C# - FICHE PRATIQUE N° 6 – Pour aller plus loin

Une petite gestion de personnel
manipulation d'enregistrements par programme.

La manipulation d'enregistrements par programme permet de réaliser des traitements personnalisés sur les données du DataSet. Le plus simple est de le faire à l'aide d'un DataView, de plusieurs DataViews, ou encore d'un dataViewManager.

1. Premier exemple

- Créer un nouveau formulaire « Fm_enumerateur », mettre en place le mécanisme lui permettant de récupérer un accès au formulaire principal, et créer un bouton permettant d'ouvrir ce formulaire depuis le formulaire principal.
- Ajouter un bouton « bt_services » à ce formulaire.
- Ajouter le code ci-dessous et tester…

```
private DataView dbDv_service;
private void bt_services_Click(object sender, System.EventArgs e)
{
    dbDv_service=new DataView(    dbDs.tp1_service,null,"code", DataViewRowState.CurrentRows);
    MessageBox.Show(dbDv_service.Count.ToString());
}
```

Ce bouton permet de créer un DatatView sur la table service et affiche le nombre de lignes.

Nous allons améliorer les choses pour afficher la désignation de chacun des services.

2. Notion d'énumérateur

> Un énumérateur (plus communément désigné par le terme itérateur) est un mécanisme permettant de parcourir les éléments d'une collection, c'est-à-dire d'obtenir un par un chacun des objets qu'elle contient.

> Un premier mécanisme simple consiste à utiliser l'instruction « foreach » permettant de balayer le contenu d'une collection :

```
ArrayList mesChaines=new ArrayList();
mesChaines.Add("un");
mesChaines.Add("deux");
mesChaines.Add("trois");
tb_resultat.Clear();
foreach (string s in mesChaines)
{
    tb_resultat.AppendText(s+Environment.NewLine);
}
```

Si les objets de la collection ne sont pas tous du même type, il faut prévoir chaque cas (ou utiliser le polymorphisme) :

```
ArrayList mixte=new ArrayList();
mixte.Add("un");
mixte.Add(2);
mixte.Add("trois");
mixte.Add(4);
tb_resultat.Clear();
foreach (object elt in mixte)
{
    if(elt.GetType()==typeof(string))
    {
        tb_resultat.AppendText(elt+Environment.NewLine);
    }
    else if(elt.GetType()==typeof(int))
    {
        tb_resultat.AppendText(elt.ToString()+Environment.NewLine);
    }
}
```

> Une seconde solution consiste à utiliser un objet énumérateur. C'est un peu plus complexe, mais cela permet par exemple de réaliser plusieurs parcours simultanés de la collection de manière indépendante.

```
ArrayList mesChaines=new ArrayList();
mesChaines.Add("un");
mesChaines.Add("deux");
mesChaines.Add("trois");
IEnumerator iter=mesChaines.GetEnumerator();
tb_resultat.Clear();
while (iter.MoveNext())
{
    tb_resultat.AppendText((string)iter.Current+Environment.NewLine);
}
```

3. Application au DataView

- Créer un TextBox tb_resultat sur le formulaire (il peut être utilisé pour tester les exemples ci-dessus).
- Paramétrer ses propriétés : Multiline (true), ScrollBars (vertical).
- Modifier le code du bouton « bt_services » :

```
private void bt_services_Click(object sender, System.EventArgs e)
{
    dbDv_service=new DataView(    dbDs.tp1_service,null,"code",
    DataViewRowState.CurrentRows);
    IEnumerator iter=dbDv_service.GetEnumerator();
    DataRowView vueLigne;
    string ligne;
    tb_resultat.Clear();
    while (iter.MoveNext())
    {
        vueLigne=(DataRowView)iter.Current;
        ligne=vueLigne.Row["code"].ToString()+"\t" ;
        ligne+=vueLigne.Row["designation"].ToString();
        tb_resultat.AppendText(ligne+Environment.NewLine);
    }
}
```

Remarque : le DataView est traité ici comme une collection d'objets DataRowView, chacun de ces objets permettant l'accès à la ligne de la table correspondante.

4. Utilisation des relations

Les relations définies entre les tables du DataSet peuvent être utilisées pour créer et exploiter des DataViews. Voici à titre d'exemple, le code permettant de retrouver et d'afficher les employés de chaque service.

```
private void bt_sceEmployes_Click(object sender, System.EventArgs e)
{
    DataView dbDv_employe;
    IEnumerator iterEmp;
    DataRowView vueLigne;
    dbDv_service=new DataView(    dbDs.tp1_service,null,"code", DataViewRowState.CurrentRows);
    IEnumerator iterSce=dbDv_service.GetEnumerator();
    tb_resultat.Clear();
    while (iterSce.MoveNext())
    {
        vueLigne=(DataRowView)iterSce.Current;
        tb_resultat.AppendText(vueLigne.Row["designation"].ToString() + Environment.NewLine);
        dbDv_employe=vueLigne.CreateChildView("serviceemploye");
        iterEmp=dbDv_employe.GetEnumerator();
        while (iterEmp.MoveNext())
        {
            vueLigne=(DataRowView)iterEmp.Current;
            tb_resultat.AppendText("\t" + vueLigne.Row["nom"].ToString() + Environment.NewLine);
        }
    }
}
```

5. Modification des données

Il est possible de modifier les données manipulées de cette manière. Le code ci-dessous augmente de 100 € le salaire de tous les employés. Attention, l'appel à la méthode « Update » du DataAdapter serait de trop, les modifications sont automatiquement répercutées sur la base de données SQL Server par la gestion des événements « RowChanged » et « RowDeleted » dans le formulaire principal.

```
private void bt_augmenter_Click(object sender, System.EventArgs e)
{
    DataView dbDv_employe=new DataView(dbDs.tp1_employe,null,"numero", DataViewRowState.CurrentRows);
    IEnumerator iterEmp=dbDv_employe.GetEnumerator();
    DataRowView vueLigne;
    string salaireValue;
    decimal salaire;
    while (iterEmp.MoveNext())
    {
        vueLigne=(DataRowView)iterEmp.Current;
        salaireValue=vueLigne.Row["salaire"].ToString();
        if (salaireValue!="")
        {
            salaire=decimal.Parse(salaireValue);
            salaire+=100;
            vueLigne.Row["salaire"]=salaire;
        }
    }
}
```

On peut également ajouter et supprimer des enregistrements :

```
private void bt_ajoutSupp_Click(object sender, System.EventArgs e)
{
    DataView dbDv_employe=new DataView(dbDs.tp1_employe,null,"numero",DataViewRowState.CurrentRows);
    DataRowView nouveau=dbDv_employe.AddNew();
    nouveau.Row["nom"]="Dupond"; nouveau.Row["prenom"]="Paul";
    nouveau.Row["sexe"]="m";    nouveau.Row["cadre"]=true;
    nouveau.Row["salaire"]=2000;    nouveau.Row["sce"]="s1";
    nouveau.EndEdit();
    dbDv_employe.Delete(0);
}
```

Remarque : On ne peut pas supprimer une ligne lors du parcours du DataView à l'aide d'un énumérateur car cet objet donne un accès en lecture seule à une collection.

6. Utilisation d'un DataViewManager

On peut également utiliser un DataViewManager pour parcourir les données.

- La technique consiste dans ce cas à utiliser la propriété « BindingContext » du formulaire (et non un énumérateur). Il devient donc possible d'ajouter ou de supprimer des enregistrements en cours de parcours.
- Attention, l'utilisation d'une variable intermédiaire (posSce et posEmp) est nécessaire car la propriété « Position » ne peut jamais prendre une valeur supérieure à « Count –1 ».

Le code ci-dessous illustre cette possibilité.

```
private void bt_dataViewManager_Click(object sender, System.EventArgs e)
{
    DataViewManager dbDm=new DataViewManager(dbDs);
    dbDm.DataViewSettings["tp1_service"].Sort="designation";
    dbDm.DataViewSettings["tp1_employe"].Sort="nom";
    DataRowView vueLigne;
    string ligne;   int posSce=0;    int posEmp;
    tb_resultat.Clear();
    while (posSce<BindingContext[dbDm,"tp1_service"].Count)
    {
        BindingContext[dbDm,"tp1_service"].Position=posSce;
        vueLigne=(DataRowView)BindingContext[dbDm,"tp1_service"].Current;
        ligne= vueLigne.Row["code"].ToString() + "\t"+vueLigne.Row["designation"].ToString();
        tb_resultat.AppendText(ligne+Environment.NewLine);
        posEmp=0;
        while (posEmp<BindingContext[dbDm,"tp1_service.serviceemploye"].Count)
        {
            BindingContext[dbDm,"tp1_service.serviceemploye"].Position=posEmp;
            vueLigne=(DataRowView)BindingContext[dbDm,"tp1_service.serviceemploye"].Current;
            ligne= "\t"+vueLigne.Row["numero"].ToString() + "\t"+vueLigne.Row["nom"].ToString();
            tb_resultat.AppendText(ligne+Environment.NewLine);
            posEmp++;
        }
        posSce++;
    }
}
```