What is difference between published and public methods / attributes?

Asked 16 years ago Modified 12 years, 7 months ago Viewed 509 times

1

According to Martin Fowler "Something can be public but that does not mean you have published it." Does this mean something like this:

```
public interface IRollsRoyceEngine
{
    void Start();
    void Stop();
    String GenerateEngineReport();
}

public class RollsRoyceEngine : IRollsRoyceEngine
{
    public bool EngineHasStarted { get; internal
set; }

    public bool EngineIsServiceable { get;
internal set; }

    #region Implementation of IRollsRoyceEngine

    public void Start()
    {
        if (EngineCanBeStarted())
            EngineHasStarted = true;
        else
            throw new
InvalidOperationException("Engine can not be
started at this time!");
    }
```

```
    public void Stop()
    {
        if (EngineCanBeStopped())
            EngineHasStarted = false;
        else
            throw new
InvalidOperationException("Engine can not be
started at this time!");
    }
```

Can it be said that published interface of this is IRollsRoyceEngine not whatever is in RollsRoyceEngine?

If so what is the real difference between public and published methods?

design-patterns **public-method**

Share

Improve this question

Follow

edited Apr 30, 2012 at 14:24

wattostudios

8,764 ● 13 ● 44 ● 58

asked Dec 14, 2008 at 3:55

Prajwal Tuladhar

533 ● 1 ● 6 ● 13

2 Answers

Sorted by: Highest score (default)

I assume he means that the contract is king - just because a method in your class is public, doesn't entitle

5

clients to assume that they can call it, or that they know what it does, or that it will be there in the next version. APIs are not defined by the source, they're defined by a contract, usually in the form of documentation.

It is the responsibility of the client not to call undocumented (unpublished) functions, not the responsibility of the implementer to hide methods which shouldn't be called.

Some people might disagree with this - typically those who don't trust documentation, and would rather find out how things work by looking at the source to see what it *actually* does, rather that what the author *claims* it does. They may well have a point, especially in practice when dealing with under-documented code. But I think that's in opposition to what Fowler is saying, which is that functionality should be formally defined, rather than inferred by examination of the particular implementation.

Share Improve this answer Follow

edited Dec 14, 2008 at 4:08

answered Dec 14, 2008 at 4:02

Steve Jessop

279k ● 40 ● 469 ● 709

1 More can be found about this issue at martinfowler.com/ieeeSoftware/published.pdf – Prajwal Tuladhar Dec 14, 2008 at 8:11

2

In my opinion mentioned white paper talks about target audience of the API rather than the distinction between interface and its implementation.

You can find analogy in Framework Design Guidelines which says that once your API shipped you have a contract with consumers. For example, if you shipped in v1 of your framework IService interface you cannot change it in v2 because it will introduce breaking changes to end developers. Instead you must create new interface IService2 inhereted from IService and ship it with v2.

So basically public API becomes published once you "sign a contract" with end developers.

Returning back to your code - it will be published when you ship it to development community for example.

Hope this explanation will help.

Share Improve this answer

Follow

answered Dec 14, 2008 at 16:39

Dzmitry Huba

4,521 ● 22 ● 19