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

Generic event handlers should be used

Analyze your code

Code Smell Major

Since .Net Framework version 2.0 it is not necessary to declare a delegate that specifies a class derived from `System.EventArgs`. The `System.EventHandler<TEventArgs>` delegate mechanism should be used instead as it allows any class derived from `EventArgs` to be used with that handler.

This rule raises an issue when an old style delegate is used as an event handler.

Noncompliant Code Example

```
public class MyEventArgs : EventArgs
{
}

public delegate void MyEventHandler(object sender, MyEventAr

public class EventProducer
{
  public event MyEventHandler MyEvent;

  protected virtual void OnMyEvent(MyEventArgs e)
  {
    if (MyEvent != null)
    {
      MyEvent(e);
    }
  }
}

public class EventConsumer
{
  public EventConsumer(EventProducer producer)
  {
      producer.MyEvent += HandleEvent;
  }

  private void HandleEvent(object sender, MyEventArgs e)
  {
    // Do something...
  }
}
```

Compliant Solution

```
public class MyEventArgs : EventArgs
{
}

public class EventProducer
```

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

```
{
  public event EventHandler<MyEventArgs> MyEvent;

  protected virtual void OnMyEvent(MyEventArgs e)
  {
    if (MyEvent != null)
    {
      MyEvent(e);
    }
  }
}

public class EventConsumer
{
  public EventConsumer(EventProducer producer)
  {
      producer.MyEvent += HandleEvent;
  }

  private void HandleEvent(object sender, MyEventArgs e)
  {
    // Do something...
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube