Buổi 3
Kết nối CSDL với ADO.NET

I. Các lớp đối tượng trong ADO.NET

ADO.NET là 1 lớp đối tượng cầu nối giữa giao diện và CSDL, cung cấp cho người dùng khả năng tương tác với CSDL.

ADO.NET cung cấp các lớp đối tượng để truy cập và thực thi truy vấn tới CSDL đặt tại SQL Server như `SqlConnection`, `SqlCommand`, `SqlParameter`, `SqlDataReader`, `SqlDataAdapter` (thuộc namespace `System.Data.SqlClient`) cũng như các lớp đối tượng để lưu dữ liệu như `DataSet`, `DataTable`, `DataRow`, `DataColumn` (thuộc namespace `System.Data`).

Trong khuôn khổ tài liệu buổi 3 này, quy ước tên một số đối tượng của một số lớp như sau:

- `ds`: lớp `DataSet`
- `dtb`: lớp `DataTable`
- `dr`: lớp `DataRow`
- `dc`: lớp `DataColumn`
- `conn`: lớp `SqlConnection`
- `cmd`: lớp `SqlCommand`
- `param`: lớp `SqlParameter`
- `reader`: lớp `SqlDataReader`
- `adapter`: lớp `SqlDataAdapter`

1. DataSet

Lớp đối tượng `DataSet` có thể xem như là 1 ánh xạ của CSDL. Tuy nhiên, giữa `DataSet` và CSDL không có kết nối đồng bộ, do đó, mọi thay đổi xảy ra ở 1 trong 2 bên sẽ không được cập nhật ở bên còn lại trừ khi kết nối được thiết lập lại.

Lớp đối tượng `DataSet` chứa các đối tượng `DataTable` và `DataRelation`. Một `DataSet` có thể có nhiều `DataTable`, và mối quan hệ giữa 2 `DatatTable` được biểu diễn bằng `DataRelation`.

Có thể đặt tên cho `DataSet` bằng thuộc tính `DataSetName`:

```
ds.DataSetName = "WebBanHang";
```

Thuộc tính `Tables` của đối tượng `ds` là 1 collection chứa tất cả các `DataTable` trong `DataSet`. Để truy cập vào 1 `DataTable` của đối tượng `ds`, chúng ta sử dụng thuộc tính `Tables` theo 2 cách:

- `DataTable dtb = ds.Tables[i];`
  Trong đó: `i` là chỉ số của `DataTable` cần truy cập trong collection.
- `DataTable dtb = ds.Tables["TaiKhoan"];`
  Trong đó: `"TaiKhoan"` là tên của `DataTable` cần truy cập.

2. DataTable

Lớp đối tượng `DataTable` có thể xem như là 1 ánh xạ của bảng trong CSDL. Tương tự như bảng, `DataTable` cũng có nhiều dòng (thuộc tính `Rows`) và nhiều cột (thuộc tính `Columns`).

Dữ liệu trong `DataTable` có thể được tạo ra theo 3 cách:

- Tự tạo các dòng dữ liệu (`DataRow`) và thêm vào `DataTable` một cách thủ công.

- Dùng `SqlDataReader` để đọc dữ liệu từ CSDL và load vào `DataTable`.
- Dùng `SqlDataAdapter` để đọc dữ liệu từ CSDL và fill vào `DataTable`.

Thuộc tính `Rows` của đối tượng `dtb` là 1 collection chứa tất cả các `DataRow` trong `DataTable`. Để truy cập dòng thứ `i` từ đối tượng `dtb`, chúng ta sử dụng câu lệnh:

```
DataRow dr = dtb.Rows[i];
```

3. DataRow

Lớp đối tượng `DataRow` có thể xem như là 1 ánh xạ của dòng trong bảng.

Để truy cập vào 1 ô (giao giữa dòng và cột) của đối tượng dr, chúng ta sử dụng 1 trong 2 cú pháp sau:

- `dr[i]`
  Trong đó: i là chỉ số của cột muốn truy cập.
- `dr["MatKhau"]`
  Trong đó: `"MatKhau"` là tên của cột muốn truy cập.

4. SqlConnection

Lớp đối tượng SqlConnection cung cấp khả năng kết nối tới CSDL.

Để kết nối tới CSDL, cần phải có một chuỗi kết nối (*connection string*) quy định rõ tên máy chủ và tên CSDL theo cấu trúc sau:

```
Data Source=localhost;Initial Catalog=WebBanHang;Integrated Security=True
```

Trong đó:

- `localhost` là tên máy chủ.
- `WebBanHang` là tên CSDL.

Chuỗi kết nối được quy định trong thuộc tính `ConnectionString` của đối tượng `conn`.

Để mở và đóng kết nối, chúng ta dùng phương thức `Open()` và `Close()`.

5. SqlCommand

Lớp đối tượng `SqlCommand` đảm nhiệm việc thực thi các câu truy vấn.

Các thuộc tính quan trọng của đối tượng `cmd`:

- `Connection`: Kết nối đến CSDL, nhận giá trị là đối tượng thuộc lớp `SqlConnection`.
- `CommandText`: Câu truy vấn muốn thực hiện.
- `Parameters`: Danh sách tham số của câu truy vấn.

Các phương thức quan trọng của đối tượng `cmd` giúp thực thi truy vấn:

- `ExecuteReader()`: Thực thi câu truy vấn `SELECT`, trả về bảng kết quả dưới dạng đối tượng thuộc lớp `SqlDataReader`.
- `ExecuteScalar()`: Thực thi câu truy vấn `SELECT`, trả về ô đầu tiên trong bảng kết quả dưới dạng đối tượng thuộc kiểu `object`.
- `ExecuteNonQuery()`: Thực thi câu truy vấn `INSERT`, `UPDATE`, `DELETE`, trả về 1 số nguyên là số dòng bị ảnh hưởng bởi câu truy vấn.

6. SqlParameter

Lớp đối tượng `SqlParameter` quy định tham số (*parameter*) cho câu truy vấn.

Các thuộc tính quan trọng của đối tượng `param`:

- `ParameterName`: Tên của parameter, bắt đầu bằng kí tự @, ví dụ: `@TenTK`
- `Value`: Giá trị của parameter.

- `SqlDbType`: Kiểu dữ liệu của tham số trong CSDL.

Để thêm parameter cho câu truy vấn, thuộc tính `Parameters` của đối tượng `cmd` cung cấp các phương thức sau:

- `Add()`: Nhận vào tham số là 1 đối tượng thuộc lớp `SqlParameter`.
- `AddWithValue()`: Nhận vào tham số là tên của parameter và giá trị của nó.
- `AddRange()`: Nhận vào tham số là 1 mảng đối tượng thuộc lớp `SqlParameter`.

7. SqlDataReader

Lớp đối tượng `SqlDataReader` phục vụ việc đọc dữ liệu từ CSDL theo 1 chiều từ trên xuống dưới. Để đọc dữ liệu, kết nối tới CSDL phải được duy trì cho đến khi đối tượng `reader` được đóng lại bằng phương thức `Close()`.

Phương thức `ExecuteReader()` của đối tượng `cmd` sẽ trả về bảng kết quả dưới dạng đối tượng thuộc lớp `SqlDataReader`:

```
reader = cmd.ExecuteReader();
```

Để lấy dữ liệu từ đối tượng `reader` này, chúng ta có thể thực hiện theo 2 cách:

- Load toàn bộ bảng kết quả vào 1 đối tượng thuộc lớp `DataTable`:

```
dtb.Load(reader());
```

- Đọc từng dòng dữ liệu[1] với phương thức `Read()` của đối tượng `reader`:

```
reader.Read();
```

8. SqlDataAdapter

Lớp đối tượng `SqlDataAdapter` phục vụ việc lấy dữ liệu từ CSDL và lưu vào `DataSet` hoặc `DataTable`. Sau khi dữ liệu đã được lấy, kết nối có thể được đóng lại.

Để lấy dữ liệu từ CSDL bằng lớp đối tượng `SqlDataAdapter`, chúng ta sử dụng cú pháp sau:

```
adapter.SelectCommand = cmd;
adapter.Fill(dtb);
```

Lớp đối tượng `SqlDataAdapter` cũng cung cấp phương thức `Update()` giúp cập nhật những thay đổi về dữ liệu từ `DataSet` hoặc `DataTable` xuống CSDL.

II. Kết nối CSDL

Trước khi thực hiện bất kì thao tác gì với CSDL, cần phải kết nối tới CSDL theo đoạn code sau:

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString = "<Chuỗi kết nối>";
conn.Open();
```

Hoặc có thể viết bằng 1 cách đơn giản hơn:

```
SqlConnection conn = new SqlConnection("<Chuỗi kết nối>");
conn.Open();
```

Chuỗi kết nối cũng có thể được lấy bằng cách kết nối với CSDL từ Visual Studio.

Sau khi kết thúc thao tác với CSDL, cần đóng kết nối bằng câu lệnh:

```
conn.Close();
```

Trong khuôn khổ tài liệu buổi 3 này, chúng ta sẽ sử dụng CSDL WebBanHang.

---

[1] Tham khảo cách đọc từng dòng dữ liệu từ reader bằng phương thức Read() tại link: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqldatareader

III. Truy xuất dữ liệu đơn giản với câu lệnh SELECT

1. Sử dụng phương thức ExecuteReader()

Đoạn code sau đây thực hiện việc lấy toàn bộ dữ liệu từ bảng SanPham, sau đó hiển thị danh sách tên sản phẩm lên giao diện, mỗi tên sản phẩm nằm trên 1 dòng:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;Initial Catalog=WebBanHang;Integrated Security=True");
conn.Open();

SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "SELECT * FROM SanPham";
SqlDataReader reader = cmd.ExecuteReader();

DataTable dtb = new DataTable();
dtb.Load(reader);

string dsTenSP = string.Empty;
foreach (DataRow row in dtb.Rows) {
    dsTenSP += row["TenSP"].ToString() + "<br>";
}
Response.Write(dsTenSP);

conn.Close();
```

2. Sử dụng SqlDataAdapter

Đoạn code sau đây thực hiện việc lấy toàn bộ dữ liệu từ bảng SanPham, sau đó hiển thị danh sách tên sản phẩm lên giao diện, mỗi tên sản phẩm nằm trên 1 dòng:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;Initial Catalog=WebBanHang;Integrated Security=True");
conn.Open();

SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "SELECT * FROM SanPham";
SqlDataAdapter adapter = new SqlDataAdapter();
adapter.SelectCommand = cmd;

DataTable dtb = new DataTable();
adapter.Fill(dtb);

string dsTenSP = string.Empty;
foreach (DataRow row in dtb.Rows) {
    dsTenSP += row["TenSP"].ToString() + "<br>";
}
Response.Write(dsTenSP);

conn.Close();
```

3. Sử dụng phương thức ExcuteScalar()

Đoạn code sau đây thực hiện việc đếm số lượng sản phẩm đang có trong bảng SanPham, sau đó hiển thị kết quả lên giao diện:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;Initial
Catalog=QLVatTu;Integrated Security=True");
conn.Open();

SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "SELECT COUNT(*) FROM SanPham";

int sl = Convert.ToInt32(cmd.ExecuteScalar());
string kq = string.Format("<script>alert('So luong san pham la
{0}')</script>", sl);
Response.Write(kq);

conn.Close();
```

IV. Thay đổi dữ liệu với câu lệnh INSERT, UPDATE, DELETE

1. Truy vấn không kèm tham số

Đoạn code sau đây thực hiện việc xóa toàn bộ bảng SanPham:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;Initial
Catalog=QLVatTu;Integrated Security=True");
conn.Open();

SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "DELETE FROM SanPham";

int kq = cmd.ExecuteNonQuery();
Response.Write(string.Format("Xóa thành công {0} dòng", kq));

conn.Close();
```

2. Truy vấn kèm tham số

Đoạn code sau đây thực hiện việc xóa sản phẩm có mã được quy định ở textbox `txtMaSP`:

```
SqlConnection conn = new SqlConnection("Data Source=localhost;Initial
Catalog=QLVatTu;Integrated Security=True");
conn.Open();

SqlCommand cmd = new SqlCommand();
cmd.Connection = conn;
cmd.CommandText = "DELETE FROM SanPham WHERE MaSP=@MaSP";
cmd.Parameters.AddWithValue("@MaSP", txtMaSP.Text);

int kq = cmd.ExecuteNonQuery();
Response.Write(string.Format("Xóa thành công {0} dòng", kq));

conn.Close();
```