

TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN
--oOo--

PHAN THỊ THANH NGA
NGUYỄN THỊ LƯƠNG
THÁI DUY QUÝ

SÁCH HƯỚNG DẪN THỰC HÀNH
**PHÁT TRIỂN ỨNG DỤNG
DESKTOP**

Đà Lạt, Tháng 6 năm 2021

LỜI NÓI ĐẦU

Sự phát triển như vũ bão của cuộc Cách mạng 4.0 đã kéo theo sự thay đổi lớn trong nhận thức của nhiều người. Với sự phát triển nhanh chóng của khoa học - kỹ thuật - công nghệ, các ứng dụng về chuyên đổi số cũng đã trở nên phong phú, đa dạng và đóng vai trò không thể thiếu trong sự phát triển chung của nền kinh tế. Những chương trình ứng dụng, những phần mềm quản lý được thay thế cho việc quản lý bằng hồ sơ, sổ sách đã quá lạc hậu, tốn chi phí và thời gian. Từ những ứng dụng quản lý trên máy tính, đến website tin tức, thương mại, đến các ứng dụng trên các thiết bị điện tử thông minh, lớn hơn nữa là những chương trình quản lý hoạt động của cả một hệ thống điều khiển tự động, tất cả đều là thành tựu vượt bậc của công nghệ thông tin.

Với sự phát triển không ngừng đó, các ứng dụng mới ngày càng đòi hỏi những tính năng ưu việt hơn, thông minh hơn. Không những về tính linh hoạt, thời gian xử lý mà còn phải kể đến khả năng lưu trữ lượng dữ liệu cực kỳ lớn với tốc độ truy xuất, thực thi lệnh nhanh chóng và hiệu quả. Để đáp ứng điều này, ngoài việc sử dụng hệ quản trị cơ sở dữ liệu phù hợp, chúng ta cũng cần phải có các công cụ, môi trường tốt để phát triển ứng dụng. Quan trọng hơn, cần phải biết cách tổ chức chương trình một cách khoa học và tinh tế nhằm cải thiện hiệu suất và nâng cao chất lượng của phần mềm ứng dụng. Đồng thời, cũng phải biết áp dụng các thành tựu nghiên cứu, các công cụ hiệu quả và mới nhất hiện nay vào trong các ứng dụng.

Nhằm tạo điều kiện để sinh viên cũng như người yêu thích lập trình ứng dụng desktop áp dụng các kiến thức đã học và nâng cao kỹ năng lập trình, chúng tôi biên soạn và giới thiệu “Sách hướng dẫn thực hành Phát triển ứng dụng desktop” này tới các độc giả như một tài liệu học tập, phát triển kỹ năng và tư duy lập trình. Sách hướng dẫn cung cấp các hướng dẫn và bài tập thực hành từ cơ bản tới nâng cao xuyên suốt từ bước thiết kế giao diện chương trình cho tới xử lý các sự kiện của form và các điều khiển. Bên cạnh đó, sách còn hướng dẫn người học phát triển các ứng dụng đọc và ghi dữ liệu từ tập tin văn bản, XML, JSON cũng như kết nối tới cơ sở dữ liệu dùng câu truy vấn đơn giản hoặc gọi thủ tục để thực thi các thao tác truy vấn và lưu trữ dữ liệu. Ngoài ra, cuốn sách còn giới thiệu tới độc giả một cách tiếp cận trong việc sử dụng mô hình ba tầng để tổ chức mã chương trình nhằm nâng cao tính chuyên nghiệp và dễ tái sử dụng mã nguồn.Thêm vào đó, chúng tôi cũng giới thiệu cho người đọc cách sử dụng một thư viện hỗ trợ kết nối cơ sở dữ liệu đang được sử dụng khá phổ biến, đó là Entity Framework. Tóm lại, thông qua cuốn sách hướng dẫn thực hành này, chúng tôi mong muốn độc giả nắm được các bước cơ bản trong việc phát triển ứng dụng desktop, cách kết nối cơ sở dữ liệu sử dụng ADO.NET và Entity framework; cách sử dụng các thư viện hỗ trợ đọc ghi các tập tin dữ liệu văn bản, các tập tin dữ liệu bán cấu trúc XML, JSON. Lượng kiến thức là vô hạn, công nghệ lập trình thường xuyên thay đổi vì thế, trong cuốn sách này, chúng tôi không thể đề cập được hết những khía cạnh, công nghệ của việc xây dựng và phát triển ứng dụng desktop. Thay vào đó, chúng tôi hi vọng, đây sẽ là tài liệu tham khảo tương đối đầy đủ về xây dựng ứng dụng desktop có kết nối cơ sở dữ liệu để quý bạn đọc tự khám phá và phát huy những kỹ năng tiềm ẩn của mình.

Sách hướng dẫn được chia làm 9 bài thực hành với những nội dung cơ bản như sau:

Bài thực hành số 1 giúp người đọc hệ thống lại cơ sở dữ liệu bằng cách sử dụng hệ quản trị cơ sở dữ liệu SQL Server. Sinh viên sẽ thiết kế các bảng, thiết kế các mối quan hệ ràng buộc giữa các bảng, thực hiện các thao tác truy vấn, viết các thủ tục (*Store Procedure*) và các hàm (*Function*).

Bài thực hành số 2 hướng dẫn người đọc sử dụng các điều khiển (control) cơ bản để xây dựng ứng dụng. Các điều khiển được giới thiệu trong bài thực hành này bao gồm Label, TextBox, ComboBox, MaskedTextBox, Button, RadioButton, CheckBox, ListBox, CheckListBox.

Bài thực hành số 3 tiếp tục hướng dẫn sinh viên sử dụng các điều khiển để xây dựng ứng dụng. Ngoài các điều khiển đã được giới thiệu ở bài thực hành số 2, bài thực hành này bổ sung thêm các điều khiển như DateTimePicker, PictureBox, ListView. Bên cạnh đó bài thực hành này còn ôn tập lại kỹ năng lập trình hướng đối tượng và hướng dẫn người đọc xử lý các sự kiện của form và điều khiển để hoàn thiện chức năng cơ bản của ứng dụng.

Bài thực hành số 4 là một bài thực hành tổng hợp nhằm giúp sinh viên ôn tập và quan trọng nhất là áp dụng các kiến thức đã học về sử dụng các control, xử lý sự kiện cho form và control để xây dựng một ứng dụng quản lý thông tin sinh viên đơn giản cho phép đọc dữ liệu từ tập tin txt hiển thị lên form và lưu dữ liệu vào tập tin txt.

Bài thực hành số 5 hướng dẫn sinh viên xây dựng một ứng dụng quản lý đơn giản có đọc ghi dữ liệu từ các loại tập tin khác nhau như txt, xml và json.

Bài thực hành số 6 giúp sinh viên tìm hiểu cách xây dựng ứng dụng có kết nối đến một cơ sở dữ liệu và thực thi trực tiếp một số lệnh truy vấn đơn giản. Sau bài thực hành sinh viên sẽ nắm rõ các thành phần của một chuỗi kết nối và ý nghĩa của chúng, cách tạo đối tượng kết nối, cách thực thi truy vấn dùng đối tượng Command và cách đọc dữ liệu dùng DataReader.

Bài thực hành số 7 hướng dẫn sinh viên kỹ hơn và nâng cao về các kỹ thuật kết nối và thao tác cơ sở dữ liệu. Cụ thể, bài thực hành hướng dẫn sinh viên cách sử dụng đối tượng Parameter để truyền tham số vào các lệnh, cách thực thi và nhận kết quả trả về từ các lệnh truy vấn có tham số, cách thực thi các thủ tục và nhận dữ liệu trả về.

Bài thực hành số 8 giúp sinh viên làm quen với mô hình đa tầng trong phát triển ứng dụng. Bài thực hành sẽ giúp sinh viên xây dựng và phát triển một số chức năng của ứng dụng theo mô hình ba tầng là Tầng giao diện, Tầng xử lý logic và Tầng Truy xuất dữ liệu

Bài thực hành số 9 giúp sinh viên biết cách sử dụng Entity Framework để kết nối tới cơ sở dữ liệu và thực hiện các thao tác truy vấn và lưu trữ dữ liệu.

Chúng tôi hi vọng cuốn sách hướng dẫn thực hành này sẽ là một tài liệu tham khảo có ích đối với sinh viên ngành công nghệ thông tin và những độc giả quan tâm tới lĩnh vực xây dựng và phát triển ứng dụng có kết nối cơ sở dữ liệu. Chúng tôi rất mong nhận được sự cổ vũ và những ý kiến đóng góp quý báu của các quý độc giả giúp bổ sung và hoàn thiện hơn nữa nội dung của cuốn sách hướng dẫn này.

Cuối cùng, chúng tôi xin gửi lời cảm ơn đến các thầy cô và đồng nghiệp đã ủng hộ và giúp đỡ chúng tôi hoàn thành giáo trình này.

Dà Lạt, tháng 6 năm 2021

ĐỀ LUẬN

MỤC LỤC

LAB 1 - ÔN TẬP CƠ SỞ DỮ LIỆU	6
I. Mục tiêu.....	6
II. Hướng dẫn thực hành.....	6
III. Bài tập	15
LAB 2 - WINDOWS FORM – CÁC CONTROL CƠ BẢN	16
I. Mục tiêu.....	16
II. Hướng dẫn thực hành.....	16
III. Bài tập	30
LAB 3 - WINDOWS FORM – XỬ LÝ SỰ KIỆN.....	32
I. Mục tiêu.....	32
II. Hướng dẫn thực hành.....	32
III. Bài tập	43
LAB 4 –ỨNG DỤNG WINDOWS FORM ĐƠN GIẢN	47
I. Mục tiêu.....	47
II. Yêu cầu thực hành	47
III. Bài tập	48
LAB 5 - ỨNG DỤNG WINDOWS FORM – ĐỌC GHI DỮ LIỆU TỪ FILE ..	50
I. Mục tiêu:.....	50
II. Hướng dẫn thực hành	50
III. Bài tập:	58
LAB 6 – KẾT NỐI VÀ TRUY VẤN DỮ LIỆU.....	59
I. Mục tiêu.....	59
II. Hướng dẫn thực hành	59
III. Bài tập	73
LAB 7 – TRUYỀN THAM SỐ VÀ THỰC THI THỦ TỤC	75
I. Mục tiêu.....	75
II. Hướng dẫn thực hành	75
III. Bài tập	88
LAB 8 – MÔ HÌNH ĐA TẦNG	90
I. Mục tiêu.....	90
II. Hướng dẫn thực hành	90

III.	Bài tập	110
LAB 9 – SỬ DỤNG ENTITY FRAMEWORK		111
I.	Mục tiêu.....	111
II.	Hướng dẫn thực hành.....	111
III.	Bài tập	132

ĐỀ LUẬN

LAB 1 - ÔN TẬP CƠ SỞ DỮ LIỆU

Thời lượng: 4 tiết

I. Mục tiêu

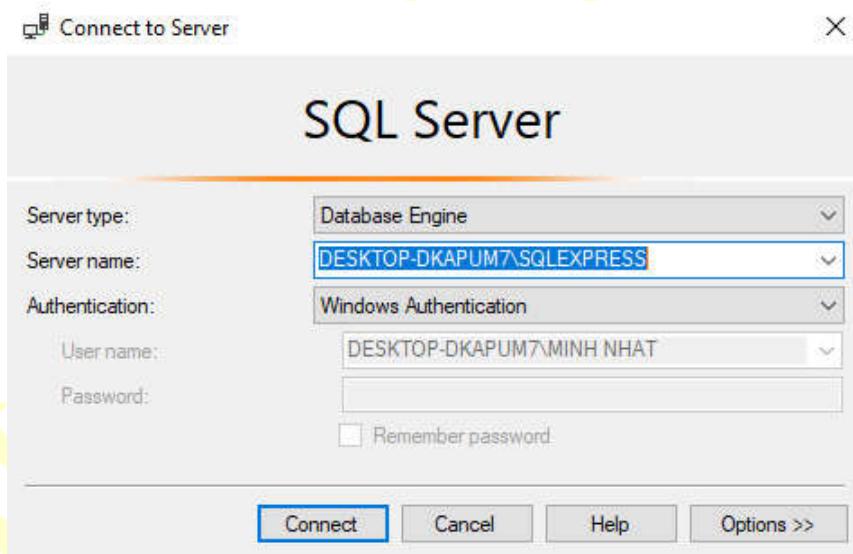
Bài thực hành này giúp sinh viên hệ thống lại cơ sở dữ liệu bằng cách sử dụng hệ quản trị cơ sở dữ liệu SQL Server. Sinh viên sẽ thiết kế các bảng, thiết kế các mối quan hệ ràng buộc giữa các bảng, thực hiện các thao tác truy vấn, viết các thủ tục (*Store Procedure*) và các hàm (*Function*).

II. Hướng dẫn thực hành

1. Giới thiệu SQL Server

SQL Server là một hệ quản trị cơ sở dữ liệu của công ty Microsoft, dùng để thiết kế dữ liệu bằng cách sử dụng các bảng (*tables*), các mối quan hệ (*relationships*), các truy vấn (*query*), thủ tục (*Store procedure*), hàm (*function*) và bảy sự kiện (*trigger*). Giống như những hệ quản trị khác (Oracle, MySQL, DB2, MongoDB...), khi làm việc với SQL Server, người dùng phải thực hiện việc kết nối bằng câu lệnh truy vấn.

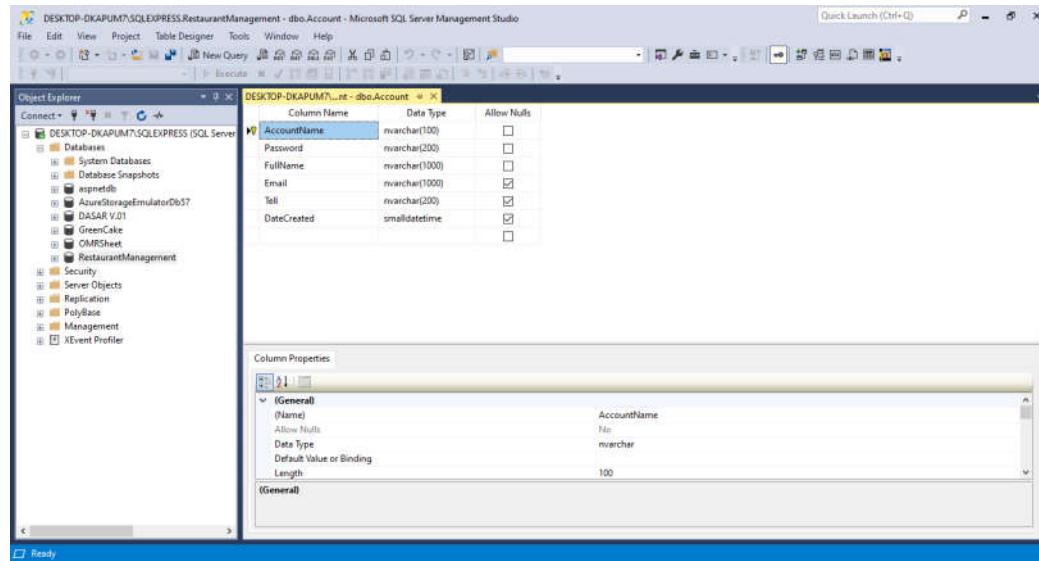
SQL Server sau khi cài đặt thành công, lúc khởi động sẽ bật màn hình như sau:



Trong đó:

- *Server type*: Kiểu Server, thường để mặc định là Database Engine;
- *Server name*: Tên của Server, đây là tên mà người dùng đặt, giống như một máy tính Server chứa thông tin về cơ sở dữ liệu. Khi kết nối bằng ngôn ngữ lập trình cần lưu ý tới đối tượng này.
- *Authentication*: Thường chọn một trong hai chế độ: Nếu là *Windows Authentication* là kết nối theo Windows, nghĩa là khi vào được Windows là sẽ vào được cơ sở dữ liệu (thường thì người dùng sẽ chọn cách này); Nếu chọn *SQL Server Authentication* thì phải nhập Username và mật khẩu để vào SQL Server (cách này áp dụng cho chế độ kết nối từ xa).
- Sau khi chọn các kiểu trên, nhấn Connect để kết nối.

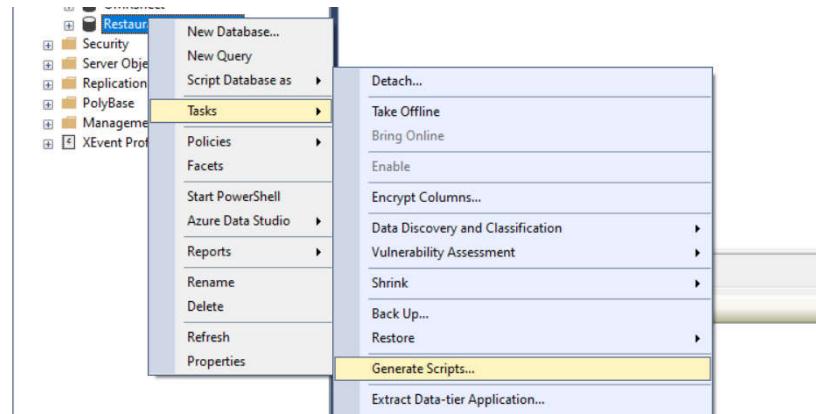
Giao diện sau khi kết nối như sau:



Một số thao tác thường dùng trên hệ quản trị cơ sở dữ liệu SQL Server như sau:

- *Tạo Cơ sở dữ liệu:* Click phải lên Databases, chọn New Database, đặt tên và nhấn OK.
- *Tạo bảng:* Vào Database, click phải lên Table, chọn New, chọn Table: Đặt tên cột, kiểu dữ liệu, chọn cho phép Null hay không, nhấn lưu để đặt tên cho bảng.
- *Tạo truy vấn:* Click phím F11 hoặc click phải lên Database, chọn New Query (hoặc chọn New Query trên thanh công cụ).
- *Tạo ràng buộc:* Click phải lên Database Diagrams, chọn New Database Diagram, thiết lập các mối quan hệ, sau đó lưu lại.
- *Tạo thủ tục (Store Procedure):* Vào Programmability, click phải lên Store Procedures chọn Store Procedure... để viết các thủ tục.
- *Tạo hàm (Function):* Vào Programmability, click phải lên Function chọn New, chọn Scalar-valued Functions để viết các hàm.
- *Sao lưu và phục hồi cơ sở dữ liệu:* Click phải lên Database, chọn Task, chọn Backup hoặc Restore.

Lưu ý: Có thể sao lưu bằng cách phát sinh script chứa dữ liệu và mô hình bằng cách lich phái lên Database, chọn Task, chọn Generate Scripts:



2. Thực hành trên SQL Server

- Bước 1:** Sinh viên mở SQL Server, kết nối bằng Windows Authentication, tạo một cơ sở dữ liệu có tên RestaurantManagement.
- Bước 2:** Tạo các bảng có tên bảng, tên cột, kiểu dữ liệu như hình sau. Lưu ý: ID là kiểu int, tự tăng khi có dữ liệu.

Category *		
Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
Name	nvarchar(1000)	<input type="checkbox"/>
Type	int	<input type="checkbox"/>
		<input type="checkbox"/>

Food *		
Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
Name	nvarchar(1000)	<input type="checkbox"/>
Unit	nvarchar(100)	<input type="checkbox"/>
FoodCategoryId	int	<input type="checkbox"/>
Price	int	<input type="checkbox"/>
Notes	nvarchar(3000)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Table *		
Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
Name	nvarchar(1000)	<input checked="" type="checkbox"/>
Status	int	<input type="checkbox"/>
Capacity	int	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Bills *		
Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
Name	nvarchar(1000)	<input type="checkbox"/>
TableID	int	<input type="checkbox"/>
Amount	int	<input type="checkbox"/>
Discount	float	<input checked="" type="checkbox"/>
Tax	float	<input checked="" type="checkbox"/>
Status	bit	<input type="checkbox"/>
CheckoutDate	smalldatetime	<input checked="" type="checkbox"/>
Account	nvarchar(100)	<input type="checkbox"/>
		<input type="checkbox"/>

BillDetails *		
Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
InvoicelD	int	<input type="checkbox"/>
FoodID	int	<input type="checkbox"/>
Quantity	int	<input type="checkbox"/>
		<input type="checkbox"/>

RoleAccount *			
Column Name	Data Type	Allow Nulls	
RoleId	int	<input type="checkbox"/>	
AccountName	nvarchar(100)	<input type="checkbox"/>	
Actived	bit	<input type="checkbox"/>	
Notes	nvarchar(3000)	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

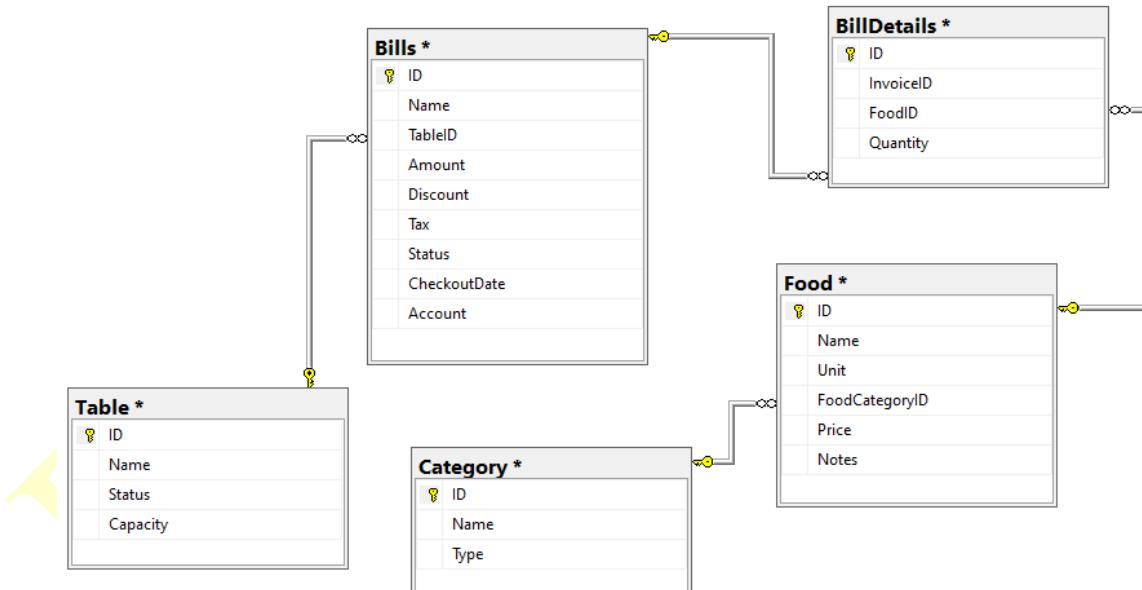
Role *			
Column Name	Data Type	Allow Nulls	
ID	int	<input type="checkbox"/>	
RoleName	nvarchar(1000)	<input type="checkbox"/>	
Path	nvarchar(3000)	<input checked="" type="checkbox"/>	
Notes	nvarchar(3000)	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

Account *			
Column Name	Data Type	Allow Nulls	
AccountName	nvarchar(100)	<input type="checkbox"/>	
Password	nvarchar(200)	<input type="checkbox"/>	
FullName	nvarchar(1000)	<input type="checkbox"/>	
Email	nvarchar(1000)	<input checked="" type="checkbox"/>	
Tell	nvarchar(200)	<input checked="" type="checkbox"/>	
DateCreated	smalldatetime	<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

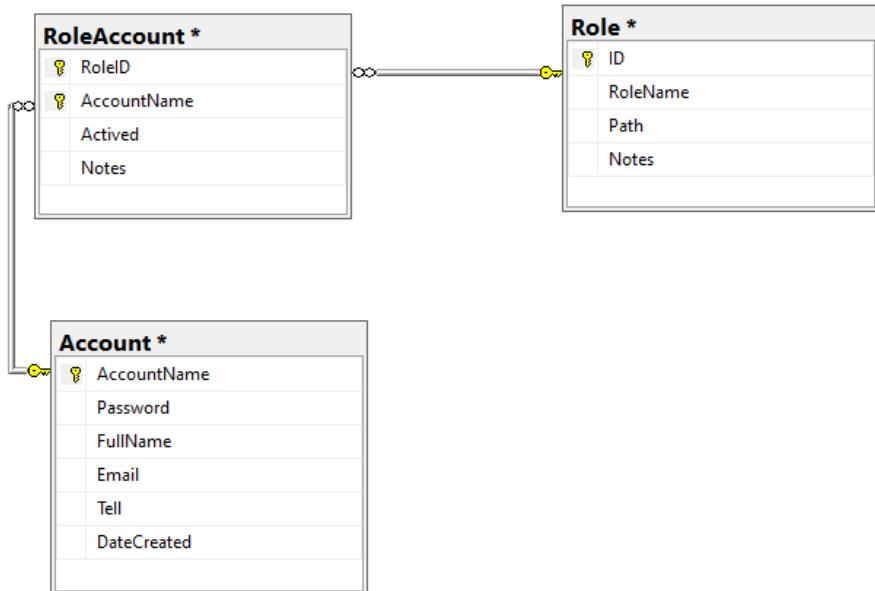
Tìm hiểu tác dụng của các bảng trên, sinh viên có thể hỏi giảng viên hướng dẫn để hiểu mô hình này muốn thể hiện điều gì.

- **Bước 3:** Tạo lược đồ quan hệ, click phải lên Database Diagrams, chọn New Database Diagrams, sau đó tạo 2 lược đồ như sau:

Lược đồ 1: Lược đồ chức năng, đặt tên là *Functional diagram*:



Lược đồ 2: Lược đồ phân quyền, đặt tên là *Role Diagram*:



- Bước 4:** Nhập dữ liệu cho các bảng theo thứ tự từ bảng 1 đến bảng nhiều.
- Bước 5:** Thực hiện truy vấn: Chọn New Query trên thanh công cụ, thực hiện các truy vấn như sau (xem hình ở dưới):
 - Lấy hết thông tin bằng lệnh `Select * from [Table]`
 - Lấy thông tin có điều kiện bằng lệnh: `Select * from [Table] where [column] = value`
 - Thêm dữ liệu vào bảng bằng lệnh: `Insert into [Table] values (...)`
 - Cập nhật dữ liệu bằng lệnh: `Update [Table] Set ...`
 - Xoá dữ liệu bằng lệnh: `Delete from [Table]`
- Bước 6:** Viết các thủ tục: Mỗi bảng sinh viên viết 05 thủ tục theo cấu trúc như sau:
 $[Table]_[TenThuTuc]$

Các thủ tục bao gồm:

- `[Table]_GetAll`: Lấy hết dữ liệu của bảng
- `[Table]_GetByID`: Lấy thông tin dữ liệu bảng theo ID (khóa ngoại)
- `[Table]_Insert`: Chèn dữ liệu vào bảng
- `[Table]_Update`: Cập nhật dữ liệu bảng
- `[Table]_Delete`: Xóa dữ liệu bảng theo khóa ngoại

Sau khi viết xong thủ tục nào thì nhấn F5 để SQL Server ghi vào hệ thống.

Ví dụ: Với bảng Category, các thủ tục được viết như các hình sau:

- `Category_GetAll`:

```

-- Thủ tục lấy hết dữ liệu trong bảng Category
CREATE PROCEDURE Category_GetAll
AS
BEGIN
    SELECT * FROM dbo.Category
END
GO
    
```

- Category_GetByID:

```
-- Thủ tục lấy hết dữ liệu trong bảng Category theo ID
CREATE PROCEDURE Category_GetAll
(
    @ID INT
)
AS
BEGIN
    SELECT * FROM dbo.Category WHERE ID = @ID
END
GO
```

- Category_Insert:

```
-- Thủ tục thêm dữ liệu vào bảng Category
CREATE PROCEDURE Category_Insert
(
    @Name NVARCHAR(1000),
    @Type INT
)
AS
BEGIN
    -- Kiểm tra tồn tại Name: Lệnh này có thể không cần thiết trong một số bài
    IF (NOT EXISTS (SELECT Name FROM dbo.Category WHERE Name = @Name))
        INSERT INTO dbo.Category (Name, Type)
                    VALUES (@Name, @Type)
END
GO
```

- Category_Update:

```
-- Thủ tục cập nhật dữ liệu trong bảng Category
CREATE PROCEDURE Category_Update
(
    @ID INT,
    @Name NVARCHAR(1000),
    @Type INT
)
AS
BEGIN
    UPDATE dbo.Category
    SET [Name] = @Name, [Type] = @Type
    WHERE ID = @ID
END
GO
```

- Category_Delete:

```
-- Thủ tục xóa mẫu tin trong bảng Category
CREATE PROCEDURE Category_Delete
(
    @ID INT
)
AS
BEGIN
    DELETE FROM dbo.Category
    WHERE ID = @ID
END
GO
```

Dùng lệnh EXEC để gọi thủ tục trên như sau:

```
EXEC dbo.Category_Insert N'Tráng miệng', 1
```

Hoặc: `EXEC dbo.Category_GetAll`

Lưu ý: Thủ tục Insert nếu cần trả về ID nào vừa mới thêm, có thể viết lại như sau:

```
ALTER PROCEDURE dbo.Category_Insert_
(
    @ID INT OUTPUT,
    @Name NVARCHAR(1000),
    @Type INT
)
AS
BEGIN
    -- Kiểm tra tồn tại Name
    IF (NOT EXISTS (SELECT Name FROM dbo.Category WHERE Name = @Name))
        INSERT INTO dbo.Category (Name, Type)
        VALUES (@Name, @Type)
        SET @ID = @@IDENTITY
END
```

Khi đó, ta gọi thủ tục để kiểm tra như sau:

```
DECLARE @ID INT = 0;
EXEC dbo.Category_Insert_ @ID = @ID OUTPUT,
                        @Name = N'Món rau',
                        @Type = 1
SELECT * FROM dbo.Category WHERE ID = @ID
```

ID	Name	Type
1	Món rau	1

Tương tự như vậy, thủ tục GetAll có thể viết chung cho tất cả các bảng (truyền vào tên bảng) như sau:

```
-- Thủ tục lấy tất cả mẫu tin theo tên bảng
CREATE PROCEDURE [dbo].[_GetAll]
(
    @TableName NVARCHAR(200)
)
AS
BEGIN
    -- Khai báo chuỗi và gán chuỗi
    DECLARE @Sql NVARCHAR(1000)
    SET @Sql = 'Select * from '+@TableName
    EXEC (@Sql) -- Thực thi
END
```

Khi đó, gọi thủ tục như sau:

`EXEC dbo._GetAll 'Category'`

hoặc `EXEC dbo._GetAll 'Food'`

- **Bước 7:** Viết hàm

Có nhiều trường hợp cần tính toán, SQL Server cho phép viết hàm để tính, hàm sẽ trả về kiểu dữ liệu (số, ngày tháng, chuỗi...) hoặc trả về bảng. Hàm được tạo ra bằng cách vào Programability, chọn Functions, click phải lên Scalar-valued Function, chọn New Scalar-valued Function và bắt đầu viết hàm.

Ngoài ra, người dùng có thể sử dụng một số hàm có sẵn của SQL Server.

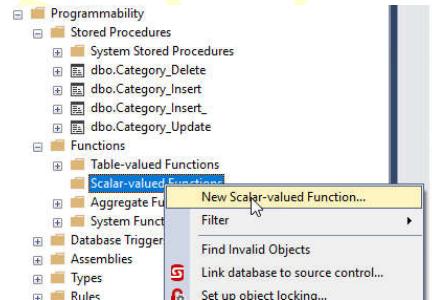
Chi tiết về hàm, sinh viên xem thêm tại đây:

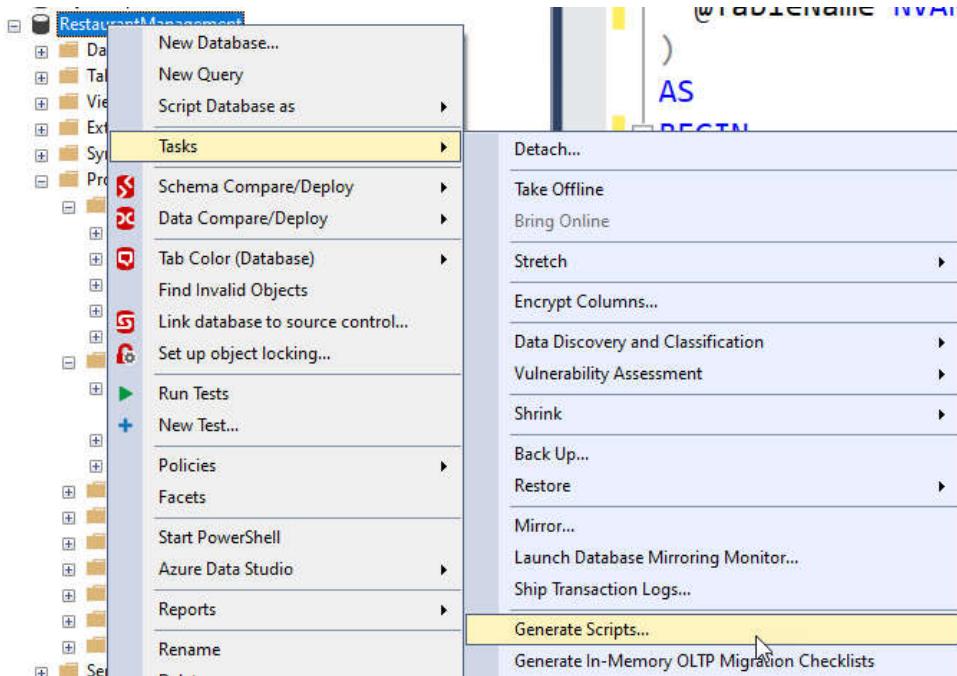
<http://giasutinhoc.vn/labs/lab-sql-server/huong-dan-tao-function-trong-sql-server/>

- **Bước 8:** Phát sinh Script

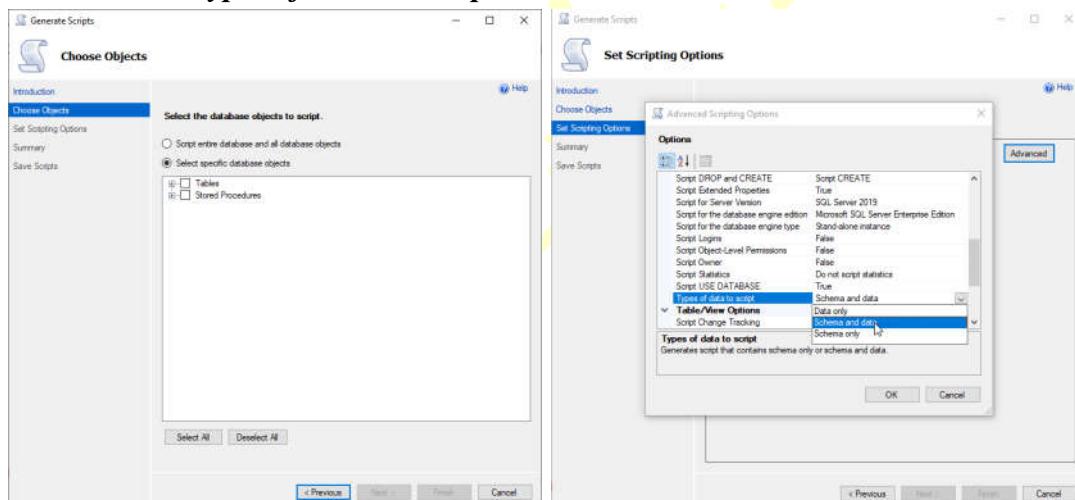
Sau khi đã tạo Database, tạo bảng, viết thủ tục, viết hàm, sinh viên có thể cho phát sinh Script để sử dụng cho lần sau. Cách phát sinh Script như sau:

- Click phải lên Database, chọn Task, chọn Generate Script:





- Chọn **Next**, chọn **Select specific database objects**, sau đó đánh dấu check hết Table, View, Store Procedure, Functions... và nhấn **Next**. Chọn **Advanced**, tìm đến **Types of Data to Script**, chọn **Schema and Data**



- Đặt tên cho Script và nhấn **Finish**. Sau đó giữ lại Script này.

III. Bài tập

1. Viết hết các thủ tục Insert, Update, Delete cho tất cả các bảng nêu trên.
2. Viết một thủ tục _GetAll để lấy dữ liệu của tất cả các bảng, truyền vào tên bảng.
3. Viết một thủ tục _GetByID để lấy dữ liệu của tất cả các bảng có ID là kiểu int, khóa chính và tự tăng. Tham số truyền vào ID và tên bảng.
4. Viết thủ tục _Delete để xóa dữ liệu của bất kỳ bảng nào có ID là kiểu int, khóa chính và tự tăng. Tham số truyền vào là ID và tên bảng.
5. Viết thủ tục để khi thêm quyền vào bảng Role thì tự động gán hết quyền cho các User (Insert vào bảng UserRole, nhưng để Active = false).
6. Viết hàm tính số tiền bán được theo ngày
7. Viết hàm đếm số lượng món ăn bán được theo ngày
8. Viết thủ tục thống kê số tiền bán được theo từng loại món ăn, theo ngày
9. *Viết thủ tục nhập hai bàn làm một
- 10.*Viết thủ tục tách bàn

LAB 2 - WINDOWS FORM – CÁC CONTROL CƠ BẢN

Thời lượng: 4 tiết

I. Mục tiêu

Sử dụng các control cơ bản để xây dựng ứng dụng.

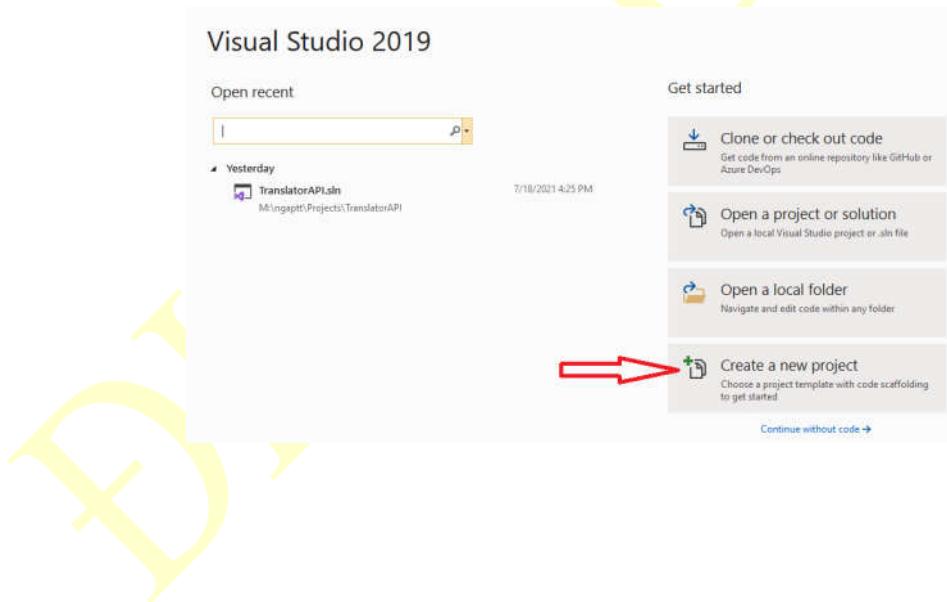
- ☞ Label
- ☞ TextBox, ComboBox, MaskedTextBox
- ☞ Button, RadioButton, CheckBox, CheckListBox
- ☞ ListBox

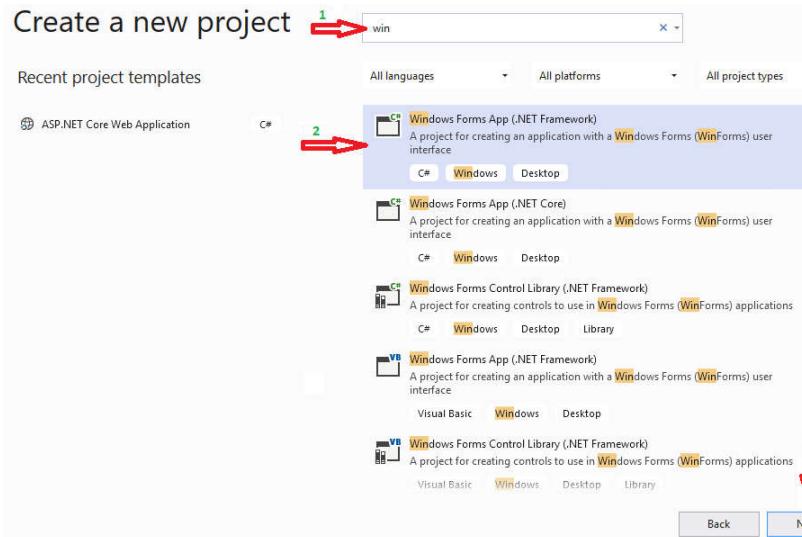
II. Hướng dẫn thực hành

1. Hướng dẫn: Tạo project Windows Application

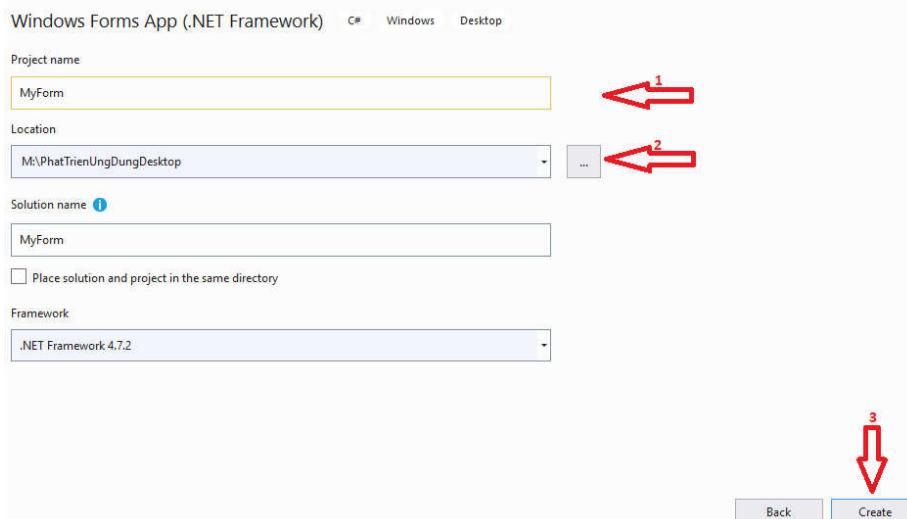
Bước 1: Khởi động Microsoft Visual Studio bằng cách click vào biểu tượng  trên desktop hoặc thanh task bar

Bước 2: Tạo mới Project và chọn Windows Forms App (.NET Framework), đặt tên Project là MyForm

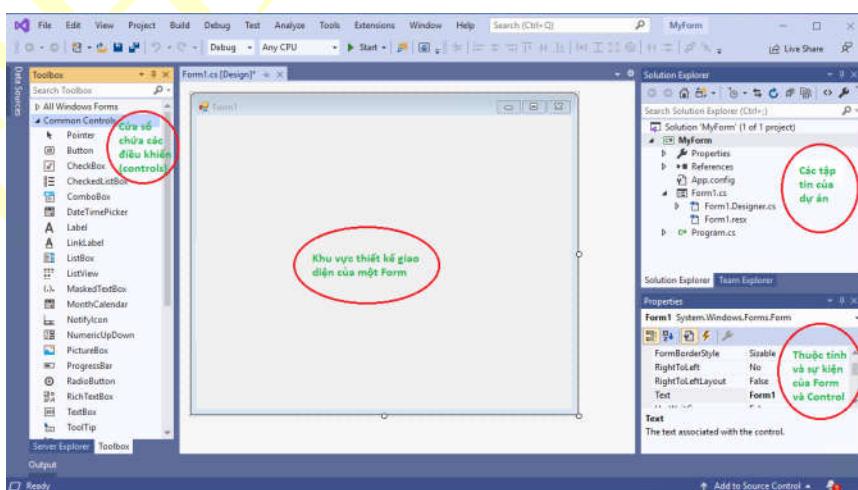




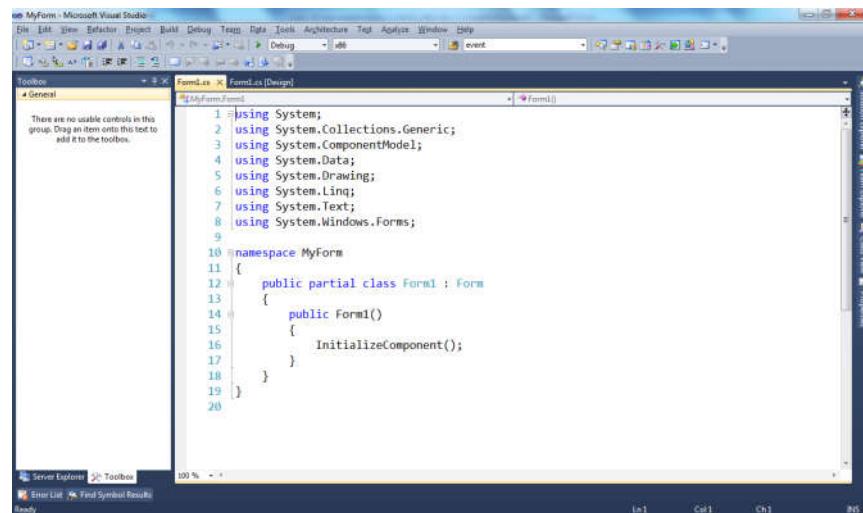
Configure your new project



Bước 3: Giao diện chính của chương trình

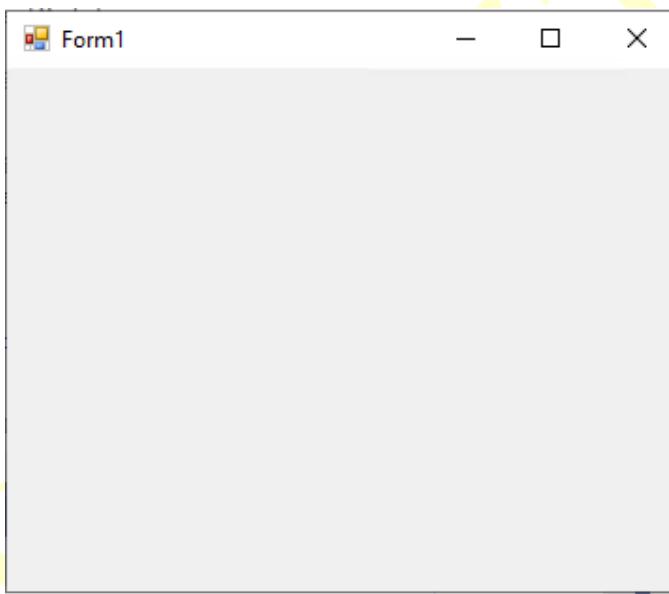


Bước 4: Để viết code Click chuột phải trên Form chon View Code hoặc nhấn F7 hoặc vào menu View → Code



```
MyForm - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Tools Data Architecture Test Analyze Window Help
File Edit View Refactor Project Build Debug Tools Data Architecture Test Analyze Window Help
Server Explorer Toolbox Error List Find Symbol Results Ready
Form1.cs [Design]
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;
namespace MyForm
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Bước 5: Nhấn Ctrl+F5 để chạy chương trình. Kết quả chạy chương trình.



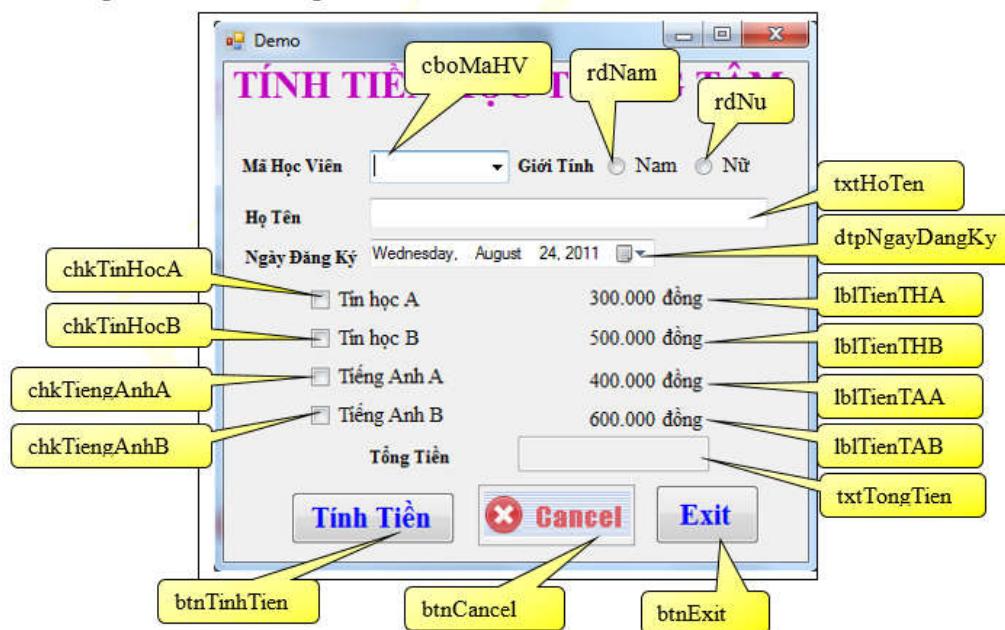
2. Ví dụ 1:

Mở Visual Studio .NET và tạo project đặt tên Lab2_Demo.Thiết kế Form có giao diện sau:



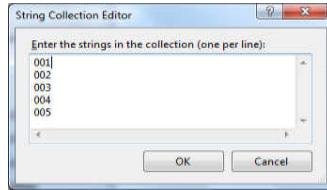
Bước 1: Thiết kế Form và đặt tên các controls như hướng dẫn:

Ý nghĩa đặt tên control: LoạiControlThôngTin. Ví dụ tên control: **cboMaHV**. Loại control sử dụng là **ComboBox** thông tin của Mã Học Viên



Bảng mô tả thông tin của các control

STT	Name	Loại Control	Tên Thuộc Tính	Giá Trị
1	frmTrungTam	Form	Text	Simple Drawing

2	cboMaHV	ComboBox	Items	
3	rdNam	RadioButton		
4	rdNu	RadioButton		
5	txtHoTen	TextBox		
6	chkTinHocA	CheckBox	Text	Tin học A
7	chkTinHocB	CheckBox	Text	Tin học B
8	chkTiengAnhA	CheckBox	Text	Tiếng Anh A
9	chkTiengAnhB	CheckBox	Text	Tiếng Anh B
10	lblTienTHA	Label	Text	300.000 đồng
11	lblTienTHB	Label	Text	500.000 đồng
12	lblTienTAA	Label	Text	400.000 đồng
13	lblTienTAB	Label	Text	600.000 đồng
14	txtTongTien	TextBox	Enable	False
15	btnTinhTien	Button	Text	Tính Tiền
16	btnCancel	Button	Image	Đường dẫn hình
17	btnExit	Button	Text	Exit

Chi tiết thông tin Lớp frmTrungTam

frmTrungTam

Class
→ Form

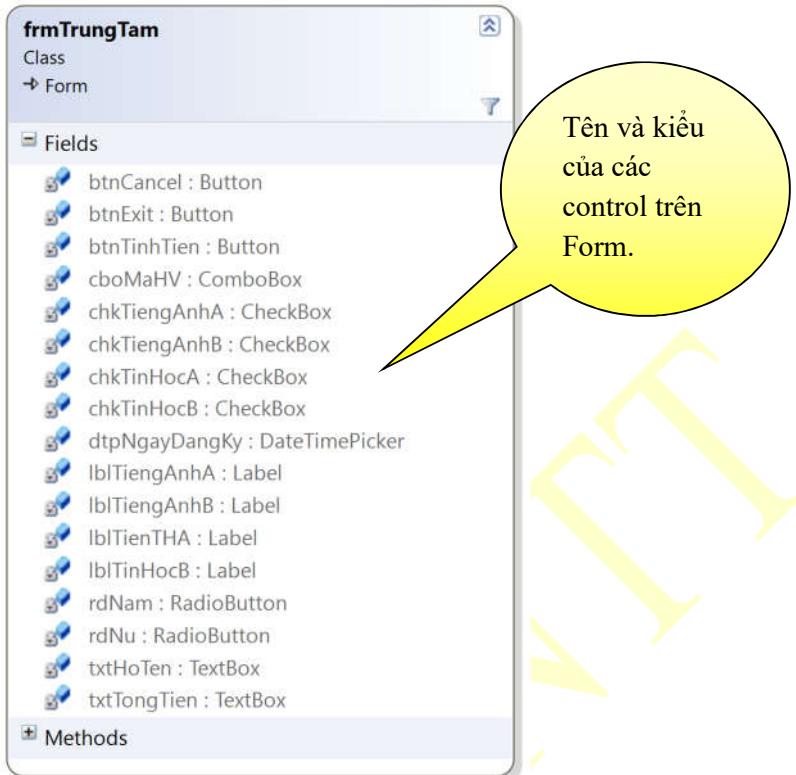
Fields

Methods

Các phương thức trên Form.

```

        btnCancel_Click(object sender, EventArgs e) : void
        btnExit_Click(object sender, EventArgs e) : void
        btnTinhTien_Click(object sender, EventArgs e) : void
        Dispose(bool disposing) : void
        frmTrungTam()
        InitializeComponent() : void
        ReSet() : void
    
```



Bước 2: Thực hiện gán Tab order cho các control trên Form:

- Vào View → Tab Order → click chuột lên số để thiết lập Tab theo thứ tự sau:

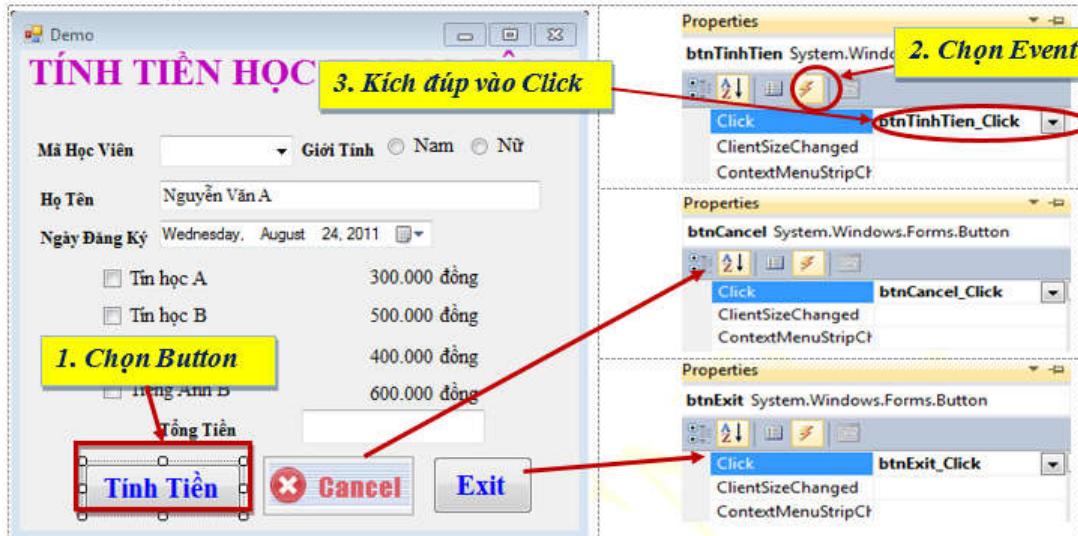


- Để quay lại màn hình thiết kế Form thực hiện vào View → Tab Order.

Bước 3: Viết Code chương trình.

Bước 3.1: Viết code cho sự kiện Click của 3 Button: btnTinhTien, btnCancel, btnExit. Để chọn sự kiện cho Button có 2 cách:

Cách 1: chọn button cần tạo trình xử lý, sau đó kích tab event trong cửa sổ Properties, kích đúp vào mục Click trong cửa sổ event.



Cách 2: Kích đúp vào button cần tạo trình xử lý sự kiện trong màn hình Form design view: khi đó VS sẽ tạo trình xử lý sự kiện gắn với sự kiện Click của button “Tính Tiền” hoặc Cancel, Exit.

Bước 3.2: Viết code cho các sự kiện trên tham khảo lớp **frmTrungTam**:

```
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;

namespace Lab2
{
    public partial class frmTrungTam : Form
    {
        public frmTrungTam()
        {
            InitializeComponent();
        }

        private void btnExit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Tắt chương trình

```

private void btnCancel_Click(object sender, EventArgs e)
{
    this.ReSet();
}
private void ReSet()
{
    this.cboMaHV.Text = "";
    this.txtHoTen.Text = "";
    this.dtpNgayDangKy.Value = DateTime.Now;
    this.rdnam.Checked = true;
    this.chkTiengAnhA.Checked = false;
    this.chkTiengAnhB.Checked = false;
    this.chkTinHocA.Checked = false;
    this.chkTinHocB.Checked = false;
    this.txtTongTien.Text = "";
}
private void btnTinhTien_Click(object sender, EventArgs e)
{
    int s = 0;
    if (chkTinHocA.Checked)
        s += int.Parse(lblTienTHA.Text.Split('.')[0]);
    if (chkTinHocB.Checked)
        s += int.Parse(lblTienHocB.Text.Split('.')[0]);
    if (chkTiengAnhA.Checked)
        s += int.Parse(lblTiengAnhA.Text.Split('.')[0]);
    if (chkTiengAnhB.Checked)
        s += int.Parse(lblTiengAnhB.Text.Split('.')[0]);
    this.txtTongTien.Text = s + ".000 đồng";
}

```

Thiết lập giá trị mặc định cho các control

Tính tổng tiền học

Xem file *Program.cs*

```

namespace Lab2
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmTrungTam());
        }
    }
}

```

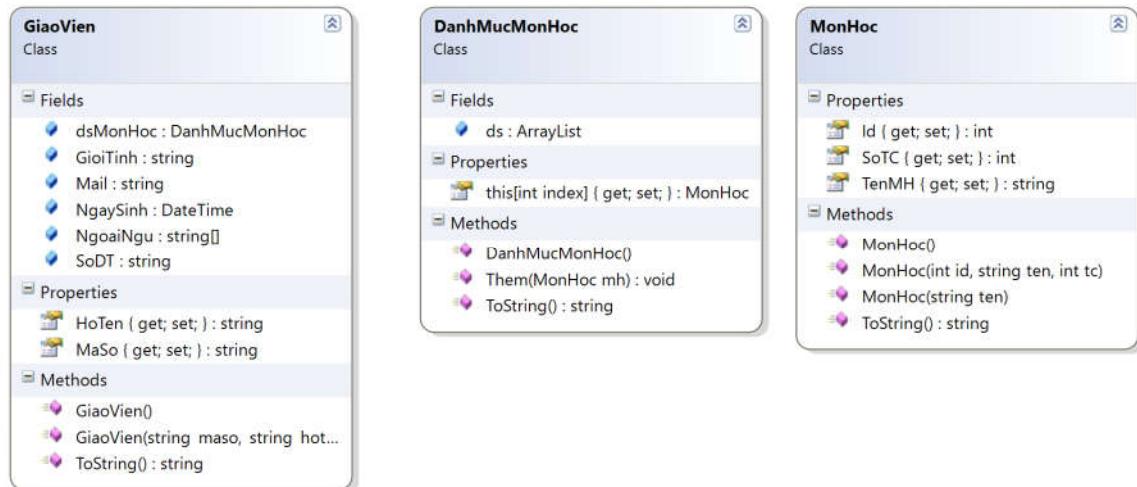
Câu lệnh chạy chương trình load form: frmTrungTam

Bước 4: Nhấn Ctrl+F5 để chạy chương trình



3. Ví dụ 2:

- Cho lược đồ sau:



Lớp GiaoVien:

```
public class GiaoVien
{
    public string MaSo { get; set; }
    public string HoTen { get; set; }
    public DateTime NgaySinh;
    public DanhMucMonHoc dsMonHoc;
    public string GioiTinh;
    public string[] NgoaiNgu;
    public string SoDT;
    public string Mail;
    public GiaoVien()
    {
        dsMonHoc = new DanhMucMonHoc();
        NgoaiNgu = new string[10];
    }
    public GiaoVien(string maso, string hoten, DateTime ngaysinh,
                    DanhMucMonHoc ds, string gt, string []nn,
                    string sdt, string mail)
    {
        this.MaSo = maso;
        this.HoTen = hoten;
        this.NgaySinh = ngaysinh;
        this.dsMonHoc = ds;
        this.GioiTinh = gt;
        this.NgoaiNgu = nn;
        this.SoDT = sdt;
        this.Mail = mail;
    }
    public override string ToString()
    {
        string s = "Mã số:" + MaSo + "\n" + "Họ tên:" + HoTen + "\n"
                  + "Ngày Sinh:" + NgaySinh.ToString() + "\n"
                  + "Giới tính:" + GioiTinh + "\n"
                  + "Số ĐT:" + SoDT + "\n"
                  + "Mail:" + Mail + "\n";
        string sngoaingu = "Ngoại ngữ:";
        foreach (string t in NgoaiNgu)
            sngoaingu += t + ";";
        string Monday = "Danh sách môn dạy:";
        foreach (MonHoc mh in dsMonHoc.ds)
            Monday += mh + ";";
        s += "\n" + sngoaingu + "\n" + Monday;
        return s;
    }
}
```



Lớp MonHoc	Lớp DanhMucMonHoc
<pre>public class MonHoc { public int Id { get; set; } public string TenMH { get; set; } public int SoTC { get; set; } public MonHoc() { } public MonHoc(string ten) { this.TenMH = ten; } public MonHoc(int id, string ten, int tc) { this.Id = id; this.TenMH = ten; this.SoTC = tc; } public override string ToString() { return TenMH ; } }</pre>	<pre>public class DanhMucMonHoc { public ArrayList ds; public DanhMucMonHoc() { ds = new ArrayList(); } public MonHoc this[int index] { get { return ds[index] as MonHoc; } set { ds[index] = value; } } public void Them(MonHoc mh) { ds.Add(mh); } public override string ToString() { string s="Danh sách mon hoc:"; foreach (object mh in ds) { s += mh as MonHoc + ";"; } return s; } }</pre>

- Thiết kế form sau: Đặt tên frmGiaoVien

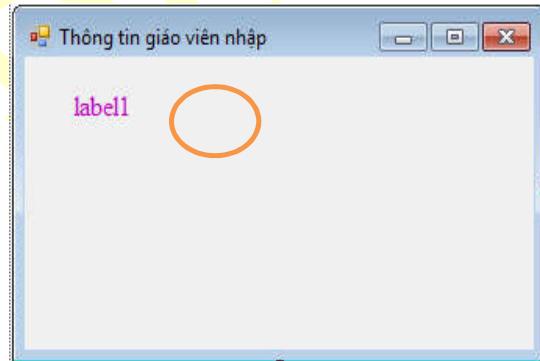


Bảng mô tả thông tin control cho frmGiaoVien

STT	Name	Loại Control	Tên Thuộc Tính	Giá Trị
1	cboMaSo	ComboBox	Items	001

				002 003 004
2	rdNam	CheckBox	Checked	true
3	rdNu	CheckBox		
4	txtHoTen	TextBox		
5	mtxtSoDT	MaskedTextBox	Mask	(0633).000.000
6	dtpNgaySinh	DateTimePicker	CustomFormat Format	dd/MM/yyyy Custom
7	txtMail	TextBox		
8	chklbNgoaiNgu	CheckListBox	Items	Tiếng Anh Tiếng Pháp Tiếng Nhật Tiếng Nga
9	lbDanhSachMH	ListBox	Items SelectionMode	“DS môn học như trên Form” MultiExtended
10	lbMonHocDay	ListBox	ListBox	One
11	btnChon	Button	Text	>>
12	btnXoa	Button	Text	<<
13	btnLuu	Button	Text	Lưu
14	btnCancel	Button	Image	
15	btnExit	Button	Text	Exit
16	linklbLienHe	LinkLabel	Text	Liên hệ

- Form frmTBGiaoVien:



Bảng mô tả thông tin control cho frmTBGiaoVien

TT	Name	Loại Control
1	lblThongBao	Label

Code xử lý frmTBGiaoVien:

```

namespace Lab2
{
    public partial class frmTBGiaoVien : Form
    {
        public frmTBGiaoVien()
        {
            InitializeComponent();
        }
        public void SetText(string s)
        {
            this.lblThongBao.Text = s;
        }
    }
}

```

Code xử lý cho frmGiaoVien:

```

public partial class frmGiaoVien : Form
{
    public frmGiaoVien()
    {
        InitializeComponent();
    }
    private void frmGiaoVien_Load(object sender, EventArgs e)
    {
        string lienhe = "http://it.dlu.edu.vn/e-learning/Default.aspx";
        this.linkLabelLienHe.Links.Add(0, lienhe.Length, lienhe);
        this.cboMaSo.SelectedItem = this.cboMaSo.Items[0];
    }
    private void btnChon_Click(object sender, EventArgs e)
    {
        int i = this.lbDanhMucMH.SelectedItems.Count - 1;
        while (i >= 0)
        {
            this.lbMonHocDay.Items.Add(lbDanhMucMH.SelectedItems[i]);
            this.lbDanhMucMH.Items.Remove(lbDanhMucMH.SelectedItems[i]);
            i--;
        }
    }
    private void btnXoa_Click(object sender, EventArgs e)
    {
        int i = this.lbMonHocDay.SelectedItems.Count - 1;
        while (i >= 0)
        {
            this.lbDanhMucMH.Items.Add(lbMonHocDay.SelectedItems[i]);
            this.lbMonHocDay.Items.Remove(lbMonHocDay.SelectedItems[i]);
            i--;
        }
    }
}

```

```
private void btnCancel_Click(object sender, EventArgs e)
{
    Reset();
}
public void Reset()
{
    this.cboMaSo.Text = "";
    this.txtHoTen.Text = "";
    this.txtMail.Text = "";
    this.mtxtSoDT.Text = "";
    this.rdBam.Checked = true;
    //Bỏ chọn trên chklbNgoaiNgu
    for (int i = 0; i < chklbNgoaiNgu.Items.Count - 1; i++)
        chklbNgoaiNgu.SetItemChecked(i, false);
    //Chuyển các môn từ lbMonHocDay sang lbDanhMucMH
    foreach (object ob in this.lbMonHocDay.Items)
        this.lbDanhMucMH.Items.Add(ob);
    this.lbMonHocDay.Items.Clear();
}
private void linklblLienHe_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
{
    string s = e.Link.LinkData.ToString();
    Process.Start(s);
}
private void btnLuu_Click(object sender, EventArgs e)
{
    frmTBGiaoVien frm = new frmTBGiaoVien();
    frm.SetText(GetGiaoVien().ToString());
    frm.ShowDialog();
}
public GiaoVien GetGiaoVien()
{
    string gt = "Nam";
    if (rdNu.Checked)
        gt = "Nữ";
    GiaoVien gv = new GiaoVien();
    gv.MaSo = this.cboMaSo.Text;
    gv.GioiTinh = gt;
    gv.HoTen = this.txtHoTen.Text;
    gv.NgaySinh = this.dtpNgaySinh.Value;
    gv.Mail = this.txtMail.Text;
    gv.SoDT = this.mtxtSoDT.Text;
    //Lấy thông tin ngoại ngữ
    string ngoaingu="";
    for(int i=0; i<chklbNgoaiNgu.Items.Count-1; i++)
        if(chklbNgoaiNgu.GetItemChecked(i))
            ngoaingu += chklbNgoaiNgu.Items[i] + ";";
    gv.NgoaiNgu = ngoaingu.Split(';');
```

```

//Lay thong tin danh sach mon hoc
DanhMucMonHoc mh = new DanhMucMonHoc();
foreach(object ob in lbMonHocDay.Items)
    mh.Them(new MonHoc(ob.ToString()));
gv.dsMonHoc = mh;

return gv;
}
}

```

III. Bài tập

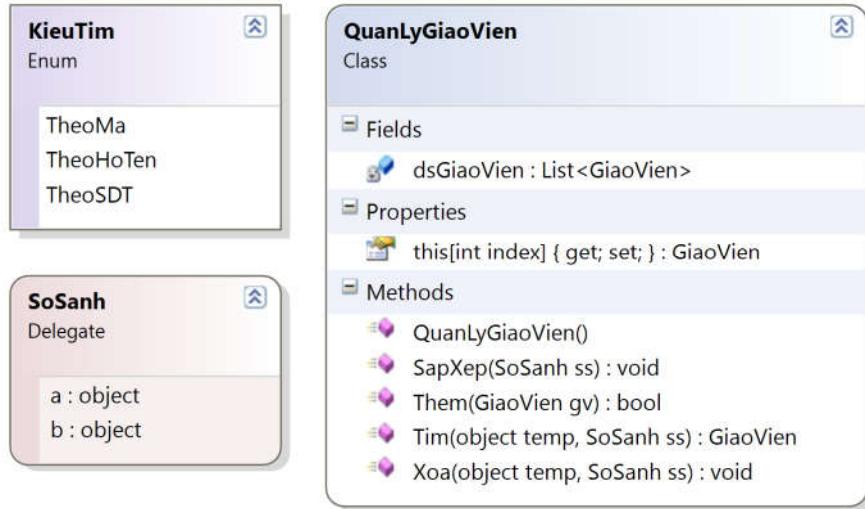
Thêm chức năng cho ví dụ 2 để xây dựng form như sau:



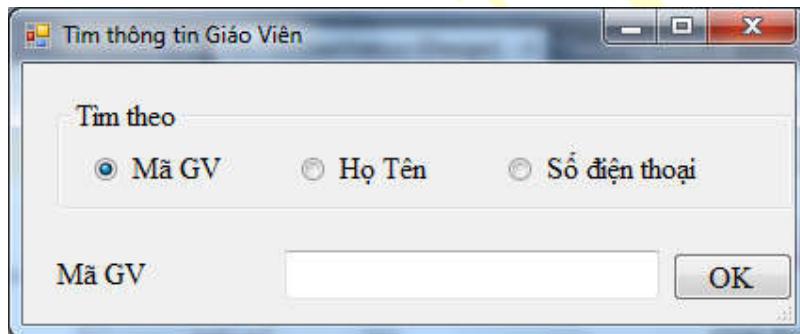
- **Chức năng 1:** Thêm button Thêm xây dựng lớp QuanLyGiaoVien. Khi click vào thêm thực hiện: Thêm dữ liệu giáo viên trên form cho danh sách giáo viên (mỗi giáo viên chỉ có 1 mã duy nhất). Nếu thêm giáo viên có mã trong danh sách thông báo người dùng:



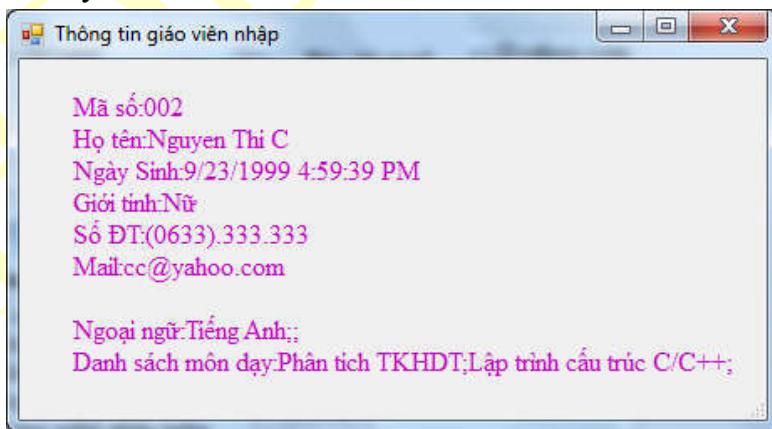
Viết theo lớp sau:



- Chức năng 2: Button Tìm có chức năng:



Khi nhấn OK thì xuất frmTBGiaoVien thông tin của Giáo Viên
Nếu tìm thấy:



Không tìm thấy:



LAB 3 - WINDOWS FORM – XỬ LÝ SỰ KIỆN

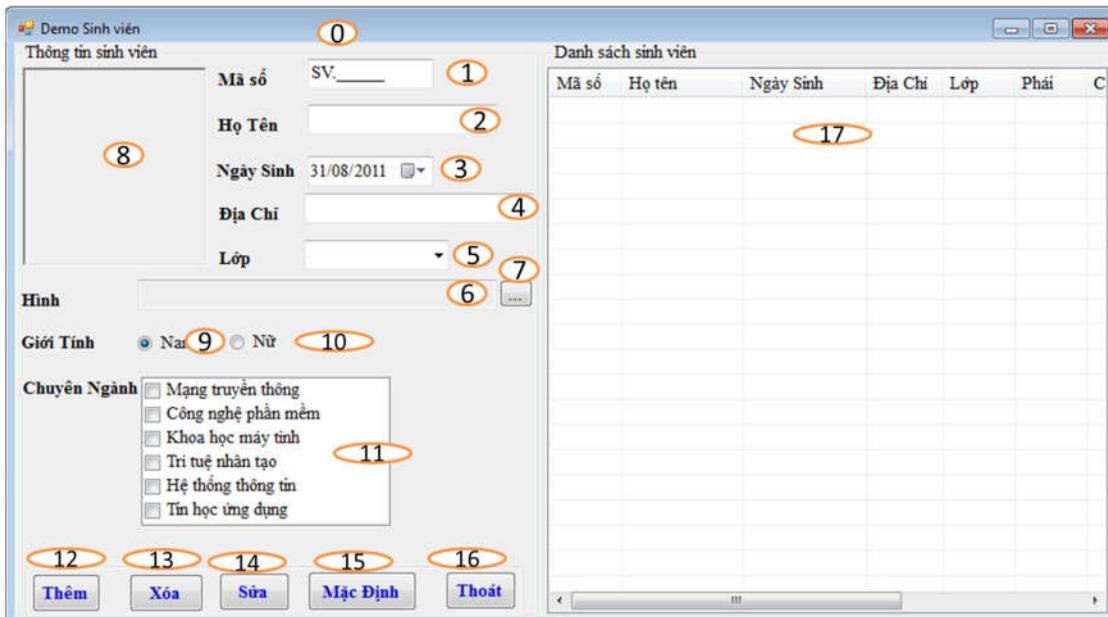
Thời lượng: 8 tiết

I. Mục tiêu

- Sử dụng các control cơ bản để xây dựng ứng dụng.
- Nắm bắt các sự kiện cơ bản

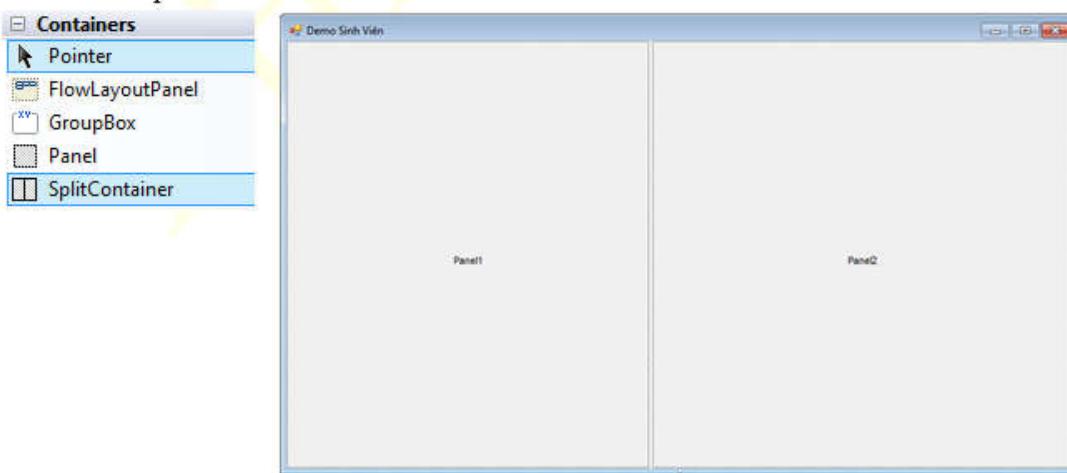
II. Hướng dẫn thực hành

Mở Visual Studio .NET và tạo project đặt tên Lab3_Demo. Thực hiện các bước như sau:

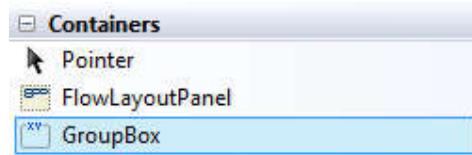


Thiết kế theo các bước sau:

1. Add control SplitContainer:



2. Trong Panel1: Add control GroupBox1; Panel2: GroupBox2



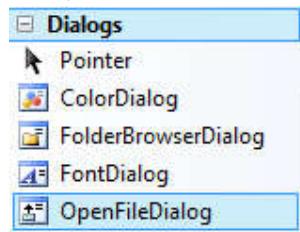
Thuộc tính GroupBox1:

STT	Thuộc tính	Giá trị
1	Dock	Fill
2	Text	Thông tin sinh viên

Thuộc tính GroupBox2:

STT	Thuộc tính	Giá trị
1	Dock	Fill
2	Text	Danh sách sinh viên

3. Add OpenFileDialog:



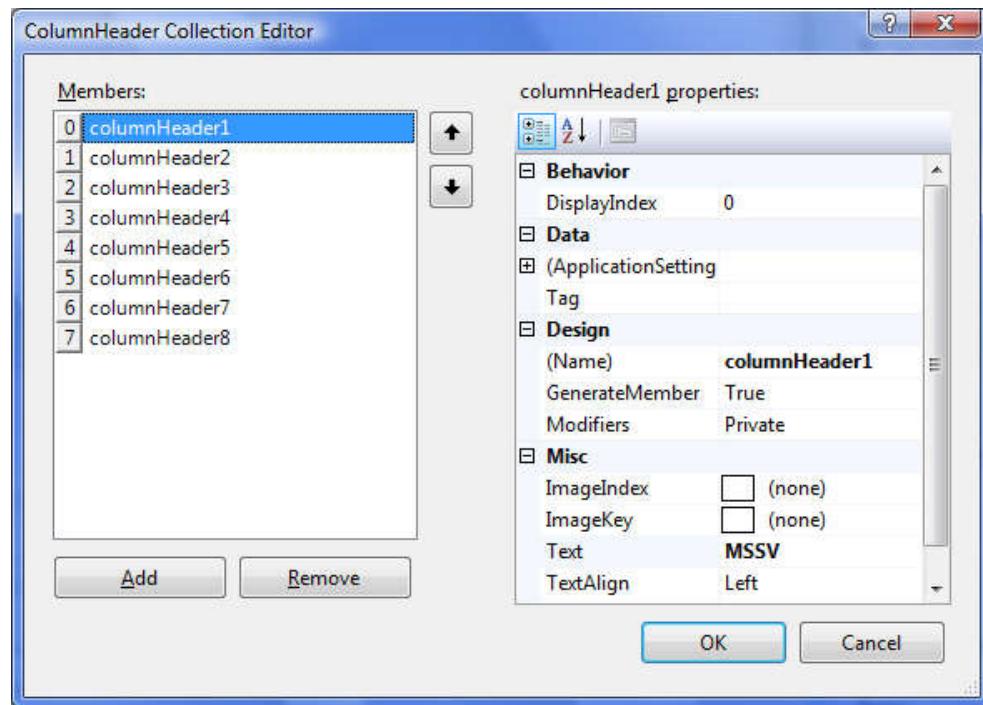
STT	Thuộc tính	Giá trị
1	(Name)	OpenFileDialog1
2	Filter	File GIF*.Gif File JPEG *.Jpg

4. Thiết kế giao diện với bảng mô tả các control như sau:

STT	Name	Loại Control	Tên Thuộc Tính	Giá Trị
0	frmSinhVien	Form	Text	Demo Sinh viên
1	mtxtMaSo	MarkedTextBox	Mask	SV.00000
2	txtHoTen	TextBox		
3	dtpNgaySinh	DateTimePicker	Format	Custom
			CustomFormat	dd/MM/yyyy
4	txtDiaChi	TextBox		
5	cboLop	ComboBox	Items	CTK31 CTK32 CTK33 CTK34 CTK32CD CTK33CD CTK34CD
6	txtHinh	TextBox	ReadOnly	True

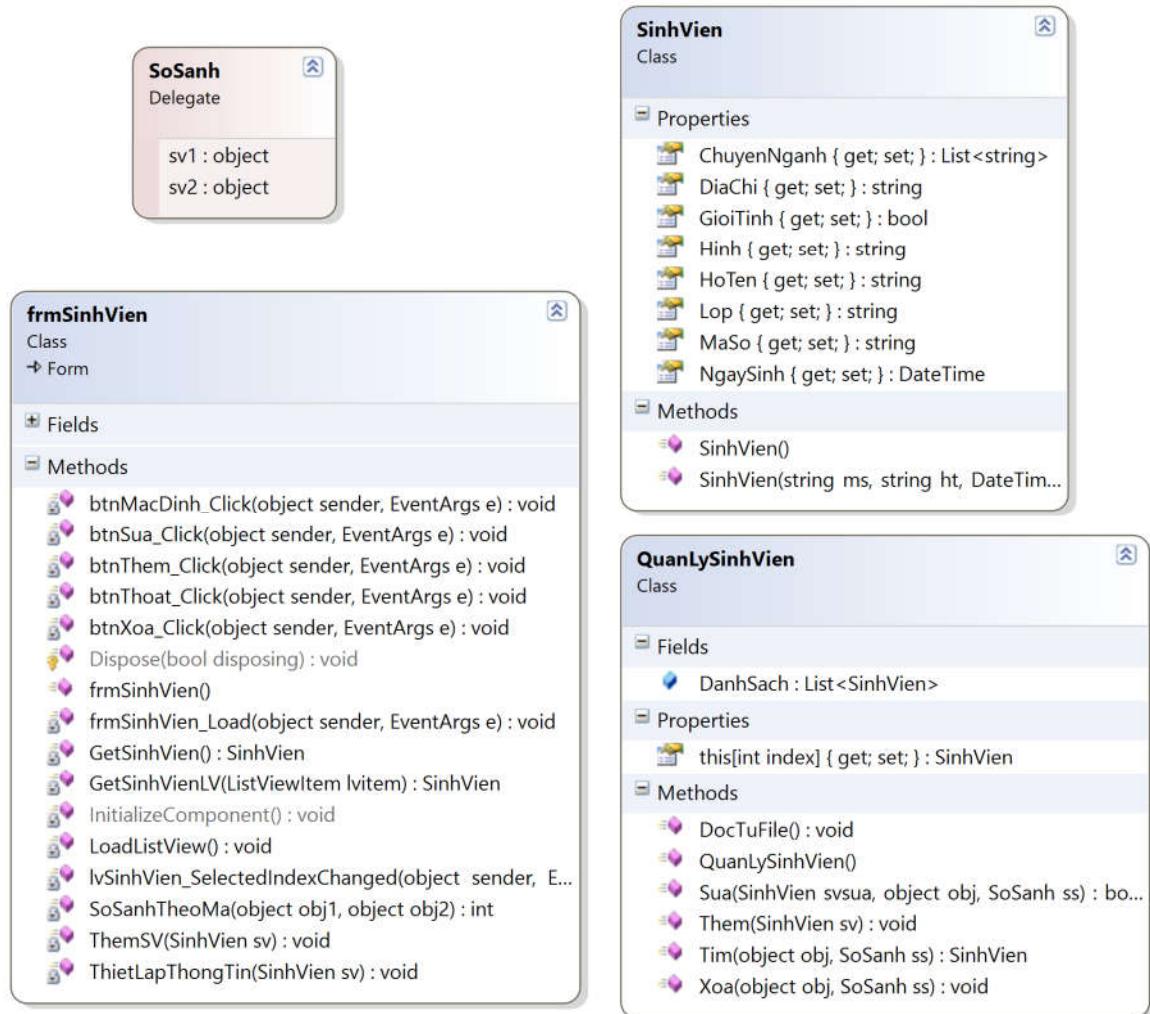
7	btnBrowse	Button	Text	...
			Chức năng	Hiển thị hộp thoại chọn hình
8	pbHinh	PictureBox	BorderStyle	Fixed3D
			SizeMode	StretchImage
9	rdNam	RadioButton	Checked	True
10	rdNu	RadioButton		
11	clbChuyenNganh	CheckListBox	CheckOnClick	True
			Items	Mạng truyền thông Công nghệ phần mềm Khoa học máy tính Trí tuệ nhân tạo Hệ thống thông tin Tin học ứng dụng
12	btnThem	Button	Text	Thêm
			Chức năng	Thêm sinh viên đã nhập
13	btnXoa	Button	Text	Xoá
			Chức năng	Xóa SV check trên ListView
14	btnSua	Button	Text	Sửa
			Chức năng	Sửa SV chọn trên ListView
15	btnMacDinh	Button	Text	Mặc Định
			Chức năng	Reset lại các controls
16	btnThoat	Button	Text	Thoát
			Chức năng	Thoát chương trình
17	lvSinhVien	ListView	CheckBox	True
			Dock	Fill
			GridLine	True
			View	Details
			Columns	Như hướng dẫn bước 5

5. Thuộc tính Columns của control ListView:



STT	Control	Thuộc tính	Giá trị
1	ColumnHeader1	Text	Mã số
		Width	60
2	ColumnHeader2	Text	Họ tên
		Width	150
3	ColumnHeader3	Text	Ngày sinh
		Width	100
4	ColumnHeader4	Text	Địa chỉ
		Width	200
5	ColumnHeader5	Text	Lớp
		Width	60
6	ColumnHeader6	Text	Phái
		Width	60
7	ColumnHeader7	Text	Chuyên Ngành
		Width	500
8	ColumnHeader8	Text	Hình
		Width	200

6. Copy tất cả các hình và file txt nằm trong thư mục File vào thư mục bin/Debug của project đã tạo.
7. Lược đồ lớp:



Danh sách file tài nguyên chương trình trong thư mục: File

7.1. Lớp SinhVien:

```
public class SinhVien
{
    //Các thuộc tính của lớp sinh viên
    public string MaSo { get; set; }
    public string HoTen { get; set; }
    public DateTime NgaySinh { get; set; }
    public string DiaChi { get; set; }
    public string Lop { get; set; }
    public string Hinhanh { get; set; }
    public bool GioiTinh { get; set; }
    public List<string> ChuyenNganh { get; set; }

    // Phương thức tạo lập mặc định
}
```

```

public SinhVien()
{
    ChuyenNganh = new List<string>();
}

// Phương thức tạo lập có tham số
public SinhVien(string ms, string ht, DateTime ngay,
                 string dc, string lop, string hinh, bool gt,
List<string>cn)
{
    this.MaSo = ms;
    this.HoTen = ht;
    this.NgaySinh = ngay;
    this.DiaChi = dc;
    this.Lop = lop;
    this.Hinh = hinh;
    this.GioiTinh = gt;
    this.ChuyenNganh = cn;
}
}

```

7.2. Lớp QuanLySinhVien:

```

// Khai báo một delegate SoSanh
public delegate int SoSanh(object sv1, object sv2);

class QuanLySinhVien
{
    public List<SinhVien> DanhSach;

    public QuanLySinhVien()
    {
        DanhSach = new List<SinhVien>();
    }

    // Thêm một sinh viên vào danh sách
    public void Them(SinhVien sv)
    {
        this.DanhSach.Add(sv);
    }
    public SinhVien this[int index]
    {
        get { return DanhSach[index]; }
        set { DanhSach[index] = value; }
    }

    // Xóa các obj trong danh sách nếu thỏa điều kiện so sánh
    public void Xoa(object obj, SoSanh ss)
    {
        int i = DanhSach.Count - 1;
        for (; i >= 0; i--)
            if (ss( obj, this[i] ) == 0)
                this.DanhSach.RemoveAt(i);
    }
}

```

```

//Tìm một sinh viên trong danh sách thỏa điều kiện so sánh
public SinhVien Tim(object obj, SoSanh ss)
{
    SinhVien svresult=null;
    foreach (SinhVien sv in DanhSach)
        if (ss(obj, sv) == 0)
    {
        svresult = sv;
        break;
    }
    return svresult;
}

//Tìm một sinh viên trong danh sách thỏa điều kiện so sánh,
//gán lại thông tin cho sinh viên này thành sv sua
public bool Sua(SinhVien sv sua, object obj, SoSanh ss)
{
    int i, count;
    bool kq = false;
    count = this.DanhSach.Count - 1;
    for (i = 0; i < count; i++)
        if (ss(obj, this[i]) == 0)
    {
        this[i] = sv sua;
        kq = true;
        break;
    }
    return kq;
}

// Hàm đọc danh sách sinh viên từ tập tin txt
public void DocTuFile()
{
    string filename = "DanhSachSV.txt", t;
    string[] s;
    SinhVien sv;
    StreamReader sr = new StreamReader(
        new FileStream(filename,
        FileMode.Open));
    while ((t = sr.ReadLine()) != null)
    {
        s = t.Split('*');
        sv = new SinhVien();
        sv.MaSo = s[0];
        sv.HoTen = s[1];
        sv.NgaySinh = DateTime.Parse(s[2]);
        sv.DiaChi = s[3];
        sv.Lop = s[4];
        sv.Hinh = s[5];
        sv.GioiTinh = false;
        if (s[6] == "1")
            sv.GioiTinh = true;
        string[] cn = s[7].Split(',');
        foreach (string c in cn)

```

```
        sv.ChuyenNganh.Add(c);
        this.Them(sv);
    }
}
```

7.3. Lớp frmSinhVien:

- Phương thức hỗ trợ: Sinh viên tự viết
 - Phương thức Sự kiện: Sinh viên phát sinh sự kiện từ các control trên form và viết code cho sự kiện đó.

```
public partial class frmSinhVien : Form
{
    QuanLySinhVien qlsv;
    public frmSinhVien()
    {
        InitializeComponent();
    }
}
```

#region Phương thức bổ trợ

```
//Lấy thông tin từ controls thông tin SV
private SinhVien GetSinhVien()
{
    SinhVien sv = new SinhVien();
    bool gt = true;
    List<string> cn = new List<string>();
    sv.MaSo = this.txtMaSo.Text;
    sv.HoTen = this.txtHoTen.Text;
    sv.NgaySinh = this.dtpNgaySinh.Value;
    sv.DiaChi = this.txtDiaChi.Text;
    sv.Lop = this.cboLop.Text;
    sv.Hinh = this.txtHinh.Text;
    if (rdNu.Checked)
        gt = false;
    sv.GioiTinh = gt;
    for (int i = 0; i < this.clbChuyenNganh.Items.Count; i++)
        if (clbChuyenNganh.GetItemChecked(i))
            cn.Add(clbChuyenNganh.Items[i].ToString());
    sv.ChuyenNganh = cn;
    return sv;
}
//Lấy thông tin sinh viên từ dòng item của ListView
private SinhVien GetSinhVienLV(ListViewItem lvitem)
{
    SinhVien sv = new SinhVien();
    sv.MaSo = lvitem.SubItems[0].Text;
    sv.HoTen = lvitem.SubItems[1].Text;
    sv.NgaySinh = DateTime.Parse(lvitem.SubItems[2].Text);
    sv.DiaChi = lvitem.SubItems[3].Text;
    sv.Lop = lvitem.SubItems[4].Text;
    sv.GioiTinh = false;
    if (lvitem.SubItems[5].Text == "Nam")
        sv.GioiTinh = true;
}
```

```

        List<string> cn = new List<string>();
        string[] s = lvitem.SubItems[6].Text.Split(',');
        foreach (string t in s)
            cn.Add(t);
        sv.ChuyenNganh = cn;

        sv.Hinh = lvitem.SubItems[7].Text;
        return sv;
    }

//Thiết lập các thông tin lên controls sinh viên
private void ThietLapThongTin(SinhVien sv)
{
    this.mtxtMaSo.Text = sv.MaSo;
    this.txtHoTen.Text = sv.HoTen;
    this.dtpNgaySinh.Value = sv.NgaySinh;
    this.txtDiaChi.Text = sv.DiaChi;
    this.cboLop.Text = sv.Lop;
    this.txtHinh.Text = sv.Hinh;
    this.pbHinh.ImageLocation = sv.Hinh;
    if (sv.GioiTinh)
        this.rdNam.Checked = true;
    else
        this.rdNu.Checked = true;
    for (int i = 0; i < this.clbChuyenNganh.Items.Count; i++)
        this.clbChuyenNganh.SetItemChecked(i, false);
    foreach (string s in sv.ChuyenNganh)
    {
        for (int i = 0; i < this.clbChuyenNganh.Items.Count;
i++)
            if
(s.CompareTo(this.clbChuyenNganh.Items[i]) == 0)
                this.clbChuyenNganh.SetItemChecked(i,
true);
    }
}
//Thêm sinh viên vào ListView
private void ThemSV(SinhVien sv)
{
    ListViewItem lvitem = new ListViewItem(sv.MaSo);
    lvitem.SubItems.Add(sv.HoTen);
    lvitem.SubItems.Add(sv.NgaySinh.ToShortDateString());
    lvitem.SubItems.Add(sv.DiaChi);
    lvitem.SubItems.Add(sv.Lop);
    string gt = "Nữ";
    if (sv.GioiTinh)
        gt = "Nam";
    lvitem.SubItems.Add(gt);
    string cn = "";
    foreach (string s in sv.ChuyenNganh)
        cn += s + ",";
    cn = cn.Substring(0, cn.Length - 1);
    lvitem.SubItems.Add(cn);
    lvitem.SubItems.Add(sv.Hinh);
    this.lvSinhVien.Items.Add(lvitem);
}

```

```

        }
        //Hiển thị các sinh viên trong qlsv lên ListView
        private void LoadListView()
        {
            this.lvSinhVien.Items.Clear();
            foreach (SinhVien sv in qlsv.DanhSach)
            {
                ThemSV(sv);
            }
        }
    }
#endregion

#region Các sự kiện
    //sự kiện Load form
    private void frmSinhVien_Load(object sender, EventArgs e)
    {
        qlsv = new QuanLySinhVien();
        qlsv.DocTuFile();
        LoadListView();
    }
    //Khi chọn dòng sinh viên bên ListView
    //thực hiện gán thông tin lên các control
    private void lvSinhVien_SelectedIndexChanged(object sender,
EventArgs e)
    {
        int count = this.lvSinhVien.SelectedItems.Count;
        if (count > 0)
        {
            ListViewItem lvitem =
this.lvSinhVien.SelectedItems[0];
            SinhVien sv = GetSinhVienLV(lvitem);
            ThietLapThongTin(sv);
        }
    }
    //Chức năng thêm sinh viên
    private void btnThem_Click(object sender, EventArgs e)
    {
        SinhVien sv = GetSinhVien();
        SinhVien kq = qlsv.Tim(sv.MaSo,
            delegate(object obj1, object obj2)
            {
                return (obj2 as
SinhVien).MaSo.CompareTo(obj1.ToString());
            });
        if (kq != null)
            MessageBox.Show("Mã sinh viên đã tồn tại!", "Lỗi
thêm dữ liệu",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        else
        {
            this qlsv.Them(sv);
            this.LoadListView();
        }
    }
}

```

```

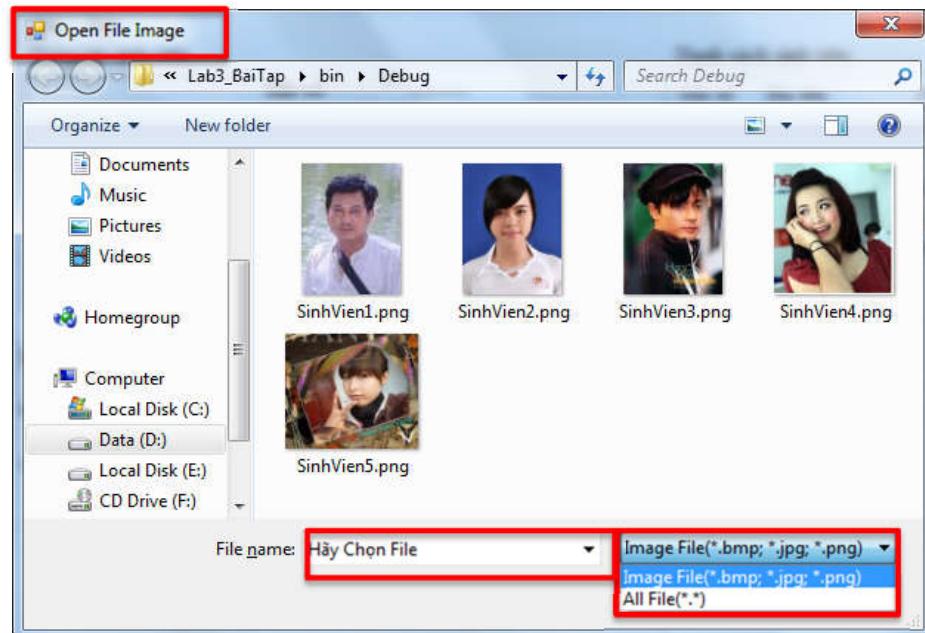
//Thoát chương trình
private void btnThoat_Click(object sender, EventArgs e)
{
    Application.Exit();
}
//Xóa tất cả sinh viên được chọn trên ListView
private void btnXoa_Click(object sender, EventArgs e)
{
    int count, i;
    ListViewItem lvitem;
    count = this.lvSinhVien.Items.Count - 1;
    for (i = count; i >= 0; i--)
    {
        lvitem = this.lvSinhVien.Items[i];
        if (lvitem.Checked)
            qlsv.Xoa(lvitem.SubItems[0].Text, SoSanhTheoMa);
    }
    this.LoadListView();
    this.btnMacDinh.PerformClick();
}
//Để các control ở giá trị mặc định
private void btnMacDinh_Click(object sender, EventArgs e)
{
    this.mtxtMaSo.Text = "";
    this.txtHoTen.Text = "";
    this.dtpNgaySinh.Value = DateTime.Now;
    this.txtDiaChi.Text = "";
    this.cboLop.Text = this.cboLop.Items[0].ToString();
    this.txtHinh.Text = "";
    this.pbHinh.ImageLocation = "";
    this.rdbName.Checked = true;
    for (int i = 0; i < this.clbChuyenNganh.Items.Count - 1;
i++)
        this.clbChuyenNganh.SetItemChecked(i, false);
}
//Sửa thông tin sinh viên được chọn
private void btnSua_Click(object sender, EventArgs e)
{
    SinhVien sv = GetSinhVien();
    bool kqsua;
    kqsua = qlsv.Sua(sv, sv.MaSo, SoSanhTheoMa);
    if (kqsua)
    {
        this.LoadListView();
    }
}
private int SoSanhTheoMa(object obj1, object obj2)
{
    SinhVien sv = obj2 as SinhVien;
    return sv.MaSo.CompareTo(obj1);
}
#endregion
}

```

III. Bài tập

Thêm chức năng cho chương trình như sau:

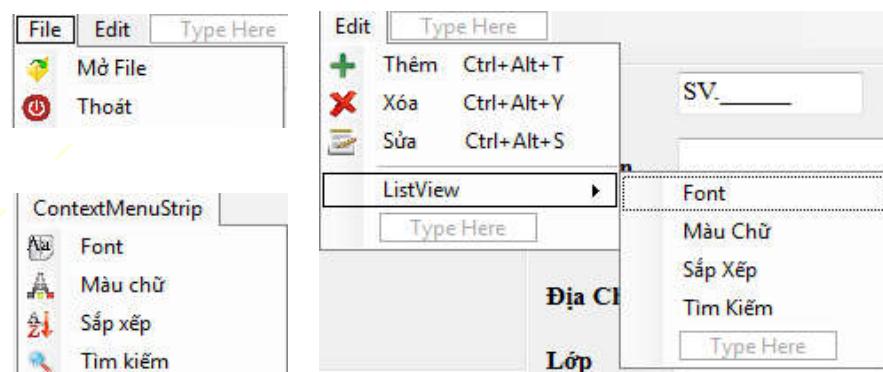
- Viết sự kiện cho button ... (btnBrowse). Chọn hình từ đĩa. Định dạng hộp thoại như sau:



Và thiết kế Statustrip: Hiển thị tổng số sinh viên trên danh sách



- Thiết kế các menu



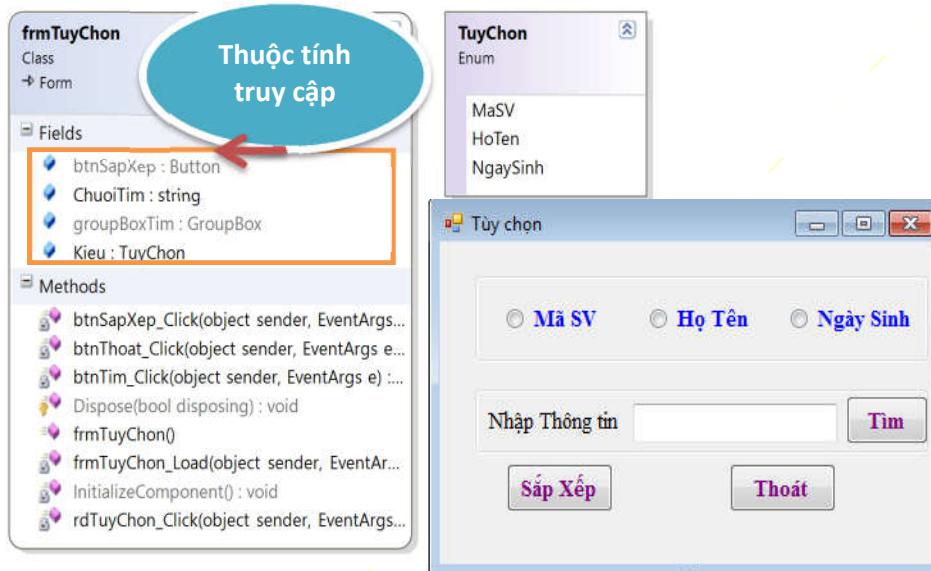
- Chức năng của các menu:

STT	Menu	MenuItem	Chức năng
1	File	Mở File	Mở file hình
		Thoát	Thoát chương trình
2	Edit	Thêm	Thêm sinh viên vào ListView
		Xóa	Xóa ds SV đánh dấu Check trên ListView

		Sửa	Sửa thông tin SV được chọn trên ListView
3	Edit → ListView ContextMenuStrip	Font	Chọn font chữ cho ListView
		Màu chữ	Chọn Màu chữ cho ListView
		Sắp xếp	Sắp xếp ds SV trên ListView
		Tìm kiếm	Tìm thông tin SV trên ListView

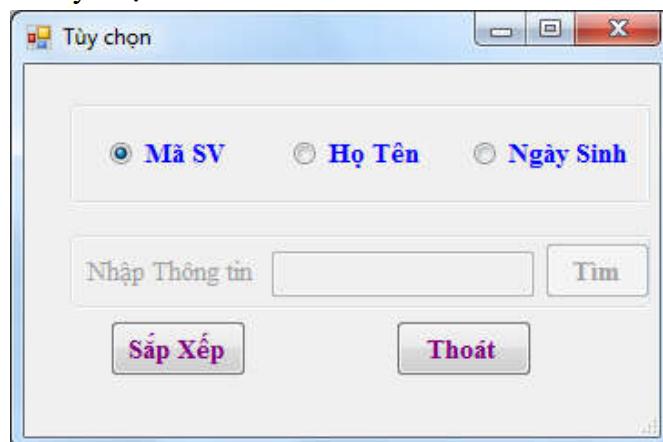
Chi tiết chức năng Sắp xếp và Tìm kiếm:

3.1. Thiết kế Form Tùy chọn với tên: frmTuyChon:

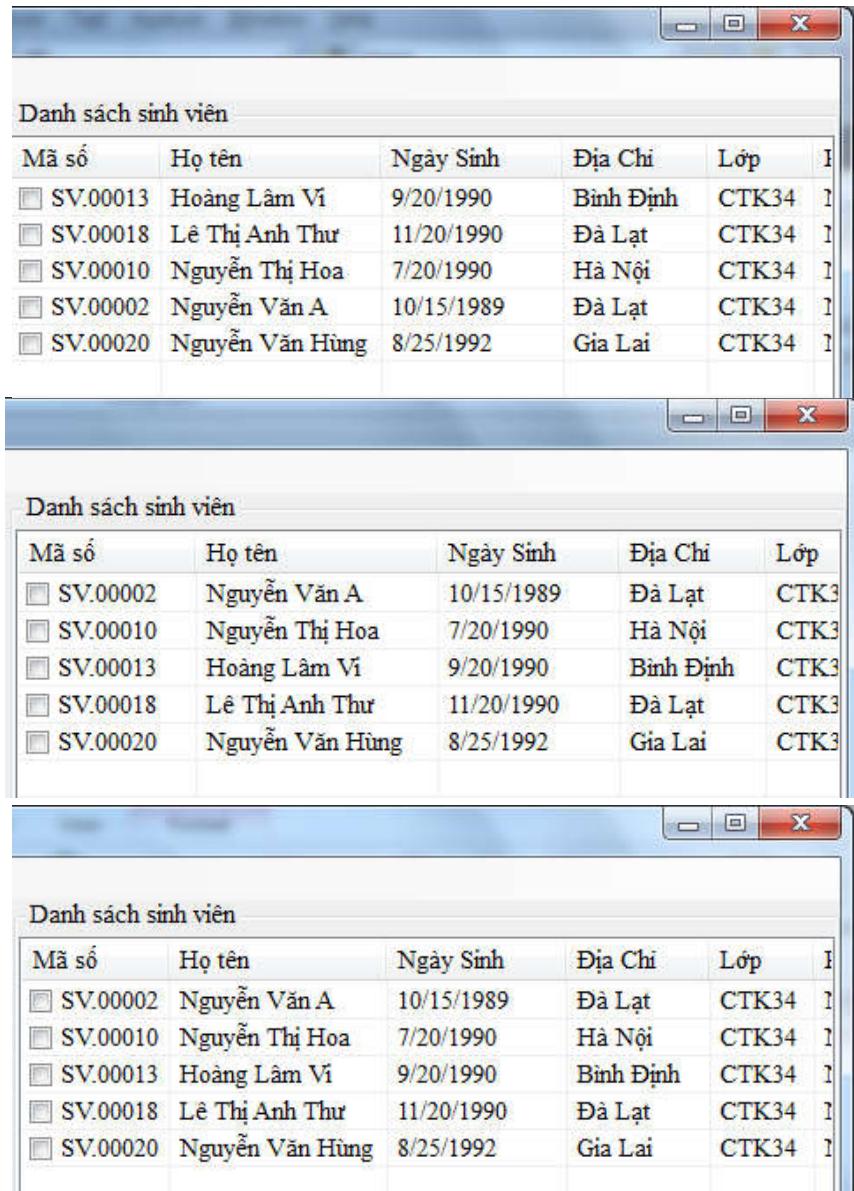


3.2. Khi người dùng Click vào menu Sắp xếp:

- Hiển thị form Tùy chọn như sau:



- Nhấn Button Sắp xếp thì danh sách trên ListView sẽ sắp theo kiểu chọn sắp:

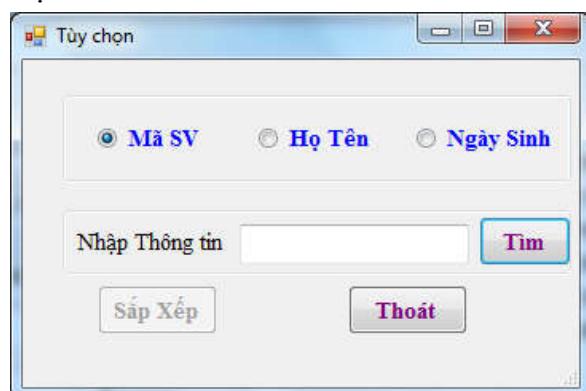


The image shows three separate windows, each titled "Danh sách sinh viên". Each window contains a table with the following columns: Mã số (Student ID), Họ tên (Name), Ngày Sinh (Date of Birth), Địa Chỉ (Address), and Lớp (Class). The data in all three windows is identical:

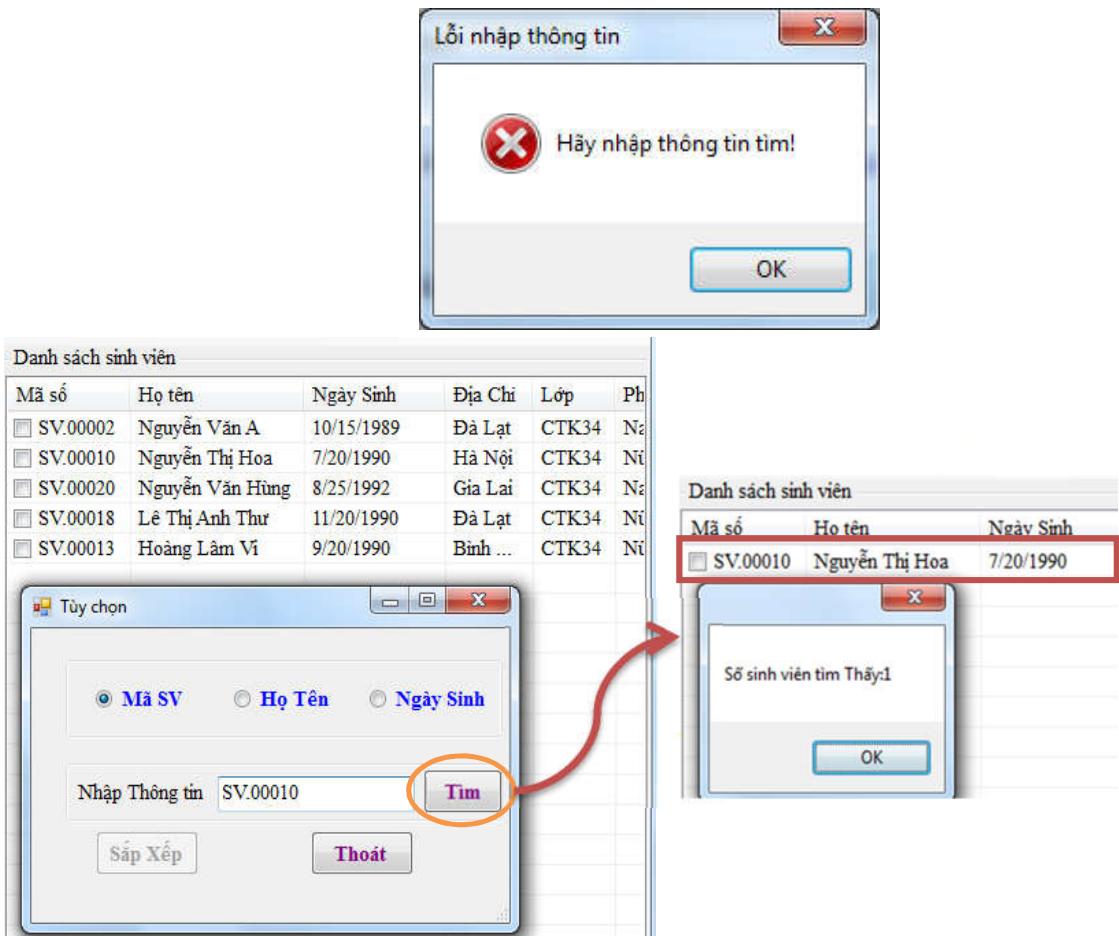
Mã số	Họ tên	Ngày Sinh	Địa Chỉ	Lớp
SV.00013	Hoàng Lâm Vĩ	9/20/1990	Bình Định	CTK34
SV.00018	Lê Thị Anh Thư	11/20/1990	Đà Lạt	CTK34
SV.00010	Nguyễn Thị Hoa	7/20/1990	Hà Nội	CTK34
SV.00002	Nguyễn Văn A	10/15/1989	Đà Lạt	CTK34
SV.00020	Nguyễn Văn Hùng	8/25/1992	Gia Lai	CTK34

3.3. Khi người dùng Click vào menu Tìm kiếm:

- Hiển thị form tùy chọn như sau:



- Nhấn Button Tìm:
 - o Nếu không nhập thông tin thông báo Lỗi:



8. Nếu nhập thông tin kết quả sẽ hiển thị lên

LAB 4 –ỨNG DỤNG WINDOWS FORM ĐƠN GIẢN

Thời lượng: 4 tiết

I. Mục tiêu

Áp dụng các kiến thức đã học về sử dụng các control, xử lý sự kiện cho form và control để xây dựng một ứng dụng đơn giản cho phép đọc dữ liệu từ tập tin txt hiển thị lên form và lưu dữ liệu vào tập tin txt

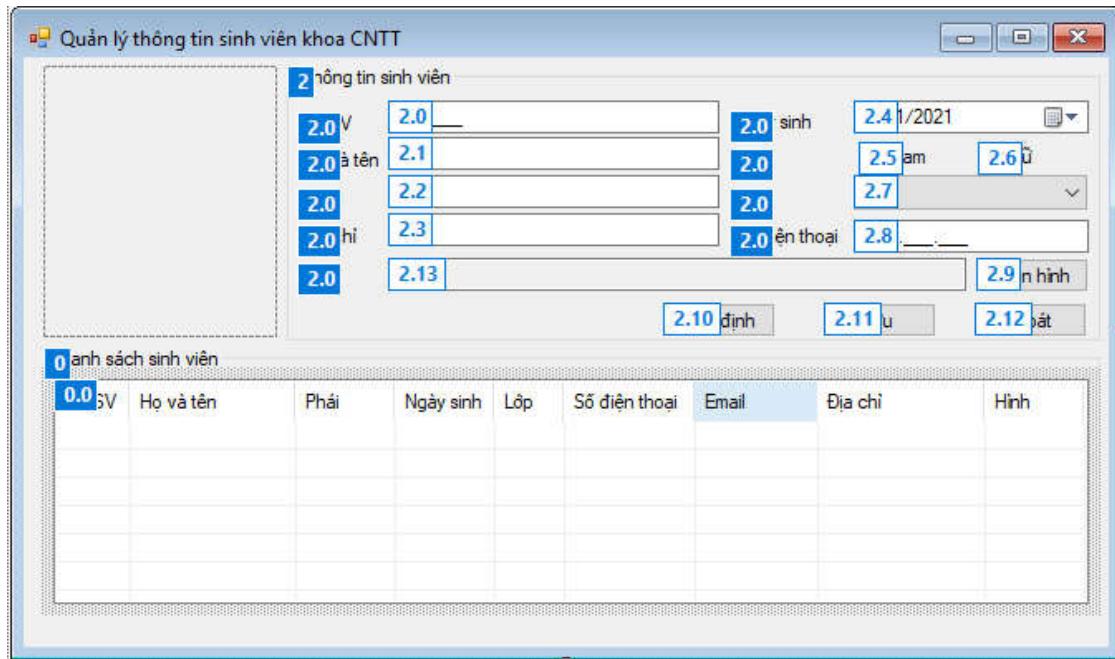
II. Yêu cầu thực hành

- Thiết kế form như hình sau:

- Các yêu cầu cần chú ý ở phần nhập thông tin sinh viên:

- a) Mã sinh viên gồm 7 kí tự số
- b) Ngày sinh hiển thị theo định dạng “dd/mm/yyyy”
- c) Giới tính: chỉ được chọn Nam hoặc Nữ
- d) Số điện thoại 10 chữ số, phân tách thành 3 nhóm.
- e) Lớp: có thể chọn 1 trong các lớp sau: “CTK43”, “CTK44”, “CTK45”, “CTK46”. Lưu ý: người dùng không có quyền nhập thêm lớp khác.
- f) Ô hình dùng để chứa đường dẫn của hình, không cho nhập

- Thiết lập thứ tự tab (Tab Order) như hình sau:



4. Xây dựng lớp Sinh viên, tạo một DSSV kiểu ArrayList để lưu danh sách sinh viên
 5. Các sự kiện cần xử lý:

- Click nút “Chọn hình” cho phép người dùng chọn 1 hình ảnh, đường dẫn hình lưu vào textbox, hình ảnh được tải lên khung hình ở bên trái
 - Click nút “Mặc định” : xóa toàn bộ nội dung được nhập ở phần nhập liệu
 - Click nút “Thoát”: đóng chương trình.
 - Click nút “Lưu”: cho phép thêm sinh viên mới vào danh sách DSSV hoặc chỉnh sửa thông tin của một sinh viên nằm trong danh sách
- Hướng dẫn:** Trước hết, thực hiện sinh viên theo mã sinh viên, nếu tìm thấy thì cập nhật thông tin của sinh viên này; nếu không tìm thấy thực hiện thêm sinh viên này vào DSSV và hiển thị DSSV ở ListView.
- Khi người dùng click chuột vào một sinh viên trong danh sách sinh viên, thông tin của sinh viên sẽ được hiển thị chi tiết lên trên phần nhập liệu
 - Khi click chuột phải vào Listview, hiển thị context menu “Xóa”, cho phép xóa một hoặc nhiều sinh viên đã chọn

III. Bài tập

- Xử lý thêm các sự kiện sau:
 - Khi form được nạp (Form_Load), tải toàn bộ danh sách sinh viên từ tập tin **DSNV.txt** vào ListView
 - Cho phép người dùng tải lại danh sách sinh viên từ tập tin **DSNV.txt** khi người dùng click chuột phải vào Listview và chọn “Tải lại danh sách”
 - Khi người dùng đóng form, kiểm tra xem người dùng có chỉnh sửa danh sách sinh viên được tải lên hay không, nếu có hiển thị MessageBox hỏi người dùng có muốn lưu danh sách đã thay đổi hay không? Nếu người dùng chọn “OK” thực hiện lưu danh sách sinh viên trên Listview vào file “**DSNV.txt**”.

2. Thực hiện thiết kế Form cho ứng dụng bài tập nhóm đã được phân công

LAB 5 - ỦNG DỤNG WINDOWS FORM – ĐỌC GHI DỮ LIỆU TỪ FILE

Thời lượng: 6 tiết

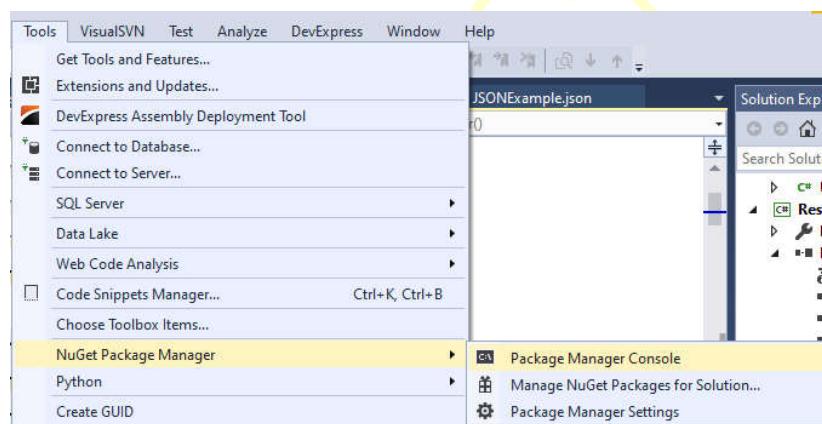
I. Mục tiêu:

- Xây dựng ứng dụng Windows Form đọc ghi dữ liệu từ các tập tin: *.txt, *.xml, *.json
- Kiểm tra giữa kỳ (2 tiết)

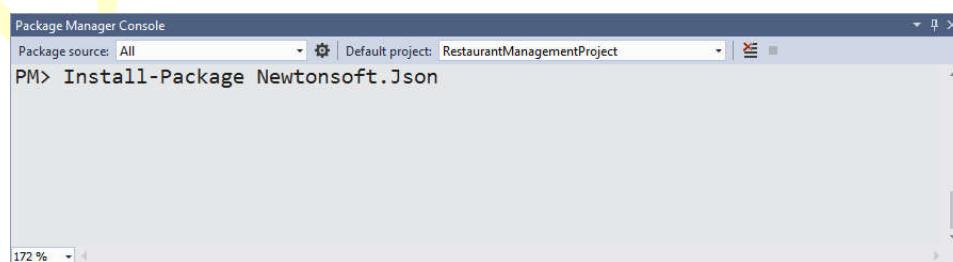
II. Hướng dẫn thực hành

1. Đọc tập tin JSON bằng Visual Studio

Để đọc được cấu trúc trong tập tin JSON bằng ngôn ngữ C#, chúng ta phải cài thêm gói thư viện Newtonsoft.Json vào dự án. Để cài gói thư viện này, thực hiện như sau: Vào menu Tools, chọn NuGet Package Manager, chọn Package Manager Console (Hình 2.7);



Khi đó cửa sổ Package Manager Console hiện ra phía dưới màn hình cho phép người dùng gõ lệnh (Hình 2.8); Từ đây có thể cài nhiều gói khác nhau vào dự án, để cài gói Newtonsoft.Json, chúng ta gõ lệnh “Install-Package Newtonsoft.Json” và nhấn Enter:



Hệ thống sẽ tìm phiên bản phù hợp nhất và cài vào dự án. Để sử dụng thư viện này và đọc được tập tin JSON, chúng ta sử dụng các câu lệnh using như sau:

```
using Newtonsoft.Json;
```

```
using System.IO;
using Newtonsoft.Json.Linq;
```

Giả sử cần đọc tập tin JSON như sau

```
JSONExample.json*  ▾ X
Schema: <No Schema Selected>
1  [
2   "sinhvien": [
3     [
4       {
5         "MSSV": "1245732", "hoten": "Thái Duy Quý",
6         "tuoi": 21, "diem": 8.5, "tongiao": false
7       },
8       {
9         "MSSV": "1245872", "hoten": "Phan Thị Thanh Nga",
10        "tuoi": 20, "diem": 9.0, "tongiao": true
11     }
12   ]
13 ]
```

Với cấu trúc sinh viên bao gồm các trường như: MSSV, họ tên, tuổi, điểm, và tôn giáo. Khi đó, để thuận tiện cần phải xây dựng một mô hình lớp bao gồm các trường như trong tập tin JSON, mô hình lớp có thể được xây dựng như trong lớp sau đây:

```
public class StudentInfo
{
    // Các thuộc tính
    public string MSSV { get; set; }
    public string Hoten { get; set; }
    public int Tuoi { get; set; }
    public double Diem { get; set; }
    public bool TonGiao { get; set; }

    // Phương thức tạo lập
    public StudentInfo(string mssv, string hoten, int tuoi, double diem, bool tongiao)
    {
        this.MSSV = mssv;
        this.Hoten = hoten;
        this.Tuoi = tuoi;
        this.Diem = diem;
        this.TonGiao = tongiao;
    }
}
```

Xây dựng phương thức đọc JSON như sau:

```
/// <summary>
/// Phương thức đọc tập tin JSON
/// </summary>
/// <param name="Path">Đường dẫn tập tin</param>
/// <returns>Danh sách các đối tượng từ tập tin JSON</returns>
private List<StudentInfo> LoadJSON(string Path)
{
    // Khai báo danh sách lưu trữ
    List<StudentInfo> List = new List<StudentInfo>();
```



```

// Đối tượng đọc tập tin
StreamReader r = new StreamReader(Path);
string json = r.ReadToEnd(); // Đọc hết
// Chuyển về thành mảng các đối tượng
var array = (JObject)JsonConvert.DeserializeObject(json);
// Lấy đối tượng sinhvien
var students = array["sinhvien"].Children();
foreach (var item in students) // Duyệt mảng
{
    // Lấy các thành phần
    string mssv = item["MSSV"].Value<string>();
    string hoten = item["hoten"].Value<string>();
    int tuoi = item["tuoi"].Value<int>();
    double diem = item["diem"].Value<double>();
    bool tongiao = item["tongiao"].Value<bool>();
    // Chuyển vào đối tượng StudentInfo
    StudentInfo info = new StudentInfo(mssv, hoten, tuoi, diem,
tongiao);
    List.Add(info); // Thêm vào danh sách
}
return List;
}

```

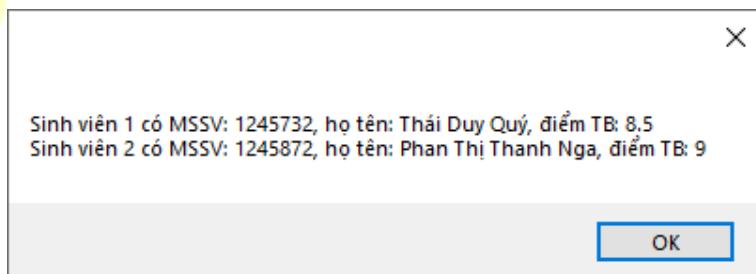
Phương thức ở trên có thể được viết trong lớp Form. Để đọc được tập tin JSON, chúng ta có thể tạo một nút và xử lý sự kiện Click như sau

```

private void btnJSON_Click(object sender, EventArgs e)
{
    string Str = ""; // chuỗi lưu trữ
    string Path = "../../JSONExample.json"; // Đường dẫn tập tin
    List<StudentInfo> List = LoadJSON(Path); // Gọi phương thức
    for (int i = 0; i < List.Count; i++) // Đọc danh sách
    {
        StudentInfo info = List[i];
        Str += string.Format("Sinh viên {0} có MSSV: {1}, họ tên: {2},\nđiểm TB: {3}\r\n", (i + 1), info.MSSV, info.Hoten, info.Diem);
    }
    MessageBox.Show(Str);
}

```

Kết quả khi đọc tập tin JSON



2. Đọc và ghi tập tin XML

XML là một ngôn ngữ có khả năng tự mô tả. Tập tin XML vừa chứa dữ liệu, vừa chứa các quy tắc và thông tin để có thể rút trích được dữ liệu từ tập tin đó. Có hai cách phổ biến để thao tác với các tập tin XML. Thứ nhất là dùng các lớp được cung cấp sẵn bởi .NET Framework trong namespace System.Xml. Bạn phải nạp namespace này vào mã nguồn trước khi sử dụng các lớp để đọc, ghi nội dung XML. Thứ hai là sử dụng LINQ to XML. LINQ to XML là một tập hợp các thư viện hàm cho phép bạn nạp tập tin XML vào bộ nhớ để xử lý và cập nhật tài liệu XML một cách thuận tiện và hiệu quả. Để sử dụng LINQ to XML, bạn cần nạp namespace System.Xml.Linq vào mã nguồn. Ngoài ra, còn có một số thư viện của bên thứ 3 nhằm giúp xử lý nội dung XML đơn giản và dễ dàng hơn.

Việc đọc tập tin XML và phân tích nội dung bên trong các thẻ XML có thể được thực hiện bằng nhiều cách khác nhau tùy thuộc vào yêu cầu cụ thể. Bạn có thể sử dụng lớp XmlDocument, XmlReader, XmlTextReader, XmlDocument trong namespace System.Xml. Hoặc bạn có thể sử dụng các lớp XDocument, XElement trong namespace System.Xml.Linq. Phần này giới thiệu cách sử dụng lớp XmlDocument để đọc, phân tích nội dung và truy vấn tài liệu XML sử dụng XPath. Lớp XmlDocument đọc toàn bộ nội dung XML vào bộ nhớ và cho phép chúng ta đọc nội dung các thẻ bất kỳ, điều hướng qua lại giữa phần tử cha-con, các phần tử cùng cấp.

Giả sử ta có tập tin books.xml như dưới đây:

```
<?xml version="1.0" encoding="utf-8" ?>
<! -- Books.xml stores information about Mahesh Chand and related books
-->
<catalog>
    <book ISBN="9831123212" yearpublished="2002">
        <title>A Programmer's Guide to ADO .Net using C#</title>
        <author>
            <first-name>Mahesh</first-name>
            <last-name>Chand</last-name>
        </author>
        <publisher>Apress</publisher>
        <price>44.99</price>
    </book>
    <book ISBN="9781484234" yearpublished="2019">
        <title>Pro Entity Framework Core 2</title>
        <author>
            <first-name>Adam</first-name>
            <last-name>Freeman</last-name>
        </author>
        <publisher>Apress</publisher>
        <price>45.09</price>
    </book>
</catalog>
```

Để đọc nội dung XML và lấy thông tin danh mục sách trong tập tin nói trên, ta có thể sử dụng lớp XmlDocument như sau:

```
public static void Main(string[] args)
{
    // Load XML file into XmlDocument instance
    var xmlDoc = new XmlDocument();
    xmlDoc.Load("../..\\books.xml");

    // Get list of nodes whose name is Book
    var nodeList = xmlDoc.DocumentElement.SelectNodes("/catalog/book");

    foreach (XmlNode node in nodeList)
    {
        // Read attribute value
        var isbn = node.Attributes["ISBN"].Value;
        // Read child node value
        var title = node.SelectSingleNode("title").InnerText;
        var price = node.SelectSingleNode("price").InnerText;
        // Read the descendant node value
        var firstName = node.SelectSingleNode("author/first-
name").InnerText;
        var lastName = node.SelectSingleNode("author/last-
name").InnerText;
        Console.WriteLine("{0,-15}{1,-50}{2,-15}{3,-15}{4,6}",
                           isbn, title, firstName, lastName, price);
    }
}
```

Kết quả đọc như sau:

9831123212	A Programmer's Guide to ADO .Net using C#	Mahesh	Chand	44.99
9781484234	Pro Entity Framework Core 2	Adam		Freeman
	45.09			

Tương tự, việc tạo tập tin XML từ dữ liệu cho trước cũng tương đối đơn giản. Bạn có thể sử dụng các lớp trong namespace System.Xml hoặc sử dụng LINQ to XML tùy theo yêu cầu dự án. Sử dụng lớp XmlWriter để ghi dữ liệu ra một tập tin books.xml như sau:

```
public static void Main(string[] args)
{
    using (XmlWriter writer = XmlWriter.Create("books.xml"))
    {
        // Write Processing Instruction
        String pi = "type='text/xsl' href='book.xsl'";
        writer.WriteProcessingInstruction("xml-stylesheet", pi);
        // Write DocumentType
        writer.WriteDocType("catalog", null, null, "<!ENTITY h
\"hardcover\"");
        // Write a Comment
        writer.WriteComment("This is a book sample XML");
        // Root element - start tag
        writer.WriteStartElement("book");
        // Write ISBN attribute
        writer.WriteString("ISBN", "9831123212");
        // Write year attribute
        writer.WriteString("yearpublished", "2002");
        // Write title
        writer.WriteString("author", "Mahesh Chand");
        // Write author
        writer.WriteString("title", "Visual C# Programming");
        // Write price
        writer.WriteString("price", "44.95");
        // Root element - end tag
        writerEndElement();
        // End Documentd
        writer.EndDocument();
        // Flush it
        writer.Flush();
    }
}
```

Tập tin books.xml được tạo bởi đoạn chương trình trên có nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="book.xsl"?>
<!DOCTYPE book [
  <!ENTITY h "hardcover">
]>
<!--This is a book sample XML-->
<book ISBN="9831123212" yearpublished="2002">
  <author>Mahesh Chand</author>
  <title>Visual C# Programming</title>
  <price>44.95</price>
</book>
```



Việc ghi nội dung XML như trên sẽ trở nên phức tạp và khó kiểm soát khi lượng dữ liệu lớn và có nhiều thông tin hơn. Nếu bạn có một danh sách các đối tượng và muốn lưu trữ dữ liệu của chúng dưới dạng tập tin XML, lớp XmlSerializer sẽ là một lựa chọn phù hợp. Giả sử, ta có lớp Book chứa 5 thuộc tính mô tả về một cuốn sách như sau:

```
public class Book
{
    public string ISBN { get; set; }
    public string Title { get; set; }
    public string Author { get; set; }
    public decimal Price { get; set; }
    public int YearPublished { get; set; }
}
```

Khi đó, để lưu dữ liệu từ một mảng đối tượng books vào tập tin books.xml, ta sử dụng đoạn mã sau:

```
private static void SaveToXmlFile(List<Book> books)
{
    var serializer = new XmlSerializer(typeof(List<Book>));

    using (var writer = new StreamWriter("books.xml"))
    {
        serializer.Serialize(writer, books, null);
        writer.Close();
    }
}
```

Tập tin books.xml kết quả:

```
<?xml version="1.0" encoding="utf-8"?>
<ArrayOfBook
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <Book>
        <ISBN>9831123212</ISBN>
        <Title>A Programmer's Guide to ADO .Net using C#</Title>
        <Author>Mahesh Chand</Author>
        <Price>44.99</Price>
        <YearPublished>2002</YearPublished>
    </Book>
    <Book>
        <ISBN>9781484234</ISBN>
        <Title>Pro Entity Framework Core 1</Title>
        <Author>Adam Freeman</Author>
        <Price>44.99</Price>
        <YearPublished>2018</YearPublished>
    </Book>
</ArrayOfBook>
```

4. Thiết kế Form nhập thông tin sinh viên như sau:

5. Định nghĩa lớp Student (SinhVien) có các thuộc tính và hàm cần thiết
6. Xây dựng các lớp để đọc, ghi danh sách sinh viên từ tập tin văn bản thông thường (*.txt)
7. Xây dựng lớp StudentManager(QLSinhVien) với các chức năng cơ bản như sau:
 - a. Thêm và cập nhật thông tin sinh viên (có lưu vào file)
 - b. Tìm kiếm sinh viên theo Tên, Lớp, MSSV
 - c. Xóa một hay nhiều sinh viên (có lưu vào file)
8. Chương trình phải đảm bảo:
 - Khi chạy chương trình danh sách sinh viên sẽ được tải từ tập tin
 - MSSV gồm 7 chữ số, Số CMND gồm 9 chữ số và Số ĐT nhập 10 chữ số
 - Môn đăng ký: cho phép click chuột phải để mở ContextMenu cho phép xóa hoặc thêm môn
 - Khi chọn một sinh viên trong danh sách sinh viên, thông tin của sinh viên phải được tự động điền lên phần thông tin phía trên danh sách.
 - Người dùng phải nhập hết thông tin rồi mới cho phép thêm mới hoặc cập nhật. Nếu người dùng chưa nhập đầy đủ thông tin mà đã nhấn nútThêm mới/ Cập nhật thì cần thông báo cho người dùng.

- Danh sách sinh viên: cho phép chọn nhiều sinh viên dùng Checkbox, cho phép click chuột phải để mở ContextMenu cho phép xóa 1 hoặc nhiều sinh viên đã chọn.
- Khi click vào nút Tìm kiếm, chương trình phải hiển thị form cho người dùng nhập điều kiện tìm kiếm (theo MSSV/Tên/Lớp)
- Khi người dùng nhấn nút Thoát phải hỏi lại người dùng có chắc chắn muốn thoát chương trình hay không. Nếu đồng ý thì thoát chương trình.

III. Bài tập:

Thêm chức năng của chương trình như sau:

1. MSSV gồm 7 chữ số có dạng AABBCCC, trong đó AA là 2 số cuối năm nhập học của sinh viên, BB = 10, CCC là số bất kỳ. Không có sinh viên nào trùng MSSV. Lưu ý: dựa vào lớp để biết năm nhập học của sinh viên.
2. Xây dựng các lớp để đọc, ghi danh sách sinh viên từ tập tin. Yêu cầu: chương trình có thể hỗ trợ các định dạng tập tin sau đây:
 - a. Tập tin văn bản thông thường: students.txt
 - b. Tập tin xml: students.xml
 - c. Tập tin JSON: students.json
3. Chương trình cho phép tìm sinh viên theo một hoặc nhiều điều kiện (điều kiện bất kỳ)

LAB 6 – KẾT NỐI VÀ TRUY VẤN DỮ LIỆU

Thời lượng: 4 tiết

I. Mục tiêu

Bài thực hành này giúp sinh viên tìm hiểu cách kết nối đến một cơ sở dữ liệu và thực thi trực tiếp một số lệnh truy vấn đơn giản:

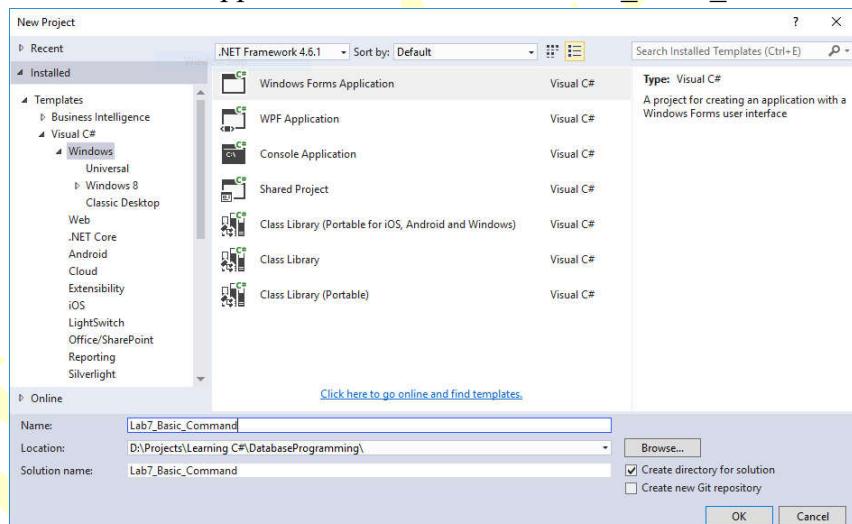
- SELECT: Lấy các mẫu tin từ một bảng hoặc khung nhìn.
- INSERT: Thêm một mẫu tin mới vào một bảng.
- UPDATE: Cập nhật một mẫu tin có sẵn trong bảng.
- DELETE: Xóa một mẫu tin khỏi bảng.

Sau bài thực hành này, sinh viên cần nắm rõ những vấn đề sau:

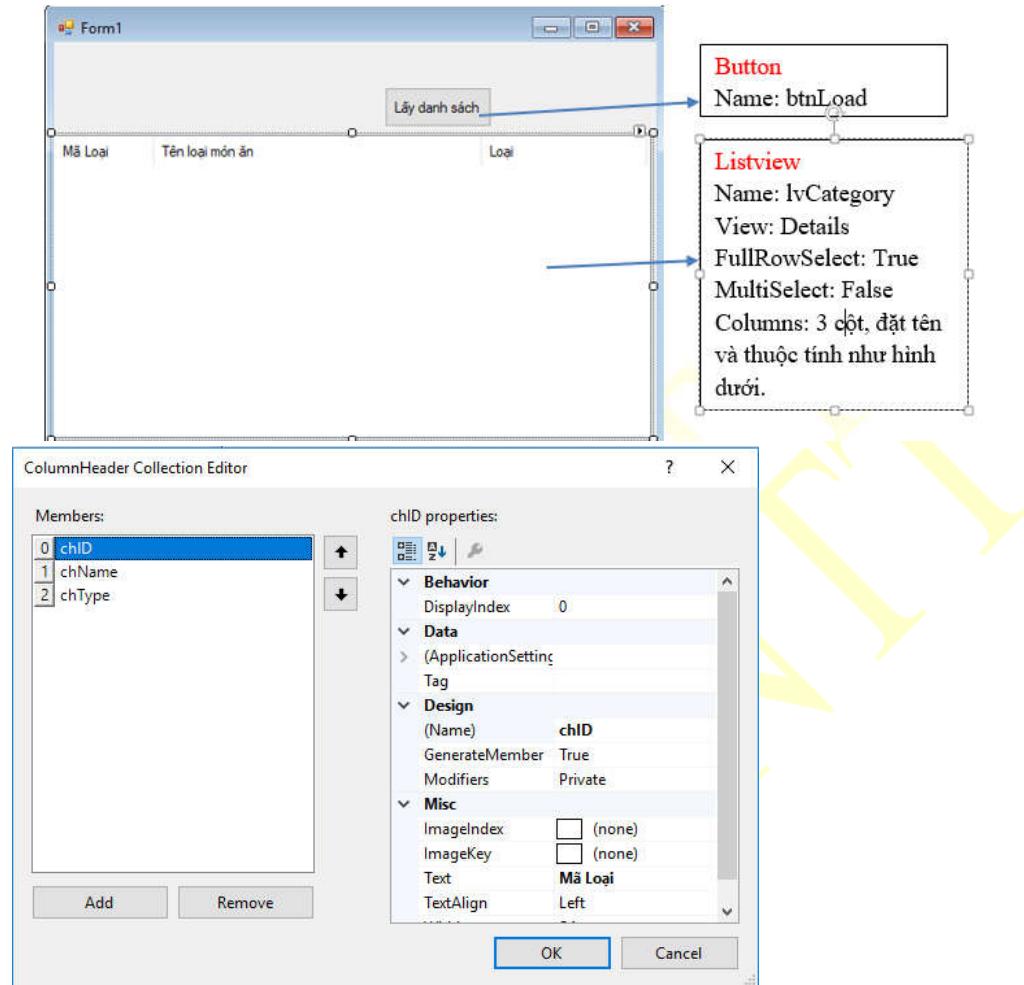
- Các thành phần của chuỗi kết nối và ý nghĩa của chúng.
- Cách tạo đối tượng kết nối đến các cơ sở dữ liệu SQL Server, Access.
- Cách sử dụng đối tượng Command để thực thi truy vấn, DataReader để đọc dữ liệu.
- Cách xây dựng ứng dụng trên nền Windows Form.

II. Hướng dẫn thực hành

Tạo một dự án Windows Application mới, đặt tên là Lab6_Basic_Command



Thiết kế Form như sau:



1. Lấy dữ liệu bằng cách dùng phương thức ExecuteReader

Nhấp đôi chuột vào nút btnLoad và thêm đoạn mã sau vào đầu lớp Form1.cs:

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Collections.Generic;
...
```

Bổ sung đoạn mã sau vào phương thức btnLoad_Click

```

private void btnLoad_Click(object sender, EventArgs e)
{
    // Tạo chuỗi kết nối tới cơ sở dữ liệu RestaurantManagement
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";

    // Tạo đối tượng kết nối
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    string query = "SELECT ID, Name, Type FROM Category";
    sqlCommand.CommandText = query;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExecuteReader
    SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

    // Gọi hàm hiển thị dữ liệu lên màn hình
    this.DisplayCategory(sqlDataReader);

    // Đóng kết nối
    sqlConnection.Close();
}

```

Để hiển thị dữ liệu lên ListView, bạn phải viết thêm hàm DisplayCategory như sau:

```

private void DisplayCategory(SqlDataReader reader)
{
    // Xóa tất cả các dòng hiện tại
    lvCategory.Items.Clear();

    // Đọc một dòng dữ liệu
    while (reader.Read())
    {
        // Tạo một dòng mới trong ListView
        ListViewItem item = new ListViewItem(reader["ID"].ToString());
        ...

        // Thêm dòng mới vào ListView
        lvCategory.Items.Add(item);

        // Bổ sung các thông tin khác cho ListViewItem
        item.SubItems.Add(reader["Name"].ToString());
        item.SubItems.Add(reader["Type"].ToString());
    }
}

```

Nhấn F5 để chạy chương trình. Nhấn nút “Lấy danh sách” để xem kết quả

Mã	Loại	Tên loại món ăn	Loại
1		Khai vị	1
2		Hải sản	1
3		Gà	1
4		Cơm	1
5		Thịt	1
6		Rau	1
8		Canh	1
9		Lẩu	1
10		Bia	0
11		Nước ngọt	0
12		Cà phê	0
13		Trà đá	0

2. Thêm một mẫu tin dùng lệnh INSERT

Thay đổi lại giao diện cho Form1 như hình sau:

Form1

Mã nhóm:	<input type="text"/>		
Tên nhóm thức ăn:	<input type="text"/>		
Loại:	<input type="text"/>		
<button>Lấy danh sách</button> <button>Thêm</button> <button>Cập nhật</button> <button>Xóa</button>			
Mã	Loại	Tên loại món ăn	Loại
<			>

Textbox
Name: txtID
ReadOnly: True

Textbox
Name: txtName
Name: txtType

Button
Name: btnAdd
Text: Thêm

Name: btnUpdate
Text: Cập nhật
Enable: False

Name: btnDelete
Text: Xóa
Enable: False

Nhấp đôi chuột vào nút btnAdd (Thêm mới), bổ sung đoạn mã sau vào phương thức btnAdd Click

```
private void btnAdd_Click(object sender, EventArgs e)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "INSERT INTO Category(Name, [Type])" +
        "VALUES (N'" + txtCategoryName.Text + "', " + txtType.Text + ")";

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExecuteReader
    int numRowsAffected = sqlCommand.ExecuteNonQuery();

    // Đóng kết nối
    sqlConnection.Close();

    if (numRowsAffected == 1)
    {
        MessageBox.Show("Thêm nhóm món ăn thành công");

        // Tải lại dữ liệu
        btnLoad.PerformClick();

        // Xóa các ô nhập
        txtCategoryName.Text = "";
        txtType.Text = "";
    }
    else
    {
        MessageBox.Show("Đã có lỗi xảy ra. Vui lòng thử lại");
    }
}
```

Nhấn nút F5 để chạy chương trình. Nhập dữ liệu như hình sau và nhấn nút “Thêm”.



Form1

Mã nhóm:

Tên nhóm thức ăn:

Loại:

Lấy danh sách Thêm Cập nhật Xóa

Mã Loại	Tên loại món ăn	Loại

Kết quả

Form1

Mã nhóm:

Tên nhóm thức ăn:

Loại:

Lấy danh sách Thêm Cập nhật Xóa

Mã Loại	Tên loại món ăn	Loại

X

Thêm nhóm món ăn thành công

OK

Mã Loại	Tên loại món ăn	Loại
4	Cơm	1
5	Thịt	1
6	Rau	1
8	Canh	1
9	Lẩu	1
10	Bia	0
11	Nước ngọt	0
12	Cà phê	0
13	Trà đá	0
16	Nhóm thức ăn mới	0

3. Cập nhật một mẫu tin dùng lệnh UPDATE

- Nhấp phải chuột vào ListView lvCategory, chọn Properties.
- Trong khung Properties, nhấn chọn nút Events. Nhấp đôi chuột vào sự kiện Click.
- Bổ sung đoạn mã sau vào phương thức lvCategory_Click

```
private void lvCategory_Click(object sender, EventArgs e)
{
    // Lấy dòng được chọn trong Listview
    ListViewItem item = lvCategory.SelectedItems[0];

    // Hiển thị dữ liệu lên Textbox
    txtCategoryID.Text = item.Text;
    txtCategoryName.Text = item.SubItems[1].Text;
    txtType.Text = item.SubItems[1].Text == "0" ? "Thức uống" : "Đồ ăn";

    // Hiển thị nút cập nhật và xóa
    btnUpdateCat.Enabled = true;
    btnDeleteCat.Enabled = true;
}
```

- Nhấn F5 để chạy chương trình, nhấn nút btnLoad rồi nhấp chuột vào ListView để xem kết quả chạy chương trình.
- Nhấn nút Close để tắt Form1.
- Nhấp đôi chuột vào nút btnUpdate (Cập nhật) và bổ sung đoạn mã sau

```

private void btnUpdate_Click(object sender, EventArgs e)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "UPDATE Category SET Name = N'" + txtCategoryName.Text +
                           "' , [Type] = " + txtType.Text +
                           " WHERE ID = " + txtCategoryID.Text;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExecuteReader
    int numRowsAffected = sqlCommand.ExecuteNonQuery();

    // Đóng kết nối
    sqlConnection.Close();

    if (numRowsAffected == 1)
    {
        // Cập nhật lại dữ liệu trên Listview
        ListViewItem item = lvCategory.SelectedItems[0];

        item.SubItems[1].Text = txtCategoryName.Text;
        item.SubItems[2].Text = txtType.Text;

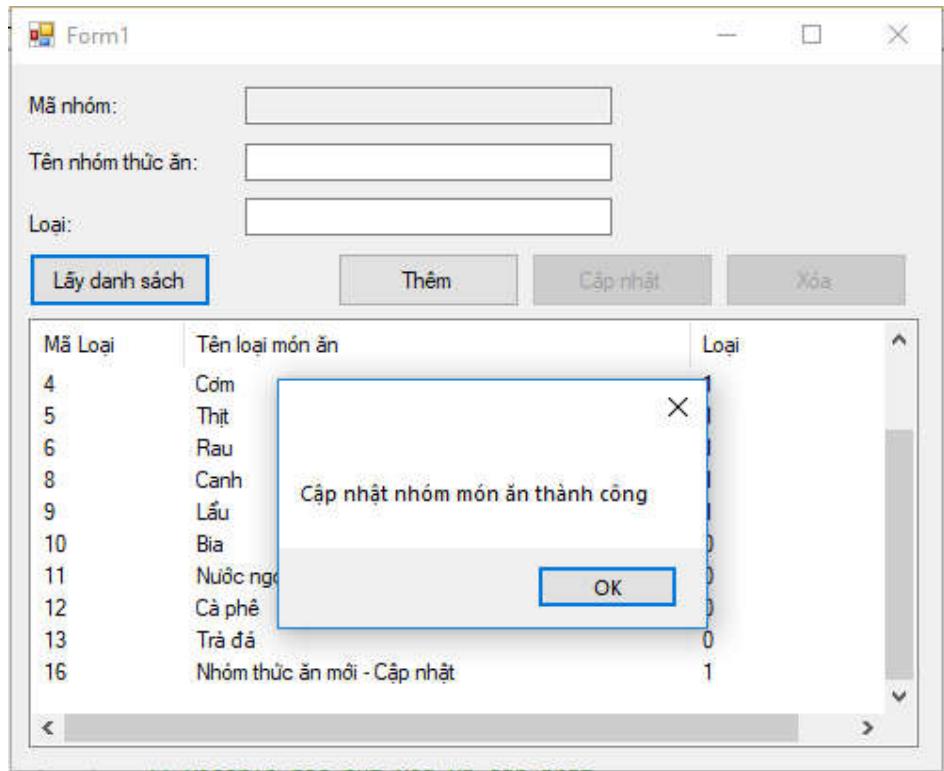
        // Xóa các ô nhập
        txtCategoryID.Text = "";
        txtCategoryName.Text = "";
        txtType.Text = "";

        // Disable các nút xóa và cập nhật
        btnUpdate.Enabled = false;
        btnDelete.Enabled = false;

        MessageBox.Show("Cập nhật nhóm món ăn thành công");
    }
    else
    {
        MessageBox.Show("Đã có lỗi xảy ra. Vui lòng thử lại");
    }
}

```

- Nhấn phím F5 để chạy và kiểm tra chương trình



4. Xóa một mẫu tin dùng lệnh DELETE

Nhấp đúi chuột vào nút btnDelete và bổ sung đoạn mã sau vào phương thức btnDelete_Click

```

private void btnDelete_Click(object sender, EventArgs e)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "DELETE FROM Category " +
                           "WHERE ID = " + txtCategoryID.Text;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Thực thi lệnh bằng phương thức ExecuteReader
    int numberOfRowsAffected = sqlCommand.ExecuteNonQuery();

    // Đóng kết nối
    sqlConnection.Close();
}

```

```

if (numOfRowsAffected == 1)
{
    // Cập nhật lại dữ liệu trên ListView
    ListViewItem item = lvCategory.SelectedItems[0];
    lvCategory.Items.Remove(item);

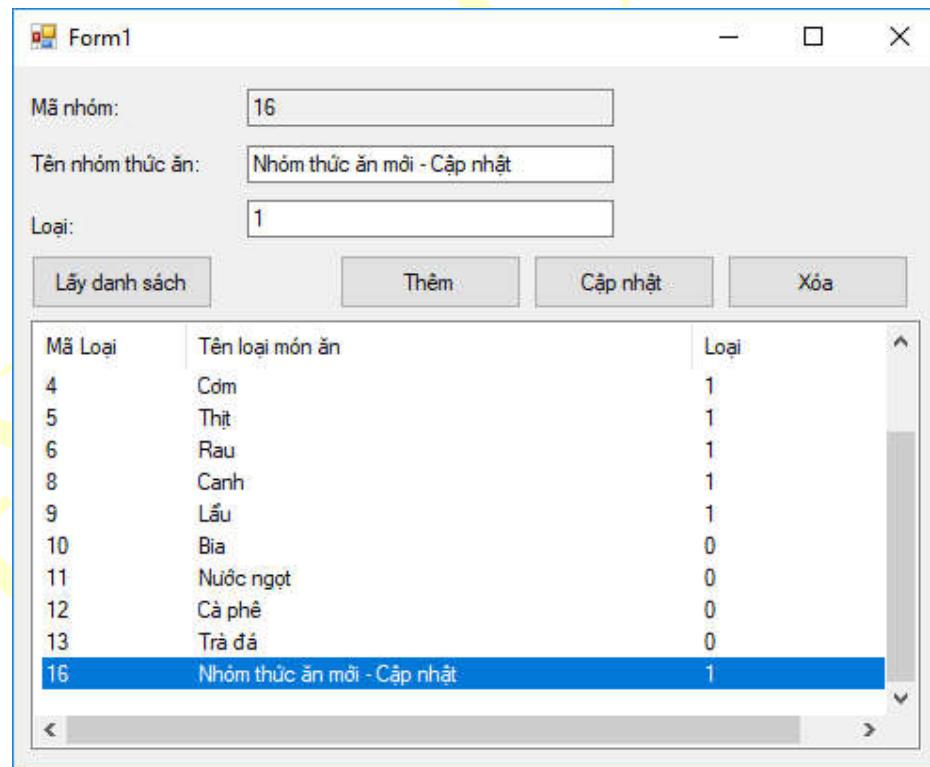
    // Xóa các ô nhập
    txtCategoryID.Text = "";
    txtCategoryName.Text = "";
    txtType.Text = "";

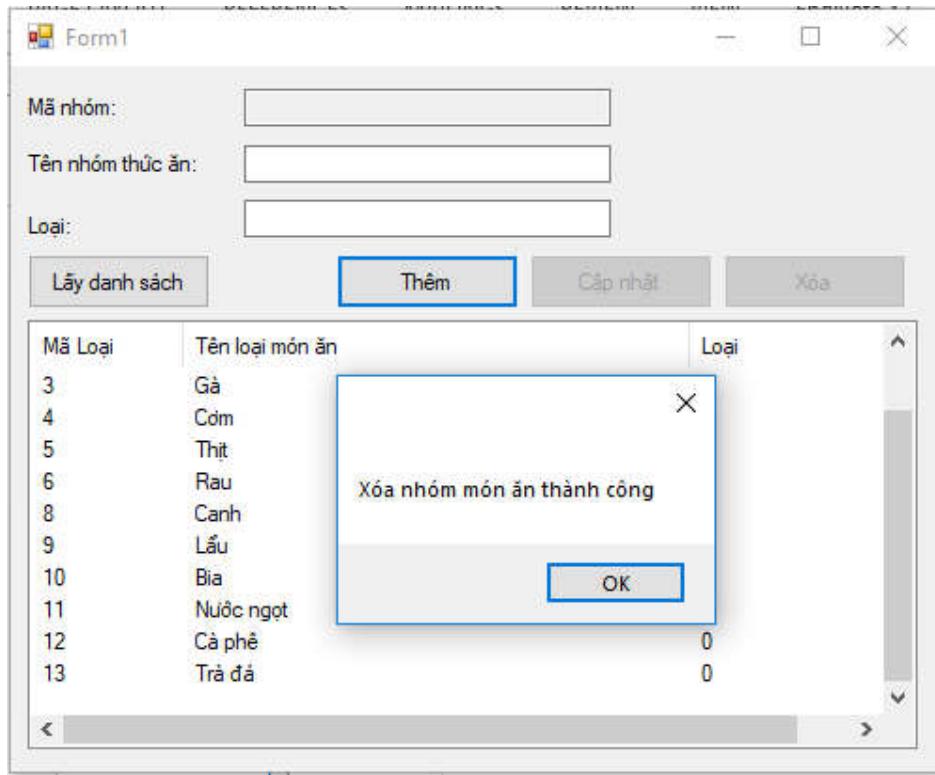
    // Disable các nút xóa và cập nhật
    btnUpdate.Enabled = false;
    btnDelete.Enabled = false;

    MessageBox.Show("Xóa nhóm món ăn thành công");
}
else
{
    MessageBox.Show("Đã có lỗi xảy ra. Vui lòng thử lại");
}
}

```

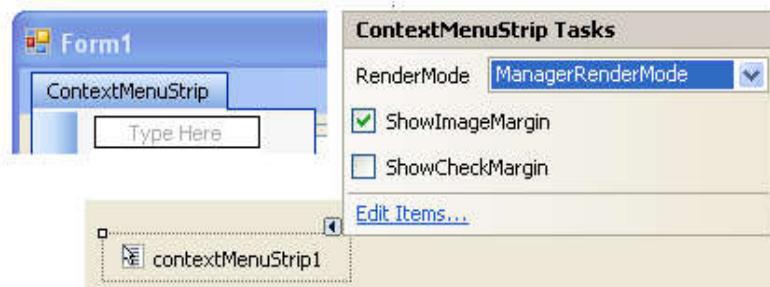
- Nhấn nút F5 để chạy chương trình.
- Nhấn nút btnLoad, nhấp chuột vào ListView, chọn dòng mới được thêm vào ở phần 2. Sau đó nhấn nút Xóa.



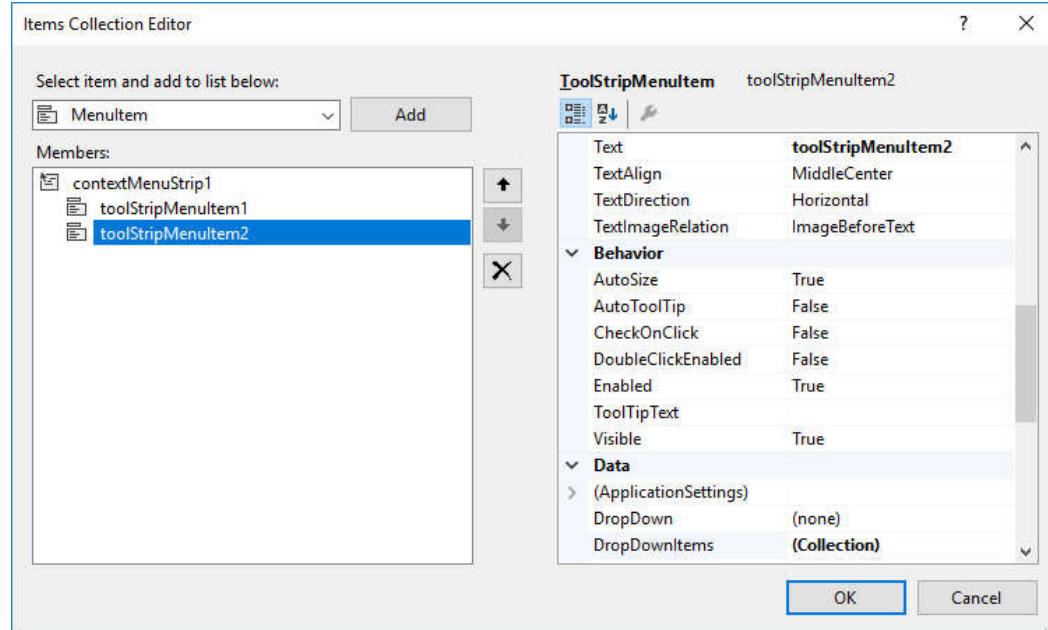


5. Lấy dữ liệu dùng bằng phương thức Fill của DataAdapter

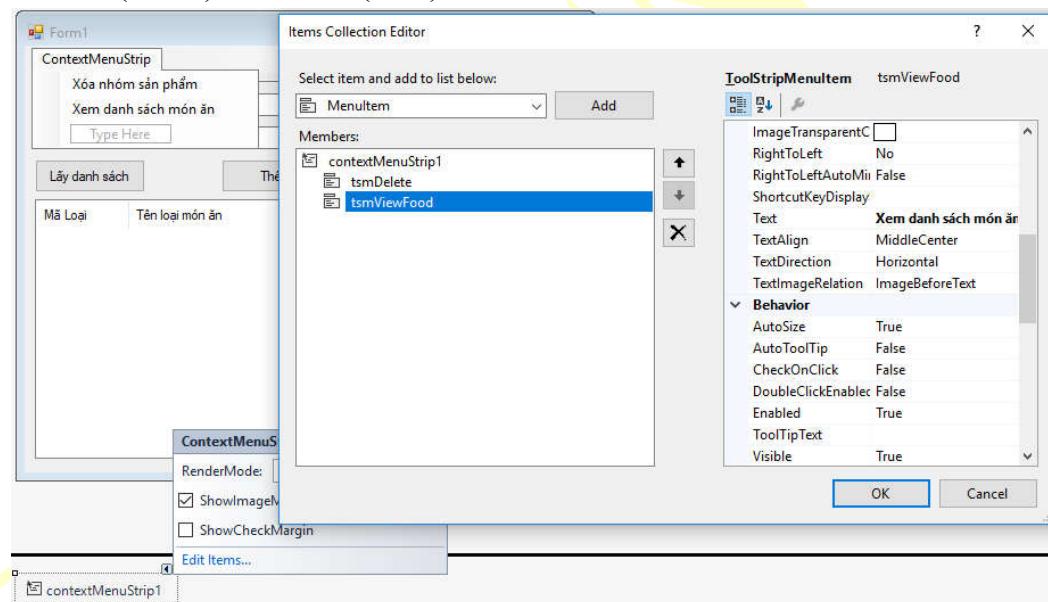
Mở Form1, trong nhóm All Windows Form hoặc Menu and Toolbar của thanh Toolboxes, chọn Context Menu Strip và kéo nó vào Form1.



- Phía dưới Form1 có một component tên là contextMenuStrip1. Nhấp chuột vào dấu mũi tên hình tam giác, chọn Edit Items...
- Chọn loại Menu Item, Nhấn nút Add để tạo 2 Menu Item như hình sau



Đổi tên (Name) và tiêu đề (Text) của các MenuItem như hình sau



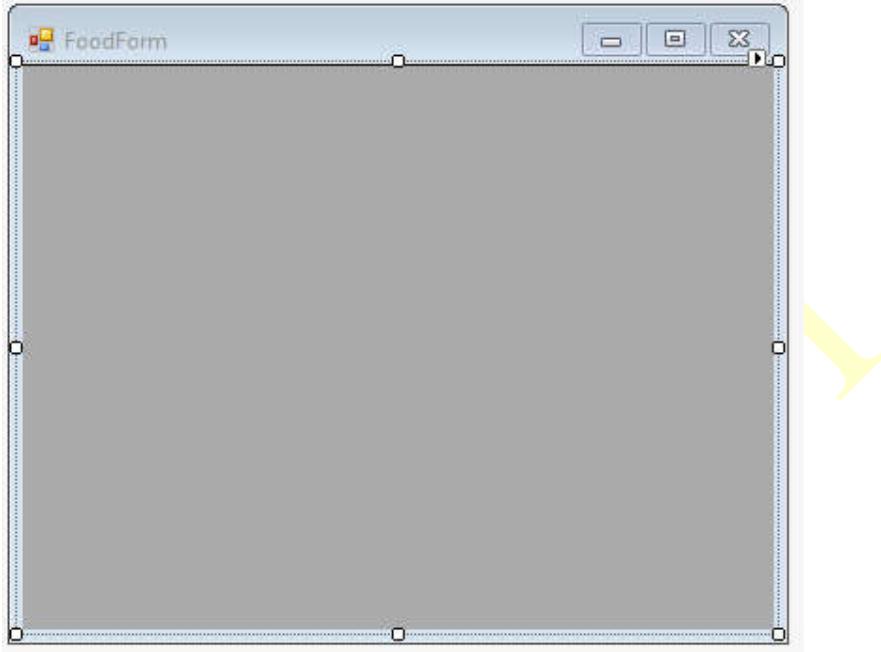
Nhấp đúp chuột vào từng Menu để tạo phương thức xử lý sự kiện Click

```
private void tsmDelete_Click(object sender, EventArgs e)
{
    if(lvCategory.SelectedItems.Count > 0)
        btnDelete.PerformClick();
}

1 reference
private void tsmViewFood_Click(object sender, EventArgs e)
{}
```

- Nhấp phải chuột lên ListView, chọn Properties. Trong khung Properties, mục ContextMenuStrip, chọn contextMenuStrip1.
- Tạo một Form mới, đặt tên là FoodForm (Name: frmFood)

- Trong nhóm Data của thanh Toolboxes, chọn DataGridView và kéo nó lên Form mới
- Đặt tên cho DataGridView là dgvFood và thiết lập thuộc tính Anchor là Top, Left, Right, Bottom.



- Nhấp phải chuột lên Form mới, chọn View Code
- Tạo một phương thức mới trong lớp FoodForm như sau:

```

public void LoadFood(int categoryID)
{
    // Tạo đối tượng kết nối
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true";
    SqlConnection sqlConnection = new SqlConnection(connectionString);

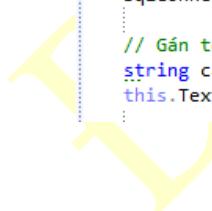
    // Tạo đối tượng thực thi lệnh
    SqlCommand sqlCommand = sqlConnection.CreateCommand();

    // Thiết lập lệnh truy vấn cho đối tượng Command
    sqlCommand.CommandText = "SELECT Name FROM Category where ID = " + categoryID;

    // Mở kết nối tới cơ sở dữ liệu
    sqlConnection.Open();

    // Gán tên nhóm sản phẩm cho tiêu đề
    string catName = sqlCommand.ExecuteScalar().ToString();
    this.Text = "Danh sách các món ăn thuộc nhóm: " + catName;
}

```



```

sqlCommand.CommandText = "SELECT * FROM Food WHERE FoodCategoryID = " + categoryID;

// Tạo đối tượng DataAdapter
SqlDataAdapter da = new SqlDataAdapter(sqlCommand);

// Tạo DataTable để chứa dữ liệu
DataTable dt = new DataTable("Food");
da.Fill(dt);

// Hiển thị danh sách món ăn lên Form
dgvFood.DataSource = dt;

// Đóng kết nối và giải phóng bộ nhớ
sqlConnection.Close();
sqlConnection.Dispose();
da.Dispose();
}

```

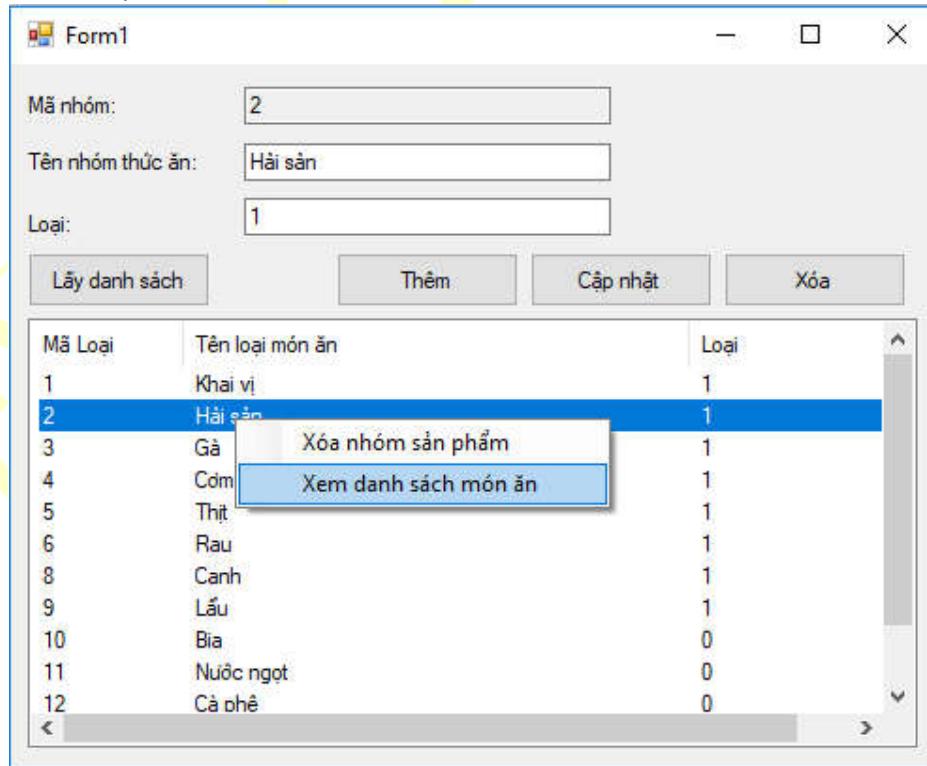
Trở lại lớp Form1.cs, bổ sung đoạn mã sau vào phương thức tsmViewFood_Click

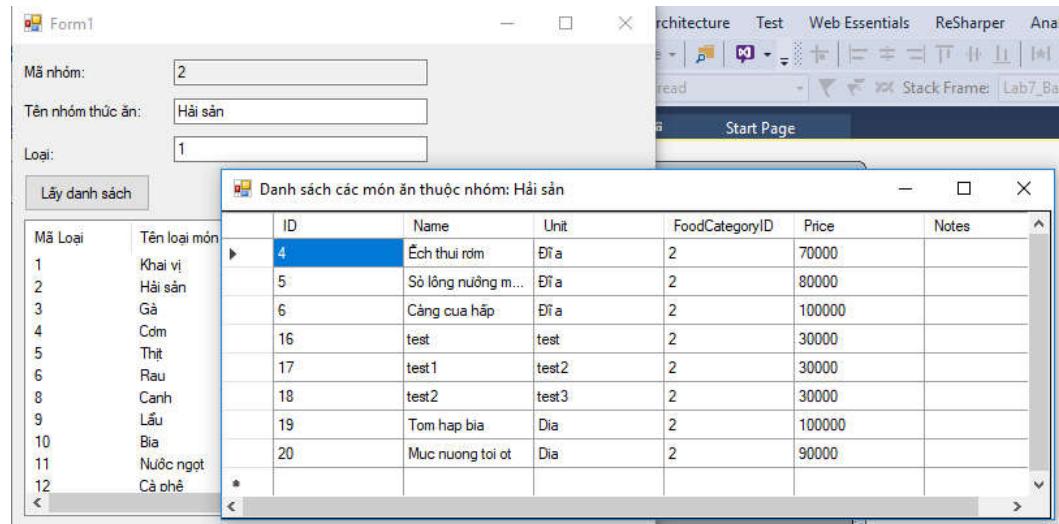
```

private void tsmViewFood_Click(object sender, EventArgs e)
{
    if (txtCategoryID.Text != "")
    {
        FoodForm foodForm = new FoodForm();
        foodForm.Show(this);
        foodForm.LoadFood(Convert.ToInt32(txtCategoryID.Text));
    }
}

```

- Nhấn F5 để chạy chương trình. Nhấn nút **btnLoad**, nhấp phải lên một nhóm sản phẩm rồi chọn **Xem danh sách món ăn**.





III. Bài tập

1. Bổ sung 2 button vào FoodForm với chức năng cụ thể như sau:
 - a. Button Save: cho phép người dùng thêm hoặc sửa thông tin trong dgvFood và lưu thông tin cập nhật hoặc thêm mới vào bảng Food
 - b. Button Delete: cho phép xóa dòng được chọn trên dgvFood
2. Thiết kế Form: BillsForm và viết hàm xử lý để
 - a. Hiển thị danh sách hóa đơn được bán trong một khoảng thời gian nào đó (yêu cầu có ô chọn từ ngày đến ngày – sử dụng control DateTimePicker), có hiển thị tổng số tiền chưa giảm giá, tổng số tiền giảm giá, thực thu
 - b. Khi nhấp đúp chuột vào một hóa đơn nào đó thì mở một Form mới (BillDetailsForm) để hiển thị danh mục các mặt hàng mua bởi hóa đơn đó.
3. Thiết kế Form AccountManager và viết hàm xử lý để
 - a. Xem danh sách tài khoản theo nhóm, theo trạng thái (Active?)
 - b. Thêm một tài khoản mới vào cơ sở dữ liệu
 - c. Cập nhật thông tin của một tài khoản
 - d. Reset mật khẩu cho tài khoản
 - e. Click chuột phải vào một tài khoản hiển thị menu sau:

Xóa tài khoản
Xem danh sách vai trò

Trong đó:

 - Nếu chọn Xóa tài khoản thì toàn bộ vai trò của tài khoản này sẽ bị đánh dấu là không kích hoạt (0)
 - Xem danh sách vai trò: Mở một Form mới để hiển thị các vai trò được gán cho tài khoản này
4. Thiết kế Form: MainForm và viết các hàm xử lý để
 - a. Hiển thị danh sách các bàn
 - b. Xem hóa đơn hiện tại của một bàn
 - c. Thêm một bàn mới
 - d. Cập nhật thông tin của bàn
 - e. Xóa một bàn.

f. Khi nhấp phải chuột vào một bàn, hiển thị menu sau

Xóa bàn
Xem danh mục hóa đơn
Xem nhật ký hóa đơn

Trong đó:

- Nếu chọn Xóa bàn thì dữ liệu về bàn đó sẽ bị xóa khỏi cơ sở dữ liệu
- Xem danh mục hóa đơn: Mở một Form mới, phần bên trái của Form là một ListBox chứa ngày lập của các hóa đơn. Khi nhấp chuột vào ngày nào thì hiển thị thông tin chi tiết (cả danh mục sản phẩm được mua) của hóa đơn ở phần bên phải Form.
- Xem nhật ký mua hàng: Liệt kê số lượng hóa đơn, tổng số tiền, tổng thuế, tổng giảm giá của tất cả các hóa đơn, thông tin liên quan đến từng hóa đơn như ngày lập, tên nhân viên lập hóa đơn. (sử dụng ListView hoặc DataGridView)

LAB 7 – TRUYỀN THAM SỐ VÀ THỰC THI THỦ TỤC

Thời lượng: 8 tiết

I. Mục tiêu

Bài thực hành này giúp sinh viên tìm hiểu:

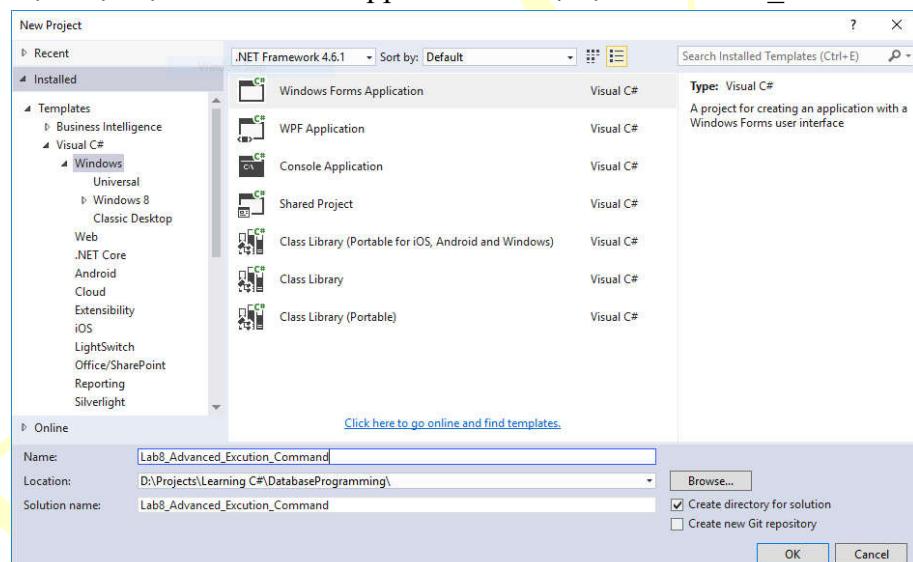
- Cách truyền tham số vào một lệnh truy vấn SQL
- Cách thực thi các lệnh SELECT, INSERT, UPDATE, DELETE có dùng tham số.
- Gọi và thực thi các Stored Procedure

Sau bài thực hành này, sinh viên cần nắm rõ những vấn đề sau:

- Sử dụng đối tượng Parameter để truyền tham số vào các lệnh.
- Cách thực thi và nhận kết quả trả về từ các lệnh truy vấn có tham số
- Cách thực thi các thủ tục và nhận dữ liệu trả về.
- Cách xây dựng ứng dụng trên nền Windows Form.

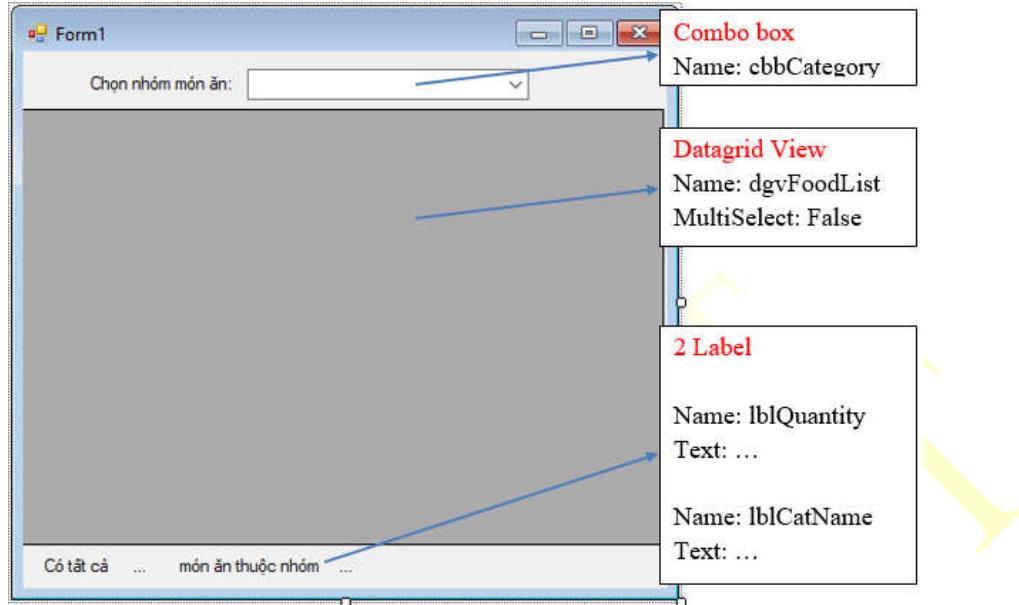
II. Hướng dẫn thực hành

Tạo một dự án Windows Application mới, đặt tên là Lab7_Advanced_Command



Nhấp đúp chuột vào Form1.cs và thiết kế Form có dạng như sau:





Nhấp đôi chuột lên Form1 để tạo phương thức xử lý sự kiện Form1_Load. Tạo hàm để tải danh sách nhóm sản phẩm lên ComboBox và bổ sung đoạn mã sau để xử lý sự kiện Form1_Load.

```

private void LoadCategory()
{
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT ID, Name FROM Category";

    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataTable dt = new DataTable();

    // Mở kết nối
    conn.Open();

    // Lấy dữ liệu từ csdl đưa vào DataTable
    adapter.Fill(dt);

    // Đóng kết nối và giải phóng bộ nhớ
    conn.Close();
    conn.Dispose();

    // Đưa dữ liệu vào Combo Box
    cbbCategory.DataSource = dt;

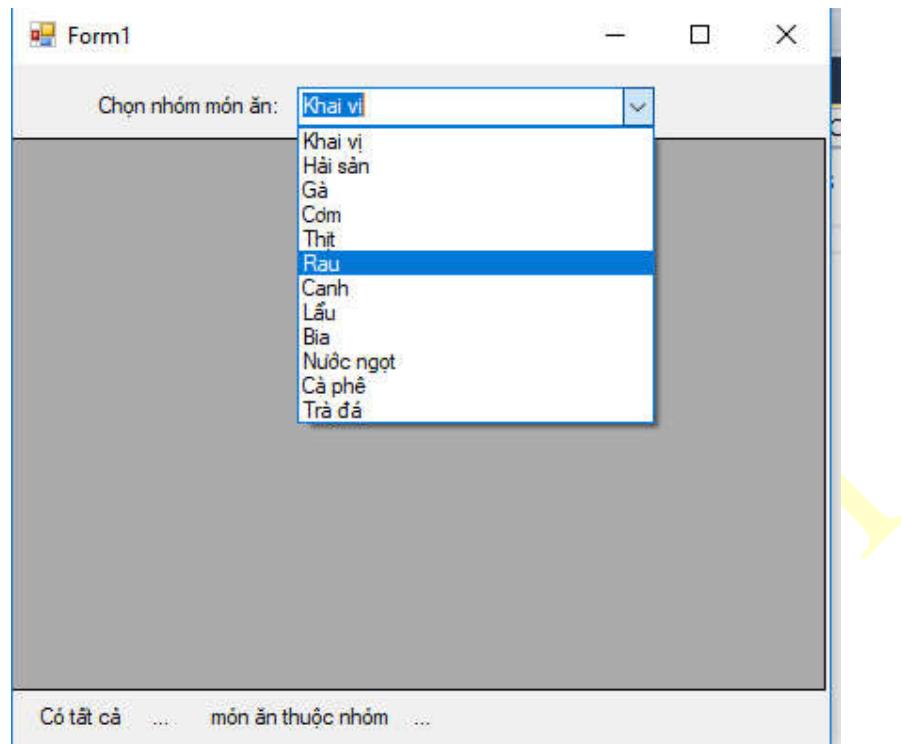
    // Hiển thị tên nhóm sản phẩm
    cbbCategory.DisplayMember = "Name";

    // Nhưng khi lấy giá trị thì lấy ID của nhóm
    cbbCategory.ValueMember = "ID";
}

private void Form1_Load(object sender, EventArgs e)
{
    this.LoadCategory();
}

```

Nhấn F5 để chạy chương trình



6. Truyền tham số vào đối tượng Command

Nhấp phải vào ComboBox, chọn Properties. Trong khung Properties, nhấn nút Events, nhấp đôi chuột vào SelectedIndexChanged. Khai báo biến cục bộ foodTable như sau:

```
public partial class Form1 : Form
{
    private DataTable foodTable;
```

Bổ sung đoạn mã sau vào phương thức cbbCategories_SelectedIndexChanged.

```

private void cbbCategory_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbbCategory.SelectedIndex == -1) return;

    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT * FROM Food WHERE FoodCategoryID = @categoryId";

    // Truyền tham số
    cmd.Parameters.Add("@categoryId", SqlDbType.Int);

    if (cbbCategory.SelectedValue is DataRowView)
    {
        DataRowView rowView = cbbCategory.SelectedValue as DataRowView;
        cmd.Parameters["@categoryId"].Value = rowView["ID"];
    }
    else
    {
        cmd.Parameters["@categoryId"].Value = cbbCategory.SelectedValue;
    }

    // Tạo bộ điều phiếu dữ liệu
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    foodTable = new DataTable();

    // Mở kết nối
    conn.Open();

    // Lấy dữ liệu từ csdl đưa vào DataTable
    adapter.Fill(foodTable);

    // Đóng kết nối và giải phóng bộ nhớ
    conn.Close();
    conn.Dispose();

    // Đưa dữ liệu vào data gridview
    dgvFoodList.DataSource = foodTable;

    // Tính số lượng mẫu tin
    lblQuantity.Text = foodTable.Rows.Count.ToString();
    lblCatName.Text = cbbCategory.Text;
}

```

Nhấn phím F5 để chạy chương trình. Nhấp chuột vào ComboBox, chọn nhóm món ăn để xem danh sách món ăn thuộc nhóm đó.



	ID	Name	Unit	FoodCategoryID	Price	Notes
▶	4	Éch thui rơm	Đĩa	2	70000	
	5	Sò lông nướng m...	Đĩa	2	80000	
	6	Càng cua hấp	Đĩa	2	100000	
	16	test	test	2	30000	
	17	test1	test2	2	30000	
	18	test2	test3	2	30000	
	19	Tom hap bia	Đĩa	2	100000	
	20	Mực nướng tỏi ớt	Đĩa	2	90000	
*						

Có tất cả 8 món ăn thuộc nhóm Hải sản

7. Nhận giá trị trả về từ tham số

Từ thanh Toolboxes, kéo một đối tượng ContextMenuStrip vào Form1. Thiết kế menu có dạng sau (tên của các menu lần lượt là tsmCalculateQuantity, tsmSeparator, tsmAddFood, tsmUpdateFood):

Nhấp phải chuột vào DataGridView, chọn Properties, thiết lập thuộc tính ContextMenuStrip của DataGridView là contextMenuStrip1.

Nhấp đúp chuột lên menu “Tính số lượng đã bán” để tạo phương thức xử lý sự kiện Click cho menu. Bổ sung đoạn mã sau:

```
private void tsmCalculateQuantity_Click(object sender, EventArgs e)
{
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT @numSaleFood = sum(Quantity) FROM BillDetails WHERE FoodID = @foodId";

    // Lấy thông tin sản phẩm được chọn
    if (dgvFoodList.SelectedRows.Count > 0)
    {
        DataGridViewRow selectedRow = dgvFoodList.SelectedRows[0];

        DataRowView rowView = selectedRow.DataBoundItem as DataRowView;

        // Truyền tham số
        cmd.Parameters.Add("@foodId", SqlDbType.Int);
        cmd.Parameters["@foodId"].Value = rowView["ID"];

        cmd.Parameters.Add("@numSaleFood", SqlDbType.Int);
        cmd.Parameters["@numSaleFood"].Direction = ParameterDirection.Output;
```

```

// Mở kết nối csdl
conn.Open();

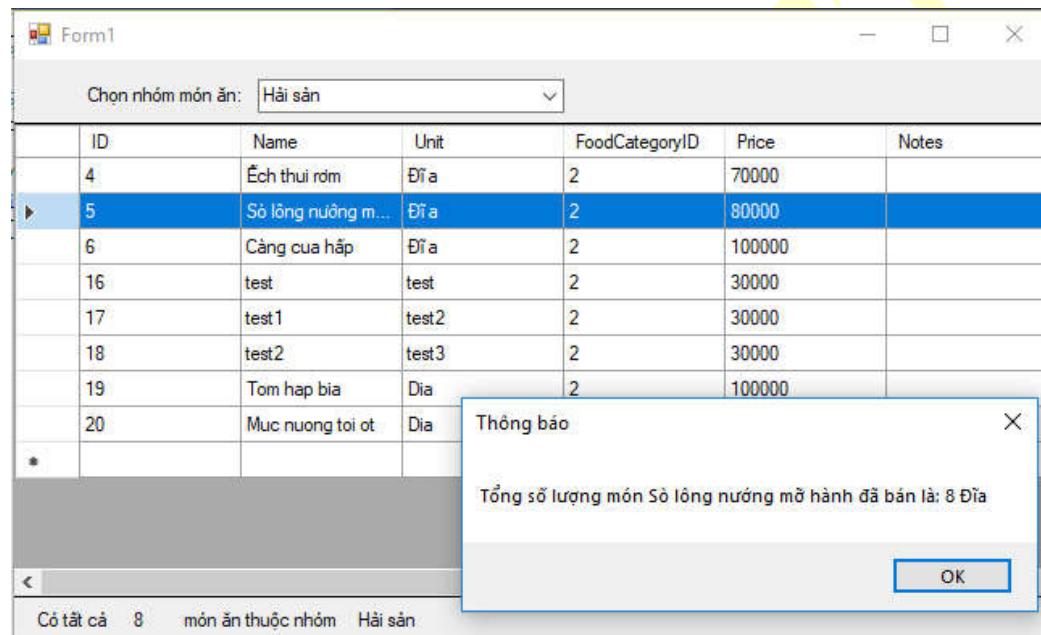
// Thực thi truy vấn và lấy dữ liệu từ tham số
cmd.ExecuteNonQuery();

string result = cmd.Parameters["@numSaleFood"].Value.ToString();
MessageBox.Show("Tổng số lượng món " + rowView["Name"] + " đã bán là: " + result + " " + rowView["Unit"]);

// Đóng kết nối csdl
conn.Close();
}
cmd.Dispose();
conn.Dispose();
}

```

Nhấn F5 chạy chương trình, chọn 1 sản phẩm và click chuột phải, kiểm tra kết quả:



Trở lại Form1, nhấp đôi chuột vào 2 menu còn lại trong contextMenuStrip1 để tạo phương thức xử lý sự kiện Click của chúng như sau:

```

private void tsmAddFood_Click(object sender, EventArgs e)
{
}

private void tsmUpdateFood_Click(object sender, EventArgs e)
{
}

```

8. Thực thi lệnh bằng cách Sử dụng Stored Procedure

Tạo Stored Procedure sau để thực hiện thêm một món ăn mới và cập nhật thông tin món ăn

```

CREATE PROCEDURE [InsertFood]
    @ID int output,
    @Name nvarchar(1000),
    @Unit nvarchar(100),
    @FoodCategoryID int,
    @Price int,
    @Notes nvarchar(3000)
AS
INSERT INTO [Food]
([Name],[Unit],[FoodCategoryID],[Price],[Notes])
VALUES (@Name, @Unit, @FoodCategoryID, @Price,@Notes)

SELECT @ID = SCOPE_IDENTITY();
GO

CREATE PROCEDURE [UpdateFood]
    @ID int,
    @Name nvarchar(1000),
    @Unit nvarchar(100),
    @FoodCategoryID int,
    @Price int,
    @Notes nvarchar(3000)
AS
UPDATE [Food]
SET
    [Name] = @Name,
    [Unit]=@Unit,
    [FoodCategoryID]= @FoodCategoryID,
    [Price]=@Price,
    [Notes]=@Notes

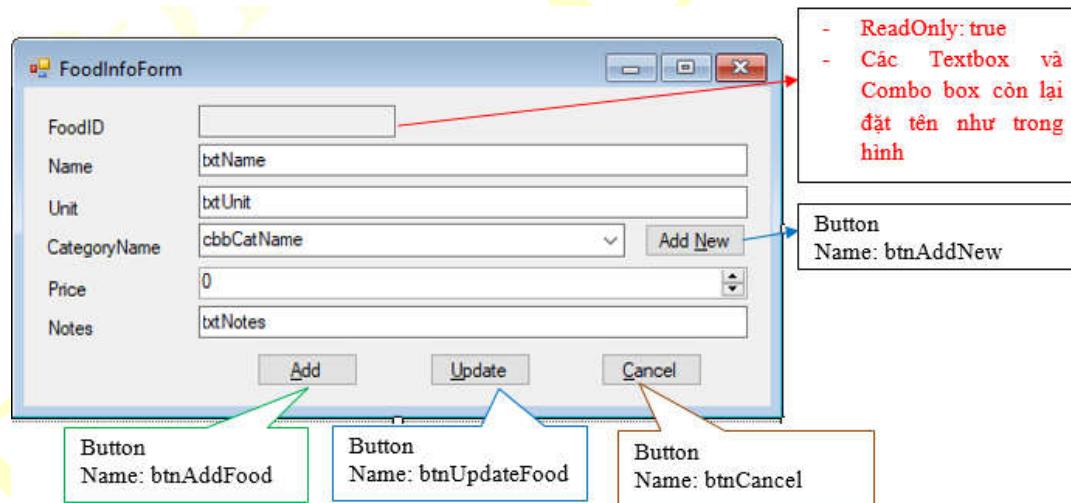
WHERE ID = @ID

IF @@ERROR <> 0
RETURN 0
ELSE
RETURN 1
GO

```

Trong phần này, ta sẽ sử dụng 2 cách khác nhau để gọi một thủ tục trong SQL Server và học cách bắt lỗi SQL (hay ngoại lệ - Exception) bằng C#.

Thêm một Form mới, đặt tên là fOODInfoForm. Thiết kế giao diện cho Form mới như hình sau:



Nhấp đúp chuột vào FoodInfoForm để xử lý phương thức Load. Thêm các hàm sau:

```

private void FoodInfoForm_Load(object sender, EventArgs e)
{
    this.InitValues();
}

```

```

private void InitValues()
{
    string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
    SqlConnection conn = new SqlConnection(connectionString);

    SqlCommand cmd = conn.CreateCommand();
    cmd.CommandText = "SELECT ID, Name FROM Category";

    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataSet ds = new DataSet();

    // mở kết nối
    conn.Open();

    // Lấy dữ liệu từ csdl đưa vào DataTable
    adapter.Fill(ds, "Category");

    // Hiển thị nhóm món ăn
    cbbCatName.DataSource = ds.Tables["Category"];
    cbbCatName.DisplayMember = "Name";
    cbbCatName.ValueMember = "ID";

    // đóng kết nối và giải phóng bộ nhớ
    conn.Close();
    conn.Dispose();
}

```

Bổ sung hàm sau để xóa dữ liệu trên các control của Form

```

private void ResetText()
{
    txtFoodID.ResetText();
    txtName.ResetText();
    txtNotes.ResetText();
    txtUnit.ResetText();
    cbbCatName.ResetText();
    nudPrice.ResetText();
}

```

Nhấp đôi chuột vào nút btnAddFood để xử lý hàm thêm một món ăn mới

```

private void btnAddFood_Click(object sender, EventArgs e)
{
    try
    {
        string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true; ";
        SqlConnection conn = new SqlConnection(connectionString);

        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "EXECUTE InsertFood @id OUTPUT, @name, @unit, @foodCategoryId, @price, @notes";

        // Thêm tham số vào đối tượng Command
        cmd.Parameters.Add("@id", SqlDbType.Int);
        cmd.Parameters.Add("@name", SqlDbType.NVarChar, 1000);
        cmd.Parameters.Add("@unit", SqlDbType.NVarChar, 100);
        cmd.Parameters.Add("@foodCategoryId", SqlDbType.Int);
        cmd.Parameters.Add("@price", SqlDbType.Int);
        cmd.Parameters.Add("@notes", SqlDbType.NVarChar, 3000);

        cmd.Parameters["@id"].Direction = ParameterDirection.Output;

        // Truyền giá trị vào thủ tục qua tham số
        cmd.Parameters["@name"].Value = txtName.Text;
        cmd.Parameters["@unit"].Value = txtUnit.Text;
        cmd.Parameters["@foodCategoryId"].Value = cbbCatName.SelectedValue;
        cmd.Parameters["@price"].Value = nudPrice.Value;
        cmd.Parameters["@notes"].Value = txtNotes.Text;

        //mở kết nối
        conn.Open();

        int numRowAffected = cmd.ExecuteNonQuery();
    }
}

```

```

// Thông báo kết quả
if (numRowAffected > 0)
{
    string foodID = cmd.Parameters["@id"].Value.ToString();

    MessageBox.Show("Successfully adding new food. Food ID = " + foodID, "Message");
    this.ResetText();
}
else
{
    MessageBox.Show("Adding food failed");
}

// đóng kết nối
conn.Close();
conn.Dispose();
}
// Bắt lỗi SQL và các lỗi khác
catch (SqlException exception)
{
    MessageBox.Show(exception.Message, "SQL Error");
}

catch (Exception exception)
{
    MessageBox.Show(exception.Message, "Error");
}
}

```

Bổ sung thêm hàm sau vào lớp FoodInfoForm.

```

public void DisplayFoodInfo(DataRowView rowView)
{
    try
    {
        txtFoodID.Text = rowView["ID"].ToString();
        txtName.Text = rowView["Name"].ToString();
        txtUnit.Text = rowView["Unit"].ToString();
        txtNotes.Text = rowView["Notes"].ToString();
        nudPrice.Text = rowView["Price"].ToString();

        cbbCatName.SelectedIndex = -1;

        // chọn nhóm món ăn tương ứng
        for (int index = 0; index < cbbCatName.Items.Count; index++)
        {
            DataRowView cat = cbbCatName.Items[index] as DataRowView;
            if (cat["ID"].ToString() == rowView["FoodCategoryID"].ToString())
            {
                cbbCatName.SelectedIndex = index;
                break;
            }
        }
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "Error");
        this.Close();
    }
}

```

Nhấp đôi chuột vào nút btnUpdateFood để xử lý hàm cập nhật thông tin món ăn

```

private void btnUpdateFood_Click(object sender, EventArgs e)
{
    try
    {
        string connectionString = "server=.; database = RestaurantManagement; Integrated Security = true";
        SqlConnection conn = new SqlConnection(connectionString);

        SqlCommand cmd = conn.CreateCommand();
        cmd.CommandText = "EXECUTE UpdateFood @id, @name, @unit, @foodCategoryID, @price, @notes";

        // Thêm tham số vào đối tượng Command
        cmd.Parameters.Add("@id", SqlDbType.Int);
        cmd.Parameters.Add("@name", SqlDbType.NVarChar, 1000);
        cmd.Parameters.Add("@unit", SqlDbType.NVarChar, 100);
        cmd.Parameters.Add("@foodCategoryId", SqlDbType.Int);
        cmd.Parameters.Add("@price", SqlDbType.Int);
        cmd.Parameters.Add("@notes", SqlDbType.NVarChar, 3000);

        // Truyền giá trị vào thủ tục qua tham số
        cmd.Parameters["@id"].Value = int.Parse(txtFoodID.Text);
        cmd.Parameters["@name"].Value = txtName.Text;
        cmd.Parameters["@unit"].Value = txtUnit.Text;
        cmd.Parameters["@foodCategoryId"].Value = cbbCatName.SelectedValue;
        cmd.Parameters["@price"].Value = nudPrice.Value;
        cmd.Parameters["@notes"].Value = txtNotes.Text;

        //mở kết nối
        conn.Open();

        int numRowAffected = cmd.ExecuteNonQuery();

        // Thông báo kết quả
        if (numRowAffected > 0)
        {
            MessageBox.Show("Successfully updating food", "Message");
            this.ResetText();
        }
        else
        {
            MessageBox.Show("Updating food failed");
        }

        // đóng kết nối
        conn.Close();
        conn.Dispose();
    }
    // Bắt lỗi SQL và các lỗi khác
    catch (SqlException exception)
    {
        MessageBox.Show(exception.Message, "SQL Error");
    }

    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "Error");
    }
}

```

Nhấp đôi chuột vào nút btnCancel để xử lý việc thoát khỏi Form

```

private void btnCancel_Click(object sender, EventArgs e)
{
    this.Close();
}

Trở lại Form1, bổ sung đoạn mã sau vào phương thức xử lý sự kiện Click của 2
menu Thêm mới và Cập nhật thông tin món ăn

private void tsmAddFood_Click(object sender, EventArgs e)
{
    FoodInfoForm foodForm = new FoodInfoForm();
    foodForm.FormClosed += new FormClosedEventHandler(foodForm_FormClosed);
    foodForm.Show(this);
}

2 references
void foodForm_FormClosed(object sender, FormClosedEventArgs e)
{
    int index = cbbCategory.SelectedIndex;
    cbbCategory.SelectedIndex = -1;
    cbbCategory.SelectedIndex = index;
}

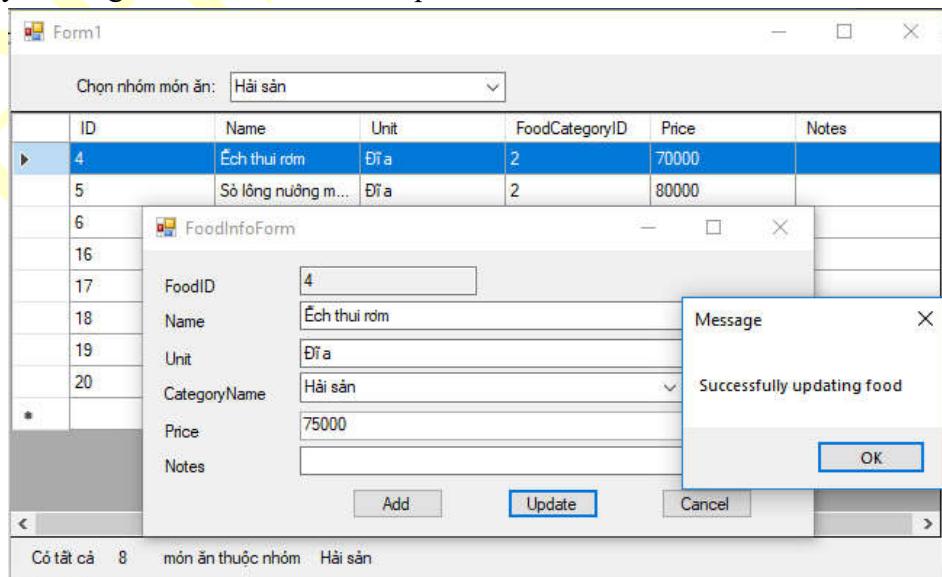
private void tsmUpdateFood_Click(object sender, EventArgs e)
{
    // Lấy thông tin sản phẩm được chọn
    if (dgvFoodList.SelectedRows.Count > 0)
    {
        DataGridViewRow selectedRow = dgvFoodList.SelectedRows[0];
        DataRowView rowView = selectedRow.DataBoundItem as DataRowView;

        FoodInfoForm foodForm = new FoodInfoForm();
        foodForm.FormClosed += new FormClosedEventHandler(foodForm_FormClosed);

        foodForm.Show(this);
        foodForm.DisplayFoodInfo(rowView);
    }
}

```

Chạy chương trình và kiểm tra kết quả



Form1

Chọn nhóm món ăn: Hải sản

	ID	Name	Unit	FoodCategoryID	Price	Notes
▶	4	Ếch thuỷ rán	Đĩa	2	75000	
	5	Sò lông nướng m...	Đĩa	2	80000	
	6	Càng cua hấp	Đĩa	2	100000	
	16	test	test	2	30000	
	17	test1	test2	2	30000	
	18	test2	test3	2	30000	
	19	Tom hap bia	Đĩa	2	100000	
	20	Mực nướng tỏi ớt	Đĩa	2	90000	
*						

< >

Có tất cả 8 món ăn thuộc nhóm Hải sản

Form1

Chọn nhóm món ăn: Lẩu

	ID	Name	Unit	FoodCategoryID	Price	Notes
--	----	------	------	----------------	-------	-------

FoodInfoForm

FoodID:

Name: Lẩu thái

Unit: Lẩu nhỏ

CategoryName: Lẩu

Price: 200000

Notes: Khoảng 3-4 người ăn

Add Update

Message

Successfully adding new food. Food ID = 23

OK

< >

Có tất cả 0 món ăn thuộc nhóm Lẩu

	ID	Name	Unit	FoodCategoryID	Price	Notes
▶	23	Lẩu thái	Lẩu nhỏ	9	200000	Khoảng 3-4 người...
*						

9. Sử dụng DataView để lọc dữ liệu

Bổ sung phần tìm kiếm theo tên món ăn vào Form1 như hình sau

Như vậy, bên cạnh việc hiển thị danh sách món ăn theo nhóm món ăn, người dùng có thể lọc món ăn dựa vào tên bằng cách sử dụng DataView để sắp xếp và trích lọc. Bổ sung sự kiện TextChange cho txtSearchByName như sau:

```

private void txtSearchByName_TextChanged(object sender, EventArgs e)
{
    if (foodTable == null) return;

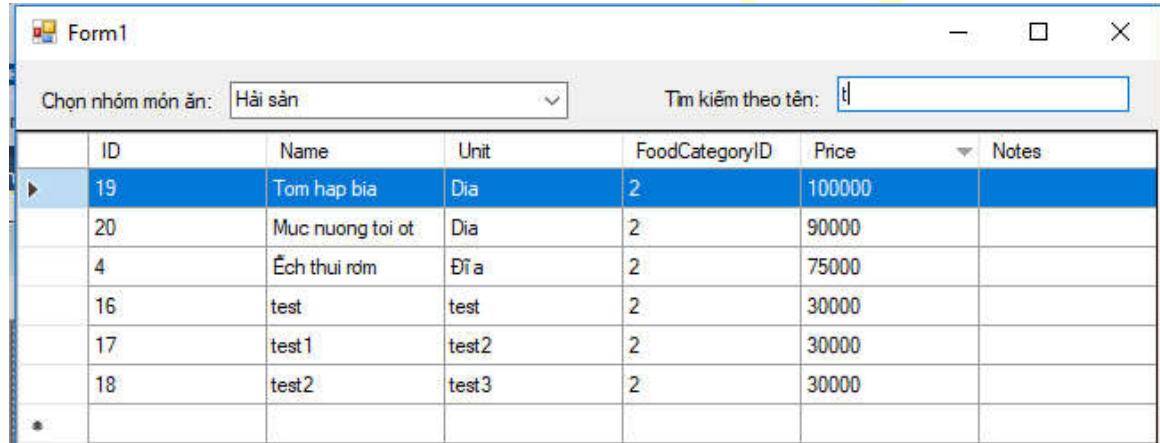
    // create filter and sort expression
    string filterExpression = "Name like '%" + txtSearchByName.Text + "%'";
    string sortExpression = "Price DESC";
    DataViewRowState rowStateFilter = DataViewRowState.OriginalRows;

    // Create a data view object to view the data in foodTable data table
    // filter by Name (contain 'ng') and sort descending by Price
    DataView foodView = new DataView(foodTable,
        filterExpression, sortExpression, rowStateFilter);

    // Assign foodTable as Data Source of data grid view
    dgvFoodList.DataSource = foodView;
}

```

Chạy chương trình, và kiểm tra kết quả



	ID	Name	Unit	FoodCategoryID	Price	Notes
▶	19	Tom hap bia	Đĩa	2	100000	
	20	Mực nướng tỏi ớt	Đĩa	2	90000	
	4	Éch thuỷ tinh	Đĩa	2	75000	
	16	test	test	2	30000	
	17	test1	test2	2	30000	
	18	test2	test3	2	30000	
*						

III. Bài tập

Trong các bài tập sau, yêu cầu sinh viên tạo Stored Procedure và dùng đối tượng Command để gọi, thực thi các thủ tục đó.

1. Trong Form FoodInfoForm, có một Button dùng để thêm mới nhóm món ăn. Hãy thiết kế Form để thêm mới nhóm món ăn. Sau khi nhấn nút Add New để thêm mới nhóm món ăn rồi tắt Form này, nhóm món ăn mới thêm phải được đưa vào ComboBox (cbbCatName).
2. Thiết kế Form: OrdersForm và viết hàm xử lý để
 - a. Hiển thị danh sách hóa đơn được bán trong một khoảng thời gian nào đó (yêu cầu có ô chọn từ ngày nào tới ngày nào – sử dụng 2 DateTimePicker). Hiển thị tổng số tiền chưa giảm giá, số tiền giảm giá, thực thu doanh thu trong ngày đã chọn
 - b. Khi nhập đôi chuột vào một hóa đơn nào đó thì mở một Form mới (OrderDetailsForm) để hiển thị danh mục các mặt hàng mua bởi hóa đơn đó.
3. Thiết kế Form: AccountForm và viết các hàm xử lý để
 - a. Hiển thị danh sách tài khoản

- b. Thêm một tài khoản mới. Có thể thêm vai trò mới trước khi thêm tài khoản bằng cách xử lý sự kiện click nút “Thêm vai trò”
- c. Cập nhật thông tin tài khoản
- d. Reset mật khẩu cho tài khoản
- e. Khi nhấp phải chuột vào một tài khoản, hiển thị menu sau

Xem danh sách các vai trò
Xem nhật ký hoạt động

Trong đó:

- Xem danh sách các vai trò: Mở một Form mới, hiển thị danh sách chi tiết các vai trò chưa hệ thống. Tài khoản được gán vai trò nào thì cột đầu tiên của danh sách sẽ được check. Form này cũng chứa 3 button là AddNew để thêm mới vai trò, Update để thay đổi danh sách vai trò gán cho người dùng, và Cancel để đóng Form.
- Xem nhật ký hoạt động: phần bên trái của Form là một ListBox chứa ngày lập của các hóa đơn. Khi nhấn chuột vào ngày nào thì hiển thị thông tin chi tiết (cả danh mục món ăn) của hóa đơn ở phần bên phải Form. Phía dưới Form, liệt kê số lượng hóa đơn tài khoản đã lập), tổng số tiền của tất cả các hóa đơn.

LAB 8 – MÔ HÌNH ĐA TẦNG

Thời lượng: 8 tiết

I. Mục tiêu

Bài thực hành này giúp sinh viên làm quen với mô hình đa tầng trong phát triển ứng dụng. Bài thực hành sẽ giúp sinh viên xây dựng và phát triển một số chức năng của ứng dụng theo mô hình ba tầng là Tầng giao diện, Tầng xử lý logic và tầng Truy xuất dữ liệu.

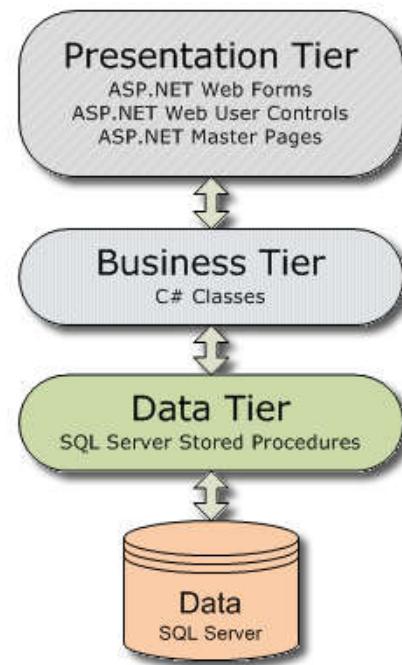
II. Hướng dẫn thực hành

1. Giới thiệu mô hình đa tầng

Mô hình đa tầng (*n-tier*) là một kiến trúc phần mềm được phân chia thành nhiều các thành phần, trong đó các phần như giao diện người dùng (*User Interface - UI*), quy tắc xử lý (*Business Logic - BL*), và lưu trữ dữ liệu (*Data Access - DA*) được phát triển như là những mô đun độc lập. Các mô đun này có thể được xây dựng và phát triển trên các nền tảng riêng biệt, chúng được nối với nhau thông qua việc giao tiếp qua các tập tin dạng thư viện (*.dll).

Khi triển khai ứng dụng ở mức vật lý, kiến trúc đa tầng thường đưa về dạng kiến trúc với ba tầng riêng biệt bao gồm:

- **Tầng giao diện (Presentation):** Dùng để hiển thị các thành phần giao diện và tương tác với người dùng như tiếp nhận thông tin nhập, thông báo kết quả, thông báo lỗi;
- **Tầng xử lý (Business Logic):** Thực hiện các hành động nghiệp vụ của phần mềm như tính toán, thêm, xóa, sửa dữ liệu;
- **Tầng truy cập dữ liệu (Data Access):** Bao gồm hai thành phần: thứ nhất là thành phần trực tiếp truy xuất dữ liệu thường là các lệnh, các hàm trong hệ quản trị cơ sở dữ liệu; Thành phần thứ hai truy xuất là các lớp tổng quát dùng để gọi các hàm và lệnh từ cơ sở dữ liệu.



Để hiểu hơn về mô hình ba tầng, sinh viên có thể tham khảo thêm các link sau:

<https://topdev.vn/blog/mo-hinh-3-lop-la-gi/>

<https://viblo.asia/p/gioi-thieu-mo-hinh-3-lop-trong-c-gDVK2Q9w5Lj>

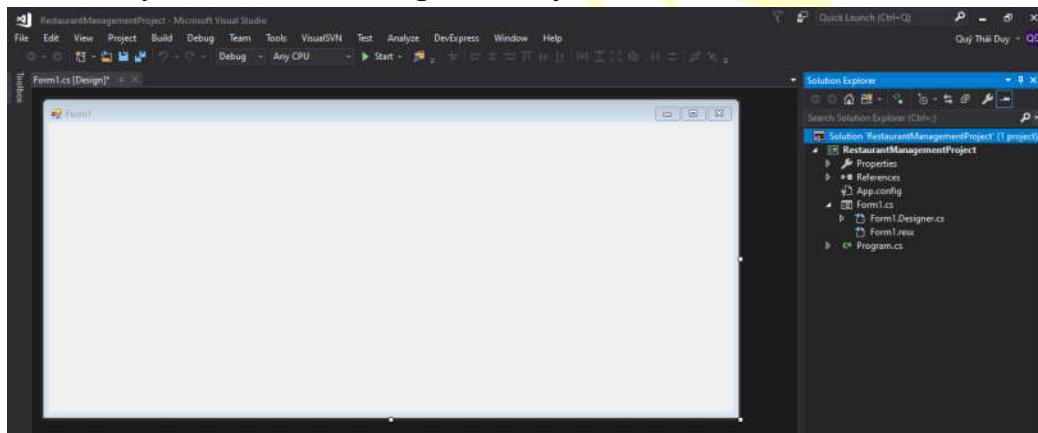
<https://stackify.com/n-tier-architecture/>

<https://docs.microsoft.com/en-us/visualstudio/data-tools/walkthrough-creating-an-n-tier-data-application?view=vs-2019>

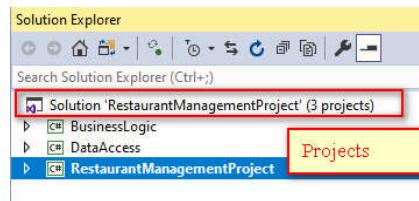
2. Tạo mô hình đa tầng với Visual Studio

Khi xây dựng mô hình đa tầng, Solution được tạo ra khi tạo tầng đầu tiên (*Web, Form*). Các tầng còn lại có thể thêm mới hoặc thêm từ dự án có sẵn (tập tin *.dll). Các tầng được liên kết với nhau thông qua việc kết nối bởi các tập tin thư viện (*.dll). Để tạo dự án với mô hình ba tầng trên Windows Form được thực hiện theo các bước như sau:

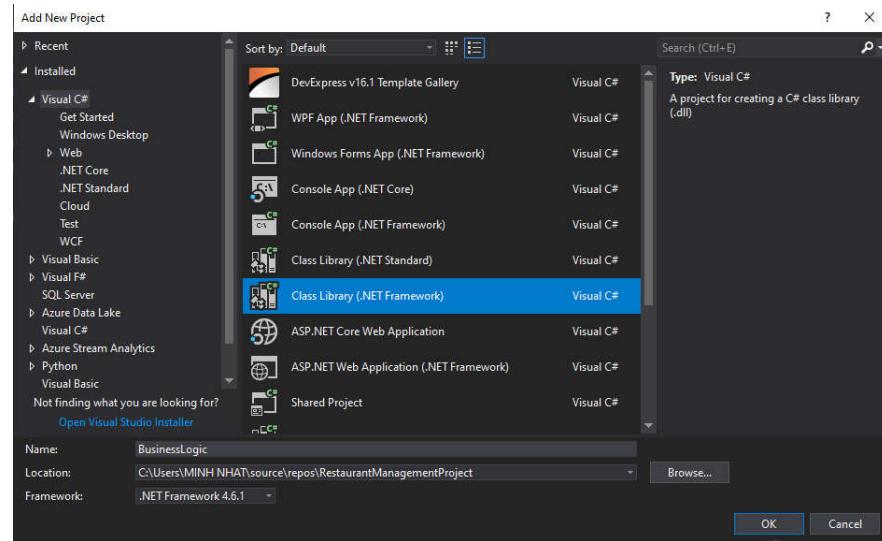
- **Bước 1:** Mở Visual Studio, tạo dự án Windows Form bằng ngôn ngữ C#, đặt tên dự án này là *RestaurantManagementProject*:



Lưu ý: Sau khi tạo xong dự án, ta thấy có một *Solution* và một *Project* được tạo ra. Từ đây, *Solution* là giải pháp chung cho cả dự án, mỗi một *Project* đại diện cho một tầng. Tầng vừa tạo chính là tầng Giao diện, dùng để xử lý các vấn đề liên quan đến giao diện chương trình.

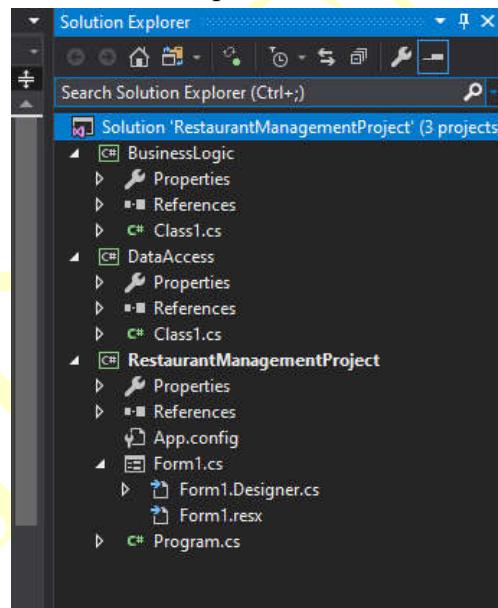


- **Bước 2:** Tạo tầng xử lý bằng cách click phải lên *Solution*, chọn *Add*, chọn *New Project*. Trong hộp thoại hiện lên chọn kiểu dự án là *Class Library (.Net Framework)*, đặt tên dự án là *BusinessLogic*.

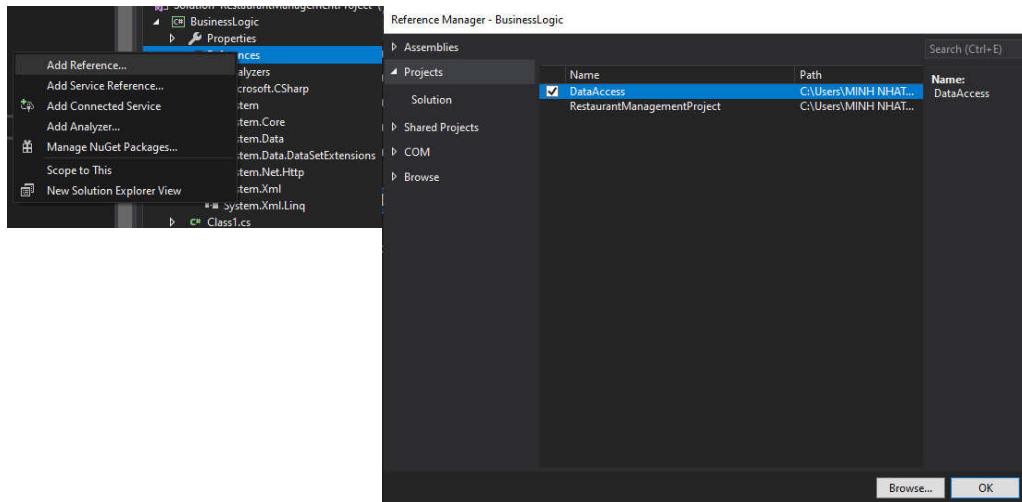


Lưu ý: Đây là một dự án dạng thư viện, thư viện này được tạo ra dưới dạng các tập tin *.dll khi chạy ứng dụng.

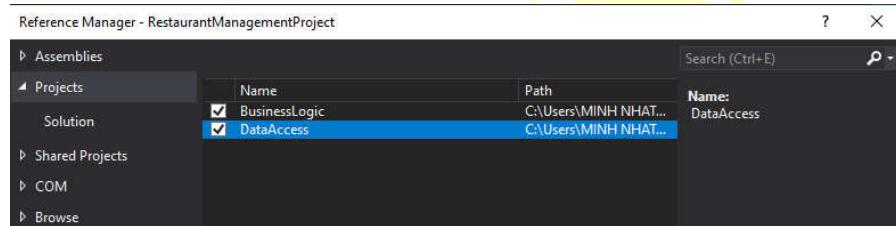
- **Bước 3:** Thực hiện thao tác tương tự Bước 2 để tạo tầng truy cập dữ liệu có tên là *DataAccess*. Kết quả trên Solution Explorer được thể hiện như sau:



- **Bước 4:** Để có thể gọi các phương thức của tầng *DataAccess* từ tầng *BusinessLogic*, click phải lên thành phần *References* của Project *BusinessLogic*, chọn *Add Reference...*, đánh dấu tích và *DataAccess*:



Thực hiện tương tự để kết nối hai tầng *BusinessLayer* và *DataAccess* với tầng giao diện:



Sau khi đã kết nối các tầng, chúng ta có thể gọi các đối tượng, phương thức, và thuộc tính từ các tầng với nhau bằng cách sử dụng lệnh *using* như sau:

`using BusinessLogic;` hoặc `using DataAccess;`

Chi tiết về cách xây dựng các tầng có thể xem tại các mục tiếp theo.

3. Tầng truy cập dữ liệu (*DataAccess*)

Tầng *DataAccess* bao gồm hai thành phần là Cơ sở dữ liệu (*Data*) xây dựng bằng SQL Server và phần Truy xuất (*Access*) xây dựng bằng ngôn ngữ C#.

- **Bước 1: Xây dựng cơ sở dữ liệu**

Cơ sở dữ liệu được dùng trong mô hình này là *RestaurantManagement* (sinh viên xem lại Lab 1 hoặc phần Phụ lục của giáo trình Lập trình cơ sở dữ liệu). Để cho đơn giản, hai bảng được dùng để minh họa là *Category* và *Food*, đây là hai bảng có kết nối khóa ngoại, có giá trị ID tự tăng làm khóa chính. Các chức năng khác, sinh viên dựa theo cách xây dựng từ hai bảng này để phát triển thêm. Các thủ tục (*Store Procedure*) cần xây dựng như sau:

Tên thủ tục	Kiểu trả về	Giải thích
-------------	-------------	------------

Category_GetAll	Toàn bộ mẫu tin bảng Category	Thủ tục này trả về tất cả mẫu tin trong bảng Category.
Category_InsertUpdateDelete	Kiểu số nguyên	Thủ tục này nhận vào các tham số bảng Category, và biến action, nếu action = 0 thì thêm, nếu bằng 1 thì sửa và nếu bằng 2 thì xoá.
Food_GetAll	Toàn bộ mẫu tin bảng Food	Thủ tục này trả về tất cả mẫu tin trong bảng Food.
Food_InsertUpdateDelete	Kiểu số nguyên	Thủ tục này nhận vào các tham số bảng Food, và biến action, nếu action=0 thì thêm, nếu bằng 1 thì sửa và nếu bằng 2 thì xoá.

Mã nguồn của các thủ tục trên được viết trong SQL Server như sau:

```
--Thủ tục lấy tất cả dữ liệu bảng Category
CREATE PROCEDURE [dbo].[Category_GetAll]
AS
```

```
    SELECT * FROM Category
```

```
--Thủ tục lấy tất cả dữ liệu bảng Food
ALTER PROCEDURE [dbo].[Food_GetAll]
AS
```

```
    SELECT * FROM Food
```

```

-- Thủ tục thêm, xóa, sửa bảng Category
ALTER PROCEDURE [dbo].[Category_InsertUpdateDelete]
    @ID int output, -- Biến ID tự tăng, khi thêm xong phải lấy ra
    @Name nvarchar(200),
    @Type int,
    @Action int -- Biến cho biết thêm, xóa, hay sửa
AS
-- Nếu Action = 0, thực hiện thêm dữ liệu
IF @Action = 0
BEGIN
    INSERT INTO [Category] ([Name],[Type])
    VALUES (@Name, @Type)
    SET @ID = @@identity -- Thiết lập ID tự tăng
END
-- Nếu Action = 1, thực hiện cập nhật dữ liệu
ELSE IF @Action = 1
BEGIN
    UPDATE [Category] SET [Name] = @Name, [Type]=@Type
    WHERE [ID] = @ID
END
-- Nếu Action = 2, thực hiện xóa dữ liệu
ELSE IF @Action = 2
BEGIN
    DELETE FROM [Category] WHERE [ID] = @ID
END

-- Thủ tục thêm, xóa, sửa bảng Food
ALTER PROCEDURE [dbo].[Food_InsertUpdateDelete]
    @ID int output, -- Biến ID tự tăng, khi thêm xong phải lấy ra
    @Name nvarchar(1000),
    @Unit nvarchar(100),
    @FoodCategoryID int,
    @Price int,
    @Notes nvarchar(3000),
    @Action int -- Biến cho biết thêm, xóa, hay sửa
AS
IF @Action = 0 -- Nếu Action = 0, thêm dữ liệu
BEGIN
    INSERT INTO [Food]
    ([Name],[Unit],[FoodCategoryID],[Price],[Notes])
    VALUES (@Name, @Unit,@FoodCategoryID,@Price,@Notes)
    SET @ID = @@identity -- Thiết lập ID tự tăng
END
ELSE IF @Action = 1 -- Nếu Action = 1, cập nhật dữ liệu
BEGIN
    UPDATE [Food]
    SET [Name] = @Name,[Unit]=@Unit,[FoodCategoryID]=@FoodCategoryID,
        [Price]=@Price,[Notes]=@Notes
    WHERE [ID] = @ID
END
ELSE IF @Action = 2 -- Nếu Action = 2, xóa dữ liệu
BEGIN
    DELETE FROM [Food] WHERE [ID] = @ID
END

```

- **Bước 2: Xây dựng các lớp tham số chung**

Click phải lên Project *DataAccess*, chọn Add, chọn Class, đặt tên lớp là *Utilities*, viết mã nguồn như sau:

```
public class Utilities
{
    // lấy chuỗi kết nối từ tập tin App.Config
    private static string StrName = "ConnectionStringName";
    public static string ConnectionString = ConfigurationManager
        .ConnectionStrings[StrName]
        .ConnectionString;

    // Các biến của bảng Food
    public static string Food_GetAll = "Food_GetAll";
    public static string Food_InsertUpdateDelete =
        "Food_InsertUpdateDelete";
    // Các biến của bảng Food
    public static string Category_GetAll = "Category_GetAll";
    public static string Category_InsertUpdateDelete =
        "Category_InsertUpdateDelete";
}
;
```

Lưu ý: Để gọi được lớp *ConfigurationManager* cần phải thêm thư viện *System.Configuration* vào dự án (click phải lên *References*, chọn *Add Reference...*, tìm đến thư viện *System.Configuration* và thêm vào dự án), sau đó gọi thư viện này bằng lệnh using:

```
using System.Configuration;
```

- **Bước 3: Xây dựng các lớp ánh xạ bảng Food và Category**

Click phải lên Project *DataAccess*, chọn Add, chọn Class, tạo hai tập tin chứa hai lớp với tên lần lượt là *Food.cs* và *Category.cs*. Khai báo các thuộc tính và kiểu dữ liệu như sau:

```
/// <summary>
/// Lớp ánh xạ bảng Category
/// </summary>
public class Category
{
    // ID của bảng, tự tăng trong CSDL
    public int ID { get; set; }
    // Tên của loại thức ăn
    public string Name { get; set; }
    // Kiểu: 0 là đồ uống; 1 là thức ăn...
    public int Type { get; set; }
}
/// <summary>
/// Lớp ánh xạ bảng Food
/// </summary>
```

```

public class FoodRecord
{
    // ID của bảng Food
    public int ID { get; set; }
    // Tên loại đồ ăn, thức uống
    public string Name { get; set; }
    // Đơn vị tính
    public string Unit { get; set; }
    // Loại thức ăn, ứng với bảng ở trên
    public int FoodCategoryID { get; set; }
    // Giá
    public int Price { get; set; }
    // Ghi chú
    public string Notes { get; set; }
}

```

- **Bước 4: Xây dựng các lớp truy xuất dữ liệu**

Bảng Category có hai thủ tục dùng để lấy tất cả (*GetAll*) và thêm, xoá, sửa mẫu tin (*InsertUpdateDelete*) thì sẽ có hai phương thức tương ứng là *GetAll()* và *Insert_Update_Delete(...)*. Phương thức GetAll thì không truyền tham số và trả về là một danh sách còn phương thức *Insert_Update_Delete* thì truyền vào một đối tượng là ánh xạ của bảng và một biến *action*:

```

//Lớp quản lý Category: DA = DataAccess
public class CategoryDA
{
    //Phương thức lấy hết dữ liệu theo thủ tục Food_GetAll
    public List<Category> GetAll()
    {
        // Khai báo đối tượng SqlConnection và mở kết nối
        // Đối tượng SqlConnection truyền vào chuỗi kết nối trong
        App.config
        SqlConnection sqlConn=new
        SqlConnection(Ultilities.ConnectionString);
        sqlConn.Open();
        //Khai báo đối tượng SqlCommand có kiểu xử lý là
        StoredProcedure
        SqlCommand command = sqlConn.CreateCommand();
        command.CommandType = CommandType.StoredProcedure;
        command.CommandText = Ultilities.Category_GetAll;
        // Đọc dữ liệu, trả về danh sách các đối tượng Category
        SqlDataReader reader = command.ExecuteReader();
        List<Category> list = new List<Category>();
        while (reader.Read())
        {
            Category category = new Category();
            category.ID = Convert.ToInt32(reader["ID"]);
            category.Name = reader["Name"].ToString();
            category.Type = Convert.ToInt32(reader["Type"]);
            list.Add(category);
        }
        // Đóng kết nối và trả về danh sách
    }
}

```

```

        sqlConn.Close();
        return list;
    }
//Phương thức thêm, xoá, sửa theo thủ tục Category_InsertUpdateDelete
public int Insert_Update_Delete(Category category, int action)
{
    // Khai báo đối tượng SqlConnection và mở kết nối
    // Đổi tượng SqlConnection truyền vào chuỗi kết nối trong
    App.config
        SqlConnection sqlConn=new
    SqlConnection(Ultilities.ConnectionString);
        sqlConn.Open();
        //Khai báo đối tượng SqlCommand có kiểu xử lý là
    StoredProcedure
        SqlCommand command = sqlConn.CreateCommand();
        command.CommandType = CommandType.StoredProcedure;
        command.CommandText = Ultilities.Category_InsertUpdateDelete;
        // Thêm các tham số cho thủ tục; Các tham số này chính là các
        tham số trong thủ tục;
        //ID là tham số có giá trị lấy ra khi thêm và truyền vào khi xoá,
        sửa
        SqlParameter IDPara = new SqlParameter("@ID", SqlDbType.Int);
        IDPara.Direction = ParameterDirection.InputOutput; // Vừa vào
        vừa ra
        command.Parameters.Add(IDPara).Value = category.ID;
        command.Parameters.Add("@Name", SqlDbType.NVarChar,200)
            .Value = category.Name;
        command.Parameters.Add("@Type", SqlDbType.Int)
            .Value = category.Type;
        command.Parameters.Add("@Action", SqlDbType.Int)
            .Value = action;
        // Thực thi lệnh
        int result = command.ExecuteNonQuery();
        if (result > 0) // Nếu thành công thì trả về ID đã thêm
            return (int)command.Parameters["@ID"].Value;
        return 0;
    }
}
//Lớp quản lý Food: DA = DataAccess
public class FoodDA
{
    // Phương thức lấy hết dữ liệu theo thủ tục Food_GetAll
    public List<Food> GetAll()
    {
        //Khai báo đối tượng SqlConnection và mở kết nối
        //Đổi tượng SqlConnection truyền vào chuỗi kết nối trong
        App.config
            SqlConnection sqlConn=new
        SqlConnection(Ultilities.ConnectionString);
            sqlConn.Open();
            //Khai báo đối tượng SqlCommand có kiểu xử lý là
        StoredProcedure
            SqlCommand command = sqlConn.CreateCommand();

```

```

        command.CommandType = CommandType.StoredProcedure;
        command.CommandText = Utilities.Food_GetAll;
        // Đọc dữ liệu, trả về danh sách các đối tượng Food
        SqlDataReader reader = command.ExecuteReader();
        List<Food> list = new List<Food>();
        while (reader.Read())
        {
            Food food = new Food();
            food.ID = Convert.ToInt32(reader["ID"]);
            food.Name = reader["Name"].ToString();
            food.Unit = reader["Unit"].ToString();
            food.FoodCategoryID =
                Convert.ToInt32(reader["FoodCategoryID"]);
            food.Price = Convert.ToInt32(reader["Price"]);
            food.Notes = reader["Notes"].ToString();
            list.Add(food);
        }
        // Đóng kết nối và trả về danh sách
        sqlConn.Close();
        return list;
    }
    // Phương thức thêm, xoá, sửa theo thủ tục Food_InsertUpdateDelete
    public int Insert_Update_Delete(Food food, int action)
    {
        // Khai báo đối tượng SqlConnection và mở kết nối
        // Đối tượng SqlConnection truyền vào chuỗi kết nối trong
        App.config
        SqlConnection sqlConn=new
        SqlConnection(Utilities.ConnectionString);
        sqlConn.Open();
        //Khai báo đối tượng SqlCommand có kiểu xử lý là
        StoredProcedure
        SqlCommand command = sqlConn.CreateCommand();
        command.CommandType = CommandType.StoredProcedure;
        command.CommandText = Utilities.Food_InsertUpdateDelete;
        // Thêm các tham số cho thủ tục; Các tham số này chính là các
        tham số trong thủ tục;
        //ID là tham số có giá trị lấy ra khi thêm và truyền vào khi xoá,
        sửa
        SqlParameter IDPara = new SqlParameter("@ID", SqlDbType.Int);
        IDPara.Direction = ParameterDirection.InputOutput;
        command.Parameters.Add(IDPara).Value = food.ID;
        //Các biến còn lại chỉ truyền vào
        command.Parameters.Add("@Name", SqlDbType.NVarChar, 1000)
            .Value = food.Name;
        command.Parameters.Add("@Unit", SqlDbType.NVarChar)
            .Value = food.Unit;
        command.Parameters.Add("@FoodCategoryID", SqlDbType.Int)
            .Value = food.FoodCategoryID;
        command.Parameters.Add("@Price", SqlDbType.Int)
            .Value = food.Price;
        command.Parameters.Add("@Notes", SqlDbType.NVarChar, 3000)
            .Value = food.Notes;
        command.Parameters.Add("@Action", SqlDbType.Int)
            .Value = action;
    }
}

```

```
        int result = command.ExecuteNonQuery();
        // Thực thi lệnh
        if (result > 0) // Nếu thành công thì trả về ID đã thêm
            return (int)command.Parameters["@ID"].Value;
        return 0;
    }
}
```

4. Tầng xử lý Logic (*BusinessLogic*)

Tầng *BusinessLogic* là tầng trung gian giữa tầng *DataAccess* và tầng *Presentation*. Tầng này có thể thêm một số phương thức khác như Tìm kiếm, tìm theo khoá chính, tìm theo trường...

- **Bước 1: Lớp *CategoryBL***

Lớp *CategoryBL* là lớp dùng để thực hiện các chức năng của bảng *Category*, các chức năng chủ yếu của bảng này bao gồm các phương thức cơ bản như: lấy hết, thêm, xoá, sửa...



```
// Lớp CategoryBL có các phương thức xử lý bảng Category
public class CategoryBL
{
    //Đối tượng CategoryDA từ DataAccess
    CategoryDA categoryDA = new CategoryDA();
    //Phương thức lấy hết dữ liệu
    public List<Category> GetAll()
    {
        return categoryDA.GetAll();
    }
    //Phương thức thêm dữ liệu
    public int Insert(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 0);
    }
    //Phương thức cập nhật dữ liệu
    public int Update(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 1);
    }
    //Phương thức xoá dữ liệu truyền vào ID
    public int Delete(Category category)
    {
        return categoryDA.Insert_Update_Delete(category, 2);
    }
}
```

- **Bước 2: Lớp FoodBL**

```
// Lớp FoodBL có các phương thức xử lý bảng Food
public class FoodBL
{
    //Đối tượng CategoryDA từ DataAccess
    FoodDA foodDA = new FoodDA();
    //Phương thức lấy hết dữ liệu
    public List<Food> GetAll()
    {
        return foodDA.GetAll();
    }
    // Phương thức lấy về đối tượng Food theo khoá chính
    public Food GetByID(int ID)
    {
        // Lấy hết
        List<Food> list = GetAll();
        // Duyệt để tìm kiếm
        foreach (var item in list)
        {
            if (item.ID == ID) // Nếu gặp khoá chính
                return item; // thì trả về kết quả
        }
        return null;
    }
    //Phương thức tìm kiếm theo khoá
    public List<Food> Find(string key)
    {
```

```

List<Food> list = GetAll(); // Lấy hết
List<Food> result = new List<Food>();
// Duyệt theo danh sách
foreach (var item in list)
{
    // Nếu từng trường chứa từ khoá
    if (item.ID.ToString().Contains(key)
        || item.Name.Contains(key)
        || item.Unit.Contains(key)
        || item.Price.ToString().Contains(key)
        || item.Notes.Contains(key))
        result.Add(item); // Thì thêm vào danh sách kết quả
}
return result;
}
//Phương thức thêm dữ liệu
public int Insert(Food food)
{
    return foodDA.Insert_Update_Delete(food, 0);
}
//Phương thức cập nhật dữ liệu
public int Update(Food food)
{
    return foodDA.Insert_Update_Delete(food, 1);
}
//Phương thức xoá dữ liệu với ID cho trước
public int Delete(Food food)
{
    return foodDA.Insert_Update_Delete(food, 2);
}
}

```

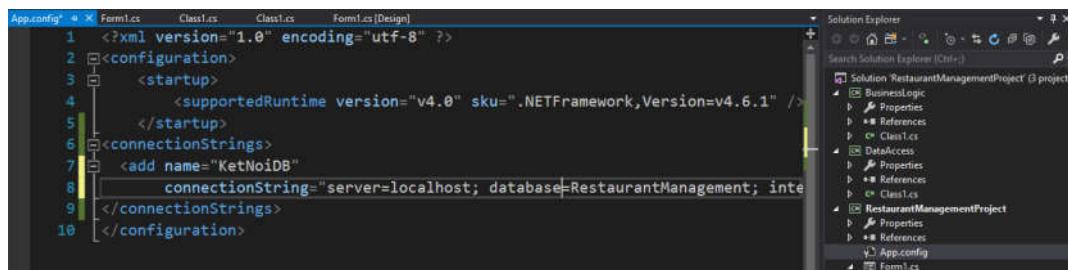
Lớp *CategoryBL* và *FoodBL* nếu có nhu cầu thêm, có thể viết thêm các phương thức khác bổ sung. Khi viết xong tầng *BusinessLogic*, có thể kiểm tra lỗi và tạo tập tin *.dll bằng cách click phải lên Project *BusinessLogic*, chọn *Build*, khi đó nếu có lỗi sai, hệ thống sẽ báo để sửa lỗi trước khi viết qua tầng mới.

5. Tầng Giao diện người dùng (*UserInterface*)

- **Bước 1: Tạo chuỗi kết nối**

Thêm thẻ sau vào trong tập tin *App.config* của dự án:

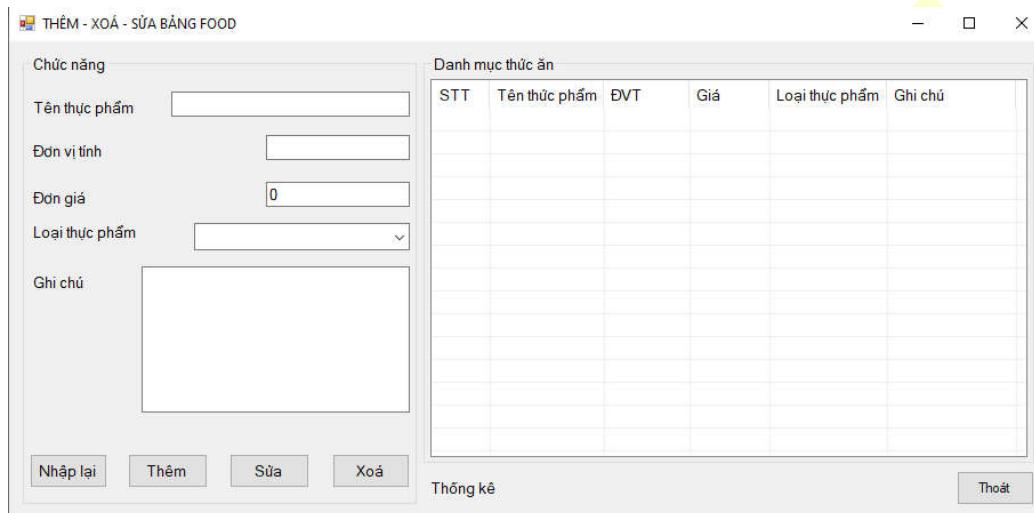
```
<add name="ConnectionStringName" connectionString="server=localhost;
database=RestaurantManagement; integrated security=true;" />
```



Lưu ý: Giá trị “ConnectionStringName” là tên của chuỗi kết nối, các giá trị được in đậm là các giá trị thay đổi theo tên của Server và tên cơ sở dữ liệu trong SQL Server.

- **Bước 2: Giao diện chương trình**

Giao diện chương trình minh họa ở đây dùng để quản lý bảng Food, bao gồm các chức năng chính như: Thêm, xoá, sửa, tìm kiếm... Để hiển thị thông tin, có thể sử dụng ListView hoặc DataGridView, ở đây sử dụng ListView. Dữ liệu Loại thực phẩm được đọc từ bảng Category và đưa vào Combobox:



Các thành phần trên giao diện được đặt các thuộc tính như sau:

Thành phần	Tên thành phần	Thuộc tính
Form	frmFood	Text= THÊM - XOÁ - SỬA BẢNG FOOD
GroupBox	grpLeft	Text = Chức năng; Anchor = Top, Bottom, Left
GroupBox	grpRight	Text = Danh mục thức ăn; Anchor = Top, Bottom, Left, Right
Label	Tên mặc định	Text là tiêu đề cho các thành phần như “Tên thực phẩm”, “Đơn vị tính”, “Đơn giá”, “Loại thực phẩm”, “Ghi chú”
TextBox	txtName txtUnit txtPrice	Anchor= Top, Left
ComboBox	cbbCategory	Anchor= Top, Left
TextBox	txtNotes	Anchor= Top, Left; Multiline = True
Button	cmdClear cmdAdd cmdUpdate cmdDelete	Anchor= Bottom; Text là tiêu đề cho các nút bấm như “Nhập lại”, “Thêm”, “Sửa”, “Xoá”.
ListView	lsvFood	Anchor= Top, Bottom, Left, Right; Columns={STT, Tên thực phẩm, ĐVT, Giá, Loại thực phẩm, Ghi chú};

		FullRowSelect= true;
		GridLines= true;
		MultiSelect= false;
		View= Details
Label	lblStatistic	Text= Thông kê
Button	cmdExit	Text= Thoát;
		Anchor= Bottom, Right

Sự kiện cho nút Thoát và Nhập lại được viết như sau:

```
private void cmdExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
private void cmdClear_Click(object sender, EventArgs e)
{
    //Gán các ô bằng giá trị mặc định
    txtName.Text = "";
    txtPrice.Text = "0";
    txtUnit.Text = "";
    txtNotes.Text = "";
    // Thiết lập index = 0 cho ComboBox
    if(cbbCategory.Items.Count >0)
        cbbCategory.SelectedIndex = 0;
}
```

- **Bước 3: Xây dựng các phương thức tự động tải dữ liệu**

Để gọi các đối tượng từ tầng *DataAccess* và *BusinessLogic*, phải sử dụng lệnh using như sau trong lớp frmFood:

```
using BusinessLogic;
using DataAccess;
```

Đầu tiên, khai báo các đối tượng toàn cục như sau trong lớp frmFood:

```
// Danh sách toàn cục bảng Category
List<Category> listCategory = new List<Category>();
// Danh sách toàn cục bảng Food
List<Food> listFood = new List<Food>();
// Đối tượng Food đang chọn hiện hành
Food foodCurrent = new Food();
```

Tiếp theo, xây dựng các phương thức tự động tải dữ liệu và gọi trong sự kiện Form_Load:

```
private void Form1_Load(object sender, EventArgs e)
{
    //Đổ dữ liệu vào ComboBox
    LoadCategory();
    // Đổ dữ liệu vào ListView
    LoadFoodDataToListView();
}
```

```

private void LoadCategory()
{
    //Gọi đối tượng CategoryBL từ tầng BusinessLogic
    CategoryBL categoryBL = new CategoryBL();
    // Lấy dữ liệu gán cho biến toàn cục listCategory
    listCategory = categoryBL.GetAll();
    // Chuyển vào Combobox với dữ liệu là ID, hiển thị là Name
    cbbCategory.DataSource = listCategory;
    cbbCategory.ValueMember = "ID";
    cbbCategory.DisplayMember = "Name";
}
public void LoadFoodDataToListView()
{
    //Gọi đối tượng FoodBL từ tầng BusinessLogic
    FoodBL foodBL = new FoodBL();
    // Lấy dữ liệu
    listFood= foodBL.GetAll();
    int count = 1; // Biến số thứ tự
    // Xóa dữ liệu trong ListView
    lsvFood.Items.Clear();
    // Duyệt mảng dữ liệu để đưa vào ListView
    foreach (var food in listFood)
    {
        // Số thứ tự
        ListViewItem item = lsvFood.Items.Add(count.ToString());
        // Đưa dữ liệu Name, Unit, price vào cột tiếp theo
        item.SubItems.Add(food.Name);
        item.SubItems.Add(food.Unit);
        item.SubItems.Add(food.Price.ToString());
        // Theo dữ liệu của bảng Category ID, lấy Name để hiển thị
        string foodName = listCategory
            .Find(x => x.ID ==
                food.FoodCategoryID).Name;
        item.SubItems.Add(foodName);
        // Đưa dữ liệu Notes vào cột cuối
        item.SubItems.Add(food.Notes);
        count++;
    }
}

```

- **Bước 4: Sự kiện click lên dòng của ListView**

Khi người dùng click chuột vào một dòng trên *ListView*, dữ liệu sẽ được lấy ra và gán cho biến của đối tượng *Food* hiện hành, đồng thời giá trị của các trường được đưa vào các ô nhập. Trong sự kiện *OnClick* của *ListView* mã nguồn được viết như sau:

```

private void lsvFood_Click(object sender, EventArgs e)
{
    // Duyệt toàn bộ dữ liệu trong ListView
    for (int i = 0; i < lsvFood.Items.Count; i++)
    {
        // Nếu có dòng được chọn thì lấy dòng đó

```

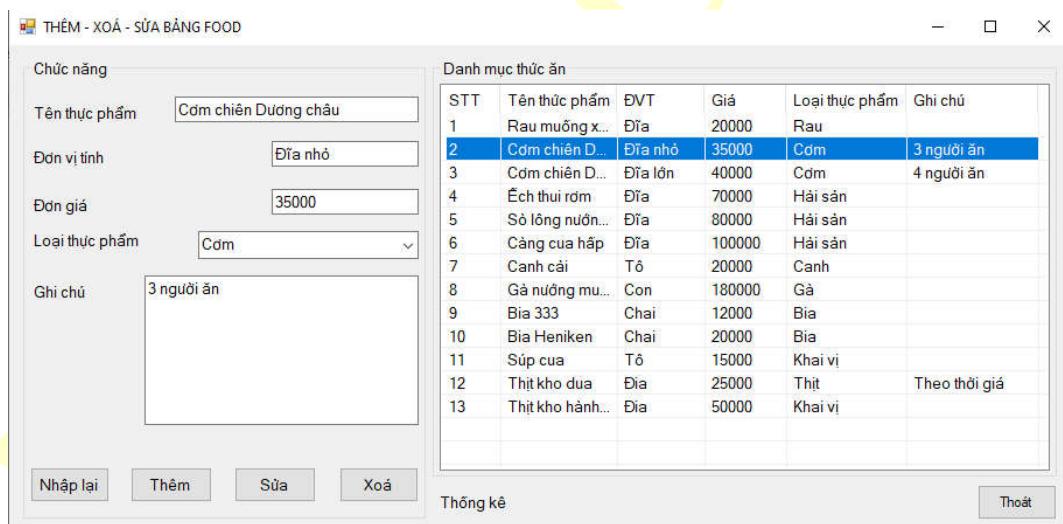
```

if (lsvFood.Items[i].Selected)
{
    // Lấy các tham số và gán dữ liệu vào các ô
    foodCurrent = listFood[i];
    txtName.Text = foodCurrent.Name;
    txtUnit.Text = foodCurrent.Unit;
    txtPrice.Text = foodCurrent.Price.ToString();
    txtNotes.Text = foodCurrent.Notes;
    // Lấy index của Combobox theo FoodCategoryID
    cbbCategory.SelectedIndex = listCategory
        .FindIndex(x => x.ID ==
            foodCurrent.FoodCategoryID);
}
}

```

Lưu ý: Do đã thiết lập từ trước các thuộc tính của *ListView* là *FullRowSelect*=
true và *MultiSelect*= false (xem bảng ở Bước 2) nên khi người dùng nhấp chuột vào
vùng dữ liệu thì chỉ 1 dòng được chọn và dòng đó phải chọn cả dòng.

Sau khi viết hết lệnh, nhấn F5 và chạy thử chương trình để xem dữ liệu được hiển
thị trong *ListView* và *ComboBox*. Click chọn dòng trong *ListView* để dữ liệu được đưa
vào trong các ô như sau:



- **Bước 5: Xử lý nút thêm**

Trước tiên, viết phương thức thêm dữ liệu:

```

/// <summary>
/// Phương thức thêm dữ liệu cho bảng Food
/// </summary>
/// <returns>Trả về số dương nếu thành công, ngược lại trả về số
/// âm</returns>
public int InsertFood()
{

```

```

//Khai báo đối tượng Food từ tầng DataAccess
Food food = new Food();
food.ID = 0;
// Kiểm tra nếu các ô nhập khác rỗng
if (txtName.Text == "" || txtUnit.Text == "" || txtPrice.Text ==
    "")
    MessageBox.Show("Chưa nhập dữ liệu cho các ô, vui lòng nhập
lại");
else {
    //Nhận giá trị Name, Unit, và Notes từ người dùng nhập vào
    food.Name = txtName.Text;
    food.Unit = txtUnit.Text;
    food.Notes = txtNotes.Text;
    // Giá trị price là giá trị số nên cần bắt lỗi khi người dùng
nhập sai
    int price = 0;
    try
    {
        // Cố gắng lấy giá trị
        price = int.Parse(txtPrice.Text);
    }
    catch
    {
        // Nếu sai, gán giá = 0
        price = 0;
    }
    food.Price = price;
    // Giá trị FoodCategoryID được lấy từ ComboBox
    food.FoodCategoryID =
int.Parse(cbbCategory.SelectedValue.ToString());
    // Khao báo đối tượng FoodBL từ tầng Business
    FoodBL foodBL = new FoodBL();
    // Chèn dữ liệu vào bảng
    return foodBL.Insert(food);
}
return -1;
}

```

Lưu ý: Vì ID là tự tăng nên giá trị nhận vào được gán mặc định (bằng 0).

Sự kiện khi nhấn nút Thêm được viết như sau:

```

private void cmdAdd_Click(object sender, EventArgs e)
{
    // Gọi phương thức thêm dữ liệu
    int result = InsertFood();
    if(result > 0) // Nếu thêm thành công
    {
        // Thông báo kết quả
        MessageBox.Show("Thêm dữ liệu thành công");
        // Tải lại dữ liệu cho ListView
        LoadFoodDataToListView();
    }
    // Nếu thêm không thành công thì thông báo cho người dùng
}

```

```
        else MessageBox.Show("Thêm dữ liệu không thành công. Vui lòng kiểm  
tra lại dữ liệu nhập");  
    }
```

DHTL-CNTT

- **Bước 6: Xử lý nút xoá**

```

private void cmdDelete_Click(object sender, EventArgs e)
{
    // Hỏi người dùng có chắc chắn xoá hay không? Nếu đồng ý thì
    if(MessageBox.Show("Bạn có chắc chắn muốn xoá mẫu tin này?", "Thông
báo",
        MessageBoxButtons.YesNo, MessageBoxIcon.Warning) ==
DialogResult.Yes)
    {
        // Khai báo đối tượng FoodBL từ BusinessLogic
        FoodBL foodBL = new FoodBL();
        if (foodBL.Delete(foodCurrent) > 0)// Nếu xoá thành công
        {
            MessageBox.Show("Xoá thực phẩm thành công");
            // Tải dữ liệu lên ListView
            LoadFoodDataToListView();
        }
        else MessageBox.Show("Xoá không thành công");
    }
}

```

Lưu ý: Thuộc tính *foodCurrent* là thuộc tính được khai báo ở Bước 3 và mỗi khi Click chuột lên dòng của ListView thì thuộc tính này sẽ lưu giữ giá trị được chọn.

- **Bước 7: Xử lý nút sửa**

Khi nhấn vào nút Sửa, hệ thống sẽ lấy giá trị hiện hành (biến *foodCurrent*) làm giá trị để sửa, ID được giữ lại, các giá trị còn lại được cập nhật bằng cách dựa trên dữ liệu mà người dùng nhập vào. Phương thức cập nhật được viết như sau:

```

/// <summary>
/// Phương thức cập nhật dữ liệu cho bảng Food
/// </summary>
/// <returns>Trả về dương nếu cập nhật thành công, ngược lại là số
âm</returns>
public int UpdateFood()
{
    //Khai báo đối tượng Food và lấy đối tượng hiện hành
    Food food = foodCurrent;
    // Kiểm tra nếu các ô nhập khác rỗng
    if (txtName.Text == "" || txtUnit.Text == "" || txtPrice.Text ==
    "")
        MessageBox.Show("Chưa nhập dữ liệu cho các ô, vui lòng nhập
lại");
    else
    {
        //Nhận giá trị Name, Unit, và Notes khi người dùng sửa
        food.Name = txtName.Text;
        food.Unit = txtUnit.Text;
        food.Notes = txtNotes.Text;
        //Giá trị price là giá trị số nên cần bắt lỗi khi người dùng
nhập sai
        int price = 0;
    }
}

```

```

try
{
    // Chuyển giá trị từ kiểu văn bản qua kiểu int
    price = int.Parse(txtPrice.Text);
}
catch
{
    // Nếu sai, gán giá = 0
    price = 0;
}
food.Price = price;
// Giá trị FoodCategoryID được lấy từ ComboBox
food.FoodCategoryID =
int.Parse(cbbCategory.SelectedValue.ToString());
// Khao báo đối tượng FoodBL từ tầng Business
FoodBL foodBL = new FoodBL();
// Cập nhật dữ liệu trong bảng
return foodBL.Update(food);
}
return -1;
}

```

Sự kiện khi người dùng click vào nút sửa được viết như sau:

```

private void cmdUpdate_Click(object sender, EventArgs e)
{
    // Gọi phương thức cập nhật dữ liệu
    int result = UpdateFood();
    if (result > 0) // Nếu cập nhật thành công
    {
        // Thông báo kết quả
        MessageBox.Show("Cập nhật dữ liệu thành công");
        // Tải lại dữ liệu cho ListView
        LoadFoodDataToListView();
    }
    // Nếu thêm không thành công thì thông báo cho người dùng
    else MessageBox.Show("Cập nhật dữ liệu không thành công. Vui lòng
kiểm tra lại dữ liệu nhập");
}

```

III. Bài tập

- Từ các bước minh họa ở trên, xây dựng chương trình mô hình ba tầng cho bảng Category và bảng Food.
- Xây dựng hết các chức năng của chương trình cho phép nhập liệu tất cả các bảng (Xem cơ sở dữ liệu phần Phụ lục trong giáo trình).
- Xây dựng phần phân quyền, cho phép gán các quyền như: Quản lý, Kế toán, Nhân viên, Admin.
- Xây dựng chương trình quản lý nhà hàng hoàn chỉnh với các chức năng như: Đặt món, tách bàn, thanh toán theo mô hình 3 tầng như trên.

--Hết--

LAB 9 – SỬ DỤNG ENTITY FRAMEWORK

Thời lượng: 8 tiết

I. Mục tiêu

Bài thực hành này giúp sinh viên tìm hiểu cách sử dụng Entity Framework để kết nối tới cơ sở dữ liệu và thực hiện các truy vấn đơn giản:

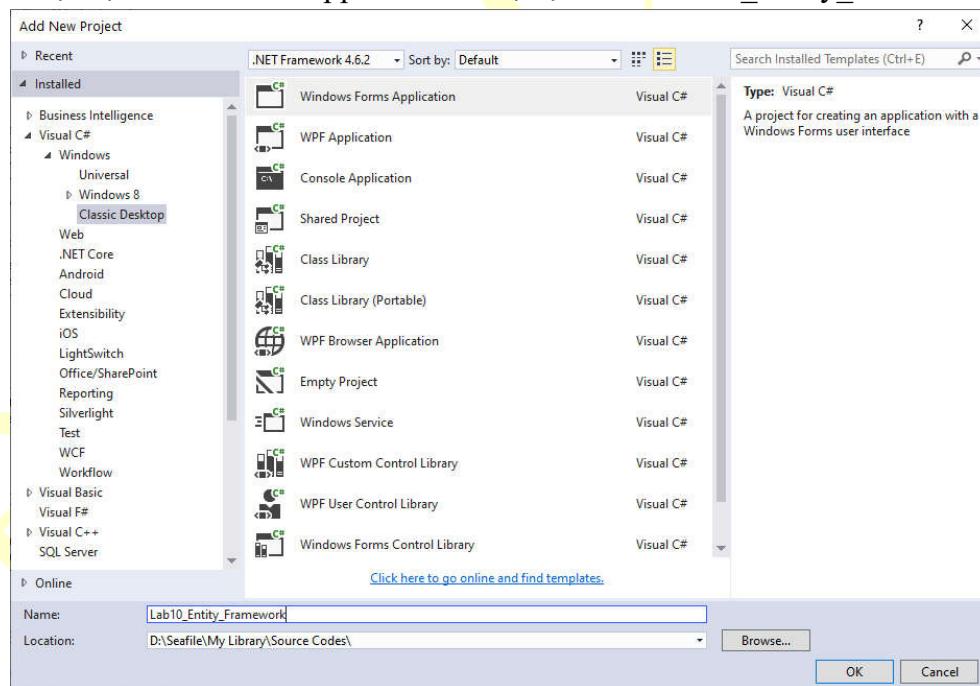
- SELECT: Lấy các mẫu tin từ một bảng.
- INSERT: Thêm một mẫu tin mới vào một bảng.
- UPDATE: Cập nhật một mẫu tin có sẵn trong bảng.
- DELETE: Xóa một mẫu tin khỏi bảng.

Sau bài thực hành này, sinh viên cần nắm rõ những vấn đề sau:

- Các thành phần của chuỗi kết nối, ý nghĩa của chúng và cách tạo chuỗi kết nối.
- Cách định nghĩa các lớp thực thể và lớp ngũ cảnh.
- Cách sử dụng đối tượng Context để thực thi truy vấn, thêm, cập nhật dữ liệu vào cơ sở dữ liệu.
- Cách sử dụng lớp DTO để lấy thông tin từ nhiều bảng.
- Cách xây dựng ứng dụng trên nền Windows Form với Entity Framework.

II. Hướng dẫn thực hành

Tạo một dự án Windows Application mới, đặt tên là Lab09_Entity_Framework



Đặt lại tên Form1 thành MainForm. Kéo một điều khiển ToolTip vào MainForm. Thiết kế MainForm như sau:

MainForm:

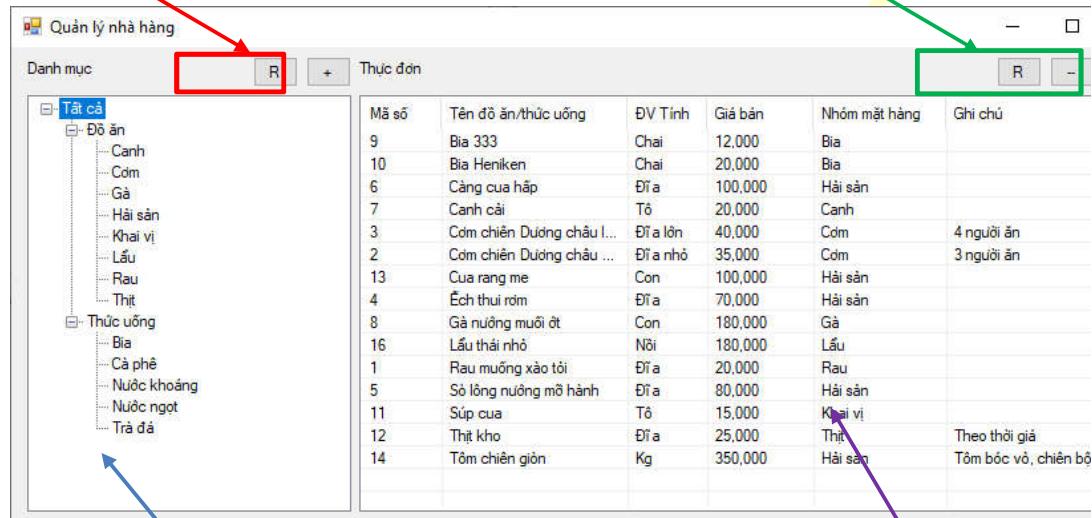
- Text: Quản lý nhà hàng

Button R:

- Name: btnReloadFood

- StartPosition: CenterScreen
- Button R:
 - Name: btnReloadCategory
 - ToolTip on toolTip1: Tải lại danh mục
 - Text: R
- Button +:
 - Name: btnAddCategory
 - ToolTip on toolTip1: Thêm danh mục mới
 - Text: +

- ToolTip: Tải lại danh sách món ăn
- Button --:
 - Name: btnDelete
 - ToolTip: Xóa món ăn được chọn
 - Text: --
- Name: btnAddFood
- ToolTip: Thêm món ăn mới
- Cả 3 buttons: Anchor: Top, Right



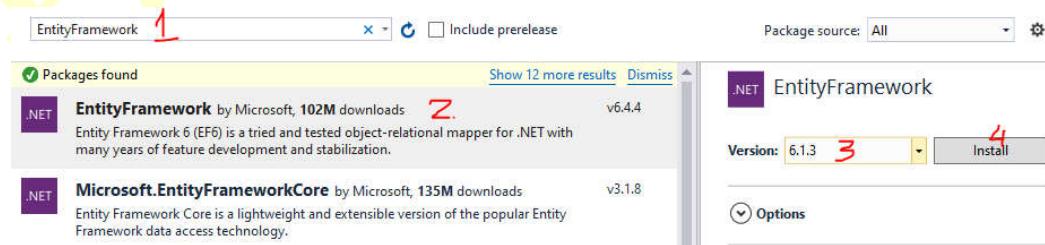
TreeView:

- Name: tvwCategory
- Anchor: Top, Bottom, Left

ListView:

- Name: lvwFood
- Anchor: Top, Bottom, Left, Right
- FullRowSelect: True
- GridLines: True
- MultiSelect: False
- View: Details

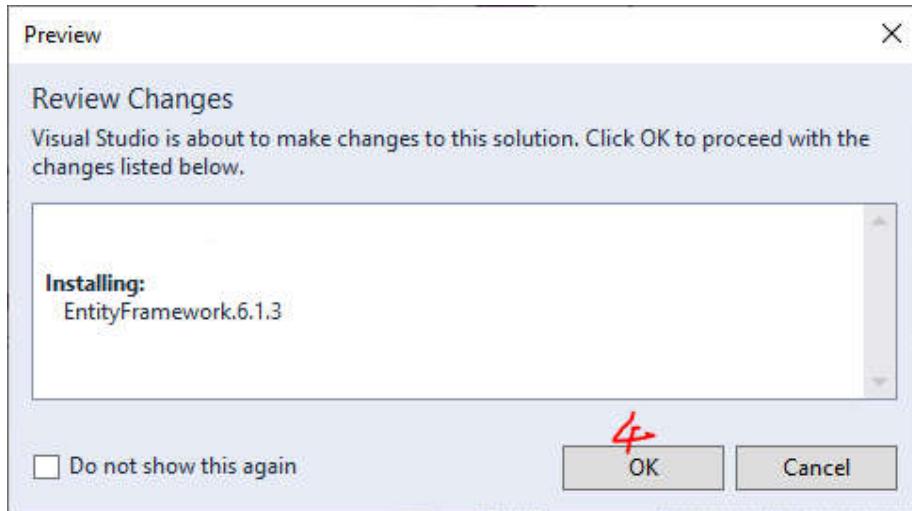
1. Cài đặt gói thư viện EntityFramework



Nhấp phải chuột vào dự án, chọn Manage Nuget Packages ...

- Trong cửa sổ mới “NuGet: Lab09_Entity_Framework”, chọn tab Browse. Trong mục Package source, chọn All. Nhập từ khóa EntityFramework vào ô tìm kiếm.
- Trong cửa sổ hiển thị kết quả tìm kiếm, chọn EntityFramework.

- Ở khung kê bên phải, chọn phiên bản muôn cài đặt là 6.1.3. Sau đó nhấn nút Install.
- Trong cửa sổ popup Preview, nhấn nút OK để xác nhận và bắt đầu cài đặt thư viện



2. Định nghĩa các lớp thực thể và lớp ngữ cảnh (DbContext)

Nhấp phải chuột vào tên dự án, chọn Add > New Folder. Đặt tên thư mục là Models.

Trong thư mục Models, tạo ra các lớp thực thể sau đây:

- Lớp Category.cs: Biểu diễn thông tin nhóm đồ ăn, thức uống.

```
8 references | 0 changes | 0 authors, 0 changes
public class Category
{
    5 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }

    9 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }

    6 references | 0 changes | 0 authors, 0 changes
    public CategoryType Type { get; set; }
}

7 references | 0 changes | 0 authors, 0 changes
public enum CategoryType
{
    Drink,
    Food
}
```

- Lớp Food.cs: Biểu diễn thông tin về một món ăn, đồ uống.

```

5 references | 0 changes | 0 authors, 0 changes
public class Food
{
    4 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }

    8 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }

    6 references | 0 changes | 0 authors, 0 changes
    public string Unit { get; set; }

    6 references | 0 changes | 0 authors, 0 changes
    public int FoodCategoryId { get; set; }

    6 references | 0 changes | 0 authors, 0 changes
    public int Price { get; set; }

    6 references | 0 changes | 0 authors, 0 changes
    public string Notes { get; set; }

    4 references | 0 changes | 0 authors, 0 changes
    public Category Category { get; set; }
}

```

Tiếp theo, để có thể truy xuất tới cơ sở dữ liệu, ta cần định nghĩa lớp ngũ cảnh RestaurantContext.cs trong thư mục Models như sau:

```

8 references | 0 changes | 0 authors, 0 changes
public class RestaurantContext : DbContext
{
    // Tham chiếu tới các nhóm món ăn trong bảng Category
    4 references | 0 changes | 0 authors, 0 changes
    public DbSet<Category> Categories { get; set; }

    // Tham chiếu tới các món ăn, đồ uống trong bảng Food
    6 references | 0 changes | 0 authors, 0 changes
    public DbSet<Food> Foods { get; set; }

    0 references | 0 changes | 0 authors, 0 changes
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // Xóa bỏ quy tắc sử dụng danh từ số nhiều cho tên bảng
        // Lúc này, thuộc tính Categories ánh xạ tới bảng Category trong db
        // Và thuộc tính Foods tương ứng với bảng Food trong cơ sở dữ liệu
        modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

        // Định nghĩa mối quan hệ một nhiều giữa hai bảng Category và Food
        modelBuilder.Entity<Food>()
            .HasRequired(x => x.Category)
            .WithMany()
            .HasForeignKey(x => x.FoodCategoryId)
            .WillCascadeOnDelete(true);
    }
}

```

Và cuối cùng, để lấy thông tin chi tiết món ăn từ cả hai bảng Food và Category, ta tạo một lớp DTO (Data Transfer Object) làm trung gian và chứa những thuộc tính cần thiết.

Nhấp phải chuột vào thư mục Models, chọn Add > Class ... Đặt tên là FoodModel. Định nghĩa các thuộc tính cho lớp FoodModel như sau:

```
6 references | 0 changes | 0 authors, 0 changes
public class FoodModel
{
    3 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }

    3 references | 0 changes | 0 authors, 0 changes
    public string Name { get; set; }

    3 references | 0 changes | 0 authors, 0 changes
    public string Unit { get; set; }

    3 references | 0 changes | 0 authors, 0 changes
    public string CategoryName { get; set; }

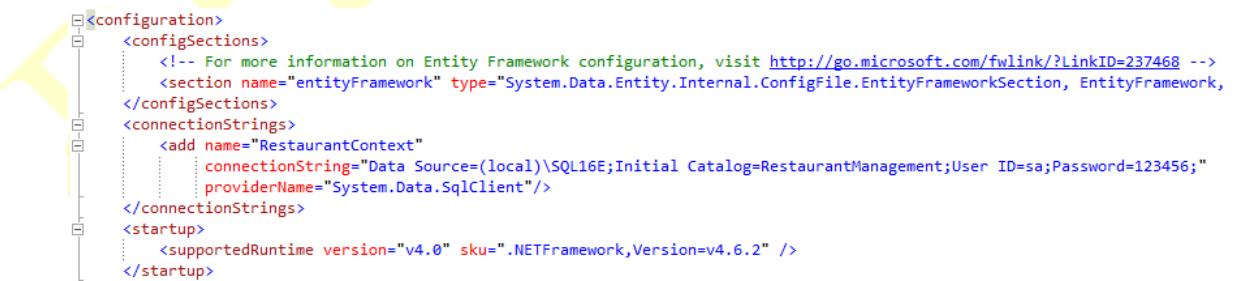
    3 references | 0 changes | 0 authors, 0 changes
    public int Price { get; set; }

    3 references | 0 changes | 0 authors, 0 changes
    public string Notes { get; set; }
}
```

3. Định nghĩa chuỗi thông tin kết nối tới cơ sở dữ liệu

Nhấp đôi chuột vào tập tin App.config để mở nó trong cửa sổ soạn thảo. Nếu chưa có, nhấp phải chuột vào tên dự án, chọn Add > New Item. Trong cửa sổ Add New Item, chọn Visual C# Items > General > Application Configuration File rồi nhấn nút Add.

Trong tập tin App.config, bổ sung thêm thẻ `<connectionStrings>` để thêm chuỗi kết nối tới cơ sở dữ liệu như trong hình sau:



```
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkId=237468 -->
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
  </configSections>
  <connectionStrings>
    <add name="RestaurantContext"
        connectionString="Data Source=(local)\SQL16E;Initial Catalog=RestaurantManagement;User ID=sa;Password=123456;" 
        providerName="System.Data.SqlClient"/>
  </connectionStrings>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.2" />
  </startup>
</configuration>
```

Có thể, bạn cần phải thay đổi thông tin trong chuỗi kết nối cho phù hợp với thông tin mà bạn đã cài đặt MS SQL Server và tạo cơ sở dữ liệu.

4. Nạp danh mục các nhóm món ăn lên TreeView

Mở MainForm, nhấp đôi chuột vào form để tạo hàm xử lý sự kiện Load trên MainForm. Tiếp đến, nhấp đôi chuột vào nút R bên trái (btnReloadCategory) để tạo hàm xử lý sự kiện Click trên nút btnReloadCategory.

```
1 reference | 0 changes | 0 authors, 0 changes
private void MainForm_Load(object sender, EventArgs e)
{
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnReloadCategory_Click(object sender, EventArgs e)
{
}
```

Cũng trong lớp MainForm.cs, định nghĩa các phương thức sau:

```
1 reference | 0 changes | 0 authors, 0 changes
private List<Category> GetCategories()
{
    // Khởi tạo đối tượng context
    var dbContext = new RestaurantContext();

    // Lấy danh sách tất cả nhóm thức ăn, sắp xếp theo tên
    return dbContext.Categories.OrderBy(x => x.Name).ToList();
}
```

```

4 references | 0 changes | 0 authors, 0 changes
private void ShowCategories()
{
    // Xóa tất cả các nút hiện có trên cây
    tvwCategory.Nodes.Clear();

    // Tạo danh sách loại nhóm thức ăn, đồ uống
    // Tên của các loại này được hiển thị trên các nút mức 2
    var cateMap = new Dictionary<CategoryType, string>()
    {
        [CategoryType.Food] = "Đồ ăn",
        [CategoryType.Drink] = "Thức uống"
    };

    // Tạo nút gốc của cây
    var rootNode = tvwCategory.Nodes.Add("Tất cả");

    // Lấy danh sách nhóm đồ ăn, thức uống
    var categories = GetCategories();

    // Duyệt qua các loại nhóm thức ăn
    foreach (var cateType in cateMap)
    {
        // Tạo các nút tương ứng với loại nhóm thức ăn
        var childNode = rootNode.Nodes.Add(cateType.Key.ToString(), cateType.Value);
        childNode.Tag = cateType.Key;

        // Duyệt qua các nhóm thức ăn
        foreach (var category in categories)
        {
            // Nếu nhóm đang xét không cùng loại thì bỏ qua
            if (category.Type != cateType.Key) continue;

            // Ngược lại, tạo các nút tương ứng trên cây
            var grantChildNode = childNode.Nodes.Add(category.Id.ToString(), category.Name);
            grantChildNode.Tag = category;
        }
    }

    // Mở rộng các nhánh của cây để thấy hết tất cả các nhóm thức ăn
    tvwCategory.ExpandAll();

    // Đánh dấu nút gốc đang được chọn
    tvwCategory.SelectedNode = rootNode;
}

```

Sau đó, gọi hàm ShowCategories() trong trong hai hàm xử lý sự kiện đã tạo lúc này:

```

1 reference | 0 changes | 0 authors, 0 changes
private void MainForm_Load(object sender, EventArgs e)
{
    ShowCategories();
}

1 reference | 0 changes | 0 authors, 0 changes
private void btnReloadCategory_Click(object sender, EventArgs e)
{
    ShowCategories();
}

```

Nhấn F5 để chạy chương trình và xem kết quả.

Nhấn nút “R” ở phía bên trái để tải lại danh mục nhóm món ăn.

5. Hiển thị danh sách món ăn, đồ uống khi chọn một danh mục

Trong phần này, ta sẽ định nghĩa các phương thức và xử lý sự kiện để hiển thị danh sách các món ăn, đồ uống vào ListView phía bên phải khi người dùng chọn một nút trong TreeView bên trái.

- Nếu người dùng chọn nút Tất cả, ListView bên phải sẽ hiển thị tất cả các món ăn, thức uống.
- Nếu người dùng chọn nút Đồ ăn, tất cả các món ăn trong các danh mục đồ ăn sẽ được hiển thị trong ListView.
- Nếu người dùng chọn nút Thức uống, ListView sẽ cho thấy tất cả các đồ uống trong thực đơn.
- Nếu người dùng chọn một danh mục, các món ăn hoặc đồ uống trong danh mục (nhóm) đó sẽ được liệt kê trong ListView.

Như vậy, ta cần phải phân biệt các nút được chọn để có thể truy vấn danh sách các món ăn, đồ uống thích hợp. Ở đây, ta dùng mức (level) của các nút trên cây để phân biệt.

Trước tiên, ta định nghĩa các phương thức sau:

- GetFoodByCategoryId(int? categoryId): Lấy danh sách món ăn theo mã danh mục. Nếu mã này bằng null thì lấy tất cả các món ăn. Sắp xếp món ăn tăng dần theo tên.
- GetFoodByCategoryType(CategoryType cateType): Lấy danh sách món ăn hoặc đồ uống, tùy theo loại danh mục.

```

1 reference | 0 changes | 0 authors, 0 changes
private List<FoodModel> GetFoodByCategory(int? categoryId)
{
    // Khởi tạo đối tượng context
    var dbContext = new RestaurantContext();

    // Tạo truy vấn lấy danh sách món ăn
    var foods = dbContext.Foods.AsQueryable();

    // Nếu mã nhóm món ăn khác null và hợp lệ
    if (categoryId != null && categoryId > 0)
    {
        // Thì tìm theo mã số nhóm thức ăn
        foods = foods.Where(x => x.FoodCategoryId == categoryId);
    }

    // Sắp xếp đồ ăn, thức uống theo tên và trả về
    // danh sách chứa đầy đủ thông tin về món ăn.
    return foods
        .OrderBy(x => x.Name)
        .Select(x => new FoodModel()
    {
        Id = x.Id,
        Name = x.Name,
        Unit = x.Unit,
        Price = x.Price,
        Notes = x.Notes,
        CategoryName = x.Category.Name
    })
    .ToList();
}

1 reference | 0 changes | 0 authors, 0 changes
private List<FoodModel> GetFoodByCategoryType(CategoryType cateType)
{
    var dbContext = new RestaurantContext();

    // Tìm các món ăn theo loại nhóm thức ăn (Category Type).
    // Sắp xếp đồ ăn, thức uống theo tên và trả về
    // danh sách chứa đầy đủ thông tin về món ăn.
    return dbContext.Foods
        .Where(x => x.Category.Type == cateType)
        .OrderBy(x => x.Name)
        .Select(x => new FoodModel()
    {
        Id = x.Id,
        Name = x.Name,
        Unit = x.Unit,
        Price = x.Price,
        Notes = x.Notes,
        CategoryName = x.Category.Name
    })
    .ToList();
}

```

Tiếp đến, ta định nghĩa hàm ShowFoodsForNode nhận đầu vào là một nút của cây. Kiểm tra xem nút đó ở mức nào để gọi các hàm tương ứng đã định nghĩa ở trên. Lưu ý

rằng, trong phương thức ShowCategories(), ta đã lưu các thông tin cần thiết vào nút của cây.

```

4 references | 0 changes | 0 authors, 0 changes
private void ShowFoodsForNode(TreeNode node)
{
    // Xóa danh sách thực đơn hiện tại khỏi listView
    lvwFood.Items.Clear();

    // Nếu node = null, không cần xử lý gì thêm
    if (node == null) return;

    // Tạo danh sách để chứa danh sách các món ăn tìm được
    List<FoodModel> foods = null;

    // Nếu nút được chọn trên TreeView tương ứng với
    // loại nhóm thức ăn (Category Type) (mức thứ 2 trên cây)
    if (node.Level == 1)
    {
        // Thì lấy danh sách món ăn theo loại nhóm
        var categoryType = (CategoryType)node.Tag;
        foods = GetFoodByCategoryType(categoryType);
    }
    else
    {
        // Ngược lại, lấy danh sách món ăn theo thể loại
        // Nếu nút được chọn là 'Tất cả' thì lấy hết
        var category = node.Tag as Category;
        foods = GetFoodByCategory(category?.Id);
    }

    // Gọi hàm để hiển thị các món ăn lên ListView
    ShowFoodsOnListView(foods);
}

1 reference | 0 changes | 0 authors, 0 changes
private void ShowFoodsOnListView(List<FoodModel> foods)
{
    // Duyệt qua từng phần tử của danh sách food
    foreach (var foodItem in foods)
    {
        // Tạo item tương ứng trên ListView
        var item = lvwFood.Items.Add(foodItem.Id.ToString());

        // Và hiển thị các thông tin của món ăn
        item.SubItems.Add(foodItem.Name);
        item.SubItems.Add(foodItem.Unit);
        item.SubItems.Add(foodItem.Price.ToString("##,###"));
        item.SubItems.Add(foodItem.CategoryName);
        item.SubItems.Add(foodItem.Notes);
    }
}

```

Để xử lý sự kiện chọn một nút của cây, ta làm như sau:

- Nhấp phải chuột vào TreeView, chọn Properties.
- Trong cửa sổ Properties, click chọn nút Events (hình tia sét)
- Sau đó nhấp đúp chuột vào sự kiện AfterSelect để tạo hàm xử lý sự kiện đó

Sau đó, gọi hàm ShowFoodsForNode() trong hàm xử lý sự kiện và truyền nút được chọn vào đối số của hàm. Nút được chọn được lưu trong thuộc tính Node của tham số e.

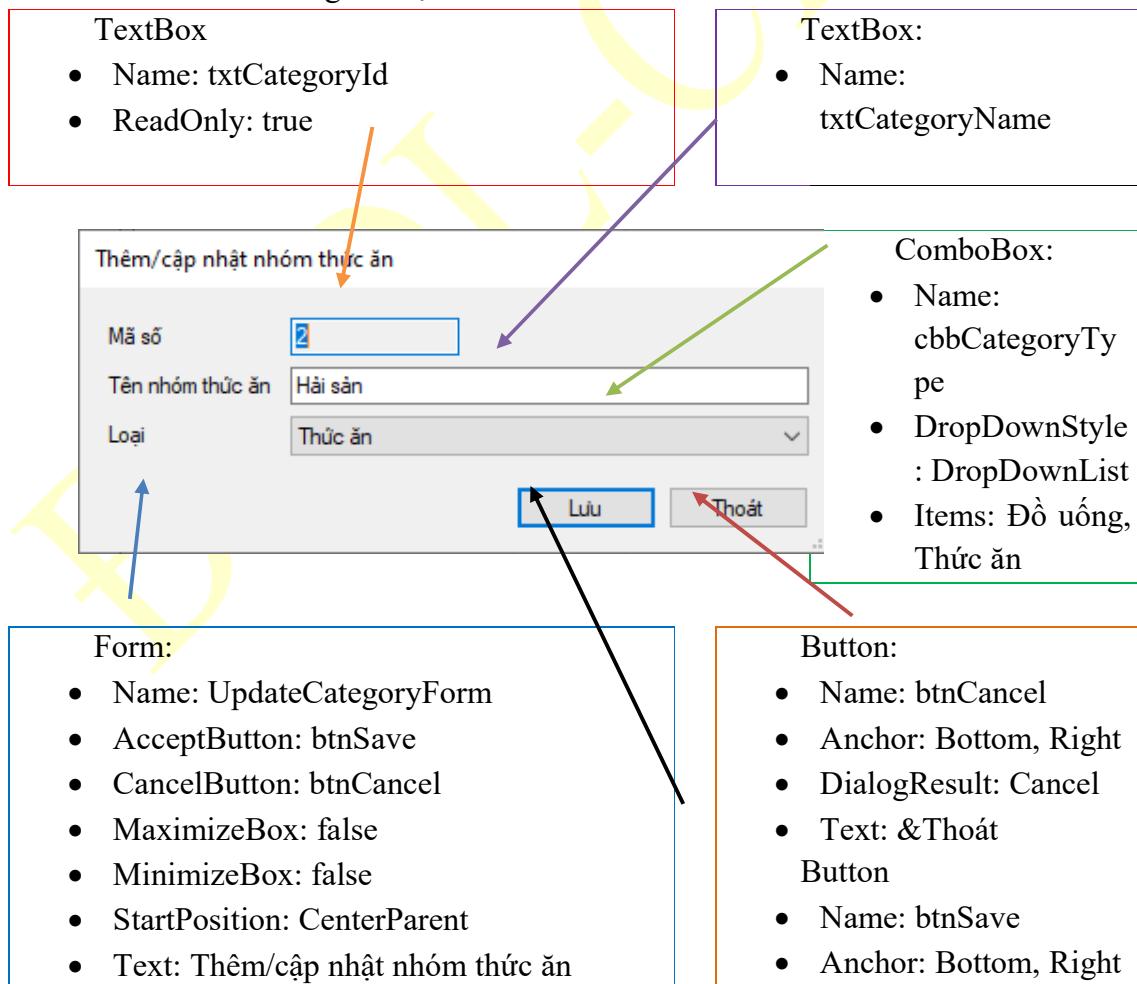
```
1 reference | 0 changes | 0 authors, 0 changes
private void tvwCategory_AfterSelect(object sender, TreeViewEventArgs e)
{
    ShowFoodsForNode(e.Node);
}
```

Nhấn phím F5 để chạy chương trình và kiểm tra kết quả.

6. Xây dựng Form cập nhật thông tin danh mục (nhóm) món ăn, đồ uống

Phần này sẽ hướng dẫn bạn cách xây dựng một form mới – có tên UpdateCategoryForm – dùng để thêm mới hoặc cập nhật thông tin một nhóm thức ăn, đồ uống.

- Nhấp phải chuột vào tên dự án, chọn Add > Windows Form.
- Đặt tên cho form mới là UpdateCategoryForm.
- Thiết kế form có giao diện như sau:



- Text: &Lưu

Tiếp theo, trong lớp UpdateCategoryForm, khai báo thêm 2 biến:

```
private RestaurantContext _dbContext;
private int _categoryId;
```

Và cập nhật lại hàm khởi tạo của form như sau:

```
2 references | 0 changes | 0 authors, 0 changes
public UpdateCategoryForm(int? categoryId = null)
{
    InitializeComponent();
    _dbContext = new RestaurantContext();
    _categoryId = categoryId ?? 0;
}
```

Tham số categoryId có kiểu Nullable<int> và nhận giá trị mặc định là null. Điều này cho phép ta khởi tạo form mới mà không cần phải truyền vào mã danh mục món ăn. Điều này được áp dụng khi muốn tạo mới một danh mục.

Trong trường hợp tham số này nhận một giá trị dương, form sẽ ở chế độ cập nhật thông tin của một danh mục có sẵn. Lúc này, khi form được mở, ta cần hiển thị thông tin của danh mục đã được chọn.

Để truy vấn và hiển thị thông tin của một danh mục có mã cho trước, ta định nghĩa các phương thức sau đây:

```
2 references | 0 changes | 0 authors, 0 changes
private Category GetCategoryById(int categoryId)
{
    // Nếu ID được truyền vào là hợp lệ, ta tìm thông tin theo ID
    // Ngược lại, chỉ đơn giản trả về null, cho biết không thấy.
    return categoryId > 0 ? _dbContext.Categories.Find(categoryId) : null;
}

1 reference | 0 changes | 0 authors, 0 changes
private void ShowCategory()
{
    // Lấy thông tin của nhóm thức ăn
    var category = GetCategoryById(_categoryId);

    // Nếu không tìm thấy thông tin, không cần làm gì cả
    if (category == null) return;

    // Ngược lại, nếu tìm thấy, hiển thị lên form
    txtCategoryId.Text = category.Id.ToString();
    txtCategoryName.Text = category.Name;
    cbbCategoryType.SelectedIndex = (int)category.Type;
}
```

Muốn hiển thị thông tin của danh mục lên màn hình khi form được mở, ta xử lý sự kiện Load của UpdateCategoryForm như sau:

```
1 reference | 0 changes | 0 authors, 0 changes
private void UpdateCategoryForm_Load(object sender, EventArgs e)
{
    // Hiển thị thông tin nhóm thức ăn lên form
    ShowCategory();
}
```

Khi người dùng nhấn nút Lưu, ta cần phải kiểm tra dữ liệu nhập có hợp lệ hay không. Nếu dữ liệu không hợp lệ, chương trình cần hiển thị hộp thoại thông báo cho người dùng biết. Ngược lại, ta lấy dữ liệu nhập và thực thi truy vấn để lưu chúng vào cơ sở dữ liệu. Trong form này, ta cần phải xử lý cả hai tình huống:

- Biến `_categoryId > 0`: nghĩa là khi form bật lên, bạn đã chọn một danh mục để cập nhật. Trong trường hợp này, chương trình cần cập nhật thông tin mới cho danh mục đó.
- Biến `_categoryId = 0`: nghĩa là form được mở ra để tạo mới một danh mục.

Ta định nghĩa thêm hai phương thức để kiểm tra dữ liệu nhập và lấy thông tin đã nhập như sau:

```

1 reference | 0 changes | 0 authors, 0 changes
private Category GetUpdatedCategory()
{
    // Tạo đối tượng Category với thông tin đã nhập
    var category = new Category()
    {
        Name = txtCategoryName.Text.Trim(),
        Type = (CategoryType)cbbCategoryType.SelectedIndex
    };

    // Gán giá trị của ID ban đầu (nếu đang cập nhật)
    if (_categoryId > 0)
    {
        category.Id = _categoryId;
    }

    return category;
}

1 reference | 0 changes | 0 authors, 0 changes
private bool ValidateUserInput()
{
    // Kiểm tra tên nhóm thức ăn đã được nhập hay chưa
    if (string.IsNullOrWhiteSpace(txtCategoryName.Text))
    {
        MessageBox.Show("Tên nhóm thức ăn không được để trống", "Thông báo");
        return false;
    }

    // Kiểm tra loại nhóm thức ăn đã được chọn hay chưa
    if (cbbCategoryType.SelectedIndex < 0)
    {
        MessageBox.Show("Bạn chưa chọn loại nhóm thức ăn", "Thông báo");
        return false;
    }

    return true;
}

```

Tiếp đến, nhấp đúp chuột vào nút Lưu (btnSave) để tạo hàm xử lý sự kiện Click và định nghĩa hàm đó như sau:

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnSave_Click(object sender, EventArgs e)
{
    // Kiểm tra nếu dữ liệu nhập vào là hợp lệ
    if (ValidateUserInput())
    {
        // Thì lấy thông tin người dùng nhập vào
        var newCategory = GetUpdatedCategory();

        // Và thử tìm xem đã có nhóm thức ăn trong CSDL chưa
        var oldCategory = GetCategoryById(_categoryId);

        // Nếu chưa có (chưa tồn tại)
        if (oldCategory == null)
        {
            // Thì thêm nhóm thức ăn mới
            _dbContext.Categories.Add(newCategory);
        }
        else
        {
            // Ngược lại, ta chỉ cần cập nhật thông tin cần thiết
            oldCategory.Name = newCategory.Name;
            oldCategory.Type = newCategory.Type;
        }

        // Lưu các thay đổi xuống CSDL
        _dbContext.SaveChanges();

        // Đóng hộp thoại
        DialogResult = DialogResult.OK;
    }
}

```

7. Thêm mới, cập nhật một danh mục món ăn

Phần này sẽ hướng dẫn bạn cách sử dụng UpdateCategoryForm để thêm mới và cập nhật một danh mục món ăn.

- Để thêm mới một danh mục, ta xử lý sự kiện Click nút dấu + (btnAddCategory) trên MainForm và gọi hàm để hiển thị hộp thoại UpdateCategoryForm.
- Để cập nhật một danh mục, người dùng nhấp đôi chuột vào một nút trên cây. Chương trình cần xử lý sự kiện NodeMouseDoubleClick để mở form cập nhật.
- Trong cả 2 trường hợp, sau khi Lưu dữ liệu thành công, chương trình cần tải và hiện thị lại danh mục thức ăn trên TreeView.

Nhấp đôi chuột vào nút btnAddCategory và định nghĩa hàm xử lý sự kiện Click như sau:

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnAddCategory_Click(object sender, EventArgs e)
{
    var dialog = new UpdateCategoryForm();
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowCategories();
    }
}

```

Nhấp phải chuột vào TreeView, chọn Properties > Events. Nhấp đôi chuột vào sự kiện NodeMouseDoubleClick để tạo hàm xử lý sự kiện. Định nghĩa hàm đó như sau:

```

1 reference | 0 changes | 0 authors, 0 changes
private void tvwCategory_NodeMouseDoubleClick(object sender, TreeNodeMouseClickEventArgs e)
{
    if (e.Node == null || e.Node.Level < 2 || e.Node.Tag == null) return;

    var category = e.Node.Tag as Category;
    var dialog = new UpdateCategoryForm(category?.Id);
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowCategories();
    }
}

```

Nhấn phím F5 để chạy chương trình. Click vào nút dấu + hoặc nhấp đôi chuột vào 1 nút trên cây để xem cách hoạt động của chương trình.

8. Xử lý việc xóa một món ăn và nạp lại danh sách món ăn, đồ uống

Việc nạp lại danh sách đồ ăn, thức uống tương đối đơn giản vì ta đã có hàm ShowFoodsForNode được định nghĩa ở phần trước. Trong màn hình thiết kế MainForm, nhấp đôi chuột vào nút R (btnReloadFood) để tạo hàm xử lý sự kiện Click. Gọi hàm ShowFoodsForNode với tham số là nút hiện đang được chọn trong TreeView.

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnReloadFood_Click(object sender, EventArgs e)
{
    ShowFoodsForNode(tvwCategory.SelectedNode);
}

```

Để xóa một món ăn, đồ uống đang được chọn trong ListView, ta cần xử lý sự kiện Click cho nút “--” (btnDeleteFood) trong MainForm.

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnDeleteFood_Click(object sender, EventArgs e)
{
    // Nếu không có món ăn nào được chọn, không cần làm gì cả
    if (lrvFood.SelectedItems.Count == 0) return;

    // Ngược lại, lấy mã số của món ăn được chọn
    var dbContext = new RestaurantContext();
    var selectedFoodId = int.Parse(lrvFood.SelectedItems[0].Text);
}

```

```

    // Truy vấn để lấy thông tin của món ăn đó
    var selectedFood = dbContext.Foods.Find(selectedFoodId);

    // Nếu tìm thấy thông tin món ăn
    if (selectedFood != null)
    {
        // Thì xóa nó khỏi cơ sở dữ liệu
        dbContext.Foods.Remove(selectedFood);
        dbContext.SaveChanges();

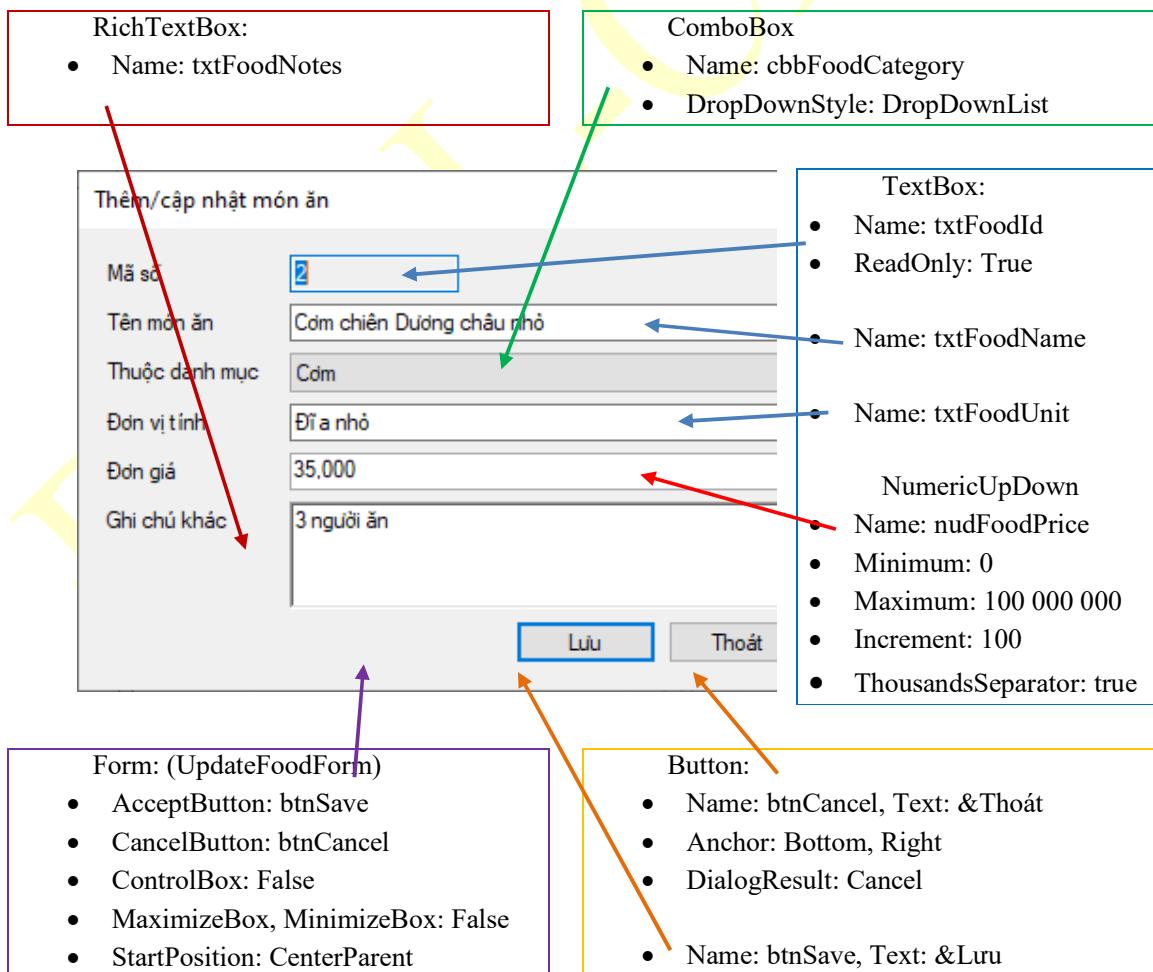
        // Và đồng thời xóa khỏi ListView
        lvwFood.Items.Remove(lvwFood.SelectedItems[0]);
    }
}

```

Chạy chương trình và thử nhấn vào nút R và nút “--” ở bên phải để xem hoạt động của chương trình.

9. Xây dựng form cập nhật thông tin món ăn, đồ uống

Để thêm và cập nhật thông tin món ăn, đồ uống, ta xây dựng form mới như hình dưới đây



- Text: Thêm/cập nhật món ăn
- Anchor: Bottom, Right
- Nhấp phải chuột vào tên dự án, chọn Add > Windows Form.
- Đặt tên cho form mới là UpdateFoodForm.
- Thiết kế form có giao diện như hình trên
- Trong lớp UpdateFoodForm, khai báo thêm 2 biến

```
private RestaurantContext _dbContext;
private int _foodId;
```

- Khởi tạo hai biến đó trong hàm khởi tạo của form như sau

```
2 references | 0 changes | 0 authors, 0 changes
public UpdateFoodForm(int? foodId = null)
{
    InitializeComponent();
    _dbContext = new RestaurantContext();
    _foodId = foodId ?? 0;
}
```

Tiếp theo, ta cần lấy danh sách các nhóm đồ ăn, thức uống và nạp chúng vào ComboBox khi form được mở lên. Ngoài ra, nếu người dùng mở form để cập nhật thông tin một món ăn đã chọn thì chương trình cũng cần phải truy vấn và hiển thị thông tin món ăn đó.

Để nạp danh mục đồ ăn, thức uống vào ComboBox, ta định nghĩa hàm sau:

```
1 reference | 0 changes | 0 authors, 0 changes
private void LoadCategoriesToCombobox()
{
    // Lấy tất cả danh mục thức ăn, sắp tăng theo tên
    var categories = _dbContext.Categories
        .OrderBy(x => x.Name).ToList();

    // Nạp danh mục vào combobox, hiển thị tên cho người
    // dùng xem nhưng khi được chọn thì lấy giá trị là ID
    cbbFoodCategory.DisplayMember = "Name";
    cbbFoodCategory.ValueMember = "Id";
    cbbFoodCategory.DataSource = categories;
}
```

Sau đó gọi tới phương thức vừa tạo trong hàm xử lý sự kiện Load của UpdateFoodForm. Bạn cần nhấp đôi chuột vào form để tạo hàm xử lý sự kiện này.

```
1 reference | 0 changes | 0 authors, 0 changes
private void UpdateFoodForm_Load(object sender, EventArgs e)
{
    // Nạp danh sách nhóm thức ăn vào combobox
    LoadCategoriesToCombobox();
}
```

Việc truy vấn để lấy thông tin món ăn dựa vào mã số cho trước sẽ được thực hiện nhiều lần. Vì vậy, ta có thể định nghĩa phương thức dưới đây để có thể sử dụng lại nó.

```
2 references | 0 changes | 0 authors, 0 changes
private Food GetFoodById(int foodId)
{
    // Tìm món ăn theo mã số
    return foodId > 0 ? _dbContext.Foods.Find(foodId) : null;
}
```

Hàm này kiểm tra nếu mã số là hợp lệ thì lấy thông tin món ăn từ cơ sở dữ liệu. Ngược lại, chỉ đơn giản là trả về null.

Để hiển thị thông tin món ăn người dùng đã chọn lên form để cập nhật, ta định nghĩa thêm một phương thức mới:

```
1 reference | 0 changes | 0 authors, 0 changes
private void ShowFoodInformation()
{
    // Tìm món ăn theo mã số đã được truyền vào form
    var food = GetFoodById(_foodId);

    // Nếu không tìm thấy, không cần làm gì cả
    if (food == null) return;

    // Ngược lại, hiển thị thông tin món ăn lên form
    txtFoodId.Text = food.Id.ToString();
    txtFoodName.Text = food.Name;
    cbbFoodCategory.SelectedValue = food.FoodCategoryId;
    txtFoodUnit.Text = food.Unit;
    nudFoodPrice.Value = food.Price;
    txtFoodNotes.Text = food.Notes;
}
```

Sau đó, bổ sung lời gọi hàm này trong hàm xử lý sự kiện form load.

```
1 reference | 0 changes | 0 authors, 0 changes
private void UpdateFoodForm_Load(object sender, EventArgs e)
{
    // Nạp danh sách nhóm thức ăn vào combobox
    LoadCategoriesToCombobox();

    // Hiển thị thông tin món ăn lên form
    ShowFoodInformation();
}
```

Việc tiếp theo cần xử lý là kiểm tra dữ liệu nhập khi người dùng nhấn nút Lưu và hiển thị thông báo nếu dữ liệu nhập chưa đúng. Nếu mọi thông tin đều hợp lệ, ta cần tái tạo lại đối tượng Food từ dữ liệu nhập và thêm hoặc cập nhật vào cơ sở dữ liệu.

```

references | 0 changes | 0 authors, 0 changes
private bool ValidateUserInput()
{
    // Kiểm tra tên món ăn đã được nhập hay chưa
    if (string.IsNullOrWhiteSpace(txtFoodName.Text))
    {
        MessageBox.Show("Tên món ăn, đồ uống không được để trống", "Thông báo");
        return false;
    }

    // Kiểm tra đơn vị tính đã được nhập hay chưa
    if (string.IsNullOrWhiteSpace(txtFoodUnit.Text))
    {
        MessageBox.Show("Đơn vị tính không được để trống", "Thông báo");
        return false;
    }

    // Kiểm tra giá món ăn đã được nhập hay chưa
    if (nudFoodPrice.Value.Equals(0))
    {
        MessageBox.Show("Giá của thức ăn phải lớn hơn 0", "Thông báo");
        return false;
    }

    // Kiểm tra nhóm món ăn đã được chọn hay chưa
    if (cbbFoodCategory.SelectedIndex < 0)
    {
        MessageBox.Show("Bạn chưa chọn nhóm thức ăn", "Thông báo");
        return false;
    }

    return true;
}

reference | 0 changes | 0 authors, 0 changes
private Food GetUpdatedFood()
{
    // Tạo đối tượng food với thông tin được lấy từ
    // các điều khiển trên form
    var food = new Food()
    {
        Name = txtFoodName.Text.Trim(),
        FoodCategoryId = (int)cbbFoodCategory.SelectedValue,
        Unit = txtFoodUnit.Text,
        Price = (int)nudFoodPrice.Value,
        Notes = txtFoodNotes.Text
    };

    // Gán giá trị của ID ban đầu (nếu đang cập nhật)
    if (_foodId > 0)
    {
        food.Id = _foodId;
    }

    return food;
}

```

Cuối cùng, để hoàn thành form cập nhật thông tin món ăn, bạn cần viết mã cho hàm xử lý khi người dùng nhấn nút Lưu. Nhấp đôi chuột vào nút Lưu (btnSave) để tạo hàm xử lý sự kiện Click và bổ sung đoạn mã sau:

```
1 reference | 0 changes | 0 authors, 0 changes
private void btnSave_Click(object sender, EventArgs e)
{
    // Kiểm tra nếu dữ liệu nhập vào là hợp lệ
    if (ValidateUserInput())
    {
        // Thì lấy thông tin người dùng nhập vào
        var newFood = GetUpdatedFood();

        // Và thử tìm xem đã có món ăn trong CSDL chưa
        var oldFood = GetFoodById(_foodId);

        // Nếu chưa có (chưa tồn tại)
        if (oldFood == null)
        {
            // Thì thêm món ăn mới
            _dbContext.Foods.Add(newFood);
        }
        else
        {
            // Ngược lại, cập nhật thông tin món ăn
            oldFood.Name = newFood.Name;
            oldFood.FoodCategoryId = newFood.FoodCategoryId;
            oldFood.Unit = newFood.Unit;
            oldFood.Price = newFood.Price;
            oldFood.Notes = newFood.Notes;
        }

        // Lưu các thay đổi xuống CSDL
        _dbContext.SaveChanges();

        // Đóng hộp thoại
        DialogResult = DialogResult.OK;
    }
}
```

10. Gọi form cập nhật thông tin món ăn, đồ uống

Trong phần này, ta bổ sung hai phương thức để xử lý sự kiện nhấn nút + (btnAddFood) để thêm món ăn mới và sự kiện nhấn đôi chuột vào một item của ListView để cập nhật thông tin món ăn, đồ uống.

```

1 reference | 0 changes | 0 authors, 0 changes
private void btnAddFood_Click(object sender, EventArgs e)
{
    var dialog = new UpdateFoodForm();
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowFoodsForNode(tvwCategory.SelectedNode);
    }
}

1 reference | 0 changes | 0 authors, 0 changes
private void lvwFood_DoubleClick(object sender, EventArgs e)
{
    if (lvwFood.SelectedItems.Count == 0) return;

    var foodId = int.Parse(lvwFood.SelectedItems[0].Text);
    var dialog = new UpdateFoodForm(foodId);
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        ShowFoodsForNode(tvwCategory.SelectedNode);
    }
}

```

Chạy chương trình và kiểm tra kết quả.

III. Bài tập

Thêm các chức năng cho chương trình như sau

1. Thiết kế Form: RoleForm và viết hàm xử lý để
 - a. Hiển thị danh sách các nhóm (hay vai trò) người dùng
 - b. Thêm role mới, cập nhật một role đã có.
 - c. Xem danh sách người dùng thuộc role được chọn
2. Thiết kế Form: AccountForm và viết hàm xử lý để
 - a. Xem danh sách tài khoản, có cho phép tìm theo role và tên người dùng
 - b. Thêm một tài khoản mới, trong đó cho phép chọn role cho người dùng
 - c. Cập nhật thông tin của một tài khoản
 - d. Reset mật khẩu cho tài khoản
 - e. Thay đổi mật khẩu của tài khoản
 - f. Click chuột phải vào một tài khoản hiển thị menu sau:

Xóa tài khoản
Xem danh sách vai trò

Trong đó:

 - Nếu chọn Xóa tài khoản thì xóa Password, gán về null. Lúc này, người dùng không thể đăng nhập được, tài khoản ở trạng thái Inactive.
 - Xem danh sách vai trò: Mở một Form mới để hiển thị các vai trò (role) được gán cho tài khoản này
3. Thiết kế Form: BillsForm và viết hàm xử lý để
 - a. Hiển thị danh sách hóa đơn được bán trong một khoảng thời gian nào đó (yêu cầu có ô chọn từ ngày đến ngày – sử dụng control DateTimePicker), có hiển thị tổng số tiền chưa giảm giá, tổng số tiền giảm giá, thực thu

- b. Khi nhấp đúp chuột vào một hóa đơn nào đó thì mở một Form mới (BillDetailsForm) để hiển thị danh mục các mặt hàng mua bởi hóa đơn đó.
4. Thiết kế Form: FoodForm với chức năng cụ thể như sau:
- Hiển thị món ăn theo nhóm món ăn
 - Tìm kiếm món ăn theo tên
 - Thêm, sửa món ăn
 - Xóa các món ăn đã chọn từ danh sách
 - Thêm nhóm món ăn
5. Thiết kế Form: TableForm và viết các hàm xử lý để
- Hiển thị danh sách các bàn
 - Xem hóa đơn hiện tại của một bàn
 - Thêm một bàn mới
 - Cập nhật thông tin của bàn
 - Xóa một bàn.
 - Khi nhấp phải chuột vào một bàn, hiển thị menu sau

Xóa bàn
Xem danh mục hóa đơn
Xem nhật ký bán hàng

Trong đó:

- Nếu chọn Xóa bàn thì dữ liệu về bàn đó sẽ bị xóa khỏi cơ sở dữ liệu
- Xem danh mục hóa đơn: Mở một Form mới, phần bên trái của Form là một ListBox chứa ngày lập của các hóa đơn. Khi nhấp chuột vào ngày nào thì hiển thị thông tin chi tiết (cả danh mục sản phẩm được mua) của hóa đơn ở phần bên phải Form.
- Xem nhật ký bán hàng: Liệt kê số lượng hóa đơn, tổng số tiền, tổng thuế, tổng giảm giá của tất cả các hóa đơn, thông tin liên quan đến từng hóa đơn như ngày lập, tên nhân viên lập hóa đơn. (sử dụng ListView hoặc DataGridView)

6. Thiết kế Form: MainForm và viết các hàm xử lý để xem trạng thái hoạt động của các bàn, quản lý việc đặt món cho các bàn, thanh toán tiền lập hóa đơn, in hóa đơn, chuyển bàn, nhập bàn. Từ main Form, tạo menu để mở các form quản trị. Nếu người dùng đăng nhập với vai trò quản trị viên sẽ truy cập được các form này. Nếu người dùng là nhân viên, chỉ cho xem thông tin tài khoản của họ.
7. Thiết kế Form: MenuForm để xem và in thực đơn.