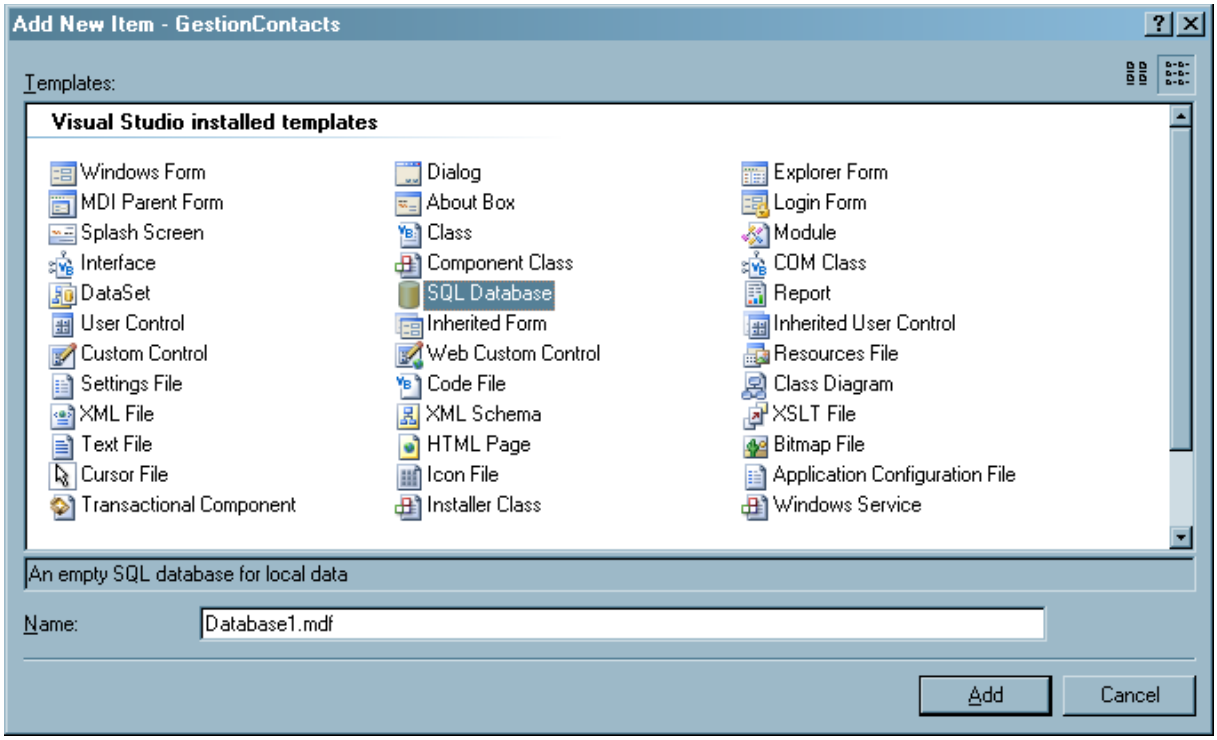
L'arrivée de SQL Server 2005 marque un grand tournant dans le domaine des bases de données.

Voici une liste non exhaustive des grands **changements apportés à SQL Server 2005** (comparé à SQL Server 2000) :

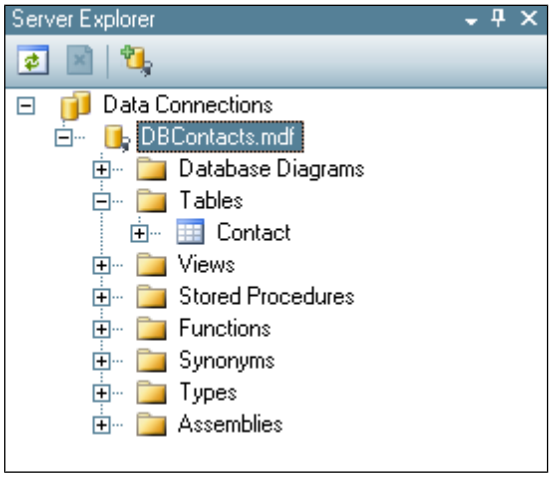
1. **Intégration de la CLR** (*Common Language Runtime*) :
    a. **UDF** : types de données utilisateurs
    b. Ecrire vos requêtes dans un langage MSIL (C#, VB.Net)
    c. Commandes **asynchrone**
    d. **MARS** (*Multiple Access RecordSet*) : permet d'ouvrir de plusieurs curseurs en même temps sur une même connexion
    e. les entités SQL ont leur correspondance « objet » (SMO)
2. **Nouveaux Types de données** :
    a. XML
    b. NVARCHAR(max)
3. Nouvelles fonctionnalités T-SQL :
    a. **CTE** (*Common Table Expression*) : WITH, requête récursives
    b. Les **opérateurs** : PIVOT, UNPIVOT, EXCEPT, INTERSECT…
    c. **Méthode de classements** : Row_Number, Rank, Dense_Rank, NTile…
4. Plus de **sécurité** !! :
    a. les définitions de droits sur les bases sont stockées dans des **schémas** ; les utilisateurs sont reliés aux schémas.
    b. cryptage
5. Encore des **services** : notification, évolution des DTS, support SOAP HTTP, reporting

2.2) Création de la base de donnée SQL Server 2005 sous Visual Studio 2005

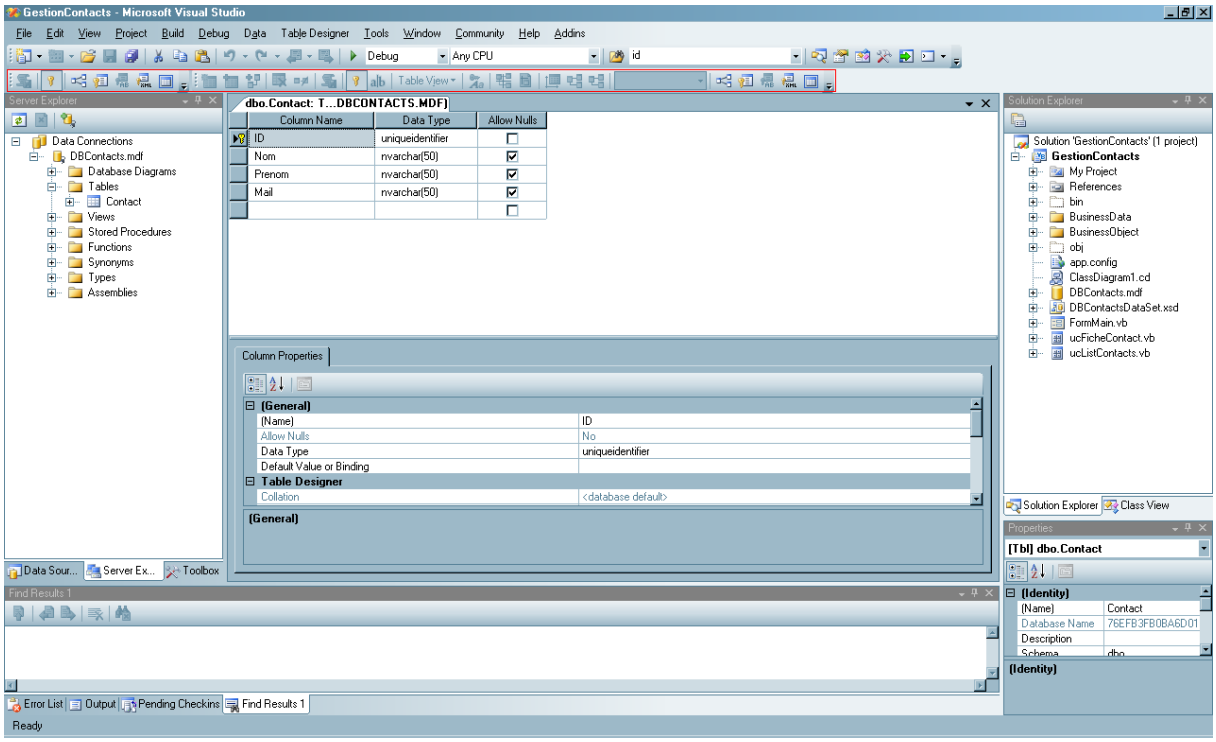
Sous Visual Studio 2005, vous avez la possibilité de venir créer directement votre base SQL Server 2005 (fichier «.mdf ») dans votre projet en cours :



L'ensemble des connections de votre projet apparaissent alors sous le nœud « Data Connections » de la fenêtre « **Server Explorer** ». Vous retrouvez ainsi directement créer et manipuler l'ensemble des éléments de votre base de données.



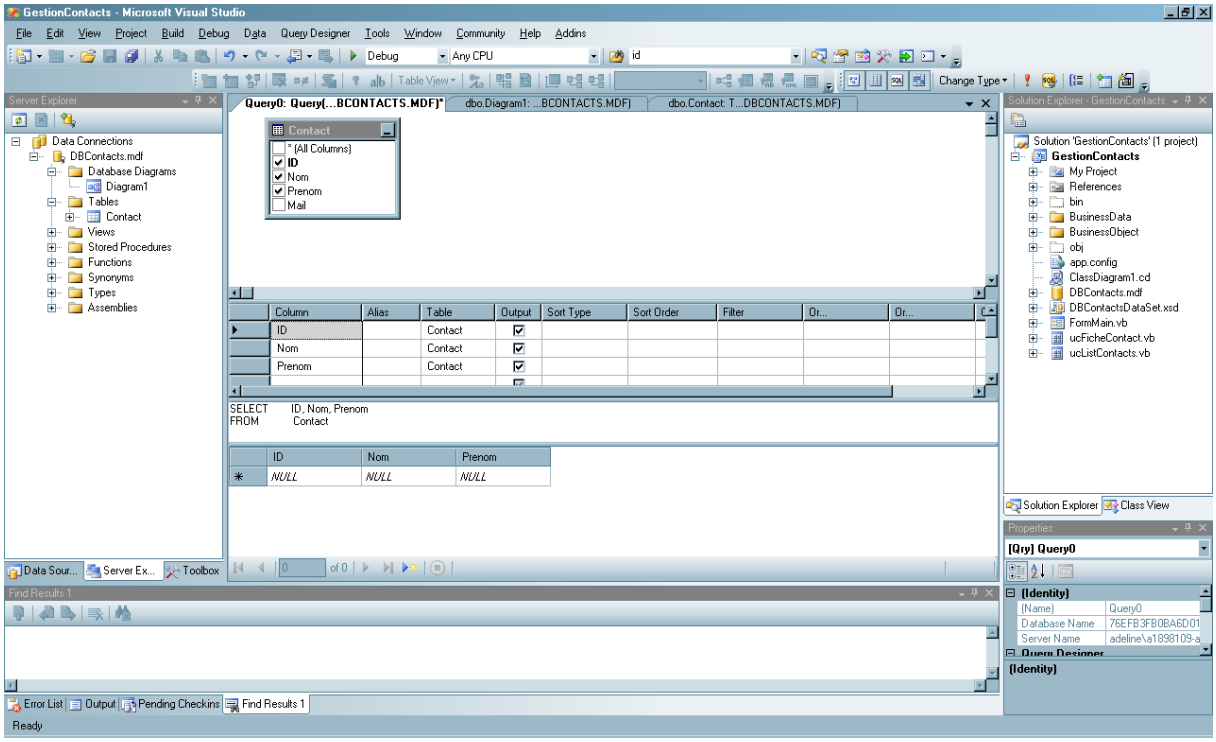
La barre d'outils « **Diagramm Class** » (encadré en rouge sur l'écran ci-dessous) vous permet de construire votre diagramme comme sous SQL Server. L'ensemble des diagrammes sont stockées dans le répertoire prévu à cet effet : « Database Diagrams ». Vous pouvez ainsi **administrer facilement le schéma de votre base de données**.



Remarquez ci-dessus le résultat de la construction de la table « Contact » (voir Lab).

2.3) Découverte du Query Analyzer

En effectuant un click droit sur votre base SQL Server, Visual Studio 2005 vous propose le menu « New Query » qui vous affichera l'environnement du « **Query Analyzer** ». Comme son nom l'indique il s'agit d'une interface vous permettant de **créer et d'exécuter facilement vos requêtes**. L'écran ci-dessous vous donne un aperçu graphique de cet environnement simple et complet.



3) Elaboration de la couche d'accès aux données via un DataSet typé

3.1) A quoi sert un DataSet typé

Comme nous l'avons abordé dans la partie sur la présentation de la technologie ADO.Net, réaliser la couche d'accès aux données n'est pas complexe en soit mais cela peut s'avérer long et fastidieux. Pour y parer, Visual Studio 2005 vous met à disposition, au travers de ce qu'on nomme les **DataSet typés**, des méthodes et outils permettant d'élaborer votre couche d'accès aux données sans la moindre ligne de code à écrire.

Un DataSet typé est en fait une classe avec les informations disponible sur les tables et colonnes au travers des propriétés.

Exemple d'accès à un champ :

- sans DataSet typé :

```
ds.Table("Contact").Rows(0).Item("ContactNom")
```

- avec DataSet typé :

```
ds.Contact(0).ContactNom()
```

De plus il expose des **méthodes personnalisées très fonctionnel**.

Exemple de récupération d'une ligne « vide » sur un DataTable définit dans le DataSet typé :

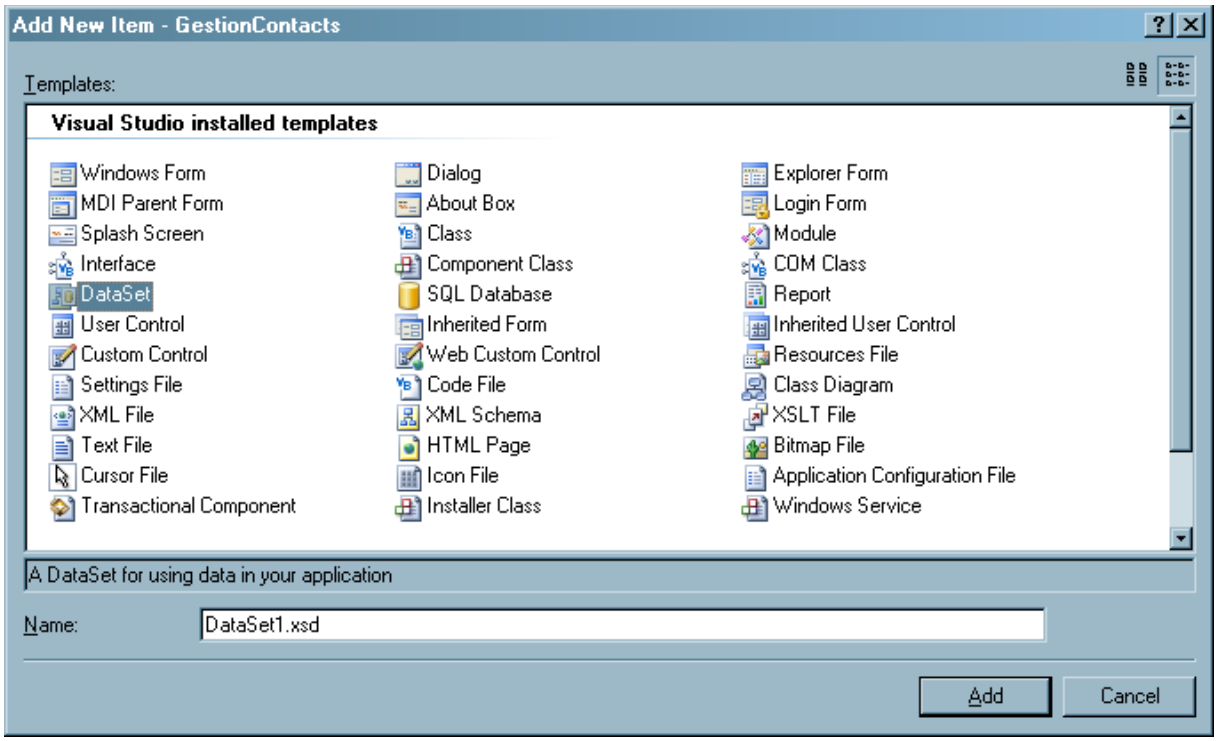
- sans DataSet typé :

```
ds.Table("Contact").NewRow()
```

- avec DataSet typé :

```
ds.Contact.NewContactsRow()
```

Visual Studio 2005 crée un fichier XSD par DataSet typé. Le format XSD définit un schéma XML ; ce n'est pas vraiment une coïncidence puisqu'en fait un DataTable est objet qui stocke en mémoire les données au format XML.

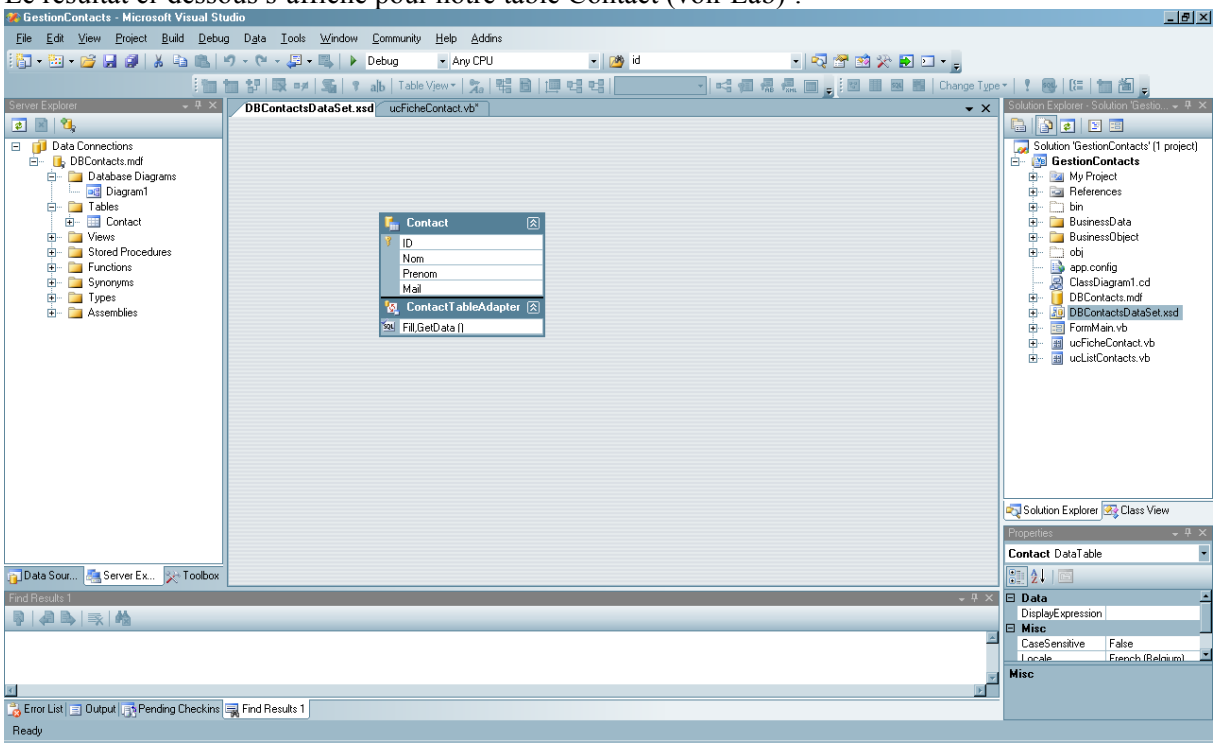

Add New Item - GestionContacts
Templates:
Visual Studio installed templates
Windows Form
MDI Parent Form
Splash Screen
Interface
DataSet
User Control
Custom Control
Settings File
XML File
Text File
Cursor File
Transactional Component
Dialog
About Box
Class
Component Class
SQL Database
Inherited Form
Web Custom Control
Code File
XML Schema
HTML Page
Icon File
Installer Class
Explorer Form
Login Form
Module
COM Class
Report
Inherited User Control
Resources File
Class Diagram
XSLT File
Bitmap File
Application Configuration File
Windows Service
A DataSet for using data in your application
Name:
DataSet1.xsd
Add
Cancel

3.2) Génération d'un DataSet typé sur base d'une table SQL Server 2005

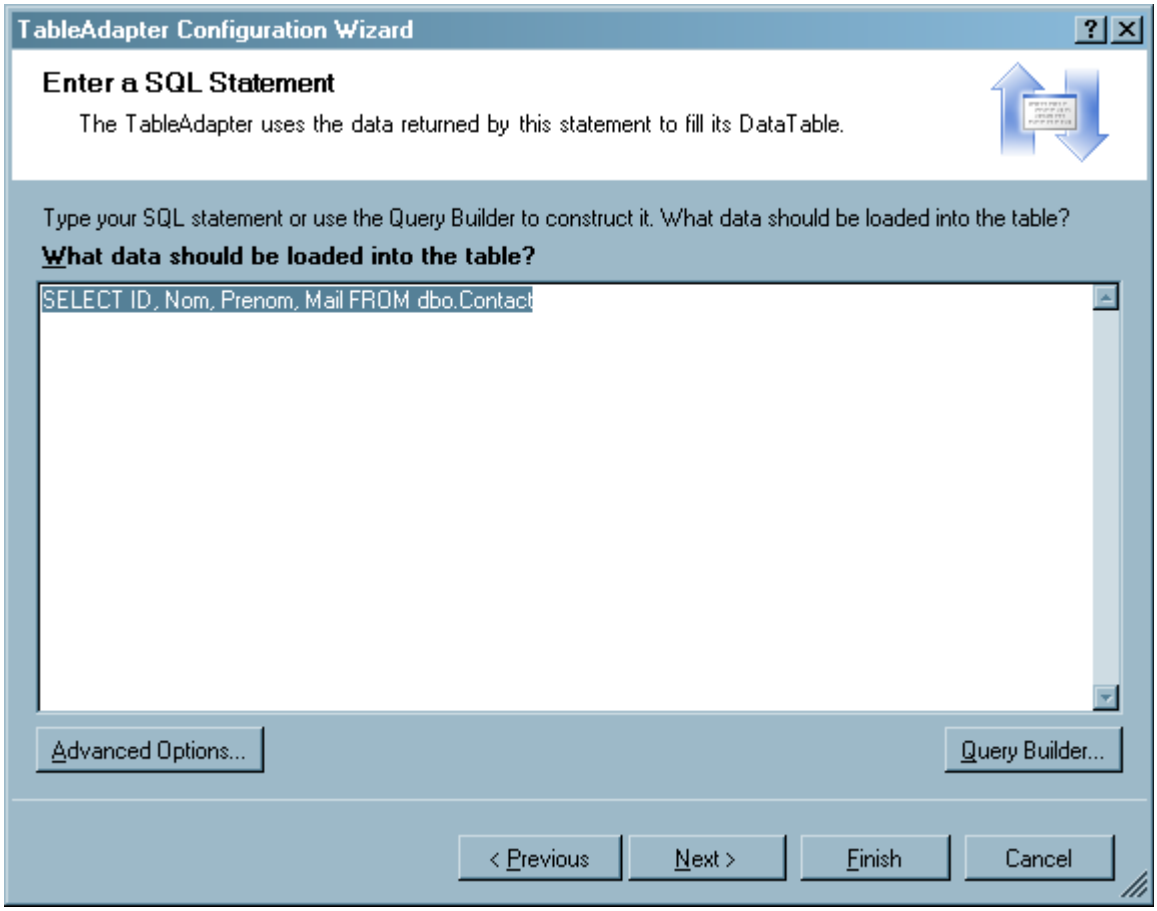
Avec Visual Studio 2005, vous avez la possibilité de générer automatiquement votre couche d'accès aux données au travers de DataSet typé.

En reprenant la fenêtre « **Server Explorer** » détaillant nos connections, vous pouvez glissez déposer les objets tables désirées sur le fichier XSD pour générer à la volé, au sein de notre DataSet, le DataTable et le DataAdapter correspondant. Le DataAdapter contiendra déjà les informations sur la connexion à la base de données.

Le résultat ci-dessous s'affiche pour notre table Contact (voir Lab) :



Il restera donc à paramétrer les **commandes SQL** (SELECT, INSERT, UPDATE, DELETE) du DataAdapter. Ceci ce fait très facilement au travers de l'**assistant** VS2005 ; cliquez droit sur le DataTable « Contact » et sélectionnez « **Configure** ».



Une fois les étapes de l'assistant achevées, notre couche d'accès aux données est réalisée pour notre table « Contact » sans que nous ayons écrit une seule ligne de code. Nous pouvons exploiter nos données récupérer de la table « Contact » au travers de notre DataSet typé.

```
' Déclaration du DataTable "Contact"
Public ListContacts As New DBContactsDataSet.ContactDataTable

' Déclaration du DataAdapter "Contact"
Dim daContacts As New DBContactsDataSetTableAdapters.ContactTableAdapter

' Chargement des données récupérer en base dans notre DataTable "Contact" au travers
de son DataAdapter
ListContacts = daContacts.GetData()
```

La méthode `GetData` viendra indirectement exécuter la requête SELECT stockée sur la propriété `CommandSelect` du DataAdapter (que nous avons généré au travers de l'assistant lors de la configuration du DataAdapter)

Une fois les données modifiées localement via le DataTable typé, il suffit alors d'appeler à la méthode « Update » du DataAdapter pour mettre à jour en base de données.

```
' Mise à jour en base de données des modifications apportées localement
daContacts.Update(ListContacts)
```

La méthode `Update` viendra indirectement exécuter les requêtes INSERT, UPDATE ou DELETE stockée sur les propriétés `CommandInsert`, `CommandUpdate`, `CommandDelete` du DataAdapter (que nous avons généré au travers de l'assistant lors de la configuration du DataAdapter)

3.3) Passage en revu de quelques méthodes sur les objets DataTable et DataSet

Ci-dessous un petit récapitulatif de méthodes déjà rencontré ou à découvrir des objets DataTable et DataSet.

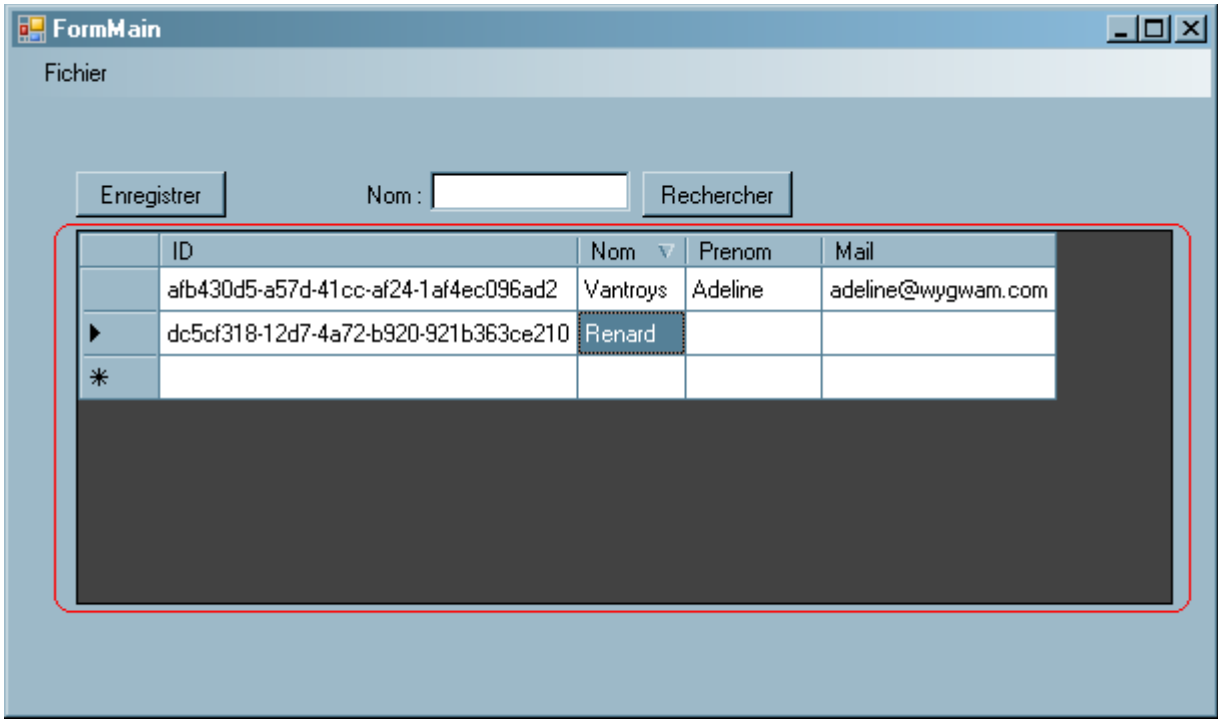
Objet	Méthode	Description
DataSet	`GetChanges`	Appel à la méthode `GetChanges` en récursif sur les DataTables compris dans le DataSet
	`Merge`	Permet de fusionner un ensemble de données de structure identique à la celle d'un des DataTables
DataTable	`AcceptChanges`	Valide les changements apportés localement sur la collection des DataTables du DataSet. L'état des lignes des DataTables (stockée sur la propriété RowState) passe à `DataRowState.Unchanged` (effectuer généralement après la mise à jour sur la source de données physique)
	`GetChanges`	Retourne l'ensemble des DataRows dont l'état a été modifié : `RowState <> DataRowState.Unchanged` (effectuer généralement avant la mise à jour sur la source de données physique)
	`Merge`	Permet de fusionner un ensemble de données de structure identique à la celle du DataTable
	`ReadXml`	Permet de charger directement le contenu d'un fichier XML dans un objet DataTable
	`WriteXml`	Permet d'écrire directement le contenu d'un DataTable dans un fichier XML

4) Liaison des données au DataGridView

4.1) Présentation du contrôle DataGridView

Le composant DataGridView est un objet, à l'apparence d'un **tableau**, très **complet permettant d'afficher et gérer des données**. On **lie** une **source de donnée** via sa propriété DataSource.

Vous pouvez laisser la génération des colonnes sur base de la structure de la source de données, ou bien venir explicitement ajouter ou modifier ces dernières (click droit sur le contrôle « DataGridView » sur la partie Design « Add Column »\ « Edit Column »).



Consultez la fenêtre de propriété de ce composant, vous trouverez bon nombre de fonctionnalités à paramétrer.

Nous l'abordons pas de le Lab mais sachez que vous pouvez effectuer ce qu'on appelle le « **master/detail** » avec un objet DataGridView. Il s'agit en fait de basculer d'une source de données parent vers sa source de données enfant au travers d'un DataRelation. L'article suivant détaille la démarche à suivre : http://msdn2.microsoft.com/en-us/library/c12c1kx4.aspx

4.2) Maîtriser la saisie des données dans le DataGridView

Le DataGridView possède de nombreux d'évènements où on peut intervenir lors de leur exécution. Entre autre, il est possible de venir contrôler et valider la saisie sur la ligne d'édition en cours. L'évènement qui nous permet d'effectuer cette validation se nomme `CellValidating`.

Parmi ces paramètres cet évènement comporte un objet de type `System.Windows.Forms.DataGridViewCellValidatingEventArgs`

Ce type d'objet offre les propriétés `RowIndex` et `ColumnIndex`, qui permettent de connaître, respectivement, la ligne et la colonne en cours de validation. La propriété `e.Cancel` indiquera si nous validons ou non la saisie ainsi faite.

4.3) Filtrer et trier à l'aide de l'objet DataView

Pour notre dernière partie de ce tutorial, nous allons nous attarder sur l'objet DataView. Pour rappel cet objet est lié à un DataTable et correspond en quelque sorte à un vue filtré ou trié du DataTable.

Si nous reprenons notre DataTable typé `ListContacts`, on peut récupérer la vue de notre DataTable via sa propriété `DefaultView`

```
Dim dv As DataView = ListContacts.DefaultView
```

Les propriétés du DataView :

Propriété	Description	Exemples
`rowFilter`	Permet de définir l'expression de filtre	`dv.RowFilter = "Nom LIKE '%" & tbNom.Text & "%'"`
`sort`	Permet de définir l'expression de tri	`dv.Sort = "ORDER BY Nom DESC"`

Tout contrôle lié, via la propriété DataSource, à notre DataTable typé, dont son DefaultView a été modifié, affichera ainsi les données filtrées et/ou triées en conséquence.

Conclusion

Au terme de ce tutorial, nous avons approché la technologie ADO.Net 2.0 qui offre une base solide pour l'élaboration d'application complexe nécessitant des accès à une source de données. Nous savons élaborer rapidement et simplement, à l'aide de DataSet typé, notre couche d'accès aux données vers une base SQL Server 2005. Nous avons découvert des objets comme le DataSet, le DataTable et le DataView permettant de manipuler rapidement des données stockées en mémoire. Visual Studio 2005 offre de nombreux outils facilitant la construction d'une base SQL Server 2005.

Enfin nous avons abordé le composant DataGridView qui offre de nombreuses fonctionnalités pour gérer l'affichage et manipuler de données. Nous savons charger un lot de données dans un objet DataGridView, mais également contrôler les saisies faite par l'utilisateur. Enfin nous avons vu que le DataView permet d'obtenir une vue de notre DataTable, et ainsi pouvoir le trier voir filtrer sur un critère.

Vous pouvez approfondir l'ensemble de ces notions au travers du lab associé.