C# - FICHE PRATIQUE N° 4 – Solutions et compléments

Une petite gestion de personnel
Procédures stockées, liaisons de données plus élaborées.

A/ Plusieurs formulaires

Structure d'une solution

A l'occasion de cette modification de l'application, il peut être intéressant de montrer aux étudiants les différentes composantes d'une application de la manière suivante :

- Faire un copier/coller du dossier « fiche3 » et le renommer « fiche4 ».
- Se fixer comme but de renommer cette application, du fichier solution (.sln) à l'exécutable.

Cela est possible simplement à l'aide de l'explorateur de solution :

- Renommer la solution et le projet dans l'explorateur de solution.
- Faire un clic droit sur le projet pour atteindre ses propriétés.
- Modifier le nom de l'assembly et l'espace de noms par défaut (utilisé pour les nouvelles classes).
- Utiliser l'option « Remplacer dans les fichiers » du menu « Edition » pour remplacer « Fiche3 » par « Fiche4 ».
- Sauvegarder et quitter sans générer l'application.
- Ouvrir la nouvelle solution « Fiche4.sln » et régénérer la solution.

On peut profiter de cette occasion pour :

- Faire le tour des propriétés d'un projet.
- Idem pour une solution (clic droit dans l'explorateur de solution), notamment lorsqu'elle contient plusieurs projets.
- Examiner les fichiers générés, notamment les « exécutables » qui se trouvent dans le répertoire « bin », sous-répertoire « debug » ou « release » en fonction du type de génération choisi.

Ouverture et fermeture de formulaire

Remarque : la méthode « ShowDialog », utilisée en lieu et place de « Show » permet d'obtenir une fenêtre modale.

Le problème qui se pose est le suivant :

- Le formulaire de type Fm_service (disont « fsce ») est instancié lors de la création du formulaire principal.
- Lorsque l'utilisateur clique sur le bouton « services », le formulaire « fsce » est montré.
- Lorsque l'utilisateur ferme ce formulaire, l'objet « fsce » est tout simplement détruit.
- Un nouveau clic sur le bouton « services » se termine mal…

Deux solutions peuvent être envisagées :

1. La plus simple

- Faire passer la propriété ControlBox de Fm_service à false (l'utilisateur peut plus le fermer).

- Y ajouter un bouton « fermer » et faire appel à la méthode « Hide » de la classe Form.

2. Une autre qui peut parfois servir

Lorsqu'un formulaire est fermé (par l'utilisateur ou par sa méthode « Close »), un événement « Closing » est généré. Cet événement fait partie des événements qui peuvent être annulés. On les reconnaît à leur second argument qui est de type « CancelEventArgs ». Il suffit d'écrire le gestionnaire ci-dessous :

```
private void Fm_service_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
    e.Cancel=true;
    Hide();
}
```

Formulaire employés

> L'examen du code généré par le concepteur permet de montrer aux étudiants qu'il n'y a rien de mystérieux dans cette affaire…

> Dans le cas où l'on souhaite faire générer la clé par le SGBD de manière automatique lors d'une insertion, il est en fait possible de récupérer la valeur générée. Le mécanisme est le suivant :

- L'application envoie un n-uplet sans clé au SGBD.
- Celui-ci génère la clé.
- L'application émet une requête pour récupérer le n-uplet inséré (et donc la clé générée).

Lorsque l'on configure un Data Adapter, la commande InsertCommand correspond par défaut à ce mode de fonctionnement. Par exemple, pour le Data Adapter « dbAd_service » on trouve :

```
INSERT INTO tp1_service(code, designation) VALUES (@code, @designation);
SELECT code, designation FROM tp1_service WHERE (code = @code)
```

La première requête insère le n-uplet, la seconde le recharge depuis la base de données dans le dataset. Le principe est le même lors d'une mise à jour.

Pour éviter la génération de cette seconde requête, il suffit de reconfigurer le Data Adapter :

- Faire un clic droit sur le Data Adapter dans la fenêtre de conception.
- Dans le dialogue « Générer les instructions SQL », cliquer sur « Options avancées ».
- Décocher « Actualiser le Dataset ».
- Ceci peut être fait dès la création du Data Adapter.
- On obtient exactement le même résultat en modifiant manuellement le texte des requêtes InsertCommand et UpdateCommand.

D'une manière générale, il est inutile de générer ces instructions « select ». Dans le cas présent, il est même indispensable de ne pas les générer. En effet, lors de l'exécution de la méthode de gestion de l 'événement « RowChanged », on obtiendrait une erreur du type « Un DataReader est déjà ouvert pour cette connexion… ». La connexion à SQL Server semble occupée à autre chose. En fait, c'est le rafraîchissement automatique du DataSet qui entre en conflit avec la modification de la ligne.

Remarque : on pourrait désactiver la gestion de l'événement « RowChanged » pendant le traitement de l'ajout pour éviter un appel lors de la modification de « e.Row ».

> En ce qui concerne la procédure stockée, rien de spécial à signaler. En utilisant « using Navigation », on peut écrire :

```
private void Fm_employe_Load(object sender, System.EventArgs e)
{
    dbNv_employe.Init(    dbAd_employe,BindingContext[dbDs_employe1,"tp1_employe"],
                          new OnDbEventFunction(surAjout));
}
```

Remarque : il existe d'autres solutions, notamment faire appel à une procédure stockée du type « insert » en lieu et place de l'instruction insert de la propriété « InsertCommand » du DataAdapter. Cette procédure stockée peut alors retourner un paramètre résultat contenant l'identifiant généré.

B/ Amélioration de l'interface

Il est nécessaire de donner une valeur par défaut pour l'attribut « cadre » car une case à cocher ne peut pas rendre compte de la valeur nulle. A ce stade, le faire dans SQL Server est inopérant. En effet, l'insertion se fait dans le DataSet et non directement dans la base de données.

Les choses sont ainsi même si la valeur par défaut a été fixée avant la génération du DataSet (ce qui ne semble pas normal…). Pour s'en convaincre, il suffit de le régénérer.

- Supprimer l'objet « dbDs_employe1 » dans la fenêtre de conception.
- La classe « dbDs_employe » (dérivée de DataSet) doit également être supprimé, ceci se fait par l'intermédiaire de l'explorateur de solution.
- Générer l'application : il y a des erreurs évidemment… (cette génération est pourtant obligatoire).
- Choisir « générer le groupe de données » (menu « données » ou clic droit dans la fenêtre de conception).
- Examiner le schéma du DataSet : le champ « cadre » n'a toujours pas de valeur par défaut, l'erreur subsiste à l'exécution.
- Remarque : il faut recommencer toutes les liaisons de données, ce qui s'avère vite fastidieux…

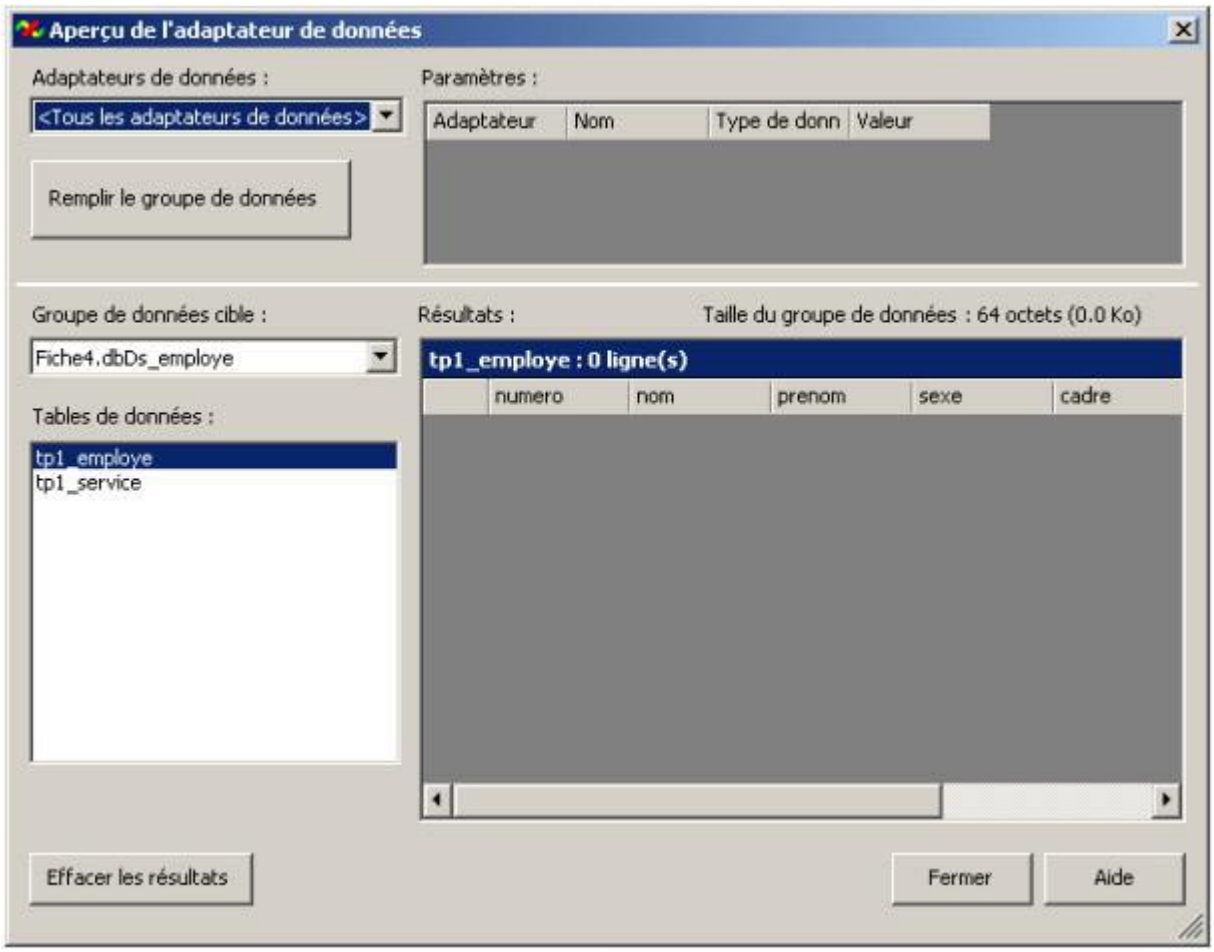
Au moins, cela a permis de voir comment générer à nouveau un DataSet !

En ce qui concerne le sexe, l'utilisation d'une valeur par défaut n'est pas obligatoire. On pourrait très bien ne cocher aucun des deux boutons dans le cas d'une valeur null. Pour affecter explicitement la valeur nulle au champ de la table du DataSet, il faut écrire :

```
dbNv_employe.Courant()["sexe"]=DBNull.Value;
```

C/ Une liste déroulante pour les services

La fenêtre ci-dessous (option « Aperçu des données » du menu « Données ») permet de visualiser les données d'un DataSet.



- La liste déroulante « Adaptateurs de données » permet de sélectionner le (ou les) DataAdapter(s) souhaité(s).
- Un clic sur le bouton « Remplir le groupe de données » déclenche la même opération que la méthode « Fill » de la classe DataAdapter.
- On peut ainsi consulter le contenu du DataSet, et voir l'espace qu'il occupe localement.
- Remarque : après l'ajout d'une relation entre les deux tables (cf fiche5), le remplissage de la table employé seule indiquera une erreur sur les contraintes. En effet, les données de la table service seront nécessaires à la vérification de l'intégrité de référence.

La mise à jour des données de la table service dans le formulaire employé doit effectuer deux opérations :

- Appeler la méthode « Clear » du DataAdapter.
- Le remplir à nouveau.

En effet, un simple appel à la méthode « Fill » reflèterait les ajouts et modifications effectués, mais ne prendrait pas en compte les suppressions. D'autre part, il semble que l'exécution successive de deux « Fill » (sans « Clear ») avec un DataSet très volumineux pose des problèmes.