Gérer les connexions aux données dans StrataFrame

Le point essentiel d'une connexion à une source de données, c'est la chaine de connexion.

Cette contribution tente de répondre aux questions suivantes :

- Comment stocker de façon permanente les sources de données nécessaires à une application ?
- Comment mettre à disposition ces sources de données à travers toute l'application ?
- Comment utiliser plusieurs sources de données dans une même application ?

Que faire avec une chaine de connexion ?

Une application StrataFrame conserve les chaines de connexion dans la collection **DataSources**, membre de la classe **DataLayer**. Le contenu de cette collection sera renseigné dans le code de lancement de l'application (voir *Figure 1 - Diagramme d'exécution d'une Application StrataFrame*)

Pour une application Winform, cette information est collectée dans la méthode ***GetConnectionString*** de l'*appmain.vb* ou du *appmain.cs*.
Pour une application Web, ce sera dans la méthode ***Application_Start*** du fichier ASP.net *global.asax*.

Comment stocker de façon permanente les sources de données nécessaires à une application ?

La question est simple : où et comment stocker les chaines de connexion qui seront utilisées dans une application ? cette question est récurrente quel que soit le langage de développement, et il existe 4 approches pour y répondre.

1) Dans l'exécutable lui-même

Avantage : la simplicité ! Inconvénient : il est impossible de changer de source de données sans recompiler l'exécutable, avec toutes les lourdeurs des mises à jour.

2) Dans un fichier texte en clair

C'est l'approche classique : les chaines de connexion sont enregistrées dans un fichier ini, ou dans le fichier config (appli .net ordinaire)
Avantage : toujours la simplicité ! Inconvénient : les chaines de connexions sont lisibles, et la sécurité n'est donc pas garantie (en particulier si un mot de passe est requis).

3) Dans la base de registre

Avantage : la confidentialité est un peu mieux assurée. Inconvénient : un utilitaire de mise à jour doit avoir accès à la base de registre pour y écrire les nouvelles valeurs quand les sources de données sont modifiées.

4) Dans un fichier crypté

C'est l'approche privilégiée dans les développements StrataFrame, qui assure à la fois la simplicité et la sécurité. Chaque connexion est identifiée par une clé, et l'ensemble des clés est enregistrée dans un fichier xml (un dataset) encrypté 3DES ; chaque paire de clé cryptée et sa chaine de connexion est enregistrée dans un autre fichier xml (un

autre dataset), lui-même crypté 3DES. Les jeux de clés publiques et privées de ces encryptages sont contenues dans les dll de runtime StrataFrame que vous livrez avec votre application.

Concrètement, comment fait-on ?

1) Application Winform

A) Dans l'exécutable

Le modèle est présenté dans la méthode ***GetConnectionStrings***, mais par défaut, il est commenté :

```
'-----------------------------------
'  Setting the data sources manually
'-----------------------------------
'-- SQL Server
'DataLayer.DataSources.Add(New SqlDataSourceItem("", "myconnectionstring"))

'-- Oracle
'DataLayer.DataSources.Add(New OracleDataSourceItem("", "myconnectionstring"))

'-- Microsoft Access
'DataLayer.DataSources.Add(New AccessDataSourceItem("", "myconnectionstring"))

'-- Visual Fox Pro
'DataLayer.DataSources.Add(New VfpDataSourceItem("", "myconnectionstring"))
```

Il suffit de supprimer le commentaire, et de renseigner la chaine de connexion… par exemple :

```
DataLayer.DataSources.Add(New SqlDataSourceItem("", _
                                                "Data Source=ACERVISTALEVY\SQL2008;"+ _
                                                "Initial Catalog=ADVENTUREWORKS2008;"+ _
                                                "Integrated Security=True;"+ _
                                                "Persist Security Info=False;"+ _
                                                "Asynchronous Processing=True "))
```

Ce code suffit si on utilise une seule source de données (un seul database) dans toute l'application. Voyons comment faire pour que notre application consomme aussi bien des données SQL que des dbf ; il suffit tout simplement de renseigner un deuxième item dans la collection.

Par exemple :

```
'-- SQL Server
DataLayer.DataSources.Add(New SqlDataSourceItem("DataSQL", _
                                                "Data Source=ACERVISTALEVY\SQL2008;"+ _
                                                "Initial Catalog=ADVENTUREWORKS2008;"+ _
                                                "Integrated Security=True;"+ _
                                                "Persist Security Info=False;"+ _
                                                "Asynchronous Processing=True "))

'-- Visual Fox Pro
DataLayer.DataSources.Add(New VfpDataSourceItem("DataVFP", _
                                                "Provider=VFPOLEDB;Data Source="+ _
            "C:\Program Files\Microsoft Visual FoxPro 9\Samples\Northwind\northwind.dbc;"))
```

Remarquez bien que nous avons maintenant identifié chaque item par une clé (la 1ère chaine de caractère). Dans le précédent exemple (une seule connexion), cette clé n'était pas requise, une chaine vide suffisait.

B) En utilisant le gestionnaire de connexions (ConnectionManager)

C'est l'approche proposée par défaut dans le AppMain.

La classe ***ConnectionManager*** assure 2 fonctionnalités, dans sa méthode ***SetConnections*** qui est invoquée dans le AppMain : si des informations de connexion sont disponibles dans les fichiers cryptés ***AppKeys.dat*** et ***Connections.dat***, ces informations sont décryptées et ajoutées à la collection DataSources, sinon, un assistant de création de ces fichiers cryptés est présenté, et les fichiers cryptés sont alors enregistrés.

Par défaut, ces deux fichiers cryptés sont enregistrés (et donc recherchés) dans le dossier %ALLUSERSPROFILE%\Application Data\MicroFour\ConnectionData

Si vous laissez ce chemin par défaut, les deux fichiers contiendront toutes les informations de connexion pour toutes les applications que vous développerez. Il est donc judicieux de prévoir un dossier spécifique pour ces fichiers, au moins en mode de débogage. Par exemple :

```
#If DEBUG
    ConnectionManager.ConnectionDataFolder=Application.StartupPath
#End If
```

Personnalisez les informations générales qui seront affichées dans l'assistant :

```
ConnectionManager.ApplicationKey                = "MultiDataSourceExemple"
ConnectionManager.ApplicationDefaultTitle       = "Connexions Multiples"
ConnectionManager.ApplicationDefaultDescription = _
    "Cette Connexion est utilisée pour l'application MultiDataSourceExemple"
```

Préparez le remplissage préalable des informations qui seront requises :

Si vous n'avez qu'une seule source de données, la clé d'identification peut être une chaine vide

```
ConnectionManager.AddRequiredDataSourceItem("", "Connexion SQL Server", _
        DataSourceTypeOptions.SqlServer, _
                "ADVENTUREWORKS2008", _
                "La connexion SQL de MultiDataSourceExemple.")
```

Si vous avez plusieurs sources de données, spécifiez des clés d'identification pour chaque item qui sera ajouté à la collection des DataSources :

```
'      SQL Connection
ConnectionManager.AddRequiredDataSourceItem("DataSQL", "Connexion SQL Server", _
    DataSourceTypeOptions.SqlServer, _
        "ADVENTUREWORKS2008", _
        "La connexion SQL de MultiDataSourceExemple.")

'      FoxPro Connection
ConnectionManager.AddRequiredDataSourceItem("DataVFP", "Connexion VFP", _
    DataSourceTypeOptions.VisualFoxPro, _
        "C:\Program Files\Microsoft Visual FoxPro 9\Samples\Northwind\northwind.dbc", _
        "La connexion DBF de MultiDataSourceExemple.")
```

2) Application Web

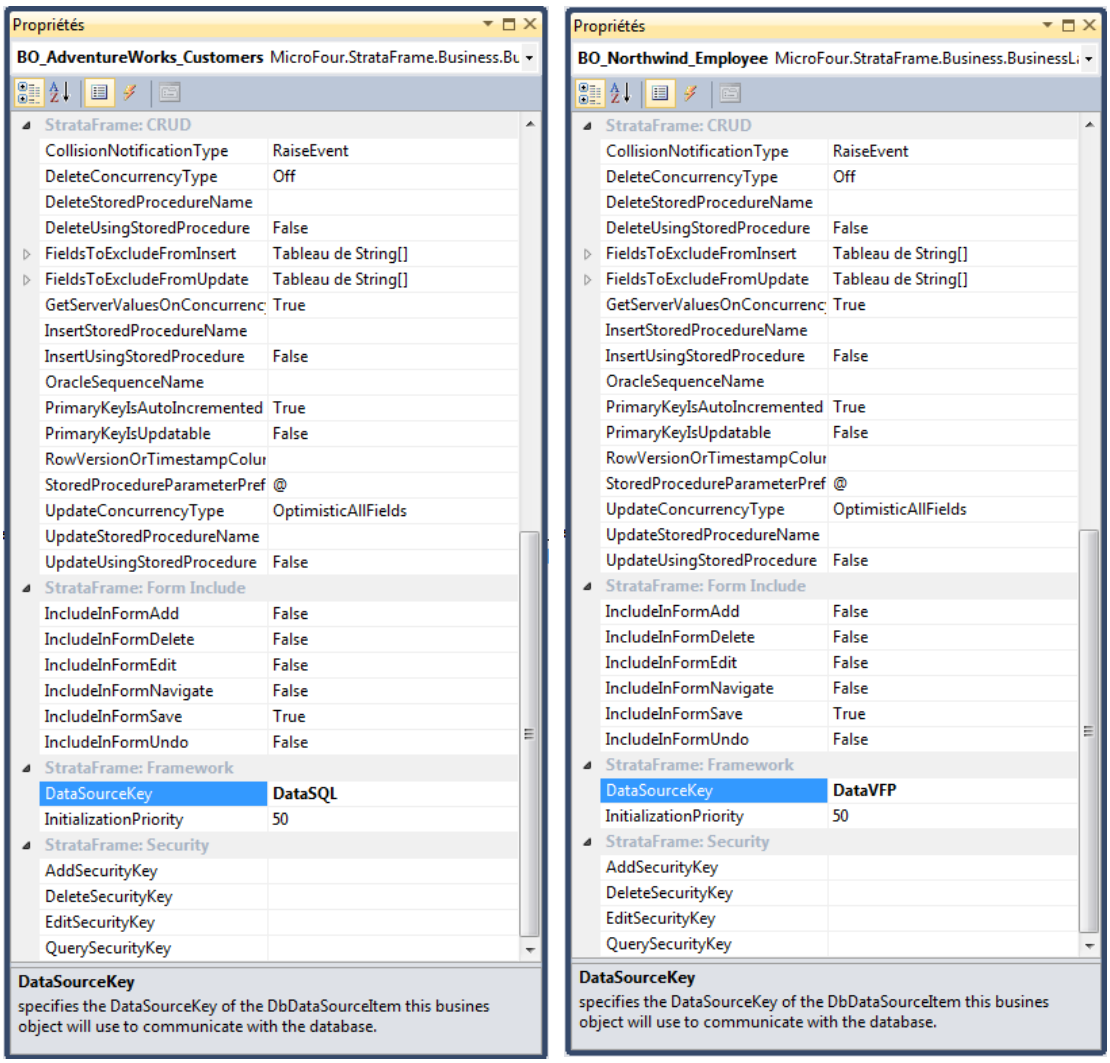
Ici, on peut sans souci de sécurité coder la ou les chaines de connexions dans le fichier ASP.net global.asax.
La syntaxe est exactement la même que dans le AppMain d'une application Winform.

Comment utiliser ces sources de données à travers toute l'application ?

Chaque Business Object présente une propriété ***DataSourceKey***.

Si vous n'avez qu'une seule source de données, nous avons vu que sa clé était par défaut une chaine vide, vous laissez donc cette propriété DataSourceKey avec sa valeur par défaut.

Si vous avez plusieurs sources de données, il suffit de spécifier la clé identifiant la source à utiliser :



Questions fréquentes

L'assistant de connexion va-t-il apparaître à chaque utilisation de mon application ?

Non, il ne sera lancé que si les fichiers AppKeys.dat et Connections.dat ne sont pas dans le dossier précisé.

Est-il possible de retrouver les chaines de connexion ?

Depuis l'application, vous pouvez utiliser la méthode ***GetApplicationActiveConnectionString*** du ConnectionManager pour lire le contenu décrypté de la chaine extraite.

Que faire si je dois modifier les chaines de connexions ?
(déplacement du serveur par exemple)

Si cette modification est occasionnelle, il suffit de recopier sur chaque poste les nouveaux fichiers cryptés.

Si vous devez effectuer fréquemment cette opération sur de nombreux postes, alors il sera probablement plus simple d'utiliser le **Shared Settings File**. C'est un fichier crypté créé en général par un administrateur ou un utilisateur avec pouvoirs, en utilisant un assistant spécifique. Les fichiers locaux contiennent alors un pointeur vers ce fichier centralisé ; il suffit d'actualiser ce dernier pour que les modifications soient prises en compte par chaque poste au prochain lancement de l'application.

Et pour aller plus loin…

Dans le code source de StrataFrame, regardez les classes *DbeConnectionData* et *DbeApplicationData*, ainsi que la classe *ConnexionWizard*.

Figure 1 - Diagramme d'exécution d'une Application StrataFrame

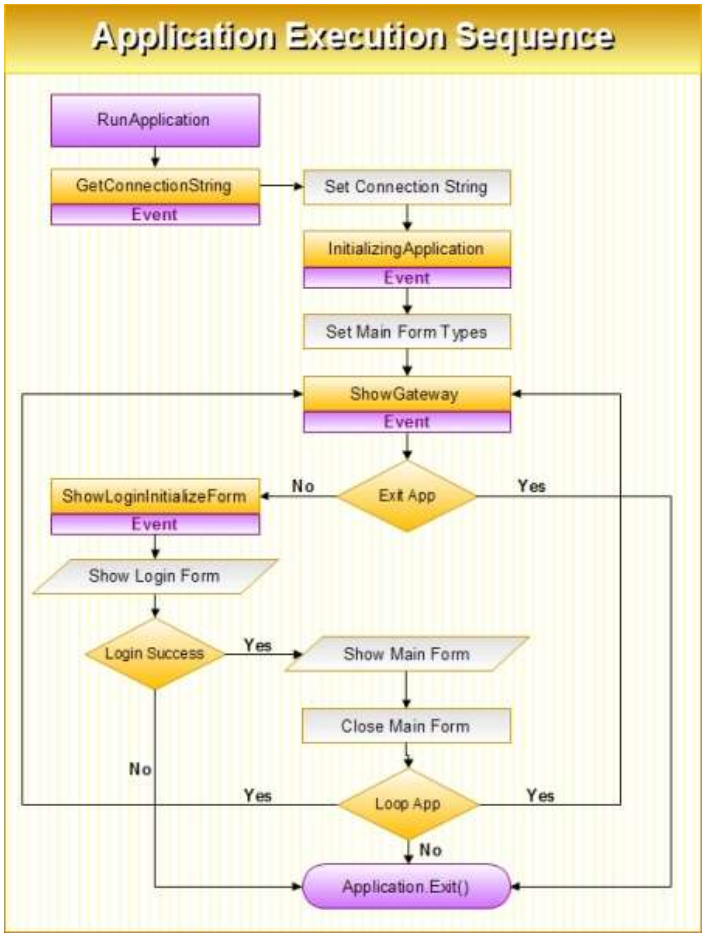


Figure 2 - Les couches d'accès aux données