C# - FICHE PRATIQUE N° 9

Agnès Kintzler - Pierre Loisel

Utilisation d'un dataset commun et définition graphique des liaisons de données.

A/ Principe

ATTENTION : pour utiliser cette technique, il faut disposer du framework DotNet version 1.1, Service Pack 1 minimum (la version du framework est affichée dans l'option « à propos » du menu « ? » de Visual Studio).

Nous avons vu qu'il était préférable d'utiliser un seul dataset pour l'ensemble de l'application. L'inconvénient de ce choix est qu'il n'est plus possible de profiter des possibilités graphiques de Visual Studio pour la définition des dataViews, des dataViewManagers et la mise en place de la liaison de données (dataBinding). Ces éléments doivent donc être définis par programme.

La solution présentée ici a été imaginée par Agnès Kintzler et repose sur le principe suivant :

- Un formulaire « Fm_donnees » contient l'objet connexion, tous les dataAdapter, et le dataset. Le seul rôle joué par ce formulaire est la définition des dataAdapter et du dataset commun. Ces objets doivent posséder le statut « protected » pour être utilisables dans les formulaires dérivés. Le dataset n'est pas rempli à la construction du formulaire.

- Un formulaire « Fm_principal », dérivé de « Fm_données », est notamment chargé de remplir le dataset dont il hérite. Cette opération est effectuée lors de son instanciation. C'est également à cette occasion que les routines de gestion des événements RowChanged et RowDeleted sont mises en place.

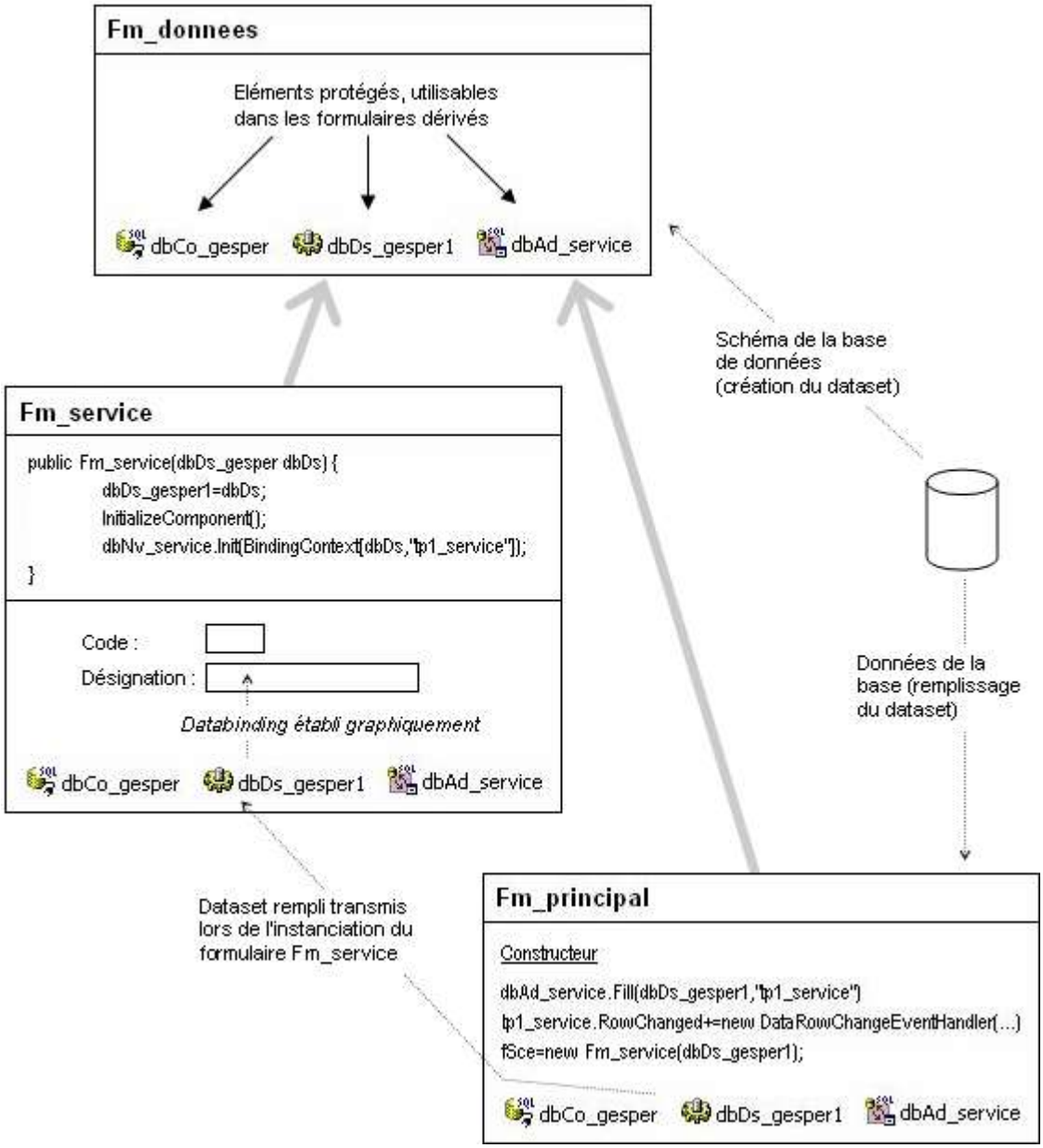
- Les autres formulaires de l'application sont également dérivés de « Fm_donnees » et possèdent donc chacun un dataset. Celui-ci ne sera jamais rempli, mais permettra de configurer graphiquement les contrôles d'accès aux données à l'aide de Visual Studio.

- Pour que ces autres formulaires disposent effectivement des données du dataset rempli par le formulaire principal, il suffit que ce dernier les instancie en leur passant en paramètre ce dataset. Leur constructeur doit donc être paramétré, le paramètre transmis étant utilisé pour renseigner leur propre dataset.

Pas de panique, c'est en fait très simple en ce qui concerne la mise en œuvre…

Ce principe de fonctionnement est illustré par le schéma figurant page suivante.

Remarque : pour profiter des facilités de paramétrage des dataViewManager, il faut ajouter ce composant à la boîte à outils.



B/ Mise en oeuvre

1. Création des formulaires

Créez un nouveau projet, renommez le formulaire créé automatiquement : « Fm_principal ».

Ajoutez un formulaire au projet : « Fm_donnees ». Créez sur ce formulaire le dataAdapter « dbAd_service ». Renommez la connexion « dbCo_gesper » et générez le groupe de données « dbDs_gesper ».

Sélectionnez les trois composants (la connexion, le dataAdapter et le dataset) et faites-en des membres protégés (propriété « Modifiers », valeur « protected »).

Modifiez le code du formulaire principal pour le faire hériter du formulaire « Fm_donnees » :

```
public class Fm_principal : Fm_donnees
{
        ...
}
```

Générez la solution, puis créez un formulaire hérité « Fm_service » (option Ajouter un formulaire hérité du menu Projet). Ce formulaire doit être dérivé de « Fm_donnees ». (Attention, générez la solution avant de créer le formulaire).

Modifiez le constructeur de ce formulaire pour mettre en place la transmission du dataset :

```
public Fm_service(dbDs_gesper dbDs)
{
        dbDs_gesper1=dbDs;
        InitializeComponent();
}
```

2. Formulaire principal

Ajoutez le code nécessaire au remplissage du dataset dans le constructeur de « Fm_principal ».

Placez-y également la mise en place de la gestion des évènements « RowChanged » et « RowDeleted ».

Ajoutez un membre privé « fSce » représentant le formulaire service. Instanciez ce formulaire dans le constructeur de « Fm_principal ».

Créez un bouton « bt_service » permettant l'ouverture du formulaire service.

On obtient :

```
private Fm_service fSce;

public Fm_principal()
{
        InitializeComponent();
        dbAd_service.Fill(dbDs_gesper1,"tp1_service");
        dbDs_gesper1.tp1_service.RowChanged+=new DataRowChangeEventHandler(tp1_service_RowChanged);
        dbDs_gesper1.tp1_service.RowDeleted+=new DataRowChangeEventHandler(tp1_service_RowDeleted);
        fSce=new Fm_service(dbDs_gesper1);
}

private void tp1_service_RowChanged(object sender, DataRowChangeEventArgs e)
{
        Outils.DbNavigateur.GererRowAction(e,dbAd_service);
}

private void tp1_service_RowDeleted(object sender, DataRowChangeEventArgs e)
{
        Outils.DbNavigateur.GererRowAction(e,dbAd_service);
}

private void bt_service_Click(object sender, System.EventArgs e)
{
        fSce.Show();
}
```

3. Formulaire service

Supprimez le menu système du formulaire « Fm_service » (propriété « ControlBox »), et ajoutez-y un bouton « bt_fermer » permettant de le cacher.

Ajoutez un DbNavigateur « dbNv_service » et initialisez-le dans le constructeur du formulaire.

Remarque : si ce composant n'est pas disponible, il faut l'ajouter à la boîte à outils (Menu outils, Ajouter des éléments…, parcourir et choisir le fichier « outils.dll »).

On obtient :

```
public Fm_service(dbDs_gesper dbDs)
{
        dbDs_gesper1=dbDs;
        InitializeComponent();
        dbNv_service.Init(BindingContext[dbDs_gesper1,"tp1_service"]);
}

private void bt_fermer_Click(object sender, System.EventArgs e)
{
        Hide();
}
```

Ajoutez des contrôles de type « TextBox » destinés à afficher les données de la table (« tb_code » et « tb_designation ».

Etablissez graphiquement la liaison de données pour ces contrôles en paramétrant leur propriété « DataBindings », item « Text ».

C'est terminé, vous pouvez tester votre application.

4. Compléments

Ajoutez un dataview sur le formulaire service et nommez-le « dbDv_service ». Associez-le graphiquement à la table « tp1_service » et indiquez « designation » pour la propriété « Sort ».

Modifiez graphiquement le DataBinding des deux TextBox pour qu'ils soient liés à ce dataView. Modifiez également l'initialisation du DbNavigateur.

Lancez l'exécution : les services sont triés par désignation.

De fait, toutes les liaisons aux données peuvent être réalisées graphiquement. En terme de productivité, le gain est considérable…

L'application fournie (fiche9) illustre l'ensemble de ces possibilités.