Method-Chaining in C#

Asked 10 years, 4 months ago Active 3 years, 11 months ago Viewed 45k times

65

I have actually no idea of what this is called in C#. But i want to add the functionallity to my class to add multiple items at the same time.

```
myObj.AddItem(mItem).AddItem(mItem2).AddItem(mItem3);
```

c#

23

edited Dec 5 '15 at 15:10 — kame — 7,868 23 84 135

asked Jul 13 '09 at 14:31 — Patrick — 783 5 12 16

stackoverflow.com/a/2055701/5558126 Gives a good example how to implement this – Peter verleg Nov 26 '18 at 12:55

9 Answers

110

The technique you mention is called chainable methods. It is commonly used when creating DSLs or fluent interfaces in C#.

The typical pattern is to have your AddItem() method return an instance of the class (or interface) it is part of. This allows subsequent calls to be chained to it.

```
public MyCollection AddItem( MyItem item )
{
   // internal logic...

   return this;
}
```

Some alternatives to method chaining, for adding items to a collection, include:

Using the `params` syntax to allow multiple items to be passed to your method as an array. Useful when you want to hide the array creation and provide a variable argument syntax to your methods:

```
public void AddItems( params MyItem[] items )
{
    foreach( var item in items )
        m_innerCollection.Add( item );
}

// can be called with any number of arguments...
coll.AddItems( first, second, third );
coll.AddItems( first, second, third, fourth, fifth );
```

Providing an overload of type IEnumerable or IEnumerable so that multiple items can be passed together to your collection class.

```
public void AddItems( IEnumerable<MyClass> items )
{
    foreach( var item in items )
         m_innerCollection.Add( item );
}
```

Use .NET 3.5 collection initializer syntax. You class must provide a single parameter `Add( item )` method, implement IEnumerable, and must have a default constructor (or you must call a specific constructor in the initialization statement). Then you can write:

```
var myColl = new MyCollection { first, second, third, ... };
```

edited Nov 23 '11 at 23:58

Eric J.
124k 48 282 506

answered Jul 13 '09 at 14:36

LBushkin
107k 30 197 252

1 +1 learnt new things :D i thought that i hava to make a different method for chaning method – GaryNg Dec 5 '13 at 14:49

Is there a significant performance hit if you do this? – starbeamrainbowlabs Jun 29 '16 at 15:38

Use this trick:

33

```
public class MyClass
{
    private List<MyItem> _Items = new List<MyItem> ();

    public MyClass AddItem (MyItem item)
    {
        // Add the object
        if (item != null)
            _Items.Add (item)

        return this;
    }
}
```

It returns the current instance which will allow you to chain method calls (thus adding multiple objects "at the same time".

edited Jul 13 '09 at 14:39

answered Jul 13 '09 at 14:33

User
21.7k 17 69 106

don't need to restrict the argument to AddItem to MyClass. Could probably Object, or Item, or something. – Mats Fredriksson Jul 13 '09 at 14:36

15

> "I have actually no idea of what this is called in c#"

A fluent API; `StringBuilder` is the most common .NET example:

```
var sb = new StringBuilder();
string s = sb.Append("this").Append(' ').Append("is a ").Append("silly way to")
     .AppendLine("append strings").ToString();
```

answered Jul 13 '09 at 14:37

Marc Gravell ♦
831k 216 2238 2636

11

Others have answered in terms of straight method chaining, but if you're using C# 3.0 you might be interested in *collection initializers*... they're only available when you make a constructor call, and only if your method has appropriate `Add` methods and implements `IEnumerable`, but then you can do:

```
MyClass myClass = new MyClass { item1, item2, item3 };
```

answered Jul 13 '09 at 14:37

Jon Skeet
1143k 722 8253 8627

5

Why don't you use the `params` keyword?

```
public void AddItem (params MyClass[] object)
{
    // Add the multiple items
}
```

edited Jul 13 '09 at 16:33

answered Jul 13 '09 at 14:37

bruno conde
43.3k 10 89 112

3

You could add an extension method to support this, provided your class inherits from ICollection:

```
[TestClass]
public class UnitTest1
{
    [TestMethod]
    public void CanChainStrings()
    {
        ICollection<string> strings = new List<string>();

        strings.AddItem("Another").AddItem("String");

        Assert.AreEqual(2, strings.Count);
    }
}
```

```
public static class ChainAdd
{
    public static ICollection<T> AddItem<T>(this ICollection<T> collection, T item)
    {
        collection.Add(item);
        return collection;
    }
}
```

edited Mar 25 '15 at 10:47

answered Mar 25 '15 at 10:31

GC.

874 1 6 19

3

How about

```
AddItem(ICollection<Item> items);
```

or

```
AddItem(params Item[] items);
```

You can use them like this

```
myObj.AddItem(new Item[] { item1, item2, item3 });
myObj.AddItem(item1, item2, item3);
```

This is not method chaining, but it adds multiple items to your object in one call.

answered Jul 13 '09 at 14:36

Martin Liversage

87.2k 17 176 226

Something like this?

1

```
class MyCollection
{
    public MyCollection AddItem(Object item)
    {
        // do stuff
        return this;
    }
}
```

answered Jul 13 '09 at 14:35

Mats Fredriksson

16.1k 6 33 54

If your item is acting as a list, you may want to implement an interface like iList or iEnumerable / iEnumerable.

1

Regardless, the key to chaining calls like you want to is returning the object you want.

```
public Class Foo
{
   public Foo AddItem(Foo object)
   {
        //Add object to your collection internally
        return this;
   }
}
```

answered Jul 13 '09 at 14:35

Tim Hoolihan

11.1k 3 36 51
