UBND TỈNH LÂM ĐỒNG

TRƯỜNG CAO ĐẲNG ĐÀ LẠT

GIÁO TRÌNH

MÔ ĐUN: LẬP TRÌNH WINDOWS 2 (ADO.NET)

NGÀNH/NGHỀ: CÔNG NGHỆ THÔNG TIN (ƯDPM)

TRÌNH ĐỘ: CAO ĐẲNG

(Ban hành kèm theo Quyết định số: /QĐ-CĐNĐL ngày …tháng…năm… của Hiệu trưởng Trường Cao đẳng Đà Lạt)

LƯU HÀNH NỘI BỘ

Lâm Đồng, năm 2017

TUYÊN BỐ BẢN QUYỀN

LỜI GIỚI THIỆU

Để học tốt môn học này, người học cần phải có kiến thức về lập trình trên Windows cơ bản, cụ thể là ngôn ngữ C#.NET.

Lập trình Windows 2 là một mô đun lập trình nâng cao nhằm giúp người học có kiến thức và kỹ năng để xây dựng một ứng dụng chuyên nghiệp với cơ sở dữ liệu. Với phạm vi của tài liệu này, chúng tôi cung cấp cho người học các kiến thức và kỹ năng chính sau:

- Tạo các ứng dụng trên nền Windows.
- Tạo được các ứng dụng cơ sở dữ liệu trên nền Windows.
- Lập trình và sử dụng các chức năng kéo và thả, kéo và kết dính.
- Lập trình và sử dụng được các đối tượng của .NET.
- Tạo được ứng dụng cơ sở dữ liệu với các báo cáo bằng CrystalReport.
- Sử dụng được hệ thống registry và tập tin .ini để lưu trữ thông tin chương trình.
- Tạo ra các ứng dụng MDI.

Trong quá trình biên soạn, chúng tôi có tham khảo nhiều nguồn tài liệu trên Internet. Mặc dù rất cố gắng nhưng chắc chắn không tránh khỏi những thiếu sót, tác giả rất mong nhận được những ý kiến đóng góp để tài liệu ngày càng hoàn thiện hơn để cung cấp cho người học những kiến thức và kỹ năng thiết thực.

Tài liệu này được thiết kế theo từng mô đun/ môn học thuộc hệ thống mô đun/ môn học của một chương trình để đào tạo hoàn chỉnh nghề Lập trình máy tính ở trình độ Cao đẳng. Tài liệu dùng làm giáo trình học tập cho sinh viên trong các khóa đào tạo và cũng có thể được sử dụng cho đào tạo ngắn hạn hoặc công nhân kỹ thuật đồng thời có thể làm tài liệu tham khảo cho các lập trình viên.

Đà Lạt, ngày 07 tháng 07 năm 2017

Tham gia biên soạn

Chủ biên: Ths. Phạm Đình Nam

MỤC LỤC

GIÁO TRÌNH MÔ ĐUN

Tên mô đun: LẬP TRÌNH WINDOWS 2 (ADO.NET)

Mã mô đun: MĐ21

Vị trí, tính chất, ý nghĩa và vai trò của mô đun:

- Vị trí: Mô đun này được học sau các mô đun chuyên môn Lập trình windows 1, Quản trị cơ sở dữ liệu SQL Server.

- Tính chất: Là mô đun chuyên môn bắt buộc..

- Ý nghĩa và vai trò của môn học/mô đun: Tài liệu này được thiết kế theo từng mô đun/ môn học thuộc hệ thống mô đun/ môn học của chương trình đào tạo hoàn chỉnh nghề Công nghệ thông tin (Ứng dụng phần mềm) ở trình độ Cao đẳng. Tài liệu dùng làm giáo trình học tập cho sinh viên trong các khóa đào tạo và cũng có thể được sử dụng đào tạo ở Trung tâm để cấp chứng chỉ, đồng thời có thể làm tài liệu tham khảo cho các lập trình viên.

Mục tiêu của mô đun:

Về kiến thức:

- Trình bày được các khái niệm về công nghệ ADO.Net trong môi trường .NET Framework;

- Trình bày được kiến trúc của ADO.Net;

- Phân biệt được sự khác nhau giữa ADO và ADO.Net;

- Chọn đúng kết nối đến các cơ sở dữ liệu khác nhau (trọng tâm là SQL Server) và tương tác trên cơ sở dữ liệu đó;

- Về kỹ năng:

- Vận dụng ADO.Net để xây dựng ứng dụng xử lý được một số yêu cầu tương tác dữ liệu trên các cơ sở dữ liệu khác nhau;

- Tích hợp XML vào ADO.Net;

Về năng lực tự chủ và trách nhiệm:

- Có khả năng tự nghiên cứu, tự học, tham khảo tài liệu liên quan đến môn học để vận dụng vào hoạt động hoc tập.

- Vận dụng được các kiến thức tự nghiên cứu, học tập và kiến thức, kỹ năng đã được học để hoàn thiện các kỹ năng liên quan đến môn học một cách khoa học, đúng quy định.

Nội dung của mô đun:

BÀI 1. TỔNG QUAN VỀ ADO.NET

Mã Bài: MĐ21_01

Giới thiệu:

Với sự phát triển liên tục và đa dạng của thế giới công nghệ thông tin ngày nay, các phần mềm, các hệ điều hành, các môi trường phát triển và các ứng dụng liên tục ra đời. Tuy nhiên, đôi khi việc phát triển không đồng nhất và nhất là do không tương thích về mặt lợi ích của các công ty phần mềm lớn đã làm ảnh hưởng đến công việc của những kỹ sư xây dựng phần mềm.

Mục tiêu:

- Trình bày được các bước phát triển của ADO.Net;
- Trình bày được kiến trúc của ADO.Net, các đối tượng ADO.Net trong môi trường .Net Framework;
- Mô tả được cách quản lý dữ liệu của ADO.Net;
- Rèn luyện tính nghiêm túc, chủ động.

Nội dung chính:

1. Lịch sử phát triển ADO.Net

Lịch sử phát triển công nghệ kết nối cơ sở dữ liệu ODBC → DAO→ RDO → OLE DB → ADO→ ADO.NET.

Tuy nhiên ADO.NET không phải là phiên bản mới của ADO (ActiveX Data Object) và ADO.NET cũng không phải là 'ActiveX Data Object .NET'.

ADO.NET là một trong các lớp nằm trong bộ thư viện lớp cơ sở của NET Framework để cho phép các ứng dụng Windows(như c#, VB.net) hay các ứng dụng Web (như ASP.Net) thao tác dễ dàng với các nguồn dữ liệu.

Mục tiêu chính của ADO.NET là:

- Cung cấp các lớp để thao tác dữ liệu trong cả hai môi trường là phi kết nối (Disconnected data) và kết nối (Connected data).
- Tích hợp chặt chẽ với XML (Extensible Markup Language)
- Tương tác với nhiều nguồn dữ liệu thông qua mô tả chung
- Tối ưu truy cập nguồn dữ liệu (OLE DB & SQL server)
- Làm việc trên môi trường Internet

2. Phần mềm cần thiết

Để học tốt mô đun này nhằm đáp ứng nhu cầu của xu thế hiện nay, sinh viên cần chuẩn bị các phần mềm sau:

+ Microsoft Visual Studio .NET (phiên bản từ 2010 trở lên).

+ Microsoft SQL Server (phiên bản 2008 trở lên).

+ Microsoft Office (phiên bản Full, 2007 trở lên).

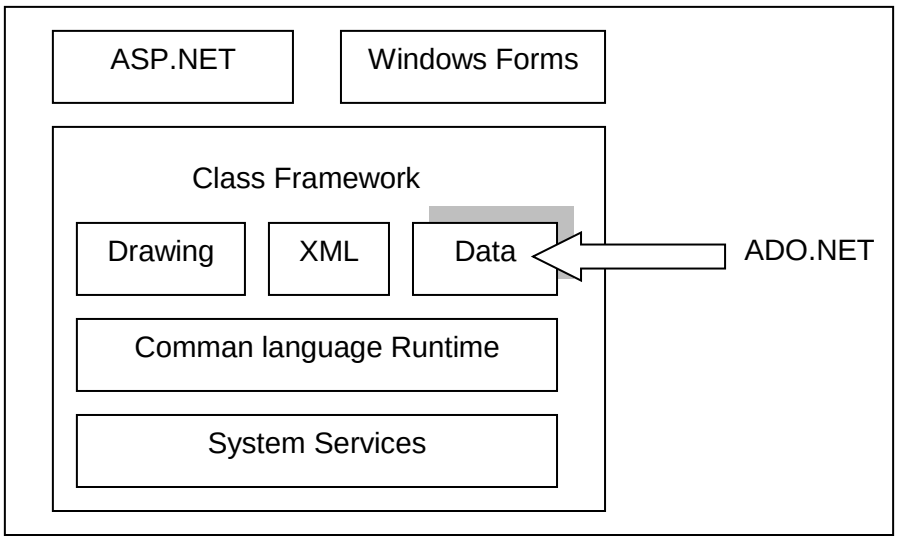
Ngoài ra cần cài thêm các công cụ hỗ trợ khác như bộ gõ tiếng Việt Unikey, bộ công cụ thiết kế báo cáo Crystal Report, …

3. Kiến trúc của ADO.Net

Các lớp của ADO.NET được đặt trong Namespase là System.Data

ADO.NET bao gồm 2 provider để thao tác với các cơ sở dữ liệu là OLEDB provider (nằm trong System.Data.OLEDB) dùng để truy xuất đến bất kỳ CSDL có hỗ trợ OLEDB; SQL Provider dữ liệu (nằm trong System.Data.SQLClient) chuyên dùng để truy xuất đến CSDL SQL Server (Không qua OLE DB nên nhanh hơn).

Vị trí ADO.NET trong kiến trúc của .NET Framework:



Hình 1.1: Vị trí của ADO.NET trong kiến trúc của .Net Framework

Từ kiến trúc ta tây rằng ADO.NET là một phần nội tại của .NET Framework, do vậy nó có thể được sử dụng trong tất cả các ngôn ngữ hộ trợ .NET như C#, VB.Net… mà không có sự khác biệt nào (Tức là các chức năng cũng như cách sử dụng hoàn toàn giống nhau).

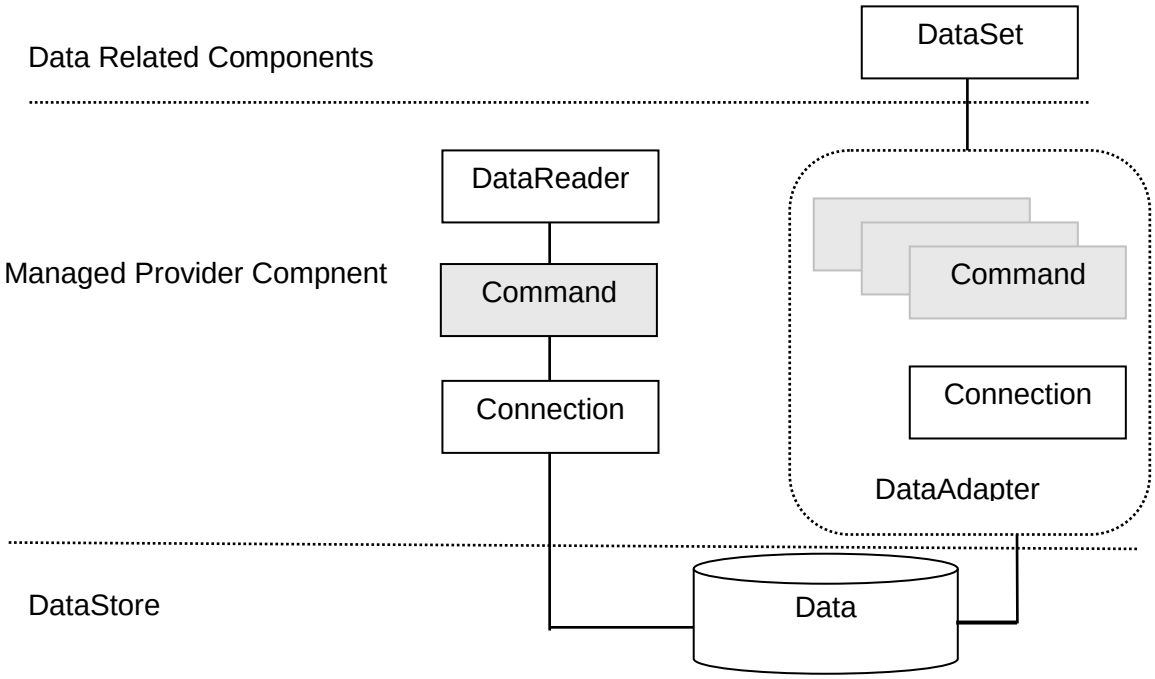
ADO.NET được thiết kế để kết nối với cả dữ liệu phi kết nối trong môi trường đa tầng (Multi – Tier). Nó sử dụng XML để trao đổi dữ liệu phi kết nối do vậy dễ dàng khi giao tiếp giữa các ứng dụng không phải trên nền Windows.

ADO.NET hỗ trợ hoàn toàn XML, nghĩa là chúng ta có thể nạp dữ liệu từ một tệp XML và thao tác như một CSDL, sau đó cũng có thể lưu kết quả ngược trở lại tệp XML do vậy có thể đi qua FireWall một cách dễ dàng.

4. Các đối tượng ADO.Net trong .NET Framework

Các thành phần chính của ADO.NET

- Connection
- Command
- Datareader
- DataAdapter
- DataSet



Hình 1.2: Các tầng kiến trúc của ADO.NET

5. Biểu diễn dữ liệu trong bộ nhớ

Kiến trúc ADO.NET có thể chia làm 2 phần chính:

Managed Provider Component: bao gồm các đối tượng như DataAdapter, DataReader,… giữ nhiệm vụ làm việc trực tiếp với dữ liệu như database, file,…

Content Component: bao gồm các đối tượng như DataSet, DataTable,… đại diện cho dữ liệu thực sự cần làm việc. DataReader là đối tượng mới, giúp truy cập dữ liệu nhanh chóng nhưng forward-only và read-only giống như ADO RecordSet sử dụng Server cursor, OpenFowardOnly và LockReadOnly.

DataSet cũng là một đối tượng mới, không chỉ là dữ liệu, DataSet có thể coi là một bản sao gọn nhẹ của CSDL trong bộ nhớ với nhiều bảng và các mối quan hệ. DataAdapter là đối tượng kết nối giữa DataSet và CSDL, nó bao gồm 2 đối tượng Connection và Command để cung cấp dữ liệu cho DataSet cũng như cập nhật dữ liệu từ DataSet xuống CSDL.

6. Quản lý Recordset

Để có thể hiểu rõ được cách làm việc của ADO.NET, chúng ta cần phải nắm được một số khái niệm cơ bản về cơ sở dữ liệu quan hệ và ngôn ngữ truy vấn dữ

liệu, như: khái niệm về dòng, cột, bảng, quan hệ giữa các bảng, khóa chính, khóa ngoại và cách truy vấn dữ liệu trên các bảng bằng ngôn ngữ truy vấn SQL : SELECT, INSERT, UPDATE, DELETE hay cách viết các hàm, thủ tục (Store Procedure) trong SQL Server. Trong phạm vi của tài liệu này, chúng ta sẽ không đề cập đến các mục trên.

BÀI 2. TƯƠNG TÁC VỚI CƠ SỞ DỮ LIỆU

Mã Bài: MĐ20_02

Giới thiệu:

Có 2 cách để tương tác với cơ sở dữ liệu đó là có thể sử dụng các đối tượng ADO.NET xây dựng sẵn để kết nối và tương tác hoặc hoặc sử dụng lệnh lập trình. Trong phạm vi bài này chỉ đề cập đến cách làm việc thứ hai, đó là sử dụng lệnh lập trình để thực hiện mọi công việc tương tác với cơ sở dữ liệu.

Mục tiêu:

- Trình bày được công dụng, tính năng của một số đối tượng trong ADO.Net;
- Phân biệt được các thuộc tính, phương thức của các đối tượng như Connection, Command, DataReder;
- Viết được một số đoạn chương trình để kết nối và tương tác trên cơ sở dữ liệu SQL Server hay Access;
- Rèn luyện tính nghiêm túc, tỉ mỉ. Tích cực làm bài tập.

Nội dung chính:

1. Đối tượng Connection

➢Chức năng: Là đối tượng có nhiệm vụ thực hiện nhiệm vụ kết nối đến CSDL để các đối tượng như Command thao tác với CSDL thông qua Connection này.

➢Khai báo:

```
using System.Data.SqlClient;
public partial class Form1 : Form
{
String str=("Server=myServer;Database=QLSVien;uid=sa;
pwd=123456");
public SqlConnection con=new SqlConnection (st);
private void Form1_Load(object sender, EventArgs e)
    {
        con.Open();
        if (con.State == ConnectionState.Open)
                        {
                       MessageBox.Show("Ket noi thanh cong" );
                       }
    }
}
```

Mở kết nối: Thi hành phương thức Open() để mở kết nối.

```
con.Open();
```

Kiểm tra kết nối: Sau khi gọi phương thức Open, có thể xem đã kết nối thành công hay không thông qua thuộc tính State của Connection:

```
if (con.State == ConnectionState.Open)
```

Đóng kết nối: Thi hành phương thức Close() để đóng kết nối.

Để tránh lỗi ta nên kiểm tra trạng thái kết nối:

```
private void Form1_FormClosing(object sender,
FormClosingEventArgs e) {
            if (con.State == ConnectionState.Open)
            {
                con.Close();
            }
        }
```

2. Đối tượng Command

Công dụng: Dùng để thực hiện các câu lệnh thao tác với CSDL như: Insert, Update, Select, Delete…

Có nhiều cách để tạo đối tượng Command, chẳng hạn chúng ta có thể khai báo đối tượng Command hoặc tạo từ phương thức CreateCommand() của đối tượng Connection.

2.1 Tạo Command từ phương thức tạo dựng

```
public SqlCommand dc = new SqlCommand("select * from sinhvien", con);
```

2.2 Tạo command từ phương thức CreateCommand của đối tượng Connection

```
SqlCommand command = con.CreateCommand();
command.CommandText = "select * from sinhvien";
```

2.3 Tạo Command bằng cách đặt các thuộc tính sau khi khai báo

```
SqlCommand com = new SqlCommand();
com.CommandType = CommandType.Text
com.CommandText ="Select * from sinhvien";
com.Connection = con;
```

Phương thức ExcuteReader: Phương thức này sẽ trả về một tập các bản ghi và thường được sử dụng để thực thi câu lệnh truy vấn Select. Kết quả có thể lưu trữ trong đối tượng DataReader để thao tác.

Cú pháp:

```
Biến_DataReader=com.ExcuteReader();   //com biến SqlCommand.
```

Phương thức ExcuteScalar(): Phương thức này sẽ trả về phần tử cột đầu tiên hàng đầu tiên trong bảng kết quả.

Phương thức này thường được sử dụng thực hiện câu lệnh truy vấn Select mà kết quả trả về chỉ có một hàng và một cột (Select Count(*) from sinhvien).

Phương thức ExcuteNonQuery: Được sử dụng để thực thi các câu lệnh truy vấn hành động: Insert, Update, Delete…

Phương thức ExcuteXMLReader: Tạo bộ đọc từ File XML.

3. Đối tượng DataReader

Công dụng: Dùng để lấy kết quả trả về từ đối tượng Command. Tuy nhiên dữ liệu là chỉ đọc theo chiều tiến (ReadOnly).

Khai báo và lấy dữ liệu từ Command. Đối tượng DataReader không có phương thức khởi tạo:

```
private void button1_Click(object sender, EventArgs e)
        {
            StringBuilder st = new StringBuilder();
            dc = new SqlCommand("select * from sinhvien", con);
               SqlDataReader dr ;          // Khai báo DataReader
            dr = dc.ExecuteReader();      // Lấy dữ liệu từ Command
            while (dr.Read() == true)     // Duyệt qua các bản ghi
            {
                st.AppendLine(dr.GetValue(2).ToString()  );
            }
            MessageBox.Show(st.ToString());
            dr.Close();
            dc.Dispose();
        }
```

BÀI 3. XỬ LÝ DỮ LIỆU

Mã Bài: MĐ21_03

Giới thiệu:

Bài này tập trung việc khai báo và sử dụng lớp đối tượng cũng như cách kế thừa trong lớp đối tượng.

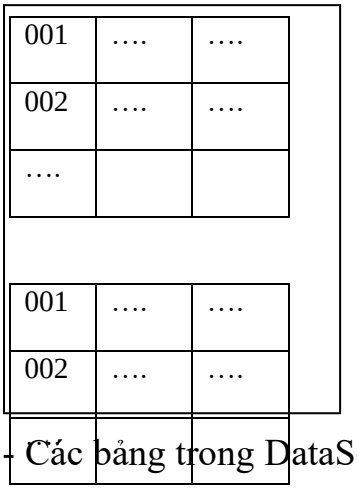
Mục tiêu:

- Liệt kê được tính năng, thành phần của đối tượng DataSet;
- Đưa được dữ liệu vào DataTable trong DataSet;
- Viết được các đoạn mã lệnh để định nghĩa dữ liệu đặc tả rồi truy xuất trực tiếp với DataTable;
- Rèn luyện thái độ tích cực, tỉ mỉ, sáng tạo. Chủ động tìm thêm nguồn bài tập thực hành.

Nội dung chính:

1. Thành phần DataSet

Đối tượng DataSet được coi như một kho chứa các bảng (Table). Người sử dụng có thể thay đổi dữ liệu trong các bảng này và khi muốn cập nhật vào cơ sở dữ liệu thì thi hành phương thức Update của đối tượng DataAdapter.



- Các bảng trong DataSet có thể do DataAdapter đổ vào (Fill) hoặc cũng có thể là các bảng được tạo thành từ đối tượng DataTable.

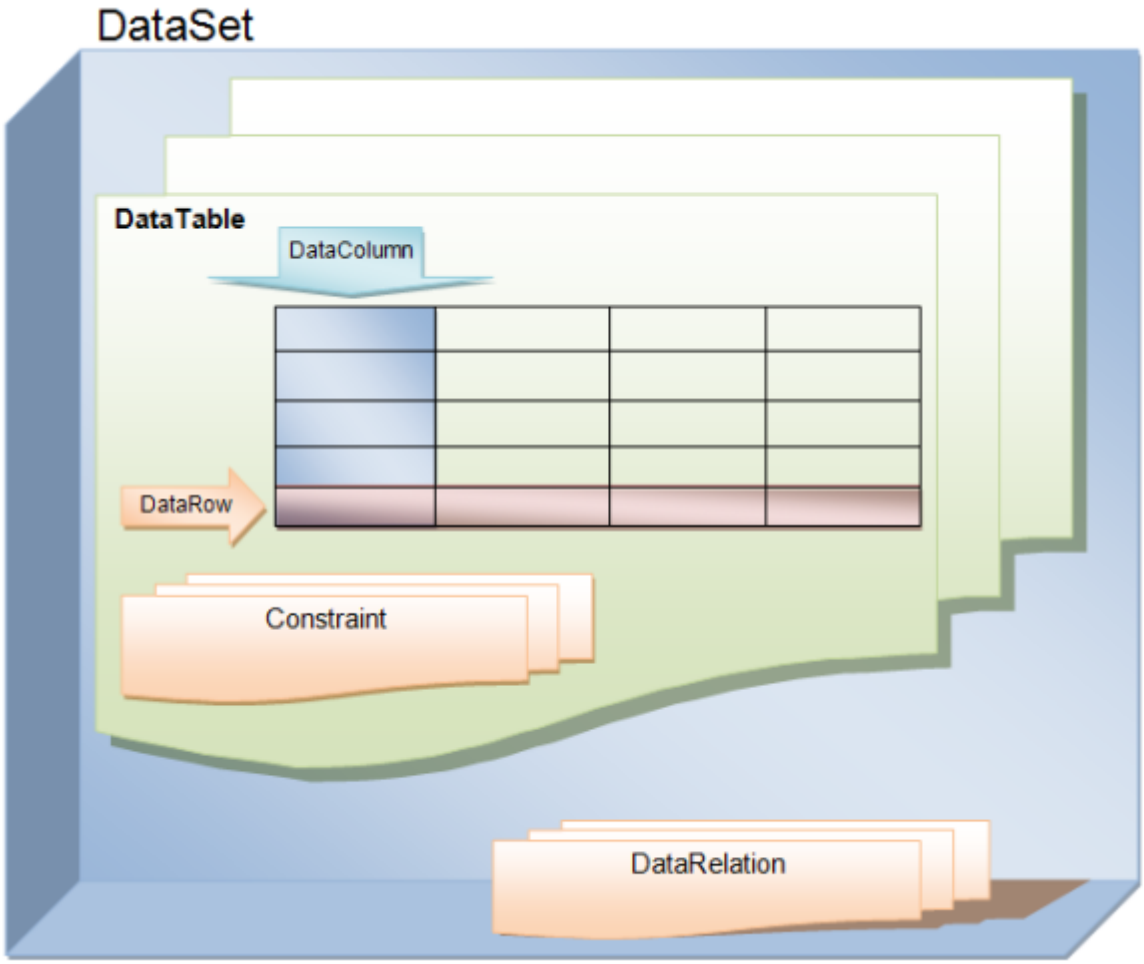
- Các bảng này được quản lý bởi tập hợp Tables của lớp DataSet

Ví dụ: Thêm một bảng vào DataSet và đặt tên bảng đó là bảng "SinhVien".

```
DataSet ds = new DataSet();
da = new SqlDataAdapter("select * from Sinhvien", con);
da.Fill(ds, "SinhVien");
```

2. Tương tác với DataSet

Một DataSet tương tự như một tập tin database vật lý hoàn chỉnh nhưng được lưu trong bộ nhớ. DataSet bao gồm các DataTable, DataTable bao gồm các DataColumn, DataRow, các constraint được minh họa như hình dưới:



Hình 3.1: Các property của DataSet và DataTable.

DataSet:

Type	Name	Description
DataTableCollection	Tables	Gets the collection of tables contained in the System.Data.DataSet
DataRelationCollection	Relations	Get the collection of relations that link tables and allow navigation from parent tables to child tables.

DataTable:

Type	Name	Description
DataColumnCollection	Columns	Gets the collection of columns that

		belong to this table.
DataRowCollection	Rows	Gets the collection of rows that belong to this table.
ConstraintCollection	Constraints	Gets the collection of constraints maintained by this table.

Nạp dữ liệu vào DataTable và DataSet

Để thực hiện các ví dụ với DataSet và DataTable, tôi đã chuẩn bị một database đơn giản trên SQL Server gồm hai bảng. Để thực hiện được bước này, yêu cầu bạn phải có sẵn kiến thức về kết nối database trong ADO.Net.

Ví dụ:

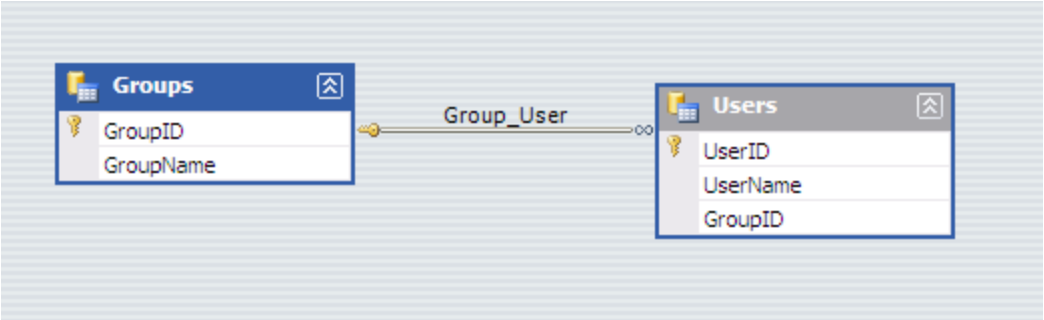


Table Groups:

GroupID	GroupName
1	Member
2	Moderator
3	Super Moderator
4	Admin

Table Users:

UserID	UserName	GroupID
1	Adon	1
2	Akuma	2
3	Balrog	1
4	Bison	1

5	Blanka	3
6	Cammy	1
7	ChunLi	1
8	Cody	4
9	Dan	1
10	DeeJay	1

Nạp dữ liệu vào DataSet từ database

Phương thức sau sẽ kết nối đến database SQL Server, sau đó nạp hai table User và Group vào DataSet. Khi làm ví dụ, bạn hãy sửa chuỗi kết nối theo máy bạn và tạo các table tương ứng.

```
private static DataSet LoadData()
{
    var conn = new SqlConnection(
        "Server=YINYANG\\SQLEXPRESS;Database=YinYangDB;Trusted_C
onnection=true");
    conn.Open();
    var cmd = "Select * from Users";
    var dataAdapter = new SqlDataAdapter(cmd, conn);
    var dataSet = new DataSet();
    dataAdapter.Fill(dataSet, "User");
    dataAdapter.SelectCommand.CommandText = "Select * from
Groups";
    dataAdapter.Fill(dataSet, "Group");
    conn.Close();
    return dataSet;
}
```

Phương thức Fill(DataSet) của DataAdapter tự động lấy tên bảng mà bạn đặt trong CommandText để đặt tên cho DataTable, tuy nhiên để chắc chắn tên bảng đúng trong trường hợp có sửa đổi, tôi dùng overload Fill(DataSet,string) để đặt tên lại cho các table.

Thay vì Fill() vào DataSet, bạn có thể dùng Fill(DataTable) để tạo ra một DataTable mới, sau đó thêm vào DataSet:

```
var table=new DataTable("Group");
dataAdapter.Fill(table);
dataSet.Tables.Add(table);
```

Một số tên table có thể trùng với từ khóa mà SQL sử dụng (ví dụ: User), mặc dù điều này ít khi xảy ra nhưng bạn có thể đề phòng trước bằng cách sử dụng quy tắc đặt tên như sử dụng tiền tố, đặt tên theo danh từ số nhiều,…

Ngoài cách nạp dữ liệu từ datatable, bạn cũng có thể tạo dữ liệu động cho DataTable thông qua các collection Columns và Rows. Phần sau sẽ giới thiệu về cách thực hiện này.

Tạo dữ liệu động cho DataTable

Kiểu dữ liệu DataColumn chứa đầy đủ các property cần thiết để bạn tạo ra một mô hình dữ liệu hoàn chỉnh cho DataTable. Ta có thể tạo một column dùng làm ID với chỉ số tự động tăng bắt đầu từ 1, không cho phép null và là duy nhất như sau:

```
DataColumn col = new DataColumn("ID", typeof(int));
col.AllowDBNull = false;
col.AutoIncrement = true;
col.AutoIncrementSeed = 1;
col.Unique = true;
```

Các DataColumn cần thiết phải có hai thông tin là tên và kiểu dữ liệu. Khi đã có một DataTable rỗng, công việc thêm các DataColumn vào rất đơn giản. Ví dụ sau tạo một DataTable với tên Persons với ba column là ID, Name và Birthday cùng với kiểu dữ liệu tương ứng là int, string và DateTime:

```
DataTable table = new DataTable("Persons");
DataColumn col = new DataColumn("ID", typeof(int));
col.AllowDBNull = false;
col.AutoIncrement = true;
col.AutoIncrementSeed = 1;
col.Unique = true;
table.Columns.Add(col);
table.Columns.Add("Name", typeof(string));
table.Columns.Add("Birthday", typeof(DateTime));
```

DataTable của chúng ta vẫn là rỗng vì chưa có dữ liệu (chỉ có mô hình dữ liệu). Để tạo một DataRow ta cần gọi phương thức DataTable.NewRow(). Phương thức này trả về một DataRow với các ô chứa dữ liệu tương ứng với các cột của

DataTable. Công việc thêm dữ liệu cũng rất đơn giản, dựa vào hai overload của DataRowCollection.Add() như bạn thấy dưới đây:

```
DataRow newRow = table.NewRow();
newRow["ID"] = 1; // remove this line
newRow["Name"] = "Boo";
newRow["Birthday"] = new DateTime(1990,3,4);
table.Rows.Add(newRow);
table.Rows.Add(null, "Bee", new DateTime(1989, 5, 3));
```

Một đoạn code nhỏ để in ra kết quả:

```
foreach (DataRow row in table.Rows)
{
    Console.WriteLine("ID={0}, Name={1}, Birthday={2}",
        row["ID"], row["Name"], row["Birthday"]);
}
```

Output:

ID=1, Name=Boo, Birthday=04/03/1990 12:00:00 AM

ID=2, Name=Bee, Birthday=03/05/1989 12:00:00 AM

Như vậy cột ID tự động tăng, ngay cả khi xóa bỏ dòng gán **newRow[“ID”] = 1**.

BÀI 4. BỘ ĐIỀU HỢP DỮ LIỆU DATAADAPTER

Mã Bài: MĐ21_04

Giới thiệu:

DataAdapter Có chức năng như một cầu nối giữa nguồn (tệp) dữ liệu và các bảng được cached trong bộ nhớ (đối tượng DataSet). DataAdapter điền dữ liệu vào một DataSet hay DataTable từ một nguồn dữ liệu sử dụng phương thức Fill(). Còn khi cập nhật dữ liệu ngược trở lại nguồn dữ liệu thì sử dụng phương thức Update() của đối tượng DataAdapter.

Một DataAdapter có thể được tạo từ một đối tượng connection đang mở hoặc từ một chuỗi kết nối (connection chưa được mở).

Mục tiêu:

- Nêu được đặc tính của các điều khiển hiển thị dữ liệu;
- Trình bày được quy trình thiết kế các dạng biểu mẫu;
- Sử dụng được các điều khiển cơ bản;
- Thiết kế các dạng biểu mẫu giao tác dữ liệu;
- Nghiêm túc, sáng tạo, chủ động trong việc thiết kế và kế thừa các dạng biểu mẫu khác nhau.

Nội dung chính:

1. Vai trò của DataAdapter

DataAdapter chính là cầu nối giữa Dataset và Datasource. Chúng ta có thể ví dụ như sau:

Có một cái bể nước (DataSource) , một cái máy bơm (DataAdapter) và một cái thùng đựng nước (DataSet). Thì khi lấy nước dùng cái bơm lấy nước từ bể, kiểm tra và lọc nước sau đó lại dùng cái bơm hút lại về cái bể nước. Đó chính là vai trò của cái bơm và DataAdapter tương tự như vậy.

2. Nạp dữ liệu từ cơ sở dữ liệu vào DataSet

Để nạp dữ liệu vào DataSet chúng ta có thể sử dụng phương thức Fill qua ví dụ sau:

```
string constring =
ConfigurationManager.ConnectionStrings["constr"].ConnectionString;
using (SqlConnection con = new SqlConnection(constring))
{
    using (SqlCommand cmd = new SqlCommand("SELECT Name, City FROM
Persons", con))
```

```
    {
        cmd.CommandType = CommandType.Text;
        using (SqlDataAdapter sda = new SqlDataAdapter(cmd))
        {
            DataSet ds = new DataSet();
            sda.Fill(ds);
            foreach (DataRow row in ds.Tables[0].Rows)
            {
                string name = row["Name"].ToString();
                string city = row["City"].ToString();
                Response.Write("Name: " + name);
                Response.Write("City: " + city);
            }
        }
    }
}
```

3. Cập nhật cơ sở dữ liệu

Đối tượng SqlDataAdapter phục vụ như cầu giữa các đối tượng ADO.NET dữ liệu và cơ sở dữ liệu SQL Server. SqlDataAdapter là một đối tượng trung gian populates với dữ liệu được lấy từ cơ sở dữ liệu SQL Server, thì bản Cập Nhật cơ sở dữ liệu để phản ánh các thay đổi (chẳng hạn như chèn, Cập Nhật và xóa) được thực hiện đối với dữ liệu bằng cách sử dụng các đối tượng dữ liệu ADO.NET dữ liệu đối tượng.

InsertCommand, UpdateCommandvà DeleteCommand thuộc tính của đối tượng SqlDataAdapter Cập Nhật cơ sở dữ liệu với các sửa đổi dữ liệu đang chạy trên một đối tượng dữ liệu . Các thuộc tính là các đối tượng SqlCommand chỉ định chèn, Cập Nhật và xoá Transact-SQL sau được sử dụng để gửi các sửa đổi dữ liệu cơ sở dữ liệu mục tiêu. SqlCommand các đối tượng được gán cho các thuộc tính có thể được tạo bằng tay trong mã hoặc tự động tạo ra bằng cách sử dụng các đối tượng SqlCommandBuilder.

Mẫu mã đầu tiên trong bài viết này giải thích cách sử dụng các đối tượng SqlCommandBuilder để tự động tạo ra thuộc tính UpdateCommand đối tượng SqlDataAdapter . Mẫu thứ hai sử dụng một tình huống mà bạn không thể sử dụng lệnh tự động tạo. Mẫu thứ hai chứng tỏ làm thế nào để tự tạo và sử dụng một đối tượng SqlCommand UpdateCommand thuộc tính của đối tượng SqlDataAdapter.

Tạo mẫu SQL Server

Tạo một mẫu SQL Server mà bạn có thể sử dụng trong mẫu mã Visual C# .NET được ghi lại trong bài viết này, hãy làm theo các bước sau:

Mở SQL Query Analyzer, và sau đó kết nối cơ sở dữ liệu mà bạn muốn tạo mẫu. Mã mẫu trong bài viết này sử dụng cơ sở dữ liệu Northwind trong Microsoft SQL Server.

Để tạo một mẫu có tên CustTest và chèn bản ghi vào bảng, chạy câu lệnh Transact-SQL sau:

```
Create Table CustTest(
 CustID int primary key,
 CustName varchar(20)
)
Insert into CustTest values(1,'John')
```

Mẫu mã 1: Lệnh được Tạo Tự động

Nếu lệnh chọn mà bạn sử dụng để truy xuất dữ liệu populates dữ liệu dựa trên một bảng cơ sở dữ liệu duy nhất, bạn có thể sử dụng các đối tượng CommandBuilder để tự động tạo ra DeleteCommand, InsertCommandvà thuộc tính UpdateCommand DataAdapter. Điều này đơn giản và giảm mã là cần thiết để thực hiện thao tác chèn, UDPATE và xoá.

Với yêu cầu tối thiểu, bạn phải đặt thuộc tính SelectCommand để tự động tạo ra lệnh. Sơ đồ bảng SelectCommand lấy xác định cú pháp chèn, Cập Nhật và báo cáo xóa tự động tạo ra.

Thuộc tính SelectCommand cũng phải trả một khoá chính hoặc cột duy nhất. Nếu không có ngoại lệ InvalidOperation được tạo ra và lệnh không được tạo ra.

Để tạo một ứng dụng bàn điều khiển Visual C# .NET mẫu trình bày cách sử dụng các đối tượng SqlCommandBuilder để tự động tạo ra DeleteCommand, InsertCommandvà UpdateCommand thuộc tính của đối tượng SqlCommand một đối tượng SqlDataAdapter , theo các bước sau:

Để tạo một ứng dụng bàn điều khiển Visual C# .NET, hãy làm theo các bước sau:

Tạo Class1 với mã sau đây:

```
using System.Data;using System.Data.SqlClient;
using System;
namespace Q308507 {
  class Class1 {
     static void Main(string[] args){
```

```
        SqlConnection cn = new SqlConnection();
        DataSet CustomersDataSet = new DataSet();
        SqlDataAdapter da;
        SqlCommandBuilder cmdBuilder;

        //Set the connection string of the SqlConnection object
to connect
        //to the SQL Server database in which you created the
sample
        //table.
        cn.ConnectionString =
"Server=server;Database=northwind;UID=login;PWD=password;";
        cn.Open();
        //Initialize the SqlDataAdapter object by specifying a
Select command
        //that retrieves data from the sample table.
        da = new SqlDataAdapter("select * from CustTest order by
CustId", cn);
        //Initialize the SqlCommandBuilder object to
automatically generate and initialize
        //the UpdateCommand, InsertCommand, and DeleteCommand
properties of the SqlDataAdapter.
        cmdBuilder = new SqlCommandBuilder(da);
        //Populate the DataSet by running the Fill method of the
SqlDataAdapter.
        da.Fill(CustomersDataSet, "Customers");
        //Display the Update, Insert, and Delete commands that
were automatically generated
        //by the SqlCommandBuilder object.
        Console.WriteLine("Update command Generated by the
Command Builder : ");
Console.WriteLine("==========================================
=====");
Console.WriteLine(cmdBuilder.GetUpdateCommand().CommandText);
        Console.WriteLine("         ");
        Console.WriteLine("Insert command Generated by the
```

```
Command Builder : ");
Console.WriteLine("==========================================
=====");

Console.WriteLine(cmdBuilder.GetInsertCommand().CommandText);
        Console.WriteLine("         ");
Console.WriteLine("Delete command Generated by the Command
Builder : ");
Console.WriteLine("==========================================
=====");
Console.WriteLine(cmdBuilder.GetDeleteCommand().CommandText);
Console.WriteLine("         ");
        //Write out the value in the CustName field before
updating the data using the DataSet.
        Console.WriteLine("Customer Name before Update : " +
CustomersDataSet.Tables["Customers"].Rows[0]["CustName"]);

        //Modify the value of the CustName field.
        CustomersDataSet.Tables["Customers"].Rows[0]["CustName"]
= "Jack";

        //Post the data modification to the database.
        da.Update(CustomersDataSet, "Customers");

        Console.WriteLine("Customer Name updated successfully");

        //Close the database connection.
        cn.Close();

        //Pause
        Console.ReadLine();
      }
   }

}
```

Sửa đổi chuỗi kết nối phù hợp cho môi trường của bạn.

Lưu và sau đó chạy ứng dụng. Thấy một cửa sổ bảng điều khiển mở và sau đó hiển thị kết quả sau đây:

```
Update command Generated by the Command Builder :
================================================
UPDATE CustTest SET CustID = @p1 , CustName = @p2 WHERE ( CustID
= @p3 AND CustName = @p4 )

Insert command Generated by the Command Builder :
================================================
INSERT INTO CustTest( CustID , CustName ) VALUES ( @p1 , @p2 )

Delete command Generated by the Command Builder :
================================================
DELETE FROM  CustTest WHERE ( CustID = @p1 AND CustName = @p2 )

Customer Name before Update : John
Customer Name updated successfully
```

Nhấn phím bất kỳ để bỏ qua cửa sổ bảng điều khiển và dừng ứng dụng.

Mã mẫu 2: Tự tạo và khởi tạo nhà UpdateCommand

Đầu ra mã mẫu 1 tạo cho biết rằng logic tạo lệnh tự động Cập Nhật báo cáo dựa trên lạc quan song song. Tức là, Hồ sơ không khoá để chỉnh sửa và người dùng hoặc quy trình khác có thể thay đổi bản ghi bất cứ khi nào.

Vì bản ghi có thể được thay đổi sau khi nó được trả về từ lệnh chọn trước khi bản Cập Nhật báo cáo được phát hành, lệnh Cập nhật tự động tạo chứa mệnh đề WHERE để liên tiếp được Cập Nhật chỉ nếu nó có chứa tất cả các giá trị ban đầu. Điều này là để tránh ghi đè dữ liệu mới. Nếu một báo cáo Cập nhật tự động tạo ra cố gắng Cập nhật hàng đã bị xóa hoặc không có giá trị ban đầu được tìm thấy trong dữ liệu, lệnh không ảnh hưởng đến bất kỳ hồ sơ, và một DBConcurrencyException ngoại lệ được tạo ra. Để kiểm tra điều này với mã mã mẫu 1, chạy mã trong trình gỡ lỗi Visual Studio, thiết lập một điểm dừng sau khi dữ liệu đã được điền nhưng trước khi cơ sở dữ liệu được Cập Nhật và sau đó xoá một hàng trong bảng từ SQL Query Analyzer. Cập Nhật gọi rồi ném một ngoại lệ.

Nếu bạn muốn báo cáo Cập Nhật hoàn tất bất kể giá trị ban đầu, bạn phải rõ ràng đặt UpdateCommand cho DataAdapter và dựa trên lệnh tự động tạo.

Để tự tạo và khởi tạo UpdateCommand thuộc tính của đối tượng SqlDataAdapter được sử dụng mã mẫu 1, hãy làm theo các bước sau:

Thay thế mã hiện có trong các chức năng chính của Class1 trong bảng điều khiển Visual C# .NET ứng dụng mà bạn đã tạo ở các mã mẫu 1: tự động tạo ra lệnh với mã sau đây:

```
SqlConnection cn = new SqlConnection();DataSet CustomersDataSet
= new DataSet();
SqlDataAdapter da;
SqlCommand DAUpdateCmd;

cn.ConnectionString =
"Server=server;Database=northwind;UID=login;PWD=password;";
cn.Open();

da = new SqlDataAdapter("select * from CustTest order by
CustId", cn);

//Initialize the SqlCommand object that will be used as the
UpdateCommand for the DataAdapter.

//Note that the WHERE clause uses only the CustId field to
locate the record to be updated.
DAUpdateCmd = new SqlCommand("Update CustTest set CustName =
@pCustName where CustId = @pCustId",
da.SelectCommand.Connection);

//Create and append the parameters for the Update command.
DAUpdateCmd.Parameters.Add(new SqlParameter("@pCustName",
SqlDbType.VarChar));
DAUpdateCmd.Parameters["@pCustName"].SourceVersion =
DataRowVersion.Current;
DAUpdateCmd.Parameters["@pCustName"].SourceColumn = "CustName";

DAUpdateCmd.Parameters.Add(new SqlParameter("@pCustId",
SqlDbType.Int));
DAUpdateCmd.Parameters["@pCustId"].SourceVersion =
```

```
DataRowVersion.Original;
DAUpdateCmd.Parameters["@pCustId"].SourceColumn = "CustId";

//Assign the SqlCommand to the UpdateCommand property of the
SqlDataAdapter.
da.UpdateCommand = DAUpdateCmd;
da.Fill(CustomersDataSet, "Customers");

Console.WriteLine("Customer Name before Update : " +
CustomersDataSet.Tables["Customers"].Rows[0]["CustName"]);
CustomersDataSet.Tables["Customers"].Rows[0]["CustName"] =
"Jack";
da.Update(CustomersDataSet, "Customers");

Console.WriteLine("Customer Name updated successfully");

cn.Close();
Console.ReadLine();
```

Sửa đổi chuỗi kết nối phù hợp cho môi trường của bạn.

Lặp lại bước 1 đến 4 trong phần mã mẫu 1: tự động tạo ra lệnh phần. Lưu ý rằng một ngoại lệ DBConcurrencyException không được tạo ra.

BÀI 5. SỬ DỤNG CÁC ĐIỀU KHIỂN RÀNG BUỘC DỮ LIỆU

Mã Bài: MĐ21_05

Giới thiệu:

Tiếp theo ta sẽ dùng các điều khiển quen thuộc như Textbox, Label, Button để trình bày cơ sở dữ liệu lên form. Để trình bày được như thế ta cần phải làm một thao tác gọi là ràng buộc dữ liệu (data binding), nghĩa là dữ liệu hiển thị lên trong các điều khiển sẽ phụ thuộc vào nguồn dữ liệu có trong DataSet hay DataAdapter.

Mục tiêu:

- Sử dụng VS 2010 để thiết kế được các giao diện, tạo lập được các ràng buộc từ các điều khiển (textbox, Listbox,…) với CSDL;
- Lập trình gắn kết được dữ liệu với các điều khiển trong ứng dụng;
- Thái độ tích cực, tỉ mỉ, sáng tạo. Chủ động tìm thêm nguồn bài tập thực hành.

Nội dung chính:

1. Tạo hộp danh sách ràng buộc

Để ràng buộn dữ liệu cho hộp danh sách (Combobox/ListBox) chúng ta sử dụng đoạn lệnh sau:

cb.DataSource = ds;

cb.DisplayMember = “Bảng.Cột1";

cb.ValueMember = "Bảng.Cột2";

cb.DataSource = ds.Tables[0];

cb.DisplayMember = “Cột1";

cb.ValueMember = "Cột2";

Lớp CurrencyManager giúp việc đồng bộ dữ liệu giữa các control

Các Property quan trong:

–Position

–Count

Phương thức

–Position++

–Position—

```
CurrencyManager cm = (CurrencyManager)this.BindingContext[ds,“Bảng"];
if (cm.Position < cm.Count - 1)
{
    cm.Position++;
}
```

Để đồng bộ dữ liệu giữa các control chúng ta sử dụng những chức năng cơ bản sau:

• Update

–cm.EndCurrentEdit();

adapter.Update(ds.Bảng);

• Next

–cm.Position++;

• Back

–cm.Position--;

• First

– cm.Position = 0;

• Last

– vt = this.BindingContext[ds, “Bảng"].Count - 1

– cm.Position = vt;

• Delete

– vt = this.BindingContext[ds, “Bảng"].Position

– cm.RemoveAt(vt);

• Add new

– cm.AddNew();

* Một số chú ý quan trọng:

- Data source của các control phải đồng nhất với nhau

- Hoặc cùng dataset

- Hoặc cùng datatable

- Tốt nhất: nên dùng datatable

2. Lọc dữ liệu trong hộp danh sách

Để minh họa chúng ta tạo một class Student.cs:

```
public class Student
{
 public string Name { get; set; }
 public int Age { get; set; }
 public string Address { get; set; }
 public string Branch { get; set; }
}
```

Tiếp theo tạo một hàm trong Form1.cs hàm khởi tạo dữ liệu cho DataGirdView và Combobox như sau:

```
public Form1()
{
 InitializeComponent();
CreateDataSource();
}
List studentList = new List () ;
```

Ta viết code cho hàm CreateDataSource():

```
void CreateDataSource()
{
//KHỏi tạo dữ liệu
Student std = new Student();

std.Name = "Phuong";
std.Branch = "IT";
std.Age = 24;
std.Address = "Ben Tre";

studentList.Add(std);
std = new Student();
std.Name = "Ngoc An";
std.Branch = "IT";
std.Age = 22;
std.Address = "Ben Tre";
studentList.Add(std);

std = new Student();
std.Name = "Thanh Tuyen";
std.Branch = "Ke Toan";
std.Age = 22;
std.Address = "Ben Tre";
studentList.Add(std);
```

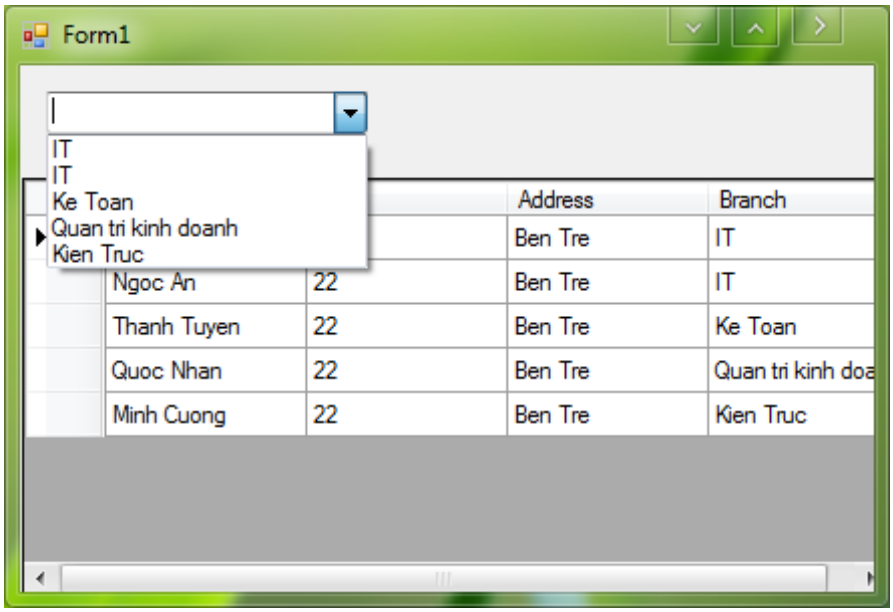
```
std = new Student();
std.Name = "Quoc Nhan";
std.Branch = "Quan tri kinh doanh";
std.Age = 22;
std.Address = "Ben Tre";
studentList.Add(std);

std = new Student();
std.Name = "Minh Cuong";
std.Branch = "Kien Truc";
std.Age = 22;
std.Address = "Ben Tre";
studentList.Add(std);
//Gán dữ liệu cho DataGirdView
this.dataGridView1.DataSource = studentList;
//Gán ngành cho combobox
var item = studentList.Select(a => a.Branch);
comboBox1.Items.AddRange(item.ToArray());
}
```

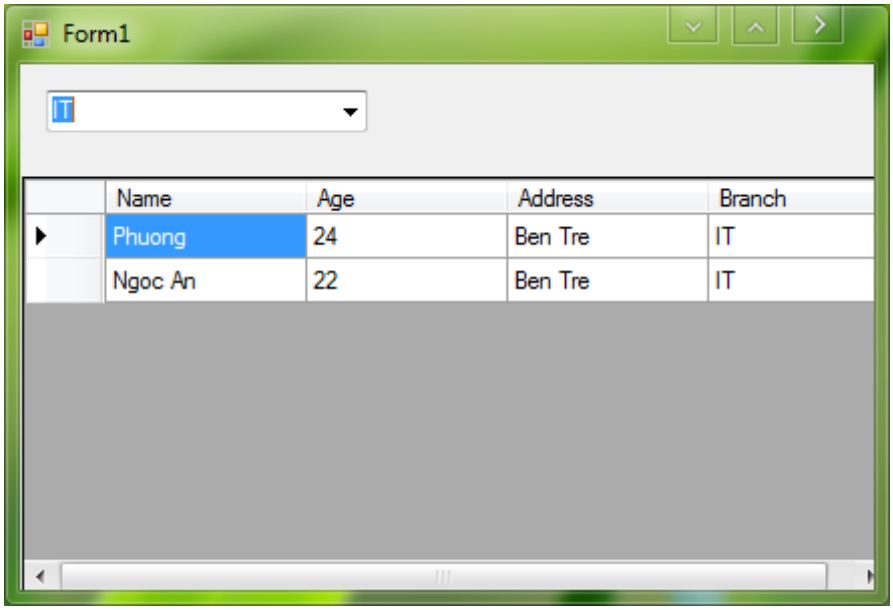
Ta viết code trong sự kiện của Combobox:

```
if (comboBox1.SelectedItem != null)
{
 string branch = comboBox1.Text;
 var students = studentList.Where(a => a.Branch.Equals(branch));
 dataGridView1.DataSource = students.ToList();
}
```

Ban đầu ta có như sau:



Kết quả ta lọc danh sách từ combobox:



3. Buộc dữ liệu vào hộp văn bản

Để buộc dữ liệu vào hộp văn bản chúng ta binding với thuộc tính Text của TextBox, ví dụ:

– txtTextBox.DataBindings.Add("Text",ds,“Bang.TênCột");

– txtTextBox.DataBindings.Add("Text",ds.Tables[0],“TênCột");

4. Sửa đổi và cập nhật dữ liệu từ hộp văn bản

Phiên bản trước của Microsoft ActiveX Data Objects (ADO) cho phép bạn Cập Nhật bản ghi trong nguồn dữ liệu khác với dữ liệu nguồn gốc của các hồ sơ, mặc dù khó khăn để thực hiện việc này. Vì ADO.NET là một mẫu thực sự bị ngắt kết nối do ADO.NET giới thiệu đối tượng DataAdapter , bạn có thể dễ dàng cập nhật nguồn dữ liệu với thông tin từ nguồn dữ liệu khác.

Bạn có thể sử dụng các thuộc tính InsertCommand, UpdateCommandvà DeleteCommand DataAdapter Cập Nhật nguồn dữ liệu với các thay đổi được thực hiện đối với các đối tượng dữ liệu . Các thuộc tính có các lệnh chèn, Cập Nhật và xoá SQL tương ứng. Bạn có thể sử dụng các lệnh này để gửi các thay đổi về nguồn dữ liệu đích. Bạn có thể sử dụng những phương pháp sau để tạo các lệnh:

Tạo thủ công các lệnh trong mã.

Sử dụng các đối tượng CommandBuilder để tự động tạo ra các lệnh.

Sử dụng thuật sỹ DataAdapter trực quan tạo lệnh.

Bài viết này bao gồm các mẫu cho các phương pháp đầu tiên.

Bài viết này mô tả bốn cách Cập Nhật bản ghi trong nguồn dữ liệu khác với dữ liệu nguồn gốc của các hồ sơ. Để chạy thành công các mẫu mã, trước tiên bạn phải tạo dự án Visual C#. Sau khi hoàn thành các bước trong phần tạo bảng tác giả mới vào cơ sở dữ liệu Northwind một Visual C# dự án mới , bạn có thể chuyển sang bất kỳ phần khác.

Tạo một bảng tác giả mới vào cơ sở dữ liệu Northwind một Visual C# dự án mới

Mỗi mẫu trong bài viết này yêu cầu dự án Visual C# và bảng trong CSDL Northwind mẫu mới.

Lưu ý Mã điền dữ liệu với các bản ghi từ bảng tác giả, bạn cũng có thể chèn dữ liệu trong nguồn dữ liệu khác nhau. Chèn dữ liệu trong nguồn dữ liệu khác nhau, bạn phải đặt thuộc tính AcceptChangesDuringFill sai. Khi bạn đặt AcceptChangesDuringFill giả, phương pháp AcceptChanges DataRow đối tượng không phải là khi một dòng được thêm vào bảng dữ liệu. Do đó, đối với mỗi hàng được thêm vào bảng dữ liệu, giá trị thuộc tính RowState là nhập thay vì không thay.

Do đó, khi dữ liệu được gửi đến phương pháp Cập Nhật DataAdpater, dữ liệu invokes lệnh chèn DataAdapter thay vì lệnh Cập Nhật cho mỗi đối tượng DataRow .

Để biết thêm thông tin, hãy tham khảo phần tham khảo.

Tạo dự án Visual C# và bảng mới trong cơ sở dữ liệu Northwind, hãy làm theo các bước sau:

Mở Query Analyzer, và sau đó chọn Northwind cơ sở dữ liệu. Chạy tập lệnh SQL sau để tạo bảng tác giả:

```
CREATE TABLE [dbo].[authors] ([au_id] [varchar] (11) primary key
NOT NULL ,
[au_lname] [varchar] (40) NOT NULL ,
[au_fname] [varchar] (20) NOT NULL ,
[phone] [char] (12) NULL ,
[address] [varchar] (40) NULL ,
[city] [varchar] (20) NULL ,
[state] [char] (2) NULL ,
[zip] [char] (5) NULL ,
[contract] [bit] NULL
) ON [PRIMARY]
GO
```

Bắt đầu Visual Studio .NET, và sau đó tạo một ứng dụng Windows mới trong Visual C# .NET.

Thêm mã sau ở đầu cửa sổ mã. Mã này cho phép bạn sử dụng các đối tượng của các tên không đầy đủ điều kiện đó.

```
      using System.Data.SqlClient;      using System.Data;
Trên khu vực lớp Form1, thêm báo cáo tuyên bố sau:
      SqlConnection PubConn;      SqlConnection NWindConn;
      SqlCommand PubCom;
      DataSet dsPub;
      SqlDataAdapter daPub;
```

Bấm đúp vào biểu mẫu để hiển thị sự kiện Form_Load trong cửa sổ mã. Thêm mã sau vào sự kiện này mở kết nối với tác giả bảng trong cơ sở dữ liệu quán và điền vào dữ liệu, dsPub, với các bản ghi từ bảng tác giả:

```
      PubConn = new SqlConnection
("server=server;uid=sa;pwd=password;initial catalog=pubs;");
      NWindConn = new SqlConnection
      ("server=server;uid=sa;pwd=password;initial
catalog=northwind;");
      PubCom = new SqlCommand ("Select * from authors",
PubConn);
      dsPub = new DataSet();
      daPub = new SqlDataAdapter();
      PubConn.Open();
```

```
      daPub.SelectCommand = PubCom;
      daPub.AcceptChangesDuringFill = false;
      daPub.Fill (dsPub, "Authors");
```

Thay đổi máy chủ, uidvà đối số mật khẩu trong mã sau để trỏ tới máy tính đang chạy Microsoft SQL Server:

```
      PubConn = new SqlConnection
("server=server;uid=sa;pwd=password;initial catalog=pubs;");
      NWindConn = new SqlConnection
      ("server=server;uid=sa;pwd=password;initial
catalog=northwind;");
```

Thêm hai điều khiển hộp (TextBox1 và TextBox2) Form1.

Thêm một nút kiểm soát để Form1. Thay đổi thuộc tính **tên** btnEdit, và sau đó thay đổi thuộc tính **văn bản** để chỉnh sửa.

Bấm đúp vào nút để hiển thị sự kiện btnEdit_Click trong cửa sổ mã. Thêm mã sau vào sự kiện này:

```
      Console.Write (dsPub.Tables["Authors"].Rows[1][1]);
Console.Write (dsPub.Tables["Authors"].Rows[1][2]);
      dsPub.Tables ["Authors"].Rows[1].BeginEdit();
      dsPub.Tables["Authors"].Rows[1][1] = textBox2.Text;
      dsPub.Tables["Authors"].Rows[1][2] = textBox1.Text;
      dsPub.Tables["Authors"].Rows[1].EndEdit();
      Console.Write (dsPub.Tables["Authors"].Rows[1][1]);
      Console.Write (dsPub.Tables["Authors"].Rows[1][2]);
```

Mã này ghi giá trị ban đầu của tác giả thứ hai đầu tiên và cuối cùng tên trong dữ liệu đầu ra cửa sổ. Mã Sửa đổi của tác giả tên trường và sau đó cuối cùng ghi giá trị mới vào cửa sổ kết quả xác nhận rằng họ đã được thay đổi trong dữ liệu. Lưu ý rằng mã Cập nhật thông tin về nguồn dữ liệu. Mã này chỉ chỉnh sửa đối tượng dữ liệu .

Cập Nhật nguồn dữ liệu khác với dữ liệu nguồn gốc

Phần này mô tả bốn cách khác nhau để Cập Nhật bản ghi trong nguồn dữ liệu khác với dữ liệu nguồn gốc của các hồ sơ.

Cập Nhật tất cả các bản ghi một bàn trống bằng cách sử dụng CommandBuilder

Phần này trình bày cách cập nhật nguồn dữ liệu thông tin từ nguồn dữ liệu khác. Trong mẫu này, bạn tạo một dữ liệu từ bảng tác giả quán cơ sở dữ liệu, chỉnh sửa bản ghi trong đó dữ liệuvà cam kết thay đổi hồ sơ và tất cả các bản ghi một bảng tác giả trống trong cơ sở dữ liệu Northwind. Bạn không cập nhật bất kỳ thay đổi về cơ sở dữ liệu quán gốc.

Mẫu này sử dụng hai đối tượng DataAdapter , một đối tượng dữ liệu và các đối tượng CommandBuilder . Bạn sử dụng đầu tiên DataAdapter để tạo một dữ liệu dựa trên tác giả bảng trong cơ sở dữ liệu quán. Bạn kết nối hai DataAdapter tác giả bảng trong CSDL Northwind, và bạn sử dụng này DataAdapter Cập Nhật. Cuối cùng, bạn sử dụng các đối tượng CommandBuilder tạo lệnh giao dịch SQL (TSQL) là cần thiết để cập nhật và chèn bản ghi vào nguồn dữ liệu.

Hoàn thành các bước trong phần tạo bảng tác giả mới vào cơ sở dữ liệu Northwind một Visual C# dự án thiết lập tác giả bảng trong CSDL Northwind và tạo dự án Visual C# .NET.

Đảm bảo rằng hồ sơ không tồn tại trong bảng cơ sở dữ liệu Northwind tác giả. Điều này có thể xảy ra khi bạn đặt DataAdapter.AcceptChangesDuringFill sai. Nếu hồ sơ tồn tại trong bảng tác giả, bạn nhận được vi phạm khoá chính khi bạn chạy mã này do bạn nhập giá trị trùng lặp.

Thêm một nút kiểm soát để Form1. Thay đổi thuộc tính **tên** btnUpdate1, và sau đó thay đổi thuộc tính **văn bản** Update1.

Bấm đúp **Update1** Hiển thị cửa sổ mã và sự kiện btnUpdate1_click . Thêm mã sau vào sự kiện này:

```
        SqlDataAdapter NwindDA = new SqlDataAdapter
("Select * from authors", NWindConn);
        SqlCommandBuilder x = new SqlCommandBuilder (NwindDA);
        NWindConn.Open();
        NwindDA.Update(dsPub, "Authors");
        Console.Write("Done");
```

Mã này tạo ra một đối tượng DataAdapter mới làm việc với bảng tác giả thứ hai. Đối tượng DataAdapter này sau đó sử dụng các đối tượng CommandBuilder tạo lệnh TSQL.

Chạy ứng dụng.

Nhập tên mới vào **TextBox1**, gõ tên mới mới **TextBox2**và sau đó nhấp vào **chỉnh sửa**. Điều này sửa hồ sơ trong dsPubdữ liệu. Điều này cũng hiển thị giá trị ban đầu của tên và tên trường cuối cùng trong cửa sổ xuất, cùng với các giá trị mới cho các trường tương tự.

Bấm **Update1**. Sau khi "Làm" xuất hiện trong cửa sổ xuất, dừng chạy ứng dụng.

Mở Bảng tác giả Northwind cơ sở dữ liệu. Thông báo rằng tác giả bảng có chứa tất cả các bản ghi từ bảng ban đầu. Ngoài ra, bản ghi thứ hai bao gồm các giá trị mà bạn đã nhập trong hộp văn bản.

Tham khảo bảng tác giả gốc trong cơ sở dữ liệu quán. Chú ý rằng các giá trị đối với hồ sơ thứ hai không thay đổi.

Cập Nhật tất cả các bản ghi một bàn trống bằng cách sử dụng lệnh TSQL

Mẫu này cho phép bạn di chuyển tất cả các bản ghi từ tác giả bảng trong cơ sở dữ liệu quán sang một bảng tác giả trống trong cơ sở dữ liệu Northwind mà không thay đổi bảng ban đầu. Để thực hiện các bản Cập Nhật, bạn tự tạo TSQL lệnh và sau đó sử dụng các lệnh TSQL đối tượng DataAdapter . Phương pháp này cung cấp các hoạt động tốt hơn với phương pháp đối tượng CommandBuilder.

Hoàn thành các bước trong phần tạo bảng tác giả mới vào cơ sở dữ liệu Northwind dự án mới Visual C# thiết lập tác giả bảng trong CSDL Northwind và tạo dự án Visual C# .NET.

Đảm bảo rằng hồ sơ không tồn tại trong bảng cơ sở dữ liệu Northwind tác giả. Điều này có thể xảy ra khi bạn đặt DataAdapter.AcceptChangesDuringFill sai. Nếu hồ sơ tồn tại trong bảng tác giả, bạn nhận được vi phạm khoá chính khi bạn chạy mã này do bạn nhập giá trị trùng lặp.

Ngay cả khi một dòng được chỉnh sửa, bạn chỉ cần một thuộc tính InsertCommand vì tất cả các hàng được đánh dấu là nhập do thiết đặt AcceptChangesDuringFill.

Thêm một nút kiểm soát để Form1. Thay đổi thuộc tính **tên** btnUpdate2, và sau đó thay đổi thuộc tính **văn bản** Update2.

Bấm đúp **Update2** Hiển thị cửa sổ mã và sự kiện btnUpdate2_click . Thêm mã sau vào sự kiện này:

```
      SqlDataAdapter NWindDA = new SqlDataAdapter       ("Select
* from Authors", NWindConn);
```

```
      NWindDA.UpdateCommand = new SqlCommand
      ("Update Authors Set au_lname=@lastname where au_id =
@au_id",
      NWindConn);
      SqlParameter workParam =
NWindDA.UpdateCommand.Parameters.Add
      ("@auID", SqlDbType.VarChar, 11);
      workParam.SourceColumn = "au_ID";
      workParam.SourceVersion = DataRowVersion.Original ;

      SqlParameter workParam4 =
NWindDA.UpdateCommand.Parameters.Add
      ("@lastname", SqlDbType.NVarChar,15);
      workParam4.SourceColumn = "au_lname";
      workParam4.SourceVersion = DataRowVersion.Current;

      NWindDA.InsertCommand = new SqlCommand
      ("Insert into authors (au_id, au_lname, au_fname) " +
      "Values (@au_id, @au_lname, @au_fname)", NWindConn);
      SqlParameter workParam1 =
NWindDA.InsertCommand.Parameters.Add
      ("@au_id", SqlDbType.VarChar,11);
      workParam1.SourceColumn = "au_ID";

      SqlParameter workParam2 =
NWindDA.InsertCommand.Parameters.Add
      ("@au_lname", SqlDbType.VarChar,40);
      workParam2.SourceColumn = "au_lname";

      SqlParameter workParam3 =
NWindDA.InsertCommand.Parameters.Add
      ("@au_fname", SqlDbType.VarChar,20);
      workParam3.SourceColumn = "au_fname";
      NWindConn.Open();
      NWindDA.Update (dsPub,"Authors");
      Console.Write ("Done");
```

Mã này tạo ra một đối tượng DataAdapter mới làm việc với bảng tác giả Northwind cơ sở dữ liệu. Này DataAdapter sau đó sử dụng lệnh TSQL bạn tạo bằng tay.

Chạy ứng dụng.

Gõ tên TextBox1, gõ tên cuối TextBox2và sau đó nhấp vào **chỉnh sửa**. Điều này sửa hồ sơ trong dsPubdữ liệu. Điều này cũng hiển thị giá trị ban đầu của tên và tên trường cuối cùng trong cửa sổ xuất, cùng với các giá trị mới cho các trường tương tự.

Bấm Update2.

Sau khi "Làm" xuất hiện trong cửa sổ xuất, mở Bảng tác giả Northwind cơ sở dữ liệu. Thông báo rằng tác giả bảng có chứa tất cả các bản ghi từ bảng tác giả ban đầu. Ngoài ra, bản ghi thứ hai bao gồm các giá trị mới.

Tham khảo bảng tác giả gốc trong cơ sở dữ liệu quán. Chú ý rằng các giá trị đối với hồ sơ thứ hai không được cập nhật do bạn gọi phương pháp Cập Nhật DataAdapter được cấu hình cơ sở dữ liệu Northwind.

Cập nhật hồ sơ chỉ thay đổi một bảng khác

Trong mẫu này, bạn gửi hồ sơ chỉ Cập Nhật cơ sở dữ liệu Northwind. Mẫu này Cập Nhật nguồn dữ liệu bằng cách sử dụng phương pháp GetChanges kéo hàng sửa đổi hoặc hàng trong gốc dữ liệu. Mẫu chèn dòng hoặc hàng vào tạm thời dữ liệu tên ChangedDS. Mẫu sau đó gọi phương pháp Cập Nhật của đối tượng DataAdapter qua này tạm thời dữ liệu cơ sở dữ liệu Northwind. Cuối cùng, mẫu gọi phương pháp AcceptChanges việc dữ liệu, dsPub, dsPub cho trạng thái phù hợp để phản ánh thay đổi nguồn dữ liệu.

Hoàn thành các bước trong phần tạo bảng tác giả mới vào cơ sở dữ liệu Northwind dự án mới Visual C# thiết lập tác giả bảng trong CSDL Northwind và tạo dự án Visual C# .NET.

Đảm bảo rằng tác giả bảng trong CSDL Northwind chứa ít hàng thứ hai từ tác giả bảng trong cơ sở dữ liệu quán. Đây là dòng mà bạn sẽ chỉnh sửa và sử dụng bản Cập Nhật.

Tác giả bảng trong CSDL Northwind ch hàng này, bạn có thể chạy mã từ phần Cập Nhật tất cả hồ sơ vào một bảng trống bằng cách sử dụng CommandBuilder thêm

tất cả các bản ghi vào bảng này. Bạn cũng có thể thêm các bản ghi theo cách thủ công.

```
Định vị dòng mã sau trong trường hợp Form_Load :
      'daPub.AcceptChangesDuringFill = False
Thảo luận trong dòng này để nó xuất hiện như sau:
//daPub.AcceptChangesDuringFill = False
```

Điều này chèn các hàng vào dữ liệu như hàng sẵn có.

Thêm một nút mới kiểm soát để Form1. Thay đổi thuộc tính **tên** btnUpdate3, và sau đó thay đổi thuộc tính **văn bản** Update3.

Bấm đúp vào **Update3** để hiển thị sự kiện btnUpdate3_click . Thêm mã sau vào sự kiện này:

```
       SqlDataAdapter NWindDA = new SqlDataAdapter
("Select * from Authors", NWindConn);

       NWindDA.UpdateCommand = new SqlCommand
       ("Update Authors Set au_lname = @lastname,
au_fname=@au_fname where
       au_ID=@auid", NWindConn);

       SqlParameter workParam =
NWindDA.UpdateCommand.Parameters.Add
       ("@auID", SqlDbType.VarChar, 11);
       workParam.SourceColumn = "au_ID";
       workParam.SourceVersion = DataRowVersion.Original;

       SqlParameter workParam4 =
NWindDA.UpdateCommand.Parameters.Add
       ("@lastname", SqlDbType.NVarChar, 15);
       workParam4.SourceColumn = "au_lname";
       workParam4.SourceVersion = DataRowVersion.Current;

       SqlParameter workParam1 =
NWindDA.UpdateCommand.Parameters.Add
       ("@au_fname", SqlDbType.VarChar, 11);
```

```
        workParam1.SourceColumn = "au_fname";
        workParam1.SourceVersion = DataRowVersion.Current;

        NWindConn.Open();
        DataSet changedDS;
        changedDS = dsPub.GetChanges(DataRowState.Modified);
        NWindDA.Update(changedDS, "Authors");
        Console.Write ("Done");
        dsPub.AcceptChanges();
```

Chạy ứng dụng.

Gõ tên **TextBox1**, gõ tên cuối **TextBox2**và sau đó nhấp vào **chỉnh sửa**. Điều này sửa hồ sơ trong dsPubdữ liệu. Điều này cũng hiển thị giá trị ban đầu của tên và tên trường cuối cùng trong cửa sổ xuất, cùng với các giá trị mới cho các trường tương tự.

Bấm **Update3**.

Sau khi "Làm" xuất hiện trong cửa sổ xuất, mở Bảng tác giả Northwind cơ sở dữ liệu. Thông báo bảng tác giả bao gồm các thay đổi bạn thực hiện DataRow.

Tham khảo bảng tác giả gốc trong cơ sở dữ liệu quán. Chú ý rằng các giá trị đối với hồ sơ thứ hai không được Cập Nhật.

Cập nhật hồ sơ chỉ thay đổi cả bảng

Để cập nhật các bản ghi trong cả hai nguồn dữ liệu, bạn có thể sử dụng hai DataAdapter đối tượng và phương pháp GetChanges đối tượng dữ liệu . Trong mẫu này, bạn tạo một đối tượng dữ liệu dựa trên tác giả bảng trong cơ sở dữ liệu quán. Bạn chỉnh sửa dữ liệu này và sau đó sao chép hồ sơ thay đổi thành một đối tượng dữ liệu mới. Bạn sử dụng này mới dữ liệu để cập nhật các bảng dữ liệu nguồn. Sau khi cập nhật bảng nguồn dữ liệu đầu tiên, bạn gọi phương pháp GetChanges lại và tạo lại tạm thời dữ liệu. Bạn có thể cập nhật bảng nguồn dữ liệu thứ hai.

Trong mẫu này, bạn sử dụng tạm thời dữ liệu và gọi GetChanges hai lần vì phương pháp Cập Nhật DataAdapter bao gọi AcceptChanges dữ liệu mà bạn vượt qua nó. Phương pháp AcceptChanges thực resynchronizes ban đầu và giá trị hiện tại cho mỗi ô được thay đổi. Ngoài ra, phương pháp thực AcceptChanges cờ DataRow như UnModified. Điều này loại bỏ các thay đổi và ngăn cản bạn gửi các thay đổi nguồn dữ liệu khác.

Trong trường hợp đặc biệt này, bạn có thể bỏ qua phương pháp GetChanges thứ hai và chuyển ban đầu dữ liệu để Cập Nhật phương pháp quán DataAdapter. Điều này giúp loại bỏ các cuộc gọi dsPubs.AcceptChanges . Tuy nhiên, hợp tạm thời dữ liệu và phương pháp GetChanges cho phép bạn quay lại các thay đổi ban đầu. Trong phiên bản trước của Microsoft ActiveX Data Objects (ADO), bạn vẫn còn tập bản ghi vào tệp để đạt được chức năng này.

Hoàn thành các bước trong phần tạo bảng tác giả mới vào cơ sở dữ liệu Northwind dự án mới Visual C# thiết lập tác giả bảng trong CSDL Northwind và tạo dự án Visual C# .NET.

Đảm bảo rằng cơ sở dữ liệu Northwind chứa một bảng tác giả và đảm bảo rằng bảng này tác giả ít chứa hồ sơ mà bạn sẽ thay đổi từ bảng tác giả quán cơ sở dữ liệu.

```
Định vị dòng mã sau trong trường hợp Form_Load :
      'daPub.AcceptChangesDuringFill = False
Thảo luận trong dòng này để nó xuất hiện như sau:
//daPub.AcceptChangesDuringFill = False
```

Thêm một nút kiểm soát để Form1. Thay đổi thuộc tính **tên** btnUpdate4, và sau đó thay đổi thuộc tính **văn bản** Update4.

Bấm đúp vào **Update4** để hiển thị sự kiện btnUpdate4_click . Thêm mã sau vào sự kiện này:

```
       SqlDataAdapter NWindDA = new SqlDataAdapter
("Select * from Authors", NWindConn);

       NWindDA.UpdateCommand = new SqlCommand
       ("UPDATE authors SET au_lname = @lastName, au_fname =
@au_fname" +
       " WHERE au_ID = @auID", NWindConn);

       SqlParameter workParam =
NWindDA.UpdateCommand.Parameters.Add
       ("@auID", SqlDbType.VarChar, 11);
       workParam.SourceColumn = "au_ID";
       workParam.SourceVersion = DataRowVersion.Original;

       SqlParameter workparam4 =
```

```
NWindDA.UpdateCommand.Parameters.Add
       ("@lastName", SqlDbType.NVarChar, 15);
       workparam4.SourceColumn = "au_lname";
       workparam4.SourceVersion = DataRowVersion.Current;

       SqlParameter workParm3 =
NWindDA.UpdateCommand.Parameters.Add
       ("@au_fname", SqlDbType.VarChar, 20);
       workParm3.SourceColumn = "au_fname";
       workParm3.SourceVersion = DataRowVersion.Current;

       NWindConn.Open();

       daPub.UpdateCommand = new SqlCommand
       ("UPDATE authors SET au_lname = @lastName, au_fname =
@au_fname" +
       " WHERE au_ID = @auID", PubConn);

       SqlParameter workParm7 =
daPub.UpdateCommand.Parameters.Add
       ("@auID", SqlDbType.VarChar, 11);
       workParm7.SourceColumn = "au_ID";
       workParm7.SourceVersion = DataRowVersion.Original;

       SqlParameter workparm8 =
       daPub.UpdateCommand.Parameters.Add
       ("@lastName", SqlDbType.NVarChar, 15);
       workparm8.SourceColumn = "au_lname";
       workparm8.SourceVersion = DataRowVersion.Current;

       SqlParameter workParm11 =
       daPub.UpdateCommand.Parameters.Add
       ("@au_fname", SqlDbType.VarChar, 20);
       workParm11.SourceColumn = "au_fname";
       workParm11.SourceVersion = DataRowVersion.Current;
```

```
        DataSet changedDS = new DataSet();
        changedDS = dsPub.GetChanges(DataRowState.Modified);

        NWindDA.Update(changedDS, "Authors");
        changedDS = dsPub.GetChanges(DataRowState.Modified);
        daPub.Update(changedDS, "Authors");
        dsPub.AcceptChanges();
        Console.Write("Done");
```

Chạy ứng dụng.

Nhập tên mới vào TextBox1, gõ tên mới mới TextBox2và sau đó nhấp vào **chỉnh sửa**. Điều này sửa hồ sơ trong dsPubdữ liệu. Điều này cũng hiển thị giá trị ban đầu của tên và tên trường cuối cùng trong cửa sổ xuất, cùng với các giá trị mới cho các trường tương tự.

Bấm **Update4**.

Sau khi "Làm" xuất hiện trong cửa sổ xuất, mở Bảng tác giả trong cơ sở dữ liệu quán và bảng tác giả Northwind cơ sở dữ liệu. Lưu ý rằng cả bảng bao gồm các giá trị mới cho hồ sơ thứ hai.

Khắc phục sự cố

Khi bạn làm việc với mã trong bản cập nhật hồ sơ chỉ thay đổi một bảng khác nhau và Cập nhật hồ sơ chỉ thay đổi cả bảng phần, bạn có thể nhận được lỗi System.ArgumentNullException. Nếu điều này xảy ra, định vị dòng mã sau:

```
      'daPub.AcceptChangesDuringFill = False
Đảm bảo rằng dòng mã được nhận xét ra và xuất hiện như sau:
\\daPub.AcceptChangesDuringFill = False
```

Dòng mã cho DataAdapter gọi phương pháp AcceptChanges khi được gọi là phương pháp điền DataAdapter .

5. Thêm mới và xóa bản ghi dùng các điều khiển ràng buộc dữ liệu

5.1 Thêm dữ liệu

```
using System;
using System.Data;
using System.Data.SqlClient;
class Class1{
public static void Main() {
```

```
SqlConnection thisConnection = new
SqlConnection("server=(local)\\SQLEXPRESS;database=MyDatabase;Integrated
Security=SSPI");
SqlDataAdapter thisAdapter = new SqlDataAdapter(
"SELECT ID, FirstName FROM Employee", thisConnection);
SqlCommandBuilder thisBuilder = new SqlCommandBuilder(thisAdapter);
DataSet thisDataSet = new DataSet();
thisAdapter.Fill(thisDataSet, "Employee");
Console.WriteLine("# rows before change: {0}",thisDataSet.Tables["Employee"].Rows.Count);
DataRow thisRow = thisDataSet.Tables["Employee"].NewRow();
thisRow["ID"] = "123";
thisRow["FirstName"] = "Ltd";
thisDataSet.Tables["Employee"].Rows.Add(thisRow);
Console.WriteLine("# rows after change: {0}", thisDataSet.Tables["Employee"].Rows.Count);
thisAdapter.Update(thisDataSet, "Employee");
}
}
```

5.2 Xóa dữ liệu

```
using System;
using System.Data;
using System.Data.SqlClient;
class PropagateDeletes {
static void Main() {
string connString = "server=(local)\\SQLEXPRESS;database=MyDatabase;Integrated
Security=SSPI";
string qry = @"select * from employee ";

string del = @"delete from employee where id = @id";
SqlConnection conn = new SqlConnection(connString);
try {
SqlDataAdapter da = new SqlDataAdapter();
da.SelectCommand = new SqlCommand(qry, conn);
DataSet ds = new DataSet();
da.Fill(ds, "employee");
DataTable dt = ds.Tables["employee"];
SqlCommand cmd = new SqlCommand(del, conn);
```

```
cmd.Parameters.Add("@id",SqlDbType.Int, 4, "id");
string filt = @"firstname = 'o' and lastname = 'B'";
foreach (DataRow row in dt.Select(filt)) {
row.Delete();
}
da.DeleteCommand = cmd;
da.Update(ds, "employee");
foreach (DataRow row in dt.Rows) {
Console.WriteLine(
"{0} {1}",
row["firstname"].ToString().PadRight(15),
row["lastname"].ToString().PadLeft(25));
}
} catch(Exception e) {
Console.WriteLine("Error: " + e);
} finally {
conn.Close();
}
}
}
```

Để sửa dữ liệu chúng ta có thể sử dụng đoạn lệnh sau:

```
using System;
using System.Data;
using System.Data.SqlClient;
class Class1{
static void Main(string[] args){
SqlConnection thisConnection = new
SqlConnection("server=(local)\\SQLEXPRESS;database=MyDatabase;Integrated
Security=SSPI");
SqlDataAdapter thisAdapter = new SqlDataAdapter("SELECT ID, FirstName FROM
Employee", thisConnection);
SqlCommandBuilder thisBuilder = new SqlCommandBuilder(thisAdapter);
DataSet thisDataSet = new DataSet();
thisAdapter.Fill(thisDataSet, "Employee");
Console.WriteLine("name before change: {0}",
thisDataSet.Tables["Employee"].Rows[9]["FirstName"]);
```

```
thisDataSet.Tables["Employee"].Rows[1]["FirstName"] = "Inc";
thisAdapter.Update(thisDataSet, "Employee");
Console.WriteLine("name after change: {0}",
thisDataSet.Tables["Employee"].Rows[9]["FirstName"]);
}
}
```

6. Quản lý lỗi khi tương tác dữ liệu

6.1 Kiểm tra sự trùng khóa khi thêm mới dữ liệu

Để kiểm tra bản ghi có tồn tại trong cơ sở dữ liệu hay chưa, chúng ta nên viết một hàm tìm kiếm bản ghi trong cơ sở dữ liệu hiện tại dựa vào khóa chính, ví dụ:

```
private bool kiemtratontai()
        {
            bool tatkt = false;
            string maso = txt_madg.Text;
            SqlConnection con = new SqlConnection (Configuration
Manager.ConnectionStrings[1].ConnectionString);
            SqlCommand cmd = new SqlCommand("Select * from Docgi
a", con);
            con.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                if (maso == dr.GetString(0))
                {
                    tatkt = true;
                    break;
                }
            }
            con.Close();
            return tatkt;
        }
```

Sau đó, trước khi thêm mới chúng ta sử dụng đoạn lệnh sau để kiểm tra:

```
if (kiemtratontai())
            {
                MessageBox.Show("Độc giả đã tồn tại !");
```

```
            }
            else
            {
             // Thực hiện khối lệnh Insert dữ liệu ở đây .
            }
```

6.2 Kiểm tra sự hợp lệ của các kiểm dữ liệu

Để kiểm tra sự hợp lệ của các kiểm dữ liệu chúng ta có thể sử dụng phương thức TryParse, ví dụ để kiểm tra n có phải là số nguyên hay không chúng ta viết lệnh như sau:

```
int n = 0;
if (int.TryParse(this.textBox6.Text, out n))
{
MessageBox.Show("dung la so nguyen");
}
else MessageBox.Show("khong phai so nguyen");
```

Các kiểu dữ liệu khác (double,float,DateTime,long….vv)cũng check in tương tự.

Ngoài ra chúng ta có thể viết hàm để kiểm tra, ví dụ:

```
//Hàm kiểm tra chuỗi có phải là số hay không
public bool IsNumber(string pText)
{
Regex regex = new Regex(@"^[-+]?[0-9]*\.?[0-9]+$");
return regex.IsMatch(pText);
}
//Sử  dụng hàm
if (IsNumber(tên chuỗi cần kiểm tra) == true)
{
// chuỗi là số
}
```

BÀI 6. TẠO BÁO CÁO VỚI CRYSTAL REPORT

Mã Bài: MĐ21_06

Giới thiệu:

Crystal Report là phần mềm thiết kế báo biểu chuyên nghiệp được tích hợp trong các phiên bản Visual Studio .NET của Microsoft.

Mục tiêu:

- Cài đặt được các công cụ báo cáo của VS .Net 2010;
- Tạo và xem báo cáo với công cụ Crystal Report của VS .Net 2010;
- Tạo và chỉnh sửa được một số báo cáo;
- Thái độ tích cực, tỉ mỉ, sáng tạo.

Nội dung chính:

1. Tạo báo cáo dùng Report Expert

Một điểm đặc biệt trong Visual Studio .NET là có thể tạo nguồn dữ liệu cho báo biểu từ một DataSet (mô hình Push). Sử dụng DataSet làm nguồn dữ liệu của báo biểu cho phép tạo ra những báo biểu không cần kết nối với CSDL hay thậm chí tạo ra những báo biểu trong các ứng dụng hoạt động không cần có CSDL đi cùng. Chẳng hạn, có thể lấy cấu trúc DataSet từ một file XML làm nguồn dữ liệu cho báo biểu.

Để gán nguồn dữ liệu cho báo biểu, chúng ta cần một biến đối tượng là một thể hiện của báo biểu đã thiết kế. Khi đã có một DataSet, chúng ta sử dụng phương thức SetDataSource của đối tượng báo biểu để gán DataSet đó làm nguồn dữ liệu:

Ví dụ:

```
CrystalReport1 Rpt = new CrystalReport1();
Rpt.SetDataSource(ds);
```

Ví dụ dưới đây minh họa cách thiết kế báo biểu sử dụng DataSet làm nguồn dữ liệu.

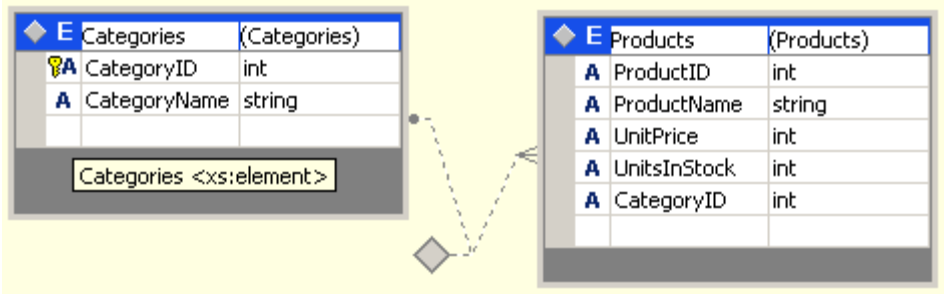
+ Trong Project, tạo một DataSet mới.

Ví dụ:

```
SqlDataAdapter bo_doc_ghi = new SqlDataAdapter ("Select CategoryID, CategoryName
From Categories;Select ProductID, ProductName,
UnitPrice, UnitsInStock, CategoryID From Products",
        "Server=MyServer;Database=Northwind; uid=sa;pwd=123456;");
bo_doc_ghi.TableMappings.Add ("Table", "Categories");
bo_doc_ghi.TableMappings.Add ("Table1", "Products");
Products_Schema DS = new Products_Schema();
CrystalReport1 rpt = new CrystalReport1();
Form f = new Form();
CrystalReportViewer cv = new CrystalReportViewer();
f.Controls.Add(cv);
cv.Dock = DockStyle.Fill;
rpt.SetDataSource(DS);
rpt.RecordSelectionFormula = "{Categories.CategoryID} > 4";
cv.ReportSource = rpt;
f.ShowDialog();
```

Vào Menu Project | Add new item, chọn mục DataSet, trong phần Name, gõ vào tên của DataSet là: Products_Schema.xsd

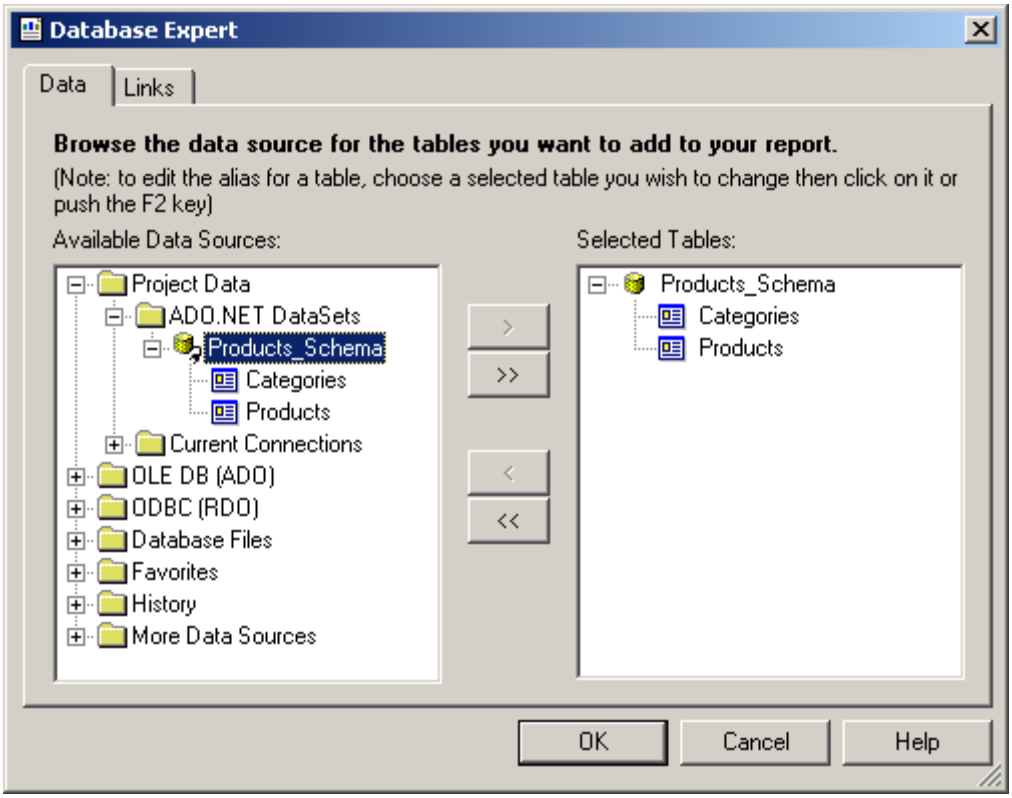
- Thiết kế DataSet như sau:



Hình 8.5: Tạo DataSet

- Lưu DataSet lại sau khi tạo xong cấu trúc bảng
- Tạo mới một báo biểu, khi chọn nguồn dữ liệu cho báo biểu chúng ta chọn mục ADO.NET DataSet.

- Chọn DataSet vừa tạo.
- Chọn 2 bảng Categories và Products làm nguồn dữ liệu.
- Chọn OK để lưu lại nguồn dữ liệu của báo biểu

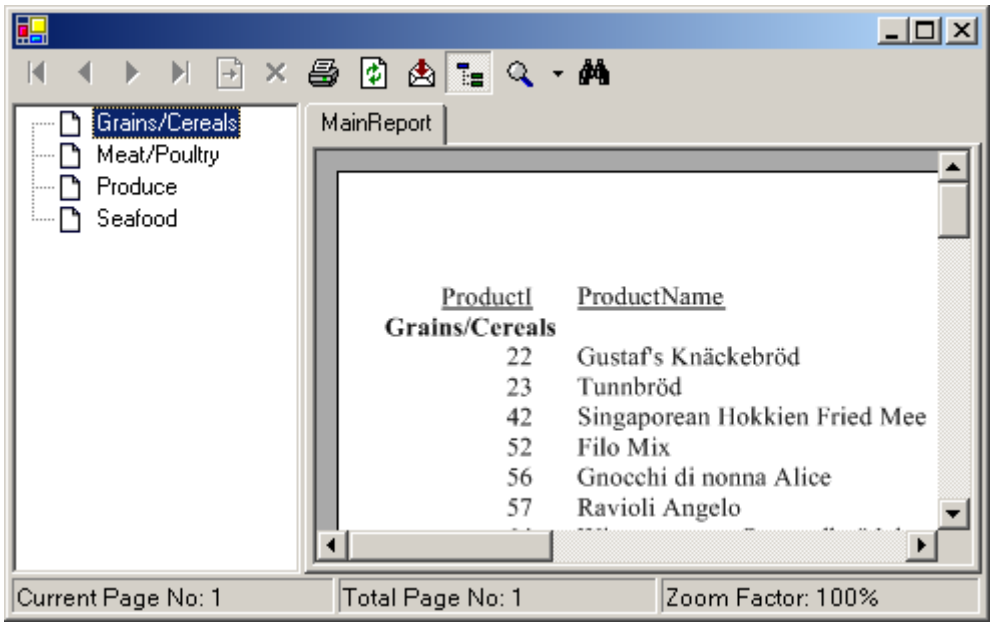


Hình 8.6: Chọn nguồn dữ liệu cho CrystalReport

- Thiết kế báo biểu tương tự như của ví dụ trước
- Trong đoạn lệnh hiển thị dữ liệu, sửa lại nội dung như sau:

Trong đoạn lệnh trên, chúng ta đã lọc dữ liệu: CategoryID > 4

Khi chạy chương trình, báo cáo sẽ được hiển thị như sau:



Hình 8.7: Lọc dữ liệu trong CrystalReport

Trên hình, dữ liệu đã được lọc lại so với báo cáo trước để chỉ hiển thị những Category có CategoryID >4.

2. Hiển thị các báo cáo đã tạo

Trong mô hình Pull đã đề cập ở trên, khi khai báo nguồn dữ liệu trong thiết kế, chúng ta phải chỉ rõ nguồn dữ liệu (thông tin đăng nhập trong SQL Server). Vì thế, lúc thực thi, cần định lại thông tin nguồn CSDL.

Trước hết cần làm quen với một thuộc tính của lớp Crystal Report trong CrystalDecisions.CrystalReports.Engine:

Đối tượng	Mô tả
Database	Nguồn CSDL sử dụng cho báo biểu.
Table	Thông qua chỉ số, tên trả về bảng tương ứng được sử dụng trong báo biểu.
Tables	Tập hợp các bảng sử dụng trên báo biểu.

Báo biểu có thành phần Database cho biết nguồn dữ liệu hiển thị. Khi vị trí nguồn thay đổi, chúng ta phải định lại vị trí nguồn mới cho báo biểu thông qua thông tin đăng nhập của Table, thuộc tập hợp Tables của Database.

Thông tin đăng nhập thuộc lớp TableLogOnInfo trong không gian tên CrystalDecisions.Shared.TableLogOnInfo với các thông tin:

Các thuộc tính của TableLogOnInfo

Thuộc tính	Mô tả
ConnectionInfo	Đối tượng chứa thông tin kết nối của bảng.
ReportName	Tên báo biểu.
TableName	Tên bảng.

Các thuộc tính của ConnectionInfo

Thuộc tính	Mô tả
DatabaseName	Tên tập tin CSDL.
Password	Mật khẩu truy cập nguồn CSDL
ServerName	Tên server hoặc nguồn dữ liệu ODBC.
UserID	Tên người dùng để truy cập CSDL.

Thông tin TableLogOnInfo của bảng có tính chỉ đọc và mặc định lưu giữ thông tin kết nối lúc chúng ta khai báo nguồn để thiết kế báo biểu. Khi thay đổi và muốn cập nhật lại, chúng ta phải sử dụng phương thức ApplyLogOnInfo của đối tượng Table với tham

số là TableLogOnInfo chúng ta đã thay đổi như ví dụ sau cho nguồn dữ liệu là MS Access:

Ví dụ:

```
Form f = new Form();
CreportVD rpt = new CreportVD();
CrystalReportViewer cv = new CrystalReportViewer();
f.Controls.Add(cv);
cv.Dock = DockStyle.Fill;
CrystalDecisions.Shared.TableLogOnInfo logOnInfo =
CrystalDecisions.Shared.TableLogOnInfo();
// Nếu các Table cùng từ một nguồn dữ liệu, chỉ cần định lại cho một bảng là đủ
logOnInfo = rpt.Database.Tables[0].LogOnInfo;
logOnInfo.ConnectionInfo.ServerName = <đường dẫn đến .mdb>;
rpt.Database.Tables[0].ApplyLogOnInfo(logOnInfo);
cv.ReportSource = rpt;
f.ShowDialog();
```

Nếu nguồn dữ liệu là SQL Server:

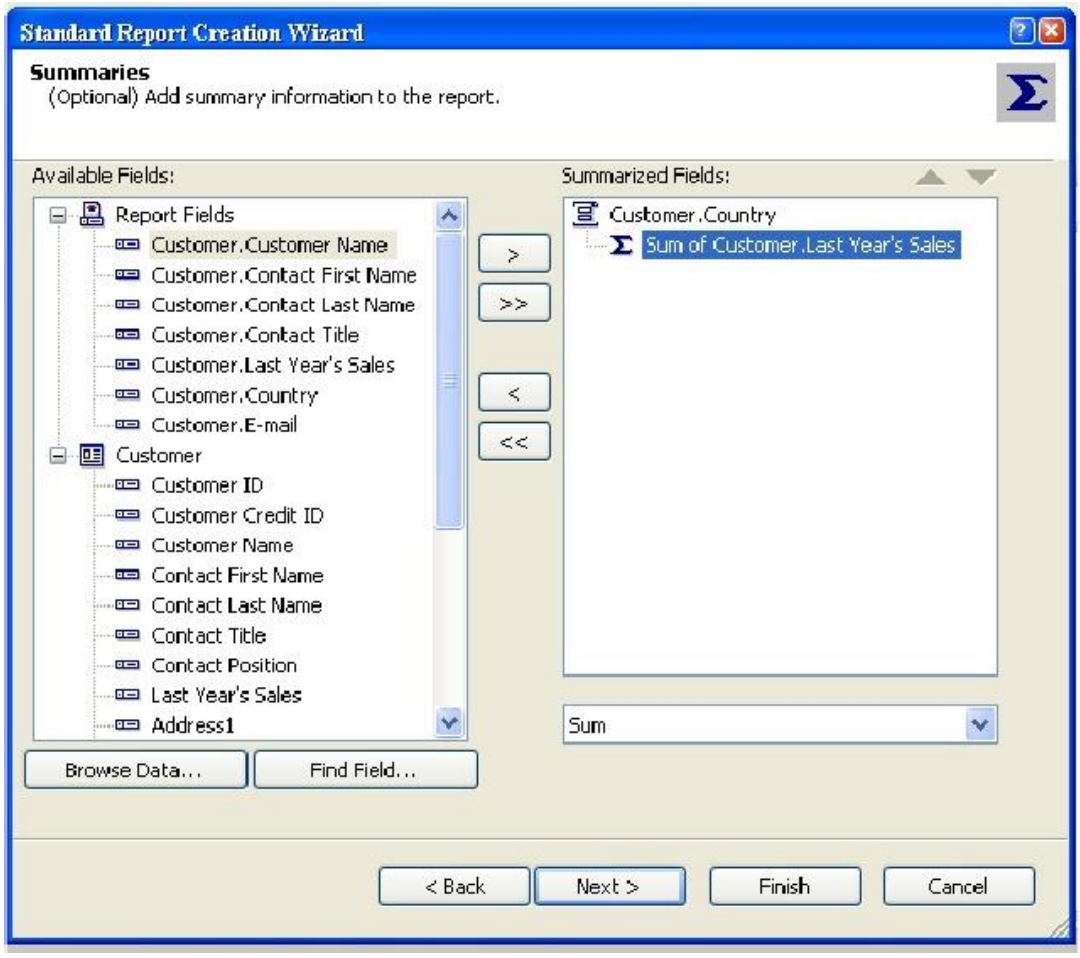
Ví dụ:

```
Form f = new Form();
// CReportVD là báo biểu đã tạo lúc thiết kế
CreportVD rpt = new CreportVD();
CrystalDecisions.Windows.Forms.CrystalReportViewer cv = new
CrystalDecisions.Windows.Forms.CrystalReportViewer();
f.Controls.Add(cv);
cv.Dock = DockStyle.Fill;
CrystalDecisions.Shared.TableLogOnInfo logOnInfo =
CrystalDecisions.Shared.TableLogOnInfo();
// Nếu các Table cùng từ một nguồn dữ liệu, chỉ cần định lại cho một bảng là đủ
logOnInfo = rpt.Database.Tables[0].LogOnInfo ;
logOnInfo.ConnectionInfo.ServerName = <Tên server>;
logOnInfo.ConnectionInfo.DatabaseName = <Tên CSDL>;
logOnInfo.ConnectionInfo.UserID = <Người dùng>;
logOnInfo.ConnectionInfo.Password = <Mật khẩu>;
rpt.Database.Tables[0].ApplyLogOnInfo(logOnInfo);
cv.ReportSource = rpt;
f.ShowDialog();
```

3. Thêm cột tính toán

Bạn click lên Tab Total trên hộp thoại Standard Report Export như hình dưới để thêm thông tin tổng kết (summary) vào báo cáo. Các field tổng kết (summaryfield) sẽ chứa kết quả của phép tính toán phổ biến nhất như sum, average ...Muốn thêm một summary field, bạn cần chọn ra một field trên khung **Available Fields** đưa vào khung **Summaried Fields** (trong ví dụ này làCustomer.Last Year's Sales), sau đó chọn ra kiểu summary từ hộp combo box **Summary Type** như trên hình dưới.Trong trường hợp này chúng ta chọn sum và check vào ô check box **Add Grand Totals**. Có tất cả 20 tác tử summary khá phổ biến mà bạn có thể sử dụng. Tùy theo kiểu dữ liệu của Field bạn chọn mà có thể có hoặc không một vài tác tử summary, ví dụ như bạn không thể tính trung bình (Average) trên một Field kiểu string. Với ví dụ này đến đây là chúng ta có thể nhấn nút **Finish** để chuyển sang **Report Designer**, nhưng để giúp các bạn hiểu rõ hơn các bước còn lại nên chúng ta sẽ tiếp tục khảo sát tất cả các bước còn lại. Bạn nhấn **Next** để qua bước tiếptheo.

Lưu ý: Bạn không thể có bước này nếu báo cáo của bạn không có ít nhất một Group:



Hình 6.1: Hộp thoại Standard Report Expert – Tab Total

4. Chọn bản ghi hiển thị

Để nạp và hiển thị Report trong chương trình chúng ta sẽ tiến hành các bước như sau:

B1. Thêm vào trong dự án một tệp DataSet.

B2. Thêm vào các bảng có cấu trúc và tên trường giống như cấu trúc của các dữ liệu cần hiển thị.

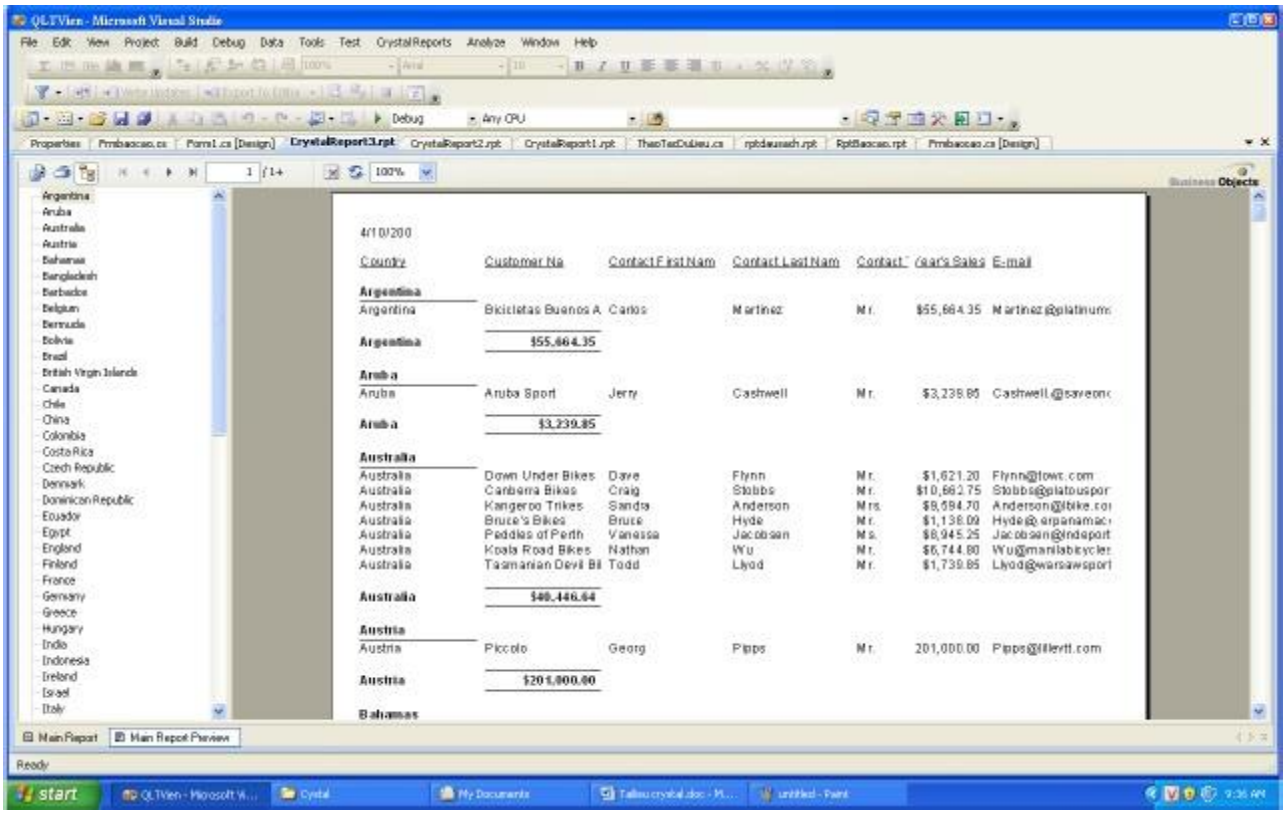
B3. Thêm vào trong dự án một CrystalReport. Xác định nguồn dữ liệu là DataSet.

B4. Thêm dự án một Form, và thêm vào From một điều khiển crystalReportViewer.

B5. Viết Code để lấy dữ liệu gán cho CrystalReport và hiển thị Report trong sự kiện FormLoad.

Các bước trên sẽ được trình bày chi tiết dưới đây:

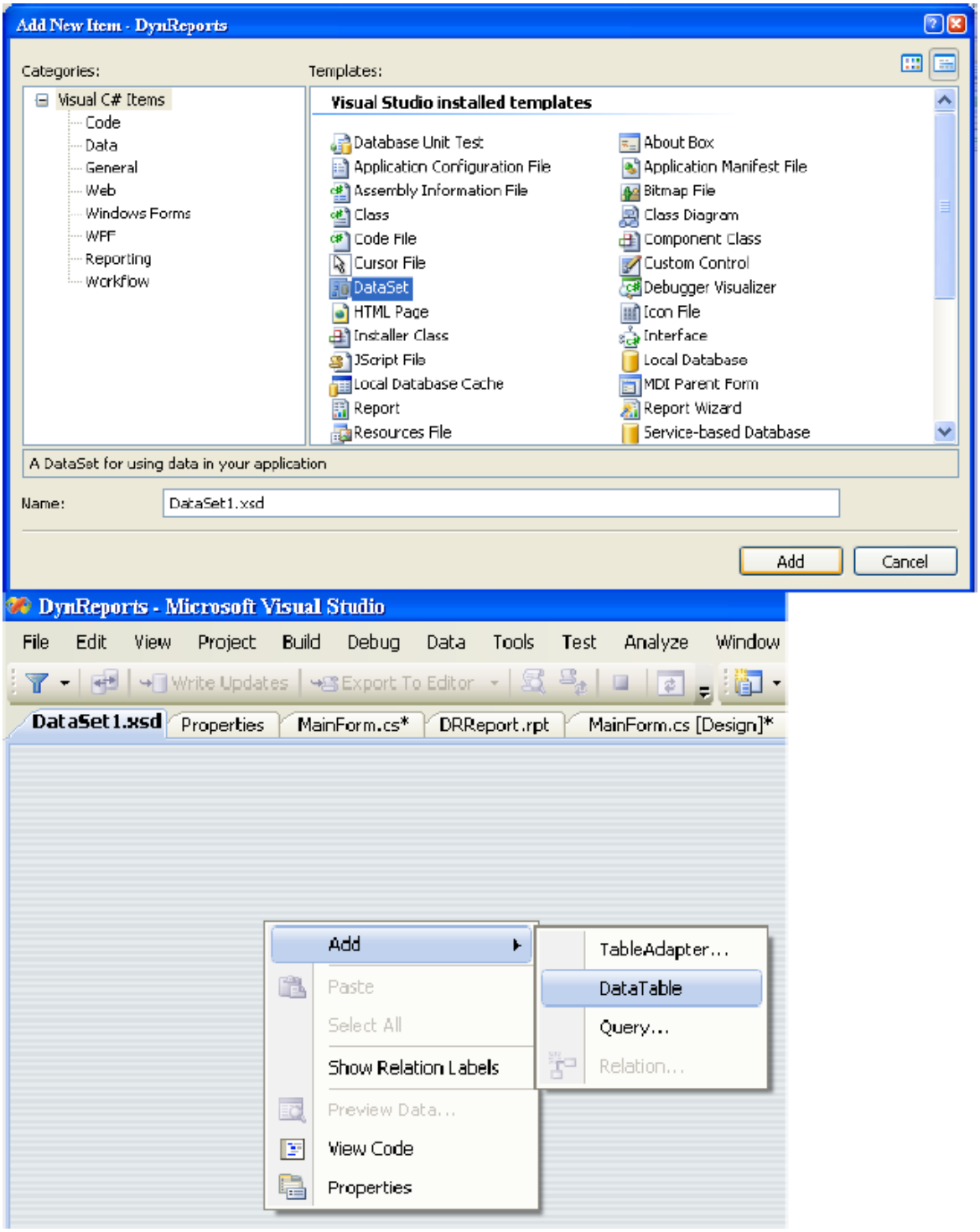
Bước 1: Thêm vào dự án một DataSet (*.xsd file) (Giả sử trong dự án ta cần hiển thị thông tin về các cuốn sách có 2 thông tin là Madau và tên đầu.)- Vào Menu Project\Addnew Item:



Bước 2: Thêm vào các bảng có cấu trúc và tên trường giống như cấutrúc của các dữ liệu cần hiển thị.

Right Click trên nền trống của DataSet để tạo bảng, tiếp tục tạo các cột đặt tên cột giống với tên trường dữ liệu cần hiển thị.

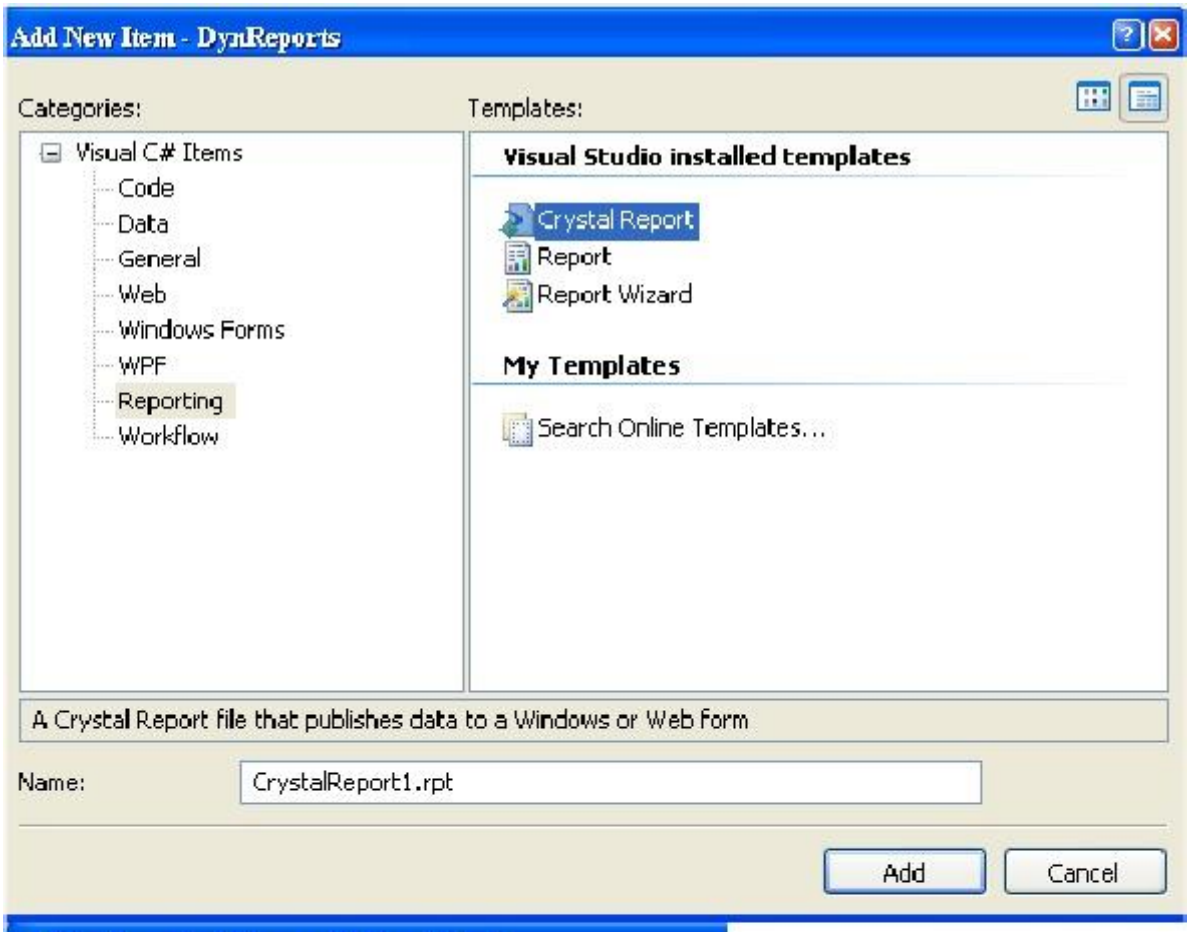
Thêm vào trong dự án một DataSet:

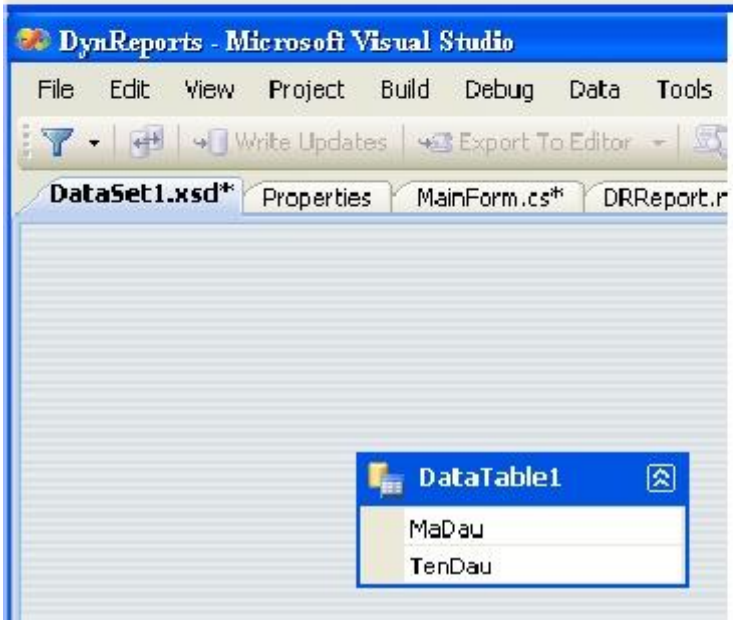


Bước 3: Thêm vào trong dự án một CrystalReport. Xác định nguồn dữ liệu là DataSet.

-Vào Menu Project\Add new Item

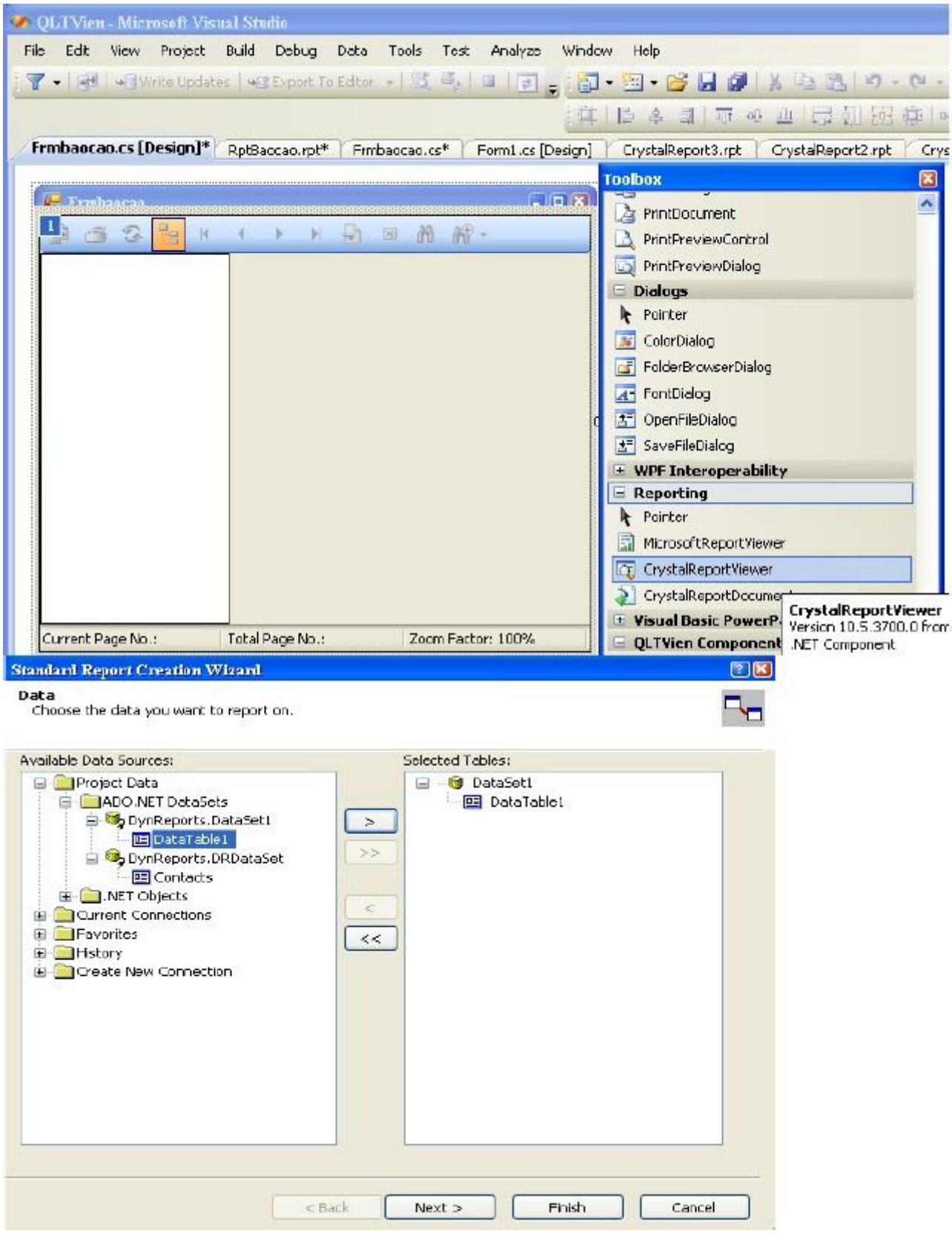
- Trong bước chọn nguồn dữ liệu thì chọn dữ liệu là DataSet1 vừa mới tạo trong bước trên trong Project Data\ ADO.NET DataSets. Và chọn các bảng cần hiển thị trong DataSet1:





-Nhấn Next trong các bước tiếp theo như là phần đã trình bày trong bài trước.

Bước 4: Thêm điều khiển crystalReportViewer để hiển thị báo cáo vàoFrom gọi Crystal.



Bước 5: Viết Code

```
ThaoTacDuLieu dl; // Khai báo một đối tượng của lớp
ThaoTacDuLieu
DataSet ds;
```

```
public Frmbaocao(){// khởi tạo lớp ThaoTacDuLieu với tham số
truyền vào là một chuỗi kết nối.
dl = new ThaoTacDuLieu(Program.f.StringConnection);
InitializeComponent();}
 private void Frmbaocao_Load(object sender, EventArgs e){
// Thi hành phương thức FillDataSet để fill dữ liệu vào DataSet
dl.FillDataSet("select * from sach", "sach");
// Lấy về đối tượng DataSet của lớp thao tác dữ liệu.
ds = dl.GetDaTaSet;
// Tạo ra đối tượng của CrystalReport tên là RptBaocao.
RptBaocao rp = new RptBaocao();
// Thiết lập nguồn dữ liệu là DataSet cho CrystalReport.
// DataSet truyền vào sẽ truyền dữ liệu cho Tệp DataSet tạo
trong bước1.
rp.SetDataSource(ds);rp.Refresh();
// Hiển thị CrystalReport lên crystalReportViewer bằng cách gán
đối tượng Report cho điều khiển crystalReportViewer.
this.crystalReportViewer1.ReportSource = rp;
}
```

BÀI 7: ADO.NET VÀ XML

Mã Bài: MĐ21_07

Giới thiệu:

XML (eXtensible Markup Langue) đóng một vai trò quan trọng trong .NET. Không chỉ vì .NET cho phép bạn sử dụng XML trong các ứng dụng của bạn, mà bản thân nó cũng sử dụng XML cho những file cấu hình và tài liệu mã nguồn, như SOAP, các dịch vụ web và Ado.net. Do đó tìm hiểu về các xử lý XML trong .Net với ngôn ngữ C# là một điều nên làm. Bài viết này sẽ đi từng bước trong quá trình với những ví dụ cụ thể.

Mục tiêu:

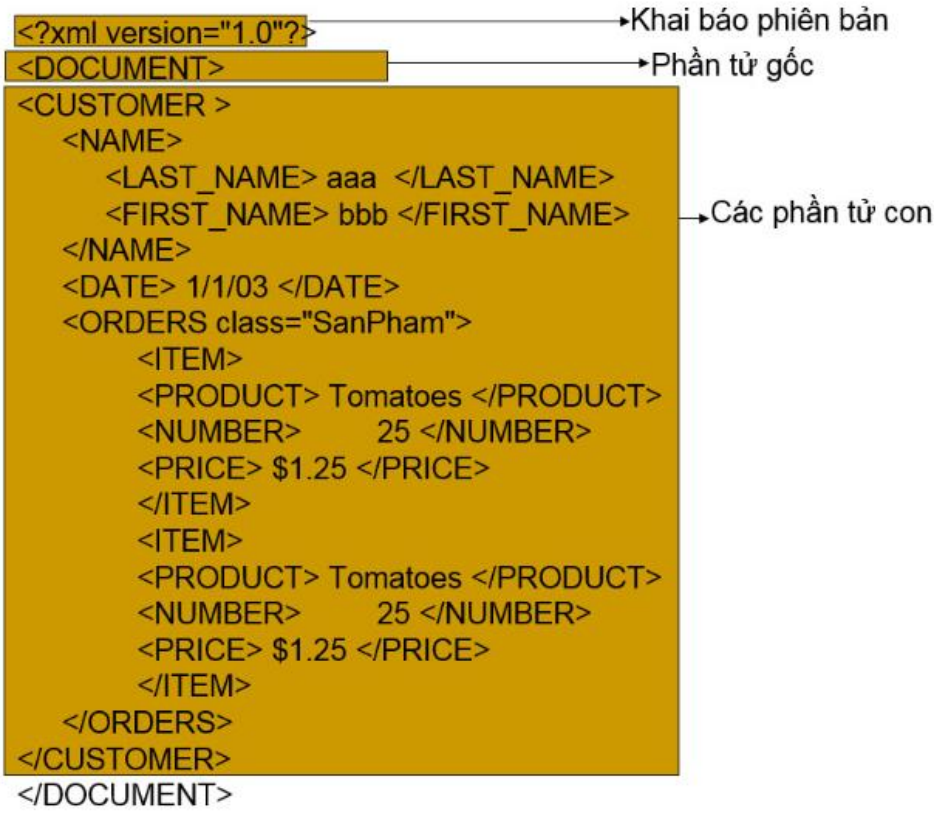
- Viết được cấu trúc XML và mối quan hệ với ADO.Net;
- Tạo được tập tin XML từ cơ sở dữ liệu, đọc lại tập tin XML và hiển thị lên lưới;
- Dùng Visual Studio 2010 tạo được tập tin XML từ cơ sở dữ liệu, đọc lại tập tin XML đã lưu trên đĩa;
- Thái độ tích cực, tỉ mỉ, sáng tạo.

Nội dung chính:

1. Tạo tài liệu XML

1.1 Định nghĩa các thẻ XML

Ví dụ về cấu trúc XML:



1.2 Lưu và quản lý tài liệu XML

Để ghi tập tin XML lên một thư mục trên đĩa cứng chúng ta có thể thực hiện đoạn lệnh sau:

```
string s = "D:\XML";
string fileName = string.Format("{0}_{1}_{2}_{3}", "products",
DateTime.Now.Date.Year, DateTime.Now.Date.Month,
DateTime.Now.Date.Day);
string path_file = string.Format("{0}/{1}.xml", s, fileName);
if (!System.IO.File.Exists(path_file))
        {
XmlTextWriter writer = newXmlTextWriter(path_file,
System.Text.Encoding.UTF8);
            writer.WriteStartDocument(true);
            writer.Formatting = Formatting.Indented;
            writer.Indentation = 2;
            writer.WriteStartElement("products");
            createNode("1", "Shirt", "1000", writer);
            createNode("2", "Jeans", "2000", writer);
            createNode("3", "Jacket", "3000", writer);
            createNode("4", "Coast", "4000", writer);
```

```
            writer.WriteEndElement();
            writer.WriteEndDocument();
            writer.Close();
        }
```

2. Nạp dữ liệu XML

2.1 Đọc dữ liệu XML vào DataSet

Chẳng hạn tập tin products.xml có cấu trúc như sau:

```
<?xmlversion="1.0"encoding="utf-8" ?>
  <Table>
    <Product>
            <Product_id>1</Product_id>
            <Product_name>Product 1</Product_name>
            <Product_price>1000</Product_price>
    </Product>
        <Product>
            <Product_id>2</Product_id>
            <Product_name>Product 2</Product_name>
            <Product_price>2000</Product_price>
    </Product>
    <Product>
            <Product_id>3</Product_id>
            <Product_name>Product 3</Product_name>
            <Product_price>3000</Product_price>
    </Product>
    <Product>
            <Product_id>4</Product_id>
            <Product_name>Product 4</Product_name>
            <Product_price>4000</Product_price>
    </Product>
  </Table>
```

Đoạn lệnh sau đây sẽ đọc tập tin XML vào DataSet:

```
string fileName = @"D:\xml\products.xml";
XmlReader xmlFile;
xmlFile = XmlReader.Create(fileName, newXmlReaderSettings());
DataSet ds = newDataSet();
```

```
ds.ReadXml(xmlFile);
```

2.2 Gán nguồn dữ liệu cho DataGridview

Với việc đọc dữ liệu vào đối tượng DataSet ở trên, việc gán nguồn dữ liệu cho DataGridview có thể thực hiện dễ dàng qua lệnh sau:

```
DataGridView1.DataSource = ds.Tables[0];
```

3. Giản đồ XML

3.1 Tạo lược đồ XML

Cú pháp:

Dạng 1:

```
<?xml version=“1.0” ?>
<schema xmlns=“URL” xmlns:namespace=“URL reference”
targetNamespace=“URL reference”>
  Định nghĩa cấu trúc lược đồ
</ schema >
```

Ví dụ:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:dcs="http://bigblue/XML"
    targetNamespace="http://bigblue/XML">
     <complexType>       ……
    </complexType>
</schema>
```

Dạng 2:

```
<?xml version=“1.0” ?>
<prefixNamespace:schema xmlns:prefixNamespace=“URL” >
  Định nghĩa cấu trúc lược đồ
</prefixNamespace:schema>
```

Ví dụ:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <xsd:complexType>
       ……
    </xsd:complexType>
</xsd:schema>
```

3.2 Lưu và quản lý lược đồ XML

Để lưu và quản lý lược đồ XML, thường chúng ta sẽ lưu cùng thư mục với tập tin XML để dễ dàng kiểm tra tính hợp lệ của cấu trúc XML.

4. Kiểm tra sự hợp lệ của dữ liệu

4.1. Các mẫu tài liệu XML tốt.

Một tài liệu XML gọi là có khuôn mẫu tốt phải có cú pháp chính xác:

– Các tài liệu XML phải có một phần tử gốc.

– Các phần tử phải có thẻ đóng.

– Các thẻ (tag) XML đòi hỏi độ chính xác.

– Các phần tử trong XML phải được lồng nhau.

– Các giá trị thuộc tính phải được trích dẫn.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

4.2. Những tài liệu XML hợp lệ

Một tài liệu XML hợp lệ 1 tài liệu XML có khuôn mẫu tốt, là sự phù hợp với các quy tắc của một DTD (Document Type Definition).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Việc khai báo DOCTYPE trong ví dụ trên là một tham chiếu đến một tập tin DTD bên ngoài. Nội dung của tập tin sẽ nói đến trong phần tiếp theo dưới đây.

4.3. XML DTD

Mục đích của DTD là để các định cấu trúc của một tài liệu XML. Nó định nghĩa các cấu trúc với một danh sách các phần tử (elements) hợp lệ.

```
<!DOCTYPE note
[
&lt;!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
```

4.4. Lượt đồ XML (XML Schema)

W3C hỗ trợ một XML cơ bản dựa trên DTD, được gọi là XML SChema (lượt đồ XML).

```
<xs:element name="note">
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

4.5. Tổng hợp về XML Validator

Nó giúp chúng ta kiểm tra cú pháp của các tập tin XML. Tổ chức W3C tạo ra một XML Validator (Máy đánh dấu XML) để kiểm tra cú pháp XML của bạn.

BÀI 8: Xây dựng ứng dụng tổng hợp

Mã Bài: MĐ21_08

Giới thiệu:

Sau khi có đầy đủ kiến thức và kỹ năng lập trình tương tác với cơ sở dữ liệu, việc xây dựng một ứng dụng tổng hợp nhằm cũng cố kỹ năng tổng hợp cho sinh viên.

Mục tiêu:

- Vận dụng các kiến thức đã học để xây dựng bài toán theo yêu cầu;
- Cài đặt các mã lệnh;
- Phân tích và kiểm thử lỗi phát sinh;
- Xây dựng được các phần mềm ứng dụng dựa trên hệ quản trị cơ sở dữ liệu có sẵn.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

1. Giới thiệu bài toán

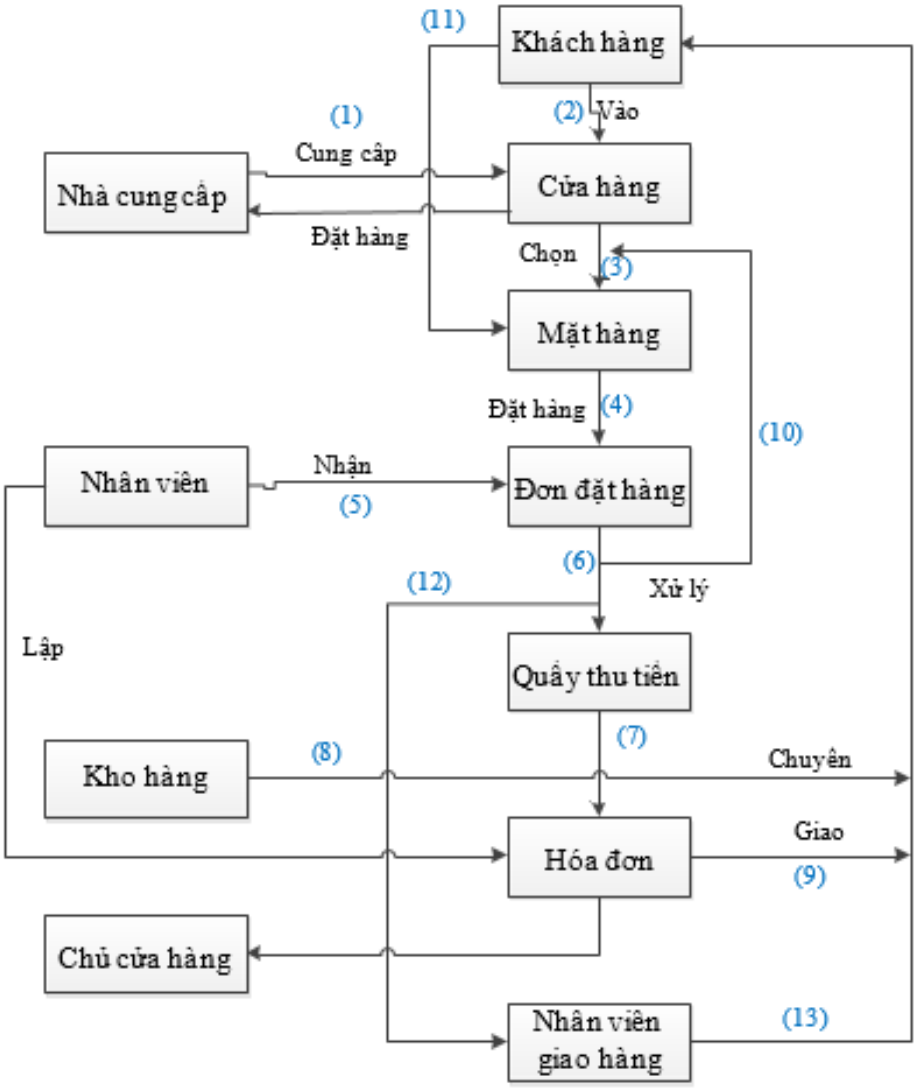
Cửa hàng bán điện thoại PHONESHOP chuyên cung cấp điện thoại của các hãng như Apple, Samsung, Nokia, LG, HTC…Các dòng điện thoại thường và điện thoại thông minh (smartphone).

Với 5 năm hoạt động, phục vụ quý khách hàng. PHONESHOP luôn là điểm đến yêu thích của giới công nghệ nói chung và giới săn điện thoại mới nói riêng. Cửa hàng luôn cấp nhật các thông tin, mặt hàng mới nhất trên thế giới cũng như ở Việt Nam. Đội ngũ nhân viên trẻ đẹp, năng động, sẵn sàng phục vụ quý khách hàng kịp thời, đầy đủ nhất.

Tiêu chí hoạt động của của hàng là làm sao mặt hàng đến tay người tiêu dùng giá rẻ nhất thị trường nhưng chất lượng phải được đảm bảo. Việc giữ uy tín với khách hàng là phương châm hoạt động, phát triển thương hiệu của cửa hàng.

Hiện tại cửa hàng chỉ phục vụ cho khách hàng đến mua hàng trực tiếp tại cửa hàng nhưng do sự mở rộng về quy mô bán hàng. Chủ cửa hàng có ý định phục vụ cho khách hàng ở xa, không có điều kiện đến mua trực tiếp.

Quy trình mua hành thực hiện như sơ đồ sau:



(1) Khi có yêu cầu đặt hàng từ cử hàng, nhà cung cấp sẽ cung cấp sản phẩm cho của hàng theo đơn đặt hàng.

(2) Khách hàng vào cửa hàng để xem các mặt hàng trong cửa hàng.

(3) Sau khi vào cửa hàng, khách hàng chọn các mặt hàng định mua.

(4) Khi quyết đinh mua hàng, thông tin mặt hàng đó được lưu ở đơn đặt hàng.

(5) Nhân viên nhận đơn đặt hàng từ khách hàng.

(6) Đơn đặt hàng được nhân viên xử lý. Nếu đơn đặt hàng đó nằm trong tầm phục vụ của cửa hàng thì chuyển đơn đặt hàng đó đến quầy thu tiền, nếu không (hết hàng, chưa có hàng..) thì khách hàng có thể quay lại cửa hàng để chọn mua phẩm khác (10).

(7) Nhân viên ở quầy thu tiền tiến hành lập hóa đơn, một bên giao cho khách hàng (9), hóa đơn đó sẽ được chủ cửa hàng kiểm tra, thống kê cuối tháng.(1x)

(8) Sản phẩm sẽ được nhân viên lấy từ kho hàng chuyển cho khách hàng.

2. Phân tích và thiết kế theo yêu cầu

- Thiết kế các bảng dữ liệu:

STT	Tên bảng	Diễn giải
1	tbl_nguoidung	Bảng người dùng
2	tbl_khachhang	Bảng khách hàng
3	tbl_sanpham	Bảng sản phẩm
4	tbl_dm_sp	Bảng danh mục sản phẩm
5	tbl_don_dh	Bảng đơn đặt hàng
6	tbl_ct_ddh	Bảng chi tiết đơn đặt hàng
7	tbl_binhluan	Bảng bình luận
8	tbl_sp_ban	Bảng sản phẩm bán ra
9	tbl_tinh_trang	Bảng tình trạng đơn đặt hàng
10	tbl_nv_gh	Bảng nhân viên giao hàng

- Bảng tbl_nguoidung

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_nd	Id người dùng	Int(11)	Khóa chính
2	Ten	Tên người dùng	Varchar(50)	
3	Tai_khoan	Tài khoản người dung	Varchar(50)	
4	Mat_khau	Mật khẩu người dung	Varchar(25)	

- Bảng tbl_khachhang

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_kh	Id khách hàng	Int(11)	Khóa chính
2	Ten_kh	Tên khách hàng	Varchar(25)	
3	Sdt	Số điện thoại khách hàng	Varchar(15)	
4	Mail	Mail khách hàng	Varchar(100)	

Bảng tbl_sanpham

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_sp	Id sản phẩm	Int(11)	Khóa chính
2	Id_km	Id khuyến mãi	Int(11)	Khóa ngoại
3	Id_dm	Id danh mục	Int(11)	Khóa ngoại
4	Ten_sp	Tên điện thoại	Varchar(100)	
5	Anh_sp	Ảnh sản phẩm	Varchar(50)	
6	Gia_sp	Giá sản phẩm	Int(12)	
7	So_luong	Số lượng điện thoại	Int(5)	
8	Kich_thuoc	Kích thước sản phẩm	Varchar(255)	
9	Trong_luong	Trọng lượng sản phẩm	Varchar(255)	
10	Mau_sac	Màu sắc sản phẩm	Varchar(255)	
11	Am_thanh	Âm thanh	Varchar(255)	
12	Bo_nho	Bộ nhớ máy	Varchar(255)	
13	He_dieu_hanh	Hệ điều hành của máy	Varchar(255)	
14	The_nho	Thẻ nhớ	Varchar(255)	
15	Camera	Camera	Varchar(255)	
16	Pin	Loại pin, số giờ hoạt động	Varchar(255)	
17	Bao_hanh	Thời gian bảo hành	Varchar(255)	
18	Ket_noi	Kết nối với các thiết bị khác	Varchar(255)	
19	Gia_km	Giá khuyến mãi	In(11)	

20	Batdau_km	Ngày bắt đầu khuyến mãi	Datetime	
21	Ketthuc_km	Ngày kết thức khuyến mãi	Datetime	

Bản tbl_dm_sp

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_dm	Id danh mục	Int(11)	Khóa chính
2	Ten_danhmuc	Tên danh mục	Varchar(50)	

Bảng tbl_don_dh

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_hd	Id hóa đơn	Int(11)	Khóa chính
2	Id_kh	Id khách hàng	Int(11)	Khóa ngoại
3	Id_tinh_trang	Id tình trạng đơn đặt hàng	Int(3)	Khóa ngoại
4	Id_nvgh	Id nhân viên giao hàng	Int(11)	Khóa ngoại
5	Ngay_lap	Ngày lập đơn đặt hàng	Datetime	
6	Tong_gia	Tổng giá trị đơn hàng	Int(11)	
7	Noi_nhan	Địa điểm nhận đơn hàng	Varchar(255)	
8	Ghi_chu	Ghi chú	Varchar(255)	

Bảng tbl_binhluan

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_bl	Id bình luận	Int(11)	Khóa chính
2	Id_sp	Id sản phẩm	Int(11)	Khóa ngoại
3	Ho_ten	Họ tên người bình luận	Varchar(25)	
4	Ngay_gio	Ngày giờ bình luận	Datetime	
5	Noi_dung	Nội dung bình luận	Varchar(255)	
6	Dien_thoai	Điện thoại người bình luận	Varchar(25)	

Bảng tbl_ct_ddh

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_ct_hd	Id chi tiết hóa đơn	Int(11)	Khóa chính
2	Id_hd	Id hóa đơn	Int(11)	Khóa ngoại
3	Id_sp	Id sản phẩm	Int(11)	Khóa ngoại
4	So_luong_mua	Số lượng mỗi sp trong hóa đơn	Int(5)	
5	Don_gia	Giá mỗi loại sp khi mua	Int(11)	

Bảng tbl_sp_ban

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_sp_ban	Id sản phẩm bán	Int(11)	Khóa chính
2	Id_sp	Id sản phẩm	Int(11)	Khóa ngoại
3	So_luong_ban	Số lượng bán	Int(11)	

Bảng tbl_tinh_trang

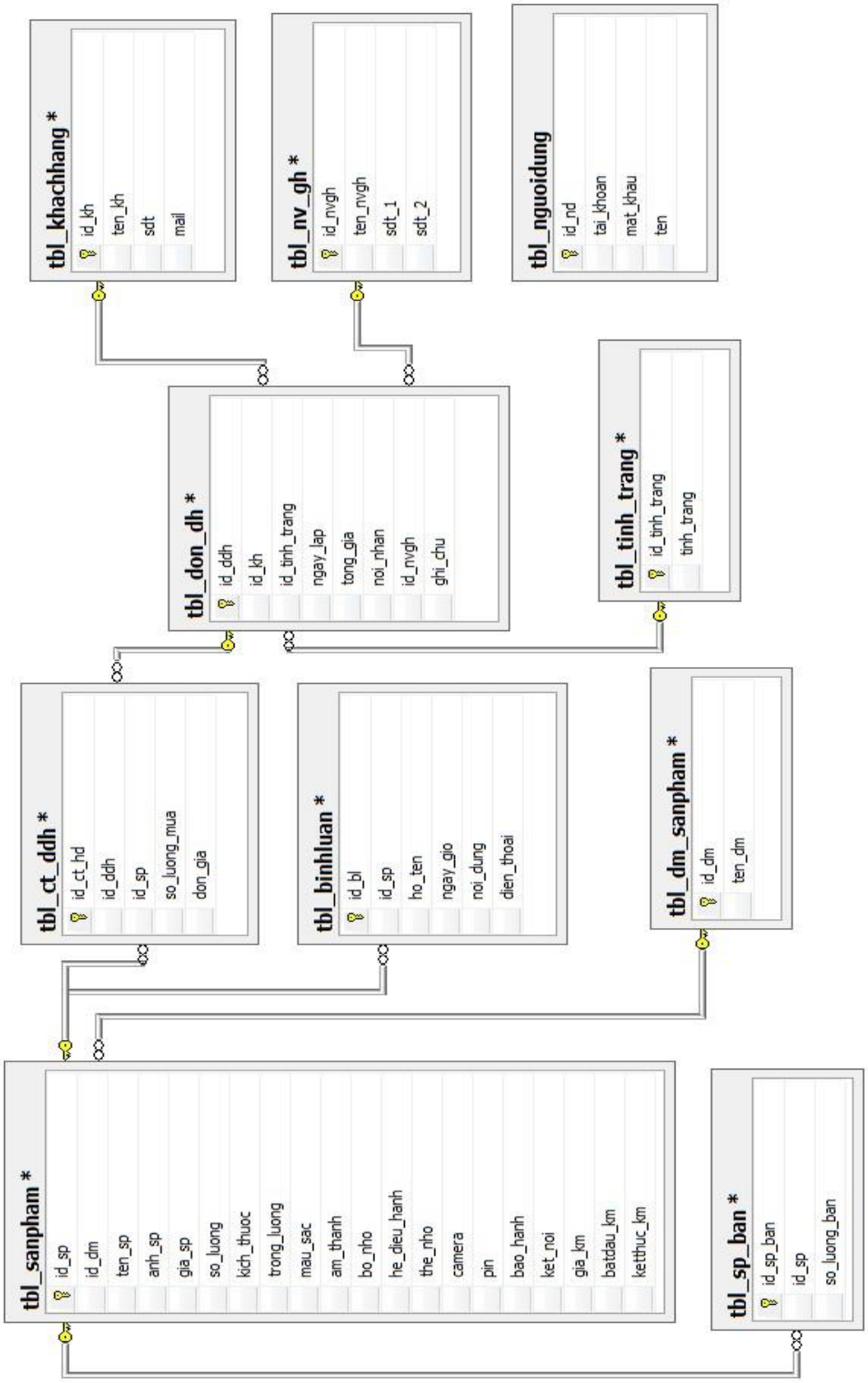
STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_tinh_trang	Id tình trạng đơn đặt hàng	Int(11)	Khóa chính
2	Tinh_trang	Tình trạng đơn đặt hàng	Varchar(255)	

Bảng tbl_nv_gh

STT	Tên trường	Diễn giải	Kiểu dữ liệu	Khóa
1	Id_nvgh	Id nhân viên giao hàng	Int(11)	Khóa chính

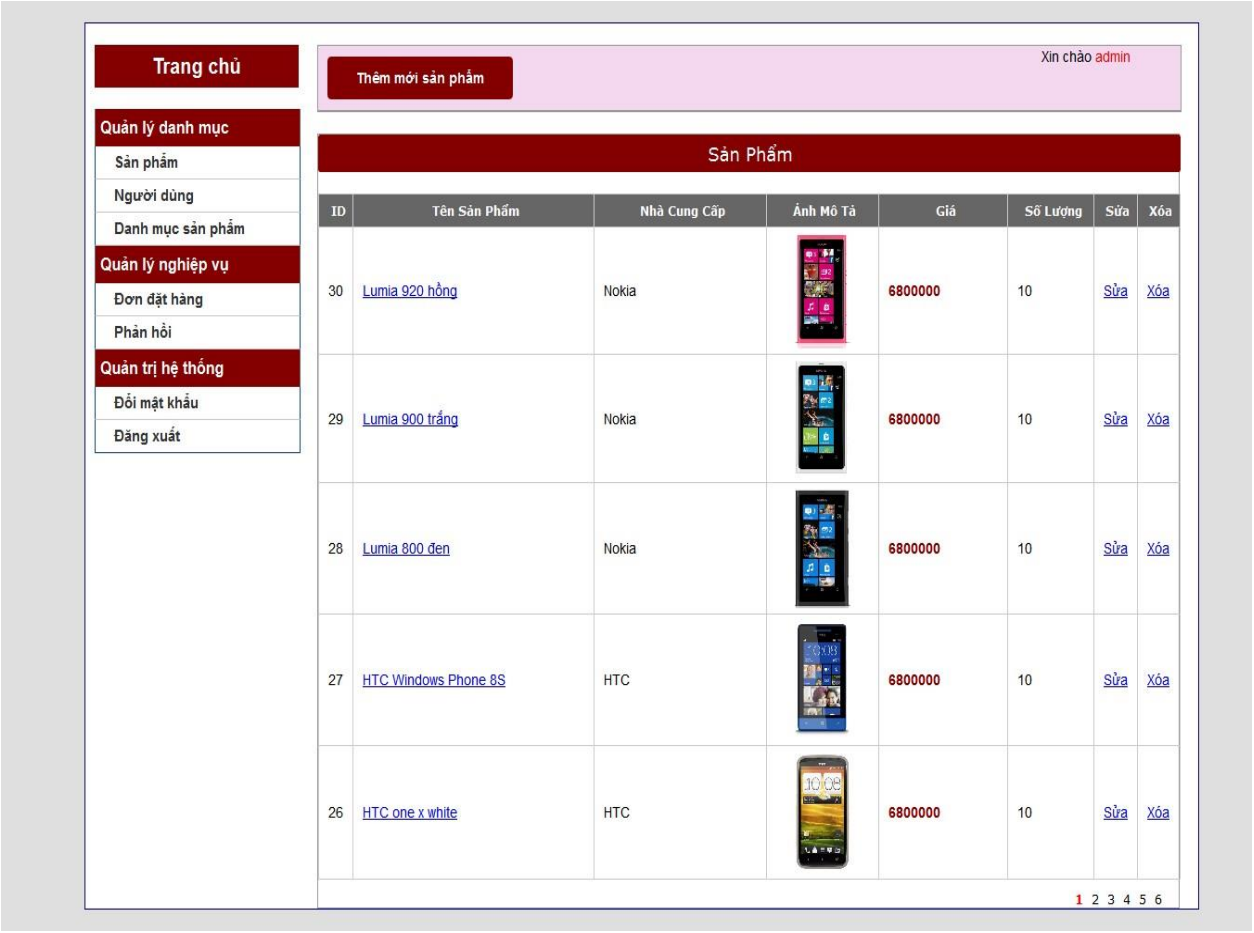
2	Ten_nvgh	Tên nhân viên giao hàng	Varchar(50)	
3	Sdt_1	Số điện thoại thứ 1 của nvgh	Varchar(11)	
4	Sdt_2	Số điện thoại thứ 2 của nvgh	Varchar(11)	

Quan hệ giữa các bảng:



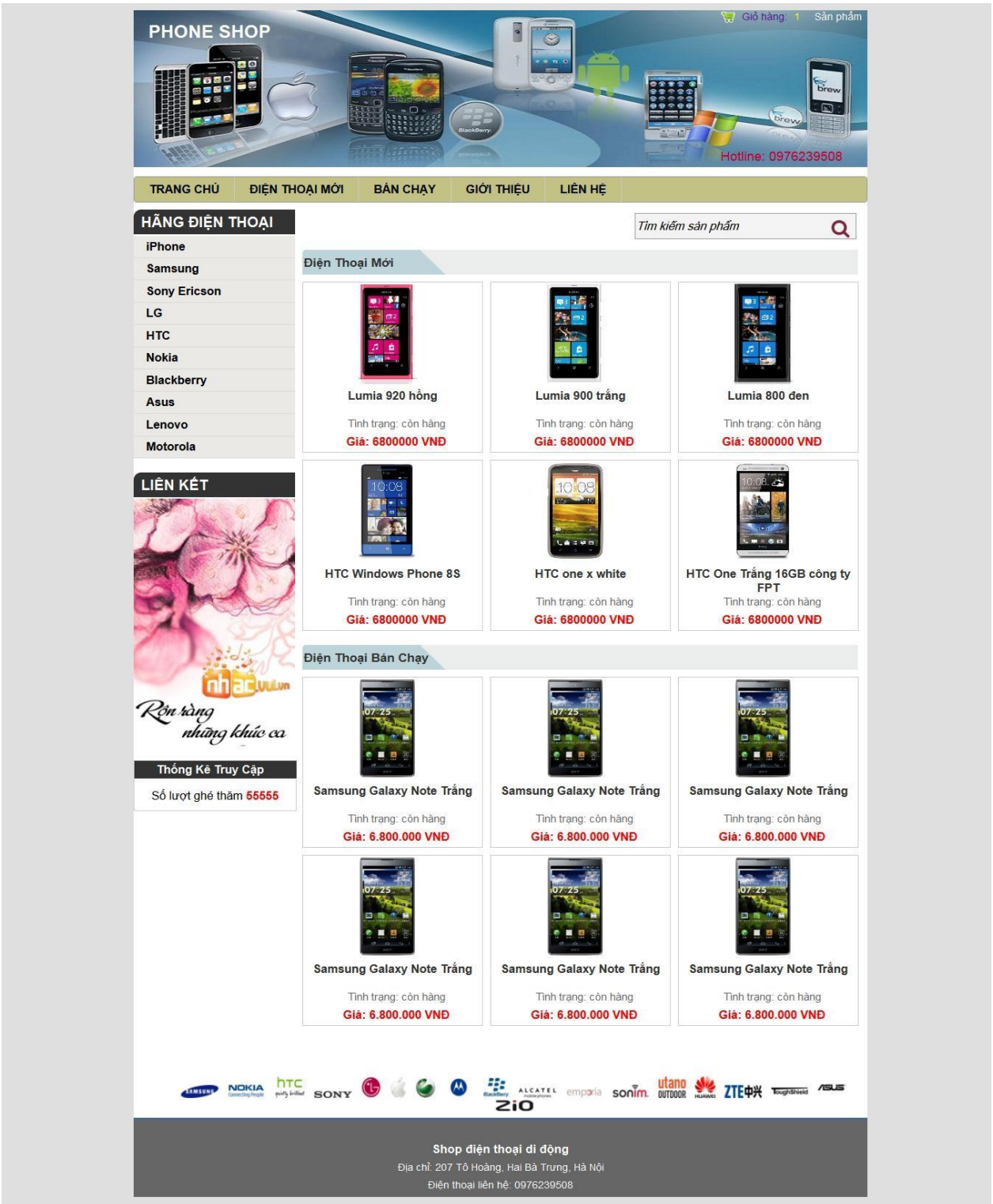
3. Thiết kế các giao diện

- Giao diện phần backend



Hình 8.1: Giao diện phần backend

Giao diện phần frontend



Hình 8.2: Giao diện phần frontend

Thông tin chi tiết sản phẩm

Thông Tin Chi Tiết

Lumia 900 Trắng

Hãng sản xuất: Nokia

Giá: 6.000.000 VNĐ

~~Giá cũ: 6.800.000 VNĐ~~

Tình trạng: Còn 5 sản phẩm

Đặt hàng

Chi tiết sản phẩm | Bình luận sản phẩm

Kích thước	2422 x23325
Trọng lượng	435g
Màu sắc	Đỏ
Âm thanh	Vibration; MP3, WAV ringtones
Bộ nhớ	Bộ nhớ trong 32g, ram 1g
Hệ điều hành	Windows
Thẻ nhớ	2g
Camera	8.7 MP
Pin	Li-Ion 2000mAh
Bảo hành	12 tháng
Kết nối	WIFI, 3G

Hình 8.3: Thông tin chi tiết sản phẩm

4. Cài đặt các mã lệnh

Trong ứng dụng này sinh viên cần thiết kế giao diện và viết code cho các chức năng sau:

+ Đăng nhập hệ thống và phân quyền cho nhân viên

+ Quản lý danh mục sản phẩm

+ Quản lý và liên hệ khách hàng

+ Xử lý các đơn đặt hàng

Các công việc đặt hàng cho khách sẽ được phát triển dạng ứng dụng web sẽ được thực hiện trong mô đun Xây dựng website bằng công cụ.

5. Kiểm thử chương trình

Để đảm bảo tính bảo mật trong việc quản lý bán hàng, sinh viên cần tạo các người sử dụng ở các quyền hạn, vai trò khác nhau để đăng nhập và quản lý theo quy trình quản lý bán hàng, sau đó chạy ứng dụng để kiểm tra các chức năng. Ghi lại các lỗi phát sinh và tiến hành khắc phục lỗi.

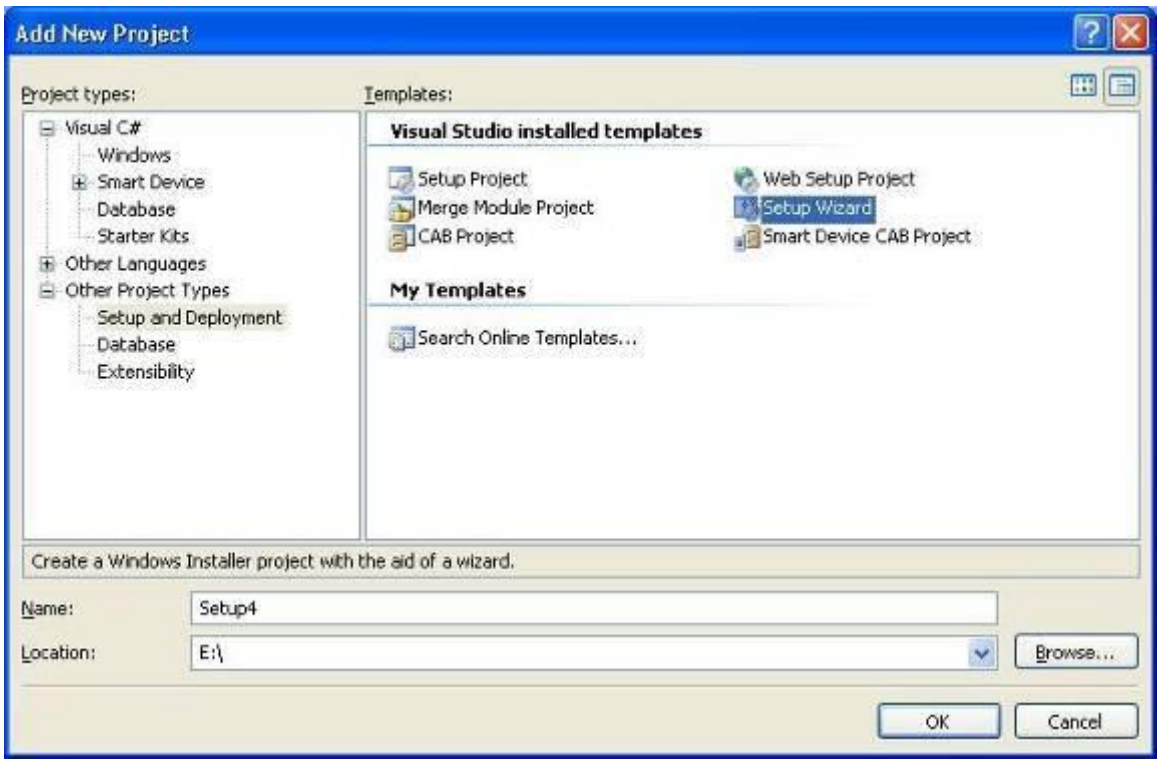
6. Triển khai – Cài đặt chương trình

Các bước tạo một chương trình setup bằng Setup Wizard

B1. Trong Solution Explorer của VS.net , right click vào Solution.

Trong menu popup, vào mục Add –> New project . Xuất hiện hộp thoại Add new project.

Trong hộp thoại Add new project:



B2. Bên trái là Project Types: Chọn Other Project Types –> Setup and Deployment

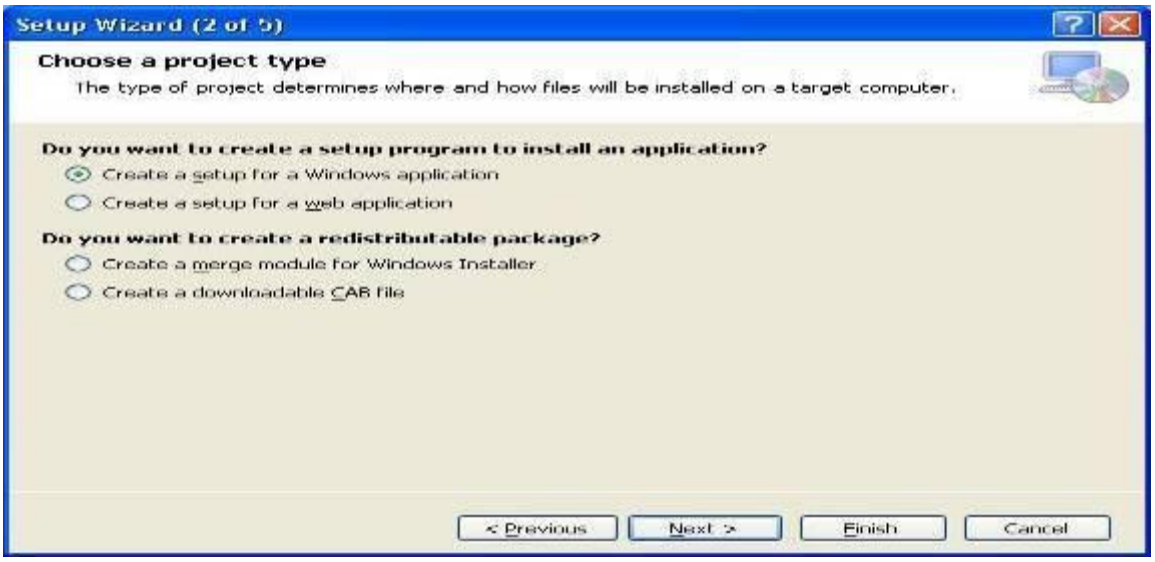
- Bên phải là Templates: Chọn Setup Wizard.

Đặt tên Project mới vào trường name

Chọn đường dẫn cần lưu thư mục tại trường Location

Sau khi click Ok, VS.net sẽ tự động hiển thị hộp thoại là Setup Wizard (1 of 5).

Click chọn Next



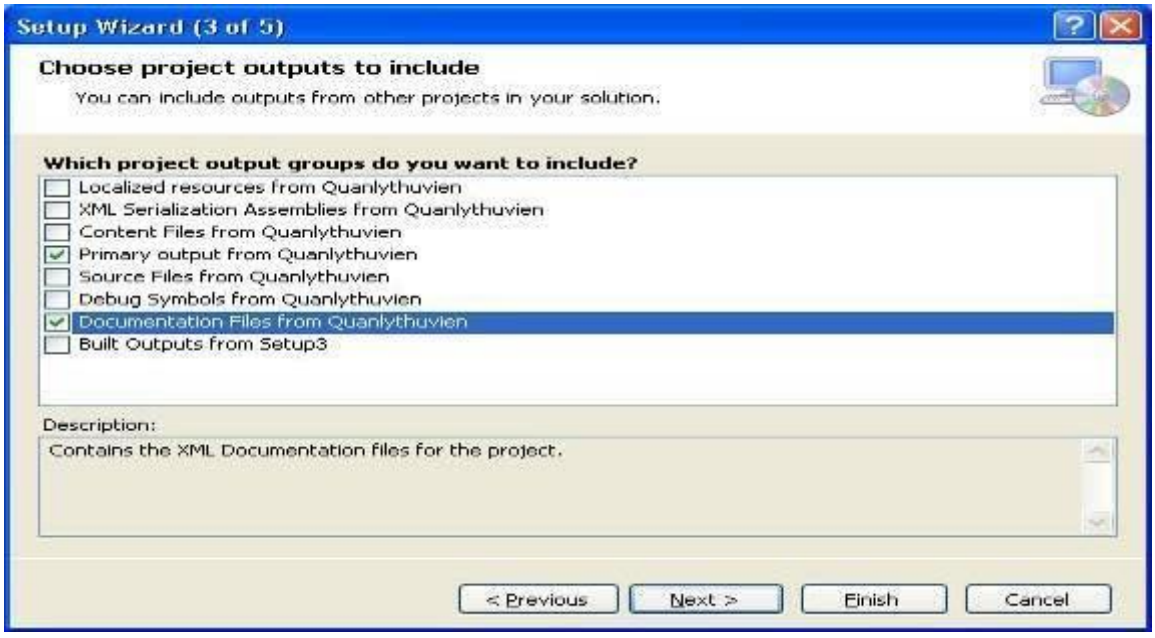
B3. Hộp thoại thứ 2 là Setup Wizard (2 of 5) xuất hiện .

Hộp thoại này cho phép chọn kiểu Project

Trong trường hợp này chúng ta giả sử đóng gói ứng dụng Windows nên đánh dấu check vào radio button : Create a setup for a Windows application.

Còn nếu là ứng dụng Web thì ta chọn radio button : Create a setup for a web application.

Next tiếp :



B4. Hộp thoại thứ 3 là Setup Wizard (3 of 5) xuất hiện . (H5)

Hộp thoại này chứa các loại output mà chúng ta có thể đưa vào file Setup

XXX là tên project cần đóng gói .

* Localized resources from XXX :

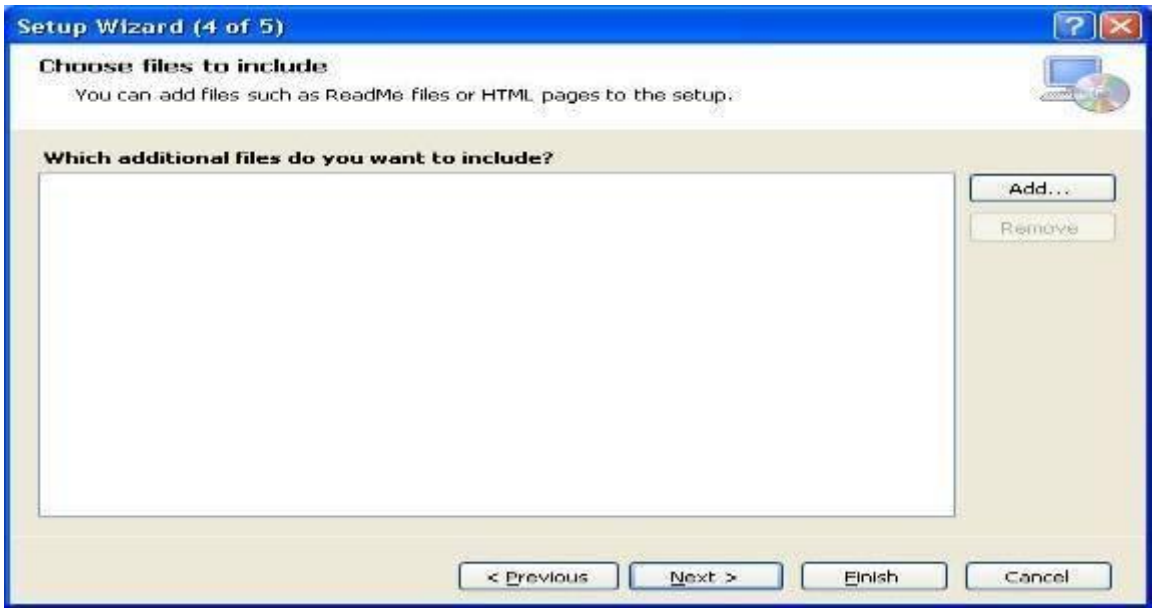
* XML Serialization Assemblies from XXX :

* Content Files from XXX : chứa toàn bộ file chứa trong project XXX , cái này không cần thiết phải chọn.
* Primary output from XXX : chứa file exe và file dll của prject XXX , cái này tất nhiên phải chọn rồi.
* Source files from XXX : chứa file source code của project cần đóng gói như file css , vb , cpp …
* Debug Symbols from XXX : chứa một số file hỗ trợ debug của dự án .
* Documentation Files XXX : (tài liệu XML của dự án, cái này liên quan đến các commnets viết theo qui tắc convention của .Net) . Có thể đưa vào hoặc không.
* Built Output from …: tạo ra file MSI
* Primary Output (chứa các DLL và EXE do Project phía trên tạo ra) - đương nhiên phải đưa vào

Bạn muốn biết thêm thông tin thì có thể xem Description ở bên dưới của hộp thoại đó.

Sau khi chọn xong nhấn Next.

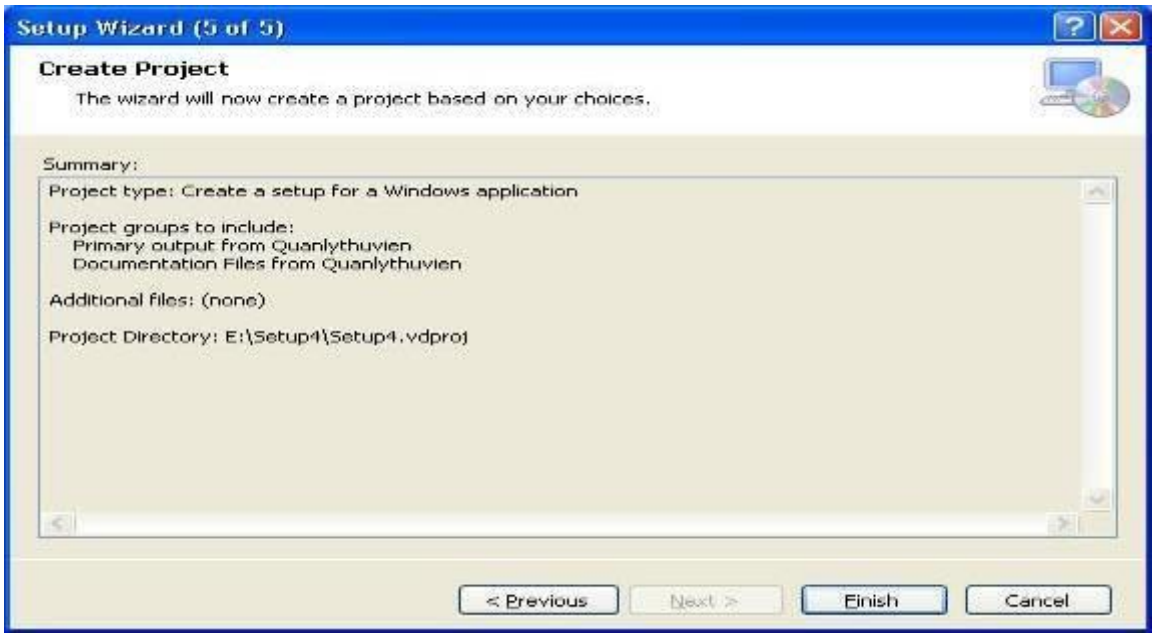
B5. Hộp thoại thứ 4 là Setup Wizard (4 of 5) xuất hiện



Hộp thoại cho phép chọn file đính kèm theo như file Help , ReadMe …

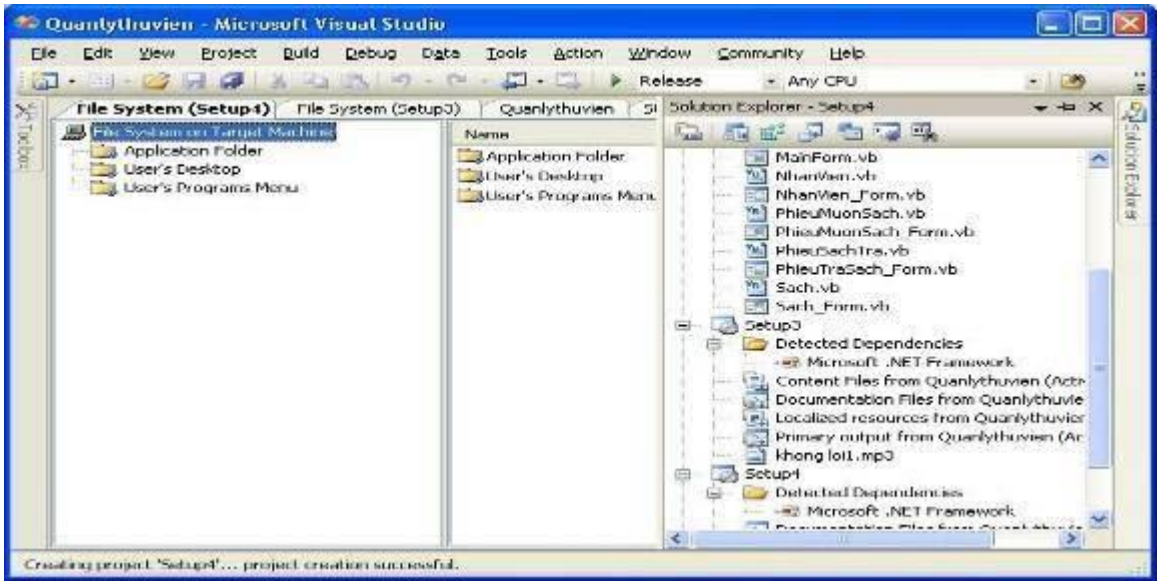
Nếu muốn thêm file nào thì chọn Add à chọn đường dẫn của file đó. Nếu không thì chọn Next.

B6. Hộp thoại thứ 5 xuất hiện Setup wizard (5 of 5) thông báo tóm tắt nội dung kết quả.



B7. Chọn Finish để kết thúc quá trình tạo file cài đặt .

B8. Sau khi nhấn nút Finish trong Wizard cuối cùng thì màn hình (xem H8) hiện ra.



Trong Solution Explorer xuất hiện thêm Project , người ta gọi là Setup Project . Tới đây có thể tạm thời gọi là xong , nhưng chúng ta phải qua một số bước cấu hình để tạo ra file setup đóng gói phần mềm theo ý mình.

Tài liệu tham khảo:

[1] Charles Petzold (2002), *Programming Microsoft Windows With C#*, Ms Press.

[2] Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner (2010), *Professional C# 4.0 and .NET 4*, Wrox.

[3] Erik Brown (2002), *Windows Forms Programming With C#*, Manning.

[4] Link http://www.codeproject.com/Articles/14122/Passing-Data-Between-Forms (2013), CodeProject.

[5] Website: http://www.google.com