What naming Convention to Use for C# Function Parameters

Asked 8 years, 11 months ago Active 3 years, 5 months ago Viewed 19k times

14

There are situations when a name passed in Parameter will be Cast into a new type, but the name of the Passed object should remain similar. For the case of Class Attributes, we can use this operator, but what about for local variable in functions. What coding convention is widely used.

example,

```
void MyFunc(BaseClass myPara)
{
  DerivedClass _mypara = (BaseClass)myPara;
}
```

or on the contrary

```
void MyFunc(BaseClass _myPara)
{
  DerivedClass mypara = (BaseClass)_myPara;
```

or any other conventionl

c# naming

asked Apr 25 '11 at 9:23

Shamim Hafiz
3,947 7 33 44

1 Whatever other answers you get below, there's a little tool to analyze and enforce stylistic rules: archive.msdn.microsoft.com/sourceanalysis – Patrick Hughes Jul 13 '11 at 8:36

6 Answers

Active | Oldest | Votes

11

Prefixing either parameters or local variables with an underscore is not very idiomatic in C#, it is not very easy to read and not often used (although it is legal, so you are free to so if you wish).

The best name for the parameter and the variable is a descriptive name. You need to think why you are changing the type, what is the reason for the cast. Then you should be able to come up with 2 different names. E.g. is you passed a "person" and converted it to a "customer" then you could use person and / or customer in the variable names perhaps.

If you really can't think of 2 different names then I would use "as" in the name (there was a question on this site a few days ago about this). E.g. you would use "myParaAsDerived" for the local variable.

If at all possible I would not use this, I would think hard about the problem you are solving and what meaningful names could be used, but if all else fails this is fairly readable.

edited Apr 12 '17 at 7:31

Community ♦
1

answered Apr 25 '11 at 9:30

Steve
5,154 1 18 26

Just a double-check (I'm not that familiar with C#). The leading underscore really is "properly" legal in C#? In C and C++, identifiers with leading (or doubled) underscores are reserved, so although they are legal in a sense, you shouldn't define your own identifiers like that. csharp.comsci.us/etymology/identifiers.html suggests C# may be similar (see bottom, last of "limitations") but doesn't actually say "reserved". – Steve314 Jul 13 '11 at 8:38

9

Firstly using

```
void MyFunc(BaseClass _myPara)
{
}
```

Is clearly wrong! As a lot of c# coding standards use a "_" prefix on all **field names**! You code need to be easy to understand by other programmer so code should not be written in a way that will mislead a lot of C# programmers.

Given all the benefits of small methods, I *personally* don't see any need for a naming convention to separate local variables from parameters. If you methods have so many parameters and local variables that you can't tell what is going on without a naming convention you have bigger problems. (This is well covered in the Clean Code Book, a Java book but I still found it of great benefit as a C# programmer)

edited Dec 5 '14 at 15:27

answered Jul 13 '11 at 8:21

Ian

4,250 14 26

4

If you do want to prefix them with something then you should use `p_` for parameter: in general I guess you would probably annoy a lot of people if you did this. BUT be consistent, don't just do it in one place just because you need two different names for variables that you want to give the same name.

A good general rule with variable naming goes like;

- If you only have one type of object name it by its function:

```
var builder = new PizzaBuilder();
```

- If you have more than one name them by their function and specialism:

```
var pizzaBuilder = new PizzaBuilder();
var milkShakeBuilder = new MilkShakeBuilder();
```

answered Apr 25 '11 at 13:27

user23157

The p_ (or just p) for parameter is an old convention that has been used a lot in C++ and C. It tends to go with l_ for local and (in C++) m_ for member-variable. I've seen it in Pascal, Modula 2 and Ada too, so it's not just a C-family thing. It *is* kinda love-it-or-hate-it, though. I've used it almost obsessively, my excuse being Steve Haighs reasoning for "As". E.g. setter methods often do `m_Whatever = p_Whatever;` - giving the two identifiers meaningfully different names would be awkward. But I've started to question if those cases are common enough to justify the consistent convention. – Steve314 Jul 13 '11 at 8:51

4

C# naming conventions will have you:

- Using PascalCasing for methods, public properties and class names
- Using IPascalCasing (notice the I at the start) for interface names
- Using camelCasing for method parameters and local variables
- Using _underscoredCamelCasing for class wide private fields

And please stay away from hungarian notation. It's pointless and doesn't adhere to C# conventions.

answered Jul 13 '11 at 8:20

Matteo Mosca

564 2 14

private fields are pascal-cased if they are static. – sara Jul 15 '16 at 14:43

2

Underscoring in variable naming can be kinda unnecesary as we have the "this" keyword to reference to class level variables specifically. If you would like to learn more about variable naming conventions from the experts I would suggest you take a look at the infamous paper called "Ottinger's Rules for Variable and Class Naming" by Tim Ottinger, an article supported by clean coding mentor Robert C. Martin

```
public void Function(string p_Parameter1, string p_Parameter2)
```

...would be more readable like...

```
public void Function(string parameter1, string parameter2)
```

...where parameter1 and 2 are descriptive names for the corresponding variables.

Here's the link, definitely worth a look: Link

answered Apr 25 '11 at 19:13

Luis Aguilar
151 4

-3

I believe in parameter suffixing: string s_, int i_, etc

I also believe that the parm names should be short and generic as possible.

Now for the reasons:

- In the function, you do not want to modify the parameter in any way, if you need a modified version, create a new variable to stick it in. The naming of parms with a suffix will make sure your not assigning to them if your paying attention.
- Exceptions to this rule come when the parm is ref or out. Though I still use the suffix on those.
- Why short generic names? You should be documenting your function so you know what s_ really is in the descriptive sense. So with that out of the way, using short generics is handy when you are creating groups of similar functions, or clipping a function body to transport to another function as a start point for modification.
- The real benefit of generic names is you dont have to recall what you called that parameter in most cases. You know your getting a string, so its s_, etc and don't have to wonder if it is 'filename' or was it 'filepath' or was it 'fullpath', its the only string so its 's_'.

Everything has trade-offs, and whether you use something or not will depend a lot on how it fits into your current style.

answered Jul 13 '11 at 7:58

Mark
336 1 4

1 @Peter K. - It looks to me like the `s` and `i` are short names because this is just an example. IOW I don't think this is Hungarian at all - I think you're misinterpreting a short name which is just the classic `string s` or `int i` I-can't-think-of-a-better-name thing, but with underscore suffixes glued on. – Steve314 Jul 13 '11 at 9:26

@Steve314: Ah, you might be right! Let's see if Mark responds. – Peter K. Jul 13 '11 at 9:39

The s_ is I guess an anonymous HG, and it is not due to example. – Mark Jul 13 '11 at 9:48

Highly active question. Earn 10 reputation in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.