

BÀI THỰC HÀNH SỐ 4 (4 tiết)

SỬ DỤNG CÁC ĐỐI TƯỢNG THUỘC NHÓM KHÔNG KẾT NỐI

I. Mục tiêu

Bài thực hành này giúp sinh viên tìm hiểu:

- Cách sử dụng các lớp DataSet, DataTable, DataRow, DataColumn, DataView
- Cách sử dụng DataAdapter để cập nhật dữ liệu từ DataSet.
- Cách tạo ràng buộc dữ liệu trên các control của Windows Form

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

- Cách quản lý dữ liệu trả về bởi phương thức Fill của DataAdapter.
- Cấu hình các thuộc tính Command của DataAdapter để cập nhật các thay đổi từ DataSet vào cơ sở dữ liệu.
- Thiết lập các thuộc tính của các control bằng giao diện và bằng cách viết mã lệnh.

II. Nội dung lý thuyết

Các đối tượng DataSet cho phép lưu trữ một bản sao thông tin của cơ sở dữ liệu. Từ đó, bạn có thể xử lý trực tiếp trên các thông tin này trong khi kết nối đã bị ngắt.

Đối tượng DataAdapter để đọc các dòng từ cơ sở dữ liệu vào một DataSet và là cầu nối giữa hai nhóm lớp: kết nối và không kết nối.

1. Tạo đối tượng DataAdapter và các đối tượng trong nhóm lớp không kết nối

Để tạo một đối tượng *SqlDataAdapter*, ta sử dụng một trong các phương thức khởi tạo của lớp *SqlDataAdapter* như sau:

```
SqlDataAdapter ()  
SqlDataAdapter (SqlCommand mySqlCommand)  
SqlDataAdapter (string selectCommandString, SqlConnection mySqlConnection)  
SqlDataAdapter (string selectCommandString, string connectionString)
```

Ví dụ:

```
SqlConnection mySqlConnection =  
    new SqlConnection(  
        "server=localhost;database=Northwind;uid=sa;pwd=sa"  
    );  
string selectCommandString =  
    "SELECT * FROM Products ORDER BY ProductID";  
  
SqlDataAdapter mySqlDataAdapter =  
    new SqlDataAdapter(selectCommandString, mySqlConnection);
```

Để tạo một đối tượng *DataSet*, ta dùng một trong hai phương thức khởi tạo sau

```
DataSet()  
DataSet(string dataSetNameString)
```

Ví dụ:

```
DataSet myDataSet = new DataSet();  
// Tạo một đối tượng DataSet mới có tên là myDataSet
```

```
DataSet myDataSet = new DataSet("myDataSet");
```

2. Đưa dữ liệu vào DataSet, DataTable bằng phương thức Fill của DataAdapter

Các dạng của phương thức Fill()

```
int Fill(DataSet myDataSet)
int Fill(DataTable myDataTable)
int Fill(DataSet myDataSet, string dataTableName)
int Fill(DataSet myDataSet, int startRow, int numOfWeeks,
        string dataTableName)
```

Cách 1: Dùng lệnh SELECT

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

// Thiết lập đối tượng Command (chứa lệnh SELECT) cho DataAdapter
mySqlDataAdapter.SelectCommand = mySqlCommand;

// Tạo đối tượng DataSet
DataSet myDataSet = new DataSet();

// Mở kết nối tới CSDL
mySqlConnection.Open();

// Đưa dữ liệu trả về vào DataSet
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "Tên_Bảng");

// Lấy bảng dữ liệu kết quả trong DataSet
DataTable myDataTable = myDataSet.Tables["Tên_Bảng"];
DataTable myDataTable = myDataSet.Tables[0];
```

Cách 2: Lấy một phần của tập kết quả

```
SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;

DataSet myDataSet = new DataSet();

// lấy các dòng từ 1 đến 3 của liệu trả về và lưu vào một bảng
// trong DataSet, đặt tên bảng là Products
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, 1, 3, "Products");
```

Cách 3: Sử dụng Stored Procedure

```
SqlCommand mySqlCommand = mySqlConnection.CreateCommand();
mySqlCommand.CommandText =
    "EXECUTE CustOrderHist @CustomerID";

mySqlCommand.Parameters.Add("@CustomerID",
    SqlDbType.NVarChar, 5).Value = "ALFKI";

SqlDataAdapter mySqlDataAdapter = new SqlDataAdapter();
mySqlDataAdapter.SelectCommand = mySqlCommand;

DataSet myDataSet = new DataSet();
mySqlConnection.Open();
int numberOfRows = mySqlDataAdapter.Fill(myDataSet, "CustOrderHist");
```

```
mySqlConnection.Close();
```

Trong trường hợp truy vấn gồm nhiều lệnh Select, ta phải dùng lệnh Fill(DataSet ds). Khi đó, các bảng kết quả sẽ được đặt tên lần lượt là Table, Table1, Table2...

3. Tìm kiếm, trích lọc và sắp xếp các dòng trong DataTable

Tìm kiếm

- Lấy các dòng từ cơ sở dữ liệu lưu vào một DataTable.
- Thiết lập giá trị thuộc tính PrimaryKey của DataTable
- Gọi phương thức Find() của DataTable và gửi giá trị trên cột khóa chính của DataRow muốn tìm vào tham số của phương thức Find().

```
mySqlDataAdapter.Fill(myDataSet, "Products");
mySqlConnection.Close();

DataTable productsDataTable = myDataSet.Tables["Products"];

productsDataTable.PrimaryKey =
    new DataColumn[]
    {
        productsDataTable.Columns["ProductID"]
    };
DataRow productDataRow = productsDataTable.Rows.Find("3");
```

Nếu khóa chính của một bảng trong cơ sở dữ liệu chứa hai hay nhiều cột, tham số được gửi vào phương thức Find() phải là một mảng các đối tượng.

```
object[] orderDetails =
    new object[]
    {
        10248,
        11
    };
DataRow orderDetailDataRow =
    orderDetailsDataTable.Rows.Find(orderDetails);
```

Trích lọc

Để lọc và sắp xếp các DataRow từ một DataTable, ta dùng phương thức Select() của đối tượng DataTable. Phương thức này có các dạng sau:

```
DataRow[] Select()
DataRow[] Select(string filterExpression)
```

Ví dụ:

```
// Lấy tất cả các dòng trong DataTable và không sắp xếp
DataRow[] productDataRows = productsDataTable.Select();

// Lấy các DataRow có giá trị trên cột ProductID <= 5
DataRow[] productDataRows =
    productsDataTable.Select("ProductID <= 5");
```

Sắp xếp

```
DataRow[] Select(string filterExpression, string sortExpression)
```

```
DataRow[] Select(string filterExpression, string sortExpression,
    DataViewRowState myDataViewRowState)
```

Ví dụ

```
// Lấy các DataRow có giá trị trên cột ProductID <= 5
// và sắp xếp các DataRow giảm dần theo ProductID
DataRow[] productDataRows =
    productsDataTable.Select("ProductID <= 5", "ProductID DESC");

// Lấy các DataRow có giá trị trên cột ProductID <= 5
// trước khi bị xóa hoặc thay đổi
// và sắp xếp các DataRow giảm dần theo ProductID
DataRow[] productDataRows =
    productsDataTable.Select("ProductID <= 5",
        "ProductID DESC", DataViewRowState.OriginalRows);
```

4. Cấu hình DataAdapter để cập nhật thay đổi vào cơ sở dữ liệu

Trước khi cập nhật các thay đổi vào cơ sở dữ liệu, đầu tiên, ta phải thiết lập các đối tượng Command chứa các lệnh SQL INSERT, UPDATE và DELETE thích hợp. Các đối tượng Command này được lưu trong các thuộc tính InsertCommand, UpdateCommand và DeleteCommand của đối tượng DataAdapter.

```
// Cấu hình lệnh INSERT
SqlCommand myInsertCommand = mySqlConnection.CreateCommand();
myInsertCommand.CommandText = "INSERT ....";

// Truyền tham số vào đối tượng Command
myInsertCommand.Parameters.Add(.....);
...

mySqlDataAdapter.InsertCommand = myInsertCommand;

// Cấu hình lệnh UPDATE
SqlCommand myUpdateCommand = mySqlConnection.CreateCommand();
myUpdateCommand.CommandText =
    "UPDATE Customers " +
    "SET CompanyName = @NewCompanyName, " +
    "    Address = @NewAddress " +
    "WHERE CustomerID = @OldCustomerID " +
    "AND CompanyName = @OldCompanyName " +
    "AND Address = @OldAddress";

// Truyền tham số vào đối tượng Command
myUpdateCommand.Parameters.Add("@NewCompanyName",
    SqlDbType.NVarChar, 40, "CompanyName");

myUpdateCommand.Parameters.Add("@NewAddress",
    SqlDbType.NVarChar, 60, "Address");

myUpdateCommand.Parameters.Add("@OldCustomerID",
    SqlDbType.NChar, 5, "CustomerID");

myUpdateCommand.Parameters.Add("@OldCompanyName",
    SqlDbType.NVarChar, 40, "CompanyName");

myUpdateCommand.Parameters.Add("@OldAddress",
    SqlDbType.NVarChar, 60, "Address");

// Truyền giá trị cho tham số
```

```

myUpdateCommand.Parameters["@OldCustomerID"].SourceVersion =
    DataRowVersion.Original;

myUpdateCommand.Parameters["@OldCompanyName"].SourceVersion =
    DataRowVersion.Original;

myUpdateCommand.Parameters["@OldAddress"].SourceVersion =
    DataRowVersion.Original;
mySqlDataAdapter.UpdateCommand = myUpdateCommand;

```

5. Cập nhật dữ liệu từ DataSet, DataTable bằng phương thức Update

Phương thức Update() của đối tượng DataAdapter có các dạng sau

```

int Update(DataRow[] myDataRows)
int Update(DataSet myDataSet)
int Update(DataTable myDataTable)
int Update(DataRow[] myDataRows, DataTableMapping myDataTableMapping)
int Update(DataSet myDataSet, string dataTableName)

```

Ví dụ:

```

// insert new row
myDataTable.Rows.Add(myNewDataRow);
int numOfRows = mySqlDataAdapter.Update(myDataTable);

// update
DataRow myEditDataRow = myDataTable.Rows.Find("J5COM");
myEditDataRow["CompanyName"] = "Widgets Inc.";
myEditDataRow["Address"] = "1 Any Street";

int numOfRows = mySqlDataAdapter.Update(myDataTable);

```

6. Các sự kiện của DataAdapter

EVENT	EVENT HANDLER	DESCRIPTION
FillError	FillErrorEventHandler	Phát sinh khi có lỗi xảy ra trong quá trình gọi phương thức Fill().
RowUpdating	RowUpdatingEventHandler	Phát sinh trước khi thêm, cập nhật hay xóa một hàng khỏi cơ sở dữ liệu bằng cách gọi phương thức Update().
RowUpdated	RowUpdatedEventHandler	Phát sinh sau khi thêm, cập nhật hay xóa một hàng khỏi cơ sở dữ liệu bằng cách gọi phương thức Update().

```

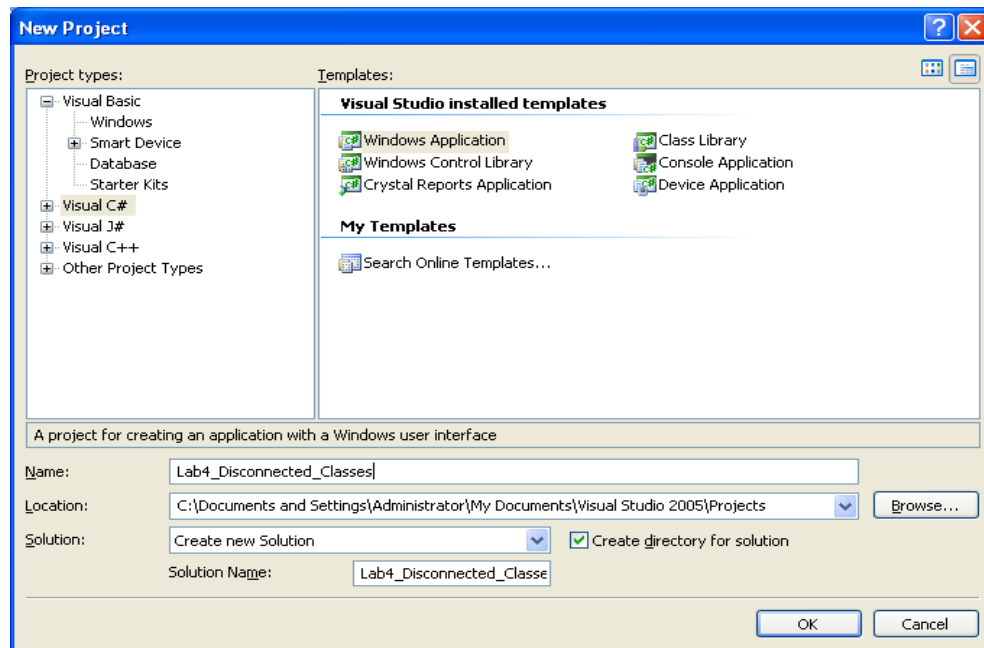
// Đăng ký trình xử lý sự kiện
mySqlDataAdapter.FillError +=
    new FillErrorEventHandler(FillErrorEventHandler);

// Hàm xử lý sự kiện
public static void FillErrorEventHandler(
    object sender, FillErrorEventArgs myFEEA ) {
    if (myFEEA.Errors.GetType() == typeof(System.OverflowException))
    {
        Console.WriteLine("A loss of precision occurred");
        myFEEA.Continue = true;
    }
}

```

III. Hướng dẫn thực hành

Tạo một dự án Windows Application mới, đặt tên là Lab4_Disconnected_Classes



Trong bài thực hành này, chúng ta sẽ tạo một Form cho phép thêm mới hóa đơn và một Form để tìm kiếm thông tin sử dụng DataView.

Nhấp đôi chuột vào Form1 và thiết kế Form có dạng như sau:

GroupBox:
Name: grbDetails

DataGridView
Name: grvOrderDetails

2 Button
Name: btnAddOrder
Name: btnCancel

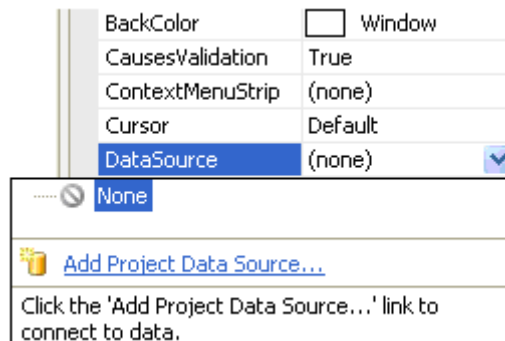
2 DateTimePicker
Name: dtpRequiredDate
Name: dtpShippedDate
CustomFormat: dd/MM/yyyy
Format: Custom

1. Cấu hình DataSource cho các ComboBox

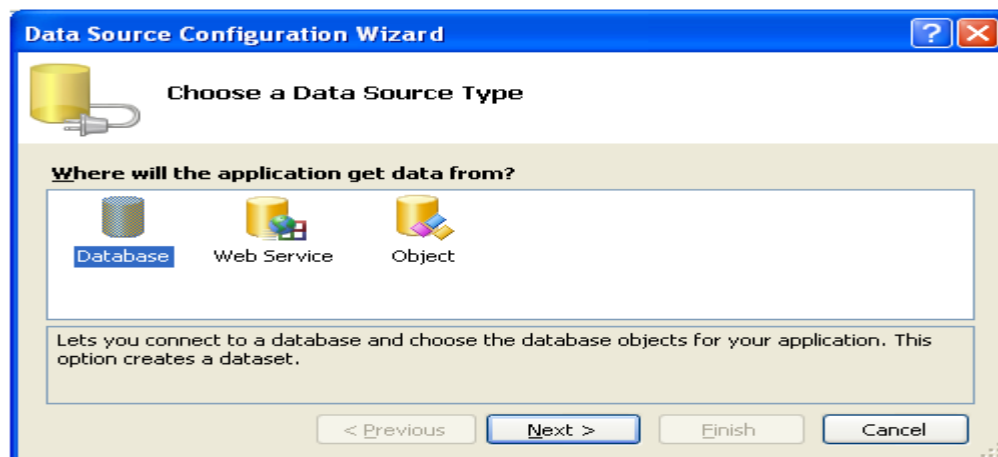
Ở bài thực hành trước, bạn đã biết cách viết mã lệnh để đưa dữ liệu vào ComboBox. Trong phần này, ta sẽ sử dụng tính năng có sẵn của VS.Net để kết nối trực tiếp tới cơ sở dữ liệu và tạo DataSource cho các ComboBox.

Cấu hình DataSource cho ComboBox chứa danh sách khách hàng

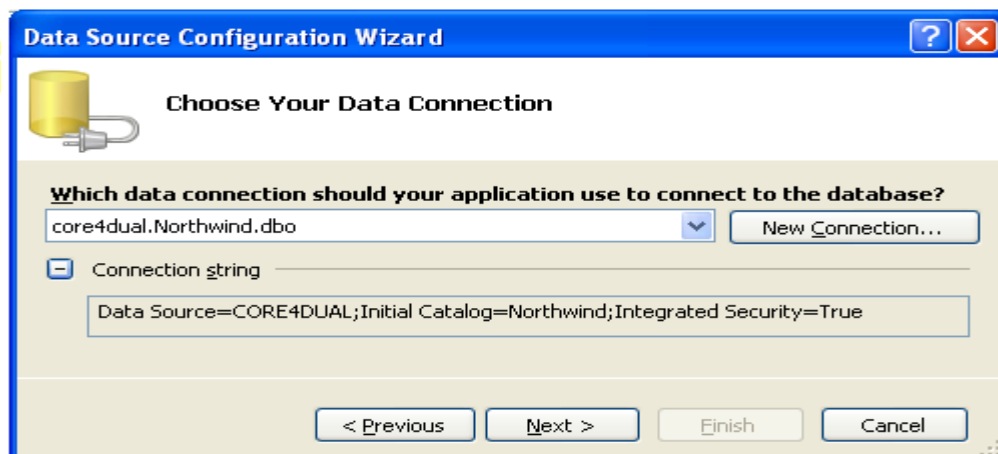
Nhấp phải chuột lên cbbCustomers, chọn Properties. Trong cửa sổ Properties, chọn thuộc tính DataSource rồi chọn Add Project Data Source...

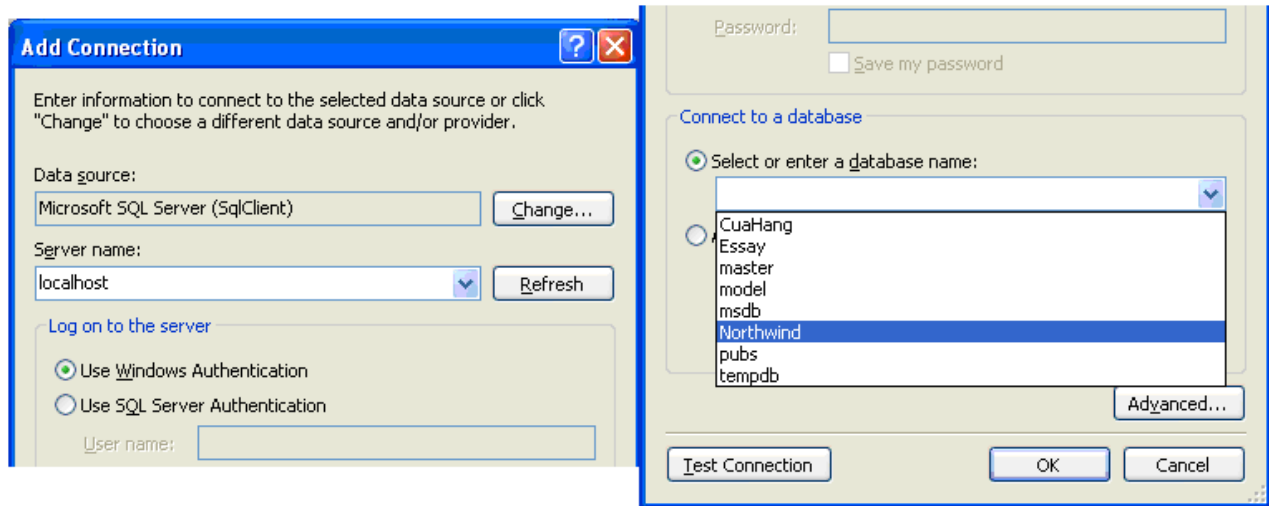


Trong cửa sổ Data Source Configuration Wizard, chọn Database. Nhấn Next.

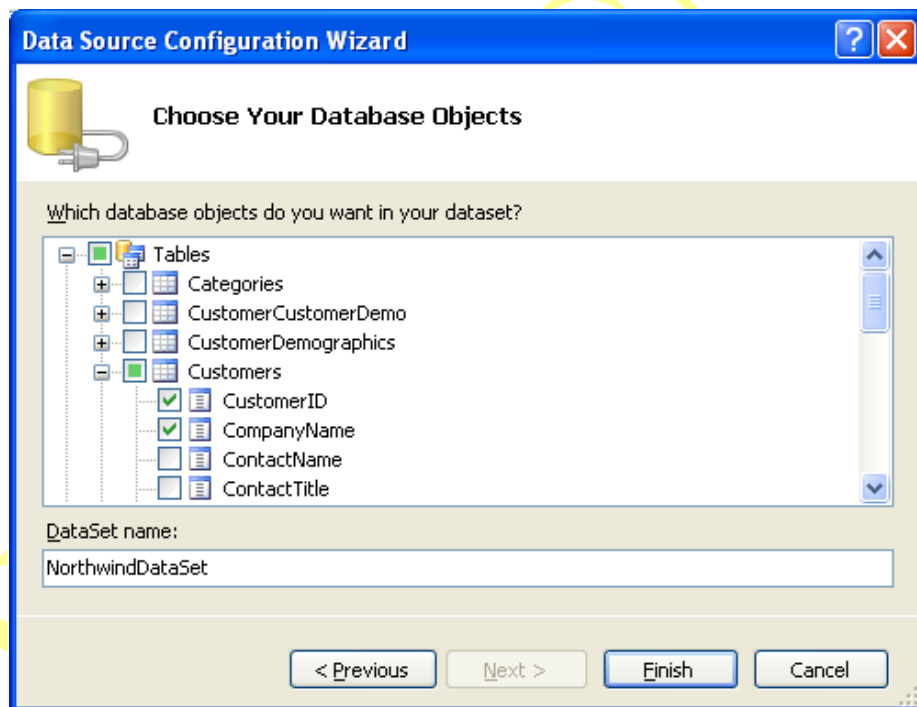


Chọn cơ sở dữ liệu cần kết nối. Nhấn Next 2 lần.

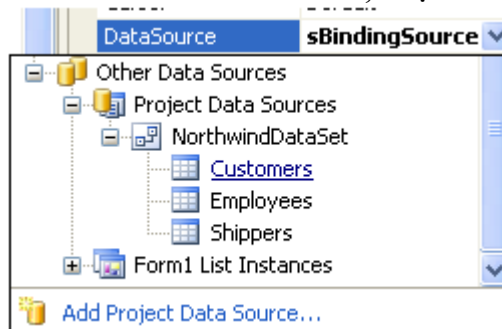




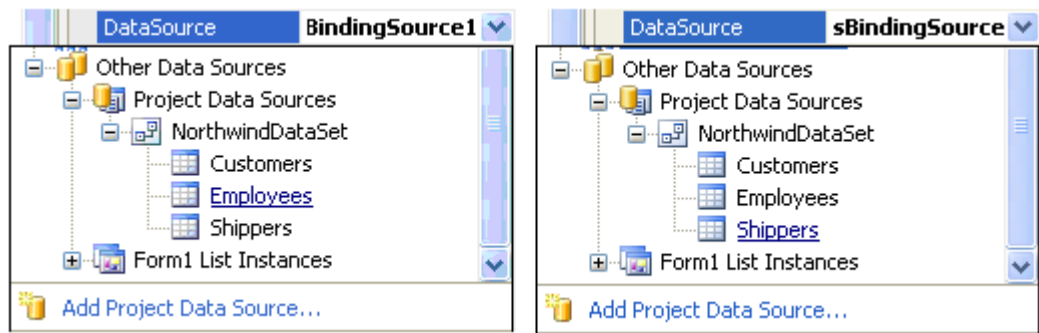
Trong mục Choose Your Database Objects, chọn Customers và chọn 2 trường đầu tiên. Chọn Employees và 3 trường EmployeeID, FirstName, LastName. Chọn Shippers và 2 trường ShipperID, CompanyName.



Trong mục DataSource của ComboBox cbbCustomer, chọn:



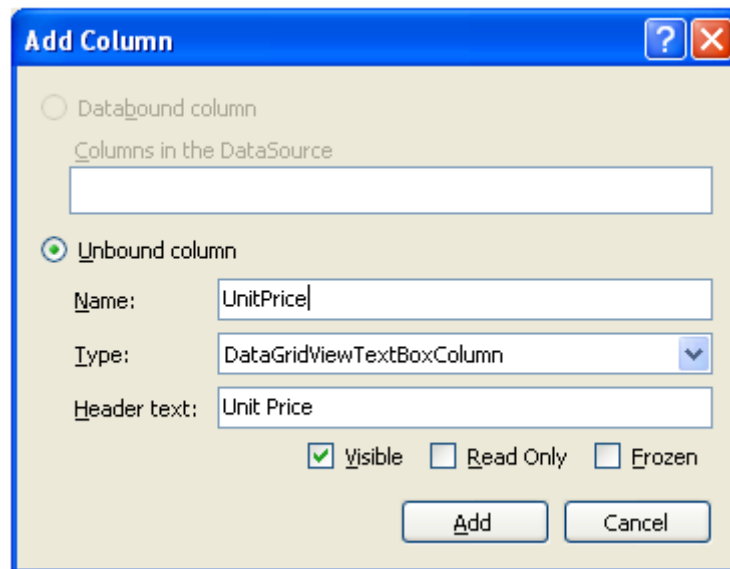
Thực hiện tương tự cho thuộc tính DataSource của cbbShipVia



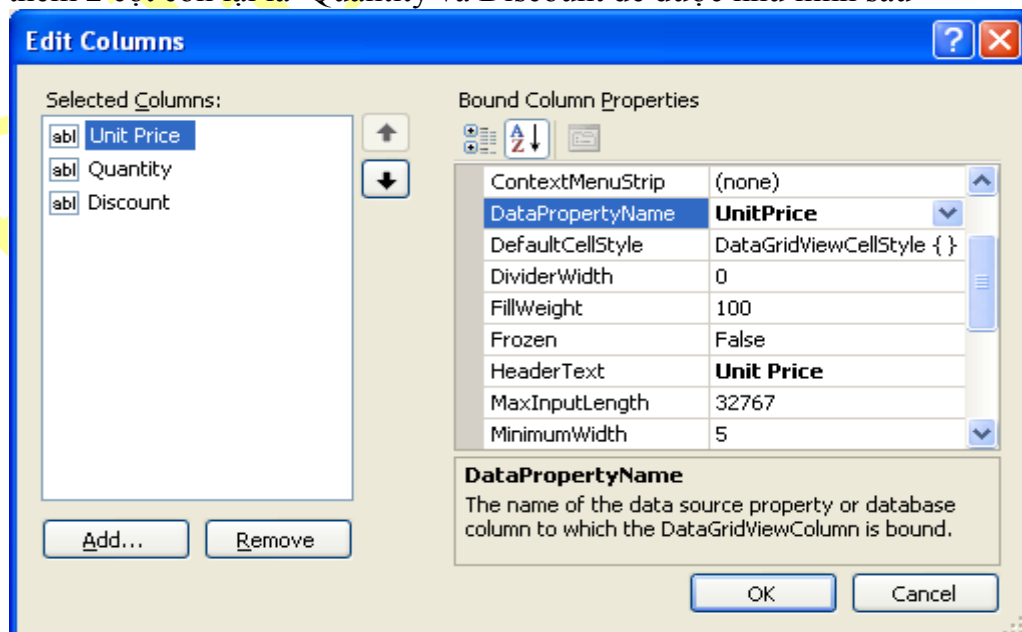
2. Tạo cấu trúc cho DataGridView

Nhấp phải chuột lên DataGridView, chọn Properties, nhấn vào nút “...” bên cạnh thuộc tính Columns để thêm các cột cho DataGridView.

Nhấn nút Add để tạo thêm cột mới, đặt tên cột và tiêu đề cho cột (Header Text)

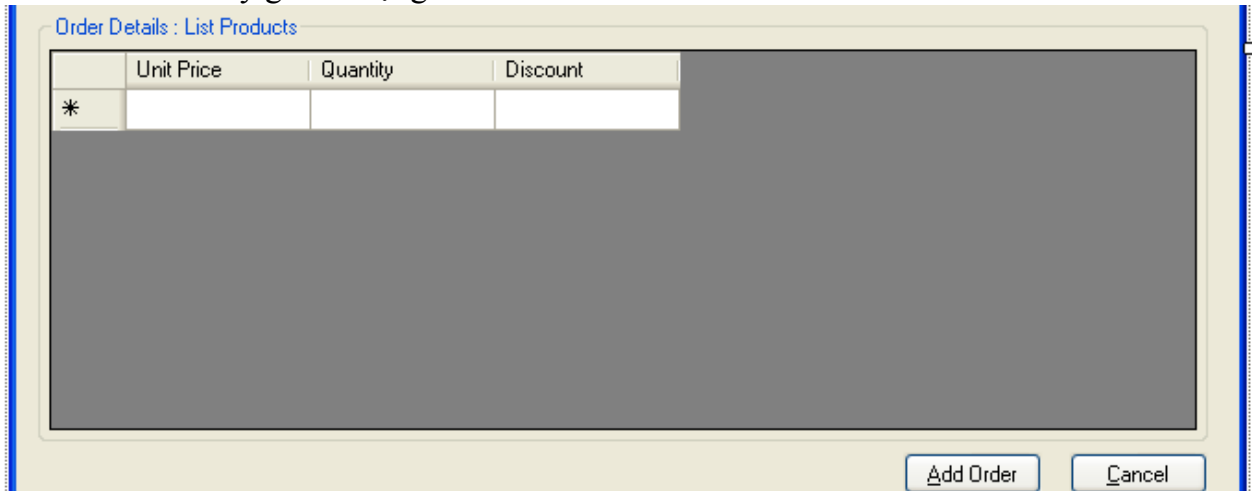


Tiếp tục thêm 2 cột còn lại là Quantity và Discount để được như hình sau



Thiết lập giá trị cho thuộc tính `DataPropertyName` lần lượt là: `UnitPrice`, `Quantity` và `Discount`. Các giá trị này tương ứng với các cột trong bảng `Order Details` của cơ sở dữ liệu `Northwind`.

`DataGridView` bây giờ có dạng sau



Nhấp đôi chuột vào 1 vùng trống của Form, thêm hàm `BuildGridViewStructure` như sau để tạo cấu trúc cho `DataGridView`

```
// Xây dựng cấu trúc cho DataGridView
private void BuildGridViewStructure()
{
    SqlConnection connection = new SqlConnection(
        "server=(local);database=Northwind;uid=sa;pwd=sa;");
    SqlCommand command = connection.CreateCommand();

    command.CommandText =
        "SELECT ProductID, UnitPrice, Quantity, Discount " +
        "FROM [Order Details]";
    SqlDataAdapter adapter = new SqlDataAdapter(command);

    productsTable = new DataTable("Products");

    // Lấy cấu trúc của bảng order details đưa vào DataTable
    adapter.FillSchema(productsTable, SchemaType.Mapped);

    // Lấy danh sách ID và tên sản phẩm
    DataTable dt = new DataTable("Products");
    command.CommandText =
        "SELECT ProductID, ProductName FROM Products";

    // Đưa vào một DataTable
    adapter = new SqlDataAdapter(command);
    adapter.Fill(dt);

    // Tạo một cột mới cho DataGridView để
    // Chọn tên sản phẩm thay vì phải nhập ID
    DataGridViewComboBoxColumn productIdColumn =
        new DataGridViewComboBoxColumn();
```

```

productIdColumn.DisplayIndex = 0;
productIdColumn.Width = 250;
productIdColumn.DataPropertyName = "ProductID";
productIdColumn.DataSource = dt;
productIdColumn.DisplayMember = "ProductName";
productIdColumn.HeaderText = "Product Name";
productIdColumn.Name = "ProductNameColumn";
productIdColumn.ValueMember = "ProductID";

// Thêm cột vừa tạo vào GridView
grvOrderDetails.Columns.Insert(0, productIdColumn);

// gán DataTable làm dữ liệu nguồn cho GridView
grvOrderDetails.DataSource = productsTable;
}

```

Trong phương thức trên có sử dụng biến `productsTable`, đây là biến toàn cục để lưu trữ danh sách sản phẩm trong chi tiết hóa đơn. Bạn khai báo biến này ngay trước phương thức khởi tạo của `Form1`.

```

public partial class Form1 : Form
{
    private DataTable productsTable;

    public Form1()

```

Sau đó, thêm đoạn mã sau vào phương thức `Form1_Load`

```

private void Form1_Load(object sender, EventArgs e)
{
    // Không tự động sinh các cột tương ứng từ csdl
    grvOrderDetails.AutoGenerateColumns = false;

Không nhập đoạn mã này



    this.BuildGridViewStructure();

    // Tạo phương thức xử lý sự kiện khi nhập giá trị
    // cho các cột trong DataGridView
    grvOrderDetails.CellValueChanged +=
        new DataGridViewCellEventHandler(grvOrderDetails_CellValueChanged);
}

void grvOrderDetails_CellValueChanged(object sender, DataGridViewCellEventArgs e)
{
    // Lấy dòng đang nhập dữ liệu
    DataGridViewRow currRow = grvOrderDetails.Rows[e.RowIndex];

    // Lấy ID của sản phẩm vừa chọn
    object nid = currRow.Cells[0].Value;

    // Duyệt qua tất cả các sản phẩm đã chọn
    for (int index = 0; index < e.RowIndex; index++)
    {

```

```
// Nếu giá trị mới nhập trùng với 1 sản phẩm đã chọn
if (grvOrderDetails.Rows[index].Cells[0].Value.ToString() == nid.ToString())
{
    // Thông báo lỗi
    MessageBox.Show("This product had been selected.", "Invalid Value");

    // Xóa dòng đang nhập
    grvOrderDetails.Rows.Remove(currRow);
    break;
}
}
```

Quay trở lại khung thiết kế Form1, chọn DataGridView, trong khung Properties, chọn Events. Nhấp đôi chuột vào sự kiện DataError để viết hàm quản lý lỗi phát sinh.

```
private void grvOrderDetails_DataError(object sender,
    DataGridViewDataErrorEventArgs e)
{
    // Hiển thị thông báo lỗi
    MessageBox.Show("Bạn chưa nhập đầy đủ dữ liệu cho các cột" +
        " hoặc không đúng kiểu dữ liệu", "Invalid value");

    // Bỏ qua lỗi vừa xảy ra, giữ nguyên dữ liệu đã nhập
    e.Cancel = true;
}
```

Nhấn F5 để chạy chương trình. Nhấp chuột vào ComboBox trong cột đầu tiên của DataGridView để chọn sản phẩm. Mặc dù chúng ta chọn tên sản phẩm nhưng khi lấy giá trị thì giá trị đó chính là ProductID của sản phẩm được chọn.

Product Name	Unit Price	Quantity	Discount
Carnarvon Tigers	2	1	1
Chocolate	4	2	1
Filo Mix			
Chocolate			
Escargots de Bourgogne			
Filo Mix			
Flotemysost			
Geitost			
Genen Shouyu			
Gnocchi di nonna Alice			
Gorgonzola Telino			

Quay trở lại khung thiết kế Form, nhấp đôi chuột lên 2 nút Add Order và Cancel để tạo hàm xử lý sự kiện Click

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

3. Gọi và thực thi Stored Procedure

Tiếp theo ta sẽ viết mã để thêm một hóa đơn với cùng với danh mục mặt hàng được mua. Đầu tiên, tạo Stored Procedure thêm hóa đơn mới như sau

```
CREATE PROCEDURE [InsertOrders]
(
    @OrderID [int] OUTPUT,
    @CustomerID [nvarchar] (5),
    @EmployeeID [int],
    @OrderDate [datetime],
    @RequiredDate [datetime],
    @ShippedDate [datetime],
    @ShipVia [int],
    @Freight [money],
    @ShipName [nvarchar] (40),
    @ShipAddress [nvarchar] (60),
    @ShipCity [nvarchar] (15),
    @ShipRegion [nvarchar] (15),
    @ShipPostalCode [nvarchar] (10),
    @ShipCountry [nvarchar] (15)
)
AS INSERT INTO [Northwind].[dbo].[Orders]
(
    [CustomerID], [EmployeeID],
    [OrderDate], [RequiredDate],
    [ShippedDate], [ShipVia],
    [Freight], [ShipName],
    [ShipAddress], [ShipCity],
    [ShipRegion], [ShipPostalCode],
    [ShipCountry]
)
VALUES
(
    @CustomerID, @EmployeeID,
    @OrderDate, @RequiredDate,
    @ShippedDate, @ShipVia,
    @Freight, @ShipName,
    @ShipAddress, @ShipCity,
    @ShipRegion, @ShipPostalCode,
    @ShipCountry
);
SELECT @OrderID = SCOPE_IDENTITY();
GO
```

Stored Procedure thêm một chi tiết hóa đơn.

```
CREATE PROCEDURE [InsertOrderDetails]
(
    @OrderID [int],
    @ProductID [int],
    @UnitPrice [money],
    @Quantity [smallint],
    @Discount [real]
)
AS
INSERT INTO [Northwind].[dbo].[Order Details]
(
    [OrderID], [ProductID],
    [UnitPrice], [Quantity],
    [Discount]
)
VALUES
(
    @OrderID, @ProductID,
    @UnitPrice, @Quantity,
    @Discount
);
GO
```

Viết thêm các hàm phụ để xử lý thao tác thêm hóa đơn, xóa một hóa đơn như sau

```
// Xóa bỏ một hóa đơn
private void RemoveOrders(int orderId)
{
    SqlConnection connection = new SqlConnection(
        "server=(local);database=Northwind;uid=sa;pwd=sa;");
    SqlCommand command = connection.CreateCommand();

    // Trước khi xóa hóa đơn, phải xóa chi tiết hóa đơn
    command.CommandText =
        "DELETE FROM [Order Details] WHERE OrderID = " + orderId + ";" +
        "DELETE FROM Orders WHERE OrderID = " + orderId;

    connection.Open();
    command.ExecuteNonQuery();
    connection.Close();
}

// Thêm mới một hóa đơn
private int InsertNewOrder()
{
    SqlConnection connection = new SqlConnection(
        "server=(local);database=Northwind;uid=sa;pwd=sa;");
    SqlCommand command = connection.CreateCommand();

    // Gán tên Stored Procedure cho Command
    command.CommandText = "InsertOrders";
    command.CommandType = CommandType.StoredProcedure;

    // Thêm các tham số cho lệnh
    command.Parameters.Add("@OrderID", SqlDbType.Int);
    command.Parameters.Add("@CustomerID", SqlDbType.NChar, 5);
    command.Parameters.Add("@EmployeeID", SqlDbType.Int);
    command.Parameters.Add("@OrderDate", SqlDbType.DateTime);
    command.Parameters.Add("@RequiredDate", SqlDbType.DateTime);
    command.Parameters.Add("@ShippedDate", SqlDbType.DateTime);
    command.Parameters.Add("@ShipVia", SqlDbType.Int);
    command.Parameters.Add("@Freight", SqlDbType.Money);
    command.Parameters.Add("@ShipName", SqlDbType.NVarChar, 40);
    command.Parameters.Add("@ShipAddress", SqlDbType.NVarChar, 60);
    command.Parameters.Add("@ShipCity", SqlDbType.NVarChar, 15);
    command.Parameters.Add("@ShipRegion", SqlDbType.NVarChar, 15);
    command.Parameters.Add("@ShipPostalCode", SqlDbType.NVarChar, 10);
    command.Parameters.Add("@ShipCountry", SqlDbType.NVarChar, 15);

    command.Parameters["@OrderID"].Direction = ParameterDirection.Output;
    command.Parameters["@OrderID"].Direction = ParameterDirection.Output;

    // truyền giá trị cho tham số
    command.Parameters["@CustomerID"].Value = cbbCustomers.SelectedValue;
    command.Parameters["@EmployeeID"].Value = cbbEmployees.SelectedValue;
    command.Parameters["@OrderDate"].Value = DateTime.Now;
    command.Parameters["@RequiredDate"].Value = dtpRequiredDate.Value;
```

```

command.Parameters["@ShippedDate"].Value = dtpShippedDate.Value;
command.Parameters["@ShipVia"].Value = cbbShipVia.SelectedValue;
command.Parameters["@Freight"].Value = txtFreight.Text;
command.Parameters["@ShipName"].Value = txtShipName.Text;
command.Parameters["@ShipAddress"].Value = txtShipAddress.Text;
command.Parameters["@ShipCity"].Value = txtShipCity.Text;
command.Parameters["@ShipRegion"].Value = txtShipRegion.Text;
command.Parameters["@ShipPostalCode"].Value = txtPostCode.Text;
command.Parameters["@ShipCountry"].Value = txtCountry.Text;

connection.Open();

// Thực thi Stored Procedure
int numRowsAffected = command.ExecuteNonQuery();

connection.Close();

// nếu thêm thành công, trả về ID của hóa đơn
if (numRowsAffected == 1)
    return Convert.ToInt32(command.Parameters["@OrderID"].Value);
else
    return 0; // Ngược lại, trả 0.
}

```

4. Cập nhật dữ liệu dùng DataAdapter

Trong phần này, ta sẽ dùng DataAdapter để thêm các dòng đã nhập trên DataGridView vào bảng Order Details của cơ sở dữ liệu Northwind.

Bổ sung hàm sau vào lớp Form1.cs

```

// Thêm các chi tiết hóa đơn
private bool InsertOrderDetails(int orderId)
{
    try
    {
        SqlConnection connection = new SqlConnection(
            "server=(local);database=Northwind;uid=sa;pwd=sa;");
        SqlCommand command = connection.CreateCommand();

        // Gán tên thủ tục cho đối tượng Command
        command.CommandText = "InsertOrderDetails";
        command.CommandType = CommandType.StoredProcedure;

        // Thêm tham số cho lệnh. Lưu ý: ta không gán giá
        // trị cho các tham số này. Thay vào đó, giá trị sẽ
        // Được lấy từ DataTable. Tên cột lấy ở tham số thứ 4
        command.Parameters.Add("@OrderID", SqlDbType.Int);
        command.Parameters.Add("@ProductID", SqlDbType.Int, 4, "ProductID");
        command.Parameters.Add("@UnitPrice", SqlDbType.Money, 8, "UnitPrice");
        command.Parameters.Add("@Quantity", SqlDbType.SmallInt, 2, "Quantity");
        command.Parameters.Add("@Discount", SqlDbType.Real, 4, "Discount");
    }
}

```

```

// Gán giá trị mặc định cho tham số OrderID
command.Parameters["@OrderID"].Value = orderId;

// Tạo đối tượng DataAdapter để đồng bộ dữ liệu
SqlDataAdapter adapter = new SqlDataAdapter();
adapter.InsertCommand = command;

// gọi phương thức Update để thêm chi tiết hóa đơn
adapter.Update(productsTable);

return true;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message + "\r\n Data will be removed.", "Error");
    return false;
}
}

```

Tạo thêm phương thức sau để thiết lập lại giá trị trên các control của Form

```

private void ResetForm()
{
    txtCountry.Text = "";
    txtFreight.Text = "";
    txtPostCode.Text = "";
    txtShipAddress.Text = "";
    txtShipCity.Text = "";
    txtShipName.Text = "";
    txtShipRegion.Text = "";

    dtpRequiredDate.Value = DateTime.Now;
    dtpShippedDate.Value = DateTime.Now;

    cbbCustomers.ResetText();
    cbbEmployees.ResetText();
    cbbShipVia.ResetText();

    // Xóa các dòng đã nhập trong GridView
    productsTable.Rows.Clear();
}

```

Bổ sung đoạn mã sau vào phương thức xử lý sự kiện Click của nút Add Order.

```

private void btnAddOrder_Click(object sender, EventArgs e)
{
    int orderId = this.InsertNewOrder();
    if (orderId > 0)
    {
        if (this.InsertOrderDetails(orderId))
            this.ResetForm();
        else
            this.RemoveOrders(orderId);
    }
    else
        MessageBox.Show("Insert order failed. Please try again.");
}

```


Nhấn phím F5 để chạy chương trình

Tạo hóa đơn mới

Customer: Antonio Moreno Taquería Employee: Leverling Janet

Required Date: 28/07/2009 Shipped Date: 28/07/2009

Ship Via: United Package Freight: 123000000

Ship Name: Peter Smart Ship Address: Wenterbourg

Ship City: Hambourg Ship Region: North East

Post Code: 0013 Country: Germany

Order Details : List Products

Product Name	Unit Price	Quantity	Discount
Boston Crab Meat	20000	14	0.01
Chai	32000	47	0
Chocolate	12000	100	0
Geitost	21500	50	0.05
Gravad lax	8000	120	0
*			

Add Order Cancel

5. Tạo Form tìm kiếm – trích lọc – sắp xếp

Tạo 2 Form mới, đặt tên lần lượt là DataFilterForm, FEBuilderForm và thiết kết giao diện như hình sau.

DataFilterForm

clbColumns

CheckedListBox
Name: clbColumns
Dock: Fill

DataGridView
Name: grvCustomers
Dock: Fill

Button
Name: btnCustomView

Button
Name: btnReset

Button
Name: btnLoadData

Button
Name: btnBuildFE

Button
Name: btnCancel

Reset Load Data Custom View Build Filter Expression Cancel

Lưu ý, đầu tiên, từ Toolboxes, kéo một control SplitterPanel vào Form, chọn thuộc tính Orientation là Horizontal. Lúc này, Form có 2 khung nằm ngang. Điều chỉnh để khung bên dưới vừa đủ chứa các Button.

Tiếp tục kéo thêm một SplitterPanel nữa vào khung phía trên của Form và điều chỉnh lại khung bên trái giống như vùng màu trắng trong hình trên.

Trong Form FilterExpressionForm, khai báo thêm thuộc tính Expression

```
public partial class FilterExpressionForm : Form
{
    private string expression = "";

    public string Expression
    {
        get { return expression; }
        //set { expression = value; }
    }
}
```

Nhấp đôi chuột lên Form để tạo phương thức xử lý sự kiện Form Load.

```
private void FilterExpressionForm_Load(object sender, EventArgs e)
{
    this.CheckRadioState();
    cbbOperators.SelectedIndex = 0;
}

private void CheckRadioState()
{
    bool enabled = (lvwExpressions.Items.Count > 0);

    radAnd.Enabled = enabled;
    radOr.Enabled = enabled;
}
```

Nhấp đôi chuột vào 2 Button Add, Close để xử lý sự kiện Click như sau:

```
private void btnAdd_Click(object sender, EventArgs e)
{
    string andor = "";

    // Kiểm tra radio nào được chọn
    if (radAnd.Enabled)
    {
        if (radAnd.Checked) andor = "AND";
        else andor = "OR";
    }

    // Tạo ListViewItem mới chứa biểu thức
    ListViewItem item = new ListViewItem(andor);
    item.SubItems.Add(cbbColumnNames.Text + ' ' +
        cbbOperators.Text + ' ' + txtValue.Text);

    // Thêm mới item vào listview
    lvwExpressions.Items.Add(item);

    this.CheckRadioState();
}

private void btnClose_Click(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder();

    foreach (ListViewItem item in lvwExpressions.Items)
    {
        sb.Append(item.Text + ' ' + item.SubItems[1].Text);
    }
    expression = sb.ToString();

    this.DialogResult = DialogResult.OK;
}
```

Bổ sung thêm phương thức sau để điền tên các cột trong DataTable vào ComboBox

```
public void AddColumns(object[] items)
{
    cbbColumnNames.Items.Clear();
    cbbColumnNames.Items.AddRange(items);

    if (items.Length > 0)
        cbbColumnNames.SelectedIndex = 0;
}
```

Trong Form DataFilterForm, nhấp đôi chuột vào các Button để xử lý sự kiện Click như sau:

```
public partial class DataFilterForm : Form
{
    private DataTable customersTable;
    private DataView customersView;

    public DataFilterForm()
    {
        InitializeComponent();

        // Ẩn khung chứa CheckListBox
        splitContainer2.Panel1Collapsed = true;
    }

    private void btnLoadData_Click(object sender, EventArgs e)
    {
        SqlConnection connection = new SqlConnection(
            "server=(local);database=Northwind;uid=sa;pwd=sa;");
        SqlCommand command = connection.CreateCommand();

        // Lấy danh sách tất cả các khách hàng
        command.CommandText =
            "SELECT * FROM Customers";

        customersTable = new DataTable("Customers");


        // đưa vào một DataTable
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(customersTable);

        // Tạo DataView để trích lọc
        customersView = new DataView(customersTable);
        grvCustomers.DataSource = customersView;

        // Hiển thị danh sách các cột của bảng Customers
        // lên một CheckListBox để chọn cột nào được hiển thị
        clbColumns.Items.Clear();
        foreach (DataColumn dc in customersTable.Columns)
        {
            clbColumns.Items.Add(dc.ColumnName, true);
        }

        // Ẩn - hiện các Button
        btnBuildFE.Enabled = true;
        btnCustomView.Enabled = true;
        btnReset.Enabled = true;
        btnLoadData.Enabled = false;
    }

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



```

private void btnCustomView_Click(object sender, EventArgs e)
{
    if (splitContainer2.Panel1Collapsed)
        splitContainer2.Panel1Collapsed = false;
    else
        splitContainer2.Panel1Collapsed = true;
}

private void btnBuildFE_Click(object sender, EventArgs e)
{
    FilterExpressionForm dialog = new FilterExpressionForm();

    object[] items = new object[clbColumns.Items.Count];
    for (int index = 0; index < clbColumns.Items.Count; index++)
    {
        items[index] = clbColumns.Items[index];
    }

    dialog.AddColumnns(items);
    if (dialog.ShowDialog(this) == DialogResult.OK)
    {
        customersView.RowFilter = dialog.Expression;
        //grvCustomers.DataSource = customersView;
    }
}

private void btnReset_Click(object sender, EventArgs e)
{
    // Trả lại trạng thái ban đầu cho DataView
    customersView.RowFilter = "";
    grvCustomers.DataSource = customersView;

    // Chọn tất cả các item trong CheckListBox
    for (int index = 0; index < clbColumns.Items.Count; index++)
    {
        if (clbColumns.GetItemChecked(index) == false)
            clbColumns.SetItemChecked(index, true);
    }
}

```

Nhấp phải chuột lên CheckedListBox, chọn Properties. Nhấn nút Events và nhấp đôi vào mục ItemCheck. Bổ sung đoạn mã sau để xử lý sự kiện này.

```

private void clbColumns_ItemCheck(object sender, ItemCheckEventArgs e)
{
    // Nếu tên cột trong CheckListBox không được chọn
    // thì ẩn cột tương ứng trong Gridview

    if (e.NewValue == CheckState.Unchecked)
        grvCustomers.Columns[e.Index].Visible = false;
    else
        grvCustomers.Columns[e.Index].Visible = true;
}

```

Nhấn phím F5 để chạy chương trình.

DataFilterForm

FilterExpressionForm

Các biểu thức lọc hiện tại

And / Or	Biểu thức
OR	CompanyName LIKE 'Fr%'
	ContactName LIKE 'F%'

Expression Builder

Kết hợp: ☐ AND ☒ OR

Tên Cột: ContactName

Phép toán: LIKE

Giá trị: 'F%'

Buttons: Add, Close

Buttons: Custom View, Build Filter Expression, Cancel

DataFilterForm

CustomerID	CompanyName	ContactName	ContactTitle	PostalCode
BLONP	Blondesdssl père et fils	Frédérique Citeaux	Marketing Manager	67000
CENTC	Centro comercial Moctezuma	Francisco Chang	Marketing Manager	05022
FRANK	Frankenversand	Peter Franken	Marketing Manager	80805
FRANR	France restauration	Carine Schmitt	Marketing Manager	44000
FRANS	Franchi S.p.A.	Paolo Accorti	Sales Representative	10100
LINOD	LIND-Delicateses	Felipe Izquierdo	Owner	4980
LONEP	Lonesome Pine Restaurant	Fran Wilson	Sales Manager	97219

Buttons: Reset, Load Data, Custom View, Build Filter Expression, Cancel

IV. Bài tập thực hành

- Trong Form tạo mới hóa đơn (Form1), khách hàng hoặc công ty vận chuyển có thể chưa tồn tại trong cơ sở dữ liệu. Vì thế, nó không được hiển thị lên ComboBox. Trong trường hợp này, bạn cần phải bổ sung các Button vào bên cạnh các ComboBox đó.
 - Khi nhấn nút thêm mới khách hàng, mở Form mới để nhập thông tin khách hàng. Khi tắt Form này, tên và ID của khách hàng mới phải được đưa vào ComboBox Customer
 - Khi nhấn nút thêm mới công ty vận chuyển, mở Form mới để nhập thông tin, khi tắt Form này, tên và ID của công ty mới phải được đưa vào ComboBox ShipVia.
- Khi nhập dữ liệu trên DataGridView của Form1, vẫn có thể có 2 dòng cùng một sản phẩm nhưng giá trị trên các cột khác thì khác nhau. Điều này sẽ phát sinh ra lỗi khi nhấn nút Add Order. Sở dĩ có lỗi phát sinh là vì bảng Order Details có khóa kép là OrderID và ProductID. Và trường hợp này, ta có 2 dòng có cùng giá trị trên tập khóa chính. Tìm cách khắc phục lỗi này để hoàn chỉnh chương trình.

3. Trong Form1, bổ sung một Control ContextMenuStrip và gán cho thuộc tính ContextMenuStrip của DataGridView. Control này chứa 2 menu: Xóa dòng hiện tại, xóa tất cả.
Viết hàm xử lý để khi chọn menu nào thì thực hiện chức năng tương ứng trên các dòng của DataGridView
4. Thiết kế và viết hàm xử lý Form để liệt kê danh sách hóa đơn theo ngày, tháng được chọn từ một control DateTimePicker. Khi nhấp phải chuột lên một hóa đơn, hiển thị menu ngữ cảnh gồm các chức năng sau:
 - a. Xem thông tin khách hàng
 - b. Xem thông tin nhân viên lập hóa đơn
 - c. Xem danh mục hàng mua bởi hóa đơn
 - d. Xem thông tin công ty giao hàng
 - e. Thống kê: chỉ sử dụng chức năng này khi danh sách hóa đơn được liệt kê theo tháng. Bảng thống kê gồm các thông tin sau: Ngày, Số lượng hóa đơn, tổng số tiền thu được, tổng số sản phẩm bán ra.