Naming Conventions for .NET / C# Projects

Martin Zahn, Akadia AG, 20.03.2003

The original of this document was developed by the Microsoft special interest group. We made some addons.

This document explains the naming conventions that should be used with .NET projects.

A consistent naming pattern is one of the most important elements of predictability and discoverability in a managed class library. Widespread use and understanding of these naming guidelines should eliminate unclear code and make it easier for developers to understand shared code.

Capitalization Styles Defined

We define three types of capitalization styles:

Pascal case

The first letter in the identifier and the first letter of each subsequent concatenated word are capitalized.

Example:

```
BackColor, DataSet
```

Camel case

The first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalized.

Example:

```
numberOfDays, isValid
```

Uppercase

All letters in the identifier are capitalized.

Example:

```
ID, PI
```

Hungarian Type Notation Defined

Hungarian notation is any of a variety of standards for organizing a computer program by selecting a schema for naming your variables so that their type is readily available to someone familiar with the notation. It is in fact a commenting technique.

Example:

```
strFirstName, iNumberOfDays
```

There are different opinions about using this kind of type notation in programming nowadays. Some say that it's useful, and it should be used everywhere to enhance clarity of your code. Others say it just obfuscates your code, because it has no real advantage in modern programming environments.

Our point of view is a moderated one: use it wisely, meaning, we only use Hungarian notation for **private or local variables**, that are only accessible and interesting to the programmer of the class.

Don't use it with public variables, properties or parameters in methods, because they are exposed to the outside world. Someone who uses your classes and accesses properties of your class, is not interested in type, but just wants to use them.

In the .NET framework, there are a lot of types, so we extended and adapted the Hungarian notation with our own type notation.

Naming Guidelines

1). Private Variables (Fields in C#) Naming Guidelines

Naming guidelines

Prefix private variables with a "_" and Hungarian-style notation.

Case guidelines

Use camel case as a general rule, or uppercase for very small words

Example:

```
_strFirstName, _dsetEmployees
```

```
// Field
private OleDbConnection _connection;

// Property
public OleDbConnection Connection
{
  get { return _connection; }
  set { _connection = value; }
}
```

2). Local Variables Naming Guidelines

Naming guidelines

Prefix private or local variables with Hungarian-style notation.

Case guidelines

Use camel case as a general rule, or uppercase for very small words

Example:

```
strFirstName, dsetEmployees
```

3). Namespace Naming Guidelines

Naming guidelines

The general rule for naming namespaces is to use the company name followed by the technology name and optionally the feature and design as follows:

```
CompanyName.TechnologyName[.Feature][.Design]
```

Prefixing namespace names with a company name or other well-established brand avoids the possibility of two published namespaces having the same name. Use a stable, recognized technology name at the second level of a hierarchical name.

Example:

```
Akadia.Traffic, System.Web.UI, System.Windows.Forms
```

Case guidelines

Use Pascal case as a general rule, or uppercase for very small words.

Example:

```
System.Windows.Forms, System.Web.UI
```

4). Class Naming Guidelines

Naming guidelines

Use a noun or noun phrase to name a class.
Do not use a type prefix, such as C for class, on a class name.
Do not use the underscore character (_).

Case guidelines

Use Pascal case. Example:

```
FileStream, Button
```

5). Interface Naming Guidelines

Naming guidelines

Prefix interface names with the letter "I", to indicate that the type is an interface.
Do not use the underscore character (_).

Case guidelines

Use Pascal case. Example:

```
IServiceProvider, IFormatable
```

6). Parameter Naming Guidelines

Naming guidelines

Use descriptive parameter names. Parameter names should be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios. To distinguish parameters from other variables the prefix **"p"** should be used.

Do not prefix parameter names with Hungarian type notation.

Do not use a prefix for parameter names of an event handler and exceptions.

Case guidelines

Use camel case. Example:

```
pTypeName, pNumberOfItems
```

7). Method Naming Guidelines

Naming guidelines

Use verbs or verb phrases to name methods.

Case guidelines

Use Pascal case. Example:

```
RemoveAll(), GetCharAt()
```

8). Property / Enumerations Naming Guidelines

Naming guidelines

Use a noun or noun phrase to name properties.
Do not use Hungarian notation.

Case guidelines

Use Pascal case. Example:

```
BackColor, NumberOfItems
```

9). Event Naming Guidelines

Naming guidelines

Use an EventHandler suffix on event handler names.

Specify two parameters named sender and e. The sender parameter represents the object that raised the event. The sender parameter is always of type object, even if it is possible to use a more specific type. The state associated with the event is encapsulated in an instance of an event class named **"e"**. Use an appropriate and specific event class for the e parameter type.

Name an event argument class with the **EventArgs** suffix.

Case guidelines

Use Pascal case. Example:

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

9). Exception Naming Guidelines

Naming guidelines

Event handlers in Visual Studio .NET tend to use an "e" parameter for the event parameter to the call. To ensure we avoid a conflict, we will use "**ex**" as a standard variable name for an Exception object.

Example

```
catch (Exception ex)
{
  // Handle Exception
}
```

10). Constant Naming Guidelines

The names of variables declared class constants should be all uppercase with words separated by underscores. It is recommended to use a grouping naming schema.

Example (for group AP_WIN):

AP_WIN_MIN_WIDTH, AP_WIN_MAX_WIDTH, AP_WIN_MIN_HIGHT, AP_WIN_MAX_HIGHT

11). C# Primitive Type Notation

sbyte	sy
short	s
int	i
long	l
byte	y
ushort	us
uint	ui
ulong	ul
float	f
double	d
decimal	dec
bool	b
char	c

12). Visual Control Type Notation

Assembly	asm
Boolean	bln
Button	btn
Char	ch
CheckBox	cbx
ComboBox	cmb
Container	ctr
DataColumn	dcol
DataGrid	dgrid
DataGridDateTimePickerColumn	dgdtpc
DataGridTableStyle	dgts
DataGridTextBoxColumn	dgtbc
DataReader	dreader
DataRow	drow
DataSet	dset
DataTable	dtable
DateTime	date
Dialog	dialog
DialogResult	dr
Double	dbl
Exception	ex
GroupBox	gbx
HashTable	htbl
ImageList	iml
Integer	int
Label	lbl
ListBox	lbx
ListView	lv
MarshallByRefObject	rmt
Mainmenu	mm
MenuItem	mi

MDI-Frame	frame
MDI-Sheet	sheet
NumericUpDown	nud
Panel	pnl
PictureBox	pbx
RadioButton	rbtn
SDI-Form	form
SqlCommand	sqlcom
SqlCommandBuilder	sqlcomb
SqlConnection	sqlcon
SqlDataAdapter	sqlda
StatusBar	stb
String	str
StringBuilder	strb
TabControl	tabctrl
TabPage	tabpage
TextBox	tbx
ToolBar	tbr
ToolBarButton	tbb
Timer	tmr
UserControl	usr
WindowsPrincipal	wpl