C# - FICHE PRATIQUE N° 6

Une petite gestion de personnel
Tri, recherche, calcul d'expressions, rafraîchissement des données.

A/ Tri et filtrage d'enregistrements

1. Principe

. Les enregistrements contenus dans un dataSet peuvent être triés et/ou filtrés. Pour cela, une première solution consiste à exprimer une condition de sélection et/ou une clause order by dans la requête utilisée pour définir le dataAdapter. Mais cette solution conduit à recharger les enregistrements à chaque modification de ces paramètres…

Le mieux est d'utiliser un objet « DataView » qui définit une manière d'accéder aux enregistrements d'une table d'un dataSet. Cette classe fait partie de l'espace de noms « System.Data ».

Voici l'exemple du formulaire Fm_service :

```
DataView dbDv;
public Fm_service(dbDs_empSce p_dbDs):this()
{
    dbDs=p_dbDs;
    dbDv=new DataView(dbDs.tp1_service);
    dbDv.Sort="designation";
    dbNv_service.Init(BindingContext[dbDv]);
    tb_code.DataBindings.Add("Text",dbDv,"code");
    tb_designation.DataBindings.Add("Text",dbDv,"designation");
    tb_masseSalariale.DataBindings.Add("Text", dbDv, "masseSalariale");
}
```

Après création du DataView « dbDv » et définition du critère de tri, celui-ci est utilisé en lieu et place du dataTable pour l'initialisation du navigateur et la mise en place des liaisons de données.

. Le tri peut s'effectuer sur plusieurs colonnes en les séparant par des virgules. D'autre part, il est possible de trier de manière décroissante en utilisant le mot-clé DESC après le nom d'une colonne.

. Le critère de tri peut même être défini dynamiquement. Créez pour le vérifier un bouton permettant d'alterner un tri par désignation croissante et un tri par code décroissant. Dans certains cas, la liaison de données doit être interrompue pour modifier le tri ou le filtrage (une erreur d'exécution signale le problème). La méthode à appliquer est la suivante :

```
this.BindingContext[dbDv].SuspendBinding();
// modification des critères de tri et de filtrage
…
this.BindingContext[dbDv].ResumeBinding();
```

. Le filtrage des données se paramètre à l'aide la propriété « RowFilter » du DataView. Par exemple, pour ne retenir que les services dont le code est supérieur à s3, on indiquera :
dbDv.RowFilter="code>'s3'";

2. A vous de jouer

. Modifiez le formulaire employé pour qu'il utilise également un DataView trié par ordre de numéro croissant.

Remarque : il est préférable de toujours fixer un ordre pour un DataView. Une autre possibilité est de demander l'application d'un ordre par défaut (DbDv.ApplyDefautSort=true), cet ordre étant alors celui de la clé primaire (ou à défaut l'ordre de création des enregistrements). Les fonctions de recherche étudiées ci-après nécessitent le tri du DataView (éventuellement par défaut).

. En ce qui concerne la liste des services, elle peut bien entendu tirer ses données de la table service du DataSet, mais il y a mieux : elle peut être alimentée par un second DataView trié en ordre croissant des désignations.

```
cb_service.DataSource=new DataView(dbDs.tp1_service,null,"designation",
DataViewRowState.CurrentRows);
```

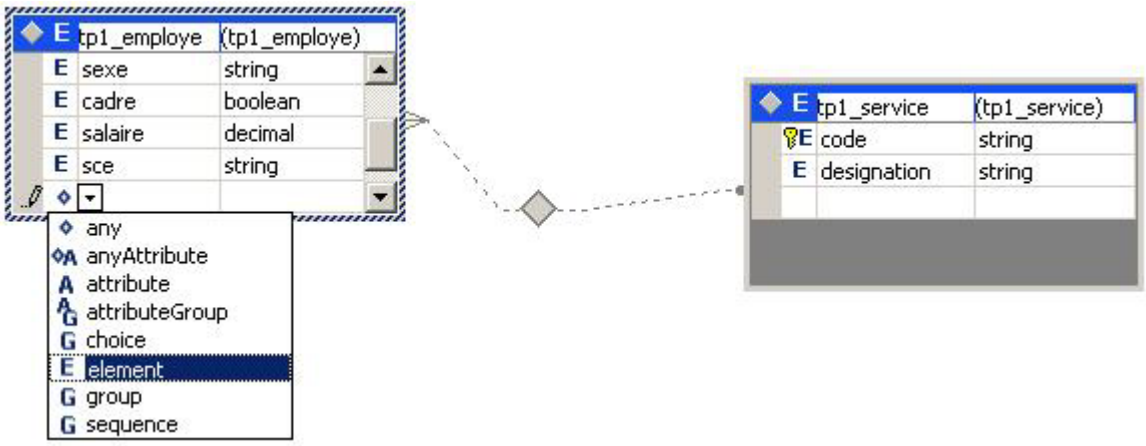
3. Conclusion

La meilleure méthode consiste à utiliser **systématiquement** un DataView pour les formulaires liés à une seule table (initialisation de dbNavigateur et liaison de données), et à utiliser **systématiquement** un DataView pour alimenter les listes déroulantes.

B/ Colonnes calculées et recherche d'enregistrements

1. Création d'un champ calculé

. Ouvrez le schéma du dataSet et créez un nouvel élément pour la table employé :



- Nom : recherche
- Type : string
- Expression : isnull(nom+' '+prenom,nom)

Ce champ sera calculé pour chaque enregistrement au fur et à mesure des besoins (sa valeur n'est pas mémorisée). La formule de calcul permet de ne prendre que le nom si le prénom à la valeur null (dans ce cas, la concaténation elle-même est nulle).

Un problème peut apparaître lors de l'ajout ou de la modification d'un employé si vous avez demandé l'actualisation du DataSet lors de la configuration du DataAdapter (bouton « options avancées »). En fait, lors de la validation de la modification, le dbNavigateur demande l'exécution de la requête SQL indiquée dans l'objet dataAdapter (propriété UpdateCommand). Cette requête effectue tout d'abord une modification des valeurs dans la table, puis lance un SELECT destiné à récupérer d'éventuelles valeurs gérées par le SGDB (numéro automatique lors des ajouts par exemple).

Ce SELECT pose problème dans la mesure où une colonne « expression » est toujours en lecture seule, c'est-à-dire qu'une fois établi, son contenu ne peut plus être modifié. La solution est donc de supprimer cette commande SELECT (le point-virgule devient alors inutile). La même opération doit être faite pour la propriété InsertCommand.

En résumé, il est préférable de ne jamais demander l'actualisation du DataSet lors du paramétrage d'un DataAdapter (sauf cas particuliers bien sûr).

. Ajoutez un TextBox destiné à afficher votre champ calculé sur le formulaire « Fm_employe ».

. Consultez l'aide concernant la propriété « Expression » de la classe « DataColumn ». Vous pouvez constater que les possibilités sont nombreuses… Regardez la fonction ISNULL pour bien comprendre l'expression utilisée.
On peut notamment utiliser la relation définie entre les tables pour calculer la valeur du champ. Ainsi, l'expression « Parent(serviceemploye).designation » utilisée pour un champ calculé de la table employé retournera la désignation de son service.

Essayez donc cela :

- Ajoutez un élément « masseSalariale » de type « decimal » à la table service.
- Entrez l'expression de calcul : « Sum(Child(serviceemploye).salaire) ».
- Ajoutez un TextBox sur le formulaire Fm_service et utilisez une liaison de données pour afficher ce champ calculé.

2. Recherche d'enregistrements et de valeurs

. Créez une liste déroulante « cb_recherche », alimentée par un DataView basé sur la table employé et trié par valeurs croissantes du champ « recherche ». Cette liste doit afficher le champ « recherche », et avoir comme valeur le numéro d'employé.

. Cette liste peut être utilisée pour rechercher rapidement un employé :

```
private void cb_recherche_SelectedIndexChanged(object sender, System.EventArgs e)
{
    string numeroRecherche=cb_recherche.SelectedValue.ToString();
    int pos=dbDv.Find(numeroRecherche);
    dbNv_employe.Position=pos;
}
```

La méthode Find du DataView utilise le critère de tri actif (le numéro d'employé dans notre cas) pour rechercher la valeur passée en paramètre. On pourrait également utiliser directement le contexte de liaison de données en écrivant : « BindingContext[dbDv].Position=pos; » au lieu de « dbNv_employe.Position=pos; ».

. La méthode « FindRows » de la classe « DataView » est également intéressante. Elle doit être utilisée lorsque la recherche renvoie plusieurs lignes (critère de tri actif « cadre » par exemple). Cette méthode retourne un tableau d'objets « DataRowView ».

. Il est également possible de rechercher une valeur directement dans le DataSet. A titre d'exemple, voici le code d'un bouton permettant de rechercher et d'afficher la masse salariale du service de l'employé courant à partir du formulaire Fm_employe. Testez-le…

```
private void bt_masseSalariale_Click(object sender, System.EventArgs e)
{
    string codeSce=dbNv_employe.Courant()["sce"].ToString();
    DataRow service=dbDs.tp1_service.FindBycode(codeSce);
    string totalSalaires=service["masseSalariale"].ToString();
    MessageBox.Show(totalSalaires);
}
```

3. Autres utilisations des expressions

. Les expressions peuvent également être utilisées pour réaliser des calculs à la demande. Voici par exemple le code associé à un bouton du formulaire principal affichant le nombre de services :

```
private void bt_nbServices_Click(object sender, System.EventArgs e)
{
    int nb=(int)dbDs_empSce1.tp1_service.Compute("Count(code)",null);
    MessageBox.Show(nb.ToString());
}
```

. En utilisant cette technique, ajoutez au formulaire employé un TextBox affichant l'effectif de son service. Le calcul peut être réalisé par une méthode « majEffectif », cette méthode étant appelée lors du changement de valeur sélectionnée dans la liste cb_service.
Attention lors de l'écriture de « majEffectif », la valeur sélectionnée dans la liste (cb_service.SelectedValue) peut être « null », le calcul étant alors impossible… (La propriété SelectedIndex vaut –1 dans ce cas).

. Remarque importante :

Pour obtenir de tels résultats, Il est également possible d'exécuter une requête en direction de la base de données à l'aide de la méthode ExecuteScalar d'un objet SqlCommand. Mais attention, ceci ne retourne pas obligatoirement le même résultat puisque les données de la base et celle du DataSet ne sont pas obligatoirement les mêmes, notamment dans un contexte multi utilisateurs (voir paragraphe suivant).

Voici un exemple de cette technique pour le calcul du nombre de services :

```
private void bt_expression_Click(object sender, System.EventArgs e)
{
    int nb=(int)dbDs_empSce1.tp1_service.Compute("Count(code)",null);
    MessageBox.Show(nb.ToString());
    SqlCommand dbSql = new SqlCommand("select count(*) from service",dbCo_gesper);
    dbCo_gesper.Open();
    nb=(int)dbSql.ExecuteScalar();
    dbCo_gesper.Close();
    MessageBox.Show(nb.ToString());
}
```

D/ Vues de données et formulaires utilisant plusieurs tables

1. DataViewManager

Lorsqu'un formulaire est lié à des données issues de plusieurs tables du DataSet, l'utilisation de plusieurs DataViews ne suffit pas. En effet, ces différents objets DataView ne possèdent aucun lien entre eux, et notamment ne prennent pas en compte les relations définies entre les tables.

Il faut dans ce cas utiliser un DataViewManager qui prend en compte l'ensemble des tables du DataSet ainsi que les relations définies au sein de ce DataSet. Un DataViewManager permet de fixer les paramètres de tri et de filtrage de chaque table du DataSet.

Voici les modifications à apporter au formulaire Fm_empSce pour utiliser cette technique :

- Ajouter un membre données pour mémoriser le DataViewManager.

- Dans le constructeur du formulaire, instancier et paramétrer cet objet.

- Toujours dans le constructeur, initialiser les dbNavigateur à l'aide du DataViewManager et mettre en place la liaison de données.

- On obtient :

```
dbDs_empSce dbDs;
DataViewManager dbDm;
public Fm_empSce(dbDs_empSce p_dbDs):this()
{
// mémorisation du dataSet
    dbDs=p_dbDs;
// création et paramétrage du DataViewManager
    dbDm=new DataViewManager(dbDs);
    dbDm.DataViewSettings["tp1_service"].Sort="code";
    dbDm.DataViewSettings["tp1_employe"].Sort="numero";
// initialisation du dbNavigateur pour les services
```

```
        dbNv_service.Init(this.BindingContext[dbDm,"tp1_service"]);
    // mise en place des liaisons de données pour les services
        this.tb_code.DataBindings.Add("Text",dbDm,"tp1_service.code");
        this.tb_designation.DataBindings.Add("Text",dbDm,"tp1_service.designation");
    // initialisation du dbNavigateur pour les employés
        dbNv_employe.Init( BindingContext[dbDm,"tp1_service.serviceemploye"],
        new Navigation.OnDbEventFunction(surAjout),
        new Navigation.OnDbEventFunction(surDeplact));
    // mise en place des liaisons de données pour les employés
        this.tb_numero.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.numero");
        this.tb_nom.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.nom");
        this.tb_prenom.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.prenom");
        this.tb_salaire.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.salaire");
    // cas particulier de la liste des services
        this.cb_service.DataSource=new DataView(    dbDs.tp1_service,null,"designation",
                                                    DataViewRowState.CurrentRows);
        this.cb_service.DataBindings.Add("SelectedValue",dbDm,"tp1_service.serviceemploye.sce");
    // cas particulier de la case à cocher cb_cadre
        this.cb_cadre.DataBindings.Add("Checked",dbDm,"tp1_service.serviceemploye.cadre");
    }
```

3. Conclusion

La meilleure méthode consiste à utiliser **systématiquement** un DataViewManager pour les formulaires liés à plusieurs tables (initialisation des dbNavigateur et liaison de données).

E/ Cohérence des données

1. Mise à jour de la base de données à partir du DataSet

. Le dataSet que nous utilisons est une copie locale de notre base de données. Il faut donc gérer la cohérence entre les deux.

. Les mises à jour du DataSet sont immédiatement répercutées sur la base par les méthodes du formulaire principal gérant les événements « RowChanged » et « RowDeleted » des deux tables lors de la saisie de l'utilisateur. Notez bien que si vous faites des modifications « par programme », elles sont également automatiquement répercutées par ces méthodes.

Exemple :

```
    private void bt_modifEmploye_Click(object sender, System.EventArgs e)
    {
        dbDs_empSce1.tp1_employe.Rows[2]["prenom"]="Gédéon";
        // mise à jour de la base de données totalement inutile !!!
        dbAd_employe.Update(dbDs_empSce1.tp1_employe);
    }
```

L'appel de la méthode « Update » est de trop, celle-ci sera automatiquement appelée par la méthode « GererRowAction ».

. Il est possible d'ajouter une ligne et ou de supprimer une ligne à une table dans le dataset :

```
    DataRow dr=dbDs_empSce1.tp1_service.NewRow();
    dr["code"]="s8";
    dr["designation"]="Atelier 3";
    dbDs_empSce1.tp1_service.Rows.Add(dr);
    dbDs_empSce1.tp1_service.Rows[8].Delete();      // suppression du 9ème enregistrement
```

Là encore, les modifications seront automatiquement répercutées.

2. Rafraîchissement des données du DataSet

- Dans un contexte multi utilisateurs, la base de données peut être modifiée à l'extérieur de notre programme. Pour refléter ces modifications, le DataSet doit être rechargé.

Voici le code destiné à recharger le DataSet :

```
private void bt_recharger_Click(object sender, System.EventArgs e)
{
    dbDs_empSce1.tp1_employe.RowChanged-=new DataRowChangeEventHandler(tp1_employe_RowChanged);
    dbDs_empSce1.tp1_employe.RowDeleted-=new DataRowChangeEventHandler(tp1_employe_RowDeleted);
    dbDs_empSce1.tp1_service.RowChanged-=new DataRowChangeEventHandler(tp1_service_RowChanged);
    dbDs_empSce1.tp1_service.RowDeleted-=new DataRowChangeEventHandler(tp1_service_RowDeleted);
    dbDs_empSce1.EnforceConstraints=false;
    dbDs_empSce1.tp1_employe.Clear();
    dbDs_empSce1.tp1_service.Clear();
    dbAd_service.Fill(dbDs_empSce1,"tp1_service");
    dbAd_employe.Fill(dbDs_empSce1,"tp1_employe");
    dbDs_empSce1.EnforceConstraints=true;
    dbDs_empSce1.tp1_employe.RowChanged+=new DataRowChangeEventHandler(tp1_employe_RowChanged);
    dbDs_empSce1.tp1_employe.RowDeleted+=new DataRowChangeEventHandler(tp1_employe_RowDeleted);
    dbDs_empSce1.tp1_service.RowChanged+=new DataRowChangeEventHandler(tp1_service_RowChanged);
    dbDs_empSce1.tp1_service.RowDeleted+=new DataRowChangeEventHandler(tp1_service_RowDeleted);
}
```

Remarques :

- La propriété « EnforceConstraints » est utilisée pour désactiver temporairement les contrôles d'intégrité au niveau du DataSet (ce n'est pas obligatoire ici puisque la table employé est vidée avant la table service, et rechargée après la table service).
- La désactivation temporaire des gestionnaires d'événements est une précaution utile. En pratique, elle n'est nécessaire que pour l'événement « RowChanged » de la table employé car il y a modification de l'enregistrement, et donc conflit avec la procédure de chargement.