# Aspect Oriented Programming in C# using DispatchProxy

## What Is Aspect Oriented Programming?

[Aspect Oriented Programming (AOP)](https://en.wikipedia.org/wiki/Aspect-oriented_programming) is very powerful approach to avoid boilerplate code and archive better modularity. The main idea is to add behavior (advice) to the existing code without making any changes in the code itself. AOP provides a way of weaving an aspect into the code. An aspect is supposed to be generic so it can be applied to any object and object should not have to know anything about aspect. AOP allows to separate [cross-cutting concerns](https://en.wikipedia.org/wiki/Cross-cutting_concern) and makes easier to follow [Single Responsibility Principle](https://en.wikipedia.org/wiki/Single_responsibility_principle) (one of the [SOLID](https://en.wikipedia.org/wiki/SOLID_(object-oriented_design)) principles). Logging, security, transactions and exceptions handling are the most common examples of using AOP. If you are not familiar with this programming technic you can read [this](https://en.wikipedia.org/wiki/Aspect-oriented_programming) or [this](http://study.com/academy/lesson/aspect-oriented-programming-definition-concepts.html). It could be very useful because this article mostly about how to use AOP in C# rather than what AOP is. Don’t be scared if you still do not understand what is all about. After locking at several examples, it becomes much easier to understand.

In Java AOP implemented in [AspectJ](https://eclipse.org/aspectj/) and [Spring](https://spring.io/) frameworks. There are [PostSharp](https://www.postsharp.net/) (not free), [NConcern](https://github.com/Virtuoze/NConcern) and some other frameworks (not very popular and easy to use) to do almost the same in .Net.

It is also possible to use **RealProxy** class to implement AOP. You can find some examples how to do it:

Example1: [Aspect-Oriented Programming: Aspect-Oriented Programming with the RealProxy Class](https://msdn.microsoft.com/en-us/magazine/dn574804.aspx).

This article also contains a lot of explanation what is AOP, how [Decorator Design Pattern](https://en.wikipedia.org/wiki/Decorator_pattern) works and examples of implementing logging and authentication using AOP.

Example2: [MSDN](https://msdn.microsoft.com/en-us/library/system.runtime.remoting.messaging.returnmessage(v=vs.110).aspx).

Unfortunately, these examples have some significant drawbacks. Example1 does not support out parameters. Example2 has limitation. Decorated class should be inherited from **MarshalByRefObject** (it could be a problem if it is not your class). Also, both examples do not support asynchronous functions as expected. Several months ago, I fixed the first example and wrote article about it.

Example3: [Aspect Oriented Programming in C# with RealProxy](https://www.codeproject.com/Articles/1204871/Aspect-Oriented-Programming-in-Csharp-with-RealPro).

Unfortunately, .Net Core does not have **RealProxy** class. There is **DispatchProxy** instead.