More efficient abridged use of IF statement C#

Asked 9 years, 9 months ago Modified 9 years, 9 months ago Viewed 237 times

0

I would like to know if there is a better more efficient way to use if statements rather than just long lines of

```
if(){
    //code
}else if(){
    //code
} else{
    //code
}
```

I've done some research on the site and I found this:

> If you have only 2 values, I strongly suggest to use the code you posted, because is likely the most readable, elegant and fast code possible (IMHO).
>
> But if you have more cases like that and more complicated, you could think to use a switch statement:

```
switch (el.type)
{
    case ElementType.Type1:
    case ElementType.Type2:
    case ElementType.Type3:
        //code here
        break;
    case ElementType.Type4:
    case ElementType.Type5:
        //code here
        break;
    case ElementType.Type6:
        //code here
        break;
}
that translated in if statements would be:

if (el.type == ElementType.Type1 ||
    el.type == ElementType.Type2 ||
    el.type == ElementType.Type3 )
{
    // code here
}else if(el.type == ElementType.Type4 ||
         el.type == ElementType.Type5)
{
    // code here
}else if(el.type == ElementType.Type6)
{
```

```
    // code here
}
```

They're perfectly equal to me, but the switch seems more readable/clearer, and you need to type less (i.e. it's "shorter" in term of code length) :)

Although I don't quite understand what it is telling me, is it saying that a switch statement is a better use for long if statements or?

To have some context surrounding my problem, I have a Windows Forms application with some radio buttons - A questionnaire if you will - I want to know if there is more efficient ways that reduces repetitive unnecessary lines of code and replaces them with short code that does the same job.

c# design-patterns refactoring

Share
Improve this question
Follow

edited Mar 6, 2015 at 14:27
st4hoo
2,204 ● 18 ● 25

asked Mar 6, 2015 at 13:15
Allahu Akbar
29 ● 2 ● 7

1 `if` and `switch` are both flow control statements. For the most part they are technically interchangeable (80% of the time). Choose the one that seems most contextually appropriate. – Sam Axe Mar 6, 2015 at 13:18

Can you post a (short) part of your if/else code? – DrKoch Mar 6, 2015 at 13:18

Could this be solved with polymorphic behavior? How do you create or assign different values to `ElementType`? – Ilya Ivanov Mar 6, 2015 at 13:20

We really need more context here. You say you have some radio buttons. How are you accessing them in an if/else cascade? Can you post some sample code? – Matthew Watson Mar 6, 2015 at 13:44

4 Answers

Sorted by: Highest score (default)

0

The goal should be more readable / clearer code, which does not always mean the shortest statement.

If you have very complex logic, with lots of nested if / else branches, try to split it in smaller, simpler routines.

Share Improve this answer Follow

answered Mar 6, 2015 at 13:25

TAMTAM

101 ● 5

0

Switch statements as well as many `if-else` statements are often a sign of the smelly design. Consider refactoring into a more object-oriented design

`If-else` not much more readable than `switch-case` and vice-versa.

Share

Improve this answer

Follow

edited Mar 6, 2015 at 13:30

answered Mar 6, 2015 at 13:22

EngineerSpock

2,675 ● 4 ● 39 ● 60

> Very interesting, I'll definitely move onto that as a learning point in the near future. Thank you very much for linking me the information, the article is clear and relevant, I shall award you the answer, while it wasn't the answer I wanted now it will help me more in the future.
> – Allahu Akbar Mar 6, 2015 at 13:26

0

You could apply CoR pattern. Here you will fine one of .NET implementations. When you combine this with IoC container you may achieve very flexible architecture...

Share

Improve this answer

Follow

edited Mar 6, 2015 at 13:34

answered Mar 6, 2015 at 13:28

st4hoo

2,204 ● 18 ● 25

0

`switch` statements are not only easier to read, they are easier for the compiler to optimize because they explicitly specify one value you are triggering all the code off of. The compiler can easily optimize this using a hash or a jump table rather than a sequence of comparisons, making it much faster (when it can do so). Technically, it could detect that the `if` statements are the same thing, but may or may not be sophisticated enough to do so. So, if you have 256 separate cases triggering off the value of a `byte`, a switch can be compiled into a hard coded array of code offsets (each entry in the array is the offset of the code handling that case), and no comparison is done at all because it can just use the byte value as an index into the hard coded array and jump to the correct code. This is *much* faster than doing 255 (or 256) compares.

Switches using `string` values are also handled specially, using hash codes for better performance. You may be able to do something similar explicitly, but your code will be

much less readable.

Share

Improve this answer

Follow

edited Mar 6, 2015 at 19:17

answered Mar 6, 2015 at 13:28

James

3,908 ● 1 ● 29 ● 43