

- Secrets
- ABAP
- Apex
- C
- C++
- CloudFormation
- COBOL
- C#**
- CSS
- Flex
- Go
- HTML
- Java
- JavaScript
- Kotlin
- Objective C
- PHP
- PL/I
- PL/SQL
- Python
- RPG
- Ruby
- Scala
- Swift
- Terraform
- Text
- TypeScript
- T-SQL
- VB.NET
- VB6
- XML



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... 🔍

Code Smell

Multiline blocks should be enclosed in curly braces

Code Smell

Boolean expressions should not be gratuitous

Code Smell

Types and methods should not have too many generic parameters

Code Smell

Write-only properties should not be used

Code Smell

Exceptions should not be thrown from property getters

Code Smell

Unused type parameters should be removed

Code Smell

Parameters should be passed in the correct order

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Unused assignments should be removed

Code Smell

Tests should not be ignored

Code Smell

"switch" statements should not have too many "case" clauses

Code Smell

"ConstructorArgument" parameters should exist in constructors

Analyze your code

Bug Major xaml wpf

When creating a custom Markup Extension that accepts parameters in WPF, the `ConstructorArgument` markup must be used to identify the discrete properties that match these parameters. However since this is done via a string, the compiler will not notice if there are typos.

This rule raises an issue when the string argument to `ConstructorArgumentAttribute` doesn't match any parameter of any constructor.

Noncompliant Code Example

```
using System;

namespace myLibrary
{
    public class MyExtension : MarkupExtension
    {
        public MyExtension() { }

        public MyExtension(object value1)
        {
            Value1 = value1;
        }

        [ConstructorArgument("value2")] // Noncompliant
        public object Value1 { get; set; }
    }
}
```

Compliant Solution

```
using System;

namespace myLibrary
{
    public class MyExtension : MarkupExtension
    {
        public MyExtension() { }

        public MyExtension(object value1)
        {
            Value1 = value1;
        }

        [ConstructorArgument("value1")]
        public object Value1 { get; set; }
    }
}
```

Sections of code should not be commented out

 Code Smell

Unused method parameters should be removed

 Code Smell

Empty arrays and collections should be returned instead of null

 Code Smell

Unused private types or members should be removed

 Code Smell

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)