C# and .NET Framework
Bài 6: Data Access and Viewing with .NET

Đoàn Quang Minh
minhdqtt@gmail.com
http://www.VTPortal.net
Last update: 30. December 2006

Mục lục

- ADO.NET Overview
- Using Database Connections
- Commands
- Quick Data Access: The Data Reader
- Working with DataSet
- Viewing .NET data
- Example

ADO.NET Overview

- ADO.NET là gì
  - ADO - Microsoft's ActiveX Data Objects: thư viện các cho phép truy cập và xử lý CSDL.
  - ADO có một số hạn chế: luôn luôn giữ kết nối, chỉ làm việc với CSDL...
  - ADO.NET: làm việc với các đối tượng dữ liệu, hỗ trợ mạnh mẽ SQL Server, đồng thời hỗ trợ các kết nối OLE DB.
- namespace System.Data
  - Để truy cập và xử lý CSDL, sử dụng các namespace System.Data, System.Data.Common, System.Data.OleDb, System.Data.SqlClient, System.Data.SqlTypes.
  - Các lớp cơ bản trong System.Data: DataSet, DataTable, DataRow, DataColumn, DataRelation, Constraint
  - Các lớp đặc biệt: SqlCommand, OleDbCommand, SqlCommandBuilder, OleDbCommandBuilder, SqlConnection, OleDbConnection, SqlDataAdapter, OleDbDataAdapter, SqlDataReader, OleDbDataReader, SqlParameter, OleDbParameter, SqlTransaction, OleDbTransaction

Using Database Connections

- Muốn truy vấn CSDL, chúng ta phải có một kết nối đến CSDL
  - Sử dụng lớp SqlConnection, OleDbConnection
    - Cung cấp chuỗi kết nối: thông thường bao gồm tên server, tên CSDL, tên truy cập, mật khẩu.
    - Sử dụng các phương thức Open() và Close().
  - Sử dụng kết nối hiệu quả
    - Đóng ngay kết nối khi không dùng nữa: thông thường, chúng ta không duy trì một kết nối "cứng" đến CSDL. Khi cần truy vấn, chúng ta mở kết nối, truy vấn xong, đóng ngay kết nối lại.
    - Khối lệnh kết nối nên đặt trong khối try…catch
    - Từ khóa using: sử dụng một đối tượng. Ra khỏi phạm vi của using, đối tượng sẽ bị huỷ.
  - Transactions
    - Transactions là gì?.
    - Sử dụng thông qua SqlTransaction hoặc OleDbTransaction.

Commands

- Là đối tượng thực thi câu lệnh
  - Một đối tượng command thông thường được sử dụng để thực thi một câu lệnh SQL hoặc một thủ tục lưu.
  - Các bước thực hiện:
    - Khai báo và mở một connection.
    - Khai báo một chuỗi chứa câu lệnh SQL hoặc tên thủ tục lưu
    - Khai báo một đối tượng command với câu lệnh truy vấn và nguồn kết nối.
    - Chỉ định thuộc tính câu truy vấn: dạng text hay thủ tục lưu.
    - Thêm các tham số của câu truy vấn.
    - Thực hiện lệnh truy vấn: tuỳ theo yêu cầu thực hiện các lệnh khác nhau

Commands

- Các lệnh truy vấn với command.
  - ExecuteNonQuery()
    - Thực thi câu lệnh, không trả về kết quả.
    - Thường sử dụng trong truy vấn không cần quan tâm đến kết quả, ví dụ các lệnh delete, insert, update…
  - ExecuteReader()
    - Trả về một DataReader.
    - Thường sử dụng trong các truy vấn hiển thị dữ liệu.
  - ExecuteScalar()
    - Trả về một đối tượng duy nhất.
    - Thường sử dụng trong các truy vấn trả về một giá trị dữ liệu đơn, ví dụ các lệnh tính tổng, tính trung bình, tính min/max…
  - ExecuteXmlReader()
    - Trả về một XmlReader.
    - Thường được xử dụng khi hiển thị dữ liệu dưới dạng XML.
    - Nên dùng nếu CSDL hỗ trợ truy vấn XML, ví dụ SQL Server 2000.

Commands

```
public class ExecuteScalarExample
{
   public static void Main(string[] args)
   {
       string source = "server=(local)\\NetSDK;" +
   "uid=QSUser;pwd=QSPassword;" + "database=Northwind";
       string select = "SELECT COUNT(*) FROM Customers";
       SqlConnection conn = new SqlConnection(source);
       conn.Open();
       SqlCommand cmd = new SqlCommand(select, conn);
       object o = cmd.ExecuteScalar();
       Console.WriteLine ( o ) ;
   }
}
```

Quick Data Access: The Data Reader

- Data Reader
  - Chỉ được tạo ra bởi giá trị trả về của câu lệnh truy vấn.
  - Kết nối tới CSDL luôn mở, cho đến nhận được lệnh đóng.
- Tính chất
  - Một data reader giống như một record set chỉ tiến (forward only) trong ADO.
    - Chỉ có thể đọc, và đi đến bản ghi tiếp.
    - Không thể quay lại các bản ghi đã đọc.
  - Tốc độ cao:
    - Một data reader không giữ các bản ghi trong bộ nhớ.
    - Data reader chỉ có nhiệm vụ lấy dữ liệu từ CSDL và chuyển về.
    - Rất hay được sử dụng khi chỉ cần hiển thị dữ liệu, nhất là trong môi trường web.

Working with DataSet

- DataSet:
  - Có tác dụng giống như một CSDL offline:
    - Trong một DataSet có thể chứa các DataTable, DataRelation,…
    - DataSet có thể được xây dựng không chỉ từ các truy vấn CSDL, mà có thể từ các tập tin khác (text, Excel, CVS,…)
  - Để tạo DataSet
    - Truy vấn CSDL, dựa trên một DataAdapter
    - Xây dựng bằng cách thêm các DataTable

Working with DataSet

- Truy vấn CSDL
  - Mở một connection.
  - Tạo một DataAdapter, chỉ định câu lệnh truy vấn cho data adapter.
  - Tạo mới một data set.
  - Sử dụng phương thức Fill() của data adapter.
- Xây dựng bằng cách thêm các data table
  - Tạo mới một DataSet.
  - Tạo mới các DataTable.
    - Khởi tạo data table bằng cách thêm mới DataColumn
    - Thêm các dòng dữ liệu vào data table.
  - Add các data table vào data set bằng cách thêm vào thuộc tính Tables của data set

Working with DataSet – Example

```
DataSet ds = new DataSet();
DataTable dt = ds.Tables.Add("SampleData");

dt.Columns.Add("MonHocID", typeof(Guid));
dt.Columns.Add("TenMon", typeof(string));
dt.Columns.Add("MaMon", typeof(string));
dt.Columns.Add(“HeSoMon", typeof(int));

DataRow dr;
for (int i = 1; i <= 20; i++)
{
   dr = dt.NewRow();
   dr["MonHocID"] = Guid.NewGuid();
   dr["TenMon"] = "Mon hoc thu " + i.ToString();
   dr["MaMon"] = "MaMon00" + i.ToString();
   dr["HeSoMon"] = i;
   dt.Rows.Add(dr);
}
```

Working with DataSet – Example

```
SqlConnection conn = new SqlConnection(source);
SqlCommand cmd = new SqlCommand(select, conn);
SqlDataAdapter adapter = new SqlDataAdapter();
adapter.SelectCommand = cmd;
DataSet data = new DataSet();
try
{
   conn.Open();
   adapter.Fill(data);
}
catch (SqlException expSQL)
{
}
finally
{
   conn.Close();
}
```

Working with DataSet

- Sau khi xây dựng, điền thông tin vào data set, có thể thay đổi dữ liệu (insert, delete, update) và cập nhật trở lại vào CSDL.
  - Thuộc tính Rows của DataTable là một collection.
    - Thêm (insert) một row mới bằng phương thức Add()
    - Cập nhật (update) một row cũ bằng cách thay đổi các giá trị của row
    - Xoá (delete) một row bằng phương thức Delete()

Viewing .NET data

- Song song với việc xử lý dữ liệu là hiển thị dữ liệu:
  - .NET cung cấp các control rất hiệu quả cho việc hiển thị dữ liệu.
  - Việc hiển thị dữ liệu trên các control này chỉ đơn giản bằng cách chỉ ra nguồn dữ liệu, gọi phương thức DataBind()
  - Các đối tượng hiển thị dữ liệu bao gồm: DataGrid, DataList, Repeter

Viewing .NET data

- DataGrid (ASP.NET)
  - Cho phép hiển thị dữ liệu dưới dạng bảng.
  - Cho phép phân trang và sắp xếp dữ liệu.
- Hiển thị dữ liệu
  - Thiết kế form:
    - Thêm một DataGrid.
    - Thay đổi các thuộc tính cần thiết.
  - Thuộc tính quan trọng:
    - DataKeyField: khoá chính của lưới, thông thường là khoá chính trong bảng dữ liệu.
    - DataMember: bảng dữ liệu (trong DataSet) cần hiển thị.
    - DataSource: nguồn dữ liệu cần hiển thị
    - AutoGenerateColumns: nếu bằng true, các cột của grid sẽ tự động sinh theo các (tên) trường dữ liệu trong bảng.

Viewing .NET data

- DataGrid (ASP.NET)
  - Cột trong grid: cho phép hiển thị theo nhiều khuôn dạng khác nhau:
    - Bound Column: chỉ hiện thị dữ liệu dạng text
    - Button Column: cho phép thực thi lệnh (xoá, soạn thảo,…)
    - HyperLink Column: siêu liên kết
    - TemplateColumn: mẫu, chứa mô tả của cột
  - Với cột là button column
    - Như là một nút lệnh bình thường
    - Thông thường, cần xử lý dòng dữ liệu được click, dòng này sẽ được xác định dựa trên khoá của lưới

Viewing .NET data

- GridView (ASP.NET): Tương tự control DataGrid, nhưng có một số khác biệt
  - Thêm một số loại cột mới:
    - CheckBoxField: hiển thị dữ liệu dạng check;
    - ImageField: hiển thị dữ liệu dạng ảnh.
  - Hỗ trợ đa ngôn ngữ hoàn toàn;
  - Hỗ trợ Ajax thông qua khái niệm callback.
- Data Source: là các control kết nối và truy xuất dữ liệu từ nguồn dữ liệu
  - Các nguồn dữ liệu hỗ trợ: Access, SQL Server, Object, Site Map, XML file;
  - Cho phép thiết kế nhanh các nguồn dữ liệu phù hợp thông qua winzard;
  - Khi gắn nguồn dữ liệu vào control hiển thị, dữ liệu sẽ được tự động truy vấn và hiển thị mà không cần viết thêm mã lệnh;
  - Rất tối ưu nếu có sử dụng callback

Example

- Ví dụ làm việc với CSDL
  - Một cửa hàng bán thiết bị vi tính cần quản lý sản phẩm theo danh mục:
    - Các danh mục như: mainboard, chip, hdd,…
    - Trong danh mục có các sản phẩm: ví dụ trong danh mục chip có chip AMD, chip Intel (các dòng khác nhau)
  - Yêu cầu:
    - Hiển thị danh mục ở một bên, chi tiết các sản phẩm ở một bên
    - Khi người dùng chọn một danh mục, hiển thị các sản phẩm tương ứng.

Example

- Phân tích
  - Các yêu cầu đầu bài.
  - Thiết kế CSDL
    - Các bảng cần thiết
    - Các thủ tục lưu, nếu cần
  - Xác định môi trường ứng dụng (Windows hay Web)
  - Design form
  - Viết mã dựa trên phân tích yêu cầu

Tài liệu tham khảo

- Professional C#, Second Edition
- http://www.asp.net
- http://www.microsoft.com/net/default.mspx
- http://www.codeproject.com
- Địa chỉ download tài liệu http://www.thanglong.edu.vn/giang-day/tab.aspx
- Diễn đàn C# & .NET http://www.thanglong.edu.vn/forum/cmd/0/category/hoc-tap-nghien-cuu/dot-net/tab.aspx