Chương 4: Viewing .NET data

Tổng quan

Chương cuối cùng sẽ chỉ các cách khác nhau để chọn và thay đổi dữ liệu. Chương này, chúng tôi sẽ minh hoạ cách bạn thể hiện dữ liệu cho người sử dụng thấy bằng cách gắn kết các control *Windows*.

Khả năng gắn kết dữ liệu của .NET giống với ADO và các control của VB. Tất cả ngôn ngữ .NET đều có khả năng sử dụng cùng những Control và phương thức. Khía cạnh mà chúng ta xem xét đó là control *DataGrid*.

Một trong những đặc tính hay nhất của *Datagrid* là tính uyển chuyển- nguồn dữ liệu có thể là mảng, DataTable, DataView, DataSet hay là một thành phần thực thi các giao diện *IListSource* hay *IList*. Với một số lượng lớn các tuỳ chọn sẽ chỉ cách mà mọi nguồn dữ liệu này được sử dụng và được xem trong *DataGrid*.

Gắn kết dữ liệu là một yêu cầu thông thường, và mặc dù VB 6 có khả năng này nhưng không bằng .NET, tất cả ngôn ngữ quản lý đều hoàn thiện khả năng gắn kết dữ liệu. Có những gì hơn khi giới hạn các cột dữ liệu, một control sẽ cập nhật tự động khi hàng dữ liệu hiện tại thay đổi. Chương này chúng ta tìm hiểu một vài khả năng gắn kết dữ liệu và chỉ cách để kết nối dữ liệu với control *Windows Forms*. Chúng ta sẽ xem vài công việc bên trong quá trình gắn kết dữ liệu để hiểu rõ hơn cách chúng hoạt động.

Nguồn dữ liệu sẽ trở nên hợp lý hơn trong Visual studio.NET, và chương này sẽ chỉ cách sử dụng *server Explorer* để tạo một sự kết nối và phát sinh *Dataset*. Chúng ta sẽ tìm hiểu cách dùng của lược đồ XSD trong visual studio.NET.

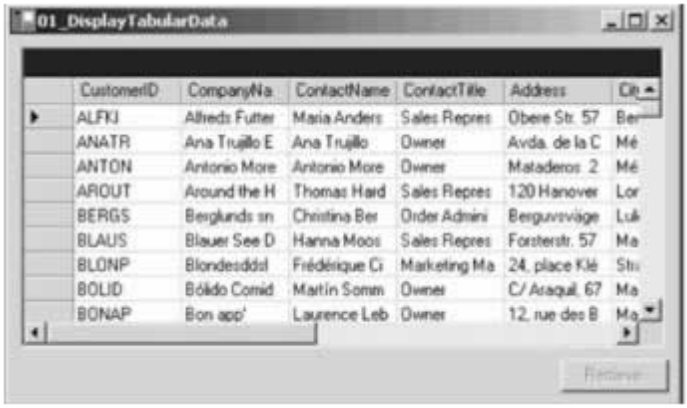
4.1 The Control DataGrid

DataGrid là một control mới hoàn toàn, được viết cho các ngôn ngữ .NET, và nó cho phép có những cái nhìn khác nhau về dữ liệu được hiển thị. Bạn có thể hiển thị dữ liệu bằng cách gọi phương thức *SetDataBinding()*.

4.1.1 Hiển thị dữ liệu xếp theo cột:

Phần cuối của chương sẽ trình bày nhiều cách chọn dữ liệu và lấy nó trong một bảng dữ liệu, mặc dù dữ liệu được hiển thị trong một kiểu rất cơ bản; chúng ta chỉ đơn giản dùng *Console.WriteLine()*

Ví dụ đầu tiên ở đây sẽ chỉ cách khôi phục dữ liệu và hiển thị trong một control *DataGrid*. Hình bên dưới là màn hình từ ứng dụng đã được xây dựng:



Ứng dụng này chọn mỗi phần tử từ bản *Customer* trong cơ sở dữ liệu *NorthWind* và hiển thị những phần tử này cho người dùng trong một *DataGrid*. Đoạn mã này khá ngắn, ta sẽ từng bước xem xét chúng như sau:

```
using System;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
```

```
public class DisplayTabularData : System.Windows.Forms.Form
{
  private System.Windows.Forms.Button retrieveButton;
  private System.Windows.Forms.DataGrid dataGrid;
  public DisplayTabularData()
  {
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(464, 253);
    this.Text = "01_DisplayTabularData";
```

Tiếp đến, ta sẽ tạo control khung lưới(grid), và cài đặt những thuộc tính của nó. Dòng thứ hai: dataGrid.BeginInit(); chỉ được dùng khi tạo nhiều sự thay đổi trên control. Nếu các sự kiện không giới hạn, mọi thay đổi trên khung lưới có thể tạo ra một *Redraw* trên màn hình. Sau đó ta xác định vị trí và kích thước của Control, định nghĩa chỉ mục tab, và neo control vào cả hai góc trên bên trái và góc dưới bên phải của cửa sổ để nó cân xứng trong cửa sổ ứng dụng chính.

```
    this.dataGrid = new System.Windows.Forms.DataGrid();
    dataGrid.BeginInit();
    dataGrid.Location = new System.Drawing.Point(8, 8);
    dataGrid.Size = new System.Drawing.Size(448, 208);
    dataGrid.TabIndex = 0;
    dataGrid.Anchor = AnchorStyles.Bottom | AnchorStyles.Top |
                AnchorStyles.Left | AnchorStyles.Right;
    this.Controls.Add(this.dataGrid);
    dataGrid.EndInit();
```

Bây giờ ta tạo nút. Cùng với những bước cơ bản theo sau trong việc khởi tạo nút:

```
        this.retrieveButton = new System.Windows.Forms.Button();
        retrieveButton.Location = new System.Drawing.Point(384, 224);
    retrieveButton.Size = new System.Drawing.Size(75, 23);
    retrieveButton.TabIndex = 1;
    retrieveButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;
    retrieveButton.Text = "Retrieve";
    retrieveButton.Click += new System.EventHandler
                        (this.retrieveButton_Click);
    this.Controls.Add(this.retrieveButton);
```

Chúng ta có một sự kiện *click* gọi bộ điều khiển sự kiện *retrieveButton_click*

```
protected void retrieveButton_Click(object sender, System.EventArgs e)
  {
    retrieveButton.Enabled = false;
    string source = "server=(local)\\NetSDK;" +
                "uid=QSUser;pwd=QSPassword;" +
                "database=Northwind";
```

Sau khi chọn dữ liệu từ bảng *Customer* và điền dữ liệu vào tập dữ liệu. Ta gọi phương thức *SetĐataBlinding* để gắn kết dữ liệu giữa tập dữ liệu và khung lưới. Phương thức này sẽ được truyền vào tập dữ liệu và tên của bảng trong *DataSet*. Một khung lưới có thể chỉ hiện dữ liệu từ một *DataTable* tại một thời điểm mặc dù *DataSet* chứa nhiều bảng.

```
    string select = "SELECT * FROM Customers" ;
    SqlConnection conn = new SqlConnection(source);
    SqlDataAdapter da = new SqlDataAdapter( select , conn);
    DataSet ds = new DataSet();
    da.Fill(ds , "Customers");
```

```
    dataGrid.SetDataBinding(ds , "Customers");
  }
  static void Main()
  {
    Application.Run(new DisplayTabularData());
  }
}
```

Để biên dịch đoạn mã trên bạn gõ dòng lệnh sau:

```
csc /t:winexe /debug+ /r:System.dll /r:System.Data.dll /r:system.windows.forms.dll
  /recurse:*.cs
```

Tham số */recurse:*.cs* sẽ biên dịch tất cả tập tin .cs trong thư mục hiện hành và các thư mục con. Đó là một cách viết tắt, nên bạn không cần phải nhớ tất cả các tập tin nhưng bạn phải chắc chắn là chỉ có những tập tin bạn cần nằm trong thư mục đó.

4.1.2 Nguồn dữ liệu:

DataGrid là một cách rất linh động để hiển thị dữ liệu; thêm vào đó là để gọi phương thức *SetDataBlinding()* với một *DataSet* và tên của bảng để hiển thị thì phương thức này sẽ được gọi với bất kỳ nguồn dữ liệu sau:

- Một mảng
- Datatable
- DataView
- DataSet hay DataViewManager
- Những thành phần thực thi giao diện IListSource
- Những thành phần thực thi giao diện IList

4.1.2.1 Hiển thị dữ liệu từ một mảng:

Nhìn thoáng qua có vẽ rất dễ dàng. Tạo một mảng, điền dữ liệu vào mảng và gọi phương thức *SetDataBlinding(array,null)* trên *DataGrid*. Như ví dụ sau:

```
string[] stuff = new string[] {"One", "Two", "Three"};
dataGrid.SetDataBinding(stuff, null);
```

Chú ý rằng phương thức *SetDataBlinding()* chỉ có hai tham số, tham số đầu là nguồn dữ liệu trong trường hợp này là mảng, tham số còn lại: nếu nguồn dữ liệu là *DataSet* hay *DataViewMannager* thì gán bằng tên của bảng muốn hiển thị còn ngược lại được gán giá trị *null*.

Bạn có thể thay thế đoạn mã trong bộ điều khiển sự kiện *retriveButton_click()* của ví dụ trước với đoạn mã ở trên. Kết quả hiển thị sẽ có vấn đề sau:

Bạn sẽ thấy kết quả hiển thị có nhiều hơn số chuỗi mà bạn định nghĩa trong mảng, Khung lưới sẽ hiển thị thêm chiều dài của những chuỗi đó. Nguyên nhân là khi sử dụng mảng là nguồn dữ liệu của *Datagrid* thì khung lưới sẽ tìm thuộc tính chung đầu tiên của đối tượng bên trong mảng để hiển thị và trường hợp này đó chính là chiều dài của chuỗi đó.

Một cách giải quyết đó là tạo một lớp bao bọc cho các chuỗi này như bên dưới:

```
protected class Item
{
  public Item(string text)
  {
    m_text = text;
  }
  public string Text
  {
    get{return m_text;}
  }
  private string m_text;
}
```

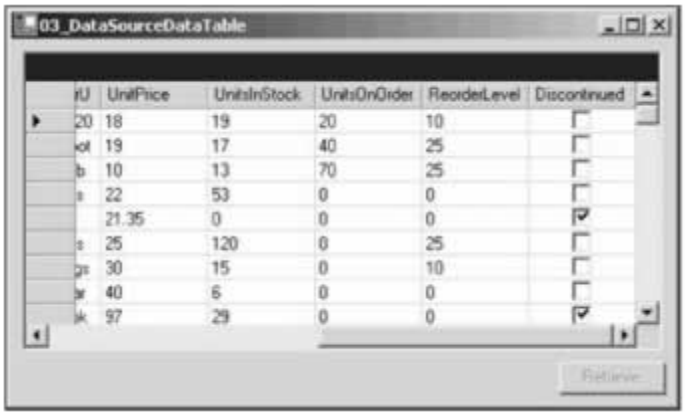
4.1.2.2 DataTable

Có hai cách chính để hiển thị một *DataTable* trong một *DataGrid*:

- Nếu *DataTable* của bạn đứng một mình thì gọi phương thức *SetDataBlingding(DataTable,null)*

- Nếu *DataTable* của bạn chứa một *DataSet* thì gọi *SetDataBlinding(DataSet,"<Table Name>")*

Ví dụ bên dưới để hiện một số cột:



Chú ý hiển thị của cột cuối cùng; nó hiện một checkbox để thay cho các control thông thường. *DataGrid* sẽ đọc lượt đồ từ nguồn dữ liệu và suy ra các kiểu cột mà control được hiển thị.

Dữ liệu trong cơ sở dữ liệu không thay đổi khi bạn thay đổi các trường trong khung lưới dữ liệu.

4.1.2.3 Hiển thị dữ liệu từ một DataView

Một *DataView* cung cấp phương tiện để lọc và sắp xếp dữ liệu bên trong một *DataTable*. Khi bạn chọn một dữ liệu từ một cơ sở dữ liệu, thông thường nó cho phép người dùng sắp xếp dữ liệu đó. Thêm vào đó, bạn muốn lọc dữ liệu để chỉ hiện những hàng nào đó. Một *DataView* cho phép bạn giới hạn số hàng hiển thị cho người dùng nhưng nó không giới hạn số cột trong *DataTable*.

Một DataView không cho phép bạn thay đổi các cột để hiển thị mà chỉ thay đổi các hàng.

Bên dưới đây là một dòng mã để tạo một *DataView* dựa trên một *DataTable* đang tồn tại:

```
DataView dv = new DataView(dataTable);
```

Khi tạo bạn có thể thay đổi các cài đặt trên *DataView*, và nó sẽ ảnh hưởng đến dữ liệu và các tác vụ khi chúng được hiển thị bên trong một *DataGrid*. Một vài ví dụ là:

- Cài đặt AllowEdit = false khoá chức năng chỉnh sửa các dòng
- Cài đặt AllowNew = false khoá chức năng tạo dòng mới
- Cài đặt AllowDelete = false khoá khả năng xoá dòng
- Cài đặt RowStateFilter chỉ hiển thị những dòng của một trạng thái được cho
- Cài đặt RowFilter để lọc hàng
- Xắp xếp các dòng bằng các cột nào đó

4.1.2.4 Lọc các hàng bằng dữ liệu:

Khi bạn tạo một *DataView*, bạn có thể thay đổi dữ liệu hiển thị bằng cách cài đặt thuộc tính *RowFilter*. Thuộc tính này như một chuỗi được dùng như một phương tiện để lọc trên những tiêu chuẩn nào đó- giá trị của chuỗi được dùng như tiêu chuẩn lọc.

Vài ví dụ về các mệnh đề lọc được chỉ trong bảng dưới đây

Mệnh đề	**Mục đích**
UnitsInStock > 50	Chỉ những hàng có UnitslnStock lớn hơn 50
Client = 'Smith'	Chỉ trả về những mẫu cho một client
County LIKE 'C*'	Trả về tất cả mẫu mà trường Country bắt đầu ký tự C

4.1.2.5 Lọc các hàng trên trạng thái

Mọi hàng trên *DataView* có một tập định nghĩa hàng, nó sẽ là một trong những giá trị sau. Tập này có thể được dùng để lọc những hàng được xem bởi người dùng:

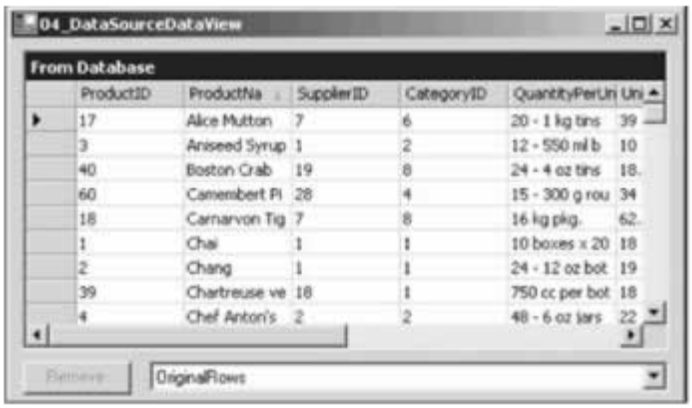
DataViewRowState	**Mục đích**
Added	Tất cả hàng được tạo mới
CurrentRows	Tất cả hàng ngoại trừ những hàng được chọn sẽ bị xoá
Deleted	tất cả hàng được chọn bị xoá
ModifiedCurrent	Dãy tất cả hàng bị sửa đổi và hiện giá trị hiện hành cho mọi cột
ModifiedOriginal	Dãy tất cả hàng bị sửa đổi nhưng hiện giá trị ban đầu cho các cột chứ không phải giá trị hiện hành
OriginalRows	Tất cả hàng được chọn ban đầu từ nguồn dữ liệu .Không có những hàng mới. Chỉ hiện những giá trị ban đầu của các cột.
Unchanged	Tất cả các hàng không thể bị thay đổi

Để xem ảnh hưởng của các trạng thái này trên một khung lưới, ta viết một ví dụ hiển thị hai khung lưới: một chứa dữ liệu được chọn từ cơ sở dữ liệu mà bạn có thể tương tác, cái còn lại hiện các hàng trong một trong những trạng thái trên.

Bộ lọc không chỉ áp dụng đối với các hàng mà còn với trạng thái của các cột bên trong những hàng này.

4.1.2.6 Sắp xếp các hàng:

Khi lọc dữ liệu, đôi lúc bạn cần sắp xếp dữ liệu trong một *Dataview*. Bạn có thể click trên tiêu đề của cột trong control *DataGrid*, và nó sẽ sắp xếp một cột theo thứ tự giảm dần hoặc tăng dần. Tuy nhiên bạn chỉ có thể sắp xếp một cột, những nơi nào có *Dataview* gạch dưới thì có thể sắp xếp theo nhiều cột.



Khi một cột được sắp xếp, *DataGrid* sẽ hiển thị một mũi tên để cho biết cột đã được sắp xếp.

Để thực hiện sắp xếp trên một cột bạn sử dụng thuộc tính *Sort* của *DataView*:

```
dataView.Sort = "ProductName";
dataView.Sort = "ProductName ASC, ProductID DESC";
```

Dòng đầu tiên sẽ được sắp xếp theo cột *ProductName*. Dòng thứ hai sẽ được sắp xếp thứ tự tăng dần theo cột *ProductName* và sau đó theo thứ tự giảm dần của *ProductID*.

Dataview hỗ trợ sắp xếp tăng dần và giảm dần trên các cột. Nếu bạn chọn sắp xếp trên nhiều cột thì *DataGrid* sẽ ngừng hiện các mũi tên sắp xếp.

4.1.2.7 Hiển thị dữ liệu từ một DataSet

Ở ví dụ trước, *DataGrid* chỉ có thể hiển thị một *DataTable* đơn tại một thời điểm. Nhưng ở ví dụ này, nó có thể điều khiển nhiều mối quan hệ trong *DataSet* trên màn hình. Đoạn mã sau được dùng để tạo ra một *DatasSet* dựa trên các bảng *Customers* và *Orders* trong cơ sở dữ liệu *Northwind*. Ví dụ này thêm hai *DataTable* và tạo một mối quan hệ giữa chúng gọi là *CustomerOrders*:

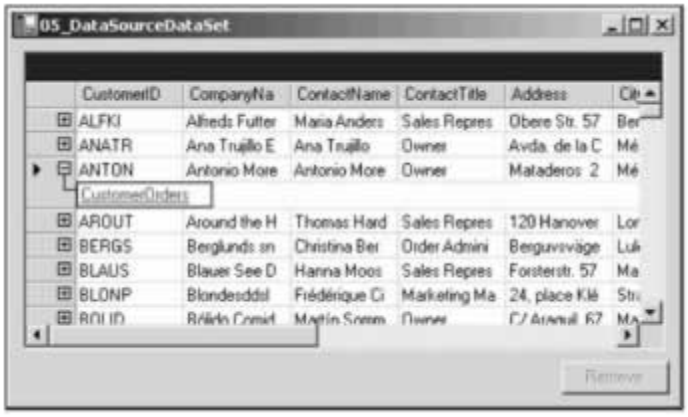
```
string source = "server=(local)\\NetSDK;" +
            "uid=QSUser;pwd=QSPassword;" +
            "database=northwind";
string orders = "SELECT * FROM Orders";
string customers = "SELECT * FROM Customers";
SqlConnection conn = new SqlConnection(source);
SqlDataAdapter da = new SqlDataAdapter(orders, conn);
DataSet ds = new DataSet();
da.Fill(ds, "Orders");
da = new SqlDataAdapter(customers , conn);
```

```
da.Fill(ds, "Customers");
ds.Relations.Add("CustomerOrders",
        ds.Tables["Customers"].Columns["CustomerID"],
        ds.Tables["Orders"].Columns["CustomerID"]);
```

Khi tạo, bạn có thể liên kết *DataSet* với *DataGrid* bằng cách gọi phương thức *SetDataBinding*:

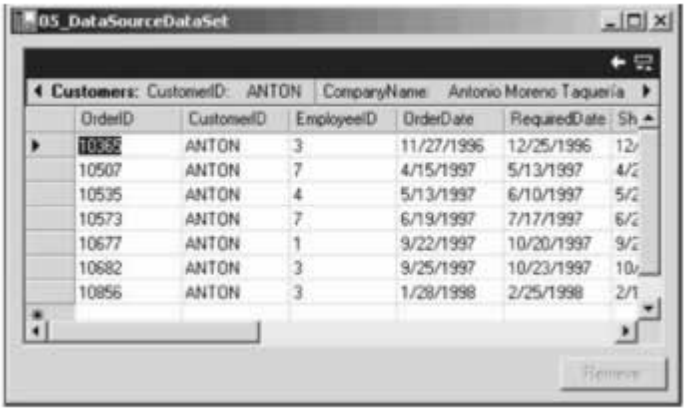
```
dataGrid1.SetDataBinding(ds, "Customers");
```

Nó sẽ tạo một hiển thị như sau:



Bạn chú ý có một dấu + bên trái mỗi mẫu tin. Để biết rằng chúng ta đã tạo một *Dataset* với một mối quan hệ điều khiển giữa *customers* và *orders*. Bạn có thể định nghĩa nhiều mối quan hệ trong đoạn mã.

Khi bạn click trên các dấu +, một danh sách các mối quan hệ hiện ra, click trên tên của mối quan hệ sẽ điều khiển khung lưới liên kết với các mẫu tin.



Control *DataGrid* bao gồm một cặp biểu tượng mới ở góc trên bên phải. Mũi tên cho phép bạn quay lại hàng cha mẹ, và sẽ thay đổi hiển thị đến trang trước đó. Tiêu đề của hàng hiện chi tiết các mẫu tin cha mẹ có thể hiện hay ẩn bằng cách click trên những nút khác.

4.1.2.8 Hiển thị dữ liệu trong một DataViewManager

Hiển thị dữ liệu trong một *DataViewManager* thì giống như *DataSet*. Nhưng khi một *DataViewManager* được tạo cho một *DataSet* thì một *DataView* đặc biệt được tạo ra cho mỗi *DataTable*, cho phép bạn có thể thay đổi hiển thị hàng, dựa vào một bộ lọc hay trạng thái hàng. Nếu bạn không muốn lọc dữ liệu, bạn sẽ đề nghị luôn luôn bao một *DataSet* trong một *DataViewManager* để hiển thị. Nó cho bạn nhiều tuỳ chọn khi sửa đổi mã của bạn.

Đoạn mã dưới tạo một *DataViewManager* dựa trên *DataSet* từ ví dụ trước, và sau đó thay đổi *DataView* cho bảng *Customers* để chỉ hiện customers từ UK:

```
DataViewManager dvm = new DataViewManager(ds);
dvm.DataViewSettings["Customers"].RowFilter = "Country='UK'";
dataGrid.SetDataBinding(dvm, "Customers");
```

Kết quả hiển thị sẽ như sau:

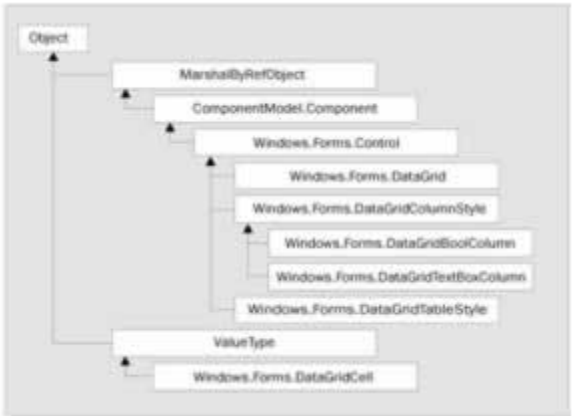
4.1.2.9 Giao diện IListSource và IList

DataGrid cũng hỗ trợ bất kỳ đối tượng mà đưa vào một trong những giao diện *IListSource* hay *IList*. I*ListSource* chỉ có một phương thức *GetList()* trả về một giao diện *IList*. *IList* được thực thi bởi rất nhiều lớp trong thời gian chạy. Vài lớp thực thi giao diện này là *Array, ArrayList, StringCollection.*

Khi sử dụng *IList*, cùng điều kiện cho đối tượng bên trong tập hợp là true thì sự thực thi *Array* sẽ hiện dễ dàng hơn- Nếu sử dụng một *StringCollection* như nguồn dữ liệu cho *DataGrid* thì chiều dài của chuỗi được hiện bên trong khung lưới.

4.1.3 Thừa kế lớp DataGrid

Thừa kế lớp cho những phần chính của DataGrid được hiện bên dưới:



DataGrid bao gồm 0 hay nhiều *DataGridTableStyles*. Những kiểu này bao gồm 0 hay nhiều *DataGridColumnStyles*. Một ô trong khung lưới có thể được truy cập bởi nhiều phương tiện của cấu trúc *DatagridCell*.

4.1.3.1 DataGridTableStyle và DataGridColumnStyle

Một *DataGridTableStyle* chứa sự miêu tả trực quan của *DataTable*. *DataGrid* chứa một tập hợp những kiểu này có thể truy cập được bằng thuộc tính *TableStype*. Khi một *DataTable* được hiển thị thì một sự kiểm tra được tạo xuyên qua tất cả đối tượng *DataGridTableStype* để tìm thuộc tính *MappingName* của nó bằng với thuộc tính *TableName* của *DataTable*. Trong khi tìm kiếm, kiểu đó sẽ được dùng trong việc hiển thị của bảng.

DataGridTableStyle cho phép bạn định nghĩa những biến hiện hình cho *DataGrid*, như là màu nền và màu cận cảnh, font dùng trong tiêu đề cột và các thuộc tính khác. *DataGridColumnStyle* cho phép bạn lọc những tuỳ chọn hiển thị trên một cột, như là cài đặt trật tự của dữ liệu trong cột, văn bản được hiển thị một giá trị *null* và chiều rộng của cột trên màn hình.

Khi *DataGrid* hiển thị một định nghĩa *DataGridTableStype*, bạn có thể định nghĩa các cột của dữ liệu được hiển thị bằng cách thêm một *DataGridColumnStype*. Chỉ những cột có một kiểu định nghĩa sẽ được hiển thị và có thể là lợi ích cho những cột ẩn như là giá trị của khoá chính không được hiển thị. Bạn cũng có thể định nghĩa một kiểu cột *ReadOnly*.

Đoạn mã bên dưới là ví dụ của việc tạo một *DataGridTableStyle*. Đoạn mã tạo ra một đối tượng *DataGridTableStyle*, thêm vào hai đối tượng *DataGridColumnStyle*, và hiển thị tất cả dữ liệu bên trong bảng *Customer*.

```
using System;
using System.Windows.Forms;
```

```
using System.Data;
using System.Data.SqlClient;
public class CustomDataGridTableStyle : System.Windows.Forms.Form
{
  private System.Windows.Forms.Button retrieveButton;
  private System.Windows.Forms.DataGrid dataGrid;
  public CustomDataGridTableStyle()
  {
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(464, 253);
    this.Text = "07_CustomDataGridTableStyle";
    this.dataGrid = new System.Windows.Forms.DataGrid();
    dataGrid.BeginInit();
    dataGrid.Location = new System.Drawing.Point(8, 8);
    dataGrid.Size = new System.Drawing.Size(448, 208);
    dataGrid.TabIndex = 0;
    dataGrid.Anchor = AnchorStyles.Bottom | AnchorStyles.Top |
                AnchorStyles.Left | AnchorStyles.Right;
    this.Controls.Add(this.dataGrid);
    dataGrid.EndInit();
    this.retrieveButton = new System.Windows.Forms.Button();
    retrieveButton.Location = new System.Drawing.Point(384, 224);
    retrieveButton.Size = new System.Drawing.Size(75, 23);
    retrieveButton.TabIndex = 1;
    retrieveButton.Anchor = AnchorStyles.Bottom | AnchorStyles.Right;
    retrieveButton.Text = "Retrieve";
    retrieveButton.Click += new
                System.EventHandler(this.retrieveButton_Click);
    this.Controls.Add(this.retrieveButton);
```

```
}
protected void retrieveButton_Click(object sender, System.EventArgs e)
{
  retrieveButton.Enabled = false;
```

DataSet sẽ được dùng tạo ra *DataGridTableStyles* để dùng trong ví dụ này và cuối cùng liên kết *DataGrid* với *DataSet*. Phương thức *CreateDataSet* không có gì mới như chúng ta thấy sau, nó chỉ đơn giản nhận tất cả hàng từ bảng *Customers*:

```
  DataSet ds = CreateDataSet();
  CreateStyles(dataGrid);
  dataGrid.SetDataBinding(ds, "Customers");
}
```

Phương thức *CreateStyles()* có nhiều đặc biệt. Dòng đầu tiên tạo đối tượng *DataGridTableStype* và cài thuộc tính *MappingName* của nó. Thuộc tính này được sử dụng khi *DataGrid* hiển thị một *DataTable*. *DataGrid* có thể hiển thị hàng trong những màu thay đổi. Đoạn mã ở đây định nghĩa màu theo từng cặp hàng.

```
private void CreateStyles(DataGrid dg)
{
  DataGridTableStyle style = new DataGridTableStyle();
  style.MappingName = "Customers";
  style.AlternatingBackColor = System.Drawing.Color.Bisque;
  DataGridTextBoxColumn customerID = new DataGridTextBoxColumn();
  customerID.HeaderText = "Customer ID";
  customerID.MappingName = "CustomerID";
  customerID.Width = 200;
  DataGridTextBoxColumn name = new DataGridTextBoxColumn();
  name.HeaderText = "Name";
```

```
    name.MappingName = "CompanyName";
    name.Width = 300;
```

Khi các cột được định nghĩa, chúng được thêm vào *GridColumnStypes* bộ các đối tượng *DataGridTableStype*, các đối tượng này có thể tự thêm thuộc tính *TableStype* của *DataGrid*:

```
    style.GridColumnStyles.AddRange
            (new DataGridColumnStyle[]{customerID , name});
    dg.TableStyles.Add(style);
  }
  private DataSet CreateDataSet()
  {
    string source = "server=(local)\\NetSDK;" +
            "uid=QSUser;pwd=QSPassword;" +
            "database=northwind";
    string customers = "SELECT * FROM Customers";
    SqlConnection con = new SqlConnection(source);
    SqlDataAdapter da = new SqlDataAdapter(customers , con);
    DataSet ds = new DataSet();
    da.Fill(ds, "Customers");
    return ds;
  }
  static void Main()
  {
    Application.Run(new CustomDataGridTableStyle());
  }
}
```

Sau khi tạo đối tượng *DataGridTableStyle*, chúng ta tạo hai đối tượng thừa hưởng từ *DataGridColumnStyle*. Mọi cột có một số lượng thuộc tính được định nghĩa. Sau đây là một danh sách các thuộc tính khoá:

Property	**Description**
Alignment	một trong những giá trị liệt kê HorizontalAlignment - Left, Center, or Right. Nó cho biết cách mà dữ liệu trong cột được định nghĩa hợp lý.
FontHeight	Kích cỡ của font theo pixels. Nó sẽ mặc định nếu không có giá trị được cài. Thuộc tính này được bảo vệ, vì thế có thể chỉ sửa đổi nếu bạn tạo lớp con.
HeaderText	Văn bản hiển thị trong cột heading.
MappingName	Cột trong DataTable mô tả bởi cột hiển thị
NullText	Văn bản hiển thị bên trong cột nếu giá trị dữ liệu nằm dưới là DBNull.
PropertyDescriptor	Nó sẽ được bàn luận phần của chương
ReadOnly	Một cờ cho biết cột là read-write or read-only.
Width	Chiều rộng của cột theo pixels.

Kết quả của đoạn mã hiện như sau:

4.2 Gắn kết dữ liệu

Ở ví dụ trước đã xem xét tất cả control *DataGrid*, đó chỉ là một phần trong thời gian chạy.NET có thể dùng để hiển thị dữ liệu. Một tiến trình gắn kết một control và một nguồn dữ liệu được gọi là **data binding**.

Nếu bạn có những kinh nghiệm với các ứngdụng lập trình Windows trong MFC. Có lúc nào đó bạn đã sử dụng chức năng **Dialog Data Exchange** (**DDX**)để móc các biến thành viên của một lớp với bộ điều khiển Win32. Bạn sẽ vui sướng khi biết rằng bạn có thể giấu cửa trên DDX, như nó dễ dàng hơn để móc dữ liệu vào bộ điều khiển trong .NET. Bạn có thể gắn kết dữ liệu không chỉ đến các bộ điều khiển Window mà còn với các trang Web ASP.NET.

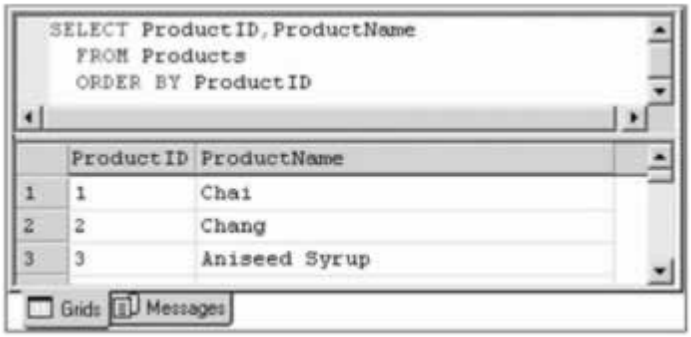
4.2.1 Gắn kết đơn giản

Một control hỗ trợ việc gắn kết đơn hiển thị chỉ những giá trị đơn tại một lúc, như là một hộp văn bản hay một nút chọn. Ví dụ sau chỉ cách gắn kết một cột từ một *DataTable* đến một hộp văn bản.

```
DataSet ds = CreateDataSet();
textBox1.DataBindings.Add("Text", ds , "Products.ProductName");
```

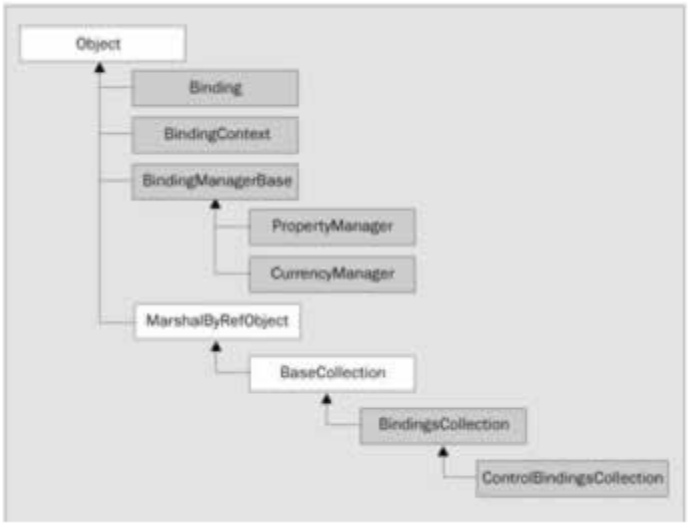
Sau khi lấy lại vài dữ liệu từ bảng Products và lưu trữ trong một *DataSet* được trả về từ phương thức *CreateDataSet()* như trên, dòng thứ hai gắn kết thuộc tính *Text* của control đến cột *Products.ProductName*. Nếu bạn viết đoạn mã này từ cơ sở dữ liệu *Northwind*, bạn sẽ thấy màn hình như bên dưới đây:

Hộp văn bản hiển thị vài thứ trong cơ sở dữ liệu. Để kiểm tra rằng nó là cột hay giá trị, bạn sẽ sử dụng công cụ SQL Server Query Analyzer để kiểm tra nội dung của bảng *Procucttool*.



4.2.2 Đối tượng gắn kết dữ liệu

Sơ đồ sau chỉ một thừa kế lớp cho các đối tượng được sử dụng trong gắn kết dữ liệu. Trong phần này ta bàn luận về *BindingContext*, *CurrencyManager*, và *PropertyManager* các lớp của *System.Windows.Forms*, và trình cách chúng tương tác khi dữ liệu giới hạn trong một hay nhiều control trên một form. Các đối tượng chuyển màu được dùng trong gắn kết.



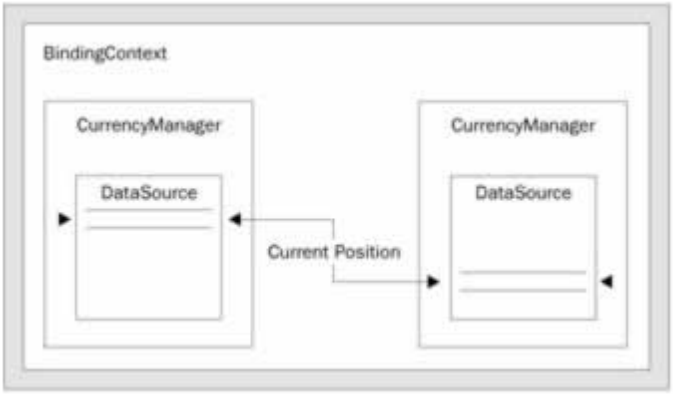
Trong ví dụ trước, chúng ta sử dụng thuộc tính *DataBinding* của control *TextBox* để gắn kết một cột từ một *DataSet* đến thuộc tính *Text* của bộ điều khiển. Thuộc tính *DataBindings* là một thể hiện của *ControlBindingsCollection* :

```
textBox1.DataBindings.Add("Text", ds, "Products.ProductName");
```

Dòng này thêm một đối tượng gắn kết từ một đối tượng *Binding* đến *ControlBindingsCollection*

4.2.2.1 Binding Context

Mọi Windows form có một thuộc tính *BindingContext*. Form được thừa hưởng từ Control . Một đối tượng *BindingContext* có một tập thể hiện *BindingManagerBase*. Những thể hiện này được tạo và thêm vào đối tượng quản lý gắn kết khi một control bị giới hạn:



BindingContext sẽ chứa vài nguồn dữ liệu, được gói trong một *CurrencyManager* hay một *PropertyManager*. Sự quyết định lớp nào được dùng dựa vào chính nguồn dữ liệu.

Nếu nguồn dữ liệu chứa một dãy item như là *DataTable*, *DataView,* hay bất kỳ đối tượng khác thực thi giao diện *IList* thì một *CurrencyManager* sẽ được dùng, như nó có thể duy trì vị trí hiện tại bên trong nguồn dữ liệu. Nếu nguồn dữ liệu chỉ trả về một giá trị đơn thì một *PropertyManager* sẽ được lưu trữ trong *BindingContext*.

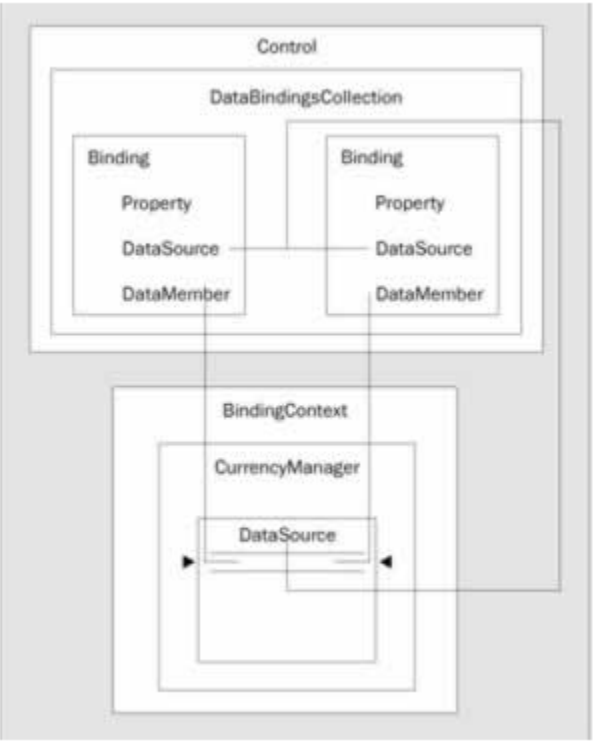
Một *CurrencyManager* hay *PropertyManager* chỉ được tạo một lần cho một nguồn dữ liệu. Nếu bạn gắn kết hai hộp văn bản với một hàng từ một *DataTable* thì chỉ một *currencyManage*r sẽ được tạo bên trong binding context.

Mọi control thêm vào một form được gắn kết với bộ quản lý gắn kết của form, vì thế tất cả control chia sẽ cùng một thể hiện. Khi một control được tạo thuộc tính *BindingContext* của nó là *null*. Khi control được thêm bộ Control của form thì nó sẽ cài *BindingContext* đến bộ đó của form.

Để gắn kết một control với một form, bạn cần thêm một thực thể vào thuộc tính DataBinding của nó. Đoạn mã bên dưới tạo một sự gắn kết mới:

```
textBox1.DataBindings.Add("Text", ds, "Products.ProductName");
```

Phương thức *Add()* của *ControlBindingsCollection* tạo một thể hiện mới của đối tượng Binding từ những thông số của phương thưc này và thêm chúng vào bộ những việc gắn kết



Hình trên trình bày những gì đang hoạt động khi bạn thêm một Binding đến một Control. Binding gắn kết control với một nguồn dữ liệu được duy trì bên trong *BindingContext* của Form. Sự thay đổi bên trong nguồn dữ liệu được phản ánh vào control như là những thay đổi trong control đó.

4.2.2.2 Binding

Lớp này gắn kết một thuộc tính của control với một thành viên của nguồn dữ liệu. Khi những thành viên này thay đổi thì những thuộc tính của control được cập nhật để phản ánh sự thay đổi này và ngược lại

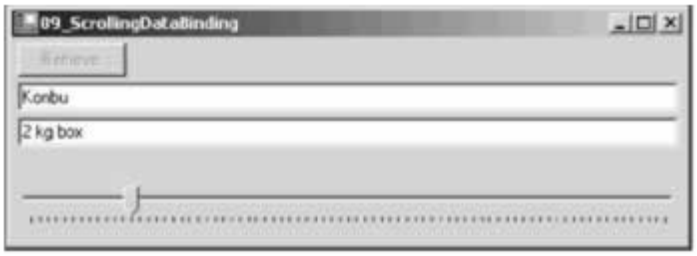
Bindings có thể cài đặt từ bất kỳ cột nào đến bất kỳ thuộc tính nào của control, vì thế bạn sẽ gắn kết một cột với một hộp văn bản và có thể gắn kết cột khác với màu hộp văn bản..

Bạn có thể gắn kết các thuôc tính của một control đến các nguồn dữ liệu khác nhau.

4.2.2.3 CurrencyManager và PropertyManager

Khi một đối tượng Binding được tạo, một đối tượng *CurrencyManager* hay *PropertyManager* sẽ được tạo nếu đó là lần đầu tiên dữ liệu đó từ nguồn bị giới hạn. Mục đích của lớp này là định nghĩa vị trí của mẫu tin hiện hành trong nguồn dữ liệu và kết hợp tất cả dãy bindings khi mẫu tin hiện hành này bị thay đổi.

Ví dụ sau sẽ hiển thị hai trường từ bảng *Product* và bao gồm một cách để di chuyển giữa các mẫu tin bằng các phương tiện của một control *TrackBar*.



Đoạn mã :

```
using System;
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;

public class ScrollingDataBinding : System.Windows.Forms.Form
{
```

```
private Button retrieveButton;
private TextBox textName;
private TextBox textQuan;
private TrackBar trackBar;
private DataSet ds;
```

Ứng dụng trên tạo cửa sổ và tất cả control cho cửa sổ đó bên trong một hàm khởi tạo *ScrollingDataBinding*:

```
public ScrollingDataBinding()
{
  this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
  this.ClientSize = new System.Drawing.Size(464, 253);
  this.Text = "09_ScrollingDataBinding";
  this.retrieveButton = new Button();
  retrieveButton.Location = new System.Drawing.Point(4, 4);
  retrieveButton.Size = new System.Drawing.Size(75, 23);
  retrieveButton.TabIndex = 1;
  retrieveButton.Anchor = AnchorStyles.Top | AnchorStyles.Left;
  retrieveButton.Text = "Retrieve";
  retrieveButton.Click += new System.EventHandler
                    (this.retrieveButton_Click);
  this.Controls.Add(this.retrieveButton);
  this.textName = new TextBox();
  textName.Location = new System.Drawing.Point(4, 31);
  textName.Text = "Please click retrieve...";
  textName.TabIndex = 2;
  textName.Anchor = AnchorStyles.Top | AnchorStyles.Left |
            AnchorStyles.Right ;
```

```
    textName.Size = new System.Drawing.Size(456, 20);
    textName.Enabled = false;
    this.Controls.Add(this.textName);
    this.textQuan = new TextBox();
    textQuan.Location = new System.Drawing.Point(4, 55);
    textQuan.Text = "";
    textQuan.TabIndex = 3;
    textQuan.Anchor = AnchorStyles.Top | AnchorStyles.Left |
                      AnchorStyles.Top;
    textQuan.Size = new System.Drawing.Size(456, 20);
    textQuan.Enabled = false;
    this.Controls.Add(this.textQuan);
    this.trackBar = new TrackBar();
    trackBar.BeginInit();
    trackBar.Dock = DockStyle.Bottom ;
    trackBar.Location = new System.Drawing.Point(0, 275);
    trackBar.TabIndex = 4;
    trackBar.Size = new System.Drawing.Size(504, 42);
    trackBar.Scroll += new System.EventHandler(this.trackBar_Scroll);
    trackBar.Enabled = false;
    this.Controls.Add(this.trackBar);
  }
```

Khi nút *Retrieve* được click, sự kiện handler chọn tất cả mẫu tin từ bảng Product và lưu trữ trong *Dataset* riêng ds:

```
  protected void retrieveButton_Click(object sender, System.EventArgs e)
  {
    retrieveButton.Enabled = false ;
```

```
    ds = CreateDataSet();
```

Tiếp theo là hai control văn bản được giới hạn

```
    textName.DataBindings.Add("Text" , ds ,
                        "Products.ProductName");
    textQuan.DataBindings.Add("Text" , ds ,
                        "Products.QuantityPerUnit");
    trackBar.Minimum = 0 ;
    trackBar.Maximum = this.BindingContext[ds,"Products"].Count – 1;
    textName.Enabled = true;
    textQuan.Enabled = true;
    trackBar.Enabled = true;
  }
```

Ở đây chúng ta có một mẫu tin cuộn để phản ứng lại với sự di chuyển của TrackBar:

```
  protected void trackBar_Scroll(object sender , System.EventArgs e)
  {
    this.BindingContext[ds,"Products"].Position = trackBar.Value;
  }
  private DataSet CreateDataSet()
  {
    string source = "server=(local)\\NetSDK;" +
                "uid=QSUser;pwd=QSPassword;" +
                "database=northwind";
    string customers = "SELECT * FROM Products";
    SqlConnection con = new SqlConnection(source);
    SqlDataAdapter da = new SqlDataAdapter(customers , con);
```

```
    DataSet ds = new DataSet();
    da.Fill(ds , "Products");
    return ds;
  }
  static void Main()
  {
    Application.Run(new ScrollingDataBinding());
  }
}
```

Khi dữ liệu được khôi phục, vị trí lớn nhất trên track bar được cài là số lượng của mẫu tin. Sau đó, trong phương thức scroll ở trên, chúng ta cài vị trí của *BindingContext* cho *DataTable* products đến vị trí của scroll bar thumb. Nó thay đổi mẫu tin hiện hành từ *DataTable*, vì thế tất cả control giới hạn đến hàng hiện hành.