MỘT SỐ QUY ĐỊNH VIẾT CODE VÀ THIẾT KẾ GIAO DIỆN TRONG LẬP TRÌNH C#

Viết code là một công việc phức tạp. Để cho ra đời một đoạn code tốt đòi hỏi lập trình viên phải tốn khá nhiều thời gian và công sức. Hầu hết những lập trình viên mới vào nghề thường viết code theo kiểu "miễn sao chạy là được" do họ rất ít khi quan tâm đến chất lượng code. Tuy nhiên, đây là một thói quen xấu mà nếu không thay đổi ngay từ đầu thì sẽ rất khó sửa về sau. Hậu quả thường thấy từ những kiểu code nguy hại này là những mã xấu lảng vảng khắp ứng dụng, và nó có thể sẽ gây khá nhiều phiền toái sau này khi họ cần phải thay đổi và chỉnh sửa.

Mỗi lập trình viên đều có một phong cách code (Coding Style[1]) khác nhau, phong cách đó sẽ được dựa trên những nguyên tắc chung trong lập trình, chưa quan tâm bạn viết ngôn ngữ nào, nhưng ngôn ngữ bạn viết và phong cách bạn viết mang lại một cách nhìn cho người đọc code của bạn. Để tạo được một phong cách code hiệu quả, lập trình viên còn phải biết nhận biết được mã xấu (thường được gọi bằng cụm từ "code smells") và biết cách tùy chỉnh nó sao cho tối ưu nhất. Dưới đây sẽ lấy ngôn ngữ C# làm ngôn ngữ chủ đạo để thể hiện được quy tắc và phong cách viết code trong lập trình phần mềm.

1. QUY ĐỊNH ĐẶT TÊN

1.1. Các kiểu đặt tên

Kiểu	Ví dụ
Pascal	`BackColor`
Camel	`backColor`
Uppercase	`BACKCOLOR`

1.2. Một số quy tắc đặt tên

Loại	Kiểu đặt tên	Ví dụ	Ghi chú
Tên biến	Camel	`backColor`	
Hằng số	Uppercase	`NUMBER_OF_STUDENT`	Có gạch chân giữa các từ
Tên class, enum	Pascal	`SmartSnake`	
Tham số	Camel	`displayTime`	
Thuộc tính	Pascal	`BackColor`	
Phương thức	Pascal	`GetPath()`	
Sự kiện	Pascal	`ClickEventHandler`	Có hậu tố EventHandler
Giao diện (interface)	Pascal	`IButtonControl`	Có tiền tố I

1.3. Tiền tố của một số điều khiển

STT	Tên điều khiển	Tiền tố	Ví dụ

[1] **Coding Style** là các quy tắc trong quá trình viết code bao gồm về ngữ pháp (syntax) và ngữ nghĩa (sematic) như quy tắc đặt tên hàm biến, cách xuống dòng, comment, ...

1.	Panel	pnl	pnlGroup
2.	Check box	chk	chkReadOnly
3.	Combo box, drop-down list box	cbo	cboEnglish
4.	Command button	btn	btnExit
5.	Common dialog	dlg	dlgFileOpen
6.	Data	dat	datBiblio
7.	Data-bound combo box	cbo	cboLanguage
8.	Data-bound grid	grd	grdQueryResult
9.	Data-bound list box	lst	lstJobType
10.	Repeater	rpt	drpLocation
11.	DateTimePicker	dtp	dtpPublished
12.	Form	frm	frmEntry
13.	Frame	fra	fraLanguage
14.	DataGridView	dgv	dgvPrices
15.	GridView	grd	grdProduct
16.	DataList	dtl	dtlOrders
17.	Horizontal scroll bar	hsb	hsbVolume
18.	Image	img	imgIcon
19.	ImageList	ils	ilsAllIcons
20.	ImageButton	ibt	ibtNext
21.	HyperLink	hpl	hplHome
22.	LinkButton	lbt	lbtClick
23.	Label	lbl	lblHelpMessage
24.	List box	lst	lstPolicyCodes
25.	ListView	lvw	lvwHeadings
26.	Menu	mnu	mnuFileOpen
27.	Option button	opt	optGender
28.	Picture box	pic	picVGA
29.	Picture clip	clp	clpToolbar
30.	ProgressBar	prg	prgLoadFile
31.	RichTextBox	rtf	rtfReport
32.	Slider	sld	sldScale
33.	Spin	spn	spnPages
34.	StatusBar	sta	staDateTime
35.	TextBox	txt	txtLastName
36.	Timer	tmr	tmrAlarm
37.	Toolbar	tlb	tlbActions

38.	TreeView	tre	treOrganization
39.	UpDown	upd	updDirection
40.	Vertical scroll bar	vsb	vsbRate
41.	SqlDataSource	sql	sqlAccounts
42.	LinqDataSource	linq	linqCategories

2. THIẾT KẾ GIAO DIỆN

2.1. Thiết kế Form

Kích thước form:

Luôn cố gắng đảm bảo tỷ lệ 4 : 3. Form rộng 4 thì cao 3 để đảm bảo cân xứng với màn hình.

Kích thước control:

Chiều cao: sử dụng chiều cao mặc định sẵn của control.

Trường hợp đặc biệt:

TextBox multi-lines đảm bảo không bị che 1 phần của dòng.

Button có image đảm bảo hiển thị vừa đủ image 16×16 pixel. Độ rộng tùy độ rộng của text.

Đảm bảo nguyên tắc các text box, combo box, button trên cùng một form có độ rộng thống nhất, text trên button không nên vượt quá 2 từ. Đối với những trường có độ rộng cố định hoặc ít khi thay đổi (ví dụ như trường có kiểu dữ liệu là Date thì độ rộng là cố định là 10 ký tự), tuân thủ theo quy định sau:

Độ rộng control được binding với trường này chỉ được phép rộng đủ để hiện thị hết thông tin trong đó. Không để độ rộng control vượt quá độ rộng của trường. Lưu ý: Label đặt AutoSize = FALSE, TextBox đặt AutoSize = TRUE.

Font & Color:

Sử dụng thiết lập mặc định. Chỉ thay đổi khi yêu cầu thiết kế chỉ rõ.

Canh lề text trên control:

Chiều ngang (HAlign) chữ canh trái, số canh phải, riêng với button thì luôn canh giữa. Chiều dọc (VAlign) canh giữa.

Tab order:

Phải thiết lập tab order trên mọi giao diện (form, control, …) theo nguyên tắc từ trái sang phải, từ trên xuống dưới.Yêu cầu bắt buộc thiết lập tab order theo đúng thứ tự cho mọi control trên form, kể cả control không focus vào được như label, group box, hay control invisible. Lưu ý tuân thủ tuyệt đối quy định này vì nó phục vụ nhiều mục đích quan trọng như tạo shortcut key, valid required data, …

Anchor & Dock:

Phải thiết lập anchor và dock cho control trên các form, container không cố định kích thước (sizable).

Lưu ý:

Với thông tin yêu cầu người dùng không được bỏ trống mà bắt buộc nhập (AllowNull = FALSE) thì label cho thông tin đó phải sử dụng ký hiệu "(*)" ở cuối và thiết lập shortcut key.

2.2. Quy định tạo MessageBox

- ***Caption***: sử dụng Application.ProductName
- ***Icon***:
    + *MessageBoxIcon.Exclamation*: dùng cho các trường hợp cảnh báo lỗi, cảnh báo xóa dữ liệu, cảnh báo nhập thiếu, nhập sai dữ liệu.
    + *MessageBoxIcon.Information*: dùng cho các thông báo không có tính chất cảnh báo, ví dụ Kết quả import, thông tin về CSDL, …
    + *Message*: không được phép viết trực tiếp nội dung message mà phải dùng Resource (sử sụng hàm String.Format để truyền tham số cho Resource nếu cần).

3. QUY ĐỊNH VIẾT CODE

3.1. Quy định viết comment

Sử dụng tiếng Việt có dấu (Unicode) để viết comment.

- ***Comment cho module, class***: mỗi module, class cần có mô tả ngắn về mục đích của module hay class đó. Nội dung gồm:
    + *Mục đích*: module hay class thực hiện những công việc gì.
    + *Người lập*: Người tạo module hay class.
    + *Những biến/hàm quan trọng (không bắt buộc):* Liệt kê tên các biến và hàm quan trọng trong module/class.
- ***Comment cho method và event:*** tất cả các method và event phải có comment. Comment cho method/event gồm hai phần:
    + *Phần 1 (không bắt buộc):* mô tả mục đích và diễn giải ngắn gọn ý nghĩa các tham số đầu vào, đầu ra. Lưu ý: mô tả method đó làm gì (what), không mô tả method đó thực hiện thế nào (how). Lập trình viên có thể không cần viết phần mô tả mục đích này với các method/event đơn giản, không phức tạp.
    + *Phần 2 (bắt buộc):* ghi thông tin về lịch sử tạo và sửa method/event đó (người tạo/ngày tạo, người sửa/ngày sửa). Thông tin này bắt buộc phải có với mọi method/event. Mẫu comment cho method/event đơn giản:

```
//Created by Thiện An - 21/07/2014: Lấy DS HS theo lớp
//Modified by Văn Hoàng - 22/07/2014: Sửa câu truy vấn SQL
//Modified by .....
protected void LoadDanhSachLop(string maLop)
{
}
```

Mẫu comment cho method/event phức tạp:

```
/// <summary>
```

```
/// Lấy bảng điểm của 1 hs theo học kỳ
/// </summary>
/// <param name="maHS">mã học sinh</param>
/// <param name="hocKy">mã học kỳ</param>
/// <remarks>nhận xét (nếu có)</remarks>
/// Created by Thiện An - 21/07/2014: Lấy DS HS theo lớp
/// Modified by Văn Hoàng - 22/07/2014: Sửa câu truy vấn SQL
/// Modified by .....
protected void LoadBangDiem(string maHS, string hocKy)
{
}
```

- ***Comment cho đoạn code:*** những đoạn code phức tạp cần có comment gắn liền bên trên để chú giải. Những đoạn code được sửa đổi (modified), bổ sung (added) hoặc rem (removed) bởi người không phải tác giả cần có comment rõ ngay tại nơi sửa đổi, bổ sung: người sửa, ngày sửa, mục đích.

3.2. Quy định phân nhóm (region) khi coding

Phải sử dụng region phân nhóm code để tiện cho việc sửa đổi, bảo trì. Phân nhóm code theo cấu trúc như sau (theo thứ tự bắt buộc, nhưng không bắt buộc có đủ tất cả các region): Declaration, Constructor, Property, Method/Function, Event. Tùy theo yêu cầu của các form, class và module, lập trình viên có thể chia nhỏ các region chính trên thành các sub-region. VD: region Method/Function có thể chứa các region con sau: Method/Function Public, Overridable (trường hợp là base form/class), Override (trường hợp là derive form/class), Private, Other.

Trường hợp form hoặc class có sử dụng các component độc lập (Security, Document, MassEmail,…) thì phải tạo các region riêng cho từng component, chứa toàn bộ code liên quan đến việc tương tác với các component đó.

3.3. Quy định bẫy lỗi khi coding

Bắt buộc bẫy lỗi (sử dụng try … catch) trong tất cả các event của form và control trên form. Nghiêm cấm sử dụng cú pháp try … catch để che dấu lỗi (không xử lý gì sau từ khóa catch).

Nguồn internet.

MỤC LỤC