**Chuẩn lập trình trong C#**

**1. Mục đích:**

* Đưa ra các quy ước khi coding với ngôn ngữ lập trình C#, với các quy tắc này giúp tiết kiệm thời gian rất lớn trong tiến trình phát triển phần mềm và cả trong quá trình bảo trì sản phẩm. Giúp sinh viên quen với làm việc theo nhóm.
* Tài liệu này chủ yếu hướng dẫn sinh viên với ngôn ngữ lập trình C#, nhưng có rất nhiều quy tắc được sử dụng trong nhiều ngôn ngữ lập trình khác tích hợp trong bộ công cụ Visual Studio .NET.

**2. Phạm vi áp dụng:**

Những lập trình viên tham gia dự án phát triển bằng ngôn ngữ C# và công cụ Visual Studio .Net.

**3. Định nghĩa và những từ viết tắt:**

Ký pháp Hungarian là cách quy định đặt tên biến như sau: bắt đầu tên biến thì viết chữ thường và các chữ đầu thể hiện kiểu dữ liệu của biến, và được gọi là các tiền tố

**4. Tài liệu liên quan**

* [**https://msdn.microsoft.com/en-us/library/ff926074.aspx**](https://msdn.microsoft.com/en-us/library/ff926074.aspx)
* [**http://www.dofactory.com/reference/csharp-coding-standards**](http://www.dofactory.com/reference/csharp-coding-standards)
* [**http://se.inf.ethz.ch/old/teaching/ss2007/251-0290-00/project/CSharpCodingStandards.pdf**](http://se.inf.ethz.ch/old/teaching/ss2007/251-0290-00/project/CSharpCodingStandards.pdf)

**5. Các nội dung quy định:**

***5.1. Các kiểu quy ước viết hoa:***

Có 3 quy tắc viết hoa:

**Pascal Case**

Chữ cái đầu tiên trong từ định danh và chữ cái đầu tiên của mỗi từ nối theo sau phải được viết hoa. Sử dụng Pascal Case để đặt tên cho một tên có từ 3 ký tự trở lên.

Ví dụ: **BackColor**

**Camel Case**

Chữ cái đầu tiên trong từ định danh là chữ thường và chữ cái đầu tiên của mối từ nối theo sau phải được viết hoa.

Ví dụ: **backColor**

**Uppercase**

Tất cả các ký tự trong từ định danh phải được viết hoa. Sử dụng quy tắc này đối với tên định danh có từ 2 ký tự trở xuống.

Ví dụ:

**System.IO**

[**System.Web.IO**](http://system.web.io/)

Bảng tóm tắt các quy tắc:

|  |  |  |
| --- | --- | --- |
| ***Kiểu dữ liệu*** | ***Kiểu quy ước*** | ***Chú ý*** |
| Interface | Pascal Casing | Dùng tiền tố **I** |
| Enum | Pascal Casing |  |
| Events | Pascal Casing |  |
| Exception | Pascal Casing | Kết thúc với hậu tố **Exception** |
| Public Fields | Pascal Casing |  |
| Methods | Pascal Casing |  |
| Namespace | Pascal Casing |  |
| Property | Pascal Casing |  |
| Protected/private Fields | Camel Casing |  |
| Parameters | Camel Casing |  |

***5.2. Một số lưu ý trong cách đặt tên cho các thành phần***

* Không dùng các tên giống nhau(chỉ phân biệt kiểu chữ in hoa hay thường) Ta khó nhận ra các định danh nhất là khi trong cùng ngữ cảnh và chỉ phân biệt các định danh bằng kiểu chữ in hoa/thường.
* Không tạo 2 namespace cùng tên và chỉ khác nhau ở kiểu chữ viết(chữ hoa/Chữ thường), ví dụ:

**Namespace IridiumSoftware**

**Namespace iridiumsoftware**

* Không nên xây dựng 1 method với các tham số có cùng tên và chỉ khác nhau kiểu chữ, ví dụ:

**void MyFunction(string a, string A)**

* Không xây dựng 1 kiểu với các tên property giống nhau và chỉ phân biệt ở kiểu chữ, ví dụ:

**int Color {get, set}**

**int COLOR {get, set}**

* Không đặt tên các phương thức có cùng tên và chỉ khác nhau ở kiểu chữ, ví dụ:

**void calculate()**

**void Calculate()**

* Không dùng từ viết tắt hoặc 1 phần của các tên định danh như: **GetWindow** viết thành **GetWin** là ko hợp quy ước.
* Ta có thể sử dụng kí tự đầu tiên của các tên có một nhóm từ để đặt tên cho nó.

Ví dụ: **UI** thay cho User Interface.

**OLAP** thay cho On-Line Analytical Processing

* Không sử dụng các từ viết tắt đã được thừa nhận từ các lĩnh vực tin học khác Ví dụ: **XML**, **TTL**, **DNS**, **UI**, **IP** và **IO** …)
* Khi đặt tên viết tắt, bạn có thể sử dụng cách viết Pascal case hay Camel case để đặt tên, như trường hợp này bạn có thể dùng tên nào cũng được **HtmlButton** hoặc **HTMLButton**. Tuy nhiên với trường hợp tên có 2 ký tự hoặc ít hơn thì bạn nên viết như **IO** thay cho **System.Io**.
* Không dùng từ viết tắt trong tên định danh hoặc tên tham số. Nếu bạn cần thiết phải dùng đến từ viết tắt, dùng cách viết Camel Case cho từ viết tắt có từ 2 kí tự trở lên thậm chí là nó phủ định lại chuẩn viết tắt của 1 từ.
* Tránh sử dùng lại các từ khóa hoặc tên các class chuẩn được dùng trong .NET Framework namespaces.

***5.3. Cách tổ chức file trong 1 project:***

***5.3.1 File sourcecode C#***

Giữ file nguồn/ các class không quá dài(không được vượt quá 2000 LOC (Lines of Code), phân chia code thành những đơn vị code nhỏ tạo nên các cấu trúc code sáng sủa hơn. Đặt mỗi class vào mỗi file source riêng biệt và tên file cũng là tên class(với phần đuôi mở rộng .cs)

Thứ tự trong file source C#:

* các lệnh **using**
* lệnh **namespace**
* Các khai báo **Class** and **interface**

Các lệnh Namespace và using:

Các dòng không phải là dòng comment ở hầu hết các file source của C# là các lệnh using, kế đến là lệnh namespace như ví dụ sau:

**using System.Data;**

**namespace Business.Framework;**

Cả 2 lệnh này đề được canh sát lề trái.

Các khai báo class và interface:

|  |  |  |
| --- | --- | --- |
| ***Thứ tự*** | ***Các phần trong khai báo class và interface*** | ***Ghi chú*** |
| 1 | Class/interface documentation | **/// <summary>**  **/// The Person class provides …**  **/// </summary>**  **public class Person** |
| 2 | class or interface |  |
| 3 | Fields | Trình tự khai báo các fields theo mức độ cho phép truy cập:  *Private*  *Protected*  *Internal*  *public* |
| 4 | Properties | Trình tự khai báo các Properties theo mức độ cho phép truy cập: *Private*  *Protected*  *Internal*  *public* |
| 5 | Constructors | Trình tự khai báo các Constructors theo mức độ cho phép truy cập: *Private*  *Protected*  *Internal*  *Public*  Trước tiên là constructor mặc định, sau đó là các constructor theo thứ tự độ phức tạp tăng dần |
| 6 | Methods | Các Method nên được nhóm theo chức năng đúng hơn là nhóm theo mức độ truy cập. |

***5.3.2 Bố trí thư mục sourcecode C#***

Tạo 1 thứ mục cho namspae cho mỗi project. (Với namespace MyProject.TestSuite.TestTier ta sẽ có 1 đường dẫn thư mục tương ứng MyProject/

TestSuite/TestTier). Và ta dễ dàng ánh xạ các namespace sang cấu trúc thư mục trong Project Explorer.

***5.4. Quy định về đặt tên biến***

***5.4.1 Hướng dẫn đặt tên Namespace***

Quy tắc chung cho việc đặt tên Namespace là dùng tên của công ty theo sau là tên công nghệ và kế đến là tùy chọn tên đặc trưng hay thiết kế:

**CompanyName.TechnologyName[.Feature][.Design]**

Ví dụ:

**IridiumSoftware.IridiumX**

**IridiumSoftware.IridiumX.Design**

**Microsoft.Office**

**5.4.2 *Hướng dẫn đặt tên*class**

* Thường dùng danh từ hoặc một cụm danh từ để đặt tên cho 1 class.
* Kí tự đầu tiên của mỗi từ là chữ in hoa.(Dùng quy tắc Pascal Case).
* Không sử dụng dạng tiền tố:

Ví dụ: **FileStream** à Tên chuẩn.

Không đặt tên theo dạng **CfileStream** hay **ClassFileStream**

* Trong tên class không sử dụng dấu gạch chân (\_).
* Trong trường hợp cần thiết, nếu kí tự đâu tiên của tên 1 class bắt đầu bằng kí tự **‘I’**và kí tự này là kí tự bắt đầu của một từ và từ đó là 1 phần của tên class thì tên đó vẫn được chấp nhận.

Ví dụ: **IdentityStore**

* Sử dụng một từ ghép để đặt tên cho 1 class dẫn xuất, từ thứ 2 trong tên của class dẫn xuất nên lấy tên của class cơ sở.

Ví dụ: **ApplicationException** là tên class dẫn xuất từ class cơ sở **Exception**, và với cách đặt tên này ta có thể hiểu rằng **ApplicationException** là một trong các loại **Exception**.

Với quy tắc này thì tùy trường hợp mà chúng ta có thể áp dụng để tránh đặt tên class dài lê thê với những từ không cần thiết.

Ví dụ như trường hợp sau:

Class **Button** được dẫn xuất từ class **Control** nhưng nếu ta đặt tên theo quy tắc trên thì tên class này là **ButtonControl**. Tên class này trở nên dài ra và không cần thiết vì mặt định thì button được xem như là một control rồi.

Dưới đây là cách đặt tên class chuẩn:

**public class FileStream**

**public class Button**

**public class String**

**5.4.3 *Hướng dẫn đặt tên*Interface**

* Thường dùng danh từ hoặc một cụm danh từ, hay một tính từ mà nó mô tả hành vi để đặt tên cho 1 interface.

Ví dụ:

Tên interface **Icomponent**sử dụng 1 danh từ.

**ICustomAttributeProvider**sử dụng 1 cụm danh từ.

**IPersistable**sử dụng một tính từ.

* Sử dụng quy tắc PacalCase.
* Tên đặt cho interface phải ngắn gọn.
* Khi định nghĩa 1 cặp class/interface mà class này là 1 thực thi chuẩn của interface thì tên giữa class và interface nên đặt tương tự nhau. Và điểm khác nhau là tên interface có kí tự tiền tố “I”.
* Không sử dụng kí tự gạch chân “\_” trong tên interface.

Dưới đây là các ví dụ đặt tên interface chuẩn:

**public interface IServiceProvider**

**public interface IFormatable**

Dưới đây là đoạn code minh họa cách định nghĩa interface, một class thực thi chuẩn của interface:

**public interface IComponent**

**{**

**// Implementation goes here.**

**}**

**public class Component : IComponent**

**{**

**// Implementation goes here.**

**}**

**5.4.4 *Hướng dẫn đặt tên*Attribute**

Bạn nên thêm vào hậu tố Attribute đối với các lớp attribute.

**public class ObsoleteAttribute{}**

**5.4.5 *Hướng dẫn đặt tên*Enumeration**

Kiểu giá trị Enum thừa kế từ Enum class. Sau đây là các quy tắc hướng dẫn đặt tên cho enumerations:

* Sử dụng Pascal Case cho các tên kiểu và giá trị enum.
* Sử dụng tên ngắn gọn và tường minh.
* Không sử dụng hậu tố Enum trong tên kiểu Enum.

**5.4.6 *Hướng dẫn đặt tên*StaticField**

Quy tắc đặt tên cho các static field:

* Dùng các danh từ, cụm danh từ hay các danh từ viết tắt để đặt tên cho static fields.
* Dùng Pascal Case.
* Sử dụng tiền tố ký pháp Hungarian trong tên của static field.
* Người ta khuyên dùng thuộc tính static để thay thế cho các static fields có cấp độ chia sẻ dạng public.

**5.4.7 *Hướng dẫn đặt tên*Parameter**

* Tên Parameter phải được mô tả một cách đầy đủ và nó bao hàm cả tên của tham số và kiểu dữ liệu của nó
* Dùng Camel Case.
* Tên tham số nên đặt theo hướng mô tả ý nghĩa của tham số tốt hơn là đặt theo kiểu dữ liệu của tham sô. Vì môi trường lập trình đã cung cấp các thông tin về kiểu dữ liệu tham số.
* Không nên dùng các tham số để dành , mà khi cần thiết ta có thể thêm vào trong các version sau này của các thư viện class.
* Không sử dụng các tên tham số tiền tố với kí pháp Hungarian.

Các tên tham số chuẩn:

**Type GetType(string typeName)**

**string Format(string format, object[] args)**

**5.4.8 *Hướng dẫn đặt tên*Method**

* Dùng các động từ hay các cụm động từ để đặt tên cho methods.
* Dùng Pascal Case.

Ví dụ tên các methods chuẩn:

**RemoveAll()**

**GetCharArray()**

**Invoke()**

**5.4.9 *Hướng dẫn đặt tên*Property**

* Dùng 1 danh từ hay 1 cụm danh từ để đặt tên cho 1 property.
* Dùng Pascal Case.
* Không dùng ký pháp Hungarian.
* Tạo 1 property và đặt tên tương tự như kiểu dữ liệu của nó.

Ví dụ minh họa cho tên một property:

**public class SampleClass**

**{**

**public Color BackColor**

**{**

**// Code for Get and Set accessors goes here.**

**}**

**}**

**5.4.10 *Hướng dẫn đặt tên*Event**

* Dùng 1 hậu tố EventHandler trong các tên của event handler.
* Chỉ định rõ 2 tham số có tên là *sender*và *e*. Tham sô *sender*mô tả object gọi event. Tham sô *sender*luôn luôn có kiểu dữ liệu là object. Trạng thái kết hợp với sự kiện được đóng gói trong một thể hiện của event class có tên là *e*. Kiểu của tham số e phải là một lớp sự kiện rõ ràng và thích hợp.
* Đặt tên 1 lớp đối số sự kiện với hậu tố EventArgs.
* Nên đặt tên 1 sự kiện là 1 động từ.
* Dùng danh động từ(dạng thêm “ing” của động từ) để đặt tên cho một sự cho một sự kiện mà nó thể hiện khái niệm pre\_event và quá khứ của động từ để thể hiện sự kiện post\_event.

Ví dụ 1 sự kiện ‘Close’ có thể được hủy bỏ vì thể nên có thêm sự kiện ‘Closing’ và ‘Closed’. Không dùng mẫu đặt tên sự kiện dạng BeforeXX/AfterXXX.

* Không dùng tiền tố và hậu tố trong khai báo event có dạng “OnXXX”. Ví dụ: dùng Close thay vì dùng OnClose.
* Tổng quát hơn, bạn nên cung cấp 1 method có mức truy cập protected gọi sự kiện có dạng “OnXXX” mà sự kiện này có thể bị ghi đè trong 1 class dẫn xuất. Method này chỉ nên có tham số sự kiện e.

Ví dụ minh họa 1 event handler với 1 tên chuẩn và các tham số:

**public delegate void MouseEventHandler(object sender, MouseEventArgs e);**

**5.4.11 *Hướng dẫn đặt tên*Control**

Bảng danh sách các tiền tố của các kiểu controls thông dụng:

|  |  |
| --- | --- |
| ***Tiền tố (prefix)*** | ***Loại Control*** |
| lbl | Label |
| llbl | LinkLabel |
| btn | Button |
| txt | Textbox |
| mnu | MainMenu |
| chk | CheckBox |
| rdo | RadioButton |
| grp | GroupBox |
| pic | PictureBox |
| grd | Grid |
| lst | ListBox |
| cbo | ComboBox |
| lstv | ListView |
| tre | TreeView |
| tab | TabControl |
| dtm | DateTimePicker |
| mon | MonthCalendar |
| sbr | ScrollBar |
| tmr | Timer |
| spl | Splitter |
| dud | DomainUpDown |
| nud | NumericUpDown |
| trk | TrackBar |
| pro | ProgressBar |
| rtxt | RichTextBox |
| img | ImageList |
| hlp | HelpProvider |
| tip | ToolTip |
| cmnu | ContextMenu |
| tbr | ToolBar |
| frm | Form |
| bar | StatusBar |
| nico | NotifyIcon |
| ofd | OpenFileDialog |
| sdl | SaveFileDialog |
| fd | FontDialog |
| cd | ColorDialog |
| pd | PrintDialog |
| ppd | PrintPreviewDialog |
| ppc | PrintPreviewControl |
| err | ErrorProvider |
| pdoc | PrintDocument |
| psd | PageSetupDialog |
| crv | CrystalReportDialog |
| pd | PrintDialog |
| sw | FileSystemWatcher |
| log | EventLog |
| dire | DirectoryEntry |
| dirs | DirectorySearcher |
| msq | MessageQueue |
| pco | PerformanceCounter |
| pro | Process |
| ser | ServiceController |
| rpt | ReportDocument |
| ds | DataSet |
| olea | OleDbDataAdapter |
| olec | OleDbConnection |
| oled | OleDbCommand |
| sqla | SqlDbDataAdapter |
| sqlc | SqlDbConnection |
| sqld | SqlDbCommand |
| dvw | DataView |

Menu Controls

Menu controls nên đặt tên bằng cách dùng tag “mnu” theo sau là đường dẫn xuống đầy đủ theo cây menu. Ví dụ về tên 1 biến control menu:

**mnuFile**

**mnuFileNew**

**mnuEdit**

**mnuEditCopy**

**mnuInsertIndexAndTables**

**mnuTableCellHeightAndWidth**

***5.4.12 Hướng dẫn đặt tên Data***

Nếu cần dùng tên **CustomerCode** và hiển thị lên 1 textbox, thì textbox phải đặt tên là **txtCustomerCode**.

Nếu dùng combobox để hiển thị các mã số khách hàng, thì control đó đặt tên là **cboCustomerCode**.

Nếu muốn lưu mã khách hàng vào biến của 1 module thì nó được đặt tên là **mCustomerCode**.

Fields in Databases:

Với các quy tắc tổng quát, chúng ta có thể dùng Property Naming Guidelines khi đặt tên cho các field trong database.