

# Double dispatch in C#?

Asked 16 years, 3 months ago   Modified 4 years, 7 months ago

Viewed 27k times



I have heard/read the term but don't quite understand what it means.

69



When should I use this technique and how would I use it?  
Can anyone provide a good code sample?



c#

design-patterns

language-features

double-dispatch



Share

Improve this question

Follow

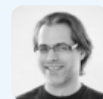
edited Dec 17, 2008 at 4:39



Jay Bazuzi

46.4k ● 16 ● 115 ● 170

asked Sep 3, 2008 at 21:03



Oded

498k ● 102 ● 893 ● 1k

Currently this is the best way to do it:

[blogs.msdn.microsoft.com/curth/2008/11/15/...](https://blogs.msdn.microsoft.com/curth/2008/11/15/...)

– Zar Shardan Oct 11, 2016 at 23:20

Great example of double dispatch at [Applying Object Composition to Build Rich Domain Models](#). – jsuddsjr Dec 16, 2016 at 19:06

[Beware of the double dispatch](#). Probably you can avoid it for better code maintenance. – Ognyan Dimitrov Jan 19, 2017 at

## 5 Answers

Sorted by:

Highest score (default) 

The visitor pattern is a way of doing double-dispatch in an object-oriented way.

58



It's useful for when you want to choose which method to use for a given argument based on its type at runtime rather than compile time.



Double dispatch is a special case of **multiple dispatch**.



When you call a virtual method on an object, that's considered single-dispatch because which actual method is called depends on the type of the single object.

For double dispatch, both the object's type and the method sole argument's type is taken into account. This is like method overload resolution, except that the argument type is determined at runtime in double-dispatch instead of statically at compile-time.

In multiple-dispatch, a method can have multiple arguments passed to it and which implementation is used depends on each argument's type. The order that the types are evaluated depends on the language. In LISP, it checks each type from first to last.

Languages with multiple dispatch make use of generic functions, which are just function declarations and aren't

like generic methods, which use type parameters.

**To do double-dispatch in C#**, you can declare a method with a sole object argument and then specific methods with specific types:

```
using System.Linq;

class DoubleDispatch
{
    public T Foo<T>(object arg)
    {
        var method = from m in GetType().GetMethods()
                      where m.Name == "Foo"
                        && m.GetParameters().Length==
                        && arg.GetType().IsAssignable
                        (m.GetParam
                        && m.ReturnType == typeof(T)
                      select m;

        return (T) method.Single().Invoke(this, new obj

    public int Foo(int arg) { /* ... */ }

    static void Test()
    {
        object x = 5;
        Foo<int>(x); //should call Foo(int) via Foo<T>
    }
}
```

Share Improve this answer

edited Feb 14, 2017 at 22:14

Follow

answered Sep 3, 2008 at 21:38



Mark Cidade

99.8k ● 33 ● 229 ● 237

---

3 Wonderful use of LINQ with reflection - I hadn't thought to apply it to those tedious xxxInfo objects before. Thanks!  
– [Jarrod Dixon](#) Sep 8, 2008 at 22:05

---

30 I'm not so sure this is a great idea. This doesn't really implement double dispatch at all, I need to know *at compile-t* what the type of the thing is in order to specify the type parameter!! You may as well not bother with all the reflection code and simply cast the object. Am I missing something?  
– [ljs](#) Feb 3, 2009 at 11:39

---

4 It does implement double dispatch in that it's dispatching on both the runtime type of the object (derived from DoubleDispatch) and the method argument. The reflection on return type is used for extending the mechanism to subclasses, so you can add "string Foo(string)" to a subclass and it'll work. – [Mark Cidade](#) Feb 3, 2009 at 16:42

---

6 Given that we now have a *dynamic* type in C#, this is only here for historical purposes. Feel free to edit my answer.  
– [Mark Cidade](#) Jun 8, 2011 at 16:39

---

2 If double dispatch is explained using reflection, I think the example is not that great. As I think reflection is not required. I think an example should display the bare essence. Probably I just fail to see the issue that the double dispatch is solving. – [Mike de Klerk](#) Nov 7, 2019 at 13:11

---



The code posted by Mark isn't complete and what ever is there isn't working.

**13**

So tweaked and complete.



```
class DoubleDispatch
{
```



```
public T Foo<T>(object arg)
{
    var method = from m in
GetType().GetMethods(System.Reflection.BindingFlags.In
System.Reflection.BindingFlags.Public |
System.Reflection.BindingFlags.NonPublic)
        where m.Name == "Foo"
            && m.GetParameters().Length
            //&& arg.GetType().IsAssign
            // (m.GetP
            &&Type.GetType(m.GetParamet
[0].ParameterType.FullName).IsAssignableFrom(arg.GetTy
            && m.ReturnType == typeof(T
        select m;

    return (T)method.Single().Invoke(this, new obj
}

public int Foo(int arg)
{
    return 10;
}

public string Foo(string arg)
{
    return 5.ToString();
}

public static void Main(string[] args)
{
    object x = 5;
    DoubleDispatch dispatch = new DoubleDispatch()

    Console.WriteLine(dispatch.Foo<int>(x));

    Console.WriteLine(dispatch.Foo<string>(x.ToStr
    Console.ReadLine();
}
}
```

Thanks Mark and others for nice explanation on Double Dispatcher pattern.

Share Improve this answer

edited May 8, 2020 at 8:31

Follow



peterh

12.6k ● 20 ● 89 ● 113

answered Jun 8, 2011 at 4:38



Zenwalker

1,909 ● 2 ● 15 ● 28

---

I agree. This code is complete and shows what double dispatch really is. The explanation from mark's answer and this code go perfectly together. – [Sachin Kainth](#) Sep 19, 2013 at 9:17

---

- 1 Wouldn't writing  
"Console.WriteLine(dispatch.Foo<x.GetType()>(x));" be a more dynamic and useful way to take advantage of the Pattern? – [Anestis Kivranoglou](#) Jul 1, 2015 at 12:17 ✎
- 



10



The other answers use generics and the runtime type system. But to be clear the use of generics and runtime type system doesn't have anything to do with double dispatch. They can be used to implement it but double dispatch is just dependent on using the concrete type at runtime to dispatch calls. It's illustrated more clearly I think in [the wikipedia page](#). I'll include the translated C++ code below. The key to this is the virtual CollideWith on SpaceShip and that it's overridden on ApolloSpacecraft. This is where the "double" dispatch takes place and the

correct asteroid method is called for the given spaceship type.

```
class Spaceship
{
    public virtual void CollideWith(Asteroid asteroid)
    {
        asteroid.CollideWith(this);
    }
}

class ApolloSpacecraft : Spaceship
{
    public override void CollideWith(Asteroid asteroid)
    {
        asteroid.CollideWith(this);
    }
}

class Asteroid
{
    public virtual void CollideWith(Spaceship target)
    {
        Console.WriteLine("Asteroid hit a Spaceship");
    }

    public virtual void CollideWith(ApolloSpacecraft target)
    {
        Console.WriteLine("Asteroid hit ApolloSpacecraft");
    }
}

class ExplodingAsteroid : Asteroid
{
    public override void CollideWith(Spaceship target)
    {
        Console.WriteLine("ExplodingAsteroid hit a Spaceship");
    }

    public override void CollideWith(ApolloSpacecraft target)
    {
        Console.WriteLine("ExplodingAsteroid hit ApolloSpacecraft");
    }
}
```

```

    }
}

class Program
{
    static void Main(string[] args)
    {
        Asteroid[] asteroids = new Asteroid[] { new As
ExplodingAsteroid() };

        ApolloSpacecraft spacecraft = new ApolloSpacec

        spacecraft.CollideWith(asteroids[0]);
        spacecraft.CollideWith(asteroids[1]);

        SpaceShip spaceShip = new SpaceShip();

        spaceShip.CollideWith(asteroids[0]);
        spaceShip.CollideWith(asteroids[1]);
    }
}

```

Share Improve this answer

edited May 19, 2020 at 21:12

Follow

answered May 18, 2020 at 19:54



MikeJ


1,359 ● 7 ● 12

You wrote "dependent on using the concrete type at runtime to dispatch calls.". But your realization also depend on static type and the overload resolution in those lines : `

asteroid.CollideWith(this); . So, additional places should be changed with introduction of new

SpaceShip: adding new Asteroid's CollideWith` method. – Koe Kto Nov 17, 2022 at 17:15



- 1 That is what double dispatch entails - using the runtime type of two objects to dispatch a function call. The rest of this is all implementation details. There are a variety of ways to implement double dispatch, this answer provides a simple direct example. Given the question, I thought it appropriate to provide a simple example. – [MikeJ](#) Nov 17, 2022 at 20:00 

This is a pretty good intermediate form between OOP and visitor pattern in its pure form. In OOP, adding a new function requires you to change many modules, while in this case, it is a little bit tidier to add more functions like `PierceThrough()`, and `GetIntoOrbit()`, but hard to add more `Spaceships`(you have to add another method to each `Asteroid`).

– [Onat Korucu](#) Feb 14, 2023 at 22:52



8



C# 4 introduces the pseudo type `dynamic` which resolves the function call at runtime (instead of compile time). (That is, the runtime type of the expression is used). Double- (or multi-dispatch) can be simplified to:

```
class C { }

static void Foo(C x) => Console.WriteLine(nameof(Foo))
static void Foo(object x) => Console.WriteLine(nameof(

public static void Main(string[] args)
{
    object x = new C();

    Foo((dynamic)x); // prints: "Foo"
    Foo(x);          // prints: "Object"
}
```

Note also by using `dynamic` you prevent the static analyzer of the compiler to examine this part of the code.

You should therefore carefully consider the use of `dynamic`.

Share Improve this answer

edited Mar 6, 2019 at 7:53

Follow

answered Jul 12, 2017 at 9:29



Micha Wiedenmann

20.8k ● 22 ● 95 ● 141



Full listing of working code

0



```
using System;
using System.Linq;

namespace TestConsoleApp
{
    internal class Program
    {
        public static void Main(string[] args)
        {
            const int x = 5;
            var dispatch = new DoubleDispatch();

            Console.WriteLine(dispatch.Foo<int>(x));
            Console.WriteLine(dispatch.Foo<string>(x.T

            Console.ReadLine();
        }
    }

    public class DoubleDispatch
    {
        public T Foo<T>(T arg)
        {
            var method = GetType()
                .GetMethod()
```

```

        .Single(m =>
            m.Name == "Foo" &&
            m.GetParameters().Length == 1 &&
            arg.GetType().IsAssignableFrom(m.G
[0].ParameterType) &&
            m.ReturnType == typeof(T));

        return (T) method.Invoke(this, new object[]
    }

    public int Foo(int arg)
    {
        return arg;
    }

    public string Foo(string arg)
    {
        return arg;
    }
}
}

```

Share Improve this answer

answered Dec 17, 2019 at 8:41

Follow



bugs-x64

11 ● 2