Chương 4

CÁC ĐIỀU KHIỂN LIÊN KẾT DỮ LIỆU

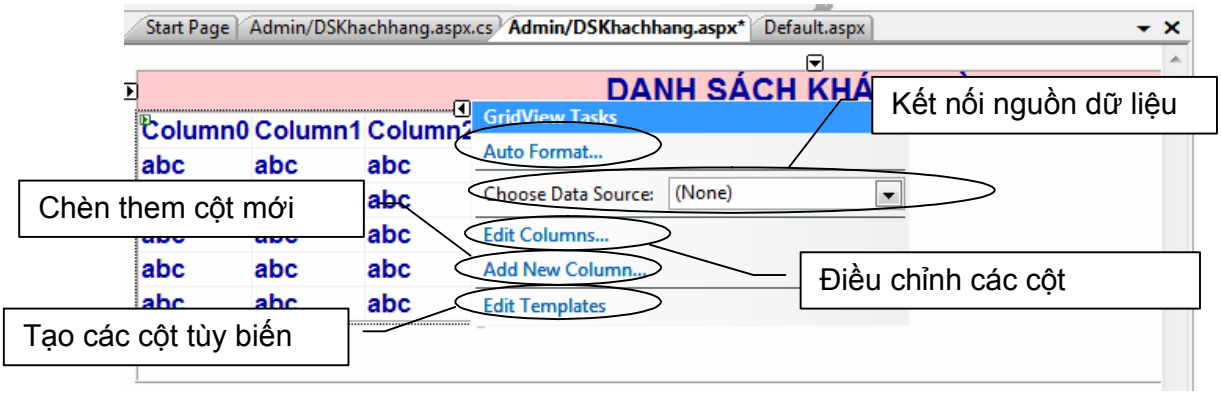
- *Sử dụng các điều khiển GridView, Data List và Repeater để hiển thị dữ liệu.*

I. Điều khiển GridView

GridView là một điều khiển khá linh hoạt và hiệu quả trong việc hiển thị, định dạng và thao tác với dữ liệu. Bên cạnh đó, chúng ta có thể thực hiện sắp xếp dữ liệu, thực hiện phân trang với sự hỗ trợ khá tốt của VS .Net trong quá trình thiết kế.

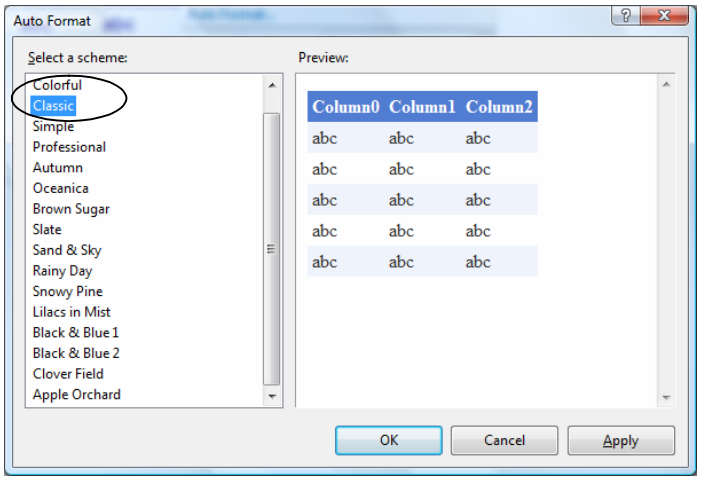
1. Tạo GridView vào trang

Kéo Control GridView vào trang



2. Định dạng tự động

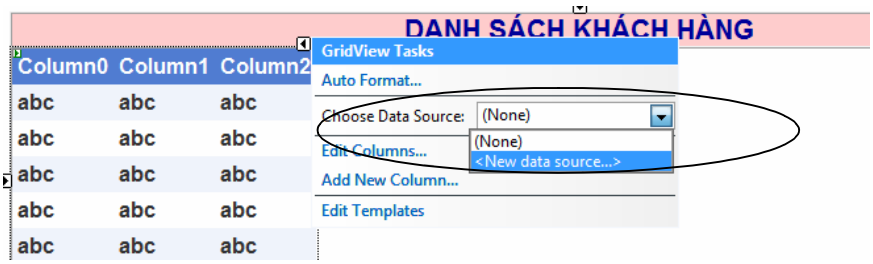
Thực hiện chọn những mẫu định dạng có sẵn quy định về khung viền màu nên bằng cách Chọn Auto Format từ khung DataGrid Task



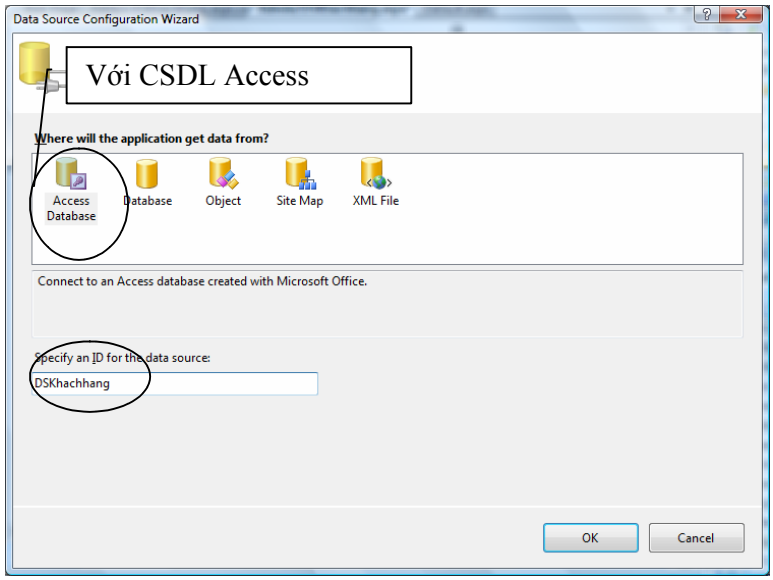
3. Kết nối nguồn dữ liệu

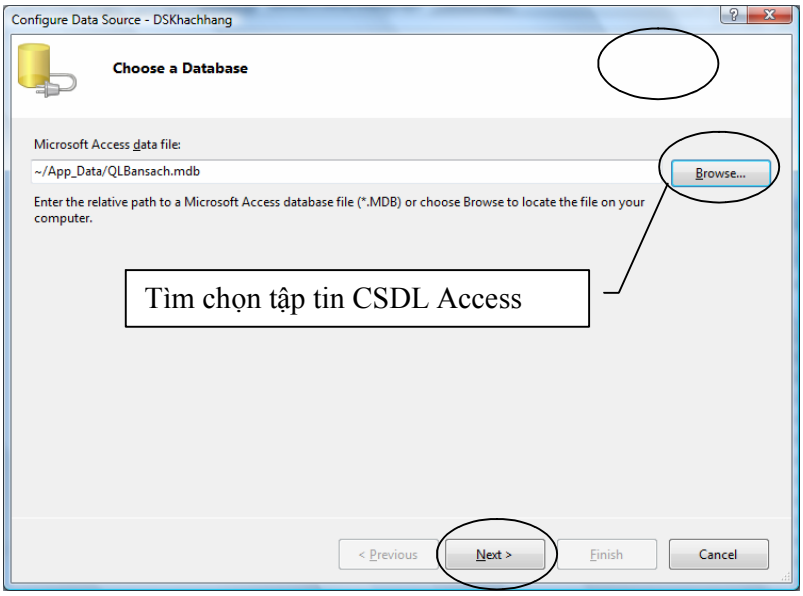
Thực hiện kết nối nguồn dữ liệu với cơ sở dữ liệu Access, SQLServer, . .

- Trong cửa sổ DataGrid Task tại mục Choose Data Source chọn : New Data Source

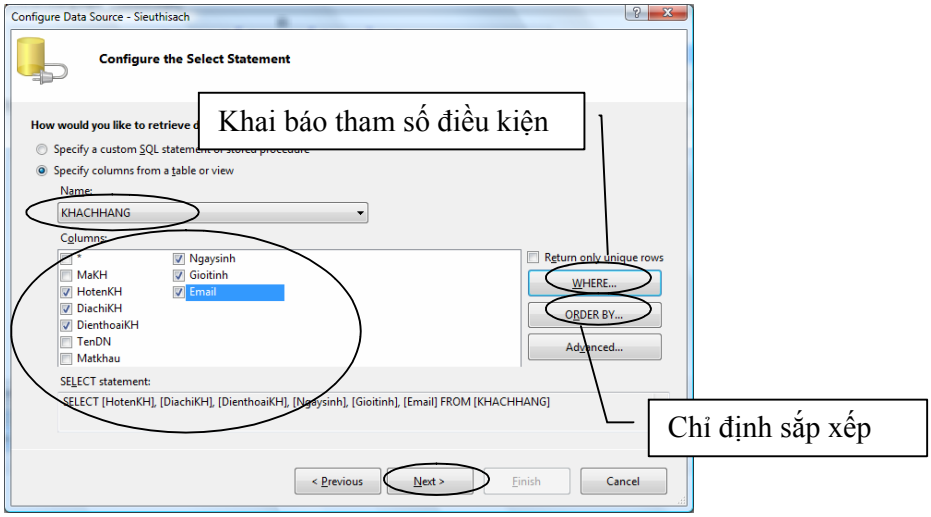


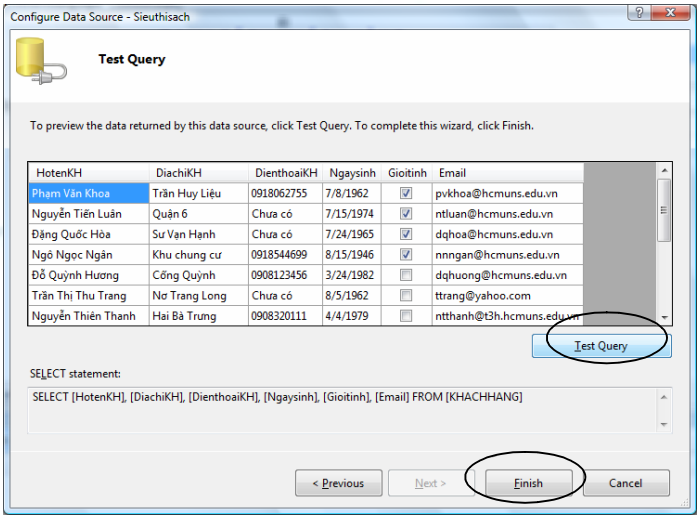
- Với CSDL Access





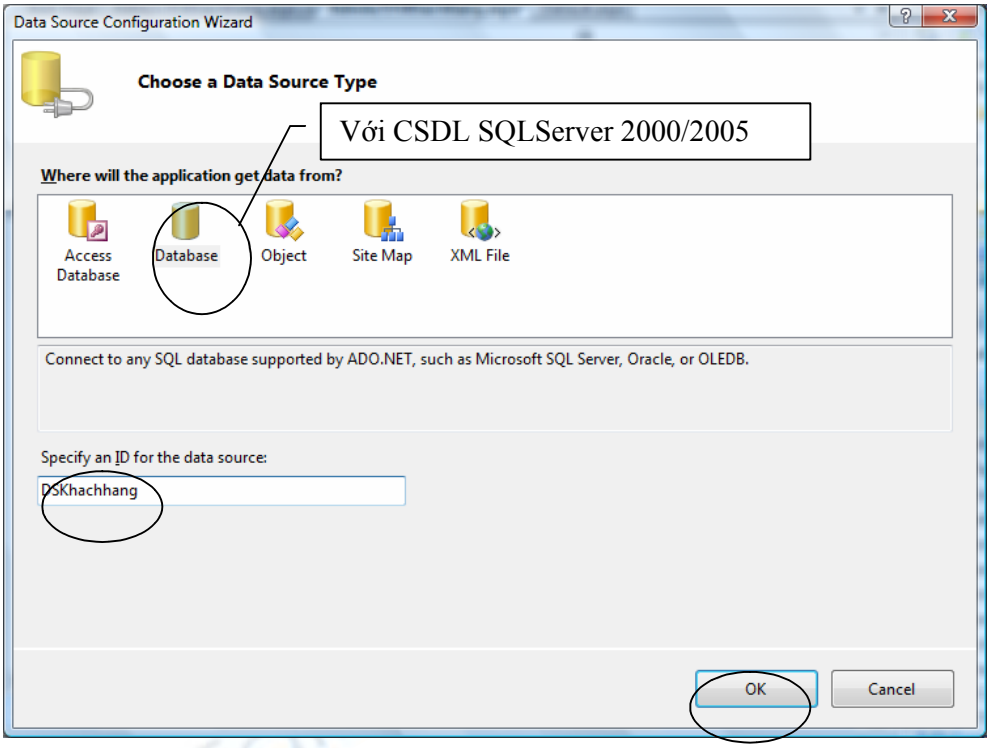
Xác định nguồn dữ liệu từ Table, Quyery hay câu lệnh truy vấn có thể chỉ định điều kiện lọc dữ liệu và sắp xếp.

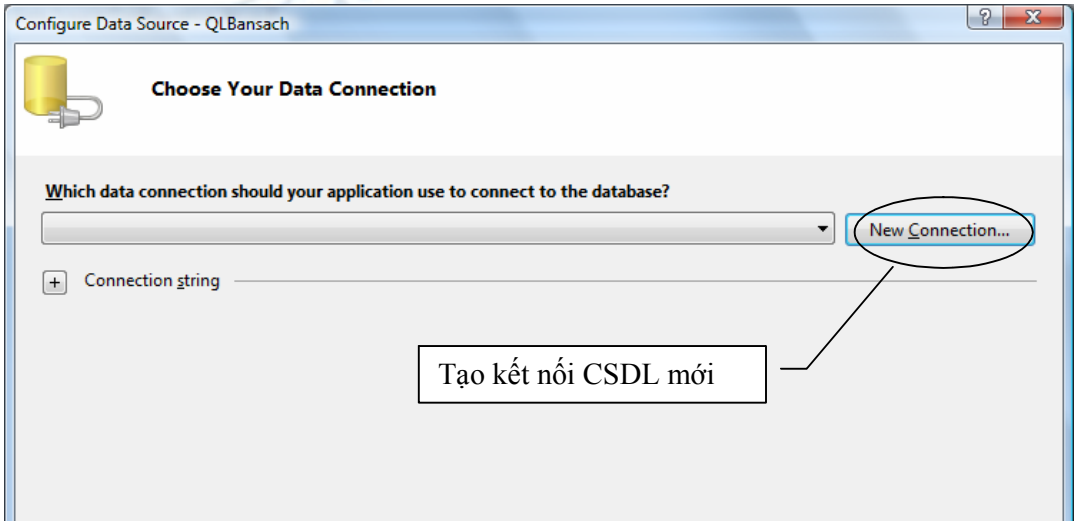




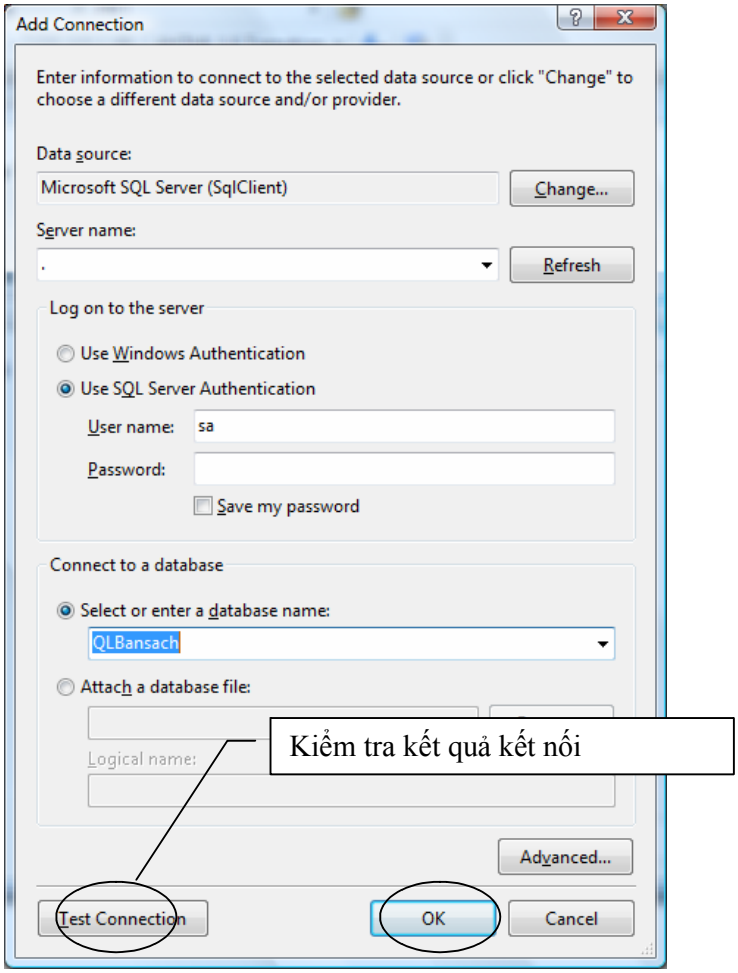
Chọn **Test Query** để kiểm tra kết quả nguồn dữ liệu. ➔ Chọn Finish để hoàn thành

- Với CSDL SQLServer

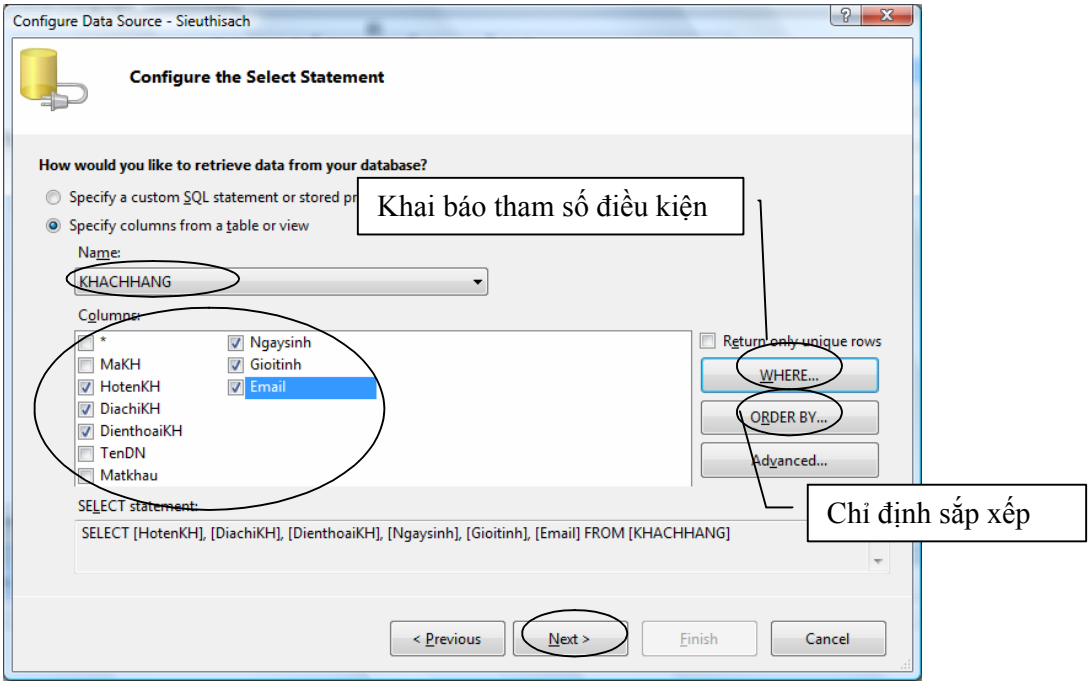


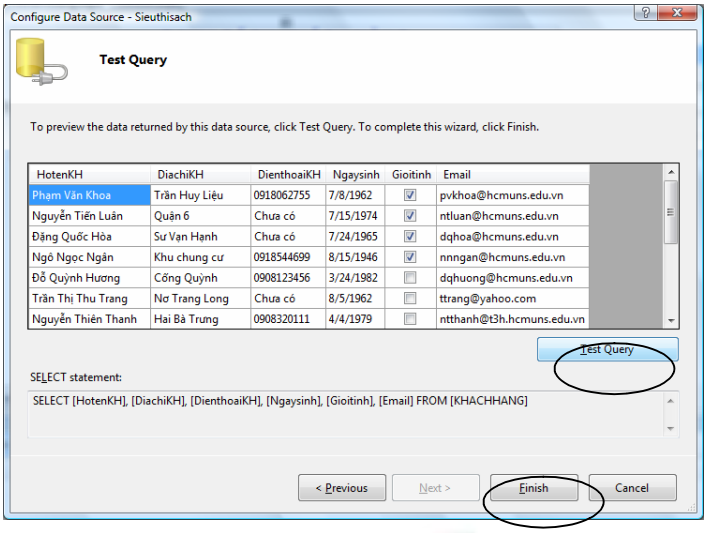


Khai báo các thông số và chọn CSDL cần kết nối



- Chọn Ok
- Xác định nguồn dữ liệu từ Table, Quyery hay câu lệnh truy vấn có thể chỉ định điều kiện lọc dữ liệu và sắp xếp.



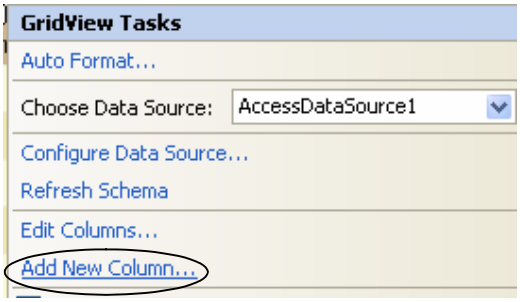


- Chọn **Test Query** để kiểm tra kết quả nguồn dữ liệu. ➔ Chọn Finish để hoàn thành

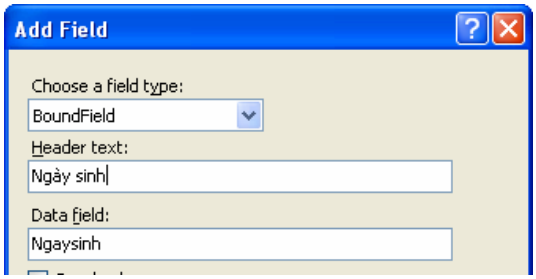
Thi hành xem kết quả:

4. Thêm cột

- Trong cửa sổ DataGrid Task chọn : Add New Column

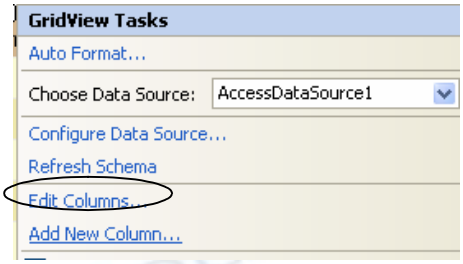


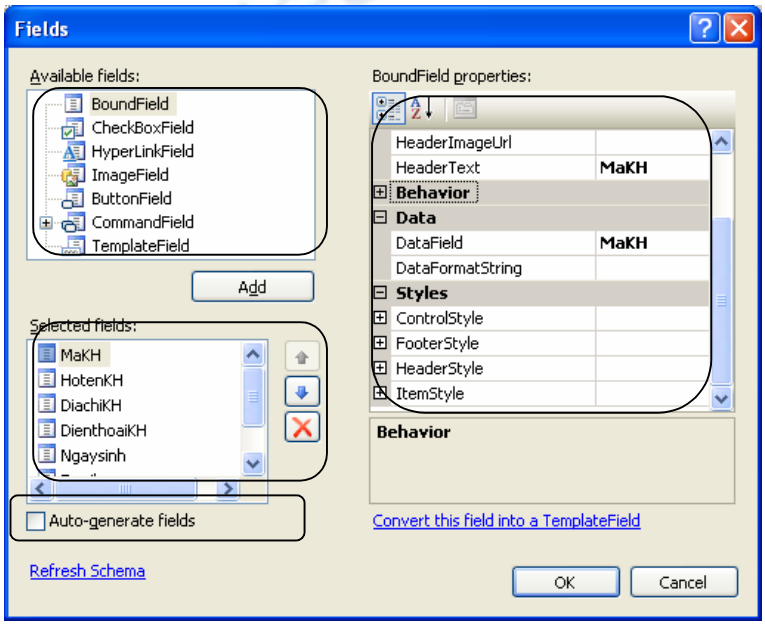
- Chọn Loại field cần tạo : BoundField
- Khai báo tiêu đề côt : Header Text
- Chỉ định tên field dữ liệu: DataField
- Ok hòan thành



5. Hiệu chỉnh, Tạo mới các cột

- Trong cửa sổ DataGrid Task chọn : Edit Column





- AvailableFields: Chọn loại Field liên kết dữ liệu
    - BoundField: Cột có liên kết với nguồn dữ liệu dạng Textbox.
    - Checkbox Field: Cột có liên kết với nguồn dữ liệu dạng Checkbox.
    - Hyperlink Field: Cột có liên kết dữ liệu dạng liên kết.
    - Button Field: Cột dạng nút lệnh
    - CommandField: Cột dạng nút lệnh đã được thiết kế sẵn. Điều khiển GridView cung cấp cho chúng ta 3 loại cột dạng này:
        - Select: Nút lệnh chọn dòng dữ liệu
        - Edit, Cancel, Update: Các nút lệnh hỗ trợ chức năng cập nhật dữ liệu trực tiếp trên lưới.
        - Delete: Nút lệnh xóa dòng dữ liệu

      Chúng ta sẽ có dịp tìm hiểu kỹ hơn về các nút lệnh này trong phần Cập nhật dữ liệu trực tiếp trên lưới.

- TemplateField: Cột do người dùng tự thiết kế. Đây là loại cột có khả năng làm việc khá linh hoạt.

  Ví dụ bạn cần hiển thị danh sách khách hàng. Tại cột Phái, bạn muốn hiển thị dạng Nam/Nữ. Chúng ta sẽ tìm hiểu sâu hơn về Template Column ở phần sau.

- BoundColumn properties: Qui định thông tin chi tiết cho các cột
  - HeaderText, Footer Text: Thông tin tiêu đề trên/dưới của cột
  - Header Image: Hình hiển thị trên tiêu đề cột (thay thế thông tin tiêu đề cột - Header Text).
  - Sort Expression: Biểu thức sắp xếp của cột.
  - Visible: Qui định cột có được hiển thị hay không.
  - DataField: Qui định tên field hay tên thuộc tính của đối tượng dữ liệu cần hiển thị.
  - Data formatting expression: Biểu thức định dạng dữ liệu.

    Mẫu định dạng: **{0:<chuỗi định dạng>}.** Ví dụ:

    + Định dạng số: {0:000.00}, {0:0.##}

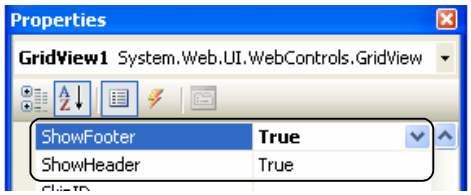
    + Định dạng ngày giờ: {0:dd/MM/yyyy}, {0:hh/mm/ss tt}

- Convert this Field into a Template Column: Chuyển cột hiện hành thành cột dạng Template Column.

6. Thiết lập các thuộc tính định dạng lưới

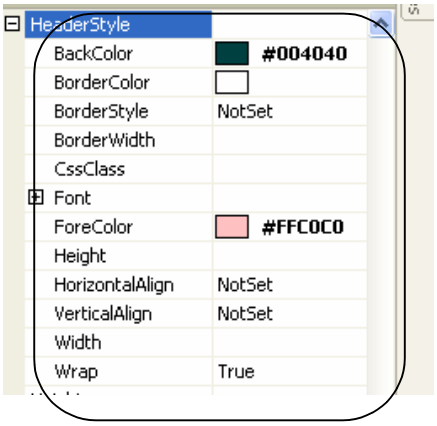
Để thực hiện các thao tác thiết lập các thuộc tính, chúng ta chọn GridView ➔ Properties.

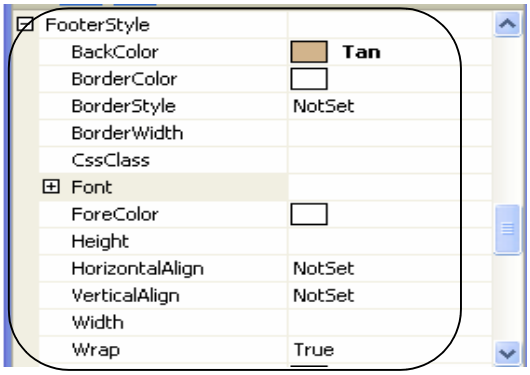
- Hiện / Ẩn : Header / Footer - Phần đầu và chân của GridView



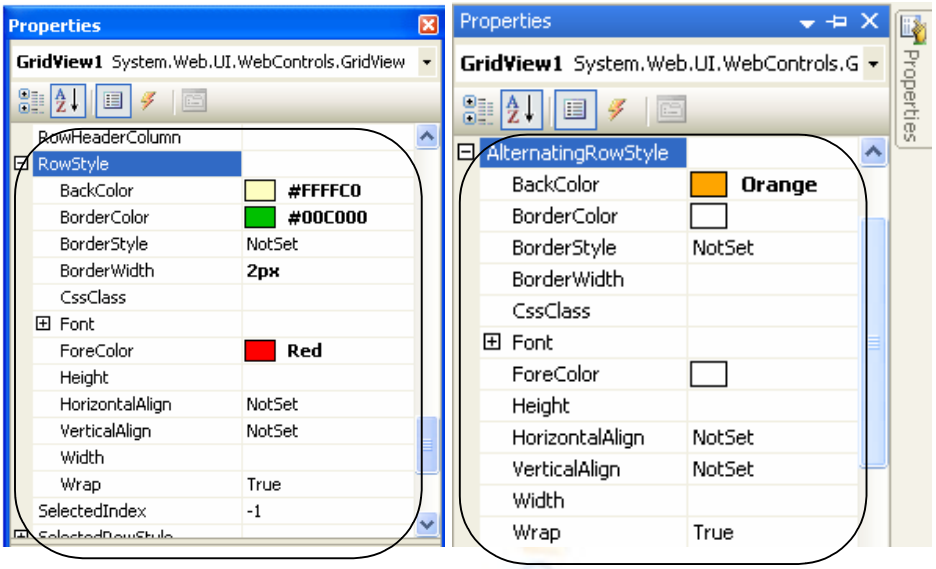
  - Show header: Qui định dòng tiêu đề trên có được phép hiển thị hay không. (mặc định là có hiển thị dòng tiêu đề)
  - Show footer: Qui định dòng tiêu đề dưới có được phép hiển thị hay không. (mặc định là không hiển thị dòng tiêu đề dưới)

- Định dạng dòng Header/Footer: HeaderStyle / FooterStyle

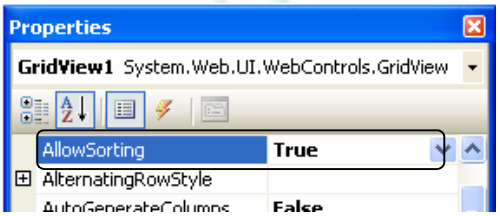




- Định dạng dòng dữ liệu lẽ/chẵn: RowStyleAlternatingRowStyle

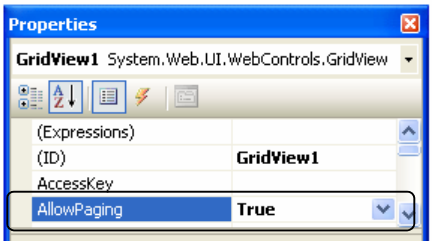


- Sắp xếp: AllowSorting



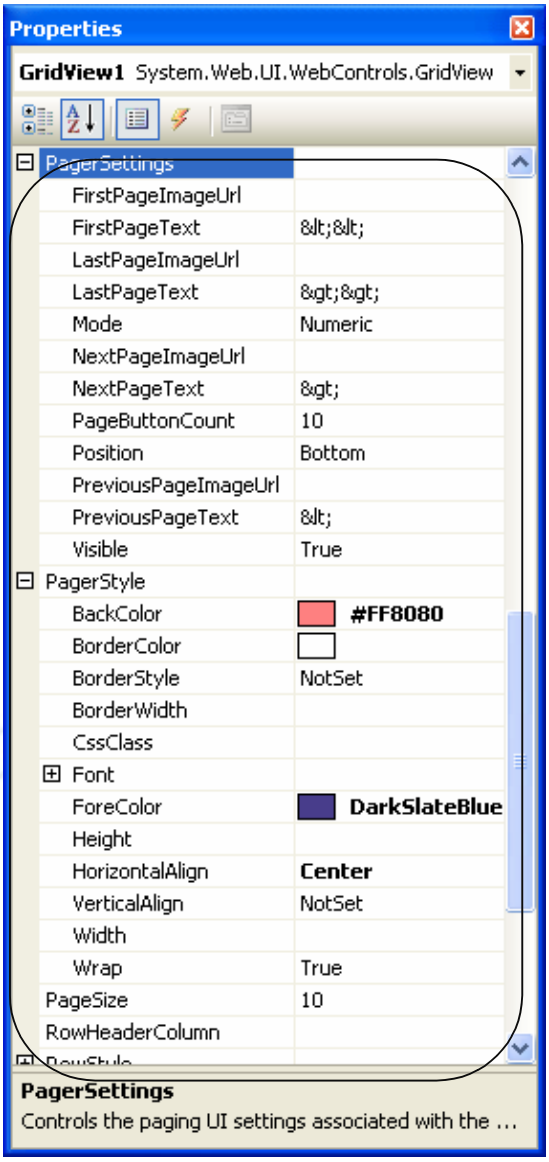
Allow sorting: Có cho phép sắp xếp dữ liệu hay không. (mặc định là không cho phép sắp xếp)

- Phân trang: AlowPaging



  - Định dạng cho phân trang
    - Page size: Qui định số dòng của mỗi trang.
    - Possition: Qui định vị trí hiển thị của bộ nút di chuyển. Ở phía trên thanh tiêu đề, ở phía dưới hay cả hai.
    - Mode: Qui định hình thức hiển thị của bộ nút di chuyển. Hiển thị dạng số trang hay là các chuỗi ký tự đại diện (Next page/Previous page button text). Trong trường hợp hiển thị dạng số, Numeric buttons qui định số nút lệnh được hiển thị tối đa.

Google hiển thị kết quả được phân trang theo dạng số



II. Điều khiển DataList

1. Sử dụng DataList để hiển thị dữ liệu

Như điều khiển Gridview, điều khiển DataList được sử dụng để hiển thị dữ liệu. Tuy nhiên, đối với DataList, chúng ta phải tự thiết kế hình thức hiển thị dữ liệu (giống như Template Column của GridView).

Sử dụng DataList hiển thị thông tin sách

Một số thuộc tính cần chú ý của DataList

- RepeatDirection: Qui định hướng hiển thị dữ liệu
  - Horizontal: Hiển thị dữ liệu theo chiều ngang

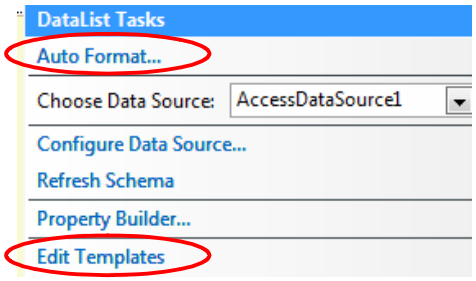
  - Vertical (mặc định): Hiển thị dữ liệu theo chiều đứng

- RepeatColumns: Qui định số cột hiển thị của DataList

RepeatColumns = 3

Thiết kế hình thức hiển thị cho DataList cũng tương tự như thiết kế cho cột Template Column của GridView

Chọn Edit Template từ thực đơn ngữ cảnh để thực hiện thiết kế hình thức hiển thị cho DataList.

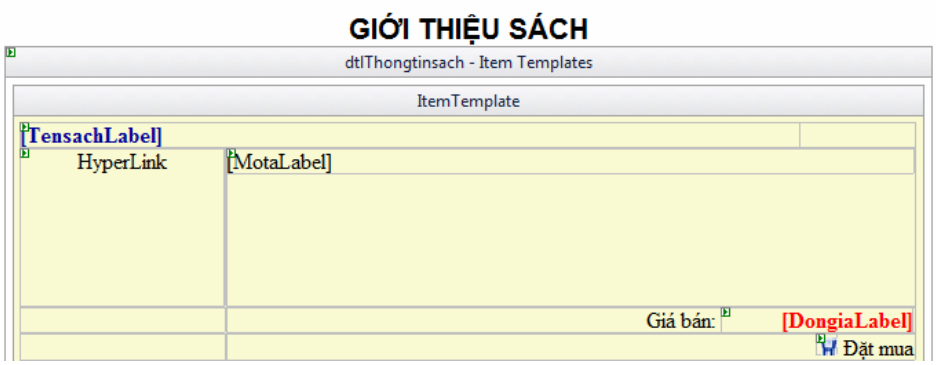


Chọn chức năng thiết kế cho DataList

Ghi chú:

Trong quá trình thực hành, để công việc thiết kế được dễ dàng, các bạn thực hiện thiết kế ở bên ngoài điều khiển DataList. Sử dụng Table để định vị trí hiển thị của các điều khiển. Sau khi hoàn tất công việc thiết kế, chúng ta kéo kết quả đã thiết kế vào vị trí cần hiển thị trong DataList.

Ví dụ: Hiển thị thông tin sách với DataList



Thiết kế thông tin sách với DataList

Kết quả trên trang Web

III. Điều khiển Repeater

Như 2 điều khiển DataList & DataGrid, điều khiển Repeater cũng được dùng để hiển thị dữ liệu. Tuy nhiên, để hiển thị dữ liệu, chúng ta phải tự thiết kế hình thức hiển thị thông qua các tag HTML.

Điều khiển Repeater có các tag sau:

- <HeaderTemplate></HeaderTemplate> (tùy chọn)

  Qui định hình thức hiển thị cho tiêu đề. (Chỉ xuất hiện 1 lần, phía trên của điều khiển)

- <ItemTemplate></ItemTemplate> (Bắt buộc phải có)

  Qui định hình thức hiển thị cho các mục dữ liệu trong điều khiển.

- <AlternatingItemTemplate></AlternatingItemTemplate> (tùy chọn)

  Qui định hình thức hiển thị cho các mục dữ liệu trong điều khiển. Nội dung được qui định trong cặp tag này sẽ hiển thị xen kẽ với các nội dung trong cặp tag

  <ItemTemplate> </ItemTemplate>

- <SeparatorTemplate></SeparatorTemplate> (tùy chọn)

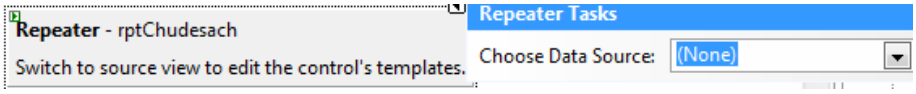
Qui định hình thức hiển thị giữa các dòng dữ liệu

- <FooterTemplate></FooterTemplate> (tùy chọn)

  Qui định hình thức hiển thị cho tiêu đề dưới. (Chỉ xuất hiện 1 lần, phía dưới của điều khiển)

Ví dụ:

Bước 1. Tạo mới điều khiển Repeater: rptChudesach vào trang Web.



Điều khiển rptChudesach trên trang Web

Bước 2. Kết nối và tạo nguồn dữ liệu cho điều khiển từ Table Chude với các cột: MaCD, Tenchude.

Bước 3:. Chuyển qua xem trang Web dưới dạng HTML

```
<asp:Repeater ID="rptChudesach" runat="server" DataSourceID="SqlDataSource1">

</asp:Repeater>
```

Bước 3. Bổ sung các tag sau

```
<asp:Repeater ID="rptChudesach" runat="server" DataSourceID="SqlDataSource1">
        <HeaderTemplate>
            <table border="1"  cellspacing="0">
            <tr>
            <td width="60" align="center">
            <strong>Mã CĐ</strong>
            </td>
            <td width="150" align="center">
            <strong>Tên chủ đề </strong>
            </td>
            </tr>
            </HeaderTemplate>
            <ItemTemplate>
            <tr>
            <td>
            <%#  Eval("MaCd")      %>
            </td>
            <td>
            <%# Eval("Tenchude")%>
            </td>
            </tr>
            </ItemTemplate>
            <AlternatingItemTemplate>
            <tr bgcolor="GhostWhite">
            <td>
            <%# Eval("MaCD")%>
            </td>
            <td>
            <%# Eval("TenChude")%>
            </td>
            </tr>
            </AlternatingItemTemplate>
            <FooterTemplate>
            </table>
            </FooterTemplate>

        </asp:Repeater>
```

Bước 4. Xem lại màn hình thiết kế và thực thi kết quả.

Mã CĐ	Tên chủ đề
1	Ngoại ngữ
2	Công nghệ thông tin
3	Luật
4	Văn học
5	Khoa học kỹ thuật
6	Nông nghiệp
7	Triết học-Chính trị
8	Lịch sử, địa lý
9	Kinh tế
10	Sách giáo khoa
11	Nghệ thuật sống
12	Du lịch
13	Kế toán

V. Sử dụng DetailView và FormView

Hai điều khiển này cho phép bạn làm việc với một trường dữ liệu đơn tại mỗi thời điểm Cả hai điều khiển này cho phép bản thay đổi, thêm mới hay xoá dữ liệu như một bản ghi cơ sở dữ liệu, và nó cho phép bạn chuyển sang trang tiếp theo hay quay lại trang trước thông qua thiết lập dữ liệu.

1. DetailView

a. Hiển thị dữ liệu với DetailView

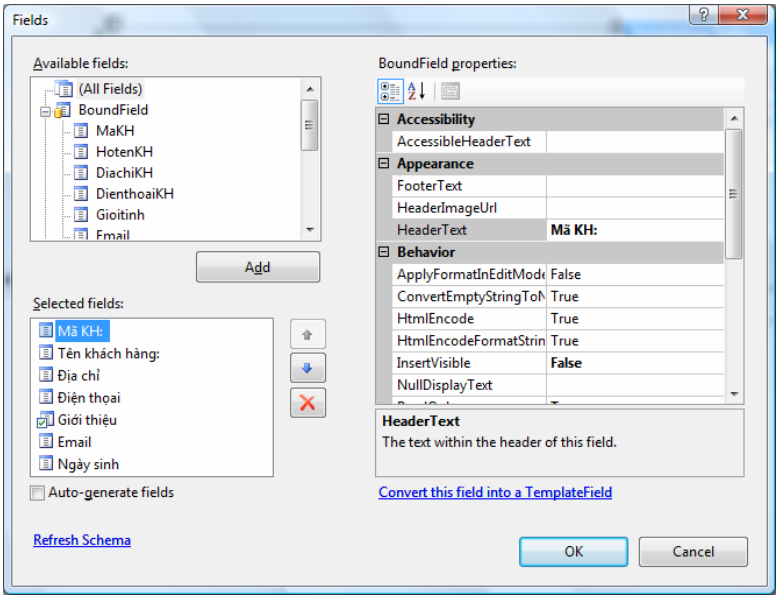
DetailView được đưa ra hiển thị như một bảng(<Table>) trong HTML để hiển thị dữ liệu một bản ghi.

Ví dụ: Trang XemthongtinKH.aspx

b. Sử dụng Fields với điều khiển DetailView

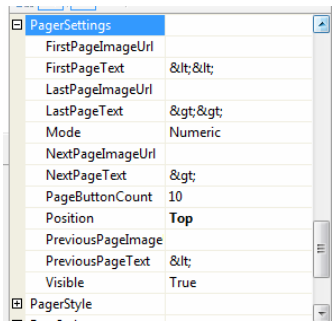
DetailView hỗ trợ tất cả các Field như GridView

- BoundField: cho phép bạn hiển thị giá trị của dữ liệu như Text
- CheckBoxField: hiển thị dữ liệu dưới dạng một CheckBox
- CommandField: hiển thị liên kết cho phép chỉnh sửa, thêm mới, xoá dữ liệu.
- ButtonField: hiển thị dữ liệu như một button(ImageButton,)
- HyperLinkField: hiển thị môt liên kết
- ImageField: hiển thị ảnh
- TemplateField: cho phép hiển thị các điều khiển tuỳ biến.



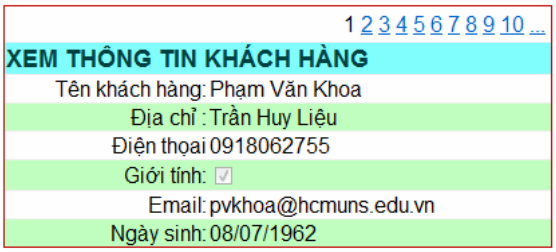
c. Xử lý phân trang với điều khiển DetailView

Để tạo phân trang chỉ định thuộc tính AllowPaging=”true” cho điều khiển DetailView. Với các thuộc tính định dạng thuộc nhóm thuộc tính: Pagersettings



d. Minh họa:

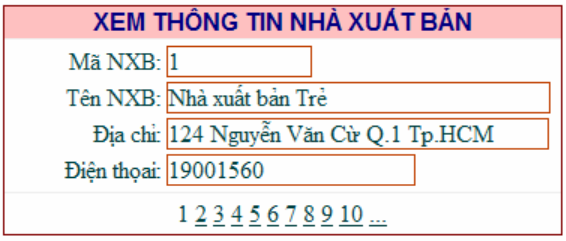
Trong ví dụ trên bạn liên kết dữ liệu với Table Khachhang đưa vào 5 BoundField và một CheckBoxField, điền vào dữ liệu với thuộc tính DataField và thiết đặt cho nó tiêu đề với HeaderText. Tạo phân trang và định dạng trình bày tại góc trên bên phải. Kết xuất của chương trình



2. DetailView

a. Hiển thị dữ liệu với FormView

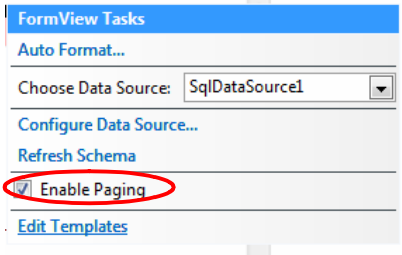
FormView được đưa ra hiển thị dữ liệu với các điều khiển tùy biến để hiển thị dữ liệu một bản ghi. Ví dụ: Trang XemthongtinNXB.aspx



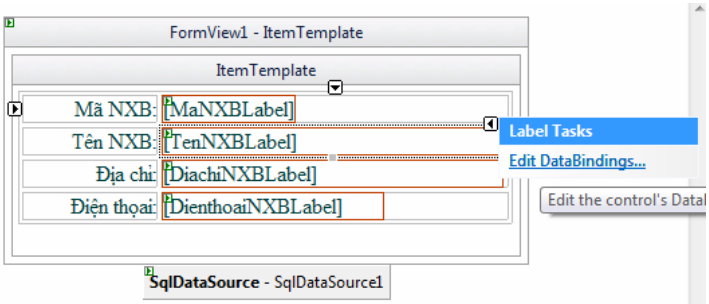
b. Trình bày dữ liệu sử dụng Edit Template

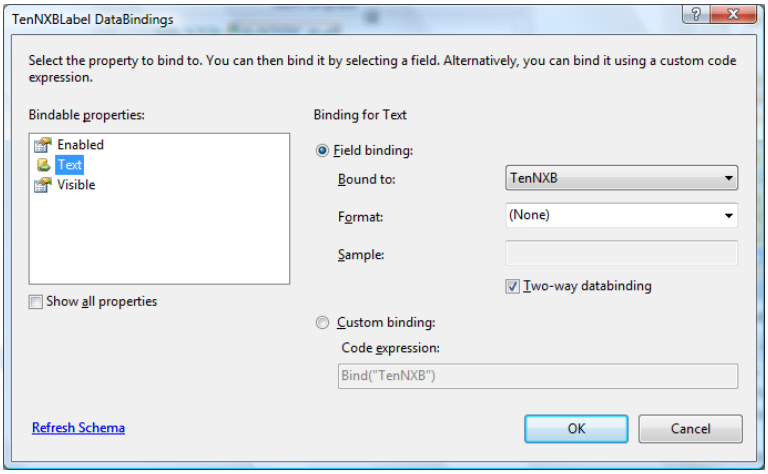
FormView hỗ trợ trình bày dữ liệu dạng tùy biến cho tất cả các Field DataList

- Tạo FormView vào trang và lien kết dữ liệu
  - Tại cửa sổ thiết kế chọn Edit Tempalte từ cửa sổ FormView Task



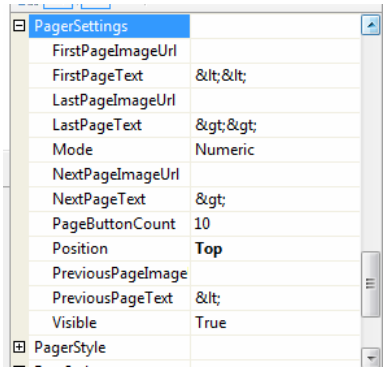
- Thiết kế trình bày với các Control điều khiển tương tự như khi thiết kế DataList với các điều khiển tùy biến





c. Xử lý phân trang với điều khiển DetailView

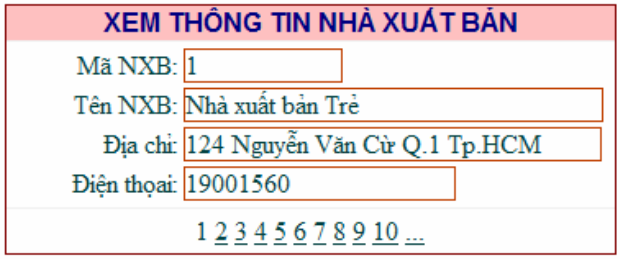
Để tạo phân trang chỉ định thuộc tính AllowPaging=”true” cho điều khiển DetailView. Với các thuộc tính định dạng thuộc nhóm thuộc tính: Pagersettings



d. Minh họa:

Trong ví dụ trên bạn liên kết dữ liệu với Table NhaXuatBan đưa vào 4 Label, điền vào dữ liệu với thuộc tính Text cho các File tương tứng và thiết đặt cho nó tiêu dề với HeaderText. Tạo phân trang và định dạng trình bày tại phía dưới giữa trang

Kết xuất của chương trình



Chương 5

XỬ LÝ DỮ LIỆU VỚI ADO.NET

I. Tìm hiểu về ADO.NET

Hầu hết ứng dụng hay các website đều cần phải có cơ sở dữ liệu, để lưu trữ dữ liệu, xử lý thông tin và đưa ra các báo cáo, hỗ trợ tìm kiếm… Khi dữ liệu trở thành trung tâm của ứng dụng thì cung cấp các chức năng tới người dùng phụ thuộc vào khả năng thao tác dữ liệu, vấn đề mà người thiết kế và người xây dựng ứng dụng quan tâm khi sử dụng dữ liệu là:

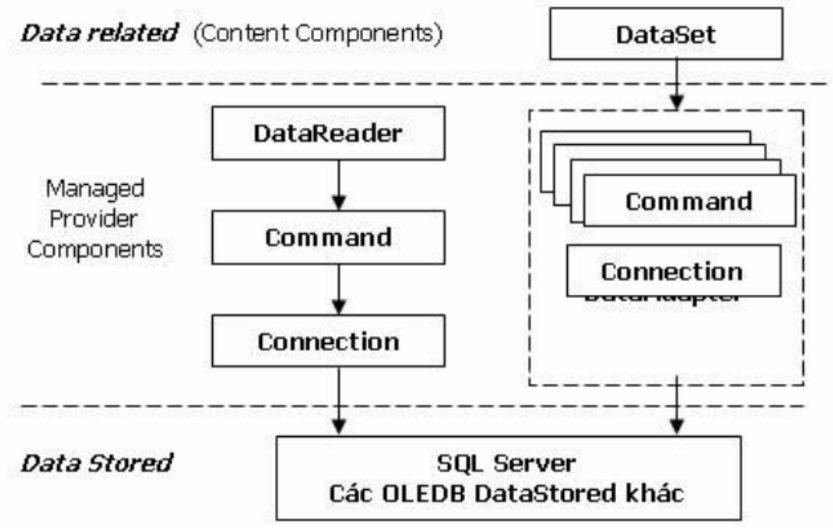
- Lưu dữ liệu tập trung
- Đảm bảo toàn vẹn dữ liệu
- Đảm bảo khả năng truy xuất đồng thời của nhiều người dùng trên dữ liệu
- Đảm bảo thời gian hồi đáp ngắn cho mỗi người dùng
- Bảo mật dữ liệu
- Trao đổi dữ liệu giữa các hệ thống khác nhau

Những vấn đề này được giải quyết dựa vào khả năng của các hệ quản trị cơ sở dữ liệu(HQT CSDL) và các phần mềm xử lý dữ liệu do HQT CSDL cung cấp.

.Net truy xuất dữ liệu qua ADO.NET, đặc điểm chính của ADO.NET là khả năng làm việc với dữ liệu không kết nối, dữ liệu được lưu trữ trong bộ nhớ như một csdl thu nhỏ gọi là dataset, nhằm tăng tốc độ tính toán, xử lý tính toán và hạn chế sử dụng tài nguyên trên Database Server.

Đặc điểm quan trọng thứ 2 là khả năng xử lý dữ liệu chuẩn XML, dữ liệu ở dạng XMl có thể trao đổi giữa bất kỳ hệ thống nào nên ứng dụng của bạn sẽ có nhiều khả năng làm việc với nhiều ứng dụng khác.

1. Kiến trúc ADO .Net



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

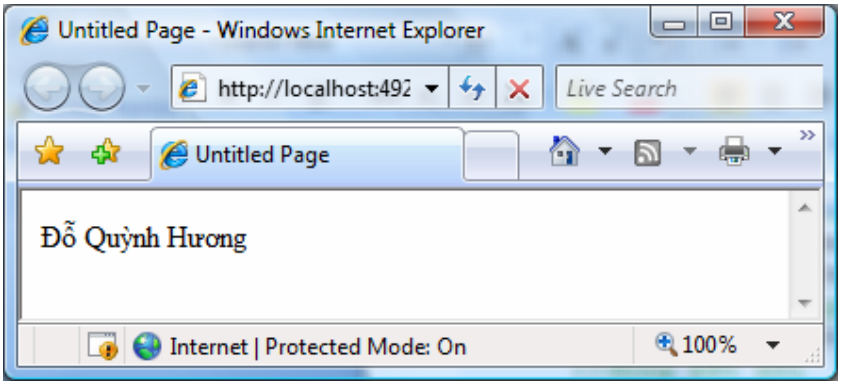
- **Managed Provider Component:** bao gồm các đối tượng như DataAdapter, DataReader,… giữ nhiệm vụ làm việc trực tiếp với dữ liệu như database, file,…
- **Content Component:** bao gồm các đối tượng như DataSet, DataTable,… đại diện cho dữ liệu thực sự cần làm việc.

- 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.

2. Minh họa tạo kết nối cơ sở dữ liệu

```
using System;
using System.Data;
using System.Data.SqlClient;
public partial class vd1 : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
    //Khai báo và khởi tạo biến Connection
      SqlConnection cnn = new SqlConnection("Data Source=(local);
                                            Initial Catalog=QLbansach;User ID=sa;Password=");
    //Mở kết nối
      cnn.Open();
    //Command điều khiển truy vấn sql
      SqlCommand cmd = cnn.CreateCommand();
      cmd.CommandText = "select HotenKH from Khachhang where MaKH=5";
    //lấy về chuỗi giá trị trong cơ sở dữ liệu
      string result = (string)cmd.ExecuteScalar();
    //đóng kết nối
      cnn.Close();
    //in giá trị ra màn hình
      Response.Write(result);
   }
}
```

Thi hành



Cơ bản các bước thực hiện với database

- Bước 1: Tạo kết nối
- Bước 2: Mở kết nối dữ liệu
- Bước 3: Tạo lệnh điều khiển truy vấn SQL
- Bước 4: Thực thi lệnh
- Bước 5: Đóng kết nối
- Bước 6: in kết quả

II. Các đối tượng trong ADO.Net

1. Đối tượng Connection

Vai trò của Connection trong ADO.net là tạo kết nối giữa ứng dụng với nguồn dữ liệu(CSDL)

Data Provider :

- System.Data.Oledb : Sử dụng với Access
- System.Data.SqlClient : Sử dụng với SQLServer

Ứng với mỗi tên miến ta có một connection tương ứng:

- System.Data.Oledb.OledbConnection
- System.Data.SqlClient.SqlConnection

Ngòai ra Ado.net còn hỗ trợ các Data Provider khác như

- System.data.OcracleClient : Dành cho Ocracle
- MicroSoft.data.Odbc : Dành cho dạng kết nối thong qua Odbc Connectionủa Hệ điều hành
- Microsoft.Data.Sqlxml: Dành cho XML trên Sqlserver

Connection String

Trước khi thực hiện kết nối Connectionần khai báo các thong tin cho Connection thông qua thuộc tính Connection String. Cách khai báo thay đổi tùy thuộc vào Data Provider. Gồm có các thành phần sau:

- **Nếu kết nối với CSDL Access**

  Provider: Khai báo Data Provider Connectionủa Hệ QT CSDL Access

  Data Source: Nguồn dữ liệu (Tên CSDL.mdb)

  User ID: Tên người dùng

  Password : Mật khẩu

Ví dụ: Tạo kết nối với CSDL Access

```
using System;
using System.Data;
using System.Data.OleDb;
public partial class VD2 : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
      //Khai báo và khởi tạo biến Connection
       String    StrCnn="Provider=Microsoft.Jet.OLEDB.4.0;
                                          Data Source=" + Server.MapPath("~/App_Data/QLBansach.mdb");
      OleDbConnection  cnn = new OleDbConnection (StrCnn);
      //Mở kết nối
      cnn.Open();
      //Command điều khiển truy vấn sql
      OleDbCommand  cmd = cnn.CreateCommand();
      cmd.CommandText = "select HotenKH from Khachhang where MaKH=2";
      //lấy về chuỗi giá trị trong cơ sở dữ liệu
      string result = (string)cmd.ExecuteScalar();
      //đóng kết nối
      cnn.Close();
      //in giá trị ra màn hình
      Response.Write(result);
   }
}
```

- **Nếu kết nối với CSDL SQLServer**

  Provider: Khai báo Data Provider Connectionủa Hệ QT CSDL SQLServer

  Data Source:Tên máy cài đặt SQLServer

  Initial Catalog: Tên CSDL

  User ID: Tên người dùng

  Password : Mật khẩu

Các thuộc tính Của Connection

- DataBase: : Tương ứng với Initial Catalog(SQL) hay tên CSDL muốn làm việc (Access)
- DataSource: Tương ứng với DataSource Tên máy SQL hay tên CSDL
- Provider: Tương ứng với Provider
- State: Tình trạng kết nối Connectionủa Connection với các giá trị
  - Broken: Kết nối đã bị ngắt chỉ xảy ra sau khi đã kết nối
  - Closed: Kết nối đã đóng
  - Connecting: Đang kết nối
  - Executing: Kết nối đang thực hiện một lệnh
  - Fetching: Kết nối đang truy xuất dữ liệu
  - Open: Kết nối đang mở

Các phương thức

Change Databse: Thay đổi DataBase làm việc

Close : Đóng kết nối sử dụng đóng Connection đang mở

Dispose: Xóa tòan bộ tài nguyên liên quan đến Connection trên vùng nhớ.

Open: Thực hiện kết nối Connectionới các thông tin đã khai báo trong ConnectionString

Ví dụ: Kiểm tra kết nối với CSDL SQLServer

```
protected void Button1_Click(object sender, EventArgs e)
{
   //Khai báo và khởi tạo biến Connection
   SqlConnection cnn = new SqlConnection("Data Source=;Initial Catalog=QLbansach;User ID=sa;Password=");
   //Mở kết nối
   cnn.Open();
   TextBox1.Text = "State = " + cnn.State; ;
   // Thực hiện các câu lệnh SELECT, INSERT, DELETE, UPDATE.
   cnn.Close();
   //Đóng kết nối

}
```