

- 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... 🔍

"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

"static" fields should be initialized inline

Analyze your code

Code Smell Minor ?

When a static constructor serves no other purpose than initializing static fields, it comes with an unnecessary performance cost because the compiler generates a check before each static method or instance constructor invocation.

Instead, inline initialization is highly recommended.

Noncompliant Code Example

```
namespace myLib
{
    public class Foo
    {
        static int i;
        static string s;

        static Foo() // Noncompliant
        {
            i = 3;
            ResourceManager sm = new ResourceManager("strings", AssemblyName);
            s = sm.GetString("mystring");
        }
    }
}
```

Compliant Solution

```
namespace myLib
{
    public class Foo
    {
        static int i = 3;
        static string s = InitString();

        static string InitString()
        {
            ResourceManager sm = new ResourceManager("strings", AssemblyName);
            return sm.GetString("mystring");
        }
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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

© 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](#)