QUY CHUẨN CODE C# TRONG FSOFT (cơ bản)

C# coding convention (basic)

=======================================

QUY CHUẨN ĐẶT TÊN

1. CÁC KIỂU ĐẶT TÊN

PASCAL CASE: BackColor ( Dùng cho identifiers có >= 3 từ )

CAMEL CASE: backColor

UPPERCASE: IO, UI,...

Dùng cho identifiers có <= 2 kí tự (\* có ngoại lệ)

(\* xem bảng tham khảo ở file tiếng Anh, trang 7)

///////////////////////////////////////

2. VIẾT TẮT

- Ko dùng từ viết tắt để đặt tên cho identifiers

Vd: Dùng GetWindow thay cho GetWin

- Ko dùng acronyms (viết tắt kiểu VND, HCM, KHTN,...) ko thông dụng

- Dùng acronyms thay cho các tên quá dài: Dùng UI thay cho User Interface

- Dùng Pascal case hoặc Camel case cho acronyms > 2 kí tự; dùng Uppercase cho acronyms chỉ chứa 2 kí tự

Vd: Dùng HtmlButton hoặc htmlButton

Vd: Dùng System.IO thay cho System.Io

- Ko dùng từ viết tắt để đặt tên cho biến hoặc identifiers.

Nếu phải viết tắt, dùng Camel case cho các tên > 2 kí tự

///////////////////////////////////////

3. ĐẶT TÊN INTERFACE

- Dùng danh từ hoặc cụm danh từ

- Đặt tên có tiền tố là kí tự I

Vd: IProvider, IService

- Dùng Pascal case

- Có thể dùng tính từ mô tả: Vd: IPersistable

- Interface và lớp thực thi nó có tên tương đồng nhau

Vd: IProvider (interface) và Provider (lớp thực thi)

- Hạn chế dùng từ viết tắt

- Ko dùng dấu gạch dưới ( \_ )

///////////////////////////////////////

4. ĐẶT TÊN LỚP (CLASS)

- Dùng danh từ hoặc cụm danh từ

- Dùng Pascal case

- Hạn chế dùng từ viết tắt

- Ko dùng dấu gạch dưới ( \_ )

- Tên lớp con có đuôi là tên lớp cha

Vd: Lớp cha là Exception, lớp con là ApplicationException

Ko áp dụng cho mọi trường hợp. Vd: lớp cha là Control, lớp con là Button => OK

///////////////////////////////////////

5. ĐẶT TÊN PHƯƠNG THỨC

- Dùng động từ hoặc cụm động từ

- Dùng Pascal case

///////////////////////////////////////

6. ĐẶT TÊN PROPERTY

- Dùng danh từ hoặc cụm danh từ

- Dùng Pascal case

- Ko dùng Hungarian notation

- Nên đặt tên property trùng với tên kiểu dữ liệu của nó khi có thể

Vd: Nếu property tên Color thì kiểu dữ liệu của nó cũng nên là Color (enum)

///////////////////////////////////////

7. ĐẶT TÊN THAM SỐ

- Dùng Camel case

- Đặt tên có ý nghĩa và mang tính gợi nhớ

- Ko dùng tham số dành riêng

- Ko dùng Hungarian notation làm tiền tố

///////////////////////////////////////

8. ĐẶT TÊN BIẾN

- Dùng danh từ hoặc cụm danh từ

- Dùng Camel case

- Ko dùng tiền tố

- Ko dùng Hungarian notation cho các biến có kiểu dữ liệu cơ bản

- Dùng 'sqlCommand' cho biến có kiểu dữ liệu là SQLCommand, 'sqlConnection' cho SQLConnection, 'param' cho SQLParameter, 'sqlDataAdapter' hoặc 'da' cho SqlDataAdapter, 'dataset' hoặc 'ds' cho DataSet

- Dùng i, j, k để đặt tên cho biến đếm

(\* xem bảng tham khảo ở file tiếng Anh, trang 16)

///////////////////////////////////////

9. ĐẶT TÊN HẰNG (CONST)

- Dùng Uppercase, giữa các từ cách nhau bởi dấu gạch dưới

Vd: Const MIN\_QUAL = 25

10. ĐẶT TÊN CONTROL

<!-- Dùng cho winform / WPF -->

(\* xem bảng tham khảo ở file tiếng Anh, trang 18)

///////////////////////////////////////

11. ĐẶT TÊN SỰ KIỆN

<!-- Dùng cho winform / WPF -->

///////////////////////////////////////

=======================================

ĐỊNH DẠNG CODE

**1. COMMENTS**

COMMENTS NHIỀU DÒNG

- Dùng để mô tả file, phương thức, cấu trúc dữ liệu hoặc thuật toán

- Căn lề bằng với phần code cần comment

- Cách code 1 dòng trống

- Vd:

//

// All rights are reserved.

// Reproduction or transmission...

//

{ code }

COMMENTS MỘT DÒNG

COMMENTS Ở ĐUÔI

COMMENTS ĐỂ "XÓA MỀM" CODE

COMMENTS KIỂU XML

- Dùng cho class, delegate, interface, event, phương thức, properties, các trường (fields) ko rõ ràng

- Bắt đầu bằng dấu '///'

- Phải viết đúng cấu trúc XML

- Chỉ dùng tag có sẵn

- Tag <param> dùng để mô tả tham số

- Thuộc tính (attribute) 'cref' dùng để tham chiếu tới code, có thể được dùng với mọi tag

- Tag <summary> dùng để mô tả thông tin, mục đích của code

- Ko nằm trong thân phương thức hoặc hàm khởi tạo

- Vd: xem ở file tiếng Anh, trang 22

(\* xem bảng tham khảo ở file tiếng Anh, trang 24)

COMMENTS VỚI TODO, HACK, UNDONE MARKERS

- Dùng để tự động thêm link vào Task List. Click đôi vào comment bất kỳ trong Task List để nhảy tới chỗ của comment đó trong code

- Ko áp dụng cho HTML, CSS, XML

- Cú pháp: // TODO/HACK/UNDONE <nội dung>

- Vd: // TODO Fix this method

///////////////////////////////////////

**2. KHAI BÁO**

KHAI BÁO BIẾN

- Khai báo mỗi biến 1 dòng

- Nên khởi tạo biến ngay khi khai báo, chỉ trừ trường hợp cần xử lý code trước

- Khai báo biến ở đầu thân khối code

Vd:

public void SomeMethod()

{

int int1 = 0; // đầu thân khối định nghĩa phương thức

if (condition)

{

int int2 = 0; // đầu thân khối if

// do something

}

else

{

int int3 = 0; // đầu thân khối else

// do something

}

}

Ngoại lệ: vòng lặp for

for (int i = 0; i < maxLoops; i++)

{

// Do something

}

KHAI BÁO LỚP VÀ INTERFACE

- Theo định dạng sau:

public class Sample : Object

{

public Sample(int i, int j)

{

// do something

}

protected void EmptyMethod() {}

}

- Đối với phương thức rỗng:

protected void EmptyMethod() {}

- Mỗi phương thức cách nhau bởi **2** dòng trống

KHAI BÁO PROPERTIES

- Nếu thân get/set chỉ gồm 1 câu lệnh:

public int Foo

{

get { return this.foo; }

set { this.foo = value; }

}

- Nếu thân get/set gồm nhiều câu lệnh:

public int Foo

{

get

{

// do something

}

set

{

// do something

}

}

///////////////////////////////////////

**3. CÂU LỆNH**

CÂU LỆNH ĐƠN

- Mỗi dòng chứa 1 câu lệnh

Vd:

argv++; // OK

argc--; // OK

argv++; argc--; // sai

KHỐI CÂU LỆNH

- Định dạng như sau:

if (condition)

{

// do something

if (condition)

{

// do something

}

}

CÂU LỆNH return

- Ko dùng cặp dấu ( ), trừ trường hợp có thể làm rõ code hơn

Vd:

return; // OK

return myDisk.size(); // OK

return (size ? size : defaultsize); // OK

CÂU LỆNH if, if-else, if else-if else

- Định dạng như sau:

// câu lệnh if

if (condition)

{

// do something

}

// câu lệnh if-else

if (condition)

{

// do something

}

else

{

// do something

}

// câu lệnh if else-if else

if (condition)

{

// do something

}

else if (condition)

{

// do something

}

else

{

// do something

}

CÂU LỆNH for

- Định dạng như sau:

for (giá trị khởi đầu; điều kiện; update biến đếm)

{

// do something

}

CÂU LỆNH while

- Định dạng như sau:

while (condition)

{

// do something

}

- Đối với câu lệnh while rỗng:

while (condition);

CÂU LỆNH do-while

- Định dạng như sau:

do

{

// do something

} while (condition);

CÂU LỆNH switch

- Định dạng như sau:

switch (condition)

{

case 1:

// Falls through

case 2;

// do something

break;

case 3:

// do something

// break

default:

// do something

break;

}

- Đối với những case falls through (ko có câu lệnh break), chèn comment

- Mỗi câu lệnh switch nên có case default

CÂU LỆNH try-catch

- Định dạng như sau:

try

{

// do something

}

catch (Exception e)

{

// do something

}

finally

{

// do something

}

- Có thể ko có khối finally

BIỂU THỨC LAMBDA

- Chỉ dùng lambda inline khi biểu thức chỉ gồm 1 câu lệnh:

SomeMethod(name => { Trace.WriteLine(name); MessageBox.Show(name); }); // sai

SomeMethod(name => MessageBox.Show(name)); // OK

TỔ CHỨC CÂU LỆNH using

- Nhóm các namespace cùng framework, đặt third-party namespace dưới cùng

using System;

using System.Collections.Generic;

using MediServe.Patient;

using MediServe.Patient.DataAccess;

using ThirdPartyCompany.SomeComponent;

GÁN CỨNG

- Dùng hằng (constant) thay vì gán cứng

const int max = 5;

for (int i; i < max; i++)

{

// do something

}

=> OK

var list = new List<int>(5); // sai

const int age = 5;

var list = new List<int>(age) // OK

- Nên dùng tên (named values) thay cho chỉ số

foreach (DataRow row in patients.Tables[4].Rows)

{

// do something

}

=> sai

foreach (DataRow row in patients.Tables["Medication"].Rows)

{

// do something

}

=> OK

CODE CHẾT

- Luôn loại bỏ các dòng code thừa, ko dùng tới

///////////////////////////////////////

**4. KHOẢNG TRẮNG**

DÒNG TRỐNG

- Thêm dòng trống vào giữa:

+ Các biến cục bộ của phương thức và câu lệnh đầu tiên của phương thức đó

Vd:

public int SomeMethod()

{

int int1 = 0;

int int2 = 0;

// do something

}

+ Các đoạn xử lý logic khác nhau

+ Sau dấu } cuối

Vd:

{

// do something

{

// do something

}

}

// do something

=> OK

{

// do something

{

// do something

}

}

// do something

=> sai

{

// do something

{

// do something

}

}

// do something

=> sai

KHOẢNG TRẮNG

- Giữa keyword và cặp dấu ( ) liền sau

Vd:

while (true) // while là keyword

{

// do something

}

\* Trừ trường hợp tên phương thức

- Sau dấu phẩy trong danh sách tham số

Vd:

public void PrintList(IEnumerable<int> list, int count) {}

- Giữa toán tử binary (toán tử gồm 2 toán hạng) và toán hạng của nó

Vd:

a += c;

// += là toán tử binary

// a, c là toán hạng

a = c + d;

//

// =, + là toán tử binary

// a, c, d là toán hạng

- Ko có khoảng trống của toán tử unary (toán tử gồm 1 toán hạng) và toán hạng của họ

Vd:

n++; // OK

c--; // OK

x ++; // sai