PHÁT TRIỂN ỨNG DUNG CSDL 1

Tháng 9/2014

LẬP TRÌNH ỨNG DỤNG VỚI NGÔN NGỮ C#

Lập trình giao diện với Windows Forms

Tóm tắt nội dung bài thực hành:

Hướng dẫn tạo và lập trình giao diện Windows Forms đơn giản bằng ngôn ngữ C#

Bộ môn **Hệ thống thông tin**

Khoa Công nghệ thông tin

ĐH Khoa hoc tư nhiên TP HCM

MỤC LỤC

1 Mục tiêu

Sau khi hoàn thành bài tập này sinh viên có thể:

- Tạo ứng dụng Windows Forms đơn giản.
- Làm quen với một số control đơn giản.
- Làm quen với một số control thể hiện dữ liệu.

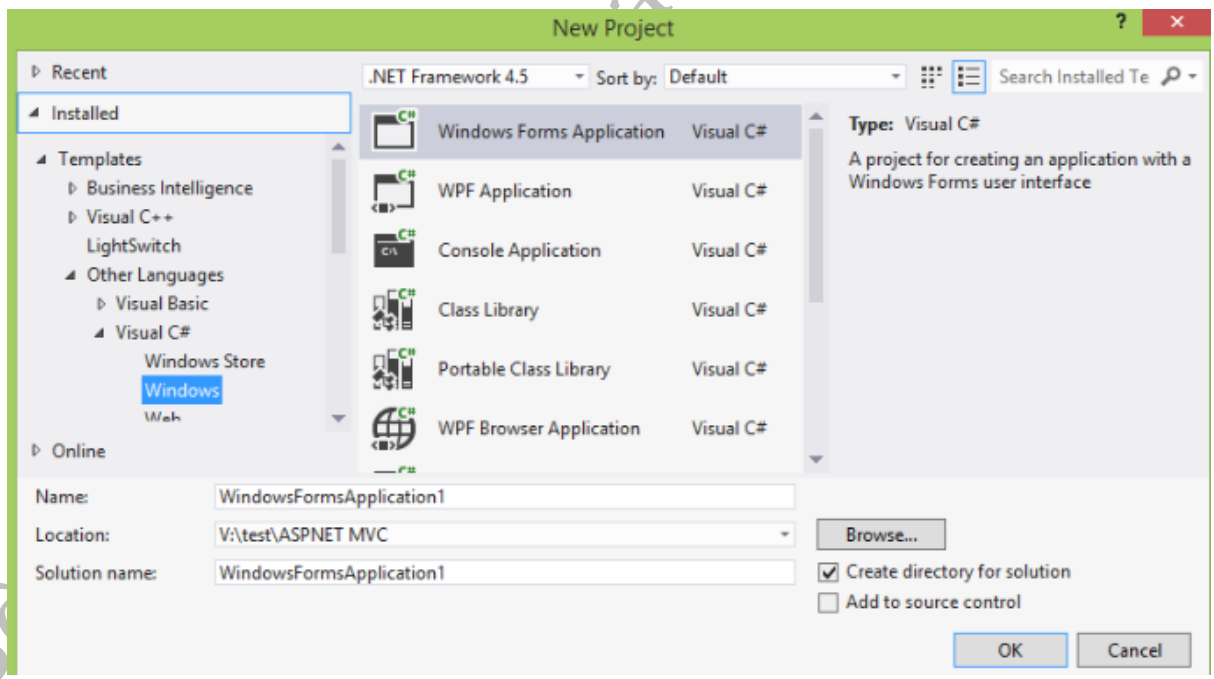
Công cụ sử dụng:

- Visual Studio: sinh viên có thể sử dụng tài khoản MSDNAA để tải và cài đặt ứng dụng visual studio.

2 Ứng dụng Windows Forms đầu tiên

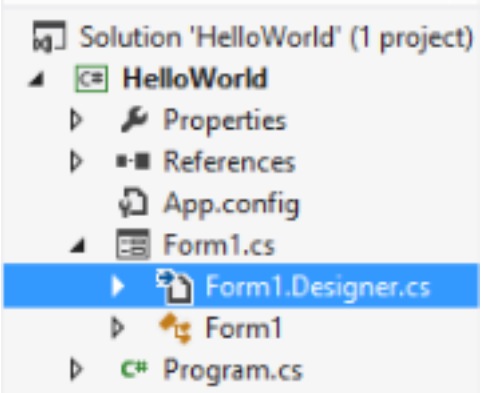
2.1 Tạo project Windows Forms

Để tạo project Windows Forms, tạo project dạng Windows Forms Application bằng ngôn ngữ C#.



2.2 Cấu trúc một chương trình Windows Forms

Khi mới tạo xong, một chương trình Windows Forms sẽ bao gồm các thành phần quan trọng sau:



– Tập tin Program.cs: đây là mã nguồn chứa hàm main của chương trình

```
static void Main()
{
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new Form1());
}
```

Mặc định, mã nguồn hàm main sẽ khởi tạo chương trình với giao diện Form1.

– Các tập tin xử lý Form1 bao gồm:

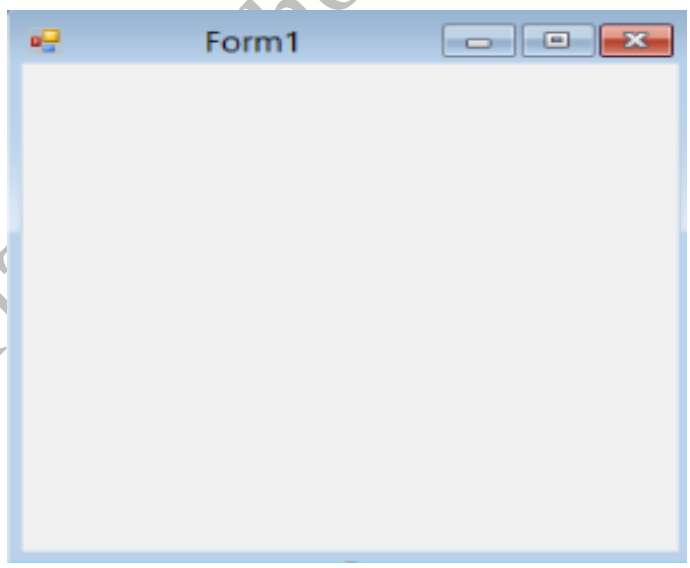
– Form1.cs: chứa mã nguồn xử lý do người dùng định nghĩa thêm. Mặc định, ban đầu Form1.cs

```
public partial class Form1 : Form
{
      public Form1()
      {
            InitializeComponent();
      }
}
```

– Form1.designer.cs: chứa mã nguồn về giao diện. Tại đây, từ khoá partial cho phép định nghĩa lớp Form1 được đặt trong cả 2 tập tin Form1.cs và Form1.designer.cs. Lúc này, mã nguồn giao diện chưa có phần nào thêm. Phần mã nguồn trong mục "Windows Forms Designer generated code" thể hiện phần mã nguồn giao diện Form1 do Visual Studio tự phát sinh thêm vào.

```
private System.ComponentModel.IContainer components = null;
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font;
            this.Text = "Form1";
        }
        #endregion
}
```

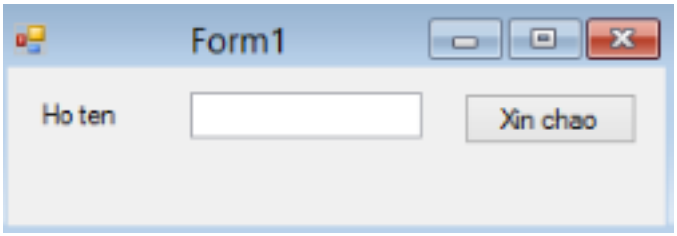
– Tập tin App.config: tập tin này cho phép định nghĩa các thuộc tính cấu hình cho ứng dụng. Có thể sử dụng để lưu trữ thông tin về kết nối cơ sở dữ liệu,...



2.3 Lập trình giao diện đơn giản

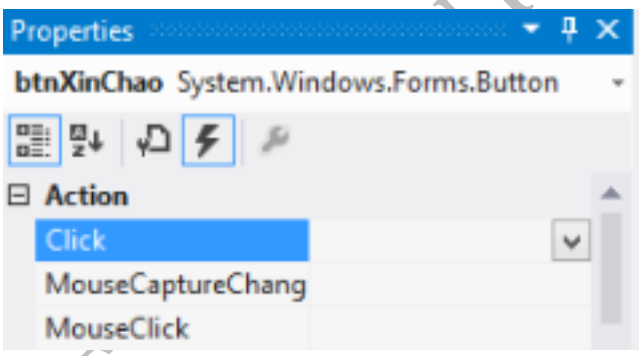
Sử dụng View/Toolbox để hiện các controls có trong Windows Forms.

Lúc này, lần lượt kéo thả các control vào giao diện Form1 như sau



Để tạo giao diện trên, lần lượt kéo control Label, Textbox và Button vào trong giao diện. Sau đó, thay đổi các thuộc tính Text tương ứng. Ngoài ra, thay đổi thuộc tính ID của các control thành: lblHoTen(cho label), txtHoTen(cho textbox) và btnXinChao(cho button). Có thể thay đổi bằng cách mở cửa sổ View/Properties và chọn vào control tương ứng.

Sau đó, chọn vào button Xin chao, chọn tab Event trên cửa sổ properties. Double Click vào sự kiện Click để phát sinh mã nguồn xử lý sự kiện Click.



Hàm xử lý sự kiện click cho button Xin Chao sẽ được phát sinh trong tập tin Form1.cs. Sau đó, cài đặt thêm chức năng lấy thông tin từ textbox Ho ten và xuất ra màn hình chuỗi "Xin chào " + <hoTen>.

```
private void btnXinChao_Click(object sender, EventArgs e)
{
      string hoTen = this.txtHoTen.Text;
      MessageBox.Show(hoTen);

}
```

Chạy chương trình để kiểm chứng lại cách hoạt động.

2.4 Nguyên tắc hoạt động của giao diện Windows Forms

Với Windows Forms, tất cả các thành phần giao diện như Form và các control như: textbox, button, label, gridview,... đều là các lớp. Trong đó, khi thêm một control vào trong giao diện thì các lớp đó sẽ được tự động phát sinh và thêm vào trong lớp Form tại tập tin Form.designer.cs.

Mã nguồn tập tin sau khi thêm label, textbox và button sẽ được thêm vào một số phần sau:

- Form1 chứa các thuộc tính là lớp button, textbox và label với tên là ID của các control đó.

```
private System.Windows.Forms.Button btnXinChao;
private System.Windows.Forms.TextBox txtHoTen;
private System.Windows.Forms.Label lblHoTen;
```

- Hàm InitializeComponent: đây là hàm sẽ được gọi khi khởi tạo Form1. Hàm này được sử dụng để khởi tạo các control đã khai báo cũng như khởi tạo cho giao diện của Form1. Đầu tiên, các thuộc tính control sẽ được khai báo. Sau đó, các thuộc tính của từng controls sẽ được khởi tạo như thiết lập lại Name, Text, và Location (vị trí của control trên giao diện Form1). TabIndex là thứ tự của các control khi người dùng nhấn tab.

```
private void InitializeComponent()
{

        this.btnXinChao = new System.Windows.Forms.Button();
        this.txtHoTen = new System.Windows.Forms.TextBox();
        this.lblHoTen = new System.Windows.Forms.Label();
        this.SuspendLayout();
        //
        // btnXinChao
        //
        this.btnXinChao.Location = new System.Drawing.Point(197, 11);
        this.btnXinChao.Name = "btnXinChao";
        this.btnXinChao.Size = new System.Drawing.Size(75, 23);
        this.btnXinChao.TabIndex = 0;
        this.btnXinChao.Text = "Xin chao";
        this.btnXinChao.UseVisualStyleBackColor = true;
        this.btnXinChao.Click += new
System.EventHandler(this.btnXinChao_Click);
        //
        // txtHoTen
        //
        this.txtHoTen.Location = new System.Drawing.Point(79, 11);
        this.txtHoTen.Name = "txtHoTen";
        this.txtHoTen.Size = new System.Drawing.Size(100, 20);
        this.txtHoTen.TabIndex = 1;
        //
        // lblHoTen
        //
        this.lblHoTen.AutoSize = true;
        this.lblHoTen.Location = new System.Drawing.Point(12, 14);
        this.lblHoTen.Name = "lblHoTen";
        this.lblHoTen.Size = new System.Drawing.Size(39, 13);
        this.lblHoTen.TabIndex = 2;
        this.lblHoTen.Text = "Ho ten";
        //
        // Form1
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(284, 69);
        this.Controls.Add(this.lblHoTen);
        this.Controls.Add(this.txtHoTen);
        this.Controls.Add(this.btnXinChao);
        this.Name = "Form1";
        this.Text = "Form1";
        this.ResumeLayout(false);
        this.PerformLayout();
}
```

Lớp Form1 có thuộc tính Controls chứa các control có trong Form1. Trong ví dụ trên, đoạn mã sau dùng để thêm label, textbox và button vào trong giao diện của Form1.

```
this.Controls.Add(this.lblHoTen);
this.Controls.Add(this.txtHoTen);
this.Controls.Add(this.btnXinChao);
```

Để xử lý sự kiện nhấn chuột của button Xin chao, Windows Forms phát sinh dòng lệnh sau trong hàm InitializeComponent:

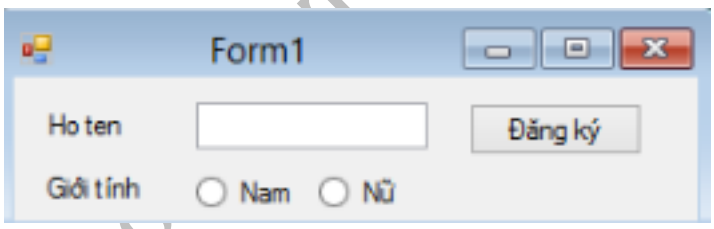
```
this.btnDangKy.Click += new System.EventHandler(this.btnXinChao_Click);
```

Trong đó, this.btnXinChao_Click là tên hàm xử lý sự kiện trong lớp Form1 ở tập tin Form1.cs

3 Một số controls phổ biến

Trong Windows Forms, rất nhiều các control thông dụng được hỗ trợ như: checkbox, combobox, radiobutton,...

3.1 Tạo thêm các lựa chọn với RadioButton



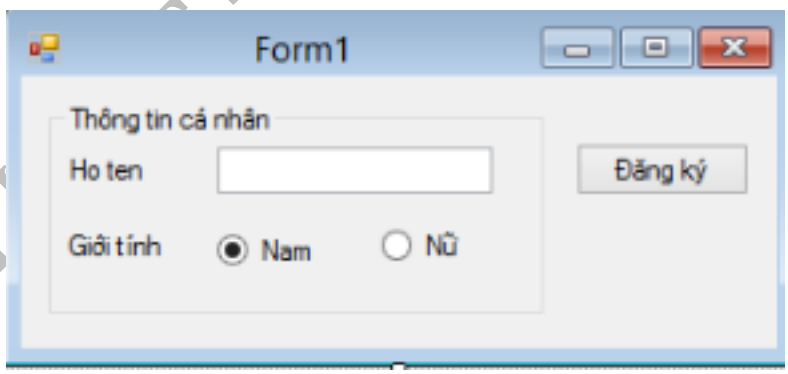
Lần lượt kéo 2 radio button vào giao diện và đặt tên như trên hình. Đồng thời, đổi ID của các control thành lblGioiTinh, rdoNam và rdoNu và thiết lập rdoNam thành lựa chọn mặc định bằng cách thiết lập thuộc tính Checked thành True.

Lập trình nút btnDangKy để hiển thị thông tin cá nhân hiện tại.

```
private void btnDangKy_Click(object sender, EventArgs e)
{
        string hoTen = txtHoTen.Text;
        string gioiTinh = null;

        if (rdoNam.Checked)
        {
                gioiTinh = "Nam";
        }
        else
        {
                gioiTinh = "Nữ";
        }

        StringBuilder builder = new StringBuilder();
        builder.AppendFormat("Họ tên: {0}", hoTen);
        builder.AppendLine();
        builder.AppendFormat("Giới tính: {0}", gioiTinh);

        MessageBox.Show(builder.ToString());

}
```

3.2 Nhóm các controls bằng các Container

Trong Windows Forms, các control có thể được nhóm lại và đặt trong các control khác để bố trí giao diện trực quan hơn. Trong hình bên dưới, giao diện được nhóm lại bằng groupbox giúp giao diện trong dễ nhìn hơn.

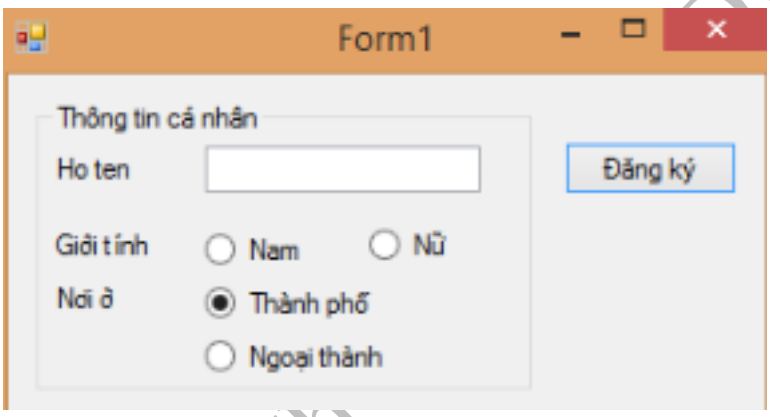


Các control dạng container luôn có thuộc tính Controls tương tự như lớp Form để thêm các control vào. Sau khi kéo các control vào trong groupbox, đoạn mã sau sẽ được phát sinh trong tập tin Form1.designer.cs:

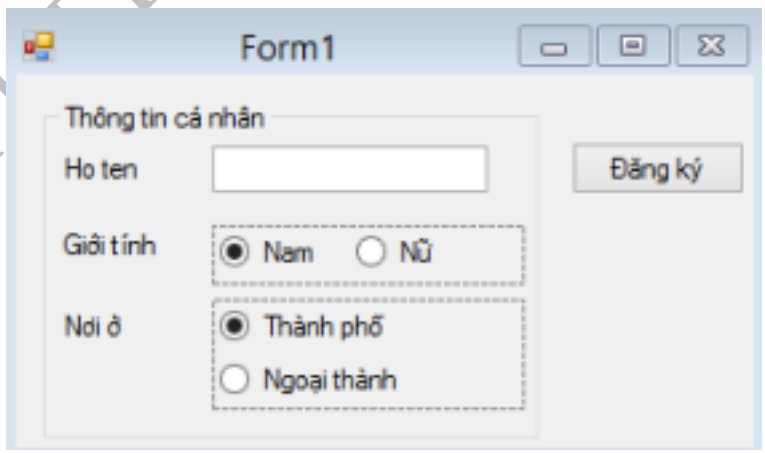
```
this.gboThongTinCaNhan.Controls.Add(this.rdoNu);
this.gboThongTinCaNhan.Controls.Add(this.lblHoTen);
this.gboThongTinCaNhan.Controls.Add(this.txtHoTen);
this.gboThongTinCaNhan.Controls.Add(this.lblGioiTinh);
this.gboThongTinCaNhan.Controls.Add(this.rdoNam);
```

Ngoài ra, có thể sử dụng một số container khác như: Panel, Split Container, Tab Control,...

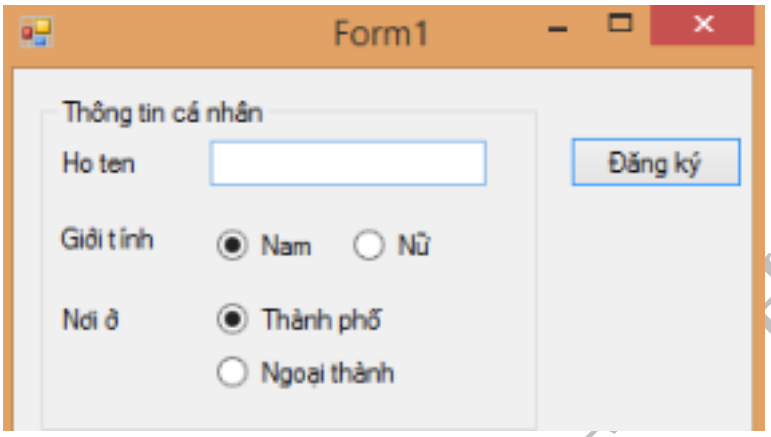
Bên cạnh vai trò thể hiện thông tin trên giao diện, các lớp container còn đóng vai trò nhóm các radio button. Trong giao diện sau, chỉ có 1 trong số 4 radio button có thể được chọn.



Để giải quyết tình trạng này, cần đặt các nhóm radio button vào trong các container khác nhau. Trong ví dụ sau, sử dụng Panel để nhóm giới tính và nơi ở vào 2 panel khác nhau.

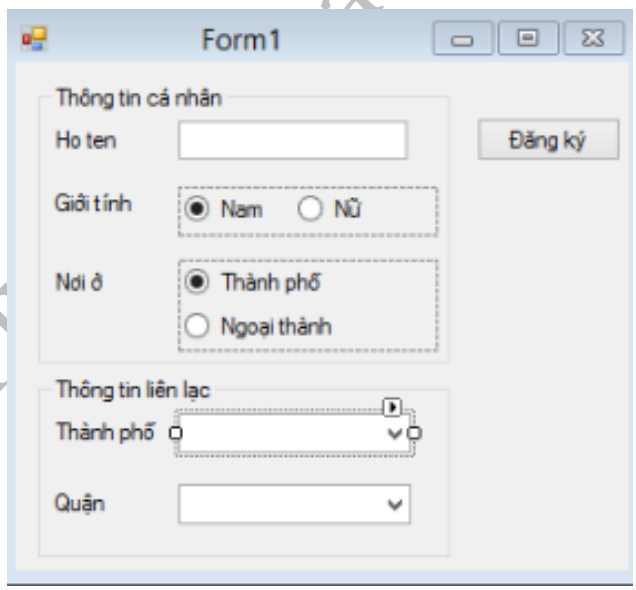


Khi chương trình dạy, phần viền của panel sẽ không được hiển thị nên sẽ cho giao diện tương tự như khi không có panel. Tuy nhiên, lúc này chúng ta có thể chọn giới tính và nơi ở riêng.

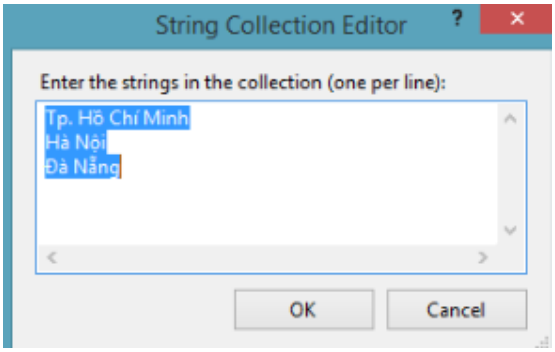


3.3 Thêm các lựa chọn trong danh sách dài bằng ComboBox

ComboBox cho phép chọn một lựa chọn từ danh sách nhiều lựa chọn hơn radio button.



Để thêm lựa chọn vào trong combo box, thay đổi giá trị thuộc tính Items trong properties của combo box cboThanhPho vừa thêm vào.

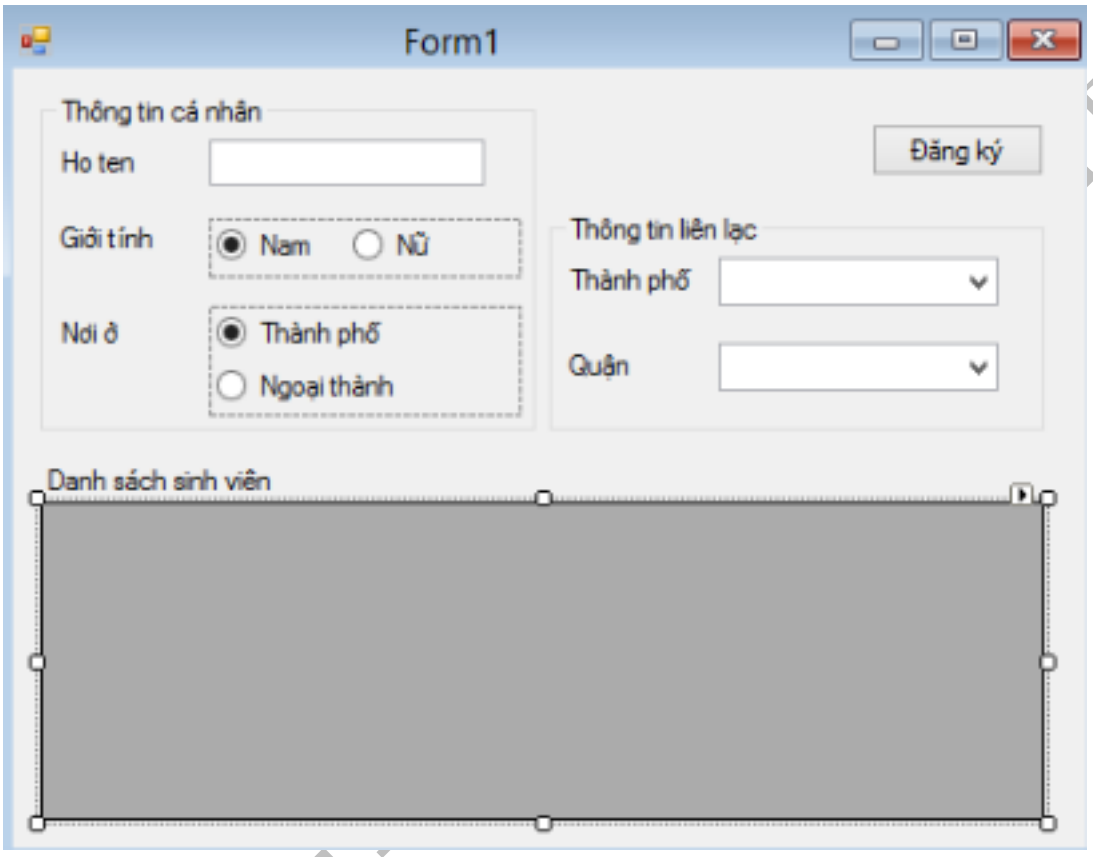


Có thể xử lý sự kiện SelectedIndexChanged để xử lý cho các hành động sau khi người dùng thay đổi lựa chọn trong combo box. Ví dụ sau xử lý sẽ nạp danh sách các quận vào combo box cboQuan dựa trên thành phố đã được chọn trong cboThanhPho.

```
private void cboThanhPho_SelectedIndexChanged(object sender,
EventArgs e)
{
      List<string> lstQuanHN = new List<string>();
      lstQuanHN.AddRange(new string[]{"Hoàn Kiếm", "Ba Đình",
                              "Đống Đa","Tây Hồ","Cầu Giấy"});

      List<string> lstQuanDN = new List<string>(new string[]{"Hải
Châu",
                                                      "Sơn Trà",
                                                      "Ngũ Hành
                                                Sơn"});

      List<string> lstQuanSG = new List<string>();
      lstQuanSG.Add("Quận 1");
      lstQuanSG.Add("Phú Nhuận");
      lstQuanSG.Add("Bình Thạnh");
      lstQuanSG.Add("Gò Vấp");

      object[] lstQuan = null;
      switch (cboThanhPho.SelectedIndex)
      {
            case 0:
                  lstQuan = lstQuanSG.ToArray();
                  break;

            case 1:
                  lstQuan = lstQuanHN.ToArray();
                  break;

            default:
                  lstQuan = lstQuanDN.ToArray();
                  break;
      };

      cboQuan.Items.Clear();
      cboQuan.Items.AddRange(lstQuan);

}
```

3.4 Thể hiện dữ liệu với DataGridView

DataGridView là một trong những control thể hiện dữ liệu dưới dạng bảng lên giao diện.



3.4.1 Kết nối DataGridView với đối tượng

DataGridView có thể được sử dụng để thể hiện dữ liệu từ một mảng. Trong ví dụ sau, sử dụng DataGridView để thể hiện mảng các họ tên sau khi nhấn nút Đăng ký.

Đầu tiên, tạo lớp SinhVien chứa thông tin các sinh viên.

```
public class SinhVien
{
        private string _hoTen;
        private string _quan;
        private string _thanhPho;
        private string _gioiTinh;
        private bool _noiO;

        public string HoTen
        {
                get { return _hoTen; }
                set { _hoTen = value; }
        }

        public string Quan
        {
                get { return _quan; }
                set { _quan = value; }
        }

        public string ThanhPho
        {
                get { return _thanhPho; }
                set { _thanhPho = value; }
        }

        public string GioiTinh
        {
                get { return _gioiTinh; }
                set { _gioiTinh = value; }
        }

        public bool NoiO
        {
                get { return _noiO; }
                set { _noiO = value; }
        }

}
```

Sau đó, thêm thuộc tính chứa danh sách sinh viên vào lớp Form1

```
private List<SinhVien> lstSinhVien = new List<SinhVien>();
```

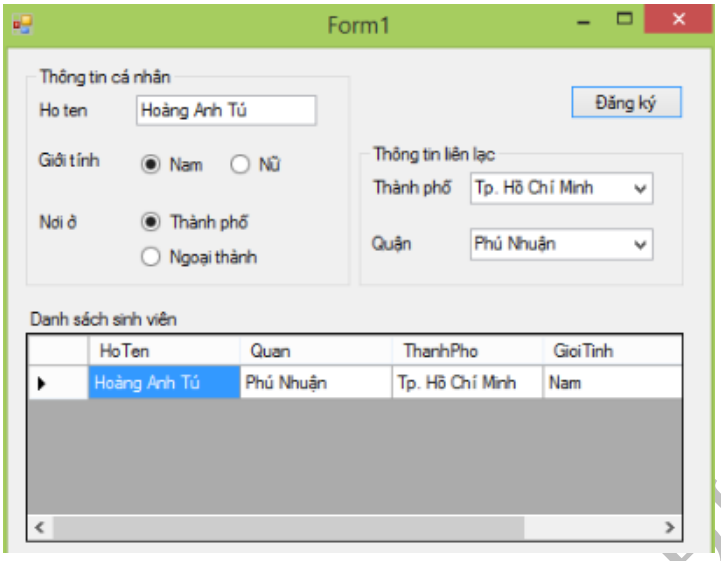
Thay đổi hàm xử lý sự kiện cho button Đăng Ký, tạo đối tượng SinhVien mới và thêm vào trong danh sách lstSinhVien.

```
private void btnDangKy_Click(object sender, EventArgs e)
{
        string hoTen = txtHoTen.Text;
        string gioiTinh = null;

        if (rdoNam.Checked)
        {
                gioiTinh = "Nam";
        }
        else
        {
                gioiTinh = "Nữ";
        }

        //true nếu là ở thành phố và false nếu ở ngoại thành
        bool noiO = rdoThanhPho.Checked;
        string thanhPho = cboThanhPho.Text;
        string quan = cboQuan.Text;

        SinhVien sv = new SinhVien();
        sv.HoTen = hoTen;
        sv.GioiTinh = gioiTinh;
        sv.NoiO = noiO;
        sv.ThanhPho = thanhPho;
        sv.Quan = quan;

        lstSinhVien.Add(sv);
        grdDanhSachSV.DataSource = lstSinhVien;
}
```

Kết nối danh sách lstSinhVien với DataGridView thông qua thuộc tính DataSouce để thể hiện dữ liệu.

```
grdDanhSachSV.DataSource = lstSinhVien;
```

Lúc này, sau khi nhập đầy đủ thông tin và nhấn vào nút Đăng ký, thông tin sinh viên bên trên sẽ được hiển thị vào DataGridView.



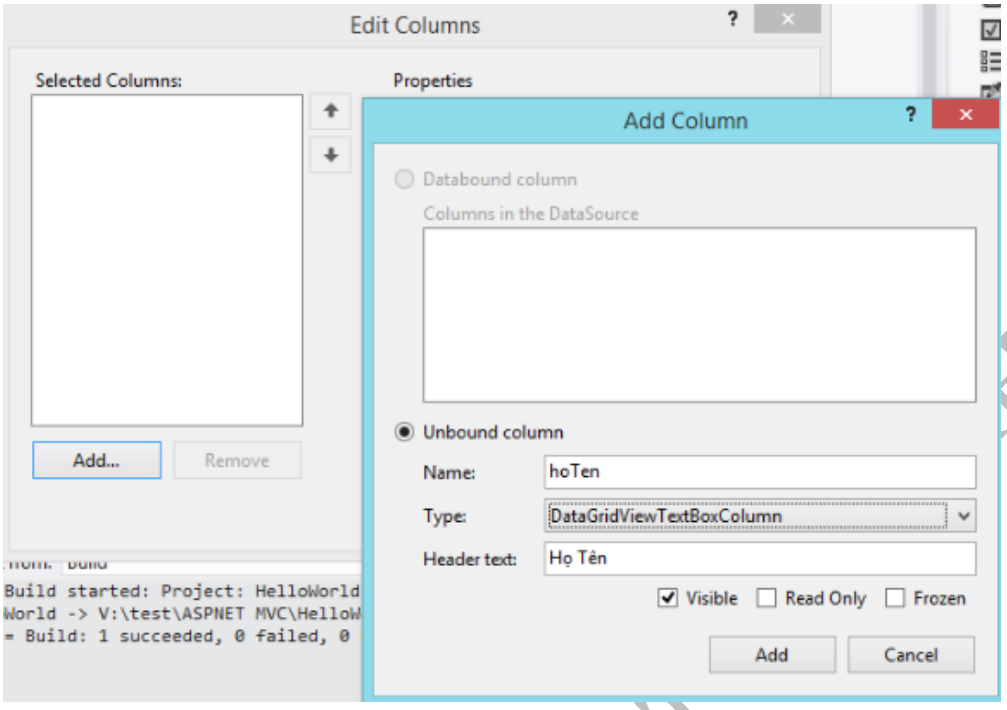
3.4.2 Chỉnh sửa các cột trong DataGridView

Mặc định, DataGridView sẽ tự động tìm các thuộc tính có trong lớp đối tượng và tạo thành các cột. Ở đây, lớp SinhVien được tạo 5 thuộc tính gồm: HoTen, GioiTinh, NoiO, Quan, ThanhPho nên grdDanhSachSinhVien sẽ hiển thị tên cột theo tên thuộc tính.

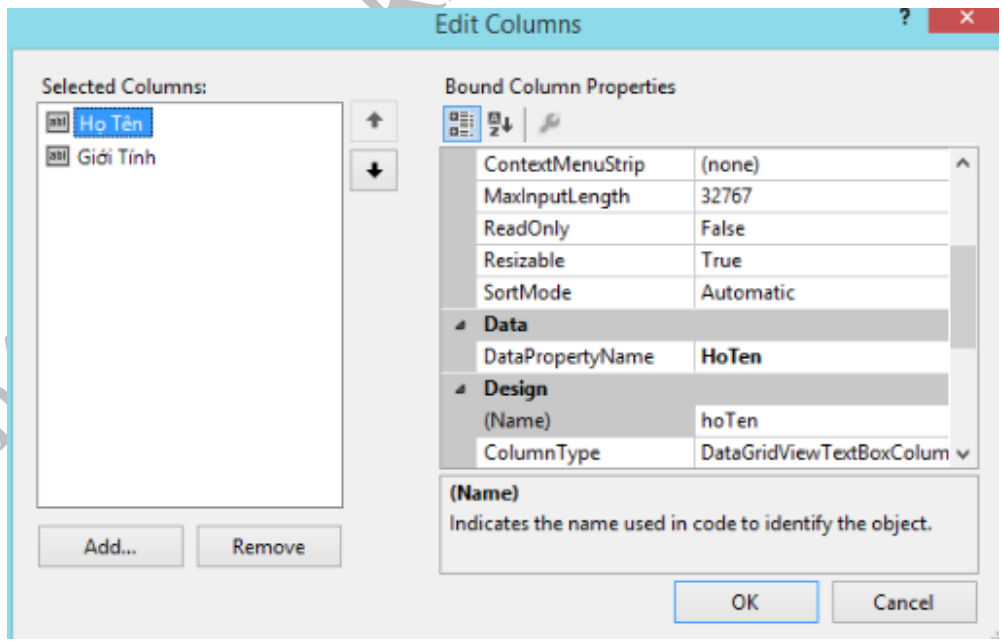
Để hiển thị tên cột riêng, không phụ thuộc vào tên cột của thuộc tính, cần thay đổi giá trị thuộc tính AutoGenerateColumns thành false. Đoạn code sau thay đổi giá trị thuộc tính và khởi tạo trong sự kiện Load của Form.

```
private void Form1_Load(object sender, EventArgs e)
{
        grdDanhSachSV.AutoGenerateColumns = false;
}
```

Sau đó, thêm danh sách các cột vào trong DataGridView bằng cách thay đổi thuộc tính Columns trong phần properties của DataGridView.



Lần lượt thêm các cột với Name là tên các cột và Header Text là phần thể hiện của cột đó trên tiêu đề của DataGridView. Với mỗi cột đã được thêm vào, cần thay đổi thuộc tính DataPropertyName thành tên thuộc tính tương ứng trong lớp đối tượng.



Lúc này, khi chạy chương trình, DataGridView grdDanhSachSV sẽ không tự động thêm các cột vào nữa mà chỉ hiện 2 cột Họ Tên và Giới Tính.