C#

C# static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C# code

All rules 409 | Vulnerability 34 | Bug 76 | Security Hotspot 28 | Code Smell 271 | Quick Fix 52

Tags

Search by name...

"protected" members
Code Smell

Underscores should be used to make large numbers readable
Code Smell

"ToString()" calls should not be redundant
Code Smell

"==" should not be used when "Equals" is overridden
Code Smell

An abstract class should have both abstract and concrete methods
Code Smell

Multiple variables should not be declared on the same line
Code Smell

Culture should be specified for "string" operations
Code Smell

"switch" statements should have at least 3 "case" clauses
Code Smell

break statements should not be used except for switch cases
Code Smell

String literals should not be duplicated
Code Smell

Files should contain an empty newline at the end
Code Smell

Unused "using" should be removed
Code Smell

Arguments of public methods should be validated against null

Analyze your code

Code Smell Major convention

A publicly accessible method can be called from anywhere, which means you should validate parameters to be within the expected constraints. In general, checking against `null` is recommended defensive programming.

This rule raises an issue when a parameter of a publicly accessible method is not validated against `null` before being dereferenced.

Noncompliant Code Example

```
public class MyClass
{
    private MyOtherClass other;

    public void Foo(MyOtherClass other)
    {
        this.other = other; // Compliant: other not being de
    }

    public void Bar(MyOtherClass other)
    {
        this.other = other.Clone(); // Noncompliant
    }

    protected void FooBar(MyOtherClass other)
    {
        this.other = other.Clone(); // Noncompliant
    }
}
```

Compliant Solution

```
public class MyClass
{
    private MyOtherClass other;

    public void Foo(MyOtherClass other)
    {
        this.other = other;
    }

    public void Bar(MyOtherClass other)
    {
        if (other != null)
        {
            this.other = other.Clone();
        }
    }

    protected void FooBar(MyOtherClass other)
```

A close curly brace should be located at the beginning of a line

Code Smell

Tabulation characters should not be used

Code Smell

Methods and properties should be named in PascalCase

Code Smell

Track uses of in-source issue suppressions

Code Smell

```
    {
        if (other != null)
        {
            this.other = other.Clone();
        }
    }
}
```

Exceptions

To create a custom null validation method declare an attribute with name `ValidatedNotNullAttribute` and mark the parameter that is validated for null in your method declaration with it:

```
using System;

public sealed class ValidatedNotNullAttribute : Attribute {

public static class Guard
{
    public static void NotNull<T>([ValidatedNotNullAttribute
    {
        if (value == null)
            throw new ArgumentNullException(name);
    }
}

public static class Utils
{
    public static string ToUpper(string value)
    {
        Guard.NotNull(value, nameof(value));
        if (value == null)
        {
            return value.ToString();
        }
        return value.ToUpper(); // Compliant
    }
}
```

Available In:

sonarlint sonarcloud sonarqube