Livret – 13

Les données relationnelles dans .Net

ADO .Net

RM di scala

Cours informatique programmation

Rm di Scala - http://www.discala.net

SOMMAIRE

ADO .Net : données relationnelles de .Net 2.0

Plan général:

1. Le modèle DataSet dans ADO.Net 2.0

DataSet
DataTable
DataColumn
DataRow
DataRelation

2. Création et utilisation d'un DataSet

Création d'un DataSet
Création de 2 DataTable
Définition du schéma de colonnes de chaque table par des DataColumn.
Définition d'une clef primaire.
Ajout des 2 tables au DataSet.
Ajout d'une relation entre les deux tables.
Exemple de création de données dans le DataSet
Affichage des données d'un DataSet
Sauvegarde des données du DataSet aux formats XML et XSL

Introduction

Nous avons vu au chapitre précédent que les fichiers simples peuvent être accédés par des flux, dès que l'organisation de l'information dans le fichier est fortement structurée les flux ne sont plus assez puissants pour fournir un accès souple aux données en particulier pour des applications à architecture multi-tiers (client/serveur, internet,…) et spécialement aux bases de données.

ADO .NET est un regroupement de types (classes, interfaces, …) dans l'espace de nom **System.Data** construits par Microsoft afin de manipuler des données structurées dans le .NET Framework.

Le modèle ADO .NET du .NET Framework fournit au développeur un ensemble d'éléments lui permettant de travailler sur des données aussi bien en mode connecté qu'en mode déconnecté (ce dernier mode est le mode préférentiel d' ADO .NET car c'est celui qui est le plus adapté aux architectures multi-tiers). ADO .NET est indépendant du mode de stockage des données : les classes s'adaptent automatiquement à l'organisation

ADO .NET permet de traiter des données situées dans des bases de données selon le modèle relationnel mais il supporte aussi les données organisées selon le modèle hiérarchique.

ADO .NET échange toutes ses informations au format XML

L'entité la plus importante d' ADO .NET permettant de gérer les données en local dans une mémoire cache complétement déconnectée de la source de données (donc indépendante de cette source) est le DataSet en fait la classe **System.Data.DataSet** et la collection de classes qui lui sont liées.

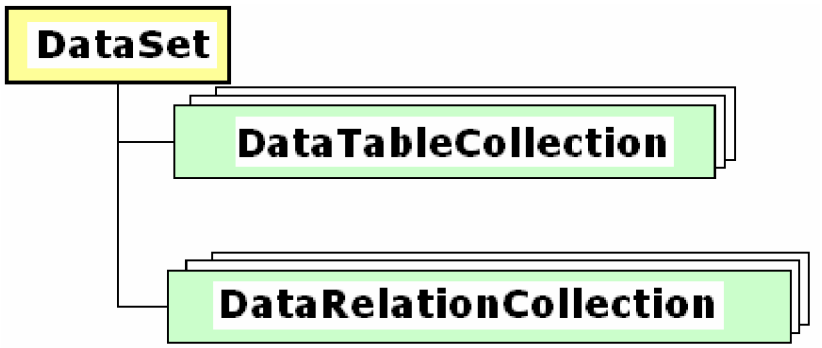
1. Le modèle DataSet dans ADO.Net 2.0

Le principe de base du DataSet est de se connecter à une source de données (par exemple une base de données) de charger toutes ses tables avec leur relations, puis ensuite de travailler en mode déconnecté sur ces tables en mémoire et enfin se reconnecter pour effectuer la mise à jour éventuelle des données.

Les classes mises en jeu lors d'une telle opération sur les données sont les suivantes :

System.Data.DataSet	
System.Data.DataTable	**System.Data.DataTableCollection**
System.Data.DataColumn	**System.Data.DataColumnCollection**
System.Data.DataRow	**System.Data.DataRowCollection**
System.Data.DataRelation	**System.Data.DataRelationCollection**
System.Data.DataConstraint	**System.Data.ConstraintCollection**
System.Data.DataView	

Un **DataSet** est en fait une collection de tables représentée par un objet de collection de la classe **DataTableCollection** et une collection de relations entres ces tables tables représentée par un objet de collection de la classe **DataRelationCollection**.



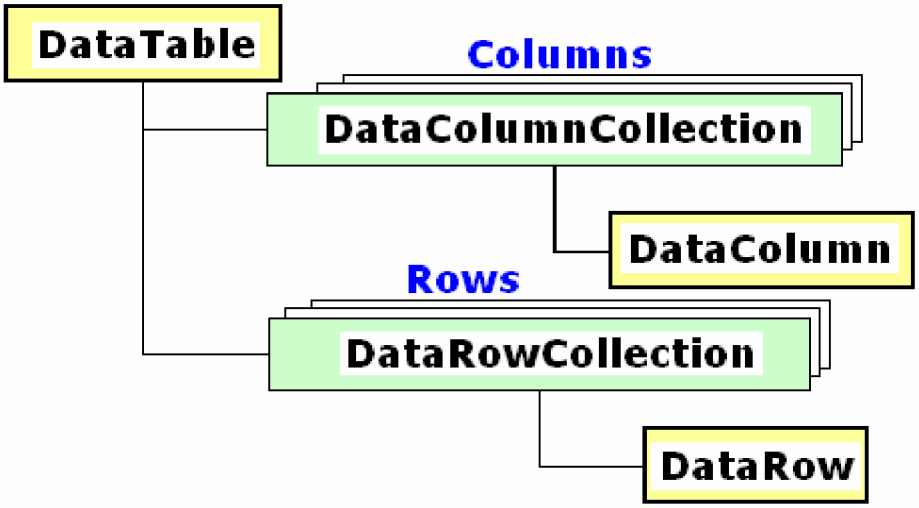
Une **DataTableCollection** est une collection (famille) d'une ou plusieurs DataTable qui représentent chacunes une table dans le DataSet.

Une table générale est une entité composée de :

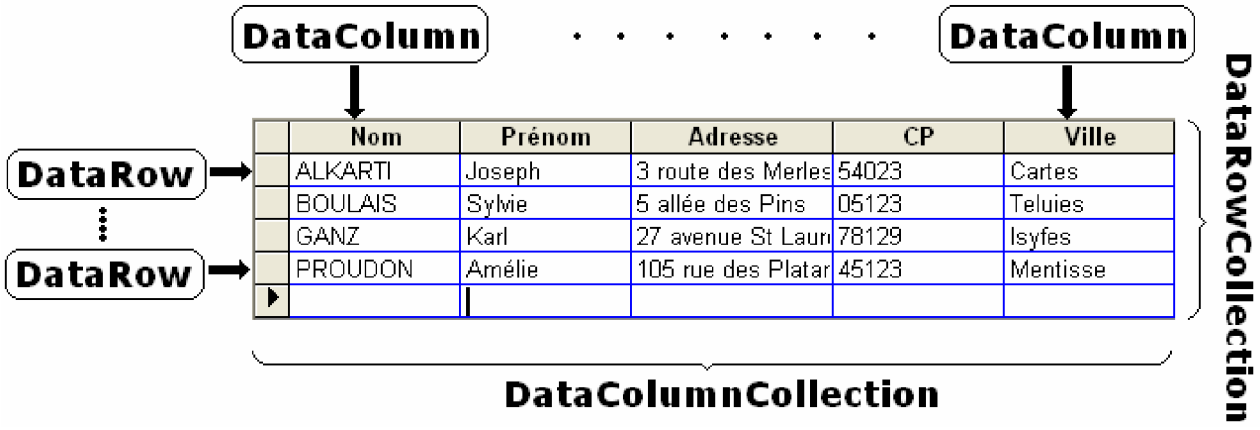
- Colonnes
- Lignes

Un objet de classe **DataTable** est composé en particulier des objets suivants :

- Une propriété **Columns** = Collection de colonnes (collection d'objets **DataColumn**)
- Une propriété **Rows** = Collection de lignes (collection d'objets **DataRow**)

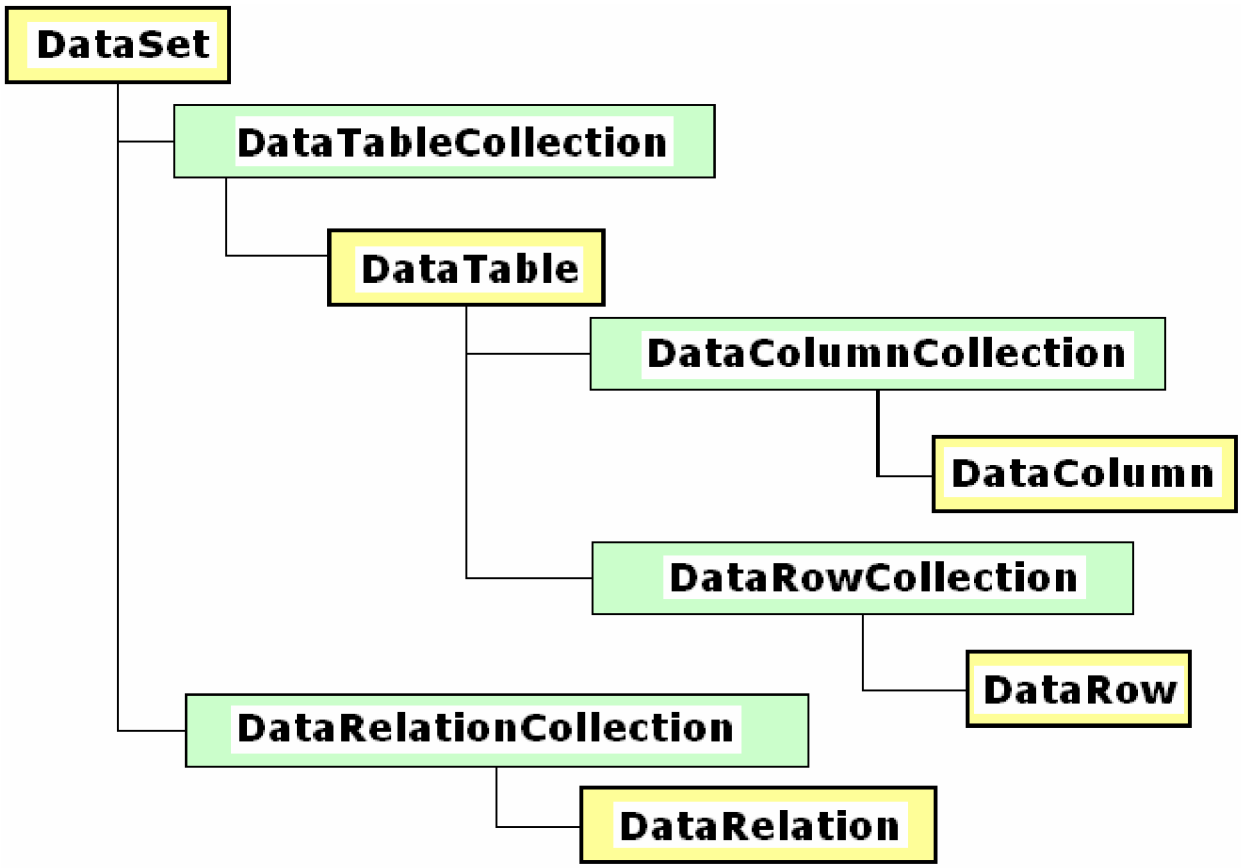


Le schéma général de la table d'un objet **DataTable** est donné par la famille des colonnes (l'objet Columns) chaque objet **DataColumn** de la collection représente une colonne de la table :

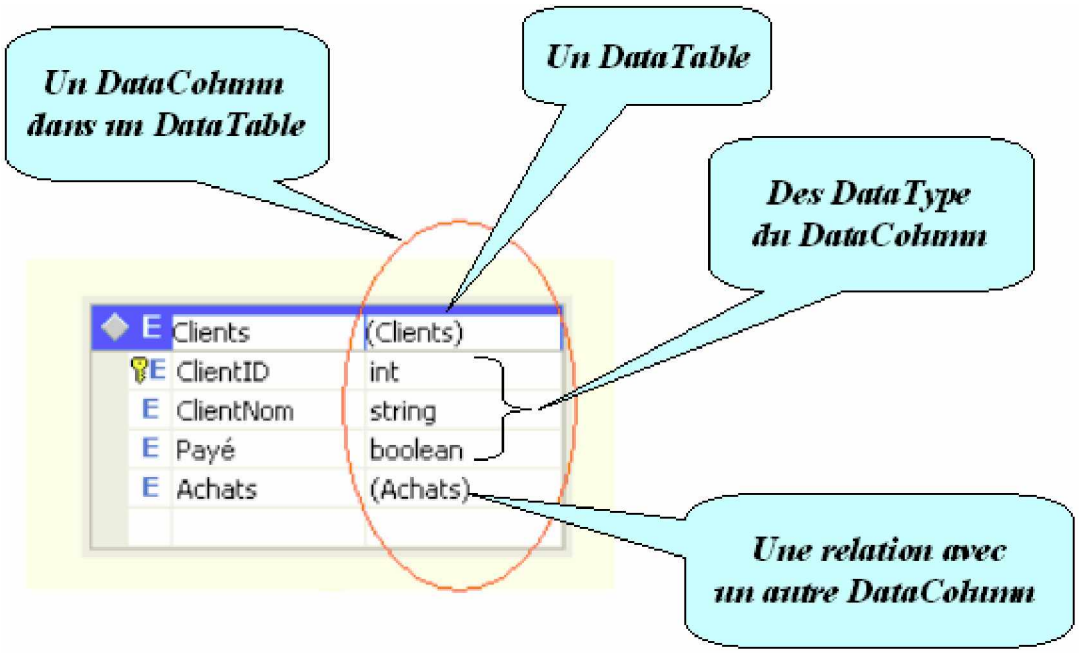


Pour résumer, un **DataSet** contient pour l'essentiel deux types d'objets :

1°) des objets **DataTable** inclus dans une **DataTableCollection**
2°) des objets **DataRelation** inclus dans une **DataRelationCollection**



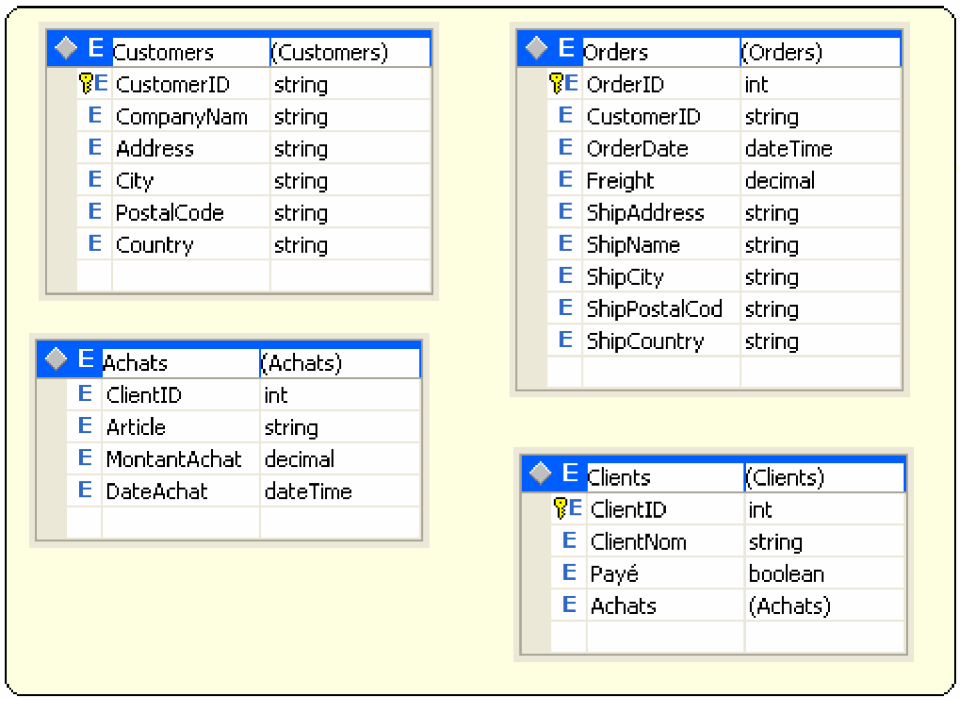
Une **DataRelationCollection** est une collection d'une ou plusieurs **DataRelation** qui représentent chacunes une relation entre deux DataTable dans le DataSet.



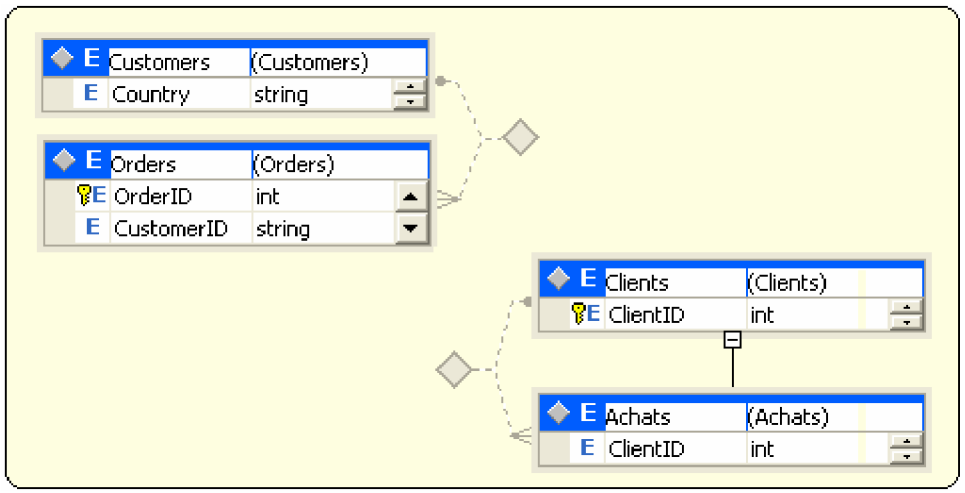
Voici une représentation fictive mais figurative d'un **DataSet** contenant 4 **DataTable** et 2

DataRelation :

La **DataTableCollection** du DataSet :



La **DataRelationCollection** du DataSet :



1. Création et utilisation d'un DataSet

Nous passons maintenant à la pratique d'utilisation d'un DataSet dans un programme C# de création de données selon la démarche suivante :

Phase de mise en place :

- Création d'un DataSet (un magasin).
- Création de 2 DataTable (une table client, une table achats).
- Définition du schéma de colonnes de chaque table par des DataColumn.
- Définition d'une clef primaire.
- Ajout des 2 tables au DataSet.
- Ajout d'une relation entre les deux tables.

Phase de manipulation des données proprement dites :

- Ajouter, supprimer ou modifier des données ligne par ligne dans chaque table.

Création d'un DataSet (un magasin) :

```
private DataSet unDataSet = new DataSet( );
unDataSet.DataSetName = "Magasin" ;
```

Création des deux tables :

```
private DataTable tabClients = new DataTable( "Clients" );
private DataTable tabAchats = new DataTable( "Achats" );
```

Définition du schéma de 3 colonnes de la table Clients :

```
DataColumn colClientID = new DataColumn( "ClientID", typeof(int) );
DataColumn colClientNom = new DataColumn( "ClientNom" );
DataColumn colPayed = new DataColumn( "Payé", typeof(bool) );
tabClients.Columns.Add(colClientID);
tabClients.Columns.Add(colClientNom);
tabClients.Columns.Add(colPayed);
```

Définition du schéma de 4 colonnes de la table Achats :

```
DataColumn colID = new DataColumn("ClientID", typeof(int));
DataColumn colArticle = new DataColumn("Article",typeof(string));
DataColumn colDateAchat = new DataColumn( "DateAchat",typeof(DateTime));
DataColumn colMontantAchat = new DataColumn("MontantAchat", typeof(decimal));
tabAchats.Columns.Add(colID);
tabAchats.Columns.Add(colArticle);
tabAchats.Columns.Add(colMontantAchat);
tabAchats.Columns.Add(colDateAchat);
```

Définition d'une clef primaire composée de 2 colonnes :

```
tabClients.PrimaryKey = new DataColumn[ ] {colClientID, colClientNom};
```

Ajouter les 2 tables au DataSet :

```
unDataSet.Tables.Add(tabClients);
unDataSet.Tables.Add(tabAchats);
```

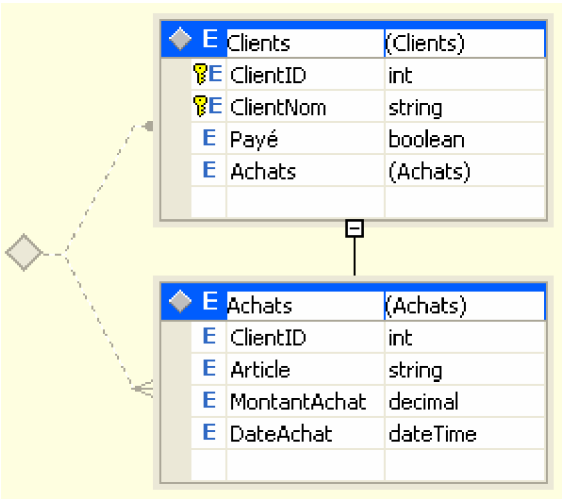
Ajouter une relation "Liste des achats" entre les 2 tables :

```
// Créer une DataRelation entre colClientID et colID
DataRelation dr = new DataRelation("Liste des achats", colClientID , colID);

// ajouter cette relation dans le DataSet
unDataSet.Relations.Add(dr);

// imbrication de relation écrite dans le fichier xml:
dr.Nested=true;
```

La validation de toutes les modifications apportées au DataSet depuis son chargement s'effectue grâce à la méthode **AcceptChanges()**, on termine donc le code par la ligne :

```
unDataSet.AcceptChanges();
```

Le DataSet ainsi construit peut maintenant travailler en lecture et en éciture sur une structure de données construite sous forme de deux tables Clients et Achats reliées entre elles comme figuré ci-desssous :

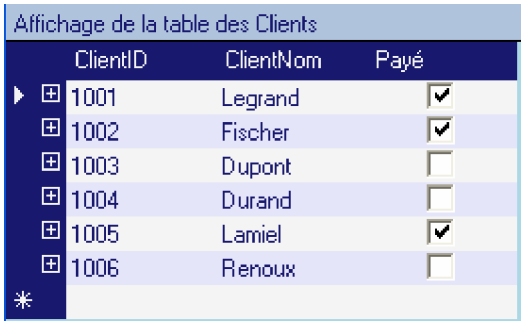


Exemple de création de données dans le DataSet :

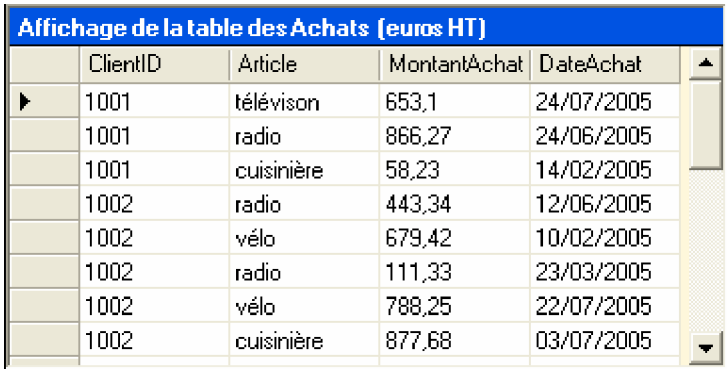
```
enum typArticle {vélo, télévision, radio, cuisinière }

//-- remplissage direct des tables
// Déclaration d'un référence de ligne de données
DataRow newRow1;
// Création de 6 lignes de clients dans la table "Clients"
for(int i = 1; i <= 6; i++) {
  // Création de la ligne newRow1 possédant le même schéma que tabClients
  newRow1 = tabClients.NewRow();
  newRow1["ClientID"] = 1000+i;
  // Ajouter de la ligne newRow1 à la table tabClients
  tabClients.Rows.Add(newRow1);
}
// Remplissage de la colonne "ClientNom" de chaque ligne de clients créée
tabClients.Rows[0]["ClientNom"] = "Legrand";
tabClients.Rows[1]["ClientNom"] = "Fischer";
tabClients.Rows[2]["ClientNom"] = "Dupont";
tabClients.Rows[3]["ClientNom"] = "Durand";
tabClients.Rows[4]["ClientNom"] = "Lamiel";
tabClients.Rows[5]["ClientNom"] = "Renoux";

// Remplissage de la colonne "ClientNom" de chaque ligne de clients créée
tabClients.Rows[0]["Payé"] = true;
tabClients.Rows[1]["Payé"] = true;
tabClients.Rows[2]["Payé"] = false;
tabClients.Rows[3]["Payé"] = false;
tabClients.Rows[4]["Payé"] = true;
tabClients.Rows[5]["Payé"] = false;

// Déclaration d'un référence de ligne de données
DataRow newRow2;

/* pour chacun des 6 clients remplissage aléatoire des 4 colonnes selon
    un maximum de 7 lignes d'achats différents.
*/
Random gen = new Random();
int max ;
do{max = gen.Next(8);}while(max==0);
for(int i = 1; i <= 6; i++){
  for(int j = 1; j < max; j++){
        // création d'une nouvelle ligne d'achats
        newRow2 = tabAchats.NewRow();
        newRow2["ClientID"]= 1000+i;
        newRow2["DateAchat"]= new DateTime(2005, gen.Next(12)+1, gen.Next(29)+1);
        newRow2["MontantAchat"] = Math.Round(gen.NextDouble()*1000,2);
        newRow2["Article"]= Enum.GetName(typeof(typArticle),gen.Next(5));
        // Ajouter la ligne à la table Achats
        tabAchats.Rows.Add(newRow2);
  }
  do{max = gen.Next(8);}while(max==0);
}
```

Affichage des données d'un DataSet

Voici affiché dans un composant visuel de NetFramework1.1 et NetFramework 2.0, le DataGrid, la table des Clients avec ses trois colonnes remplies par le programme précédent :



Voici affiché dans un autre composant visuel DataGrid, la table des Achats avec ses quatre colonnes remplies par le programme précédent :



Le composant DataGrid destiné à afficher une table de données, est remplacé dans la version 2.0 par le DataGridView, mais est tojours présent et utilisable.

```
private System.Windows.Forms.DataGrid  DataGridSimple;
```

Voici deux façons de lier ce composant visuel à la première table (la table Clients) du DataSet du programme de gestion du Magasin :

```
// Lie directement par le nom de la table le DataGrid au DataSet.
DataGridSimple.SetDataBinding(unDataSet, "Clients");

// Lie indirectement par le rang, le DataGrid au DataSet sur la première table :
DataGridSimple.SetDataBinding(unDataSet, unDataSet.Tables[0].TableName);
```

Enfin pour terminer la description des actions pratiques d'un DataSet, indiquons qu'il est possible de sauvegarder le schéma structuré (relation, clef primaire, tables,…) d'un DataSet dans un fichier de schémas au format **XSL**; il est aussi possible de sauvegarder toutes les données et leur structuration dans un fichier au format **XML**.

Sauvegarde des données du DataSet aux formats XML et XSL :

```
// Stockage uniquement du schéma général du DataSet dans un fichier séparé
unDataSet.WriteXmlSchema("Donnees.xsl");

// Stockage à la fois dans un même fichier du schéma et des données
unDataSet.WriteXml("Donnees.xml", XmlWriteMode.WriteSchema);
```

Fichier "Donnees.xsl" obtenu au format XSL

```
<?xml version="1.0" standalone="yes"?>
<xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="fr-FR">
    <xs:complexType>
      <xs:choice maxOccurs="unbounded">
        <xs:element name="Clients">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ClientID" type="xs:int" />
              <xs:element name="ClientNom" type="xs:string" />
              <xs:element name="Payé" type="xs:boolean" minOccurs="0" />
              <xs:element name="Achats" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ClientID" type="xs:int" minOccurs="0" />
                    <xs:element name="Article" type="xs:string" minOccurs="0" />
                    <xs:element name="MontantAchat" type="xs:decimal" minOccurs="0" />
                    <xs:element name="DateAchat" type="xs:dateTime" minOccurs="0" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:choice>
    </xs:complexType>
    <xs:unique name="Constraint1">
      <xs:selector xpath=".//Clients" />
      <xs:field xpath="ClientID" />
    </xs:unique>
    <xs:unique name="Constraint2" msdata:PrimaryKey="true">
      <xs:selector xpath=".//Clients" />
      <xs:field xpath="ClientID" />
      <xs:field xpath="ClientNom" />
    </xs:unique>
    <xs:keyref name="Liste_x0020_des_x0020_achats" refer="Constraint1"
msdata:IsNested="true">
```

```
    <xs:selector xpath=".//Achats" />
    <xs:field xpath="ClientID" />
   </xs:keyref>
  </xs:element>
</xs:schema>
```

Fichier "Donnees.xml" obtenu format XML

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="fr-FR">
      <xs:complexType>
        <xs:choice maxOccurs="unbounded">
          <xs:element name="Clients">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="ClientID" type="xs:int" />
                <xs:element name="ClientNom" type="xs:string" />
                <xs:element name="Payé" type="xs:boolean" minOccurs="0" />
                <xs:element name="Achats" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                    <xs:sequence>
                      <xs:element name="ClientID" type="xs:int" minOccurs="0" />
                      <xs:element name="Article" type="xs:string" minOccurs="0" />
                      <xs:element name="MontantAchat" type="xs:decimal" minOccurs="0" />
                      <xs:element name="DateAchat" type="xs:dateTime" minOccurs="0" />
                    </xs:sequence>
                  </xs:complexType>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
      <xs:unique name="Constraint1">
        <xs:selector xpath=".//Clients" />
        <xs:field xpath="ClientID" />
      </xs:unique>
      <xs:unique name="Constraint2" msdata:PrimaryKey="true">
        <xs:selector xpath=".//Clients" />
        <xs:field xpath="ClientID" />
        <xs:field xpath="ClientNom" />
      </xs:unique>
      <xs:keyref name="Liste_x0020_des_x0020_achats" refer="Constraint1"
msdata:IsNested="true">
        <xs:selector xpath=".//Achats" />
        <xs:field xpath="ClientID" />
      </xs:keyref>
```

```
  </xs:element>
</xs:schema>
<Clients>
  <ClientID>1001</ClientID>
  <ClientNom>Legrand</ClientNom>
  <Payé>true</Payé>
  <Achats>
    <ClientID>1001</ClientID>
    <Article>cuisinière</Article>
    <MontantAchat>456.74</MontantAchat>
    <DateAchat>2005-12-12T00:00:00.0000000+01:00</DateAchat>
  </Achats>
</Clients>
<Clients>
  <ClientID>1002</ClientID>
  <ClientNom>Fischer</ClientNom>
  <Payé>true</Payé>
  <Achats>
    <ClientID>1002</ClientID>
    <Article>radio</Article>
    <MontantAchat>297.58</MontantAchat>
    <DateAchat>2005-02-25T00:00:00.0000000+01:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1002</ClientID>
    <Article>télévison</Article>
    <MontantAchat>715.1</MontantAchat>
    <DateAchat>2005-07-19T00:00:00.0000000+02:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1002</ClientID>
    <Article>télévison</Article>
    <MontantAchat>447.55</MontantAchat>
    <DateAchat>2005-08-16T00:00:00.0000000+02:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1002</ClientID>
    <Article>cuisinière</Article>
    <MontantAchat>92.64</MontantAchat>
    <DateAchat>2005-09-23T00:00:00.0000000+02:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1002</ClientID>
    <Article>cuisinière</Article>
    <MontantAchat>171.07</MontantAchat>
    <DateAchat>2005-01-23T00:00:00.0000000+01:00</DateAchat>
  </Achats>
</Clients>
<Clients>
  <ClientID>1003</ClientID>
  <ClientNom>Dupont</ClientNom>
```

```
  <Payé>false</Payé>
  <Achats>
    <ClientID>1003</ClientID>
    <Article>aspirateur</Article>
    <MontantAchat>445.89</MontantAchat>
    <DateAchat>2005-02-11T00:00:00.0000000+01:00</DateAchat>
  </Achats>
</Clients>
<Clients>
  <ClientID>1004</ClientID>
  <ClientNom>Durand</ClientNom>
  <Payé>false</Payé>
  <Achats>
    <ClientID>1004</ClientID>
    <Article>télévison</Article>
    <MontantAchat>661.47</MontantAchat>
    <DateAchat>2005-11-15T00:00:00.0000000+01:00</DateAchat>
  </Achats>
</Clients>
<Clients>
  <ClientID>1005</ClientID>
  <ClientNom>Lamiel</ClientNom>
  <Payé>true</Payé>
</Clients>
<Clients>
  <ClientID>1006</ClientID>
  <ClientNom>Renoux</ClientNom>
  <Payé>false</Payé>
  <Achats>
    <ClientID>1006</ClientID>
    <Article>cuisinière</Article>
    <MontantAchat>435.17</MontantAchat>
    <DateAchat>2005-06-22T00:00:00.0000000+02:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1006</ClientID>
    <Article>cuisinière</Article>
    <MontantAchat>491.3</MontantAchat>
    <DateAchat>2005-12-25T00:00:00.0000000+01:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1006</ClientID>
    <Article>cuisinière</Article>
    <MontantAchat>388.81</MontantAchat>
    <DateAchat>2005-10-13T00:00:00.0000000+02:00</DateAchat>
  </Achats>
  <Achats>
    <ClientID>1006</ClientID>
    <Article>radio</Article>
    <MontantAchat>864.93</MontantAchat>
    <DateAchat>2005-07-23T00:00:00.0000000+02:00</DateAchat>
```

```
    </Achats>
  </Clients>
</NewDataSet>
```