Method parameter array default value [duplicate]

Asked 7 years, 1 month ago Active 3 years, 6 months ago Viewed 10k times

9

This question already has an answer here:
Passing an empty array as default value of an optional parameter [duplicate] *3 answers*

In c# it is possible to use default parameter values in a method, in example:

```
public void SomeMethod(String someString = "string value")
{
    Debug.WriteLine(someString);
}
```

But now I want to use an array as the parameter in the method, and set a default value for it.
I was thinking it should look something like this:

```
public void SomeMethod(String[] arrayString = {"value 1", "value 2", "value 3"})
{
    foreach(someString in arrayString)
    {
        Debug.WriteLine(someString);
    }
}
```

But this does not work.
Is there a correct way to do this, if this is even possible at all?

c# arrays parameters default-value optional-parameters

edited May 4 '16 at 22:23
colinfang
8,288 10 57 115

asked Sep 26 '12 at 17:25
Gabi Barrientos
774 2 15 33

marked as duplicate by colinfang, Rob ♦ c# May 5 '16 at 2:05

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.

Define "does not work". – Tudor Sep 26 '12 at 17:27

There is a workaround for reference types. Set the argument default to "null". Then, inside the code block check if parameter is set to null, if it is null set the default value for the reference type parameter. – Bill Moore Sep 26 '17 at 20:00

2 Answers

16

✓

> Is there a correct way to do this, if this is even possible at all?

This is not possible (directly) as the default value must be one of the following (from Optional Arguments):

- a constant expression;
- an expression of the form new ValType(), where ValType is a value type, such as an enum or a struct;
- an expression of the form default(ValType), where ValType is a value type.

Creating an array doesn't fit any of the possible default values for optional arguments.

The best option here is to make an overload:

```
public void SomeMethod()
{
    SomeMethod(new[] {"value 1", "value 2", "value 3"});
}

public void SomeMethod(String[] arrayString)
{
    foreach(someString in arrayString)
    {
        Debug.WriteLine(someString);
        }
    }
```

answered Sep 26 '12 at 17:27

Reed Copsey
486k 61 1017 1300

Ok, thanks. I will have a go at this. I will accept this as my answer in about 11 minutes. – Gabi Barrientos Sep 26 '12 at 17:30

12

Try this:

```
public void SomeMethod(String[] arrayString = null)
{
    arrayString = arrayString ?? {"value 1", "value 2", "value 3"};
    foreach(someString in arrayString)
    {
        Debug.WriteLine(someString);
    }
}
someMethod();
```

answered Sep 26 '12 at 17:38

Nathan
2,227 18 26

Thanks for your input. It seems like a good way to go, however I am going to stick with Reed's answer on this one because it comes in handy in some other ways as well for me. – Gabi Barrientos Sep 26 '12 at 17:40

4 +1 This is a slick approach, but it doesn't give you a way to differentiate between a user passing in null *explicitly* and just not using the parameter (which is why I typically prefer the overload approach). That may or may not be important in this case. – Reed Copsey Sep 26 '12 at 17:42

You're welcome. – Nathan Sep 26 '12 at 17:42

What about `int[] arrayString` ? I tried with `arrayString = arrayString ?? { 0 };` but I got the compiler error `CS1525: Invalid expression term '{'` – John Oct 31 '17 at 2:51

4 It's okay I got it resolved by changing it to `arrayString = arrayString ?? new int[] { 0 };` – John Oct 31 '17 at 2:57