When do you use extension methods, ext. methods vs. inheritance?

Asked 14 years ago Modified 1 year, 2 months ago Viewed 33k times

63

We started using C# (.NET 3.0) and I wonder how you guys are using extension methods? When do you use them?

Also, I would appreciate if you also list all dark prerequisites for using them.

c# .net

Share Edit Follow Flag

edited Apr 27, 2009 at 21:04

GEOCHET
21k 15 73 98

asked Apr 24, 2009 at 19:55
PuzzleCracker

1 See my answer to another question for guidelines I use. – Michael Meadows Apr 24, 2009 at 20:01

Also see this question http://stackoverflow.com/questions/474074/overriding-extension-methods/474108#474108 – Jacob Adams Apr 24, 2009 at 20:04

1 extension methods are also useful for checking of values for certain objects like a http cookie. example public static bool IsValid(this HttpCookie cookie) { return cookie != null && !string.IsNullOrEmpty(cookie.Value)); } – Mike Geise Apr 25, 2009 at 0:57

7 Answers

Sorted by: Highest score (default)

66

Times to use extension methods:

- when you don't control the types being extended
- where you don't want to force the implementor to provide code that can be done using the existing methods

For an example of the second point; you might have an extension method on `IList<T>` (for example, `Sort`) that can be written entirely using the existing `IList<T>` members... so why force anybody else to write anything? This is the foundation block of LINQ, and allowed Microsoft to provide *much* more functionality without breaking anything.

Times to **not** use extension methods:

- when polymorphism is critical; you cannot guarantee that your code will be the version that gets executed with an extension method, as methods directly on the type take precedence
- when you need access to private/protected members

Share Edit Follow Flag

answered Apr 24, 2009 at 19:59

Marc Gravell
1.0m 261 2547 2886

3 I did not understood 1st point of **not** using extension methods. – Zameer Ansari Jul 2, 2015 at 14:24

@student, it's related to this (mentioned in another answer): "An extension method with the same name and signature as an instance method will not be called". If some inherited class implements the exact same signature of the extension method, the class method will be called instead of the extension. – drizin Jul 7, 2016 at 18:55

1 Is there any functional difference between an extension method and a normal static method? That is to say, instead of calling the method on an instance of a class, why not just pass that instance as a parameter of the method? Wouldn't an extension method be able to do all the same things if you just left out the `this` keyword? Is how you type it the only advantage? – Kyle Delaney Feb 18, 2017 at 22:15

29

Extension methods allow existing classes to be extended without relying on inheritance or having to change the class's source code. This means that if you want to add some methods into the existing String class you can do it quite easily. Here's a couple of rules to consider when deciding on whether or not to use extension methods:

- Extension methods cannot be used to override existing methods
- An extension method with the same name and signature as an instance method will not be called
- The concept of extension methods cannot be applied to fields, properties or events
- Use extension methods sparingly....overuse can be a bad thing!

Share Edit Follow Flag

answered Apr 24, 2009 at 19:58

Sasha

10

This link http://geekswithblogs.net/BlackRabbitCoder/archive/2010/04/26/c-extension-methods---to-extend-or-not-to-extend.aspx provides good guidance on when to use Extension methods and when not.

To quote from this article:

A good extension method should:
- Apply to any possible instance of the type it extends.
- Simplify logic and improve readability/maintainability.
- Apply to the most specific type or interface applicable.
- Be isolated in a namespace so that it does not pollute IntelliSense.

Share Edit Follow Flag

edited Jul 4, 2014 at 16:09

AbcAeffchen
14.2k 15 49 66

answered Jul 4, 2014 at 15:44

user3395318
101 1 2

Yea ... "Apply to any possible instance of the type it extends." is a important factor. If not, a utility/helper method of some kind probably makes more sense. – Andrew Jun 21, 2019 at 13:59

7

I use extension methods when it makes sense. If you control a class and its code, you usually don't need extension methods.

If you *don't*, an extension method might be useful.

One place I frequently use extension methods is for [Flags] enumerations. When you have a flag-based enumeration, there's a rather large expression that's necessary to determine whether or not an enumeration value has a particular flag set. And so I build the following extension method whenever I build a [Flags] enumeration:

```
[Flags]
public enum MyEnum
{
    FlagA,
    FlagB,
    // etc.
}

public static class MyEnumExt
{
    public static bool HasFlags(this MyEnum item, MyEnum query)
    {
        return ((item & query) == query);
    }
}
```

That way my code looks like:

```
MyEnum flags = MyEnum.FlagA;
if(flags.HasFlags(MyEnum.FlagA))
{
  // handle FlagA
}
```

Rather than:

```
MyEnum flags = MyEnum.FlagA;
if((flags & MyEnum.FlagA) == MyEnum.FlagA)
{
  // handle FlagA
}
```

Share Edit Follow Flag

answered Apr 24, 2009 at 20:03

Randolpho
55.1k 17 144 179

4

Honestly, I would say it is easier to explain when it is ***NOT*** a good idea than when it is a good idea.

I think, the main benefit to extension methods is that they may ***increase the adoption of your method***. This is because the user will not have to instantiate an instance of another class in order to use your method and intellisense will advertise your method when a developer is looking for methods for the class you are extending. This could be important if you are trying to get other developers to follow a new standard in your company.

When contemplating whether or not to create an extension method, remember the **SOLID** principles.

> **S**ingle Responsibility: - You are almost always at least bending the single responsibility principle with an extension method because you are tacking on to something that is already a class (that you either don't have control over or are too afraid to touch).
>
> **O**pen/Close Principle: - Extension methods cannot be overridden, which means your method may not be adequately "open for extension".
>
> **L**iskov substitution principle: - If you have any sort of inheritance structure you will not be able to use the extension methods on the sub types.
>
> **I**nterface segregation principle: - Although you can "extend" an interface, you have to provide a concrete implementation for that extension. So you can not program towards an interface >that can be implemented differently in a different context (eg Unit Test)
>
> **D**ependency inversion principle: - If your code has any dependencies, how are you going to expose those dependencies for factories and unit tests (dependency inversion principle)?

Finally, extension methods are just ***static methods wearing a new dress***. So all of the difficulties with static methods (such as ***thread safety, garbage collection, etc***) come along with extension

methods when you implement them.

So, I would think long and hard about writing a method as an extension and **reconsider using a factory and a basic helper** class instead.

If you do write an extension method, please make it very simple. Try to avoid having any dependencies (or pass all of your dependencies in as parameters). And be careful for how you manage memory and lock resources in a multi-threaded environment.

Share Edit Follow Flag

edited Nov 23, 2018 at 5:46

Tejus
684 1 6 19

answered Feb 6, 2017 at 23:27

Rober
65 2

9 This answer comes across as a vague hodge-podge of software engineering generalities, some of which have no special connection to extension methods in particular (i.e. they'd apply equally to regular instance methods), and some of the claims made are factually incorrect. Sorry, but this is not helpful. – stakx - no longer contributing Dec 28, 2017 at 14:07

2

What are extension methods?

Extension methods enable you to add methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type.

An extension method is a special kind of static method, but they are called as if they were instance methods on the extended type.

How to use extension methods?

An extension method is a static method of a static class, where the "this" modifier is applied to the first parameter. The type of the first parameter will be the type that is extended.

Extension methods are only in scope when you explicitly import the namespace into your source code with a using directive.

Important points for the use of extension methods:

1.An extension method must be defined in a top-level static class.

2.An extension method with the same name and signature as an instance method will not be called.

3.Extension methods cannot be used to override existing methods.

4.The concept of extension methods cannot be applied to fields, properties or events.

5.Overuse of extension methods is not a good style of programming.

Share Edit Follow Flag

answered Jul 8, 2018 at 17:05

AutomationNerd
400 1 4 12

0

Use inheritance when "Is-A" relationship makes sense. Inheritance create unwanted dependencies between the classes.

I use extension method when I feel "what good it would have been if XYZ class had ABC method".

I remember the best exammple when I used extension method when I needed to break collection into chunks. Here is the details.

https://www.codingcrest.com/extension-method-in-c/

Share Edit Follow Flag

answered Jan 30, 2022 at 18:08

Imad
7,042 11 53 109
