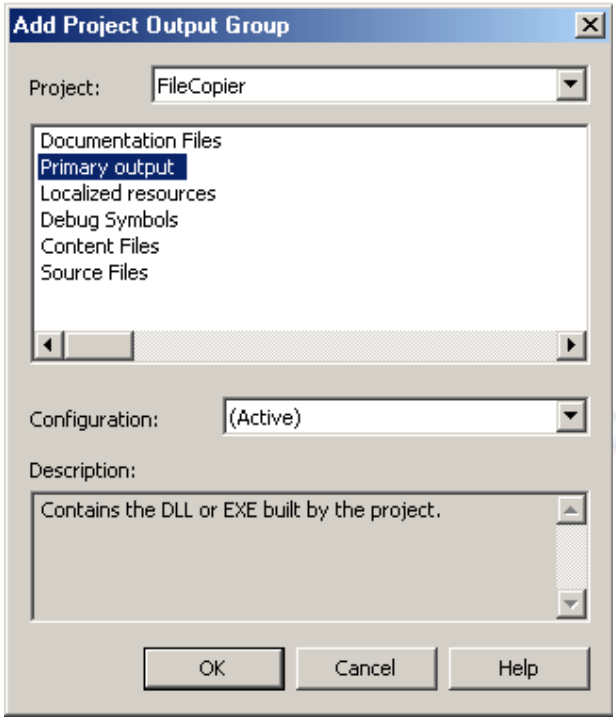
Hình 13-13 Lựa chọn loại kết xuất để đóng gói.



Ở đây, ta sẽ chọn loại **Primary Output** để tạo tập tin FileCopier.exe cho ứng dụng **FileCopier** của ta. Khi chạy chương trình thì .NET sẽ tạo ra một gói có tên là **FileCopierCabProject.CAB.** Trong gói này có chứa 2 tập tin :

- FileCopier.exe : tập tin chạy của ứng dụng
- Osd8c0.osd : tập tin này mô tả gói .CAB theo dạng XML.

Mã mô tả XML tập tin .CAB

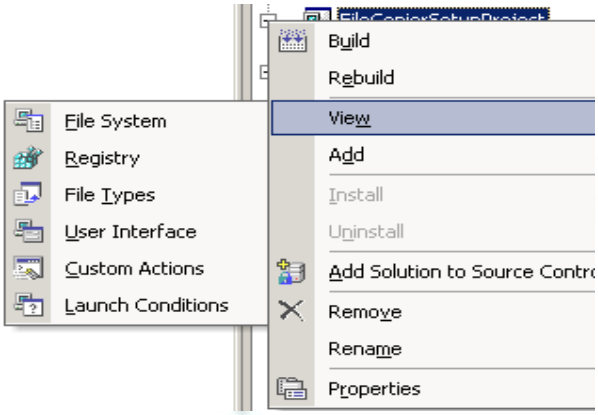
```
<?XML version="1.0" ENCODING='UTF-8'?>
<!DOCTYPE SOFTPKG SYSTEM
"http://www.microsoft.com/standards/osd/osd.dtd">
<?XML::namespace
href="http://www.microsoft.com/standards/osd/msicd.dtd"
        as="MSICD"?>
<SOFTPKG NAME="FileCopierCabProject" VERSION="1,0,0,0">
<TITLE> FileCopierCabProject </TITLE>
<MSICD::NATIVECODE>
        <CODE NAME="FileCopier">
          <IMPLEMENTATION>
           <CODEBASE FILENAME="FileCopier.exe">
           </CODEBASE>
     </IMPLEMENTATION>
        </CODE>
     </MSICD::NATIVECODE>
</SOFTPKG>
```

13.1.4.2 Việc cài đặt dự án (Setup Project)

Để tạo được chương trình tự động cài đặt cho ứng dụng, ta thêm dự án khác và chọn kiểu là **Setup Project,** dự án kiểu này khá linh động. Nó tạo thành tập tin có phần mở rộng là **MSI**, có thể tự cài đặt ứng dụng của ta lên máy tính khác.

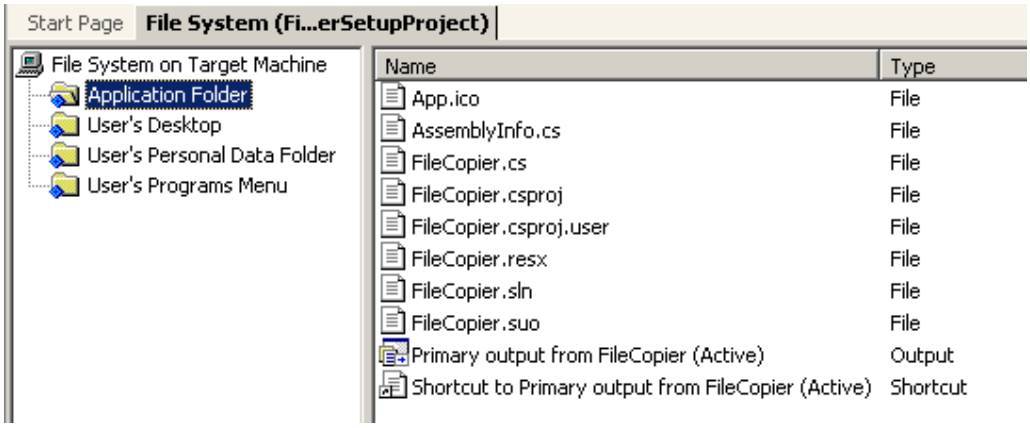
UI phục vụ chủ yếu cho việc cài đặt là cửa số **View Menu** :

Hình 13-14 Giao diện người dùng View Menu.



Ứng với mỗi chọn lựa trên View Menu, ta sẽ thấy các hiển thị tương ứng trong hai cửa sổ bên trái. Chẳng hạn như, khi ta chọn **View \ File System** thì ở bên trái sẽ hiện ra hai cửa sổ như như hình bên dưới, ta có thể thêm các tập tin hay tài nguyên liên quan đến ứng dụng theo ý muốn :

Hình 13-15 Cửa sổ File System của ứng dụng FileCopier



13.1.4.3 Triển khai trên các vị trí khác nhau

Mặc nhiên thì ứng dụng sẽ được cài đặt trên thư mục sau :

```
[ProgramFilesFolder]\[Manufacturer]\[Product Name.
  ProgramFilesFolder: thư mục Program Files trên máy người dùng
  Manufacturer: tên của nhà sản xuất
  Product Name: tên của ứng dụng
```

Ta có thể thay đổi các thông số này trong cửa sổ thuộc tính FileCopierSetupProject hoặc trong quá trình cài đặt.

Tạo biểu tượng của ứng dụng trên màn hình Desktop.

Để tạo biểu tượng cho ứng dụng, ta chỉ cần chọn **Application Folder\Primary output from FileCopier (Active)** ở cửa sổ bên trái, sau đó nhấn chuột phải và chọn **Create Shortcut to Primary output from FileCopier (Active).** Ta hiệu chỉnh đường dẫn của biểu tượng cho thích hợp.

Thêm các mục vào thư mục My Documents

Ta cũng có thể thêm các tài liệu cần thiết vào thư mục **My Documents** trên máy của người dùng khi cài đặt, bằng cách đặt chúng vào thư mục **User's Personal Data Folder**

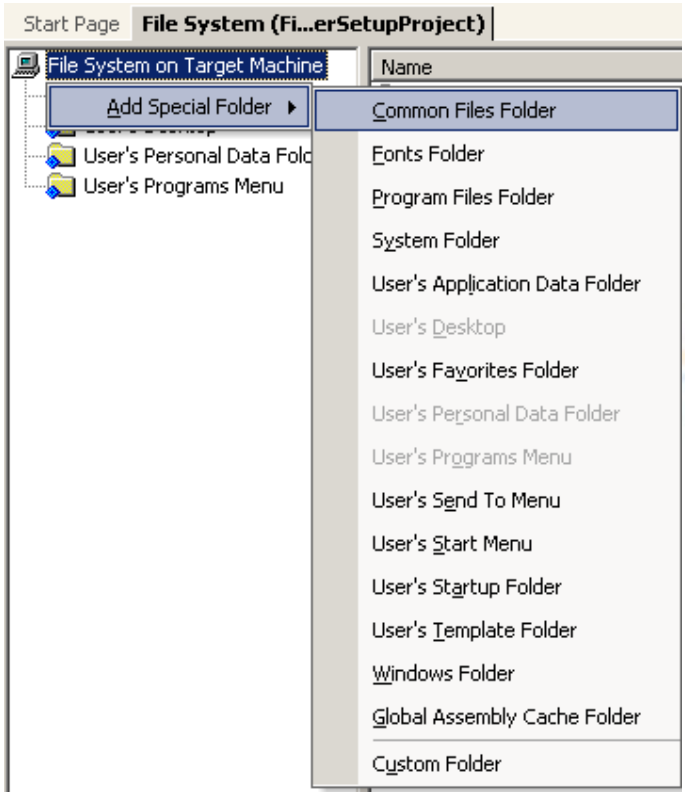
Tạo biểu trượng trong cửa sổ Start Menu

Để thêm các thành phần khác của ứng vào cửa sổ **Start / Programs**, ta thêm chúng vào thư mục **User's Program Menu** ở cửa sổ bên phải.

13.1.4.4 Thêm các chức năng cài đặt khác cho ứng dụng triển khai

Ngoài bốn mục được liệt kê trong hình trên, ta có thể bổ sung thêm các thư mục khác, như hình dưới đây :

Hình 13-16 Bổ sung các thư mục trong cửa sổ File System.



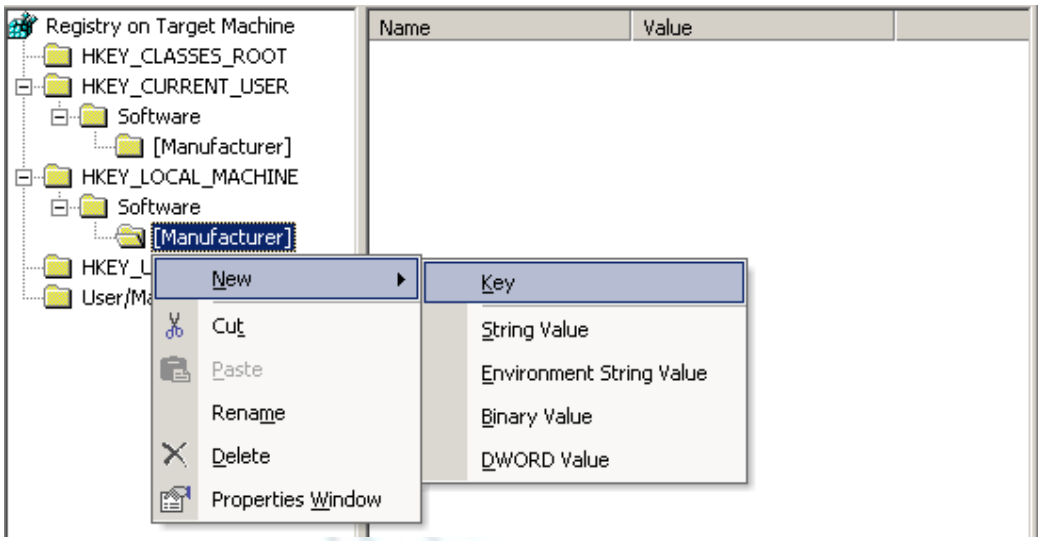
13.1.4.5 Các thành phần khác

Ta đã khảo sát qua cách cài đặt dùng **File System** trong **Menu View**, nay ta tìm hiểu thêm một số cách khác.

Tạo các thay đổi trong sổ đăng ký (Registry)

Cửa sổ đăng ký của View Menu cho phép làm cho trình cài đặt của chúng ta tạo ra các thay đổi trong sổ đăng ký chương trình trên các máy cài đặt ứng dụng. Nhấn chuột phải trên các thư mục được liệt trong hình dưới đây để hiệu chỉnh sổ đăng ký theo ý muốn :

Hình 13-17 Hiệu chỉnh sổ đăng ký.

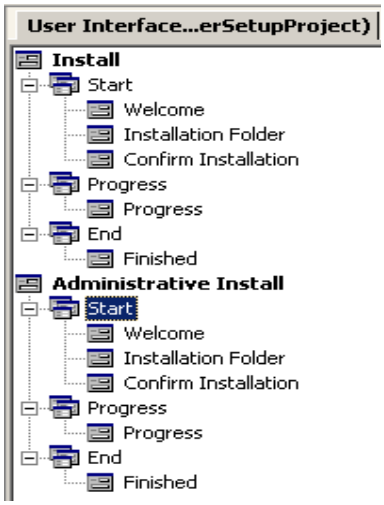


Chú ý: Phải rất thật cẩn thận khi cài đặt sổ đăng ký. Trong hầu hết các ứng dụng .NET đều không cần thiết phải liên quan đến sổ đăng ký, .NET quản lý ứng dụng không cần dùng sổ đăng ký (Registry).

Quản lý giao diện người dùng trong quá trình cài đặt

Để có thể kiểm soát các chữ hay giao diện đồ họa hiển thị trong suốt quá trình cài đặt ứng dụng, ta chọn mục User Interface trong View Menu. Hình dưới đây thể hiện luồng công việc trong quá trình cà đặt ứng dụng :

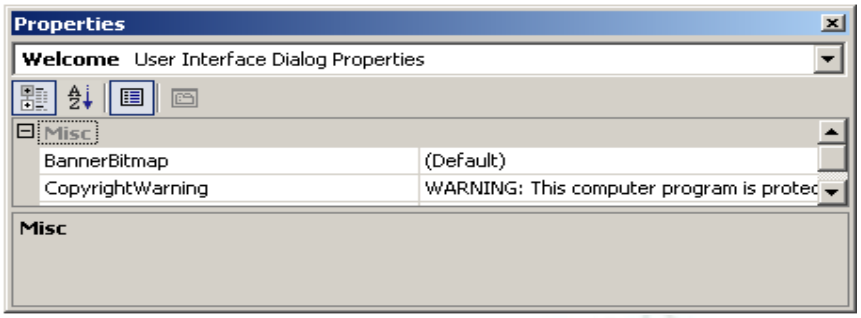
Hình 13-18 Luồng công việc trong quá trình cài đặt.



Khi ta nhấn chuột phải trên một bước nào đó trong tiến trình cài đặt và chọn **Properties** thì sẽ hiện lên một cửa sổ tương ứng với mục đó, nhờ hộp thoại thuộc tính này ta có thể hiệu chỉnh các chuỗi hay ảnh hiển thị thích hợp. Ta cũng có thể

thêm hay bớt đi một bước trong luồng công việc cài đặt. Ví dụ hộp thoại thuộc tính của mục **Welcome**.

Hình 13-19 Cửa sổ thuộc tính của mục Welcome trong quá trình cài đặt.



Hiệu chỉnh thêm cho quá trình cài đặt

Nếu quá trình cài đặt trong cửa sổ User Interface vẫn không đủ thỏa mãn nhu cầu cài đặt của ứng dụng, thì ta có thể tự hiệu chỉnh các thông số cho tiến trình cài đặt : điều kiện để chạy tiến trình cài đặt … Ta chọn mục Custom Option trong View Menu để thực hiện mục đích này.

Kết thúc việc tạo chương trình cài đặt

Sau khi hoàn tất mọi hiệu chỉnh, ta chỉ cần chạy ứng dụng và .NET sẽ tạo ra một tập tin cài đặt **MSD** để cài đặt ứng dụng của ta.

Chương 14 Truy cập dữ liệu với ADO.NET

Trong thực tế, có rất nhiều ứng dụng cần tương tác với cơ sở dữ liệu. .NET Framework cung cấp một tập các đối tượng cho phép truy cập vào cơ sở dữ liệu, tập các đối tượng này được gọi chung là ADO.NET.

ADO.NET tương tự với ADO, điểm khác biệt chính ở chỗ ADO.NET là một kiến trúc dữ liệu rời rạc, không kết nối (*Disconnected Data Architecture*). Với kiến trúc này, dữ liệu được nhận về từ cơ sở dữ liệu và được lưu trên vùng nhớ cache của máy người dùng. Người dùng có thể thao tác trên dữ liệu họ nhận về và chỉ kết nối đến cơ sở dữ liệu khi họ cần thay đổi các dòng dữ liệu hay yêu cầu dữ liệu mới.

Việc kết nối không liên tục đến cơ sở dữ liệu đã đem lại nhiều thuận lợi, trong đó điểm lợi nhất là việc giảm đi một lưu lượng lớn truy cập vào cơ sở dữ liệu cùng một lúc, tiết kiệm đáng kể tài nguyên bộ nhớ. Giảm thiểu đáng kể vấn đề hàng trăm ngàn kết nối cùng truy cập vào cơ sở dữ liệu cùng một lúc.

ADO.NET kết nối vào cơ sở dữ liệu để lấy dữ liệu và kết nối trở lại để cập nhật dữ liệu khi người dùng thay đổi chúng. Hầu hết mọi ứng dụng đều sử dụng nhiều thời gian cho việc đọc và hiển thị dữ liệu, vì thế ADO.NET đã cung cấp một tập hợp con các đối tượng dữ liệu không kết nối cho các ứng dụng để người dùng có thể đọc và hiển thị chúng mà không cần kết nối vào cơ sở dữ liệu.

Các đối tượng ngắt kết nối này làm việc tương tự đối với các ứng dụng Web.

14.1 Cơ sở dữ liệu và ngôn ngữ truy vấn SQL

Để có thể hiểu rõ được cách làm việc của ADO.NET, chúng ta cần phải nắm được một số khái niệm cơ bản về cơ sở dữ liệu quan hệ và ngôn ngữ truy vấn dữ liệu, như: khái niệm về dòng, cột, bảng, quan hệ giữa các bảng, khóa chính, khóa ngoại và cách truy vấn dữ liệu trên các bảng bằng ngôn ngữ truy vấn SQL : SELECT, UPDATE, DELETE … hay cách viết các thủ tục (Store Procedure) …. Trong phạm vi của tài liệu này, chúng ta sẽ không đề cập đến các mục trên.

Trong các ví dụ sau, chúng ta sẽ dùng cơ sở dữ liệu NorthWind, được cung cấp bởi Microsoft để minh họa cho các ví dụ của chúng ta.

14.2 Một số loại kết nối hiện đang sử dụng

1982 ra đời ODBC driver (Open Database Connectivity) của Microsoft. Chỉ truy xuất được thông tin quan hệ, không truy xuất được dữ liệu không quan hệ như : tập tin văn bản, email …Ta phải truy cập ODBC thông qua DSN.

Để truy cập được tất cả Datastore, dùng OLEDB provider thông qua ODBC. Là vỏ bọc của ODBC hoặc không. OLEDB dễ sử dụng hơn ODBC, nhưng chỉ có 1 số ít ngôn ngữ có thể hiểu được (C++), vì thế ra đời ADO. OLEDB là giao diện ở mức lập trình hệ thống để quản lý dữ liệu. OLEDB đơn giản chỉ là một tập các giao diện COM đóng gói thành các system service để quản trị các CSDL khác nhau. Gồm 4 đối tượng chính : Datasource, Session, Command, Rowset.

ADO là một COM, do đó được dùng với bất kỳ ngôn ngữ nào tương thích với COM. ADO không độc lập OS, nhưng độc lập ngôn ngữ : C++,VB, JavaScript, VBScript …Là vỏ bọc của OLEDB và ADO gồm 3 đối tượng chính : Connection, Command, Recordset.

Remote Data Services (RDS) của Microsoft cho phép dùng ADO thông qua các giao thức HTTP, HTTPS và DCOM để truy cập dữ liệu qua Web.

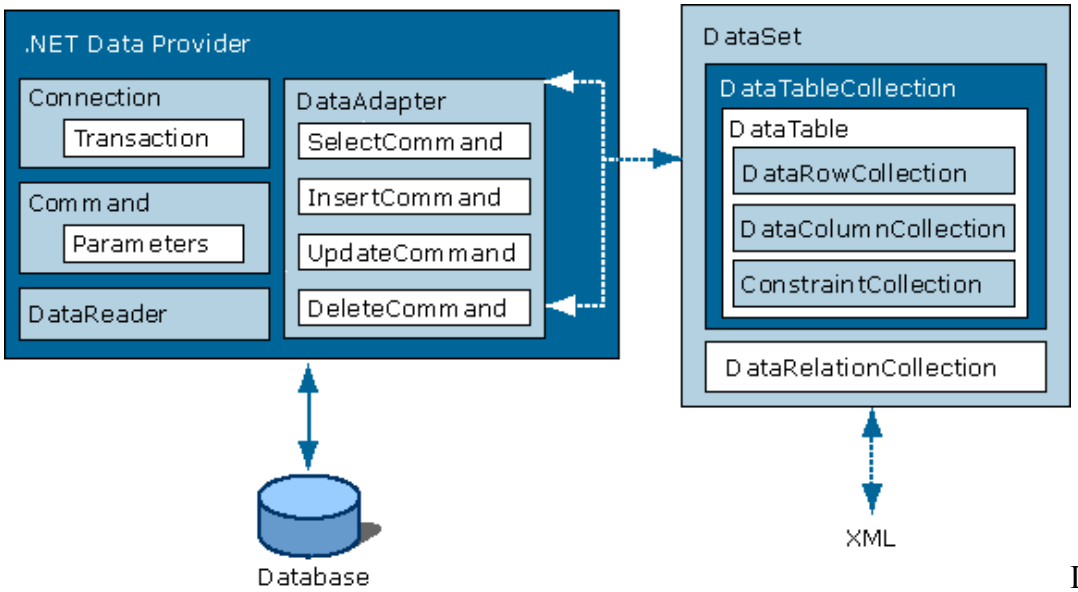
Microsoft Data Access Components (MDAC) là tổ hợp của ODBC, OLEDB, ADO và cả RDS.

Ta có thể kết nối dữ liệu bằng một trong các cách: dùng ODBC driver (DSN), dùng OLEDB thông qua ODBC hoặc OLEDB **không** thông qua ODBC.

14.3 Kiến trúc ADO.NET

ADO.NET được chia ra làm hai phần chính rõ rệt, được thể hiện qua hình

Hình 14-1 Kiến trúc ADO.NET



D

DataSet là thành phần chính cho đặc trưng kết nối không liên tục của kiến trúc ADO.NET. DataSet được thiết kế để có thể thích ứng với bất kỳ nguồn dữ liệu nào. DataSet chứa một hay nhiều đối tượng DataTable mà nó được tạo từ tập các dòng và cột dữ liệu, cùng với khoá chính, khóa ngoại, ràng buộc và các thông tin liên

quan đến đối tượng DataTable này. Bản thân DataSet được dạng như một tập tin XML.

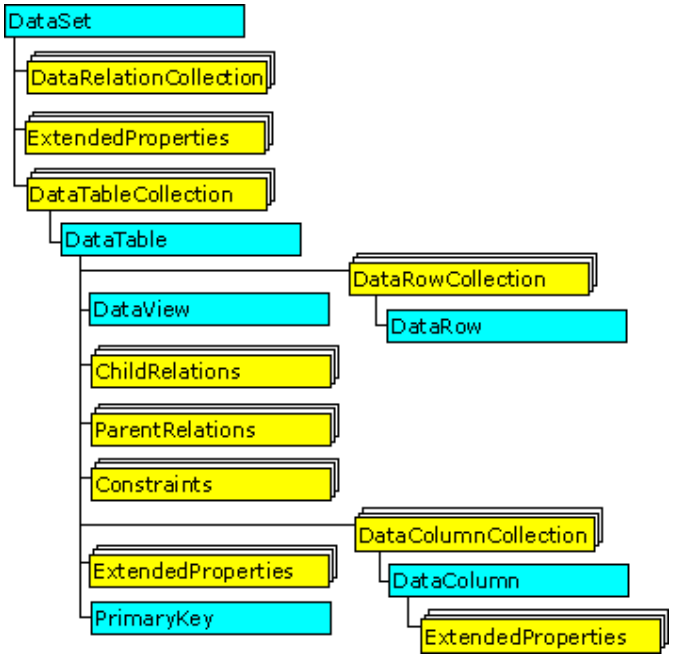
Thành phần chính thứ hai của ADO.NET chính là NET Provider Data, nó chứa các đối tượng phục vụ cho việc thao tác trên cơ sở dữ liệu được hiệu quả và nhanh chóng, nó bao gồm một tập các đối tượng Connection, Command, DataReader và DataAdapter. Đối tượng Connection cung cấp một kết nối đến cơ sở dữ liệu, Command cung cấp một thao tác đến cơ sở dữ liệu, DataReader cho phép chỉ đọc dữ liệu và DataAdapter là cấu nối trung gian giữa DataSet và nguồn dữ liệu.

14.4 Mô hình đối tượng ADO.NET

Có thể nói mô hình đối tượng của ADO.NET khá uyển chuyển, các đối tượng của nó được tạo ra dựa trên quan điểm đơn giản và dễ dùng. Đối tượng quan trọng nhất trong mô hình ADO.NET chính là **Dataset. Dataset** có thể được xem như là thể hiện của cả một cơ sở dữ liệu con, lưu trữ trên vùng nhớ cache của máy người dùng mà không kết nối đến cơ sở dữ liệu.

14.4.1 Mô hình đối tượng của Dataset

Hình 14-2 Mô hình đối tượng Dataset



DataSet bao gồm một tập các đối tượng DataRelation cũng như tập các đối tượng DataTable. Các đối tượng này đóng vai trò như các thuộc tính của DataSet.

14.4.2 Đối tượng DataTable và DataColumn

Ta có thể viết mã C# để tạo ra đối tượng **DataTable** hay nhận về từ kết quả của câu truy vấn đến cơ sở dữ liệu. **DataTable** có một số thuộc tính dùng chung (public) như thuộc tính **Columns**, từ thuộc tính này ta có thể truy cập đến đối tượng **DataColumnsCollection** thông qua chỉ mục hay tên của cột để nhận về các đối tượng **DataColumn** thích hợp, mỗi **DataColumn** tương ứng với một cột trong một bảng dữ liệu. Ví dụ :

```
DataTable dt = new DataTable("tenBang");
DataColumn dc = dt.Columns["tenCot"];
```

14.4.3 Đối tượng DataRelation

Ngoài tập các đối tượng **DataTable** được truy cập thông qua thuộc tính **Tables**, **DataSet** còn có một thuộc tính **Relations**. Thuộc tính này dùng để truy cập đến đối tượng **DataRelationCollection** thông qua chỉ mục hay tên của quan hệ và sẽ trả về đối tượng **DataRelation** tương ứng. Ví dụ :

```
DataSet ds = new DataSet("tenDataSet");
DataRelation dre = ds.Relations["tenQuanHe"];
```

14.4.4 Các bản ghi (Rows)

Tương tự như thuộc tính **Columns** của đối tượng **DataTable**, để truy cập đến các dòng ta cũng có thuộc tính **Rows**. ADO. NET không đưa ra khái niệm RecordSet, thay vào đó để duyệt qua các dòng (Row), ta có thể truy cập các dòng thông qua thuộc tính **Rows** bằng vòng lặp **foreach**.

14.4.5 Đối tượng SqlConnection và SqlCommand

Đối tượng SqlConnection đại diện cho một kết nối đến cơ sở dữ liệu, đối tượng này có thể được dùng chung cho các đối tượng SqlCommand khác nhau. Đối tượng SqlCommand cho phép thực hiện một câu lệnh truy vấn trực tiếp : như SELECT, UPDATE hay DELETE hay gọi một thủ tục (Store Procedure) từ cơ sở dữ liệu.

14.4.6 Đối tượng DataAdapter

ADO.NET dùng DataAdapter như là chiếc cầu nối trung gian giữa DataSet và DataSource (nguồn dữ liệu), nó lấy dữ liệu từ cơ sở dữ liệu sau đó dùng phương Fill() để đẩy dữ liệu cho đối tượng DataSet. Nhờ đối tượng DataAdapter này mà DataSet tồn tại tách biệt, độc lập với cơ sở dữ liệu và một DataSet có thể là thể hiện của một hay nhiều cơ sở dữ liệu. Ví dụ :

```
//Tạo đối tượng SqlDataAdapter
SqlDataAdapter sda = new SqlDataAdapter();

// cung cấp cho sda một SqlCommand và SqlConnection ...
// lấy dữ liệu ...

//tạo đối tượng DataSet mới
DataSet ds = new DataSet("tenDataSet");
```