Switch case on type c# [duplicate]

Ask Question

54

7

Possible Duplicate:
C# - Is there a better alternative than this to 'switch on type'?

Hello suppose i get a big if/else on class type. it's there a way to do it with a switch case ?

Example :

```
function test(object obj)
{
if(obj is WebControl)
{

}else if(obj is TextBox)
{

}
else if(obj is ComboBox)
{

}
```

etc ...

I would like to create something like

```
switch(obj)
{
case is TextBox:
break;
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
    }

}
```

c# .net optimization switch-statement

edited May 23 '17 at 12:10
Community ♦
1 1

asked Aug 31 '11 at 3:15
Cédric Boivin
5,130 9 45 87

marked as duplicate by ChrisF ♦, Michael Myers ♦ Sep 26 '12 at 16:28

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

1 see also stackoverflow.com/questions/156467/switch-pattern-matching-idea stackoverflow.com/questions/7542793/… stackoverflow.com/questions/4478464/c-sharp-switch-on-type stackoverflow.com/questions/298976/… – Mikhail May 1 '12 at 11:49

1 and stackoverflow.com/questions/94305/… stackoverflow.com/questions/7149788/… stackoverflow.com/questions/6304815/… stackoverflow.com/questions/5947343/… stackoverflow.com/questions/10115028/… stackoverflow.com/questions/2551773/… – Mikhail May 1 '12 at 11:49

7 UPDATE for anyone landing on this page: The answer is yes

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

See: blogs.msdn.microsoft.com/dotnet/2016/08/24/… – LanderV May 14 '17 at 12:19

see example here – Thierry Prost May 6 at 13:32

5 Answers

58

✓

No.

http://blogs.msdn.com/b/peterhal/archive/2005/07/05/435760.aspx

> We get a lot of requests for adddditions to the C# language and today I'm going to talk about one of the more common ones - switch on type. Switch on type looks like a pretty useful and straightforward feature: Add a switch-like construct which switches on the type of the expression, rather than the value. This might look something like this:

```
switch typeof(e) {
        case int:    ... break;
        case string: ... break;
        case double: ... break;
        default:     ... break;
}
```

> This kind of statement would be extremely useful for adding virtual method like dispatch over a disjoint type hierarchy, or over a type hierarchy containing types that you don't own. Seeing an example like this, you could easily conclude that the feature would be straightforward and useful. It might even get you thinking "Why don't those #*&%$ lazy C# language

Home

PUBLIC

Stack Overflow

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Jobs

Teams
Q&A for work
Learn More

> Unfortunately, like many 'simple' language features, type switch is not as simple as it first appears. The troubles start when you look at a more significant, and no less important, example like this:

```
class C {}
interface I {}
class D : C, I {}

switch typeof(e) {
case C: … break;
case I: … break;
default: … break;
}
```

Link: https://blogs.msdn.microsoft.com/peterhal/2005/07/05/many-questions-switch-on-type/

Update C# 7

Yes: Source

```
switch(shape)
{
    case Circle c:
        WriteLine($"circle with radius {c.Radius}");
        break;
    case Rectangle s when (s.Length == s.Height):
        WriteLine($"{s.Length} x {s.Height} square");
        break;
    case Rectangle r:
        WriteLine($"{r.Length} x {r.Height} rectangle");
        break;
    default:
        WriteLine("<unknown shape>");
        break;
    case null:
        throw new ArgumentNullException(nameof(shape));
}
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Thierry Prost
190 2 14

answered Aug 31 '11 at 3:18
Steve
25.2k 13 89 117

4 I disagree with the language designers. Lots of languages have type switches. Anyway while there is no type-switch it is easy enough to implement see stackoverflow.com/questions/7252186/switch-case-on-type-c/... – cdiggins Sep 4 '11 at 19:08

37 Where's the rest of the answer? it now ends on "... , example like this:" – ɹǝɯoɔ Oct 18 '11 at 3:54

1 I don't care what he says. It is 'simple'. Rather it would be if they designed C# better from the start. I hear there are 10+ passes for all the grammar :(. Switch is essentially obj.member. The vtable is an inaccessible member. If its treated as a value (like int) you can use it. – user34537 Jan 5 '13 at 2:52

5 @ɹǝɯoɔ I supplied the link to the source of my information if you want to read the entire article. I wasn't going to paste the entire quote when you can read it at the source. The simple answer was "No". – Steve Jan 5 '13 at 3:25

6 UPDATE for anyone landing on this page: The answer is yes since C# 7. You can totally write switch cases over types now. See: blogs.msdn.microsoft.com/dotnet/2016/08/24/... – LanderV May 14 '17 at 12:19

50 The following code works more or less as one would expect a type-switch that only looks at the actual type (e.g. what is returned by `GetType()`).

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
        var ts = new TypeSwitch()
            .Case((int x) => Console.WriteLine("int"))
            .Case((bool x) => Console.WriteLine("bool"))
            .Case((string x) => Console.WriteLine("string"));

        ts.Switch(42);
        ts.Switch(false);
        ts.Switch("hello");
    }
```

Here is the machinery required to make it work.

```
public class TypeSwitch
{
    Dictionary<Type, Action<object>> matches = new Diction
    public TypeSwitch Case<T>(Action<T> action) { matches.
action((T)x)); return this; }
    public void Switch(object x) { matches[x.GetType()](x)
}
```

answered Sep 4 '11 at 19:07

cdiggins
12.1k 3 78 84

Very interesting solution... in some ways this is terrible :) ... but in some ways this is incredible (especially in the event that some other external developer could tap into this system by creating class "X" and then supplying the 'what to do with X' logic ... sorta like a mini DI/Ioc) – Timothy Khouri Sep 5 '11 at 1:28

8 When "in some ways this is terrible"? – Pedro77 Jul 18 '13 at 22:34

1 @cdiggins, is there a way I can add something like "default" or "none of them" ? – Pedro77 Jul 18 '13 at 22:51

Wow!! Very interesting solution! – edotassi Nov 6 '14 at 14:45

1 @Pedro77 Yes you can check the existence of the Key in the Switch method and if it doesn't exist, then use the

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

22

Yes, you can switch on the name...

```
switch (obj.GetType().Name)
{
    case "TextBox":...
}
```

answered Aug 31 '11 at 3:18
Timothy Khouri
22k 17 73 120

1 FYI, this solution won't work when given a `public class MyCustomTextBox : TextBox` – Steve Aug 31 '11 at 3:20

1 You would just do `case "TextBox: "MyCustomTextBox": ...` – Timothy Khouri Aug 31 '11 at 3:25

7 Assuming you know all possible subclasses, sure. – Steve Aug 31 '11 at 3:29

3 I imagine that the person writing this code is writing it for a reason... and knows all of the subclasses :P - Example, he's probably trying to add a CssClass name for all "textboxes" and then all "datepickers" or whatever. – Timothy Khouri Aug 31 '11 at 3:44

3 I know this is old, but I like your answer, and I agree with your reasoning that the person writing this would know all of their subclasses, as long as they're using the switch statement to specify action for specific classes, and not all children classes – cost May 16 '13 at 5:06

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

13

hard enough it almost looks like a real switch statement.

The calling code looks like this:

```
var @switch = this.Switch(new []
{
    this.Case<WebControl>(x => { /* WebControl code here *,
    this.Case<TextBox>(x => { /* TextBox code here */ }),
    this.Case<ComboBox>(x => { /* ComboBox code here */ })
});

@switch(obj);
```

The `x` in each lambda above is strongly-typed. No casting required.

And to make this magic work you need these two methods:

```
private Action<object> Switch(params Func<object, Action>[
{
    return o =>
    {
        var @case = tests
            .Select(f => f(o))
            .FirstOrDefault(a => a != null);

        if (@case != null)
        {
            @case();
        }
    };
}

private Func<object, Action> Case<T>(Action<T> action)
{
    return o => o is T ? (Action)(() => action((T)o)) : (A
}
```

Almost brings tears to your eyes, right?

Nonetheless, it works. Enjoy.

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Enigmativity
79.4k 9 67 135

1 +1 definitely clever. Not sure I'd use it in production code (not that there is anything *wrong* with it per-se). – Steve Aug 31 '11 at 6:37

2 The "new [] {" and "}" are superfluous. – cdiggins Sep 4 '11 at 19:12

Hehe, nice one. I agree with @Steve but +1 – Doctor Jones Sep 4 '11 at 19:20

@cdiggins - I agree they are superflous, but I wrote the code to allow dropping the array initializer in favour of a list of parameters. The array initializer syntax is a little more readable, IMHO, once you have more than three parameters. And it looks more like the normal `switch` syntax - if you squint hard enough! – Enigmativity Sep 5 '11 at 0:28

1 @Enigmativity loved the clever solution, but as I squinted hard enough found it, same as if, else if looping, not genuinely a switch case, this would again do the chaining of calls till it finds the first case which is not null. But agreed nice camouflage. – Mrinal Kamboj Feb 13 '16 at 13:18

9

The simplest thing to do could be to use dynamics, i.e. you define the simple methods like in Yuval Peled answer:

```
void Test(WebControl c)
{
...
}

void Test(ComboBox c)
{
...
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Then you cannot call directly Test(obj), because overload resolution is done at compile time. You have to assign your object to a dynamic and then call the Test method:

```
dynamic dynObj = obj;
Test(dynObj);
```

answered Aug 31 '11 at 4:55

Francesco Baruchelli
6,097 2 28 34

1 Cool solution! Got my vote :-) – Volker von Einem Jul 3 '12 at 14:42

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.