

MỤC LỤC

Mục lục	1
Lời Mở Đầu	
Chương 1. Giới Thiệu ADO.NET	6
1.1. Giới thiệu.....	6
1.1.1. ADO.NET Là Gì ?	7
1.1.2. Hiểu Đơn Giản Về ADO.NET	9
1.2. Các Data provider thông dụng:.....	11
1.2.1. SQL Data Provider.....	12
1.3. Dataset	
Chương 2. Cơ Sở Dữ Liệu SQL SERVER	16
2.1. SQL là gì?.....	16
2.2. Các thao tác trên cơ sở dữ liệu	16
2.3. Định nghĩa dữ liệu SQL.....	16
2.4. Phát biểu SQL Select.....	17
2.4.1. Mệnh đề WHERE	19
2.4.2. Sắp xếp dữ liệu	22
2.5. Chèn thêm dữ liệu (INSERTING DATA).....	24
2.6. Thay đổi dữ liệu (UPDATING DATA)	24
2.7. Xóa dữ liệu (DELETING DATA).....	25
2.8. Kiểu dữ liệu T-SQL.....	25
2.8.1. Dữ liệu kiểu số (Numeric Data Types)	25
2.8.2. Dữ liệu kiểu tiền tệ (Money Data Types)	26
2.8.3. Kiểu dữ liệu chuỗi ký tự (Character String Data Types)	26

2.8.4. Kiểu dữ liệu ngày giờ (Date and Time Data Types).....	26
2.8.5. Kiểu dữ liệu khác (Other Data Types)	27
2.8.6. Độ ưu tiên các kiểu dữ liệu (Data Type Precedence)	27
2.9. Stored Procedure	28
2.9.1. Stored Procedure cơ bản	28
2.9.2. Tạo Stored Procedure.....	30
2.9.3. Thực thi Stored Procedure	31
2.9.4. Thay đổi nội dung Stored Procedure	32
2.9.5. Xóa Stored Procedure.....	32
2.9.6. Tham số trong Stored Procedure	33
2.9.7. Trả về giá trị trong Stored Procedure.....	34
2.9.8. Kết hợp Stored Procedure với các lệnh T-SQL	36
Chương 3. Các Thành Phần ADO.NET	37
3.1. Giới thiệu.....	38
3.2. Đối tượng Connection	38
3.2.1. Kết nối cơ sở dữ liệu SQL Server.....	38
3.2.2. Tập tin lưu trữ chuỗi kết nối	43
3.2.3. Tham khảo một vài chuỗi kết nối CSDL của các .Net Data Provider khác	49
3.2.4. Mở và đóng kết nối cơ sở dữ liệu	50
3.3. Đối tượng Command	54
3.3.1. Đối tượng SqlCommand	54
3.3.2. Đối tượng DataReader	67
3.3.3. Đối tượng DataAdapter.....	70
3.3.4. Đối tượng SqlParameter	77
Chương 4. Các Thành Phần Phía Ứng Dụng.....	81

4.1.	Giới thiệu.....	81
4.2.	Datatable.....	82
4.2.1.	DataColumn	83
4.2.2.	DataRow	84
4.2.3.	DataView	85
4.2.4.	Nạp dữ liệu vào DataSet	85
4.2.5.	Nạp dữ liệu vào DataSet bằng DataAdapter	86
4.2.6.	Cập nhật CSDL bằng DataAdapter.....	87
4.3.	Các đối tượng trình diễn dữ liệu.....	108
4.3.1.	DataView – Một góc nhìn của DataTable	108
4.3.2.	Tạo và sử dụng DataView.....	108
4.3.3.	Lọc dữ liệu với RowFilter, sắp xếp dữ liệu với thuộc tính Sort().....	109
4.4.	Trình diễn dữ liệu với DataGridView	111
4.4.1.	Xác định dòng, ô hiện hành.	112
4.4.2.	Tùy biến điều khiển nhập liệu trên DataGridView	114
4.4.3.	Thêm cột điều khiển Button vào DataGridView.	121
Chương 5.	Thiết Kế Báo Biểu	125
5.1.	Sử DÙNG REPORT CHUẨN CỦA VISUAL STUDIO 2010.....	125
5.1.1.	Quy trình tạo Report.	125
5.1.2.	Ví dụ minh họa:	126
5.2.	BÀI TẬP ÚNG DỤNG CHO PHẦN REPORT.....	140
5.2.1.	Bài tập 1: Thiết kế report có sử dụng ComboBox	140
5.2.2.	Bài tập 2:.....	141
5.2.3.	Bài tập 3:.....	141
5.2.4.	Bài tập 04:.....	142
5.3.	Hướng dẫn giải bài tập:	143

5.3.1. Hướng dẫn bài tập 1	143
5.3.2. Bài tập 2: (tương tự bài tập 1).....	151
5.3.3. Hướng dẫn Bài tập 4:	151
Chương 6. Mô hình 3 lớp	155
6.1. Giới thiệu mô hình 3 lớp	155
6.2. Các thành phần trong 3-Layer	155
6.2.1. Presentation Layers.....	155
6.2.2. Business Logic Layer.....	155
6.2.3. Data Access Layer	155
6.3. Cách vận hành của mô hình.....	156
6.4. Tổ chức mô hình 3-Layer	156
Phụ Lục	
Chuẩn viết code trong C#	
Tài liệu tham khảo	

MỞ ĐẦU

Tư duy lập trình trong những năm gần đây đã có sự biến đổi lớn. Đó là sự ra đời của phương pháp lập trình hướng đối tượng. Với phương pháp lập trình này, những người lập trình không còn lo ngại trước những chương trình lớn và phức tạp. Cùng với xu hướng đó, giáo trình lập trình ứng dụng cơ sở dữ liệu với C# cung cấp cho sinh viên một cách tiếp cận lập trình hướng đối tượng hiện đại, mà cụ thể là hướng dẫn sinh viên lập trình theo mô hình 3 lớp.

Giáo trình gồm 6 chương giới thiệu cho sinh viên đầy đủ về những vấn đề trong lập trình ứng dụng cơ sở dữ liệu. Chương 1, giới thiệu tổng quan về mô hình lập trình ADO.NET với kiến thức trong chương này sẽ giúp sinh viên có cái nhìn tổng quan về lập trình kết nối cơ sở dữ liệu. Chương 2, đưa ra các thành phần của ngôn ngữ SQL sẽ sử dụng trong khi viết chương trình. Chương 3, nêu nội dung cụ thể của các thành phần chính trong ADO.NET và cách sử dụng từng đối tượng. Chương 4, tập trung vào những đối tượng phía ứng dụng như DataSet, DataTable, DataColumn, DataRow, DataView.... Chương 5, hướng dẫn về sử dụng Report làm báo cáo. Chương 6, nêu cơ bản về mô hình lập trình 3 lớp.

Trong mỗi Chương đều có câu hỏi hiểu bài và bài tập thực hành giúp sinh viên ôn lại những kiến thức đã được học trong giờ giảng lý thuyết đồng thời cũng có kiến thức cho sinh viên và tạo điều kiện cho sinh viên có nhiều thời gian thực hành tại nhà.

CHƯƠNG 1. GIỚI THIỆU ADO.NET

Sau khi học xong bài này sinh viên nắm được:

- ADO.NET là gì?
- Liệt kê được các phần chính trong kiến trúc ADO.NET
- Mô tả được chức năng của các đối tượng trong các thành phần chính của kiến trúc ADO.NET.
- Các Data Provider chuẩn của .NET là gì? Các lớp, phương thức, thuộc tính của các Provider này giống và khác nhau như thế nào?
- Cơ chế Disconnected là như thế nào? Ưu điểm của cơ chế này?

Trọng tâm bài giảng

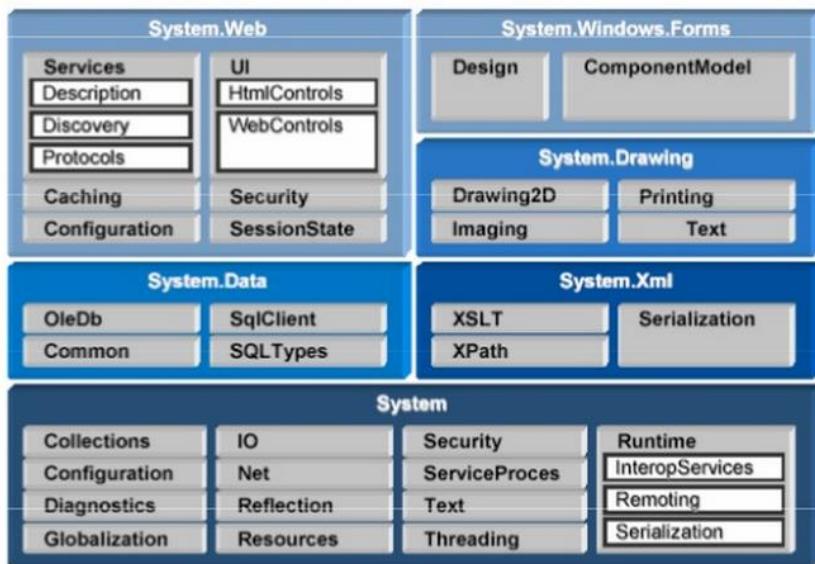
Hiểu được kiến trúc của ADO.NET từ đó thấy được cách thức truy cập nguồn dữ liệu và cách thức lấy dữ liệu về các đối tượng lưu trữ xử lý trong ADO.NET.

Hiểu được chức năng của các đối tượng: Connection, DataAdapter, Command, DataReader, DataSet.

1.1.GIỚI THIỆU

Trong phần đầu này chúng ta sẽ cùng nhau đi tìm hiểu công nghệ ADO.NET của Microsoft. Nội dung thì bao la nên chúng ta sẽ chỉ tìm hiểu sơ lược để cho bạn có thể nắm rõ chức năng cũng như cách thức sử dụng vào thực tế.

Hệ thống namespace trong .NET Framework



1.1.1. ADO.NET Là Gì ?

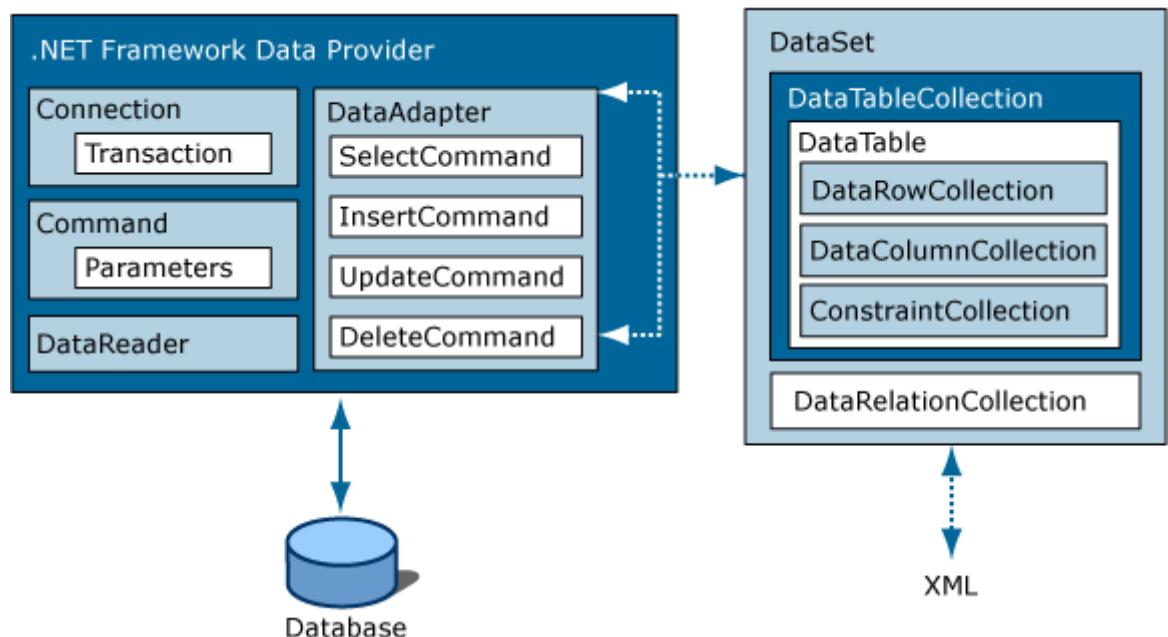
ActiveX Data Object.NET (ADO.NET) là một thư viện phần mềm .NET Framework. Bao gồm các thành phần phần mềm cung cấp dịch vụ truy cập dữ liệu. ADO.NET được thiết kế để cho phép các nhà phát triển viết mã được quản lý cho việc tiếp cận các nguồn dữ liệu bị ngắt kết nối, có thể có quan hệ hoặc không quan hệ (như XML hoặc dữ liệu ứng dụng). Tính năng này của ADO.NET giúp ứng dụng có thể chia sẻ dữ liệu hay các ứng dụng phân tán.

ADO.NET cung cấp kết nối truy cập với CSDL bằng cách sử .NET Managed Provider (mặc định .Net Framework có hai Managed Providers là SQL Managed Provider and OleDb Managed Provider) và truy cập bị ngắt kết nối bằng cách sử dụng Datasets, đó là các ứng dụng sử dụng kết nối cơ sở dữ liệu chỉ trong thời gian truy xuất dữ liệu hoặc cập nhật dữ liệu. Datasets là thành phần giúp đỡ để lưu trữ các dữ liệu liên tục trong bộ nhớ để cung cấp truy cập khi bị ngắt kết nối để sử dụng các nguồn tài nguyên cơ sở dữ liệu một cách hiệu quả và khả năng mở rộng tốt hơn.

ADO.NET được phát triển từ ADO, cũng là một công nghệ tương tự như ADO.NET với một vài thay đổi cấu trúc cơ bản. Mặc dù có một vài trường hợp để làm việc trong chế độ bị ngắt kết nối bằng cách sử dụng ADO, nhưng dữ liệu được chuyển đến CSDL trong ADO.NET hiệu quả hơn bằng cách sử dụng Data Adapters. Các đại diện trong bộ nhớ của dữ liệu giữa ADO và ADO.NET là khác nhau. ADO.NET có thể giữ các dữ liệu trong một bảng kết quả duy nhất, nhưng ADO giữ

nhiều bảng cùng với các chi tiết của mối quan hệ. Không giống như ADO, việc truyền dữ liệu giữa các ứng dụng bằng cách sử dụng ADO.NET không sử dụng COM (Component Object Model) để sắp xếp mà dùng datasets, nó truyền dữ liệu như là một dòng XML.

Kiến trúc ADO.NET dựa trên hai yếu tố chính là : Dataset và .NET Framework data provider



Data provider gồm các thành phần sau :

- Datasets
- Một tập hợp đầy đủ của CSDL bao gồm các table có liên quan , các ràng buộc và mối quan hệ của chúng.
- Chức năng giống như truy cập dữ liệu từ xa từ dịch vụ Web XML
- Thao tác với dữ liệu động
- Xử lý dữ liệu theo kiểu không kết nối
- Cung cấp cho xem thứ bậc XML của dữ liệu quan hệ
- Xử dụng các công cụ XSLT và XPath Query để hoạt động trên dữ liệu

.NET Framework Data Provider bao gồm các thành phần sau đây để thao tác dữ liệu :

- **Connection:** Cung cấp kết nối với nguồn dữ liệu (database). Hay nói cách khác nó là đối tượng có nhiệm vụ thực hiện kết nối tới CSDL.

- **Command:** Thực hiện các thao tác cần thiết với CSDL để lấy dữ liệu, sửa đổi dữ liệu hoặc thực hiện các thủ tục được lưu trữ. Hiểu đơn giản là nó đưa ra các mệnh lệnh đối với CSDL.
- **DataReader:** Cung cấp việc đọc dữ liệu và chỉ đọc 1 dòng dữ liệu tại một thời điểm và nó đọc theo chiều tiến từ đầu đến cuối.
- **DataUpdater:** Nó đóng vai trò như là cầu nối giữa Dataset và CSDL, tải dữ liệu lên dataset hoặc đồng bộ các thay đổi ở dataset về lại CSDL.
- **Datasets:** Để lưu trữ, truy cập từ xa và lập trình với dữ liệu phẳng (Flat), dữ liệu XML và dữ liệu quan hệ

1.1.2. Hiểu Đơn Giản Về ADO.NET

Là công nghệ truy cập dữ liệu cốt lõi cho các ngôn ngữ nhắm vào mức CLR (Common Language Runtime). Sử dụng không gian tên System.Data.SqlClient để truy cập vào SQL Server, hoặc các Providers từ các nhà cung cấp khác để truy cập các dữ liệu của họ. Sử dụng System.Data.ODBC hoặc System.Data.OleDb truy cập dữ liệu từ ngôn ngữ .NET sử dụng công nghệ truy cập dữ liệu khác. Sử dụng System.Data.Dataset khi bạn cần một bộ nhớ cache dữ liệu ẩn trong các ứng dụng của khách hàng (client). ADO.NET là một phần của NET Framework. Để tạo ra các ứng dụng mà sử dụng NET Framework. ADO.NET được xây dựng để hỗ trợ cho cơ sở dữ liệu bao gồm MS SQL Server databases cùng với OLEDB và các nguồn dữ liệu ODBC. Tối ưu hóa đáp ứng cho các nhà cung cấp bên thứ ba.

ADO.NET cho phép:

- Thao tác với CSDL trong cả hai môi trường là **Connected data & Disconnected data.**
- Làm việc tốt với XML
- Tương tác được với nhiều nguồn dữ liệu
- Làm việc trên môi trường internet

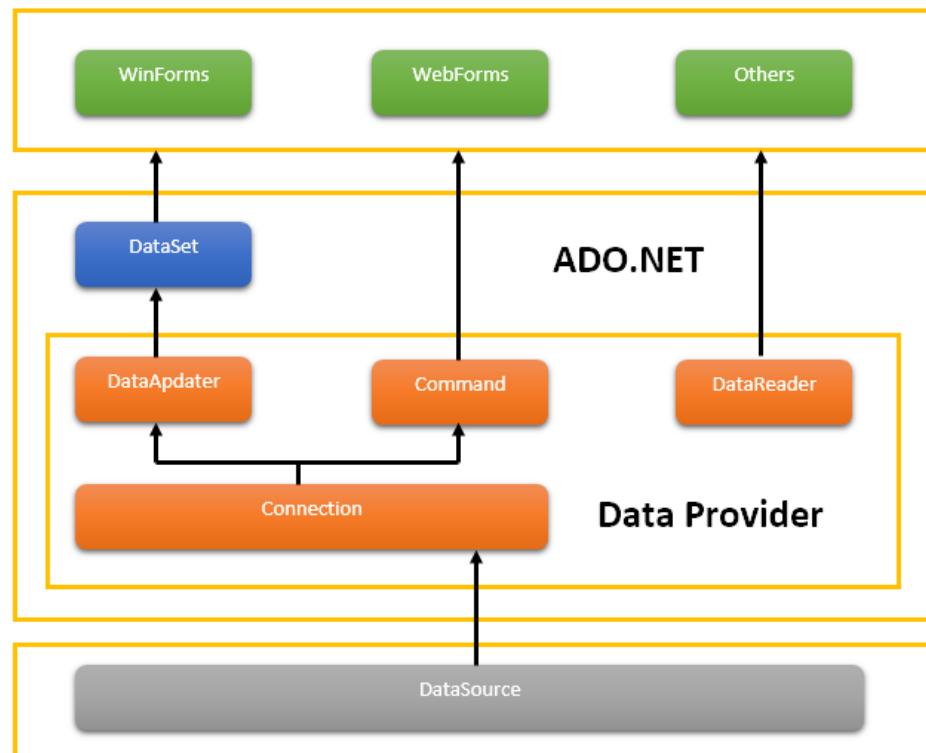
Các class của nó được đặt trong namespace: System.Data

Ba bộ thư viện chính của nó là:

- **System.Data.OleDb:** Dùng để truy xuất bất kỳ CSDL nào hỗ trợ OLEDB

- **System.Data.Odbc:** Dùng truy xuất các nguồn dữ liệu ODBC
- **System.Data.SqlClient:** Dùng để truy xuất đến CSDL SQL Server mà không cần thông qua OLEDB.

ADO.NET là thành phần của .NET nên nó có thể sử dụng hầu như tất cả các ngôn ngữ mà .NET hỗ trợ như: C#, VB.NET, C++ . . .

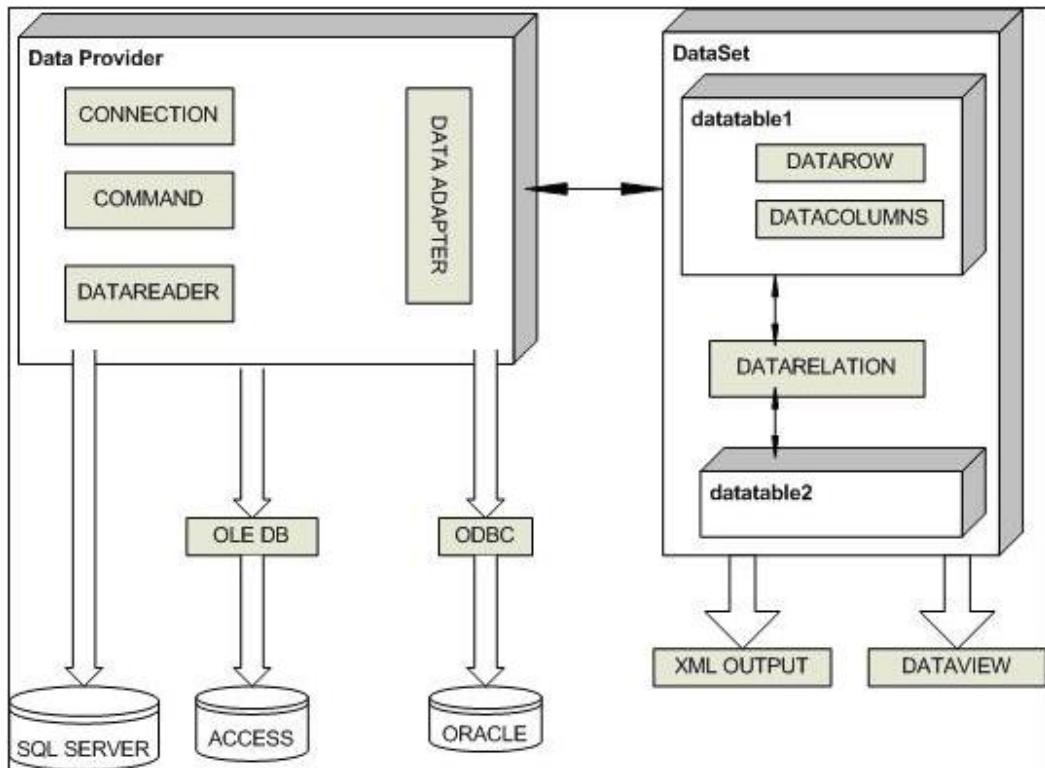


Đặc biệt khi sử dụng ADO.NET bạn có hai lựa chọn:

- Làm việc với môi trường cần kết nối data. Ứng dụng client kết nối tới CSDL và thao tác trên CSDL này nằm ở Server. Không có kết nối thì không truy xuất được dữ liệu
- Làm việc với môi trường không kết nối. Có nghĩa là chúng ta sẽ dùng dataset để mô tả lại các table tương đương với CSDL. Dữ liệu sẽ được lấy từ CSDL lên Dataset và lưu dưới dạng XML. Khi trong điều kiện ko có kết nối tới CSDL, mọi thao tác dữ liệu sẽ được lưu trên Dataset và sau đó có thể đồng bộ với CSDL. Giống như là một bộ nhớ tạm thời nằm tại phía client. Khi có kết nối sẽ lựa chọn đồng bộ với Server

Nhìn tổng thể kiến trúc ADO.Net gồm hai phần:

- **Phần kết nối:** Thực hiện kết nối với CSDL và thực hiện các thao tác.
Sử dụng Connection, Command, DataReader, DataAdapter.
- **Phần ngắt kết nối:** Lấy dữ liệu thông qua DataAdapter vào Dataset để xử lý. Bạn có thể hình dung Dataset là một CSDL nằm trong ứng



1.2.CÁC DATA PROVIDER THÔNG DỤNG:

TÊN DATA PROVIDER	TIỀN TÓ	MÔ TẢ
ODBC Data Provider	Odbc	Data Sources with an ODBC interface. Normally older databases.
OleDb Data Provider	OleDb	Data Sources that expose an OleDb interface, i.e. Access or Excel.
Oracle Data Provider	Oracle	For Oracle Databases.
SQL Data Provider	Sql	For interacting with Microsoft SQL Server.
Borland Data Provider	Bdp	Generic access to many

		databases such as Interbase, SQL Server, IBM DB2, and Oracle.
--	--	---

1.2.1. SQL Data Provider

1.2.1.1. SQL connection

SQL Connection là đối tượng chứa các thông tin kết nối tới Database. Đầu tiên, chúng ta phải tạo chuỗi kết nối tới database trước khi thao tác với nó.

Có 2 dạng chuỗi kết nối (ConnectionString):

```
server=. tenserver; database=tendatabase;
uid=tendangnhap; pwd= matkhau

Data Source=. tenserver;Initial Catalog=tendatabase;User
Id=tendangnhap;Password=matkhau
và các thông tin bảo mật.

String connString = "server=THANHLAM; database=
DemoADO; uid=sa; pwd=123456";
```

Hoặc

```
String connString = "Data Source=THANHLAM;Initial
Catalog=DemoADO;User Id=sa;Password=123456";
Công việc tiếp theo tạo một đối tượng SqlConnection với
tham số truyền vào là chuỗi kết nối (connString):
SqlConnection connect = new SqlConnection(
connString);
```

Chú ý: khi sử dụng Connection trên đối tượng DataReader, các bạn phải đóng/mở Connection một cách rõ ràng và tường minh. Còn khi sử dụng trên mô hình phi kết nối(Disconnect), các bạn không cần phải đóng/mở Connection một cách thủ công, nó sẽ tự động mở khi cần và đóng khi hoàn thành.

1.2.1.2. SQL Command

Quá trình thao tác với Database được thực hiện bởi một đối tượng command. Đối tượng này sẽ gửi câu lệnh SQL (Select, insert, update, delete...) đến Database, command sẽ dựa vào connection để xác định Database nào muốn thao tác. Bạn có thể dùng một command riêng lẻ để thực thi câu lệnh SQL hoặc gán command cho một đối tượng SQL Data Adapter(Sẽ được đề cập ở phần dưới).

Khởi tạo đối tượng Command:

```
String querySelect = "SELECT * FROM SINHVIEN";
SqlCommand command = new SqlCommand(querySelect,
connect);
```

Đối tượng command khai báo ở trên có 2 tham số:

- Tham số thứ nhất: câu truy vấn SQL.
- Tham số thứ hai: một tham chiếu tới connection.

Các phương thức của Command:

- **ExecuteScalar()**: thực thi các câu lệnh mà kết quả trả về chỉ một ô (VD: Select count(*)...).

```
String querySelect = "SELECT COUNT(*) FROM
SINHVIEN";
SqlCommand command = new SqlCommand(querySelect,
connect);
int count = (int)command.ExecuteScalar();
```

Phương thức này trả về một đối tượng Object, vì vậy chúng ta phải ép kiểu.

- **ExecuteReader()**: thực thi các câu lệnh SELECT và kết quả trả về là một DataReader.

```
String querySelect = "SELECT * FROM SINHVIEN";
SqlCommand command = new SqlCommand(querySelect,
connect);
command.ExecuteReader();
```

- **ExecuteNonQuery()**: thực thi các câu lệnh (Update, Delete, Insert....).

```
String querySelect = "UPDATE SINHVIEN SET TEN='Ngo
Thanh Lam' WHERE ID=1";
SqlCommand command = new SqlCommand(querySelect,
connect);
command.ExecuteNonQuery();
```

- **ExecuteXMLReader()**: tạo một bộ đọc file XML (phương thức này không có trong OleDb Data Provider).

```
String querySelect = "SELECT * FROM SINHVIEN";
SqlCommand command = new SqlCommand(querySelect,
connect);
XmlReader xmlReader = command.ExecuteXmlReader();
```

1.2.1.3. SQL Data Reader

Data Reader dùng để đón nhận kết quả của câu lệnh SELECT từ đối tượng command, nó không thể tự khởi tạo bằng từ khóa new. Dữ liệu trả về cho Data Reader là luồng dữ liệu nhanh, chỉ truy xuất dữ liệu ở cấp độ dòng(row) và theo chiều hướng nhất định. Có nghĩa là dữ liệu trong đối tượng Data Reader chỉ được truy xuất 1 chiều và chiều đó bắt đầu từ dòng thứ nhất cho đến dòng cuối cùng của bảng. Thực hiện bởi hàm Read() – là hàm duy chuyển xuống dòng tiếp theo.

Cách đọc dữ liệu:

```
SqlDataReader dr = command.ExecuteReader();
while(dr.Read())
{
    for(int i=0;i<dr.FieldCount; i++)
    {
        Console.WriteLine("{0,20}", dr[i]);
    }
    Console.WriteLine();
}
Console.ReadLine();
```

Trong đó:

- dr.Read(): sẽ trả về giá trị true/false. True khi còn dòng dữ liệu, False khi hết dòng dữ liệu.
- dr.FieldCount(): sẽ trả về số cột trong table.

Đoạn code trên có nghĩa là sẽ đọc nếu còn dòng(dr.Read()==true), và đọc từ cột trái sang cột phải. chúng ta cần lưu ý: nên lưu lại dữ liệu nếu cần thiết, vì khi đọc xong một dòng, sẽ không đọc lại nữa. Nếu cần thao tác với dữ liệu thì nên làm việc với đối tượng Data Set (sẽ được đề cập ở phần tiếp theo).

1.2.1.4. SQL Data Adapter

Data Adapter đóng vai trò là cầu nối giữ Database và Data Set(sẽ được đề cập ở phần tiếp theo). Data Adapter sẽ đỡ dữ liệu vào DataSet bằng phương thức Fill() khi đọc hoặc thay đổi dữ liệu một lượt vào Database.

Data Adapter chứa một tham chiếu tới đối tượng tham chiếu tới connection và tự động đóng/mở kết nối khi đọc hoặc ghi dữ liệu trong Database.

Khởi tạo đối tượng SqlDataAdapter:

```
public SqlDataAdapter(string selectCommandText,  
SqlConnection selectConnection)
```

- **selectCommandText** : Là 1 câu lệnh T-SQL SELECT hoặc 1 stored procedure được sử dụng bởi thuộc tính SelectCommand của SqlDataAdapter.
- **selectConnection** : Là 1 tham chiếu tới đối tượng Connection.

CHƯƠNG 2. CƠ SỞ DỮ LIỆU SQL SERVER

2.1.SQL LÀ GÌ?

SQL là Structured Query Language – Ngôn ngữ Truy vấn có Cấu trúc

- SQL cho phép bạn truy xuất một cơ sở dữ liệu
- SQL là một ngôn ngữ theo chuẩn ANSI
- SQL có thể thực hiện các truy vấn đến một cơ sở dữ liệu
- SQL có thể truy tìm dữ liệu từ một cơ sở dữ liệu
- SQL có thể chèn các mẫu tin mới vào trong một cơ sở dữ liệu
- SQL có thể xóa các mẫu tin trong một cơ sở dữ liệu
- SQL có thể cập nhật các mẫu tin trong một cơ sở dữ liệu
- SQL rất dễ học

SQL là một chuẩn ANSI (American National Standards Institute - Viện Tiêu chuẩn Quốc gia Mỹ) cho các hệ thống truy xuất cơ sở dữ liệu. Các phát biểu SQL dùng để truy tìm và cập nhật dữ liệu trong một cơ sở dữ liệu. SQL làm việc với các trình quản lý cơ sở dữ liệu như Access, DB2, Informix, Microsoft SQL Server, Oracle, Sybase, và nhiều trình khác (đáng tiếc là đa số trong chúng có các phần mở rộng ngôn ngữ SQL riêng).

2.2.CÁC THAO TÁC TRÊN CƠ SỞ DỮ LIỆU

SQL là một cú pháp để thực hiện các truy vấn. Nhưng ngôn ngữ SQL cũng chứa các cú pháp cập nhật các mẫu tin (record), chèn các mẫu tin mới và xóa các mẫu tin đang tồn tại. Các lệnh truy vấn và cập nhật này thuộc dạng Ngôn ngữ Thao tác Dữ liệu (Data Manipulation Language - DML) một phần của SQL:

- **SELECT** – trích dữ liệu từ một cơ sở dữ liệu
- **UPDATE** – cập nhật dữ liệu trong một cơ sở dữ liệu
- **DELETE** – xóa dữ liệu từ một cơ sở dữ liệu
- **INSERT** – chèn dữ liệu mới vào trong một cơ sở dữ liệu

2.3.ĐỊNH NGHĨA DỮ LIỆU SQL

Ngôn ngữ Định nghĩa Dữ liệu (Data Definition Language - DDL) một phần của SQL, cho phép tạo hay xóa các bảng cơ sở dữ liệu. Chúng ta cũng có thể định

nghĩa các chỉ mục (các khóa - key), chỉ định liên kết giữa các bảng, và ràng buộc giữa các bảng cơ sở dữ liệu.

Các phát biểu DDL quan trọng nhất trong SQL là::

CREATE TABLE – tạo một bảng cơ sở dữ liệu mới

ALTER TABLE – thay đổi (alters) một bảng cơ sở dữ liệu

DROP TABLE – xóa một bảng cơ sở dữ liệu

CREATE INDEX – tạo một chỉ mục (khóa tìm kiếm)

DROP INDEX – xoá một chỉ mục

2.4. PHÁT BIỂU SQL SELECT

Một query của SQL dùng để lấy thông tin từ database. Dữ liệu được chứa trong các hàng (rows) của bảng (tables). Hàng (Rows) gồm một nhóm các cột (Columns) chứa dữ liệu tương ứng. Biểu thức query lấy dữ liệu có cấu trúc :

- Một danh sách SELECT (lựa chọn), tại vị trí các cột được gọi để lấy dữ liệu chỉ định.
- Một mệnh đề FROM, xác định bảng (tables) cần truy cập để lấy dữ liệu.

Khi viết các query SQL thì viết hoa, đơn giản là để xác định nó là từ khóa của SQL; SQL không phân biệt chữ hoa chữ thường cho nên nếu viết thường không ảnh hưởng gì cả; chẳng qua viết hoa tránh nhầm lẫn. Cái này theo chuẩn lập trình ANSI để phân biệt thôi.

Viết một query đơn giản

```
SELECT * FROM Employees;
```

Dấu * nghĩa là lấy dữ liệu của tất cả các cột nằm trong bảng. Khi thực thi (!Execute) query này trong “Northwind” thì sẽ thu được tất cả các hàng và cột nằm trong ‘Employees’ Table.

Chú ý: khi làm việc với cơ sở dữ liệu của SQL Server 2008 thì dùng GUI để xác định các thành phần và đặc điểm của cấu trúc khi làm việc với cơ sở dữ liệu.

Query có thể hiểu là làm việc theo kiểu code. Thực tế để code với cơ sở dữ liệu một cách chuẩn thì câu lệnh sẽ rất phức tạp và tốn nhiều dòng.

Chú ý:

Nếu bạn chưa biết thực thi (! Execute) query như thế nào thì làm như sau :

1. Mở Microsoft SQL Server Management Studio 2008 Express ra.
2. Chọn db ‘Northwind’.
3. Chọn nút ‘New Query’ trên Toolbar (thanh công cụ).
4. Viết query vào.
5. Phải chuột và chọn ‘! Execute’. Hay nhấn F5
6. Sẽ thấy kết quả tại tab ‘Results’ ngay bên dưới phần mà bạn vừa code query xong.

Giải thích câu query

Câu query vừa thực hiện là :

```
SELECT * FROM Employees;
```

Nghĩa là : Lấy dữ liệu của tất cả các cột các hàng của bảng ‘Employees’. Hoặc là : lấy tất cả dữ liệu có trong bảng ‘Employees’.

Giả sử muốn lấy dữ liệu của một số columns thôi vì có nhiều cột không cần thiết, lấy vào chỉ tốn tài nguyên khi thực thi thì query như sau :

```
SELECT
    <column 1>,
    <column 2>,
    ...
    <column n>
FROM Employees;
```

Trong đó <column X> là tên column bạn cần lấy dữ liệu. Chẳng hạn từ chỉ muốn lấy tên của nhân viên trong bảng ‘Employees’ thôi thì query như sau :

```
SELECT Lastname
```

```
FROM Employees;
```

Hoặc lấy số ID của nhân viên và họ tên của nhân viên :

```
SELECT
```

```
    employeeid,  
    firstname,  
    lastname
```

```
FROM
```

```
Employees;
```

Giảm bớt số lượng dữ liệu không cần thiết sẽ rất tiết kiệm tài nguyên khi bạn làm việc với một cơ sở dữ liệu lớn, đồ sộ.

2.4.1. Mệnh đề WHERE

Yếu tố thêm trong query này là WHERE để xác định hàng có tính chất nào đó.

2.4.1.1. Cú pháp:

```
WHERE <column1> <operator> <column2>
```

Trong đó : <column1><column2> là tên 2 cột với toán tử so sánh <operator>.

Ví dụ: Mệnh đề WHERE trong câu query trên cơ sở dữ liệu

```
SELECT
```

```
    employeeid,  
    firstname,  
    lastname
```

```
FROM
```

```
Employees
```

```
WHERE
```

```
country = 'USA';
```

Chú ý: Nếu là string thì phải để trong dấu '' như ở trên.

Toán tử so sánh của mệnh đề WHERE

Toán tử	Giải thích	Ví dụ
=	Bằng	EmployeeID = 1
<	Nhỏ hơn	EmployeeID < 1
>	Lớn hơn	EmployeeID > 1
<=	Nhỏ hơn hoặc bằng	EmployeeID <= 1
>=	Lớn hơn hoặc bằng	EmployeeID >= 1
<>, !=	Khác	EmployeeID <> 1
!<	Không nhỏ hơn	EmployeeID !< 1
!>	Không lớn hơn	EmployeeID !> 1

Chú ý: Trong chuẩn SQL không có toán tử ‘!=’; chỉ áp dụng với kiểu dữ liệu T-SQL.

2.4.1.2. Toán tử LIKE

Toán tử này đưa ra kết quả theo kiểu pattern cho trước (cái này liên quan Regular Expression).

Ví dụ

```
WHERE Title LIKE 'Sale%'
```

Sẽ lấy ra tất cả các hàng nào có cột có tiêu đề bắt đầu với cụm từ Sale , có thể là ‘Sale’ , ‘Sale0’, ‘Saleman’...

Có 4 trường hợp để xác định pattern :

1. % : bắt cứ ký tự hoặc một nhóm nào đều hợp lệ kể cả rỗng (empty).
2. _ : một ký tự bất kỳ . Ví dụ : LIKE ‘_ales’ có thể là : Sales, Bales, Cales... nhưng chỉ 1 ký tự.
3. [] : Một vài ký tự xác định cho phép là hợp lệ. Ví dụ: LIKE ‘[bs]ales’ thì chỉ có 2 kết quả là : bales và sales.

4. [^] : kí tự không phải nhóm kí tự xác định. Ví dụ : [^a-h] thì không lấy kí tự nào từ a đến h.

Đôi lúc bạn sẽ thấy những cột chẵng có giá trị gì được gắn cho nó cả, ta gọi là NULL (column is NULL). Vì vậy có toán tử giúp ta xác định các giá trị này :

2.4.1.3. Toán tử IS NULL và IS NOT NULL

IS NULL : Cho phép lấy ra hàng có cột chẵng có giá trị gì.

Ví dụ :

```
WHERE Region IS NULL
```

IS NOT NULL : Cho phép lấy ra hàng có cột có giá trị.

Ví dụ :

```
WHERE Region IS NOT NULL
```

Một query đúng :

```
SELECT * FROM Employees WHERE Region IS NULL;
```

Một query sai :

```
SELECT * FROM Employees WHERE Region = NULL;
```

Lý do sai: vì từ khóa NULL không đi với toán tử =

2.4.1.4. Toán tử BETWEEN và IN

Nhiều lúc muốn lấy kết quả trong một khu vực mình muốn thu hẹp lại (range), ước chừng khoảng thế nào đó. Ta dùng BETWEEN và IN

BETWEEN: Trả về *true* nếu giá trị nằm trong một khoảng.

Ví dụ :

```
WHERE extension BETWEEN 400 AND 500
```

Lấy hàng có cột 'extension' có giá trị trong khoảng 400 và 500

IN : Trả về *true* nếu giá trị nằm trong một danh sách (list). Danh sách có thể là một query con (sub-query)

Ví dụ : `WHERE city IN ('Seattle', 'London')`

Lấy hàng có cột ‘city’ mang giá trị là ‘Seattle’ và ‘London’.

2.4.1.5. Toán tử Logic : AND – NOT – OR

Kết nối hai hay nhiều điều kiện trong một mệnh đề WHERE.

AND : Toán tử AND hiển thị một cột nếu TẤT CẢ các điều kiện liệt kê đều đúng

Ví dụ :

```
WHERE ( title LIKE 'Sale%' AND lastname = 'Peacock'
)
```

NOT :

```
ví dụ : WHERE ( title LIKE 'Sale%' NOT lastname =
'Peacock' )
```

OR : Toán tử OR hiển thị một cột nếu MỘT TRONG các điều kiện liệt kê là đúng

Ví dụ :

```
WHERE ( title = 'Anh Tuấn' OR title = 'Pete' )
```

Cái này dễ hiểu khỏi giải thích. ^^!

2.4.2. Sắp xếp dữ liệu

Khi lọc ra được các dữ liệu muốn tìm nhưng mà nó không theo trật tự nào cả. Bạn muốn kết quả thu được tự sắp xếp theo một hướng nào đó để bạn dễ hiểu dễ nhìn.

2.4.2.1. Dùng mệnh đề ORDER BY

Cú pháp :

```
ORDER BY <column_name> [ASC | DESC] {, n}
```

ASC : Ascending

DESC : Descending

Chú ý: Nếu không có ASC hay DESC thì mặc định (default) là ASC.

Query chung mẫu :

```
SELECT <Danh sách các cột dữ liệu muốn lấy>
FROM < Danh sách bảng lấy dữ liệu >
WHERE <Danh sách điều kiện>
ORDER BY <Tên cột cần sắp xếp> ASC | DESC
```

Một ví dụ về câu lệnh truy vấn (Query)

Yêu cầu:

Lấy các đơn đặt hàng (orders) được nhận bởi nhân viên có id là 5 (employeeid = 5). Đơn đặt hàng chuyển tới Pháp (France) hoặc Brazil. Chỉ lấy thông tin : OrderID, EmployeeID, CustomerID, OrderDate và ShipCountry. Sắp xếp theo nước nhận hàng và ngày đặt hàng

Câu lệnh SQL tương ứng:

```
SELECT
    Orderid,
    Employeeid,
    Customerid,
    Orderdate,
    Shipcountry,
FROM
    Orders
WHERE
    Employeeid = 5
    AND
    Shipcountry IN ( 'Brazil' , 'France' )
ORDER BY
    Shipcountry ASC,
```

Orderdate ASC

2.5.CHÈN THÊM DỮ LIỆU (INSERTING DATA)

Bạn đã biết cách lấy dữ liệu từ cơ sở dữ liệu bây giờ phải biết thêm (insert) dữ liệu vào cơ sở dữ liệu. Để thêm dữ liệu vào dùng câu lệnh ‘INSERT’. Rất đơn giản không phức tạp lắm vì khi thêm dữ liệu đâu cần phải lọc, sắp xếp phân loại nên không xài WHERE và ORDER BY khi thêm dữ liệu.

Cú pháp INSERT :

```
INSERT INTO <table_name>
( <column-1>, <column-2>, ... , <column-N> )
VALUES
( <value-1>, <value-2>, ... , <value-N> )
```

Ví dụ: Thêm vào trong bảng Shippers.

```
INSERT INTO shippers
( CompanyName, Phone )
VALUES
( 'CongDongCViet.COM', '000-123456' );
```

2.6.THAY ĐỔI DỮ LIỆU (UPDATING DATA)

Bây giờ một việc quan trọng là thay đổi dữ liệu. ta dùng câu lệnh ‘UPDATE’. Khi làm việc với câu lệnh ‘UPDATE’ nên rất cẩn thận vì sự thay đổi sẽ có tác dụng ảnh hưởng tất cả các hàng trong mệnh đề WHERE; Nên chú ý điều này.

Cú pháp UPDATE

```
UPDATE <Tên bảng dữ liệu cần cập nhật>
SET
<cột 1> = <giá trị cột 1>,
<cột 2> = <giá trị cột 2>,
...
<cột N> = <giá trị cột N>
```

`WHERE <điều kiện>`

Ví dụ:

```
UPDATE shippers
SET
    CompanyName = N'Pete - Vô danh tiêu tốt'
WHERE
    ShipperID = 4
```

2.7.XÓA DỮ LIỆU (DELETING DATA)

Để xóa dữ liệu ta dùng câu lệnh ‘DELETE’, giống như ‘UPDATE’ bạn cần cẩn thận với query này.

Cú pháp DELETE

```
DELETE FROM <Bảng cần xóa dữ liệu>
WHERE <Điều kiện>
```

Ví dụ:

```
DELETE FROM Shippers
WHERE ShipperID = 4
```

Chú ý:

Chỉ thực hiện được câu lệnh xóa khi dữ liệu trên dòng dữ liệu muốn xóa không có liên quan đến bảng dữ liệu khác (là khóa ngoại của bảng dữ liệu khác).

Nhiều lúc bạn muốn xóa từng hàng trong bảng dữ liệu thì TRUNCATE là sự lựa chọn tốt hơn là DELETE. Vì khi xóa mỗi hàng DELETE thường log lại thông tin xóa trong khi TRUNCATE thì không log lại gì.

2.8.KIỂU DỮ LIỆU T-SQL

T-SQL cung cấp một cơ sở kiểu dữ liệu rất tốt – có chính xác 27 kiểu. Trong đó có cả UDT (User-defined data type : kiểu dữ liệu người dùng định nghĩa).

2.8.1. Dữ liệu kiểu số (Numeric Data Types)

Có 8 kiểu số trong T-SQL và tương ứng với C#.

Kiểu dữ liệu SQL	Kiểu tương ứng trong	Giải thích
------------------	----------------------	------------

	C#	
Bigint	Long	64-bit integer
Bit	Bool	0,1 hoặc NULL
Decimal (numeric)	Decimal	128-bit signed integer
Float	Double	64-bit floating-point num
Int	Int	32-bit signed integer
Real	Float	32-bit floating-point num
Smallint	Short	16-bit signed integer
Tinyint	Byte	8-bit unsigned integer

2.8.2. Dữ liệu kiểu tiền tệ (Money Data Types)

Kiểu dữ liệu SQL	Kiểu tương ứng trong C#	Giải thích
Money	Decimal	Giá trị trong khoảng : -922,337,203,685,477.5808 đến 922,337,203,685,477.5807
Smallmoney	Decimal	Giá trị trong khoảng : -214,748.3648 Đến 214,748.3647

2.8.3. Kiểu dữ liệu chuỗi kí tự (Character String Data Types)

Kiểu dữ liệu SQL	Kiểu tương ứng trong C#	Giải thích
Char	String	1->8k bytes
Nchar	String	1->4k bytes
Text	String	1->231-1 characters
Ntext	String	1->230-1 bytes
Varchar	String	1->231-1 bytes
Nvarchar	String	1->231-1 bytes

2.8.4. Kiểu dữ liệu ngày giờ (Date and Time Data Types)

Kiểu dữ liệu SQL	Kiểu tương ứng trong C#	Giải thích
Datetime	SqlDateTime	Từ Jan-01-1753 đến Dec-31-9999, chính xác 1/300 giây. Tương đương kiểu timestamp của chuẩn SQL
Smalldatetime	SqlDateTime	Từ Jan-01-1900 đến Jun-06-2079, chính xác đến phút.

Kiểu dữ liệu nhị phân (Binary Data Types)

Kiểu dữ liệu SQL	Kiểu tương ứng trong C#	Giải thích
Binary	Byte[]	Fixed_1->8k bytes
Image	Byte[]	Var_0->231-1 bytes
Varbinary	Byte[]	Var_0->231-1 bytes

2.8.5. Kiểu dữ liệu khác (Other Data Types)

Kiểu dữ liệu SQL	Kiểu tương ứng trong C#	Giải thích
Cursor		Sử dụng trong SQL Server
Sq1_variant	Object	Chứa int, binary, char
Table		Sử dụng trong SQL Server
Timestamp	Byte[]	8-byte int db unique
Uniqueidentifier	System.Guid	128-bit unique int
Xml	String	Chứa XML đến 2GB

2.8.6. Độ ưu tiên các kiểu dữ liệu (Data Type Precedence)

Sắp xếp theo thứ tự ưu tiên cao đến thấp

Kiểu dữ liệu SQL	Độ ưu tiên	Kiểu dữ liệu SQL	Độ ưu tiên
UDT	1	Tinyint	14
Sq1_variant	2	Bit	15
Xml	3	Ntext	16
Datetime	4	Text	17
Smalldatetime	5	Image	18
Float	6	Timestamp	19
Real	7	Uniqueidentifier	20
Decimal (numeric)	8	Nvarchar	21
Money	9	Nchar	22
Smallmoney	10	Varchar	23
Bigint	11	Varbinary	24
Int	12	binary	25
Smallint	13		

2.9. STORED PROCEDURE

Nội dung chính bao gồm những phần sau:

- Hướng dẫn cú pháp Stored Procedure trong SQL Server.
- Cách gọi, sử dụng lại các Stored Procedure đã tạo.
- Một số ví dụ Stored Procedure.

2.9.1. Stored Procedure cơ bản

Khi chúng ta tạo một ứng dụng với Microsoft SQL Server, ngôn ngữ lập trình T-SQL (Transact-SQL) là ngôn ngữ chính giao tiếp giữa ứng dụng và database của SQL Server. Khi chúng ta tạo các chương trình bằng T-SQL, hai phương pháp chính có thể dùng để lưu trữ và thực thi cho các chương trình là:

- Chúng ta có thể lưu trữ các chương trình cục bộ và tạo các ứng dụng để gọi các lệnh đến SQL Server và xử lý các kết quả.
- Chúng ta có thể lưu trữ những chương trình như các stored procedure trong SQL Server và tạo ứng dụng để gọi thực thi các stored procedure và xử lý các kết quả.

Đặc tính của Stored-procedure trong SQL Server:

- Stored Procedure là hàm cho phép truyền tham số vào và trả về giá trị.
- Bao gồm 1 tập các lệnh T-SQL để xử lý 1 chức năng nào đó trong cơ sở dữ liệu.

Ta có thể dùng T-SQL EXECUTE để thực thi các stored procedure. Stored procedure khác với các hàm xử lý (User-defined Function) là giá trị trả về của chúng không chứa trong tên và chúng không được sử dụng trực tiếp trong biểu thức.

Stored procedure có những thuận lợi so với các chương trình T-SQL lưu trữ cục bộ là:

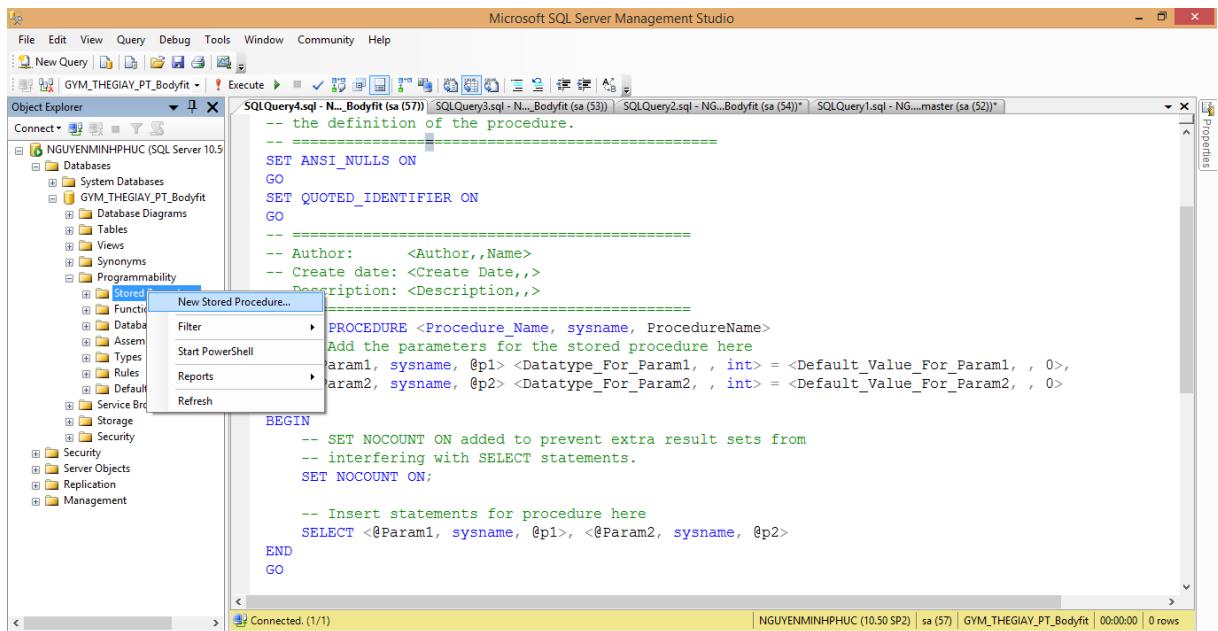
- **Stored procedure cho phép điều chỉnh chương trình cho phù hợp:**
Chúng ta có chỉ tạo stored procedure một lần và lưu trữ trong database một lần, trong chương trình chúng ta có thể gọi nó với số lần bất kỳ.
Stored procedure có thể được chỉ rõ do một người nào đó tạo ra và sự

thay đổi của chúng hoàn toàn độc lập với source code của chương trình.

- **Stored procedure cho phép thực thi nhanh hơn:** nếu sự xử lý yêu cầu một đoạn source code Transact – SQL khá lớn hoặc việc thực thi mang tính lặp đi lặp lại thì stored procedure thực hiện nhanh hơn việc thực hiện hàng loạt các lệnh Transact-SQL. Chúng được phân tích cú pháp và tối ưu hóa trong lần thực thi đầu tiên và một phiên bản dịch của chúng trong đó sẽ được lưu trong bộ nhớ để sử dụng cho lần sau, nghĩa là trong những lần thực hiện sau chúng không cần phải phân tích cú pháp và tối ưu lại, mà chúng sẽ sử dụng kết quả đã được biên dịch trong lần đầu tiên.
- **Stored procedure có thể làm giảm bớt vấn đề kẹt đường truyền mạng:** giả sử một xử lý mà có sử dụng hàng trăm lệnh của Transact-SQL và việc thực hiện thông qua từng dòng lệnh đơn, như vậy việc thực thông qua stored procedure sẽ tốt hơn, vì nếu không khi thực hiện chúng ta phải gửi hàng trăm lệnh đó lên mạng và điều này sẽ dẫn đến tình trạng kẹt mạng.
- **Stored procedure có thể sử dụng trong vấn đề bảo mật của máy:** vì người sử dụng có thể được phân cấp những quyền để sử dụng các stored procedure này, thậm chí họ không được phép thực thi trực tiếp những stored procedure này.

2.9.2. Tạo Stored Procedure

2.9.2.1. Tạo bằng SQL Server Management Studio (SSMS)



Hình 1. Tạo Stored Procedure bằng chương trình SSMS

Tạo bằng giao diện SSMS sẽ phát sinh script tạo Stored Procedure sẵn, ta chỉ cần thêm nội dung vào trong phần thân hàm.

2.9.2.2. Tạo bằng script

Cú pháp đơn giản:

```

CREATE PROCEDURE procedure_name
@parameter1 data_type [output] /*các tham số*/,
@parameter2 data_type [output]
AS
BEGIN
[khai báo các biến cho xử lý]
{Các câu lệnh transact-sql}
END
GO
  
```

Phần [output] là phần có thể có hoặc không để xác định loại tham số.

Ví dụ:

```
CREATE PROCEDURE XinChao
@hoTen  nvarchar(50)
AS
BEGIN
print N'Xin chào ' + @hoTen
END
GO
```

Hay viết cách khác

```
CREATE PROC Hello
AS
BEGIN
print N'Hello ' + @hoTen
END
GO
```

2.9.3. Thực thi Stored Procedure

2.9.3.1. Thực thi bằng giao diện SSMS

Sử dụng lệnh EXECUTE (có thể viết tắt là EXEC) để thực thi một stored procedure

```
EXECUTE procedure_name parameter_value1,
parameter_value2, . . .
```

```
EXEC procedure_name parameter_value1, parameter_value2,
. . .
```

Ví dụ:

```
EXEC XinChao N'Minh Phúc'
```

Đoạn lệnh trên sẽ tạo kết quả như sau

```

SQLQuery5.sql - N...unghiep (sa (56))*
CREATE PROCEDURE XinChao
@hoTen nvarchar(50)
AS
BEGIN
print N'Xin chào ' + @hoTen
END
GO

EXEC XinChao N'Minh Phúc'

```

Messages

Xin chào Minh Phúc

2.9.4. Thay đổi nội dung Stored Procedure

Cú pháp:

```

ALTER PROCEDURE procedure_name
@parameter1 data_type [output] /*các tham số*/,
@parameter2 data_type [output]
AS
BEGIN
[khai báo các biến cho xử lý]
{Các câu lệnh transact-sql}
END
GO

```

Lúc này, SQL Server sẽ thay thế stored procedure có tên “procedure_name” bằng 1 stored procedure mới có cùng tên

2.9.5. Xóa Stored Procedure

Cú pháp:

```

DROP PROCEDURE procedure_name
hay

```

```
DROP PROC procedure_name
```

2.9.6. Tham số trong Stored Procedure

Stored Procedure là 1 hàm được lưu trữ sẵn trong cơ sở dữ liệu. Hàm này có thể có 2 loại tham số chính: tham số đầu vào và tham số đầu ra

2.9.6.1. Tham số đầu vào

Đây là loại tham số mặc định, cho phép truyền các giá trị vào trong stored procedure để hỗ trợ xử lý.

Ví dụ:

```
CREATE PROC Cong
@So1 int,
@So2 int
AS
BEGIN
declare @Kq int
set @Kq = @So1 + @So2
print @Kq
END
GO
exec Cong 1, 2
```

Kết quả đoạn lệnh

The screenshot shows the SQL Query window with the following content:

```
SQLQuery5.sql - N...unghiep (sa (56))*
CREATE PROC Cong
@So1 int,
@So2 int
AS
BEGIN
declare @Kq int
set @Kq = @So1 + @So2
print @Kq
END
GO
exec Cong 1, 2
```

Below the query window, the Messages pane displays the number "3", indicating the successful execution of the stored procedure.

2.9.6.2. Tham số đầu ra

Tham số dùng để nhận kết quả trả về từ stored procedure. Sử dụng từ khóa OUTPUT (hoặc viết tắt là OUT) để xác định tham số.

Ví dụ:

```
create PROC Tru
@So1 int,
@So2 int,
@Kq int output
AS
BEGIN
set @Kq = @So1 - @So2
END
GO
DECLARE @test int
EXEC Tru 1, 2, @test output
PRINT @test
```

The screenshot shows the SQL Query window with the following code:

```
SQLQuery5.sql - N...unghiep (sa (56))*
create PROC Tru
@So1 int,
@So2 int,
@Kq int output
AS
BEGIN
set @Kq = @So1 - @So2
END
GO
DECLARE @test int
EXEC Tru 1, 2, @test output
PRINT @test
```

Below the query window, there is a 'Messages' pane showing the output: '-1'

2.9.7. Trả về giá trị trong Stored Procedure

Ngoài cách sử dụng tham số đầu ra để trả về giá trị. Có thể sử dụng RETURN để trả về giá trị từ stored procedure hoặc các câu lệnh SELECT khi truy vấn dữ liệu.

2.9.7.1. Trả về giá trị từ lệnh RETURN

Lệnh RETURN được sử dụng để trả về giá trị từ stored procedure mà không cần sử dụng tham số đầu ra. Giá trị trả về này có một số đặc điểm:

- Giá trị trả về chỉ có thể là số nguyên. Nếu trả về các loại giá trị khác thì lúc thực thi stored procedure sẽ báo lỗi (ngoại trừ 1 số kiểu dữ liệu được tự động chuyển đổi sang kiểu số nguyên như:float, double,...).
- Giá trị trả về mặc định là 0.
- Có thể nhận giá trị trả về này bằng 1 biến.
- Sau khi gọi RETURN, stored procedure sẽ trả về giá trị và kết thúc xử lý.

Ví dụ:

```
CREATE PROC Test
@Lenh int
AS
BEGIN
if (@Lenh = 1)
return 1
if (@Lenh = 2) begin
declare @float float
set @float = 2.6
return @float
end
if (@Lenh = 3) begin
declare @char varchar(50)
set @char = 'hello'
return @char
end
END
```

GO

Nếu giá trị truyền vào là 1: stored procedure trả về giá trị “1”.

Nếu giá trị truyền vào là 2: stored procedure trả về giá trị “2”.

Nếu giá trị truyền vào là 3: stored procedure báo lỗi không thể chuyển chuỗi ‘hello’ thành số nguyên.

Nếu truyền các giá trị khác: stored procedure trả về giá trị “0”.

2.9.7.2. Trả về dữ liệu từ lệnh SELECT

Mỗi lệnh SELECT đặt trong stored procedure sẽ trả về 1 bảng.

Ví dụ:

```
CREATE PROC TestSelect
AS
BEGIN
SELECT * FROM SINHVIEN
SELECT * FROM LOP
END
GO
EXEC TestSelect
```

2.9.8. Kết hợp Stored Procedure với các lệnh T-SQL

Các stored procedure thông thường được tạo ra nhằm giúp thực hiện một số chức năng cần thao tác trong cơ sở dữ liệu. Khi đó, ta cần phải kết hợp nhiều lệnh T-SQL thao tác với dữ liệu như (SELECT, INSERT, UPDATE, DELETE) và các cấu trúc điều khiển (IF, WHILE, CASE,...)

Ví dụ: Ứng dụng thêm sinh viên vào cơ sở dữ liệu.

```
CREATE PROC ThemSinhVien
@mssv varchar(10),
@hoTen nvarchar(100),
@namSinh int,
@danToc nvarchar(20),
@maLop varchar(10)
AS
BEGIN
IF EXISTS(SELECT * FROM SinhVien s WHERE s.ma = @mssv)
BEGIN
PRINT N'Mã số sinh viên ' + @mssv + N' đã tồn tại'
RETURN -1
END
IF NOT EXISTS(SELECT * FROM Lop L WHERE L.ma = @maLop)
BEGIN
PRINT N'Mã số lớp ' + @maLop + N' chưa tồn tại'
RETURN -1
END
INSERT INTO SinhVien(ma, hoTen, namSinh, danToc, maLop)
VALUES (@mssv, @hoTen, @namSinh, @danToc, @maLop)
RETURN 0 /* procedure tự trả về 0 nếu không RETURN */
END
GO
DECLARE @kq INT
```

```

EXEC @kq = ThemSinhVien '0212005', N'Nguyễn Văn A', 1987,
'Kinh',
'TH2002/01'

PRINT @kq

```

Ví dụ 2: Ứng dụng trả về danh sách sinh viên trong lớp

```

CREATE PROC XuatDanhSachSinhVien
@maLop  varchar(10)
AS
BEGIN
IF(NOT EXISTS(SELECT * FROM Lop L WHERE L.ma = @maLop))
BEGIN
PRINT N'Mã số lớp ' + @maLop + N' chưa tồn tại'
RETURN -1
END
SELECT * FROM Lop l where l.ma = @maLop
/*procedure luôn trả về 0 nếu không RETURN*/
END

GO

```

CHƯƠNG 3. CÁC THÀNH PHẦN ADO.NET

Mục đích - Yêu cầu

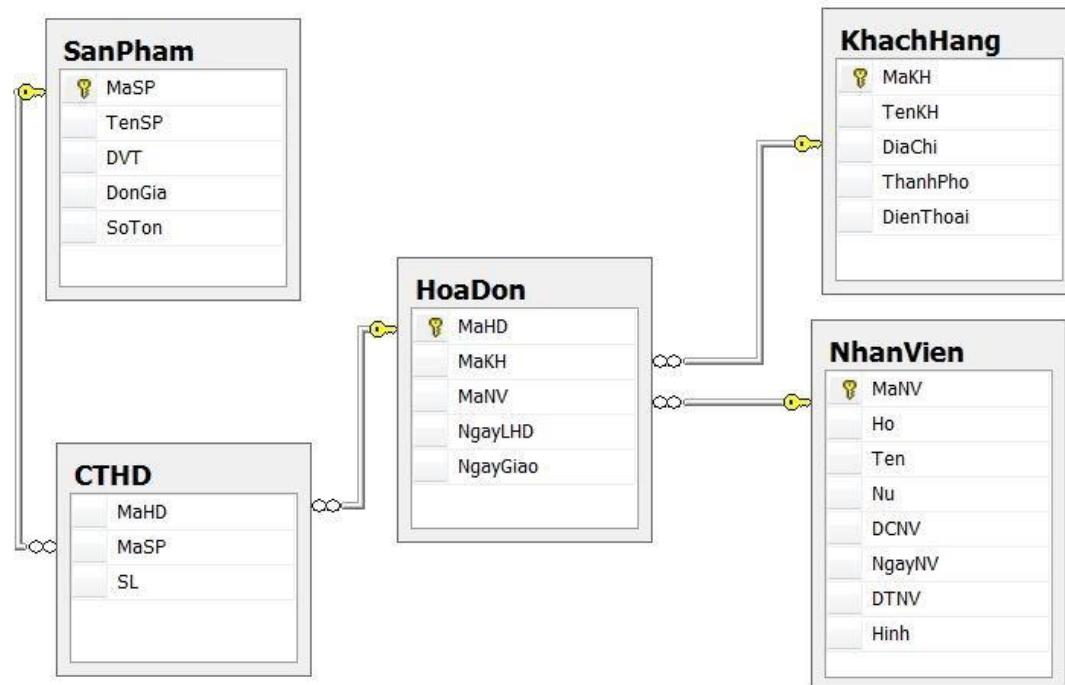
Sau khi học xong bài này sinh viên nắm được:

- Cách kết nối các hệ CSDL như SQL Server, Access,...
- Biết và sử dụng được các thuộc tính, phương thức của các đối tượng quan trọng trong kiến trúc ADO.NET như: Connection, Command, DataReader, DataAdapter, DataSet.
- Đọc dữ liệu và lấy dữ liệu với đối tượng DataReader.
- Cập nhật dữ liệu với các lệnh SQL tương ứng, cùng với các đối tượng Connection, Command, DataAdapter.
- Biết cách kết hợp thành thao các đối tượng lập trình như SqlConnection, SqlCommand, SqlDataAdapter, DataSet, DataTable.

3.1.GIỚI THIỆU

Ở chương 1 chúng ta đã biết được các khái niệm ban đầu về ADO.NET, biết được kiến trúc cũng như chức năng chính của các thành phần trong ADO.NET, ... Chương này chúng ta sẽ lần lượt tìm hiểu chi tiết các thành phần đó.

Chúng ta sẽ sử dụng cơ sở dữ liệu HoaDon để minh họa cho các ví dụ trong toàn bộ bài giảng của môn học. CSDL HoaDon có Diagram như sau:



Hình 3.1: Diagram CSDL Hóa Đơn

ADO.NET cung cấp cho ta một số Provider để làm việc hiệu quả với các CSDL khác nhau. Tuy nhiên, về phía lập trình thì cách thức sử dụng chúng rất giống nhau. Do vậy, trong bài giảng này chỉ giới thiệu Provider là SQL Server.

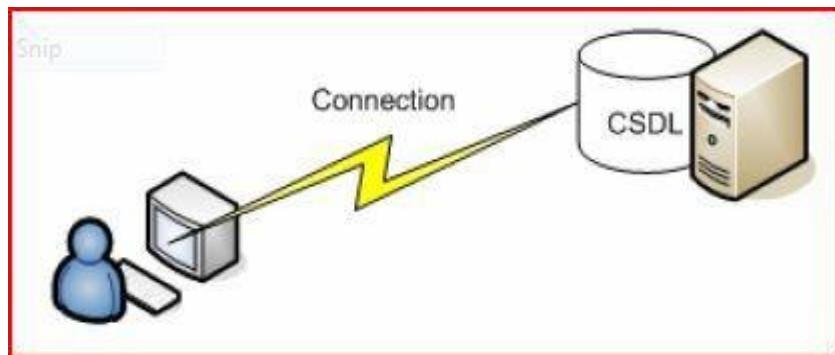
3.2.ĐỐI TƯỢNG CONNECTION

Khi làm việc với dữ liệu của cơ sở dữ liệu SQL Server bằng ngôn ngữ lập trình C#, bạn sử dụng đối tượng SqlConnection. Đối tượng này có chức năng kết nối với cơ sở dữ liệu nhờ các thuộc tính và phương thức mà chúng có.

3.2.1. Kết nối cơ sở dữ liệu SQL Server

Mỗi loại cơ sở dữ liệu sẽ sử dụng một không gian tên tương ứng. Trong trường hợp làm việc với cơ sở dữ liệu SQL Server thì không gian tên là System.Data.SqlClient. Bước đầu tiên và quan trọng nhất khi làm việc với CSDL là kết nối từ ứng dụng đến SQL Server,

và để thực hiện công việc này chúng ta sử dụng đối tượng SqlConnection. Các bước khai báo và thực hiện lệnh kết nối như sau:



Hình 3.2: Hình ảnh kết nối đến CSDL

3.2.1.1. Khai báo và Sử dụng đối tượng SQLConnection

Sử dụng không gian tên, bạn khai báo trên phần đầu của class bằng từ khóa using như sau:

```
using System.Data.SqlClient;
```

Sau khi khai báo không gian tên cho cơ sở dữ liệu SqlServer để kết nối đến cơ sở dữ liệu thì ta tiến hành khai báo biến đối tượng SqlServer có tên là sqlConnection như sau:

```
SqlConnection _sqlconnection;
```

Và để sử dụng đối tượng này bạn cần phải khởi tạo nó:

```
_sqlconnection = new SqlConnection();
```

Lưu ý: lớp SqlConnection có hai Constructor được cài đặt Overload là SqlConnection và SqlConnection(connectionString).

Với khai bao new SqlConnection() thì bạn phải cung cấp chuỗi kết nối cơ sở dữ liệu thông qua thuộc tính ConnectionString. Trong trường hợp bạn muốn truyền chuỗi kết nối cơ sở dữ liệu thông qua tham số của Constructor thứ hai thì bạn khai báo như sau:

```
_sqlconnection = new SqlConnection(connectionString);
```

Trong đó connectionString là biến chứa chuỗi kết nối.

Ví dụ cho hai trường hợp này:

```
SqlConnection _sqlconnection;
string _connectionString = "server=.;database=hoadon;integrated
security=true";
public Form1()
{
```

```

        _sqlconnection = new SqlConnection (_connectionString);
    }
}

```

Hoặc

```

SqlConnection _sqlconnection;
string _connectionString = "server=.;database=hoadon;integrated
security=true";
public Form1()
{
    InitializeComponent();
    _sqlconnection = new SqlConnection();
    _sqlconnection.ConnectionString = _connectionString;
}

```

3.2.1.2. Chuỗi kết nối cơ sở dữ liệu

Chuỗi kết nối cơ sở dữ liệu phụ thuộc vào hình thức kết nối cơ sở dữ liệu. Khi kết nối cơ sở dữ liệu SQL Server, bạn có thể sử dụng đặc quyền của hệ điều hành hoặc của SQL Server. Với đặc quyền của hệ điều hành thì người dùng sử dụng quyền của tài khoản hiện hành truy cập hệ điều hành để đăng nhập vào cơ sở dữ liệu Sql Server. Trong trường hợp sử dụng đặc quyền của SQL Server thì cần được cung cấp tài khoản và mật khẩu của cơ sở dữ liệu để đăng nhập.

Các thông tin kết nối được đặt trong một chuỗi, Provider sẽ tự động phân tích nội dung chuỗi để lấy ra thông số kết nối thích hợp.

Kết nối theo đặc quyền của hệ điều hành

Đối với trường hợp này thì chuỗi kết nối cơ sở dữ liệu không tồn tại hai thuộc tính User Id và Password, Thay vào đó, chuỗi kết nối sẽ có thuộc tính Integrated Security=SSPI hay Integrated Security=True.

```

string _connectionString = "server=.;database=hoadon;integrated
security=true";
hay
string _connectionString = "server=.;database=hoadon;integrated
security=SSPI";

```

Ngoài ra, thuộc tính database còn có thể thay thế bằng Initial Catalog.

Sau đây là một số thuộc tính của chuỗi kết nối

Data Source hoặc **Server** là tên máy chủ hay địa chỉ IP máy chủ nơi CSDL hoạt động;

Database hoặc **Initial Catalog** là tên của CSDL chúng ta muốn sử dụng; Trong một hệ quản trị có rất nhiều cơ sở dữ liệu khác nhau.

Connection Timeout qui định thời gian chờ của ADO.NET cho kết nối thành công, nếu quá thời gian chờ này mà vẫn chưa kết nối được chúng ta sẽ nhận được thông báo lỗi ngoại lệ Connection Timeout;

Integrated Security=True hoặc **Integrated Security=SSPI** nếu chúng ta muốn cơ chế đăng nhập SQL Server sử dụng quyền chứng thực của hệ điều hành Windows.

Kết nối theo đặc quyền của hệ SQL Server

Trường hợp chúng ta muốn cơ chế đăng nhập sử dụng quyền chứng thực của SQL Server thì cần thay **Integrated Security=True** bằng 2 thông số **User ID**: Tài khoản được tạo bởi SQL Server (giả sử là sa: Đây là tài khoản cao nhất trong SQL Server) và **PWD**: Mật khẩu của tài khoản muốn kết nối (giả sử là bimat), khi đó thông tin chuỗi kết nối sẽ như sau:

```
conn.ConnectionString = "Server=.; Database=HoaDon; Connection
Timeout=60; User ID = sa; Password = hihipass"
```

hay có thể viết tắt như sau:

```
conn.ConnectionString = "Server=.; Database=HoaDon; Connection
Timeout=60; Uid = sa; Pwd = hihipass"
```

Connection cần phải được mở trước khi thực thi các thao tác với CSDL: Select, Insert, Update và Delete.

Connection cần phải được đóng khi kết thúc thao tác với CSDL để giải phóng tài nguyên cho hệ thống sau khi đã thực thi các thao tác với CSDL.

Ví dụ: Sau đây ta sẽ thực hiện kết nối đến Database HoaDon trong SQL Server bằng hai cơ chế chứng thực: của Windows và của SQL Server.



Hình 3.3: Tùy chọn cơ chế chứng thực

```
using System.Data.SqlClient;
```

.....

◊ Code của Button “Connect” như sau:

```
private void btnconnect_Click(object sender, EventArgs e)
{
    try
    {
        if (rdwin.Checked == true)
        {
            connectionstring = "Server=" + txtserver.Text;
            connectionstring += ";database=" + txtdatabase.Text;
            connectionstring += ";integrated security=true";
            cnn.ConnectionString = connectionstring;
        }
        else
        {
            connectionstring = "Server=" + txtserver.Text;
            connectionstring += ";database=" + txtdatabase.Text;
            connectionstring += ";uid=" + txtuser.Text;
            connectionstring += ";pwd=" + txtpass.Text;
            cnn.ConnectionString = connectionstring;
        }
        cnn.Open();
        MessageBox.Show("ket noi thanh cong");
        cnn.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("loi ket noi" + ex.Message);
    }
}
```

```

    }
}
```

3.2.2. Tập tin lưu chuỗi kết nối

Để cho phép người sử dụng có thể cấu hình giá trị cho các thuộc tính trong chuỗi kết nối cơ sở dữ liệu, bạn có thể sử dụng các loại định dạng tập tin như sau: *.ini, *.txt. Tuy nhiên khi làm việc với .NET, bạn có thể sử dụng tập tin App.config với cấu trúc nội dung được tổ chức theo dạng XML

Lưu ý: Tập tin App.Config là tập tinh có định dạng XML được giới thiệu từ phiên bản .NET 1.0 nó cho phép bạn khai báo các thuộc tính của chuỗi kết nối cơ sở dữ liệu SQL

Chẳng hạn, bạn có thể khai báo các thuộc tính của chuỗi kết nối cơ sở dữ liệu SQL Server trong tập tin SQLServer.ini như sau:

```

serverName=(local)
instanceName=SQLEXPRESS
databaseName=Hoadon
userName=
password=
portNo
timeOut=
```

Để đọc được các thuộc tính của chuỗi kết nối này phải sử dụng đến lớp StreamReader trong namespace System.IO.

Trong trường hợp sử dụng file App.config thì cấu trúc file như sau

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add name="SqlServer"
      connectionString="Server=(local);database=hoadon;uid=sa;pwd=hihipass"
      providerName="System.Data.SqlClient"></add>
  </connectionStrings>
</configuration>
```

Hay theo dạng như sau:

```

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key ="server" value="(local)"/>
    <add key="instance" value="" />
    <add key="database" value="hoadon" />
```

```

<add key="port" value="" />
<add key="timeout" value="" />
</appSettings>
</configuration>

```

Sau đây là code dùng để đọc chuỗi kết nối từ tập tin *.ini và tập tin app.config. Code được xây dựng trong lớp có tên là SQLServer gồm các thuộc tính và phương thức với chức năng đọc chuỗi kết nối từ tập tin ini và app.config với hai đặc quyền truy cập nói ở trên.

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration;
using Manager = System.Configuration.ConfigurationManager;
using Settings = System.Configuration.ConnectionStringSettings;
using System.IO;
using System.Data.SqlClient;

namespace ketnoi
{
    public class SQLServer
    {
        #region bienthanhvien
        private string connectionString = "";
        // SqlConnection
        public string ConnectionString
        {
            get { return connectionString; }
            set { connectionString = value; }
        }
        private string iniFile = "";

        public string IniFile
        {
            get { return iniFile; }
            set { iniFile = value; }
        }
        // thuộc tính dùng kiểm tra quyền đăng nhập
        private bool windowNT = false;

        public bool WindowNT
        {
            get { return windowNT; }
            set { windowNT = value; }
        }
        private string serverName = "(local)";

        public string ServerName
        {
            get { return serverName; }
            set { serverName = value; }
        }
    }
}

```

```

        }
        private string instanceName = "";

        public string InstanceName
        {
            get { return instanceName; }
            set { instanceName = value; }
        }
        private string databaseName = "";

        public string DatabaseName
        {
            get { return databaseName; }
            set { databaseName = value; }
        }
        private string userName = "sa";

        public string UserName
        {
            get { return userName; }
            set { userName = value; }
        }
        private string password = "";

        public string Password
        {
            get { return password; }
            set { password = value; }
        }
        private string connectionTimeout = "";

        public string ConnectionTimeout
        {
            get { return connectionTimeout; }
            set { connectionTimeout = value; }
        }
        private string portNo = "";

        public string PortNo
        {
            get { return portNo; }
            set { portNo = value; }
        }
#endregion

#region Constructor
public SQLServer()
{
}
public SQLServer(string connectstr)
{
    ConnectionString = connectstr;
}
public SQLServer(string server, string database, string userid,

```

```

string password)
{
    serverName = server;
    databaseName = database;
    userName = userid;
    passWord = password;
}
public SQLServer(string server, string database)
{
    serverName = server;
    databaseName = database;
}
public SQLServer(bool appfile,string filename)
{
    if(!appfile)
        iniFile = filename;
}

#endregion

#region method
//phương thức dùng để tạo ra chuỗi kết nối từ các thuộc
//tính trong chuỗi kết nối.
/// <summary>
/// Ý nghĩa : Nối các thuộc tính thành chuỗi kết nối.
/// </summary>
/// <returns></returns>
public string GetconnectionString()
{
    string conString = "";
    //trường hợp có instance
    if (!instanceName.Equals(""))
    {
        serverName += @"\\" + instanceName;
    }
    //hàm if kiểm tra quyền đăng nhập
    if (windowNT)
    {
        conString = "server=" + serverName + ";database=" +
databaseName + ";Integrated Security= true";
    }
    //đăng nhập theo quyền của SQL
    else
    {
        conString = "server=" + serverName + ";database=" +
databaseName + ";uid=" + userName + ";pwd=" + passWord;
    }
    //kiểm tra giá trị thuộc tính timeout
    if (!connectionTimeout.Equals(""))
    {
        conString += ";connection Timeout=" + connectionTimeout;
    }
    //kiểm tra port
    if (!portNo.Equals(""))
    {

```

```

        conString += ";Port=" + portNo;
    }
    return conString;
}
/// <summary>
/// Ý nghĩa: Lấy chuỗi kết nối từ file App.config
/// </summary>
/// <param name="nodeName">Tên của node chứa chuỗi kết nối trong
file AppConfig</param>
/// <returns></returns>
public string GetConnectionWithAppConfig(string nodeName)
{
    string StrApp = "";
    //namespace sử dụng để đọc giá trị trong file xml

    Settings settings;
    settings = Manager.ConnectionStrings[nodeName];
    //namespace Manager dùng đọc giá trị trong tab có tên là
nodeName trong file config
    StrApp = settings.ConnectionString;

    return StrApp;
}
//hàm đọc chuỗi kết nối trong file ini theo từng thuộc tính.
/// <summary>
/// Ý nghĩa : Lấy chuỗi kết nối trong file ini
/// </summary>
/// <returns></returns>
public string GetConnectionWithinifile()
{
    string StrApp = "";
    using (StreamReader sReader = new StreamReader(iniFile))
    {
        string line = "";
        while ((line=sReader.ReadLine())!=null)
        {
            switch
((line.Substring(0,line.IndexOf("="))).ToLower())
            {
                case "serverName":
                    serverName = GetValue(line);
                    break;
                case "databaseName":
                    databaseName = GetValue(line);
                    break;
                case "instanceName":
                    instanceName = GetValue(line);
                    break;
                case "portNo":
                    portNo = GetValue(line);
                    break;
                case "timeout":
                    connectionTimeout = GetValue(line);
                    break;
            }
        }
    }
}
```

```

        }
        StrApp = GetconnectionString();
    }
    return StrApp;
}
/// <summary>
/// Ý nghĩa: Lấy chuỗi kết nối từ file ini
/// Có chỉ đường dẫn cụ thể.
/// </summary>
/// <param name="path">Đường dẫn file ini</param>
/// <returns></returns>
public string GetConnectionWithinifile(string path)
{
    string StrApp = "";
    using (StreamReader sReader = new StreamReader(path))
    {
        string line = "";
        while ((line = sReader.ReadLine()) != null)
        {
            switch ((line.Substring(0,
line.IndexOf("="))).ToLower())
            {
                case "serverName":
                    serverName = GetValue(line);
                    break;
                case "databaseName":
                    databaseName = GetValue(line);
                    break;
                case "instanceName":
                    instanceName = GetValue(line);
                    break;
                case "portNo":
                    portNo = GetValue(line);
                    break;
                case "timeout":
                    connectionTimeout = GetValue(line);
                    break;
            }
        }
        StrApp = GetconnectionString();
    }
    return StrApp;
}
//hàm lấy giá trị trong từng thuộc tính trong file ini
/// <summary>
/// Lấy giá trị trên từng dòng trong file ini
/// </summary>
/// <param name="line">dữ liệu từng dòng (string)</param>
/// <returns></returns>
private string GetValue(string line)
{
    string stringvalue = "";
    if (!line.Equals(""))
        stringvalue = line.Substring(line.LastIndexOf("=") + 1);
    return stringvalue;
}

```

```

        }

        //hàm đọc chuỗi kết nối trong file config theo dạng từng thuộc
tính.
        /// <summary>
        /// Ý nghĩa: Lấy giá trị từ File App.Config lấy theo từng giá
tri
        /// </summary>
        /// <param name="server">Server</param>
        /// <param name="instance">Đối tượng sql</param>
        /// <param name="database">Tên Database cần kết nối</param>
        /// <param name="userid">Người dùng</param>
        /// <param name="password">Mật khẩu</param>
        /// <param name="port">Cổng</param>
        /// <param name="timeout">Thời gian kết nối</param>
        /// <returns></returns>
        public string GetConnectionStringWithAppConfig(string server,
string instance, string database, string userid, string password, string
port, string timeout)
        {
            string strApp = "";

            //doc thuc tinh servername
            serverName = Manager.AppSettings.Get(server);
            //doc thuoc tinh instance

            instanceName = Manager.AppSettings.Get(instance);
            //doc thuoc tinh database
            databaseName = Manager.AppSettings.Get(database);
            //doc thuoc tinh username
            userName = Manager.AppSettings.Get(userid);
            //doc thuoc tinh pass
            passWord = Manager.AppSettings.Get(password);
            //doc thuoc tinh port
            portNo = Manager.AppSettings.Get(port);
            //doc thuoc tinh timeout
            connectionTimeout = Manager.AppSettings.Get(timeout);
            strApp = GetConnectionString();

            return strApp;
        }

#endregion
    }
}

```

3.2.3. Tham khảo một vài chuỗi kết nối CSDL của các .Net Data Provider khác

- ✓ OdbcConnection: namespace System.Data.Odbc
- ✓ OleDbConnection: namespace System.Data.OleDb

- ✓ OracleConnection: namespace System.Data.OracleClient

Kết nối đến Access

Ví dụ:

```
using System.Data.OleDb;
OleDbConnection _oledbconnection = new OleDbConnection();
    _oledbconnection.ConnectionString = "Data
Source='D:\DoaDon.mdb';
Provider='Microsoft.Jet.OLEDB.4.0';";
    _oledbconnection.Open()

..... Các lệnh truy xuất đến CSDL: Select, Insert, Update, Delete

    _oledbconnection.Close()
```

Kết nối đến Excel Ví dụ:

```
using System.Data.Odbc;
OdbcConnection _odbcconnection = new OdbcConnection();
    _odbcconnection.ConnectionString = "
DBQ=D:\HoaDon.xls;Driver={Driver do Microsoft Excel(*.xls)} ";
    _odbcconnection.Open()

..... Các lệnh truy xuất đến CSDL: Select, Insert, Update, Delete

    _odbcconnection.Close()
```

3.2.4. Mở và đóng kết nối cơ sở dữ liệu

Sau khi khai báo và khởi tạo đối tượng SqlConnection với chuỗi kết nối vừa trình bày ở trên, để mở kết nối cơ sở dữ liệu bạn gọi phương thức Open() với khai báo như sau:

```
SqlConnection.Open();
```

Chú ý: để kiểm tra tính hợp lệ của người sử dụng khi đăng nhập cơ sở dữ liệu, bạn chỉ cần khai báo để mở cơ sở dữ liệu với chuỗi kết nối cơ sở dữ liệu theo hai loại đặc quyền hệ điều hành và SQL Server.

Sau khi mở kết nối để sử dụng và sau khi hoàn tất các tác vụ xử lý với cơ sở dữ liệu ta tiến hành đóng kết nối bằng phương thức close() với cú pháp như sau

```
SqlConnection.Close();
```

Code hướng dẫn xây dựng lớp kết nối

Class này cung cấp các phương thức cho phép kiểm tra trạng thái kết nối và mở đóng kết nối khi sử dụng. trong class này cũng sử dụng class SQLServer ở trên để đọc chuỗi kết nối

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace SqlServer
{
    public class Connection
    {
        public static SqlConnection sqlconnection;
        SQLServer constring = new SQLServer();

        static Connection()
        {
            sqlconnection = new SqlConnection();
        }
        private static string strError = "";
        public static string Error
        {
            get{return strError;}
        }
        private static string connectionstring=
constring.GetConnectionString("SqlServer");
;
        public static string ConnectionString
        {
            get { return connectionstring; }
            set { connectionstring = value; }
        }
        //phuong thức
        public static bool Connect()
        {
            bool IsExit = false;
            try
            {
                sqlconnection.ConnectionString = connectionstring;
                sqlconnection.Open();
                IsExit = true;
            }
            catch (Exception ex)
            {
                strError = "Error:" + ex.Message;
            }
            return IsExit;
        }
        //phuong thuc ket noi lai
        /// <summary>
        /// phuong thuc su dung de ket noi lai
        /// trong truong hop muon mo mot ket noi khac
        /// </summary>
        /// <returns></returns>
        public static bool ReConnect()
    }
}

```

```
        {
            bool IsExit = false;
            if (sqlconnection.State==ConnectionState.Open)
            {
                sqlconnection.Close();
            }
            connectionstring = ConnectionString;
            sqlconnection.ConnectionString = connectionstring;
            try
            {
                sqlconnection.Open();
                IsExit = true;
            }
            catch (Exception ex)
            {
                strError = "Error:" + ex.Message;

            }
            return IsExit;
        }
        //dòng kết nối
        public static void CloseConnection()
        {
            if (sqlconnection.State != ConnectionState.Closed)
            {
                sqlconnection.Close();
                sqlconnection.Dispose();
            }
        }
        //hàm kiểm tra trạng thái kết nối
        public static bool SqlConnectionState()
        {
            if (sqlconnection.State == ConnectionState.Open)
                return true;
            else
                return false;
        }
    }
}
```

Câu hỏi hiểu bài

- Đối tượng Connection như thế nào thì các câu lệnh T-SQL mới có thể tác động đến CSDL?
 - Để sử dụng Provider cho SQL Server ta phải imports namespace nào?
 - Từ khóa **Data Source** hoặc **Server** trong chuỗi kết nối có ý nghĩa gì?
 - Ngoài từ khóa **Database** ta còn có từ khóa tương đương **Initial Catalog**, vậy chúng mang ý nghĩa gì trong chuỗi kết nối?
 - Connection Timeout=60** trong chuỗi kết nối là như thế nào? Chuỗi kết nối nhất thiết phải có cụm từ khóa này không?

6. Có bao nhiêu cơ chế chứng thực khi truy xuất vào data của một CSDL?
7. **Integrated Security = ?** Giá trị tại dấu hỏi có thể là gì?
8. **UID, PWD** lần lượt là user name và password của máy chứa CSDL cần truy xuất đúng hay sai? Nếu sai vậy nó là gì?
9. Tại sao phải đóng kết nối sau khi đã kết nối thành công và thực hiện điều ta muốn làm?
10. Chuỗi kết nối đối với các Provider khác như Access, Excel, ... giống, khác với Provider của SQL như thế nào?
11. Tại sao cần phải xây dựng chuỗi kết nối động trong file XML?
12. Giả sử ta có chuỗi kết nối “**DataSource=.; Data base=HoaDon; Integrated Security=SSPI**”, chuỗi kết nối là đúng hay sai, nếu sai hãy chỉ chỗ sai và sửa lại cho đúng.
13. Hai chuỗi kết nối sau là tương đương nhau đúng hay sai?
 - Chuỗi 1 “**Server=.; Initial Catalog=HoaDon; Integrated Security=True**”
 - Chuỗi 2 “**Data Source=.; Database=HoaDon; Integrated Security=True**”

Bài tập tại lớp

14. Viết chương trình cho nhập thông tin tên máy chứa CSDL, tên database, tên tài khoản SQL và mật khẩu của tài khoản, có hai tùy chọn chứng thực bởi Windows và bởi SQL, một nút nhấn sẽ kiểm tra việc kết nối đến cơ sở dữ liệu, xuất thông báo kết quả là kết nối được hay không.
15. Viết chương trình với chức năng như trên nhưng dùng Provider OleDb
(namespace System.Data.OleDb)

Bài tập về nhà

16. Viết chương trình kết nối đến CSDL là một file access bất kỳ, xuất thông báo cho biết là có kết nối được hay không
17. Viết chương trình kết nối đến CSDL là một Sheet Excel bất kỳ, xuất thông báo cho biết là có kết nối được hay không.
18. Viết chương trình cho nhập thông tin tên máy chứa CSDL, tên database , tên tài khoản SQL và mật khẩu của tài khoản, có hai tùy chọn chứng thực bởi Windows và bởi SQL, một nút nhấn sẽ kiểm tra

việc kết nối đến cơ sở dữ liệu, xuất thông báo kết quả là kết nối được hay không. Chuỗi kết nối lấy từ File app.conf (file .XML)

3.3. ĐỐI TƯỢNG COMMAND

Trong phần trước chúng ta đã tìm hiểu về đối tượng SqlConnection dùng để kết nối đến cơ sở dữ liệu SQL Server. Trong phần này chúng ta tiếp tục tìm hiểu chi tiết về đối tượng SqlCommand ứng với đối tượng SqlConnection dùng để thực thi phát biểu SQL hay thủ tục nội tại Stored Procedure (như đã nêu ở chương 2) của SQL Server.

SqlCommand là đối tượng cho phép truy cập CSDL và thực thi phát biểu SQL hay thủ tục Store Procedure của CSDL, truyền tham số và trả về dữ liệu. Các lệnh chỉ thực thi sau khi đối tượng Connection được thiết lập thành công (tức là đã kết nối đến CSDL)

3.3.1. Đối tượng SqlCommand

3.3.1.1. Khai báo và khởi tạo

Để khai báo đối tượng SqlCommand, bạn sử dụng cú pháp tương tự như sau:

```
SqlCommand _sqlcommand;
```

Sau khi khai báo như trên, để sử dụng đối tượng này, bạn cần khai báo khởi tạo chúng với 1 trong 4 Constructor với cú pháp như sau:

```
_sqlcommand = new SqlCommand();
```

Hay

```
_sqlcommand=new SqlCommand(string commandtext);
```

Hay

```
_sqlcommand=new SqlCommand(string commandText,SqlConnection  
sqlconnection);
```

Hay

```
_sqlcommand=new SqlCommand(string commandText,SqlConnection  
sqlConnection,SqlTransaction sqltransaction);
```

3.3.1.2. Một số thuộc tính thường dùng của Command .

Đối tượng SqlCommand cung cấp một số thuộc tính cho phép bạn đa dạng hóa các tham số truyền vào đối tượng. sau đây là danh sách các thuộc tính thường sử dụng như: CommandText, CommandType, CommandTimeout, và Connection

CommandText

Cho phép bạn khai báo phát biểu SQL thay vì truyền vào đối tượng này thông qua tham số của Constructor

Chẳng hạn, bạn có thể khai báo thuộc tính CommandText ứng với phát biểu SQL dạng select như sau:

```
string strquery = "Select MaKhachHang,TenKhachHang from khachhang";
cmd = new SqlCommand();
cmd.CommandText=strquery;
```

CommandType

Bạn có thể chọn một trong ba giá trị enum là Text, TableDirect và Stored Procedure.

Trong đó, Command.Text ứng với phát biểu SQL, CommandType.TableDirect ứng với tên bảng dữ liệu và CommandType.StoredProcedure ứng với tên thủ tục nội tại của SQL Server. Giá trị mặc định là CommandType.Text.

Chẳng hạn, trong trường hợp bạn muốn thực thi thủ tục nội tại của Sql Server thì khai báo như sau:

```
SqlCommand cmd = new SqlCommand();
cmd.CommandText = "tên thủ tục nội tại";
cmd.CommandType = CommandType.StoredProcedure;
```

Lưu ý: Trong trường hợp thủ tục nội tại yêu cầu truyền tham số thì bạn sử dụng SqlParameterCollection hay đối tượng SqlParameter sẽ được trình bày trong phần kế tiếp.

CommandTimeout

Cho phép bạn khai báo thời gian tính bằng giây (mặc định là 30 giây) ứng với khoảng thời gian chờ cho đến khi thực thi phát biểu.

Chẳng hạn, bạn cho phép thực thi thủ tục này với thời gian là 60 giây thi khai báo như sau:

```
SqlCommand cmd = new SqlCommand();
cmd.CommandText = "tên thủ tục nội tại";
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandTimeout = 60;
```

Connection

Mỗi đối tượng SqlCommand được sử dụng phải kèm theo đối tượng SqlConnection, chính vì vậy khi khởi tạo đối tượng SqlCommand, bạn có thể sử dụng thuộc tính Connection thay vì truyền đối tượng SqlConnection thông qua Constructor.

Ví dụ: có thể khai báo và sử dụng đối tượng SqlConnection với thuộc tính connection như sau:

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnn;
cmd.CommandText = "tên thủ tục nội tại";
cmd.CommandType = CommandType.StoredProcedure;
cmd.CommandTimeout = 60;
```

Parameters

Là thuộc tính cho phép bạn truyền tham số vào trong thủ tục nội tại của SQL ví dụ khi ta có một câu lệnh Sql sử dụng tham số ta sẽ khai báo như sau:

```
SqlConnection conn = New SqlConnection();
conn.ConnectionString =
ConfigurationSettings.AppSettings("strconn")
conn.Open();

SqlCommand cmd = New SqlCommand();
cmd.Connection = conn;
cmd.CommandType = CommandType.Text;
cmd.CommandText = "Select count(*) From SanPham Where DVT
= @dvt and SoTon < @soton"
cmd.Parameters.Add("@dvt", SqlDbType.NVarChar).Value =
"Kg"
cmd.Parameters.Add("@soton", SqlDbType.Int).Value = 50
int i = cmd.ExecuteScalar();
conn.Close();
MessageBox("Sản Phẩm có DVT là Kg và số tồn nhỏ hơn
50 có:
" & i.ToString());
```

3.3.1.3. Các phương thức thường dùng của SqlCommand.

Đối tượng SqlCommand cung cấp 4 phương thức chính, cho phép bạn thực thi lệnh SQL bao gồm phát biểu SQL (Select, Delete, Update, Insert, Alter, Drop, Create,...) thủ tục nội tại của SQL Server và tên bảng dữ liệu, chúng bao gồm: ExecuteScalar, ExecuteNonQuery, ExecuteReader và ExecuteXmlReader.

Lưu ý, nếu bạn làm việc với Server, Database, cấu trúc Table, View, StoredProcedure, Trigger,... thì nên sử dụng đối tượng tương ứng thuộc không gian tên Microsoft.SqlServer.Management.Smo thay vì sử dụng đối tượng SqlCommand, mặc dù

đối tượng này cung cấp phương thức cho phép bạn thực thi phát biểu SQL hay thủ tục nội tại hệ thống để làm việc với các đối tượng cơ sở dữ liệu.

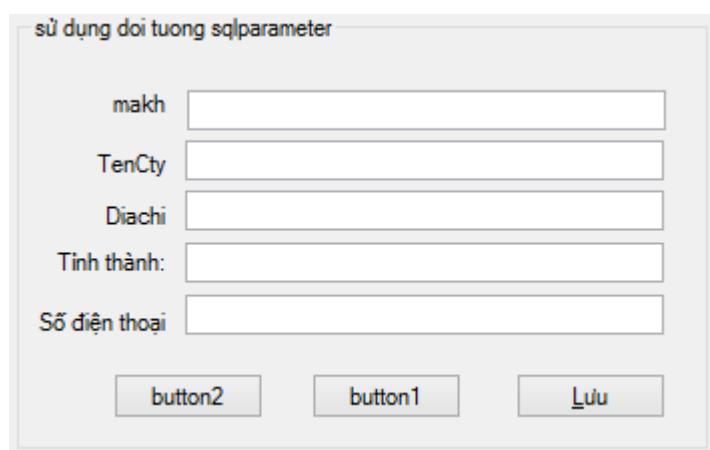
Phương thức ExecuteNonQuery

Phương thức này thực thi phát biểu SQL, thủ tục nội tại của SQL Server, tên bảng dữ liệu và trả về số mẫu tin thực được dạng số int với cú pháp như sau:

```
cmd.ExecuteNonQuery();
int records= cmd.ExecuteNonQuery();
```

Trong đó, records là số mẫu tin thực thi được nếu bạn sử dụng phát biểu SQL.

Một ví dụ sử dụng phương thức để thực thi phương thức insert để thêm dữ liệu vào trong bảng:



```
SqlConnection cnn = new SqlConnection(strconn);
try
{
    cnn.Open();
    string strSQL = "insert into khachhang
(makh,tencty,diachi,thanpho,dienthoai)+"+
"values(@makh,@tencty,@diachi,@thanpho,@dienthoai)";
    cmd = new SqlCommand(strSQL, cnn);
    //khai tao doi tuong sqlparameter
    //cot dau ma khach hang
    sqlparam = new SqlParameter();
    sqlparam.ParameterName = "@makh";
    sqlparam.SqlDbType = SqlDbType.NVarChar;
    sqlparam.Size = 10;
    sqlparam.SqlValue = txtmakh.Text;
    cmd.Parameters.Add(sqlparam);
    //cot thu hai ten cong ty
    sqlparam = new SqlParameter("@tencty",
    SqlDbType.NVarChar, 40);
    sqlparam.SqlValue = txttencty.Text;
    cmd.Parameters.Add(sqlparam);
    //cot thu 3 diachi
```

```

        sqlparam = new SqlParameter("@diachi",
SqlDbType.NVarChar, 60);
        sqlparam.SqlValue = txtdiachi.Text;
        cmd.Parameters.Add(sqlparam);
//cot thu 4
        sqlparam = new SqlParameter("@thanhpho",
SqlDbType.NVarChar, 40);
        sqlparam.SqlValue = txtthanhpho.Text;
        cmd.Parameters.Add(sqlparam);
//cot thu 5
        sqlparam = new SqlParameter("@dienthoai",
SqlDbType.NVarChar, 10);
        sqlparam.SqlValue = txtsodienthoai.Text;
        cmd.Parameters.Add(sqlparam);
//thuc thi cau lenh ExecuteNonQuery
        cmd.ExecuteNonQuery();
        cnn.Close();
        cnn.Dispose();
        MessageBox.Show("thanh cong");
        reset();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

```

Phương thức ExecuteScalar

Trong khi phương thức ExecuteNonQuery dùng để thực thi phát biểu SQL và trả về số mẫu tin thực thi được. Phương thức ExecuteScalar cũng dùng để thực thi phát biểu SQL nhưng giá trị là kiểu object.

Thông thường phương thức ExecuteScalar được sử dụng để thực hiện phát biểu SQL hay thủ tục nội tại SQL và nhận giá trị trả về là hàng và cột thứ nhất. ta thực hiện theo cú pháp sau:

```
object obj=sqlCommand.ExecuteScalar();
```

Phương thức này thực hiện lệnh của Command và chỉ trả về giá trị của cột đầu tiên và dòng đầu tiên (ô đầu tiên trong bảng). Chúng ta thường sử dụng phương thức này khi muốn Command thực hiện các hàm tính toán thống kê như SUM, COUNT, AVG, MAX, MIN,... trên nguồn dữ liệu ngay lúc thực thi.

Ví dụ

```

SqlConnection conn = New SqlConnection();
conn.ConnectionString =
ConfigurationSettings.AppSettings("strconn");
conn.Open();

SqlCommand cmd = New SqlCommand();

```

```

cmd.Connection = conn;
cmd.CommandText = "Select count(*) From SanPham
Where DVT = @dvt";
cmd.Parameters.Add(New SqlParameter("@dvt", "Kg"));
object i = cmd.ExecuteScalar();
conn.Close();
MessageBox.Show("Sản Phẩm có DVT là Kg: " &
i.ToString());

```

Phương thức ExecuteReader

Khác với hai phương thức vừa trình bày ở trên, phương thức ExecuteReader trả về tập dữ liệu chỉ đọc một chiều và sử dụng đối tượng SqlDataReader để nắm giữ tập dữ liệu đó.

Phương thức này trả về một đối tượng DataReader, thuộc tính CommandText là một câu lệnh Select.

Ví dụ:

```

cnn = new SqlConnection(strconn);
cnn.ConnectionString = strconn;
try
{
    cnn.Open();
    cmd = new SqlCommand();
    cmd.CommandText = "select * from khachhang where
makh= "+"""+textBox1.Text+""";
    // cmd.Parameters.Add("@makh", SqlDbType.NVarChar,
    10).Value = textBox1.Text;
    cmd.Connection = cnn;
    SqlDataReader sqldatareader;
    sqldatareader = cmd.ExecuteReader();
    if (sqldatareader.HasRows)
    {
        textBox2.Text =Convert.ToString(
sqldatareader.GetString(1));
        button1.Text = "Update";
        MessageBox.Show("có dữ liệu");
    }
    else
    {
        button1.Text = "Save";
        MessageBox.Show("khong co du lieu");
    }
    sqldatareader.Close();
    sqldatareader.Dispose();
    cnn.Close();
    cnn.Dispose();
}
catch (Exception ex)
{

```

```

        MessageBox.Show(ex.Message);
    }
}

```

Phương thức ExecuteXmlReader

Tương tự phương thức ExecuteReader, phương thức ExecuteXmlReader trả về một tập dữ liệu có định dạng XML và sử dụng đối tượng XmlReader để nắm giữ tập dữ liệu đó.

Ví dụ:

```

SqlConnection cnn = new SqlConnection(connectionString);
cnn.Open();
string sql = "select * from khachhang";
SqlCommand cmd = new SqlCommand(sql, cnn);
XmlReader _xmlreader = cmd.ExecuteXmlReader();
 XmlDocument doc = new XmlDocument();
doc.Load(_xmlreader);
doc.Save(@"c:\test.xml");
_xmlreader.Close();
cnn.Close();

```

Câu hỏi hiểu bài

1. Nêu chức năng của đối tượng Command?
2. Nêu các cách tạo đối tượng Command.
3. Một số thuộc tính thường dùng của Command là gì?
4. Một số phương thức thường dùng của Command là gì?
5. Thuộc tính CommandText nhận giá trị có thể là gì?
6. Thuộc tính CommandType có thể là những loại nào và ý nghĩa mỗi loại?
7. Thuộc tính Parameters giữ vai trò gì trong một command, trong một command chỉ sử dụng một Parameters đúng hay sai?
8. Các phương thức thường dùng của command.
9. Khi nào dùng phương thức ExecuteReader.
10. Cách truy xuất tập kết quả trả về bởi phương thức ExecuteReader có gì đặc biệt?
11. Khi nào dùng phương thức ExecuteScalar.

12. Có phát biểu cho rằng phương thức ExecuteScalar trả về duy nhất một giá trị kiểu số nguyên, phát biểu ấy đúng hay sai và vì sao?
13. Phương thức ExecuteNonQuery dùng trong những trường hợp nào? Giá trị trả về của phương thức này là gì?
14. So sánh ba phương thức ExecuteReader, ExecuteScalar và ExecuteNonQuery về cách dùng (dùng khi nào?)

Bài tập tại lớp

15. Viết chương trình dùng đối tượng command, cho biết có bao nhiêu sản phẩm trong bảng sản phẩm. Câu truy vấn dùng viết thành thủ tục procedured.
16. Viết chương trình dùng đối tượng command, cho biết có bao nhiêu nhân viên trong bảng sản phẩm. Câu truy vấn dùng viết thành thủ tục procedured.
17. Viết chương trình dùng đối tượng command, cho biết sản phẩm có đơn giá lớn nhất. Câu truy vấn dùng viết thành thủ tục procedured.
18. Viết chương trình dùng đối tượng command, cho biết sản phẩm có đơn giá nhỏ nhất. Câu truy vấn dùng viết thành thủ tục procedured.

Bài tập về nhà

19. Viết chương trình dùng đối tượng command, cho biết sản phẩm có số tồn lớn nhất. Câu truy vấn dùng viết thành thủ tục procedured.
20. Viết chương trình dùng đối tượng command, cho phép thêm mới sản phẩm, thông tin sản phẩm nhập từ bàn phím. Câu truy vấn dùng viết thành thủ tục procedured.
21. Viết chương trình dùng đối tượng command, cho phép thêm mới nhân viên, thông tin nhân viên nhập từ bàn phím. Câu truy vấn dùng viết thành thủ tục procedured.
22. Viết chương trình dùng đối tượng command, cho phép xóa sản phẩm vừa được thêm mới, sản phẩm cần xóa nhập từ bàn phím. Câu truy vấn dùng viết thành thủ tục procedured.

23. Viết chương trình dùng đối tượng command, cho phép xóa nhân viên vừa được thêm mới, nhân viên cần xóa nhập từ bàn phím. Câu truy vấn dùng viết thành thủ tục procedured.
24. Viết chương trình cập nhật thông tin các sản phẩm có đơn vị tính là “tan” thành “ta”. Câu truy vấn dùng viết thành thủ tục procedured.
25. Viết chương trình cập nhật thông tin các nhân viên có địa chỉ ở “Biên Hòa” thành “Biên Hòa – Đồng Nai”. Câu truy vấn dùng viết thành thủ tục procedured.

Code mẫu để xây dựng lớp dùng chung datalayer

Lớp này xây dựng với mục đích tạo ra các phương thức thực thi của đối tượng command như ExecuteReader, ExecuteScalar, ExecuteNonQuery.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace DLLDungChung
{
    /// <summary>
    /// Lớp này sử dụng để thực hiện các phương thức của đối tượng
    /// SQL: SqlConnection, SqlCommand, SqlDataAdapter...
    /// Là lớp ở tầng kết nối dữ liệu trong mô hình 3 lớp.
    /// </summary>
    public class Database
    {
        private SqlConnection cnn;
        private SqlDataAdapter da;
        private SqlCommand cmd;
        DocchuoiketnoiSQLSerVer constring = new
DocchuoiketnoiSQLSerVer();
        /// <summary>
        /// Hàm tạo lấy chuỗi kết nối trong tập tin App.config
        /// </summary>
        public Database()
        {
            try
            {
                string connectring =
constring.GetConnectionString("SqlServer");
                cnn = new SqlConnection(connectring);
                cmd = cnn.CreateCommand();
            }
        }
    }
}
```

```

        catch (SqlException sqle)
    {
        System.Windows.Forms.MessageBox.Show(sqle.Message);
    }

}

/// <summary>
/// Hàm tạo lấy chuỗi kết nối với tham số
/// </summary>
/// <param name="sqlserver">tên của tab chứa chuỗi kết nối trong
file App.Config</param>
public Database(string sqlserver)
{
    try
    {
        string connectring =
constring.GetConnectionWithAppConfig(sqlserver);
        cnn = new SqlConnection(connectring);
        cmd = cnn.CreateCommand();
    }
    catch (SqlException sqle)
    {
        System.Windows.Forms.MessageBox.Show(sqle.Message);
    }

}

/// <summary>
/// hàm tạo cho phép lấy chuỗi kết nối từ file *.ini
/// </summary>
/// <param name="Path">đường dẫn tập tin *.ini</param>
/// <param name="t">true hay false đều được</param>
public Database(string Path, bool t)
{
    try
    {
        string connectring =
constring.GetConnectionWithinifile(Path);
        cnn = new SqlConnection(connectring);
        cmd = cnn.CreateCommand();
    }
    catch (SqlException sqle)
    {
        System.Windows.Forms.MessageBox.Show(sqle.Message);
    }

}

/// <summary>
/// Phương Thực Thực hiện NonQuery sử dụng có các câu lệnh
insert, delete, update.
/// </summary>
/// <param name="strSql">Tên thủ tục hay câu lệnh truy vấn
SQL</param>
/// <param name="ct">CommandType</param>
/// <param name="err">Tham số tham chiếu lưu trữ lỗi</param>
/// <param name="param">Danh sách các tham số truyền vào cho thủ
tục sql</param>

```

```

    /// <returns>Trả về True nếu thực hiện thành công, người lại trả
    /// về False</returns>
    public bool MyExecuteNonQuery(string strSql, CommandType ct, ref
string err, params SqlParameter[] param)
    {
        bool f = false;
        cnn.Open();
        cmd.Parameters.Clear();
        cmd.CommandText = strSql;
        cmd.CommandType = ct;
        foreach (SqlParameter p in param)
            cmd.Parameters.Add(p);
        try
        {
            cmd.ExecuteNonQuery();
            f = true;
        }
        catch (SqlException ex)
        {
            err = ex.Message;
        }
        finally
        {
            cnn.Close();
        }
        return f;
    }
    /// <summary>
    /// Thực thi NonQuery sử dụng cho insert, update, Delete không
    có biến tham chiếu lỗi
    /// </summary>
    /// <param name="strSql">Tên Thủ tục hay câu lệnh truy vấn
    SQL</param>
    /// <param name="ct">CommandType</param>
    /// <param name="param">Danh sách các tham số truyền vào cho thủ
    tục sql</param>
    /// <returns></returns>
    public bool MyExecuteNonQuery(string strSql, CommandType ct,
params SqlParameter[] param)
    {
        bool f = false;
        cnn.Open();
        cmd.Parameters.Clear();
        cmd.CommandText = strSql;
        cmd.CommandType = ct;
        foreach (SqlParameter p in param)
            cmd.Parameters.Add(p);
        try
        {
            cmd.ExecuteNonQuery();
            f = true;
        }
        catch (SqlException ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}

```

```

        }
        finally
        {
            cnn.Close();
        }
        return f;
    }
    /// <summary>
    /// Phương thức sử dụng để lấy một giá trị trả về từ cơ sở dữ
liệu
    /// </summary>
    /// <param name="strsql">Tên thủ tục hay câu truy vấn
Sql</param>
    /// <param name="ct">CommandType</param>
    /// <param name="err">tham số tham chiếu lấy lỗi</param>
    /// <returns>Trả về kiểu object</returns>
    public object Laygiatri(string strsql, CommandType ct, ref
string err)
{
    cnn.Open();
    cmd.CommandText = strsql;
    cmd.CommandType = ct;
    object obj;
    try
    {
        obj = cmd.ExecuteScalar();
    }
    catch (Exception ex)
    {
        err = ex.Message;
        throw;
    }
    finally
    {
        cnn.Close();
    }

    return obj;
}
    /// <summary>
    /// Phương thức sử dụng để lấy một giá trị trả về từ cơ sở dữ
liệu
    /// </summary>
    /// <param name="strsql">Tên thủ tục hay câu truy vấn
Sql</param>
    /// <param name="ct">CommandType</param>
    /// <param name="err">tham số tham chiếu lấy lỗi</param>
    /// <param name="param">Danh sách tham số truyền vào cho thủ tục
sql</param>
    /// <returns>Trả về kiểu object</returns>
    public object Laygiatri(string strsql, CommandType ct, ref
string err, params SqlParameter[] param)
{
    cnn.Open();
    cmd.CommandText = strsql;
}

```

```

        cmd.CommandType = ct;
        foreach (SqlParameter p in param)
        {
            cmd.Parameters.Add(p);
        }
        object obj;
        try
        {
            obj = cmd.ExecuteScalar();
        }
        catch (Exception ex)
        {
            err = ex.Message;
            throw;
        }
        finally
        {
            cnn.Close();
        }
        return obj;
    }

    public SqlDataReader ThucHienReader(string sql)
    {

        cnn.Open();
        SqlCommand mysqlcommand = cnn.CreateCommand();

        mysqlcommand.CommandText = sql;
        SqlDataReader datareader = mysqlcommand.ExecuteReader();

        return datareader;
    }

    /// <summary>
    /// Phương thức thực thi Reader
    /// </summary>
    /// <param name="sql">Thủ tục sql hay câu truy vấn sql</param>
    /// <param name="ct">CommandType</param>
    /// <param name="err">Tham số tham chiếu chưa lỗi</param>
    /// <param name="param">danh sách tham số sử dụng cho thủ
    tucus</param>
    /// <returns></returns>
    public SqlDataReader ThucHienReader(string sql, CommandType
ct, ref string err, params SqlParameter[] param)
    {
        cnn.Open();
        SqlDataReader datareader;
        SqlCommand mysqlcommand = cnn.CreateCommand();
        mysqlcommand.CommandText = sql;
        mysqlcommand.CommandType = ct;
        mysqlcommand.Parameters.Clear();
        foreach (SqlParameter p in param)
        {
            mysqlcommand.Parameters.Add(p);
        }
    }
}

```

```

        }
    try
    {
        datareader = mysqlcommand.ExecuteReader();
    }
    catch (Exception ex)
    {
        err = ex.Message;
        throw;
    }
    finally
    {
        //cnn.Close();
    }

    return datareader;
}
}
}

```

3.3.2. Đối tượng DataReader

Dùng để đón nhận kết quả trả về từ phương thức ExecuteReader của đối tượng Command, chỉ đọc (Readonly) nhanh dữ liệu theo chiều tiến.

Khai báo DataReader:

```

SqlDataReader <Biến DataReader>
Lấy kết quả từ Command

<Biến DataReader> = <Biến
Command>.ExecuteReader()

```

Sau khi lấy kết quả về, để đọc dữ liệu ta dùng phương thức Read với cú pháp:

```
<Biến DataReader>.Read()
```

Nếu cần duyệt qua danh sách mẫu tin trong đối tượng SqlDataReader thì ta sử dụng phát biểu While với phương thức Read() như sau:

```

while( <Biến DataReader>.read())
{
    //“Xử lý mẫu tin đọc được”;
    MessageBox.Show(<Biến DataReader>.GetValue
(FieldIndex))
    //FieldIndex là số thứ tự của cột dữ liệu.
    .....
}

```

Chú ý: SqlDataReader không cung cấp thuộc tính hay phương thức cho phép chúng ta kiểm soát được số mẫu tin đang nắm giữ, thay vào đó phương thức Read() trả về True nếu mẫu tin tiếp theo là mẫu tin hợp lệ.

Trong trường hợp chỉ cần kiểm tra sự tồn tại của mẫu tin đầu tiên thì có thể sử dụng phát biểu If thay cho phát biểu While như sau:

```
if( <Biến DataReader>.Read())
{
    //Xử lý mẫu tin đầu tiên đọc được
}
```

Các thuộc tính của DataReader:

- ✓ FieldCount: Trả về số cột trên dòng hiện hành của DataReader.
- ✓ IsClosed: Cho biết DataReader đã đóng chưa.
- ✓ Item: Trị của cột truyền vào, tham số truyền có thể là tên cột hoặc số thứ tự.
- ✓ HasRows: cho biết có dữ liệu trả về hay không, tuy nhiên HasRows không đọc bất kỳ mẫu tin nào cho chúng ta xử lý cả.
- ✓ ...

Các phương thức của DataReader:

- ✓ Close: Đóng DataReader.
- ✓ GetDataTypeName: Trả về kiểu dữ liệu của cột truyền vào.
- ✓ GetName: Trả về tên của cột truyền vào.
- ✓ GetOrdinal: Trả về số thứ tự của cột truyền vào.
- ✓ GetValue: Trả về giá trị trên cột truyền vào.
- ✓ Read: Di chuyển đến dòng kế tiếp và trả về True nếu còn dòng để di chuyển, ngược lại trả về False.
- ✓ ...

Ví dụ:



Hình 2.4: Ví dụ đọc dữ liệu với SqlDataReader

Ban đầu ComboBox rỗng, khi ta nhấn Button “Đọc thông tin” đối tượng DataReader sẽ lấy dữ liệu từ phương thức ExecuteReader của đối tượng Command và Fill vào ComboBox thông tin gồm: Tên cột, số thứ tự của cột và kiểu dữ liệu của cột trên bảng SanPham trong Database HoaDon.

↳ Code của Button “Đọc thông tin” như sau:

```
SqlConnection conn = new SqlConnection(); conn.ConnectionString =
ConfigurationManager.AppSettings("strconn") ;
conn.Open() ;
SqlCommand cmd = new SqlCommand();
cmd.Connection = conn ;
cmd.CommandText = "Select * From SanPham" ;
cmd.CommandType = CommandType.Text ;
SqlDataReader rd = cmd.ExecuteReader();
if( rd.HasRows)
{
    for (int i = 0; i < rd.FieldCount - 1; i++)
    {
        cboThongTin.Items.Add(rd.GetName(i) + " - " +
rd.GetOrdinal(rd.GetName(i)) + " - " + rd.GetDataTypeName(i));
    }
}
```

Câu hỏi hiểu bài

- Giả sử ta muốn lấy dữ liệu từ CSDL về và dùng đối tượng DataReader để đọc dữ liệu đó, thứ tự sau đây hợp lý hay không? Nếu không, phải thay đổi như thế nào mới hợp lý.
 - Cần khai báo một biến đối tượng DataConnection, và mở kết nối đến cơ sở dữ liệu.
 - Khi cần lấy dữ liệu thì tạo một biến DataCommand gắn với biến DataConnection đã mở, với câu lệnh Select truy vấn dữ liệu cần lấy về.
 - Khai báo biến DataReader
 - Thi hành phương thức ExecuteReader() của biến đối tượng DataCommand và gán cho biến đối tượng DataReader.
 - Đọc và trình bày dữ liệu
 - Đóng và giải phóng các biến đối tượng.

2. Đối tượng DataReader có thể đọc dữ liệu một cách ngẫu nhiên không?
3. DataReader không cung cấp thuộc tính hay phương thức cho phép chúng ta kiểm soát được số mẫu tin đang nắm giữ, đúng hay sai? Nếu sai, chỉ ra thuộc tính đó.
4. Thuộc tính HasRows của DataReader dùng để đọc mẫu tin cho chúng ta xử lý, đúng hay sai? Nếu sai thì chức năng của nó là gì?
5. Thuộc tính GetName của DataReader có chức năng gì?
6. Thuộc tính GetOrdinal của DataReader có chức năng gì?

Bài tập tại lớp

1. Đọc dữ liệu với DataReader lấy thông tin bao gồm: tên cột số thứ tự của cột và kiểu dữ liệu của các cột trong bảng sản phẩm và hiển thị vào một combobox. Câu truy vấn viết thành thủ tục store procedure
2. Đọc dữ liệu với DataReader lấy thông tin bao gồm: tên cột số thứ tự của cột, kiểu dữ liệu và giá trị của các cột trong bảng nhân viên và hiển thị vào một combobox. Câu truy vấn viết thành thủ tục store procedure

Bài tập về nhà

1. Đọc dữ liệu với DataReader lấy thông tin bao gồm: tên cột, số thứ tự của cột, kiểu dữ liệu và giá trị của các cột trong bảng khách hàng và hiển thị vào một combobox. Câu truy vấn viết thành thủ tục store procedure
2. Đọc dữ liệu với DataReader lấy thông tin bao gồm: tên cột, số thứ tự của cột, kiểu dữ liệu và giá trị của các cột trong bảng khách hàng và hiển thị vào một combobox. Câu truy vấn viết thành thủ tục store procedure
3. Đọc dữ liệu với DataReader lấy thông tin bảng sản phẩm và hiển thị vào một listview. Câu truy vấn viết thành thủ tục store procedure

3.3.3. Đối tượng DataAdapter

Ý nghĩa của DataAdapter như tên gọi của nó, “Adapter” có nghĩa là điều phối còn “Data” là dữ liệu. Bộ điều phối này sẽ giúp chúng ta lưu chuyển dữ liệu, xử lý logic một cách hiệu quả giữa trình ứng dụng và CSDL. Bản chất nó không lưu trữ dữ liệu, nó chỉ phát đi các lệnh SQL và nhận dữ liệu hay kết quả trả về từ CSDL. DataAdapter cung cấp cho chúng ta cùng lúc bốn đối tượng Command: Select, Insert, Update và Delete để thực hiện đồng loạt các lệnh SQL thành một đơn vị xử lý duy nhất. DataAdapter điều dữ liệu nguồn vào một DataSet hay DataTable sử dụng phương thức **Fill()**. Còn khi cập nhật dữ liệu ngược trở lại nguồn từ ứng dụng thì sử dụng phương thức **Update()**.

Đối tượng SqlDataAdapter nằm trong namespace System.Data.SqlClient do đó để sử dụng chúng ta phải thêm namespace này vào trước khi sử dụng.

```
using System.Data.SqlClient;
```

Cú pháp khai báo:

`SqlDataAdapter <tên biến>`

Ví dụ: `SqlDataAdapter da;`

Khởi tạo:

`<tên biến> = new SqlDataAdapter ([[lệnh],[biến Connection]])`

Ví dụ : `da = new SqlDataAdapter("Select * From KhachHang,cnn);`

Trong đó:

- ✓ [lệnh]: Câu lệnh truy vấn Select, tên Store Procedure,... để thực hiện truy vấn từ nguồn dữ liệu.
- ✓ [biến Connection]: Đối tượng Connection đã kết nối với CSDL.

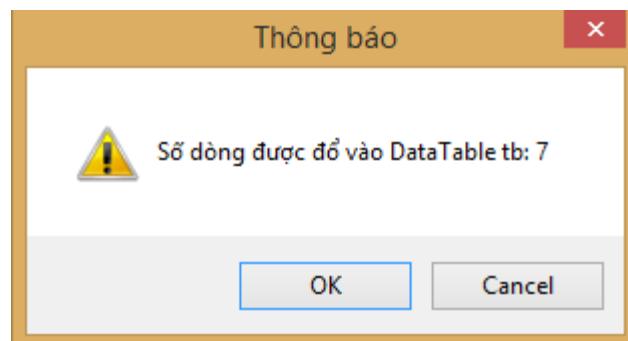
3.3.3.1. Phương thức Fill – Trích rút dữ liệu.

- ✓ **Fill(<DataTable>):** Đỗ dữ liệu vào DataTable có sẵn.
- ✓ **Fill(<DataSet>):** Đỗ dữ liệu vào DataSet có sẵn. Dữ liệu được lấy về DataSet dưới dạng các DataTable với tên mặc định là Table, Table1, Table2,...
- ✓ **Fill(<DataSet>,<Tên DataTable>):** Đỗ dữ liệu vào bảng <Tên DataTable> trong DataSet. Nếu chưa có bảng sẽ được tạo.

Ví dụ Sau đây SqlDataAdapter sẽ đọc dữ liệu của bảng NhanVien. Nó sử dụng thuộc tính SelectCommand để thực thi câu lệnh Select của SQL gián tiếp thông qua phương thức Fill() của SqlDataAdapter. Dữ liệu sau đó được đỗ vào DataTable.

```
private void Form1_Load(object sender, EventArgs e)
{
    SqlConnection cnn=new SqlConnection();
    cnn.ConnectionString=ConfigurationSettings.AppSettings("strconn");
    cnn.Open();
    SqlCommand cmd=new SqlCommand("Select * From NhanVien",cnn);
    //Khai báo định nghĩa đối tượng DataAdapter
    SqlDataAdapter da=new SqlDataAdapter(cmd);
    DataTable dt=new DataTable();
    da.Fill(dt);
    MessageBox.Show("Số dòng được đổ vào DataTable tb:
"+dt.Rows.Count);
}
```

Kết quả ta có được:



Hình 3.5: Kết quả trích rút dữ liệu

Khác với đối tượng DataReader chỉ được phép đọc một chiều, đối tượng DataTable cho phép chúng ta định vị mọi mảng tin trong tập dữ liệu thông qua chỉ số. Chúng ta có thể xem DataTable như một bảng tính Excel bao gồm các dòng và các cột và các dòng và cột đó có thể tham chiếu thông qua chỉ số mảng.

Ví dụ: Để xem dòng 7, cột 3 có giá trị là gì chúng ta có thể truy xuất như sau:

```
MessageBox.Show("Dữ liệu dòng 7 cột 3: " + dt.Rows[7][3].ToString());
```

Lưu ý: Chỉ số dòng và cột được đánh từ 0.

Chúng ta có thể dùng đối tượng DataGridView của Windows Forms để hiển thị toàn bộ dữ liệu của biến đối tượng DataTable tb:

Ví dụ:

```
.....
DataTable dt = new DataTable("NhanVien")
'Thực hiện lệnh Select và đổ dữ liệu vào
bảng tb          da.Fill(tb)
'Gán nguồn dữ liệu cho lưới là DataTable tb
DataGridView1.DataSource = tb
```

Và kết quả ta được:

	MaNV	Ho	Ten	Nu
▶	55	Trần Kiều	Van	
	57	Nguyễn Thị Linh	Giang	
	58	Nguyễn Hùng	Anh	
	59	Trịnh Nguyễn...	Linh	
	60	Trần Thị Mỹ	Hậu	
	61	Ngô Bảo	Châu	
	

Hình 3.6: Hiển thị dữ liệu rút trích lên lưới

3.3.3.2. Phương thức Update – Một hàm làm mới thứ.

Để cập nhật dữ liệu chúng ta tạo câu lệnh UpdateCommand của SqlDataAdapter như sau:

Ví dụ:

```
SqlConnection cnn=new SqlConnection();
cnn.ConnectionString=ConfigurationSettings.AppSettings
("strconn");
cnn.Open();

SqlDataAdapter da=new SqlDataAdapter(cmd);
da.UpdateCommand = New SqlCommand();
da.UpdateCommand.Connection = conn;
da.UpdateCommand.CommandType = CommandType.Text
da.UpdateCommand.CommandText = "Update SanPham
Set
SoTon = @soton Where MaSP = @masp"
.....
```

Để xóa một dòng dữ liệu thì sử dụng lệnh SQL Delete và gán nó cho thuộc tính DeleteCommand của SqlDataAdapter như sau:

Ví dụ:

```
SqlDataAdapter da=new SqlDataAdapter(cmd);
da.DeleteCommand = New SqlCommand();
da.DeleteCommand.Connection = conn;
da.DeleteCommand.CommandType = CommandType.Text;
da.DeleteCommand.CommandText = "Delete From SanPham
Where MaSP = @masp";
```

Tương tự như Update và Delete, để Insert dữ liệu thì sử dụng lệnh SQL Insert và gán cho nó thuộc tính InsertCommand của SqlDataAdapter.

Ví dụ:

```
SqlDataAdapter da=new SqlDataAdapter(cmd);
da.InsertCommand = New SqlCommand();
da.InsertCommand.Connection = conn;
da.InsertCommand.CommandType = CommandType.Text;
da.InsertCommand.CommandText = "Insert Into
SanPham(MaSP, TenSP) Values (@masp,@tensp)" ;
```

Tại sao nói một hàm làm mọi thứ? Mọi thứ đều riêng biệt đó chứ: Insert, Delete, Update và cả Select.

Tuy nhiên, SqlDataAdapter với phương thức Fill() dùng thực thi lệnh SelectCommand và đổ dữ liệu vào đối tượng DataTable. Các Command còn lại có thể tự động phát sinh lệnh cập nhật thông qua phương thức Update() của SqlDataAdapter, chúng ta phải sử dụng *CommandBuilder* thuộc không gian tên System. Data. SqlCommandBuilder.

Cú pháp khởi tạo các đối tượng Insert, Update và Delete.

```
SqlCommandBuilder <tên biến> =new SqlCommandBuilder(<Đối
tượng SqlDataAdapter> )
```

Ví dụ: Sau đây thông qua phương thức Fill() của SqlDataAdapter sẽ đổ dữ liệu vào DataTable. Lưới DataGridView sẽ lấy nguồn dữ liệu từ DataTable và hiển thị. Chúng ta có thể thêm, sửa, xóa trực tiếp trên lưới và cập nhật vào CSDL thông qua duy nhất phương thức Update() của SqlDataAdapter như sau:



Hình 2.7: Update – một hàm làm mới thứ

```
//Khai báo các biến toàn cục  
SqlConnection cnn;  
SqlDataAdapter da;  
DataTable dt;  
SqlCommandBuilder cmb;
```

Code cho sự kiện Load của Form.

```
cnn = new SqlConnection();
cnn.ConnectionString =
ConfigurationSettings.AppSettings("strconn");
cnn.Open() ;
da = new SqlDataAdapter();
da.SelectCommand = new SqlCommand();
da.SelectCommand.Connection = cnn ;
da.SelectCommand.CommandType = CommandType.Text;
da.SelectCommand.CommandText = "Select * From SanPham";
dt = new DataTable("SanPham");
da.Fill(dt) ;
```

```
DataGridView1.DataSource = dt ;
```

Code cho sự kiện Click của Button "Update"

```
//Khai báo DataTable để chứa các thay đổi trên lưới  
DataTable tbl =new DataTable();  
tbl = dt.GetChanges();  
    //Nếu có sự thay đổi sẽ phát sinh các lệnh cập nhật  
if(tbl.Rows.Count>0)  
{  
    cmb = new SqlCommandBuilder(da);  
    da.Update(dt);  
}
```

Một vài lưu ý:

- ✓ CommandBuilder chỉ phát sinh nội dung lệnh cập nhật cho các DataAdapter có nội dung SelectCommand truy xuất đến một bảng nguồn.
- ✓ Nội dung trong SelectCommand phải có ít nhất một khóa chính hoặc một khóa duy nhất (Unique Key) để DataAdapter phân biệt các dòng khi cập nhật, nếu không CommandBuilder sẽ không phát sinh được nội dung lệnh cho các Command Insert, Update và Delete.
- ✓ Trường hợp nội dung của SelectCommand là truy vấn từ hơn một bảng hoặc từ một Store Procedure các Command cập nhật không tự động phát sinh, chúng ta phải khai báo các *Command cập nhật cách tường minh*.
- ✓ Trường hợp nội dung của SelectCommand là nhiều câu lệnh Select SQL, CommandBuilder chỉ tạo được lệnh cập nhật cho lệnh truy vấn đầu tiên.

Để cập nhật dữ liệu về nguồn:

- ✓ <biến DataAdapter>.Update(<DataTable>): Cập nhật các thay đổi trên DataTable vào nguồn dữ liệu.
- ✓ < biến DataAdapter >.Update(<DataSet>.< DataTable()>): Cập nhật các thay đổi trên DataTable chỉ định trong DataSet vào nguồn dữ liệu.
- ✓ < biến DataAdapter>.Update(<DataSet>): Cập nhật các thay đổi trên tất cả các bảng của DataSet vào nguồn dữ liệu.

Câu hỏi hiểu bài

1. Các thao tác trên cơ sở dữ liệu có thể là thêm mới, sửa, xóa ta sẽ có một trình tự như sau? Thứ tự đã hợp lý chưa và nếu chưa hãy giúp chúng hợp lý.

- ✓ Cần khai báo một biến đối tượng DataConnection, và mở kết nối đến cơ sở dữ liệu.
- ✓ Khi cần lấy dữ liệu thì tạo một biến DataCommand gắn với biến DataConnection đã mở.
- ✓ Gán giá trị cho thuộc tính CommandType của biến đối tượng DataCommand - Gán câu lệnh truy vấn (insert, update, delete, gọi store procedure...) cho thuộc tính CommandText của biến đối tượng DataCommand

Thi hành phương thức ExecuteNonQuery() của biến đối tượng DataCommand - Đóng và giải phóng các biến đối tượng.

2. SqlDataAdapter có namespace là gì?
3. Nêu đặc điểm của đối tượng SqlDataAdapter.

Bài tập tại lớp

1. Viết chương trình hiển thị thông tin nhân viên lên lưới DataGridView, giao diện cho phép nhập thông tin nhân viên từ các textbox để thêm mới nhân viên, khi double click vào nhân viên nào trên lưới sẽ hiển thị thông tin tương ứng của nhân viên đó lên các textbox để chỉnh sửa thông tin nhân viên. Ngoài ra chương trình cho phép xóa nhân viên với mã nhân viên được nhập vào từ bàn phím.
2. Viết chương trình với yêu cầu như trên (câu 1), thao tác thêm sửa xóa thực hiện trực tiếp trên lưới nhưng sử dụng phương thức Update của DataAdapter để thấy được chức năng một hàm làm được cùng lúc thêm, sửa, xóa.

Bài tập về nhà

1. Viết chương trình hiển thị thông tin sản phẩm lên lưới DataGridView, giao diện cho phép nhập thông tin sản phẩm từ các textbox để thêm mới, khi double click vào sản phẩm nào trên lưới sẽ hiển thị thông tin tương ứng của sản phẩm đó lên các textbox để chỉnh sửa thông tin. Ngoài ra chương trình cho phép xóa sản phẩm với mã sản phẩm được nhập vào từ bàn phím.
2. Viết chương trình với yêu cầu như trên (câu 1), thao tác thêm sửa xóa thực hiện trực tiếp trên lưới, sử dụng phương thức Update của DataAdapter để thấy được chức năng một hàm làm được cùng lúc thêm, sửa, xóa.

3.3.4. Đối tượng SqlParameter

Trong phần trước bạn đã tìm hiểu đối tượng chính là SqlCommand dùng để thao tác sơ sở dữ liệu SQL Server. Trong phần này chúng ta tiếp tục tìm hiểu chi tiết về thuộc tính Parameters (thuộc tính Parameter trả về đối tượng SqlParameterCollection) của đối tượng SqlCommand ứng với đối tượng SqlConnection dùng để thực thi phát biểu SQL hay thủ tục nội tại.

Nếu khi sử dụng đối tượng SqlCommand với câu lệnh dạng chuỗi truyền vào tồn tại một dấu nháy đơn thì bạn cần sử dụng phương thức Replace để thay thế chúng thành hai dấu nháy đơn liên tiếp. mục đích của việc thay thế chúng là tạo ra một phát biểu SQL với

dữ liệu truyền từ bên ngoài vào SQL có cú pháp hợp lệ. và việc này để dàng tạo cơ hội cho những hacker tấn công hệ thống bằng kỹ thuật SQL Injection.

Để khắc phục việc này ta sử dụng đối tượng SQLParameters hay SqlParameterCollection của đối tượng SqlCommand.

Khai báo và khởi tạo

Để sử dụng đối tượng SqlParameter, bạn cũng thực hiện các bước như đối tượng SqlCommand là chúng cũng phải được khai báo, khởi tạo, định nghĩa phương thức hay thuộc tính.

```
SqlParameter _sqlparameter;
_sqlparameter=new SqlParameter();
```

Trong trường hợp bạn muốn khởi tạo đối tượng SqlParameter với hai tham số là tên tham số và giá trị tương ứng thì sử dụng cú pháp như sau:

```
SqlParameter _sqlparameter;
_sqlparameter=new SqlParameter(string ParameterName,object value);
```

Hay

```
SqlParameter _sqlparameter;
_sqlparameter=new SqlParameter(string
parameterName,SqlDbType DbType);
```

Hay sử dụng 3 tham số

```
SqlParameter _sqlparameter;
_sqlparameter=new SqlParameter(string
parameterName,SqlDbType DbType,int Size);
```

Hay sử dụng 4 tham số:

```
SqlParameter _sqlparameter;
_sqlparameter=new SqlParameter(string
parameterName,SqlDbType DbType,int Size,string
sourceColumn);
```

Các thuộc tính của SqlParameter

- ParameterName:
- SqlValue: giá trị truyền vào.
- SqlDbType: kiểu dữ liệu
- Size: kích thước

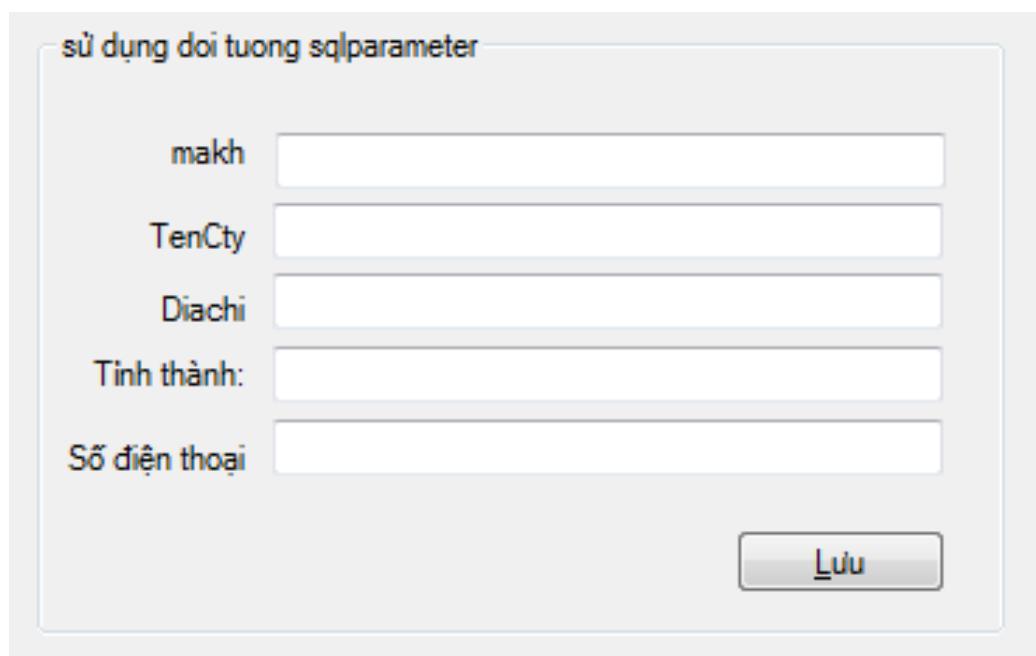
Sử dụng:

```

SqlParameter sqlparam;
sqlparam = new SqlParameter();
sqlparam.ParameterName = "@makh";
sqlparam.SqlValue = "FPT";
sqlparam.SqlDbType = SqlDbType.NVarChar;
sqlparam.Size = 50;
cmm.Parameters.Add(sqlparam);

```

Ví dụ và cách sử dụng SqlParameter



Code cho nút lưu

```

private void btnluusqlparam_Click(object sender, EventArgs e)
{
    cnn = new SqlConnection();
    cnn.ConnectionString = constring.GetConnectionString("SqlServer");
    try
    {
        cnn.Open();
        //buoc 1
        string sqlstring = "sp_Themkhachang";
        cmm = new SqlCommand(sqlstring, cnn);
        cmm.CommandType = CommandType.StoredProcedure;
        //buoc 2, và 3
        //thuoc tinh dau tien
        SqlParameter sqlparameter = new SqlParameter();
        sqlparameter.ParameterName = "@makh";
        sqlparameter.SqlDbType = SqlDbType.NVarChar;
        sqlparameter.Size = 10;
        sqlparameter.SqlValue = txtmakh.Text;
        cmm.Parameters.Add(sqlparameter);
    }
}

```

```
//thuoc tinh 2
sqlparameter = new SqlParameter("@tencty", SqlDbType.NVarChar, 40);
sqlparameter.SqlValue = txtten.Text;
cmm.Parameters.Add(sqlparameter);
//thuoc tinh 3
sqlparameter = new SqlParameter("@diachi", txtdiachi.Text);
sqlparameter.SqlDbType = SqlDbType.NVarChar;
sqlparameter.Size = 40;
cmm.Parameters.Add(sqlparameter);
sqlparameter = new SqlParameter("@thanhpho", SqlDbType.NVarChar, 15);
sqlparameter.SqlValue = txttinhthanh.Text;
cmm.Parameters.Add(sqlparameter);
sqlparameter = new SqlParameter("@dienthoai", SqlDbType.NVarChar, 10);
sqlparameter.SqlValue = txtsodienthoai.Text;
cmm.Parameters.Add(sqlparameter);
int records = cmm.ExecuteNonQuery();
MessageBox.Show("da them thanh cong " + records + " khach hang");
cnn.Close();
cnn.Dispose();
}
catch (Exception ex)
{
```

CHƯƠNG 4. CÁC THÀNH PHẦN PHÍA ỦNG DỤNG

Mục đích - Yêu cầu

Sau khi học xong bài này sinh viên nắm được:

- Các đối tượng nắm giữ dữ liệu như DataSet, DataTable, DataRow, DataColumn, BindingSource,...
- Chi tiết cách tạo cũng như nắm được các thuộc tính và phương thức của các đối tượng trên.
- Vận dụng các đối tượng hiển thị, xử lý dữ liệu như DataView, DataGridView để có thể: lọc, sắp xếp, trình bày dữ liệu....
- Cách thêm các Control như TextBox, ComboBox, CheckBox, Button vào khung lưới DataGridView.
- Cách thức ràng buộc dữ liệu với các đối tượng Windows Form.
- Kết hợp giữa các đối tượng nắm giữ dữ liệu và trình diễn dữ liệu, quản lý dữ liệu hiệu quả.

Trọng tâm bài giảng

- Tìm hiểu chi tiết các đối tượng nắm giữ dữ liệu.
- Các phương thức trình diễn dữ liệu thông qua các đối tượng trình diễn dữ liệu, ràng buộc dữ liệu với các Windows Form.
- Kết hợp giữa các đối tượng xử lý và trình diễn dữ liệu, quản lý dữ liệu một cách hiệu quả nhất.

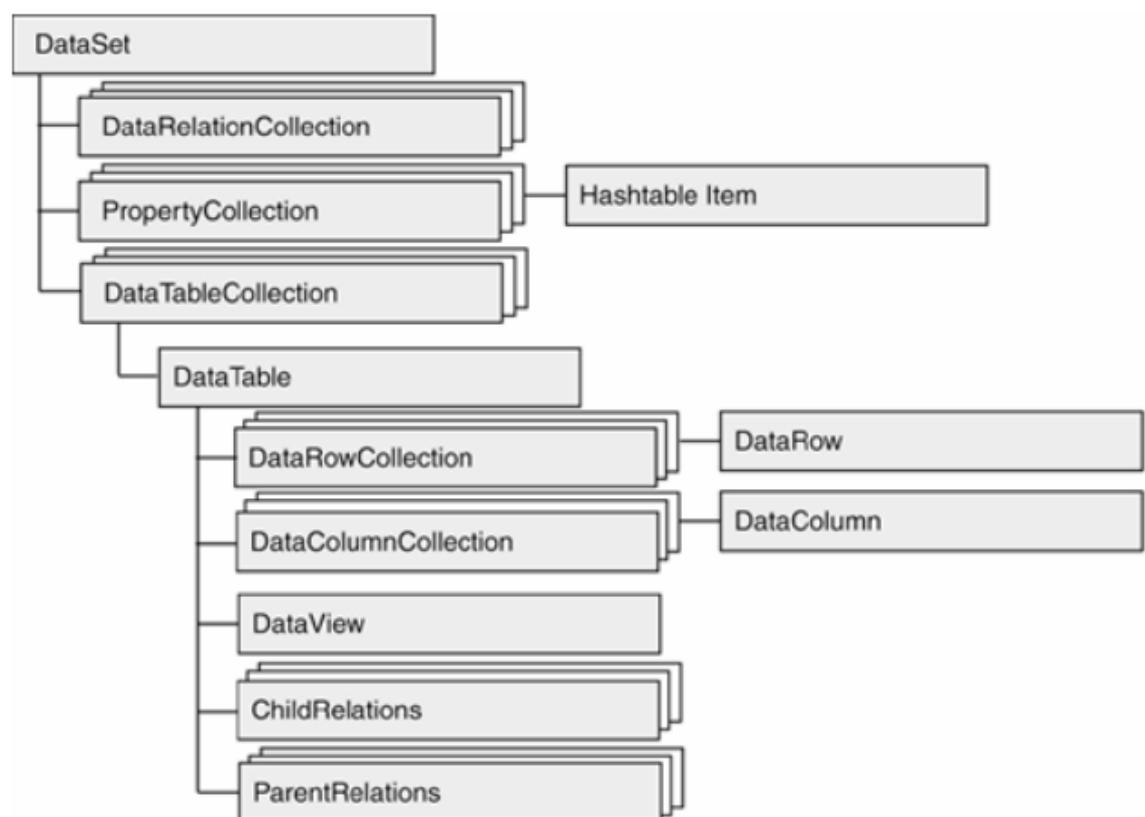
4.1.GIỚI THIỆU

Chương 3 chúng ta đã học cách trích rút dữ liệu từ hệ quản trị CSDL bằng các đối tượng như SqlConnection, SqlCommand, SqlDataAdapter. Chương này chúng ta sẽ học về các đối tượng quan trọng giúp lập trình phía ứng dụng và các đối tượng trình diễn dữ liệu. Các đối tượng này hoàn toàn độc lập với CSDL và không phụ thuộc vào .NET Data Provider. Chúng là các công cụ hữu hiệu, mạnh mẽ có thể giúp chúng ta giải quyết hầu như mọi vấn đề. Các đối tượng đó bao gồm: DataSet, DataTable, DataRow, DataColumn, DataView, DataGridView, BindingSource, ... Trong đó đối tượng DataSet đã được chúng ta tìm hiểu trong chương các thành phần của ADO.NET (chương 3), chương này chúng ta tiếp tục với các đối tượng còn lại.

DataSet đóng vai trò của một CSDL in-memory (CSDL nằm trong bộ nhớ). Thuộc tính Tables của DataSet là một tập hợp các DataTable chứa dữ liệu và lược đồ dữ liệu (data schema) mô tả dữ liệu trong DataTable. Thuộc tính Relations chứa tập hợp các đối tượng

DataRelation xác định cách thức liên kết các đối tượng DataTable của DataSet. Lớp DataSet cũng hỗ trợ việc sao chép, trộn, và xóa DataSet thông qua các phương thức tương ứng là Copy, Merge, và Clear. DataSet và DataTable là phần lõi của ADO.NET và chúng không là đặc trưng của một data provider nào (giống như ở các lớp Connection, DataReader, DataAdapter). Một ứng dụng có thể định nghĩa và nạp dữ liệu từ nguồn bất kỳ (chứ không nhất thiết là từ một CSDL) vào DataSet.

Bên cạnh các DataTable và các DataRelation, một DataSet còn có thể chứa các thông tin tùy biến khác được định nghĩa bởi ứng dụng. Hình dưới đây mô tả cả lớp chính trong DataSet. Trong số các thuộc tính này, chú ý thuộc tính PropertyCollection; đó là các thuộc tính được lưu trữ dưới dạng một hash table (bảng băm), thường chứa một giá trị time stamp hay các thông tin đặc tả như các yêu cầu hợp lệ hóa (validation requirements) cho column trong các DataTable trong DataSet.



Hình 3.8: cấu trúc của DataSet

4.2. DATATABLE

Thuộc tính DataSet.Tables chứa các đối tượng DataTable. Mỗi đối tượng trong tập hợp này có thể được truy xuất bằng chỉ số hoặc bằng tên. Các DataTable trong tập hợp DataSet.Tables mô phỏng các Table trong CSDL quan hệ (các row, column, ...). Các

thuộc tính quan trọng nhất của lớp DataTable là Columns và Rows định nghĩa cấu trúc và nội dung bảng dữ liệu.

4.2.1. DataColumn

Thuộc tính DataTable.Columns chứa một tập các đối tượng DataColumn biểu diễn các trường dữ liệu trong DataTable. Bảng dưới đây tóm tắt các thuộc tính quan trọng của lớp DataColumn.

Phương thức	Mô tả
ColumnName	Tên column
DataType	Kiểu của dữ liệu chứa trong column này Ví dụ: col1.DataType = System.Type.GetType("System.String")
MaxLength	Độ dài tối đa của một text column. -1 nếu không xác định độ dài tối đa
ReadOnly	Cho biết giá trị của column có được chỉnh sửa hay không
AllowDBNull	Giá trị Boolean cho biết column này có được chứa giá trị NULL hay không
Unique	Giá trị Boolean cho biết column này có được chứa các giá trị trùng nhau hay không
Expression	Biểu thức định nghĩa cách tính giá trị của một column Ví dụ: colTax.Expression = "colSales * .085";
Caption	Tiêu đề hiển thị trong thành phần điều khiển giao diện đồ họa
DataTable	Tên của đối tượng DataTable chứa column này

Các column của DataTable được tạo ra một cách tự động khi table được nạp dữ liệu từ kết quả của một database query hoặc từ kết quả đọc được ở một file XML. Tuy nhiên, chúng ta cũng có thể viết code để tạo động các column. Đoạn code dưới đây sẽ tạo ra một đối tượng DataTable, sau đó tạo thêm các đối tượng DataColumn, gán giá trị cho các thuộc tính của column, và bổ sung các DataColumn này vào DataTable.

```
DataTable tb = new DataTable("DonHang");
DataColumn dCol = new DataColumn("MaSo", Type.GetType("System.Int16"));
dCol.Unique = true; // Dữ liệu của các dòng ở column này không được trùng nhau
dCol.AllowDBNull = false;
tb.Columns.Add(dCol);
dCol = new DataColumn("DonGia", Type.GetType("System.Decimal"));
tb.Columns.Add(dCol);
dCol = new DataColumn("SoLuong", Type.GetType("System.Int16"));
tb.Columns.Add(dCol);
dCol = new DataColumn("ThanhTien", Type.GetType("System.Decimal"));
dCol.Expression = "SoLuong*DonGia";
tb.Columns.Add(dCol);
// Liệt kê danh sách các Column trong DataTable
foreach (DataColumn dc in tb.Columns)
{
    Console.WriteLine(dc.ColumnName);
    Console.WriteLine(dc.DataType.ToString());
```

Để ý rằng column MaSo được định nghĩa để chứa các giá trị duy nhất. Ràng buộc này giúp cho column này có thể được dùng như là trường khóa để thiết lập relationship kiểu parent-child với một bảng khác trong DataSet. Để mô tả, khóa phải là duy nhất – như trong trường hợp này – hoặc được định nghĩa như là một primary key của bảng. Ví dụ dưới đây mô tả cách xác định primary key của bảng:

```
DataTable[] col = {tb.Columns["MaSo"]};
tb.PrimaryKey = col;
```

Nếu một primary key chứa nhiều hơn 1 column – chẳng hạn như HoDem và Ten – bạn có thể tạo ra một ràng buộc unique constraint trên các như ví dụ dưới đây:

```
DataTable[] cols = {tb.Columns["HoDem"], tb.Columns["Ten"]};
tb.Constraints.Add(new UniqueConstraint("keyHoVaTen", cols));
```

4.2.2. DataRow

Dữ liệu được đưa vào table bằng cách tạo mới một đối tượng DataRow, gán giá trị cho các column của nó, sau đó bổ sung đối tượng DataRow này vào tập hợp Rows gồm các DataRow của table.

```
DataRow row;
row = tb.NewRow(); // Tạo mới DataRow
row["DonGia"] = 22.95;
row["SoLuong"] = 2;
row["MaSo"] = 12001;
tb.Rows.Add(row); // Bổ sung row vào tập Rows
Console.WriteLine(tb.Rows[0]["ThanhTien"].ToString()); // 45.90
```

Một DataTable có các phương thức cho phép nó có thể commit hay roll back các thay đổi được tạo ra đối với table tương ứng. Để thực hiện được điều này, nó phải nắm giữ trạng thái của mỗi dòng dữ liệu bằng thuộc tính DataRow.RowState. Thuộc tính này được thiết lập bằng một trong 5 giá trị kiểu enumeration DataRowState sau: Added, Deleted, Detached, Modified, hoặc Unchanged. Xem xét ví dụ sau:

```
tb.Rows.Add(row); // Added
tb.AcceptChanges(); // ...Commit changes
Console.WriteLine(row.RowState); // Unchanged
tb.Rows[0].Delete(); // Deleted
// Undo deletion
tb.RejectChanges(); // ...Roll back
Console.WriteLine(tb.Rows[0].RowState); // Unchanged
DataRow myRow;
MyRow = tb.NewRow(); // Detached
```

Hai phương thức AcceptChanges và RejectChanges của DataTable là tương đương với các thao tác commit và rollback trong một CSDL. Các phương thức này sẽ cập nhật mọi thay đổi xảy ra kể từ khi table được nạp, hoặc từ khi phương thức AcceptChanges

được triệu gọi trước đó. Ở ví dụ trên, chúng ta có thể khôi phục lại dòng bị xóa do thao tác xóa là chưa được commit trước khi phương thức RejectChanges được gọi. Điều đáng lưu ý nhất đó là, những thay đổi được thực hiện là ở trên table chứ không phải là ở data source.

ADO.NET quản lý 2 giá trị - ứng với 2 phiên bản hiện tại và nguyên gốc - cho mỗi column trong một dòng dữ liệu. Khi phương thức RejectChanges được gọi, các giá trị hiện tại sẽ được đặt khôi phục lại từ giá trị nguyên gốc. Điều ngược lại được thực hiện khi gọi phương thức AcceptChanges. Hai tập giá trị này có thể được truy xuất đồng thời thông qua các giá trị liệt kê DataRowVersion là: Current và Original:

```
DataRow r = tb.Rows[0];
r["DonGia"] = 14.95;
r.AcceptChanges();
r["DonGia"] = 16.95;
Console.WriteLine("Current: {0} Original: {1}",
r["Price", DataRowVersion.Current],
r["Price", DataRowVersion.Original]);
Kết quả in ra:
Current: 16.95 Original: 14.95
```

4.2.3. DataView

DataView đóng vai trò như tầng hiển thị dữ liệu lưu trữ trong DataTable. Nó cho phép người sử dụng sắp xếp, lọc và tìm kiếm dữ liệu.

```
//Giả sử đã có 1 dataset có tên là ds chứa dữ liệu của bảng DonHang
DataView dv = new DataView(ds.Tables["DonHang"]);
// Lọc ra tất cả các hàng có giá từ 10 đến 100
dv.RowFilter = "soluong>=10 and soluong<=100";
//Sắp xếp tăng dần theo số lượng nếu số lượng bằng nhau thì sắp xếp giảm
dần thêm đơn
giá
dv.Sort = "soluong, dongia DESC";
```

4.2.4. Nạp dữ liệu vào DataSet

Chúng ta đã biết cách thành lập một DataTable và xử lý dữ liệu theo kiểu từng dòng một. Phần này sẽ trình bày phương pháp để dữ liệu và lược đồ dữ liệu được nạp tự động từ CSDL quan hệ vào các table trong DataSet.

4.2.4.1. Dùng DataReader để nạp dữ liệu vào DataSet

Đối tượng DataReader có thể được sử dụng để liên hợp đối tượng DataSet hay DataTable trong việc nạp các dòng dữ liệu kết quả (của query trong DataReader).

```
cmd.CommandText = "SELECT * FROM nhanvien";
DBDataReader rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
DataTable dt = new DataTable("nhanvien");
dt.Load(rdr); // Nạp dữ liệu và lược đồ vào table
```

```
Console.WriteLine(rdr.IsClosed); // True
```

Đối tượng DataReader được tự động đóng sau khi tất cả các dòng dữ liệu được nạp vào table. Do đã sử dụng tham số CommandBehavior.CloseConnection trong phương thức ExecuteReader nên connection được đóng sau khi DataReader được đóng.

Nếu table đã có dữ liệu, phương thức Load sẽ trộn dữ liệu mới với các dòng dữ liệu đang có trong nó. Việc trộn này xảy ra chỉ khi các dòng dữ liệu có chung primary key. Nếu không có primary key được định nghĩa, các dòng dữ liệu sẽ được nối vào sau tập dữ liệu hiện tại. Chúng ta có thể sử dụng phương thức nạp chồng khác của phương thức Load để quy định cách thức làm việc. Phương thức Load với tham số kiểu enumeration LoadOption gồm 1 trong 3 giá trị OverwriteRow, PreserveCurrentValues, hoặc UpdateCurrentValues tương ứng với tùy chọn ghi đè nguyên dòng, giữ lại các giá trị hiện tại, hoặc cập nhật các giá trị hiện tại. Đoạn code dưới đây minh họa cách trộn dữ liệu vào các dòng hiện tại theo kiểu ghi đè các giá trị hiện tại:

```
cmd.CommandText = "SELECT * FROM nhanvien WHERE diachi='a'";
DBDataReader rdr = cmd.ExecuteReader();
DataTable dt = new DataTable("nhanvien");
dt.Load(rdr);
Console.WriteLine(dt.Rows[0]["HoTen"]); // giả sử giá trị nhận được là "tnv spider"
// Gán khóa chính
DataColumn[] col = new DataColumn[1];
col[0] = dt.Columns["Manv"];

dt.PrimaryKey = col;
DataRow r = dt.Rows[0]; // lấy dòng đầu tiên
r["HoTen"] = "ten moi"; // thay đổi giá trị của cột HoTen
// Do reader đã bị đóng sau khi nạp vào data table nên phải giờ phải
fill lại
rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);
// Trộn dữ liệu với các dòng hiện tại. Ghi đè các giá trị hiện tại
dt.Load(rdr, LoadOption.UpdateCurrentValues);
// Giá trị cập nhật đã bị ghi đè!!!
Console.WriteLine(dt.Rows[0]["HoTen"]); // "tnv spider"
```

4.2.5. Nạp dữ liệu vào DataSet bằng DataAdapter

Đối tượng DataAdapter có thể được dùng để nạp một table hiện có vào một table khác, hoặc tạo mới và nạp dữ liệu cho table từ kết quả của một query. Bước đầu tiên là tạo ra một đối tượng DataAdapter tương ứng với data provider cụ thể. Dưới đây là các ví dụ để tạo ra đối tượng DataAdapter:

Tạo từ Connection string và câu truy vấn SELECT:

```
String sql = "SELECT * FROM nhanvien";
```

```
SqlDataAdapter da = new SqlDataAdapter(sql, connStr);
Tạo từ đối tượng Connection và câu truy vấn SELECT:
```

```
SqlConnection conn = new SqlConnection(connStr);
SqlDataAdapter da = new SqlDataAdapter(sql, conn);
Gán đối tượng Command cho thuộc tính SelectCommand
```

```
SqlDataAdapter da = new SqlDataAdapter();
SqlConnection conn = new SqlConnection(connStr);
da.SelectCommand = new SqlCommand(sql, conn);
```

Sau khi đối tượng DataAdapter đã được tạo ra, phương thức Fill của nó được thực thi để nạp dữ liệu vào table (đang tồn tại hoặc tạo mới). Ở ví dụ dưới đây, một table mới được tạo ra với tên mặc định là “Table”:

```
DataSet ds = new DataSet();
// Tạo ra một DataTable, nạp dữ liệu vào DataTable, và đưa DataTable vào
// DataSet
int nRecs = da.Fill(ds); // trả về số lượng record được nạp vào
DataTable
// Nếu muốn đặt tên cho DataTable trong DataSet thay vì lấy tên mặc định
// thì sử dụng code như thế này
int nRecs = da.Fill(ds, "nhanvien ")
```

Với một table đang tồn tại, tác dụng của lệnh Fill tùy thuộc vào table có primary hay không. Nếu có, những dòng dữ liệu có khóa trùng với dòng dữ liệu mới sẽ được thay thế. Các dòng dữ liệu mới không trùng với dữ liệu hiện có sẽ được nối vào sau DataTable. (Xem thêm phần bài tập bên dưới).

4.2.6. Cập nhật CSDL bằng DataAdapter

Sau khi DataAdapter đã nạp dữ liệu vào table, connection sẽ được đóng, và các thay đổi sau đó đối sau đó tạo ra cho dữ liệu sẽ chỉ có ảnh hưởng trong DataSet chứ không phải là ở dữ liệu nguồn! Để thực sự cập nhật các thay đổi này lên nguồn dữ liệu, DataAdapter phải được sử dụng để khôi phục connection và gửi các dòng dữ liệu đã được thay đổi lên CSDL.

Ngoài SelectCommand, DataAdapter có thêm 3 thuộc tính Command nữa, gồm InsertCommand, DeleteCommand và UpdateCommand, làm nhiệm vụ thực hiện các thao tác tương ứng với tên thuộc tính của chúng (chèn, xóa, cập nhật). Các Command này được thực thi khi phương thức Update của DataAdapter được triệu gọi. Khó khăn nằm ở chỗ tạo ra các query command phức tạp này (cú pháp của câu lệnh SQL tương ứng càng dài dòng và phức tạp khi số lượng column nhiều lên). Rất may là các data provider đều có cài đặt một lớp gọi là CommandBuilder dùng để quản lý việc tạo các Command nói trên một cách tự động.

4.2.6.1. *CommandBuilder*

Một đối tượng CommandBuilder sẽ sinh ra các Command cần thiết để thực hiện việc cập nhật nguồn dữ liệu tạo ra bởi DataSet. Cách tạo đối tượng CommandBuilder là truyền đối tượng DataAdapter cho phương thức khởi dựng của nó; sau đó, khi phương thức DataAdapter.Update được gọi, các lệnh SQL sẽ được sinh ra và thực thi. Đoạn code dưới đây minh họa cách thức thay đổi dữ liệu ở một DataTable và cập nhật lên CSDL tương ứng bằng DataAdapter:

```
//Giả sử đã có 1 DataSet ds chứa dữ liệu của bảng khoa
DataTable dt= ds.Tables["khoa"];
// (1) Dùng commandBuilder để sinh ra các Command cần thiết để update
SqlCommandBuilder sb = new SqlCommandBuilder(da);
// (2) Thực hiện thay đổi dữ liệu: thêm 1 khoa mới
DataRow drow = dt.NewRow();
drow["Makhoa"] = 12;
drow["tenkhoa"] = "abc";
dt.Rows.Add(drow);
// (3) Thực hiện thay đổi dữ liệu: xóa 1 khoa
dt.Rows[4].Delete();
// (4) Thực hiện thay đổi dữ liệu: thay đổi giá trị 1 dòng dữ liệu
dt.Rows[5]["tenkhoa"] = "this must be changed";
// (5) Tiến hành cập nhật lên CSDL
int nUpdate = da.Update(ds, "khoa");
MessageBox.Show("Số dòng được thay đổi: " + nUpdate.ToString()); // → 3
```

Có một số hạn chế khi sử dụng CommandBuilder: Command Select ứng với DataAdapter chỉ được tham chiếu đến 1 table, và table nguồn trong CSDL phải bao gồm một primary key hoặc một column chứa các giá trị duy nhất. Column này (hay tổ hợp các columns) phải được bao gồm trong command Select ban đầu.

4.2.6.2. *Đồng bộ hóa dữ liệu giữa DataSet và CSDL*

Như đã minh họa trong ví dụ này, việc sử dụng DataAdapter làm đơn giản hóa và tự động hóa quá trình cập nhật CSDL hoặc bất kỳ data source nào. Tuy nhiên, có một vấn đề ở đây: multi-user (nhiều người sử dụng). Mô hình Ngắt kết nối được dựa trên cơ chế Optimistic Concurrency, một cách tiếp cận mà trong đó các dòng dữ liệu ở data source không bị khóa (lock) giữa thời gian mà chúng được đọc và thời gian mà các cập nhật được áp dụng cho data source. Trong khoảng thời gian này, user khác có thể cũng cập nhật data source. Nếu có thay đổi xảy ra kể từ lần đọc trước đó thì phương thức Update sẽ nhận biết được và không cho áp dụng thay đổi đối với các dòng dữ liệu đó. Có hai phương án cơ bản để giải quyết lỗi concurrency (tương tranh) khi có nhiều cập nhật được áp dụng: roll back tất cả các thay đổi nếu như xuất hiện xung đột (violation), hoặc áp dụng các cập nhật không gây ra lỗi và xác định những cập nhật có gây ra lỗi để có thể xử lý lại.

4.2.6.3. Sử dụng Transactions để Roll Back nhiều cập nhật

Khi thuộc tính DataAdapter.ContinueUpdateOnErrors được thiết lập là false, một ngoại lệ sẽ được ném ra khi một thay đổi dòng dữ liệu không thể thực hiện được. Điều này sẽ ngăn các cập nhật tiếp theo được thực thi, nhưng lại không ảnh hưởng đến các cập nhật đã xuất hiện trước ngoại lệ đó. Do những cập nhật có thể có liên quan với nhau, ứng dụng thường là cần chiến lược hoặc là tất cả, hoặc là không (all-or-none). Cách dễ nhất để thực thi chiến lược này là tạo ra một transaction trong đó tất cả các command update sẽ được thực thi. Để thực hiện điều này, tạo ra một đối tượng SqlTransaction và gắn nó với SqlDataAdapter.SelectCommand bằng cách truyền nó cho hàm khởi dựng của nó. Nếu có ngoại lệ xảy ra, phương thức Rollback sẽ được thực thi để undo mọi thay đổi trước đó; nếu không có ngoại lệ nào xuất hiện, phương thức Commit được thực thi để áp dụng tất cả các command update. Dưới đây là một ví dụ:

```

SqlDataAdapter da = new SqlDataAdapter();
SqlCommandBuilder sb = new SqlCommandBuilder(da);
SqlTransaction tran;
SqlConnection conn = new SqlConnection(connStr);
conn.Open(); // Connection phải được dùng với Transaction
// (1) Tạo ra một transaction
SqlTransaction tran = conn.BeginTransaction();
// (2) Gắn SelectCommand với transaction
da.SelectCommand = new SqlCommand(sql, conn, tran);
DataSet ds = new DataSet();
da.Fill(ds, "docgia");
//
// Code ở phần này thực hiện các cập nhật lên các dòng dữ liệu ở DataSet
try
{
    int updates = da.Update(ds, "docgia");
    MessageBox.Show("Cập nhật: " + updates.ToString());
}
// (3) Nếu có ngoại lệ xuất hiện, roll back mọi cập nhật trong
// transaction
catch (Exception ex)
{
    MessageBox.Show(ex.Message); // Lỗi khi cập nhật
    if (tran != null)
    {
        tran.Rollback(); // Roll back mọi cập nhật
        tran = null;
        MessageBox.Show("Tất cả các cập nhật đã bị roll back.");
    }
}
finally
{
    // (4) Nếu không có lỗi, commit tất cả các cập nhật
    if (tran != null)
}

```

```

{
tran.Commit();
MessageBox.Show("Tất cả đã được cập nhật thành công. ");
tran = null;
}
}
conn.Close();

```

4.2.6.4. Xác định các dòng gây ra lỗi cập nhật

Khi thuộc tính DataAdapter.ContinueUpdateOnErrors được thiết lập là True, các xử lý sẽ không ngưng nếu có một dòng dữ liệu không thể cập nhật được. Thay vào đó, DataAdapter sẽ cập nhật tất cả các dòng dữ liệu không gây ra lỗi. Sau đó, lập trình viên có thể xác định các dòng dữ liệu không cập nhật được và quyết định cách xử lý chúng. Các dòng dữ liệu không cập nhật được có thể dễ dàng được xác định qua thuộc tính DataRowState của chúng (đã được trình bày trong phần mô tả DataRow).

Các dòng dữ liệu đã được cập nhật thành công sẽ có trạng thái Unchanged; trong khi đó các dòng dữ liệu không cập nhật thành công sẽ mang trạng thái Added, Deleted hoặc Modified. Đoạn code dưới đây minh họa cách lặp qua các dòng dữ liệu và xác định các dòng chưa được cập nhật.

```

// SqlDataAdapter da nạp dữ liệu của bảng docgia
da.ContinueUpdateOnError = true;
DataSet ds = new DataSet();
try
{
    da.Fill(ds, "docgia");
    DataTable dt = ds.Tables["docgia"];
    SqlCommandBuilder sb = new SqlCommandBuilder(da);
    // ... Các thao tác cập nhật
    dt.Rows[29].Delete(); // Delete
    dt.Rows[30]["HoTen"] = "try to change"; // Update
    dt.Rows[30][Madocgia] = 1234; // Update
    dt.Rows[31]["HoTen"] = "XYZ"; // Update
    DataRow drow = dt.NewRow();
    drow["HoTen"] = "tnv spider";
    drow["Madocgia"] = 25;
    dt.Rows.Add(drow); // insert
    // Submit updates
    int updates = da.Update(ds, "docgia");
    if (ds.HasChanges())
    {
        // Load rows that failed into a DataSet
        DataSet failures = ds.GetChanges();
        int rowsFailed = failures.Rows.Count;
        Console.WriteLine("Số dòng không thể cập nhật: " + rowsFailed);
        foreach (DataRow r in failures.Tables[0].Rows)
        {
            string state = r.RowState.ToString());
        }
    }
}

```

```

// Phải hủy bỏ thay đổi để hiển thị dòng đã bị xóa
if (r.RowState == DataRowState.Deleted)
    r.RejectChanges();
string iMadocgia= ((int)r["Madocgia"]).ToString();
string msg = state + " Madocgia: " + iMadocgia;
Console.WriteLine(msg);
}
}

```

Lưu ý rằng ngay cả khi thao tác xóa xuất hiện trước, nó cũng không có tác dụng đối với các thao tác khác. Câu lệnh SQL xóa hay cập nhật một dòng dữ liệu được dựa theo giá trị của primary key chứ không liên quan đến vị trí của nó. Ngoài ra các cập nhật trên cùng một dòng được kết hợp và đếm như là 1 cập nhật cho dòng đó bởi phương thức Update. Ở ví dụ trên, các cập nhật cho dòng 30 được tính như là 1 cập nhật.

4.2.6.5. Định nghĩa Relationships giữa các Table trong DataSet

Một DataRelation là một mối quan hệ parent-child giữa hai đối tượng DataTables. Nó được định nghĩa dựa trên việc so khớp các columns trong 2 DataTable. Cú pháp hàm khởi dụng là như sau:

```
public DataRelation(string relationName, DataColumn parentColumn,
DataColumn childColumn)
```

Một DataSet có một thuộc tính Relations giúp quản lý tập hợp các DataRelation đã được định nghĩa trong DataSet. Sử dụng phương thức Relations.Add để thêm các DataRelation vào tập hợp Relations. Ví dụ dưới đây thiết lập mối quan hệ giữa hai bảng khoa và docgia để có thể liệt kê danh sách các docgia của mỗi khoa.

```

string connStr="Data Source=tên máy chủ;Initial Catalog=quanlythuvien;
Trusted_Connection=yes";
DataSet ds = new DataSet();
// (1) Fill bảng docgia
string sql = "SELECT * FROM docgia";
SqlConnection conn = new SqlConnection(connStr);
SqlCommand cmd = new SqlCommand();
SqlDataAdapter da = new SqlDataAdapter(sql, conn);
da.Fill(ds, "docgia");
// (2) Fill bảng khoa
sql = "SELECT * FROM khoa";
da.SelectCommand.CommandText = sql;
da.Fill(ds, "khoa");
// (3) Định nghĩa relationship giữa 2 bảng khoa và docgia
DataTable parent = ds.Tables["khoa"];
DataTable child = ds.Tables["docgia"];
DataRelation relation = new DataRelation("khoa_docgia",
parent.Columns["makhoa"],
child.Columns["makhoa"]);
// (4) Đưa relationship vào DataSet
ds.Relations.Add(relation);

```

```

// (5) Liệt kê danh sách các đọc giả của từng khoa
foreach (DataRow r in parent.Rows)
{
    Console.WriteLine(r["tenkhoa"]); // Tên khoa
    foreach (DataRow rc in r.GetChildRows("khoa_docgia"))
    {
        Console.WriteLine(" " + rc["HoTen"]);
    }
}
/*
Ví dụ kết quả:
Khoa Tin
Nguyễn Văn Trung
Ngô Anh Tuấn
Lê Thanh Hoa
Khoa Toán
Nguyễn Thị Hoa
Trần Văn Phúc
*/
}
}

```

Khi một relationship được định nghĩa giữa 2 tables, nó cũng sẽ thêm một ForeignKeyConstraint vào tập hợp Constraints của DataTable con. Constraint này quyết định cách mà DataTable con bị ảnh hưởng khi các dòng dữ liệu ở phía DataTable cha bị thay đổi hay bị xóa. Trong thực tế, điều này có nghĩa là khi bạn xóa 1 dòng trong DataTable cha, các dòng con có liên quan cũng bị xóa – hoặc cũng có thể là các key bị đặt lại giá trị thành NULL. Tương tự như thế, nếu một giá trị key bị thay đổi trong DataTable cha, các dòng dữ liệu có liên quan trong DataTable con cũng có thể bị thay đổi theo hoặc bị đổi giá trị thành NULL.

Các luật như trên được xác định bởi các thuộc tính DeleteRule và UpdateRule của constraint. Các luật này được nhận các giá trị liệt kê sau đây:

- ✓ Cascade. Xóa/Cập nhật các dòng dữ liệu có liên quan trong DataTable con.
Đây là giá trị mặc định.
- ✓ None. Không làm gì.
- ✓ SetDefault. Thiết lập các giá trị khóa trong các dòng dữ liệu có liên quan của DataTable con thành giá trị mặc định của column tương ứng.
- ✓ SetNull. Thiết lập các giá trị khóa trong các dòng dữ liệu có liên quan của DataTable con thành null

Xem xét ví dụ dưới đây:

```
// (1) Thêm một dòng với khóa mới vào DataTable con
```

```

DataRow row = child.NewRow();
row["Makhoa"] = 999; // giả sử trong bảng Khoa không có record nào có
Makhoa = 999
child.Rows.Add(row); // Không được do 999 không tồn tại trong DataTable
// (1) Xóa dòng mà không có khóa
// (2) Xóa một dòng trong DataTable cha
row = parent.Rows[0];
row.Delete(); // Xóa các dòng trong DataTable con có khóa này
// (3) Tạm thời vô hiệu hóa constraints và thử thêm dòng mới
ds.EnforceConstraints = false;
row["Makhoa"] = 999;
child.Rows.Add(row); // Được chấp nhận!!!
ds.EnforceConstraints = true; // Kích hoạt constraint trở lại
// (4) Thay đổi constraint để đặt các dòng dữ liệu thành null nếu
// DataTable thay đổi
((ForeignKeyConstraint)child.Constraints[0]).DeleteRule = Rule.SetNull;

```

Lưu ý rằng thuộc tính EnforeceConstraint được đặt thành false sẽ làm vô hiệu hóa tất cả các constraint – điều này trong thuật ngữ CSDL gọi là bỏ qua tính toàn vẹn tham chiếu. Điều này cho phép một khoa được bổ sung vào thậm chí khi cả column Makhoa không tương ứng với dòng nào trong bảng Khoa. Nó cũng cho phép một dòng Khoa được xóa ngay cả khi có nhiều dòng tương ứng với nó trong bảng docgia.

4.2.6.6. Sử dụng Data Binding

Chúng ta đã đề cập đến các điều khiển để thiết kế giao diện như TextBox, ListBox, RadioButton, ComBoBox và các điều khiển của ADO.NET như DataSet, DataTable và DataView. Các điều khiển này làm việc một cách độc lập với nhau, tuy nhiên trong một số tình huống chúng cần kết hợp lại với nhau. Ví dụ ta cần hiển thị tên Khoa từ cơ sở dữ liệu ra 1 TextBox, khi đó ta cần tạo ra 1 DataSet chứa dữ liệu của bảng Khoa và 1 TextBox, sau đó liên kết dữ liệu trong DataSet vào TextBox. Sự kết hợp giữa hai điều khiển này có thể sử dụng DataBinding.

4.2.6.7. Các loại của Binding

ADO.NET cung cấp 2 loại Binding:

- DataBinding đơn giản (Simple DataBinding): Tại một thời điểm, một giá trị đơn trong DataSet có thể bị buộc vào bất kỳ một điều khiển.

Ví dụ: giả sử đã có 1 DataSet ds chứa dữ liệu của bảng Khoa, cần buộc tên Khoa vào TextBox txttenkhoa:

```
txttenkhoa.DataBindings.Add("Text", ds, "khoa.tenkhoa");
```

Khi đó mọi thay đổi trên DataSet ds sẽ ảnh hưởng đến TextBox txtdocgia và ngược lại.

- DataBinding phức tạp (Complex DataBinding): Các dữ liệu trong DataSet bị buộc vào một điều khiển thay vì chỉ một giá trị đơn. Chỉ có DataGridView và ComboBox hỗ trợ chức năng DataBinding phức tạp.

Ví dụ: giả sử đã có 1 DataSet ds chứa dữ liệu của bảng Khoa, cần buộc tên khoa vào ComboBox cmbkhoa và buộc toàn bộ dữ liệu của bảng Khoa vào DataSet ds:

```
//Buộc tênkhoa của bảng Khoa trong DataSet ds vào cmbkhoa
cmbkhoa.DataSource = ds;
cmbkhoa.DisplayMember = "tenkhoa";
//Buộc toàn bộ dữ liệu của bảng Khoa trong DataSet ds vào DataGridView
dgvKhoa
dgvKhoa.DataSource = ds;
dgvKhoaDataMember = "Khoa";
```

4.2.6.8. Các nguồn dữ liệu của DataBinding

Nhiều thành phần có thể hoạt động như nguồn dữ liệu của DataBinding. Bất kỳ các thành phần được cài đặt từ giao diện Ilist có thể xem là như nguồn dữ liệu của DataBinding. Các ví dụ sau minh họa bằng cách nào để sử dụng DataTable, DataView, DataSet và Mảng như là nguồn dữ liệu để cài đặt DataBinding đơn giản và phức tạp.

DataTable:

Loại dữ liệu này lưu trữ dữ liệu của một bảng trong cơ sở dữ liệu.

```
DataTable t = ds.Tables["Khoa"];
//DataBinding đơn giản
txtTenKhoa.DataBindings.Add("Text", t, "tenkhoa");
//DataBinding phức tạp
cmbKhoa.DataSource = t;
cmbKhoa.DisplayMember = "tenkhoa";
dgvKhoa.DataSource = t;
```

DataView:

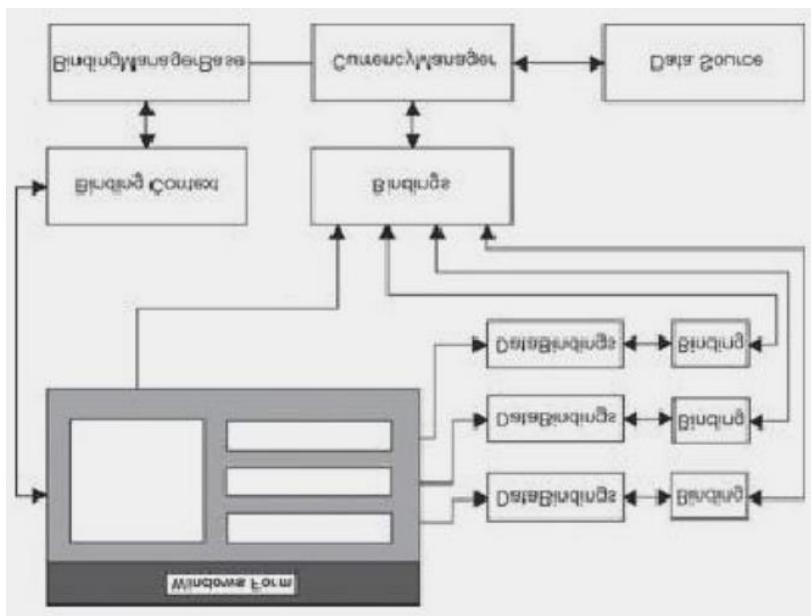
```
DataView dv = new DataView(ds.Tables["Khoa"]);
//DataBinding đơn giản
txtTenKhoa.DataBindings.Add("Text", dv, "tenkhoa");
//DataBinding phức tạp
cmbKhoa.DataSource = dv;
cmbKhoa.DisplayMember = "tenkhoa";
dgvKhoa.DataSource = dv;
```

Mảng:

```
int[] t = new int[4] { 12, 2, 3, 4 };
//DataBinding đơn giản
txtTenKhoa.DataBindings.Add("Text", t, "");
//DataBinding phức tạp
cmbKhoa.DataSource = t;
```

BindingContext

Sơ đồ bên dưới chỉ ra cách buộc dữ liệu từ nguồn dữ liệu vào các điều khiển trên Form. Phần này chủ yếu thảo luận về các lớp BindingContext, CurrencyManager và chỉ ra bằng cách nào chúng tương tác khi dữ liệu bị buộc vào một hoặc nhiều điều khiển trên Form:



Hình :

Mỗi Windows Form đều có một thuộc tính BindingContext. Một BindingContext có một tập hợp các BindingManagerBase, các đối tượng này được tạo ra khi dữ liệu buộc vào một điều khiển.

Nếu nguồn dữ liệu bao gồm một danh sách các đối tượng như a DataTable, DataView hoặc bất kỳ đối tượng nào cài đặt trên giao diện IList khi đó CurrencyManager sẽ được sử dụng. Một CurrencyManager có thể duy trì một vị trí hiện thời (current position) bên trong nguồn dữ liệu. Nếu nguồn dữ liệu chỉ trả về một giá trị đơn khi đó PropertyManager sẽ được lưu trữ bên trong BindingContext.

Một CurrencyManager hoặc PropertyManager chỉ được tạo ra một lần cho một nguồn dữ liệu. Nếu hai TextBox bị buộc vào 1 dòng của DataTable khi đó chỉ có một CurrencyManager được tạo ra trong BindingContext

Khi tạo ra một điều khiển trên form, điều khiển này sẽ được liên kết với bộ quản lý buộc dữ liệu trên form (form's binding manager). Khi tạo ra một điều khiển thì thuộc tính BindingContext của nó bằng null. Để buộc dữ liệu vào một điều khiển ta dùng thuộc tính

DataBindings như ví dụ bên dưới buộc mã khoa và tên khoa trong DataSet ds vào TextBox1 và textBox2 .

```
textBox1.DataBindings.Add("Text", ds, "khoa.makhoa");
textBox2.DataBindings.Add("Text", ds, "khoa.tenkhoa");
```

CurrencyManager and PropertyManager

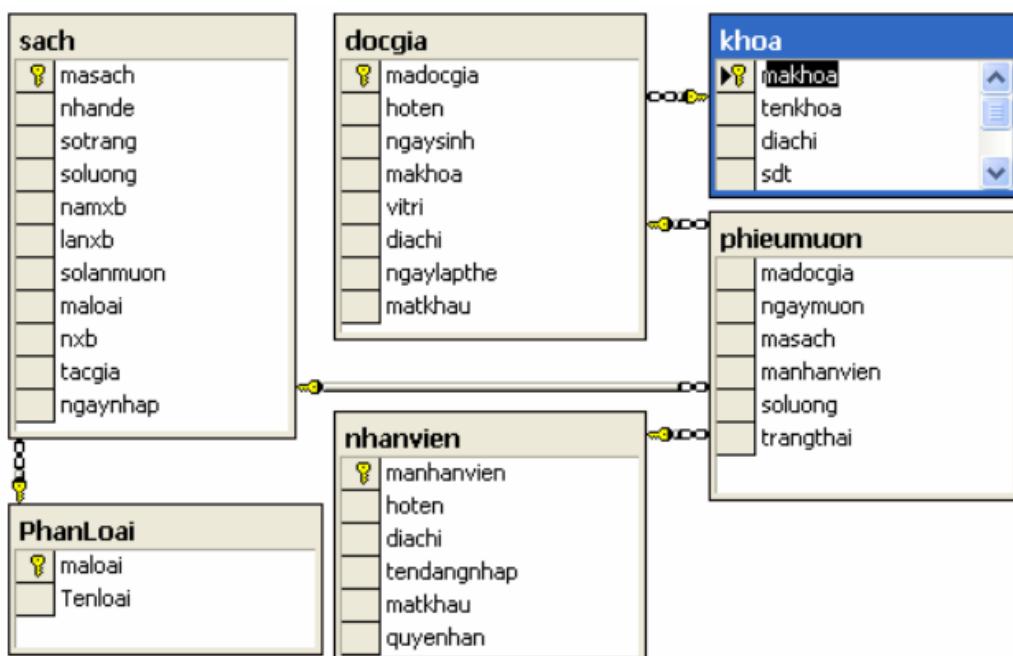
Khi buộc dữ liệu vào 1 điều khiển trên form khi đó một CurrencyManager hoặc một PropertyManager tương ứng sẽ được tạo ra. Mục đích của lớp này là xác định vị trí của mẫu tin hiện thời bên trong nguồn dữ liệu và khi vị trí này thay đổi thì dữ liệu trên các điều khiển bị buộc trên form sẽ tự động thay đổi theo.

Các thuộc tính của BindingContext:

Thuộc tính	Mô tả
Bindings	Tập hợp các đối tượng Binding được quản lý bởi CurrencyManager
Count	Số dòng được quản lý trong CurrencyManager
Current	Giá trị của các đối tượng hiện thời trong nguồn dữ liệu
Position	Gets hoặc sets đối tượng hiện thời trong danh sách các đối tượng được quản lý trong CurrencyManager

Bài tập thực hành về Dataset và binding

Ví dụ này sử dụng cơ sở dữ liệu quanlythuvien



Thiết kế form frmKhoa để nhập, xóa, lưu và duyệt qua các mẫu tin trong bảng Khoa như sau:



frmKhoa sử dụng các trường, phương thức và sự kiện sau:

```

using System.Data.SqlClient;
namespace baimau
{
    public partial class frmKhoa : Form
    {
        SqlConnection cn = new SqlConnection("Data Source=nhha;Initial Catalog=" -
            "quanlythuvien;Trusted_Connection=yes");
        SqlCommand cmdkhoa = new SqlCommand();
        SqlDataAdapter dakhoa = new SqlDataAdapter();
        DataSet ds = new DataSet();
        SqlCommandBuilder cb;
        public frmKhoa() { }

        private void BuocCacDieuKhien() { }
        private void frmKhoa_Load(object sender, EventArgs e) { }
        private void butFirst_Click(object sender, EventArgs e) { }
        private void butPre_Click(object sender, EventArgs e) { }
        private void butNext_Click(object sender, EventArgs e) { }
        private void butLast_Click(object sender, EventArgs e) { }
        private void butBosung_Click(object sender, EventArgs e) { }
        private void butLuu_Click(object sender, EventArgs e) { }
        private void butXoa_Click(object sender, EventArgs e) { }

    }
}

```

Các điều khiển	Thuộc tính
Form	Name: frmKhoa Text: Thông tin về bảng Khoa
Label	Tạo ra 4 lable với các Text: Mã Khoa, Tên Khoa, Địa chỉ, Số điện thoại
TextBox	Tạo ra 4 TextBox với các Name: txtMakhoa, txtTenKhoa, txtDiachi, txtSodienthoai

Button	Tạo ra 7 Button với các Name: butBosung, butLuu, butXoa, butFirst, butPre, butNext, butLast
dataGridView	Name: dataGridView1

Các phương thức

Phương thức BuocCacDieuKhien(): Buộc dữ liệu vào dataGridView1 và các textBox

```
private void BuocCacDieuKhien()
{
    //Buộc dữ liệu vào dataGridView1
    dataGridView1.DataSource = ds; dataGridView1.DataMember =
    "khoa";
    // Buộc dữ liệu vào các textBox
    txtMaKhoa.DataBindings.Add("Text", ds, "khoa.makhoa");
    txtTenKhoa.DataBindings.Add("Text", ds, "khoa.tenkhoa");
    txtdiachi.DataBindings.Add("Text", ds, "khoa.diachi");
    txtSodienthoai.DataBindings.Add("Text", ds, "khoa.sdt");
}
```

Sự kiện: frmKhoa_Load() được sử dụng để kết nối dữ liệu, tạo ra DataSet ds chứa toàn bộ dữ liệu của bảng khoa, buộc dữ liệu vào cho các điều khiển và tạo ra 1 SqlCommandBuilder cb để quản lý việc nhập thêm, xóa và lưu dữ liệu của SqlDataAdapter dakhoa .

```
private void frmKhoa_Load(object sender, EventArgs e)
{
    cn.Open(); // Kết nối dữ liệu
    cmdkhoa = new SqlCommand("select * from khoa", cn);
    dakhoa = new SqlDataAdapter(cmdkhoa);
    dakhoa.Fill(ds, "khoa");
    BuocCacDieuKhien();
    cb = new SqlCommandBuilder(dakhoa);
}
```

Sự kiện: butFirst_Click: Di chuyển con trỏ về mẫu tin đầu tiên

```
private void butFirst_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa"].Position = 0;
}
```

Sự kiện: butPre_Click: Di chuyển con trỏ về mẫu tin trước mẫu tin hiện thời

```
private void butPre_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa"].Position--;
}
```

Sự kiện: butNext_Click: Di chuyển con trỏ đến mẫu tin kế tiếp

```
private void butNext_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa"].Position++;
}
```

```
}
```

Sự kiện: butLast_Click: Di chuyển con trỏ về mẫu tin cuối cùng

```
private void butLast_Click(object sender, EventArgs e)
{
    int ViTriMauTinCuoiCung = this.BindingContext[ds,
"khoa"].Count - 1;
    this.BindingContext[ds, "khoa"].Position =
ViTriMauTinCuoiCung;
}
```

Sự kiện: butBosung_Click: Tạo mới một dòng

```
private void butBosung_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa"].AddNew();
}
```

Sự kiện: butLuu_Click: Di chuyển con trỏ về mẫu tin cuối cùng, nếu có thay đổi trong DataSet ds thì cập nhật lại dữ liệu, việc cập nhật nhờ vào SqlCommandBuilder cb. Các thao tác bổ sung và xóa chỉ được cập nhật vào cơ sở dữ liệu khi người sử dụng kích chuột vào nút Lưu

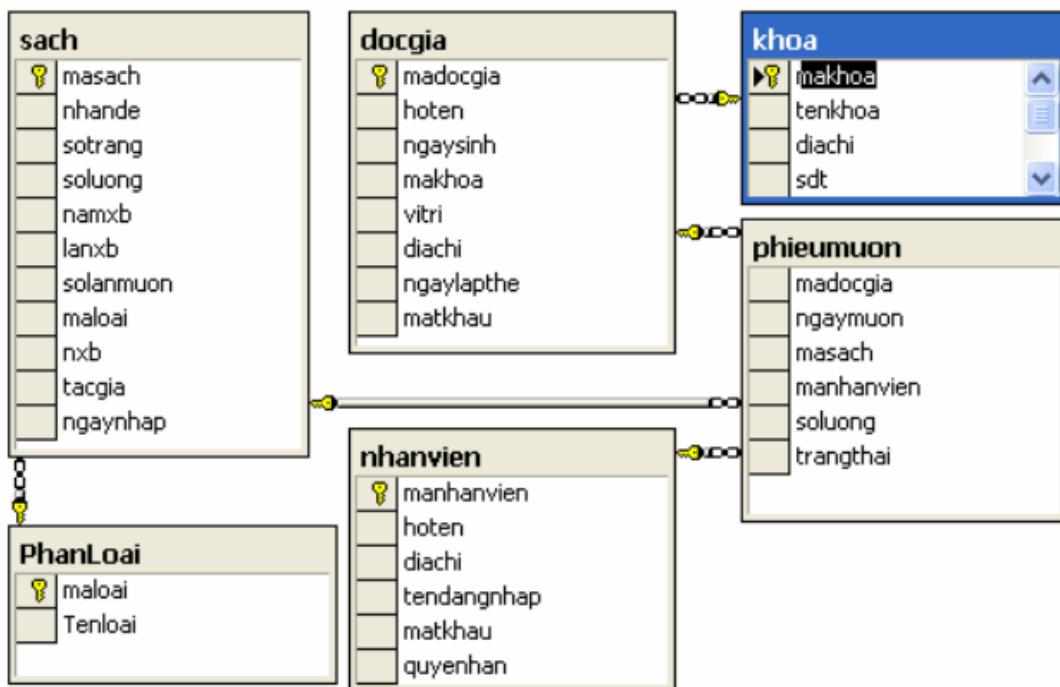
```
private void butLuu_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa"].EndCurrentEdit();
    if (ds.HasChanges() == true)
    {
        try
        {
            dakhoa.Update(ds, "khoa");
            MessageBox.Show("Da cap nhat");
        }
        catch (Exception ll) { MessageBox.Show(ll.Message); }
    }
}
```

Sự kiện: butXoa_Click: Lấy vị trí của con trỏ hiện thời, sau đó xóa đi mẫu tin này.

```
private void butXoa_Click(object sender, EventArgs e)
{
    int donghientai;
    donghientai = this.BindingContext[ds, "khoa"].Position;
    this.BindingContext[ds, "khoa"].RemoveAt(donghientai);
}
```

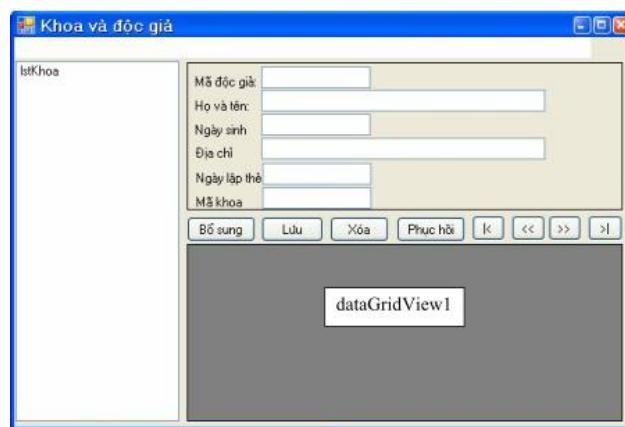
Bài thực hành về đặt quan hệ giữa các bảng, DataSet và DataBinding

Ví dụ này sử dụng cơ sở dữ liệu quanlythuvien.



Ví dụ này liên quan đến hai bảng dữ liệu: Khoa và docgia

Thiết kế form frmKhoa_Docgia để nhập, xóa, lưu, phục hồi và duyệt qua các mẫu tin trong bảng docgia cho từng khoa như sau:



frmKhoa sử dụng các trường, phương thức và sự kiện sau:

```

namespace baimau
{
    public partial class frmKhoa_Docgia : Form
    {
        SqlConnection cn = new SqlConnection("Data Source=nhha;Initial Catalog=" +
            "quanlythuvien;Trusted_Connection=yes");
        SqlCommand cmdkhoa = new SqlCommand(); SqlCommand cmddocgia = new SqlCommand();
        SqlDataAdapter dakhoa = new SqlDataAdapter();
        SqlDataAdapter dadocgia = new SqlDataAdapter();
        DataSet ds = new DataSet(); SqlCommandBuilder cb;
        public frmKhoa_Docgia()
        {
            InitializeComponent();
            Datquanhe(bangchinh, bangphu, khoachinh, khoaphu, tenquanhe);
        }
        private void Datquanhe(string bangchinh, string bangphu,
            string khoachinh, string khoaphu, string tenquanhe)
        {
            BuocCacDieuKhien();
            frmKhoa_Docgia_Load(sender, e);
            butLuu_Click(sender, e);
            butXoa_Click(sender, e);
            butBosung_Click(sender, e);
            butFirst_Click(sender, e);
            butLast_Click(sender, e);
            butPre_Click(sender, e);
            butNext_Click(sender, e);
            butPhuchoi_Click(sender, e);
        }
        private void BuocCacDieuKhien()
        {
        }
        private void frmKhoa_Docgia_Load(object sender, EventArgs e)
        {
        }
        private void butLuu_Click(object sender, EventArgs e)
        {
        }
        private void butXoa_Click(object sender, EventArgs e)
        {
        }
        private void butBosung_Click(object sender, EventArgs e)
        {
        }
        private void butFirst_Click(object sender, EventArgs e)
        {
        }
        private void butLast_Click(object sender, EventArgs e)
        {
        }
        private void butPre_Click(object sender, EventArgs e)
        {
        }
        private void butNext_Click(object sender, EventArgs e)
        {
        }
        private void butPhuchoi_Click(object sender, EventArgs e)
        {
        }
    }
}

```

Các điều khiển

Tên điều khiển	
ListBox	Name: lstKhoa
Form	Name: frmKhoa_Docgia Text: Khoa và đọc giả
Label	Tạo ra 6 label với các Text: Mã đọc giả, Họ và tên, Ngày sinh, Địa chỉ, Ngày lập thẻ và Mã khoa
TextBox	Tạo ra 6 TextBox với các Name: txtMadocgia, txtHoten, txtNgaysinh, txtDiachi, txtNgaylapthe, txtMakhoa
Button	Tạo ra 8 Button với các Name: butBosung, butLuu, butXoa, butPhuchoi, butFirst (<), butPre (<<), butNext (>>), butLast (>)
dataGridView	

Các phương thức

Hàm dựng frmKhoa_Docgia() để tạo giao diện:

```

public frmKhoa_Docgia()
{
    InitializeComponent();
}

```

Phương thức Datquanhe bao gồm các tham số: bảng chính, bảng phụ, khóa chính, khóa phụ và tên quan hệ. Phương thức này nạp dữ liệu của 2 bảng: bảng chính và bảng phụ vào DataSet DataSet ds, sau đó đặt quan hệ giữa 2 bảng trong DataSet DataSet ds.

```

private void Datquanhe(string bangchinh, string bangphu,

```

```

string khoachinh, string khoaphu, string tenquanhe)
{
    cn.Open();
    cmdkhoa = new SqlCommand("select * from " + bangchinh, cn);
    dakhoa = new SqlDataAdapter(cmdkhoa);
    cmddocgia = new SqlCommand("select * from " + bangphu, cn);
    dadocgia = new SqlDataAdapter(cmddocgia);
    ds = new DataSet();
    dakhoa.Fill(ds, bangchinh);
    dadocgia.Fill(ds, bangphu);
    DataColumn chinh = ds.Tables[bangchinh].Columns[khoachinh];
    DataColumn phu = ds.Tables[bangphu].Columns[khoaphu];
    DataRelation r = new DataRelation(tenquanhe, chinh, phu);
    ds.Relations.Add(r);
}

```

Phương thức BuocCacDieuKhien(): Buộc dữ liệu vào lstKhoa, dataGridView1 và các textBox

```

private void BuocCacDieuKhien()
{
    lstKhoa.DataSource = ds;
    lstKhoa.DisplayMember = "khoa.tenkhoa";
    dataGridView1.DataSource = ds;
    dataGridView1DataMember = "khoa.khoa_docgia";
    //khoa_docgia là tên quan hệ của 2 bảng khoa và docgia trong
DataSet ds
    txtMadocgia.DataBindings.Add("Text", ds,
"khoa.khoa_docgia.madocgia");
    txtHoten.DataBindings.Add("Text", ds,
"khoa.khoa_docgia.hoten");
    txtNgaysinh.DataBindings.Add("Text", ds,
"khoa.khoa_docgia.ngaysinh");
    txtdiachi.DataBindings.Add("Text", ds,
"khoa.khoa_docgia.diachi");
    txtNgaylapthe.DataBindings.Add("Text", ds,
"khoa.khoa_docgia.ngaylapthe");
    txtMakhoa.DataBindings.Add("Text", ds,
"khoa.khoa_docgia.makhoa");
}

```

Sự kiện frmKhoa_Docgia_Load: Đặt quan hệ giữa 2 bảng khoa và docgia trong DataSet DataSet ds, tạo ra 1 SqlCommandBuilder để quản lý việc lưu dữ liệu vào cơ sở dữ liệu ,buộc dữ liệu vào các điều khiển trên form:

```

private void frmKhoa_Docgia_Load(object sender, EventArgs e)
{
    Datquanhe("khoa", "docgia", "makhoa", "makhoa",
"khoa_docgia");
    cb = new SqlCommandBuilder(dadocgia);
    BuocCacDieuKhien();
}

```

Sự kiện: butFirst_Click: Di chuyển con trỏ về mẫu tin đầu tiên

```
private void butFirst_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa.khoa_docgia"].Position = 0; ;
}
```

Sự kiện: butPre_Click: Di chuyển con trỏ về mẫu tin trước mẫu tin hiện thời

```
private void butPre_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa.khoa_docgia"].Position--;
}
```

Sự kiện: butNext_Click: Di chuyển con trỏ đến mẫu tin kế tiếp

```
private void butNext_Click(object sender, EventArgs e)
{
    this.BindingContext[ds, "khoa.khoa_docgia"].Position++;
}
```

Sự kiện: butLast_Click: Di chuyển con trỏ về mẫu tin cuối cùng

```
private void butLast_Click(object sender, EventArgs e)
{
int ViTri = this.BindingContext[ds["khoa.khoa_docgia"]].Count - 1;
this.BindingContext[ds, "khoa"].Position = ViTri;
}
```

Sự kiện: butBosung_Click: Tạo mới một dòng

```
private void butBosung_Click(object sender, EventArgs e)
{
this.BindingContext[ds, "khoa.khoa_docgia"].AddNew();
}
```

Sự kiện: butLuu_Click: Di chuyển con trỏ về mẫu tin cuối cùng, nếu có thay đổi trong DataSet ds thì cập nhật lại dữ liệu, việc cập nhật nhờ vào SqlCommandBuilder cb. Các thao tác bổ sung và xóa chỉ được cập nhật vào cơ sở dữ liệu khi người sử dụng kích chuột vào nút Lưu

```
private void butLuu_Click(object sender, EventArgs e)
{
this.BindingContext[ds, "khoa.khoa_docgia"].EndCurrentEdit();
if (ds.HasChanges() == true)
{
try
{
dakhoa.Update(ds, "docgia");
MessageBox.Show("Da cap nhat");
}
catch (Exception ll) { MessageBox.Show(ll.Message); }
}
}
```

Sự kiện: butXoa_Click: Lấy vị trí của con trỏ hiện thời, sau đó xóa đi mẫu tin này.

```
private void butXoa_Click(object sender, EventArgs e)
{
```

```

        int donghientai;
        donghientai = this.BindingContext[ds,
    "khoa.khoa_docgia"].Position;
        this.BindingContext[ds,
    "khoa.khoa_docgia"].RemoveAt(donghientai);
    }

```

Sự kiện: buttPhuchoi_Click: Phục hồi lại các thao tác Bỏ sung và xóa, dữ liệu chỉ được phục hồi khi chưa lưu vào cơ sở dữ liệu

```

private void buttPhuchoi_Click(object sender, EventArgs e)
{
    this.BindingContext[ds,
    "khoa.khoa_docgia"].CancelCurrentEdit();
    ds.RejectChanges();
}

```

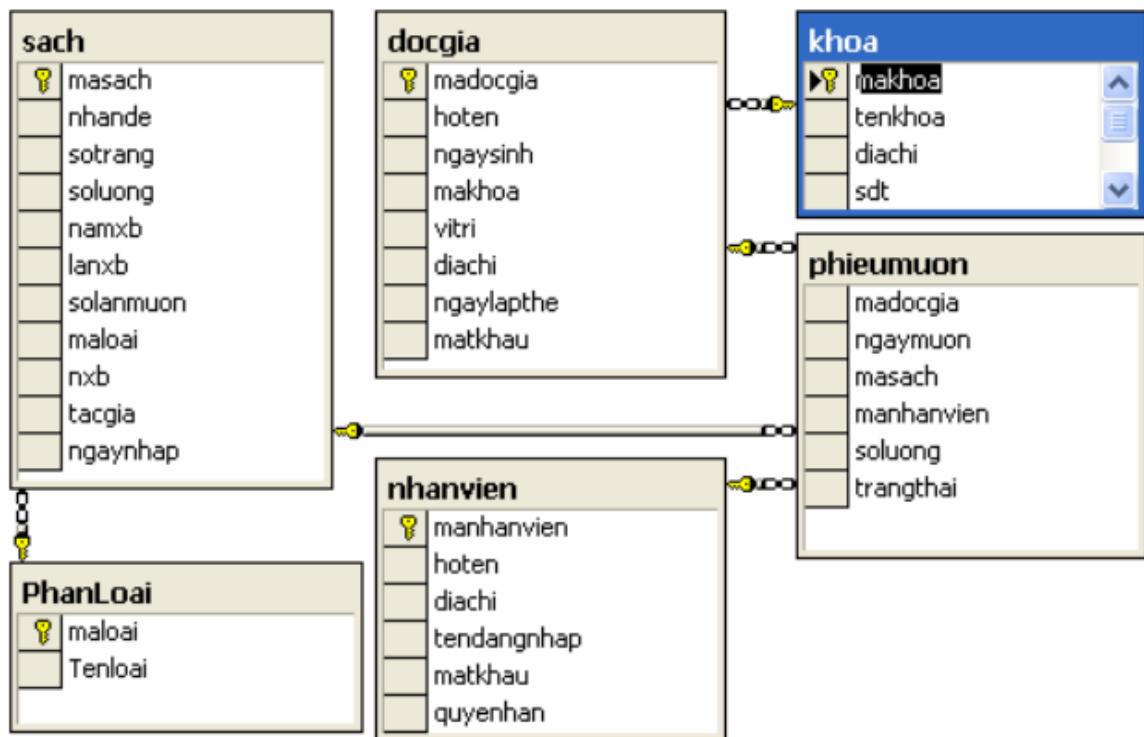
Câu hỏi trên lớp

1. Đối tượng DataTable nằm trong namespace nào?
2. DataTable trong DataSet mang ý nghĩa như một bảng trong CSDL đúng hay sai?
3. Cách tạo DataTable, tạo dòng, cột cho DataTable.
4. Thành phần chính của DataTable là gì?
5. Để duyệt qua số phần tử của một dòng hay cột trong DataTable thì có thể sử dụng phương thức nào?
6. Phương thức GetChanges của DataTable có tác dụng gì?
7. Tác dụng phương thức RejectChanges ?
8. Tác dụng phương thức AcceptChanges ?
9. Tác dụng phương thức Clear ?
10. Tác dụng phương thức Copy ?
11. Tác dụng phương thức Clone ?
12. Tác dụng phương thức WriteXml ?
13. Tác dụng phương thức ReadXml ?
14. Di chuyển dữ liệu với BindingSource có ý nghĩa như thế nào?

Bài tập tại lớp

Bài thực hành về DataAdapter và DataSet

Ví dụ này sử dụng cơ sở dữ liệu quanlythuvien



Thiết kế form frmtimkiemsach để tìm theo tên sách hoặc tên tác giả như sau:



frmtimkiemsach sử dụng các trường, phương thức và sự kiện sau:

```

using System.Data;
using System.Data.SqlClient;
namespace baimau
{
    public partial class frmtimkiemsach : Form
    {
        SqlConnection cn = new SqlConnection("Data Source=nhha;Initial Catalog=+"
            "quanlythuvien;Trusted_Connection=yes");
        SqlCommand cmd=new SqlCommand();
        SqlDataAdapter da= new SqlDataAdapter() ;
        DataSet ds= new DataSet() ;
        public frmtimkiemsach(){}
        private int Tongsoluong(){}
        private int Tongsoluongtk(DataView dv){}
        private void txttimkiem_KeyPress(object sender, KeyPressEventArgs e){}
        private void frmtimkiemsach_Load(object sender, EventArgs e){}
    }
}

```

Các điều khiển

Tên điều khiển	Thuộc tính
Form	Name: frmtimkiemsach Text:Tìm kiếm theo nhan đề hoặc tên tác giả
Label	Text: Nhập tên sách hoặc tên tác giả cần tìm
TextBox	Name: txttimkiem
dataGridView	Name: dataGridView1
statusStrip	Name: thanhtrangthai Items: Add thêm 2 statusLabel: với tên ketquatim và tongoluong

Các trường dữ liệu:

Tên Trường	Ý nghĩa
Cn	Dùng để kết nối đến cơ sở dữ liệu quanlythuvien
cmd	sqlCommand sử dụng câu lệnh select để hiển thị và tìm kiếm sách
da	SqlDataAdapter chứa cmd và cn
ds	DataSet chứa dữ liệu của bảng sách hoặc chứa kết quả tìm kiếm

Các phương thức

+ Hàm dựng frmtimkiemsach để tạo giao diện

```

public frmtimkiemsach()
{
    InitializeComponent();
}

```

+ Phương thức Tongsoluong: được sử dụng để tính tổng số lượng sách của các sách lưu trong Dataset ds.

```

private int Tongsoluong()
{
    int s=0;
    foreach (DataRow r in ds.Tables["sach"].Rows)
    {
        s += (int)r["soluong"];
    }
    return s;
}

```

+ Phương thức Tongsoluongtk tính tổng số lượng sách trong DataView dv, dv chứa thông tin các sách tìm kiếm được.

```

private int Tongsoluongtk(DataView dv)
{
    int s = 0;
    foreach (DataRow r in dv.ToTable("sach").Rows )
    {
        s += (int)r["soluong"];
    }
    return s;
}

```

+ **Sự kiện frmtimkiemsach_Load:** Nạp thông tin của 4 quyển sách đầu tiên theo thứ tự giảm dần của ngaynhap vào DataSet ds với tên bảng trong DataSet là sach, sau đó hiển thị thông tin của bảng sach trong DataSet vào dataGridView1, đưa tổng số sách và tổng số lượng sách trong DataSet vào thanh trạng thái.

```

private void frmtimkiemsach_Load(object sender, EventArgs e)
{
    cn.Open(); // Mở kết nối
    cmd.CommandText = "select top 4 * from sach order by ngaynhap desc";
    cmd.Connection = cn;
    da.SelectCommand = cmd;
    da.Fill(ds, "sach"); // Nạp dữ liệu vào DataSet
    dataGridView1.DataSource = ds.Tables["sach"]; // Nạp dữ liệu vào
    dataGridView1
    // Nạp dữ liệu vào thanh trạng thái
    thanhtrangthai.Items[0].Text = "Tổng số sách:" +
    ds.Tables["sach"].Rows.Count.ToString();
    thanhtrangthai.Items[1].Text = "Tổng số lượng:" +
    Tongsoluong().ToString();
}

```

+ **Sự kiện txttimkiem_KeyPress:** Khi người sử dụng nhấn Enter trên txttimkiem thì việc tìm kiếm tương đối bắt đầu: Tạo ra 1 DataView dv chứa dữ liệu của bảng sách trong DataSet ds, lọc trong DataView dv ra thông tin của các quyển sách gần giống với dữ liệu nhập trên txttimkiem, sau đó đưa kết quả lọc ra trên dataGridView1 và thanh trạng thái.

```

private void txttimkiem_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        DataView dv = new DataView(ds.Tables["sach"]); //Nạp dữ liệu
        vào DataView
        //bắt đầu lọc dữ liệu
        dv.RowFilter = "nhande like '%" + txttimkiem.Text + "%' or
        tacgia like '%'
        + txttimkiem.Text + "%'";
        dataGridView1.DataSource = dv; //Nạp kết quả lọc trong dv vào
        dataGridView1
        // Đưa số quyển sách và tổng số lượng sách lọc được vào thanh
        trang thái
        thanhtrangthai.Items[0].Text = "Số kết quả tìm thấy được: " +
        dv.Count.ToString() + "/" +
        ds.Tables["sach"].Rows.Count.ToString();
        thanhtrangthai.Items[1].Text = "Tổng số lượng tìm thấy được:" +
        Tongsoluongtk(dv).ToString() + "/" + Tongsoluong().ToString();
    }
}

```

Bài tập thêm

1. Tạo một table sinh viên với các cột mã sinh viên, họ tên, nơi sinh. Thiết kế giao diện cho phép nhập các thông tin trên, một button khi được nhấn sẽ thêm thông tin sinh viên vào lưới DataGridView, một button khác khi được nhấn sẽ hiện thông báo thông tin của từng sinh viên trên lưới.
2. Viết chương trình minh họa các phương thức của DataTable, gồm hai lưới DataGridView, mỗi phương thức gọi thực hiện bởi một button có thuộc tính text tương ứng với tên của phương thức. Truy vấn một bảng bất kỳ trong CSDL HoaDon, dùng store procedure.
3. Viết chương trình minh họa chức năng của đối tượng Binding, việc di chuyển giữa các mẫu tin trên lưới DataGridView được điều khiển bởi các button tương ứng.

4.3.CÁC ĐỐI TƯỢNG TRÌNH DIỄN DỮ LIỆU

4.3.1. DataView – Một góc nhìn của DataTable

Trong CSDL chúng ta có đối tượng View là một câu lệnh Select thể hiện các cách kết hợp dữ liệu khác nhau. Trong ADO.NET phía ứng dụng Client, đối tượng DataView cũng có ý nghĩa tương tự, DataView dùng để trình bày dữ liệu có trong DataTable dưới hình thức sắp xếp, lọc, tìm kiếm.

4.3.2. Tạo và sử dụng DataView

- Khai báo và khởi tạo DataView

```
    DataView view = new DataView();
    - Mỗi đối tượng DataView cần một nguồn dữ liệu DataTable
```

Ví dụ:

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString=ConfigurationSettings.AppSettings("strco
nn");
SqlDataAdapter da=new SqlDataAdapter("select * From
SanPham",conn);
DataTable dt=new System.Data.DataTable();
da.Fill(dt);
DataView view=new DataView();
view.Table=dt;
```

Khi chạy chương trình những gì hiển thị trong khung lưới không phải là DataTable mà chính là DataView. Ở đây chúng ta đã tạo mới một đối tượng DataView từ nguồn dữ liệu tb. Thực chất thì bất kỳ đối tượng DataTable nào cũng cung cấp thuộc tính DefaultView là một đối tượng DataView mặc định khi chúng ta gán DataGridView.DataSource = dt nghĩa là lệnh gán này tương đương với DataGridView.DataSource = tb.DefaultView

Để duyệt qua các dòng trong DataView chúng ta có thể dùng vòng lặp For **Ví dụ:**

```
for (int i = 0; i < dt.DefaultView.Count; i++)
{
    DataRowView _datarowview=dt.DefaultView[i];
    MessageBox.Show(_datarowview.Row["MaSanPham"].ToString());
}
```

Hoặc **foreach**

```
foreach (DataRowView item in dt.DefaultView)
{
    MessageBox.Show(item.Row["MaSanPham"].ToString());
}
```

4.3.3. Lọc dữ liệu với RowFilter, sắp xếp dữ liệu với thuộc tính Sort()

Thuộc tính RowFilter giúp ta lọc dữ liệu trong đối tượng DataView tương tự như mệnh đề Where trong phát biểu SQL của CSDL SQL Server.

Ví dụ sau đây, chúng ta truy vấn dữ liệu bảng khách hàng, đồng thời cung cấp một ô TextBox trên thanh ToolStrip. Khi người dùng gõ chuỗi văn bản vào ô TextBox chương trình sẽ tìm các tên khách hàng so khớp gần đúng nhất, các mục dữ liệu không khớp sẽ bị loại khỏi DataView.

	MaKH	TenKH	Địa Chỉ	Thành Phố	Điện Thoại
▶	BUMEM	Xây dựng Bình Mi...	155 Tô Hiến Thành	Cần Thơ	0718547896
	NASOCO	Xây dựng Nam Sơn	541 Phan Văn Trị...	Huế	0548523690
	PECCO	Xây lắp dầu khí	388 Trần Phú P7...	TP HCM	0886014797
	SECMC	Xây dựng điện	6 Tân Hóa P11 Q1	TP HCM	0887512087
	WACO	Xây dựng nước n...	1A Đề Thám	Hải Phòng	0318459047
*	WASECO	Xây dựng cảng th...	174 Nguyễn Thiệ...	Cần Thơ	0718123941

Hình 4.4: Lọc dữ liệu với DataView

Sau khi đưa dữ liệu lên lưới, trong sự kiện `KeyDown` của `TextBox` ta có

```
switch (e.KeyCode)
{
    case Keys.Enter:
        dt.DefaultView.RowFilter = String.Format("TenKH
Like '{0}%', {1}.Text) Me.Text =
String.Format("Found {0} Items ",
t.DefaultView.Count);
        break;

}
```

Chúng ta có thể sắp xếp dữ liệu trong `DataView` thông qua thuộc tính **Sort**. Cần chỉ định tên cột cần sắp xếp và chiều sắp xếp tăng hay giảm, mọi việc còn lại ADO.NET sẽ tự thực hiện. Ví dụ sau sẽ đọc tất cả các cột của bảng `KhachHang` vào `ComboBox` có tên `comboBoxTenCot`, khi chúng ta chọn một cột trên `ComboBox`, tên cột sẽ được dùng để sắp xếp. Mặc định sắp xếp tăng dần do đó để sắp xếp giảm dần ta phải nhấn chọn `“DESC”` có tên `cmdDESC`

Ví dụ

	TenKH	DiaChi	ThanhPho	DienThoai
	Bot giặt LIX	79 Bàn Cờ P3	Huế	0548562389
	Cơ điện nông ng...	311 Hai Bà Trưn...	TP HCM	0889703647
	SAMECO	Cơ khí chế tạo m...	13 Lê Quang Su...	Hà Nội
	CODACO	Cơ khí dân dụng	534 Lê Văn Sỹ P14	Nha Trang
	SCITEC	Cổ phần tư vấn &...	189 Kỳ Đồng P9 ...	TP HCM
	COPHACO	Cơ Phát	129 Nguyễn Du	Hải Phòng
	COTEC	Công nghệ mới	81 Trung Tự	Hà Nội

Hình 3.5: Sắp xếp dữ liệu với DataView

Trong sự kiện `TextChanged` của `ComboBox` ta có

```
string direc ="ASC";
    if(cmdDESC.Checked)
        direc="DESC";
    dt.DefaultView.Sort=String.Format("[{0}] {1}",
comboBoxTenCot.Text, direc);
```

Ngoài ra khi hiển thị dữ liệu trên khung lưới tùy theo mục đích nếu chúng ta không muốn cho người dùng tương tác như xóa mẫu tin, hiệu chỉnh mẫu tin, hay thêm mới mẫu tin chúng ta có thể sử dụng các thuộc tính:

- ✓ `AllowNew`: điều khiển cho hoặc không thêm mới mẫu tin.
- ✓ `AllowDelete`: điều khiển cho hoặc không xóa mẫu tin.
- ✓ `AllowEdit`: điều khiển cho hoặc không chỉnh sửa mẫu tin.

Ví dụ: `dt.DefaultView.AllowEdit = False` sẽ không cho chỉnh sửa mẫu tin trên khung lưới.

4.4. TRÌNH DIỄN DỮ LIỆU VỚI DATAGRIDVIEW

Như chúng ta đã biết, đối tượng `DataGridView` dùng để tiếp nhận và trình bày dữ liệu có trong `DataSet`, `DataTable`, `DataView`. Có thể nói `DataGridView` là đối tượng tiện dụng nhất để nhập dữ liệu dạng bảng.

Nguồn dữ liệu của `DataGridView` thường được lấy từ đối tượng `DataSet`, `DataTable`. Cho dù chúng ta gán dữ liệu là `DataSet` hay `DataTable` thì dữ liệu mà `DataGridView` dùng để hiển thị cuối cùng cũng là `DataView` như chúng ta vừa được biết ở

trên. Trường hợp DataTable thì DefaultView chính là dữ liệu mà DataGridView sẽ hiển thị. Với DataSet thì DefaultView là bảng được chỉ định trong thuộc tính DataMember.

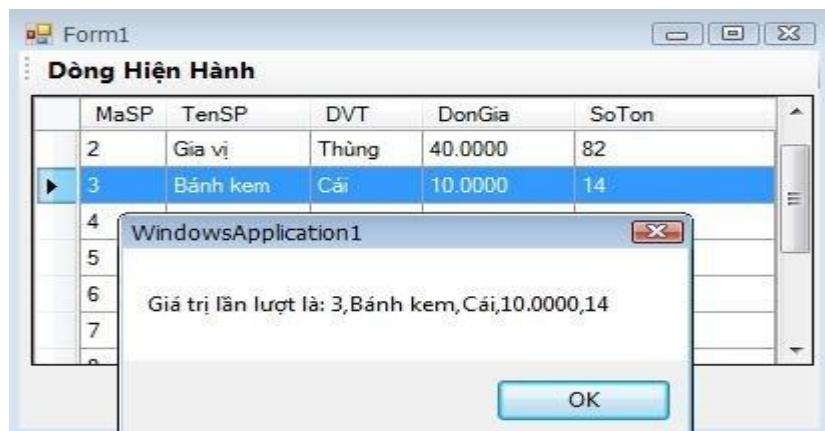
Ví dụ: trường hợp gán nguồn dữ liệu lấy từ DataSet

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString=ConfigurationSettings.AppSettings("strconn");
SqlDataAdapter da=new SqlDataAdapter("select * From SanPham",conn);
DataSet dt=new System.Data.DataSet();
da.Fill(ds);
dataGridView1.DataSource = ds.Table[0];
```

4.4.1. Xác định dòng, ô hiện hành.

Để lấy ra mẫu tin đang chọn trên điều khiển DataGridView, chúng ta sử dụng thuộc tính CurrentRow. Thuộc tính CurrentRow trả về đối tượng DataGridViewRow chứa thông tin về dòng đang chọn trên khung lưới.

Ví dụ:



Hình 4.6: Giá trị của CurrentRow

Chạy chương trình và nhấn button “Dòng Hiện Hành” sẽ hiện thông báo gồm các thông tin của dòng đang được chọn thông qua thuộc tính CurrentRow. Sau đây là code cho Button “Dòng Hiện Hành”.

```
dataGridView1.CurrentRow.Cells["Chỉ số cột hay tên cột"];
```

Tương tự như thuộc tính CurrentRow, thuộc tính CurrentCell sẽ giúp chúng ta lấy thông tin ô (Cell) hiện hành như sau:

Ví dụ:



Hình 3.7: Giá trị của CurrentCell

Khi nhấn Button “Ô Hiện Hành” sẽ hiện thông báo về ô đang được chọn trên khung lưới. Button “Ô Hiện Hành” như sau:

```
MessageBox.Show(dataGridView1.CurrentCell.Value.ToString());
```

Ngoài ra, thuộc tính ColumnCount và RowCount sẽ giúp chúng ta biết được số dòng và số cột đang trình bày trên khung lưới.

Bài tập tại lớp

1. Viết chương trình cho phép lọc dữ liệu (Filter) cột được chọn trong combobox (bảng khách hàng) theo nội dung nhập từ một textbox. Áp dụng thuộc tính tương ứng của DataView để làm.
2. Viết chương trình cho phép sắp xếp (Sort) dữ liệu theo cột được chọn trong combobox (bảng khách hàng). Áp dụng thuộc tính tương ứng của DataView để làm.

Bài tập về nhà

1. Viết chương trình cho phép lọc (Filter) dữ liệu theo cột được chọn trong combobox (bảng sản phẩm) theo nội dung nhập từ một textbox. Áp dụng thuộc tính tương ứng của DataView để làm.
2. Viết chương trình cho phép sắp xếp (Sort) dữ liệu theo cột được chọn trong combobox (bảng sản phẩm). Áp dụng thuộc tính tương ứng của DataView để làm.

3. Viết chương trình cho phép lọc (Filter) dữ liệu theo cột được chọn trong combobox (bảng khách hàng) theo nội dung nhập từ một textbox. Việc lọc dữ liệu thực hiện bởi câu truy vấn viết bằng Store Procedured.
4. Viết chương trình cho phép sắp xếp (Sort) dữ liệu theo cột được chọn trong combobox (bảng khách hàng). Việc sắp xếp dữ liệu thực hiện bởi câu truy vấn Store Procedured.

4.4.2. Tùy biến điều khiển nhập liệu trên DataGridView

Chúng ta có thể dễ dàng thêm các điều khiển như: Button, TextBox, ComboBox, CheckBox, ... vào điều khiển DataGridView phục vụ cho các mục đích công việc khác nhau. Việc thêm các điều khiển có thể thực hiện trực quan trong môi trường thiết kế hoặc viết mã.

4.4.2.1. Thêm cột điều khiển TextBox vào DataGridView.

Chúng ta thường gán thẳng DataTable vào thuộc tính DataSource của DataGridView và trong DataTable có bao nhiêu cột thì các cột hiển thị bấy nhiêu trên khung lưới. Tuy nhiên, chúng ta cũng có thể tạo thêm các cột cho DataGridView hoặc ánh xạ các cột thay thế cột dữ liệu nguồn trong DataSource.

Ví dụ:

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString=ConfigurationSettings.AppSettings("strconn");
SqlDataAdapter da=new SqlDataAdapter("select * From SanPham",conn);
DataTable dt=new System.Data.DataTable();
da.Fill(dt);

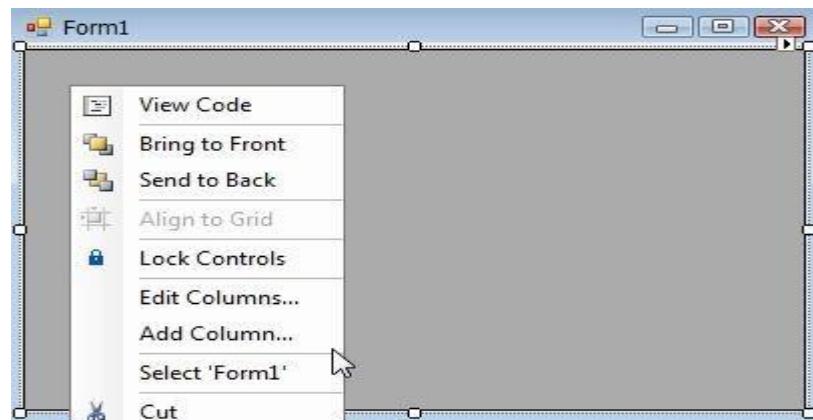
dataGridView1.DataSource = dt;
```

Chúng ta có được hình ảnh lưới DataGridView1 như sau:

	MaSP	TenSP	DVT	DonGia	SoTon
1	Rượu	Chai	230.5000	201	
2	Gia vị	Thùng	40.0000	82	
3	Bánh kem	Cái	10.0000	14	
4	Bơ	Kg	38.0000	30	
5	Bánh mì	Cái	8.0000	435	
6	Nem	Kg	23.7900	97	
7	Táo Đỏ	Kg	5.0000	33	
8	Giò lợn	Thùng	20.5000	100	

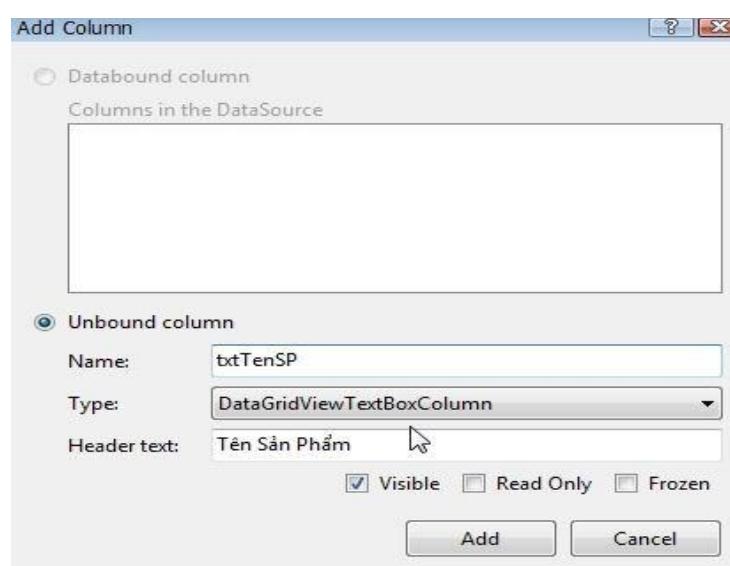
Hình 4.8: Hình ảnh lưới ban đầu

Bây giờ chúng ta tạo mới một cột TextBox như sau: Nhấn chuột phải lên khung lưới và chọn “Add Column...”



Hình 4.9: Menu khi chọn thêm control

Sau đó chúng ta khai báo tên cột, loại cột và tiêu đề cột



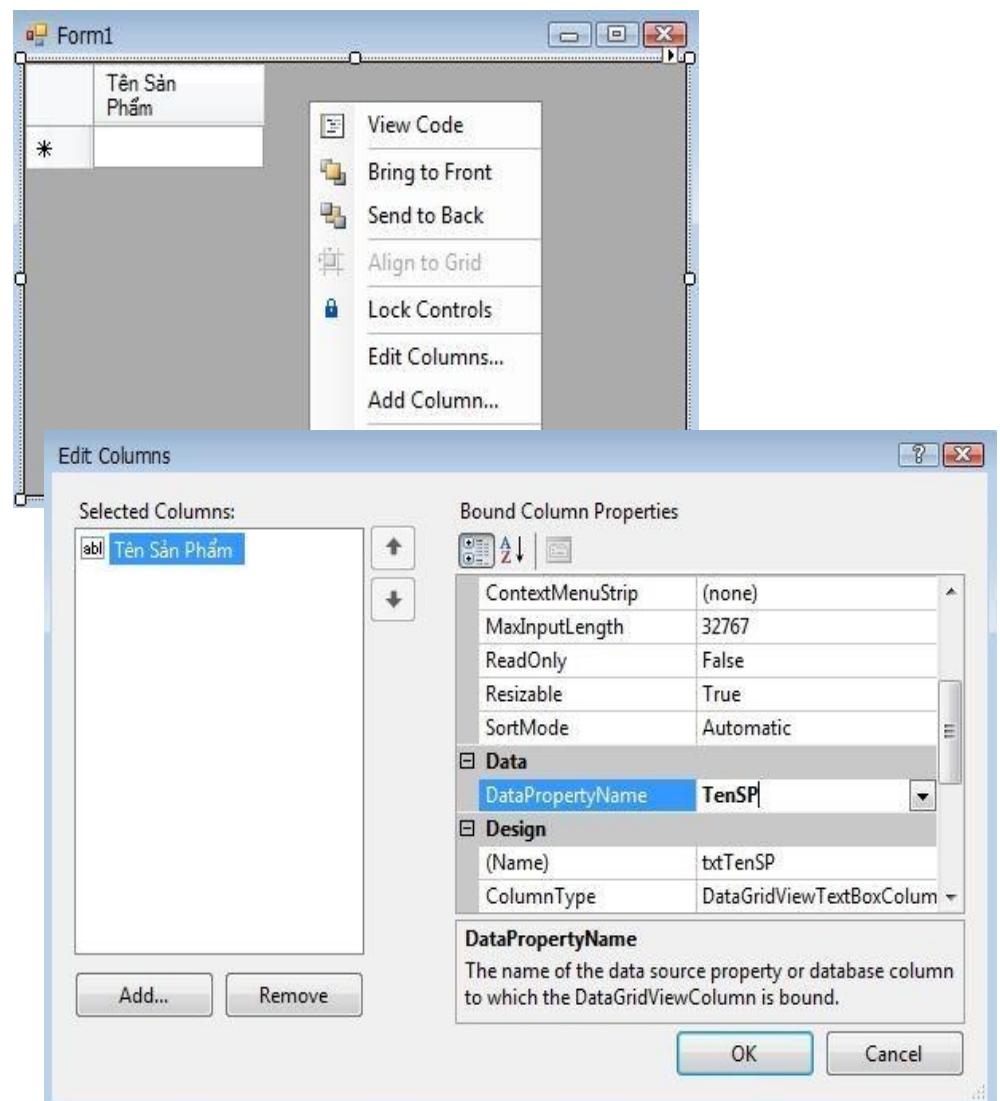
Hình 4.10: Hình ảnh khai báo thuộc tính column

Chạy chương trình chúng ta có hình ảnh của lưới

Tên Sản Phẩm	MaSP	TenSP	DVT	DonGia	SoTon
	1	Rượu	Chai	230.5000	201
	2	Gia vị	Thùng	40.0000	82
	3	Bánh kem	Cái	10.0000	14
	4	Bơ	Kg	38.0000	30
	5	Bánh mì	Cái	8.0000	435
	6	Mì...	Kg	20.0000	67

Hình 4.11: Hình ảnh sau khi thêm Column Textbox

Ta thấy, dữ liệu của DataTable nằm bên phải, còn cột dữ liệu trống do chúng ta tạo nằm bên trái. Nay giờ chúng ta ánh xạ tên của một cột dữ liệu vào cột dữ liệu do chúng ta mới tạo ra. Nhấn chuột phải vào khung lưới, chọn Edit Columns... từ menu hiện ra, hộp thoại Edit xuất hiện, ta chọn thuộc tính DataPropertyName và nhập tên cột dữ liệu của DataSource, chẳng hạn TenSP.



Hình 4.12: Gán nguồn cho columTextbox

Lúc này chạy chương trình DataGridView1 có hình ảnh như sau

The screenshot shows the running application 'Form1'. A DataGridView control displays a list of products. A tooltip is shown over the first row, pointing to the 'Tên Sản Phẩm' column. The tooltip text is: 'Dữ liệu cột TenSP được ánh xạ sang cột chúng ta vừa tạo.'

	Tên Sản Phẩm	Mã SP	DVT	Đơn Giá
▶	Rượu	1	Chai	230.5
	Gia vị		Thùng	40.00
	Bánh kem	3	Cái	10.00
	Bơ	4		
	Bánh mì	5		
	Nem	6		
	Táo Đỏ	7	Kg	5.000
	Cá hộp	8	Thùng	62.50

Hình 4.13: Hình ảnh lưới sau khi gán nguồn.

Ngoài ra còn có nhiều tùy biến khác từ menu chuột phải lên khung lưới như *Readonly, Visible, SortMode, DefaultCellStyle...* để tùy chỉnh tương tác với cột theo yêu cầu đặt ra cũng như định dạng trang trí cho khung lưới.

4.4.2.2. Thêm cột điều khiển ComboBox vào DataGridView

Trường hợp một cột dữ liệu là số nhưng chúng ta muốn hiển thị dưới dạng chuỗi mô tả liên quan, lúc này đối tượng ComboBox thêm vào lưới sẽ giải quyết vấn đề này.

Ví dụ:

```
SqlConnection conn=new SqlConnection();
conn.ConnectionString=ConfigurationSettings.AppSettings("strconn");
SqlDataAdapter da=new SqlDataAdapter("Select * From HoaDon ",conn);
DataTable dt=new System.Data.DataTable();
da.Fill(dt);

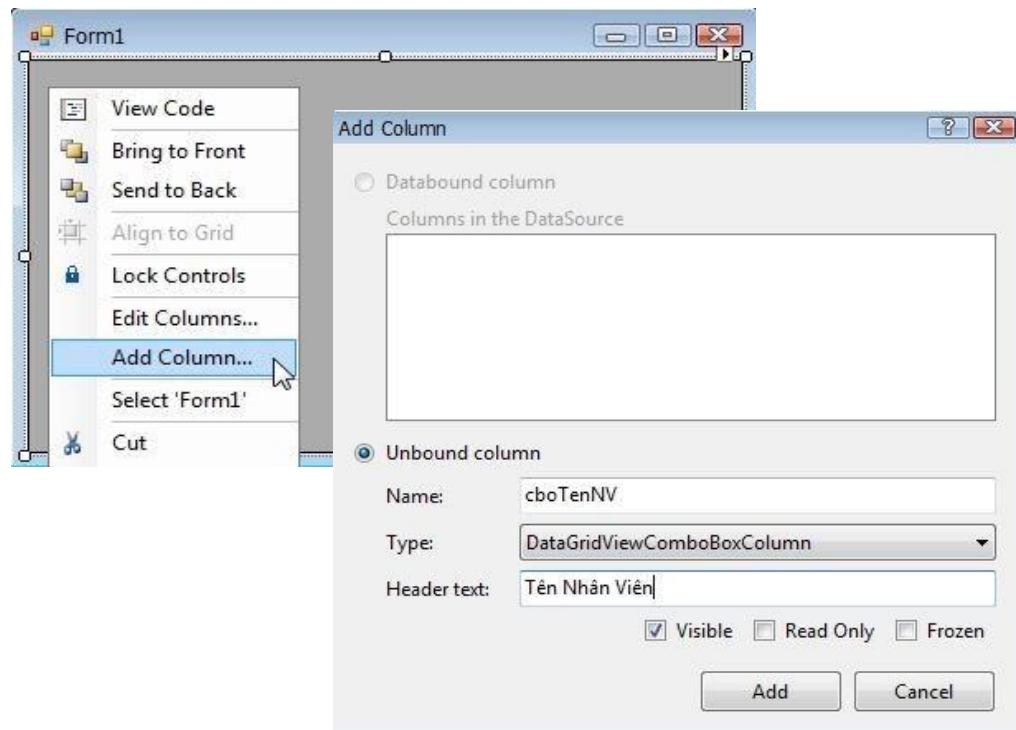
dataGridView1.DataSource = dt;
```

Chạy code trên ta được hình ảnh thông tin các hóa đơn như sau

MaHD	MaKH	MaNV	NgayLHD	NgayGiao
10144	SCITEC	7	3/30/2003	4/14/2003
10145	SAMACO	9	4/1/2003	4/16/2003
10148	VITICO	1	4/6/2003	4/21/2003
10150	CODACO	4	4/9/2003	4/24/2003
10156	COTEC	4	4/20/2003	5/5/2003
10157	DHP	4	4/21/2003	5/6/2003
10158	SAMECO	6	4/22/2003	5/7/2003
10162	ASECO	6	4/29/2003	5/14/2003
10163	TRACODI	4	4/30/2003	5/15/2003

Hình 4.14: Hình ảnh hóa đơn trên lưới ban đầu.

Bây giờ chúng ta muốn các hóa đơn hiển thị tên nhân viên lập chử không phải mã nhân viên, lúc này ta phải thêm một ComboBox vào lưới như sau. Nhấn chuột phải lên lưới, chọn Add Column... và điền tên, chọn loại cột là ComboBoxColumn và đặt tiêu đề cho cột.



Hình 4.15: Hình ảnh khi chọn thêm column.

Lúc này chạy chương trình hình ảnh sẽ như sau

Tên Nhân Viên	MaHD	MaKH	MaNV	NgayLHD	NgayGiao
	10144	SCITEC	7	3/30/2003	4/14/2003
	10145	SAMACO	9	4/1/2003	4/16/2003
	10148	VITICO	1	4/6/2003	4/21/2003
	10150	CODACO	4	4/9/2003	4/24/2003
	10156	COTEC	4	4/20/2003	5/5/2003
	10157	DHP	4	4/21/2003	5/6/2003
	10158	SAMECO	6	4/22/2003	5/7/2003
	10162	ASECO	6	4/29/2003	5/14/2003
	10163	TRACODI	4	4/30/2003	5/15/2003

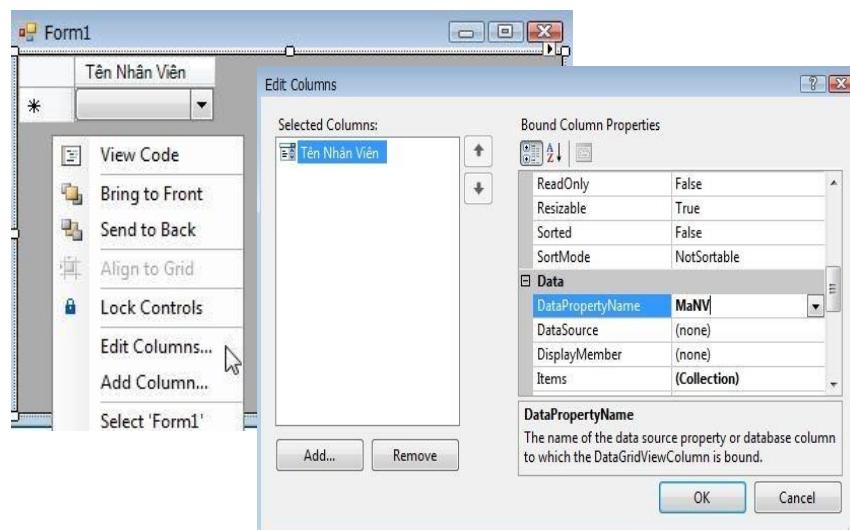
Hình 4.16: Hình ảnh sau khi chọn thêm column Combobox.

Cột ComboBox vừa tạo chưa hiển thị tên nhân viên tương ứng với mã nhân viên được. Do đó ta sẽ ánh xạ cột MaNV sang bằng cách gán MaNV cho thuộc tính DataPropertyName trong hộp thoại Edit

Columns...đồng thời khai báo DataSource, DisplayMember, ValueMember cho ComboBox cboTenNV như sau:

```
da.Fill(ds, "NhanVien") ;
cboTenNV.DataSource = ds.Tables("NhanVien");

cboTenNV.DisplayMember = "Ten" ;
cboTenNV.ValueMember = "MaNV"
DataGridView1.DataSource = ds.Tables("HoaDon")
```



Hình 4.17: Gán thuộc tính cho column Combobox.

Và chạy chương trình lúc này sẽ có được điều ta muốn là các nhân viên lập hóa đơn sẽ hiển thị tên thay vì hiển thị mã như lúc trước.

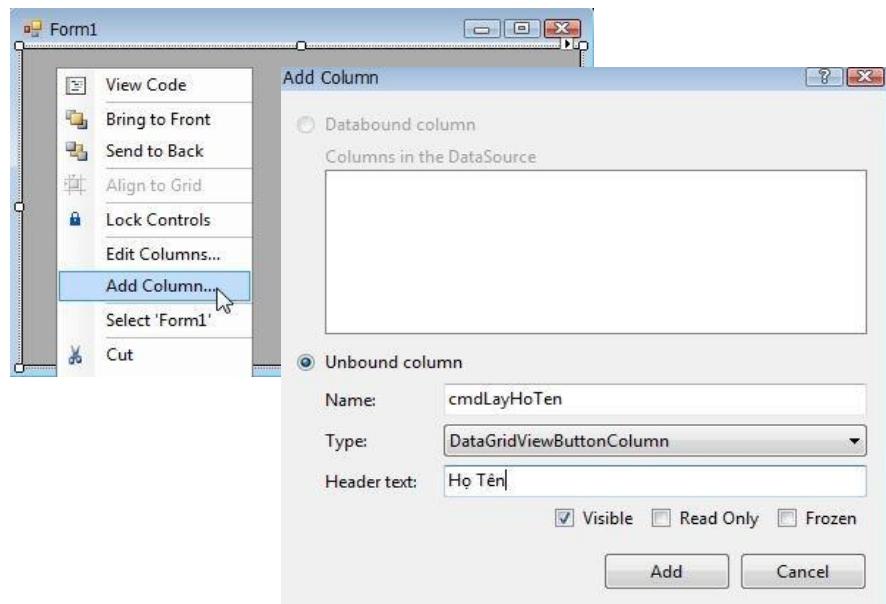
Tên Nhân Viên	MaHD	MaKH	NgayLHD	NgayGiao
Hoàng	10144	SCITEC	3/30/2003	4/14/2003
Lan	10145	SAMACO	4/1/2003	4/16/2003
Nga	10148	VITICO	4/6/2003	4/21/2003
Ngọc	10150	CODACO	4/9/2003	4/24/2003
Ngọc	10156	COTEC	4/20/2003	5/5/2003
Ngọc	10157	DHP	4/21/2003	5/6/2003
Thắng	10158	SAMECO	4/22/2003	5/7/2003
Thắng	10162	ASECO	4/29/2003	5/14/2003

Hình 4.18: Kết quả sau khi gán thuộc tính cho column Combobox.

4.4.3. Thêm cột điều khiển Button vào DataGridView.

Để thêm điều khiển Button vào điều khiển DataGridView, chúng ta thực hiện tương tự như hai trường hợp trên. Áp dụng khi muốn người dùng tác động lên dữ liệu của một dòng nào đó.

Ví dụ: Thêm điều khiển Button vào DataGridView, người dùng nhấn vào



Hình 4.19: Hình ảnh khi thêm column

Trong sự kiện Load của Form ta có code sau.

```
SqlConnection conn=new SqlConnection();

conn.ConnectionString=ConfigurationSettings.AppSettings("strconn") ;
    DataTable dt=new DataTable();
    SqlDataAdapter da=new SqlDataAdapter("Select * From
NhanVien",conn);
    da.Fill(dt);
    dataGridView1.DataSource=dt;
    foreach (DataGridViewRow item in dataGridView1.Rows)
{
    item.Cells[0].Value="Lấy họ tên";
}
```

Chạy chương trình chúng ta thấy

Họ Tên	MaNV	Ho	Ten	Nu
Lấy Họ Tên	1	Nguyễn Ngọc	Nga	<input type="checkbox"/>
Lấy Họ Tên	2	Hà Vĩnh	Phát	<input type="checkbox"/>
Lấy Họ Tên	3	Trần Tuyết	Oanh	<input checked="" type="checkbox"/>
Lấy Họ Tên	4	Nguyễn Kim	Ngọc	<input checked="" type="checkbox"/>
Lấy Họ Tên	5	Trương Duy	Hùng	<input type="checkbox"/>
Lấy Họ Tên	6	Lương Bá	Thắng	<input type="checkbox"/>
Lấy Họ Tên	7	Lâm Sơn	Hoàng	<input type="checkbox"/>
Lấy Họ Tên	8	Nguyễn Minh	Hồng	<input type="checkbox"/>
Lấy Họ Tên	9	Vương Ngọc	Lan	<input checked="" type="checkbox"/>
	!!!			

Hình 4.20: Hình ảnh sau khi thêm button column

Trong sự kiện `CellClick` của `DataGridView` ta có code lấy thông tin họ tên của dòng được nhấn như sau:

Ví dụ:

```
switch (e.ColumnIndex)
    case 0:
        MessageBox.Show(dataGridView1.CurrentRow.Cells["Ho"].Value.ToString() +
            dataGridView1.CurrentRow.Cells["Ten"].Value.ToString());
        Break;
    }
```

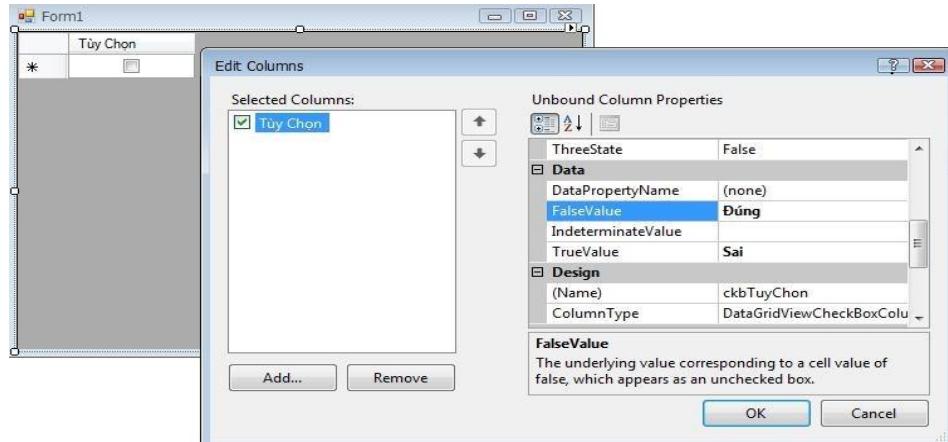
Đến lúc này chạy chương trình nhấn vào nút nhấn “Lấy Họ Tên” sẽ hiện thông báo họ tên của nhân viên dòng tương ứng.



Hình 4.21: Kết quả nhấn button.

4.4.3.1. Thêm cột điều khiển CheckBox vào DataGridView

Thông thường nếu giá trị một cột là kiểu Boolean (đúng/sai) thì DataGridView tự chuyển thành CheckBox khi hiển thị. Tuy nhiên chúng ta có thể thiết lập các cột hiển thị kiểu CheckBox phục vụ cho các ý tưởng riêng của chúng ta.



Cách thức thực hiện hoàn toàn tương tự như trên, chúng ta phải đánh dấu (gán giá trị) cho thuộc tính TrueValue, FalseValue để khi lập trình xử lý theo các giá trị đúng sai đó.

Câu hỏi trên lớp

1. DataView trong ADO.NET một đối tượng tương tự View-một câu lệnh truy vấn trong CSDL. Trình bày hiểu biết về đối tượng này.
2. Cách tạo và duyệt dữ liệu trong DataView.
3. Với tb là một đối tượng DataTable, nhận xét gì về hai câu lệnh sau:

`DataGridView1.DataSource = tb` và

`DataGridView1.DataSource = tb.DefaultView`

4. Để lọc dữ liệu trong DataView ta dùng thuộc tính nào?
5. Để sắp xếp dữ liệu trong DataView dùng thuộc tính nào?
6. Khi hiển thị DataView trong khung lưới nếu không cho phép người dùng thêm mới, hiệu chỉnh, xóa chúng ta có thể dùng những thuộc tính nào của DataView.
7. Thuộc tính nào giúp lấy ra mẫu tin hiện hành trên khung lưới DataGridView?
8. Thuộc tính nào giúp lấy ra giá trị ô hiện hành trên khung lưới DataGridView?

9. Khung lưới DataGridView có thể thêm được các Control như Button, TextBox, CheckBox,...không?
10. Thêm các control vào DataGridView chỉ có thể thực hiện trong môi trường thiết kế, ý kiến trên đúng hay sai?

Bài tập tại lớp

1. Viết chương trình thêm một TextBoxColumn vào lưới DataGridView và tham chiếu đến một columns bất kỳ trong bảng được chọn load dữ liệu lên lưới.
2. Viết chương trình load bảng hóa đơn lên lưới, thêm một ComboboxColumn vào lưới DataGridView và hiển thị tên của nhân viên tương ứng với mã nhân viên đó trong bảng hóa đơn.
3. Viết chương trình load bảng hóa đơn lên lưới, thêm một ButtonColumn vào lưới DataGridView và thực hiện một chức năng bất kỳ cho ButtonColumn được thêm.
4. Viết chương trình load bảng hóa đơn lên lưới, thêm một CheckBoxColumn vào lưới DataGridView và thực hiện một chức năng bất kỳ cho CheckBoxColumn được thêm.

Bài tập về nhà

1. Viết chương trình thêm một TextBoxColumn vào lưới DataGridView và tham chiếu đến một columns bất kỳ trong bảng khách hàng trong CSDL HoaDon.
2. Viết chương trình load bảng hóa đơn lên lưới, thêm một ComboboxColumn vào lưới DataGridView và hiển thị tên của khách hàng tương ứng với mã khách hàng đó trong bảng hóa đơn.
3. Viết chương trình load bảng khách hàng lên lưới, thêm một ButtonColumn vào lưới DataGridView và thực hiện một chức năng bất kỳ cho ButtonColumn được thêm.
4. Viết chương trình load bảng khách hàng lên lưới, thêm một CheckBoxColumn vào lưới DataGridView và thực hiện một chức năng bất kỳ cho CheckBoxColumn được thêm.

CHƯƠNG 5. THIẾT KẾ BÁO BIỂU

Sau khi học xong bài này sinh viên nắm được:

- ✓ Các bước để tạo một Report chuẩn của Visaul Studio 2010
- ✓ Vận dụng để thiết kế các mẫu Report khác nhau phục vụ cho việc kết suất dữ liệu.

Trọng tâm bài họa

- ✓ Cách tạo một Report chuẩn
- ✓ Các bước tiến hành thiết kế.
- ✓ Đỗ dữ liệu lên Report
- ✓ Hiển thị và in Report

5.1. SỬ DÙNG REPORT CHUẨN CỦA VISUAL STUDIO 2010

5.1.1. Quy trình tạo Report.

Gồm 3 phần chính

5.1.1.1. Phần 1: Cơ sở dữ liệu

- ✓ Viết Procedure dùng để hiển thị dữ liệu cho Report
- ✓ Trong procedure viết luôn phần hiển thị số thứ tự
- ✓ Thực thi thử procedure trong SQL Server

5.1.1.2. Phần 2: Thiết kế cấu trúc Report theo yêu cầu để bài

- ✓ Không dùng Crystal Report (hạng thứ ba)
- ✓ Sử dụng Report chuẩn của Visual Studio 20110
- ✓ Đuôi tập tin Report là .rdlc

5.1.1.3. Phần 3: Thiết kế 1 form để hiển thị Report

- ✓ Thiết kế 01 khung nhìn để hiển thị report ở phần 2
- ✓ Viết code lấy dữ liệu từ procedure vào DataTable. Sau đó, đưa dữ liệu từ DataTable vào trong Report
- ✓ Gọi hàm đúng sự kiện cần
 - Trong Form_Load
 - Trong sự kiện chọn ComboBox
 - Trong sự kiện Click của Button

Chú ý

- ✓ Phần 1 & Phần 2 độc lập với nhau

- ✓ Phần 3 cần có 02 phần
 - Phần lấy dữ liệu từ Procedure vào DataTable => Liên quan đến procedure ở phần 1
 - Phần hiển thị Report vào khung nhìn => Liên quan đến report (.rdlc) ở phần 2
- ✓ Phần 2: Phần trọng tâm thiết kế Report
 - Tính thẩm mỹ & tính chính xác

5.1.2. Ví dụ minh họa:

Thiết kế 1 Report theo yêu cầu của đề bài

DANH SÁCH SẢN PHẨM ĐANG BÁN

STT	Mã SP	Tên SP	Đơn vị tính	Đơn giá
1	12 dfd	dfddd		5656
2	11 l...lkiuku	.l.I.I.I		670
3	8 Cá hộp	Thùng		62.5
4	7 Táo	Kg		5
5	6 Nem	Kg		23.5
6	5 Bánh mì	Cái		8
7	4 Bơ	Kg		38
8	3 Bánh kem	Cái		10
9	2 Gia vị	Thùng		40
10	1 Rượu 123	Chai		230.5

Số sản phẩm 10

5.1.2.1. Quy trình thực hiện

Phần 1: Viết 1 procedure dùng để hiển thị thông tin của sản phẩm.

Chú ý: Trong Procedure bao gồm luôn field STT

Code phần Store procedure

```
Create procedure sp_rptSanPhamSelect
@MaSP as int
as
if @MaSP=0
select rOW_NUMBER() OVER (order by masp desc) AS
STT,* from SanPham
else
select rOW_NUMBER() OVER (order by masp desc) AS
STT,* from SanPham
where MaSP=@masp
```

Thực thi thủ tục

sp_rptSanPhamSelect 0

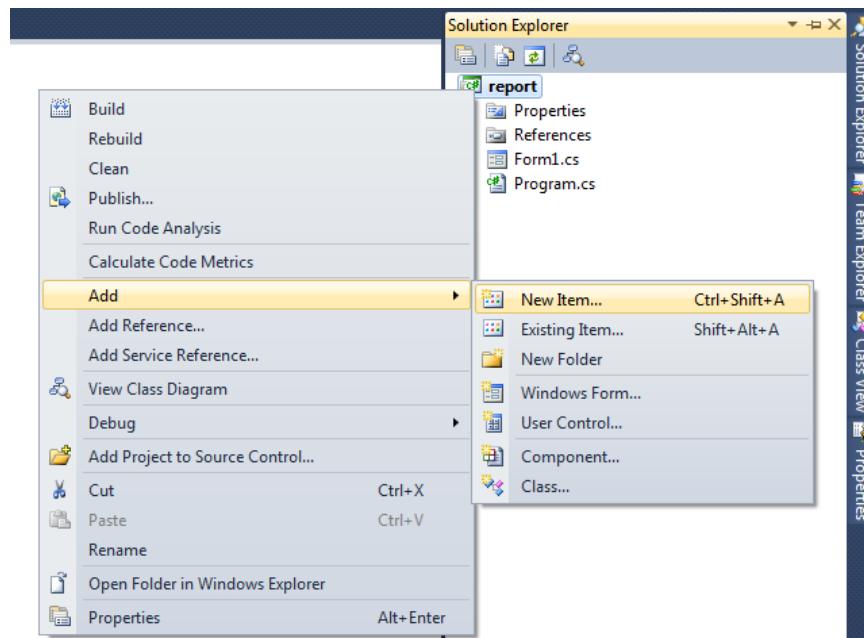
Kết quả :

STT	MaSP	TenSP	DonViTinh	DonGia
1	1	12	dfd	5656
2	2	11	I.Ilkiiuku	670
3	3	8	Cá hộp	62.5
4	4	7	Táo	Kg
5	5	6	Nem	Kg
6	6	5	Bánh kem	Click to select all grid cells
7	7	4	Bd	Kg
8	8	3	Bánh kem	Cái
9	9	2	Gia vị	Thùng
10	10	1	Rượu 123	Chai
				230.5

5.1.2.2. Phần 2: Thiết kế 1 Report

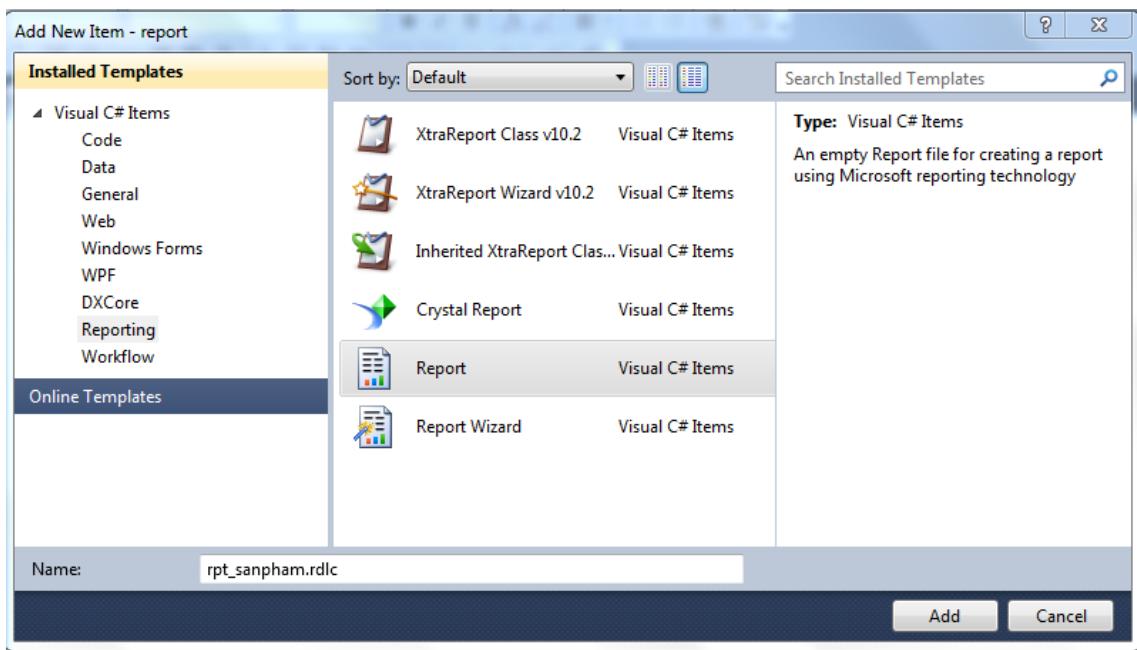
Tạo mới 1 report

Bước 1: Nhấn chuột phải vào dự án và chọn Add -> New Item

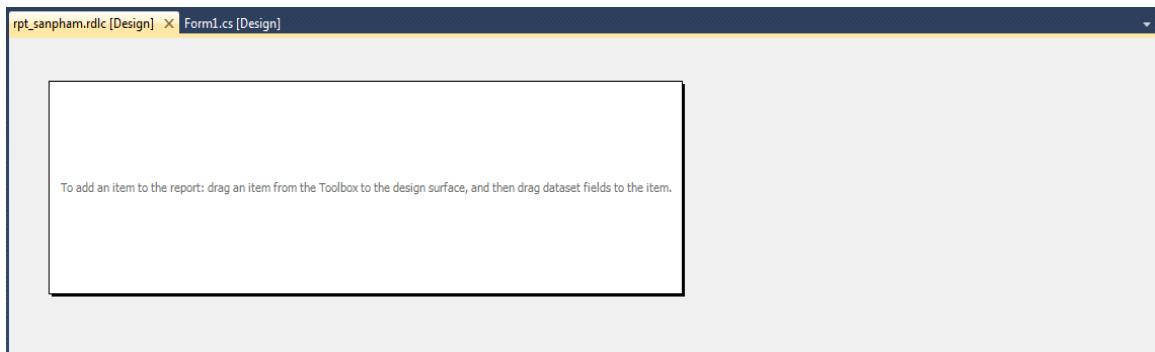


Bước 2: Chọn đối tượng Report và đặt tên

- ✓ Đặt tên theo nguyên tắc rpt_Tên đối tượng (Ví dụ: rpt_SanPham)
- ✓ Đuôi của tập tin Report là *.rdlc
- ✓ Chọn mục Reporting (bên menu trái) -> Chọn đối tượng Report (bên menu phải) và tiến hành đặt tên như hình vẽ



Bước 3: Một giao diện Report xuất hiện để thiết kế



Tạo cấu trúc cho Report

Dùng DataSet chứa 1 DataTable

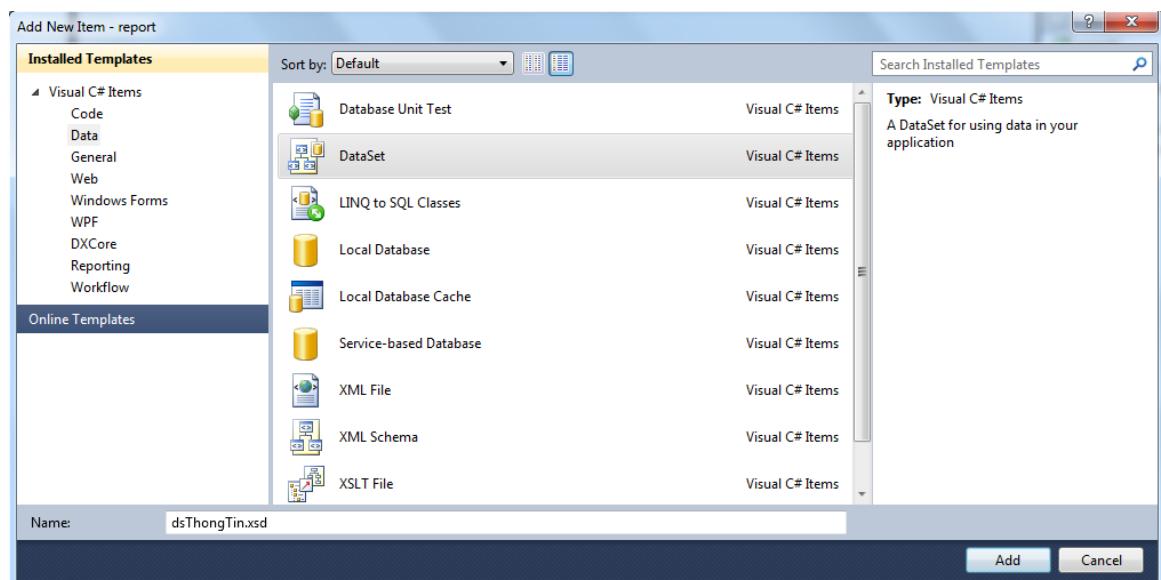
- ✓ Cấu trúc của DataTable là cấu trúc của Report
 - Số lượng Field trong Report
 - Kiểu dữ liệu của từng Field

Trong ví dụ: thông tin trong Report rpt_SanPham gồm các field sau đây:

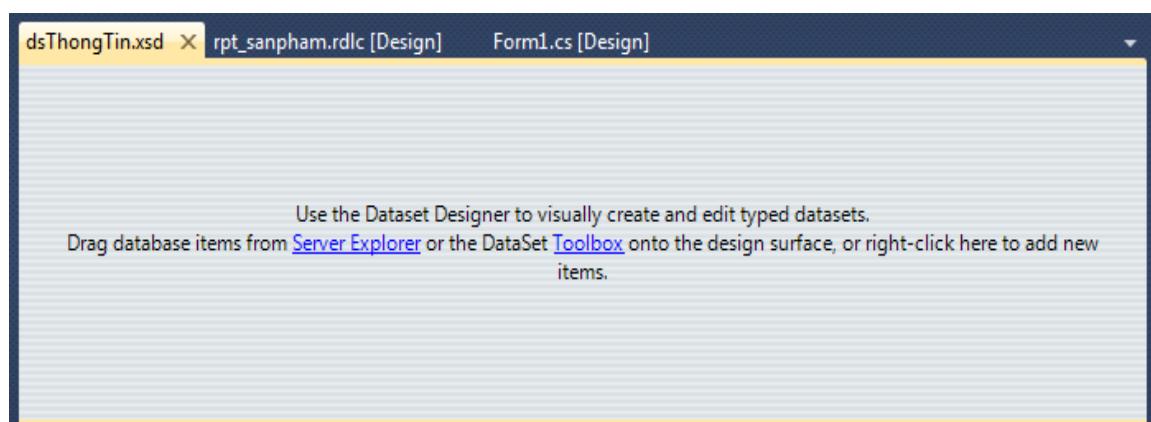
Tên field	Kiểu dữ liệu
STT	Integer
MaSP	Integer
TenSP	String
DonViTinh	String
DonGia	Double

Bước 1: Tạo 1 DataSet mới

- ✓ Nhấn chuột phải lên dự án
- ✓ Chọn Add ->New Item
- ✓ Chọn Data (bên phải) -> DataSet bên trái
- ✓ Đặt tên cho Dataset (đuôi là *.xsd). Trong ví dụ đặt tên là dsThongTin.xsd

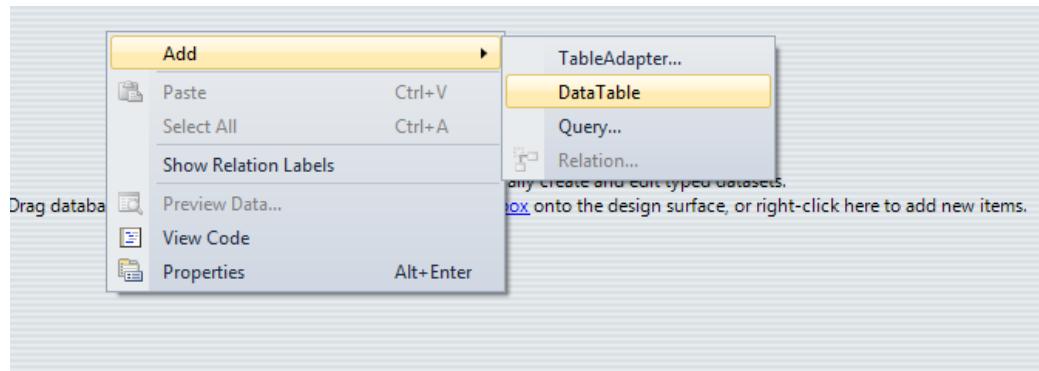


Bước 2: Một giao diện DataSet xuất hiện



Bước 3: Tạo 1 DataTable trong DataSet chứa cấu trúc của Report

- Bấm chuột phải vào vùng trống trong DataSet
- Chọn mục Add -> DataTable

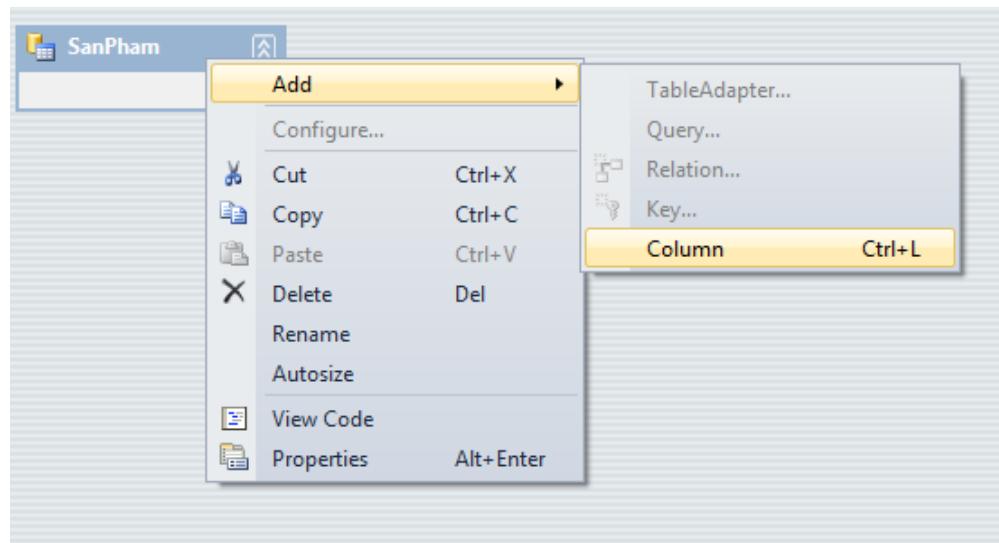


Bước 4: Đặt tên cho DataTable là SanPham



Bước 5: Khai báo các field cho DataTable SanPham

- Bấm chuột phải lên DataTable SanPham
- Chọn mục Add -> Column



Bước 6: Hoàn thành các field trong DataTable SanPham

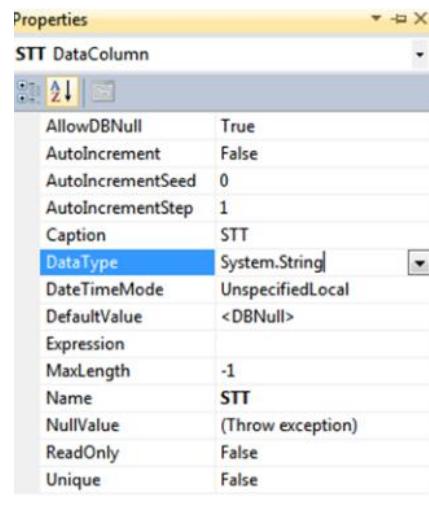
STT
MaSP
TenSP
DonViTinh
DonGia

Chú ý:

Tên các field nên trùng với tên các cột trong Procedure để dễ thao tác

Bước 7: Hiệu chỉnh lại kiểu dữ liệu cho các field

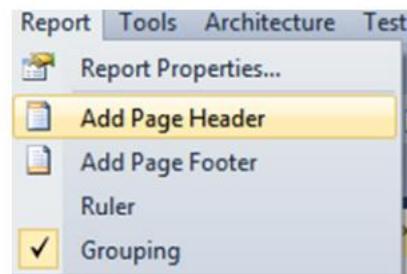
- ✓ Click chọn field cần hiệu chỉnh
- ✓ Vào Properties của field, hiệu chỉnh trong mục DataType



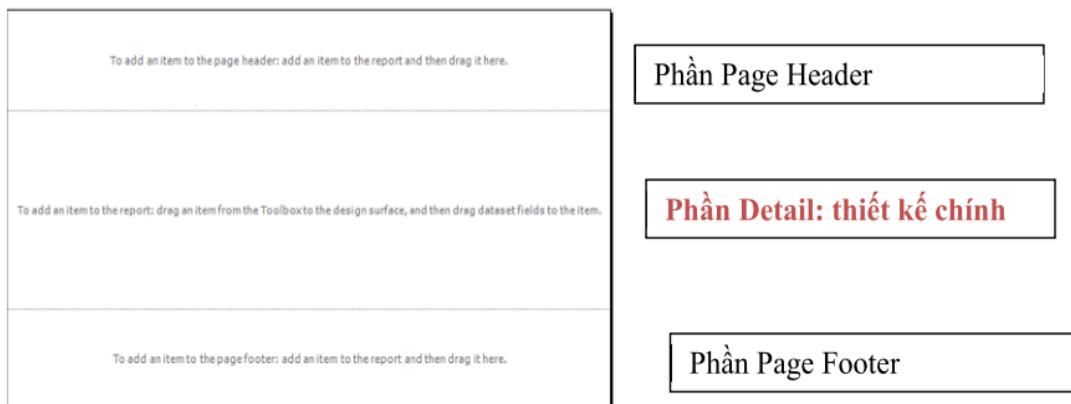
Thiết kế cấu trúc cho Report

Bước 1: Hiển thị PageHeader & PageFooter cho Report

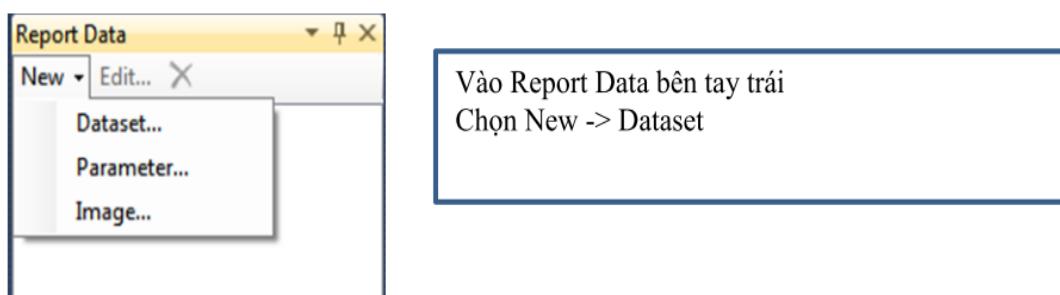
- ✓ Vào menu Report
- ✓ Check chọn 2 mục Add Page Header và Add PageFooter



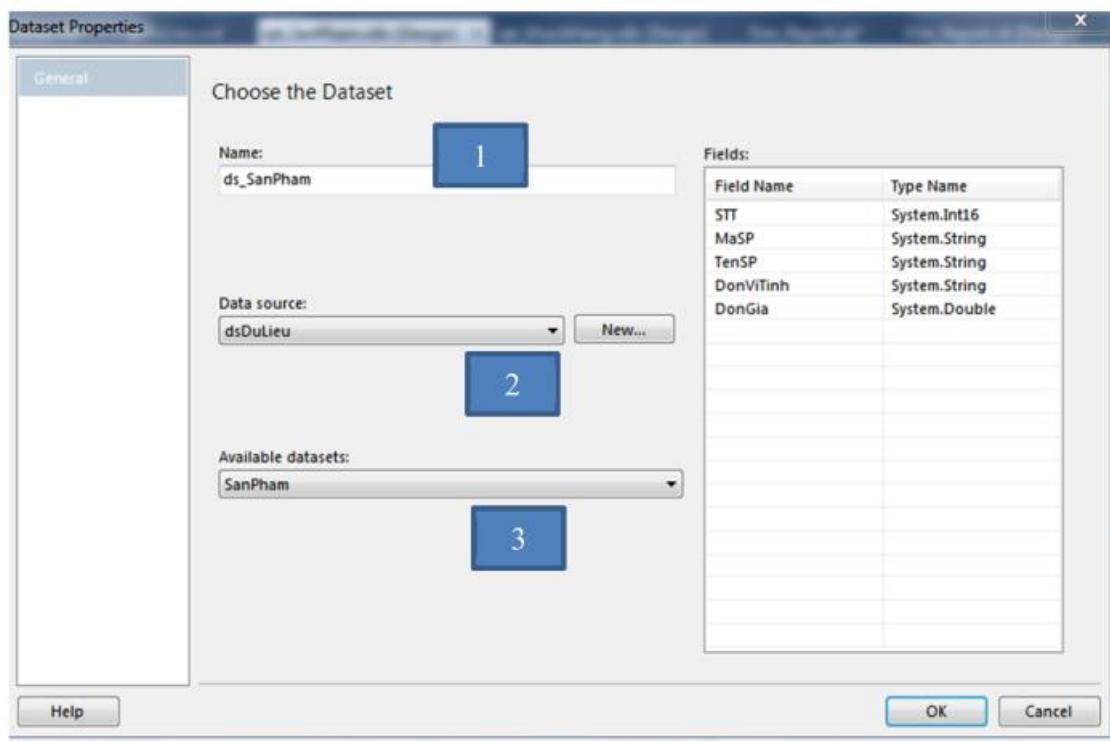
- ✓ Giao diện thiết kế Report trở thành



Bước 2: Lấy cấu trúc cho report dựa vào DataTable đã xây dựng trong DataSet



Bước 3: Cấu hình cấu trúc của Report



Ý nghĩa các mục

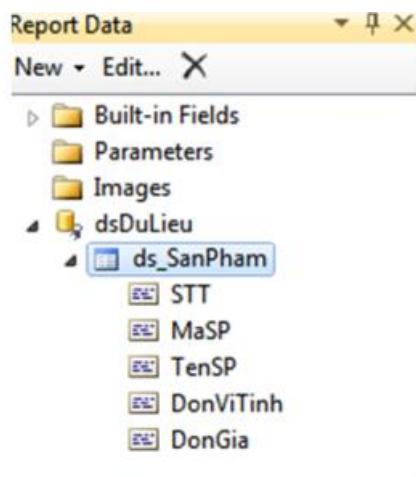
1. Name: Tên dataset của Report. Rất quan trọng dùng để lấy dữ liệu.

Lưu ý đặt tên cho dễ nhớ

2. DataSource: Chọn Dataset chứa DataTable, dùng để tạo cấu trúc cho Report

3. Chọn DataTable dùng để tạo cấu trúc

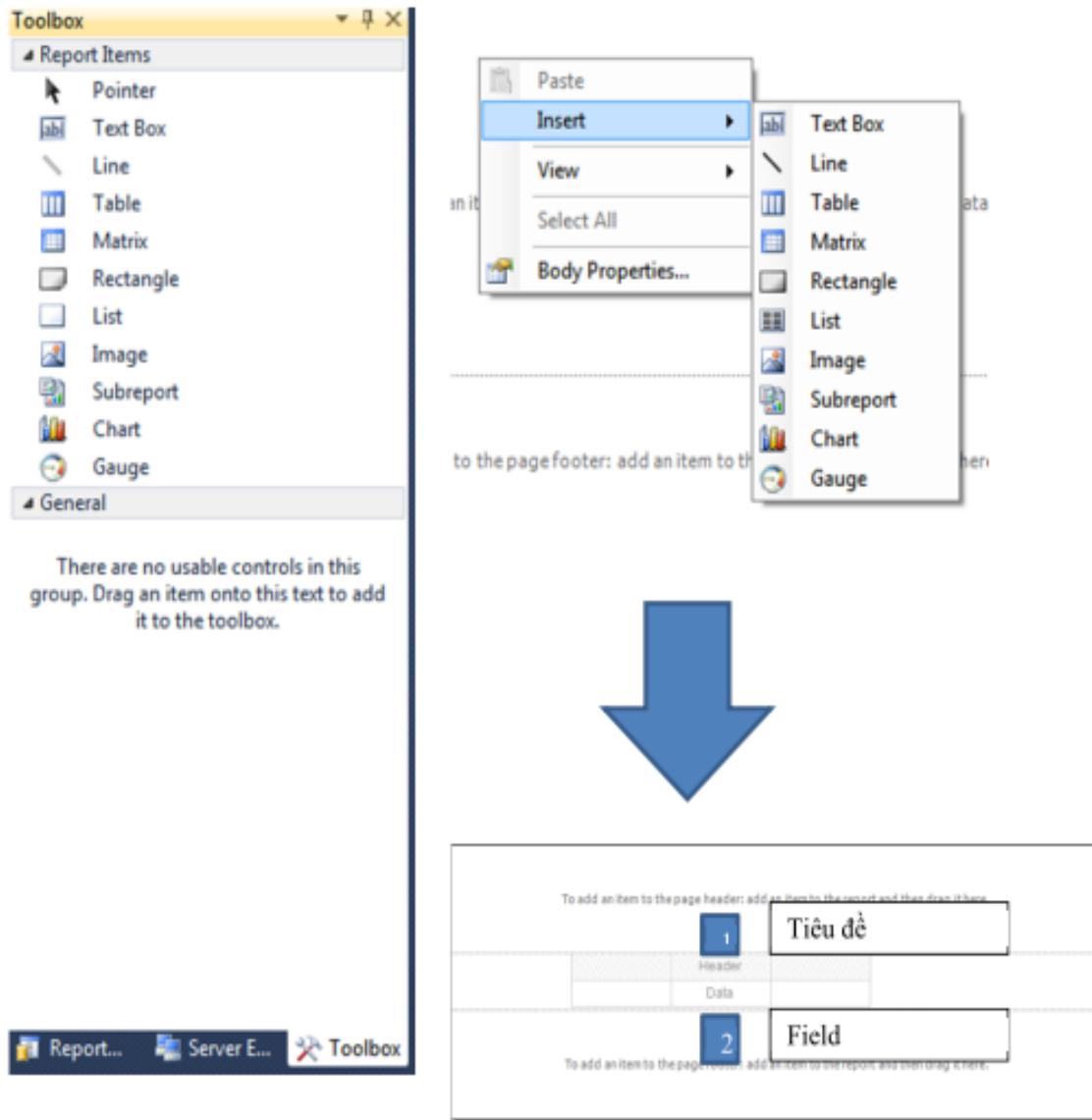
Sau khi nhấn OK, kết quả đạt được



Bước 3: Tạo bảng dùng để chứa cấu trúc report

Dùng hộp thoại Toolbox

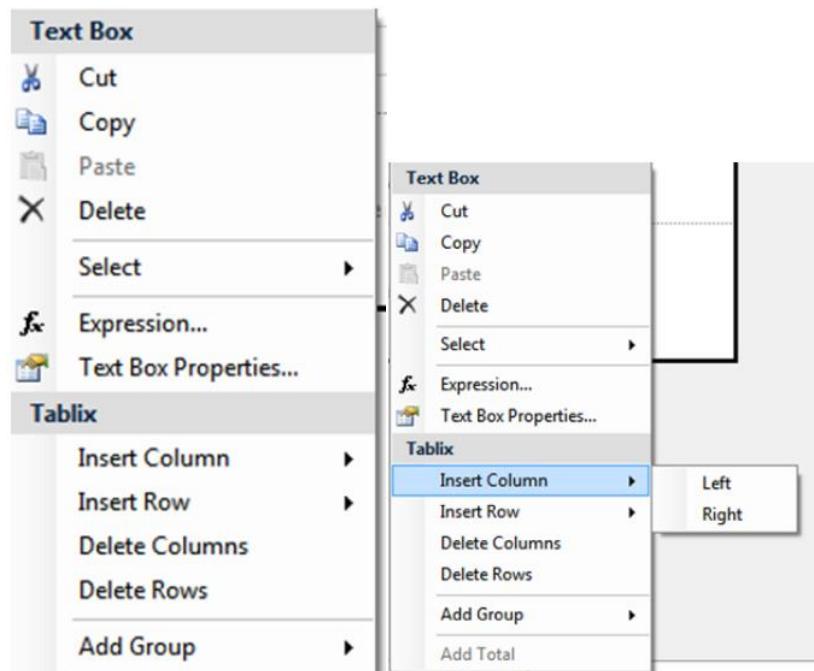
Nhấn chuột phải lên vùng Detail của Report và chọn Insert



Bước 4: Kéo các field từ Dataset của Report sang Table vừa tạo. Lưu ý kéo field vào vùng Data của Table

STT	Ma SP	Ten SP
[STT]	[MaSP]	[TenSP]
1	Header	Tiêu đề
2	Data	Field

Lưu ý: Thêm dòng, thêm cột thì nhán chuột phải vào Table và chọn mục tương ứng như hình vẽ

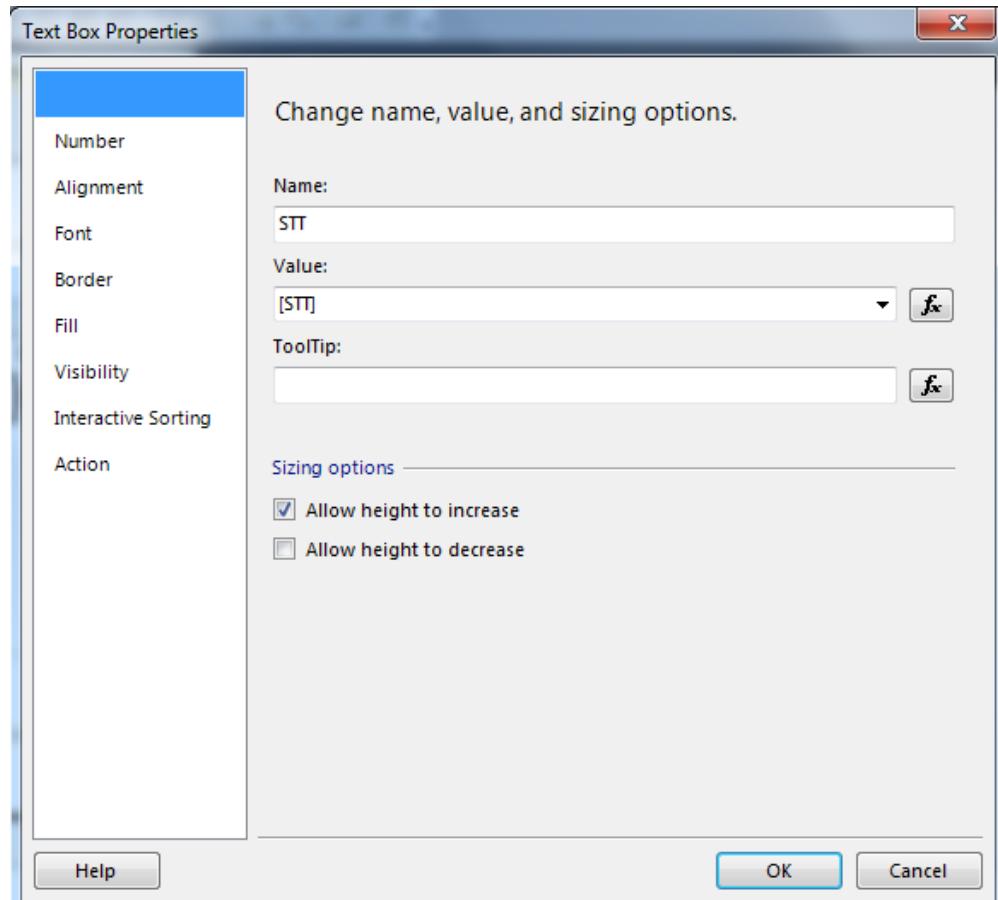


Câu hình cho các ô.

Chọn ô cần hiệu chỉnh

Chọn TextBox Properties

Hộp thoại câu hình cho text box



Bước 5: Tổng hợp dữ liệu

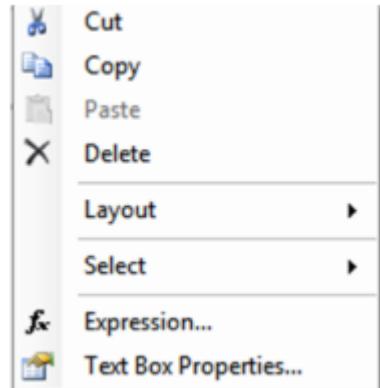
DANH SÁCH SẢN PHẨM ĐANG BÁN				
STT	Mã SP	Tên SP	Đơn vị tính	Đơn giá
[STT]	[MaSP]	[TenSP]	[DonViTinh]	[DonGia]
Số sản phẩm				
1		2		

Mục 1 và 2 nằm ở Page Footer

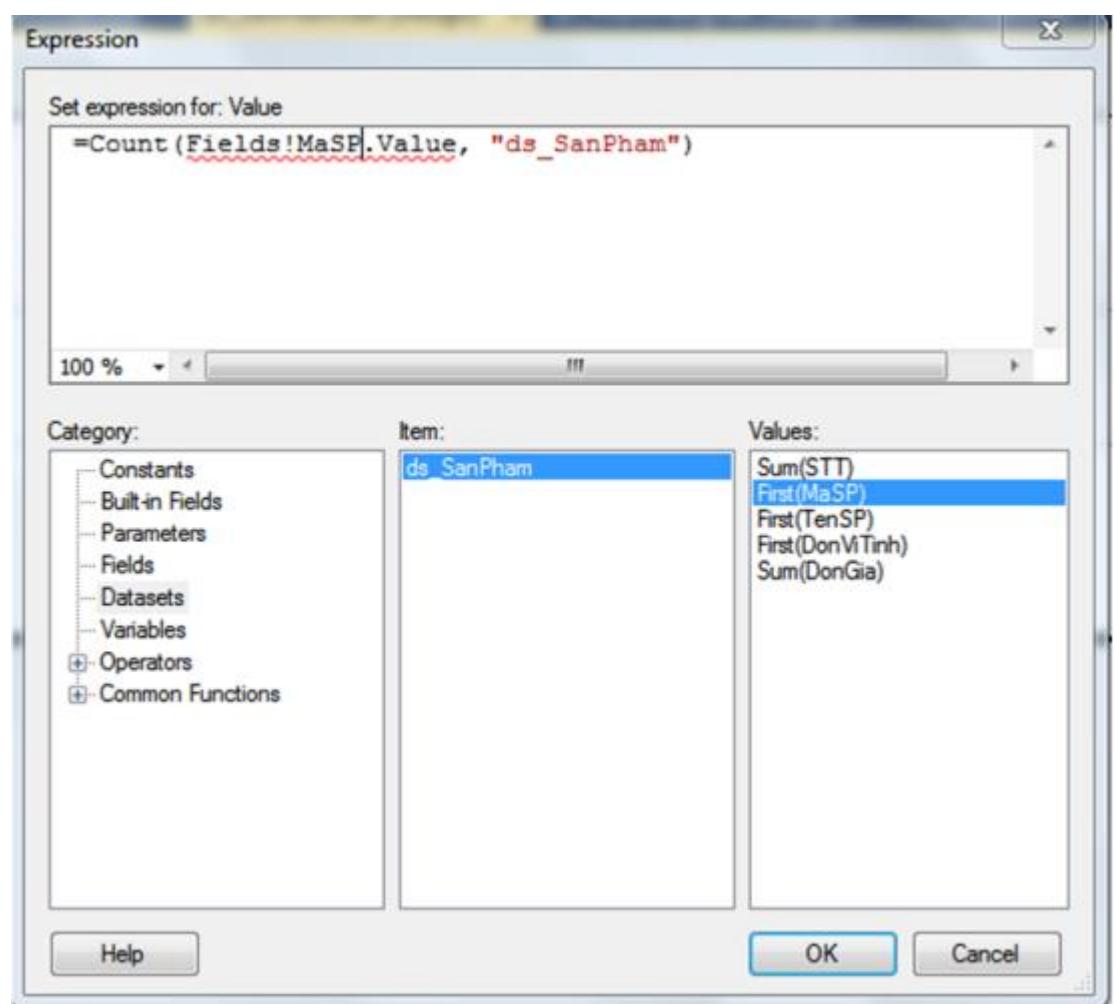
1.Text box: nhập chữ

2. Text box: nhập công thức tính tổng

Nhập chuột phải lên textbox, chọn mục Expression



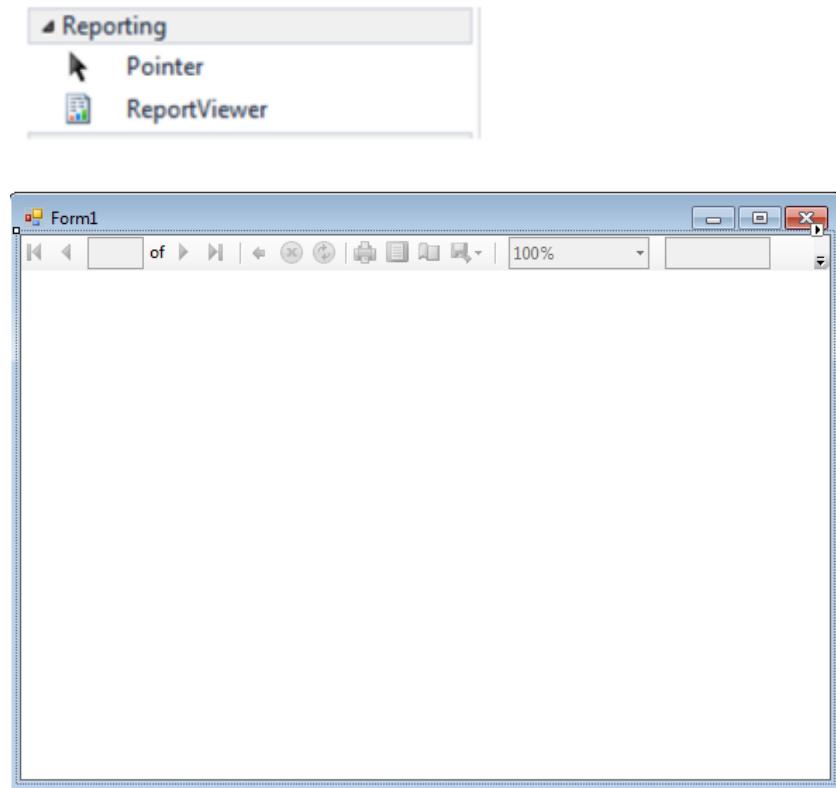
Chọn mục DataSet -> Chọn DataTable -> Chọn field và gõ hàm phù hợp



5.1.2.3. Phản 3: Tạo form để hiển thị Report

Bước 1: Tạo 1 form mới

Bước 2: Chọn Control Report Viewer kéo vào form Đặt tên Report Viewer là rptViewer



Bước 3: Các đoạn code cơ bản

Khai báo namespace

```
using System.Windows.Forms;
using Microsoft.Reporting.WinForms;
```

Hàm lấy dữ liệu từ Procedure

```
private DataTable Laydulieusanpham(int masp)
{
    cmd = new SqlCommand("sp_rptSanPhamSelect", cnn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@masp", SqlDbType.Int);
    da = new SqlDataAdapter(cmd);
    dt = new DataTable();
    da.Fill(dt);
```

```

    return dt;
}

```

Hàm đưa dữ liệu vào Report

```

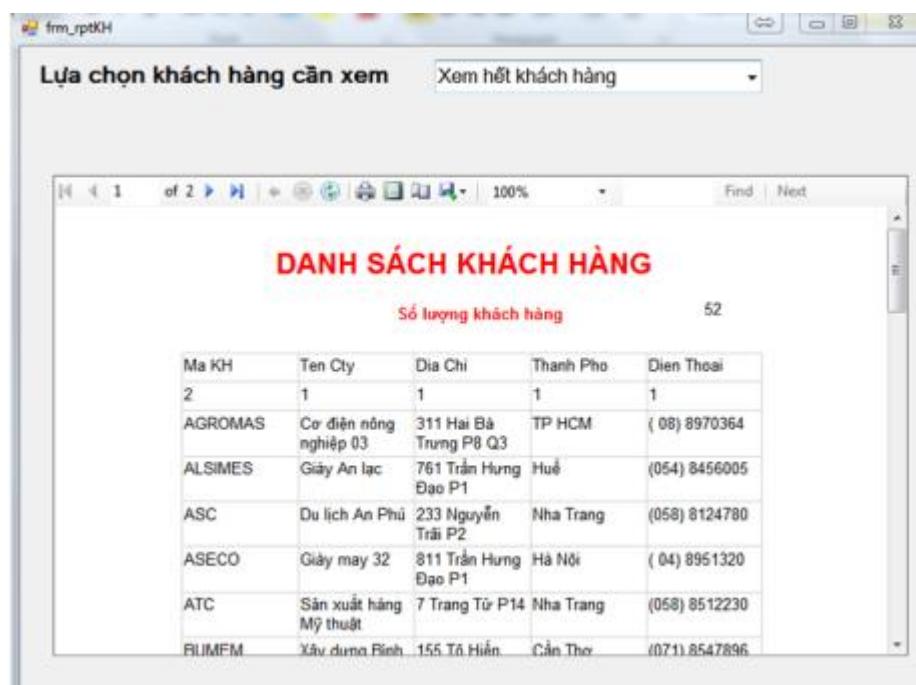
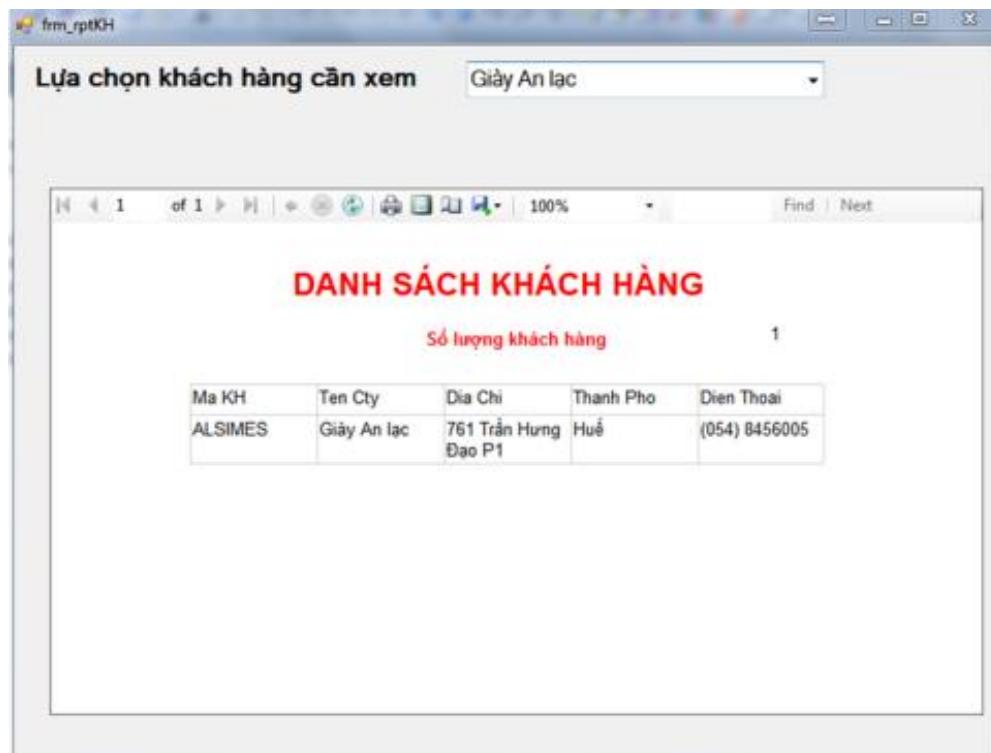
private void loaddulieuvaoreport()
{
    DataTable table = new DataTable();
    table.Clear();
    table = Laydulieusanford();
    //reset lại khung hiển thị report
    rptViewer.Reset();
    //gán tên report cần hiển thị trong khung nhìn viewer
    rptViewer.LocalReport.ReportPath=
        @"D:\TaiLieuHoc\LAP_TRINH_C#\project\laptrinhc2\report\report
        \rpt_sanpham.rdlc";
    //làm sạch khung nhìn
    rptViewer.LocalReport.DataSources.Clear();
    ReportDataSource newDataSource = new
    ReportDataSource("ds_sanpham",table);
    rptViewer.LocalReport.DataSources.Add(newDataSource);
    //lam tuoi report
    // rptViewer.LocalReport.DataSources.Add(new
    ReportDataSource("ds_sanpham",table));
    rptViewer.RefreshReport();
}

```

5.2. BÀI TẬP ỦNG DỤNG CHO PHẦN REPORT.

5.2.1. Bài tập 1: Thiết kế report có sử dụng ComboBox

Chọn 1 khách hàng trong combox thì hiển thị thông tin của khách hàng đó. Còn chọn xem hết thì sẽ hiển thị tất cả thông tin của khách hàng.



5.2.2. Bài tập 2:

Thiết kế report theo mẫu sau:

Lựa chọn hóa đơn cần xem 10144

THÔNG TIN CHI TIẾT CỦA HÓA ĐƠN SỐ 10144

TỔNG TIỀN BÁN: **8637.5**

STT	Mã HD	Tên SP	Soluong	Đơn Giá	Thanh Tiền
1	10144	Nem	20	23.5	470
2	10144	Bánh kem	10	10	100
3	10144	Rượu 123	35	230.5	8067.5

5.2.3. Bài tập 3:

Tạo report theo mẫu sau:

Quản lý hóa đơn theo khách hàng

Chọn khách hàng: Sản xuất hàng Mỹ thuật

Tổng số hóa đơn 4

Delete	Mã HD	Mã KH	Mã NV	Ngày lập HD	Ngày nhận hàng
<input checked="" type="checkbox"/>	10175	ATC	4	26/02/1992	25/03/1992
<input type="checkbox"/>	10408	ATC	4	02/12/1992	30/12/1992
<input type="checkbox"/>	10634	ATC	8	09/07/1993	06/08/1993
<input type="checkbox"/>	10763	ATC	3	27/10/1993	24/11/1993

Trên form tạo thêm 1 nút “Xem báo cáo”, khi nhấn vào sẽ mở report chứa các thông tin các hóa đơn của khách hàng được chọn trong Combo Box (Ảnh minh họa)

DANH SÁCH HÓA ĐƠN CỦA KHÁCH HÀNG
Cơ điện nông nghiệp 03

Số hóa đơn 24

STT	MaHD	MaKH	MaNV
1	10359	AGROMAS	5
2	10523	AGROMAS	7
3	10804	AGROMAS	6
4	10869	AGROMAS	5
5	123	AGROMAS	3
6	678	AGROMAS	3
7	asd11	AGROMAS	3
8	asd56	AGROMAS	3
9	dff	AGROMAS	3
10	fg546	AGROMAS	3

5.2.4. Bài tập 04:

Trên form tạo thêm 1 nút “Xem báo cáo”, khi nhấn vào sẽ mở report chứa các thông tin các hóa đơn của nhân viên được chọn trong Combo Box (Ảnh minh họa)

Quản lý hóa đơn theo nhân viên

Chọn nhân viên		Nguyễn Minh Hồng				Delete
Tổng số hóa đơn 23						
Delete	MaHD	MAKH	MaNV	NgayLapHD	NgayNhanHa	
<input checked="" type="checkbox"/>	10166	VTP	4	13/02/1992	12/03/1992	
<input type="checkbox"/>	10175	ATC	4	26/02/1992	25/03/1992	
<input type="checkbox"/>	10214	ASC	4	15/04/1992	13/05/1992	
<input type="checkbox"/>	10262	PECCO	4	15/06/1992	13/07/1992	
<input type="checkbox"/>	10287	REXCO	4	16/07/1992	13/08/1992	
<input type="checkbox"/>	10305	LIPEXIM	4	07/08/1992	04/09/1992	
<input type="checkbox"/>	10380	TAFACO	4	05/11/1992	03/12/1992	
<input type="checkbox"/>	10383	RUBIMEX	4	09/11/1992	07/12/1992	
<input type="checkbox"/>	10408	ATC	4	02/12/1992	30/12/1992	
<input type="checkbox"/>	10416	VAFACO	4	10/12/1992	07/01/1993	
<input type="checkbox"/>	10421	TAFACO	4	15/12/1992	26/01/1993	

STT	Ma HD	Ma KH	Ma NV
1	10857	SAMECO	4
2	10852	PECCO	4
3	10845	WASECO	4
4	10811	SECMC	4
5	10786	WACO	4
6	10756	INEXIM	4
7	10729	SECMC	4
8	10722	SCITEC	4
9	10635	TRANACO	4
10	10623	TRACODI	4
11	10534	COMEKO	4
12	10498	DHP	4
13	10421	TAFACO	4

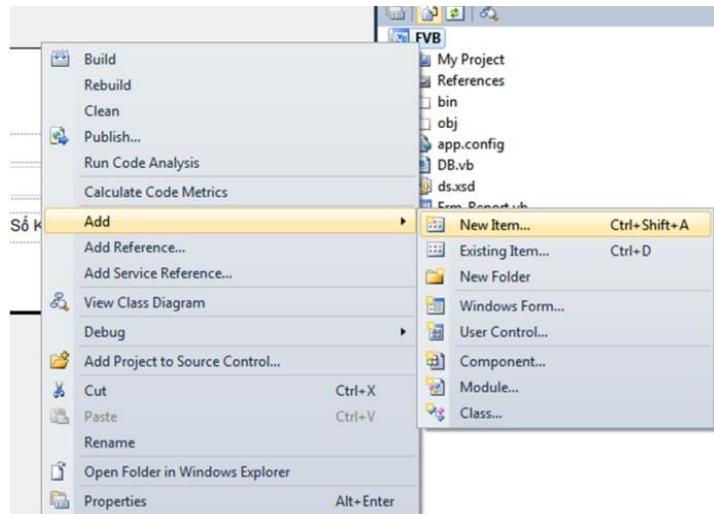
5.3. HƯỚNG DẪN GIẢI BÀI TẬP:

5.3.1. Hướng dẫn bài tập 1

Viết procedure lấy thông tin khách hàng:

```
Create procedure sp_rptKhachHangSelect
@MaKH as varchar(20)
as
-- Lấy dữ liệu cho ComboBox
if @MaKH=-1'
Select '0' as MaKH, N'Xem hết khách hàng' as TenCty
union
select MaKH, tencty
from khachhang
else
-- Lấy dữ liệu tất cả khách hàng
if @MaKH='0'
select *
from khachhang
else
-- Lấy dữ liệu khách hàng ứng với mã khách hàng truyền
vào
select *
from khachhang
where MaKH=@MaKH
```

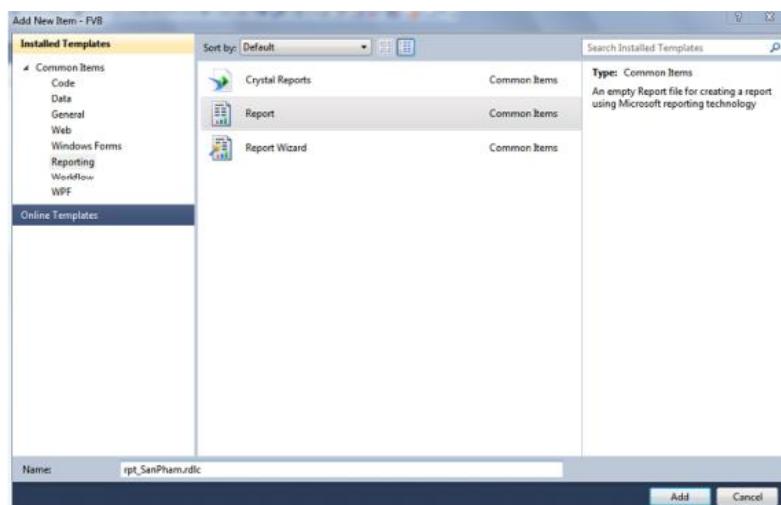
Thiết kế 1 Report

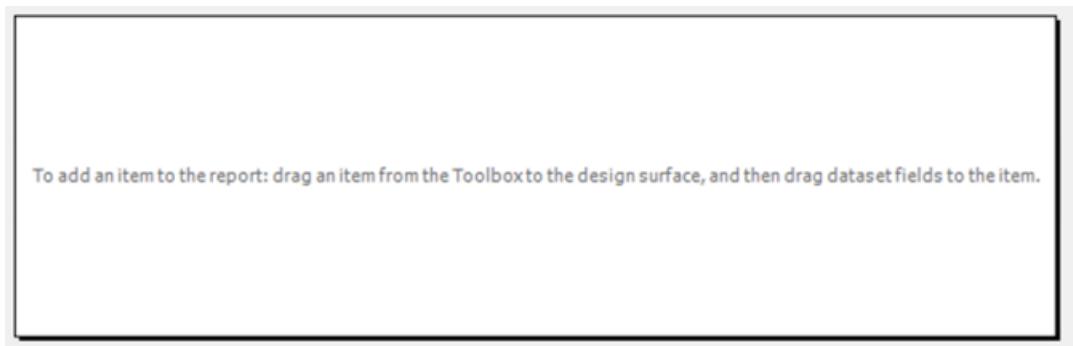


Bước 1: Nhấn chuột phải vào dự án và chọn Add -> New Item

Bước 2: Chọn đối tượng Report và đặt tên

- Đặt tên theo nguyên tắc rpt_Tên đối tượng (Ví dụ: rpt_KhachHang)
- Đuôi của tập tin Report là *.rdlc
- Chọn mục Reporting (bên menu trái) -> Chọn đối tượng Report (bên menu phải) và tiến hành đặt tên như hình vẽ





Bước 3: Một giao diện Report xuất hiện dùng để thiết kế

Tạo cấu trúc cho Report

- ✓ Dùng DataSet chứa 1 DataTable; Cấu trúc của DataTable là cấu trúc của Report
 - Số lượng Field trong Report
 - Kiểu dữ liệu của từng Field

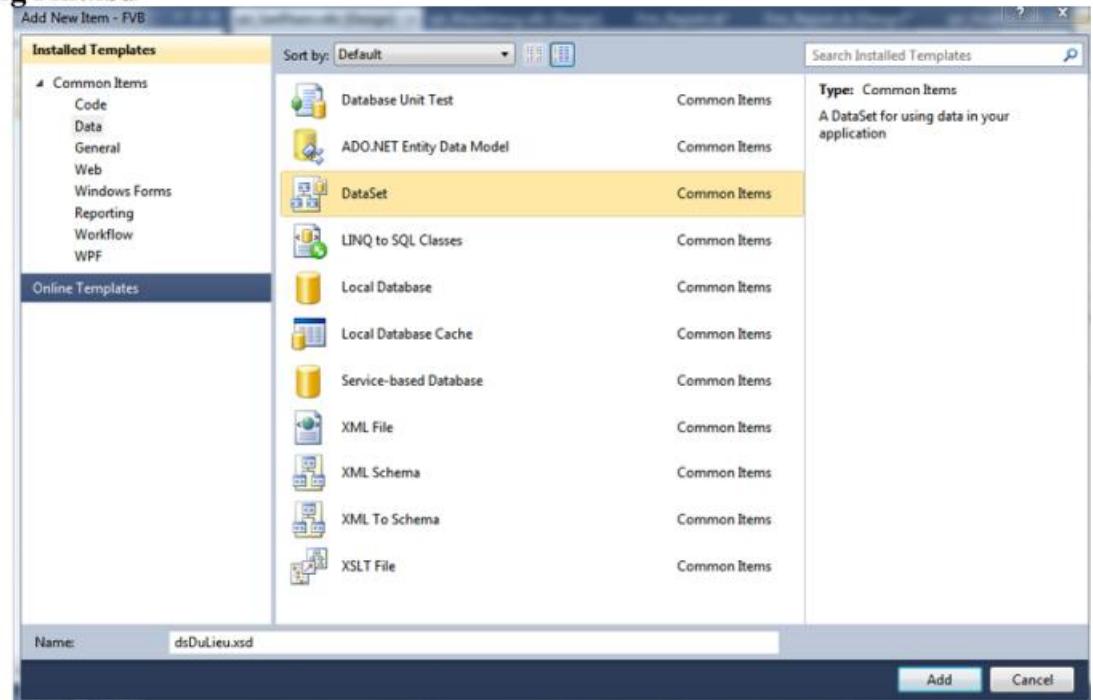
Tên field	Kiểu dữ liệu
MaKH	String
TenCTy	String
DiaChi	String
ThanhPho	String
DienThoai	String

Trong ví dụ: thông tin trong Report rpt_KhachHang gồm các field sau đây

Bước 1: Tạo 1 DataSet mới

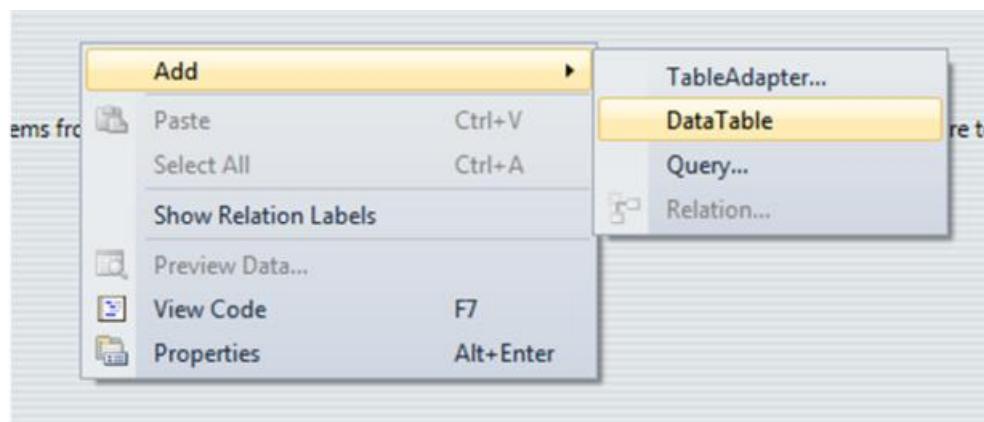
- Nhấn chuột phải lên dự án
- Chọn Add ->New Item
- Chọn Data (bên phải) -> DataSet bên trái
- Đặt tên cho Dataset (đuôi là *.xsd). Trong ví dụ đặt tên l

dsThongTin.xsd



Bước 3: Tao 1 DataTable trong DataSet chứa cấu trúc của Report

- Bấm chuột phải vào vùng trốngng trong DataSet
- Chọn mục Add -> DataTable

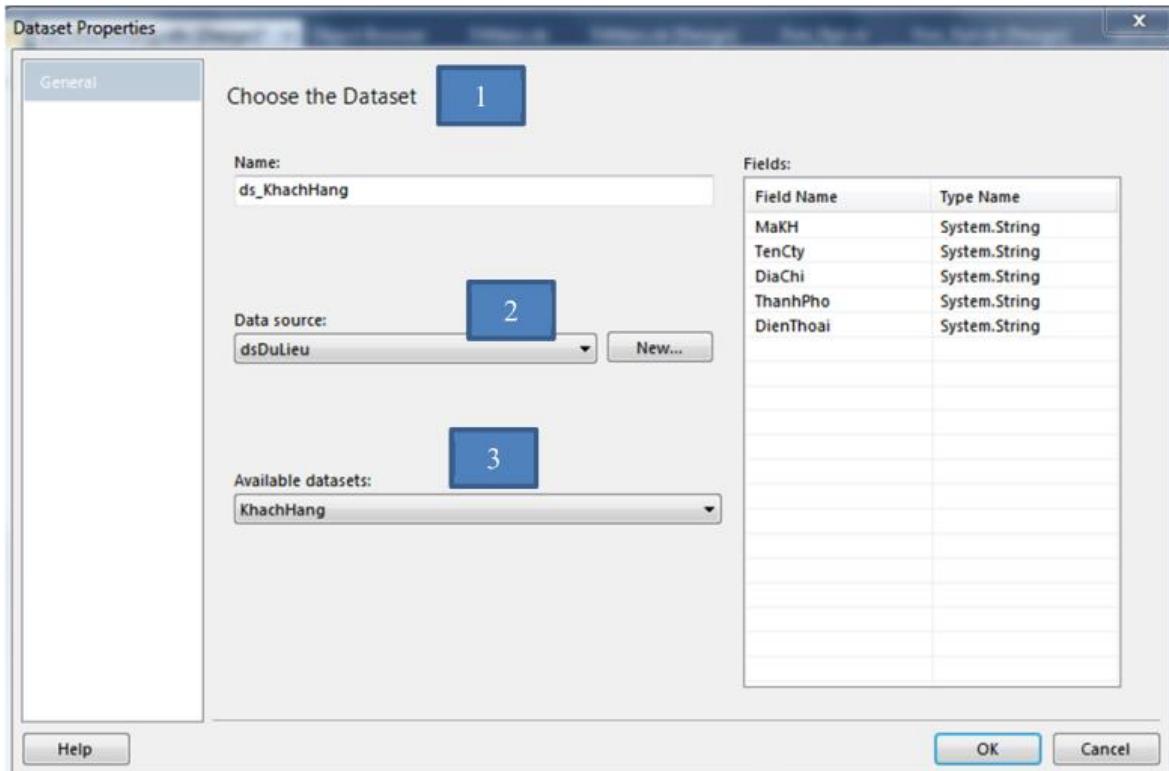


Thiết kế cấu trúc cho Report

Bước 1: Hiển thị PageHeader & PageFooter cho Report

Bước 2: Lấy cấu trúc cho report dựa vào DataTable đã xây dựng trong DataSet

Bước 3: Cấu hình cấu trúc của Report



Ý nghĩa các mục

1. Name: Tên dataset của Report. Rất quan trọng dùng để lấy dữ liệu. Lưu ý đặt tên cho dễ nhớ
2. DataSource: Chọn Dataset chứa DataTable, dùng để tạo cấu trúc cho Report
3. Chọn DataTable dùng để tạo cấu trúc

Sau khi nhấn OK, kết quả đạt được

Bước 4: Tạo bảng dùng để chứa cấu trúc report

Bước 5: Kéo các field từ Dataset của Report sang Table vừa tạo. Lưu ý kéo field vào vùng Data của Table.

Bước 6: Tổng hợp dữ liệu.

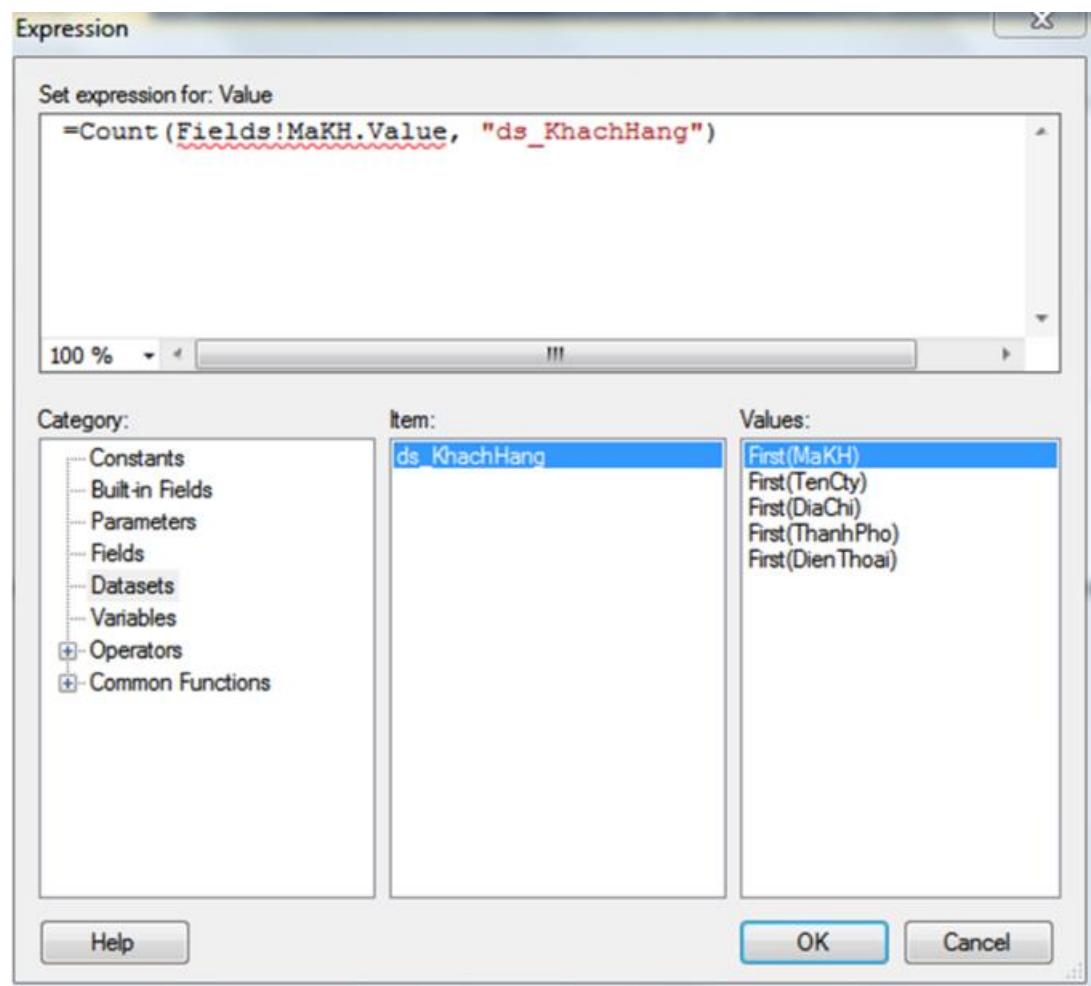


Mục 1 và 2 nằm ở Page Footer

1.Text box: nhập chữ

2. Text box: nhập công thức tính tổng

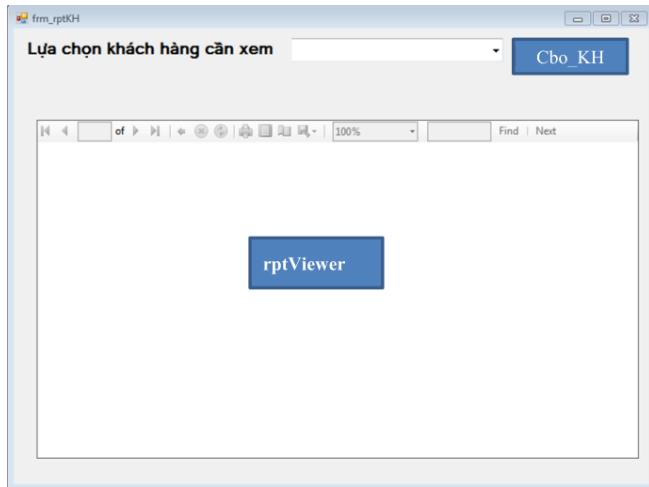
Nhập chuột phải lên textbox, chọn mục Expression



Phần 3: Tạo form để hiển thị Report

Bước 1: Tạo 1 form mới

Bước 2: Chọn Control Report Viewer kéo vào form Đặt tên Report Viewer là rptViewer.



Các đoạn code cơ bản:

Lấy dữ liệu từ Procedured

```
private DataTable Laydulieusanford(int makh)
{
    cmd = new SqlCommand("sp_rptkhachhangSelect", cnn);
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@makh", SqlDbType.Int).Value = makh;
    da = new SqlDataAdapter(cmd);
    dt = new DataTable();
    da.Fill(dt);
    return dt;
}
```

Đưa dữ liệu vào ComboBox

```
private void dudulieuvaocombox()
{
    DataTable dtdulieu = new DataTable();
    dtdulieu = Laydulieukhachhang("-1");
    comboBox1.DataSource = dtdulieu;
    comboBox1.ValueMember = "makh";
    comboBox1.DisplayMember = "tencty";
}
```

Lấy dữ liệu đưa lên report

```

private void loaddulieuvaoreport(string makh)
{
    DataTable table = new DataTable();
    table.Clear();
    table = laydulieukhachhang(makh);
    //reset lại khung hiển thị report
    reportViewer1.Reset();
    //gán tên report cần hiển thị trong khung nhìn
    viewer
    reportViewer1.LocalReport.ReportPath =
@"D:\TaiLieuHoc\LAP_TRINH_C#\project\laptrinhc2\report\report
\rpt_khachhang.rdlc";

    //làm sạch khung nhìn
    reportViewer1.LocalReport.DataSources.Clear();
    ReportDataSource newDataSource = new
ReportDataSource("ds_khachhang", table);

    reportViewer1.LocalReport.DataSources.Add(newDataSource);
    //lam tuoi report
    // rptViewer.LocalReport.DataSources.Add(new
ReportDataSource("ds_sanpham", table));
    reportViewer1.RefreshReport();
}

```

Gọi hàm trong sự kiện Load của form

```

private void Form2_Load(object sender, EventArgs e)
{
    cnn = new SqlConnection(connectionstring);
    dudulieuvaocombox();

    this.reportViewer1.RefreshReport();
    loaddulieuvaoreport(
comboBox1.SelectedValue.ToString());
}

```

Thực thi sự kiện SelectionChangeCommitted để view dữ liệu khi chọn
ComboBox

```

private void comboBox1_SelectionChangeCommitted(object
sender, EventArgs e)
{
    if (comboBox1.SelectedIndex >= 0)

        loaddulieuvaoreport(comboBox1.SelectedValue.ToString());
}

```

5.3.2. Bài tập 2: (tương tự bài tập 1)

Procedure lấy dữ liệu vào Combo Box

```
create procedure sp_rptHoaDonSelect
as
Select MaHD
from hoadon
```

Procedure lấy dữ liệu vào report

```
Create procedure sp_rptCTHDSelect
@MaHD as varchar(20)
as
select rOW_NUMBER() OVER (order by a.masp desc) AS
STT,a.MaHD,b.TenSP,a.SoLuong,b.DonGia,
a.SoLuong*b.DonGia as ThanhTien
from ChiTietHoaDon a, SanPham b
where a.Masp=b.Masp and a.MaHD=@MaHD
```

5.3.3. Hướng dẫn Bài tập 4:

Phần 1: Viết 1 procedure dùng để hiển thị thông tin của khách hàng.

- 1.1. Dựa vào Report để xác định số lượng Field cần tạo trong Procedure của Report
- 1.2. Theo đề bài, ta phải tạo procedure gồm các field sau đây:

STT,MAHD,MAKH,MANV => Trong Report

Ten => Tên nhân viên phía trên đầu (Oanh)

Code Procedure lấy dữ liệu cho Report

```
create procedure sp_rptHoaDonSelectByMaNV
@MaNV as int
as
Select rOW_NUMBER() OVER (order by a.mahd desc) AS
STT,a.MaHD,a.Makh,a.Manv,b.Ten
from HoaDon a, NhanVien b
where a.Manv=b.Manv
and b.Manv=@MaNV
```

STT	MaHD	MaKH	MaNV	Ten
1	10872	COFIDECK	5	Hùng
2	10869	AGROMAS	5	Hùng
3	10866	SAMECO	5	Hùng
4	10851	REXCO	5	Hùng
5	10841	HOTICO	5	Hùng
6	10823	MINEXIM	5	Hùng
7	10730	TDE	5	Hùng
8	10714	SCITEC	5	Hùng
9	10711	SCITEC	5	Hùng
10	10675	TRACODI	5	Hùng
11	10654	SAMECO	5	Hùng

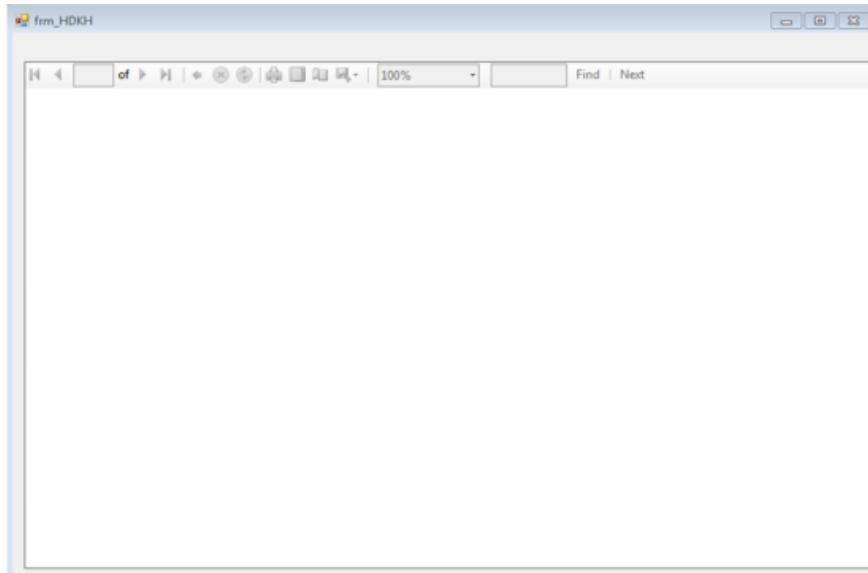
Phần 2: Thiết kế mẫu Report

The screenshot shows a Microsoft Report Designer interface. At the top, there are two main sections: "THÔNG TIN BÁN HÀNG CỦA NHÂN VIÊN" and "TỔNG HÓA ĐƠN". Each section contains a table with four columns: STT, Ma HD, Ma KH, and Ma NV. Below the tables, there is a note: "To add an item to the page footer: add an item to the report and then drag it here." On the right side of the interface, there are two blue buttons labeled "1" and "2".

1. Kéo field TEN từ Dataset của Report vào
2. Tạo mới 1 text box và đếm số lượng hóa đơn dùng hàm Count

Phản 3: Tao form hiển thi Report

Giao diện form cần tạo



Các đoạn code cơ bản

Bên form mở

```

private void dudulieuvaocombox()
{
    DataTable dtdulieu = new DataTable();
    dtdulieu = laynv();
    comboBox1.DataSource = dtdulieu;
    comboBox1.ValueMember = "manv";
    comboBox1.DisplayMember = "ten";
}
private DataTable laynv()
{
    cmd = new SqlCommand("select manv,Ten from
nhanvien", cnn);
    da = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();
    da.Fill(dt);
    return dt;
}

private void button2_Click(object sender, EventArgs e)
{
    Manv=Convert.ToInt32(comboBox1.SelectedValue);
    // manv khai báo là biến toàn cục
    view frmview = new view();
    frmview.ShowDialog();
}

Form view
public DataTable Laydulieuohoadon(int manv)

```

```

{
    DataTable dt = new DataTable();
    cmd = new
SqlCommand("sp_rptHoaDonSelectByMaNV", cnn);
    cmd.CommandType = CommandType.StoredProcedure;

    cmd.Parameters.Add("@manv",
SqlDbType.Int).Value = manv;
    da = new SqlDataAdapter(cmd);
    dt = new DataTable();
    da.Fill(dt);
    return dt;
}

private void loaddulieuvaoreport(int manv)
{
    DataTable table = new DataTable();
    table.Clear();
    table = Laydulieuohoadon(manv);
    //reset lại khung hiển thị report
    reportViewer1.Reset();
    //gán tên report cần hiển thị trong khung nhìn
    viewer
        reportViewer1.LocalReport.ReportPath =
@"D:\TaiLieuHoc\LAP_TRINH_C#\project\laptrinhc2\report\repo
rt\Report2.rdlc";

        //làm sạch khung nhìn
        reportViewer1.LocalReport.DataSources.Clear();
        ReportDataSource newDataSource = new
ReportDataSource("ds_hoadon", table);

reportViewer1.LocalReport.DataSources.Add(newDataSource);
        //lam tuoi report
        // rptViewer.LocalReport.DataSources.Add(new
ReportDataSource("ds_sanpham",table));
        reportViewer1.RefreshReport();
    }

private void view_Load(object sender, EventArgs e)
{
    cnn = new SqlConnection(connectionstring);
    loaddulieuvaoreport(Form3.manv);
    this.reportViewer1.RefreshReport();
}

```

CHƯƠNG 6. MÔ HÌNH 3 LỚP

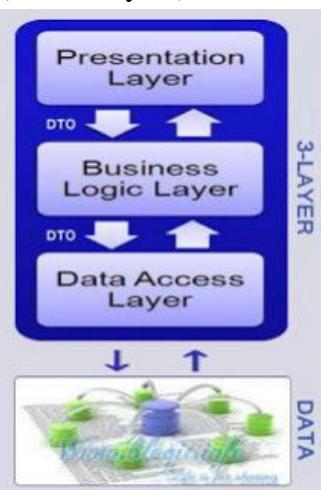
6.1. GIỚI THIỆU MÔ HÌNH 3 LỚP

Để dễ quản lý các thành phần của hệ thống, cũng như không bị ảnh hưởng bởi các thay đổi, người ta hay nhóm các thành phần có cùng chức năng lại với nhau và phân chia trách nhiệm cho từng nhóm để công việc không bị chồng chéo và ảnh hưởng lẫn nhau. Một trong những mô hình lập trình như vậy đó là Mô hình 3 lớp (Three Layers).

6.2. CÁC THÀNH PHẦN TRONG 3-LAYER

Mô hình 3 lớp được cấu thành từ:

- Presentation Layers
- Business Logic Layers
- Data Access



6.2.1. Presentation Layers

Lớp này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả/dữ liệu thông qua các thành phần trong giao diện người sử dụng. Trong .NET thì bạn có thể dùng Windows Forms, ASP.NET hay Mobile Forms để hiện thực lớp này.

Lưu ý : Lớp này không nên sử dụng trực tiếp các dịch vụ của lớp Data Access mà nên sử dụng thông qua các service của lớp Business Logic vì khi bạn sử dụng trực tiếp như vậy, bạn có thể bỏ qua các ràng buộc, các logic nghiệp vụ mà ứng dụng cần phải có. Và hơn nữa nếu sử dụng như vậy thì đâu cần đến 3 lớp phải không bạn?

6.2.2. Business Logic Layer

Đây là layer xử lý chính các dữ liệu trước khi được đưa lên hiển thị trên màn hình hoặc xử lý các dữ liệu trước khi chuyển xuống Data Access Layer để lưu dữ liệu xuống cơ sở dữ liệu.

Đây là nơi để kiểm tra ràng buộc, các yêu cầu nghiệp vụ, tính toán, xử lý các yêu cầu và lựa chọn kết quả trả về cho Presentation Layers.

6.2.3. Data Access Layer

Lớp này thực hiện các nghiệp vụ liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng như đọc, lưu, cập nhật cơ sở dữ liệu.

6.3. CÁCH VẬN HÀNH CỦA MÔ HÌNH

Đối với 3-Layer, yêu cầu được xử lý tuần tự qua các layer như hình.

- Đầu tiên User giao tiếp với Presentation Layers (GUI) để gửi đi thông tin và yêu cầu. Tại layer này, các thông tin sẽ được kiểm tra, nếu OK chúng sẽ được chuyển xuống Business Logic Layer (BLL).
- Tại BLL, các thông tin sẽ được nhào nặn, tính toán theo đúng yêu cầu đã gửi, nếu không cần đến Database thì BLL sẽ gửi trả kết quả về GUI, ngược lại nó sẽ đẩy dữ liệu (thông tin đã xử lý) xuống Data Access Layer (DAL).
- DAL sẽ thao tác với Database và trả kết quả về cho BLL, BLL kiểm tra và gửi nó lên GUI để hiển thị cho người dùng.
- Một khi gặp lỗi (các trường hợp không đúng dữ liệu) thì đang ở layer nào thì quăng lên layer cao hơn nó 1 bậc cho tới GUI thì sẽ quăng ra cho người dùng biết
- Các dữ liệu được trung chuyển giữa các Layer thông qua một đối tượng gọi là Data Transfer Object (DTO), đơn giản đây chỉ là các Class đại diện cho các đối tượng được lưu trữ trong Database.

6.4. TỔ CHỨC MÔ HÌNH 3-LAYER

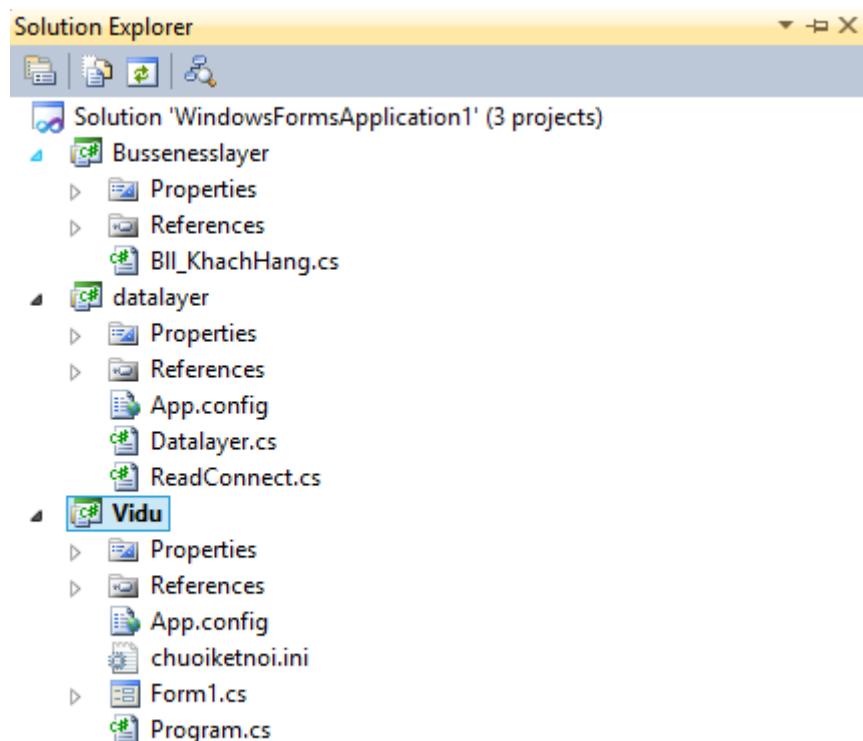
Có rất nhiều cách đặt tên cho các thành phần của 3 lớp như:

Cách 1: GUI, BUS, DAL

Cách 2: GUI, BLL, DAO, DTO

Cách 3: Presentation, BLL, DAL

Một cấu trúc mẫu theo cách 1:



Code mẫu xây dựng từng lớp

Lớp DataLayer

Thiết kế hai lớp: lớp Datalayer.cs tạo các phương thức với chức năng insert, update, delete, select đến cơ sở dữ liệu và lớp ReadConnect gồm các phương thức đọc chuỗi kết nối từ file App.config và *.ini

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration;
using Manager = System.Configuration.ConfigurationManager;
using Settings = System.Configuration.ConnectionStringSettings;
using System.IO;
using System.Data.SqlClient;

namespace datalayer
{
    class ReadConnect
    {

```

```

#region bienthanhvien
private string connectionString = "";
// SqlConnection
public string ConnectionString
{
    get { return connectionString; }
    set { connectionString = value; }
}
private string iniFile = "";

public string IniFile
{
    get { return iniFile; }
    set { iniFile = value; }
}
// thuộc tính dùng kiểm tra quyền đăng nhập
private bool windowNT = false;

public bool WindowNT
{
    get { return windowNT; }
    set { windowNT = value; }
}
private string serverName = "(local)";

public string ServerName
{
    get { return serverName; }
    set { serverName = value; }
}
private string instanceName = "";

public string InstanceName
{
    get { return instanceName; }
    set { instanceName = value; }
}
private string databaseName = "";

public string DatabaseName
{
    get { return databaseName; }
    set { databaseName = value; }
}
private string userName = "sa";

public string UserName
{
    get { return userName; }
    set { userName = value; }
}
private string password = "";

public string Password
{

```

```

        get { return password; }
        set { password = value; }
    }
    private string connectionTimeout = "";

    public string ConnectionTimeout
    {
        get { return connectionTimeout; }
        set { connectionTimeout = value; }
    }
    private string portNo = "";

    public string PortNo
    {
        get { return portNo; }
        set { portNo = value; }
    }
#endregion

#region Constructor
public ReadConnect()
{
}
/// <summary>
/// Khởi tạo chuỗi kết nối với chuỗi kết nối trực tiếp.
/// </summary>
/// <param name="connectstr">Giá trị chuỗi kết nối</param>
public ReadConnect(string connectstr)
{
    ConnectionString = connectstr;
}
/// <summary>
/// Hàm khởi tạo giá trị chuỗi kết nối
/// </summary>
/// <param name="server">Tên server</param>
/// <param name="database">Tên Cơ sở dữ liệu</param>
/// <param name="userid">tên người dùng</param>
/// <param name="password">Mật khẩu người dùng</param>
public ReadConnect(string server, string database, string
userid, string password)
{
    serverName = server;
    databaseName = database;
    userName = userid;
    passWord = password;
}
/// <summary>
/// Hàm khởi tạo với hai thuộc tính
/// </summary>
/// <param name="server">ServerName</param>
/// <param name="database">Tên Cơ sở dữ liệu</param>
public ReadConnect(string server, string database)
{
    serverName = server;
}

```

```

        databaseName = database;
    }
    /// <summary>
    /// hàm tạo chuỗi kết nối từ tập tin ini
    /// </summary>
    /// <param name="appfile">thuộc tính xác định tập tin ini hay
app.config: true là file app.config; false là file *ini</param>
    /// <param name="filename"></param>
    public ReadConnect(bool appfile, string filename)
    {
        if (!appfile)
            iniFile = filename;
    }

#endregion

#region method
//phương thức dùng để tạo ra chuỗi kết nối từ các thuộc
//tính trong chuỗi kết nối.
/// <summary>
/// Ý nghĩa : Nối các thuộc tính thành chuỗi kết nối.
/// </summary>
/// <returns></returns>
public string GetconnectionString()
{
    string conString = "";
    //trường hợp có instance
    if (!instanceName.Equals(""))
    {
        serverName += @"\\" + instanceName;
    }
    //hàm if kiểm tra quyền đăng nhập
    if (windowNT)
    {
        conString = "server=" + serverName + ";database=" +
databaseName + ";Integrated Security= true";
    }
    //đăng nhập theo quyền của SQL
    else
    {
        conString = "server=" + serverName + ";database=" +
databaseName + ";uid=" + userName + ";pwd=" + password;
    }
    //kiểm tra giá trị thuộc tính timeout
    if (!connectionTimeout.Equals(""))
    {
        conString += ";connection Timeout=" + connectionTimeout;
    }
    //kiểm tra port
    if (!portNo.Equals(""))
    {
        conString += ";Port=" + portNo;
    }
    return conString;
}

```

```

    }
    /// <summary>
    /// Ý nghĩa: Lấy chuỗi kết nối từ file App.config
    /// </summary>
    /// <param name="nodeName">Tên của node chứa chuỗi kết nối trong
file AppConfig</param>
    /// <returns></returns>
    public string GetConnectionWithAppConfig(string nodeName)
    {
        string StrApp = "";
        //namespace sử dụng để đọc giá trị trong file xml

        Settings settings;
        settings = Manager.ConnectionStrings[nodeName];
        //namespace Manager dùng đọc giá trị trong tab có tên là
nodeNode trong file config
        StrApp = settings.ConnectionString;

        return StrApp;
    }
    //hàm đọc chuỗi kết nối trong file ini theo từng thuộc tính.
    /// <summary>
    /// Ý nghĩa : Lấy chuỗi kết nối trong file ini
    /// </summary>
    /// <returns></returns>
    public string GetConnectionWithinifile()
    {
        string StrApp = "";
        using (StreamReader sReader = new StreamReader(iniFile))
        {
            string line = "";
            while ((line = sReader.ReadLine()) != null)
            {
                switch ((line.Substring(0,
line.IndexOf("="))).ToLower())
                {
                    case "servername":
                        serverName = GetValue(line);
                        break;
                    case "databasename":
                        databaseName = GetValue(line);
                        break;
                    case "instancename":
                        instanceName = GetValue(line);
                        break;
                    case "uid":
                        userName = GetValue(line);
                        break;
                    case "pwd":
                        passWord = GetValue(line);
                        break;
                    case "portno":
                        portNo = GetValue(line);
                        break;
                    case "timeout":

```

```

        connectionTimeout = GetValue(line);
        break;
    }
}
StrApp = GetConnectionString();
}
return StrApp;
}
public string docchuoiketnoi()
{
    using (StreamReader sReader = new StreamReader(iniFile))
    {
        return sReader.ReadLine();
    }

}
/// <summary>
/// Ý nghĩa: Lấy chuỗi kết nối từ file ini
/// Có chỉ đường dẫn cụ thể.
/// </summary>
/// <param name="path">Đường dẫn file ini</param>
/// <returns></returns>
public string GetConnectionWithinifile(string path)
{
    string StrApp = "";
    using (StreamReader sReader = new StreamReader(path))
    {
        string line = "";
        while ((line = sReader.ReadLine()) != null)
        {
            switch ((line.Substring(0,
line.IndexOf("="))).ToLower())
            {
                case "servername":
                    serverName = GetValue(line);
                    break;
                case "databasename":
                    databaseName = GetValue(line);
                    break;
                case "instancename":
                    instanceName = GetValue(line);
                    break;
                case "portno":
                    portNo = GetValue(line);
                    break;
                case "uid":
                    userName = GetValue(line);
                    break;
                case "pwd":
                    passWord = GetValue(line);
                    break;
                case "timeout":
                    connectionTimeout = GetValue(line);
                    break;
            }
        }
    }
}

```

```

        }
        StrApp = GetconnectionString();
    }
    return StrApp;
}
//hàm lấy giá trị trong từng thuộc tính trong file ini
/// <summary>
/// Lấy giá trị trên từng dòng trong file ini
/// </summary>
/// <param name="line">dữ liệu từng dòng (string)</param>
/// <returns></returns>
private string GetValue(string line)
{
    string stringvalue = "";
    if (!line.Equals(""))
        stringvalue = line.Substring(line.LastIndexOf("=") + 1);
    return stringvalue;
}

//hàm đọc chuỗi kết nối trong file config theo dạng từng thuộc
tính.
/// <summary>
/// Ý nghĩa: Lấy giá trị từ File App.Config lấy theo từng giá
tri
/// </summary>
/// <param name="server">Server</param>
/// <param name="instance">Đối tượng sql</param>
/// <param name="database">Tên Database cần kết nối</param>
/// <param name="userid">Người dùng</param>
/// <param name="password">Mật khẩu</param>
/// <param name="port">Cổng</param>
/// <param name="timeout">Thời gian kết nối</param>
/// <returns></returns>
public string GetConnectionStringWithAppConfig(string server,
string instance, string database, string userid, string password, string
port, string timeout)
{
    string strApp = "";

    //doc thuc tinh servername
    serverName = Manager.AppSettings.Get(server);
    //doc thuoc tinh instance

    instanceName = Manager.AppSettings.Get(instance);
    //doc thuoc tinh database
    databaseName = Manager.AppSettings.Get(database);
    //doc thuoc tinh username
    userName = Manager.AppSettings.Get(userid);
    //doc thuoc tinh pass
    passWord = Manager.AppSettings.Get(password);
    //doc thuoc tinh port
    portNo = Manager.AppSettings.Get(port);
    //doc thuoc tinh timeout
    connectionTimeout = Manager.AppSettings.Get(timeout);
    strApp = GetconnectionString();
}

```

```

        return strApp;
    }

#endregion

}

}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data;

namespace datalayer
{
    public class Datalayer
    {
        SqlConnection cnn;
        SqlCommand cmd;
        SqlDataAdapter da;
        ReadConnect _read = new ReadConnect();
        public Datalayer()
        {
            cnn=new
SqlConnection(_read.GetConnectionWithAppConfig("SqlServer"));
            cmd = cnn.CreateCommand();

        }
        public DataTable GetData(string sql, CommandType ct, ref string
err, params SqlParameter[] param)
        {
            DataTable dt = new DataTable();
            try
            {
                cnn.Open();
                cmd = new SqlCommand(sql, cnn);
// cmd.CommandText = sql;
                cmd.CommandType = ct;
                cmd.Parameters.Clear();
                if (param != null)
                {
                    foreach (SqlParameter p in param)
                    {
                        cmd.Parameters.Add(p);
                    }
                }

                da = new SqlDataAdapter(cmd);
                da.Fill(dt);
            }
        }
    }
}
```

```

        }
        catch (Exception ex)
        {
            err = ex.Message.ToString();
        }
        finally
        {
            cnn.Close();
        }
        return dt;
    }
    public bool myExecuteNonQuery(string sql, CommandType ct, ref
string err, params SqlParameter[] param)
{
    bool ketqua = false;
    cnn.Open();
    try
    {
        cmd = new SqlCommand(sql, cnn);
        // cmd.CommandText = sql;
        cmd.CommandType = ct;
        cmd.Parameters.Clear();
        if (param != null)
        {
            foreach (SqlParameter p in param)
            {
                cmd.Parameters.Add(p);
            }
        }
        cmd.ExecuteNonQuery();
        ketqua = true;
    }
    catch (Exception ex)
    {
        err = ex.Message.ToString();
    }
    finally
    {
        cnn.Close();
    }
    return ketqua;
}
/// <summary>
/// Hàm lấy giá trị duy nhất trong sql
/// </summary>
/// <param name="sql">thủ tục hay câu truy vấn</param>
/// <param name="ct">CommandType</param>
/// <param name="err">Biến tham chiếu lưu lỗi</param>
/// <param name="param">Danh sách tham số, có thể để null nếu
thu tục không tham số</param>
/// <returns>đối tượng object </returns>
public object MyExecuteScalar(string sql, CommandType ct, ref
string err, params SqlParameter[] param)
{

```

```

        object o = null;
        cnn.Open();
        try
        {
            cmd = new SqlCommand(sql, cnn);
            cmd.CommandType = ct;
            cmd.Parameters.Clear();
            if (param != null)
            {
                foreach (SqlParameter p in param)
                {
                    cmd.Parameters.Add(p);
                }
            }
            o = cmd.ExecuteScalar();

        }
        catch (Exception ex)
        {
            err = ex.Message.ToString();
        }
        finally
        {
            cnn.Close();
        }
        return o;
    }
}
}

```

Lớp businesslayer:

Làm công việc liên hệ giữa lớp giao diện và lớp datalayer.

Ví dụ về lớp Business cho form khách hàng

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using datalayer;
using System.Data;
using System.Data.SqlClient;

namespace Bussenesslayer
{
    public class Bll_KhachHang
    {
        Datalayer da;
        public Bll_KhachHang()
        {
            da = new Datalayer();
        }
    }
}

```

```

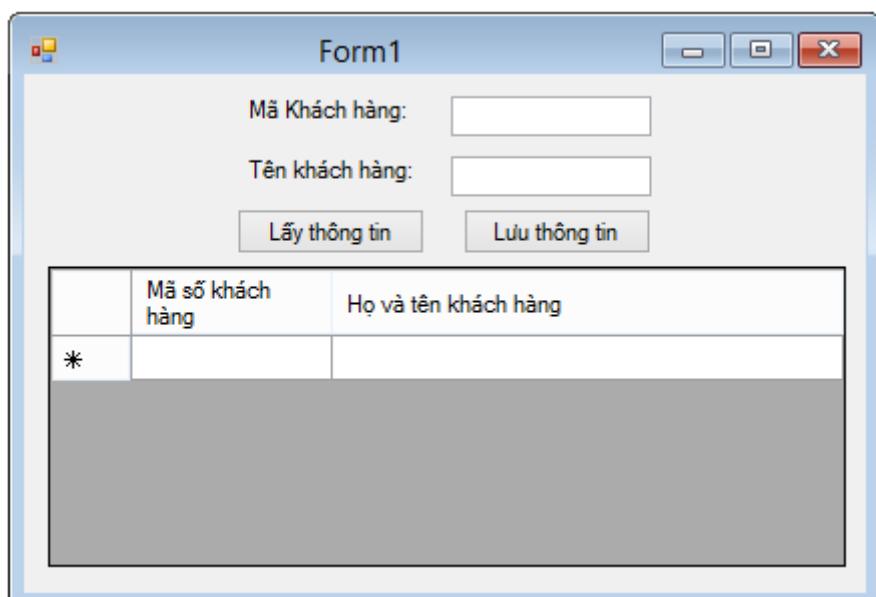
public DataTable LayKhachHang(ref string err)
{
    return da.GetData("select * from khachhang",
 CommandType.Text, ref err, null);
}
public DataTable LayKhachHang(ref string err, int makhachhang)
{
    return da.GetData("sp_laykhachhangtheomaso",
 CommandType.Text, ref err, new
 SqlParameter("@makhachhang", makhachhang));
}
public DataTable LayKhachHang(ref string err, string
 tenkhachhang)
{
    return da.GetData("sp_laykhachhangtheoten",
 CommandType.Text, ref err, new SqlParameter("@tenkhachhang",
 tenkhachhang));
}

public bool themkhachhang(ref string err, int makhachhang,
 string tenkhachhang)
{
    return da.myExecuteNonQuery("sp_themkhachhang",
 CommandType.StoredProcedure, ref err, new SqlParameter("@makhachhang",
 makhachhang), new SqlParameter("@tenkhachhang", tenkhachhang));
}
public int laysoluongkhachhang(ref string err)
{
    return (int)da.MyExecuteScalar("select count(1) from
 KhachHang", CommandType.Text, ref err, null);
}

}
}

```

Lớp giao diện



Code

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
using Bussenesslayer;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }

        Bll_KhachHang bd = new Bll_KhachHang();
        private void Form1_Load(object sender, EventArgs e)
        {
            laythongtin();
        }
        string err = "";
        private void laythongtin()
        {
            dataGridView1.DataSource = bd.LayKhachHang(ref err);
            if (!String.IsNullOrEmpty(err))
                MessageBox.Show(err);
        }
        private void btnlaythongtin_Click(object sender, EventArgs e)
        {
            laythongtin();
        }
        int makhachhang;
        string tenkhachhang;
        private void btnluuuthongtin_Click(object sender, EventArgs e)
        {
            if (txtmakhachhang.Text.Length > 0)
            {
                if (txttenkhachhang.Text.Length > 0)
                {
                    makhachhang = Convert.ToInt32(txtmakhachhang.Text);
                    tenkhachhang = txttenkhachhang.Text;
                    if (bd.themkhachhang(ref err, makhachhang,
tenkhachhang))
                    {
                        MessageBox.Show("Thanhcong");
                        laythongtin();
                    }
                }
            }
        }
    }
}

```

```
        }
        else
        {
            MessageBox.Show("thất bại"+err);
        }
    }
    else
    {
        MessageBox.Show("nhapten", "Thong
bao", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
        txttenkhachhang.Focus();
    }

}
else
{
    MessageBox.Show("nhập mã số", "Thong bao",
MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
    txttenkhachhang.Focus();
}
}
```

PHỤ LỤC

CHUẨN VIẾT CODE TRONG C#

Đưa ra các quy ước khi viết với ngôn ngữ lập trình C#, với các quy tắc này giúp tiết kiệm thời gian rất lớn trong tiến trình phát triển phần mềm và cả trong quá trình bảo trì sản phẩm. Giúp sinh viên quen với làm việc theo nhóm.

Tài liệu này chủ yếu hướng dẫn sinh viên với ngôn ngữ lập trình C#, nhưng có rất nhiều quy tắc được sử dụng trong nhiều ngôn ngữ lập trình khác tích hợp trong bộ công cụ Visual Studio .NET.

Các nội dung quy định

Các kiểu quy ước viết hoa

Có 3 quy tắc :

- ✓ **Pascal Case:** Chữ cái đầu tiên trong từ định danh và chữ cái đầu tiên của mỗi từ nối theo sau phải được viết hoa. Sử dụng Pascal case để đặt tên cho một tên có từ 3 ký tự trở lên.
Ví dụ: BackColor
- ✓ **Camel Case :** Chữ cái đầu tiên trong từ định danh là chữ thường và chữ cái đầu tiên của mỗi từ nối theo sau phải được viết hoa.
Ví dụ: backColor

- ✓ **Uppercase :** Tất cả các ký tự trong từ định danh phải được viết hoa. Sử dụng quy tắc này đối với tên định danh có từ 2 ký tự trở xuống.

Ví dụ: System.IO

Một số lưu ý trong cách đặt tên cho các thành phần

Không dùng các tên giống nhau(chỉ phân biệt kiểu chữ in hoa hay thường). Ta khó nhận ra các định danh nhất là khi trong cùng ngữ cảnh và chỉ phân biệt các định danh bằng kiểu chữ in hoa/thường.

- ✓ Không tạo 2 namespace cùng tên và chỉ khác nhau ở kiểu chữ viết(chữ hoa/Chữ thường)

Ví dụ: 2 namespace này rất khó để nhận biết:

Namespace IridiumSoftware

Namespace iridiumsoftware

- ✓ Không nên xây dựng 1 phương thức với các tham số có cùng tên và chỉ khác nhau kiểu chữ

Ví dụ:

```
Void MyFunction(string a, string A)
```

- ✓ Không xây dựng 1 kiểu với các tên property giống nhau và chỉ phân biệt ở kiểu chữ

Ví dụ: 2 property dưới đây rất khó để nhận biết:

```
int Color {get, set}  
int COLOR {get, set}
```

- ✓ Không đặt tên các phương thức có cùng tên và chỉ khác nhau ở kiểu chữ

Ví dụ:

```
void calculate()  
void Calculate()
```

- ✓ Không dùng từ viết tắt hoặc 1 phần của các tên định danh như: GetWindow viết thành GetWin là ko hợp quy ước. Ta có thể sử dụng kí tự đầu tiên của các tên có một nhóm từ để đặt tên cho nó.

Ví dụ:

UI thay cho User Interface.

- ✓ Không sử dụng các từ viết tắt đã được thừa nhận từ các lĩnh vực tin học khác. (Ví dụ: XML, TTL, DNS, UI, IP và IO ...)

Khi đặt tên viết tắt, bạn có thể sử dụng cách viết **Pascal case** hay **Camel case** để đặt tên. Trong trường hợp này bạn có thể dùng tên nào cũng được HtmlButton hoặc HTMLButton. Tuy nhiên với trường hợp tên có 2 ký tự hoặc ít hơn thì bạn nên viết như System.IO thay cho System.Io.

Không dùng từ viết tắt trong tên định danh hoặc tên tham số. Nếu bạn cần thiết phải dùng đến từ viết tắt, dùng cách viết Camel Case cho từ viết tắt có từ 2 ký tự trở lên thậm chí là nó phủ định lại chuẩn viết tắt của 1 từ. Tránh sử dụng lại các từ khóa hoặc tên các class chuẩn được dùng trong .NET Framework namespaces.

Cách tổ chức file trong một project

File mã nguồn

Giữ file nguồn các class không quá dài (không được vượt quá 2000 LOC - Lines of Code), phân chia code thành những đơn vị code nhỏ tạo nên các cấu trúc code sáng sủa hơn. Đặt mỗi class vào mỗi file source riêng biệt và tên file cũng là tên class (với phần đuôi mở rộng .cs)

Thứ tự trong file source C#:

- ✓ Các lệnh using
- ✓ Lệnh namespace
- ✓ Các khai báo Class and interface

Các lệnh Namespace và using

Các dòng không phải là dòng comment ở hầu hết các file source của C# là các lệnh using, kế đến là lệnh namespace như sau:

```
using System.Data;
namespace Business.Framework;
```

Cả 2 lệnh này đều được canh sát lề trái.

Các khai báo class và interface

Khai báo class -> Ghi chú -> Nội dung class

Ví dụ: Khai báo một class Class/interface documentation

```
/// <summary>
/// The Person class provides ...
/// </summary>
public class Person
```

Trình tự khai báo các fields theo mức độ cho phép truy cập

Private

Protected

Internal

public

Trình tự khai báo các Properties theo mức độ cho phép truy cập

Private

Protected
Internal
public

Trình tự khai báo các Constructors theo mức độ cho phép truy cập

Private
Protected
Internal
Public

Quy Tắc đặt tên biến

Quy tắc đặt tên cho namespace

Quy tắc chung cho việc đặt tên Namespace là dùng tên của công ty theo sau là tên công nghệ và kế đến là tùy chọn tên đặc trưng hay thiết kế :

CompanyName.TechnologyName[.Feature][.Design]

Ví dụ: Quy tắc đặt tên Namespace

IridiumSoftware.IridiumX
IridiumSoftware.IridiumX.Design
Microsoft.Office

Quy tắc đặt tên cho class

Thường dùng danh từ hoặc một cụm danh từ để đặt tên cho 1 class. Kí tự đầu tiên của mỗi từ là chữ in hoa.(Dùng quy tắc Pascal Case). Không sử dụng dạng tiền tố

Ví dụ: FileStream là tên chuẩn. Không đặt tên theo dạng CfileStream hay ClassFileStream

Trong tên class không sử dụng dấu gạch chân (_).

Trong trường hợp cần thiết, nếu kí tự đầu tiên của tên 1 class bắt đầu bằng kí tự ‘I’ và kí tự này là kí tự bắt đầu của một từ và từ đó là 1 phần của tên class thì tên đó vẫn được chấp nhận.

Ví dụ: IdentityStore. Sử dụng một từ ghép để đặt tên cho 1 class dẫn xuất, từ thứ 2 trong tên của class dẫn xuất nên lấy tên của class cơ sở.

Ví dụ: ApplicationException là tên class dẫn xuất từ class cơ sở Exception, và với cách đặt tên này ta có thể hiểu rằng ApplicationException là một trong các loại Exception.

Với quy tắc này thì tùy trường hợp mà chúng ta có thể áp dụng để tránh đặt tên class dài lê thê với những từ không cần thiết.

Ví dụ: Class Button được dẫn xuất từ class Control nhưng nếu ta đặt tên theo quy tắc trên thì tên class này là ButtonControl. Tên class này trở nên dài ra và không cần thiết vì mặc định thì button được xem như là một control rồi. Dưới đây là cách đặt tên class chuẩn:

```
public class FileStream
public class Button
public class String
```

Quy tắc đặt tên cho Interface

Thường dùng danh từ hoặc một cụm danh từ, hay một tính từ mà nó mô tả hành vi để đặt tên cho 1 interface.

Ví dụ: Cách đặt tên Interface

interface Icomponent sử dụng 1 danh từ.

interface ICUSTOMATTRIBUTEPROVIDER sử dụng 1 cụm danh từ.

interface IPERSISTABLE sử dụng một tính từ.

Sử dụng quy tắc PascalCase.

Tên đặt cho interface phải ngắn gọn.

Khi định nghĩa 1 cặp class/interface mà class này là 1 thực thi chuẩn của interface thì tên giữa class và interface nên đặt tương tự nhau. Và điểm khác nhau là tên interface có kí tự tiền tố “I”.

Không sử dụng kí tự gạch chân “_” trong tên interface.

Ví dụ: Đặt Interface chuẩn

```
public interface IServiceProvider
public interface IFormattable
```

Ví dụ: Cách định nghĩa interface, một class thực thi chuẩn của interface:

```
public interface IComponent
{
    // Implementation goes here.
}
```

```
public class Component : IComponent
{
    // Implementation goes here.
}
```

Quy tắc đặt tên cho Attribute

Bạn nên thêm vào hậu tố Attribute đối với các lớp attribute.

```
public class ObsoleteAttribute{ }
```

Quy tắc đặt tên cho Enumeration

- Sử dụng Pascal Case cho các tên kiểu và giá trị enum.
- Sử dụng tên ngắn gọn và tường minh.
- Không sử dụng hậu tố Enum trong tên kiểu Enum.

Quy tắc đặt tên cho StaticField

- Dùng các danh từ, cụm danh từ hay các danh từ viết tắt để đặt tên cho static fields.
- Dùng Pascal Case.
- Sử dụng tiền tố ký pháp Hungarian trong tên của static field.
- Người ta khuyên dùng thuộc tính static để thay thế cho các static fields có cấp độ chia sẻ dạng public.

Quy tắc đặt tên cho Parameter

- Tên Parameter phải được mô tả một cách đầy đủ và nó bao hàm cả tên của tham số và kiểu dữ liệu của nó
- Dùng Camel Case.
- Tên tham số nên đặt theo hướng mô tả ý nghĩa của tham số tốt hơn là đặt theo kiểu dữ liệu của tham số. Vì môi trường lập trình đã cung cấp các thông tin về kiểu dữ liệu tham số.
- Không nên dùng các tham số để dành, mà khi cần thiết ta có thể thêm vào trong các version sau này của các thư viện class.
- Không sử dụng các tên tham số tiền tố với kí pháp Hungarian.

Ví dụ:

```
Type GetType(string typeName)
```

string Format(string format, object[] args)

Quy tắc đặt tên cho Method

- Dùng các động từ hay các cụm động từ để đặt tên cho methods.
- Dùng Pascal Case.

Ví dụ:

```
RemoveAll()
GetCharArray()
Invoke()
```

Quy tắc đặt tên cho Property

- Dùng 1 danh từ hay 1 cụm danh từ để đặt tên cho 1 property.
- Dùng Pascal Case.
- Không dùng ký pháp Hungarian.
- Tạo 1 property và đặt tên tương tự như kiểu dữ liệu của nó.

```
public class SampleClass
{
    public Color BackColor
    {
        // Code for Get and Set accessors goes here.
    }
}
```

Quy tắc đặt tên cho Event

- Dùng 1 hậu tố EventHandler trong các tên của event handler.
- Chỉ định rõ 2 tham số có tên là sender và e. Tham số sender mô tả object gọi event. Tham số sender luôn luôn có kiểu dữ liệu là object. Trạng thái kết hợp với sự kiện được đóng gói trong một thể hiện của event class có tên là e. Kiểu của tham số e phải là một lớp sự kiện rõ ràng và thích hợp.
- Đặt tên 1 lớp đối số sự kiện với hậu tố EventArgs.
- Nên đặt tên 1 sự kiện là 1 động từ.
- Dùng danh động từ (dạng thêm “ing” của động từ) để đặt tên cho một sự cho một sự kiện mà nó thể hiện khái niệm pre_event và quá khứ của động từ để thể hiện sự kiện post_event.

- Không dùng tiền tố và hậu tố trong khai báo event có dạng “OnXXX”. Ví dụ: dùng Close thay vì dùng OnClose.
- Tông quát hơn, bạn nên cung cấp 1 method có mức truy cập protected gọi sự kiện có dạng “OnXXX” mà sự kiện này có thể bị ghi đè trong 1 class dẫn xuất. Method này chỉ nên có tham số sự kiện e.

Ví dụ:

```
public delegate void MouseEventHandler(object sender,
MouseEventArgs e);
```

Quy tắc đặt tên cho Control

Bảng danh sách các tiền tố của các kiểu controls thông dụng:

Prefix	Control
lbl	Label
llbl	LinkLabel
but	Button
txt	Textbox
mnu	MainMenu
chk	CheckBox
rdo	RadioButton
grp	GroupBox
pic	PictureBox
grd	Grid
lst	ListBox
cbo	ComboBox
lstv	ListView
tre	TreeView
tab	TabControl
dtm	DateTimePicker
mon	MonthCalendar
sbr	ScrollBar
tmr	Timer
spl	Splitter
dud	DomainUpDown
nud	NumericUpDown
trk	TrackBar
pro	ProgressBar
rtxt	RichTextBox
img	ImageList
hlp	HelpProvider
tip	ToolTip
cmnu	ContextMenu
tbr	ToolBar

frm	Form
bar	StatusBar
nico	NotifyIcon
ofd	OpenFileDialog
sfd	SaveFileDialog
fd	FontDialog
cd	ColorDialog
pd	PrintDialog
ppd	PrintPreviewDialog
ppc	PrintPreviewControl
err	ErrorProvider
pdoc	PrintDocument
psd	PageSetupDialog
crv	CrystalReportViewer
pd	PrintDialog f
sw	FileSystemWatcher
log	EventLog
dire	DirectoryEntry
dirs	DirectorySearcher
msq	MessageQueue
pco	PerformanceCounter
pro	Process
ser	ServiceController
rpt	ReportDocument
ds	DataSet
olea	OleDbDataAdapter
olec	OleDbConnection
oled	OleDbCommand
sqla	SqlDbDataAdapter
sqlc	SqlDbConnection
sqld	SqlDbCommand
dvw	DataView

Menu Controls. Menu controls nên đặt tên bằng cách dùng tag “mnu” theo sau là đường dẫn xuống đầy đủ theo cây menu. Ví dụ về tên 1 biến control menu:

```
mnuFile
    mnuFileNew
    mnuEdit
        mnuEditCopy
        mnuInsertIndexAndTables
        mnuTableCellHeightAndWidth
```

Quy tắc đặt tên cho Data

- Nếu cần dùng tên CustomerCode và hiển thị lên 1 text box, thì text box phải đặt tên là txtCustomerCode.
- Nếu dùng combobox để hiển thị các mã số khách hàng, thì control đó đặt tên là cboCustomerCode.
- Nếu muốn lưu mã khách hàng vào biến của 1 module thì nó được đặt tên là mCustomerCode.
- Nếu muốn chuyển sang chuỗi thì dùng câu lệnh sau:

String mCustomerCode = CustomerCode.ToString()

Quy tắc Comment

- Các chương trình C# có thể có 2 loại comment là implementation comments và documentation comments
- Implementation comments là những comment trong C++, mà được định nghĩa bởi các dấu /*...*/, và //.
- Documentation comments thì chỉ có trong C#, và được định nghĩa bởi các tag XML. Implementation comments dùng để giải thích cho 1 đoạn code hay để chú thích cho 1 implementation một cách cụ thể hơn.

Các kỹ thuật Comment

- Khi sửa code nên nhớ comment ngày sửa code.
- Comments nên chứa các câu giải thích hoàn.
- Tránh sử dụng comment cuối dòng cho các đoạn code (trừ các khai báo biến).
- Tránh dùng các dòng comment chứa các đài dâu hoa thị mà nên dùng các khoảng trắng thay vào đó.
- Tránh dùng các khối comment có khung viền nghệ thuật. Mặc dù nhìn thì đẹp nhưng rất khó khăn khi bảo trì.
- Trước khi chuẩn bị triển khai nên xóa các comment tạm thời hay các comment không liên quan để tránh gặp rối trong quá trình bảo trì sản phẩm sau này.
- Dùng các câu comment hoàn thiện. Comment nên sáng sủa không nên thêm vào các câu nhập nhằng.
- Để tránh comment lặp lại nhiều lần thì ta nên thực hiện động tác comment vào lúc fix bug cho code, đặc biệt là khi làm việc trong môi trường nhóm.
- Nên dùng comments cho code chứa các vòng lặp và điều kiện rẽ nhánh. Nó sẽ hỗ trợ người đọc khi đọc source code.

Implementation Comments

Block comments

- Với C# thì comment (///) thường được sử dụng (do VS tự generate). Khi dùng comments cho khôi chú thích thì nên dùng theo style sau:

```
/* Line 1
 * Line 2
 * Line 3
 */
```

- Comment như bên dưới là rất hiếm: /* http://www.vi-infotech.com */
- Block comments được dùng để mô tả cho các file, method, cấu trúc dữ liệu và các thuật toán.
- Mỗi file code nên có khôi comment header, khôi header nên trình bày như bên dưới:

```
#region Copyright vi-infotech.com @ 2012
//
// All rights are reserved. Reproduction or transmission
in whole or
    // in part, in any form or by any means, electronic,
mechanical or
    // otherwise, is prohibited without the prior written
consent of the
    // copyright owner.
//
// Filename: abc.cs
//
#endregion
```

Single Line comments

- Nên dùng dấu comment // để comment đối với 1 dòng code và có thể dùng để comment cho 1 đoạn code cũng được.
- Single line comments phải thuộc vào so với mức canh lề của code khi dùng cho tài liệu code. Comment nên được đặt ở dòng đầu tiên của đoạn code muốn comment để làm tăng tính trong sáng và tường minh của đoạn code.
- Thông thường thì độ dài của comment không được vượt quá chiều dài của đoạn code muốn được comment, điều này sẽ làm cho code trở nên rối rắm thêm.

```
if (condition)
{
    // Handle the condition.

    ...
}
```

Trailing comments

Dùng với việc mô tả các khai báo biến hay là các mô tả ngắn cho các câu lệnh:

```
if (a == 2)
{
    return true; // Special case
}
else
{
    return isPrime(a); // Works only for odd a
}
```

Code-Disabling comments

Dấu comment // dùng để che đoạn code được chọn. Để dùng comment này ta dùng các tổ hợp phím CTRL+K, CTRL+C. Để hủy bỏ án CTRL+K, CTRL+U.

```
if (foo > 1)
{
    // Do a double-flip.

    ...
}

else
{
    return false; // Explain why here.
}

// if (bar > 1)
// {
//     // // Do a triple-flip.

//     ...
// }
```

```
// else
// {
// return false;
// }
```

Documentation Comments

Trong .net framework, Microsoft đưa ra hệ thống phát sinh tài liệu (documentation generation system) dựa vào các comment dạng thẻ XML. Những comments sử dụng dấu comment /// cùng với các thẻ XML. Ví dụ lời comment trên 1 dòng:

```
/// <summary>
/// This class...
/// </summary>
```

Ví dụ về lời comment trên nhiều dòng:

```
/// <exception cref="BogusException">
/// This exception gets thrown as soon as a
/// Bogus flag gets set.
/// </exception>
```

Trước tất cả các dòng đều có 3 dấu / đi trước nó. Các Tag chia thành 2 loại:

Documentation items
Formatting/Referencing

Loại đầu tiên bao gồm các tag như <summary>, <param> hay <exception>. Những tag này sẽ mô tả các thành phần API của chương trình và nó được tài liệu hóa để cho các lập trình viên khác có thể dùng lại nó. Những tag này thường có các thuộc tính như name.

Loại thứ 2 liên quan đến việc bố trí của tài liệu, sử dụng các tag như <code>, <list> hoặc <para>.

TÀI LIỆU THAM KHẢO

[1]. Phạm Hữu Khang, Trần Tiến Dũng (2006), C# 2005 , tập 4 – quyển 1 Lập trình Cơ sở dữ liệu NXB Lao Động Xã Hội, Tp. Hồ Chí Minh.