

Chương 11

Tạo đối tượng giao diện cá nhân hóa bằng VC#

11.0 Dẫn nhập

11.1 Tổng quát về giao diện cá nhân hóa

11.2 Xây dựng User control & ứng dụng

11.3 Xây dựng Inherited control & ứng dụng

11.4 Xây dựng Owner-drawn control & ứng dụng

11.5 Kết chương



11.0 Dẫn nhập

- ❑ Chương này giới thiệu cách thức dùng tính thừa kế để tạo mới 3 loại đối tượng giao diện cá nhân hóa phổ biến là User Control, Inherited Control và Owner-drawn Control.
- ❑ Chương này cũng giới thiệu cách thức viết chương trình sử dụng lại các đối tượng giao diện cá nhân hóa.



11.1 Tổng quát về giao diện cá nhân hóa

- ❑ Mỗi chương trình dùng giao diện đồ họa thường có nhiều cửa sổ giao diện. Mỗi cửa sổ giao diện chứa nhiều đối tượng giao diện. Microsoft đã cung cấp sẵn nhiều đối tượng giao diện (control) phổ dụng để ta thiết kế form giao diện dễ dàng. Tuy nhiên trong từng ứng dụng, có thể ta cần 1 số đối tượng giao diện đặc thù, ta gọi chúng là đối tượng cá nhân hóa (user control).
- ❑ Thường có 3 dạng đối tượng giao diện cá nhân hóa :
 1. **User Control** : là dạng đơn giản nhất, nó thừa kế class UserControl sẵn có, tích hợp nhiều control có sẵn để tạo đối tượng cá nhân hóa. Thí dụ 1 LoginControl gồm 2 TextBox để nhập username, password và 1 Button đăng nhập.



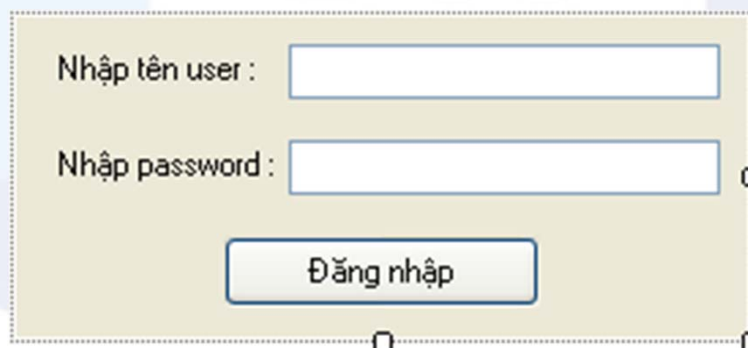
11.1 Tổng quát về giao diện cá nhân hóa

2. **Inherited Control** : chức năng và hành vi của nó gần giống control đã có sẵn. Để xây dựng nó, ta thừa kế class có sẵn mà chức năng gần giống nhất, rồi hiệu chỉnh (override) 1 số tác vụ để thể hiện chức năng thay đổi. Ta cũng có thể thêm mới 1 số tác vụ để thể hiện các chức năng tăng cường. Thí dụ MyTextBox có chức năng gần giống như TextBox có sẵn, nhưng nó có nhiều chế độ khác nhau, ở mỗi chế độ nó phản ứng khác nhau. Thí dụ nếu ở chế độ nhập số nguyên, nó chỉ cho phép nhập ký số, chứ không cho nhập ký tự khác.
3. **Owner-drawn control** : chức năng giống y như control có sẵn nhưng bộ mặt giao diện thì khác. Ta sẽ thừa kế class có sẵn mà chức năng giống y rồi override tác vụ OnPaint để vẽ lại bộ mặt mới. Thí dụ HeartControl là một Button nhưng bộ mặt không phải là khung chữ nhật bình thường mà là trái tim màu đỏ tươi.



11.2 Xây dựng User control

- ❑ Qui trình xây dựng 1 hay nhiều User Control gồm các bước chính :
 1. chạy Visual Studio .Net, mở/tạo Project loại "Windows Control Library" để quản lý 1 hay nhiều user control.
 2. Tạo mới 1 User Control rồi thiết kế giao diện/viết code cho nó.
 3. Dịch project ra file *.dll, ta gọi file này là thư viện chứa các user control.
- ❑ Thí dụ ta hãy xây dựng 1 User Control có tên là LoginControl, nó gồm 2 TextBox và 1 Button để giúp người dùng đăng ký tài khoản để truy xuất hệ thống. Hình ảnh LoginControl như sau :



The image shows a user control interface for login. It consists of a light yellow rectangular box with a dashed border. Inside the box, there are two text input fields. The first field is labeled "Nhập tên user :" and the second field is labeled "Nhập password :". Below these two fields is a button labeled "Đăng nhập". The entire control is shown within a larger window, with a faint "BK" watermark visible in the background.

11.2 Xây dựng User control

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Control Library" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. MyUserControls), click button OK để tạo Project theo các thông số đã khai báo.
3. Ngay sau Project vừa được tạo ra, nó có sẵn 1 User Control mới có tên mặc định là UserControl1, nó chỉ là 1 vùng hình chữ nhật trống, chứ chưa có gì. Dời chuột về cửa sổ Solution Explorer (thường ở trên phải màn hình), ấn kép chuột vào mục UserControl1.cs để hiển thị menu lệnh, chọn option Rename, nhập tên mới là LoginControl.cs và chọn button Yes khi được hỏi.



11.2 Xây dựng User control

4. Nếu cửa sổ ToolBox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Duyệt tìm phần tử Label (trong nhóm Common Controls), chọn nó, dờ chuột về vị trí thích hợp trong LoginControl và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Nhập tên user :". Nếu cần, hãy thay đổi vị trí và kích thước của Label và của LoginControl.
5. Dờ chuột về cửa sổ ToolBox, duyệt tìm phần tử TextBox (trong nhóm Common Controls), chọn nó, dờ chuột về vị trí thích hợp trong LoginControl (bên phải Label vừa vẽ) và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính (Name) = txtUser. Nếu cần, hãy thay đổi vị trí và kích thước của TextBox.
6. Lặp lại các bước 4 và 5 để vẽ Label "Nhập password :", TextBox có (Name) = txtPassword, 1 button "Đăng nhập" có (Name) = btnLogin.




11.2 Xây dựng User control

4. Nếu cửa sổ Toolbox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Duyệt tìm phần tử Label (trong nhóm Common Controls), chọn nó, dời chuột về vị trí thích hợp trong LoginControl và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Nhập tên user :". Nếu cần, hãy thay đổi vị trí và kích thước của Label và của LoginControl.
5. Dời chuột về cửa sổ Toolbox, duyệt tìm phần tử TextBox (trong nhóm Common Controls), chọn nó, dời chuột về vị trí thích hợp trong LoginControl (bên phải Label vừa vẽ) và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính (Name) = txtUser. Nếu cần, hãy thay đổi vị trí và kích thước của TextBox.
6. Lặp lại các bước 4 và 5 để vẽ Label "Nhập password :", TextBox có (Name) = txtPassword, 1 button "Đăng nhập" có (Name) = btnLogin.



11.2 Xây dựng User control

7. Dời chuột về và chọn button "Đăng nhập", cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện Button, duyệt tìm sự kiện Click, ấn kép chuột vào comboBox bên phải của Click để máy tạo tự động hàm xử lý rồi viết code cho hàm này như sau :

```
private void btnLogin_Click(object sender, EventArgs e) {  
    //kiểm tra đã nhập user name chưa  
    if (txtUser.Text.Length == 0) {  
        MessageBox.Show("Hãy nhập tên user."); return;  
    }  
    //kiểm tra đã nhập password chưa  
    if (txtPassword.Text.Length == 0) {  
        MessageBox.Show("Hãy nhập password."); return;  
    }  
}
```



11.2 Xây dựng User control

```
//tạo sự kiện Click để gọi hàm xử lý sự kiện Click
//do người lập trình ứng dụng viết
OnSubmitClicked(sender,e);
}
```

8. Viết thêm đoạn code định nghĩa delegate, event và 2 thuộc tính UserName, Password như sau (nằm trước hay sau hàm xử lý Click chuột cho button) :

```
//định nghĩa delegate phục vụ cho event
public delegate void SubmitClickedHandler(object sender, EventArgs e);
//định nghĩa event SubmitClicked
public event SubmitClickedHandler SubmitClicked;
```



11.2 Xây dựng User control

```
//định nghĩa hàm xử lý sự kiện SubmitClicked
protected virtual void OnSubmitClicked(object sender, EventArgs e) {
    // kiểm tra xem có hàm xử lý sự kiện SubmitClicked ?
    //nếu có thì gọi nó
    if (SubmitClicked != null) {
        SubmitClicked(sender, e); // Notify Subscribers
    }
}

//định nghĩa thuộc tính giao tiếp có tên là UserName
public string UserName {
    get { return txtUser.Text; }
    set { txtUser.Text = value; }
}
```



11.2 Xây dựng User control

//định nghĩa thuộc tính giao tiếp có tên là Password

```
public string Password {  
    get { return txtPassword.Text; }  
    set { txtPassword.Text = value; }  
}
```

9. Chọn menu Build.Build Solution để dịch và tạo file thư viện chứa các user control. Nếu có lỗi thì sửa và dịch lại.
10. Nếu dịch thành công, file thư viện có tên là MyUserControls.dll sẽ được tạo ra trong thư mục con Debug (hay Release tùy chế độ dịch) trong thư mục chứa Project. Ta nên copy file này vào thư mục chung chứa các file thư viện để sau này dùng tiện lợi hơn.



Xây dựng ứng dụng dùng User Control

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. UseLoginControl), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lập 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.



Xây dựng ứng dụng dùng User control

4. Nếu cửa sổ Toolbox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Dời chuột vào trong cửa sổ Toolbox, ấn phải chuột để hiển thị menu lệnh, chọn option "Choose Items". Khi cửa sổ "Choose Toolbox Items" hiển thị, click chuột vào button Browse để hiển thị cửa sổ duyệt tìm file, hãy duyệt tìm đến thư mục chứa file MyUserControls.dll vừa xây dựng được trong các slide trước, chọn file dll rồi click button OK để "add" các usercontrol trong thư viện này vào cửa sổ Toolbox của Project ứng dụng. Bây giờ việc dùng LoginControl giống y như các điều khiển có sẵn khác.
5. Duyệt tìm phần tử LoginControl (trong nhóm General ở cuối cửa sổ Toolbox), chọn nó, dời chuột về vị trí thích hợp trong Form và vẽ nó với kích thước mong muốn.



Xây dựng ứng dụng dùng User control

6. Chọn đối tượng LoginControl để hiển thị cửa sổ thuộc tính của nó, click chuột vào button Events để hiển thị các event của nó. duyệt tìm event SubmitClicked vào tạo hàm xử lý cho event này. Viết code cho hàm xử lý như sau :

```
private void loginControl1_SubmitClicked(object sender, EventArgs e) {  
    //viết code xử lý việc đăng nhập tài khoản  
    //ở đây chỉ hiển thị thông báo để kiểm tra  
    MessageBox.Show("Đã đăng ký tài khoản : "  
        + loginControl1.UserName);  
}
```

7. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hãy thử sử dụng đối tượng LoginControl và đánh giá kết quả.



11.3 Xây dựng Inherited control

- ❑ Qui trình xây dựng 1 hay nhiều Inherited Control gồm các bước chính :
 1. chạy Visual Studio .Net, mở/tạo Project loại "Windows Control Library" để quản lý 1 hay nhiều user control.
 2. Ấn phải chuột vào gốc của cây Project trong cửa sổ "Solution Explorer", chọn option Add.User Control để tạo mới 1 User Control.
 3. Hiện thị cửa sổ soạn mã nguồn của user Control, hiệu chỉnh lại tên class base cần thừa kế rồi override/tăng cường các tác vụ chức năng mong muốn.
 4. Dịch project ra file *.dll, ta gọi file này là thư viện chứa các user control.



11.3 Xây dựng Inherited control

- ❑ Thí dụ ta hãy xây dựng 1 Inherited Control có tên là MyTextBox, nó là TextBox nhưng có thể hoạt động ở 1 trong nhiều chế độ khác nhau :
 - Common (giống như textbox của .Net),
 - Text (chỉ cho nhập các ký tự alphabet),
 - NumInt (chỉ cho phép nhập các ký số),
 - NumReal (chỉ cho phép nhập các ký số và dấu chấm thập phân).



11.3 Xây dựng Inherited control

- ❑ Qui trình xây dựng MyTextBox và chứa nó trong thư viện có sẵn (thư viện chứa đối tượng LoginControl) như sau :
 1. Chạy VS .Net, chọn menu File.Open.Project để hiển thị cửa sổ duyệt file. Duyệt và tìm file *.sln quản lý Project "Windows Control Library" có sẵn để mở lại Project này.
 2. Quan sát cây Project, chúng ta đã thấy có mục LoginControl.cs quản lý user control đã xây dựng trong mục 9.2. Ấn phải chuột vào gốc của cây Project trong cửa sổ "Solution Explorer", chọn option Add.User Control để tạo mới 1 User Control, nhập tên là MyTextBox.cs rồi click button Add để tạo mới nó.
 3. Lúc này control mới chỉ là 1 vùng hình chữ nhật trống. Dời chuột về mục MyTextBox.cs trong cửa sổ Project, ấn phải chuột trên nó rồi chọn option "View Code" để hiển thị cửa sổ soạn mã nguồn cho MyTextBox control.



11.3 Xây dựng Inherited control

4. Thêm lệnh định nghĩa kiểu liệt kê các chế độ làm việc của MyTextBox :

```
public enum ValidationType {  
    Common = 0,           //giống như TextBox bình thường  
    NumInt,               //chỉ nhận các ký số  
    NumReal,              //chỉ nhận các ký số và dấu chấm thập phân  
    Text };               //chỉ nhận các ký tự chữ
```

5. Hiệu chỉnh lại lệnh định nghĩa class MyTextBox để thừa kế class TextBox (thay vì UserControl như mặc định). Nội dung chi tiết của class MyTextBox được liệt kê ở các slide sau.
6. Chọn menu Build.Build Solution để dịch và tạo file thư viện chứa các user control. Nếu có lỗi thì sửa và dịch lại. Lưu ý khi máy báo lỗi ở hàng lệnh `this.AutoScaleMode = ...` thì hãy chú thích hàng lệnh này hay xóa nó luôn cũng được.



11.3 Xây dựng Inherited control

```
public partial class MyTextBox : TextBox {  
    bool fPoint;  
    //hàm contructor  
    public MyTextBox() : base() {  
        InitializeComponent();  
        //đăng ký hàm xử lý sự kiện KeyPress  
        this.KeyPress += new KeyPressEventHandler(OnKeyPress);  
    }  
    //định nghĩa thuộc tính ValidateFor miêu tả chế độ làm việc  
    private int intValidType = (int)ValidationType.Text;  
    public ValidationType ValidateFor {  
        get { return (ValidationType)intValidType; }  
        set { intValidType = (int)value; }  
    }  
}
```



11.3 Xây dựng Inherited control

```
//hàm xử lý sự kiện gõ phím KeyPress
protected void OnKeyPress(object sender,
    KeyPressEventArgs e) {
    //xác định mã ký tự được nhập
    char ch = e.KeyChar;
    //kiểm tra chế độ hoạt động để phản ứng
    switch (intValidType) {
        case (int)ValidationType.Common:
            //nếu là kiểu tổng quát, thì không xử lý thêm gì cả
            return;
        case (int)ValidationType.NumInt:
            //nếu là kiểu số nguyên thì chỉ nhận ký số
            if (!Char.IsDigit(ch)) e.KeyChar = (char)0;
            return;
    }
}
```



11.3 Xây dựng Inherited control

```
case (int)ValidationType.NumReal:
    //nếu là kiểu số thực thì chỉ nhận ký số + dấu .
    if (Char.IsDigit(ch)) return;
    if (ch == '.' && fPoint==false) {
        fPoint = true; return;
    }
    e.KeyChar = (char)0;
    return;
case (int)ValidationType.Text:
    //nếu là kiểu chuỗi văn bản thì chỉ nhận ký tự chữ
    ch = Char.ToLower(ch);
    if (ch < 'a' || 'z' < ch) e.KeyChar = (char)0;
    return;
}} //kết thúc lệnh switch, hàm OnKeyPress, ...
```



Xây dựng ứng dụng dùng Inherited control

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. UseMyTextBox), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lập 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.



Xây dựng ứng dụng dùng Inherited control

4. Nếu cửa sổ Toolbox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Dời chuột vào trong cửa sổ Toolbox, ấn phải chuột để hiển thị menu lệnh, chọn option "Choose Items". Khi cửa sổ "Choose Toolbox Items" hiển thị, click chuột vào button Browse để hiển thị cửa sổ duyệt tìm file, hãy duyệt tìm đến thư mục chứa file MyUserControls.dll vừa xây dựng được trong các slide trước, chọn file dll rồi click button OK để "add" các usercontrol trong thư viện này vào cửa sổ Toolbox của Project ứng dụng.
5. Duyệt tìm phần tử Label (trong nhóm Common Controls), chọn nó, dời chuột về vị trí thích hợp trong form và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Nhập chuỗi bất kỳ :". Nếu cần, hãy thay đổi vị trí và kích thước của Label và Form.



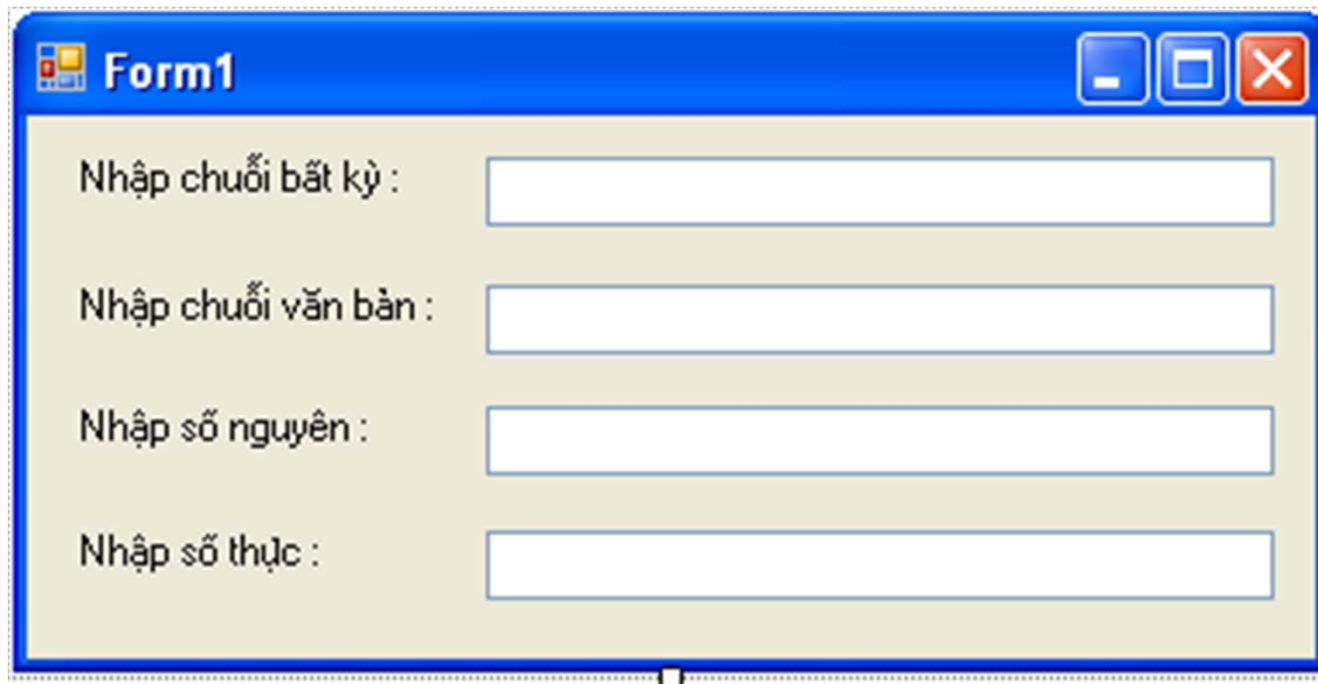
Xây dựng ứng dụng dùng Inherited control

6. Duyệt tìm phần tử MyTextBox (trong nhóm General ở cuối cửa sổ Toolbox), chọn nó, dời chuột về vị trí thích hợp trong Form (bên phải Label vừa vẽ) và vẽ nó với kích thước mong muốn. Vào cửa sổ thuộc tính của đối tượng MyTextBox vừa vẽ, đặt thuộc tính (Name) = txtCommon, thuộc tính ValidateFor = Common để nó hoạt động ở chế độ nhập ký tự bình thường.
 7. Lặp 2 bước 5 và 6 ba lần để tạo thêm 3 cặp (Label, MyTextBox) khác, các MyTextBox tạo mới lần lượt có thuộc tính ValidateFor = Text, NumInt, NumReal để hoạt động trên hoặc chuỗi văn bản, hoặc số nguyên, hoặc số thực.
- ❑ Đối với các đối tượng giống nhau, ta có thể dùng kỹ thuật Copy-Paste để nhân bản vô tính chúng cho dễ dàng.



Xây dựng ứng dụng dùng Inherited control

- ❑ Sau khi thiết kế xong, Form có dạng sau :



The screenshot shows a Windows application window titled "Form1". Inside the window, there are four text input fields arranged vertically. Each field is preceded by a label in Vietnamese: "Nhập chuỗi bất kỳ :", "Nhập chuỗi văn bản :", "Nhập số nguyên :", and "Nhập số thực :". The fields are empty and have a standard Windows text box appearance with a light gray border and a white background.

8. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hãy thử nhập các loại ký tự vào các đối tượng MyTextBox và đánh giá kết quả.

11.4 Xây dựng Owner-drawn control

- ❑ Qui trình xây dựng 1 hay nhiều Owner-drawn Control cũng giống như Inherited control, gồm các bước chính :
 1. chạy Visual Studio .Net, mở/tạo Project loại "Windows Control Library" để quản lý 1 hay nhiều user control.
 2. Ấn phải chuột vào gốc của cây Project trong cửa sổ "Solution Explorer", chọn option Add.User Control để tạo mới 1 User Control.
 3. Hiện thị cửa sổ soạn mã nguồn của User Control, hiệu chỉnh lại tên class base cần thừa kế rồi override/tăng cường các tác vụ chức năng mong muốn, trong đó thiết yếu nhất là hàm OnPaint để vẽ bộ mặt của đối tượng giao diện.
 4. Dịch project ra file *.dll, ta gọi file này là thư viện chứa các user control.



11.4 Xây dựng Owner-drawn control

- ❑ Thí dụ ta hãy xây dựng 1 Owner-drawn Control có tên là HeartButton, nó là Button nhưng bộ mặt không phải là hình chữ nhật có đường viền thông thường mà là một trái tim màu đỏ tươi.
- ❑ Qui trình xây dựng HeartButton và chứa nó trong thư viện có sẵn (thư viện đã tạo ra trong mục 9.2 và 9.3) như sau :
 1. Chạy VS .Net, chọn menu File.Open.Project để hiển thị cửa sổ duyệt file. Duyệt và tìm file *.sln quản lý Project "Windows Control Library" có sẵn để mở lại Project này.
 2. Quan sát cây Project, ta thấy có mục LoginControl.cs quản lý usercontrol đã xây dựng trong mục 9.2, mục MyTextBox.cs quản lý inherited control đã xây dựng trong mục 9.3. Ấn phải chuột vào gốc cây Project trong cửa sổ "Solution Explorer", chọn option Add.User Control để tạo mới 1 User Control, nhập tên là HeartButton.cs rồi click button Add để tạo mới nó.



11.4 Xây dựng Owner-drawn control

3. Lúc này control mới chỉ là 1 vùng hình chữ nhật trống. Dời chuột về mục HeartButton.cs trong cửa sổ Project, ấn phải chuột trên nó rồi chọn option "View Code" để hiển thị cửa sổ soạn mã nguồn cho HeartButton control.
4. Hiệu chỉnh lại lệnh định nghĩa class HeartButton để thừa kế class Button (thay vì UserControl như mặc định). Nội dung chi tiết của class HeartButton được liệt kê ở các slide sau.
5. Chọn menu Build.Build Solution để dịch và tạo file thư viện chứa các user control. Nếu có lỗi thì sửa và dịch lại. Lưu ý khi máy báo lỗi ở hàng lệnh `this.AutoScaleMode = ...` thì hãy chú thích hàng lệnh này hay xóa nó luôn cũng được.



11.4 Xây dựng Owner-drawn control

```
public partial class HeartButton : Button {  
    //hàm constructor của class  
    public HeartButton() {  
        InitializeComponent();  
    }  
    //hàm vẽ bộ mặt của button  
    protected override void OnPaint(PaintEventArgs e) {  
        //xác định đối tượng vẽ của Button  
        Graphics g = e.Graphics;  
        //tạo đối tượng image gốc chứa ảnh trái tim màu đỏ  
        Image bgimg = Image.FromFile("c:\\bgbutton.jpg");  
        //vẽ inage gốc theo chế độ zoom vào button  
        g.DrawImage(bgimg, 0, 0, this.Width, this.Height);  
    }  
}
```



11.4 Xây dựng Owner-drawn control

//định nghĩa đối tượng miêu tả cách thức hiển thị chuỗi

```
StringFormat format1 =
```

```
    new StringFormat (StringFormatFlags.NoClip);
```

```
format1.Alignment = StringAlignment.Center;
```

//vẽ chuỗi caption của button

```
g.DrawString(this.Text, this.Font, Brushes.White,  
             this.Width / 2, this.Height / 3, format1);
```

```
} //hết hàm OnPaint
```

```
} //hết class HeartButton
```



Xây dựng ứng dụng dùng Owner-drawn control

1. Chạy VS .Net, chọn menu File.New.Project để hiển thị cửa sổ New Project.
2. Mở rộng mục Visual C# trong TreeView "Project Types", chọn mục Windows, chọn icon "Windows Application" trong listbox "Templates" bên phải, thiết lập thư mục chứa Project trong listbox "Location", nhập tên Project vào textbox "Name:" (td. UseHeartButton), click button OK để tạo Project theo các thông số đã khai báo.
3. Form đầu tiên của ứng dụng đã hiển thị trong cửa sổ thiết kế, việc thiết kế form là quá trình lập 4 thao tác tạo mới/xóa/hiệu chỉnh thuộc tính/tạo hàm xử lý sự kiện cho từng đối tượng cần dùng trong form.



Xây dựng ứng dụng dùng Owner-drawn control

4. Nếu cửa sổ Toolbox chưa hiển thị, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Dời chuột vào trong cửa sổ Toolbox, ấn phải chuột để hiển thị menu lệnh, chọn option "Choose Items". Khi cửa sổ "Choose Toolbox Items" hiển thị, click chuột vào button Browse để hiển thị cửa sổ duyệt tìm file, hãy duyệt tìm đến thư mục chứa file MyUserControls.dll vừa xây dựng được trong các slide trước, chọn file dll rồi click button OK để "add" các usercontrol trong thư viện này vào cửa sổ Toolbox của Project ứng dụng.
5. Duyệt tìm phần tử HeartButton (trong nhóm General ở chuỗi cửa sổ), chọn nó, dời chuột về vị trí thích hợp trong form và vẽ nó với kích thước mong muốn. Hiệu chỉnh thuộc tính Text = "Bắt đầu thực hiện". Nếu cần, hãy thay đổi vị trí và kích thước của Button và Form.



Xây dựng ứng dụng dùng Owner-drawn control

6. Ấn kép chuột vào button vừa tạo để tạo hàm xử lý sự kiện Click của Button rồi viết code như sau :

//hàm xử lý Click chuột trên button

```
private void btnStart_Click (object sender, EventArgs e) {
```

```
    MessageBox.Show("Bạn vừa ấn chuột trên Button");
```

//thử thay đổi nội dung Caption

```
    btnStart.Text = "Kết thúc";
```

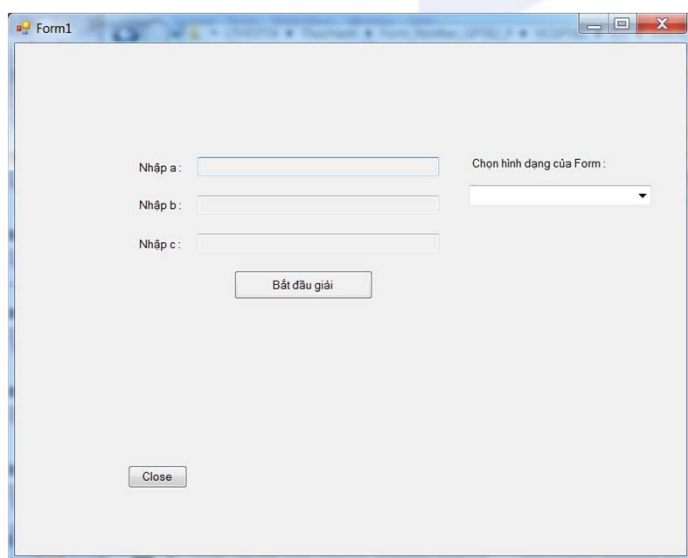
```
}
```

7. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hãy thử click chuột trên đối tượng HeartButton và đánh giá kết quả.



11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Các đối tượng giao diện, dù nhỏ hay lớn (Button, TextBox, ListBox, TreeView,...), đều được Windows quản lý giống nhau : Windows xử lý chúng như là window.
- ❑ Mỗi window sẽ được hiển thị ở dạng mặc định là hình chữ nhật có đường viền xung quanh và titlebar ở phía trên. Tuy nhiên ta có thể miêu tả lại hình dạng cho window theo nhu cầu riêng của mình.



11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Các thuộc tính sau đây sẽ xác định chính xác hình dạng của một window :
 - BackgroundImage : miêu tả hình bitmap được dùng để hiển thị nền window và để xác định hình dạng của window.
 - FormBorderStyle : miêu tả chế độ hiển thị các đường biên và titlebar của window.
 - Region : miêu tả vùng hiển thị và làm việc của window, nó gồm từ 1 tới nhiều vùng rời rạc, mỗi vùng rời rạc được bao đóng bởi 1 đường viền khép kín.



11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Đường viền khép kín của 1 vùng độc lập có thể được xác định bằng 1 trong 2 phương pháp :
 - Danh sách các đoạn thẳng hay cong liên tiếp và khép kín, mỗi đoạn thẳng hay cong có thể miêu tả bởi 1 hàm toán học như Line, Arc,....
 - Do hình bitmap nào đó xác định.
- ❑ Có 2 kỹ thuật xây dựng window có hình dạng bất kỳ :
 - Khai báo các thuộc tính liên quan 1 cách trực quan tại thời điểm thiết kế.
 - Lập trình động để thiết lập các giá trị phù hợp cho các thuộc tính liên quan đến window.



11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Qui trình xây dựng đối tượng giao diện có hình dạng bất kỳ bằng cách khai báo các thuộc tính liên quan 1 cách trực quan tại thời điểm thiết kế : Tạo form cần dùng, chọn nó để hiển thị cửa sổ thuộc tính, tìm và thiết lập giá trị cho các thuộc tính sau đây :
 - `BackgroundImage` : khai báo file bitmap được dùng để hiển thị nền window và để xác định hình dạng của window. Lưu ý hình bitmap cần có tính chất : các vùng diện tích của bitmap phải có màu khác với màu nền của hình bitmap; kích thước hình bitmap nên phù hợp với nhu cầu sử dụng của form tương ứng.
 - `FormBorderStyle = None`.
 - `TransparencyKey` : miêu tả màu RGB nền của hình bitmap cần lọc bỏ.



11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Thí dụ hãy xây dựng ứng dụng giải phương trình bậc 2 có hình dạng như sau :



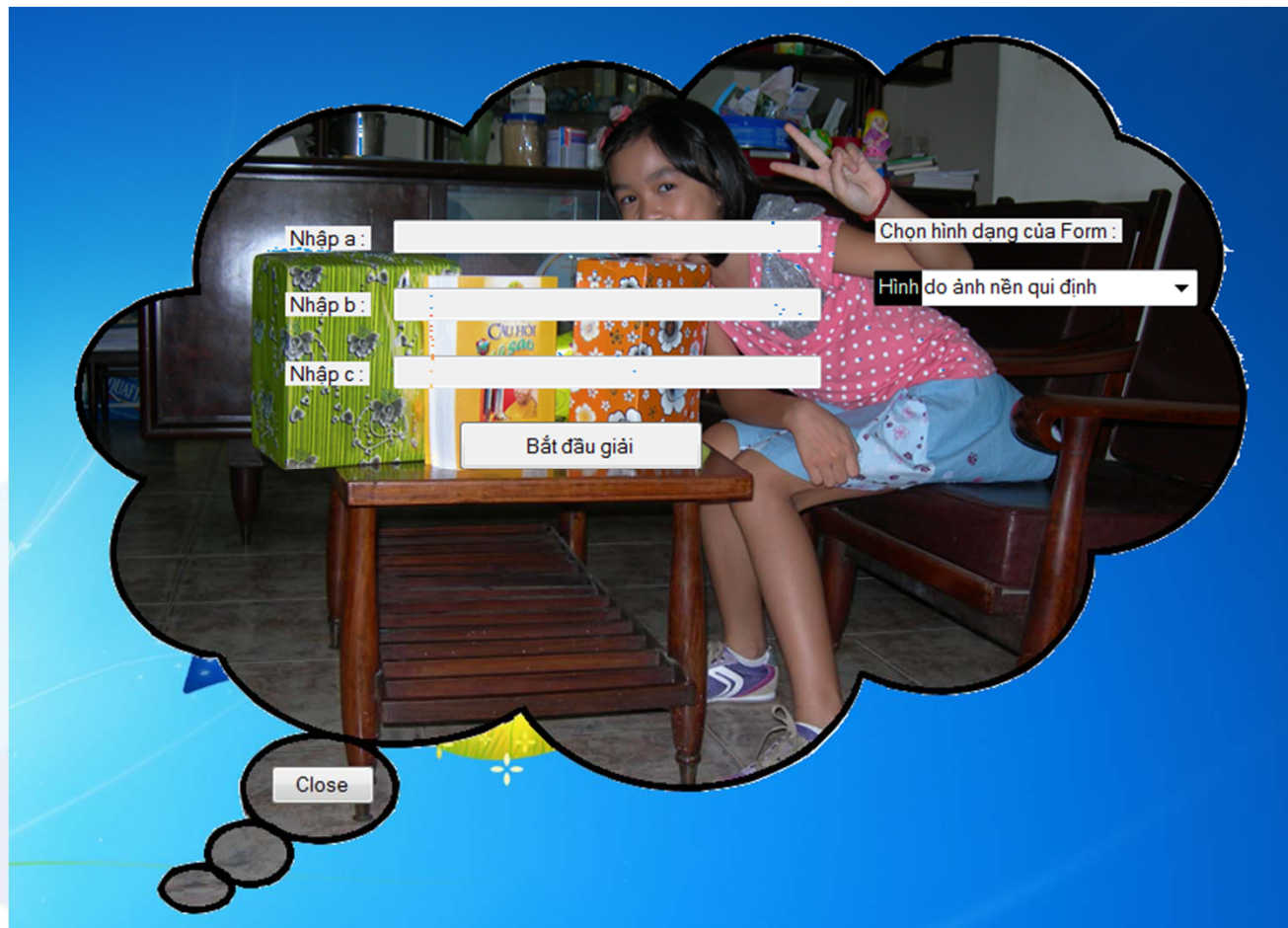
11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Qui trình xây dựng đối tượng giao diện có hình dạng bất kỳ bằng cách viết code thiết lập động các thuộc tính liên quan : Tạo form cần dùng, viết đoạn code thiết lập 3 thuộc tính liên quan khi cần thiết :
 - `BackgroundImage` : miêu tả hình bitmap được dùng để hiển thị nền window.
 - `FormBorderStyle = None`.
 - `Region` : miêu tả vùng diện tích làm việc của đối tượng.
- ❑ Thường `Region` được xác định thông qua đối tượng `Path`, đối tượng này miêu tả đường viền của `Region`.
- ❑ Để tạo `Path`, ta có thể dùng các hàm toán học miêu tả từng đoạn viền hay dùng đường viền của hình bitmap bất kỳ.



11.5 Xây dựng đối tượng giao diện có hình dạng tùy ý

- ❑ Thí dụ hãy xây dựng ứng dụng giải phương trình bậc 2 có hình dạng như sau :



11.6 Kết chương

- ❑ Chương này đã giới thiệu cách thức dùng tính thừa kế để tạo mới 3 loại đối tượng giao diện cá nhân hóa phổ biến là User Control, Inherited Control và Owner-drawn Control.
- ❑ Chương này cũng đã giới thiệu cách thức viết chương trình sử dụng lại các đối tượng giao diện cá nhân hóa.

