
































-  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

"ToString()" calls should not be redundant

Analyze your code

Code Smell Minor Quick Fix finding clumsy

Invoking a method designed to return a string representation of an object which is already a string is a waste of keystrokes. Similarly, explicitly invoking ToString() when the compiler would do it implicitly is also needless code-bloat.

This rule raises an issue when ToString() is invoked:

- on a string
- on a non-string operand to concatenation
- on an argument to string.Format

Noncompliant Code Example

```
var s = "foo";
var t = "fee fie foe " + s.ToString(); // Noncompliant
var someObject = new object();
var u = "" + someObject.ToString(); // Noncompliant
var v = string.Format("{0}", someObject.ToString()); // Noncompliant
```

Compliant Solution

```
var s = "foo";
var t = "fee fie foe " + s;
var someObject = new object();
var u = "" + someObject;
var v = string.Format("{0}", someObject);
```

Exceptions

The rule does not report on value types, where leaving off the ToString() call would result in automatic boxing.

```
var v = string.Format("{0}", 1.ToString());
```

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