

Forum:[Hack Forums](#)

Thread: [Grouping functions ?](#)

System[X] - March 1, 2019, 11:10 am

I have a class(Class1) and a few functions then need to call them like this

```
class1.test1()
class1.test2()
class1.test3()
```

This works fine but i want to kinda group the functions into 1 (myfunction)and to do something like this

```
class1.myfunction.test1
class1.myfunction.test2
class1.myfunction.test3
```

all 3 do the same thing but produce a different return value

Im not exactly sure what its called so i can look up some examples.

Any help would be great, Thanks

Vexallus - March 1, 2019, 12:07 pm

how about using a variable as a counter for each time u use that function ? that way will make it easier for you, or try being more specific on what you need so we can try helping you out :)

System[X] - March 1, 2019, 12:19 pm

(Mar 1, 2019 01:07 PM)Vexallus Wrote: [\[CITE\]](#) how about using a variable as a counter for each time u use that function ? that way will make it easier for you, or try being more specific on what you need so we can try helping you out :)

I did not say anything about a variable counter and i think i made it quite clear on what i am wanting to achieve. This is not a question for any coder but a semi pro who has delt with creating advanced class libraries befor.

Thanks

itcorp - March 1, 2019, 1:17 pm

(Mar 1, 2019 12:10 PM)System[X] Wrote: [\[CITE\]](#) I have a class(Class1) and a few functions then need to call them like this

```
class1.test1()
class1.test2()
class1.test3()
```

This works fine but i want to kinda group the functions into 1 (myfunction)and to do something like this

```
class1.myfunction.test1  
class1.myfunction.test2  
class1.myfunction.test3
```

all 3 do the same thing but produce a different return value

Im not exactly sure what its called so i can look up some examples.

Any help would be great, ThanksIf I understand you correctly,this is the solution for you

Code Public Class Class1

```
Public Sub test1()  
    'your code  
End Sub
```

```
Public Sub test2()  
    'your code  
End Sub
```

```
Public Sub test3()  
    'your code  
End Sub
```

```
Private Shared instance As Class1  
Private Sub New()
```

```
End Sub
```

```
Public Shared Function myfunction() As Class1  
    If instance Is Nothing Then instance = New Class1()  
    Return instance  
End Function
```

```
End Class
```

Usage:

```
Code Class1.myfunction.test1()  
Class1.myfunction.test2()  
Class1.myfunction.test3()
```

System[X] - March 1, 2019, 1:29 pm

(Mar 1, 2019 02:17 PM)itcorp Wrote: [CITE](#) If I understand you correctly,this is the solution for you

Code Public Class Class1

```
Public Sub test1()  
    'your code  
End Sub
```

```
Public Sub test2()  
    'your code  
End Sub
```

```
Public Sub test3()  
    'your code  
End Sub
```

```
Private Shared instance As Class1  
Private Sub New()
```

```
End Sub
```

```
Public Shared Function myfunction() As Class1  
    If instance Is Nothing Then instance = New Class1()  
    Return instance  
End Function  
End Class  
Usage:  
Code Class1.myfunction.test1()  
      Class1.myfunction.test2()  
      Class1.myfunction.test3()
```

Yes this is kinda what i needed and can work with it.

Just 1 question why is there a new constructor in there that's not used ?

```
Code Private Sub New()
```

```
End Sub
```

Or is it that is is needed for the Shared instance ?

Thanks

itcorp - March 1, 2019, 1:57 pm

(Mar 1, 2019 02:29 PM)System[X] Wrote: [\[CITE\]](#) Yes this is kinda what i needed and can work with it.

Just 1 question why is there a new constructor in there that's not used ?

```
Code Private Sub New()
```

```
End Sub
```

Or is it that is is needed for the Shared instance ?

Thanks

It's necessary in order to have the correct singleton pattern and if you want something more to configure on instance creation

System[X] - March 2, 2019, 1:30 am

(Mar 1, 2019 02:57 PM)itcorp Wrote: [\[CITE\]](#) It's necessary in order to have the correct singleton pattern and if you want something more to configure on instance creation

Well it actually works without the constructor, i have no use for them in this class unless i decide to add static variables for this class.

Thanks

itcorp - March 2, 2019, 2:31 am

(Mar 2, 2019 02:30 AM)System[X] Wrote: [\[CITE\]](#) Well it actually works without the constructor, i have no use for them in this class unless i decide to add static variables for this class.

Thanks

In this way you have only one instance,because singleton can have only one instance if you want to prevent second instance creation.if you throw out the constructor from code multiple instances will be allowed.

So while the constructor is present you will not be able to do this:

```
Code Dim classone = New Class1()
```

System[X] - March 2, 2019, 5:25 am

Ok, so how would i go about adding another function for example:

if i add this to the above code

```
Code Public Sub test6()  
    'your code  
End Sub
```

```
Public Shared Function newfunction() As Class1  
    If instance Is Nothing Then instance = New Class1()  
    Return instance  
End Function
```

then i can call newfunction.test6, this works however i can also do myfunction.test6 or newfunction.test1

I dont want to make a new class for this tho.

itcorp - March 2, 2019, 5:42 am

(Mar 2, 2019 06:25 AM)System[X] Wrote: [\[CITE\]](#) Ok, so how would i go about adding another function for example:

if i add this to the above code

```
Code Public Sub test6()  
    'your code  
End Sub
```

```
Public Shared Function newfunction() As Class1  
    If instance Is Nothing Then instance = New Class1()  
    Return instance  
End Function
```

then i can call newfunction.test6, this works however i can also do myfunction.test6 or newfunction.test1

I dont want to make a new class for this tho.you can add as much as you want functions in that class and call them,but you always will have one instance of that class you will not make a new instance you dont need to make new class i was just explaining to you which is the result when you throw out the constructor and why it is in the code to prevent creation of new instanse of class outside of pattern.

System[X] - March 2, 2019, 6:00 am

(Mar 2, 2019 06:42 AM)itcorp Wrote: [\[CITE\]](#) you can add as much as you want functions in that class and call them, but you always will have one instance of that class you will not make a new instance you don't need to make new class i was just explaining to you which is the result when you throw out the constructor and why it is in the code to prevent creation of new instance of class outside of pattern.

Yes, i'll try to explain it a bit better.

Ok, so with your code, if i add

```
Code Public Sub test6()
```

```
    'your code
```

```
End Sub
```

I then can call it with "Class1.myfunction.test6" correct

that works perfectly fine with the function of "myfunction" but...

What if i want to add a different function for "test6" like "Class1.MyNewFunction.test6" ?

so test1, test2, test3 works with myfunction but test6 works with MyNewFunction.

Thanks

itcorp - March 2, 2019, 6:18 am

(Mar 2, 2019 07:00 AM)System[X] Wrote: [\[CITE\]](#) Yes, i'll try to explain it a bit better.

Ok, so with your code, if i add

```
Code Public Sub test6()
```

```
    'your code
```

```
End Sub
```

I then can call it with "Class1.myfunction.test6" correct

that works perfectly fine with the function of "myfunction" but...

What if i want to add a different function for "test6" like "Class1.MyNewFunction.test6" ?

so test1, test2, test3 works with myfunction but test6 works with MyNewFunction.

Thanks! It would not be a violation of the pattern and for that I recommend creating factory pattern

System[X] - March 2, 2019, 6:21 am

(Mar 2, 2019 07:18 AM)itcorp Wrote: [\[CITE\]](#) It would not be a violation of the pattern and for that I recommend creating factory pattern

Ok, but how would this be done with the code you posted ?

Thanks

itcorp - March 2, 2019, 7:54 am

(Mar 2, 2019 07:21 AM)System[X] Wrote: [\[CITE\]](#) Ok, but how would this be done with the code you posted ?

Thanks

You can do something like this.

Usage:

```

Code FactoryPattern.Pattern().Init(TestType.Test1).Test1()
FactoryPattern.Pattern().Init(TestType.Test2).Test1()
Code:
Code Public Enum TestType
    Test1
    Test2
End Enum
Code Public Class FactoryPattern
    Implements FactoryCore
    Private Property _testType As TestType

    Public Function Init(_testType As TestType) As Core Implements FactoryCore.Init
        Select Case _testType
            Case TestType.Test1
                Return Class1.myfunction
            Case TestType.Test2
                Return Class2.MyNewFunction
            Case Else
                Return Nothing
        End Select
    End Function

    Private Sub New()

    End Sub

    Private Shared instance As FactoryPattern

    Public Shared Function Pattern() As FactoryPattern
        If instance Is Nothing Then instance = New FactoryPattern()
        Return instance
    End Function

End Class
Code Public Interface FactoryCore
    Function Init(ByVal _testType As TestType) As Core
End Interface

Public Interface Core
    Sub Test1()
    Sub Test2()
    Sub Test3()
End Interface
Code Public Class Class1
    Implements Core

    Public Sub Test1() Implements Core.Test1
        Console.WriteLine("class1 test1")
    End Sub

    Public Sub Test2() Implements Core.Test2
        Console.WriteLine("class1 test2")
    End Sub

```

```

End Sub

Public Sub Test3() Implements Core.Test3
    Console.WriteLine("class1 test3")
End Sub

Private Shared instance As Class1
Private Sub New()

End Sub

Public Shared Function myfunction() As Class1
    If instance Is Nothing Then instance = New Class1()
    Return instance
End Function
End Class
Code Public Class Class2
    Implements Core

    Public Sub Test1() Implements Core.Test1
        Console.WriteLine("calss2 test1")
    End Sub

    Public Sub Test2() Implements Core.Test2
        Console.WriteLine("class2 test2")
    End Sub

    Public Sub Test3() Implements Core.Test3
        Console.WriteLine("class2 test3")
    End Sub

    Private Shared instance As Class2

    Public Shared Function MyNewFunction() As Class2
        If instance Is Nothing Then instance = New Class2()
        Return instance
    End Function
End Class

```

0xDEAD10CC - March 2, 2019, 4:34 pm

(Mar 2, 2019 08:54 AM)itcorp Wrote: [\[CITE\]](#) You can do something like this.

...

Why use an enum for branching over a generic type constraint? That's usually a sign of a very poorly designed factory pattern implementation. I would also brush up on your .NET naming conventions.

I would question whether any of this even needs to be singletons, and if you just need a factory pattern that initializes a derived class with overridden functions from an abstract base class holding virtual methods. -- OR, classes that implement a common interface.

A factory class that attempts to return singletons? That branching doesn't make sense and seems more like an anti-pattern in the making... IMO it should be an abstract class, remove the private

constructor since that's redundant, and use it as it was intended; a factory.

I think the best approach here would be to not try to solve what he's asking but get an idea of what he's trying to do before suggesting a solution to accomplish a very weird and uncommon paradigm...

You can create somewhat of a lazy initialization factory that also doubles as a pseudo-singleton by caching instances of a specific generic type in a dictionary. You may desire to add thread-safety to this depending on what the goal is but this is just for demonstration only...

```
Code public interface ICoreInitialization
```

```
{  
    void Init();  
}
```

```
public sealed class MyClass1 : ICoreInitialization
```

```
{  
    private int _counter = 0;  
    public void Init() => Console.WriteLine("{0} -> {1}", nameof(MyClass1), ++_counter);  
}
```

```
public sealed class MyClass2 : ICoreInitialization
```

```
{  
    private int _counter = 0;  
    public void Init() => Console.WriteLine("{0} -> {1}", nameof(MyClass2), ++_counter);  
}
```

```
public abstract class CoreFactory<T>
```

```
    where T : ICoreInitialization, new()  
{  
    private static readonly Dictionary<int, T> _map = new Dictionary<int, T>();  
    public static T CreateOrGet()  
    {  
        int hashCode = typeof(T).GetHashCode();  
        if (!_map.TryGetValue(hashCode, out T obj))  
        {  
            obj = new T();  
            _map[hashCode] = obj;  
        }  
        return obj;  
    }  
}
```

```
public static void Main()
```

```
{  
    CoreFactory<MyClass1>.CreateOrGet().Init();  
    CoreFactory<MyClass2>.CreateOrGet().Init();  
    CoreFactory<MyClass1>.CreateOrGet().Init();  
    CoreFactory<MyClass2>.CreateOrGet().Init();  
    Console.ReadKey();  
}
```

* Probably better to use Lazy<T> or something here though.

I'm just guessing though but this seems like a singleton branching problem on the surface; or a polymorphism problem unknown to OP at this time because he's structuring his code in an odd way.

itcorp - March 2, 2019, 5:45 pm

(Mar 2, 2019 05:34 PM)0xDEAD10CC Wrote: [\[CITE\]](#) Why use an enum for branching over a generic type constraint? That's usually a sign of a very poorly designed factory pattern implementation. I would also brush up on your .NET naming conventions.

I would question whether any of this even needs to be singletons, and if you just need a factory pattern that initializes a derived class with overridden functions from an abstract base class holding virtual methods. -- OR, classes that implement a common interface.

A factory class that acts as a singleton? Makes no sense and seems more like an anti-pattern in the making... IMO it should be an abstract class, remove the private constructor since that's redundant, and use it as it was intended; a factory.

I think the best approach here would be to not try to solve what he's asking but get an idea of what he's trying to do before suggesting a solution to accomplish a very weird and uncommon paradigm... It's a little weird,i agree with you!This is just most similar example of what he wants to achieve and I did not go into the topic to imagine the ideas which he imagines.They both are different design patterns and this is a mix of singleton and factory pattern.I also do not think it is necessary for factory to have more than one instance in some cases as well the usage of abstraction because there is no need for complexity which follow the abstraction in more serious application solutions.I also understand what you want to say ,factory is responsible for creating instances of object and singleton ensures that one instance is created and that why is not necessary for each class to have one instance in this case factory class.But I do not think there is any violation of pattern and i know this is not always the right choice of implementation and everything is a matter of need.

Thanks for the reply,you're a very smart guy and to be honest I avoid going into the debate with you i see that you have a lot of things in your little finger.
All the best

0xDEAD10CC - March 2, 2019, 6:06 pm

(Mar 2, 2019 06:45 PM)itcorp Wrote: [\[CITE\]](#) It's a little weird,i agree with you!This is just most similar example of what he wants to achieve and I did not go into the topic to imagine the ideas which he imagines.They both are different design patterns and this is a mix of singleton and factory pattern.I also do not think it is necessary for factory to have more than one instance in some cases as well the usage of abstraction because there is no need for complexity which follow the abstraction in more serious application solutions.I also understand what you want to say ,factory is responsible for creating instances of object and singleton ensures that one instance is created and that why is not necessary for each class to have one instance in this case factory class.But I do not think there is any violation of pattern and i know this is not always the right choice of implementation and everything is a matter of need.

Thanks for the reply,you're a very smart guy and to be honest I avoid going into the debate with you i see that you have a lot of things in your little finger.
All the best

VB.net syntax eludes me as I haven't touched it really in 7+ years. I was going by what it "looked" like from my own conceptual translation to C# by viewing your code. I may be wrong about what I thought, but I think the question is hidden in layers of abstraction which makes it difficult to provide an answer for OP.

itcorp - March 2, 2019, 6:22 pm

(Mar 2, 2019 07:06 PM)0xDEAD10CC Wrote: [\[CITE\]](#) VB.net syntax eludes me as I haven't touched it really in 7+ years. I was going by what it "looked" like from my own conceptual translation to C# by viewing your code. I may be wrong about what I thought, but I think the question is hidden in layers of abstraction which makes it difficult to provide an answer for OP.

Regardless you haven't touched VB.net 7+ years ,personally i understood what you wanted to say and to point up and you have made real and smart conclusions as always and thanks for that.

0xDEAD10CC - March 2, 2019, 6:29 pm

From what he said initially and throughout this thread; based on his first example of the calls:

Code class1.myfunction.test1

class1.myfunction.test2

class1.myfunction.test3

It seems like 'class1' could represent a number of things; a factory, proxy, IoC container, etc.

In retrospect to a factory pattern implementation, 'myfunction' is the factory method which instantiates an object with similar methods 'test1/2/3'.

But there are many ways to achieve this, as mentioned earlier.

1. 'class1' could act as an IoC container for a common base type that gets instantiated with a derived type (as you don't see any generics in the call examples). The 'myfunction' could return this private object that exposes a common set of the 3 methods 'test1/2/3' either through inheritance, or common implementation.

2. 'class1' could be a factory class as mentioned above, and generics (which aren't shown here) could be used to return a specific derived type, that exposes the 3 common methods.

3. Maybe this could be a proxy implementation of some flavor?

A decision on direction for some of these would depend on the implementation itself however. I'd like to wait to see what OP is using this for. (Ex: if it's an IoC container, then perhaps it's restricted to a branch of specific execution determined by instantiation of the 'class1' type, in which case, this doesn't make it interchangeable with other common functionality for the lifetime of that object, whereas a factory pattern would be. If he intends on using singletons too, then we need to know why, or if it is even necessary, as this would be something to consider as well; whether he wants lazy initialization, or a pure singleton, or a hybrid of the two.)

tl;dr - "Grouping functions" can be related to many things.