What's the best way to add custom functionality to existing code in C#?

Asked 15 years, 11 months ago Modified 15 years, 11 months ago

Viewed 1k times

2

Say, for example, I have many web applications using the same class library compiled into a dll in C# and I want to update a specific method of a class for a particular application. I don't want to change the existing code because it will affect everyone, and I don't want to inherit from a class to create a new one. I just want to change an existing method of a class for an individual application.

One way to achieve this would be to have the compiled classes as 'base' classes and have a separate file containing all the override classes inheriting the base classes. Then the applications will use the inherited classes. Then to change a class for an individual application you could just update the inherited classes. The only problem I see with this method is that every application would need 2 files, one of base classes and one of inherited (probably not compiled).

Can anyone recommend a better way of doing this?

c# inheritance

Share

Improve this question

Follow

asked Jan 21, 2009 at 12:20

Mark Clancy
7,889 ● 8 ● 44 ● 51

8 Answers

Sorted by: Highest score (default)

5

Maybe extension methods would work?

Share Improve this answer

Follow

answered Jan 21, 2009 at 12:28

Dmitri Nesteruk
23.8k ● 23 ● 106 ● 169

Extension methods are good for adding new methods to existing classes but this is not suitable for my scenario, thanks. – Mark Clancy Jan 24, 2009 at 13:03

2

You can't change the implementation of a compiled method; polymorphism is one option; the other is encapsulation, perhaps via a decorator pattern over an interface, using a factory to create the instance - i.e.

```
interface IFoo { void Bar();}
class Foo : IFoo {public void Bar() { /* imp 1 */ } }
class FooWrapper : IFoo {
    IFoo parent;
    public FooWrapper(IFoo parent) {this. parent = par
    public void Bar() { /* imp 2, perhaps using "paren
}
```

(I won't bother with an example of polymorphism)

Polymorphism and decoration achieve a similar end, but have different pros/cons - for example, decorators can be assembled in flexible chains for different scenarios.

With any other approach (such as extension methods), you end up calling a different method; if that suits, then fine.

Share Improve this answer Follow

answered Jan 21, 2009 at 12:50

Marc Gravell
1.1m ● 273 ● 2.6k ● 3k

2

If you simply want to **add** stuffs in the methods, and not modify its contents, you can use Aspect Oriented Programming (AOP). In C#, this can be done using PostSharp or AspectDNG.

AOP focuses on the separation of concerns: you write methods containing only business logic, and every other aspects of the application (security, logging, and so on) are encapsulated in their own modules. At compile time, these aspects' code are injected into the business code in particular locations you specified.

You can also have a look at this question. Even though the question aims specifically at runtime modification of methods, some answers might give you some hints.

Share Improve this answer Follow

edited May 23, 2017 at 12:13
Community Bot
1 ● 1

answered Jan 21, 2009 at 13:52
Luc Touraille
81.9k ● 16 ● 99 ● 139

0

I think extension methods is the best option. For more information go here.

Share Improve this answer Follow

answered Jan 21, 2009 at 12:31
Gerrie Schenck
22.4k ● 21 ● 69 ● 96

0

It seems strange that you can't override an existing method without using an inherited class. This is very straightforward in languages like javascript. The above suggestions have given me food for thought but nether really improve on my initial idea. Extension methods don't allow you to modify the existing method. With encapsulation I would still need to use a new class.

Say I had 10 applications all using the class User with method login(). I don't want to change where this method is called (ie. changing classname or method parameters).

I just want to change the method itself for that application so additional checks can be made before returning.

Share Improve this answer

Follow

answered Jan 21, 2009 at 13:22

Mark Clancy

7,889 ● 8 ● 44 ● 51

0

Is it possible to make the method in the class library Virtual? This would allow you to override it where you want to but keep it "as is" otherwise.

Share Improve this answer

Follow

answered Jan 21, 2009 at 13:25

Perry Neal

765 ● 4 ● 7

0

✓

Polymorphism seems to be the best solution for this. This way base classes can be compiled and left alone and inherited classes can be changed for individual applications without changing the way the original is used (unless necessary).

BaseClasses.cs

```
public class BaseUser {

    public BaseUser() {}

    public bool login() {

        return false;
    }

}
```

Classes.cs

```
public class User : BaseUser {}
```

Application

```
User u = new User();
u.login();
```

Share Improve this answer

Follow

answered Jan 21, 2009 at 16:09

Mark Clancy

7,889 ● 8 ● 44 ● 51

Part of the reson for this is that when you come to change the library classes you can see whether you will break anything without having to go into each of the applications and check for modifications. The more stuff you make abstract/virtual the harder changes to the library become. – Martin Brown Jan 21, 2009 at 16:32

0

Extension methods. This should be added to a static class. You should visit: http://weblogs.asp.net/scottgu/archive/2007/03/13/new-orcas-language-feature-extension-methods.aspx

Example of an extension method for the DateTime class to make it return the week number:

```
public static int WeekNumber(this DateTime dtPassed)
    {
        CultureInfo ciCurr = CultureInfo.CurrentCultur
        int weekNum = 0;
        try
        {
            weekNum = ciCurr.Calendar.GetWeekOfYear(dt
           CalendarWeekRule.FirstFourDayWeek, DayOfWee
        }
        catch (Exception ex)
        {
            //TODO: Add error handling code
        }

        return weekNum;
    }
```

Share Improve this answer

Follow

answered Jan 21, 2009 at 16:13

Raúl Roa

12.3k ● 13 ● 50 ● 64

Extension methods only allow you to add new methods to classes, not override existing ones. This is interesting but not what I'm looking for. – Mark Clancy Jan 21, 2009 at 16:24