

BÀI TẬP THIẾT KẾ FORM – PRN211

Bài Tập C# - Bài 1 - Đối Tượng Form Và Một Số Control Thông Dụng Trong Lập Trình Windows Form

BÀI TẬP THỰC HÀNH C# Bài 1. Đối tượng Form và một số control thông dụng trong lập trình Windows Form

BÀI TẬP THỰC HÀNH C# Bài 1. Đối tượng Form và một số control thông dụng trong lập trình Windows Form

Trong bài này mình sẽ giới thiệu chi tiết hơn về Form và các control (*điều khiển*) thông dụng sử dụng trong Form (*Windows Forms Application*). Đây là một phần rất quan trọng vì nó chính là giao diện của ứng dụng.

Chúng ta sẽ lần lượt tìm hiểu cách thêm một Form, xử lý các sự kiện với Form và một số ví dụ cơ bản sử dụng các control thông dụng.

Table of Content

- 1. Thêm, xóa một Form trong project
 - Thêm một Form vào project
 - Xóa một Form khỏi project
- 2. Một số thuộc tính của Form
- 3. Một số sự kiện thông dụng của Form
- 4. Một số phương thức thường dùng trong Form
- 5. Phân loại Form
 - Phân loại Form
 - Gọi hiển thị Form
 - Đóng Form, dừng chương trình
- 6. Ví dụ sử dụng các sự kiện thông dụng trong Form
- 7. Kết luận

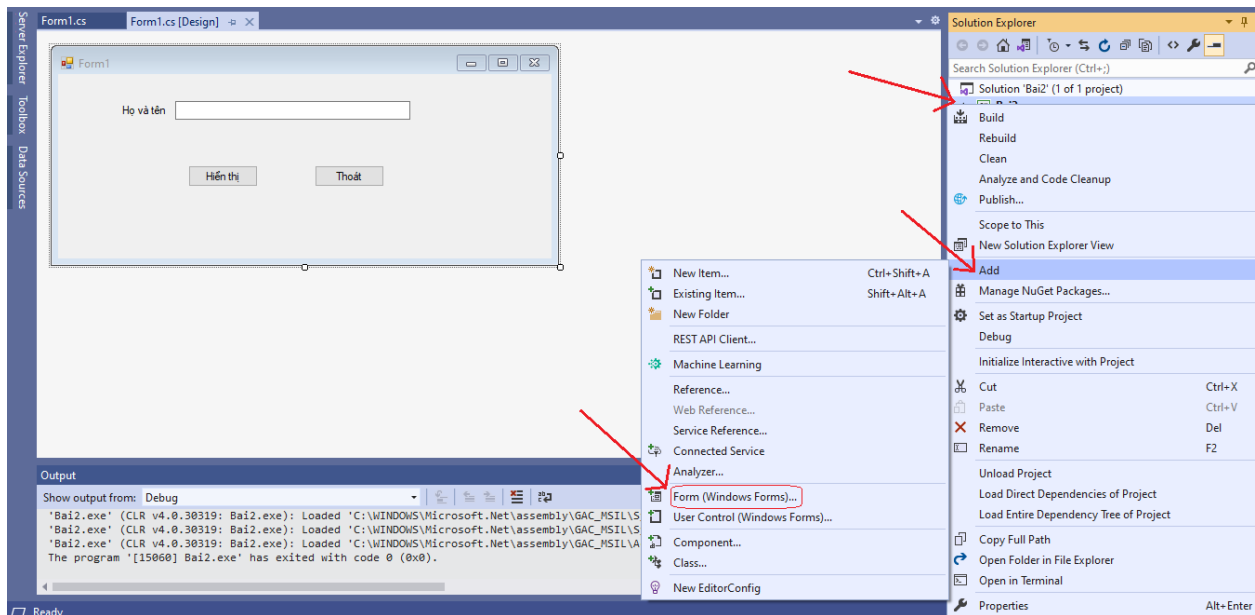
1. Thêm, xóa một Form trong project

Trong phần này mình sẽ hướng dẫn các bạn cách để thêm mới một Form hoặc thêm một Form đã có sẵn vào project. Cùng với đó là cách xóa một Form khỏi project như thế nào.

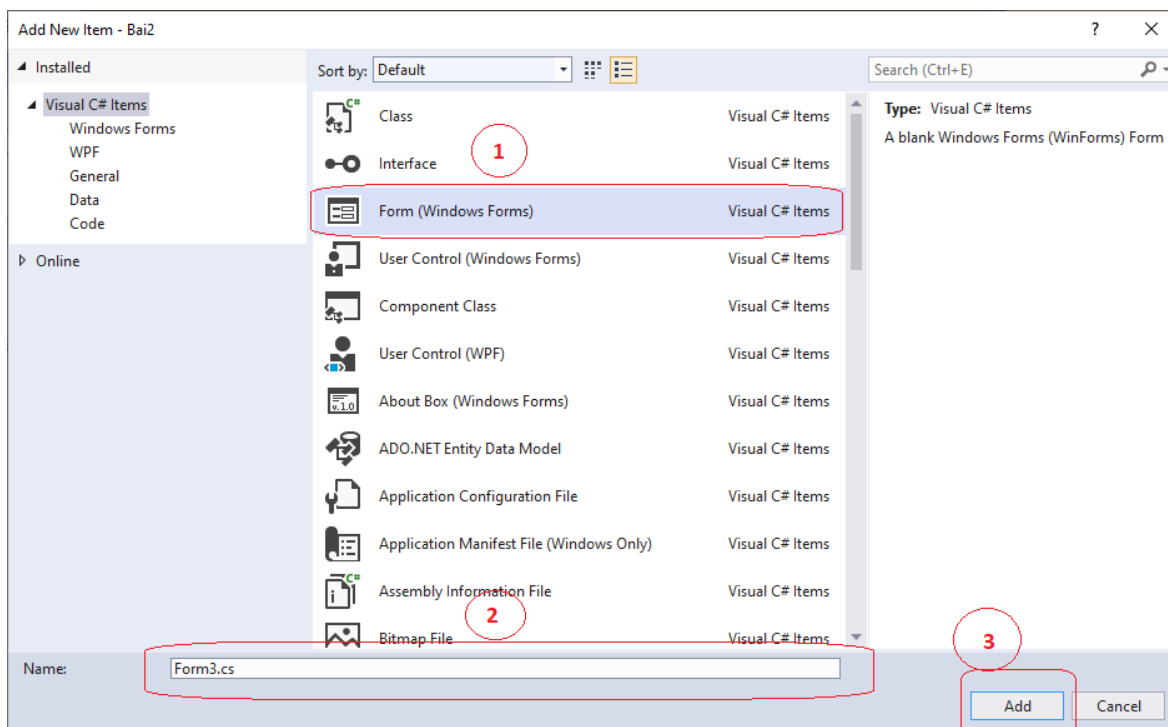
Thêm một Form vào project

Thêm mới một Form vào project: Với mỗi project chúng ta sẽ cần nhiều Form để thể hiện giao diện của người dùng . Việc thêm mới một Form mới vào project được thực hiện như sau:

Bước 1: Vào menu *Project | Add Windows Form* hoặc vào menu *Project | Add New Item*



Bước 2: Thiết lập trên cửa sổ Add New Item như sau:



Thêm một Form có sẵn vào project: Ta cũng có thể thêm một Form đã có sẵn bằng cách:

Bước 1: Vào menu *Project | Add Existing Item*.

Bước 2: Theo đường dẫn, chọn Form cần add

Bước 3: Nhấn nút Add

Xóa một Form khỏi project

Để xóa một Form đang có trong dự án:

Bước 1: *Nhấn chuột phải* vào Form cần gỡ bỏ (ở cửa sổ Solution Explorer).

Bước 2: Chọn *Delete* trong menu hiện ra

2. Một số thuộc tính của Form

Form là control chứa (*chứa các control khác khi tạo giao diện cho ứng dụng*). Các bạn cùng mình tìm hiểu một số thuộc tính của Form ở bảng dưới đây

| Thuộc tính | Diễn giải |
|------------------------|---|
| Text | Tiêu đề (titlebar) của Form |
| WindowState | Trạng thái thể hiện của Form (normal, minimized,maximized) |
| StartPosition | Vị trí Form hiển thị khi chương trình gọi đến Form |
| TopMost | Form có chọn hiện ưu tiên trong các action Form hay không |
| Locked | Khóa vị trí các controls trong quá trình thiết kế |
| Icon | Chọn biểu tượng cho Form |
| ControlBox | Mang giá trị True hoặc False. Nếu thiết lập thuộc tính là False thì sẽ loại bỏ các nút minimize và nút maximize trên Form |
| MinimizeBox | Có hiện nút thu nhỏ của Form |
| MaximizeBox | Có hiện nút phóng to của Form |
| BackColor | Lựa chọn màu nền cho Form |
| BackgroundImage | Xác định file hình làm hình nền cho Form |
| FormBoderStyle | Lựa chọn kiểu đường viền cho Form |
| ForeColor | Chọn màu chữ cho Form |
| Font | Lựa chọn font chữ, font size, font style cho Form |
| Cursor | Thiết lập hình dạng con trỏ khi di chuyển con trỏ vào Form |
| IsMDIContainer | Mang giá trị True hoặc False; True: Form ở dạng MDI Form (Form cha), False: Form ở dạng bình thường |

| Thuộc tính | Diễn giải |
|------------------------------|---|
| BackgroundImageLayout | Thiết lập việc hiển thị hình vừa thêm trong thuộc tính BackgroundImage sẽ hiển thị trên Form ở dạng: bình thường (None), giữa (Center),... |
| AcceptButton | Giá trị thuộc tính này nhận là tên của một Button trên Form. Khi đó thay vì nhấp chuột vào Button để thực thi thì người dùng có thể nhấn phím Enter trên bàn phím |
| CancleButton | Giá trị thuộc tính này nhận là tên của một Button trên Form. Khi đó thay vì nhấp chuột vào Button để thực thi thì người dùng có thể nhấn nút Escape trên bàn phím |
| Keypreview | Cho phép Form nhận các giá trị từ bàn phím, mặc định là False (không nhận) |

3. Một số sự kiện thông dụng của Form

Trong phần này mình sẽ liệt kê một số sự kiện thông dụng của Form, đây là các sự kiện rất quan trọng vì vậy các bạn nên học và ghi nhớ nó thật kỹ.

Các sự kiện trong Form được liệt kê trong bảng dưới đây:

| Tên sự kiện | Diễn giải |
|-------------------------|---|
| FormLoad | Xảy ra khi Form bắt đầu chạy, ta thường dùng sự kiện Load để khởi tạo các giá trị ban đầu |
| FormClosing | Xảy ra trước khi đóng Form, ta thường dùng sự kiện này để giải phóng tài nguyên hệ thống |
| KeyDown | Xảy ra khi một phím được nhấn trên Form. muốn cho sự kiện này xảy ra ta phải đặt thuộc tính KeyPreview có giá trị là True, để lấy mã của phím được nhấn (keyCode) |
| MouseClick | Xảy ra khi nhấn một trong ba nút của chuột: chuột trái, chuột phải và chuột giữa |
| BackColorChanged | Giá trị thuộc tính BackColor thay đổi |
| ForeColorChanged | Giá trị thuộc tính ForeColor bị thay đổi |
| Click | Nhấp chuột vào vùng làm việc của Form |
| DoubleClick | Nhấp đúp chuột vào vùng làm việc của Form |
| FormClosed | Form đã được đóng hoàn toàn |
| KeyDown | Phím được nhấn xuống |
| KeyUp | Phím được thả ra |
| KeyPress | Phím được nhấn xuống và thả ra |
| MouseEnter | Chuột nằm trong vùng thấy được của Form |
| MouseHover | Chuột nằm trong vùng hiển thị một khoảng thời gian |
| MouseDown | Nhấn chuột trên vùng hiển thị của Form |
| MouseLeave | Chuột ra khỏi vùng thấy được của Form |

| Tên sự kiện | Diễn giải |
|--------------------|---|
| MouseMove | Chuột được di chuyển trên Form |
| Move | Form được di chuyển |
| Resize | Form bị thay đổi kích thước |
| TextChanged | Giá trị của thuộc tính Text bị thay đổi |

4. Một số phương thức thường dùng trong Form

Trong phần này mình sẽ giới thiệu các bạn một số phương thức thường được dùng trong Form, hầu hết trong các ứng dụng đều sử dụng các phương thức này.

- **Show()**: Sử dụng để hiển thị Form.
- **ShowDialog()**: Tương tự như Show() nhưng Form hiển thị bằng phương thức ShowDialog() sẽ bắt buộc người dùng phải thao tác cho tới khi đóng Form (khi chưa đóng Form thì không được thao tác với Form khác).
- **Hide()**: Ẩn Form, tương tự như việc thiết lập thuộc tính Visible = False.
- **Close()**: Đóng Form.

5. Phân loại Form

Trong phần này mình sẽ phân loại các Form thường gặp cũng như cách gọi hiển thị Form, đóng Form và dừng chương trình.

Phân loại Form

Trong Form sẽ được phân làm 3 loại Form:

- **Form bình thường (Normal Form)**: Một Form bình thường khi được gọi hiển thị bên trong một Form cha và được chỉ định Form cha chứa nó là Form nào thông qua thuộc tính **MdiParent** khi đó sẽ trở thành Form con.
- **Form cha (MdiParent Form)**: Là Form có thể chứa các Form khác bên trong nó, để một Form trở thành Form cha ta cần thiết lập thuộc tính **isMdiContainer** của Form có giá trị là True.
- **Form con (MdiChildren Form)**

Gọi hiển thị Form

Gọi hiển thị Form2 từ Form1 dạng: Form2 là Normal Form.

```
1Form frm = new Form2();
```

```
2frm.Show();
```

Gọi hiển thị Form2 thì Form1 dạng: Form2 là Form con.

```
1Form frm = new Form2();
```

```
2frm.MdiParent = this;
```

```
3frm.Show();
```

Đóng Form, dừng chương trình

Để đóng Form hoặc dừng chương trình ta sử dụng 2 dòng lệnh sau:

- **This.Close()**: Đóng Form hiện hành
- **Application.Exit()**: Đóng chương trình

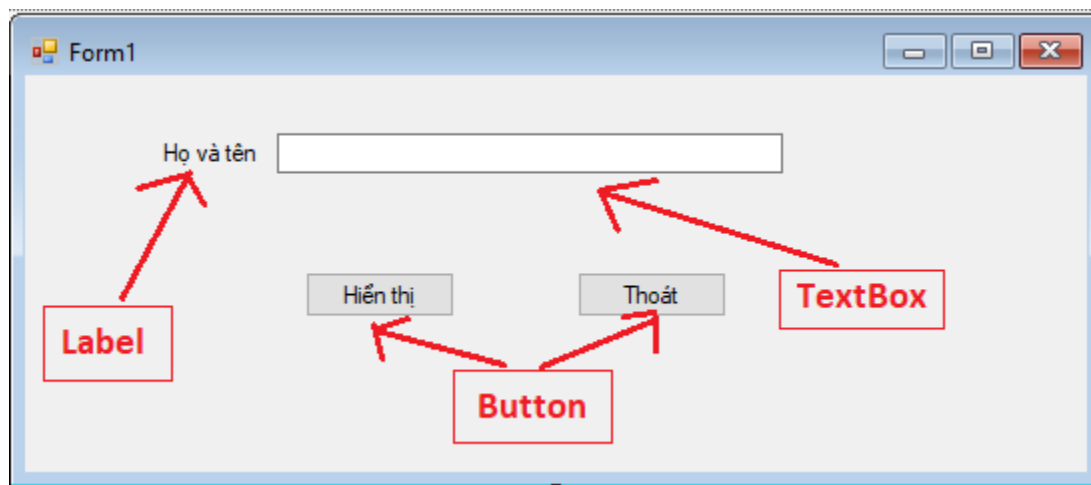
6. Ví dụ sử dụng các sự kiện thông dụng trong Form

Trong phần này mình sẽ tạo một ứng dụng đơn giản sử dụng một số control thông dụng trong Form, cụ thể mình sẽ tạo một Form với các sự kiện sau:

- Sử lý sự kiện ở **FormLoad** xuất hiện hộp thoại hỏi người dùng có muốn mở ứng dụng hay không.
- Nhấn nút "**Hiển thị**" thì hiển thị một hộp thoại với nội dung được nhập ở ô **TextBox**.
- Nhấn tổ hợp phím **ALT + H** sẽ xuất dòng chữ "**Hello Freetuts.net**" trong ô **TextBox**.
- Click chuột phải sẽ hiển thị hộp thoại thông báo người dùng vừa nhấp chuột phải, tương tự như vậy cho chuột trái và chuột giữa.
- Nhấn nút "**Thoát**" sẽ xuất hiện hộp thoại hỏi người dùng có muốn thoát hay không.

Việc đầu tiên chúng ta tạo giao diện với các điều khiển cần thiết, sau đó sẽ xử lý từng sự kiện theo như yêu cầu của đề bài.

Bước 1: Tạo giao diện cho ứng dụng



Ta sẽ tạo một **Label** để hiển thị dòng chữ "**Họ và tên**", một ô **TextBox** để người dùng nhập dữ liệu cũng như để hiển thị dữ liệu và hai **Button** để xử lý hai sự kiện là hiển thị và

thoát. Ở bài trước mình đã có hướng dẫn đổi nội dung Text và đặt tên cho điều khiển, các bạn có thể xem lại nhé.

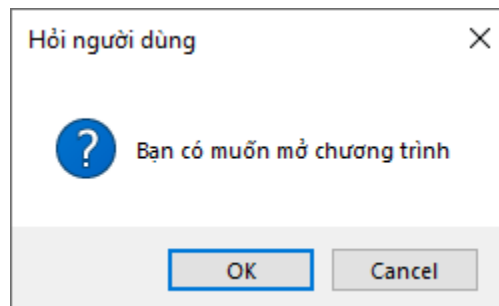
Bước 2: Xử lý các sự kiện

Trong phần này ta sẽ xử lý lần lượt các sự kiện mà đề bài đã yêu cầu, đầu tiên ta sẽ xử lý sự kiện **FormLoad**. Trước khi Form bắt đầu chạy, hệ thống sẽ hiển thị một hộp thoại với nội dung hỏi xem người dùng có muốn mở ứng dụng hay không, nếu chọn "OK" thì mở ứng dụng, ngược lại chọn "Cancel" thì không mở.

Ta sẽ sử dụng hộp thoại **MessageBox.Show()** với thuộc tính **OK/Cancel** để hiển thị thông báo cho người dùng.

```
1 private void Form1_Load(object sender, EventArgs e)
2     {
3         if (MessageBox.Show("Bạn có muốn mở chương trình", "Hỏi người dùng",
4             MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.Cancel)
5             Dispose();
6     }
```

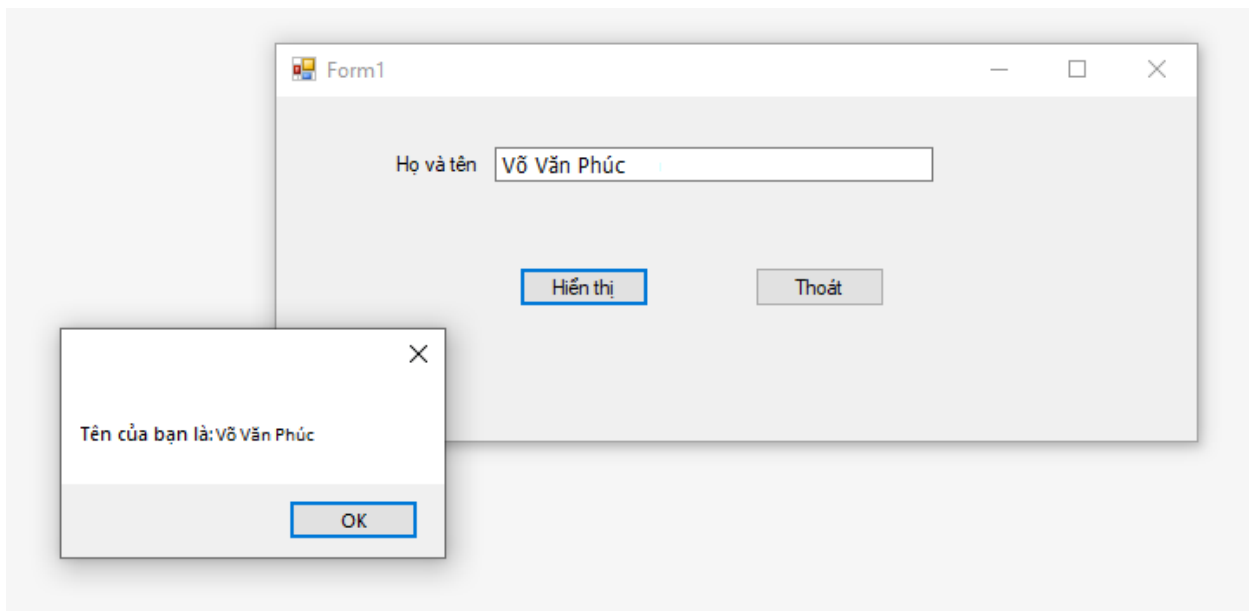
Kết quả:



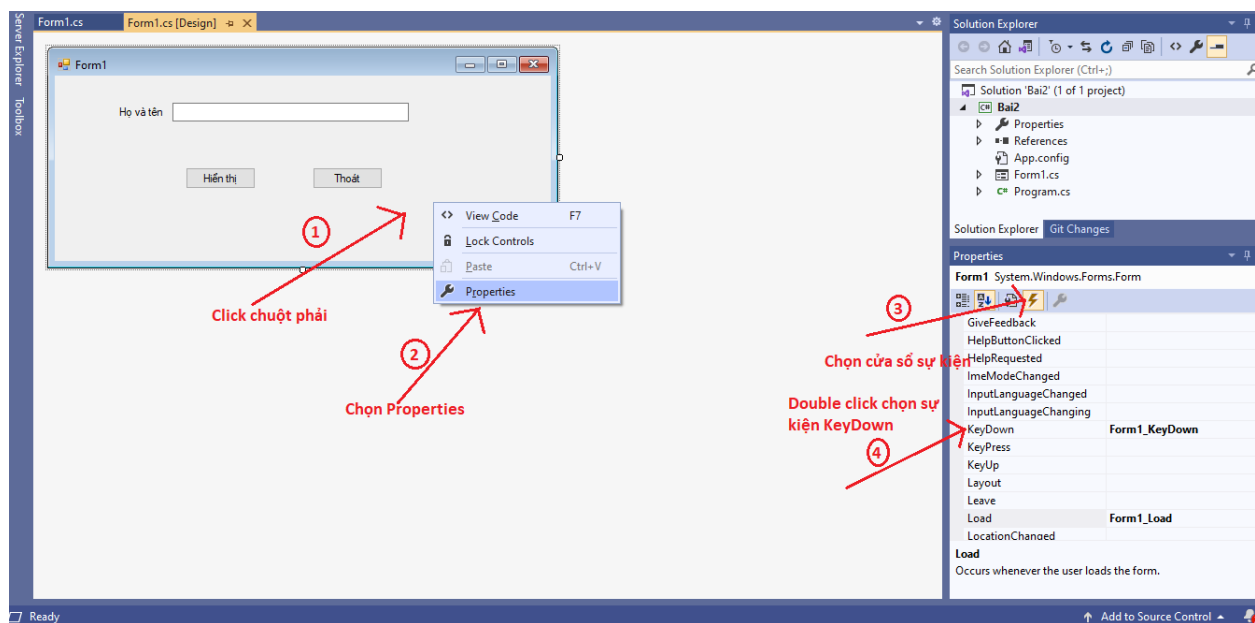
Tiếp theo sẽ là sự kiện nhấp vào nút Button "**Hiển thị**" thì nội dung được nhập ở ô TextBox sẽ được hiện ra trên hộp thoại, các bạn Double Click vào nút Button "**Hiển thị**" để viết sự kiện cho nó. Chúng ta cũng sẽ sử dụng **MessageBox.Show()** để tạo hộp thoại.

```
1 private void btn_hienthi_Click(object sender, EventArgs e)
2     {
3         MessageBox.Show("Tên của bạn là: " + txt_hoten.Text);
4     }
```

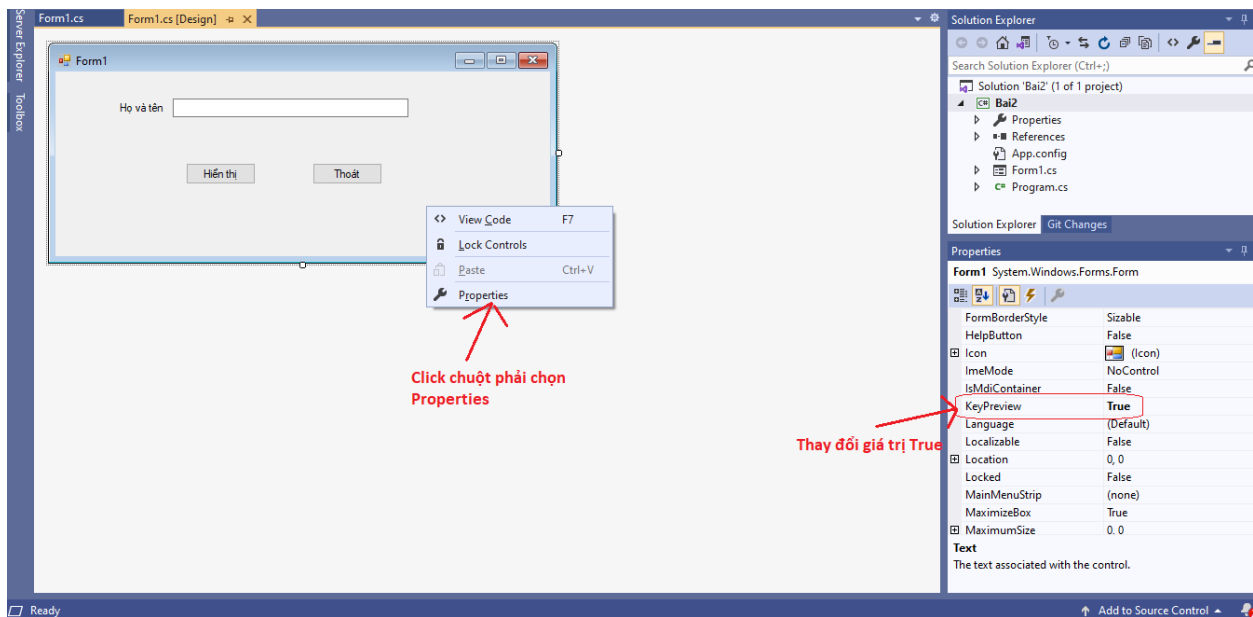
Kết quả:



Sự kiện tiếp theo khi các bạn nhấn tổ hợp phím **ALT + H** thì nội dung "**Hello Freetuts.net**" sẽ được hiển thị lên ô TextBox. Chúng ta sẽ viết trên sự kiện **KeyDown** (phím được nhấn xuống) bằng cách chuột phải vào Form chọn Properties rồi Double click vào sự kiện Keydown.



***Lưu ý:** Các bạn phải thay đổi giá trị thuộc tính **KeyPreview** với giá trị **True** để nhận các giá trị từ bàn phím, vì mặc định nó sẽ bằng **False**.



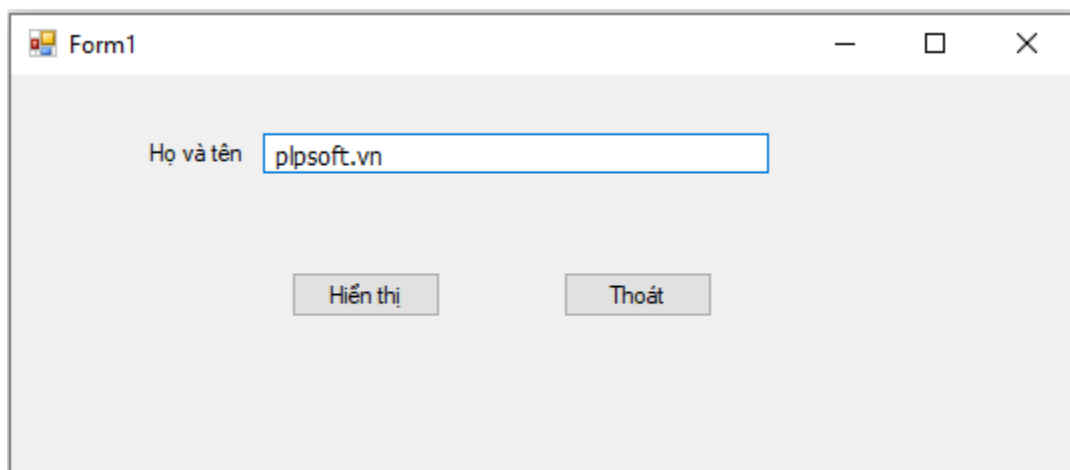
Các bạn xử lý sự kiện theo đoạn code sau:

```

1 private void Form1_KeyDown(object sender, KeyEventArgs e)
2     {
3         if(e.Modifiers == Keys.Alt && e.KeyCode == Keys.H)
4             //Lưu ý: bật KeyPreview == true trước khi chạy Form
5             {
6                 // gán nội dung "Hello Freetuts.net" vào ô TextBox
7                 txt_hoten.Text = "Hello Freetuts.net !!!";
8             }
9     }

```

Kết quả:

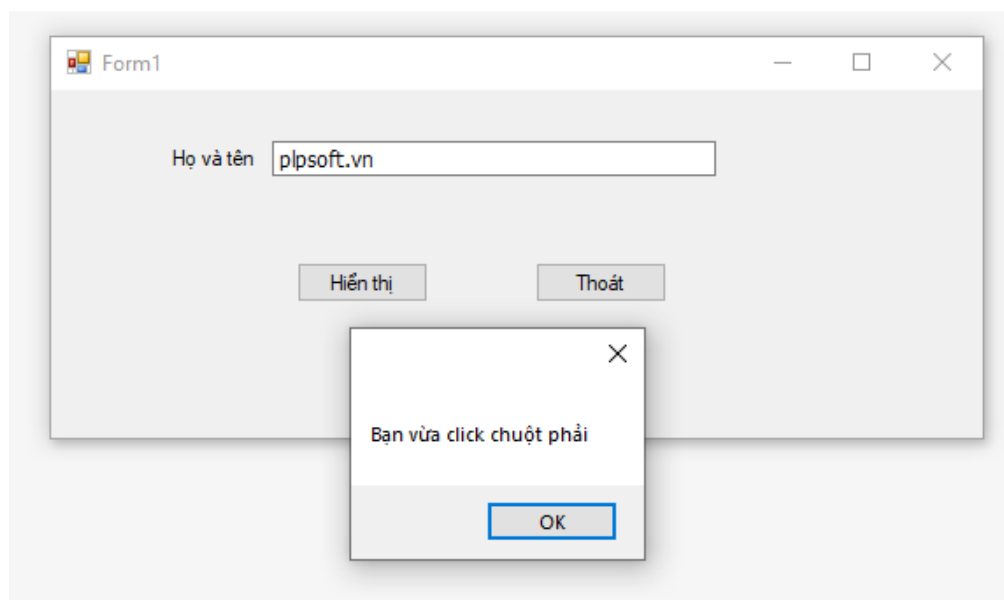


Sự kiện tiếp theo là khi các bạn nhấp chuột hệ thống sẽ hiển thị hộp thoại thông báo các bạn vừa nhấp chuột, ví dụ khi các bạn nhấp chuột phải thì hệ thống sẽ hiển thị hộp thoại thông báo bạn vừa nhấp chuột phải.

Các bạn sẽ xử lý trên sự kiện **MouseClick**, Tương tự như **KeyDown** các bạn sẽ vào Properties của Form rồi Double click vào sự kiện **MouseClick** để viết sự kiện.

```
1 private void Form1_MouseClick(object sender, MouseEventArgs e)
2     {
3         if(e.Button == MouseButtons.Left)
4         {
5             MessageBox.Show("Bạn vừa click chuột trái");
6         }
7         if (e.Button == MouseButtons.Right)
8         {
9             MessageBox.Show("Bạn vừa click chuột phải");
10        }
11        if(e.Button == MouseButtons.Middle)
12        {
13            MessageBox.Show("Bạn vừa click chuột giữa");
14        }
15    }
```

Kết quả:

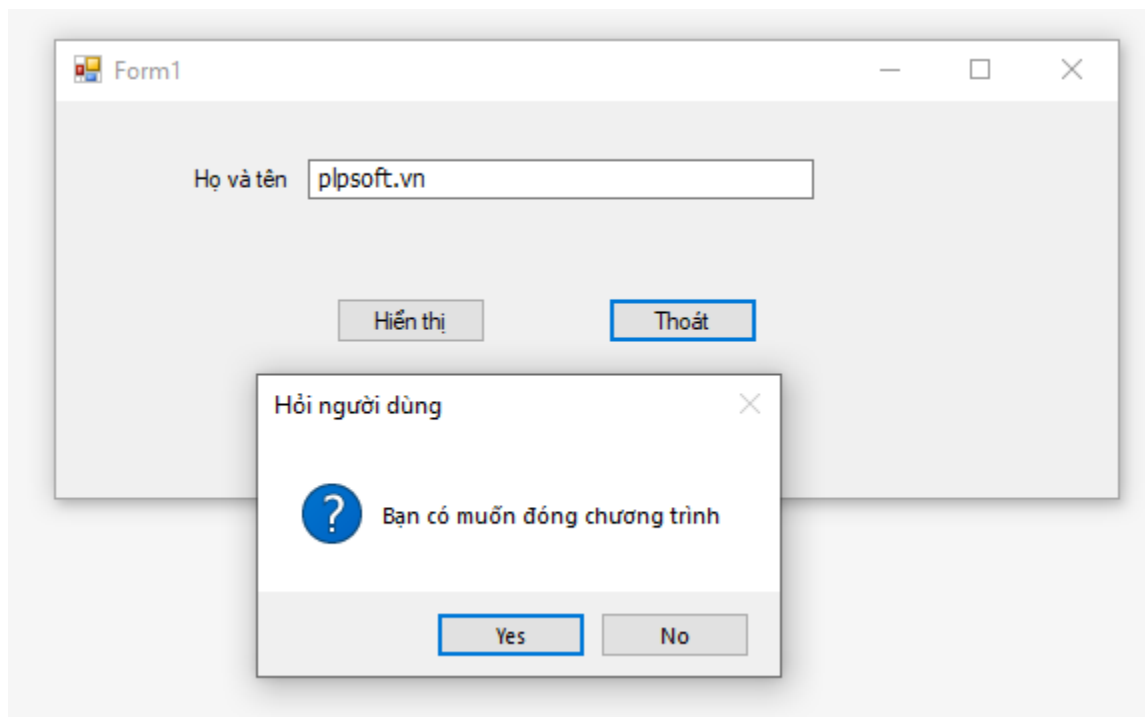


Và cuối cùng là sự kiện khi các bạn nhấn vào nút Button "**Thoát**" thì hệ thống sẽ hiển thị hộp thoại hỏi các bạn có muốn thoát hay không, nếu chọn "**Yes**" thì thoát ứng dụng, nếu chọn "**No**" thì thoát khỏi hộp thoại và quay lại ứng dụng.

Chúng ta dùng phương thức **Application.Exit()** để thoát khỏi chương trình, đây là một phương thức rất thông dụng và được sử dụng rất nhiều.

```
1 private void btn_thoat_Click(object sender, EventArgs e)
2     {
3         DialogResult dg = MessageBox.Show("Bạn có muốn đóng chương trình", "Hỏi
4 người dùng", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
5         if (dg == DialogResult.Yes)
6         {
7             Application.Exit();
8         }
9     }
```

Kết quả:



Bài 2. Cách dùng Label - Button - Textbox trong C# winforms

Trong bài này mình sẽ giới thiệu các bạn một trong số các điều khiển thông dụng trong C# winforms đó chính là Label, Button, Textbox. Đây là các điều khiển được sử dụng rất nhiều khi lập trình ứng dụng với winforms.

Chúng ta sẽ cùng nhau tìm hiểu lần lượt các điều khiển trên, sau đó mình sẽ tạo một ứng dụng đơn giản sử dụng các điều khiển và xử lý sự kiện cho nó.

Table of Content

- 1. Label
- 2. Button
- 3. Textbox
- 4. Ví dụ sử dụng các điều khiển Label, Button, Textbox
 - Đề bài
 - Tạo ứng dụng máy tính bỏ túi
- 5. Kết luận

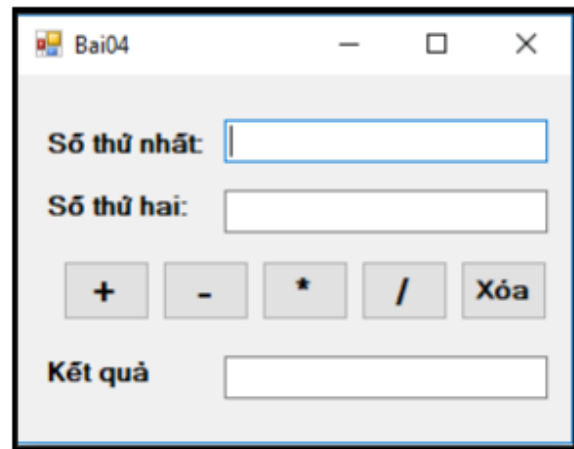
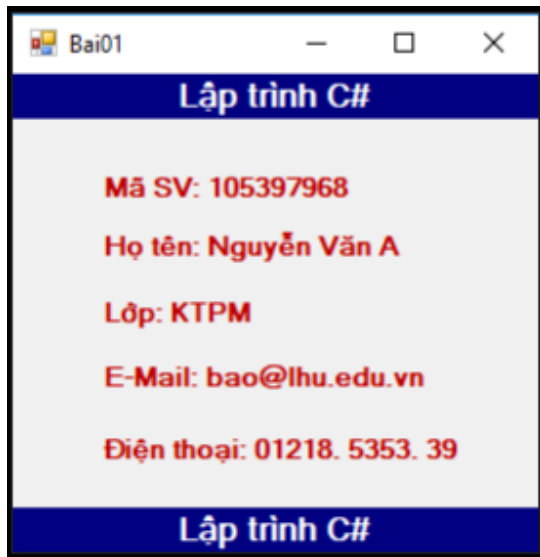
1. Label

Label thường được dùng hiển thị thông tin chỉ đọc và thường sử dụng kèm với các điều khiển khác để mô tả chức năng.

Một số thuộc tính thường dùng của Label

| Thuộc tính | Mô tả |
|-------------|--|
| BorderStyle | Thiết lập đường viền |
| Font | Thiết lập font chữ hiển thị trên Label |
| Text | Nội dung hiển thị trên Label |
| BackColor | Màu nền của Label |
| ForeColor | Thiết lập màu chữ hiển thị trên Label |
| Name | Tên của Label |
| TextAlign | Canh lề nội dung của Label (Left Center Right) |
| Visible | Ẩn hoặc hiện Label |

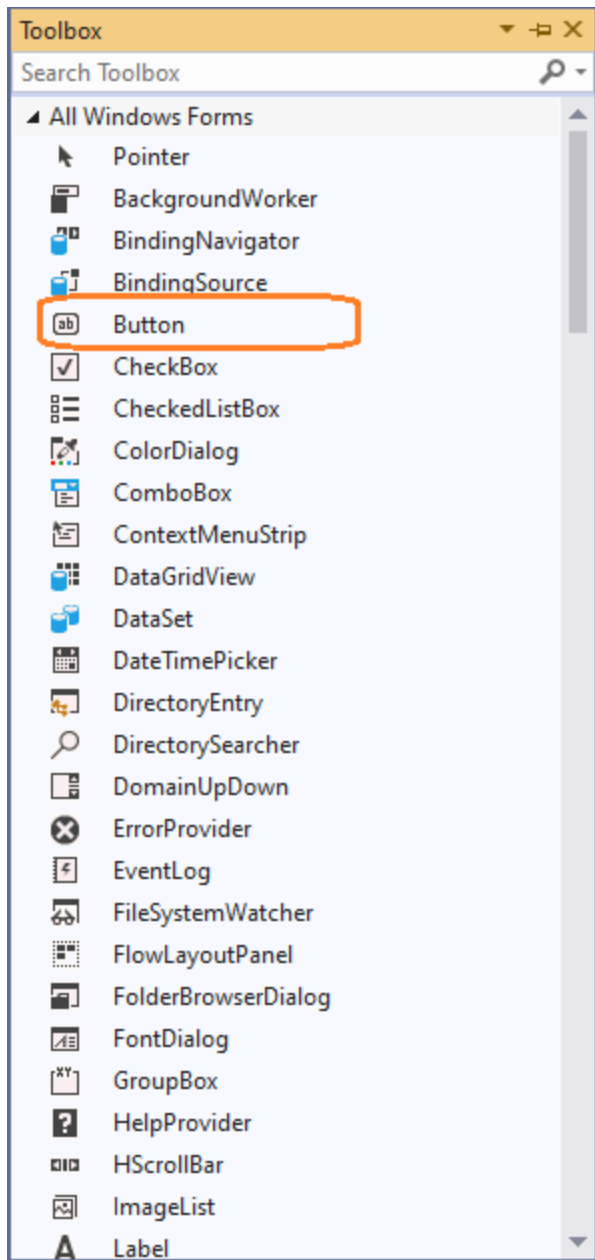
Ví dụ:



Các dòng chữ ở hai ví dụ trên đều sử dụng Label để mô tả, sử dụng thuộc tính Text trong Label để có thể đặt nội dung tùy ý.

2. Button

Button là điều khiển tạo giao diện nút lệnh trên Form, khi người dùng nhấn chuột vào nút lệnh thì chương trình sẽ thực hiện một hành động nào đó. Button được đặt trong nhóm Common Controls của cửa sổ Toolbox.



Công dụng của Button:

- Dùng để thực thi lệnh.
- Khi nhấp chuột lên Buton, chương trình nhận được tính hiệu Click và lệnh được thi hành.

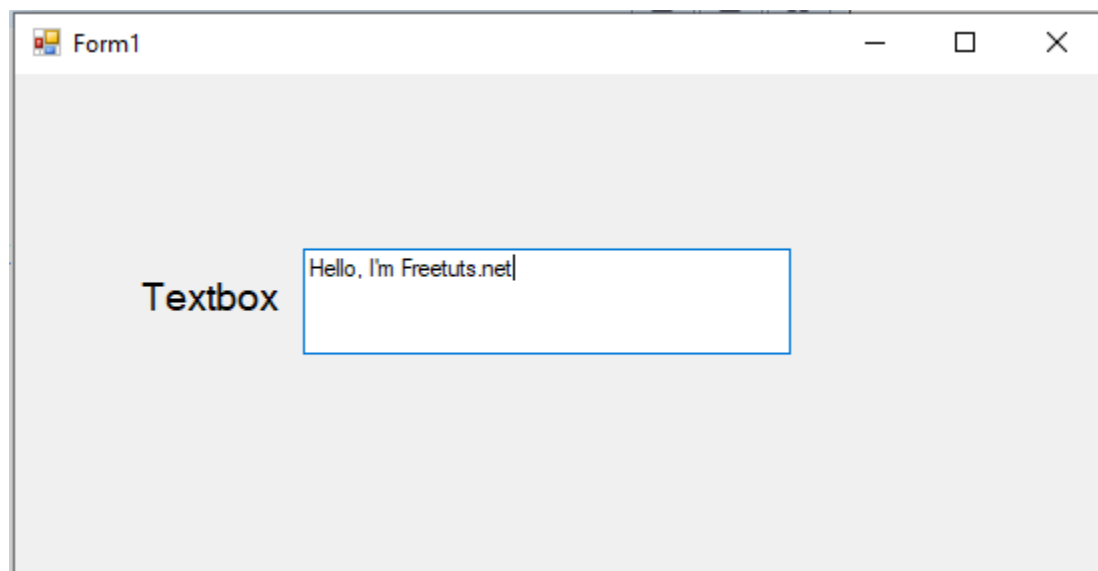
| Thuộc tính | Mô tả |
|------------|----------------------|
| Name | Đặt tên cho nút lệnh |

| Thuộc tính | Mô tả |
|------------|---|
| Text | Nội dung hiển thị lên nút nhấn |
| Visible | Ẩn, hiện nút nhấn |
| Enable | Cho phép/ không cho phép tương tác với nút lệnh |
| Font | Chỉ định kiểu chữ, kích cỡ chữ hiển thị |
| Image | Hình ảnh hiển thị trên nút lệnh |
| ForeColor | Màu chữ hiển thị trên nút lệnh |
| BackColor | Màu của nút lệnh |
| TabIndex | Chỉ định thứ tự tab của các Button trên Form |

| Sự kiện | Diễn giải |
|-------------|-------------------------------------|
| Click | Sự kiện nhấn chuột vào Button |
| TextChanged | Giá trị thuộc tính Text bị thay đổi |

3. Textbox

Trong phần này chúng ta sẽ tìm hiểu về công dụng của Textbox là gì, cũng như các thuộc tính và các phương thức thường dùng trong Textbox.



Công dụng của Textbox:

- Dùng để trình bày văn bản và cho phép người dùng được thay đổi nội dung văn bản.
- Công dụng chính là cho người dùng nhập văn bản.
- Ngoài các thuộc tính chung của các control, thì các bạn cần lưu ý thêm một số thuộc tính bên dưới:

| Thuộc tính | Diễn giải |
|-------------------|---|
| PasswordChar | Ký tự thay thế khi nhập vào TextBox |
| Multiline | Cho phép Textbox có nhiều dòng hay không |
| ScrollBars | Thanh cuộn (None / Horizontal / Vertical / Both) |
| MaxLength | Quy định chuỗi Max sẽ nhập vào Textbox, mặc định là 32767 |
| Focus | Textbox sẵn sàng được tương tác bởi người dùng |

Một số sự kiện thường dùng của Textbox:

| Sự kiện | Diễn giải |
|----------------|---|
| KeyDown | Thực hiện công việc nào đó khi một phím được nhấn xuống |
| KeyUp | Thực hiện công việc nào đó khi một phím được thả ra |
| KeyPress | Xảy ra khi người dùng nhấn một phím và thả ra. Mỗi sự kiện KeyPress cho ta một cặp sự kiện KeyDown và KeyUp |
| TextChanged | Giá trị của thuộc tính Text bị thay đổi |
| MouseEnter | Chuột nằm trong vùng thấy được của Textbox |
| MouseHover | Chuột nằm trong vùng hiển thị một quãng thời gian |
| MouseLeave | Chuột ra khỏi vùng nhập liệu của Textbox |
| MouseMove | Chuột được di chuyển trên Textbox |

Một số phương thức của Textbox:

| Phương thức | Diễn giải |
|--------------------|---|
| Clear() | Xóa tất cả chuỗi hiển thị trong Textbox |
| Cut() | Di chuyển phần nội dung bôi đen của chuỗi |

| Phương thức | Diễn giải |
|---------------|--|
| Paste() | Dán phần nội dung được chọn của chuỗi |
| Copy() | Sao chép phần nội dung được bôi đen của chuỗi |
| Undo() | Khôi phục thao tác trước |
| Select() | Chọn một phần nội dung của chuỗi trong Textbox |
| SelectAll() | Chọn tất cả nội dung của chuỗi trong Textbox |
| DeselectAll() | Bỏ chọn chuỗi trong Textbox |

4. Ví dụ sử dụng các điều khiển Label, Button, Textbox

Trong phần này mình sẽ tạo một ứng dụng nhập xuất đơn giản sử dụng 3 điều khiển nói trên đó chính là máy tính bỏ túi dùng để tính toán các phép tính đơn giản như cộng, trừ, nhân, chia.

Bài tập:

Tạo giao diện cho Form như hình

The screenshot shows a Windows application window titled "Form1". Inside the window, there is a calculator-like interface. It features two text boxes at the top, each preceded by a label: "Số thứ nhất" and "Số thứ hai". Below these text boxes is a row of five buttons: a plus sign (+), a minus sign (-), an asterisk (*), a forward slash (/), and a button labeled "Del". At the bottom of the form is another text box preceded by the label "Kết quả".

Xử lý các sự kiện sau:

- Ở ô Textbox thêm điều kiện chỉ cho phép người dùng nhập số và xóa lùi trên Textbox
- Xử lý các phép toán cộng, trừ, nhân, chia tương ứng với các Button.
- Khi người dùng nhấn Del thì các số ở trên Textbox sẽ được xóa hết.
- Kết quả sẽ được hiển thị ở ô Textbox "Kết quả"

Tạo ứng dụng máy tính bỏ túi

Chúng ta sẽ lần lượt thực hiện theo yêu cầu của đề bài, đầu tiên sẽ tạo giao diện cho Form như đề bài. Chúng ta sẽ có tất cả 3 Label, 3 Textbox, 5 Button.

***Lưu ý:** Các bạn nên đặt tên cho từng Label, Button, Textbox để khi chúng ta xử lý sự kiện dễ dàng truy xuất nó ra và không bị nhầm lẫn với các điều khiển khác.

Sau khi các bạn đã tạo xong giao diện Form giống như đề bài yêu cầu, bây giờ các bạn sẽ bắt tay vào thực hiện xử lý các sự kiện trên các điều khiển.

Sự kiện đầu tiên là các ô TextBox chỉ cho nhập số và được xóa lùi. Ta sẽ viết trên sự kiện KeyPress, nếu e.KeyChar (*ký tự người dùng nhập vào*) ≥ 0 và ≤ 9 thì cho nhập vào.

```
1private void txt_num1_KeyPress(object sender, KeyPressEventArgs e)
2    {
3        if (!(e.KeyChar >= '0' && e.KeyChar <= '9' || e.KeyChar == (char)8))
4            {
5                e.Handled = true;
6            }
7    }
```

Tương tự như vậy cho ô Textbox của số thứ hai.

Tiếp theo sẽ viết sự kiện cho các Button là các phép toán, trong phần này các bạn cần sử dụng một phương thức chuyển đổi từ Text sang Number đó chính là Convert(). Trong phương thức này các bạn có thể chuyển đổi từ Text sang các số dạng Number như số nguyên, số thực,... .

```
1private void btn_plus_Click(object sender, EventArgs e)
2    {
3        int result = Convert.ToInt32(txt_num1.Text) + Convert.ToInt32(txt_num2.Text);
4        txt_result.Text = result.ToString();
5    }
```

Tương tự như vậy cho các phép toán khác, lưu ý ở phép toán chia các bạn cần đổi từ Text sang kiểu Double nhé, vì phép chia sẽ có kết quả là số thực.

```
1private void btn_chia_Click(object sender, EventArgs e)
2    {
```

```

3      Double result = Convert.ToDouble(txt_num1.Text) /
4      Convert.ToDouble(txt_num2.Text);
5      txt_result.Text = result.ToString();
    }

```

Kết quả:

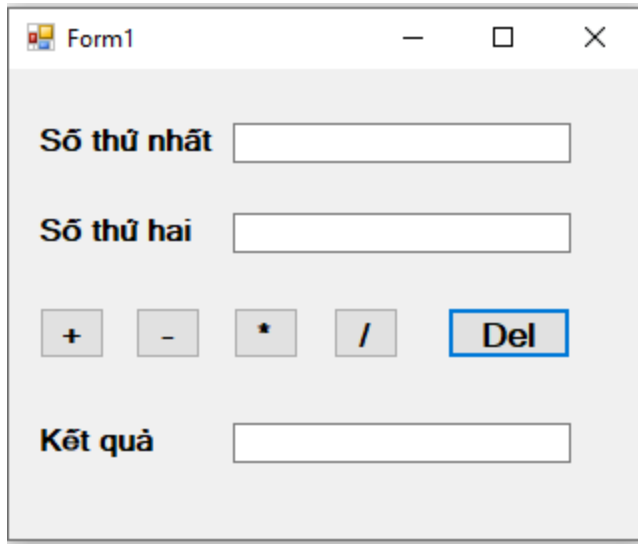
Và cuối cùng sẽ là sự kiện khi chúng ta nhấn vào nút Del thì các số ở ô Textbox sẽ được xóa hết, ta sẽ sử dụng phương thức `Clear()` để xóa đi các số trong Textbox.

```

1 private void btn_xoa_Click(object sender, EventArgs e)
2     {
3         txt_num1.Clear();
4         txt_num2.Clear();
5         txt_result.Clear();
6     }

```

Kết quả:



The screenshot shows a Windows application window titled "Form1". Inside the window, there is a simple calculator interface. It consists of two input fields for numbers, labeled "Số thứ nhất" (First number) and "Số thứ hai" (Second number). Below these are five buttons: a plus sign (+), a minus sign (-), a multiplication sign (*), a division sign (/), and a "Del" button. At the bottom, there is an output field labeled "Kết quả" (Result).

5. Kết luận

Như vậy là chúng ta đã cùng nhau tìm hiểu về một trong số các điều khiển thông thường Label, Button, Textbox. Vì đây là những điều khiển căn bản và được sử dụng rất nhiều nên các bạn hãy luyện tập thật nhiều để có thể sử dụng nó thành thạo nhé. Ở bài tiếp theo mình sẽ giới thiệu thêm một số điều khiển thông thường khác như là Checkbox - Radio Button.

Bài 3. Checkbox - RadioButton trong lập trình C# winforms

Trong bài này mình sẽ giới thiệu các bạn một trong số điều khiển thông thường của lập trình C# winforms đó chính là Checkbox và RadioButton. Đây là hai điều khiển được sử dụng rất nhiều, vì nó có thể trình bày đa dạng các giá trị trên Form.

Chúng ta sẽ cùng nhau tìm hiểu về Checkbox và RadioButton là gì? công dụng của nó như thế nào và nó có các thuộc tính và sự kiện gì? Sau đó mình sẽ viết một ứng dụng đơn giản sử dụng Checkbox và RadioButton.

Table of Content

- 1. Checkbox
- 2. RadioButton
- 3. Ví dụ sử dụng các điều khiển Checkbox và RadioButton
- 4. Kết luận

1. Checkbox

Checkbox là điều khiển cho phép người dùng chọn hoặc không chọn trên Form, người dùng có thể chọn một hoặc nhiều giá trị cùng lúc.



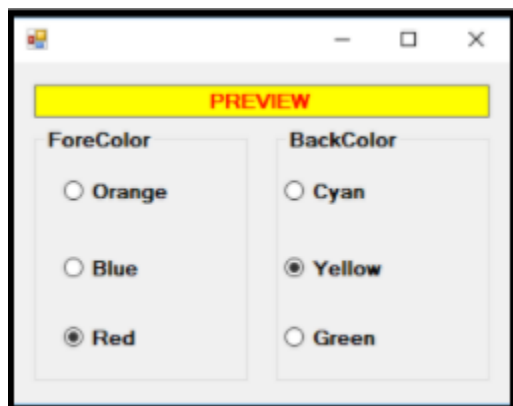
Chúng ta có một số thuộc tính của Checkbox dưới đây:

| Thuộc tính | Mô tả |
|------------|---|
| Checked | Giá trị True là được chọn, nếu thiết lập False thì không được chọn |
| CheckState | Thường dùng để kiểm tra tình trạng Checkbox có được chọn hay không. Mang 3 giá trị UnChecked, Checked và Indeterminate. - Checked: Điều khiển đang được chọn - UnChecked: Điều khiển không được chọn |

| Thuộc tính | Mô tả |
|-------------|---|
| | - Indeterminate: Điều khiển ở trạng thái không hoạt động |
| AutoCheck | Mang giá trị True hoặc False, nếu giá trị True thì cho phép người dùng nhấp chuột để chọn, nếu là False thì không cho phép người dùng nhấp chuột chọn |
| Text | Chuỗi văn bản hiển thị bên cạnh Checkbox |
| ThreeState | Mang giá trị True hoặc False; nếu là True thì cho phép Checkbox có 3 trạng thái: Checked, UnChecked, Indeterminate |
| RightToLeft | Mang giá trị Yes hoặc No; cho biết chuỗi văn bản hiển thị (thuộc tính Text) nằm bên trái hay bên phải của Checkbox |

2. RadioButton

RadioButton tương tự như Checkbox là điều khiển cho phép người dùng chọn hoặc không chọn, điểm khác biệt là với RadioButton người dùng chỉ có thể chọn một giá trị trong một nhóm.



Cả hai điều khiển Checkbox và RadioButton đều có chung một sự kiện đó chính là **CheckedChanged**, xảy ra khi thay đổi trạng thái chọn của Checkbox, RadioButton.

3. Ví dụ sử dụng các điều khiển Checkbox và RadioButton

Trong ví dụ này mình sẽ thực hiện một viết một ứng dụng sử dụng Checkbox và RadioButton, cụ thể như sau:

Thực hiện thiết kế giao diện cho Form như hình dưới đây:

Xử lý một số sự kiện sau:

- Khi click vào nút "**Sở thích của bạn**" thì những Checkbox được chọn bởi người dùng sẽ hiển thị trên hộp thoại MessageBox. Nếu không có Checkbox nào được chọn thì thông báo cho người dùng biết.
- Khi click vào nút "**Màu bạn thích**" thì RadioButton được chọn bởi người dùng sẽ được hiển thị trên hộp thoại MessageBox.

Chúng ta sẽ lần lượt thực hiện các yêu cầu nhé. Đầu tiên sẽ tạo giao diện cho Form giống như ứng dụng mẫu.

- Hai GroupBox (*ở bài sau mình sẽ hướng dẫn chi tiết*)
- 5 Checkbox với 5 sở thích tương ứng
- 5 RadioButton với 5 màu tương ứng
- 2 Button

Các bạn sẽ thay đổi một số thuộc tính như: Text, name, font,... cho giống với Form mẫu.

Sau khi thiết kế xong giao diện cho Form, bây giờ đến lúc các bạn xử lý sự kiện cho hai nút Button. Sự kiện đầu tiên là khi các bạn Click vào nút "**Sở thích của bạn**" thì các Checkbox được chọn sẽ hiển thị trên MessageBox.

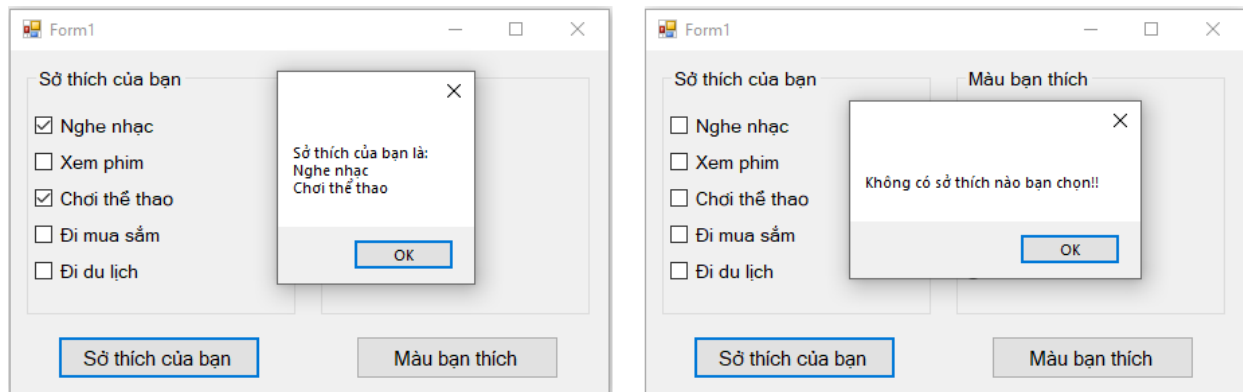
Các bạn Double Click vào Button cần xử lý để viết sự kiện. Chúng ta sẽ sử dụng thuộc tính Checked để kiểm tra xem Checkbox có được chọn hay không bằng cách cho **Checked == true**.

Ta sẽ tạo một biến **String str** để lưu các Checkbox được chọn, nếu Checked == true thì thêm vào str.

```
1 private void btn_othich_Click(object sender, EventArgs e)
2     {
3         string str = "";
4         if(chkbox_muasam.Checked == true)
5         {
6             str = chkbox_muasam.Text + "\n";
7         }
8         if(chkbox_nghenhac.Checked == true)
9         {
10            str = str + chkbox_nghenhac.Text + "\n";
11        }
12        if(chkbox_thethao.Checked == true)
13        {
14            str = str + chkbox_thethao.Text + "\n";
15        }
16        if(chkbox_xemphim.Checked == true)
17        {
18            str = str + chkbox_xemphim.Text + "\n";
19        }
20        if(chxbox_dulich.Checked == true)
21        {
22            str = str + chxbox_dulich.Text;
23        }
24
25        if(str.Length > 0)
26        {
27            MessageBox.Show("Sở thích của bạn là: \n" + str);
28        }
29        else
30        {
31            MessageBox.Show("Không có sở thích nào bạn chọn!!");
32        }
```


33 }

Kết quả:



Tiếp đến ta sẽ xử lý sự kiện khi Click vào nút "**Màu bạn thích**", tương tự như nút "**Sở thích của bạn**" ta sử dụng thuộc tính Checked để kiểm tra xem RadioButton có được chọn, nếu được chọn thì hiển thị ra MessageBox.

```

1 private void btn_mauthich_Click(object sender, EventArgs e)
2     {
3         string str = "";
4         if (rdbtn_do.Checked == true)
5         {
6             str = rdbtn_do.Text + "\n";
7         }
8         if (rdbtn_hong.Checked == true)
9         {
10            str = rdbtn_hong.Text + "\n";
11        }
12        if (rdbtn_tim.Checked == true)
13        {
14            str = rdbtn_tim.Text + "\n";
15        }
16        if (rdbtn_trang.Checked == true)
17        {
18            str = rdbtn_trang.Text + "\n";
19        }
20        if (rdbtn_vang.Checked == true)

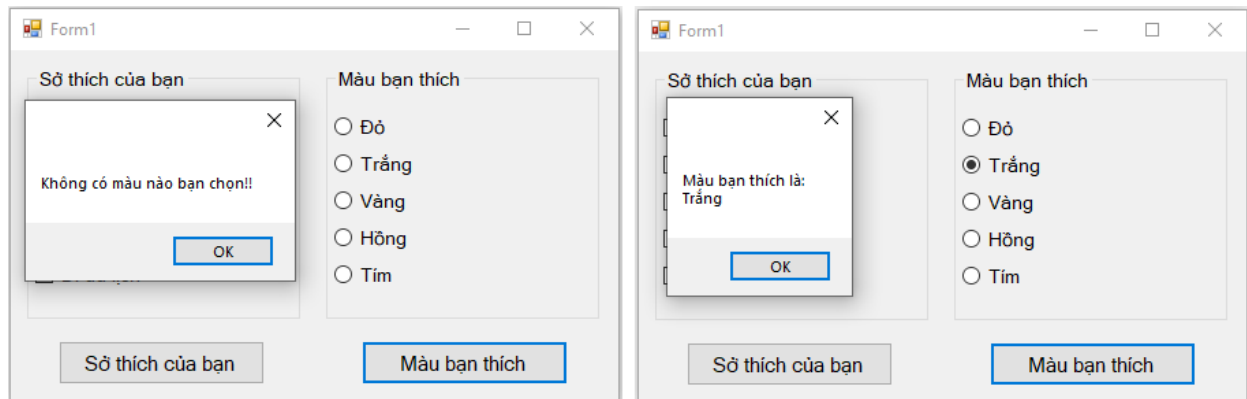
```

```

21      {
22          str = rdbtn_vang.Text;
23      }
24
25      if (str.Length > 0)
26      {
27          MessageBox.Show("Màu bạn thích là: \n" + str);
28      }
29      else
30      {
31          MessageBox.Show("Không có màu nào bạn chọn!!");
32      }
33  }

```

Kết quả:



4. Kết luận

Như vậy là chúng ta đã tìm hiểu xong về hai điều khiển thông thường trong winforms là Checkbox và RadioButton. Đây là hai điều khiển rất quan trọng vì vậy các bạn hãy luyện tập thật nhiều để có thể sử dụng nó thật thành thạo.

Bài 4. ComboBox - ListBox trong lập trình C# winforms

Trong hướng dẫn này mình sẽ giới thiệu các bạn một trong những điều khiển thông thường tiếp theo đó chính là ComboBox và ListBox.

ComboBox và **ListBox** là hai điều khiển có nhiều điểm tương đồng, đều sử dụng để chứa dữ liệu cho phép người dùng lựa chọn.

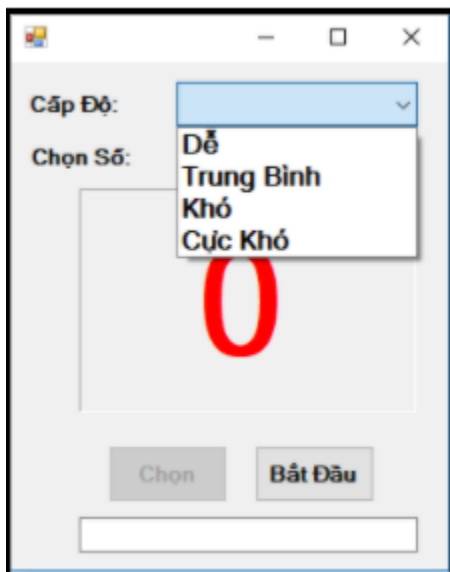
Chúng ta sẽ cùng nhau tìm hiểu về các khai niệm cũng như các sự kiện và thuộc tính của ComboBox và ListBox nhé.

Table of Content

- 1. ComboBox
- 2. ListBox
- 3. Ví dụ sử dụng các điều khiển ComboBox và ListBox
 - Ví dụ sử dụng ListBox
 - Ví dụ sử dụng ComboBox
- 4. Kết luận

1. ComboBox

ComboBox được dùng để hiển thị một danh sách những mỗi lần người dùng chỉ có thể chọn một lựa chọn, có thể nhập mới.



Một số thao tác với ComboBox:

- **Add()**: Thêm một mục chọn vào cuối danh sách ListBox.

- **Insert()**: Chèn thêm mục chọn vào vị trí **i**.
- **Count**: Trả về số mục chọn hiện đang có.
- **Item()**: Trả về mục chọn ở vị trí thứ **i**.
- **Remove()**: Bỏ mục chọn.
- **RemoveAt()**: Bỏ mục chọn ở vị trí thứ **i**.
- **Contains()**: Trả về True nếu có mục chọn trong danh sách, trả về False nếu không có mục chọn trong danh sách.
- **Clear**: Xóa tất cả các mục chọn.
- **IndexOf()**: Trả về vị trí mục chọn trong danh sách, nếu không tìm thấy sẽ trả về - 1.

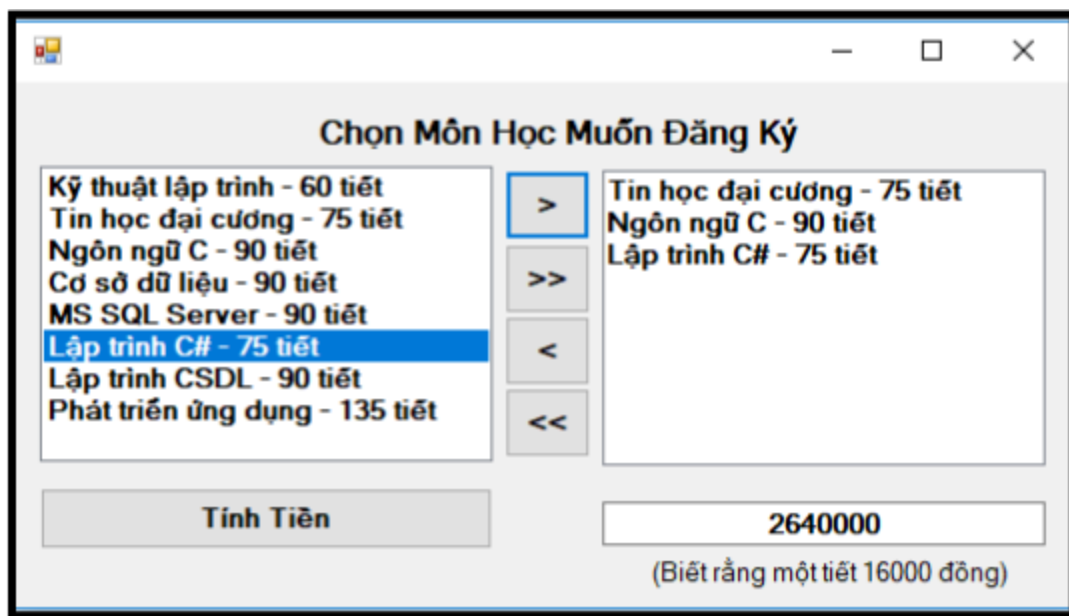
Một số thuộc tính thường dùng:

| Thuộc tính | Mô tả |
|-----------------------|---|
| Text | Trả về nội dung dòng dữ liệu đang hiển thị trên ComboBox |
| DropDownStyle | Quy định định dạng của ComboBox, nhận một trong các giá trị: - Simple: hiển thị theo dạng ListBox + TextBox có thể chọn dữ liệu từ ListBox hoặc nhập mới vào TextBox - DropDownList: Chỉ cho phép chọn dữ liệu trong ComboBox - DropDown: Giá trị mặc định, có thể chọn hoặc nhập mới mục dữ liệu vào ComboBox |
| Items | Trả về các mục chứa trong ComboBox |
| DropDownHeight | Thiết lập chiều cao tối đa khi sổ xuống của ComboBox |
| DropDownWidth | Thiết lập độ rộng của mục chọn trong ComboBox |
| SelectedIndex | Lưu chỉ số mục được chọn, chỉ số mục đầu tiên là 0 |
| SelectedItem | Trả về mục được chọn |
| SelectedText | Lấy chuỗi hiển thị của mục chọn trên ComboBox |
| DataSource | Chọn tập dữ liệu điền vào ComboBox. Tập dữ liệu có thể là mảng, chuỗi, ArrayList,... |
| DisplayMember | Gán dữ liệu thành viên sẽ hiển thị trên ComboBox |
| ValueMember | Thuộc tính này chỉ định dữ liệu thành viên sẽ cung cấp giá trị cho ComboBox |
| SelectedValue | Trả về giá trị của mục chọn (ValueMember) nếu ComboBox có liên kết dữ liệu. Nếu không liên kết dữ liệu hoặc ValueMember không được thiết lập thì giá trị SelectedValue là giá trị chuỗi của thuộc tính SelectedItem |

Trong **ComboBox** có một sự kiện là **SelectedIndexChanged**, sự kiện này xảy ra khi thay đổi mục chọn trong ComboBox.

2. ListBox

ListBox được dùng để hiển thị một danh sách các lựa chọn, người dùng có thể chọn một hoặc nhiều lựa chọn cùng lúc.



Một số thao tác với ListBox:

- **Add()**: Thêm một mục chọn vào cuối danh sách **ListBox**.
- **Insert()**: Chèn thêm mục vào vị trí **i**.
- **Count**: Trả về số mục chọn hiện đang có.
- **Item()**: Trả về mục chọn ở vị trí **i**.
- **Remove()**: Bỏ mục chọn.
- **RemoveAt()**: Bỏ mục chọn tại vị trí **i**.
- **Contains()**: Trả về True nếu có mục chọn trong danh sách và trả về False nếu không có mục chọn trong danh sách.
- **Clear**: Xóa tất cả các mục chọn.
- **IndexOf()**: Trả về vị trí mục chọn trong danh sách, nếu không tìm thấy sẽ trả về -1.

Một số thuộc tính của ListBox thường dùng:

| Thuộc tính | Mô tả |
|----------------------|--|
| DataSource | Chọn tập dữ liệu điền vào ListBox. Tập dữ liệu có thể là mảng, chuỗi, ArrayList,... |
| DisplayMember | Dữ liệu thành viên sẽ được hiển thị trên ListBox |
| ValueMember | Thuộc tính này chỉ định dữ liệu thành viên sẽ cung cấp giá trị cho ListBox |
| SelectedValue | Trả về giá trị của mục chọn nếu ListBox có liên kết dữ liệu. Nếu không liên kết với dữ liệu hoặc thuộc tính ValueMember không được thiết lập |

| Thuộc tính | Mô tả |
|----------------------|---|
| | thì giá trị thuộc tính <code>SelectedValue</code> là giá trị chuỗi của thuộc tính <code>SelectedItem</code> |
| Items | Các mục chứa trong <code>ListBox</code> |
| SelectedItem | Trả về mục được chọn |
| SelectedIndex | Lấy chỉ số mục được chọn, chỉ số mục chọn đầu tiên là 0 |
| SelectionMode | Cho phép chọn một hoặc nhiều dòng dữ liệu trên <code>ListBox</code> , bao gồm: <ul style="list-style-type: none"> - <i>One</i>: Chỉ chọn một giá trị - <i>MultiSimple</i>: Cho phép chọn nhiều, chọn bằng cách Click vào mục chọn, bỏ chọn bằng cách Click vào mục đã chọn - <i>MultiExtended</i>: Chọn nhiều bằng cách nhấn kết hợp với Shift hoặc Ctrl |
| SelectedItems | Được sử dụng khi <code>SelectionMode</code> là <code>MultiSimple</code> hoặc <code>MultiExtended</code> . Thuộc tính <code>SelectedItems</code> chứa các chỉ số của các dòng dữ liệu được chọn |
| SelectedItems | Được sử dụng khi <code>SelectionMode</code> là <code>MultiSimple</code> hoặc <code>MultiExtended</code> . Thuộc tính <code>SelectedItems</code> chứa các chỉ số của các dòng dữ liệu được chọn |

Trong `ListBox` có một sự kiện được sử dụng rất nhiều đó chính là ***SelectedIndexChanged***, sự kiện này xảy ra khi thay đổi mục chọn trong `ListBox`.

3. Ví dụ sử dụng các điều khiển `ComboBox` và `ListBox`

Trong phần này mình sẽ thực hiện tạo hai ứng dụng sử dụng **`ComboBox`** và **`ListBox`**, các bạn chú ý theo dõi nhé.

Ví dụ sử dụng `ListBox`

Ở ví dụ này, mình sẽ thực hiện tạo giao diện cho Form như mẫu dưới đây, sau đó viết các sự kiện cho các điều khiển `Button`.

Form1

KHAI BÁO Y TẾ ĐIỆN TỬ

Họ tên

Tổng số người đã khai báo

Yêu cầu:

- Nhấn vào Button "**Nhập thông tin**" thì nội dung trong ô TextBox "Họ tên" sẽ được thêm vào ListBox.
- Nhấn vào Button "**Xóa thông tin đang chọn**" sẽ xóa nội dung họ tên đang chọn trong ListBox.
- Nhấn vào Button "**Xóa thông tin đầu**" sẽ xóa nội dung họ tên đầu tiên trong ListBox.
- Nhấn vào Button "**Xóa thông tin cuối**" sẽ xóa nội dung họ tên cuối cùng trong ListBox.
- Nhấn vào Button "**Xóa tất cả thông tin**" sẽ xóa toàn bộ nội dung có bên trong ListBox.
- Mỗi khi thêm hoặc xóa thông tin người khai báo trong ListBox, "**Tổng số người đã khai báo**" sẽ được cập nhật lại.

Các bước thực hiện:

Bước 1: Thực hiện tạo giao diện cho Form giống như mẫu, bao gồm:

- 2 ô TextBox để nhập và hiển thị thông tin.
- 5 Button với 5 sự kiện tương ứng.

- 1 ListBox để hiển thị các khai báo.

Bước 2: Xử lý sự kiện trên Button "Nhập thông tin".

Để xử lý cho Button này, ta sẽ Click Double vào Button để sang cửa sổ xử lý sự kiện. Ta sẽ lần lượt xử lý sự kiện theo yêu cầu của đề bài:

- Sử dụng **String.IsNullOrEmpty()** để đặt điều kiện cho ô TextBox không được để trống, nếu TextBox trống thì hiển thị hộp thoại cho người dùng biết.
- Sử dụng **Items.Add()** để thêm nội dung từ ô TextBox vào ListBox
- Sử dụng **Items.Count** để đếm tổng số phần tử có trong ListBox và đưa nó vào ô TextBox hiển thị.
- Sử dụng **Clear()** để xóa hết các thông tin trong ô TextBox nhập sau khi đã thêm vào ListBox.

```

1 private void btn_nhap_Click(object sender, EventArgs e)
2     {
3         if (!String.IsNullOrEmpty(txt_nhap.Text))
4         {
5             lstBox_hienthi.Items.Add(txt_nhap.Text);
6             txt_hienthi.Text = lstBox_hienthi.Items.Count.ToString();
7             txt_nhap.Clear();
8             txt_nhap.Focus();
9         }
10        else
11            MessageBox.Show("Vui lòng điền đầy đủ thông tin !!!");
12    }

```

Bước 3: Xử lý sự kiện trên Button "Xóa thông tin đang chọn".

Trong Button này, ta cần xét điều kiện nếu trong ListBox đã tồn tại phần tử thì khi đó chúng ta mới thực hiện xóa. Ta sẽ sử dụng **Items.Remove** tại vị trí **SelectedItems[0]** để xóa phần tử đang được chọn.

Sau khi xóa xong ta cần cập nhật lại tổng số phần tử ở trong ô TextBox hiển thị.

```

1 private void btn_xoadangchon_Click(object sender, EventArgs e)
2     {

```



```

3      int a = Convert.ToInt32(txt_hienthi.Text);
4      if(lstBox_hienthi.SelectedItems.Count != 0)
5      {
6          lstBox_hienthi.Items.Remove(lstBox_hienthi.SelectedItems[0]);
7      }
8      txt_hienthi.Text = (a - 1).ToString();
9  }

```

Bước 4: Xử lý sự kiện trên Button "**Xóa thông tin đầu**" và trên Button "**Xóa thông tin cuối**".

Về cơ bản ở hai Button này việc xử lý sự kiện khá giống nhau, đều sử dụng **Items.RemoveAt()** để thực hiện xóa đầu và xóa cuối.

```

1private void btn_xoadau_Click(object sender, EventArgs e)
2  {
3      int a = Convert.ToInt32(txt_hienthi.Text);
4      lstBox_hienthi.Items.RemoveAt(0);
5      txt_hienthi.Text = (a - 1).ToString();
6  }
1private void btn_xoacuo_Click(object sender, EventArgs e)
2  {
3      int a = Convert.ToInt32(txt_hienthi.Text);
4      lstBox_hienthi.Items.RemoveAt(lstBox_hienthi.Items.Count - 1);
5      txt_hienthi.Text = (a - 1).ToString();
6  }

```

Bước 5: Xử lý sự kiện trên Button "**Xóa tất cả thông tin**".

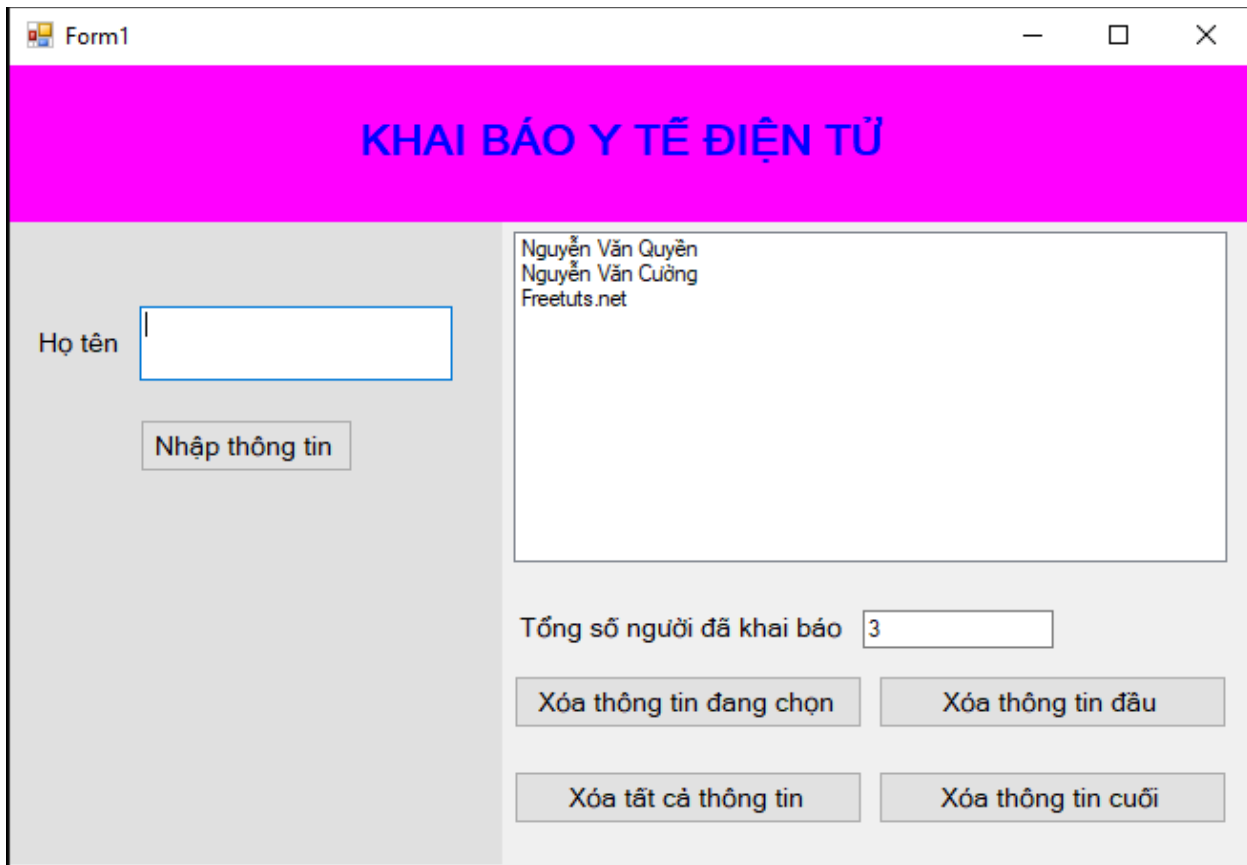
Trong Button này khá đơn giản, ta sử dụng **Items.Clear()** để xóa toàn bộ phần tử trong ListBox, sau đó cập nhật lại ô TextBox hiển thị là "**0**".

```

1private void btn_xoatatca_Click(object sender, EventArgs e)
2  {
3      lstBox_hienthi.Items.Clear();
4      txt_hienthi.Text = "0";
5  }

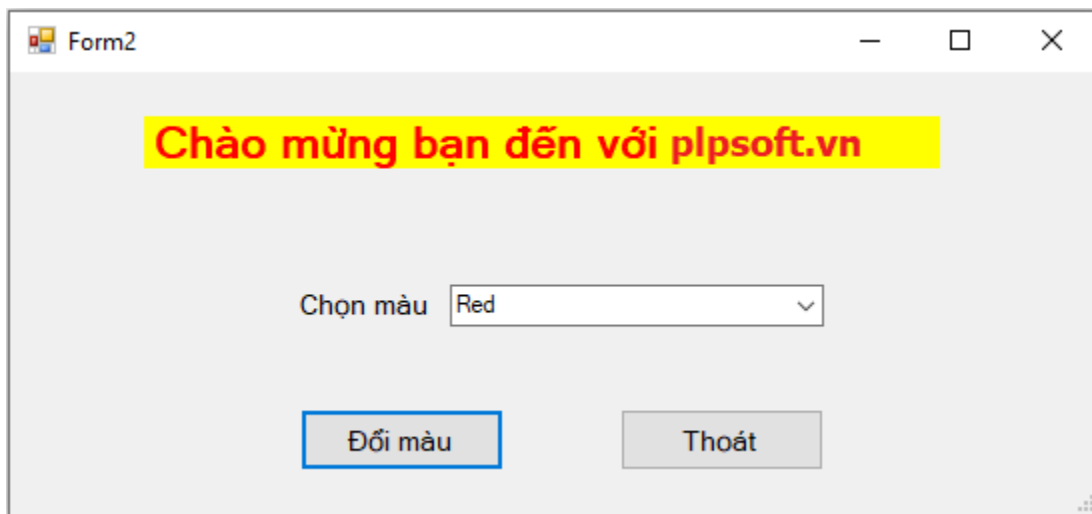
```

Kết quả: Sau khi thực hiện xử lý các sự kiện, các bạn có thể kiểm tra nó, dưới đây mình sẽ Demo Button "**Nhập thông tin**", còn các Button khác các bạn có thể tự kiểm tra nhé.



Ví dụ sử dụng ComboBox

Ở ví dụ này mình sẽ viết một ứng dụng sử dụng **ComboBox** với giao diện như mẫu dưới đây, kèm theo các sự kiện.



Yêu cầu:

- Xây dựng ứng dụng với giao diện như trên gồm ComboBox chứa danh sách 4 màu (*Yellow, Red, Blue, Black*) và một Label hiển thị dòng chữ "**Chào mừng bạn đến với plpsoft.vn**".
- Khi chọn màu trong ComboBox và nhấn vào nút Button "**Đổi màu**" thì màu của dòng chữ và màu nền của Label sẽ thay đổi tương ứng.
- Nhấn vào nút Button "**Thoát**" xác nhận người dùng muốn đóng Form hiện hành. Nếu chọn **YES** thì kết thúc chương trình thực thi, nếu chọn **NO** thì hủy bỏ lệnh kết thúc.

Các bước thực hiện:

Bước 1: Xây dựng giao diện Form như mẫu, bao gồm:

- 2 Label để hiển thị dòng chữ.
- 1 ComboBox với 4 Items là 4 màu (*Yellow, Red, Blue, Black*).
- 2 Button với hai sự kiện là đổi màu và thoát.

Bước 2: Viết sự kiện cho Button "**Đổi màu**".

Trong Button này, ta chỉ cần sử dụng **.Text** để kiểm tra, nếu màu đang chọn là Yellow thì đổi **ForeColor** thành Yellow, tương tự như vậy cho 3 màu còn lại.

```

1 private void btn_doimau_Click(object sender, EventArgs e)
2     {
3         if(cbo_color.Text == "Yellow")
4         {
5             lb_doimau.BackColor = Color.Red;
6             lb_doimau.ForeColor = Color.Yellow;
7         }
8         if (cbo_color.Text == "Red")
9         {
10            lb_doimau.BackColor = Color.Yellow;
11            lb_doimau.ForeColor = Color.Red;
12        }
13        if (cbo_color.Text == "Blue")
14        {
15            lb_doimau.BackColor = Color.Yellow;
16            lb_doimau.ForeColor = Color.Blue;
17        }

```

```

18     if (cbo_color.Text == "Black")
19     {
20         lb_doimau.BackColor = Color.White;
21         lb_doimau.ForeColor = Color.Black;
22     }
23 }

```

Bước 3: Viết sự kiện cho Button "**Thoát**".

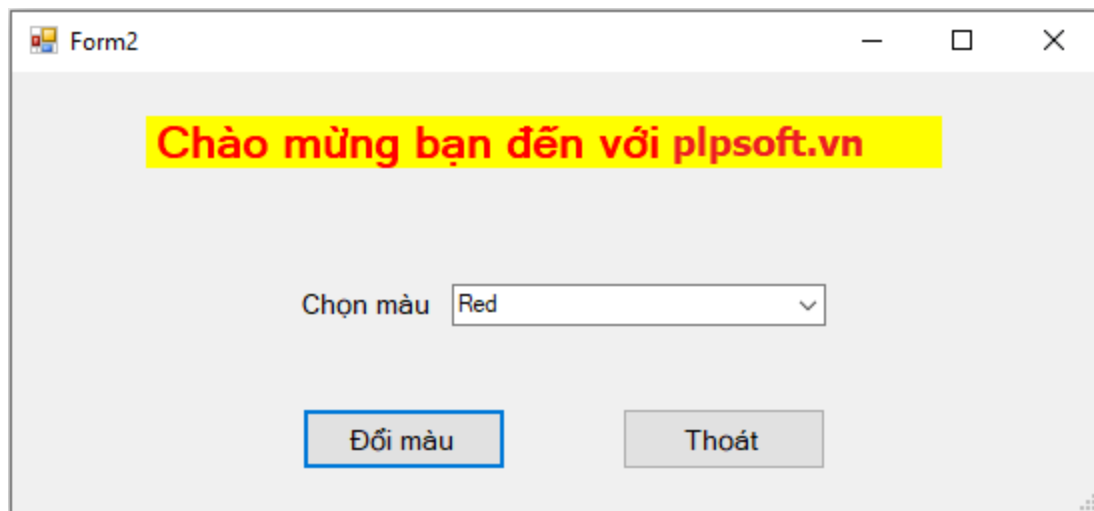
Ở các bài trước mình đã có nói về sự kiện trong Button "**Thoát**", đơn giản ta chỉ cần sử dụng `MessageBox.Show()` kết hợp với `Application.Exit()` để tạo hộp thoại hỏi người dùng có muốn thoát hay không.

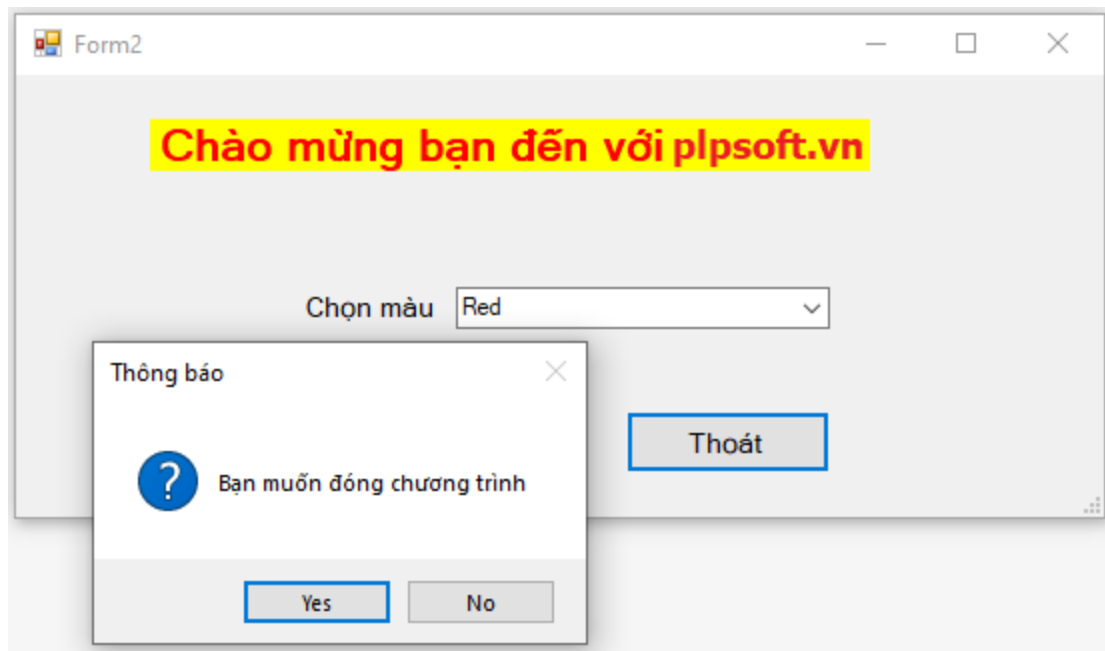
```

1 private void btn_thoat_Click(object sender, EventArgs e)
2 {
3     DialogResult dg = MessageBox.Show("Bạn muốn đóng chương trình", "Thông
4 báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
5     if(dg == DialogResult.Yes)
6     {
7         Application.Exit();
8     }
9 }

```

Kết quả:





4. Kết luận

Như vậy là chúng ta đã tìm hiểu xong về hai điều khiển thông thường trong winforms đó chính là ComboBox và ListBox. Đây cũng là hai điều khiển cuối cùng trong Series điều khiển thông thường. Ở các bài tiếp theo chúng ta sẽ cùng tìm hiểu về các điều khiển đặc biệt trong winforms.

Bài 5. ToolTip - HelpProvider - ErrorProvider trong C# winforms

Trong hướng dẫn này mình sẽ giới thiệu các bạn một trong số các điều khiển đặc biệt trong lập trình C# winforms, cụ thể là ToolTip, HelpProvider, ErrorProvider.

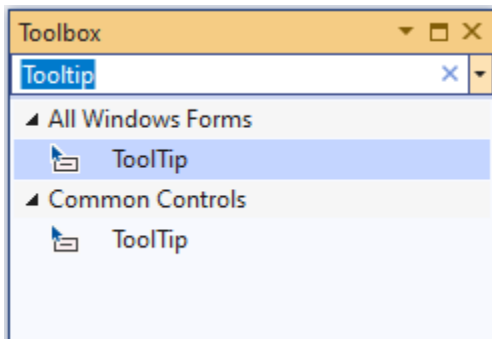
Đây là các điều khiển rất quan trọng để hoàn thiện một ứng dụng với đầy đủ các chức năng, vì vậy hãy nắm thật rõ nhé. Chúng ta sẽ cùng nhau tìm hiểu về công dụng của nó cũng như các sự kiện và phương thức của nó.

Table of Content

- 1. ToolTip
- 2. HelpProvider
- 3. ErrorProvider
- 4. Ví dụ sử dụng các điều khiển đặc biệt ToolTip, HelpProvider, ErrorProvider
- 5. Kết luận

1. ToolTip

ToolTip là điều khiển cho phép hiển thị các thông tin chú thích khi người dùng đưa chuột qua các điều khiển có thiết lập ToolTip.



Một số thuộc tính thường dùng của ToolTip:

| Thuộc tính | Mô tả |
|-----------------------|---|
| Active | Mang giá trị True hoặc False, nếu thiết lập True thì ToolTip có hiệu lực hiển thị thông báo, nếu mang giá trị False thì ToolTip không hiển thị được thông báo |
| AutomaticDelay | Thiết lập thời gian xuất hiện ToolTip khi vừa đưa chuột đến điều khiển, thời gian tính bằng mili giây |
| AutoPopDelay | Thời gian hiển thị ToolTip cho đến khi kết thúc khi người dùng đã đưa chuột đến điều khiển, thời gian tính bằng mili giây |
| IsBalloon | Quy định kiểu hiển thị của ToolTip (<i>False/True</i>) |

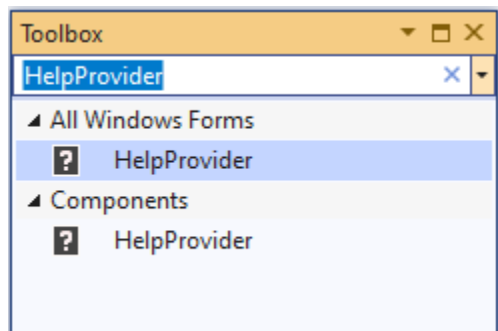
| Thuộc tính | Mô tả |
|---------------------|--|
| ReshowDelay | Thời gian mà ToolTip tắt từ khi người dùng đưa chuột ra khỏi điều khiển, thời gian tính bằng mili giây |
| ToolTipIcon | Biểu tượng xuất hiện bên cạnh chuỗi khai báo trong thuộc tính ToolTipTitle |
| ToolTipTitle | Chuỗi hiển thị bên cạnh biểu tượng ToolTipIcon |
| UseAnimation | Thiết lập hiệu ứng ảnh động được biểu diễn khi ToolTip được hiển thị |
| UseFading | Thiết lập hiệu ứng mờ dần được biểu diễn khi ToolTip hiển thị |

Một số phương thức thường dùng của ToolTip:

| Phương thức | Mô tả |
|---------------------|--|
| SetToolTip() | Thiết lập chuỗi hiển thị của ToolTip trên điều khiển |
| GetToolTip() | Biểu tượng xuất hiện bên cạnh chuỗi khai báo trong thuộc tính ToolTipTitle |
| Clear() | Loại bỏ tất cả ToolTipText cho các điều khiển trên Form |

2. HelpProvider

Điều khiển **HelpProvider** cung cấp cửa sổ trợ giúp cho điều khiển. Với những ứng dụng có sử dụng HelpProvider, người dùng có thể gọi sự trợ giúp bằng cách ấn phím F1.



Một số thuộc tính thường dùng của HelpProvider:

| Thuộc tính | Mô tả |
|----------------------|---|
| HelpNamespace | Chỉ định tên tập trình trợ giúp định dạng chm hoặc html |
| HelpKeyword | Từ khóa tìm kiếm, từ khóa này là chỉ mục hoặc chủ đề được truyền vào tập tin tìm kiếm. Thuộc tính HelpNavigator sẽ quy định từ khóa này tìm kiếm theo chủ đề hay theo chỉ mục |
| HelpString | Hiển thị chuỗi trợ giúp cho điều khiển. Nếu thuộc tính HelpProvider không được thiết lập thì khi người dùng nhấn F1 sẽ hiển thị chuỗi trợ giúp này |
| ShowHelp | Nếu thiết lập giá trị True thì cho phép nội dung trợ giúp hiển thị trên một điều khiển nào đó. Nếu thiết lập giá trị False thì không hiển thị được |

| Thuộc tính | Mô tả |
|----------------------|---|
| HelpNavigator | <p>Thiết lập cách thức hiển thị của tập tin trợ giúp. Gồm các thuộc tính thành viên như:</p> <ul style="list-style-type: none"> • <i>AssociateIndex</i>: Mở tập tin trợ giúp và hiển thị danh sách chỉ mục có ký tự trùng với ký tự đầu tiên trong thuộc tính HelpKeyWorf • <i>Find</i>: Giúp hiển thị nội dung trợ giúp có chuỗi ký tự trùng với từ khóa trong thuộc tính HelpKeyWord • <i>Index</i>: Hiển thị danh mục các chỉ mục trong tập tin trợ giúp • <i>KeywordIndex</i>: Hiển thị danh mục là các chỉ mục như từ khóa trong thuộc tính HelpKeyWord • <i>TableOfContentsL</i> Hiển thị tất cả cá nội dung trong tập tin trợ giúp • <i>Topic</i>: Hiển thị danh mục các chủ đề trong tập tin trợ giúp, áp dụng với tập tin có hỗ trợ danh mục các chủ đề • <i>TopicId</i>: Hiển thị chủ đề có mã trùng với mã chủ đề chỉ định, áp dụng với tập tin có hỗ trợ danh mục các chủ đề và các chủ đề được đánh số. |

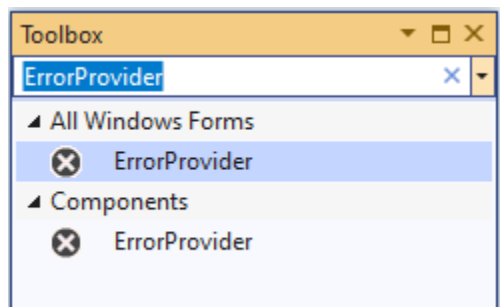
Một số phương thức thương dùng của HelpProvider:

| Phương thức | Mô tả |
|-------------------------|--|
| SetHelpKeyWord | Thiết lập giá trị cho thuộc tính HelpKeyWord |
| SetHelpNavigator | Thiết lập giá trị cho thuộc tính HelpNavigator |
| SetHelpString | Thiết lập giá trị cho thuộc tính HelpString |
| SetShowHelp | Thiết lập giá trị cho thuộc tính ShowHelp |

3. ErrorProvider

ErrorProvider giúp báo cho người dùng biết thông tin lỗi của điều khiển trên Form.

Thông thường khi điều khiển trên Form lỗi, ErrorProvider sẽ cung cấp một biểu tượng để thông báo lỗi bên cạnh điều khiển đó.



Một số thuộc tính có trong ErrorProvider:

| Thuộc tính | Mô tả |
|-------------------|---|
| Icon | Chọn biểu tượng thể hiện lỗi của điều khiển |
| BlinkRate | Tốc độ nhấp nháy của biểu tượng trong thuộc tính Icon. Tốc độ tính theo mili giây |
| BlinkStyle | Kiểu nhấp nháy của biểu tượng. Nếu thiết lập giá trị NeverBlink thì biểu tượng sẽ hiển thị mà không nhấp nháy |

Một số phương thức của **ErrorProvider**:

| Phương thức | Mô tả |
|---|---|
| SetError (<Điều khiển>, <Thông báo lỗi>) | Giúp hiển thị lỗi và thông báo lỗi của điều khiển. Thông báo lỗi hiển thị dưới dạng ToolTip |
| Clear() | Xóa biểu tượng ErrorProvider của điều khiển tương ứng trên Form |
| GetError() | Lấy chuỗi thông báo lỗi của điều khiển |

4. Ví dụ sử dụng các điều khiển đặc biệt **ToolTrip**, **HelpProvider**, **ErrorProvider**

Trong phần này mình sẽ thực hiện viết một ứng dụng sử dụng 3 điều khiển đặc biệt trên, cụ thể là thiết kế giao diện như dưới đây sau đó xử lý một số sự kiện theo yêu cầu.

Yêu cầu:

- Thiết kế giao diện cho Form giống như Form mẫu ở trên.
- Khi rê chuột vào ô TextBox "**Tên đăng nhập**" thì hiển thị dòng ghi chú "Chỉ được nhập ký tự a-z và 0-9".

- Khi rê chuột vào ô TextBox "**Mật khẩu**" thì hiển thị dòng ghi chú "Chỉ được nhập ký tự từ 0-9".
- Khi load Form thì nội dung trong ô TextBox "**Mật khẩu**" sẽ được mã hóa thành dấu *, khi Tick vào ô "**Hiển thị mật khẩu**" thì mật khẩu sẽ được hiển thị.
- Xử lý nút Button "**Đăng nhập**", nếu hai ô TextBox ở trên không để trống thì thông báo đăng nhập thành công, ngược lại nếu một trong hai để trống thì thông báo điền đầy đủ thông tin.
- Xử lý nút Button "**Thoát**" để thoát khỏi chương trình.
- Khi nhấn F1 thì sẽ hiển thị trang hướng dẫn theo đường link (<https://plpsoft.vn/30236-Bai-tap-C-Bai-5-Su-dung-ToolTip-HelpProvider-ErrorProvider-trong-C-windows-Form>).

Các bước thực hiện:

Bước 1: Thiết kế giao diện như Form mẫu, bao gồm:

- 4 Label với nội dung hiển thị tương ứng.
- 2 TextBox để nhập tên đăng nhập và mật khẩu.
- 1 CheckBox để hiển thị mật khẩu.
- 2 Button để đăng nhập và thoát.
- 1 ToolTip để hiển thị thông báo lúc rê chuột vào ô TextBox.
- 1 HelpProvider để tạo đường dẫn hướng dẫn khi nhấn F1.

Bước 2: Xử lý trên nút Button "Đăng nhập".

Sử dụng `string.IsNullOrEmpty()` để kiểm tra điều kiện ô TextBox không được để trống.

```

1 private void btn_dangnhap_Click(object sender, EventArgs e)
2     {
3         if (!string.IsNullOrEmpty(txt_tendangnhap.Text) ||
4             !string.IsNullOrEmpty(txt_matkhau.Text))
5             {
6                 MessageBox.Show("Đăng nhập thành công !!!");
7             }
8         else
9             {
10                MessageBox.Show("Vui lòng điền đầy đủ thông tin !!!");
11            }
12    }

```

Bước 3: Xử lý trên ô TextBox "Mật khẩu".

Trong TextBox ta sử dụng sự kiện KeyPress để đặt điều kiện cho TextBox.

```
1 private void txt_matkhau_KeyPress(object sender, KeyPressEventArgs e)
2     {
3         if (!(e.KeyChar >= '0' && e.KeyChar <= '9' || e.KeyChar == (char)8))
4             e.Handled = true;
5     }
```

Bước 4: Xử lý trên ô CheckBox.

- Sử dụng Checked để kiểm tra xem ô CheckBox có được tick hay không.
- Sử dụng PasswordChar để mã hóa cho ô TextBox "Mật khẩu".

***Lưu ý:** Trong phần Properties của ô TextBox "Mật khẩu" các bạn cần điền dấu "*" vào thuộc tính PasswordChar.

```
1 private void chk_hienthi_CheckedChanged(object sender, EventArgs e)
2     {
3         if (chk_hienthi.Checked == true)
4         {
5             txt_matkhau.PasswordChar = (char)0;
6         }
7         else
8         {
9             txt_matkhau.PasswordChar = '*';
10        }
11    }
```

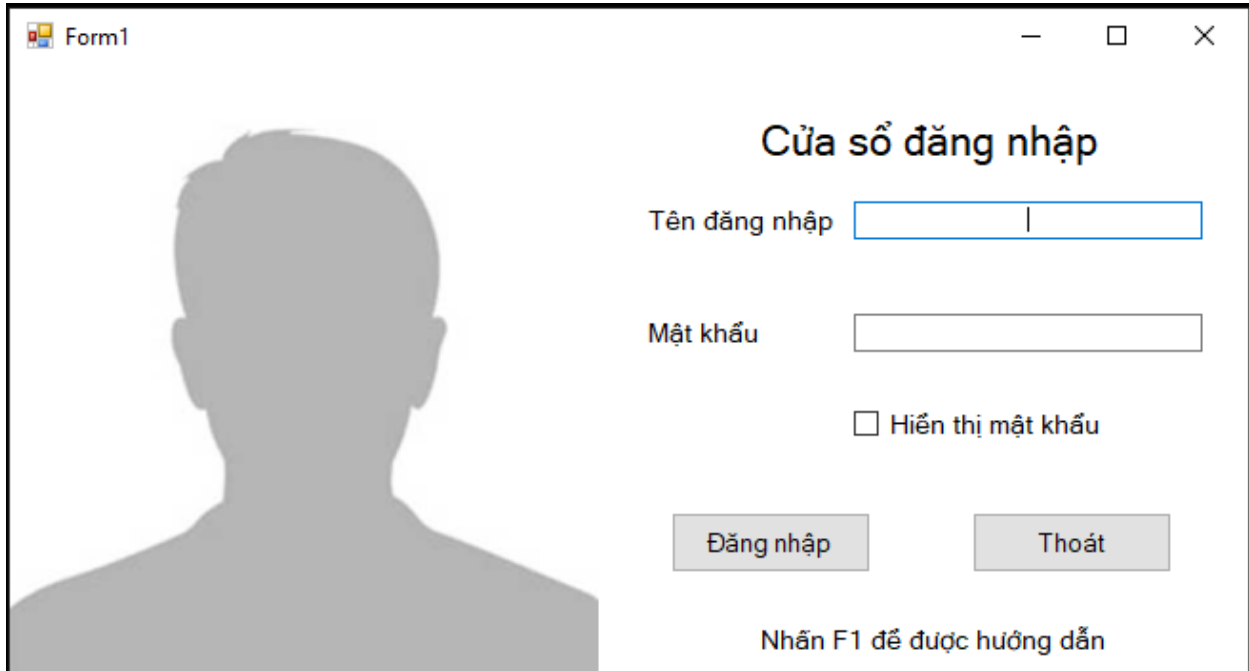
Bước 5: Xử lý khi rê chuột vào ô TextBox và khi nhấn F1 sẽ hiển thị trang hướng dẫn.

Ta sẽ viết sự kiện ở FormLoad. Sử dụng **SetToolTip** để viết sự kiện lúc rê chuột vào TextBox và sử dụng **HelpNamespace** để viết sự kiện khi nhấn **F1**.

```
1 private void Form1_Load(object sender, EventArgs e)
2     {
3         toolTip1.SetToolTip(txt_tendangnhap, "Chỉ được nhập ký tự a-z và 0-9");
4         toolTip1.SetToolTip(txt_matkhau, "Chỉ được nhập ký tự từ 0-9");
5     }
```

6 `helpProvider1.HelpNamespace = "https://plpsoft.vn/30236-Bai-tap-C-Bai-5-Su-dung-ToolTip-HelpProvider-ErrorProvider-trong-C-windows-Form";`
 `}`

Kết quả:



5. Kết luận

Như vậy là chúng ta đã tìm hiểu xong về ba điều khiển đặc biệt trong C# winforms. Đây là một trong các điều khiển rất quan trọng vì vậy hãy nắm rõ nó bằng cách luyện tập thật nhiều.

Bài 6. Cách dùng ProgressBar - Timer trong C# winforms

Trong hướng dẫn này mình sẽ giới thiệu các bạn một trong các điều khiển đặc biệt tiếp theo đó chính là ProgressBar và Timer. Đây là hai điều khiển được sử dụng khá phổ biến vì nó liên quan đến thời gian.

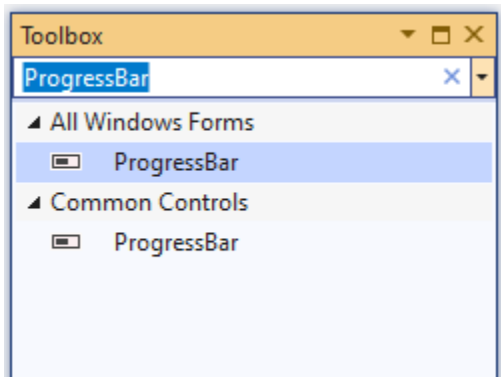
Chúng ta sẽ cùng nhau tìm hiểu về công dụng cũng như là các thuộc tính và phương thức của ProgressBar, Timer. Sau đó mình sẽ thực hiện một chương trình áp dụng hai điều khiển trên.

Table of Content

- 1. ProgressBar
- 2. Timer
- 3. Ví dụ sử dụng các điều khiển ProgressBar và Timer
- 4. Kết luận

1. ProgressBar

ProgressBar được sử dụng để hiển thị thời gian thực hiện của một công việc nào đó.



Một số thuộc tính của ProgressBar:

| Thuộc tính | Mô tả |
|------------|---|
| Maximum | Giá trị tối đa của ProgressBar. Khi ProgressBar được lấp đầy nghĩa là ProgressBar đã đạt giá trị Maximum |
| Minimum | Giá trị nhỏ nhất của ProgressBar. Khi ProgressBar trông rỗng nghĩa là ProgressBar đang có giá trị Minimum |
| Value | Giữ giá trị hiện tại của ProgressBar, giá trị này nằm trong đoạn Minimum và Maximum |
| Style | Kiểu hiển thị của ProgressBar |

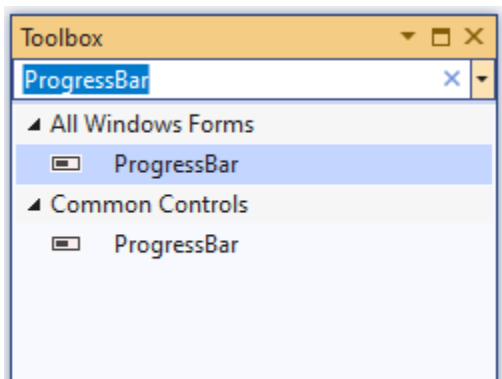
| Thuộc tính | Mô tả |
|------------|---|
| Step | Lượng giá trị thêm vào Value khi phương thức PerformStep() được gọi |

Một số phương thức của ProgressBar:

| Phương thức | Mô tả |
|----------------------|---|
| PerformStep() | Phương thức giúp tăng ProgressBar. Giá trị tăng là giá trị được thiết lập trong thuộc tính Step |
| Increment(<giá trị>) | Phương thức giúp tăng ProgressBar. Giá trị tăng là tham số đầu vào <giá trị> của phương thức |

2. Timer

Điều khiển Timer cho phép thực thi lại một hành động sau một khoảng thời gian xác định.



Một số thuộc tính của Timer:

| Thuộc tính | Mô tả |
|------------|--|
| Interval | Thiết lập giá trị là một số nguyên. Giá trị nguyên này là thời lượng của một chu kỳ tính |
| Enable | Thiết lập giá trị True hoặc False. Nếu là giá trị True thì điều khiển Timer hoạt động, nếu là giá trị False thì điều khiển Timer không hoạt động |

Một số phương thức thường dùng của Timer:

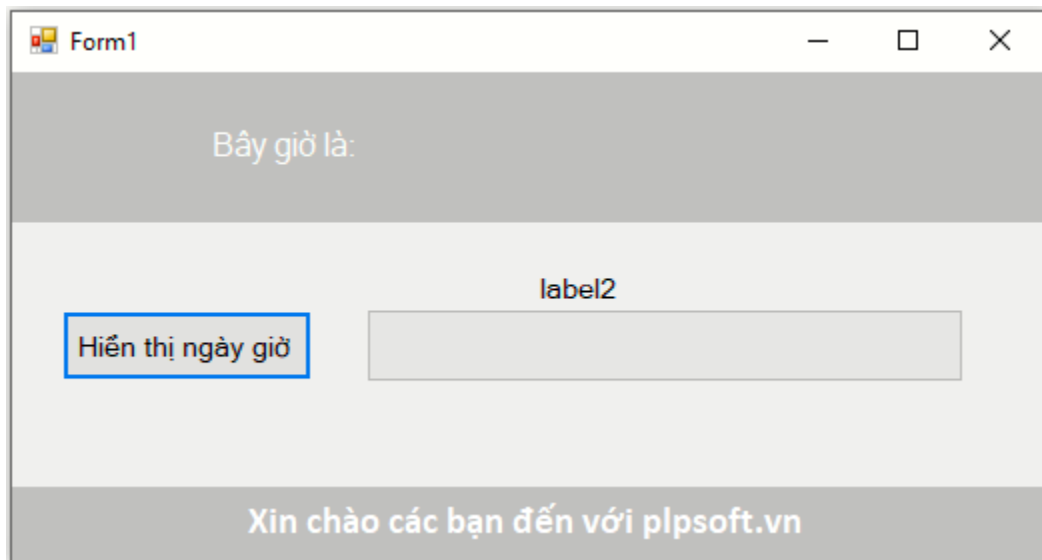
| Phương thức | Mô tả |
|-------------|--|
| Start() | Start() kích hoạt điều khiển Timer hoạt động. Phương thức này tương ứng với việc thiết lập giá trị thuộc tính Enable là True |
| Stop() | Dừng hoạt động của điều khiển Timer. Phương thức này tương ứng với việc thiết lập giá trị thuộc tính Enable là False |

Trong điều khiển Timer chỉ có một sự kiện đó là Tick:

| Sự kiện | Mô tả |
|---------|--|
| Tick | Tick là sự kiện được gọi trong mỗi chu kỳ Interval |

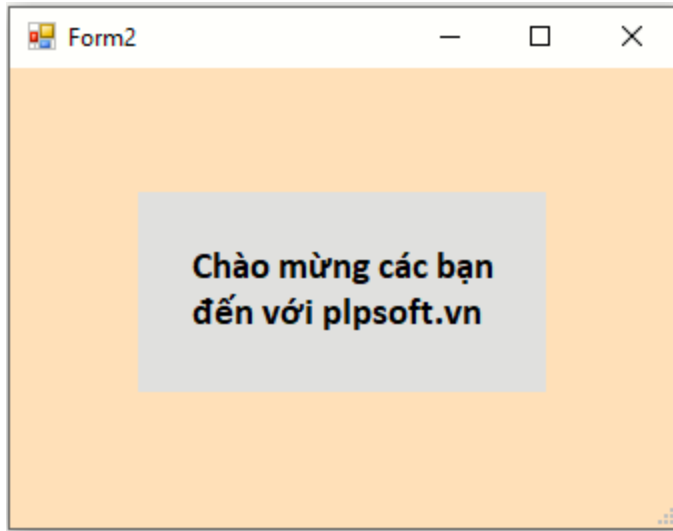
3. Ví dụ sử dụng các điều khiển ProgressBar và Timer

Trong ví dụ này mình sẽ thực hiện viết một chương trình áp dụng hai điều khiển ProgressBar và Timer, cụ thể sẽ tạo giao diện cho Form như sau rồi thực hiện một số sự kiện.



Xử lý một số sự kiện sau:

- Sử dụng Timer để hiển thị ngày giờ hiện tại.
- Sử dụng ProgressBar để hiển thị % công việc đang thực hiện.
- Xử lý sự kiện cho nút Button, khi click vào thì ngày giờ hiện tại sẽ hiển thị ra và thanh ProgressBar sẽ hiển thị % thực hiện công việc. Khi đạt đến 100% thì load Form hai.



Bây giờ chúng ta sẽ bắt đầu tạo giao diện cho **Form1** và **Form2**. Đối với **Form1** thì ta cần một số điều khiển như sau:

- 2 label để hiển thị ngày và hiển thị giờ.
- 1 ProgressBar để hiển thị thời gian thực hiện công việc.
- 2 Timer, **Timer1** để tạo sự kiện hiển thị ngày giờ hiện tại, **Timer2** để tạo sự kiện hiển thị % thực hiện công việc.

Sau khi tạo xong giao diện cho 2 Form, bây giờ đến lúc ta đi xử lý sự kiện cho điều khiển. Đối với Timer thì ta xử lý trên sự kiện Tick, ta vào Properties của Timer Click double vào sự kiện Tick để viết.

Trong **Timer1** ta sẽ tạo sự kiện hiển thị ngày tháng hiện tại. Ta tạo mới một **DateTime**, sau đó sử dụng **String.Format()** để format kiểu hiển thị.

```
1 private void timer1_Tick(object sender, EventArgs e)
2     {
3         DateTime dt = DateTime.Now.Add(new TimeSpan());
4         lbl_gio.Text = String.Format("{0:hh:mm:ss tt}", dt);
5         lbl_ngaythang.Text = String.Format("{0:dd/MM/yyyy}", dt);
6     }
```

Trong **Timer2** ta sẽ tạo sự kiện hiển thị % thực hiện công việc và khi đạt đến 100% thì load **Form2**. Ta sử dụng thuộc tính **Maximum** để thực hiện điều này.

```
1 private void timer2_Tick(object sender, EventArgs e)
2     {
```



```

3      Form2 frm = new Form2();
4      progressBar1.Increment(1);
5      lbl_complete.Text = "Connecting to from " + progressBar1.Value.ToString()
6 + "%";
7      if (progressBar1.Value == progressBar1.Maximum)
8      {
9          timer2.Enabled = false;
10         frm.ShowDialog();
11     }
12 }

```

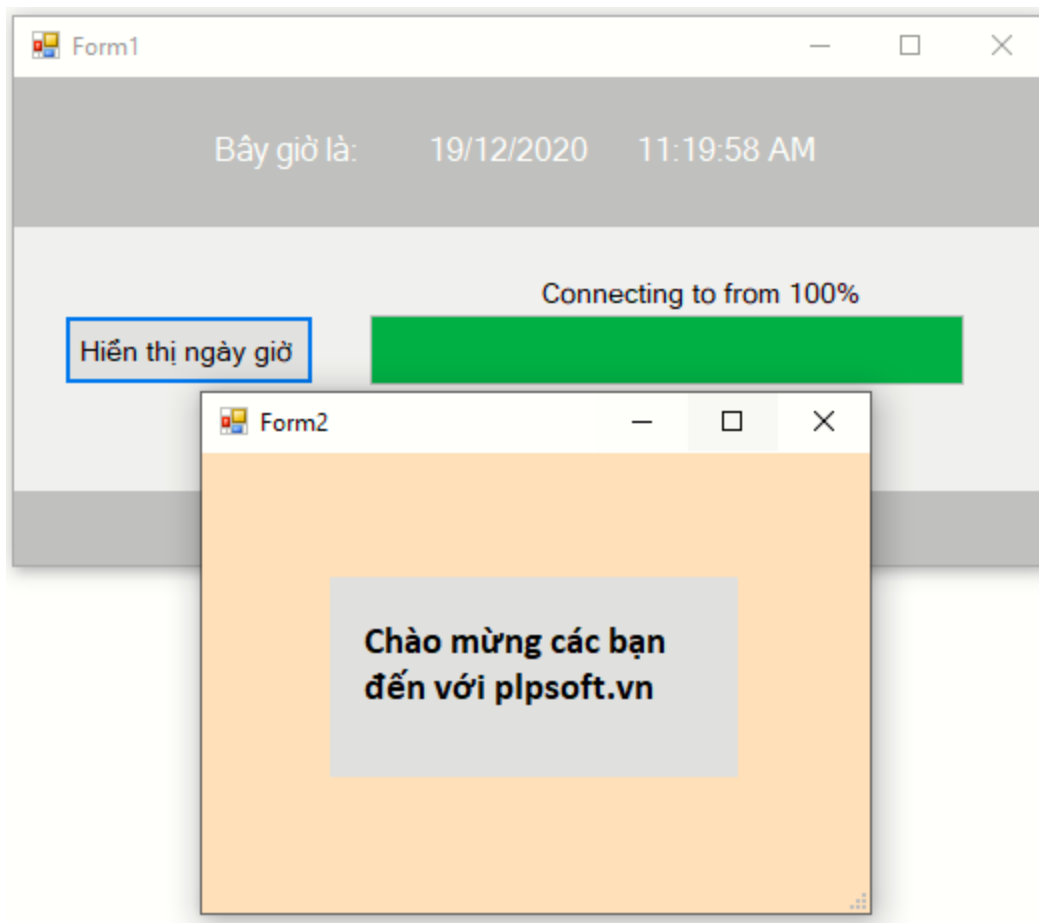
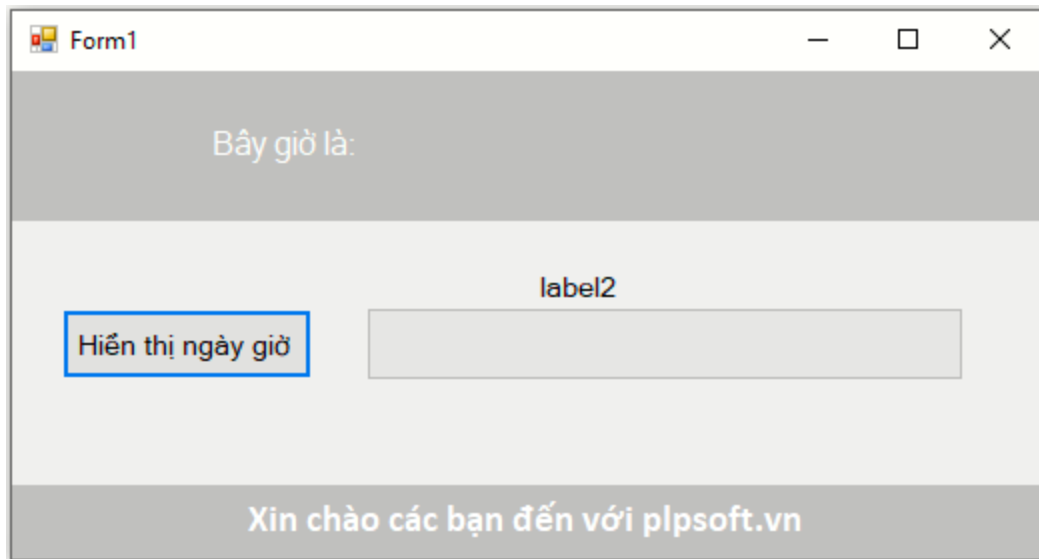
Sau khi tạo sự kiện cho ProgressBar và Timer, bây giờ ta sẽ viết sự kiện cho nút Button, khi Click vào thì sẽ bật Timer1 và Timer 2. Trong trường hợp ProgressBar đang ở trạng thái Enabled thì ta cho nó bằng False và ngược lại.

```

1 private void button1_Click(object sender, EventArgs e)
2     {
3         if(progressBar1.Enabled == true)
4         {
5             progressBar1.Enabled = false;
6             timer2.Start();
7             timer1.Start();
8         }
9         else
10        {
11            progressBar1.Enabled = true;
12            timer2.Stop();
13            timer1.Stop();
14        }
15    }

```

Kết quả: Trước khi Click vào Button "Hiển thị ngày giờ" và sau khi Click.



4. Kết luận

Như vậy là chúng ta đã tìm hiểu xong hai điều khiển đặc biệt trong C# winform. Đây là hai điều khiển được sử dụng rất nhiều, vì trong ứng dụng luôn cần đến thời gian để có thể thực hiện một cách chính xác

Bài 7. Điều khiển ListView trong lập trình C# winforms

Trong hướng dẫn này mình sẽ giới thiệu các bạn một trong các điều khiển đặc biệt tiếp theo đó chính là ListView. Đây là một điều khiển rất quan trọng vì nó có thể hiển thị các đối tượng một cách đa dạng.

Chúng ta sẽ cùng nhau tìm hiểu về công dụng của nó cũng như các thuộc tính và phương thức, sự kiện của ListView. Sau đó mình sẽ thực hiện viết một chương trình áp dụng ListView, các bạn cùng theo dõi nhé.

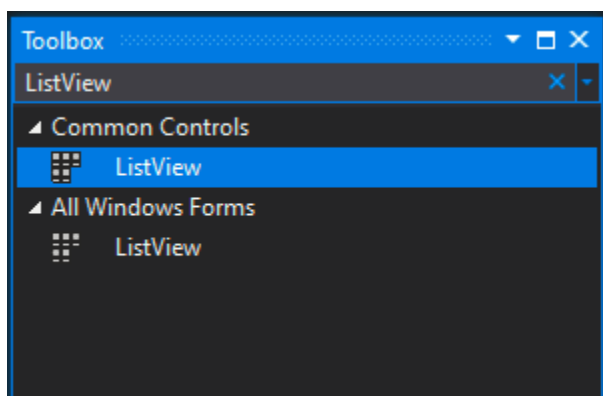
Table of Content

- 1. ListView
- 2. Ví dụ sử dụng điều khiển ListView
- 3. Kết luận

1. ListView

ListView là điều khiển cho phép hiển thị danh sách các đối tượng. Mỗi đối tượng hiển thị trong ListView được gọi là Item.

Item là đối tượng được tạo từ lớp **ListViewItem**. Mỗi Item có thuộc tính Text là chuỗi ký tự hiển thị ở cột đầu tiên trong ListView, mỗi Item có các SubItem hiển thị ở các cột tiếp theo trong ListView.



Một số thuộc tính thường dùng của ListView:

| Thuộc tính | Mô tả |
|-----------------------|---|
| View | Thuộc tính View quy định cách hiển thị các Item trong ListView. Thuộc tính View có 5 giá trị: Detail, LargeIcons, List, SmallIcons, Titles |
| MultiSelect | True/ False: Cho phép hoặc không cho phép chọn một lúc nhiều Item trong ListView |
| FullRowSelect | FullRowSelect khi chọn dòng dữ liệu highlighted cả dòng hay chỉ ô được chọn |
| GridLines | GridLines nếu thiết lập True sẽ hiển thị các dòng và cột dạng lưới, thiết lập False không hiển thị dạng lưới |
| SelectedItems | Trả về tập các Items được chọn trong ListView |
| LargeImageIcon | Gán đối tượng ImageList cho ListView |
| SmallImageIcon | Gán đối tượng ImageList cho ListView |
| FocusedItem.Index | Trả về chỉ số dòng được chọn trong ListView |
| SelectedIndices.Count | Trả về số lượng Item được chọn trong ListView |
| SelectedIndices | Trả về danh sách chỉ mục các Item được chọn |
| Items | <p>Trả về các Item chứa trong ListView. Một số phương thức và thuộc tính thường dùng của ListView.</p> <ul style="list-style-type: none"> • Count: Đếm số lượng Item trong L • Insert(<i>i</i>, <Item mới>): Chèn thêm Item mới vào vị trí <i>i</i> trong ListView • Add(<Item mới>): Thêm Item vào cuối ListView • Remove(<Item cần xóa>): Xóa Item khỏi ListView • RemoveAt(<i>i</i>): Xóa Item có chỉ số <i>i</i> khỏi ListView • Contains(<Item cần tìm>): Trả về True nếu tìm thấy trong ListView, trả về False nếu không có trong ListView • IndexOf(<Item cần tìm>): Nếu có trong ListView thì trả về chỉ số của item tìm thấy trong ListView, nếu không tìm thấy sẽ trả về -1 |

Một số phương thức thường dùng của ListView:

| Phương thức | Mô tả |
|----------------------------------|---|
| Clear() | Xóa tất cả các Item và Column trong ListView |
| Sort() | Sắp xếp các Item trong ListView |
| GetItemAt(<i>x</i> , <i>y</i>) | Lấy Item tại vị trí tọa độ <i>x</i> và <i>y</i> (<i>x</i> và <i>y</i> có thể lấy được thông qua sự kiện Click chuột) |

Một số sự kiện thường dùng của ListView:

| Sự kiện | Mô tả |
|----------------------|--|
| SelectedIndexChanged | Sự kiện phát sinh khi có sự thay đổi về chỉ mục được chọn của Item trên ListView |
| ItemSelectionChanged | Sự kiện phát sinh khi có sự thay đổi lựa chọn một Item trên ListView |
| ItemCheck | Xảy ra khi trạng thái chọn của Item thay đổi |
| ColumnClick | Sự kiện phát sinh khi có sự thay đổi lựa chọn một Item trên ListView |
| MouseClicked | Sự kiện phát sinh khi nhấp chuột chọn một Item trong ListView |

2. Ví dụ sử dụng điều khiển ListView

Trong ví dụ này mình sẽ thực hiện viết một chương trình áp dụng ListView và các thuộc tính cũng như các sự kiện của nó. Cụ thể sẽ tạo giao diện cho Form giống dưới đây, sau đó viết các sự kiện theo yêu cầu.

The screenshot shows a Windows application window titled "Form1". Inside the window, there is a form titled "Thêm Phòng Ban Mới". The form contains two text input fields: "Tên phòng" and "Số lượng nhân sự". Below these fields is a ListView control displaying a table with two columns: "Phòng" and "Số lượng". The table has several empty rows for data entry. At the bottom of the form, there are four buttons: "Thêm", "Xóa", "Xóa hết", and "Thoát".

Yêu cầu:

- Button "**Thêm**": Khi người dùng nhập xong tên phòng ban và số lượng nhân viên, sau đó nhấn nút Button "**Thêm**" thì trong ListView sẽ chèn một dòng vào cuối với tên phòng ban và số lượng vừa nhập.
- Button "**Xóa**": Khi người dùng chọn vào tên phòng ban cần xóa và nhấn nút xóa thì sẽ xóa phòng ban vừa chọn trong ListView.
- Button "**Xóa hết**": Khi người dùng chọn vào Button sẽ xóa tất cả các Item trong ListView.
- Button "**Thoát**": Thông báo "**Bạn có muốn thoát?**" xác nhận người dùng muốn đóng Form hiện hành, nếu chọn Yes thì kết thúc chương trình thực thi, nếu chọn No thì hủy bỏ lệnh kết thúc.

Bây giờ chúng ta sẽ bắt đầu tạo giao diện cho Form, cụ thể sẽ cần các điều khiển sau:

- **3 Label** để hiển thị nội dung tương ứng.
- **2 TextBox** để nhập nội dung cần thêm.
- **1 ListBox** để hiển thị thông tin.
- **4 Button** với 4 sự kiện khác nhau.

Sau khi tạo xong giao diện cho Form, ta bắt đầu viết sự kiện cho các Button.

Bước 1: Xử lý sự kiện cho nút Button "**Thêm**", khi Click vào Button thì nội dung trong ô TextBox sẽ được đưa vào trong ListBox cùng với các điều kiện ràng buộc khác.

Ta sẽ sử dụng `string.IsNullOrEmpty()` để xét sự kiện trong trường hợp một trong hai ô TextBox không có dữ liệu thì sẽ thông báo cho người dùng biết.

Tiếp đến ta tạo mới một ListViewItem, sau đó sử dụng thuộc tính `Items.Add()` để thêm nội dung từ ô TextBox và sử dụng `SubItems.Add()` để thêm vào cột tương ứng.

Sau khi thêm ta cần xóa nội dung trong ô TextBox đã nhập bằng cách sử dụng thuộc tính `Clear()`.

```
private void btnThem_Click(object sender, EventArgs e)
{
    if ((string.IsNullOrEmpty(txtTenphong.Text)) ||
        (string.IsNullOrEmpty(txtSoluong.Text)))
    {
        MessageBox.Show("Vui lòng điền đủ thông tin");
    }
}
```

```

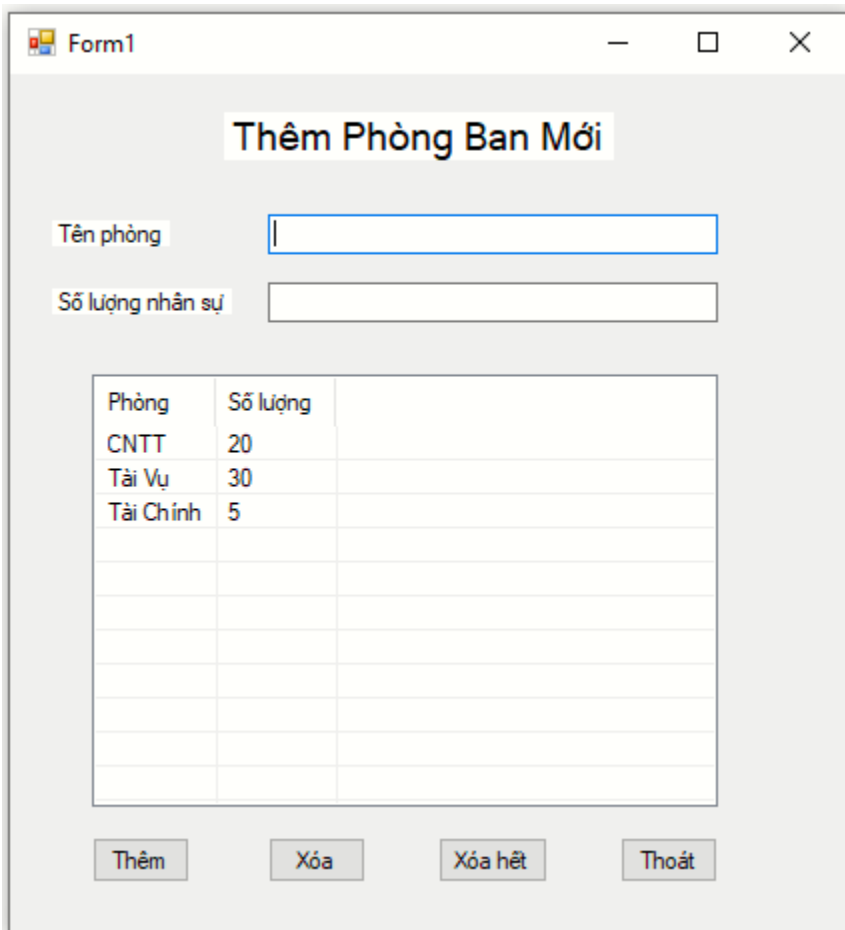
else
{
    ListViewItem item = new ListViewItem();
    item.Text = txtTenphong.Text;
    listView1.Items.Add(item);

    ListViewItem.ListViewSubItem subitem = new
ListViewItem.ListViewSubItem(item, (txtSoluong.Text));
    item.SubItems.Add(subitem);

    txtTenphong.Clear();
    txtSoluong.Clear();
    txtTenphong.Focus();
}
}

```

Kết quả: Khi nhấn vào Button "**Thêm**".



Thêm Phòng Ban Mới

Tên phòng

Số lượng nhân sự

| Phòng | Số lượng | |
|-----------|----------|--|
| CNTT | 20 | |
| Tài Vụ | 30 | |
| Tài Chính | 5 | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Thêm Xóa Xóa hết Thoát

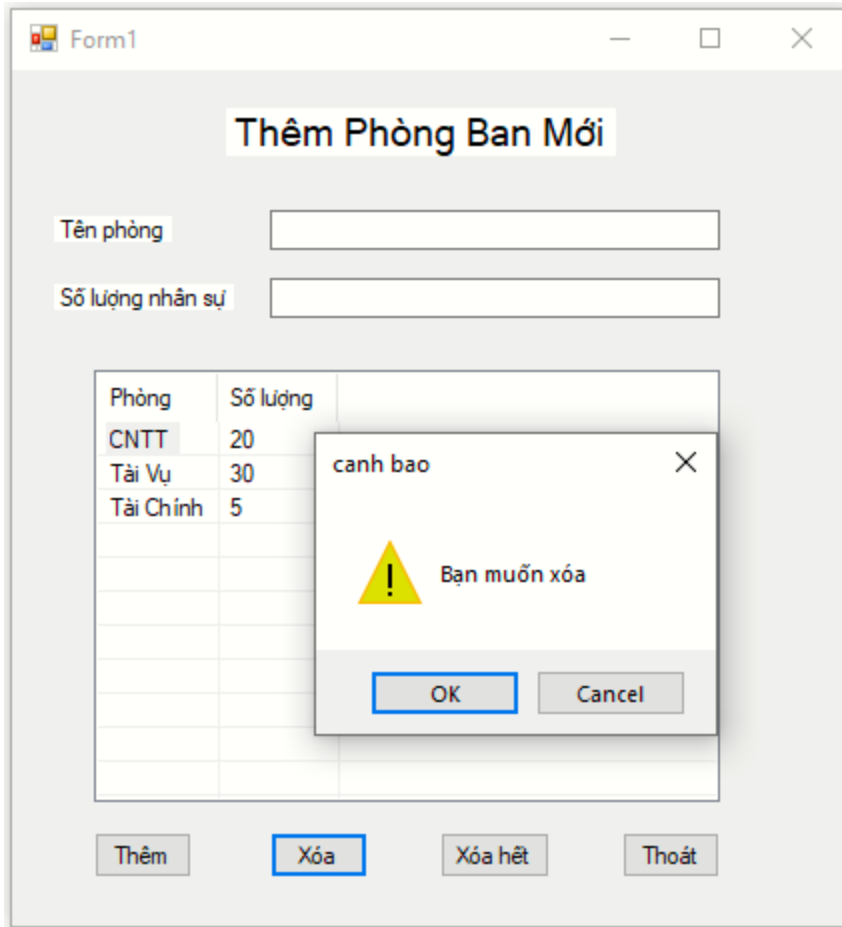
Bước 2: Xử lý sự kiện cho Button "**Xóa**", khi chọn một Item và Click vào Button thì Item đó sẽ được xóa khỏi ListView. Trước khi xóa sẽ hiển thị hộp thoại hỏi người dùng có muốn xóa hay không?

Ta sử dụng **SelectedItems.Count** để xét điều kiện nếu tồn tại Item trong ListView thì ta mới thực hiện xóa, ngược lại nếu ListView rỗng thì thông báo cho người dùng biết.

Tiếp đến ta sử dụng thuộc tính **Items.Remove()** để xóa phần tử khỏi ListView.

```
private void btnXoa_Click(object sender, EventArgs e)
{
    if (listView1.SelectedItems.Count > 0)
    {
        DialogResult dl = MessageBox.Show("Bạn muốn xóa", "canh bao",
        MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
        if (dl == DialogResult.OK)
            listView1.Items.Remove(listView1.SelectedItems[0]);
    }
    else MessageBox.Show("Xóa lỗi");
}
```

Kết quả: Khi nhấn vào Button "**Xóa**".

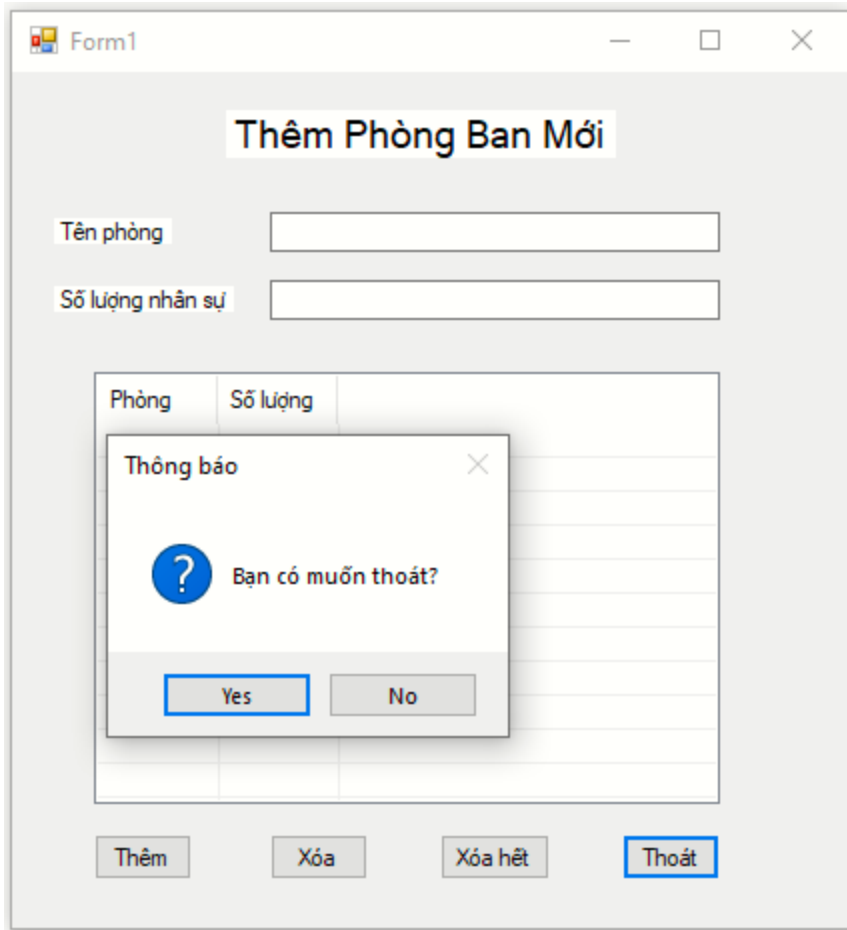


Bước 3: Xử lý sự kiện cho Button "**Xóa hết**", đối với Button này thì khá đơn giản vì trong ListView có thuộc tính thực hiện việc này khá dễ dàng đó chính là thuộc tính **Clear()**.

```
private void btnXoahet_Click(object sender, EventArgs e)
{
    listView1.Items.Clear();
}
```

Bước 4: Xử lý sự kiện cho Button "**Thoát**", tương tự như các bài trước mình đã thực hiện, ta sẽ sử dụng MessageBox để hỏi người dùng có muốn thoát hay không, nếu Yes thì sử dụng **Application.Exit()** để thoát, ngược lại chọn No thì hủy bỏ lệnh.

Kết quả: khi nhấn Button "**Thoát**".



Full Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace cau2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```

    }

    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private void btnThem_Click(object sender, EventArgs e)
    {
        if ((string.IsNullOrEmpty(txtTenphong.Text)) ||
            (string.IsNullOrEmpty(txtSoluong.Text)))
        {
            MessageBox.Show("Vui lòng điền đủ thông tin");
        }
        else
        {
            ListViewItem item = new ListViewItem();
            item.Text = txtTenphong.Text;
            listView1.Items.Add(item);

            ListViewItem.ListViewSubItem subitem = new
            ListViewItem.ListViewSubItem(item, (txtSoluong.Text));
            item.SubItems.Add(subitem);

            txtTenphong.Clear();
            txtSoluong.Clear();
            txtTenphong.Focus();
        }
    }

    private void btnXoa_Click(object sender, EventArgs e)
    {
        if (listView1.SelectedItems.Count > 0)
        {
            DialogResult dl = MessageBox.Show("Bạn muốn xóa", "canh bao",
            MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
            if (dl == DialogResult.OK)
                listView1.Items.Remove(listView1.SelectedItems[0]);
        }
        else MessageBox.Show("Xóa lỗi");
    }

    private void btnXoahet_Click(object sender, EventArgs e)

```

```

    {
        listView1.Items.Clear();
    }

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

```

3. Kết luận

Như vậy là chúng ta đã cùng nhau tìm hiểu về ListView là gì? công dụng cũng như các thuộc tính và phương thức, sự kiện của nó. Đây là một điều khiển được sử dụng rất nhiều trong việc quản lý dữ liệu, vì vậy các bạn hãy luyện tập thật nhiều để có thể thành thạo nó nhé. Ở bài tiếp theo mình sẽ giới thiệu các bạn một điều khiển cũng được dùng để quản lý dữ liệu đó chính là TreeView

Bài 8. Cách dùng TreeView trong C# winforms

Trong bài này chúng ta sẽ tìm hiểu cách sử dụng TreeView trong C# Winforms. Đây là một điều khiển khá quen thuộc, nó được sử dụng rất nhiều trong việc quản lý dữ liệu.

Chúng ta sẽ cùng nhau tìm hiểu về công dụng cũng như các thuộc tính và phương thức, sự kiện của TreeView. Sau đó mình sẽ viết một chương trình áp dụng TreeView để xem sự khác nhau giữa TreeView và ListView nhé.

Table of Content

- 1. Giới thiệu TreeView trong C#
 - Thuộc tính, phương thức và sự kiện của TreeView
 - Thuộc tính và phương thức của TreeNode
- 2. Sử dụng TreeView trong C#
- Kết luận

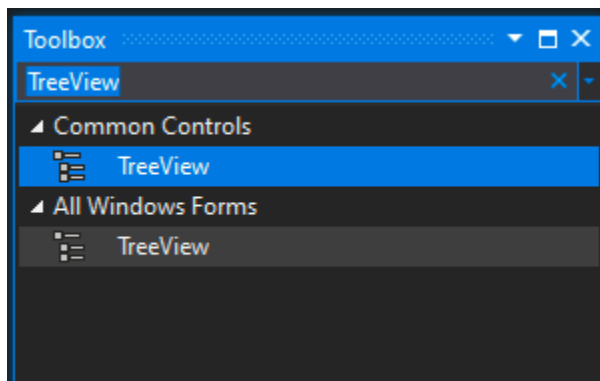
1. Giới thiệu TreeView trong C#

TreeView là điều khiển dùng để hiển thị danh sách các đối tượng dưới dạng phân cấp.

Đối tượng trong **TreeView** thường được gọi là Node và cấu trúc phân cấp của TreeView được biểu diễn bởi lớp **TreeNode**. Mỗi một Node trong TreeView có thể chứa các Node khác.

Node chứa một Node khác gọi là Node cha (*RootNode*) và Node được chứa gọi là Node con (*ChildNode*).

Việc sử dụng điều khiển TreeView để hiển thị rất hữu ích, vì trình bày theo dạng phân cấp giúp việc hiển thị được rõ ràng và có hệ thống hơn.



Thuộc tính, phương thức và sự kiện của TreeView

Một số thuộc tính thường dùng của TreeView:

| Thuộc tính | Mô tả |
|---------------------------|---|
| Node | Trả về một đối tượng thuộc lớp <i>TreeNode</i> |
| SelectedNode | Trả về Node đang được chọn trong <i>TreeView</i> |
| ShowPlusMinus | Hiển thị dấu + và - trước mỗi <i>TreeNode</i> |
| ShowRootLines | Hiển thị đường thẳng nối giữa các <i>RootNode</i> trong một <i>TreeView</i> |
| ImageList | Hiển thị hình trước mỗi Node trong <i>TreeView</i> . *Lưu ý: phải sử dụng thêm điều khiển <i>ImageList</i> và gán tên đối tượng của điều khiển <i>ImageList</i> cho thuộc tính <i>ImageList</i> của <i>TreeView</i> |
| ImageIndex | Giá trị của thuộc tính <i>ImageIndex</i> là chỉ số của hình trong điều khiển <i>ImageList</i> . Khi gán chỉ số cho thuộc tính <i>ImageIndex</i> thì hình hiển thị trước mỗi Node sẽ là hình có chỉ số tương ứng. *Lưu ý: Phải sử dụng thuộc tính <i>ImageList</i> trước |
| SelectedImageIndex | Giá trị của thuộc tính <i>SelectedImageIndex</i> là chỉ số của hình trong điều khiển <i>ImageList</i> . Khi người dùng chọn Node nào thì Node đó sẽ có hình tương ứng như thuộc tính <i>SelectedImageIndex</i> chỉ định |

Một số phương thức thường dùng của TreeView:

| Phương thức | Mô tả |
|------------------------|--|
| GetNodeCount() | Đếm số Node trong một <i>TreeView</i> |
| ExpandAll() | Hiển thị tất cả các Node trên <i>TreeView</i> |
| CollapseAll() | Thu gọn tất cả các Node trên <i>TreeView</i> |
| GetNodeAt(x, y) | Lấy một Node tại một vị trí có tọa độ (x, y) trên màn hình. *Lưu ý: Thường sử dụng sự kiện <i>MouseDown</i> hoặc <i>NodeMouseClick</i> |

Một số sự kiện thường dùng của TreeView:

| Sự kiện | Mô tả |
|-----------------------|---|
| AfterCollapse | Phát sinh khi thu gọn một <i>TreeNode</i> |
| AfterExpand | Phát sinh khi hiển thị các Node trong <i>TreeNode</i> |
| AfterSelect | Phát sinh khi chọn một <i>TreeNode</i> |
| NodeMouseClick | Phát sinh khi chọn một Node |

Thuộc tính và phương thức của TreeNode

Một số thuộc tính thường dùng của TreeNode:

| Thuộc tính | Mô tả |
|--------------|---------------------|
| Nodes | Trả về tập các Node |

| Thuộc tính | Mô tả |
|------------------|--|
| Text | Đọc/gán chuỗi ký tự người dùng sẽ nhìn thấy ở mỗi Node |
| FirstNode | Trả về Node đầu tiên |
| LastNode | Trả về Node cuối cùng |
| NextNode | Chuyển đến Node tiếp theo |
| PrevNode | Lùi lại Node trước đó |
| Parent | Trả về Node cha của Node hiện tại |
| Index | Trả về chỉ số của Node |

Một số phương thức thường dùng của TreeNode:

| Phương thức | Mô tả |
|---------------------|--|
| Nodes.Add | Thêm một Node |
| Nodes.Remove | Xóa một Node |
| Nodes.Insert | Chèn vào một Node |
| Nodes.Clear | Xóa tất cả các Node con và Node hiện tại |

2. Sử dụng TreeView trong C#

Trong ví dụ này mình sẽ thực hiện một chương trình theo giao diện Form như dưới đây, với một số chức năng.

The screenshot shows a Windows Form titled "Form1" with a light orange background. On the left, there is a "Treeview" control with a large empty rectangular area. To the right of the treeview, there is a text label "Tiêu đề Node" followed by an empty text box. Below this, there are five buttons stacked vertically: "Thêm Node gốc", "Thêm Node con", "Xóa tất cả các Node", "Xóa Node đang chọn", and "Đếm tổng Node Treeview". At the bottom of the form, there are two buttons: a "+" button on the left and a "-" button on the right.

Tạo hệ thống cây TreeView gồm:

- Button "**Thêm Node gốc**": Thêm một Node gốc.
- Button "**Thêm Node con**": Thêm một Node con.
- Button "**Xóa tất cả các Node**": Hủy tất cả các Node trong TreeView.
- Butotn "**Xóa Node được chọn**": Xóa Node được chọn khỏi TreeView.
- Button "-": Thu hẹp lại vị trí chọn - thu hẹp lại các Node là con của Node.
- Button "+": Mở rộng tại vị trí chọn - mở rộng các Node là con của Node.
- Button "**Đếm tổng Node TreeView**": Đếm tổng Node có trên TreeView.

Việc đầu tiên chúng ta sẽ thiết kế giao diện tương tự như Form mẫu, bao gồm:

- 1 TextBox để nhập tiêu đề cho Node.
- 7 Button với các chức năng tương ứng.
- 1 TreeView để hiển thị danh sách các Node.

Sau khi tạo giao diện cho Form, ta bắt đầu thực hiện lần lượt các chức năng trên các nút Button.

Bước 1: xử lý sự kiện trên Button "**Thêm Node gốc**", với chức năng khi người dùng Click vào nút thì nội dung trong ô TextBox sẽ được thêm vào Node gốc trong TreeView.

Ta sử dụng `string.IsNullOrEmpty()` để xét điều kiện nếu như trong ô TextBox rỗng mà người dùng nhấn vào Button thì thông báo cho người dùng biết.

Tiếp đến ta sử dụng `string.Equals()` để so sánh nội dung trong ô TextBox với các Node trong TreeView, nếu bằng nhau thì thông báo cho người dùng biết đã tồn tại Node trong TreeView.

Sau khi xét xong điều kiện, bây giờ ta chỉ cần sử dụng `Node.Add()` để thêm nội dung từ TextBox vào Node. Khi thêm xong ta sẽ xóa nội dung trong ô TextBox bằng phương thức `Clear()`.

```

1 private void btnThemnodegoc_Click(object sender, EventArgs e)
2 {
3     bool t = false;
4     if (!string.IsNullOrEmpty(txtNode.Text))
5     {
6         TreeNode Node = new TreeNode();
7         Node.Text = txtNode.Text;

```



```

8      foreach(TreeNode nodex in treeView1.Nodes)
9      {
10         if(string.Equals(Node.Text,nodex.Text))
11         {
12             MessageBox.Show("Node đã tồn tại");
13             t = true;
14         }
15     }
16     if (t == false) treeView1.Nodes.Add(Node);
17     txtNode.Clear();
18     txtNode.Focus();
19 }
20 else
21     MessageBox.Show("Node không được để trống");
22 }

```

Kết quả: Sau khi thêm các Node gốc và các Node con.



Bước 2: Xử lý sự kiện trên Button "**Thêm Node con**", với chức năng khi người dùng nhấp chuột vào Button thì nội dung trong ô TextBox sẽ được thêm vào Node con của Node được chọn trong TreeView.

Tương tự như Button "**Thêm Node gốc**", ta cũng sử dụng `string.IsNullOrEmpty()` để thêm điều kiện cho nút. Sau đó sử dụng `SelectedNode.Nodes.Add()` để thêm Node con vào Node được chọn.

```
1 private void btnThemnodecon_Click(object sender, EventArgs e)
2     {
3         if (!string.IsNullOrEmpty(txtNode.Text))
4         {
5             if (treeView1.SelectedNode != null)
6             {
7                 TreeNode Subnode = new TreeNode();
8                 Subnode.Text = txtNode.Text;
9                 treeView1.SelectedNode.Nodes.Add(Subnode);
10                txtNode.Clear();
11                txtNode.Focus();
12            }
13            else
14                MessageBox.Show("Bạn chưa chọn vị trí tạo Node con");
15        }
16        else
17            MessageBox.Show("Node không được để trống");
18    }
```

Bước 3: Xử lý sự kiện trên Button "**Xóa tất cả các Node**", với chức năng khi người dùng nhấn vào thì tất cả các Node sẽ được xóa.

Đơn giản ta chỉ cần sử dụng phương thức `Clear()` có trong Nodes để xóa tất cả các Node.

```
1 private void btnXoaallNode_Click(object sender, EventArgs e)
2     {
3         treeView1.Nodes.Clear();
4     }
```

Bước 4: Xử lý sự kiện trên Button "**Xóa Node được chọn**", với chức năng xóa Node mà người dùng chọn khỏi TreeView.

Ta sẽ sử dụng **SelectedNode.Remove()** để xóa Node đang được chọn khỏi TreeView.

```
1private void btnXoaNodechon_Click(object sender, EventArgs e)
2 {
3     if(treeView1.SelectedNode !=null)
4         treeView1.SelectedNode.Remove();
5 }
```

Bước 5: Xử lý sự kiện trên hai Button "-" và "+", với chức năng thu hẹp và mở rộng các Node trong TreeView.

Trong TreeView có hai sự kiện hỗ trợ cho việc này đó chính là **ExpandAll()** và **CollapseAll()**.

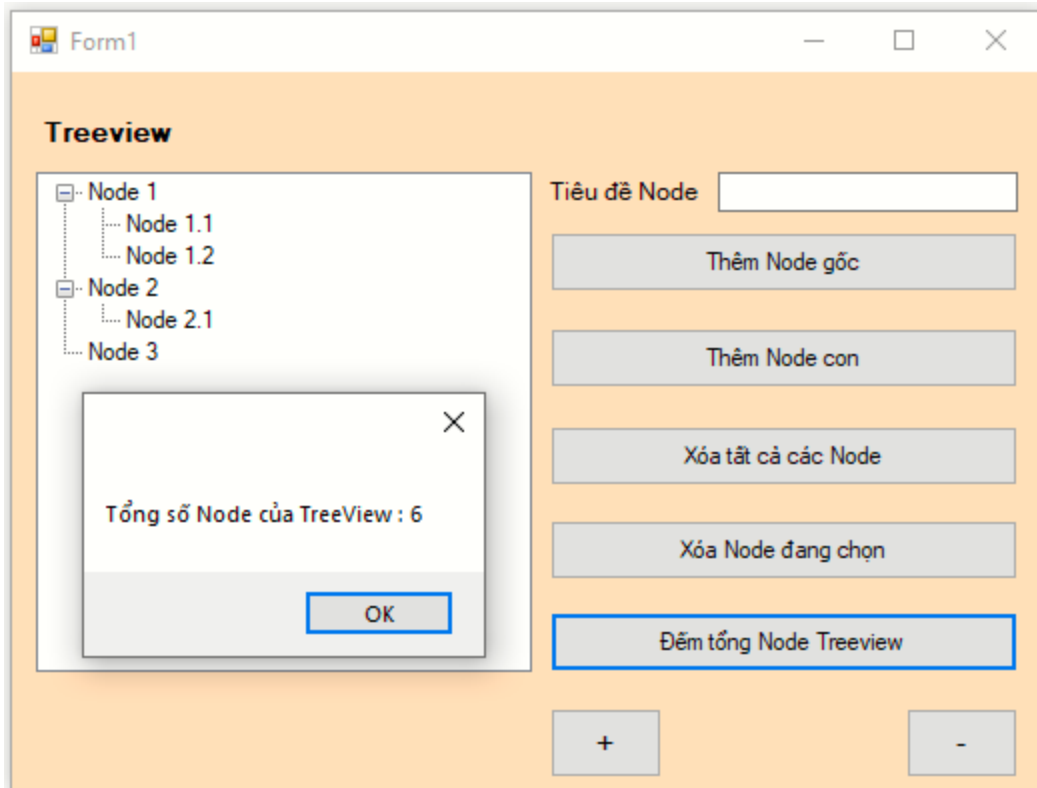
```
1private void btnShow_Click(object sender, EventArgs e)
2 {
3     treeView1.ExpandAll();
4 }
5
6private void btnAn_Click(object sender, EventArgs e)
7 {
8     treeView1.CollapseAll();
9 }
```

Bước 6: Xử lý sự kiện trên Button "**Đếm tổng Node TreeView**", với chức năng đếm tất cả các Node có trong TreeView.

Ta sẽ sử dụng **GetNodeCount()** để đếm tổng số Node trong **TreeView()**.

```
1private void btnTong_Click(object sender, EventArgs e)
2 {
3     int n = treeView1.GetNodeCount(true);
4     MessageBox.Show("Tổng số Node của TreeView : "+n);
5 }
```

Kết quả: khi nhấn vào Button "**Đến tổng Node TreeView**".



Full Code:

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace cau4
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {

```

```

17     InitializeComponent();
18 }
19
20 private void btnThemnodegoc_Click(object sender, EventArgs e)
21 {
22     bool t = false;
23     if (!string.IsNullOrEmpty(txtNode.Text))
24     {
25         TreeNode Node = new TreeNode();
26         Node.Text = txtNode.Text;
27         foreach(TreeNode nodex in treeView1.Nodes)
28         {
29             if(string.Equals(Node.Text,nodex.Text))
30             {
31                 MessageBox.Show("Node đã tồn tại");
32                 t = true;
33             }
34         }
35         if (t == false) treeView1.Nodes.Add(Node);
36         txtNode.Clear();
37         txtNode.Focus();
38     }
39     else
40         MessageBox.Show("Node không được để trống");
41
42 }
43
44 private void btnThemnodecon_Click(object sender, EventArgs e)
45 {
46     if (!string.IsNullOrEmpty(txtNode.Text))
47     {
48         if (treeView1.SelectedNode != null)
49         {
50             TreeNode Subnode = new TreeNode();

```

```

51         Subnode.Text = txtNode.Text;
52         treeView1.SelectedNode.Nodes.Add(Subnode);
53
54         txtNode.Clear();
55         txtNode.Focus();
56     }
57     else
58         MessageBox.Show("Bạn chưa chọn vị trí tạo Node con");
59     }
60     else
61         MessageBox.Show("Node không được để trống");
62 }
63
64 private void btnShow_Click(object sender, EventArgs e)
65 {
66     treeView1.ExpandAll();
67 }
68
69 private void btnAn_Click(object sender, EventArgs e)
70 {
71     treeView1.CollapseAll();
72 }
73
74 private void btnXoaallNode_Click(object sender, EventArgs e)
75 {
76     treeView1.Nodes.Clear();
77 }
78
79 private void btnXoaNodechon_Click(object sender, EventArgs e)
80 {
81     if(treeView1.SelectedNode !=null)
82         treeView1.SelectedNode.Remove();
83 }
84

```

```

85     private void btnTong_Click(object sender, EventArgs e)
86     {
87         int n = treeView1.GetNodeCount(true);
88         MessageBox.Show("Tổng số Node của TreeView : "+n);
89     }
90
91     private void Form1_Load(object sender, EventArgs e)
92     {
93
94     }
95 }
96}

```

Kết luận

Như vậy là chúng ta đã cùng nhau tìm hiểu về điều khiển TreeView trong C#, cũng như các thuộc tính, sự kiện và phương thức của nó. Đây là một điều khiển được sử dụng rất nhiều, vì vậy các bạn hãy luyện tập thật nhiều để thành thạo nó nhé. Ở bài tiếp theo mình sẽ hướng dẫn các bạn các điều khiển khác trong winforms.

Bài 9. Cách dùng DateTimePicker - MonthCalendar trong C# winforms

Trong bài này chúng ta sẽ tìm hiểu hai đối tượng **DateTimePicker** và **MonthCalendar** trong C#, đây là hai điều khiển được sử dụng để hiển thị ngày tháng dưới hai hình thức khác nhau trong lập trình Winforms.

Chúng ta sẽ cùng nhau tìm hiểu về công dụng cũng như các thuộc tính và phương thức, sự kiện của DateTimePicker và MonthCalendar nhé. Sau đó mình sẽ thực hiện một chương trình áp dụng hai điều khiển trên.

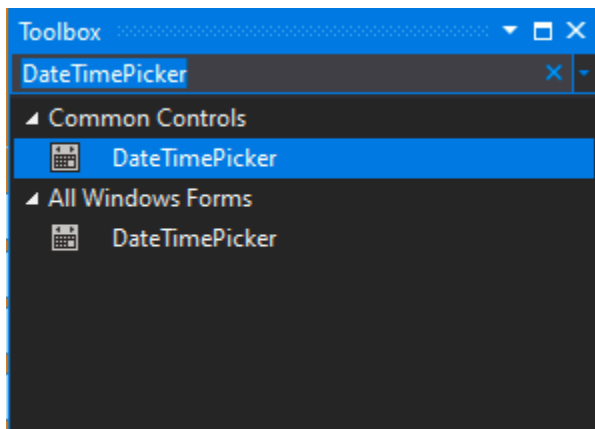
Table of Content

- 1. DateTimePicker trong C#
- 2. MonthCalendar trong C#
- 3. Ví dụ sử dụng DateTimePicker và MonthCalendar trong C#
 - Ví dụ sử dụng điều khiển DateTimePicker
 - Ví dụ sử dụng điều khiển MonthCalendar
- 4. Kết luận

1. DateTimePicker trong C#

Điều khiển **DateTimePicker** cho phép người dùng chọn ngày tháng như một lịch biểu nhưng biểu diễn ở dạng ComboBox.

Các đối tượng ngày tháng biểu diễn trong DateTimePicker thực chất là các đối tượng thuộc lớp DateTime. Điều khiển DateTimePicker được đặt trong nhóm Common Controls của cửa sổ Toolbox.



Một số thuộc tính thường dùng trong DateTimePicker:

| Thuộc tính | Mô tả |
|-------------------------|--|
| Format | Định dạng kiểu hiển thị của ngày tháng <i>*Lưu ý:</i> Thường sử dụng giá trị kiểu Short |
| Value | Trả về giá trị hiện thời của điều khiển DateTimePicker |
| Value.Date | Trả về ngày tháng năm |
| Value.Day | Trả về ngày của tháng |
| Value.Month | Trả về tháng |
| Value.Year | Trả về năm |
| Value.DateOfWeek | Trả về ngày của tuần (0 là chủ nhật, 1 là thứ hai, ..., 6 là thứ bảy) |
| CustomFormat | Cho phép lập trình tạo ra một định dạng khác nhau về ngày tháng. <i>*Lưu ý:</i> Định dạng ngày tháng năm như kiểu Việt Nam thì kiểu định dạng phải là dd/MM/yyyy. Khi đó thuộc tính Format phải thiết lập là Custom |
| MaxDate | Thiết lập ngày lớn nhất cho phép người dùng chọn trên điều khiển DateTimePicker |
| MinDate | Thiết lập ngày nhỏ nhất cho phép người dùng chọn trên điều khiển DateTimePicker |

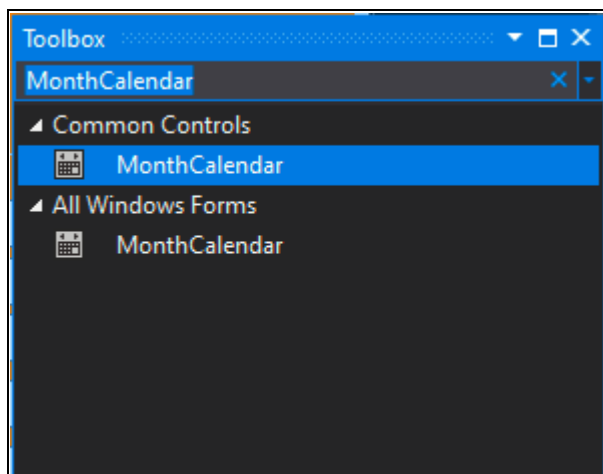
Một số sự kiện thường dùng:

| Sự kiện | Mô tả |
|---------------------|--|
| ValueChanged | Phát sinh khi người dùng chọn giá trị khác với giá trị trước đó trên điều khiển DateTimePicker |
| CloseUp | Phát sinh người dùng kết thúc việc chọn ngày trên điều khiển DateTimePicker |

2. MonthCalendar trong C#

MonthCalendar là điều khiển hiển thị lịch dưới dạng một lịch biểu cho phép người dùng chọn ngày tháng. Nhưng khác biệt là MonthCalendar cho phép người dùng có thể chọn một tập các ngày hay nói cách khác là một tập các đối tượng thuộc lớp DateTime.

Điều khiển MonthCalendar được đặt trong nhóm Common Controls.



Một số thuộc tính thường dùng:

| Thuộc tính | Mô tả |
|----------------------------|---|
| MaxDate | Thiết lập ngày lớn nhất cho phép người dùng chọn trên điều khiển MonthCalendar |
| MinDate | Thiết lập ngày nhỏ nhất cho phép người dùng chọn trên điều khiển MonthCalendar |
| SelectionRange | Trả về một dãy các ngày liên tục được chọn bởi người dùng |
| SelectionStart | Trả về ngày đầu tiên trong dãy tại thuộc tính SelectionRange |
| SelectionEnd | Trả về ngày cuối cùng trong dãy tại thuộc tính SelectionRange |
| AnnuallyBoldedDates | Chứa một mảng các ngày. Trong mỗi năm, các ngày trong mảng sẽ được bôi đen MonthCalendar |
| BoldedDates | Chứa mảng các ngày. Các ngày này sẽ được bôi đen trên điều khiển MonthCalendar tại những năm chỉ định |
| MaxSelectCount | Thiết lập số lượng ngày tối đa mà người dùng có thể chọn |
| MonthlyBoldedDates | Chứa mảng các ngày trong mỗi tháng, các ngày trong mảng sẽ được bôi đen trên MonthCalendar |

Trong MonthCalendar có một sự kiện được sử dụng rất thông dụng đó chính là **DateChanged**. Đây là một sự kiện được phát sinh khi một ngày mới hoặc một dãy các ngày mới được chọn.

3. Ví dụ sử dụng DateTimePicker và MonthCalendar trong C#

Trong ví dụ này mình sẽ thực hiện viết chương trình có sử dụng DateTimePicker và MonthCalendar để các bạn hiểu rõ hơn về công dụng cũng như cách nó hoạt động.

Ví dụ sử dụng điều khiển DateTimePicker

Trong ví dụ đầu tiên này mình sẽ thực hiện một chương trình đơn giản trong đó có sử dụng DateTimePicker, cụ thể sẽ thực hiện theo Form dưới đây với một số chức năng tương ứng.

The screenshot shows a Windows application window titled "Form1". The main heading is "ĐĂNG KÝ KHÓA HỌC TRỰC TUYẾN" in orange. On the left is an illustration of a hot air balloon with orange, blue, and white stripes. On the right are four input fields: "Họ Tên", "Số Điện Thoại", "Email", and "Thời Gian" (which shows "Friday, December 25, 2020" and a calendar icon). Below these are two buttons: "Đăng ký" and "Thoát". At the bottom is a green city skyline with various icons (Wi-Fi, mail, globe, play button, etc.) and a logo for "CNTT Learn with the expert".

Chương trình có các chức năng như sau:

- TextBox "**Họ Tên**": Chỉ cho phép nhập ký tự hoa hoặc thường, cho phép nhập khoảng trắng và cho phép xóa lùi.
- TextBox "**Số Điện Thoại**": Chỉ cho phép nhập số và được xóa lùi.
- DateTimePicker "**Thời Gian**": Chỉ cho phép chọn thời gian đăng ký học lớn hơn thời gian hiện tại.
- Button "**Đăng ký**": Hiện thị hộp thoại bao gồm các nội dung trong ô TextBox và thời gian được chọn trong DateTimePicker.
- Button "**Thoát**": Đóng chương trình.

Đầu tiên chúng ta sẽ tạo Form tương tự như trên:

- Một số Label với nội dung tương ứng.
- 3 TextBox để nhập tên, sdt, email.
- 2 Button để đăng ký và thoát.
- 1 DateTimePicker để chọn thời gian học.

Sau khi tạo được Form, ta sẽ bắt đầu xử lý sự kiện cho từng TextBox và Button.

Bước 1: Xử lý sự kiện cho TextBox "**Họ Tên**", với chức năng chỉ cho phép nhập lý tự hoa hoặc thường. cho phép nhập khoảng trắng và xóa lùi.

Ta sẽ viết trên sự kiện **KeyPress** trong TextBox, sử dụng **KeyChar** để làm điều kiện.

```
1 private void txtHoTen_KeyPress(object sender, KeyPressEventArgs e)
2 {
3     if (!((e.KeyChar >= 'a' && e.KeyChar <= 'z') || (e.KeyChar == ' ') || (e.KeyChar
4 >= 'A' && e.KeyChar <= 'Z') || e.KeyChar == (char)8))
5         e.Handled = true;
6 }
```

Bước 2: Xử lý sự kiện cho TextBox "**Số Điện Thoại**", với chức năng chỉ cho phép nhập số và được xóa lùi.

Tương tự như TextBox "**Họ Tên**" ta cũng sẽ sử dụng **KeyPress** và **KeyChar** để viết điều kiện.

```
1 private void txtSDT_KeyPress(object sender, KeyPressEventArgs e)
2 {
3     if (!((e.KeyChar >= '0' && e.KeyChar <= '9') || e.KeyChar == (char)8))
4         e.Handled = true;
5 }
```

Bước 3: Xử lý sự kiện cho DateTimePicker "**Thời Gian**", với chức năng chỉ cho phép chọn thời gian đăng ký học lớn hơn thời gian hiện tại. Ta sẽ viết sự kiện ở **FormLoad**.

```
private void Form1_Load(object sender, EventArgs e)
1 {
2     dateTimePicker1.MinDate = DateTime.Now;
3     int dayofmonth = DateTime.DaysInMonth(DateTime.Now.Year,
4 DateTime.Now.Month);
5     dateTimePicker1.MaxDate = new DateTime(DateTime.Now.Year,
6 DateTime.Now.Month, dayofmonth);
7 }
```

Bước 4: Xử lý sự kiện cho Button "**Đăng Ký**", với chức năng sẽ hiển thị nội dung trong các ô TextBox và thời gian được chọn trong DateTimePicker trên hộp thoại MessageBox.

```

private void btnDangKy_Click(object sender, EventArgs e)
{
1      if (string.IsNullOrEmpty(txtHoTen.Text) || string.IsNullOrEmpty(txtSDT.Text)
2 || string.IsNullOrEmpty(txtEmail.Text))
3      {
4          MessageBox.Show("Bạn chưa nhập đầy đủ thông tin");
5      }
6      else if (!string.IsNullOrEmpty(txtHoTen.Text) ||
7 !string.IsNullOrEmpty(txtSDT.Text) || !string.IsNullOrEmpty(txtEmail.Text) == true)
8      {
9          MessageBox.Show("Chúc mừng bạn: " + txtHoTen.Text + "\nSố Điện Thoại:
10" + txtSDT.Text + "\nEmail: " + txtEmail.Text + "\nThời gian lựa chọn: " +
11dateTimePicker1.Value.ToShortDateString());
12      }
13  }

```

Bước 5: Xử lý sự kiện cho Button "**Thoát**", với chức năng sẽ đóng chương trình khi người dùng nhấn vào nút này.

```

1private void bntThoat_Click(object sender, EventArgs e)
2 {
3     Application.Exit();
4 }

```

Kết quả:

Full Code:

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace Cau5
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19     }

```

```

20 private void btnDangKy_Click(object sender, EventArgs e)
21 {
22     if (string.IsNullOrEmpty(txtHoTen.Text) || string.IsNullOrEmpty(txtSDT.Text)
23 || string.IsNullOrEmpty(txtEmail.Text))
24     {
25         MessageBox.Show("Bạn chưa nhập đầy đủ thông tin");
26     }
27     else if (!string.IsNullOrEmpty(txtHoTen.Text) ||
28 !string.IsNullOrEmpty(txtSDT.Text) || !string.IsNullOrEmpty(txtEmail.Text) == true)
29     {
30         MessageBox.Show("Chúc mừng bạn: " + txtHoTen.Text + "\nSố Điện Thoại:
31 " + txtSDT.Text + "\nEmail: " + txtEmail.Text + "\nThời gian lựa chọn: " +
32 dateTimePicker1.Value.ToShortDateString());
33     }
34 }
35
36
37 private void txtHoTen_KeyPress(object sender, KeyPressEventArgs e)
38 {
39     if (!((e.KeyChar >= 'a' && e.KeyChar <= 'z') || (e.KeyChar == ' ') || (e.KeyChar
40 >= 'A' && e.KeyChar <= 'Z') || e.KeyChar == (char)8))
41         e.Handled = true;
42 }
43
44 private void txtSDT_KeyPress(object sender, KeyPressEventArgs e)
45 {
46     if (!(e.KeyChar >= '0' && e.KeyChar <= '9' || e.KeyChar == (char)8))
47         e.Handled = true;
48 }
49
50
51 private void Form1_Load(object sender, EventArgs e)
52 {
53     dateTimePicker1.MinDate = DateTime.Now;

```

```

54     int dayofmonth = DateTime.DaysInMonth(DateTime.Now.Year,
55 DateTime.Now.Month);
56     dateTimePicker1.MaxDate = new DateTime(DateTime.Now.Year,
57 DateTime.Now.Month, dayofmonth);
58 }

```

```

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

```

Ví dụ sử dụng điều khiển MonthCalendar

Trong ví dụ này mình sẽ thực hiện viết một chương trình trong đó có sử dụng MonthCalendar, cụ thể ta sẽ tạo giao diện như Form dưới đây với các chức năng tương ứng.

Các chức năng trong chương trình:

- Button "**Hiển thị**": Hiển thị ngày bắt đầu chọn (*SelectionStart*), ngày cuối cùng chọn (*SelectionEnd*), *MinDate*, *MaxDate*, *Today*.
- Button "**Count Day**": Đếm tổng ngày từ lúc *SelectionStart* đến *SelectionEnd*.
- Button "**Count Hour**": Đếm tổng giờ từ lúc *SelectionStart* đến *SelectionEnd*.

Việc đầu tiên ta cần tạo giao diện cho Form tương tự như Form ở trên, bao gồm:

- Các Label với nội dung tương ứng.
- 7 TextBox để hiển thị các nội dung.
- 3 Button để viết sự kiện.
- 1 MonthCalendar làm lịch biểu.

Tiếp đến ta sẽ lần lượt viết các chức năng cho chương trình.

Bước 1: Xử lý sự kiện cho Button "**Hiển thị**", với chức năng hiển thị ngày bắt đầu và ngày cuối cùng được chọn, *MinDate*, *MaxDate*, *Today*.

```
1 private void btnHienthi_Click(object sender, EventArgs e)
2 {
3     txtStart.Text = monthCalendar1.SelectionStart.ToShortDateString();
4     txtEnd.Text = monthCalendar1.SelectionEnd.ToShortDateString();
5     txtmin.Text = monthCalendar1.MinDate.ToShortDateString();
6     txtMax.Text = monthCalendar1.MaxDate.ToShortDateString();
7     txtDayDate.Text = monthCalendar1.TodayDate.ToShortDateString();
8 }
```

Bước 2: Xử lý sự kiện cho Button "**Count Day**", với chức năng đếm số ngày được chọn.

```
1 private void btnCountday_Click(object sender, EventArgs e)
2 {
3     int numdays;
4     numdays = Convert.ToInt32((monthCalendar1.SelectionEnd -
5 monthCalendar1.SelectionStart).TotalDays);
6     txtday.Text = numdays.ToString();
7 }
```

Bước 3: Xử lý sự kiện cho Button "**Count Hour**", với chức năng đếm số giờ được chọn.

```
1 private void btnCountHour_Click(object sender, EventArgs e)
2 {
3     int numhour;
```

```

4    numhour = Convert.ToInt32((monthCalendar1.SelectionEnd -
5monthCalendar1.SelectionStart).TotalHours);
6    txthour.Text = numhour.ToString();
    }

```

Kết quả:

Full Code:

```

1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10

```

```

11 namespace Cau2
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void btnHienthi_Click(object sender, EventArgs e)
21         {
22             txtStart.Text = monthCalendar1.SelectionStart.ToShortDateString();
23             txtEnd.Text = monthCalendar1.SelectionEnd.ToShortDateString();
24             txtmin.Text = monthCalendar1.MinDate.ToShortDateString();
25             txtMax.Text = monthCalendar1.MaxDate.ToShortDateString();
26             txtDayDate.Text = monthCalendar1.TodayDate.ToShortDateString();
27         }
28
29         private void btnCountday_Click(object sender, EventArgs e)
30         {
31             int numdays;
32             numdays = Convert.ToInt32((monthCalendar1.SelectionEnd -
33 monthCalendar1.SelectionStart).TotalDays);
34             txtday.Text = numdays.ToString();
35         }
36
37         private void btnCountHour_Click(object sender, EventArgs e)
38         {
39             int numhour;
40             numhour = Convert.ToInt32((monthCalendar1.SelectionEnd -
41 monthCalendar1.SelectionStart).TotalHours);
42             txthour.Text = numhour.ToString();
43         }
44

```

```
45     private void Form1_Load(object sender, EventArgs e)
46     {
47
48     }
    }
}
```

4. Kết luận

Như vậy là chúng ta đã tìm hiểu xong về DateTimePick và MonthCalendar trong lập trình C# winforms. Đây là hai điều khiển khá quan trọng và được sử dụng nhiều, vì vậy các bạn hãy luyện tập nhiều để có thể thành thạo nó nhé. Ở bài tiếp theo mình sẽ tiếp tục giới thiệu các bạn các điều khiển quan trọng khác trong C# winforms.

Bài 10. Cách dùng MenuStrip trong C# winforms

Trong bài viết này mình sẽ hướng dẫn sử dụng điều khiển MenuStrip trong C#, đây là một điều khiển rất quan trọng trong lập trình ứng dụng với winforms, vì mỗi ứng dụng đều có menu để người dùng có thể lựa chọn.

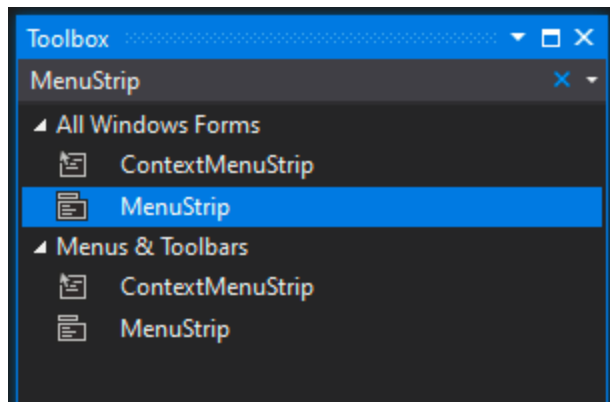
Chúng ta sẽ cùng nhau tìm hiểu về công dụng cũng như các thuộc tính và phương thức, sự kiện của MenuStrip. Sau đó mình sẽ thực hiện một vài ví dụ áp dụng MenuStrip để các bạn hiểu rõ hơn.

Table of Content

- 1. MenuStrip trong C#
- 2. Ví dụ sử dụng điều khiển MenuStrip trong C#
- 3. Kết luận

1. MenuStrip trong C#

MenuStrip là một điều khiển cho phép lập trình viên xây dựng hệ thống Menu trên Form. Menu có thể xây dựng ở dạng một cấp hoặc nhiều cấp.



MenuStrip cho phép xây dựng Menu với các điều khiển:

- ToolStripSeparator.
- ToolStripMenuItem (*Menu con*).
- ToolStripComboBox (*ComboBox*).
- ToolStripTextBox (*TextBox*).

Một số thuộc tính thường dùng của MenuStrip:

| Thuộc tính | Mô tả |
|----------------------|---|
| TextDirection | Chọn hình thức trình bày Menu <ul style="list-style-type: none"> Hình thức Horizontal Hình thức Vertical 90 Hình thức Vertical 270 |
| Items | Thêm các menu con. Kiểu menu có thể chọn một trong 4 dạng: <i>MenuItem</i> , <i>ComboBox</i> , <i>TextBox</i> , <i>Seperator</i> |
| RightToLeft | Mang giá trị Yes hoặc No. Nếu là Yes thì sẽ trình bày menu từ phải qua trái. Nếu là No thì trình bày menu từ trái qua phải |
| Checked | Mang giá trị True hoặc False. Nếu là giá trị True thì hiển thị biểu tượng CheckBox bên cạnh chuỗi Text. Nếu là False thì không hiện biểu tượng CheckBox |
| CheckOnClick | Mang giá trị True hoặc False <ul style="list-style-type: none"> Nếu là True: Biểu tượng CheckBox sẽ xuất hiện bên cạnh chuỗi Text của menu con khi người dùng nhấp chuột chọn. Nếu là False: Thao tác nhấp chuột của người dùng sẽ không bị ảnh hưởng gì. |
| CheckState | Cho biết trạng thái của CheckBox trên menu con. Có 3 trạng thái: <i>Unchecked</i> , <i>Checked</i> , <i>Indeterminate</i> . * Lưu ý: Trạng thái Indeterminate chỉ có hiệu lực khi thuộc tính Checked là true. |
| DisplayStyle | Hình thức trình bày của menu con có 4 kiểu hiển thị. <ul style="list-style-type: none"> None: Không hiển thị gì trên menu con Text: Cho phép hiển thị chuỗi mô tả Image: Cho phép hiển thị hình hoặc biểu tượng cạnh bên Text ImageAndText: Cho phép hiển thị hình (<i>biểu tượng</i>) và chuỗi mô tả |
| Image | Hình ảnh xuất hiện bên cạnh chuỗi Text |
| ImageScaling | Kiểu trình bày của hình trong thuộc tính Image. Có thể thiết lập một trong hai giá trị |

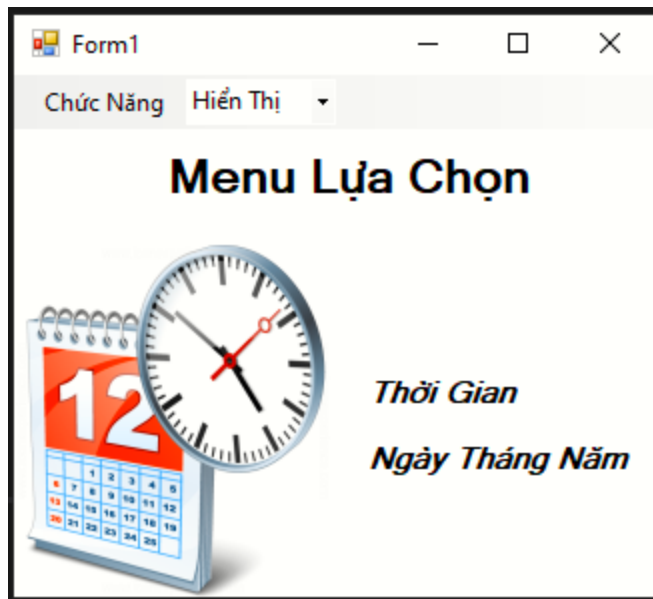
| Thuộc tính | Mô tả |
|---------------------------------|---|
| | <ul style="list-style-type: none"> • None: Hiển thị bình thường • SizeToFit: Hiển thị đúng kích cỡ của hình hoặc biểu tượng |
| ShortcutKeyDisplayString | Chuỗi trình bày ứng với phím tắt mô tả cho menu đó |
| ShortcutKeys | Tổ hợp phím tắt ứng với menu |
| ShowShortcutKeys | Mang giá trị True hoặc False. <ul style="list-style-type: none"> • Nếu là True: Cho phép hiển thị giá trị trong thuộc tính ShortcutKeyDisplayString. • Nếu là False: Giá trị trong thuộc tính ShortcutKeyDisplayString sẽ không hiển thị. |
| Text | Chuỗi ký tự hiển thị trên menu |
| ToolTipText | Chuỗi ký tự hiển thị khi rê chuột vào menu |

Các sự kiện thường dùng của MenuItem:

| Sự kiện | Mô tả |
|----------------------|--|
| CheckedChange | Phát sinh khi trạng thái (<i>CheckState</i>) của CheckBox thay đổi |
| Click | Phát sinh khi người dùng nhấp chuột vào menu |

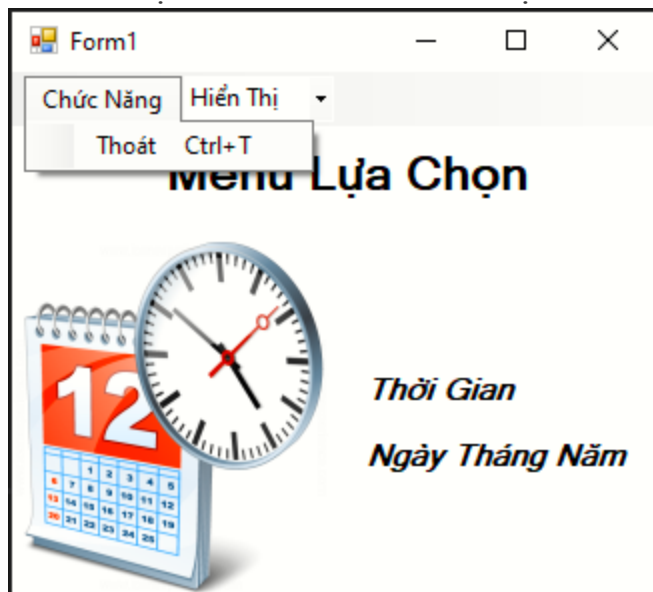
2. Ví dụ sử dụng điều khiển MenuStrip trong C#

Trong ví dụ này mình sẽ viết một chương trình trong đó có sử dụng MenuStrip và kết hợp thêm Timer để hiển thị ngày giờ hiện hành. Cụ thể mình sẽ tạo giao diện cho Form như dưới đây và thực hiện một số chức năng tương ứng.

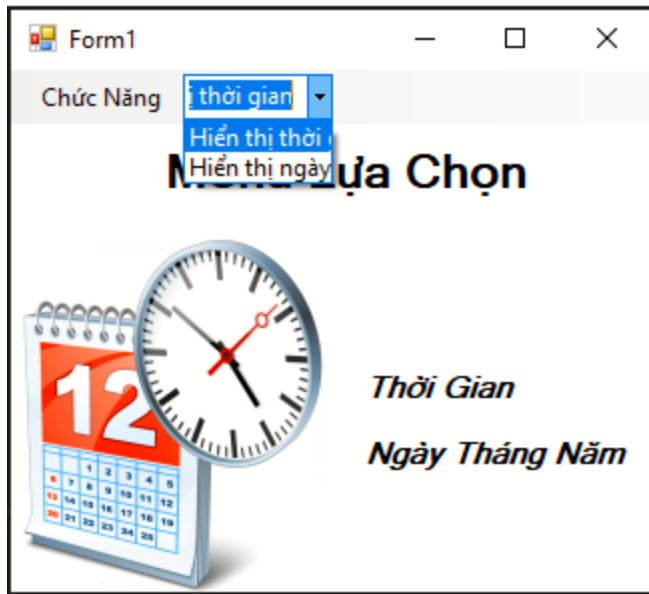


Chương trình có các chức năng:

- Menu chức năng: có chức năng mục **Thoát** dạng MenuItem. Khi người dùng nhấn chuột trái vào menu **Thoát** hoặc nhấn tổ hợp phím **Ctrl + T** sẽ thoát chương trình.

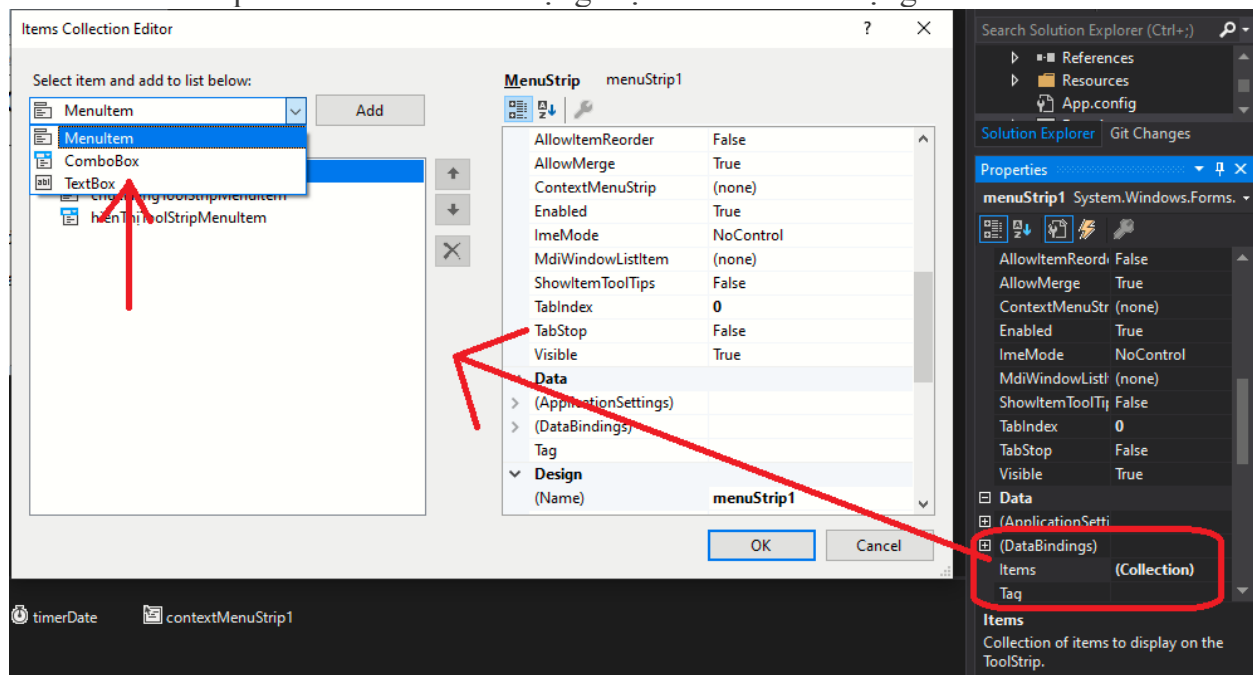


- Menu hiển thị: có dạng ComboBox chứa hai mục chọn, hiển thị thời gian hiện hành và hiển thị ngày tháng năm hiện hành.



Việc đầu tiên ta cần tạo giao diện cho Form, bao gồm:

- 2 Label để hiển thị ngày giờ hiện hành.
- 2 Timer để lấy ngày giờ hiện hành.
- 1 MenuStrip với 2 Item là "**chứcNăngToolStripMenuItem**" và "**hiểnThịToolStripMenuItem**". để thêm hai Item này ta vào thuộc tính Items trên MenuStrip rồi thêm 1 Item có dạng mục và 1 Item có dạng ComboBox.



Sau khi đã tạo giao diện cho Form và thêm một số điều khiển cần thiết, bây giờ ta sẽ đi xử lý sự kiện.

Bước 1: Xử lý sự kiện Tick cho hai Timer để lấy ngày giờ hiện hành. Chú ý rằng trong Timer thuộc tính **Enabled** mặc định ở trạng thái False, vì vậy nó sẽ không hiển thị ngay khi chạy chương trình. Khi chúng ta chọn hiển thị ở Menu thì khi đó mới Start Timer lên.

```
1 private void timerTime_Tick(object sender, EventArgs e)
2 {
3     DateTime dt = DateTime.Now.Add(new TimeSpan());
4     lblTime.Text = String.Format("{0:hh:mm:ss tt}", dt);
5 }
6 private void timerDate_Tick(object sender, EventArgs e)
7 {
8     DateTime dt = DateTime.Now.Add(new TimeSpan());
9     lblDate.Text = String.Format("{0:dd/MM/yyyy}", dt);
10 }
```

Bước 2: Xử lý sự kiện cho Menu với chức năng là khi người dùng chọn hiển thị ngày hoặc giờ thì ngày giờ hiện hành sẽ được hiển thị trên Label. Và khi người dùng chọn **Thoát** hoặc nhấn tổ hợp **Ctrl + T** thì sẽ đóng chương trình.

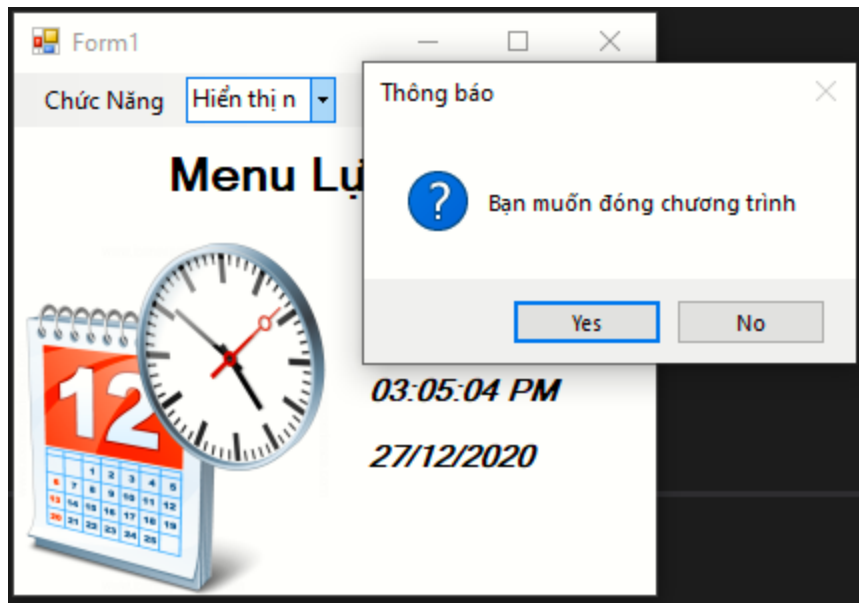
Khi tạo Menu ta đã tạo hai Item là "**chứcNăngToolStripMenuItem**" và "**hiểnThịToolStripMenuItem**" bây giờ chỉ cần viết sự kiện cho nó. Ở Item "**hiểnThịToolStripMenuItem**" sẽ viết trên sự kiện **SelectedIndexChanged**.

```
private void hiểnThịToolStripMenuItem_SelectedIndexChanged(object sender,
EventArgs e)
{
    if(hiểnThịToolStripMenuItem.SelectedItem == "Hiển thị thời gian")
    {
        timerTime.Start();
    }
    else if(hiểnThịToolStripMenuItem.SelectedItem == "Hiển thị ngày tháng")
    {
        timerDate.Start();
    }
}
```

Còn Item "**chứcNăngToolStripMenuItem**" ta sẽ viết trên sự kiện **Click**, bằng cách nhấn đúp chuột vào chức năng **Thoát**.

```
1 private void thoátToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     DialogResult dt = MessageBox.Show("Bạn muốn đóng chương trình", "Thông
4 báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
5     if(dt == DialogResult.Yes)
6     {
7         Application.Exit();
8     }
9 }
```

Kết quả:



Full Code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
```

```

9 using System.Windows.Forms;
10
11 namespace Cau1
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void timerTime_Tick(object sender, EventArgs e)
21         {
22             DateTime dt = DateTime.Now.Add(new TimeSpan());
23             lblTime.Text = String.Format("{0:hh:mm:ss tt}", dt);
24         }
25
26         private void timerDate_Tick(object sender, EventArgs e)
27         {
28             DateTime dt = DateTime.Now.Add(new TimeSpan());
29             lblDate.Text = String.Format("{0:dd/MM/yyyy}", dt);
30         }
31
32         private void hiểnThịToolStripMenuItem_SelectedIndexChanged(object sender,
33 EventArgs e)
34         {
35             if(hiểnThịToolStripMenuItem.SelectedItem == "Hiển thị thời gian")
36             {
37                 timerTime.Start();
38             }
39             else if(hiểnThịToolStripMenuItem.SelectedItem == "Hiển thị ngày tháng")
40             {
41                 timerDate.Start();
42             }
43         }
44     }
45 }

```

```

43     }
44
45     private void thoátToolStripMenuItem_Click(object sender, EventArgs e)
46     {
47         DialogResult dt = MessageBox.Show("Bạn muốn đóng chương trình", "Thông
48 báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
49         if(dt == DialogResult.Yes)
50         {
51             Application.Exit();
52         }
53     }
54
55     private void Form1_Load(object sender, EventArgs e)
56     {
57
58     }
59 }

```

3. Kết luận

Như vậy là chúng ta đã tìm hiểu xong công dụng của MenuStrip cũng như các thuộc tính và sự kiện của nó. Đây là một điều khiển rất quan trọng trong việc tạo các ứng dụng, vì đa số các ứng dụng đều sử dụng menu. Vậy nên các bạn hãy luyện tập thật nhiều để thành thạo nó nhé. Ở bài tiếp theo mình sẽ giới thiệu đến các bạn một điều khiển Menu nữa là ContextMenuStrip

Bài 11. Cách dùng ContextMenuStrip trong C# winforms

Trong hướng dẫn này mình sẽ giới thiệu các bạn một điều khiển Menu tiếp theo đó chính là ContextMenuStrip. Đây cũng là một dạng menu được sử dụng rất nhiều trong các ứng dụng trên Desktop.

Chúng ta sẽ cùng nhau tìm hiểu về công dụng cũng như các thuộc tính, phương thức và sự kiện của ContextMenuStrip. Sau đó mình sẽ thực hiện một chương trình áp dụng ContextMenuStrip để các bạn hiểu rõ hơn về cách hoạt động của nó nhé.

Table of Content

- 1. ContextMenuStrip trong C#
- 2. Ví dụ sử dụng điều khiển ContextMenuStrip C#
- 3. Kết luận

1. ContextMenuStrip trong C#

ContextMenuStrip dùng để thiết kế menu Popup (*menu ngữ cảnh*). Menu Popup là menu dạng như loại menu khi người dùng nhấn chuột phải vào màn hình Desktop thì hiện lên một menu. Trong lập trình ứng dụng Windows Form, menu Popup sẽ xuất hiện khi người dùng nhấn chuột phải vào các điều khiển như: *Form, Label, Button, TextBox,...*

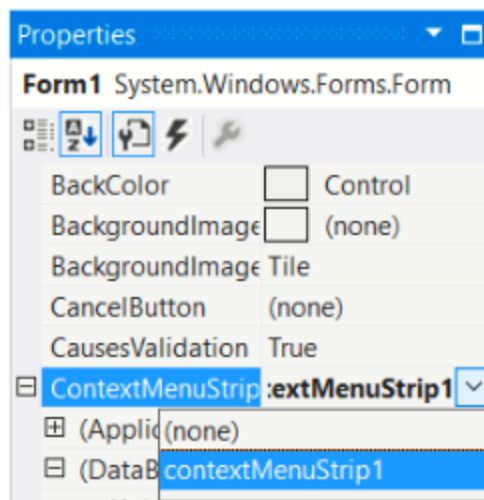
Phần lớn các điều khiển trong cửa sổ Toolbox đều hỗ trợ thuộc tính ContextMenuStrip. Do đó muốn menu Popup xuất hiện trên điều khiển nào thì cần khai báo thuộc tính ContextMenuStrip của điều khiển đó là một điều khiển ContextMenuStrip.

Một số thuộc tính thường dùng:

| Thuộc tính | Mô tả |
|--------------------|---|
| Items | Thêm các menu con, kiểu menu con thuộc một trong bốn dạng: <i>Separator, MenuItem, ComboBox, TextBox</i> . Lập trình viên có thể thêm các menu trong cửa sổ Items Collection Edition. Cửa sổ Items Collection Edition có thể mở lên bằng cách nhấp chuột trái vào biểu tượng của thuộc tính Items trong cửa sổ Properties. |
| RightToLeft | <ul style="list-style-type: none">• Mang giá trị true: Trình bày menu từ phải qua trái• Mang giá trị False: Trình bày menu từ trái qua phải |

Để ContextMenuStrip xuất hiện khi người dùng nhấp chuột phải vào Form thì phải thiết lập thuộc tính ContextMenuStrip của Form là tên của control ConTextMenuStrip

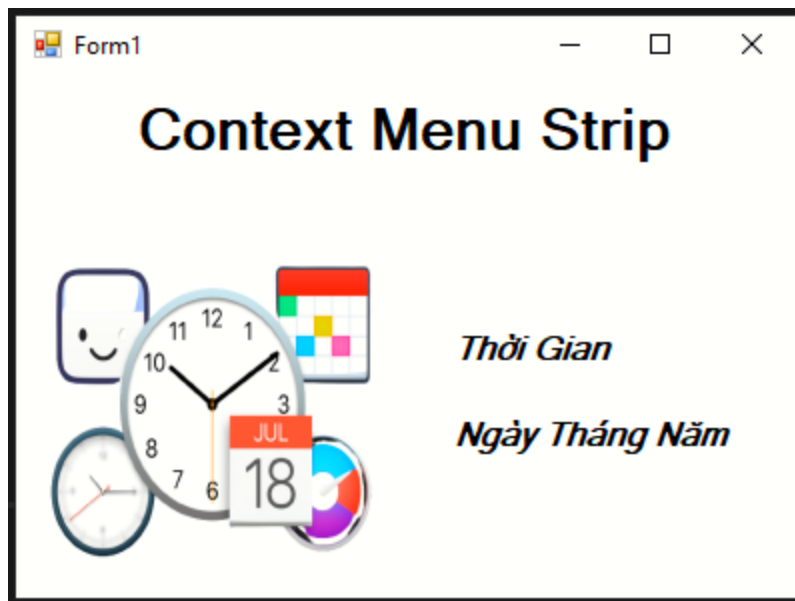
Ví dụ tên control ContextMenuStrip của bạn là contextMenuStrip1, thì thuộc tính ContextMenuStrip trong Form có tên là contextMenuStrip1



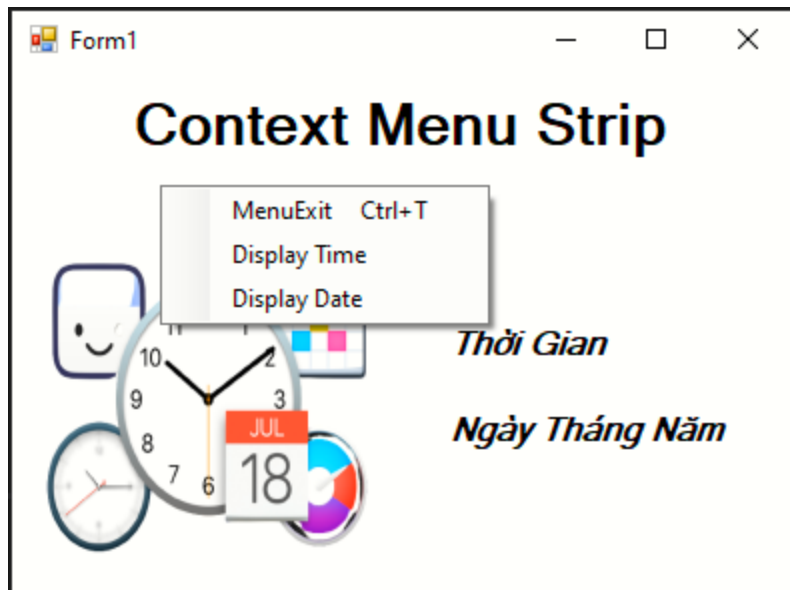
Trong ContextMenuStrip có một sự kiện rất phổ biến đó chính là **Click**, để cài đặt nó ta nhấp đúp chuột vào sự kiện thì tự động nó sẽ chuyển đến cửa sổ viết Code của bạn.

2. Ví dụ sử dụng điều khiển ContextMenuStrip C#

Trong ví dụ này mình sẽ viết một chương trình sử dụng ContextMenuStrip để mô tả công dụng của nó. Cụ thể sẽ tạo Form như mẫu, sau đó viết một số chức năng cho chương trình.



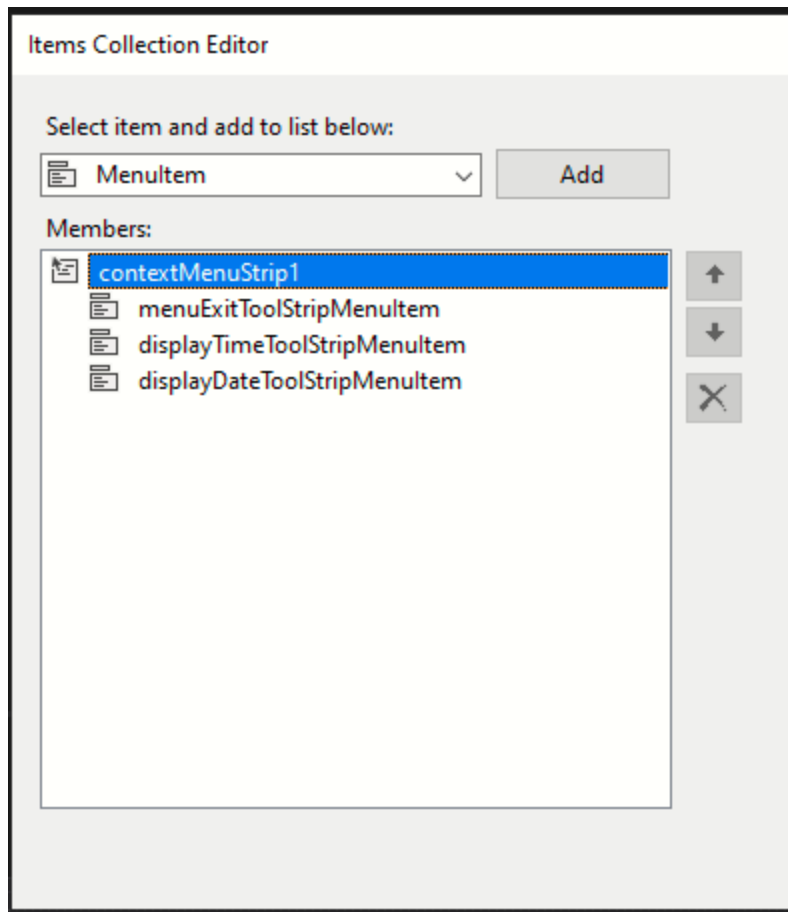
Yêu cầu: Tạo ContextMenuStrip cho Panel chứa Label ngày tháng và thời gian. Khi nhấn chuột phải lên Form thì một Menu sẽ hiển thị ra các chức năng như hình dưới đây.



- Khi chọn mục "**MenuExit**" hoặc nhấn tổ hợp phím **Ctrl + T** thì chương trình sẽ đóng.
- Khi chọn mục "**Display Time**" thì hiển thị thời gian hiện hành trên Label "**Thời Gian**".
- Khi chọn mục "**Display Date**" thì hiển thị ngày tháng năm hiện hành trên Label "**Ngày Tháng Năm**".

Việc đầu tiên ta cần tạo giao diện cho Form tương tự hình mẫu, bao gồm:

- 2 Label để hiển thị ngày tháng năm và thời gian hiện hành.
- 1 ContextMenuStrip để tạo Menu khi nhấn chuột phải lên Form. Trong ContextMenuStrip ta sẽ tạo 3 Items ở thuộc tính Items:



Sau khi tạo giao diện và thêm các Items trong ContextMenuStrip, bây giờ ta sẽ đi xử lý lần lượt các sự kiện.

Bước 1: Xử lý sự kiện *Click* cho Item **displayTimeToolStripMenuItem**, với chức năng khi người dùng chọn sẽ hiển thị thời gian hiện hành lên Label.

```
1 private void displayTimeToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     DateTime dt = DateTime.Now.Add(new TimeSpan());
4     lblTime.Text = String.Format("{0:hh:mm:ss tt}", dt);
5 }
```

Bước 2: Xử lý sự kiện *Click* cho Item **displayDateToolStripMenuItem**, với chức năng khi người dùng chọn sẽ hiển thị ngày tháng năm hiện hành trên Label.

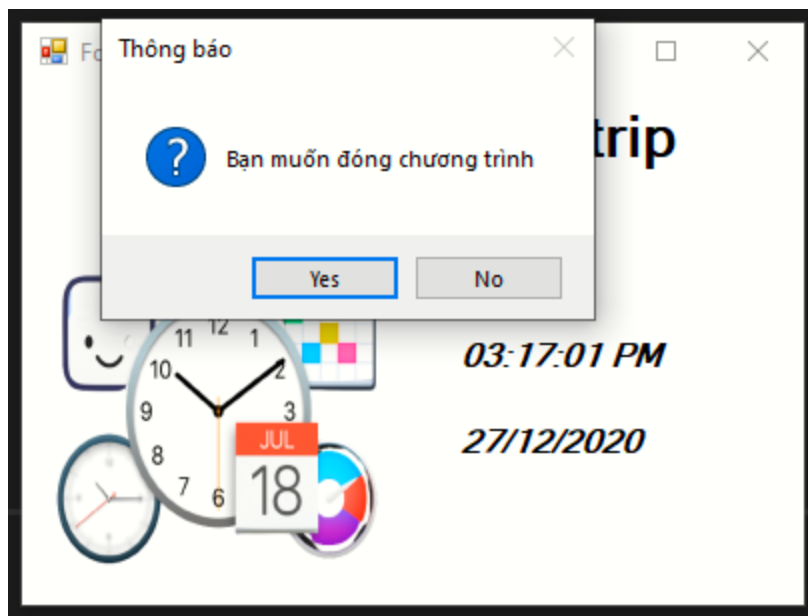
```
1 private void displayDateToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     DateTime dt = DateTime.Now.Add(new TimeSpan());
4     lblDate.Text = String.Format("{0:dd/MM/yyyy}", dt);
```

5 }

Bước 3: Xử lý sự kiện *Click* cho Item **menuExitToolStripMenuItem**, với chức năng khi người dùng chọn thì đóng chương trình.

```
1 private void menuExitToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     DialogResult dt = MessageBox.Show("Bạn muốn đóng chương trình", "Thông
4 báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
5     if (dt == DialogResult.Yes)
6     {
7         Application.Exit();
8     }
9 }
```

Kết quả:



Full Code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
```

```

7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace Cau2
12 {
13     public partial class Form1 : Form
14     {
15         public Form1()
16         {
17             InitializeComponent();
18         }
19
20         private void displayTimeToolStripMenuItem_Click(object sender, EventArgs e)
21         {
22             DateTime dt = DateTime.Now.Add(new TimeSpan());
23             lblTime.Text = String.Format("{0:hh:mm:ss tt}", dt);
24         }
25
26         private void displayDateToolStripMenuItem_Click(object sender, EventArgs e)
27         {
28             DateTime dt = DateTime.Now.Add(new TimeSpan());
29             lblDate.Text = String.Format("{0:dd/MM/yyyy}", dt);
30         }
31
32         private void menuExitToolStripMenuItem_Click(object sender, EventArgs e)
33         {
34             DialogResult dt = MessageBox.Show("Bạn muốn đóng chương trình", "Thông
35 báo", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
36             if (dt == DialogResult.Yes)
37             {
38                 Application.Exit();
39             }
40         }

```

```
41
42     private void Form1_Load(object sender, EventArgs e)
43     {
44
45     }
46 }
}
```

3. Kết luận

Như vậy là chúng ta đã tìm hiểu xong về công dụng của ContextMenuStrip cũng như các thuộc tính và sự kiện của nó. Đây là một điều khiển cũng khá quan trọng nên các bạn hãy luyện tập thật nhiều để thành thạo nó nhé. Bài tiếp theo mình sẽ giới thiệu về ToolStip.

Bài 12. Cách dùng ToolStrip trong C# winforms

Trong hướng dẫn này mình sẽ giới thiệu các bạn một điều khiển menu cuối cùng đó chính là ToolStrip trong C#. Đây cũng là một điều khiển menu quan trọng trong list control menu.

Chúng ta sẽ cùng nhau tìm hiểu về công dụng cũng như các thuộc tính và sự kiện của nó. Sau đó mình sẽ thực hiện một chương trình áp dụng ToolStrip để các bạn hiểu rõ hơn về cách hoạt động.

Table of Content

- 1. ToolStrip trong C# là gì?
- 2. Ví dụ sử dụng ToolStrip trong C#
- 3. Kết luận

1. ToolStrip trong C# là gì?

ToolStrip là điều khiển cho phép tạo thanh công cụ trên Form. Thông thường trong các ứng dụng Windows Forms, ToolStrip thường được bố trí phía dưới điều khiển MenuStrip.

Một số chức năng thường dùng của phần mềm ví dụ như *Save, Open, Paste, Copy,...* Sẽ được thể hiện trực quan, giúp đơn giản và tiện dụng cho người sử dụng thay vì phải chọn từ thanh Menubar.

Một số thuộc tính của ToolStrip C# thường dùng:

| Thuộc tính | Mô tả |
|-------------------------|---|
| Items | Quản lý việc thêm xóa các điều khiển trên ToolStrip |
| AllowItemReorder | <ul style="list-style-type: none">• Mang giá trị True: Cho phép người dùng sắp xếp lại vị trí của các điều khiển trên ToolStrip. Thay đổi vị trí bằng cách giữ phím Alt và nhấn chuột trái vào điều khiển và kéo đến vị trí mới trên ToolStrip• Mang giá trị False: Các vị trí của điều khiển trên ToolStrip cố định không thể thay đổi bởi người dùng |
| AllowMerge | Cho phép người dùng giữ phím Alt và giữ chuột tái vào điều khiển trên ToolStrip này và kéo thả vào một ToolStrip khác. |

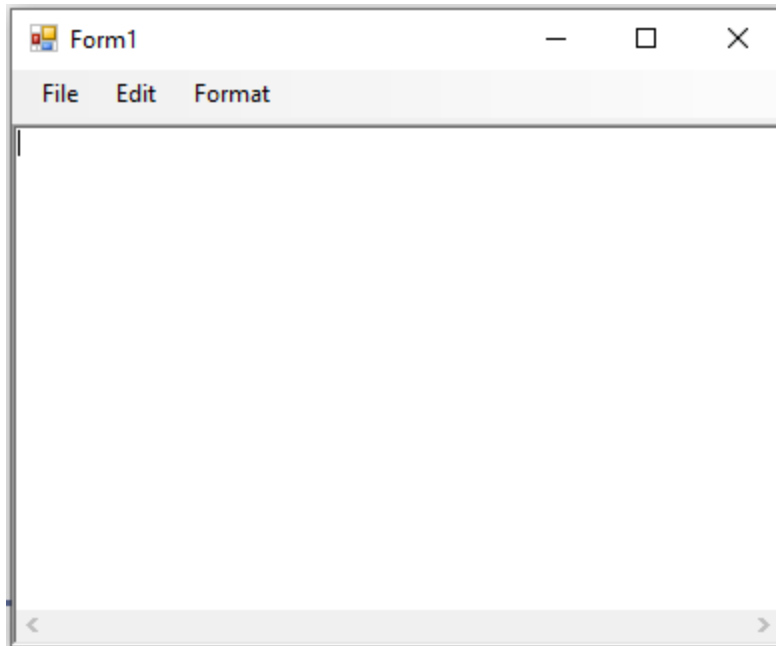
| Thuộc tính | Mô tả |
|-------------------------|---|
| | *Lưu ý: Thuộc tính này chỉ có hiệu lực khi thuộc tính AllowItemReoder là True. |
| Dock | Quy định vị trí hiển thị của ToolStrip trên Form |
| ShowItemTooltips | <ul style="list-style-type: none"> Mang giá trị True: Cho phép hiển thị chuỗi khai báo trong thuộc tính ToolTipText của mỗi điều khiển chứa trong ToolStrip Mang giá trị False: Chuỗi khai báo trong ToolTipText của các điều khiển chứa trong ToolStrip không được hiển thị |
| LayoutStyle | Kiểu trình bày của ToolStrip |
| CanOverflow | <ul style="list-style-type: none"> Mang giá trị True: Khi số lượng điều khiển trong ToolStrip vượt ra khỏi phạm vi kích thước thì những điều khiển này sẽ được thu nhỏ ở góc phải của ToolStrip Mang giá trị False: Những điều khiển nằm ngoài phạm vi kích thước sẽ không được thu nhỏ trong biểu tượng ở góc phải của ToolStrip |

Về lý thuyết thì nó chỉ có một số cái quan trọng như vậy, bây giờ chúng ta sẽ cùng nhau thực hiện để hiểu rõ hơn về công dụng cũng như cách thức hoạt động nhé.

2. Ví dụ sử dụng ToolStrip trong C#

Trong ví dụ này mình sẽ thực hiện một chương trình áp dụng điều khiển ToolStrip. Chương trình có chức năng tương tự như Notepad, có thể thực hiện các thao tác phím tắt và gõ văn bản.

Cụ thể ta sẽ thực hiện tạo giao diện như Form dưới đây sau đó xử lý một số sự kiện tương ứng cho nó.



Sau khi tạo giao diện tương tự như trên, ta thực hiện một số sự kiện.

- Menu File gồm có các menu con như sau:
 - **Exit**: dùng để thoát chương trình.
- Menu Edit gồm có các menu con như sau:
 - **Copy**: Dùng để sao chép đoạn văn bản có sẵn.
 - **Cut**: Dùng để đoạn văn bản có sẵn.
 - **Paste**: Dùng để dán đoạn văn bản.
 - **Delete**: Dùng để xóa đoạn văn bản.
 - **Select All**: Chọn tất cả nội dung trong RichTextBox.
 - **Undo**: Thao tác thực hiện trước đó sẽ được quay ngược trở lại.
- Menu Format gồm có các menu con như sau:
 - **Font**: Dùng thay đổi kiểu chữ trong RichTextBox.
 - **Color**: Dùng thay đổi màu chữ trong RichTextBox.
 - **WordWrap**: Dùng để hiển thị thành cuộn.

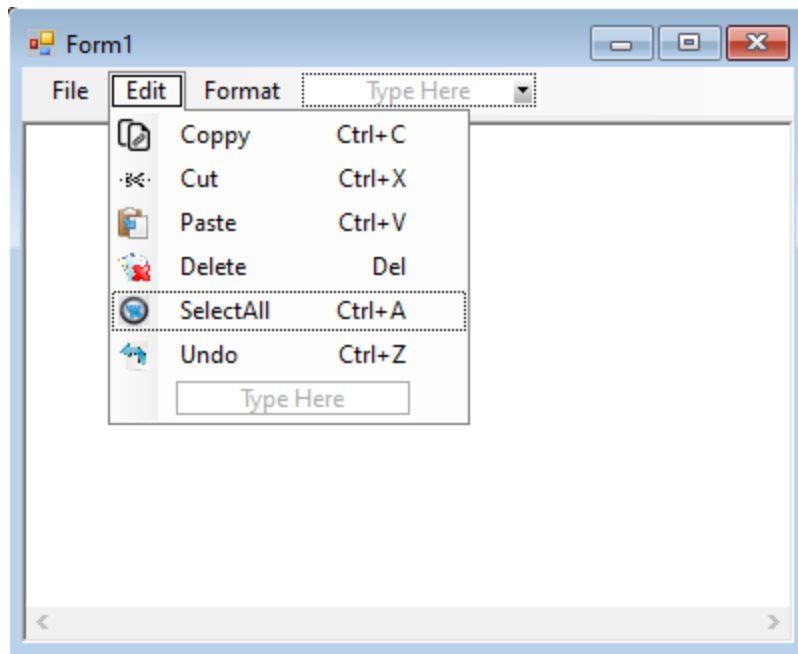
Bây giờ ta bắt đầu tạo giao diện cho Form, ta cần một số các control như sau:

- RichTextBox để viết văn bản và thực hiện các thao tác phím tắt.
- Một ToolStrip để tạo các menu.

Trên thanh ToolStrip ta sẽ tạo 3 menu lớn đó chính là File, Edit và Format.



Tiếp đến ta sẽ tạo các menu con cho từng menu lớn, đầu tiên sẽ là MenuFile với menu con là Exit. Sau đó lần lượt tới các menu lớn còn lại.



Bây giờ ta sẽ xử lý sự kiện cho các phím tắt Copy, Cut, Paste, Select All, Undo.

```
private void copyToolStripMenuItem_Click(object sender, EventArgs e)
{
    richTextBox1.Copy();
}

private void cutToolStripMenuItem_Click(object sender, EventArgs e)
{
    richTextBox1.Cut();
}

private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
{
    richTextBox1.Paste();
}

private void selectAllToolStripMenuItem_Click(object sender, EventArgs e)
{
    richTextBox1.SelectAll();
}

private void unToolStripMenuItem_Click(object sender, EventArgs e)
{

```



```

        richTextBox1.Undo();
    }

```

Đối với Delete ta sử dụng thuộc tính SelectionStart để xóa phần tử trong RichTextBox.

```

1 private void deleteToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     int i;
4     i = richTextBox1.SelectionStart;
5     richTextBox1.Text = richTextBox1.Text.Remove(i,
6     richTextBox1.SelectionLength);
7     richTextBox1.SelectionStart = i;
8 }

```

Tiếp theo ta sẽ thực hiện tạo Font và Color cho RichTextBox bằng cách sử dụng FontDialog và ColorDialog.

```

1 private void fontToolStripMenuItem_Click(object sender, EventArgs e)
2 {
3     FontDialog f = new FontDialog();
4     if (f.ShowDialog() == DialogResult.OK)
5     {
6         richTextBox1.Font = f.Font;
7     }
8 }
9
10 private void colorToolStripMenuItem_Click(object sender, EventArgs e)
11 {
12     ColorDialog c = new ColorDialog();
13     if (c.ShowDialog() == DialogResult.OK)
14     {
15         richTextBox1.ForeColor = c.Color;
16     }
17 }

```

Và cuối cùng là WordWrap để tạo thanh cuộn cho RichTextBox. Mặc định thì trạng thái trong WordWrap sẽ là False vì vậy ta cần bật nó lên thành True nhé.

```

1 private void wordRapToolStripMenuItem_Click(object sender, EventArgs e)

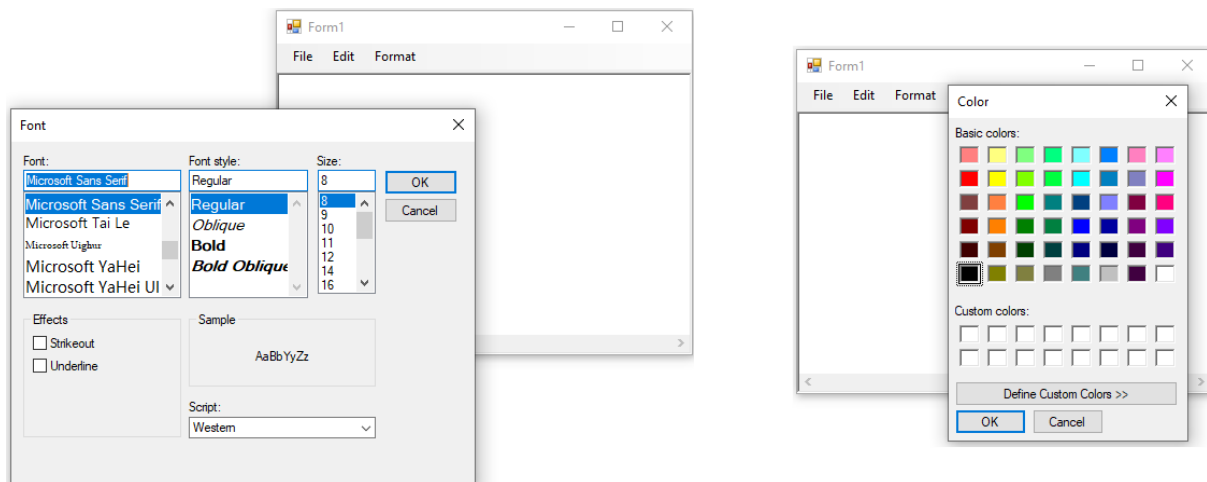
```

```

2  {
3      if(richTextBox1.WordWrap == true)
4      {
5          wordRapToolStripMenuItem.Checked = false;
6          richTextBox1.WordWrap = false;
7      }
8      else
9      {
10         wordRapToolStripMenuItem.Checked = true;
11         richTextBox1.WordWrap = true;
12     }
13 }

```

Kết quả: Mình sẽ Show kết quả khi chọn vào Font và Color, các thao tác còn lại các bạn có thể kiểm tra nhé.



3. Kết luận

Như vậy là chúng ta đã cùng nhau tìm hiểu xong ToolStrip và cũng kết thúc Series Control Menu. Hy vọng qua các bài học về Menu các bạn sẽ hiểu và có thể áp dụng để thực hiện tạo ứng dụng có chứa Menu. Ở bài tiếp theo mình sẽ giới thiệu các bạn các điều khiển trong nhóm Dialog.

Bài 13. Ứng dụng quản lý tài khoản đơn giản với C# Winforms

Trong bài viết này mình sẽ hướng dẫn các bạn thực hiện tạo một ứng dụng quản lý tài khoản đơn giản với các thao tác nhập xuất thông tin và lưu thông tin vào ListView.

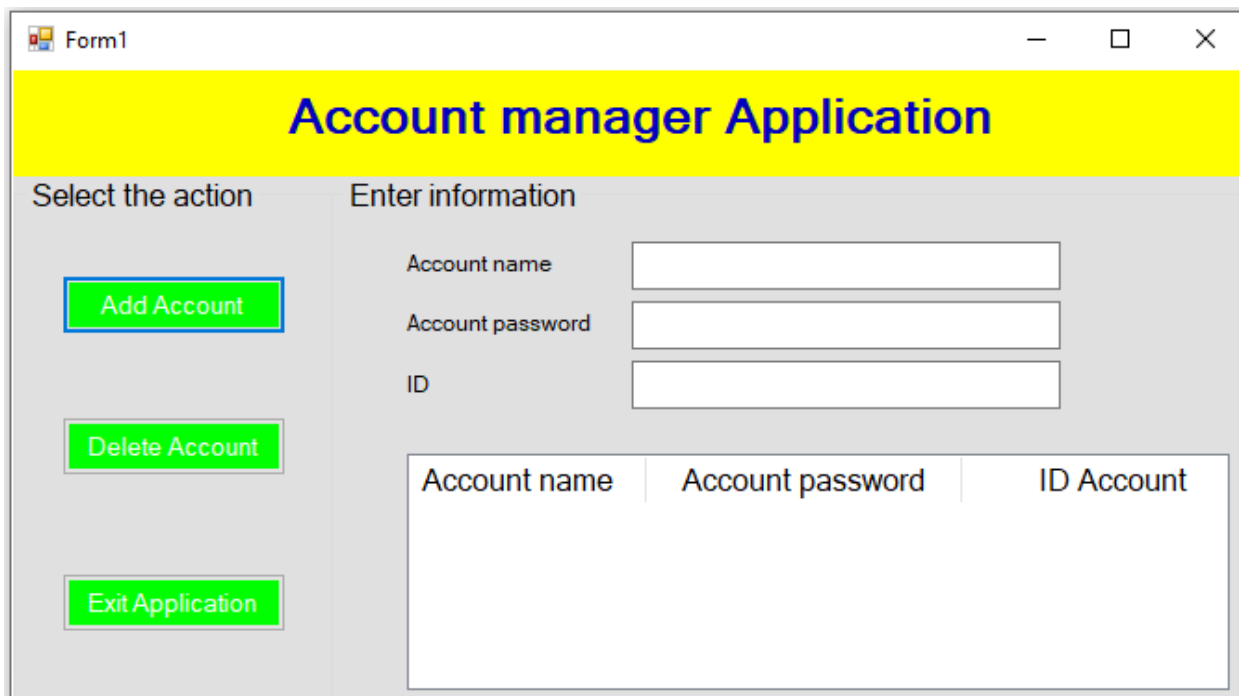
Để làm được bài này các bạn cần có kiến thức cơ bản về winforms, nếu các bạn là người mới thì có thể xem qua các bài học mà mình [đã hướng dẫn tại đây!](#)

Table of Content

- Gọi ý thực hiện ứng dụng
 - Gọi ý tạo giao diện
 - Gọi ý viết sự kiện
- Viết ứng dụng quản lý tài khoản
- Kết luận

Gợi ý thực hiện ứng dụng

Trong ví dụ này chúng ta sẽ thực hiện tạo ứng dụng với giao diện như sau:



| Account name | Account password | ID Account |
|--------------|------------------|------------|
|--------------|------------------|------------|

Ứng dụng có chức năng thêm tài khoản và xóa tài khoản, thông tin sau khi được nhập sẽ được lưu vào ListView với các cột thông tin tương ứng.

Gợi ý tạo giao diện

Tạo giao diện cho Form với các control sau đây:

- 1 Panel để tạo tiêu đề cho ứng dụng
- 2 GroupBox với hai phần khác nhau, 1 phần với các thao tác và 1 phần để thực hiện nhập và hiển thị thông tin.
- 3 Button với 3 chức năng đó là Add, Del và Exit.
- 3 TextBox để nhập thông tin tương ứng là Name Acc, Password Acc và ID.
- 1 ListView để lưu các thông tin của tài khoản đã được tạo.

Gợi ý viết sự kiện

Ta thực hiện viết sự kiện cho 3 Button "**Add Account**", "**Delete Account**", "**Exit Application**".

- Đối với "**Add Account**" ta cần kiểm tra xem thông tin trong ô TextBox đã được điền đầy đủ hay chưa, nếu một trong các ô vẫn chưa điền ta sẽ thông báo cho người dùng biết. Nếu thông tin đã được điền đầy đủ ta sẽ đưa thông tin đó vào các cột tương ứng trong ListView.
- Đối với "**Exit Application**" ta sử dụng *DialogResult* để hỏi người dùng có muốn đóng chương trình hay không, nếu YES thì đóng chương trình nếu NO thì hủy bỏ lệnh.
- Đối với "**Delete Account**" ta sử dụng phương thức *Remove()* trong ListView để xóa tài khoản được chọn trong ListView.

Viết ứng dụng quản lý tài khoản

Việc đầu tiên các bạn cần tạo giao diện cho Form, như hướng dẫn của mình ở trên, các bạn có thể tạo giao diện một cách dễ dàng.

Sau khi tạo xong giao diện, bây giờ ta bắt đầu xử lý sự kiện, ta sẽ lần lượt thực hiện từng Button nhé.

Bước 1: Xử lý sự kiện cho Button "**Add Account**".

Ta cần sử dụng thuộc tính *string.IsNullOrEmpty()* để kiểm tra xem các ô TextBox đã được nhập đầy đủ thông tin hay chưa. Nếu đã đầy đủ thông tin ta bắt đầu thêm các thông tin vào các cột tương ứng trong ListView.

```

private void button1_Click(object sender, EventArgs e)
{
    if(string.IsNullOrEmpty(txt_ID.Text) || string.IsNullOrEmpty(txt_name.Text)
1 || string.IsNullOrEmpty(txt_pass.Text))
2     {
3         MessageBox.Show("You have not entered full information
4 !!", "Notify", MessageBoxButtons.OK, MessageBoxIcon.Warning);
5     }
6     else
7     {
8 // tạo mới một ListViewItem
9         ListViewItem lstvItem = new ListViewItem();
10        lstvItem.Text = txt_name.Text;
11        ListViewItem.ListViewSubItem lstvsub
12= new ListViewItem.ListViewSubItem(lstvItem, txt_pass.Text);
13        ListViewItem.ListViewSubItem lstvsub1
14= new ListViewItem.ListViewSubItem(lstvItem, txt_ID.Text);
15// thêm các thông tin vào các cột tương ứng trong ListView
16        lstvItem.SubItems.Add(lstvsub);
17        lstvItem.SubItems.Add(lstvsub1);
18        listview_show.Items.Add(lstvItem);
19        MessageBox.Show("You have successfully added your
20account", "Notify", MessageBoxButtons.OK, MessageBoxIcon.Information);
21// sau khi thêm thông tin ta sẽ xóa các thông tin đã nhập ở các ô textbox để có thể nhập
22và thêm mới tài khoản khác.
23        txt_ID.Clear();
24        txt_name.Clear();
        txt_pass.Clear();
    }
}

```

Sau khi thêm các thông tin xong, ta cần xóa các thông tin trong ô TextBox để có thể nhập và thêm các tài khoản khác.

Kết quả:

Bước 2: Xử lý sự kiện cho Button "Delete Account".

Ta sẽ kiểm tra xem trong ListView đã có tài khoản hay chưa, nếu chưa có thì ta sẽ thông báo cho người dùng biết. Ngược lại nếu đã có tài khoản thì ta sẽ thực hiện xóa tài khoản mà người dùng đã chọn.

Để xóa một phần tử trong ListView ta sẽ sử dụng phương thức **Remove()** để xóa.

```

1 private void btn_del_Click(object sender, EventArgs e)
2 {
3     if(listview_show.SelectedIndices.Count <= 0)
4     {
5         MessageBox.Show("You have not selected the item you want to
6 delete", "Notify", MessageBoxButtons.OK, MessageBoxIcon.Warning);
7     }
8     if(listview_show.SelectedIndices.Count > 0)
9     {
10         DialogResult dl = MessageBox.Show("Bạn muốn xóa", "canh bao",
11 MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
12         if (dl == DialogResult.OK)

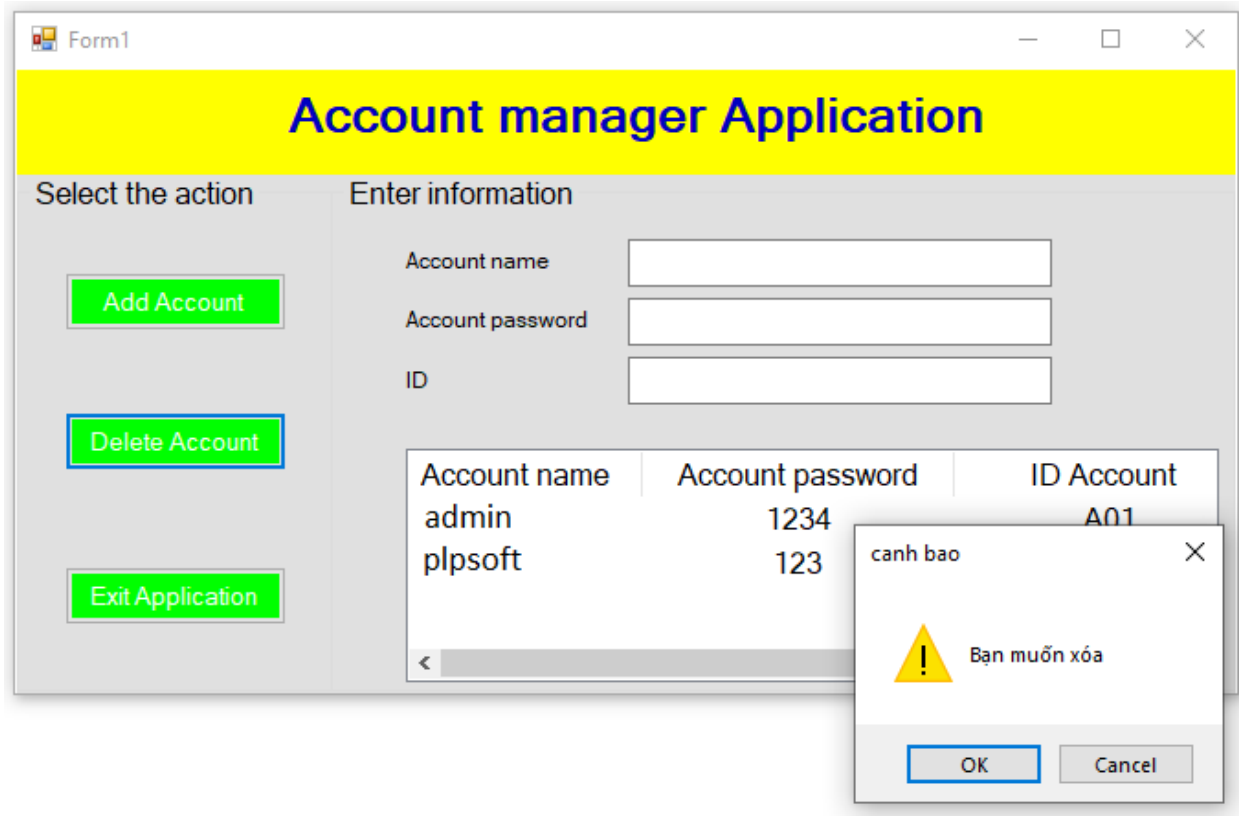
```

```

13         listview_show.Items.Remove(listview_show.SelectedItems[0]);
        }
    }
}

```

Kết quả:



Bước 3: Xử lý sự kiện cho Button "Exit Application".

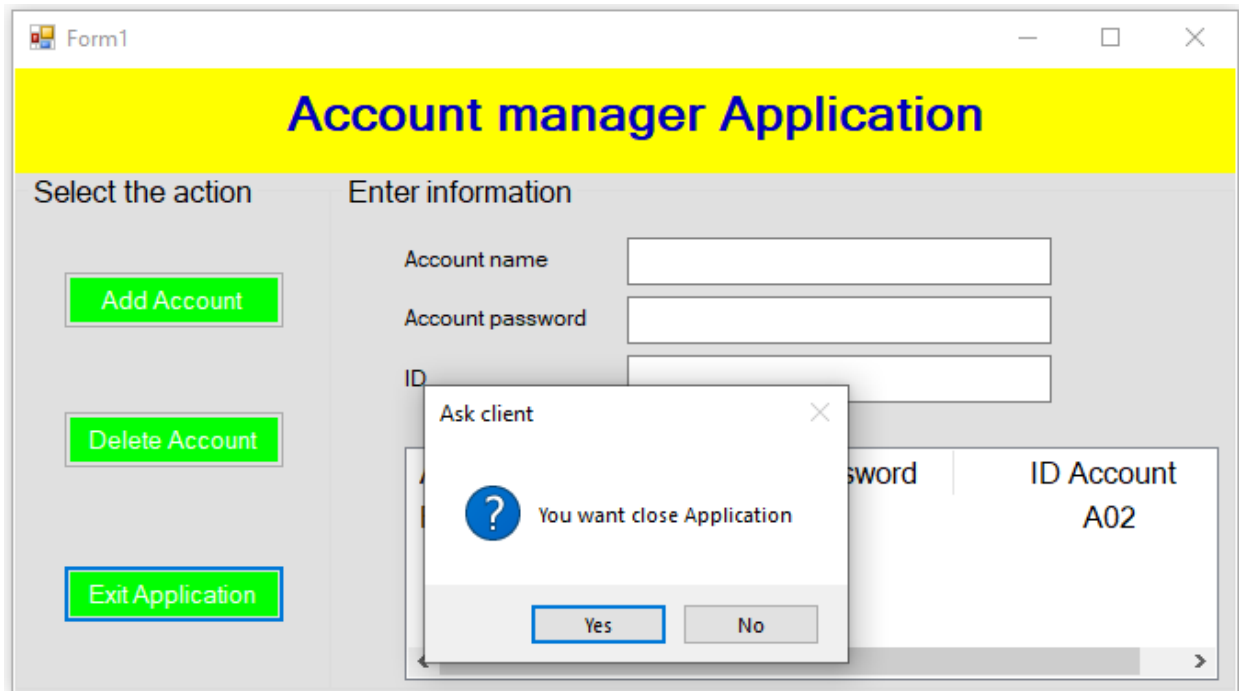
Ở sự kiện này thì khá đơn giản, ta chỉ cần sử dụng **DialogResul** để hỏi người dùng có muốn thoát hay không. Nếu chọn **YES** thì thoát chương trình, nếu chọn **NO** thì hủy bỏ lệnh.

```

1 private void Form1_Load(object sender, EventArgs e)
2 {
3     if (MessageBox.Show("You want open Application", "Ask client",
4         MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.Cancel)
5         Dispose();
6 }

```

Kết quả:



Kết luận

Như vậy là chúng ta đã thực hiện xong ứng dụng quản lý tài khoản đơn giản với các thao tác nhập xuất thông tin. Qua ví dụ này chúng ta có thể luyện tập tạo giao diện cho ứng dụng cùng với đó là thực hiện các thao tác đơn giản mà bất kỳ ứng dụng nào cũng có.

NGUỒN

Bài Tập C# - Giới Thiệu Về Lập Trình C# Và Các Bước Lập Trình Windows Form Căn Bản

<https://plpsoft.vn/30230-Bai-tap-C-Gioi-thieu-ve-lap-trinh-C-va-cac-buoc-lap-trinh-Windows-Form-can-ban>

Bài 1 - Bài Tập C# - Đối Tượng Form Và Một Số Control Thông Dụng Trong Lập Trình Windows Form

<https://plpsoft.vn/30231-Bai-tap-C-Bai-1-Doi-tuong-Form-va-mot-so-control-thong-dung-trong-lap-trinh-Windows-Form>

Bài 2 - Bài Tập C# - Cách Dùng Label - Button - Textbox Trong C# Winforms

<https://plpsoft.vn/30232-Bai-tap-C-Bai-2-Cach-dung-Label-Button-Textbox-trong-C-winforms>

Bài 3 - Bài Tập C# - Sử Dụng Checkbox - RadioButton Trong Lập Trình C# Winforms

<https://plpsoft.vn/30233-Bai-tap-C-Bai-3-Su-dung-Checkbox-RadioButton-trong-lap-trinh-C-winforms>

Bài 4 - Bài Tập C# - Sử Dụng ComboBox - ListBox Trong Lập Trình C# Winforms

<https://plpsoft.vn/30235-Bai-tap-C-Bai-4-Su-dung-ComboBox-ListBox-trong-lap-trinh-C-winforms>

Bài 5 - Bài Tập C# - Sử Dụng ToolTip - HelpProvider - ErrorProvider Trong C# Windows Form

<https://plpsoft.vn/30236-Bai-tap-C-Bai-5-Su-dung-ToolTip-HelpProvider-ErrorProvider-trong-C-windows-Form>

Bài 6 - Bài Tập C# - Cách Dùng ProgressBar - Timer Trong C# Windows Form

<https://plpsoft.vn/30237-Bai-tap-C-Bai-6-Cach-dung-ProgressBar-Timer-trong-C-Windows-Form>

Bài 7 - Bài Tập C# - Điều Khiển ListView Trong Lập Trình C# Winforms

<https://plpsoft.vn/30238-Bai-tap-C-Bai-7-Dieu-khien-ListView-trong-lap-trinh-C-winforms>

Bài 8 - Bài Tập C# - Cách Dùng TreeView Trong C# Winforms

<https://plpsoft.vn/30239-Bai-tap-C-Bai-8-Cach-dung-TreeView-trong-C-winforms>

Bài 9 - Bài Tập C# - Cách Dùng DateTimePicker - MonthCalendar Trong C# Winforms

<https://plpsoft.vn/30240-Bai-tap-C-Bai-9-Cach-dung-DateTimePicker-MonthCalendar-trong-C-winforms>

Bài 10 - Bài Tập C# - Cách Dùng MenuStrip Trong C# Winforms

<https://plpsoft.vn/30241-Bai-tap-C-Bai-10-Cach-dung-MenuStrip-trong-C-winforms>

Bài 11 - Bài Tập C# - Cách Dùng ContextMenuStrip Trong C# Winforms

<https://plpsoft.vn/30242-Bai-tap-C-Bai-11-Cach-dung-ContextMenuStrip-trong-C-winforms>

Bài 12 - Bài Tập C# - Cách Dùng ToolStrip Trong C# Winforms

<https://plpsoft.vn/30243-Bai-tap-C-Bai-12-Cach-dung-ToolStrip-trong-C-winforms>

Bài Tập C# - Bài 13 - Ứng Dụng Quản Lý Tài Khoản Đơn Giản Với C# Winforms

<https://plpsoft.vn/30244-Bai-tap-C-Bai-13-Ung-dung-quan-ly-tai-khoan-don-gian-voi-C-Winforms>