sonar RULES

Products

- 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

Method parameters should be declared with base types

Analyze your code

Code Smell Minor api-design

When a derived type is used as a parameter instead of the base type, it limits the uses of the method. If the additional functionality that is provided in the derived type is not required then that limitation isn't required, and should be removed.

This rule raises an issue when a method declaration includes a parameter that is a derived type and accesses only members of the base type.

Noncompliant Code Example

```
using System;
using System.IO;

namespace MyLibrary
{
  public class Foo
  {
    public void ReadStream(FileStream stream) // Noncomplian
    {
      int a;
      while ((a = stream.ReadByte()) != -1)
      {
            // Do something.
      }
    }
  }
}
```

Compliant Solution

```
using System;
using System.IO;

namespace MyLibrary
{
  public class Foo
  {
    public void ReadStream(Stream stream)
    {
      int a;
      while ((a = stream.ReadByte()) != -1)
      {
            // Do something.
      }
    }
  }
}
```

Available In:

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