Enforce Attribute Decoration of Classes/Methods

Asked 16 years, 4 months ago Modified 5 years, 7 months ago

Viewed 5k times

18

Following on from my recent question on Large, Complex Objects as a Web Service Result. I have been thinking about how I can ensure all future child classes are serializable to XML.

Now, obviously I could implement the IXmlSerializable interface and then chuck a reader/writer to it but I would like to avoid that since it then means I need to instantiate a reader/writer whenever I want to do it, and 99.99% of the time I am going to be working with a *string* so I may just write my own.

However, to serialize to XML, I am simply decorating the class and its members with the *Xml???* attributes (*XmlRoot* , *XmlElement* etc.) and then passing it to the *XmlSerializer* and a *StringWriter* to get the string. Which is all good. I intend to put the method to return the string into a generic utility method so I don't need to worry about type etc.

The this that concerns me is this: If I do not decorate the class(es) with the required attributes an error is not thrown until run time.

Is there any way to enforce attribute decoration? Can this be done with FxCop? (I have not used FxCop yet)

UPDATE:

Sorry for the delay in getting this close off guys, lots to do!

Definitely like the idea of using reflection to do it in a test case rather than resorting to FxCop (like to keep everything together).. Fredrik Kalseth's answer was fantastic, thanks for including the code as it probably would have taken me a bit of digging to figure out how to do it myself!

+1 to the other guys for similar suggestions :)

c# xml serialization coding-style .net-attributes

Share

Improve this question

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

asked Aug 21, 2008 at 7:49

Rob Cooper

28.9k ● 26 ● 105 ● 142

5 Answers

Sorted by: Highest score (default)

19

I'd write a unit/integration test that verifies that any class matching some given criteria (ie subclassing X) is decorated appropriately. If you set up your build to run with tests, you can have the build fail when this test fails.

UPDATE: You said, "Looks like I will just have to roll my sleeves up and make sure that the unit tests are collectively maintained" - you don't have to. Just write a general test class that uses reflection to find all classes that needs to be asserted. Something like this:

```
[TestClass]
public class When_type_inherits_MyObject
{
    private readonly List<Type> _types = new List<Type

    public When_type_inherits_MyObject()
    {
        // lets find all types that inherit from MyObj
indirectly
        foreach(Type type in typeof(MyObject).Assembly
        {
            if(type.IsClass && typeof(MyObject).IsAssi
            {
                _types.Add(type);
            }
        }
    }

    [TestMethod]
    public void Properties_have_XmlElement_attribute
    {
        foreach(Type type in _types)
        {
            foreach(PropertyInfo property in type.GetP
            {
                object[] attribs =
property.GetCustomAttributes(typeof(XmlElementAttribut
```

```
                Assert.IsTrue(attribs.Count > 0, "Miss
on property " + property.Name + " in type " + type.Ful
            }
        }
    }
}
```

Share Improve this answer

Follow

edited Aug 21, 2008 at 10:29

answered Aug 21, 2008 at 7:54

Fredrik Kalseth

14.2k ● 4 ● 26 ● 18

1

You can write unit tests to check for this kind of thing - it basically uses reflection.

Given the fact this is possible I guess it would also be possible to write a FxCop rule, but I've never done such a thing.

Share Improve this answer

Follow

answered Aug 21, 2008 at 8:17

samjudson

56.8k ● 7 ● 60 ● 69

1

You can write an FxCop rule or even check for the attributes by calling GetType() in the base class's constructor and reflecting over the returned type.

Share Improve this answer

Follow

answered Aug 21, 2008 at 10:08

Mark Cidade
99.8k ● 33 ● 229 ● 237

0

A good FXCop rule (and one which I am finding I need right now) would be to check that all objects that are being added to the ASP.NET Session have the Serializable attribute. I'm trying to move from InProc session state to SQL Server. First time I requested a page, my site blew up on me because non-serializable objects were being stored in Session. Then came the task of hunting through all the source code looking for every instance where an object is set in the Session... FXCop would be a nice solution. Something to work on...

Share Improve this answer

Follow

answered Oct 3, 2008 at 22:11

Larry Silverman
1,053 ● 1 ● 11 ● 32

0

You can also use this concept/post-processor to enforce relationships between attributes and use similar login to enforce relationships between classes and attributes at compile time:

http://www.st.informatik.tu-darmstadt.de/database/publications/data/cepa-mezini-gpce04.pdf?id=92

Share Improve this answer

Follow

answered Mar 6, 2009 at 5:18

Professional Sounding Name