C# - FICHE PRATIQUE N° 4

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

A/ Plusieurs formulaires

1. Formulaire principal

En fait, le formulaire Fm_service n'est pas le formulaire principal de notre application. Il ne sert qu'à la gestion des services.

Choisissez « Ajouter un nouvel élément » dans le menu fichier. Ajoutez un élément « windows form » et nommez-le « Fm_principal». Faites en sorte que l'application affiche en premier le formulaire Fm_principal en déplaçant la méthode Main (et l'attribut [STAThread] qui la précède) de la classe Fm_service à la classe Fm_principal (attention, il y a une petite modification à opérer…).

Munissez le formulaire principal d'un bouton "Services" (bt_service) permettant l'ouverture du formulaire Fm_service. Comment faire ?

Une première approche consiste à déclarer un objet Fm_service, l'instancier et appeler sa méthode Show, tout cela dans la méthode bt_serviceClick. L'inconvénient est qu'il y a création d'un nouvel objet à chaque ouverture (avec tout ce que cela implique (connexion à la base, création et remplissage du dataset…).

Une meilleure approche consiste à déclarer un membre donnée de type Fm_service dans la classe Fm_principal, de l'instancier dans le constructeur de Fm_principal (ou plus simplement de l'instancier lors de sa déclaration) et de simplement l'afficher sur le clic du bouton.

Essayez les deux…

Un problème se pose avec la seconde solution. A vous de le trouver et de le résoudre… Un conseil, lisez bien le message d'erreur ! Une aide : la propriété ControlBox du formulaire est intéressante.

Remarque : la méthode « BringToFront » permet d'amener le formulaire au premier plan après son affichage.

2. Formulaire Employés

Ajoutez quelques employés dans votre base de données via l'analyseur de requêtes SQL Server (n'indiquez aucune valeur pour le numéro d'employé, un identifiant sera généré par SQL Server).

Créez un nouveau formulaire Fm_employe et mettez en place le mécanisme permettant son ouverture à partir d'un bouton dans le formulaire principal.

Créez un nouveau SlqDataAdapter pour gérer la table employé. Vous pouvez bien sûr réutiliser la définition de la connexion créée pour l'accès à la table service.

ATTENTION : lors du paramétrage de ce DataAdapter, décochez l'option « Actualiser le dataset » après avoir cliqué sur le bouton « Options avancées ».

Renommez les objets créés sur le formulaire.

Choisissez « générer le groupe de données » dans le menu « données ». Créez un nouveau dataSet « dbDs_employe ». N'oubliez pas de remplir ce dataset à l'aide de la méthode « fill » du dataAdapter dans le constructeur du formulaire et de mettre en place la validation automatique des modifications à l'aide des événements « RowChanged » et « RowDeleted » (utilisez pour cela la méthode « GererRowAction » de la classe « DbNavigateur »).

Créez un label et un textbox pour chacun des attributs de la table employé, mettez en place la liaison de données et ajoutez un dbNavigateur. Testez le tout… mais sans ajouter d'enregistrements…

3. Ajouter un employé

Le problème de l'ajout dans une table possédant une clé générée par le sgbd est classique.

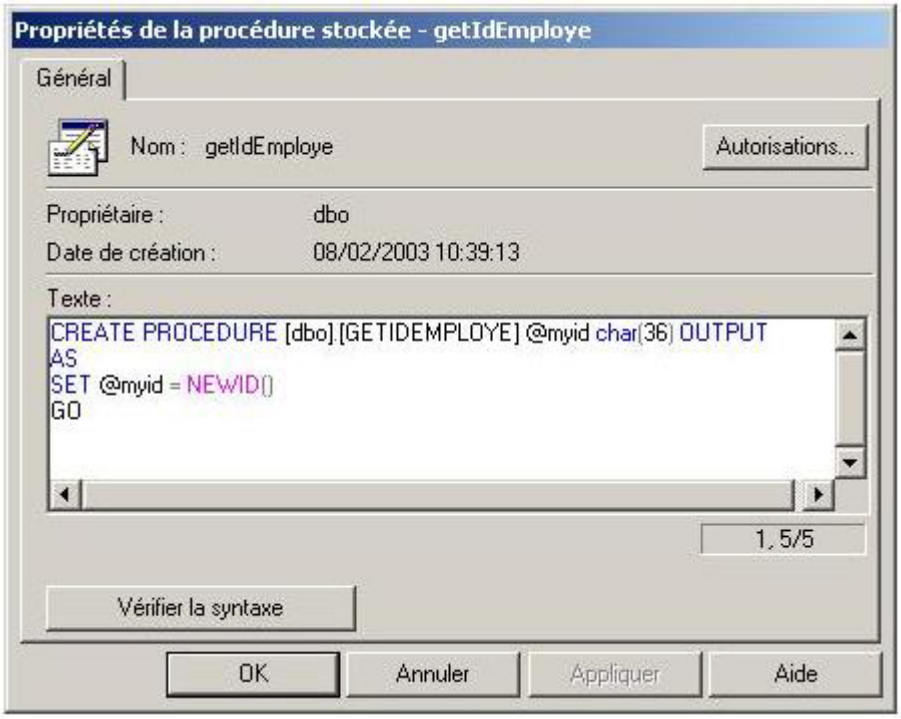
Une première solution est de ne gérer que les autres attributs dans l'application, le sgbd se chargeant de créer la clé primaire. Cependant, cela a deux conséquences gênantes : d'une part l'utilisateur ne peut pas connaître la clé générée, d'autre part il est impossible de la manipuler dans l'application (contrôles, renseignement d'une clé étrangère, …).

Une meilleure solution fonctionne selon le principe suivant :

- Lorsqu'elle veut insérer un enregistrement, l'application demande une nouvelle valeur de clé au sgbd. Ceci peut être réalisé à l'aide d'une procédure stockée
- L'application possédant une valeur pour la clé, elle peut insérer l'enregistrement et montrer cette valeur à l'utilisateur ou s'en servir pour des contrôles…

Mise en œuvre :

Créez une procédure stockée SqlServer à partir de « Sql Server enterprise manager ».



Pour pouvoir l'exécuter :

- créez un objet « SqlCommand » nommé « dbPs_idEmploye »
- paramétrez sa propriété « connection » : « dbCo_gesper »
- propriété « commandType » : « StoredProcedure »
- propriété « Text » : « getIdEmploye » (le nom de la procédure stockée à exécuter)
- contrôlez la propriété « Parameters », vous devez retrouver votre paramètre « @myid ».

Il faut maintenant appeler cette procédure stockée lors de l'ajout d'un enregistrement et renseigner la colonne « numero » de la table du DataSet local. Cela peut être fait lors de la gestion de l'événement « RowChanged ».

- Modifiez la méthode de gestion de cet événement :

```
private void tp1_employe_RowChanged(object sender, DataRowChangeEventArgs e)
{
    if (e.Action==DataRowAction.Add)
    {
        dbCo_gesper.Open();
        dbPs_idEmploye.ExecuteNonQuery();
        dbCo_gesper.Close();
        e.Row["numero"]=dbPs_idEmploye.Parameters["@myid"].Value.ToString();
    }
    DbNavigateur.GererRowAction(e,dbAd_employe);
}
```

- Testez l'ajout d'un employé, cela ne se passe pas correctement. En fait, l'insertion dans la table locale pose problème car la colonne « numero » possède à ce moment la valeur nulle, ce qui n'est pas autorisé car cette colonne est clé primaire de la table locale.

- Pour résoudre le problème, il suffit de donner à cette colonne une valeur par défaut (« auto » par exemple). Cette valeur sera utilisée lors de l'insertion dans la table locale, puis immédiatement remplacée par l'identifiant généré par SQL Server lors de l'exécution de la méthode « tp1_employe_RowChanged ».

Pour cela :

- Faites un clic droit sur le DataSet « dbDs_employe1 » dans la fenêtre de conception.
- Choisissez l'option « afficher le schéma ».
- Sélectionnez la colonne « numero » de la table « employe ».
- Affichez la fenêtre de propriétés et indiquée la valeur par défaut de la colonne.

Dernier détail, l'utilisateur ne doit pas pouvoir modifier la valeur générée : la propriété enabled du textBox doit être mise à la valeur « false ».

4. Exécution d'une méthode lors de l'ajout d'un employé

Il est souhaitable de sélectionner le contrôle tb_nom lors de l'ajout d'un employé. Ceci peut fait à l'aide de la méthode ci-dessous :

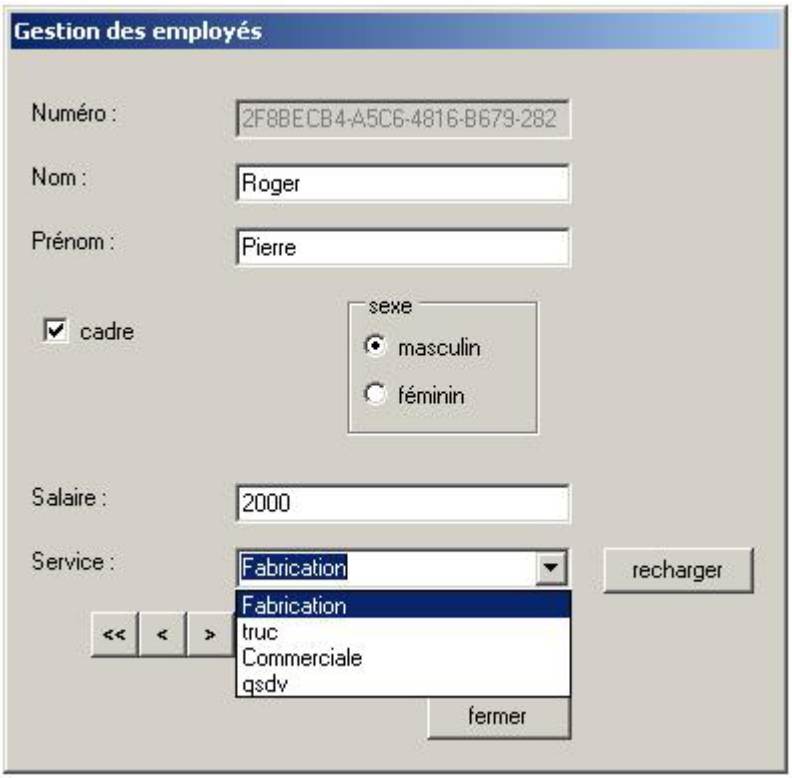
```
private void surAjout()
{
    tb_nom.Focus();
}
```

Il reste à appeler cette procédure lors de l'ajout d'un employé. Le composant dbNavigateur a été prévu pour cela. Il est possible d'ajouter un 3ème paramètre lors de son initialisation. Ce paramètre permet d'indiquer le nom d'une méthode à exécuter lors de l'insertion d'un enregistrement.

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

B/ Amélioration de l'interface

Nous allons progressivement arriver à une interface plus conviviale :



1. Case à cocher et boutons d'option

- Affichez le statut cadre ou non d'un employé à l'aide d'une case à cocher (la propriété à lier aux données est « Checked »). Le contrôle tb_cadre peut être supprimé. Un petit problème subsiste lors d'un ajout d'enregistrement… La solution est de donner une valeur par défaut à la propriété « cadre » (true ou false). Pour cela, faites un clic droit sur le dataSet et choisissez l'option « afficher le schéma ».
- Remarque : si vous avez saisi des employés avec une valeur nulle pour le champ « cadre », le formulaire refusera de s'ouvrir… Modifiez les à l'aide du SGBD.

- Gestion des boutons d'options :

- Un GroupBox (gb_sexe)
- Deux RadioButton (rb_feminin et rb_masculin)
- Les deux boutons radio s'excluent mutuellement car ils sont placés dans un même GroupBox

Il s'agit de synchroniser la valeur de la propriété « sexe » et cet ensemble de boutons d'options. La solution est la suivante :

. La méthode Init de DbNavigateur possède un autre paramètre qui permet d'indiquer une fonction à appeler lorsque l'on change d'enregistrement courant :

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

La méthode surDeplact doit activer le bon bouton radio (propriété Checked) en fonction du sexe de l'employé courant. Cet enregistrement courant est fourni par la méthode Courant de la classe dbNavigateur. On peut donc écrire la méthode surDeplact (classe Fm_employe) de la manière suivante :

```
private void surDeplact()
{
    if (dbNv_employe.Position!=-1)   // la position est-elle valide ?
    {
        if (dbNv_employe.Courant()["sexe"].ToString()=="f")
            rb_feminin.Checked=true;
        else
            rb_masculin.Checked=true;
    }
}
```

. Il faut maintenant mettre à jour le dataset lorsque l'utilisateur clique sur un des boutons d'option. Pour cela, sur l'événement « Click » de chaque bouton radio, il faut appeler une même procédure événementielle qui modifie la valeur de l'attribut sexe de l'employé courant.

```
private void gb_sexe_Changed(object sender, System.EventArgs e)
{
    if (rb_feminin.Checked)
        dbNv_employe.Courant()["sexe"]="f";
    else
        dbNv_employe.Courant()["sexe"]="m";
}
```

. Enfin, le mieux est de donner une valeur par défaut à cet attribut, modifiez le schéma du DataSet en conséquence (et supprimez les éventuelles valeurs nulles à l'aide de SQL Server).

C/ Une liste déroulante pour les services

1. Ajout de la table service dans le dataSet

Le principe est de remplir une liste déroulante avec le contenu de la table service. La valeur de cette liste sera liée au champ « sce » de la table employé.

Pour pouvoir remplir la liste, il faut un accès à la table service. Nous allons l'ajouter dans le dataset « dbDs_employe ».

- Créez un datatAdpater « dbAd_service » pour la table service en utilisant la même connexion que pour la table employé et en ne générant pas les instructions delete, update et insert (bouton « options avancées »). En effet, cet accès à la table service se limitera au chargement de la liste. Demandez à générer le groupe de données (dataset) pour cet adaptateur (clic droit). Ajoutez la table service au dataset existant « dbDs_employe ».

- Vérifiez l'opération en regardant les propriétés du groupe de données (clic droit sur db_Ds_employe1). Vous pouvez également avoir un aperçu des données par l'intermédiaire du menu « données ».

- Il reste à ajouter le chargement des données à l'aide de l'adaptateur dans le constructeur du formulaire :

```
public Fm_employe()
{
    InitializeComponent();
    dbDs_employe1.Clear();
    dbAd_employe.Fill(dbDs_employe1,"tp1_employe");
    dbAd_service.Fill(dbDs_employe1,"tp1_service");
    dbDs_employe1.tp1_employe.RowChanged+=new DataRowChangeEventHandler(tp1_employe_RowChanged);
    dbDs_employe1.tp1_employe.RowDeleted+=new DataRowChangeEventHandler(tp1_employe_RowDeleted);
}
```

Remarque : il est plus clair d'indiquer le nom de la table à remplir pour chacun des dataAdapter puisque le DataSet en comporte plusieurs.

2. Création de la liste déroulante

- - Créez un nouveau contrôle « comboBox » : cb_service.
- - Pour le remplir avec la table service, il suffit de paramétrer les propriétés « datasource », « displayMember » et « valueMember ». A vous de voir comment les paramétrer…
- - Pour lier la comboBox avec le champ sce de la table employé, il suffit d'effectuer une liaison de données (propriété dataBindings). Mais attention, c'est la propriété « SelectedValue » et non la propriété « Text » qui doit être liée au champ « tp1_employe.sce ».
- - Il reste à s'assurer qu'aucun service n'est sélectionné lors de l'ajout d'un employé (le champ « sce » a par défaut une valeur null). Il suffit d'ajouter la ligne « cb_service.SelectedIndex=-1 ; » à la méthode « surAjout » du formulaire.

3. Mise à jour de la table service

Un problème subsiste : si la table service est mise à jour par l'intermédiaire du formulaire Fm_service, le dataset utilisé dans le formulaire Fm_employe n'est pas actualisé. Une solution simple consiste à permettre à l'utilisateur de recharger la table service depuis le formulaire Fm_employe. Ajoutez pour cela un bouton « recharger » permettant d'exécuter le code ci-dessous :

```
private void bt_recharger_Click(object sender, System.EventArgs e)
{
    dbDs_employe1.service.Clear();
    dbAd_service.Fill(dbDs_employe1,"tp1_service");
}
```

Notez bien que la solution consiste à maintenir deux copies de la table service dans l'application… Nous reviendrons sur ce problème.