C# - FICHE PRATIQUE N° 6 – Solutions et compléments

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

A/ Tri et filtrage d'enregistrements

Voici le code du bouton permettant de modifier l'ordre de tri :

```
private void bt_modifTri_Click(object sender, System.EventArgs e)
{
    BindingContext[dbDv].SuspendBinding();
    if (dbDv.Sort=="code")
    {
        dbDv.Sort="designation";
    }
    else
    {
        dbDv.Sort="code";
    }
    BindingContext[dbDv].ResumeBinding();
}
```

La technique de création d'un DataView « à la volée » pour alimenter une liste déroulante devrait être systématisée.

Il convient de noter qu'un DataView n'est qu'un objet de présentation des données. Il n'est donc pas coûteux en terme de ressources.

Enfin, il faut signaler qu'un DataView peut également être créé en mode graphique.

B/ Colonnes calculées et recherche d'enregistrements

> La remarque concernant le problème lié à l'instruction « select » ajoutée aux commandes d'insertion et de modification devrait conduire à systématiquement ne pas demander l'actualisation du DataSet lors de la création d'un DataAdapter.

> Si l'expression de calcul d'un élément comporte une erreur, il est possible que le type lié au DataSet (dbDs_empSce) ne puisse être construit. De ce fait, la déclaration de la variable représentant le DataSet lui-même (dbDs_empSce1) ne peut plus être compilée, le projet semble « cassé ». Il faut alors reprendre le schéma du DataSet (toujours accessible via l'explorateur de solutions), et corriger l'erreur. Malheureusement, le problème posé conduit Visual studio à supprimer l'initialisation de la variable « dbDs_empSce1 » dans la fonction « InitializeComponent ». Pour « réparer », il faut supprimer la déclaration de cette variable dans le code et en recréer une de même nom. Il suffit d'utiliser l'outil DataSet de la boîte à outils et de choisir un DataSet du type « dbDs_empSce », le nom généré automatiquement convient : « dbDs_empSce1 ».

> Voici le code à placer dans le constructeur de Fm_employe pour initialiser la liste déroulante de recherche :

```
cb_recherche.DataSource=new DataView(dbDs.tp1_employe,null,"recherche",DataViewRowState.CurrentRows);
cb_recherche.DisplayMember="recherche";
cb_recherche.ValueMember="numero";
```

> La procédure de gestion de la modification de la liste est la suivante :

```
private void cb_service_SelectedIndexChanged(     object sender,
System.EventArgs e)
{
    dbNv_employe.Valider(); // intégrer l'employé dans le service
    majEffectif();
}
```

> La fonction de calcul de l'effectif du service d'un employé est :

```
private void majEffectif()
{
    if (cb_service.SelectedIndex!=-1)
    {
        int nb;
        string filtre;
        filtre="sce='" + cb_service.SelectedValue + "'";
        nb=(int)dbDs.tp1_employe.Compute("Count(numero)",filtre);
        tb_effectif.Text=nb.ToString();
    }
    else
    {
        tb_effectif.Text="";
    }
}
```

D/ Vues de données et formulaires utilisant plusieurs tables

Il est bien entendu possible de modifier l'ordre de tri et d'ajouter une sélection des enregistrements. Par exemple, en écrivant :

```
dbDm.DataViewSettings["tp1_service"].Sort="designation DESC";
dbDm.DataViewSettings["tp1_service"].RowFilter="code>'s3'";
dbDm.DataViewSettings["tp1_employe"].Sort="salaire,nom DESC";
dbDm.DataViewSettings["tp1_employe"].RowFilter="salaire>1000";
```

On obtient ici les services dont le code est supérieur à 's3', triés en ordre décroissant des désignations ainsi que les employés gagnant plus de 1000 € associés à chaque service triés par salaire croissant et nom décroissant…

E/ Cohérence des données

Avec la version 1 du framework, un problème se pose : après le rafraîchissement des données du DataSet, la liaison de données du formulaire fm_empSce ne fonctionne plus correctement. En fait, le fait de recharger la table enfant de la relation a fait perdre la liaison de données.

Ce problème n'existe plus avec les versions suivantes.

. Une première solution simple : il suffit de fermer le formulaire (ce qui le détruit), et de le recréer après le chargement des nouvelles données, ce qui rétablit la liaison.

```
private void bt_recharger_Click(object sender, System.EventArgs e)
{
    bool fempSceOpen=fmse.Visible;
    fempSce.Close();
    dbDs_empSce1.Clear();
    dbAd_service.Fill(dbDs_empSce1,"tp1_service");
    dbAd_employe.Fill(dbDs_empSce1,"tp1_employe");
    fempSce =new fm_empSce(dbDs_empSce1);
    if (fempSceOpen)
    {
        fempSce.Show();
    }
}
```

Remarque : il est possible de mémoriser également la position de chaque dbNavigateur avant la fermeture, le fait que le formulaire soit actif ou non, sa position dans la fenêtre…

. Une autre solution consiste à remplir un dataSet temporaire, à vider le dataSet principal, puis à les fusionner à l'aide de la méthode « merge » de la classe DataSet. Cependant, on a alors deux copies de la base de données pendant un certain temp…

. Une troisième solution consiste à désactiver toutes les liaisons de données du formulaire avant de recharger les données et à les réactiver après le chargement.

On obtient :

```
// Dans la classe Fm_principal

private void bt_recharger_Click(object sender, System.EventArgs e)
{
    fempSce.EffacerLiaisons();
    dbDs_empSce1.Clear();
    dbAd_service.Fill(dbDs_empSce1,"tp1_service");
    dbAd_employe.Fill(dbDs_empSce1,"tp1_employe");
    fempSce.ActiverLiaisons();
}



// Dans la classe Fm_empSce

public void EffacerLiaisons()
{
    foreach (Control c in Controls)    // parcours d'une collection
    {
        c.DataBindings.Clear();
    }
}
```

```
    public void ActiverLiaisons()
    {
        // mise en place des liaisons de données pour les services
        tb_code.DataBindings.Add("Text",dbDm, "tp1_service.code");
        tb_designation.DataBindings.Add("Text",dbDm,"tp1_service.designation");
        // mise en place des liaisons de données pour les employés
        tb_numero.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.numero");
        tb_nom.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.nom");
        tb_prenom.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.prenom");
        tb_salaire.DataBindings.Add("Text",dbDm,"tp1_service.serviceemploye.salaire");
        cb_service.DataSource=new DataView(     dbDs.tp1_service,null,"designation",
                                                DataViewRowState.CurrentRows);
        cb_service.DataBindings.Add("SelectedValue",dbDm,"tp1_service.serviceemploye.sce");
        cb_cadre.DataBindings.Add("Checked",dbDm,"tp1_service.serviceemploye.cadre");
    }

// Le constructeur de Fm_empSce peut alors être réécrit de la manière suivante

    public Fm_empSce(dbDs_empSce p_dbDs):this()
    {
        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 des dbNavigateur
        dbnv_service.init(BindingContext[dbDm,"tp1_service"]);
        dbnv_employe.init(
                        BindingContext[dbDm,"tp1_service.serviceemploye"],
                        new Navigation.OnDbEventFunction(surAjout),
                        new Navigation.OnDbEventFunction(surDeplact));
        // mise en place des liaisons de données
        ActiverLiaisons();
    }
```

En clair, passez à la version suivante…