how explicit should I be with my overloads?

Asked 15 years, 8 months ago Modified 15 years, 8 months ago

Viewed 188 times

1

I'm building a wrapper for a jquery plugin to C# and I dont like the usage of [Optional] because its not as "optional" as it says it is(meaning you still have to declare that System.Missing library) so I decided to use overloaded methods. I want to give the user alot of customization but I'm not sure how explicit I should be with my overloads. Should I break it down in terms of importance for the parameters or do an overload for every scenario.

- Whats the foot print on overloads?
- Does it decrease in efficiency?
- Is it frowned upon in terms of OOP?

P.S I'm calling a base class' constructor and having to use `base(param1, param2, ...)` method but I would also like to use `this(param1, param2, ...`) instead of having to initialize my member variables in each scenario, is there a way around this?

c# parameters constructor wrapper overloading

Share

Improve this question

Follow

asked Mar 30, 2009 at 16:08

Ayo

1,228 ● 3 ● 17 ● 36

2 Answers

Sorted by: Highest score (default)

2

A confusing API is the eternal bane of library consumers. Make it easy, even trivial, for people to decide exactly what they want to do given the interface you provide. If they must pick from a lengthy list of overloaded functions, that's probably too much cognitive overhead. Also, it's worth noting that C# 4 will have support for optional and named parameters, so your `System.Missing` problem will go away by itself.

> Is it frowned upon in terms of OOP?

OOP favors the [Single Responsibility Principle](), or SRP. If you have a single class with many of these functions, each having many overloads, that suggests that it may be doing too much.

> Does it decrease in efficiency?

If you have a lot of overloads for the same method, it takes longer to *statically* resolve each function call (i.e., "which method with this name is the right one?"). But that's not going to have a *runtime* impact on performance if they're nonvirtual calls -- that is, where the compiler can statically know exactly what type something is going to be. Either way, though, I don't think that should be your motivating factor here.

Share Improve this answer Follow

edited Mar 30, 2009 at 16:51

answered Mar 30, 2009 at 16:17

John Feminella
311k ● 48 ● 346 ● 361

1

If you have a lot of parameters and need to be extensible consider using a class that contains your parameters.

E.g.

```
public class FileOptions
{
  public bool Indented { get; set; }
  public string Namespace { get; set; }
  public System.Text.Encoding Encoding { get; set; }
}

public void Save(string fileName, FileOptions options)

// usage:

obj.Save("a.xml", new FileOptions { Indented=true });
obj.Save("b.xml", new FileOptions { Namespace="urn:foo
Encoding=System.Text.Encoding.UTF8 });
```

Share Improve this answer Follow

answered Mar 30, 2009 at 16:38

laktak
59.8k ● 17 ● 138 ● 167

I like this approach but the only problem I have with it is that I want an overrided method from the base class to initiate in the constructor or on initialization of the object. – Ayo Mar 31, 2009 at 13:00