C# - FICHE PRATIQUE N° 5 – Pour aller plus loin

Une petite gestion de personnel
modification de la source de données.

Nous avons vu que le mécanisme d'accès aux données rend l'application relativement indépendante de la source utilisée (base de données et/ou SGDB). Pour le constater de manière pratique, il est possible de transformer l'application pour qu'elle utilise une base de données Access 2000 (ou version supérieure).

1. Création de la base de données

Copier le répertoire de l'application « Fiche5 », renommer cette copie « Fiche5Access » et créer une base de données Access « FichesCSharp.mdb » :

- Table tp1_service : code texte(2), designation texte(30). Clé code.
- Table tp1_employe : numero(numéroAuto), nom texte(30), prenom texte(25), sexe texte(1), cadre (oui/non), salaire (monétaire), sce texte(2). Clé numero.
- Une relation entre tp1_service (code) et tp1_employe (sce).

La seule différence tient au type du champ « numero » de la table employé.

2. Modification de l'application en mode conception

Seul le formulaire principal est concerné puisque c'est lui qui fournit l'accès aux données aux autres éléments de l'application.

- Supprimer les deux DataAdapters, l'objet connexion dbCo_gesper, l'objet SqlCommand dbPs_idEmploye et le DataSet dbDs_empSce1.
- Supprimer le type dbDs_empSce depuis l'explorateur de solution.
- Ajouter un élément « OleDbDataAdapter »
- Choisir une nouvelle connexion pour cet objet : fournisseur « Microsoft Jet 4.0 » connectée à la base de données « FichesCSharp.mdb ».
- Paramétrer ce DataAdapter pour qu'il retourne tous les champs de la table service et valider (ne pas mettre de mot de passe).
- Renommer les objets créés : DataAdapater « dbAd_service », connexion « dbCo_gesper ».
- Faire de même pour la table employé : DataAdapter « dbAd_employe » (réutiliser la connexion qui vient d'être créée).
- Générer le groupe de données et nommez-le dbDs_empSce : un objet DataSet dbDs_empSce1 est créé automatiquement.
- Afficher le schéma de ce DataSet, donner une valeur par défaut aux champs « cadre » et « sexe ».
- Créer une relation « serviceemploye » entre les deux tables, indiquer « none, none et cascade » pour les règles associées à cette relation.
- Créer un objet OleDbCommand nommé « dbPs_idEmploye », connexion « dbCo_gesper », texte « select @@identity ».

3. Modification du code

Là encore, seul le code du formulaire principal est concerné. En fait, le seul point à modifier concerne l'obtention d'une valeur pour la clé de la table employé. Access ne permet pas d'écrire une procédure stockée retournant une valeur, la technique est donc différente.

Depuis la version Access 2000, il est possible de récupérer un numéro automatique qui vient d'être généré à l'aide de la requête « select @@identity » appliquée à la table concernée via son dataAdapter. Ce numéro doit être récupéré après l'insertion, lors de l'événement « RowUpdatedEventHandler » du DataAdapter.

- Inclure l'espace de noms dédiée à OleDb.

```
using System.Data.OleDb;
```

- Ajouter la gestion de l'événement dans le constructeur du formulaire et programmer la méthode correspondante.

```
public Fm_principal()
{
    dbAd_employe.RowUpdated+=new OleDbRowUpdatedEventHandler(dbAd_employe_RowUpdated);
}

private void dbAd_employe_RowUpdated(object sender, OleDbRowUpdatedEventArgs e)
{
    if (e.StatementType==StatementType.Insert)
    {
        int id;
        id=int.Parse(dbPs_idEmploye.ExecuteScalar().ToString());
        e.Row["numero"]=id;
    }
}
```

- Supprimer le code destiné à gérer l'ajout sur l'événement « RowChanged » pour la table employé, la procédure de gestion devient simplement :

```
private void tp1_employe_RowChanged(object sender,DataRowChangeEventArgs e)
{
    DbNavigateur.GererRowAction(e,dbAd_employe);
}
```

C'est fini, il ne reste plus qu'à tester le tout !