Les objets DataAdapter et Dataset dans ADO.NET

Table des matières

Cet article traite des objets Connection, Command et Datareader de ADO.NET

L'auteur

Leduke

L'article

Publié le 3 juin 2003

TOUT PUBLIC

Durée : 30 à 45 minutes

Version PDF Version hors-ligne

ePub, Azw et Mobi

I. Introduction▲

Dans mon précédent article : nous avons abordé les objets Connection, Command et DataReader qui permettent la récupération de données en mode connecté. Aujourd'hui nous allons vous présenter un nouvel objet le DataSet qui est probablement l'une des innovations les plus importantes sous ADO .NET.

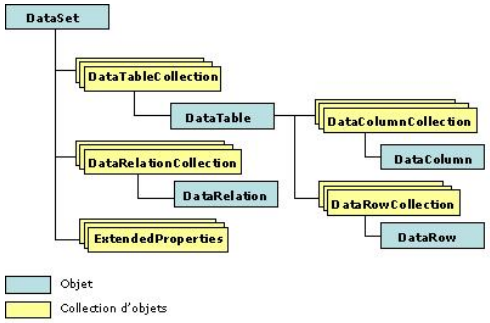
II. L'objet DataSet▲

Le DataSet est un objet qui réside en mémoire et qui correspond à une copie locale des données d'une base. Il contient les tables d'une base, mais aussi les relations entre ces différentes tables et les contraintes appliquées aux données.

XML est sous .NET utilisé comme le standard de description et de persistance des données et à ce titre l'objet DataSet est représenté sous cette forme. Les données d'un DataSet sont écrites en XML et le schéma est écrit en XSD (XML Schema Definition Language).

Comme l'objet DataSet est déconnecté de toute source de données, il peut être utilisé pour la transmission de données entre différents tiers d'une application Web (rôle assuré par les Recordsets déconnectés sous ADO), mais aussi à d'autres systèmes grâce à sa représentation XML.

Regardons plus précisément le modèle objet d'un DataSet.



Comme vous pouvez le constater, le modèle objet du DataSet s'apparente à celui d'une base de données relationnelle. Il est principalement constitué des trois collections

suivantes :

- la collection "**DataTableCollection**" :
cette collection peut contenir de zéro à n objets de type DataTable. Chaque objet DataTable représente une table d'une source de données. Chaque DataTable est constituée d'une collection Columns et d'une collection Rows qui peuvent contenir respectivement de zéro à n objets DataRow et DataColumn ;
- la collection "**DataRelationCollection**" :
cette collection peut contenir de zéro à n objets de type DataRelation. Un objet DataRelation définit une relation parent-enfant entre deux tables à partir des valeurs des clés étrangères ;
- la collection "**ExtendedProperties**" :
cette collection correspond à un objet de type PropertyCollection qui peut contenir de zéro à n propriétés définies par un utilisateur. Cette collection peut être utilisée afin de stocker des informations personnalisées liées au DataSet utilisé (date et heure de génération des données, ordre select passé pour générer les données).

La création d'un DataSet

La création d'une instance d'un objet DataSet se fait de la manière suivante :

Sélectionnez

```
// Création d'un objet DataSet

DataSet oDataSet = new DataSet("Customers");
```

Il est possible de préciser en argument le nom du DataSet, si vous l'omettez c'est le nom "NewDataSet" qui est attribué automatiquement.

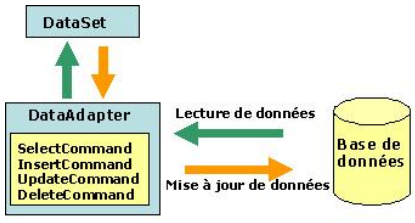
III. L'objet DataAdapter▲

Un objet DataSet doit être capable d'interagir avec une ou plusieurs sources de données. Pour réaliser cette action le framework Microsoft .Net fournit un objet nommé DataAdapter. L'objet DataAdapter sert de liaison entre un objet DataSet et une source de données. Un fournisseur de données .NET va se servir d'un objet DataAdapter pour remplir de données un DataSet puis répercuter les modifications réalisées sur une source de données. Il est possible de définir les actions à réaliser par le DataAdapter en utilisant l'une des quatre propriétés suivantes. Chaque propriété exécutera soit un ordre SQL soit une procédure stockée.

Les propriétés proposées par la classe DataAdapter sont les suivantes :

Propriété	Description
SelectCommand	La propriété SelectCommand permet de récupérer des données d'une source
InsertCommand	La propriété InsertCommand écrit les lignes insérées d'un DataSet vers une source de données
UpdateCommand	La propriété UpdateCommand écrit les lignes modifiées d'un DataSet vers une source de données
DeleteCommand	La propriété DeleteCommand efface les lignes supprimées d'un DataSet d'une source de données

Le schéma ci-joint présente le fonctionnement de l'objet DataAdapter.



Comme pour les objets Connection et Command, chaque fournisseur de données va proposer son objet DataAdapter spécifique. Le fournisseur de données OLEDB .NET inclut un objet **OleDbDataAdapter** et le fournisseur de données SQL Server .NET inclut un objet **SqlDataAdapter**. C'est ce dernier que nous utiliserons dans la suite de cet article.

La lecture de données et le remplissage d'un DataSet :
La méthode Fill de l'objet DataAdapter permet d'extraire les données d'une source de données en exécutant la requête SQL spécifiée dans la propriété SelectCommand. Elle

prend en paramètre le DataSet et le nom du DataTable à remplir avec les données retournées.

Sélectionnez

```
// Création d'un objet SqlDataAdapter
SqlDataAdapter oSqlDataAdapter = new SqlDataAdapter("select * from customers", oConnection);

DataSet oDataSet = new DataSet("Customers");

oSqlDataAdapter.Fill(oDataSet,"Customers");
```

La mise à jour de données avec un DataAdapter et un DataSet :
La méthode Update de l'objet DataAdapter permet de répercuter sur une source de données les modifications effectuées dans un DataSet. Cette méthode admet un objet DataSet qui contient les données modifiées et un objet DataTable optionnel qui contient les modifications à extraire.

Sélectionnez

```
oSqlDataAdapter.Update(oDataSet);
```

Mais me direz-vous comment ADO .NET assure-t-il la relation entre les données d'un DataSet et celles contenues dans une base de données ?. En réalité, lorsque la méthode Update est invoquée l'objet DataAdapter analyse les modifications effectuées au niveau de la collection DataRow. En fonction des modifications rencontrées, les commandes InsertCommand, UpdateCommand et DeleteCommand sont appelées par l'objet DataAdapter.

Mettons désormais en pratique ce que nous venons d'étudier à travers un exemple. Toujours en utilisant la base d'exemple SQL2000 Microsoft nommée "NorthWind".

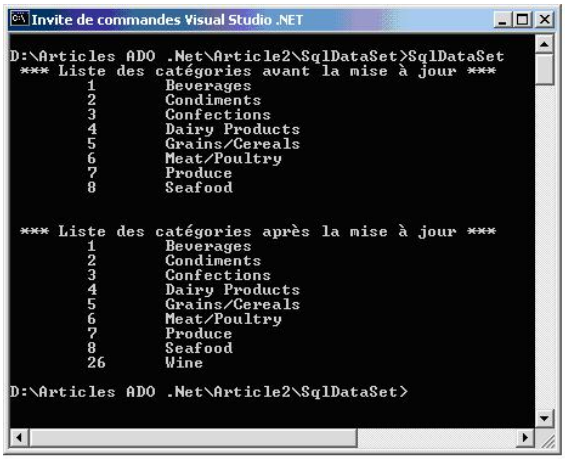
Dans cet exemple, nous allons :

- récupérer la liste des catégories de la table "Categories" dans un DataSet ;
- visualiser la liste des catégories ;
- insérer de nouvelles catégories via une propriété InsertCommand ;
- actualiser le DataSet et visualiser les modifications apportées à la table "Categories".

Sélectionnez

```
// Exemple d'utilisation d'un DataSet en mise à jour
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace ExempleAdoNetCSharp
{
/// Description résumée de SqlDataSet.

public class SqlDataSet
{
public static void Main()
{
string strConnexion = "Data Source=localhost; Integrated Security=SSPI;" + "Initial Catalog=
string strRequete = "SELECT * FROM Categories ORDER BY CategoryID";
try
{
SqlConnection oConnection = new SqlConnection(strConnexion);
oConnection.Open();

// Chargement de la liste des catégories dans oDataSet
SqlDataAdapter oSqlDataAdapter = new SqlDataAdapter(strRequete, oConnection);

DataSet oDataSet = new DataSet("Categories");
oSqlDataAdapter.Fill(oDataSet,"Categories");

// Affichage du contenu de oDataSet avant insertion de données

Console.WriteLine(" *** Liste des catégories avant la mise à jour *** ");
for (int i=0 ; i < oDataSet.Tables["Categories"].Rows.Count ; i++)
{
Console.WriteLine("\t{0}\t{1}",oDataSet.Tables["Categories"].Rows[i][0].ToString(),oDataSet.
}
Console.WriteLine("\n");

// Remplissage de la commande InsetCommand
oSqlDataAdapter.InsertCommand = new SqlCommand("INSERT INTO Categories(CategoryName, Descrip
oSqlDataAdapter.InsertCommand.Parameters.Add("@CategoryName", SqlDbType.NVarChar, 15, "Categ
oSqlDataAdapter.InsertCommand.Parameters.Add("@Description", SqlDbType.NText, 16, "Descripti
oSqlDataAdapter.InsertCommand.Parameters.Add("@Picture", SqlDbType.Image, 16, "Picture");

DataRow oDataRow;
byte [] byteArray = { 0x00,0x00};

oDataRow = oDataSet.Tables["Categories"].NewRow();
oDataRow["CategoryName"]= "Wine";
oDataRow["Description"]= "French Wine";
oDataRow["Picture"] = byteArray;

oDataSet.Tables["Categories"].Rows.Add(oDataRow);

// Mise à jour de la source de données à partir du DataSet
oSqlDataAdapter.Update(oDataSet,"Categories");
```

```
// Rechargement des données de la source mise à jour
oDataSet.Clear();
oSqlDataAdapter.Fill(oDataSet,"Categories");

// Affichage du contenu de oDataSet après insertion d'une ligne de données
Console.WriteLine(" *** Liste des catégories après la mise à jour *** ");
for (int i=0 ; i < oDataSet.Tables["Categories"].Rows.Count ; i++)
{
Console.WriteLine("\t{0}\t{1}",oDataSet.Tables["Categories"].Rows[i][0].ToString(),oDataSet.
}
oConnection.Close();
}
catch (Exception e)
{
Console.WriteLine("L'erreur suivante a été rencontrée :" + e.Message);
}
}
}
}
```



IV. Conclusion▲

Nous avons abordé dans cet article les objets DataSet et DataAdapter du modèle objet ADO .Net. Ces objets sont recommandés lorsqu'il est nécessaire de réaliser de nombreux traitements sur des données (modifications, filtres, tris.) mais aussi pour agréger des données issues de plusieurs sources.

Voir également mes autres articles :

Migrer de C++ vers C#

La FAQ .NET par Leduke et Neo.51

L'authentification par formulaire en ASP.NET

Vous avez aimé ce tutoriel ? Alors partagez-le en cliquant sur les boutons suivants :

Partager