Chương 10

Truy xuất database trong chương trình VC#

10.0 Dẫn nhập

Chương này giới thiệu cách thức dùng các đối tượng trong thư viện ADO .Net để truy xuất database dễ dàng, tin cậy.

Chương này cũng giới thiệu cách thức dùng khả năng databinding của các đối tượng giao diện trong môi trường VS .Net để xây dựng chương trình truy xuất database được dễ dàng, nhanh chóng, tin cậy, và nhiều trường hợp không cần viết code cho chương trình.

10.1 Tổng quát về truy xuất database

Mục tiêu của chương trình là xử lý các dữ liệu của nó. Dữ liệu của chương trình có thể rất nhiều và đa dạng phong phú về tính chất. Trong chương 7, chúng ta đã giới thiệu cách lập trình để ghi/đọc dữ liệu cổ điển hay đối tượng ra/vào file.

Hầu hết các ứng dụng hiện nay (nhất là các ứng dụng nghiệp vụ) đều phải truy xuất dữ liệu rất lớn. Thí dụ chương trình quản lý công dân Việt Nam phải xử lý hàng trăm triệu hồ sơ chứa thông tin về các công dân.

Việc xử lý dữ liệu bao gồm nhiều tác vụ như tạo file mới với cấu trúc record cụ thể, thêm/bớt/hiệu chỉnh/duyệt các record, tìm kiếm các record thỏa mãn 1 tiêu chuẩn nào đó,... Để thực hiện các tác vụ trên (nhất là tìm kiếm record thỏa mãn 1 số tiêu chuẩn nào đó) hiệu quả, tin cậy, ta cần nhiều kiến thức khác nhau và phải tốn nhiều công sức.

Hiện nay các record dữ liệu có cùng cấu trúc (thí dụ như các record sinh viên) cần xử lý của chương trình thường được lưu giữ trong 1 bảng dữ liệu (table). Nhiều bảng dữ liệu có mối quan hệ lẫn nhau được chứa trong 1 database quan hệ. Có nhiều định dạng

database quan hệ khác nhau đang được dùng như FoxPro, Access, SQL Server, MySQL, Oracle...

Để giải phóng ứng dụng khỏi các chi tiết quản lý database, người ta đã xây dựng ứng dụng đặc biệt : DBMS (Database Management System).

Mỗi DBMS cung cấp ít nhất 1 Provider. Provider là module phần mềm cung cấp các hàm chức năng để chương trình ứng dụng gọi khi cần thiết hầu truy xuất dữ liệu trong database mà không cần biết chi tiết về cấu trúc vật lý của các record dữ liệu trong database.

Mỗi lần cần truy xuất dữ liệu trong database, ứng dụng sẽ nhờ DBMS thực hiện dùm thông qua việc dùng 1 trong các cấp dịch vụ sau đây (từ cao xuống thấp) :

- Các lệnh truy vấn của ngôn ngữ SQL
- Các đối tượng trong thư viện ADO .Net (ActiveX Data Objects)
- Các đối tượng trong thư viện ADO (ActiveX Data Objects)
- Các đối tượng trong thư viện DAO (Data Access Objects)
- Các hàm trong thư viện ODBC (Open Database Connectivity)

Ngôn ngữ truy vấn SQL là ngôn ngữ phi thủ tục, nó cung cấp 1 tập các lệnh SQL rất mạnh và dễ dàng dùng để xử lý database. Thí dụ để tìm tất cả sinh viên nam quê ở Bến tre đang theo học tại trường Bách Khoa Tp.HCM, ta chỉ cần dùng 1 lệnh SQL như sau :

```
Select * from Sinhvien where Phai = 1 and Quequan = 71
```

Thư viện ADO .Net cung cấp 1 số đối tượng để giúp người lập trình truy xuất database rất dễ dàng thông qua mô hình hướng đối tượng.

Ngôn ngữ VC# cho phép ta kết hợp 2 cấp truy xuất database dễ dàng, đơn giản nhất : dùng các đối tượng ADO .Net để thực hiện các lệnh truy vấn SQL.

10.2 Truy xuất database thông qua ADO .Net

Các đối tượng ADO .Net được tổ chức theo từng namespace, mỗi namespace chứa đối tượng dùng cùng Provider truy xuất database :

- System.Data.OleDb chứa các đối tượng ADO .Net để truy xuất database do bộ Microsoft Office quản lý như Visual FoxPro, Access, Excel,...
- System.Data.Sql và System.Data.SqlClient chứa các đối tượng ADO .Net để truy xuất database do serer "SQLServer" quản lý.
- System.Data.Odbc chứa các đối tượng ADO .Net để truy xuất database thông qua chuẩn giao tiếp ODBC. Hầu hết các hệ quản trị database (DBMS) hiện nay đều hỗ trợ chuẩn giao tiếp này.
- ...

Trong từng namespace, để lập trình truy xuất database, ta thường dùng các đối tượng ADO .Net chính yếu sau đây :

1. preConnection, trong đó pre là phần tiếp đầu ngữ miêu tả tên namespace như OleDb, Obdc, Sql, SqlClient,... Đối tượng này có chức năng quản lý cầu nối đến nguồn database mà chúng ta cần truy xuất.
2. preCommand có chức năng quản lý lệnh truy vấn SQL mà ta cần thực hiện.
3. preDataReader cho phép duyệt đọc/xử lý các record từ 1 bảng dữ liệu.

4. preDataAdapter quản lý 1 tập các lệnh truy vấn và 1 connection tới nguồn database để cho phép việc đọc/ghi dữ liệu.
5. ...

10.3 Thí dụ lập trình dùng ADO .Net

Thí dụ trong một tổ chức quản lý việc nhập/xuất/tồn các sản phẩm, ta dùng 1 database quản lý dữ liệu. Database chứa 3 bảng dữ liệu sau đây :

- SPNhap chứa số lượng các sản phẩm nhập, mỗi record có các field như MaSP, Soluong,...
- SPXuat chứa số lượng các sản phẩm xuất, mỗi record có các field như MaSP, Soluong,...
- SPTon chứa số lượng các sản phẩm tồn kho, mỗi record có các field như MaSP, Soluong,...

Thường thì người làm công tác nghiệp vụ sẽ thực hiện việc cập nhật bảng sản phẩm nhập, bảng sản phẩm xuất theo thời gian. Chúng ta hãy viết chương trình tạo bảng sản phẩm tồn theo nội dung hiện hành của 2 bảng sản phẩm nhập/xuất.

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Console Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. TaoSPTon), click button OK để tạo Project theo các thông số đã khai báo.
3. Ngay sau khi Project vừa được tạo ra, cửa sổ soạn code cho chương trình được hiển thị. Thêm lệnh using sau đây vào đầu file :

   using System.data.OleDb;

4. Viết code cho thân hàm Main như sau :

```
static void Main(string[] args) {
    //định nghĩa các biến đối tượng cần dùng
    String ConnectionString;
    OleDbConnection cn;
    OleDbCommand cmd = new OleDbCommand();
    //xây dựng chuỗi đặc tả database cần truy xuất
    ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=d:\\MyDatabase.mdb;";
    //tạo đối tượng Connection đến database & mở Connection
    cn = new OleDbConnection(ConnectionString);
    cn.Open();
    //cấu hình cho đối tượng Command
    cmd.Connection = cn;
    //thực hiện lệnh SQL xóa bảng SPTon nếu đã có
    cmd.CommandText = "Drop Table SPTon";
    try { cmd.ExecuteNonQuery(); }
    catch { }
    //thực hiện lệnh SQL tạo bảng sản phẩm tồn
    cmd.CommandText = "Create Table SPTon(MaSP Text, Soluong int)";
    cmd.ExecuteNonQuery();
    //thực hiện lệnh SQL tạo các sản phẩm tồn có MaSP tồn tại trong bảng
SPNhap
    cmd.CommandText = "Insert into SPTon select SPNhap.MaSP,
IIf(IsNull(SPNhap.Soluong), 0, SPNhap.Soluong)-
IIf(IsNull(SPXuat.Soluong), 0, SPXuat.Soluong) as Soluong from
SPXuat right join SPNhap on SPXuat.MaSP = SPNhap.MaSP";
    cmd.ExecuteNonQuery();
    //thực hiện lệnh SQL tạo bảng sản phẩm Tam
    cmd.CommandText = "Create Table Tam(MaSP Text, Soluong int)";
    cmd.ExecuteNonQuery();
    //thực hiện lệnh SQL tạo các sản phẩm tồn có MaSP tồn tại trong bảng
SPXuat
```

```
    cmd.CommandText = "Insert into Tam select SPXuat.MaSP,
IIf(IsNull(SPNhap.Soluong), 0, SPNhap.Soluong) -
IIf(IsNull(SPXuat.Soluong), 0, SPXuat.Soluong) as Soluong from
SPNhap right join SPXuat on SPXuat.MaSP = SPNhap.MaSP";
    cmd.ExecuteNonQuery();
    //Trộn 2 bảng kết quả lại thành bảng SPTon
    cmd.CommandText = "Insert into SPTon select MaSP, Soluong
from Tam where Soluong < 0";
    cmd.ExecuteNonQuery();
    //Xóa bảng Tam
    cmd.CommandText = "Drop Table Tam";
    cmd.ExecuteNonQuery();
}
```

5. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng.
6. Sau khi ứng dụng chạy xong, chạy ứng dụng Access, mở file database, kiểm tra nội dung bảng SPTon để đánh giá kết quả có đúng yêu cầu không.

10.4 Databinding (Kết nối động đến dữ liệu)

Khi viết chương trình truy xuất database có giao diện đồ họa trực quan, chúng ta thường phải thực hiện các chức năng chính :

- thiết kế các đối tượng giao diện để giúp người dùng tương tác với database.
- viết code thiết lập các thông tin về database, về các dữ liệu cần truy xuất.
- mỗi lần record trên database bị thay đổi nội dung (do bị xử lý bên trong, do bị thay đổi vị trí truy xuất), ta phải viết code đọc thông tin của record hiện hành và hiển thị lên các đối tượng giao diện tương ứng.
- mỗi lần người dùng cập nhật nội dung trong các đối tượng giao diện, ta phải viết code đọc thông tin của các đối tượng

giao diện và ghi lên record tương ứng trên database để đảm bảo tính nhất quán dữ liệu.

Trong 4 công việc cần thực hiện trong slide trước thì công việc 3 và 4 khá nặng nề. VC# cung cấp khả năng Databinding để giúp người lập trình không cần thực hiện 2 công việc nặng nề này.

Thật vậy, hầu hết các đối tượng giao diện có sẵn trong framework .Net như TextBox, ListBox, ComboBox, TreeView, DataGridView,... đều có thuộc tính DataBindings. Nó giúp ta chỉ cần thiết lập sự kết hợp giữa đối tượng giao diện với dữ liệu nào đó trong database để khi chương trình hoạt động, máy sẽ tự hiển thị thông tin từ database lên đối tượng giao diện và mỗi khi nội dung đối tượng giao diện bị thay đổi bởi người dùng, máy cũng sẽ tự động ghi lên database để đảm bảo tính nhất quán dữ liệu theo suốt thời gian làm việc.

Việc thiết lập sự kết hợp giữa đối tượng giao diện với dữ liệu trong database có thể thực hiện bằng thiết kế trực quan hay viết code tường minh.

Nếu ta thiết lập sự kết hợp giữa đối tượng giao diện với dữ liệu trong database bằng thiết kế trực quan thì sẽ dẫn đến kết quả hết sức hấp dẫn : xây dựng chương trình truy xuất database mà không cần viết code nào cả. Thí dụ trong mục 10.5 sẽ cho thấy kết luận này.

Tuy nhiên để chủ động hơn trong việc xác lập mối quan hệ ràng buộc giữa các nội dung hiển thị trong các đối tượng giao diện, ta thường sẽ viết đoạn code để thiết lập sự kết hợp giữa đối tượng giao diện với dữ liệu trong database cũng như các ràng buộc giữa chúng. Thí dụ trong mục 10.6 sẽ cho thấy kết luận này.

10.5 Thí dụ về databinding mà không viết code

Trong bộ Microsoft Office, Microsoft có cung cấp 1 database Access có tên là NorthWind.mdb, database này chứa thông tin

quản lý của 1 siêu thị điện tử ảo, gồm nhiều bảng dữ liệu, trong đó có bảng Customers miêu tả các khách hàng của siêu thị.

Chúng ta hãy thử viết 1 chương trình đơn giản có chức năng hiển thị thông tin về các khách hàng trong bảng Customers cho người dùng xem.

Sau đây là qui trình điển hình để xây dựng chương trình theo yêu cầu trên mà không cần viết code cho chương trình.

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. DBAccess), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lặp 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.
4. Duyệt tìm phần tử DataGridView (trong nhóm Data), chọn nó, dời chuột vào trong Form và vẽ nó với kích thước mong muốn (chiếm hết form). Hiệu chỉnh thuộc tính (Name) = grdCustomers.
5. Ngay sau khi vẽ xong DataGridView, máy sẽ hiển thị cửa sổ "DataGridView Tasks". Nếu sơ xuất làm mất nó thì bạn hãy click chuột vào button nhỏ ở phía trên phải DataGridView để hiển thị lại. Click chuột vào icon chỉ xuống trong listbox "Choose data source" để hiển thị cửa sổ hỗ trợ. Click chuột vào mục "Add Project Data Source", để hiển thị cửa sổ "Choose a Data Source type". chọn icon Database rồi click button Next để hiển thị cửa sổ "Choose Your Database Connection".

6. Click button New Connection để hiển thị cửa sổ "Add Connection", xác định Provider truy xuất database, thí dụ như Provider "Microsoft Access Database File (OLE DB)" để truy xuất file Access, provider "Microsoft SQL Server" để truy xuất database do SQL Server quản lý...
7. Xác định database cần truy xuất trong "Database file name" rồi click button OK để quay về cửa sổ trước. Click buttonNext để hiển thị cửa sổ "Choose Your Database Object".
8. Mở rộng mục Tables để hiển thị đầy đủ các tên bảng dữ liệu có trong database, duyệt tìm và đánh dấu chọn vào bảng Customers rồi click button Finish để hoàn tất việc khai báo trực quan.
9. Bây giờ chương trình đã được viết xong. hãy chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Cửa sổ ứng dụng sẽ hiển thị đối tượng DataGridView, đối tượng này hiển thị đầy đủ danh sách thông tin các khách hàng trong bảng Customers, người dùng có thể "scroll" lên/xuống hay trái/phải để xem thông tin thích hợp.

10.6 Thí dụ về databinding có viết code khởi tạo

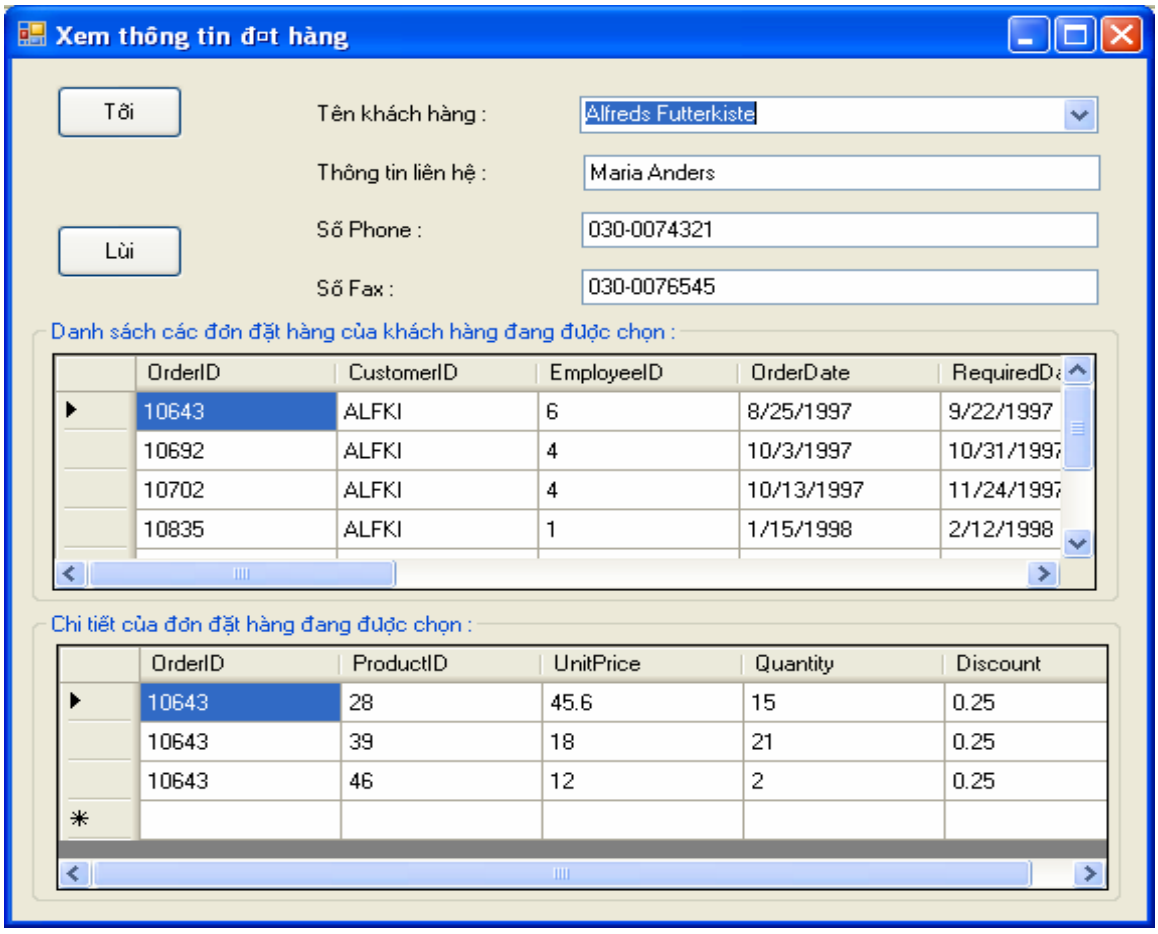
Trong bộ Microsoft Office, Microsoft có cung cấp 1 database Access có tên là NorthWind.mdb, database này chứa thông tin quản lý của 1 siêu thị điện tử ảo, gồm nhiều bảng dữ liệu và mối quan hệ giữa chúng :

- Customers : là bảng dữ liệu miêu tả các khách hàng của siêu thị, mỗi khác hàng có các field thông tin như CustomerID, CustomerName,...
- Orders : là bảng dữ liệu miêu tả các đơn đặt hàng của các khách hàng, mỗi đơn đặt hàng có các field thông tin như CustomerID, OrderID,...

- OrderDetails : là bảng dữ liệu miêu tả nội dung chi tiết của từng đơn đặt hàng, có các field thông tin như OrderID, ProductID,...

Chúng ta hãy viết chương trình cho phép người dùng xem thông tin mua hàng của các khách hàng, mỗi thời điểm xem chi tiết 1 khách hàng cần khảo sát, người dùng có thể dời tới/dời lui để xem thông tin khách hàng kế cận, người dùng có thể chọn ngẫu nhiên 1 khách hàng để xem thông tin. Thông tin chi tiết về khách hàng gồm tên, địa chỉ liên hệ, số phone, số fax, danh sách các đơn đặt hàng đã đặt, nội dung chi tiết của từng đơn đặt hàng.

Sau khi phân tích kỹ yêu cầu của chương trình trên, ta thiết kế được nội dung chi tiết của form giao diện của chương trình như sau :



Vai trò cụ thể của các đối tượng giao diện như sau :

- 2 Button : cho phép người dùng tiến tới/lùi 1 khách hàng.

- ComboBox : hiển thị tên khách hàng đang chọn khảo sát, nó còn cho phép người dùng chọn ngẫu nhiên 1 khác hàng khác nếu muốn.
- 3 textbox : hiển thị các thông tin về khác hàng như địa chỉ liên hệ, số phone, số fax.
- 1 DataGridView : hiển thị danh sách chi tiết về các đơn hàng của khách hàng đang quan tâm.
- 1 DataGridView : hiển thị danh sách chi tiết về các mặt hàng của đơn hàng đang quan tâm.

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. DBAccess), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lặp 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.
4. Nếu cửa sổ ToolBox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Thay đổi kích thước của form lớn ra theo yêu cầu.
5. Duyệt tìm phần tử Button (trong nhóm Common Controls), chọn nó, dời chuột về vị trí thích hợp trong form và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Tới" và thuộc tính (Name) = btnToi.
6. Lặp lại bước 5 để vẽ Button thứ 2 với thuộc tính Text = "Lùi" và thuộc tính (Name) = btnLui.

7. Duyệt tìm phần tử Label (trong nhóm Common Controls), chọn nó, dời chuột về vị trí thích hợp trong form và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Tên khách hàng :".
8. Lặp lại bước 7 để vẽ 3 Label còn lại với thuộc tính Text tuần tự là "Địa chỉ liên hệ :", "Số Phone : ", "Số Fax :"
9. Duyệt tìm phần tử ComboBox (trong nhóm Common Controls), chọn nó, dời chuột về bên phải Label "Tên khách hàng :" và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính (Name) = cbCust.
10. Duyệt tìm phần tử TextBox (trong nhóm Common Controls), chọn nó, dời chuột về vị trí bên phải Label "Địa chỉ liên hệ :" và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính (Name)= txtContact.
11. Lặp lại bước 10 để vẽ 2 TextBox còn lại với thuộc tính (Name) tuần tự là txtPhoneNo, txtFaxNo.
12. Duyệt tìm phần tử GroupBox (trong nhóm Common Controls), chọn nó, dời chuột về vị trí ngay dưới Label "Số Fax :" và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Danh sách các đơn đặt hàng của khách hàng đang được chọn :".
13. Duyệt tìm phần tử DataGridView (trong nhóm Data), chọn nó, dời chuột vào trong GroupBox vừa vẽ và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính (Name)= grdOrders.
14. Lặp lại bước 12 và 13 để vẽ GroupBox có thuộc tính Text = "Chi tiết của đơn đặt hàng đang được chọn :" và DataGridView bên trong có thuộc tính (Name) = grdOrderDetails.
15. Tạo hàm xử lý sự kiện cho 2 button btnToi và btnLui rồi viết code cho chúng như sau :

```
//hàm xử lý Click chuột trên button "Tới"
private void btnToi_Click(object sender, EventArgs e) {
   CurrencyManager cm = (CurrencyManager)this.BindingContext[
dsView, "Customers"];
   //nếu không phải khách hàng cuối thì tiến tới 1 khách hàng
   if (cm.Position < cm.Count - 1) cm.Position++;
}
//hàm xử lý Click chuột trên button"Lùi"
private void btnLui_Click(object sender, EventArgs e) {
   //nếu không phải khách hàng đầu tiên thì lùi 1 khách hàng
   if (this.BindingContext[dsView, "Customers"].Position > 0)
      this.BindingContext[dsView, "Customers"].Position--;
}
```

16. Thêm lệnh using sau vào đầu file đặc tả class Form :

```
using System.Data.OleDb;
```

17. Thêm các lệnh định nghĩa các thuộc tính dữ liệu cần dùng sau đây vào ở vị trí đầu class đặc tả Form :

```
   //định nghĩa các thuộc tính dữ liệu cần dùng
   private String ConnectionString;
   private DataViewManager dsView;
   private DataSet ds;
   private OleDbConnection cn;
```

18. Hiệu chỉnh lại hàm constructor của Form để có nội dung như sau:

```
 public Form1() {
   InitializeComponent();
   //xây dựng chuỗi đặc tả database cần truy xuất
   ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=c:\\NorthWind.mdb;";
   //tạo đối tượng Connection đến database
   cn = new OleDbConnection(ConnectionString);
   //tạo đối tượng DataSet
   ds = new DataSet("CustOrders");
```

```
    //tạo đối tượng DataApdater quản lý danh sách các khách hàng
    OleDbDataAdapter da1 = new OleDbDataAdapter
        ("SELECT * FROM Customers", cn);
    //ánh xạ Tablename "Table" tới bảng dữ liệu "Customers"
    da1.TableMappings.Add("Table","Customers");
    //chứa bảng Customers vào Dataset
    da1.Fill(ds);
    //tạo đối tượng DataApdater quản lý danh sách các đơn đặt
hàng
    OleDbDataAdapter da2 = new OleDbDataAdapter
        ("SELECT * FROM Orders", cn);
    //ánh xạ Tablename "Table" tới bảng dữ liệu "Orders"
    da2.TableMappings.Add("Table","Orders");
    // chứa bảng Orders vào Dataset
    da2.Fill(ds);
    //tạo đối tượng DataApdater quản lý danh sách các mặt hàng
    OleDbDataAdapter da3 = new OleDbDataAdapter
        ("SELECT * FROM [Order Details]", cn);
    //ánh xạ Tablename "Table" tới bảng dữ liệu "Orders"
    da3.TableMappings.Add("Table","OrderDetails");
    // chứa bảng [Orders Details] vào Dataset
    da3.Fill(ds);
    //thiết lập quan hệ "RelCustOrd" giữa bảng Customers và
Orders
    System.Data.DataRelation relCustOrd;
    System.Data.DataColumn  colMaster1;
    System.Data.DataColumn  colDetail1;
    colMaster1 = ds.Tables["Customers"].Columns["CustomerID"];
    colDetail1 = ds.Tables["Orders"].Columns["CustomerID"];
    relCustOrd = new System.Data.DataRelation
    ("RelCustOrd",colMaster1,colDetail1);
    //"add" quan hệ vừa tạo vào dataSet
    ds.Relations.Add(relCustOrd);
```

```
    //thiết lập quan hệ "relOrdDet" giữa bảng Orders & [Order Details]
    System.Data.DataRelation relOrdDet;
    System.Data.DataColumn  colMaster2;
    System.Data.DataColumn  colDetail2;
    colMaster2 = ds.Tables["Orders"].Columns["OrderID"];
    colDetail2 = ds.Tables["OrderDetails"].Columns["OrderID"];
    relOrdDet = new DataRelation("RelOrdDet",colMaster2,colDetail2);
    //"add" quan hệ vừa tạo vào dataSet
    ds.Relations.Add(relOrdDet);
    //Xác định DataViewManager của DataSet.
    dsView = ds.DefaultViewManager;
    //thiết lập Databinding giữa database với 2 DataGridView
    grdOrders.DataSource = dsView;
    grdOrders.DataMember = "Customers.RelCustOrd";
    grdOrderDetails.DataSource = dsView;
    grdOrderDetails.DataMember = "Customers.RelCustOrd.RelOrdDet";
    //thiết lập Databinding giữa database với ComboBox
    cbCust.DataSource = dsView;
    cbCust.DisplayMember = "Customers.CompanyName";
    cbCust.ValueMember = "Customers.CustomerID";
    //thiết lập Databinding giữa database với 3 Textbox
    txtContact.DataBindings.Add("Text",dsView,"Customers.ContactName");
    txtPhoneNo.DataBindings.Add("Text",dsView,"Customers.Phone");
    txtFaxNo.DataBindings.Add("Text",dsView,"Customers.Fax");
}
```

19. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Lúc đầu, form sẽ hiển thị thông tin về khách hàng đầu tiên trong bảng, khi bạn click vào button "Tới" hay "Lùi", thông tin khách hàng tương ứng sẽ tự được hiển thị. Bạn cũng có thể chọn 1 khách hàng tùy ý trong ComboBox "Tên khách hàng :" để chương trình tự hiển thị thông tin chi tiết về khách hàng đó.

20. Tóm lại Databinding trong VC# giúp ta giảm nhẹ rất nhiều công sức viết chương trình truy xuất database : chúng ta chỉ viết đoạn code thiết lập databinding giữa các đối tượng giao diện với dữ liệu tương ứng trong database chứ chúng ta không cần viết đoạn code cập nhật nội dung của các phần tử giao diện theo sự biến động của database, chúng ta cũng không cần viết code cập nhật database theo nội dung mà người dùng thay đổi trên các đối tượng giao diện.

10.7 Kết chương

Chương này đã giới thiệu cách thức dùng các đối tượng trong thư viện ADO .Net để truy xuất database dễ dàng, tin cậy.

Chương này cũng đã giới thiệu cách thức dùng khả năng databinding của các đối tượng giao diện trong môi trường VS .Net để xây dựng chương trình truy xuất database được dễ dàng, nhanh chóng, tin cậy, và nhiều trường hợp không cần viết code cho chương trình.